ADSP-SC58x EZ-KIT Lite® Board Support Package v1.0.0 Release Notes

Thank you for installing the ADSP-SC58x EZ-KIT Lite® Support Package (BSP). The BSP provides software and documentation in support of the ADSP-SC58x EZ-KIT Lite®.

The EZ-Kit Lite 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-SC58x EZ-KIT Lite® BSP provides comprehensive software support for the ADSP-SC58x EZ-KIT Lite®. Specifically, drivers, examples and code sketches are included for the following components:

- INA3221 shunt and bus voltage monitor ADC driver
- 12-Channel, 192 kHz, 24-Bit DAC (ADAU1962A) 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 CrossCore® Embedded Studio version 2.0.0

Release Testing

The BSP has been tested with the ADSP-SC589 EZ-KIT version 1.0, BOM 1.4 and ADSP-SC584 EZ-KIT version 0.1 BOM 1.1.

License Checking

There are no license requirements for the ADSP-SC58x EZ-KIT Lite® 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-SC58x_EZ-KIT-Rel1.0.0.exe  /v"/l*v c:\temp\installer.log"
```

Software Requirements

To build the example projects included in the ADSP-SC58x EZ-KIT Lite® BSP, CrossCore® Embedded Studio version 2.0.0 or later is required.

Getting Started

Adding a Driver to a Project

When adding an ADSP-SC58x 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-SC58x 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-SC58x EZ-KIT™ under the "Device Drivers and System Services" category. Within this catgeory you will see "ADSP-SC58x EZ-KIT" 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-SC58x EZ-KIT Lite® add-in generates a call to adi_initComponents(). For more information on adi_initComponents(), please refer to the CCES help

Adding a ADSP-SC58x 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-SC58x EZ-KIT 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-SC58x EZ-KIT Lite® 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.

Code Sketches and Examples

Code Sketches

CrossCore® Embedded Studio provides a mechanism by which small code fragments, called code 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-SC58x EZ-KIT® are provided in the BSP. To locate the sketches specific to the ADSP-SC58x EZ-KIT® BSP, open up the example browser (Help -> Browse Examples) and then select ADSP-SC58x EZ-KIT Lite 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-KIT. 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-KIT flash. This POST was designed so that you can use the EZ-KIT push buttons to select a specific test to run.

Device_Programmer

This example allows the user to program the flash device on the ADSP-SC58x EZ-KIT Lite® 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. ADC Example which uses the ADAU1979 ADC and the ADAU1962a DAC to operate either a four channel audio playback or a four channel analog loopback
2. INA3221 example to read the 3.3 V source, VDD_INT and VDD_EXT bus and shunt voltage on the ADSP-SC58x EZ-KIT
3. ASRC Example which uses the ASRC, PCG, ADAU1979 ADC and the ADAU1962a DAC to operate either a four channel audio playback

or a four channel analog loopback
4. CRC examples which use the on-board CRC controller
5. FIR example which demonstrates how to configure the FIR device driver in floating point single rate mode
6. HADC example which uses the on-chip HADC controller
7. HAE examples which uses the on-chip HAE controller
8. IIR example which demonstrates how to configure the IIR device driver in channel interrupt mode
9. Linkport loopback example which uses the linkport driver
10. SINC example which demonstrates how to configure the SINC filter device driver
11. SPDIF examples which uses the SPDIF interface
12. SPI_flash_read example which uses SPI driver to access the on-board serial flash memory
13. SPI example which tests the 1K-bit SPI Serial EEPROM (Microchip 25LC010A) memory device.
14. TMU examples which use the on-board Thermal Monitoring Unit
15. TWI example which demonstrates how to configure the Silicon Labs Si5356A clock generator output
16. UART examples which demonstrate auto-baud and character echo using UART

Examples for Services:

1. Timer_Callback example which demonstrates the General Purpose timer service
2. Watchdog example which demonstrates the Watchdog service
3. EachDayAlarm example which uses the Real Time Clock service to generate an alarm each day
4. SetGetDateTime example which uses the Real Time Clock service to set and get the time
5. Power service example which demonstrates how to use the Power Service to change the processor core clock and system clock frequencies
6. Trigger Routing Unit (TRU) example which demonstrates the TRU functionality using Memory DMA (MDMA)
7. PWM example which demonstrates the use of the PWM driver to output a signal capable of driving a Hitec servo motor. This example requires driver sources that are not in CrossCore® Embedded Studio 2.0.0 M6 but will be in the production release of CrossCore® Embedded Studio 2.0.0

Location

In order to locate the ADSP-SC58x 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-SC58x EZ-KIT v1.0.0" for a full list of examples and sketches.

Documentation

API documentation for the ADSP-SC58x on-chip drivers and services for both SHARC+ and Cortex-A5 can be found in the CCES Help.
    CrossCore® Embedded Studio 2.0.0 > System Runtime Documentation > System Services > ADSP-SC58x API Reference and
    CrossCore® Embedded Studio 2.0.0 > System Runtime Documentation > System Services > ADSP-SC58x Cortex API Reference
General information on the driver model can be found in CCES help under
    CrossCore® Embedded Studio 2.0.0 > System Runtime Documentation > Device Drivers Users Guide
API documentation for the off-chip drivers (controllers populated on the EZ-KIT, ADAU1962A and INA231) in a future version of the BSP will be found under
    ADSP-SC58x Board Support Package 1.0.0 > ADSP-SC58x 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 processor.support@analog.com.

Known issues with the ADSP-SC58x EZ-KIT® Board Support Package (BSP)

None.