# ADSP-BF707 EZ-Board™ Support Package v1.0.0 Release Notes

Thank you for installing the ADSP-BF707 EZ-Board™ Support Package (BSP). The BSP provides software and documentation in support of the ADSP-BF707 EZ-Board™.

The EZ Board is designed for use with CrossCore® Embedded Studio (CCES) for Analog Devices Processors software development tools. The CCES development environment aids advanced application code development and debug, such as:

- Create, compile, assemble, and link application programs written in C++, C, and assembly
- Load, run, step, halt, and set breakpoints in application programs
- Read and write data and program memory
- Read and write core and peripheral registers

- Plot memory

For more details on CCES, please visit www.analog.com/cces.

The ADSP-BF707 EZ-Board™ BSP provides comprehensive software support for the ADSP-BF707 EZ-Board™. Specifically, drivers, examples and code sketches are included for the following components:

- INA230 current-shunt and power monitor ADC driver.
- INA3221 shunt and bus voltage monitor ADC driver

The BSP also provides comprehensive examples which demonstrates the on-chip drivers and services.

The CCES Help environment provides complete hardware and software documentation.

## Release Dependencies

Requires CCES version 1.1.0

## Release Testing

The BSP has been tested with the ADSP-BF707 EZ-Board version 1.0, BOM 1.4

## License Checking

There are no license requirements for the ADSP-BF707 EZ-Board™ BSP.

## Installation Logging

The installer does not create a log file by default. If you encounter installation issues, you can generate an installation log file by running the installer from the command prompt.

Change to the directory containing downloaded installer executable and run the following from the command prompt:

```
ADI_ADSP-BF707_EZ-Board-Rel1.0.0.exe  /v"/l*v c:\temp\installer.log"
```

# Software Requirements

To build the example projects included in the ADSP-BF707 EZ-Board™ BSP, CrossCore Embedded Studio version 1.1.0 or later is required.

# Getting Started

## Adding a Driver to a Project

When adding an ADSP-BF707 Driver to your project, the IDE will add the sources for the driver to the CCES Project folders, starting at "system". There will be a folder specific to the driver(s) or service(s) you have added under this folder.

## Creating a project which includes a ADSP-BF707 driver/service

In order to create a project you should follow the instructions provided in the CrossCore Embedded Studio help. As part of the project creation, the page "Add-in selection" contains a list of all the available add-ins for the project that you are creating based on the installed products and the project's chosen processor and type. You can see the drivers in support of the ADSP-BF707 EZ-Board™ under the "Device Drivers and System Services" category. Within this catgeory you will see "ADSP-BF707 EZ-Board" which contains the drivers for the on-board peripherals (INA230 and INA3221). The on-chip peripheral drivers will be listed in "On-chip peripheral drivers" folder and the system services are listed in the "System Services" folder.

The ADSP-BF707 EZ-Board add-in generates a call to adi_initComponents(). For more information on adi_initComponents(), please refer to the CCES help

## Adding a ADSP-BF707 driver to an existing project

Every CrossCore Embedded Studio project contains a System Configuration file called `system.svc` which is located in the root of the project. The file is the IDE's interface for managing the various pre-written software components used in the "system" implemented by a project. Double-clicking any `system.svc` file in a navigation view opens that file in the System Configuration Utility which allows you to see the add-ins that you currently have in your project. Click on "Add..." and select the ADSP-BF707 EZ-Board Drivers add-in which is under the "Device Drivers and System Services" for the on-board INA230 and INA3221 ADC drivers. For adding on-chip peripherals drivers select the "On-chip peripheral drivers" and for the system services select the "System Services".

**Notes:**

- If the IDE detects that `adi_initComponents()` is not yet present in `main()`, it prompts you to add it and offers to insert it for you.

## Configuration

There are no ADSP-BF707 EZ-Board™ driver configuration options available in the IDE.

## Interrupts

CrossCore Embedded Studio provides a coherent interrupt management mechanism which allows for the same interface to be used in RTOS and non-RTOS applications. This means that interrupt service routines in all applications must be written in C and use the adi_int interface. Any thread-safety requirements or interactions with tasks are handled by the adi_int interface. For more information on the adi_int API, in CrossCore Embedded Studio go to Help > Search and enter adi_int.

Examples of the usage of this interrupt management mechanism are the System Services and Device Drivers provided with Crosscore Embedded Studio. By using the adi_int interface, the same services and drivers can be used in all applications regardless of whether an operating system is used.

## Sketches and Examples

## Sketches

CrossCore Embedded Studio provides a mechanism by which small code fragments, called sketches, can be generated with parameterized input provided by the user. The resulting code can then be copied and pasted to a project. Sketches for the on-board peripherals on the ADSP-BF707 EZ-Board™ are provided in the BSP. To locate the sketches specific to the ADSP-BF707 EZ-Board™ BSP, open up the example browser (Help -> Browse Examples) and then select ADSP-BF707 EZ-Board™ product in the "Product:" pulldown. The sketches for the on-chip drivers and system services can be located by selecting the CrossCore Embedded Studio product in the "Product" pulldown.

## Examples

### Power_On_Self_Test:

This example allows the user to test the many peripherals of the EZ-Board. This example is also pre-programmed into the on-board flash memory. By following the directions in the readme.html you can also program this example (or one of your own) into the EZ-Board flash. This POST was designed so that you can use the EZ-Board push buttons to select a specific test to run.

### Device_Programmer

This example allows the user to program the flash device on the ADSP-BF707 EZ-Board in conjunction with the Device Programmer Command-line tool. This code only serves as an example of how to program the flash and may not be optimized for the fastest programming speed. A pre-built binary exists so that users can just program the flash device without having to build the example.

### EXAMPLES FOR DRIVERS:

1. CRC examples which uses on-board CRC controller.
2. HADC example which uses the on-chip HADC controllerr
3. SPORT example which uses the on-board SPORT.
4. UART examples to demonstrate auto-baud and character echo using UART.
5. SPI_flash_read example which uses SPI driver to access the on-board serial flash memory (SST25WF040).
6. INA230 current-shunt and power monitor examples.

7. INA3221 example to read the 3.3 V source, VDD_INT and VDD_EXT bus and shunt voltage on the ADSP-BF707 EZ-Board.

### EXAMPLES FOR SERVICES:

1. Timer_Callback example demonstrates the General Purpose timer service.
2. Watchdog example demonstrates the Watchdog service.
3. EachDayAlaram example demonstrates the Real Time Clock service which programs RTC to generate alarm each day.
4. SetGetDateTime example demonstrates the Real Time Clock service to set and get the time.
5. MemCopyArrayMode example demonstrates the Memory DMA service to do a memory copy using Array Mode.
6. MemCopyListMode example demonstrates the Memory DMA service to do a memory copy using List Mode.
7. MemCopyOneShot1D example demonstrates the Memory DMA service to do a memory copy using One Dimensional DMA mode.
8. MemCopyOneShot1D example demonstrates the Memory DMA service to do a memory copy using Two Dimensional DMA mode.
9. MemCopyOneToMany example demonstrates the Memory DMA service to do a memory copy from single location to many locations in the memory.
10. GPIO examples demonstrate how to control the general purpose IO pins.
11. Power service examples demonstrates how to put the processor in and out of deep sleep and how to set the core and system frequencies.
12. System memory Protection Unit (SMPU) examples that demonstrate using Global Security and region protection.
13. System Protection Unit (SPU) examples demonstrate core and system write protection and slave protection
14. Trigger Routing Unit (TRU) example demonstrates TRU functionality using Memory DMA (MDMA).

## Location

In order to locate the ADSP-BF707 BSP examples and sketches, open CrossCore Embedded Studio's Example Browser which can be found in CrossCore Embedded Studio under Help. Select in the Product section "ADSP-BF707 EZ-Board v1.0.0" for a full list of examples and sketches.

# Documentation

API documentation for the drivers included in the ADSP-BF707 EZ-Board™ BSP can be found in CCES Help.

CrossCore ® Embedded Studio 1.1.0 > System Runtime Documentation > System Services and Device Drivers > ADSP-BF70x API Reference

General information on the driver model can be found in CCES help under

CrossCore ® Embedded Studio 1.1.0 > System Runtime Documentation > System Services and Device Drivers > Device Drivers Users Guide

API documentation for the off-chip drivers (controllers populated on the EZ-Board, INA230 and INA231) can be found under

ADSP-BF707 Board Support Package 1.0.0 > ADSP-BF707 EZ-KIT Lite® API Reference

# MISRA-C Support

MISRA C is a software development standard for the C programming language developed by the Motor Industry Software Reliability Association (MISRA). Its aims are to facilitate code safety, portability, and reliability in the context of embedded systems, specifically those systems programmed in ANSI C. The compiler detects violations of the MISRA rules at compile-time, link-time, and run-time.

# System Services and Device Driver Thread Safety

All system services and device drivers (SSDD) use mutexes and semaphores to ensure thread-safety. If an RTOS is present then the SSDD will use the RTOS mutex and semaphores. If an RTOS is not present then the SSDD will use a non-RTOS implementation of mutexes and semaphores (spin locks).

# Contacting Technical Support

Please contact your local ADI FAE and processor.support@analog.com.

# Known issues with the ADSP-BF707 EZ-Board™ Support Package (BSP)

None.