ANALOG DEVICES

ADI_KPAD
DEVICE DRIVER

DATE: FEBRUARY 20, 2007

Table of Contents

List of Tables

Document Revision History

Date	Description of Changes
2007-02-20	Initial version

Table 1 – Revision History

1. Overview

This document describes the usage of the KPAD(key pad) device driver. As written, the driver supports Press-Release mode and has no hardware dependencies and has been tested on the ADSP-BF548 EZ-Kit Lite development boards.

The KPAD device driver is an interrupt driven device driver.

2. Files

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

2.1. Include Files

The driver sources include the following include files:

- <services/services.h> This file contains all definitions, function prototypes etc. for all the System Services.
- <drivers/adi_dev.h> This file contains all definitions, function prototypes etc. for the Device Manager and general device driver information.
- <drivers/kpad/adi_kpad.h> This file contains all commands, event and return codes specific to the KPAD device driver.

2.2. Source Files

The driver sources are contained in the following files, as located in the default installation directory:

- <Blackfin/lib/src/drivers/kpad/adi_kpad.c> This file contains all source code for the KPAD device driver. All source code is written in ‘C’.

3. Lower Level Drivers

The KPAD device driver does not use any lower level device drivers.

4. Resources Required

Device drivers typically consume some amount of system resources. This section describes the resources required by the device driver.

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.

Because dynamic memory allocations are not used in the Device Drivers or System Services, all memory used by the Device Drivers and System Services must be supplied by the application. The Device Drivers and System Services supply macros that can be used by the application to size the amount of base memory and/or the amount of incremental memory required to support the needed functionality. Memory for the Device Manager and System Services is provided in the initialization functions (adi_xxx_Init()).

4.1. Interrupts

This driver uses one interrupt, the KPAD hardware interrupt.
The KPAD device driver uses the default Interrupt Vector Group (IVG) mappings. Changes to the IVG mappings can be made by appropriate calls into the Interrupt Manager service prior to opening the KPAD driver.

4.2. DMA

This driver does not use any DMA resources.

4.3. Timers

This driver does not use any timers.

4.4. Real-Time Clock

This driver does not use the real-time clock.

4.5. Programmable Flags

This driver does not use any programmable flag pins

4.6. Pins

The KPAD makes use of maximum 8 pins for active row and 8 pins for active column.
The number of pins used depends on the number of active rows and columns. For example, a 4x4 matrix uses 8 pins.
On processors where pin muxing is used, the KPAD device driver uses the Port Control service to automatically configure pins for use by the

The pins are configured when the KPAD is enabled via adi_dev_Control().

5. Supported Features of the Device Driver

This section describes what features are supported by the device driver.

5.1. Directionality

There is no dataflow between Key-Pad hardware and processor. The driver does not support the dataflow directions .

5.2. Dataflow Methods

This driver does not support any dataflow methods.

5.3. Buffer Types

This driver does not support any buffer types.

5.4. Command IDs

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

5.4.1. 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

5.4.2. 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_GET_PERIPHERAL_DMA_SUPPORT
  - Determines if the device driver is supported by peripheral DMA
  - Value – u32 * (location where TRUE or FALSE is stored)

5.4.3. Device Driver Specific Commands

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

5.4.3.1. Commands to set device registers

ADI_KPAD_CMD_SET_CONTROL_REG

- Sets the KPAD_CTL register.
- Value – u16 (register value).

ADI_KPAD_CMD_SET_MSEL_REG

- Sets the KPAD_MSEL register
- Value – u16 (register value).

5.4.3.2. Commands to read device registers

ADI_KPAD_CMD_GET_CONTROL_REG

- Gets the KPAD_CTL register
- Value – u16 * (location where contents of KPAD_CTL will be stored)

ADI_KPAD_CMD_GET_PRESCALE_REG

- Gets the KPAD_PRESCALE register
- Value – u16 * (location where contents of KPAD_PRESCALE will be stored)

ADI_KPAD_CMD_GET_MSEL_REG

- Gets the KPAD_MSEL register
- Value – u16 * (location where contents of KPAD_MSEL will be stored)

ADI_KPAD_CMD_GET_ROWCOL_REG

- Gets the KPAD_ROWCOL register
- Value – u16 * (location where contents of KPAD_ROWCOL will be stored)

ADI_KPAD_CMD_GET_STAT_REG

- Gets the KPAD_STAT register
- Value – u16 * (location where contents of KPAD_STAT will be stored)

ADI_KPAD_CMD_GET_SOFTEVAL_REG

- Gets the KPAD_SOFTEVAL register
- Value – u16 * (location where contents of KPAD_SOFTEVAL will be stored)

5.4.3.3. Commands to configure device register fields

ADI_KPAD_CMD_SET_KPAD_ENABLE

- Enables/disables the Keypad Interface Module
- Value – (TRUE – enabled, FALSE – disabled)

ADI_KPAD_CMD_SET_IRQMODE

- Sets the Key press Interrupt mode
- Value: 00 – Interrupt disabled
  01 – Single key press interrupt enable
  10 – Single key and multiple key press interrupt enable

ADI_KPAD_CMD_SET_ROW_NUMBER

- Sets the active number of rows.
- Value: 000 – 1 row
  001 – 2 rows
  010 – 3 rows
  011 – 4 rows
  100 – 5 rows
  101 – 6 rows
  110 – 7 rows
  111 – 8 rows

ADI_KPAD_CMD_SET_COLUMN_NUMBER

- Sets the active number of columns.
- Value: 000 – 1 column
  001 – 2 columns
  010 – 3 columns
  011 – 4 columns
  100 – 5 columns
  101 – 6 columns
  110 – 7 columns
  111 – 8 columns

ADI_KPAD_CMD_SET_PRESCALE_VAL

- Sets the key pre-scale value
- Value: 1 – 63. Formula to get to 0.1 ms timescale : (SCLK(MHz) * 100) / 1024

ADI_KPAD_CMD_SET_DBON_SCALE

- Sets the debounce delay multiplier
- Value: (8 bits value)

ADI_KPAD_CMD_SET_COLDRV_SCALE

- Sets the column driver period multiplier
- Value: (8 bits value)

ADI_KPAD_CMD_SET_SOFTEVAL

- Sets key programmable force evaluate
- Value: 0(normal mode) or 1 (force evaluate mode)

5.5. 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.

5.5.1. 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.

5.5.2. 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_KPAD_EVENT_KEYPRESSED
  - Notifies the callback function that a key or multiple key has been pressed.
  - Value – Pointer to ADI_KPAD_KEY_PRESSED_RESULT structure will be passed as callback argument

5.6. 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_Xxxx(…) == ADI_DEV_RESULT_SUCCESS) {
      // normal processing
} else {
      // error processing
}
```

5.6.1. 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

5.6.2. Device Driver Specific Return Codes

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

- ADI_KPAD_RESULT_BAD_ACCESS
    - This error is a result of the client attempting to configure the control register when key-pad is enabled.

- ADI_KPAD_RESULT_BAD_ACCESS_WIDTH
    - This error is a result of the client attempts to write an invalid data to selected register field location. Example: writes value 0x11 to 1 bit field register.

6. 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.

6.1. Entry Point

When opening the device driver with the adi_dev_Open() function call, the client passes a parameter to the function that identifies the specific device driver that is being opened. This parameter is called the entry point. The entry point for this driver is listed below.

- ADIKPADEntryPoint

6.2. 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.

Item	Default Value	Possible Values	Command ID
Number of columns = 4	0x011	0x000 to 0x111	ADI_KPAD_CMD_SET_COLUMN_NUMBER
Number of rows = 4	0x011	0x000 to 0x111	ADI_KPAD_CMD_SET_ROW_NUMBER
Single/multiple key press interrupt enable	0x10	0x00 to 0x10	ADI_KPAD_CMD_SET_IRQMODE
Key prescale time base of 0.1 msec	4	0 to 63	ADI_KPAD_CMD_SET_PRESCALE_VAL
Debounce delay multiplier for 1 msec	9	0 to 255	ADI_KPAD_CMD_SET_DBON_SCALE
Column drive period multiplier for 1 msec	9	0 to 255	ADI_KPAD_CMD_SET_COLDRV_SCALE

Table 2 – Default Settings

6.3. Additional Required Configuration Settings

In addition to the possible overrides of the default driver settings, the device driver requires the application to specify the additional configuration information listed in the table below.

Item	Possible Values	Command ID
Enables/disables KPAD	1(enable) , 0(disable)	ADI_KPAD_CMD_SET_KPAD_ENABLE

Table 3 – Additional Required Settings

7. Hardware Considerations

There are no special hardware considerations for the KPAD device driver.