ADI ACM SERVICE

ANALOG DEVICES, INC.

www.analog.com

(REV 0.1, DECEMBER 22, 2009)

Table of Contents

1 Introduction 6
1.1 Scope 6
1.2 Organisation of this Guide 6
1.3 Acronyms 6
1.4 References 6
1.5 Additional Information 6

2 Programmer's Reference 7
2.1 Files 7
2.1.1 Libraries 7
2.1.2 Include Files 7
2.1.3 Source Files 7
2.2 API Functions 8
2.2.1 adi_acm_Open 8
2.2.2 adi_acm_Close 9
2.2.3 adi_acm_Control 10
2.3 Control Commands 12
2.3.1 Batch processing commands 12
2.3.1.1 ADI_ACM_CMD_TABLE 12
2.3.1.2 ADI_ACM_CMD_END 12
2.3.1.3 ADI_ACM_CMD_PAIR 12
2.3.2 ACM Register Configuration commands 13
2.3.2.1 ADI_ACM_CMD_SET_CTRL_REG 13
2.3.2.2 ADI_ACM_CMD_SET_EVENT_CTRL_REG 13
2.3.2.3 ADI_ACM_CMD_SET_EVENT_TIME_REG 13
2.3.2.4 ADI_ACM_CMD_SET_EVENT_CTRL_REG_TABLE 13
2.3.2.5 ADI_ACM_CMD_SET_EVENT_TIME_REG_TABLE 13
2.3.3 ACM Register Query commands 14
2.3.3.1 ADI_ACM_CMD_GET_STAT_REG 14
2.3.4 ACM Register Field Configuration commands 14
2.3.4.1 ADI_ACM_CMD_ENABLE_ACM 14
2.3.4.2 ADI_ACM_CMD_ENABLE_EVENT 14
2.3.4.3 ADI_ACM_CMD_DISABLE_EVENT 14
2.3.4.4 ADI_ACM_CMD_CONFIG_TMR 14
2.3.4.5 ADI_ACM_CMD_SET_CS_POL 14
2.3.4.6 ADI_ACM_CMD_SET_ADC_CLK_POL 15

2.3.4.7 ADI_ACM_CMD_SET_EXT_PERIPHERAL_SEL 15
2.3.4.8 ADI_ACM_CMD_ENABLE_EVENT_COMPLETE_INT 15
2.3.4.9 ADI_ACM_CMD_DISABLE_EVENT_COMPLETE_INT 15
2.3.4.10 ADI_ACM_CMD_ENABLE_EVENT_MISS_INT 15
2.3.4.11 ADI_ACM_CMD_DISABLE_EVENT_MISS_INT 15
2.3.4.12 ADI_ACM_CMD_SET_EVENT_PARAMS 15
2.3.4.13 ADI_ACM_CMD_SET_EVENT_PARAMS_TABLE 16
2.3.4.14 ADI_ACM_CMD_SET_ACLK_FREQ 16
2.3.4.15 ADI_ACM_CMD_SET_ACLK_DIVISOR 16
2.3.4.16 ADI_ACM_CMD_SET_SETUP_CYCLES 16
2.3.4.17 ADI_ACM_CMD_SET_CS_WIDTH 16
2.3.4.18 ADI_ACM_CMD_SET_HOLD_CYCLES 16
2.3.4.19 ADI_ACM_CMD_SET_ZERO_CYCLES 16
2.3.5 ACM Register Field Query commands 17
2.3.5.1 ADI_ACM_CMD_GET_EVENT_STAT 17
2.3.5.2 ADI_ACM_CMD_GET_EVENT_COMPLETE_STAT 17
2.3.5.3 ADI_ACM_CMD_GET_EVENT_MISS_STAT 17
2.3.5.4 ADI_ACM_CMD_GET_EVENT_COMPLETE_LIST 17
2.3.5.5 ADI_ACM_CMD_GET_EVENT_MISS_LIST 17
2.3.6 Callback Control commands 18
2.3.6.1 ADI_ACM_CMD_SET_CALLBACK_FN 18
2.3.6.2 ADI_ACM_CMD_SET_CALLBACK_PARAM 18
2.3.6.3 ADI_ACM_CMD_SET_DCB_MANAGER_HANDLE 18
2.4 Callback Events 19
2.5 API Data Types 20
2.5.1 Structures 20
2.5.1.1 ADI_ACM_CMD_VALUE_PAIR 20
2.5.1.2 ADI_ACM_TMR_CONFIG 21
2.5.1.3 ADI_ACM_EVENT_CONFIG 22
2.5.1.4 ADI_ACM_EVENT_CONFIG_TABLE 23
2.5.1.5 ADI_ACM_EVENT_PARAMS 24
2.5.1.6 ADI_ACM_EVENT_PARAMS_TABLE 25
2.5.1.7 ADI_ACM_EVENT_STAT 26
2.5.1.8 ADI_ACM_EVENT_LIST 27

List of Figures

Figure 1: References 6

Copyright, Disclaimer & 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++, EZ-KIT Lite, 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.

Implementation Notice

1 Introduction

This document describes the usage of the ADC control module (ACM) service. The ADC control module (ACM) is available on the Blackfin ADSP-BF50x family of processors and provides an interface that synchronizes the controls between the processor and an analog-to-digital converter (ADC). The analog-to-digital conversions are initiated by the processor, based on either external or internal events. This document lists the APIs, commands, enumerations and data structures of the system services designed to control ACM peripheral.

1.1 Scope

The document is intended to assist software developers using ADC control module (ACM) service to control an on-chip or off-chip ADC device. Developers are assumed to be familiar with ADI Device Drivers and System Services model.

1.2 Organisation of this Guide

Section 1: this section contains the introduction.

Section 2: contains the programmer's reference.

1.3 Acronyms

ADC	Analog to Digital Converter
ADI	Analog Devices Inc.
ACM	ADC control module
API	Application Program Interface

1.4 References

[1] "ADSP-BF50x Blackfin Embedded Processor Data Sheet", Revision PrC, 2009
[2] ADSP-BF50x Blackfin Processor Hardware Reference, Revision 0.1, December 2009

Figure 1: References

1.5 Additional Information

None

2 Programmer's Reference

The API of ACM service is organised as follows -:

- Section 2.1: Files.
- Section 2.2: API Functions.
- Section 2.3: Control Commands
- Section 2.4: Callback Events
- Section 2.5: API Data Types

2.1 Files

ACM service consists of following files.

2.1.1 Libraries

ACM service is released as part of the ADI System Services library.

2.1.2 Include Files

ACM Service have the following header files.

- ***<services/services.h>***
  This file contains the API and definitions common to all system services.
- ***<services/acm/adi_acm.h>***
  This file contains the API and definitions specific to ACM service.

2.1.3 Source Files

It is not required to include the source files into the application project as the sources are part of the ADI System Services library. ACM Service source file is located at:

```
<VDSP Installation directory>/Blackfin/lib/src/services/acm>
```

2.2 API Functions

This section contains detailed descriptions of the API functions of ACM Service

2.2.1 adi_acm_Open

Prototype

```
ADI_ACM_RESULT  adi_acm_Open (
    u32                     nDeviceNumber,
    void                    *pCriticalRegionArg,
    ADI_ACM_DEV_HANDLE      *phAcmDevice
);
```

Description

This function opens an ADC Control Module (ACM) device

Preconditions

Power, EBIU, Interrupt, Timer, Flag and Port control service must be initialised before opening an ACM device

Parameters

Name: nDeviceNumber
Type: u32
Direction: Input
Description: Physical ACM Device number to open

Name: pCriticalRegionArg
Type: void
Direction: Input
Description: Critical region parameter

Name: phAcmDevice
Type: ADI_ACM_DEV_HANDLE
Direction: Output
Description: Pointer to location to store handle to this ACM device

Return value

Event (Hex Code)	Event	Description
0	ADI_ACM_RESULT_SUCCESS	Successfully opened ACM device
0x00110001	ADI_ACM_RESULT_ALREADY_IN_USE	Specified ACM device has been initialised and already in use
0x00110003	ADI_ACM_RESULT_BAD_DEVICE_NUMBER	ACM device number is invalid

2.2.2 adi_acm_Close

Prototype

```
ADI_ACM_RESULT  adi_acm_Close (
    ADI_ACM_DEV_HANDLE      hAcmDevice
);
```

Description

This function closes an ADC Control Module (ACM) device

Preconditions

The device must be open.

Parameters

Name: hAcmDevice
Type: ADI_ACM_DEV_HANDLE
Direction: Input
Description: Handle to an active ACM device to close

Return value

Event (Hex Code)	Event	Description
0	ADI_ACM_RESULT_SUCCESS	Successfully closed ACM device
0x00110002	ADI_ACM_RESULT_BAD_HANDLE	Supplied ACM device handle is invalid

2.2.3 adi_acm_Control

Prototype

```
ADI_ACM_RESULT  adi_acm_Control (
    ADI_ACM_DEV_HANDLE      hAcmDevice,
    ADI_ACM_COMMAND         eCommandID,
    void                    *pValue
);
```

Description

This function sets/senses ADC Control Module (ACM) device specific settings

Preconditions

ACM device must be opened to issue control commands.

Parameters

Name: hAcmDevice
Type: ADI_ACM_DEV_HANDLE
Direction: Input
Description: Handle to an active ACM device to set/sense

Name: eCommandID
Type: ADI_ACM_COMMAND
Direction: Input
Description: ACM specific command ID to process

Name: pValue
Type: void
Direction: Input/Output
Description: Command specific value

Return value

Event (Hex Code)	Event	Description
0	ADI_ACM_RESULT_SUCCESS	Successfully set/sensed ACM device settings
0x00110001	ADI_ACM_RESULT_BAD_HANDLE	Supplied ACM device handle is invalid
0x00110004	ADI_ACM_RESULT_COMMAND_INVALID	Given command is invalid/not recognised by ACM
0x00110005	ADI_ACM_RESULT_CALLBACK_FN_INALID	ACM device does not have a valid callback function to report interrupts to application
0x00110006	ADI_ACM_RESULT_COMMAND_NOT_PERMITTED	Issued command is not permitted at this stage
0x00110007	ADI_ACM_RESULT_EVENT_NUMBER_INVALID	Issued Event number is invalid
0x00110008	ADI_ACM_RESULT_TIMER_NUMBER_INVALID	Issued ACM Timer Number is invalid
0x00110009	ADI_ACM_RESULT_TIMER_CONFIG_INVALID	Issued ACM Timer configuration is invalid
0x0011000A	ADI_ACM_RESULT_EVENT_PARAM_INVALID	Issued Event parameter configuration is invalid
0x0011000B	ADI_ACM_RESULT_EXT_PERIPHERAL_INVALID	Issued ACM External Peripheral ID is invalid
0x0011000C	ADI_ACM_RESULT_ACLK_OUT_OF_RANGE	Issued ACLK frequency is out of range and can't be supported
0x0011000D	ADI_ACM_RESULT_CANNOT_DERIVE_ACLK	Issued ACLK frequency can't be derived from current SCLK
0x0011000E	ADI_ACM_RESULT_CLKDIV_INVALID	Issued ACLK Divisor is invalid
0x0011000F	ADI_ACM_RESULT_SETUP_CYCLES_INVALID	Issued ACM Setup cycle value is invalid
0x00110010	ADI_ACM_RESULT_CS_WIDTH_INVALID	Issued CS Width value is invalid
0x00110011	ADI_ACM_RESULT_HOLD_CYCLES_INVALID	Issued Hold Cycles value is invalid
0x00110012	ADI_ACM_RESULT_ZERO_CYCLES_INVALID	Issued Zero Cycles value is invalid
0x00110013	ADI_ACM_RESULT_INT_HOOK_FAILED	Failed to Hook ACM Status interrupt to interrupt manager
0x00110014	ADI_ACM_RESULT_INT_UNHOOK_FAILED	Failed to Unhook ACM Status interrupt from interrupt manager
0x00110015	ADI_ACM_RESULT_PORT_CONTROL_FAILED	Failed to configure ports for ACM
0x00110016	ADI_ADC_RESULT_GET_SCLK_FAILED	Failed to get present system clock frequency

2.3 Control Commands

This section lists the ACM service specific command ids to be used with `'adi_acm_Control'` function to set/sense ACM device specific parameters. Refer section 2.2.3 for API Prototype

2.3.1 Batch processing commands

This section contains a group of commands that can be used to issue a table of control commands to the ACM service. An ideal use case for these commands would be during ACM device initialisation, where a set of commands configuring device specific parameters can be grouped together and passed in single `'adi_acm_Control'` call

2.3.1.1 ADI_ACM_CMD_TABLE

Command to issue a table of control commands to an ACM device. The issued command table MUST be terminated with `ADI_ACM_CMD_END`.

Command Specific Value ADI_ACM_CMD_VALUE_PAIR* (refer section 2.5.1.1)

(Address of a table of structure of type ADI_ACM_CMD_VALUE_PAIR. Table must be terminated with ADI_ACM_CMD_END)

2.3.1.2 ADI_ACM_CMD_END

Command to indicate end of a control command table issued to an ACM device.

Command Specific Value NULL

2.3.1.3 ADI_ACM_CMD_PAIR

Command to issue a single command-value pair.

Command Specific Value ADI_ACM_CMD_VALUE_PAIR* (refer section 2.5.1.1)

(Address of a structure of type ADI_ACM_CMD_VALUE_PAIR)

2.3.2 ACM Register Configuration commands

2.3.2.1 ADI_ACM_CMD_SET_CTRL_REG

Command to set ACM Control register value.

Command Specific Value u32 (Control register value)

2.3.2.2 ADI_ACM_CMD_SET_EVENT_CTRL_REG

Command to set Event control register corresponding to an event number.

Command Specific Value ADI_ACM_EVENT_CONFIG *

(Address of a structure of type ADI_ACM_EVENT_CONFIG – refer section 2.5.1.3)

2.3.2.3 ADI_ACM_CMD_SET_EVENT_TIME_REG

Command to configure time value register corresponding to an event number.

Command Specific Value ADI_ACM_EVENT_CONFIG *

(Address of a structure of type ADI_ACM_EVENT_CONFIG – refer section 2.5.1.3)

2.3.2.4 ADI_ACM_CMD_SET_EVENT_CTRL_REG_TABLE

Command to configure a table of event numbers and its corresponding control registers.

Command Specific Value ADI_ACM_EVENT_CONFIG_TABLE *

(Address of a structure of type ADI_ACM_EVENT_CONFIG_TABLE – refer section 2.5.1.4)

2.3.2.5 ADI_ACM_CMD_SET_EVENT_TIME_REG_TABLE

Command to configure a table of event numbers and its corresponding time value registers.

Command Specific Value ADI_ACM_EVENT_CONFIG_TABLE *

(Address of a structure of type ADI_ACM_EVENT_CONFIG_TABLE – refer section 2.5.1.4)

2.3.3 ACM Register Query commands

2.3.3.1 ADI_ACM_CMD_GET_STAT_REG

Command to get current value of ACM Status register.

Command Specific Value u32 * (Location to store status register value)

2.3.4 ACM Register Field Configuration commands

2.3.4.1 ADI_ACM_CMD_ENABLE_ACM

Command to enable/Disable ACM.

Command Specific Value true/false (true to enable, false to disable)

2.3.4.2 ADI_ACM_CMD_ENABLE_EVENT

Command to enable a selected ACM event.

Command Specific Value u8 (Event number to enable)

2.3.4.3 ADI_ACM_CMD_DISABLE_EVENT

Command to disable a selected ACM event.

Command Specific Value u8 (Event number to disable)

2.3.4.4 ADI_ACM_CMD_CONFIG_TMR

Command to configure ACM Timer specific fields.

Command Specific Value ADI_ACM_TMR_CONFIG *
(Address of a structure of type ADI_ACM_TMR_CONFIG – refer section 2.5.1.2)

2.3.4.5 ADI_ACM_CMD_SET_CS_POL

Command to set chip select / start conversion (CS) polarity.

Command Specific Value 0 or 1 (0 for Active low, 1 for Active high)

2.3.4.6 ADI_ACM_CMD_SET_ADC_CLK_POL

Command to set ADC Clock (ACLK) polarity after CS gets activated.

Command Specific Value 0 or 1 (0 for falling edge, 1 for raising edge)

2.3.4.7 ADI_ACM_CMD_SET_EXT_PERIPHERAL_SEL

Command to set External Peripheral Select.

Command Specific Value 0 or 1 (SPORT device number)

2.3.4.8 ADI_ACM_CMD_ENABLE_EVENT_COMPLETE_INT

Command to enable event completion interrupt for a specific event number.

Command Specific Value u8 (Event number)

2.3.4.9 ADI_ACM_CMD_DISABLE_EVENT_COMPLETE_INT

Command to disable event completion interrupt for a specific event number.

Command Specific Value u8 (Event number)

2.3.4.10 ADI_ACM_CMD_ENABLE_EVENT_MISS_INT

Command to enable event missed interrupt for a specific event number.

Command Specific Value u8 (Event number)

2.3.4.11 ADI_ACM_CMD_DISABLE_EVENT_MISS_INT

Command to disable event missed interrupt for a specific event number.

Command Specific Value u8 (Event number)

2.3.4.12 ADI_ACM_CMD_SET_EVENT_PARAMS

Command to pass all parameters applicable to an event.

Command Specific Value ADI_ACM_EVENT_PARAMS *
(Address of a structure of type ADI_ACM_EVENT_PARAMS – refer section 2.5.1.5)

2.3.4.13 ADI_ACM_CMD_SET_EVENT_PARAMS_TABLE

Command to pass a table of event parameters.

Command Specific Value ADI_ACM_EVENT_PARAMS_TABLE *

(Address of a structure of type ADI_ACM_EVENT_PARAMS_TABLE – refer section 2.5.1.6)

2.3.4.14 ADI_ACM_CMD_SET_ACLK_FREQ

Command to set ACM Clock frequency (Clock output for ADC and SPORT)

Command Specific Value u32 (Frequency in Hertz)

2.3.4.15 ADI_ACM_CMD_SET_ACLK_DIVISOR

Command to set ACM Clock Divisor.

Command Specific Value u8 (Clock Divisor - 0 to 255)

2.3.4.16 ADI_ACM_CMD_SET_SETUP_CYCLES

Command to set setup cycle time in terms of SCLK.

Command Specific Value u16 (Number of SCLKs - between 1 and 256 for ADSP-BF50x family of processors)

2.3.4.17 ADI_ACM_CMD_SET_CS_WIDTH

Command to set Chip Select/Start Conversion (CS) Width in terms of ACLK. Active duration of active CS in ACLK cycles

Command Specific Value u16 (Number of ACLKs - between 1 and 256 for ADSP-BF50x family of processors)

2.3.4.18 ADI_ACM_CMD_SET_HOLD_CYCLES

Command to set Hold Cycles in terms of ACLK (Hold in ACLK cycles after the inactive edge of CS for all ADC controls).

Command Specific Value u8 (Number of ACLKs - between 0 and 15 for ADSP-BF50x family of processors)

2.3.4.19 ADI_ACM_CMD_SET_ZERO_CYCLES

Command to set Zero Cycles in terms of ACLK (Zero duration (driven low) in ACLK cycles for all ADC controls).

Command Specific Value u8 (Number of ACLKs - between 0 and 15 for ADSP-BF50x family of processors)

2.3.5 ACM Register Field Query commands

2.3.5.1 ADI_ACM_CMD_GET_EVENT_STAT

Command to get status of a particular event number.

Command Specific Value ADI_ACM_EVENT_STAT *
(Address of a structure of type ADI_ACM_EVENT_STAT – refer section 2.5.1.7)

2.3.5.2 ADI_ACM_CMD_GET_EVENT_COMPLETE_STAT

Command to query if all events related to current trigger is completed.

Command Specific Value bool * (Pointer to a boolen type to store ACM status)
(returns 'true' when all enabled events of current trigger completed,
returns 'false' on pending/incomplete events)

2.3.5.3 ADI_ACM_CMD_GET_EVENT_MISS_STAT

Command to query if any events related to current trigger was missed.

Command Specific Value bool * (Pointer to a boolen type to store ACM status)
(returns 'true' when ACM has missed event(s),
returns 'false' no events were missed)

2.3.5.4 ADI_ACM_CMD_GET_EVENT_COMPLETE_LIST

Command to get a list of event numbers that are completed/done.

Command Specific Value ADI_ACM_EVENT_LIST *
(Address of a structure of type ADI_ACM_EVENT_LIST – refer section 2.5.1.8)

2.3.5.5 ADI_ACM_CMD_GET_EVENT_MISS_LIST

Command to get a list of event numbers missed.

Command Specific Value ADI_ACM_EVENT_LIST *
(Address of a structure of type ADI_ACM_EVENT_LIST – refer section 2.5.1.8)

2.3.6 Callback Control commands

2.3.6.1 ADI_ACM_CMD_SET_CALLBACK_FN

Command to set Application callback function.

Command Specific Value ADI_DCB_CALLBACK_FN

2.3.6.2 ADI_ACM_CMD_SET_CALLBACK_PARAM

Command to set Callback parameter passed to Application callback function.

Command Specific Value void *

2.3.6.3 ADI_ACM_CMD_SET_DCB_MANAGER_HANDLE

Command to set Deferred Callback manager handle.

Command Specific Value of type ADI_DCB_HANDLE or NULL

2.4 Callback Events

When an ACM device is opened, by default, the device is set to operate in polling mode. None of the callback events listed below is supported in polling mode. Application has to use callback control commands (section 2.3.6) and ACM register field configuration commands (section 2.3.4) to enable/disable interrupts for ACM complete/miss events.

Event	Description
`ADI_ACM_EVENT_EVENT_COMPLETE`	Reports completion of a particular event Callback Argument = `u8` (Event number completed)
`ADI_ACM_EVENT_EVENT_MISS`	Reports when an event was missed Callback Argument = `u8` (Event number missed)

2.5 API Data Types

This section describes the data types supported by ACM Service

2.5.1 Structures

2.5.1.1 ADI_ACM_CMD_VALUE_PAIR

```
typedef struct ADI_ACM_CMD_VALUE_PAIR
{
    /* Command ID */
    ADI_ACM_COMMAND     eCommandID;

    /* Command specific value */
    void                *Value;

} ADI_ACM_CMD_VALUE_PAIR;
```

Description

This structure is used to pass a command – value pair to the ACM service. Pointer to an instance of this type is passed as an argument with `ADI_ACM_CMD_PAIR` command (section 2.3.1.3), and a pointer to a table of instance of this type is passed as an argument with `ADI_ACM_CMD_TABLE` command (section 2.3.1.1)

Fields

- `eCommandID`
  ACM Command ID
- `Value`
  Command specific value

2.5.1.2 ADI_ACM_TMR_CONFIG

```
typedef struct __AdiAcmTmrConfig
{

    /* Timer number to configure
       Accepted values = 0, 1 */
    u8      nTmrNumber;

    /* Flag to enable/disable Timer
       'true' to enable, 'false' to disable */
    bool    bEnable;

    /* Trigger select value
       Accepted values = between 0 and 3 (inclusive) */
    u8      nTriggerSelect;

    /* Trigger polarity value
       Accepted values = 0 or 1 (0 for raising edge, 1 for falling edge) */
    u8      nTriggerPolarity;

} ADI_ACM_TMR_CONFIG;
```

Description

This structure is used to pass a ACM Timer specific configuration values. Pointer to an instance of this type is passed as an argument with `ADI_ACM_CMD_CONFIG_TMR` command (section 2.3.4.4)

Fields

- `nTmrNumber`
  Timer number to configure
  Accepted values = 0, 1

- `bEnable`
  Flag to enable/disable Timer.
  'true' to enable, 'false' to disable

- `nTriggerSelect`
  Trigger select value
  Accepted values = between 0 and 3 (inclusive)

- `nTriggerPolarity`
  Trigger polarity value
  Accepted values = 0 or 1 (0 for raising edge, 1 for falling edge)

2.5.1.3 ADI_ACM_EVENT_CONFIG

```
typedef struct __AdiAcmEventConfig
{

    /* Holds the Event number to configure */
    u8      nEventNumber;

    /* Configuration value of the register/register field
       corresponding to the event number */
    u32     nConfigValue;

} ADI_ACM_EVENT_CONFIG;
```

Description

This structure is used to configure a register/register field corresponding to an event. Pointer to an instance of this type is passed as an argument with commands `ADI_ACM_CMD_SET_EVENT_CTRL_REG` and `ADI_ACM_CMD_SET_EVENT_TIME_REG`

Fields

- `eCommandID`
  ACM Command ID
- `Value`
  Command specific value

2.5.1.4 ADI_ACM_EVENT_CONFIG_TABLE

```
typedef struct __AdiAcmEventConfigTable
{

    /* Number of Event register/register field configuration enteries */
    u32                     nNumEntries;

    /* Pointer to an array of event register/register field configurations with
       number of entries mentioned above */
    ADI_ACM_EVENT_CONFIG    *paConfig;

} ADI_ACM_EVENT_CONFIG_TABLE;
```

Description

This structure is used to configure a table of registers/register fields corresponding to specified event numbers. Pointer to an instance of this type is passed as an argument with commands `ADI_ACM_CMD_SET_EVENT_CTRL_REG_TABLE` and `ADI_ACM_CMD_SET_EVENT_TIME_REG_TABLE`

Fields

- `nNumEntries`
  Number of Event register/register field configuration enteries
- `paConfig`
  Pointer to an array of event register/register field configurations with number of entries mentioned above

2.5.1.5 ADI_ACM_EVENT_PARAMS

```
typedef struct __AdiAcmEventParams
{

    /* Event number to configure (Value between 0 and 15 (inclusive)) */
    u8      nEventNumber;

    /* Flag to enable/disable event ('true' to enable, 'false' to disable) */
    bool    bEnable;

    /* ADC Channel select value (Value between 0 and 7 (inclusive)) */
    u8      nAdcCannelSelect;

    /* ADC Range (Accepted values = 0, 1) */
    u8      nRange;

    /* ADC Logic (Accepted values = 0, 1) */
    u8      nLogic;

    /* Time value to execute the corresponding event */
    u32     nTime;

} ADI_ACM_EVENT_PARAMS;
```

Description

This structure is used to pass configuration values of fields related to a particular event. Pointer to an instance of this type is passed as an argument with `ADI_ACM_CMD_SET_EVENT_PARAMS` command

Fields

- `nEventNumber`
  Event number to configure (value must be between 0 and 15, inclusive)
- `bEnable`
  Flag to enable/disable event ('true' to enable, 'false' to disable)
- `nAdcCannelSelect`
  ADC Channel select value (value must be between 0 and 7, inclusive)
- `nRange`
  ADC Range selector (Value must be 0 or 1)
- `nLogic`
  ADC Logic (single/differential) selector (Value must be 0 or 1)
- `nTime`
  Time value to execute the corresponding event

2.5.1.6 ADI_ACM_EVENT_PARAMS_TABLE

```
typedef struct __AdiAcmEventParamsTable
{

    /* Number of Event parameter configuration enteries */
    u32                     nNumEntries;

    /* Pointer to an array of event - parameter configurations with
       number of entries mentioned above */
    ADI_ACM_EVENT_PARAMS    *paEventParams;

} ADI_ACM_EVENT_PARAMS_TABLE;
```

Description

This structure is used to configure a table of events and its fields. Pointer to an instance of this type is passed as an argument with `ADI_ACM_CMD_SET_EVENT_PARAMS_TABLE` command

Fields

- `nNumEntries`
  Number of Event parameter configuration enteries
- `paEventParams`
  Pointer to an array of event - parameter configurations with number of entries mentioned above

2.5.1.7 ADI_ACM_EVENT_STAT

```
typedef struct __AdiAcmEventStat
{

    /* Holds the Event number to query
       Accepted values = between 0 and 15 (inclusive) */
    u8      nEventNumber;

    /* returns 'true' when conversion is completed, 'false' otherwise */
    bool    bIsCompleted;

    /* returns 'true' when event is missed, 'false' otherwise */
    bool    bIsMissed;

    /* returns 'true' when this is the current event in progress,
       'false' otherwise */
    bool    bIsCurrent;

} ADI_ACM_EVENT_STAT;
```

Description

This structure is used to query current status of a particular event. Pointer to an instance of this type is passed as an argument with `ADI_ACM_CMD_GET_EVENT_STAT` command

Fields

- nEventNumber
  Event number to query
- `bIsCompleted`
  Returns as 'true' when conversion is completed, 'false' otherwise
- bIsMissed
  Returns as 'true' when event is missed, 'false' otherwise
- bIsCurrent
  Returns as 'true' when this is the current event in progress, 'false' otherwise

2.5.1.8 ADI_ACM_EVENT_LIST

```
typedef struct __AdiAcmEventList
{

    /* Number of valid event number entries in the below list */
    u32     nNumEntries;

    /* Pointer to an array that can support ADI_ACM_MAX_EVENTS entries
       ACM service will fill this array with valid event numbers that are
       completed or missed */
    u8      *paEventList;

} ADI_ACM_EVENT_LIST;
```

Description

This structure is used to get the list of event numbers that were completed or missed. Pointer to an instance of this type is passed as an argument with `ADI_ACM_CMD_GET_EVENT_COMPLETE_LIST` and `ADI_ACM_CMD_GET_EVENT_MISS_LIST` commands

Fields

- `nNumEntries`
  Number of valid event number entries in the below list

- `paEventList`
  Pointer to an array that can support `ADI_ACM_MAX_EVENTS` entries ACM service will fill this array with valid event numbers that are completed or missed.

  Note: `ADI_ACM_MAX_EVENTS` value for ADSP-BF50x processor family is 16.