

# Cell/B.E. SDK: Code sample directory

Keep track of where to find code examples for the SDK

Skill Level: Introductory

[Anita Bateman \(ajbatema@us.ibm.com\)](mailto:ajbatema@us.ibm.com)  
Certified Senior IT Architect, Cell Solutions  
IBM

[VanDung To \(vto@us.ibm.com\)](mailto:vto@us.ibm.com)  
Software Engineer  
IBM

15 Jul 2008

In this article, you'll find tables indicating the locations of code samples that illustrate how to use the IBM SDK for Multicore Acceleration. This article will be updated with new code samples.

## About the SDK for Multicore Acceleration

To enable you to take full advantage of the Cell Broadband Engine® (Cell/B.E.) architecture and the PowerXCell 8i processor, IBM has developed a software development kit designed to accelerate production-ready, multi-core programming.

The [IBM Software Development Kit \(SDK\) for Multicore Acceleration](#) provides the libraries, tools, and resources that businesses need to develop and tune applications for Cell/B.E. technology. You can easily:

- Port and optimize applications and algorithms quickly.
- Increase ease-of-programming and developer productivity.
- Obtain a reliable development tool kit with warranty and support.

- Plug in third-party ISV libraries to integrate and build your software ecosystem.

## Cell/B.E. demos

These full demonstration programs are designed and optimized to show the capabilities of the Cell/B.E.

**Table 1. Cell/B.E. demos**

Example	Description	RPM Default installed directory	SDK
<b>FFT16M</b>	Tuned Fast Fourier Transform. Performs a 4-way SIMD single-precision complex FFT on an array of size 16,777,216 elements.	cell-demos-source /opt/cell/sdk/src/demos/FFT16M	3.0
<b>Julia Set</b>	Quaternion Julia Set Ray-tracing Sample.	cell-demos-source /opt/cell/sdk/src/demos/julia_set	3.0
<b>Matrix Multiply</b>	Parallel matrix multiplication workload for CBE.		3.0

## Cell/B.E. examples

These small examples show different features of the SDK and the Cell/B.E. architecture.

**Table 2. Cell/B.E. examples**

Example	Description	RPM Default installed directory	SDK
<b>cache-cp</b>	Demonstrates a standalone <i>spu</i> let that copies one file to another. The file contents are mapped into the effective address space, and a software managed cache is used to stage the data into LS. The example demonstrates the use of	cell-examples-source /opt/cell/sdk/src/examples/cache/cache-cp	3.0

	software-managed cache.		
<b>cache: sort</b>	Demonstrates examples of quicksort and heapsort that use a software managed data cache. The example demonstrates the use of software-managed cache.	cell-examples-source	3.0 /opt/cell/sdk/src/examples/cache/sort
<b>ch_prof</b>	Demonstrates an example of a statistical profiling utility that can be used to determine channel read and write activity of an SPU program. This example also contains an example application on how to use the ch_prof utility.	cell-examples-source	3.0 /opt/cell/sdk/src/examples/ch_prof
<b>DMA: simple</b>	Demonstrates simple DMA calls within one SPE,	cell-examples-source	3.0 /opt/cell/sdk/src/examples/DMA/simple
<b>DMA: complex</b>	Demonstrates DMA calls for multiple SPEs using single-buffered DMA, double-buffered DMA, single-buffered DMA List, and double-buffered DMA list.	cell-examples-source	3.0 /opt/cell/sdk/src/examples/DMA/complex
<b>overlay: overview</b>	Demonstrates an example implementation for SPU code overlay,	cell-examples-source	3.0 /opt/cell/sdk/src/examples/overlay/overview
<b>overlay: simple</b>	Demonstrates a simple example of using SPU code overlay.	cell-examples-source	3.0 /opt/cell/sdk/src/examples/overlay/simple
<b>overlay: large matrix</b>	Demonstrates an example of using SPU code overlay for large-matrix math.	cell-examples-source	3.0 /opt/cell/sdk/src/examples/overlay/large_matrix
<b>ppe_address_space</b>	Demonstrates an implementation of quick-sort on a random list of floats using compiler PPE address space support. The example illustrates the	cell-examples-source	3.0 /opt/cell/sdk/src/examples/ppe_address_space

	<p>use of the automatic software caching feature supported by the SPU compiler. The <code>"__ea"</code> qualifier is used within the source file to indicate to the SPU compiler that a memory reference is in the effective address space, rather than in local store.</p>	
<b>spe_interrupt</b>	<p>Demonstrates the use of an assembly first level interrupt handle (FLIH) that calls ABI compliant, registered, C-sourced second level interrupt handlers (SLIH) as a function of the interrupt events.</p>	<p>cell-examples-source 3.0 /opt/cell/sdk/src/examples/spu_interrupt</p>
<b>spe_interrupt_fast</b>	<p>Demonstrates the use of the fast interrupt handler.</p>	<p>cell-examples-source 3.0 /opt/cell/sdk/src/examples/spu_interrupt</p>
<b>spu_let</b>	<p>Demonstrates examples of standalone SPU programs called <i>spulets</i>. These are C-language programs that have been compiled to run on an SPU and can invoke C-library functions, such as <code>printf(3)</code>, <code>open(2)</code>, and so on. A spulet can be executed directly from the Linux® command prompt. The following entries are spulet examples.</p>	<p>/opt/cell/sdk/src/examples/spulet</p>
<b>spu_let: hello</b>	<p>Demonstrates a spulet version of <i>hello world</i>.</p>	<p>cell-examples-source 3.0 /opt/cell/sdk/src/examples/spulet/hello</p>
<b>spu_let: spe-sum</b>	<p>Demonstrates a simple spulet that computes a checksum for a file. The file contents are <code>mmap</code>'ed into the effective address space, and DMAs are used to stage data into the LS. The example illustrates how an</p>	<p>cell-examples-source 3.0 /opt/cell/sdk/src/examples/spulet/spe-sum</p>

	SPE-based filter might be constructed.		
<b>spu_let: spe-audio</b>	Demonstrates audio filtering on the SPU. Two examples are included. cpaudio copies one channel of a stereo audio file to the other channel. normalize normalizes the volume of a mono audio file. Both examples take raw, unsigned, halfword audio files as input, and they output the same format.	cell-examples-source	3.0 /opt/cell/sdk/src/examples/spulet/spe-audio
<b>spu_let: spe-cp</b>	Demonstrates how a simple spulet copies one file to another. The file contents are mmap'ed into the effective address space, and multi-buffered DMAs are used to stage data into and then out from LS.	cell-examples-source	3.0 /opt/cell/sdk/src/examples/spulet/spe-cp
<b>sync</b>	Demonstrates and tests low-level atomic and thread-type synchronization functions from the sample library libsync.	cell-examples-source	3.0 /opt/cell/sdk/src/examples/sync

## Cell/B.E. libraries

These example libraries show different sets of functions to the programmer. The examples are optimized for running on the PPE and SPE. Find detailed information about each of the routines in each of the libraries in the 3.0 document [SDK\\_Example\\_Library\\_API\\_v3.0.pdf](#).

**Table 3. Cell/B.E. libraries**

Example	Description	RPM Default installed directory	SDK
<b>fft</b>	Contains a collection of fast fourier transform routines. Includes the	cell-libs-source /opt/cell/sdk/src/lib/fft	3.0

	following two routines.		
<b>fft_1d_r2</b>	Performs a single precision, complex Fast Fourier Transform using Discrete Fourier Transform with radix-2 decimation in time. This function is built only for use on the SPE.	cell-libs-source /opt/cell/sdk/src/lib/fft	3.0
<b>fft_2d</b>	Transforms 4 rows of complex 2D data from the time domain to the frequency domain or vice versa. This function is available on both the PPE and the SPE.	cell-libs-source	3.0
<b>gmath</b>	Contains the following game math functions: cos8, cos8_v, cos14, cos14_v, cos18, cos18_v, pack_normal16, pack_normal16_v, pack_rgba8_v, sin8, sin8_v, sin14, sin14_v, sin18, sin18_v, spec9, spec9_v, unpack_normal16, unpack_normal16_v, unpack_rgba8, unpack_rgba8_v, pack_color8, pack_rgba8, set_spec_exponent9, specThresholds, unpack_color8.	cell-libs-source /opt/cell/sdk/src/lib/gmath	3.0
<b>image</b>	Contains a series of convolution and histogram functions.	cell-libs-source /opt/cell/sdk/src/lib/image	3.0
<b>large_matrix</b>	Contains several large matrix functions for the SPE. The matrix must fit the SPE local store.	cell-libs-source /opt/cell/sdk/src/lib/large_matrix	3.0
<b>matrix</b>	Contains matrix routines for the PPE and SPE.	cell-libs-source /opt/cell/sdk/src/lib/matrix	3.0
<b>misc</b>	Contains miscellaneous functions for PPE and SPE, including: calloc_align, clamp,	cell-libs-source /opt/cell/sdk/src/lib/misc	3.0

	clamp_0_to_1, clamp_0_to_1_v, clamp_minus1_to_1, clamp_minus1_to_1_v, clamp_v, free_align, load_vec_unaligned, malloc_align, max_float_v, max_int_v, max_vec_float3, max_vec_float4, max_vec_int3, max_vec_int4, min_float_v, min_int_v, min_vec_float3, min_vec_float4, min_vec_int3, min_vec_int4, rand_0_to_1, rand_0_to_1_v, rand_minus1_to_1, rand_minus1_to_1_v, rand_v, realloc_align, srand_v, store_vec_unaligned, copy_from_ls, copy_from_ls_aligned, copy_to_ls, copy_to_ls_aligned, ls_sync.		
<b>mpm</b>	Contains several functions that perform math on multi-precision large numbers on the SPE.	cell-libs-source /opt/cell/sdk/src/lib/mpm	3.0
<b>sync</b>	Contains several interfaces that are patterned after POSIX threads mutex and that condition variables for the Cell.	cell-libs-source /opt/cell/sdk/src/lib/sync	3.0
<b>vector</b>	Contains several vector math functions.	cell-libs-source /opt/cell/sdk/src/lib/vector	3.0

## Cell/B.E. tutorials

These examples are contained within the SDK Programming Tutorial.

**Table 4. Cell/B.E. tutorials**

Example	Description	RPM	SDK
---------	-------------	-----	-----

		Default installed directory	
<b>simple</b>	Demonstrates a simple program that creates 8 identical SPE threads that take turns printing messages to stdout and then terminate.	cell-tutorial-source	3.0
<b>euler</b>	Demonstrates how a simplified scalar function can be ported and accelerated for parallel execution on Cell.	cell-tutorial-source	3.0

## ALF Cell/B.E. examples

These examples show how the Accelerated Library Framework works on Cell/B.E.

**Table 5. ALF Cell/B.E. examples**

Example	Description	RPM Default installed directory	SDK
<b>BlackScholes_ALF</b>	Demonstrates how to perform a Black Scholes pricing model. It demonstrates a closed form solution for Black Scholes using ALF.	alf-examples-source /opt/cell/sdk/src/alf/BlackScholes_ALF/	3.0
<b>FFT16M_ALF</b>	Demonstrates an ALF version of the FFT16M workload in the cell-examples-source RPM.	alf-examples-source /opt/cell/sdk/src/alf/FFT16M_ALF	3.0
<b>hello_world</b>	Demonstrates a <i>hello_world</i> program that shows a simple ALF program. It contains an SPU computational kernel that issues a printf of the <i>hello_world</i> program.	alf-examples-source /opt/cell/sdk/src/alf/hello_world	3.0
<b>inout_buffer</b>	Demonstrates the usage of overlapped I/O buffers. The first example implements	alf-examples-source /opt/cell/sdk/src/alf/inout_buffer	3.0

	C=A+B, where A, B, and C are matrices. The second example implements A=A+B, where matrix A is overwritten by the result.		
<b>inverse_matrix_ovl</b>	Demonstrates the use of overlay with ALF by calculating the inverse of a block diagonal matrix.	alf-examples-source	3.0 /opt/cell/sdk/src/alf/inverse_matrix_ovl
<b>matrix_add</b>	Demonstrates how to program with ALF API. By adding two 1024x512 single precision floating point matrices, the sample code demonstrates how a simplified scalar function can be ported and accelerated for parallel execution on ALF. This simple example illustrates different features in ALF, including data partitioning on the host, data partitioning on the accelerator, overlapped in/out buffer, and data set.	alf-examples-source	3.0 /opt/cell/sdk/src/alf/matrix_add
<b>matrix_transpose</b>	Demonstrates how to program with the ALF API. By transposing a 1024x512 single precision floating point matrix, the sample code shows how a simplified scalar function can be ported and accelerated for parallel execution on ALF. The sample contains a scalar, non-ALF version of the code, a data partition on host version, a data partition on accelerator version, and a tuned version with SIMD.	alf-examples-source	3.0 /opt/cell/sdk/src/alf/matrix_transpose
<b>PI</b>	Demonstrates computing PI by	alf-examples-source	3.0 /opt/cell/sdk/src/alf/PI

	<p>Buffon's Needle method using ALF. It shows how to use the ALF context buffer for global parameters and how to collect computing results on each task instance. This program also shows a progress bar during the computation by using the sync point feature in ALF.</p>	
<p><b>pipe_line</b></p>	<p>Demonstrates the task dependency feature in ALF. It shows how task dependency is used in a two-stage pipeline application. The application is a simple simulation. An object P is placed in the middle of a flat surface with a bounding rectangular box. On each simulation step, the object moves a random distance in a random direction. It moves back to the initial position when it hits the side walls of the bounding box. The problem is to calculate the number of hits to the four walls in a given time period.</p>	<p>alf-examples-source 3.0 /opt/cell/sdk/src/pipe_line</p>
<p><b>task_context</b></p>	<p>Contains a collection of small samples that demonstrate the use of task context in ALF. Four included samples follow.</p>	<p>alf-examples-source 3.0 /opt/cell/sdk/src/task_context</p>
<p><b>task_context: dot_prod</b></p>	<p>Computes the dot product of two large vectors. It shows how to use the bundled work block distribution together with the task context to handle situations where the work block cannot hold the partitioned data because of a local</p>	<p>/opt/cell/sdk/src/task_context/dot_prod</p>

	memory size limit.	
<b>task_context: dot_prod_multi</b>	Computes the dot product of two large vectors. It shows how to use the multi-use workblocks together with work block parameter and context buffers.	/opt/cell/sdk/src/task_context/dot_prod_multi
<b>task_context: min_max</b>	Shows how to use the task context to keep the partial computing results for each task instance and then to combine these partial results into the final result.	/opt/cell/sdk/src/task_context/min_max
<b>task_context: table_lookup</b>	Shows how the task context buffer is used as a large lookup table to convert the 16-bit input data into 8-bit output data.	/opt/cell/sdk/src/task_context/table_lookup

## ALF hybrid examples

These examples show how the Accelerated Library Framework works on the hybrid system.

**Table 6. ALF hybrid examples**

Example	Description	RPM Default installed directory	SDK
<b>BlackScholes_ALF</b>	Demonstrates how to perform a Black Scholes pricing model. It demonstrates a closed form solution for Black Scholes using ALF.	alf-hybrid-examples-source /opt/cell/sdk/prototype/src/alf/BlackScholes_ALF/	3.0
<b>FFT16M_ALF</b>	Demonstrates an ALF version of the FFT16M workload in the cell-examples-source RPM.	alf-hybrid-examples-source /opt/cell/sdk/prototype/src/alf/FFT16M_ALF	3.0
<b>hello_world</b>	Demonstrates a <i>hello_world</i> program that shows a simple	alf-hybrid-examples-source /opt/cell/sdk/prototype/src/alf/hello_world	3.0

	ALF program. It contains an SPU computational kernel that issues a printf of the hello_world program.	
<b>inout_buffer</b>	Demonstrates the usage of overlapped I/O buffers. The first example implements $C=A+B$ , where A, B, and C are matrices. The second example implements $A=A+B$ , where matrix A is overwritten by the result.	alf-hybrid-examples-source0 /opt/cell/sdk/prototype/src/alf/inout_buffer
<b>inverse_matrix_ovl</b>	Demonstrates the use of overlay with ALF by calculating the inverse of a block diagonal matrix.	alf-hybrid-examples-source0 /opt/cell/sdk/prototype/src/alf/inverse_matrix_ovl
<b>matrix_add</b>	Demonstrates how to program with ALF API. By adding two 1024x512 single precision floating point matrices, the sample code demonstrates how a simplified scalar function can be ported and accelerated for parallel execution on ALF. This simple example illustrates different features in ALF, including data partitioning on the host, data partitioning on the accelerator, overlapped in/out buffer, and data set.	alf-hybrid-examples-source0 /opt/cell/sdk/prototype/src/alf/matrix_add
<b>matrix_transpose</b>	Demonstrates how to program with the ALF API. By transposing a 1024x512 single precision floating point matrix, the sample code shows how a simplified scalar function can be ported and accelerated for parallel execution on	alf-hybrid-examples-source0 /opt/cell/sdk/prototype/src/alf/matrix_transpose

	<p>ALF. The sample contains a scalar, non-ALF version of the code, a data partition on host version, a data partition on accelerator version, and a tuned version with SIMD.</p>
<p><b>PI</b></p>	<p>Demonstrates computing PI by Buffon's Needle method using ALF. It shows how to use the ALF context buffer for global parameters and how to collect computing results on every task instance. This program also shows a progress bar during the computation by using the sync point feature in ALF.</p> <p>alf-hybrid-examples-source00 /opt/cell/sdk/prototype/src/alf/PI</p>
<p><b>pipe_line</b></p>	<p>Demonstrates the task dependency feature in ALF. It shows how task dependency is used in a two-stage pipeline application. The application is a simple simulation. An object P is placed in the middle of a flat surface with a bounding rectangular box. On each simulation step, the object moves a random distance in a random direction. It moves back to the initial position when it hits the side walls of the bounding box. The problem is to calculate the number of hits to the four walls in a given time period.</p> <p>alf-hybrid-examples-source00 /opt/cell/sdk/prototype/src/pipe_line</p>
<p><b>task_context</b></p>	<p>Contains a collection of small samples that demonstrate the use of task context in ALF. (The same samples under task_context in Table 5.)</p> <p>alf-hybrid-examples-source00 /opt/cell/sdk/prototype/src/task_context</p>

## LAPACK examples

These examples show how to use the LAPACK library.

**Table 7. LAPACK examples**

Example	Description	RPM Default installed directory	SDK
<b>dsteqr_F</b>	Computes all eigenvalues and optionally eigenvectors of a symmetric tridiagonal matrix using the implicit QL or QR method.	lapack-examples-source /opt/cell/sdk/src/lapack-examples/dsteqr_F	3.0.0.3
<b>inverse_matrix</b>	Computes inverse of a matrix on Cell.	lapack-examples-source /opt/cell/sdk/src/lapack-examples/inverse_matrix	3.0.0.3

## BLAS examples

These examples show how to use the BLAS library.

**Table 8. BLAS examples**

Example	Description	RPM Default installed directory	SDK
<b>blas_simple</b>	Demonstrates the use of the BLAS library at the PPU level, which has the standard BLAS interface.	blas-devel /opt/cell/sdk/src/blas-examples/blas_simple	3.0
<b>blas_thread</b>	Demonstrates the use of the BLAS library with a multi-threaded PPU application.	blas-devel /opt/cell/sdk/src/blas-examples/blas_thread	3.0
<b>spulet</b>	Demonstrates the use of the SPU BLAS library containing SPU BLAS kernel routines.	blas-devel /opt/cell/sdk/src/blas-examples/spulet	3.0

## LIB FFT examples

These examples show how to use the FFT library.

**Table 9. LIB FFT examples**

Example	Description	RPM Default installed directory	SDK
<b>FFT1D</b>	Demonstrates the use of the prototype libfft code. In particular, this example demonstrates how the library can be made to perform with high efficiency on carefully crafted 1D FFTs.	libfft-examples-source /opt/cell/sdk/prototype/src/libfft/FFT1D	3.0
<b>FFT1D.spu_only</b>	Demonstrates the use of the prototype libfft code. In particular, this example demonstrates how the library can be made to perform with high efficiency on carefully crafted 1D FFTs on an SPU.	libfft-examples-source /opt/cell/sdk/prototype/src/libfft/FFT1D.spu_only	3.0
<b>FFT2D</b>	Demonstrates the use of the prototype libfft code. In particular, this example demonstrates how the library can be made to perform with high efficiency on carefully crafted 2D FFTs.	libfft-examples-source /opt/cell/sdk/prototype/src/libfft/FFT2D	3.0
<b>FFT2D.stream</b>	Demonstrates the use of the prototype libfft code. In particular, this example demonstrates how the library can be made to perform with high efficiency on carefully crafted 2D FFTs.	libfft-examples-source /opt/cell/sdk/prototype/src/libfft/FFT2D.stream	3.0

# Resources

## Learn

- Learn more about Cell/B.E. programming from the developerWorks series:
  - ["Programming high-performance applications on the Cell/B.E. processor"](#)
  - ["PS3 fab-to-lab"](#)
  - ["The little broadband engine that could"](#)
- Refer to the [Cell Broadband Engine documentation](#) section of the IBM Semiconductor Solutions Technical Library for a wealth of downloadable manuals, specifications, and more.
- Sign up for the [developerWorks newsletter](#) and get the latest developer news and Cell/B.E. happenings delivered to your inbox each week. Check *Power Architecture*® when you sign up to receive Cell/B.E. news in your newsletter.

## Get products and technologies

- Download [additional code samples](#) by choosing the IBM Cell Education link at this Information Center site.
- Get your copy of the [IBM SDK for Multicore Acceleration 3.0](#) or browse through the [library of Cell/B.E. documentation](#).
- Find all Cell/B.E.-related articles, discussion forums, downloads, and more at the IBM developerWorks [Cell Broadband Engine resource center](#): your definitive resource for all things Cell/B.E.
- Contact IBM about [custom Cell/B.E.-based or custom-processor based solutions](#).

## Discuss

- Check out the [Cell Broadband Engine Architecture forum](#) to get your technical questions about the processor answered. Juicy problems and answers from the forums are rounded up periodically and highlighted in the ["Forum watch" blog series](#).
- Go to the [Cell Broadband Engine/Power Architecture blog](#) for [news](#), downloads, instructional resources, and event notifications for Cell/B.E. and other Power Architecture-related technologies. You can find the popular ["Forum watch"](#) blog series (Q&A roundup), the ["FixIt"](#) technology updates, and the [Infobomb](#) quick-read technology introductions.

## About the authors

### Anita Bateman

Anita Bateman is a Senior IT Architect with the IBM Corporation. She has been with IBM doing software development and architecture since 1998. She is a certified architect with both IBM and The Open Group, and she has filed and published several patents. She holds an M.S. in computer science from the University of Texas at Austin and a B.S. in computer science from Hope College in Holland, Michigan. She is currently a Cell Broadband Engine solutions architect, working with partners and customers to adopt the IBM multi-core technology and to improve Cell/B.E. programmability.

---

### VanDung To

VanDung To received a computer science degree from Rice University in 2002 then joined IBM shortly afterwards. She is an advisory software engineer in the Quasar Design Center at IBM, and she has been working with Cell/B.E. technology since 2002.

## Trademarks

IBM and developerWorks are trademarks of IBM Corporation in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Cell Broadband Engine is a trademark of Sony Computer Entertainment Inc.

Other company, product, or service names may be trademarks or service marks of others.