ANALOG DEVICES

ADI_AD7147
DEVICE DRIVER

DATE: MARCH 18, 2009

Table of Contents

List of Tables

Document Revision History

Date	Description of Changes
March 11, 2009	Initial draft
March 18, 2009	First release

Table 1 – Revision History

1 Introduction

This document describes use of the Analog Devices AD7147 Capacitive Touch (CapTouch) device driver, adi_ad7147.

The Analog Devices AD7147 CapTouch controller is a generally available integrated circuit that is designed into the BLACKFIN LANDSCAPE LCD EZ-EXTENDER expansion card. The CapTouch controller is used to monitor an array of single-electrode, capacitance-sensing touch pad "buttons" arranged below the combination LCD/TouchScreen display on the extender card; it detects touch events by sensing minute changes in virtual capacitance between the sensor pad and the user's finger.

The CapTouch controller has on-chip calibration logic to compensate for changes in the ambient environment. It uses a 16-bit Sigma-Delta A/D converter with 13 dedicated input channels and an input multiplexer to perform "Capacitance-to-Digital Conversions" (CDC). It has numerous operational modes, input conditioning and conversion options. The CapTouch controller is available in two versions, based on the serial communications channel protocol build into the IC: the AD7147, which is SPI based; and the AD7147-1, which TWI-based (similar to I2C). The AD7147-1 (TWI) controller is used on the BLACKFIN LANDSCAPE LCD EZ-EXTENDER card.

The adi_ad7147 CapTouch device driver described here allows the AD7147 to be used in the Blackfin System Services and Device Driver environment. The CapTouch device driver isolates the low-level configuration and communication tasks of managing the AD7147 from the application by providing a high-level integrated programming API alternative.

The Analog Devices AD7147 CapTouch driver is tested using the BLACKFIN LANDSCAPE LCD EZ-EXTENDER expansion card interfaced to both the ADSP-BF518 EZ-BOARD and the ADSP-BF526 EZ-BOARD development platforms.

1.1 Scope

This document is intended for software engineers. It provides all the information necessary to write an application using the AD7147 driver. Previous or concurrent training in device drivers and digital signal processing (DSP) would be highly advantageous to understanding this document.

This document does not describe the internal design of the driver, except when this information is required to use the driver in an application.

1.2 Organization of this Document

This Document begins with an overview of the supported devices and platforms.

A "Quick Start" guide provides an example to get the user started in writing a real application.

It then describes the device driver source files which comprise the driver and explains how to add the appropriate files to the application to use the driver.

The "Lower Level Drivers" section lists the dependencies of this driver upon other drivers.

The "Required Resources" section describes the memory requirements for buffers and the System Services.

The "Detailed Device Driver API" section is the programmer's reference. It provides the enumerated values, macros and API functions of the AD7147 driver, as well as explaining their meaning and usage within the context of the Device Manager Service.

Finally, references and resources for further reading complete the manual.

1.3 Acronyms

The following acronyms are used within this document.

Acronym	Meaning
ACK	Acknowledge Handshake
API	Application Programming Interface
CDC	Capacitance-to-Digital Conversion
DCB	Deferred CallBack
DPJ	VisualDSP++ project filename extension
DSP	Digital Signal Processing or Digital Signal Processor
GPIO	General-Purpose I/O
IC	Integrated Circuit
IRQ	Interrupt Request
ISR	Interrupt Service Routine
LDF	Linker Description File
PCB	Printed Circuit Board
PLL	Phase Locked Loop
RTOS	Real-Time Operating System
SPI	A full-duplex Serial Peripheral Interface
TWI	A half-duplex Two Wire Interface. Similar to I2C
I/O	Input/Output
I2C	Inter-Integrated Circuit interface; a half-duplex multi-master low-speed serial computer bus communications protocol. Similar to TWI.
VisualDSP++	Visual Digital Signal Processing Development Environment
VDK	VisualDSP++ Kernel (Analog Devices proprietary RTOS)

1.4 Copyright, Disclaimers and Trademark Statements

Copyright Information

Disclaimer

Analog Devices, Inc. reserves the right to change this product without prior notice. Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use; nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under the patent rights of Analog Devices, Inc.

Trademark and Service Mark Notice

Analog Devices, the Analog Devices logo, Blackfin, SHARC, TigerSHARC, CROSSCORE, VisualDSP, VisualDSP++, VisualAudio, EZ-KIT Lite, EZ-BOARD, EZ-Extender and Collaborative are trademarks and/or registered trademarks "®" of Analog Devices, Inc.
All other brand and product names are trademarks or service marks of their respective owners.

Trademarks and Service Marks must be reproduced according to ADI's Trademark Usage guidelines. Any licensee wishing to reproduce ADI's Trademarks and Service Marks must obtain and follow these guidelines for the specific marks to be reproduced.

2 Overview

2.1 Supported Devices

The adi_ad7147 device driver supports the following physical devices:

- Analog Devices AD7147-1 (I2C/TWI-based) CapTouch Programmable Controller for Single-Electrode Capacitance Sensors

2.2 Supported Platforms

The adi_ad7147 device driver is tested on the following platform combinations.

- BLACKFIN LANDSCAPE LCD EZ-EXTENDER

With either:

- ADSP-BF518F EZ-BOARD
- ADSP-BF526 EZ-BOARD

2.3 Development Tools

Build Tools:

- Analog Devices VisualDSP++ 5.0, Update 6 (minimum)

Debugging Tools:

One of:

- Analog Devices ADDS HPPCI-ICE emulator
- Analog Devices ADDS HPUSB-ICE emulator
- Analog Devices ADDS USB-ICE emulator

2.4 Test Suite

No Blackfin Test Project (BTP) tests exist for the adi_ad7147 device driver at this time. The SketchPad example application (install_root/VisualDSP++ 5.0/Blackfin/Landscape LCD Ez-Extender/SketchPad) demonstrates the use of this driver. See http://sourceforge.net/projects/bfin-test-proj for details on the public domain Blackfin Test Project.

3 Quick Start Guide

SketchPad is a stand-alone example application shipped as part of the VisualDSP++ 5.0 release starting with Update 6. It is installed under the Examples directory (under the "Landscape LCD EZ-Extender" subdirectory).

SketchPad is a complete, high-level working example application illustrating the use of the CapTouch device driver, the TouchScreen device driver and the LCD device driver, all of which is integrated on the stand-alone plug-in expansion board, the BLACKFIN LANDSCAPE LCD EZ-EXTENDER.

SketchPad illustrates all of the configuration settings and normal operation of the CapTouch device driver (as well as the TouchScreen and LCD display) and serves as the Quick Start Guide and Example Code for developing applications with the CapTouch device driver.

The calibration and environmental settings used by SketchPad were obtained using the recommended calibration procedure detailed in the AD7147 Data Sheet. Different hardware will require calibration of that hardware resulting in different calibration and environmental settings.

Please refer to the "README.txt" file under either of the following directories for complete details of configuring the CapTouch controller (as well as for configuring the TouchScreen controller and the LCD display module) on the BLACKFIN LANDSCAPE LCD EZ-EXTENDER board.

- "C:\Program Files\Analog Devices\VisualDSP 5.0\Blackfin\Examples\Landscape LCD EZ-EXTENDER\SketchPad\ADSP-BF518\README.txt"
- "C:\Program Files\Analog Devices\VisualDSP 5.0\Blackfin\Examples\Landscape LCD EZ-EXTENDER\SketchPad\ADSP-BF526\README.txt"

4 Device Driver Files

The files listed below comprise the device driver API and source files.

4.1 Include Files

The driver sources include the following include files, located in the default installation directory: "C:\Program Files\Analog Devices\VisualDSP 5.0\Blackfin\include".

Include File	Description
services\services.h	This is the common include file for all System Services.
drivers\adi_dev.h	This is the common include file for the Device Manager.
drivers\deviceaccess\adi_device_access.h	This is the include file for the Device Access System Service.
drivers\captouch\adi_ad7147.h	This is the include file for the CapTouch Device Driver.

4.2 Source Files

The driver sources are contained in the following files, as located in the default installation directory: "C:\Program Files\Analog Devices\VisualDSP 5.0\Blackfin\lib\src":

Source File	Description
drivers\captouch\adi_ad7147.c	This is the source file for the CapTouch Device Driver.

4.3 Project Files

Applications deploying the AD7147 add the CapTouch device driver files directly to the application's project. The SketchPad example application project file illustrates this.

The CapTouch device driver is available as source code only and is not included in any pre-built device driver libraries. This is because the CapTouch controller is an external device that is not integral to the Blackfin Processor.

5 Lower Level Drivers

The AD7147 device driver is considered a "high-level" device driver; it leverages various "low-level" drivers, as needed. The programming API for the AD7147 device driver defines command control codes, parameters and return codes to manage these low-level drivers transparently. The application need not interact with the low-level drivers directly.

The complete AD7147 device driver API is documented in the "Supported Features of the Device Driver" chapter of this document. The same device driver (adi_ad7147) is used to manage the AD7147 and the AD7147-1 controllers.

5.1 Lower Level TWI Device Driver

The AD7147-1 CapTouch controller IC (used on the BLACKFIN LANDSCAPE LCD EZ-ENTENDER) employs a TWI-based (I2C) serial communications channel for managing the device. All control and data transactions between the Blackfin processor and the AD7147-1 controller are conducted over this channel. The AD7147-1 device driver (adi_ad7147) defines control codes, parameters and return codes for managing the AD7147-1 controller. All TWI device driver communication with the AD7147-1 controller is managed by the AD7147-1 device driver and no direct management of the TWI protocol or TWI device driver is required.

The application uses the CapTouch device driver to configure the TWI channel parameters such as device number, device address, timing, retries, etc. The SketchPad example application illustrates TWI configuration from the high-level CapTouch device driver API.

5.2 Lower Level Device Access Device Driver

The AD7147 (SPI-based) and the AD7147-1 (TWI-based) CapTouch controllers both employ the Device Access Device Driver for underlying implementation of the TWI serial communications protocol. All low-level Device Access transactions are transparent to the application.

6 Required Resources

6.1 System Services

Unless explicitly noted in the sections below, this device driver uses the System Services to access and control any required hardware. The information in this section may be helpful in determining the resources this driver requires, such as the number of interrupt handlers or number of DMA channels etc., from the System Services.

Wherever possible, this device driver uses the System Services to perform the necessary low-level hardware access and control.

6.2 System Services Initialization

The application code must initialize each system service before use. If applicable, VDK initialization *must* precede System Service initializations (using “VDK_Initialize()” or “VDK::Initialize()”, depending on namespace, see *VDK User’s Guide*). System Services *and* VDK initializations are illustrated in the SketchPad example application (see file “main.cpp”).

System services are typically initialized with a series of calls to “adi_xxx_Init()”, where xxx is “ebiu”, “pwr”, “int”, “dcb”, etc. The SketchPad example application has rolled up the System Service initializations it requires into a single “adi_ssl_Init()” call in file “…/SketchPad/adi_ssl_Init.c”.

6.3 Interrupt Manager System Service

The CapTouch device driver uses interrupts to manage both the serial communication internal buffering and to respond to CapTouch touch events. Both system interrupts and core events are described. The CapTouch device driver uses the interrupt manager and the low-level device drivers to hook interrupt handlers as follows.

NOTE: The following interrupts and IVG allocations are unique to the SketchPad example application. Different applications and hardware platforms will allocate these resources differently.

Peripheral ID	Default IVG	Description
ADI_INT_TWI	ADI_INT_IVG_10	This is a TWI System Service interrupt which is raised on completion of TWI data buffer transfers. The TWI System Service handles the interrupt. The TWI interrupt ID and IVG are identical on the ADSP-BD518 and ADSP-BF526 target platforms.
ADI_INT_PORTH_INTA	ADI_INT_IVG_11	***ADSP-BF518 Target ONLY***. This interrupt is routed from the AD7147 CapTouch controller "INT~" output (pin 17) to Port H, bit-5 (PH5, pin 155) on the Blackfin. The CapTouch device driver supports a registration API to install an application layer interrupt callback handler to process this interrupt.
ADI_INT_PORTG_INTA	ADI_INT_IVG_12	***ADSP-BF526 Target ONLY***. This interrupt is routed from the AD7147 CapTouch controller "INT~" output (pin 17) to Port G, bit-12 (PG12, pin J2) on the Blackfin. The CapTouch device driver supports a registration API to install an application layer interrupt callback handler to process this interrupt.

6.4 Deferred Callbacks Manager System Service

A deferred callback (DCB) is a mechanism for delaying non-critical extended interrupt processing to idle cycles, outside context of the formal interrupt service routine (ISR). Simple interrupt processing may be performed within the ISR, but the guiding System Services principal is to defer non-critical interrupt processing to avoid blocking and missed interrupts. The DCB system service provides for registering, queuing and dispatching callbacks to perform extended interrupt processing. Using DCB enhances overall system performance and callbacks are scheduled during idle cycles.

The CapTouch device driver passes the DCB manager handle (or NULL), as defined in the application layer, when the CapTouch device driver is opened.

The SketchPad example application enables deferred callbacks by default. Deferred callbacks may be disabled by commenting out the "USE_DEFERRED_CALLBACKS" macro in source file: "…\SketchPad\adi_ssl_init.h".

6.5 Power Management System Service

The CapTouch device driver inherits Power Management settings from the application layer; it does not use the Power Management system service explicitly.

The SketchPad example application (which runs the CapTouch driver) uses the Power Management system service to set the system and core clock frequencies and the PLL divider on the target hardware. Function call "adi_pwr_SetFreq()" in source file "main.cpp" illustrates this use, in which the Blackfin system and core clock frequencies are set to their maximum values.

6.6 Port Control System Service

The Port Control system service defines the appropriate set of processor-dependant port mapping defines for use by other system services. The Port Control system service resolves Blackfin port routing of internal device control and data, pin mappings, interrupts, flags, and general purpose I/O multiplexing. The Port Control system service is used implicitly by other system services.

Neither the CapTouch device driver nor the SketchPad example application make explicit use of the Port Control system service, other than to initialize it at the application layer.

6.7 Programmable Flags System Service

The Flag system service is used to manage the general-purpose port I/O capabilities of the Blackfin interface ports. Primarily, the Blackfin interface ports are configurable to service various internal devices, e.g., Ethernet, PPI, SPI, TWI, Timers, UART, SPORT, etc. But, the interface ports may also service general purpose I/O (GPIO). The Flag system service provides an API to configure GPIO, including management of the port interface multiplexing registers, function enable register bits, flag direction, interrupt mapping, interrupt callback registration, etc.

The CapTouch device driver requires a single flag pin to capture the singular CapTouch controller interrupt output. The specific pin used is target platform-specific. The CapTouch API supports a set of application-level commands for configuring interrupts, mapping pins and registering application-level callbacks. The CapTouch device driver uses the Flag system service (transparently to the application) to configure the Blackfin processor to receive interrupt inputs from the CapTouch controller on one of the GPIO-configured port pins. The CapTouch application-layer need only designate the desired pin and interrupt group.

6.8 Pin Usage

The AD7147-1 CapTouch device driver uses the TWI serial interface. Depending on the target platform, the TWI serial interface pins are routed to different Blackfin port locations. The CapTouch device driver also uses one flag input to sense the interrupt line from the CapTouch controller.

The following tables define the CapTouch device driver pin and multiplexer assignments used by the SketchPad example application. (Note: the SketchPad example application is based on the hardware resources of the BLACKFIN LANDSCAPE LCD EZ-EXTENDER expansion card, which are specific and unique to that card, as well as the particular Blackfin processor on the respective EZ-BOARD evaluation system.)

ADSP-BF518 Pin assignments for AD7147-1 (TWI)

Pin	Function	Multiplexer Status
PH5	Interrupt Input	GPIO
PJ0	SCL (TWI Clock)	Dedicated pin
PJ1	SDA (TWI Data)	Dedicated pin

ADSP-BF526 Pin assignments for AD7147-1 (TWI)

Pin	Function	Multiplexer Status
PG12	Interrupt Input	GPIO
PJ2	SCL (TWI Clock)	Dedicated pin
PJ3	SDA (TWI Data)	Dedicated pin

6.9 Memory Allocation

Device drivers typically consume some amount of system resources for buffers and System Services.

Dynamic memory is not allocated by the Device Drivers or the System Services, but must be supplied by the application.

The application can allocate the memory resources required by the device driver, using "supply macros" to select the amount of base memory and/or the amount of incremental memory required to support the required service functionality.

6.9.1 Buffer Allocation

The application allocates the memory it needs for the buffers which are passed to the 'adi_dev_Read()' and 'adi_dev_Write()' functions. A compiler "section" directive may be used, to force the buffer to a specific memory section, as defined in the default linker description file (LDF).

```
section("sdram0_bank0") volatile u16 sFrame0[FRAMESIZE];
```

A customized linker description file (LDF) can also be used to define memory sections for buffers, as shown below.

```
section("buffer0_sdram") volatile u16 sFrame0[FRAMESIZE];
section("buffer1_sdram") volatile u16 sFrame1[FRAMESIZE];
```

The Quick Start reference guide provides examples of buffer creation and allocation.

The CapTouch device driver manages all buffer allocations for the TWI serial data communications channels internally. The application need not address buffer allocations.

6.9.2 Allocating Memory for the Deferred Callback Service

In this example there are three deferred callbacks to be supported by the Deferred Callback Service.

```
#define ADI_SSL_DCB_NUM_SERVERS 3

static u8 DeferredCallbackServiceData [ADI_DCB_QUEUE_SIZE * ADI_SSL_DCB_NUM_SERVERS];

adi_dcb_Init(DeferredCallbackServiceData, sizeof(DeferredCallbackServiceData), &i,
ADI_SSL_ENTER_CRITICAL);
```

The SketchPad example application defines one DCB server, see include file "adi_ssl_init.h".

6.9.3 Allocating Memory for the DMA Manager Service

In this example there are four DMA channels for the DMA Manager Service to support.

```
#define ADI_SSL_DMA_NUM_CHANNELS 4

static u8 DMAServiceData  [ADI_DMA_BASE_MEMORY + (ADI_DMA_CHANNEL_MEMORY *
ADI_SSL_DMA_NUM_CHANNELS)];

adi_dma_Init(DMAServiceData, sizeof(DMAServiceData), &i, &adi_dma_ManagerHandle,
ADI_SSL_ENTER_CRITICAL);
```

The SketchPad example application defines one DMA channel for streaming display frames to the LCD over the PPI device driver, see include file "adi_ssl_init.h". This has nothing to do with the CapTouch device driver.

6.9.4 Allocating Memory for the Flag Manager Service

In this example there are eight flags for the Flag Manager Service to support.

```
#define ADI_SSL_FLAG_NUM_CALLBACKS 8

static u8 FlagServiceData [ADI_FLAG_CALLBACK_MEMORY *  ADI_SSL_FLAG_NUM_CALLBACKS];

adi_flag_Init(FlagServiceData, sizeof(FlagServiceData), &i, ADI_SSL_ENTER_CRITICAL);
```

The SketchPad example application defines two Flag callbacks, one for the CapTouch controller, and one for the TouchScreen controller, see include file "adi_ssl_init.h".

6.9.5 Allocating Memory for the Semaphore Service

In this example there are two semaphores for the Semaphore Service to support.

```
#define ADI_SSL_SEM_NUM_SEMAPHORES 2

static u8 SemaphoreServiceData [ADI_SEM_SEMAPHORE_MEMORY *
ADI_SSL_SEM_NUM_SEMAPHORES];

adi_sem_Init(SemaphoreServiceData, sizeof(SemaphoreServiceData), &i,
ADI_SSL_ENTER_CRITICAL);
```

The SketchPad example application defines one Semaphore for managing inter-thread data mailbox notifications used for passing data from the CapTouch and TouchScreen threads to the LCD display thread, see include file "adi_ssl_init.h".

6.9.6 Allocating Memory for the Device Manager Service

In this example there are three devices for the Device Manager Service to support.

```
#define ADI_SSL_DEV_NUM_DEVICES 3

static u8 DevMgrData [ADI_DEV_BASE_MEMORY + (ADI_DEV_DEVICE_MEMORY *
ADI_SSL_DEV_NUM_DEVICES)];

adi_dev_Init(DevMgrData, sizeof(DevMgrData), &i, &adi_dev_ManagerHandle,
ADI_SSL_ENTER_CRIITICAL);
```

The SketchPad example application defines six devices: CapTouch, LCD, PPI, SPI, TouchScreen, and TWI, see include file "adi_ssl_init.h".

6.9.7 Allocating Memory for the Interrupt Manager Service

Multiple interrupt sources can share the same interrupt vector group (IVG), multiple handlers can be hooked within the same IVG. The first handler to be hooked, is referred to as the Primary Interrupt Handler, while subsequent handlers to be hooked are referred to as the Secondary Interrupt Handlers. In this example there are four secondary handlers for the Interrupt Manager Service to support.

```
#define ADI_SSL_INT_NUM_SECONDARY_HANDLERS 4

static u8 DevMgrData InterruptServiceData  [ADI_INT_SECONDARY_MEMORY *
ADI_SSL_INT_NUM_SECONDARY_HANDLERS];

adi_int_Init(InterruptServiceData, sizeof(InterruptServiceData), &i,
ADI_SSL_ENTER_CRITICAL);
```

The SketchPad example application defines one secondary interrupt handler; for switching between the TouchScreen PENIRQ and PEN/DAV interrupt modes, which share the same pin.

7 Detailed Device Driver API

This section describes which of the available device driver features are supported by this device driver. Refer to the "VisualDSP++ 5.0 Device Drivers and System Services User Manual for Blackfin Processors" for details.

7.1 Directionality

The driver supports the dataflow directions listed in the table below.

ADI_DEV_DIRECTION	Description
ADI_DEV_DIRECTION_INBOUND	Supports the reception of data in through the device.
ADI_DEV_ DIRECTION_OUTBOUND	Supports the transmission of data out through the device.
ADI_DEV_ DIRECTION_BIDIRECTIONAL	Supports both the reception of data and transmission of data through the device.

Table 2 – Supported Dataflow Directions

7.2 Dataflow Methods

The driver supports the dataflow methods listed in the table below.

ADI_DEV_MODE	Description
ADI_DEV_MODE_SEQ_CHAINED	Supports the sequential I/O dataflow method

Table 3 – Supported Dataflow Methods

7.3 Buffer Types

The driver supports the buffer types listed in the table below.

The CapTouch device driver only uses a single data buffer type for passing data over the TWI channel.

- ADI_DEV_1D_BUFFER
  - Linear one-dimensional buffer
  - pAdditionalInfo – Conveys the device address control word prefix.
  - ProcessedFlag – Conveys buffer transfer completion status.
  - ElementCount – Conveys the number of bytes per transaction (2)†.

† 2-byte (16-bit) buffering is used throughout the driver. See Data Sheet for examples of TWI read/write timing.

7.4 Command IDs

This section enumerates the device driver commands. The commands are divided into three sections. The first section describes commands supported directly by the Device Manager. The next section describes common Device Manager commands. The remaining section describes driver specific commands.

Commands are sent to the device driver via the adi_dev_Control() function. The adi_dev_Control() function accepts three arguments:

- DeviceHandle – This parameter is a ADI_DEV_DEVICE_HANDLE type that uniquely identifies the device driver. This handle is provided to the client in the adi_dev_Open() function call.
- CommandID – This parameter is a u32 data type that specifies the command ID.
- Value – This parameter is a void * whose value is context sensitive to the specific command ID.

The sections below enumerate the command IDs that are supported by the driver and the meaning of the Value parameter for each command ID.

7.4.1 Device Manager Commands

The commands listed below are supported and processed directly by the Device Manager. As such, all device drivers support these commands.

- ADI_DEV_CMD_TABLE
  - Table of command pairs being passed to the driver
  - Value – ADI_DEV_CMD_VALUE_PAIR *
- ADI_DEV_CMD_END
  - Signifies the end of a command pair table
  - Value – ignored
- ADI_DEV_CMD_PAIR
  - Single command pair being passed
  - Value – ADI_DEV_CMD_PAIR *
- ADI_DEV_CMD_SET_SYNCHRONOUS
  - Enables/disables synchronous mode for the driver
  - Value – TRUE/FALSE

7.4.2 Common Commands

The command IDs described in this section are common to many device drivers. The list below enumerates all common command IDs that are supported by this device driver.

- ADI_DEV_CMD_REGISTER_READ
  - Reads a single device register
  - Value – ADI_DEV_ACCESS_REGISTER * (register specifics)
- ADI_DEV_CMD_REGISTER_FIELD_READ
  - Reads a specific field location in a single device register
  - Value – ADI_DEV_ACCESS_REGISTER_FIELD * (register specifics)
- ADI_DEV_CMD_REGISTER_TABLE_READ
  - Reads a table of selective device registers
  - Value – ADI_DEV_ACCESS_REGISTER * (register specifics)
- ADI_DEV_CMD_REGISTER_FIELD_TABLE_READ
  - Reads a table of selective device register fields
  - Value – ADI_DEV_ACCESS_REGISTER_FIELD * (register specifics)
- ADI_DEV_CMD_REGISTER_BLOCK_READ
  - Reads a block of consecutive device registers
  - Value – ADI_DEV_ACCESS_REGISTER_BLOCK * (register specifics)
- ADI_DEV_CMD_REGISTER_WRITE
  - Writes to a single device register
  - Value – ADI_DEV_ACCESS_REGISTER * (register specifics)
- ADI_DEV_CMD_REGISTER_FIELD_WRITE
  - Writes to a specific field location in a single device register
  - Value – ADI_DEV_ACCESS_REGISTER_FIELD * (register specifics)
- ADI_DEV_CMD_REGISTER_TABLE_WRITE
  - Writes to a table of selective device registers
  - Value – ADI_DEV_ACCESS_REGISTER * (register specifics)
- ADI_DEV_CMD_REGISTER_FIELD_TABLE_WRITE
  - Writes to a table of selective device register fields
  - Value – ADI_DEV_ACCESS_REGISTER_FIELD * (register specifics)
- ADI_DEV_CMD_REGISTER_BLOCK_WRITE
  - Writes to a block of consecutive device registers
  - Value – ADI_DEV_ACCESS_REGISTER_BLOCK * (register specifics)

7.4.3 Device Driver Specific Commands

The command IDs listed below are supported and processed by the CapTouch device driver. These command IDs are unique to the CapTouch device driver.

- ADI_AD7147_CMD_SET_TWI_DEVICE_NUMBER,
  - Sets TWI Device Number to use to access AD7147 registers
  - Value = u32 (TWI Device number to use)
  - Default = 0xFF (TWI device number invalid)
- ADI_AD7147_CMD_SET_TWI_CONFIG_TABLE,
  - Sets TWI Configuration table specific to the application
  - Value = pointer to ADI_DEV_CMD_VALUE_PAIR
  - Default = NULL pointer
- ADI_AD7147_CMD_SET_TWI_DEVICE_ADDRESS,
  - Sets TWI Device Address specific to the HW setting
  - Value = u32 (TWI Device address to use)
  - Default = 0x00 (TWI device address invalid)
- ADI_AD7147_CMD_INSTALL_INTIRQ,
  - Sets AD7147 driver to monitor INTIRQ interrupts
  - Value = ADI_AD7147_INTERRUPT_PORT
  - Default = Interrupt monitoring disabled
- ADI_AD7147_CMD_UNINSTALL_INTIRQ,
  - Removes INTIRQ interrupt from being monitored by AD7147 driver
  - Value = NULL
  - Default = none
- ADI_AD7147_CMD_DISABLE_INTIRQ,
  - Disable INTIRQ interrupt from being monitored by AD7147 driver
  - Value = NULL
  - Default = none
- ADI_AD7147_CMD_REENABLE_INTIRQ,
  - Reenable INTIRQ interrupt from being monitored by AD7147 driver
  - Value = NULL
  - Default = none

7.5 Callback Events

This section enumerates the callback events the device driver is capable of generating. The events are divided into two sections. The first section describes events that are common to many device drivers. The next section describes driver specific event IDs. The client should prepare its callback function to process each event described in these two sections.

The callback function is of the type ADI_DCB_CALLBACK_FN. The callback function is passed three parameters. These parameters are:

- ClientHandle – This void * parameter is the value that is passed to the device driver as a parameter in the adi_dev_Open() function.
- EventID – This is a u32 data type that specifies the event ID.
- Value – This parameter is a void * whose value is context sensitive to the specific event ID.

The sections below enumerate the event IDs that the device driver can generate and the meaning of the Value parameter for each event ID.

7.5.1 Common Events

The events described in this section are common to many device drivers. The list below enumerates all common event IDs that are supported by this device driver.

- ADI_DEV_EVENT_BUFFER_PROCESSED
    - Notifies callback function that a chained or sequential I/O buffer has been processed by the device driver. This event is also used to notify that an entire circular buffer has been processed if the driver was directed to generate a callback upon completion of an entire circular buffer.
    - Value – For chained or sequential I/O dataflow methods, this value is the CallbackParameter value that was supplied in the buffer that was passed to the adi_dev_Read(), adi_dev_Write() or adi_dev_SequentialIO() function. For the circular dataflow method, this value is the address of the buffer provided in the adi_dev_Read() or adi_dev_Write() function.

7.5.2 Device Driver Specific Events

The events listed below are supported and processed by the device driver. These event IDs are unique to this device driver.

- ADI_AD7147_EVENT_INT_IRQ
    - Callback Event indicates that the CapTouch hardware interrupt request has been detected and the application should see what caused it.
    - Value – NULL

7.6 Return Codes

All API functions of the device driver return status indicating either successful completion of the function or an indication that an error has occurred. This section enumerates the return codes that the device driver is capable of returning to the client. A return value of ADI_DEV_RESULT_SUCCESS indicates success, while any other value indicates an error or some other informative result. The value ADI_DEV_RESULT_SUCCESS is always equal to the value zero. All other return codes are a non-zero value.

The return codes are divided into two sections. The first section describes return codes that are common to many device drivers. The next section describes driver specific return codes. The client should prepare to process each of the return codes described in these sections.

Typically, the application should check the return code for ADI_DEV_RESULT_SUCCESS, taking appropriate corrective action if ADI_DEV_RESULT_SUCCESS is not returned. For example:

```
if (adi_dev_DriverName(…) == ADI_DEV_RESULT_SUCCESS) {
   // normal processing
} else {
   // error processing
}
```

7.6.1 Common Return Codes

The return codes described in this section are common to many device drivers. The list below enumerates all common return codes that are supported by this device driver.

- ADI_DEV_RESULT_SUCCESS
  - The function executed successfully.
- ADI_DEV_RESULT_NOT_SUPPORTED
  - The function is not supported by the driver.
- ADI_DEV_RESULT_DEVICE_IN_USE
  - The requested device is already in use.
- ADI_DEV_RESULT_NO_MEMORY
  - There is insufficient memory available.
- ADI_DEV_RESULT_BAD_DEVICE_NUMBER
  - The device number is invalid.
- ADI_DEV_RESULT_DIRECTION_NOT_SUPPORTED
  - The device cannot be opened in the direction specified.
- ADI_DEV_RESULT_BAD_DEVICE_HANDLE
  - The handle to the device driver is invalid.
- ADI_DEV_RESULT_BAD_MANAGER_HANDLE
  - The handle to the Device Manager is invalid.
- ADI_DEV_RESULT_BAD_PDD_HANDLE
  - The handle to the physical driver is invalid.
- ADI_DEV_RESULT_INVALID_SEQUENCE
  - The action requested is not within a valid sequence.

- ADI_DEV_RESULT_ATTEMPTED_READ_ON_OUTBOUND_DEVICE
  - The client attempted to provide an inbound buffer for a device opened for outbound traffic only.
- ADI_DEV_RESULT_ATTEMPTED_WRITE_ON_INBOUND_DEVICE
  - The client attempted to provide an outbound buffer for a device opened for inbound traffic only.
- ADI_DEV_RESULT_DATAFLOW_UNDEFINED
  - The dataflow method has not yet been declared.
- ADI_DEV_RESULT_DATAFLOW_INCOMPATIBLE
  - The dataflow method is incompatible with the action requested.
- ADI_DEV_RESULT_BUFFER_TYPE_INCOMPATIBLE
  - The device does not support the buffer type provided.
- ADI_DEV_RESULT_CANT_HOOK_INTERRUPT
  - The Interrupt Manager failed to hook an interrupt handler.
- ADI_DEV_RESULT_CANT_UNHOOK_INTERRUPT
  - The Interrupt Manager failed to unhook an interrupt handler.
- ADI_DEV_RESULT_NON_TERMINATED_LIST
  - The chain of buffers provided is not NULL terminated.
- ADI_DEV_RESULT_NO_CALLBACK_FUNCTION_SUPPLIED
  - No callback function was supplied when it was required.
- ADI_DEV_RESULT_REQUIRES_UNIDIRECTIONAL_DEVICE
  - Requires the device be opened for either inbound or outbound traffic only.
- ADI_DEV_RESULT_REQUIRES_BIDIRECTIONAL_DEVICE
  - Requires the device be opened for bidirectional traffic only.

Return codes specific to TWI/SPI Device access service.

- ADI_DEV_RESULT_TWI_LOCKED
  - Indicates the present TWI device is locked in other operation
- ADI_DEV_RESULT_REQUIRES_TWI_CONFIG_TABLE
  - Client need to supply a configuration table for the TWI driver
- ADI_DEV_RESULT_CMD_NOT_SUPPORTED
  - Command not supported by the Device Access Service
- ADI_DEV_RESULT_INVALID_REG_ADDRESS
  - The client attempting to access an invalid register address
- ADI_DEV_RESULT_INVALID_REG_FIELD
  - The client attempting to access an invalid register field location
- ADI_DEV_RESULT_INVALID_REG_FIELD_DATA
  - The client attempting to write an invalid data to selected register field location
- ADI_DEV_RESULT_ATTEMPT_TO_WRITE_READONLY_REG
  - The client attempting to write to a read-only location
- ADI_DEV_RESULT_ATTEMPT_TO_ACCESS_RESERVE_AREA
  - The client attempting to access a reserved location
- ADI_DEV_RESULT_ACCESS_TYPE_NOT_SUPPORTED
  - Device Access Service does not support the access type provided by the driver

7.6.2 Device Driver Specific Return Codes

The return codes listed below are supported and processed by the CapTouch device driver. These event IDs are unique to the CapTouch device driver.

- ADI_AD7147_RESULT_CMD_NOT_SUPPORTED
  - Occurs when client issues a command that is not supported by this driver
- ADI_AD7147_RESULT_TWI_ADDRESS_INVALID
  - Given TWI Device address is invalid
- ADI_AD7147_RESULT_TWI_NUMBER_INVALID
  - Given TWI Device number is invalid
- ADI_AD7147_RESULT_TWI_CONFIG_TABLE_INVALID
  - Given pointer to TWI configuration table is invalid
- ADI_AD7147_RESULT_REGISTER_ADDR_INVALID
  - Results when client tries to write to an invalid register address
- ADI_AD7147_RESULT_REGISTER_ADDR_RO
  - Results when client tries to write to a read-only register address

8 Opening and Configuring the Device Driver

This section describes the default configuration settings for the device driver and any additional configuration settings required from the client application.

8.1 Entry Point

When opening the device driver at the application level, the “adi_dev_Open()” function call is passed a parameter that identifies the specific device driver to be opened. This parameter is called the device entry point. The entry point for the CapTouch device driver is declared in include file, “adi_ad7147.h” (for access by the application) as:

```
extern ADI_DEV_PDD_ENTRY_POINT      ADIAD7147EntryPoint;
```

The entry point definition for the CapTouch device driver is in source file, “adi_ad7147.c”.

The ADI_DEV_PDD_ENTRY_POINT macro defines a standardized function table pointer shared by all device drivers in which the predefined device driver function addresses are held.

8.2 Default Settings

The table below describes the default configuration settings for the device driver. If the default values are inappropriate for the given system, the application should use the command IDs listed in the table to configure the device driver appropriately. Any configuration settings not listed in the table below are undefined.

The SketchPad example application illustrates setting all these configurations. Initially, it opens the CapTouch driver with "adi_dev_Open()" and then configures it in a single "adi_dev_control()" call using the generic device driver "ADI_DEV_CMD_TABLE" command. This allows a series of commands to be executed in one call, as a table of "ADI_DEV_CMD_VALUE_PAIR" command pairs (subcommand and value).

See the "README.txt" file under either of the following directories for complete details of configuring the CapTouch controller (as well as for configuring the TouchScreen controller and the LCD display module) on the BLACKFIN LANDSCAPE LCD EZ-EXTENDER board.

- "C:\Program Files\Analog Devices\VisualDSP 5.0\Blackfin\Examples\Landscape LCD EZ-EXTENDER\SketchPad\ADSP-BF518\README.txt"
- "C:\Program Files\Analog Devices\VisualDSP 5.0\Blackfin\Examples\Landscape LCD EZ-EXTENDER\SketchPad\ADSP-BF526\README.txt"

Item	Default Value	Possible Values	Command ID
Interrupt Flag & IVG Group	Undefined	See section 6.4	ADI_AD7147_CMD_INSTALL_INTIRQ
TWI Device Number	Invalid	See Section 6.4	ADI_AD7147_CMD_SET_TWI_DEVICE_NUMBER
TWI Device Address	Invalid	See Section 6.4	ADI_AD7147_CMD_SET_TWI_DEVICE_ADDRESS

Table 4 – Default Settings

8.3 Additional Required Configuration Settings

In addition to the overrides of the default driver settings, the device driver requires the application to specify the additional configuration information listed in the table below. The settings shown here are passed to the underlying low-level TWI device driver by the CapTouch device driver.

The SketchPad example application illustrates setting all these configurations after initially opening the CapTouch device driver in a single call to "adi_dev_control()" call with the generic device driver "ADI_DEV_CMD_TABLE" command to execute a table of "ADI_DEV_CMD_VALUE_PAIR" command pairs (subcommand and value).

See the "README.txt" file under either of the following directories for complete details of configuring the CapTouch controller (as well as for configuring the TouchScreen controller and the LCD display module) on the BLACKFIN LANDSCAPE LCD EZ-EXTENDER board.

- "C:\Program Files\Analog Devices\VisualDSP 5.0\Blackfin\Examples\Landscape LCD EZ-EXTENDER\SketchPad\ADSP-BF518\README.txt"
- "C:\Program Files\Analog Devices\VisualDSP 5.0\Blackfin\Examples\Landscape LCD EZ-EXTENDER\SketchPad\ADSP-BF526\README.txt"

Item	SketchPad Value	Command ID
Serial Channel	ADI_INT_TWI	ADI_TWI_CMD_SET_HARDWARE
TWI Timing	Frequency & Duty Cycle	ADI_TWI_CMD_SET_RATE
FIFO Interrupt	0	ADI_TWI_CMD_SET_FIFO
Retries before reporting lost arbitration	5	ADI_TWI_CMD_SET_LOSTARB
Callback for Address ACK	0	ADI_TWI_CMD_SET_ANAK
Callback for Data ACK	0	ADI_TWI_CMD_SET_DNAK
Dataflow Method	ADI_DEV_MODE_SEQ_CHAINED	ADI_DEV_CMD_SET_DATAFLOW_METHOD
Dataflow Enable	TRUE	ADI_DEV_CMD_SET_DATAFLOW

Table 5 – Additional Required Settings

9 References and Resources

9.1 Hardware Reference Manuals

- BLACKFIN LANDSCAPE LCD EZ EXTENDER Manual & Schematic
- EVAL-AD7147EBZ CapTouch Evaluation Board and Demo Software
- HPUSB, USB, HPPCI, and MSP430 Emulators User's Guide
- ADSP-BF51x Hardware Reference Manual
- ADSP-BF518F EZ-Board Evaluation System Manual & Schematic
- ADSP-BF52x Hardware Reference Manual (Vol. 1 & 2)
- ADSP-BF526 EZ-Board Evaluation System Manual & Schematic

9.2 Data Sheets

- AD7147 CapTouch Programmable Controller Data Sheet
- AD7879 Low Voltage Controller for Touch Screens Data Sheet
- ADSP-BF518 Blackfin Embedded Processor Data Sheet
- ADSP-BF526 Blackfin Embedded Processor Data Sheet
- EVAL-AD7147 CapTouch Evaluation Board Data Sheet

9.3 Software Manuals

- VisualDSP++ 5.0 ADI_AD7147 Device Driver (CapTouch Controller)
- VisualDSP++ 5.0 ADI_AD7879 Device Driver (TouchScreen Controller)
- VisualDSP++ 5.0 ADI_LQ035Q1DH02 Device Driver (LCD Module)
- VisualDSP++ 5.0 Device Drivers and System Services User Manual for Blackfin Processors
- VisualDSP++ 5.0 VDK User's Guide

9.4 Application Notes and White Papers

- *Sensors for the AD7147 and AD7148 CapTouch Controllers*, AN-925, Analog Devices, Inc.
- *Tuning the AD714x for CapTouch Applications*, AN-929, Analog Devices, Inc.
- *Blackfin Processor Troubleshooting Tips using VisualDSP++ Tools*, EE-307, Analog Devices, Inc.
- *System Optimization Techniques for Blackfin Processors*, EE-324, Analog Devices, Inc.
- *Writing Drivers for Common Touch-Screen Interface Hardware*, by Kenneth G. Maxwell, Embedded Systems Design, 6/15/05
- *How to Calibrate Touch Screens*, by Carlos E. Vidales, Embedded Systems Design, 5/31/02