Audio Capacitive Touch BoosterPack C5535 Software

User's Guide

Literature Number: TIDA015
September 2014

Contents

List of Figures

List of Tables

Audio Capacitive Touch BoosterPack C5535 Software

1 About Audio Capacitive Touch BoosterPack (ACTBP) C5535 Software

- Runs on the C5535 DSP on the Audio Capacitive Touch BoosterPack board
- Accepts universal asynchronous receiver/transmitter (UART) commands to control audio player and recorder
- Black box "audio accelerator" for MSP430™
- Supports:
  - MP3 encode and decode
  - SD card file system
  - USB mass storage
  - OLED display
  - Codec control, playback and record
  - Low-power mode
- UART client interface to MSP430 on LaunchPad™ board

2 Software Architecture

2.1 Block Diagram

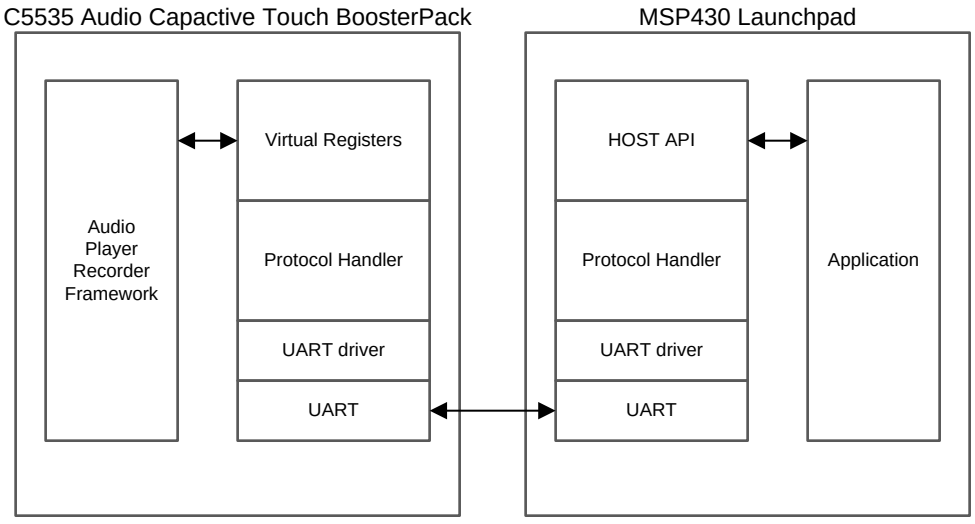

Figure 1. Block Diagram

2.2 Overview

The C5000™ Virtual Register Control Interface (VRCI) is a standard control protocol defined and implemented on many of software frameworks running on C5000 DSPs. VRCI allows an external host to control and communicate to C5000 software framework using the common UART hardware interface. VRCI consists of the following software components:

- Virtual register model

  To make the rich and powerful features easily available to the host on the DSP side, C5000 software framework virtualized the functionality by a set of virtual registers. The virtual register model insulates you from the complexity of the software framework. The model makes the complex software appear to be **lox**. The virtual register set-by-design emulates the look and feel of the register set in many of the fixed function system-on-chip (SoC), hence, presenting itself as a programming model that most of the embedded software programmers are most familiar with.

  For more details, see Section 4 and Section 6.

- Host API

  A set of APIs is defined to allow the host to access the virtual registers of the software framework. The APIs are easy to use and free the developer from the details of the low-level protocol across UART. As part the UIF software package, a reference implementation is provided on hosts running on Windows. Additional implementation will be made available in future releases for MP430 family microcontrollers and other MCU offerings.

  For more details, see Section 7.

- Low-level UART control protocol

  UART is used as the physical link between the host and the C5000 DSP. UART is commonly integrated in many microcontrollers and micro-processors so it is readily available with no added cost. A simple and efficient low-level packet protocol is defined over the UART to allow the host and the DSP to exchange information. The protocol is simple, robust and yet flexible and powerful, which provides the foundation of the upper level UIF software.

  For more details, see Section 8.

3 Download

- BOOTIMG.BIN
  - Bootimg.bin for Audio Capacitive Touch BoosterPack: File:AUDIOCAPDSP-C5535.zip
- Host APIs
  - Doxygen documentation for virtual register host APIs: File:C55 uif host api.zip
- Virtual Registers
  - Doxygen documentation for ACTBP virtual register enums: File:C55 acbp vreg.zip

3.1 UART Virtual Register Control Interface and Host APIs

- Powerful and flexible features made easy via a simple UART virtual register control interface and host APIs
- UART virtual register control interface supports both UART and USB virtual COM port
- Simple and easy-to-use virtual register programming model
- Black box approach simplifies system integration, porting and upgrading

3.2 Audio Playback

- MP3 file playback
- Supports many bit rates and sample rates
- Volume
- Balance
- Fast-forward and rewind
- Time scale modification (TSM)
- Stereo and mono
- Cycle and shuffle modes

3.3 Audio Record

- MP3 file recording
- Stereo and mono
- Selectable bit rates and sample rates
- Default configurations for record: 192 kbps bit rate, 48 KHz sample rate and stereo mode

WARNING

The Audio Capacitive Touch BoosterPack has only a mono-input. The codec on the hardware has been configured to treat the signal as a mono signal, and contains the same information on both left and right channels.

- Bit rate, sample rate, and channel mode (mono and stereo) combinations supported for recording:

Table 1. Sample Rate(1)

	SAMPLING RATE			
[kbps]	8	16	44.1	48
8	M	M	NO	NO
32	MS	MS	NO	NO
64	NO	MS	MS	MS
96	NO	MS	MS	MS
128	NO	MS	MS	MS
192	NO	NO	S	S
320	NO	NO	S	S

(1) M = Only mono mode; S = Only stereo mode; MS = Both mono and stereo mode; NO = Not supported

3.4 Storage Media Support

- MicroSD card
- USB2.0 compliant

3.5 File System Services

- Change to a directory
- Create a directory
- Delete a file or a directory
- Format the SD card
- List a directory

3.6 Boot Mode

- Secure boot from SD card

3.7 Low-Power Playback, Record and Low-Power Modes

- Normal playback and record
- Idle mode
- Sleep mode
- Powerdown mode

4 Virtual Register Model

C5000 software framework that supports the UART virtual register control interface implements a scheme called "Virtual Register Model". The Virtual Register Model virtualizes the rich functionalities and the complicated operations of the software framework via a set of emulated registers. The register model is very similar to the register model often encountered by embedded software developers working with an digital or analog ICs, which have a set of memory-mapped registers. This approach has many benefits:

- Quick learning curve

  Most software developers can quickly become familiar with the virtual register model, and will be able to learn the rich features and functionalities of the software framework and apply what they learn to the products they are working on.

- Black box approach

  The software framework appears to be a black box to the developers. The developers can save the tedious and error-prone process of integrating the software framework code to the application code. Developers are able to continue to stay with their favorite application platform, be a TI MCU and other microcontroller, processors, and DSPs while enjoying the full feature set and capabilities that the C5000 software framework has to offer.

- Portability and ungradable

  On the DSP side, the virtual register model of a software framework stays compatible from one generation to the next. However, the performance, the power and the resource utilization continues to improve. The same virtual register model will also be supported in the future generations of C5000 DSPs. On the application platform side, if the applications use the standard UIF APIs, the application code can be migrated easily to a different platform, which may have a different processor running on a different OS.

  The virtual register set is software framework dependant. Different software frameworks will have different virtual register sets. The idea is to provide a unique set of virtual registers to make the rich and powerful features of a software framework easy to use for the applications. As an example and to give customers a feel of the virtual register model, Table 2 is a quick summary of the virtual register set for the Audio Player Recorder Framework/UIF framework used by Audio Booster Pack Product offered by TI. TI continues to develop additional software frameworks for targeted applications.

5 Table of Virtual Registers for Audio Capacitive Touch BoosterPack

Table 2. C55 Audio Player Recorder Framework Virtual Registers

VIRTUAL REGISTER ADDRESS	VIRTUAL REGISTER NAME	TYPE	R/W	DESCRIPTION
0x00	operation	scalar	R/W	This register invokes the control operations like play, record, stop, pause, resume, and so forth. Each control operation will have a value which is defined by the enum 'UifCommand'
0x01	play_file	array	R/W	This register holds the file name to be played or directory name from which the files will be played.
0x02	record_file	array	R/W	This register holds the name of the file being recorded.
0x03	volume	scalar	R	This register holds the value for the volume configuration.
0x04	balance	scalar	R/W	This register holds the value for the balance configuration.
0x05	stereo	scalar	R/W	This register provides channel selection playback/record mono/stereo.
0x06	control_eq	scalar	R/W	This register provides configurations for the equalizer.
0x07	control_ks	scalar	R/W	This register provides configurations for the TSM.
0x08	record_bit rate	scalar	R/W	This register holds the bit rate for record operation.
0x09	record_sample_rate	scalar	R/W	This register holds the sample rate for record operation.
0x0A	record_format	scalar	R/W	This register provides selection of the file format for the record operation.
0x0B	play_format	scalar	R	This register holds the format of the file being played.
0x0C	play_bit_rate	scalar	R	This register holds the bit rate of the file being played.
0x0D	play_sample_rate	scalar	R	This register holds the sample rate of the file being played.
0x0E	audio_output	scalar	R/W	This register provides selection for the audio output on the HW EVM.
0x0F	audio_input	scalar	R/W	This register provides selection for the audio input on the HW EVM.
0x10	dir_info	array	R	This register holds the directory listing information for the DIR command on the operation register.
0x11	sys_file	array	R/W	This register is used with CD, DEL, DIR and MKDIR command on the operation register.
0x12	power_io	array	R	Register to get the IO power value
0x13	power_core	array	R	Register to get the core power value
0x14	baud_rate	scalar	R/W	Register to get the operating baud rate
0x15	play_number	scalar	R/W	Register to get/set the playback number
0x16	current_play_file	array	R	Register to get name of the file being played
0x17	record_status	scalar	R	Register to get record status
0x18	current_record_file	array	R	Register to get name of the file being recorded
0x19	power_mode	scalar	R/W	Register to configure the power mode
0x1A	sys_status	scalar	R	This register indicates the current system status.
0x1B	file_count	scalar	R	This register holds the number of files with extensions of .mp3 in the current directory.
0x1C	dir_count	scalar	R	Ths register holds the number of the sub-directories in the current directory.
0x1D	play_status	scalar	R	This register indicates the current play status.
0x1E	play_mode	scalar	R	This register indicates the current play mode.
0x1F	shuffle_status	scalar	R	This register indicates if the SHUFFLE is on or off.
0x20	usb_control	scalar	R/W	This register is used to enable or disable USB Mass Storage Class (MSC) mode.
0x21	usb_status	scalar	R	This register indicates if the USB cable is connected or not.
0x22	event_control	scalar	R/W	This register enables or disables the event control.
0x23	version	scalar	R	This register holds the revision number of the Audio Player Recorder Framework/UIF software.
0x24	param	scalar	R/W	This register is a general purpose register that holds a scalar value.
0x25	str	array	R/W	This register is general purpose register that holds a string.

As seen from Table 2, there are two types of virtual registers: scalar and array. All the registers can be read or written by the host. Scalar registers hold a single value used for operations that need only one value to set up the operations. Array registers holds a array of values and are mostly used to hold the name of a file, listing of a directory and any other data that requires long string of values. Section 7 describes a set of host APIs that are provided to enable easy access to these registers.

6 Description of Virtual Registers and Commands

For more information, see the Audio Capacitive Touch BoosterPack Virtual Registers and Commands.

7 Host APIs

The most accurate and detailed description of host APIs can be found File:C55 uif host api.zip and File:C55 acbp vreg.zip.

Table 3. Host API Descriptions

uif_open

This API should open the resource port corresponding to the port specified with the value in PortName.

Syntax

```
Syntax Status uif_open(UIF *hPortHandle, char PortName[5])
```

Arguments

IN	UIF*	hPortHandle
	Handle to the UIF	
	char*	portname
	The name of the port (User specified – COMn)	

Return Value

Status	Success or Failure

Comments None

Constraints None

See Also None

uif_control

This API should open the resource port corresponding to the port specified with the value in PortName.

Syntax

```
Syntax Status uif_control(UIF *hPortHandle,int baud, int datalen, int parity, int stopbit)
```

Arguments

IN	UIF*	hPortHandle
	Handle to the UIF	
	int	baud
	User specified Baudrate	
	int	datalen
	User specified data length	
	int	parity
	User specified parity	
	int	stopbit
	User specified stop bit	
OUT	None	

Return Value

Status	Success or Failure

Comments If an invalid value is entered, the default values of baud=9600, datalen=8, stopbit=0, and parity=0 is set.

Constraints None

See Also None

uif_writeScalar

This API should be called for writing scalar data to port. This takes the virtual reg address and the value to write to port as its arguments.

Syntax

```
Status uif_writeScalar(UIF uif,INT8 virtualRegisterAddr,INT32 value,INT16 timeout)
```

Arguments

IN	UIF*	hPortHandle
	Handle to the UIF	
	INT8	virtualRegisterAddr
	Virtual register name like play_file, time_out, dir_info, and so forth	
	INT32	value
	Value to write	
	INT16	timeout
	Timeout value	
OUT	None	

Return Value

Status	Success or Failure

Comments None

Constraints None

See Also None

uif_writeArray

This function is used to write array content to port. This API is called for instructions that take a string as parameter (for example, $play_file).

Syntax

```
Status uif_writeArray(UIF uif, INT8 virtualRegisterAddr, INT16 length, CHAR*
arrayToWrite, INT16 timeout)
```

Arguments

IN	UIF*	hPortHandle
	Handle to the UIF	
	INT8	virtualRegisterAddr
	Virtual register name like play_file, time_out, dir_info, and so forth	
	INT16	length
	Length of array to write	
	CHAR*	arrayToWrite
	Array to write	
	INT16	timeout
	Timeout value	
OUT	None	

Return Value

Status	Success or Failure

Comments None

Constraints None

See Also None

uif_readScalar

This function is used for instructions where there is a file name or content of variable length to be received from Audio Player Recorder Framework (File=$current_play_file).

Syntax

```
Status uif_ReadScalar(UIF hPortHandle, INT8 virtualRegisterAddr, INT32 *value,
INT16 timeout
```

Arguments

IN	UIF*	hPortHandle
	Handle to the UIF	
	INT8	virtualRegisterAddr
	Virtual register name like play_file, time_out, dir_info, and so forth	
	INT16	timeout
	Timeout value	
	INT32	value
OUT	Read-back value	

Return Value

Status	Success or Failure

Comments None

Constraints None

See Also None

uif_readArray

This function is used for instructions where there is a file name or content of variable length to be received from Audio Player Recorder Framework (File=$current_play_file).

Syntax

```
Status uif_ReadArray(UIF uif, INT8 virtualRegisterAddr, INT16* length, CHAR*
arrayToRead, INT16 timeout
```

Arguments

IN	UIF*	hPortHandle
	Handle to the UIF	
	INT8	virtualRegisterAddr
	Virtual register name like play_file, time_out, dir_info, and so forth	
	INT16	timeout
	Timeout value	
OUT	INT16*	length
	Length of array read back	
	CHAR*	arrayToRead
	Array read back	

Return Value

Status	Success or Failure

Comments None

Constraints None

See Also None

7.1 Programming Examples Using Host APIs

```
include <uif_app.h>
UIF uif; void main() { int status; int my_play_number;
```

```
status=uif_open (&uif,"\\\\.\\COM4");                                    // Open COM4 as the
communication port
status=uif_control (&uif,9600,8,NOPARITY, ONESTOPBIT);                   // Set up the port
```

```
uif_WriteScalar(uif,UIF_CMD_BAURATE,BAU3);                               // $baurate= BAU3
uif_WriteArray(uif, UIF_CMD_SYS_FILE, "\\");                             // $sys_file="\"
uif_WriteScalar(uif,UIF_CMD_OPERATION,DIR);                              // $operation=DIR
uif_ReadArray(uif,UIF_CMD_DIR_INFO, &len, my_dir_info);                  // my_dir_info=$dir_info
printf("%s\n",my_dir_info); //display my_dir_info

uif_WriteArray(uif, UIF_CMD_SYS_FILE, "\\");                             // $sys_file="\Pop"
uif_WriteScalar(uif,UIF_CMD_OPERATION,DIR);                              // $operation=DIR
uif_ReadArray(uif,UIF_CMD_DIR_INFO, &len, my_dir_info);                  // my_dir_info=$dir_info
printf("%s\n",my_dir_info);                                              // display my_dir_info
uif_WriteArray(uif, UIF_CMD_PLAY_FILE, "Song2");                         // $play_file="Song2"
```

```
uif_writeScalar(uif,UIF_CMD_CONTROL_EQ,0x5678);                          // $control_eq=0x5678
uif_writeScalar(uif,UIF_CMD_CONTROL_KS,2);                               // $control_ks=2
uif_writeScalar(uif,UIF_CMD_STEREO,1);                                   // $stereo=1
uif_writeScalar(uif,UIF_CMD_OPERATION,UIF_CMD_PLAY_LIST);                // $operation=PLAY_LIST
uif_writeScalar(uif,UIF_CMD_OPERATION,UIF_CMD_VOLUME_UP10);              // $operation=VOLUME_UP10
uif_writeScalar(uif,UIF_CMD_OPERATION,UIF_CMD_BALANCE_LEFT05);           // $operation=BALANCE_LEFT05
```

```
uif_ReadScalar(uif,UIF_CMD_PLAY_NUMBER, &my_play_number);                // display $play_number
printf("%d\n", &my_play_number);
```

8 Low-Level UART Control Protocol

For more information, see the Low-level UART Virtual Register Control Protocol.

9 Frequently Asked Questions (FAQ)

9.1 What Debugging Ability Do I Have Over the C5535 DSP Code?

A: The C5535 DSP code is a black box for the ACTBP due to licensing restrictions.

9.2 How Can I Update the ACTBP C5535 DSP Software?

A: The ACTBP C5535 DSP software cannot update.

A: If you have lost, broken, or formatted the SD card that shipped with the ACTBP kit, see Section 9.51.

9.3 Is a WAV File Supported (PCM, MS-ADPCM, IMA-ADPCM)?

A: It is not supported at this time.

9.4 Is FLAC Supported?

A: It is not supported at this time.

A: Vinjey, a TI 3rd party has this codec.

9.5 Is OGG Supported?

A: It is not supported at this time.

A: Vinjey, a TI 3rd party has this codec.

9.6 What MP3 Playback Rates Can be Supported?

A. For more information, see the data sheet from http://software-dl.ti.com/dsps/dsps_public_sw/codecs/C55X_Audio/index_FDS.html.

9.7 What ID3 Tag Versions are Supported?

A: ID3 version 1.1 and 2.4

9.8 How Many Bands of EQ Can be Supported?

A: Equalizer module is currently configured for 5 bands. Band number is configurable, and could be configured to support up to 14 bands.

9.9 What Kind of filesystem is Supported?

A: FAT16 and FAT32

9.10 Is exFAT Supported?

A: This is unknown and is being investigated (as of Dec 8, 2011).

9.11 Is SDHC, SDXC Supported?

A: SDHC is supported, and it has been tested with a 16GB SD card. It should be capable of supporting up to 32GB card, though it has not been tested (as of Dec 8, 2011).

9.12 What Version of USB is Supported?

A: Currently only USB 2.0 is supported.

9.13 Why is USB MSC Not Working with My PC?

A: You may have an older PC with USB 1.0. USB 1.0 is not currently supported.

A: Please try a different USB port. Differences have been seen in PC USB ports among PC manufacturers

9.14 What Bit Rates and Sample Rates are Supported for Recording?

A: For more recording information, see Table 4.

Table 4. Bit and Sample Rates (1)

kbps\KHz	8	16	44.1	48
8	M	M	NO	NO
32	MS	MS	NO	NO
64	NO	MS	MS	MS
96	NO	MS	MS	MS
128	NO	MS	MS	MS
196	NO	NO	S	S
320	NO	NO	S	S

(1) M = Only mono mode; S = Only stereo mode; MS = Both mono and stereo mode; NO = Not supported

9.15 Why Do My Recordings Have Noise?

A: You may be using a 16GB Kingston SD card or other untested SD cards. This is a known issue in work for resolution. The 16GB SDs card that has been tested and is known to work is SanDisk Class-10 SD cards.

9.16 In Audio Player Recorder Framework/UIF, What Baud Rates are Supported with the UIF Interface?

The following baud rates are supported:

9600, 14400, 19200, 38400, 57600,115200, and 230400

To Change the buad rate, on the command console, issue:

```
$baudrate=<value>
```

<value> is one of the supported numerical values of the baud rates. For example, to change to 19200 baud rate, issue:

```
$baudrate=19200
```

You could also use host APIs in your applications to change the baud rate by following the steps below:

1. You could also use host APIs in your applications to change the baud rate by following the steps below:
2. Check the return value of the *uif_WriteScalar()* function. Return value will be ‘0’ if baud rate configuration on Audio Player Recorder Framework/UIF is successful.

NOTE:

- Audio Player Recorder Framework sends acknowledgment for baud rate command and waits for 100 msecs (to allow the transfer completion of ACK) before changing the baud rate to new value. Host applications should accommodate for this time delay.
- Parameter 'value' of the *uif_WriteScalar()* function should be the actual value of the baud rate to be configured. Valid values are 9600, 14400, 19200, 38400, 57600, 115200, and 230400.

9.17 What Happens if an Invalid Baud Rate Value is Sent to Audio Player Recorder Framework?

A: Audio Player Recorder Framework sends error code 0x03 along with the acknowledgment for the set baud rate command. This error code will be returned by the *uif_WriteScalar()* function.

9.18 In Audio Player Recorder Framework, How Many Files and Directories Can be in the Root Directory and How Many Files Can be in Each Subdirectory?

A: Audio Player Recorder Framework only supports one level of subdirectories. In the root directory, the number of files plus the number of subdirectories must be less or equal to 50. In each sub directory, there can be no more than 50 files. Only files with .MP3 ending will be recognized.

9.19 In Audio Player Recorder Framework, How Do I Enable and Disable USB Mass Storage Class (MSC) Function?

A: Use $usb_control=1 to enable the USB MSC and $usb_ocntrol=0 on the command console or corresponding host APIs to disable USB MSC. Since it takes 15 to 20 seconds for the host to enumerate the USB MSC, Audio Player Recorder Framework will be in a state, which lasts for 20 seconds waiting for the host to complete the enumeration. During this period, all UIF commands are ignored, including $usb_control=0 command. The host is advised to wait at least 20 seconds to issue any command.

9.20 In Audio Player Recorder Framework, How Do I Control the Volume?

A: $volume is read-only register allows the host to read the level of volume. For example, after power-up reset:

```
my_volume=$volume
display my_volume
```

shows the default value of $volume and, in the current implementation, a default value of 81 will be shown. $volume ranges from 0 to 99.

To change the value of $volume, use:

```
$operation=VOLUME_UPxx
```

or

```
$operation=VOLUME_DOWNxx
```

to turn up or turn down the volume. xx ranges from 00 to 16.

Note in the current implementation, when you issue $operation=VOLUME_UPxx command, the value of $volume will not inclrease by a exact value of xx. Similar is true for the case of VOLUME_DOWNxx. Please refer to the next Q&A for a related issue.

9.21 Why is the Zero Level Around 34 to 40? The Volume Doesn't Seem Linear with the Number.

A: The value of $volume does not relate linearly to the perception. There is not enough steps in the high volume range but too many in the low volume ranage. This is a known issue and will be addressed in a future release (ENH 1415)

9.22 Why Does the Record Sometimes Feel Like It is Hanging When It Starts?

A: With slower SD cards, when the file system searches for a free cluster to create the recorded file it searches from the first sector in the card until it finds enough free clusters to store the file data. If there are more free clusters at the beginning of the card, the faster it will find the space it needs. If the card is more full, then the search will take longer. On some SD cards with varying amounts of data, search times up to 18-20 seconds have been observed.

9.23 What are the Baud Rates Supported in Audio Player Recorder Framework?

A: Audio Player Recorder Framework supports the baud rates 9600, 14400, 19200, 38400, 57600, 115200 and 230400 bps.

9.24 How Do I Configure the Baud Rate of the UART Interface?

A: To change the baudrate from command console use the below command as an example:

```
$baudrate = 19200
```

This command will change the baud rate to 19200.

9.25 What is the Procedure for an Application to Configure the Baud Rate With Host APIs?

A: Instead of using the command console, an application can use host APIs to set the baud rate. Follow the steps below when setting baud rate:

1. Call the *uif_WriteScalar()* function with parameter ‘virtualRegisterAddr’ set to ‘UIF_CMD_BAUDRATE’ and parameter ‘value’ set to value of the baudrate that needs to configurted.
2. Check the return value of the *uif_WriteScalar()* function. The return value will be ‘0’ if the baud rate configuration on Audio Player Recorder Framework is successful. If successful, change the baud rate on the host.

NOTE:

- Audio Player Recorder Framework sends acknowledgment for baud rate command and waits for 100 msecs (to allow the transfer completion of ACK) before changing the baud rate to new value. Host applications should accommodate for this time delay.
- Parameter 'value' of the *uif_WriteScalar()* function should be the actual value of the baud rate to be configured. Valid values are 9600, 14400, 19200, 38400, 57600, 115200, and 230400.

9.26 What Happens If an Invalid Baud Rate Value is Sent to Audio Player Recorder Framework?

A: Audio Player Recorder Framework sends an error code 0x03 along with the acknowledgment for the set baud rate command. This error code will be returned by the *uif_WriteScalar()* function.

9.27 What is the Function of $play_number Virtual Register?

A: $play_number virtual registers contains the index of the song from $play_file to be played next. This register could be written and read by the user. $play_file contains the list of songs to be played. $play_file contains either the name of a directory or the name of a file.

9.28 How Does play_number Command Work?

A: Behaviour of the play_number command differs based on the system state. During a playback initiated by PLAY_LIST, PLAY_NEXT or PLAY_PREV commands, $play_number contains the index of the song that are currently being played. The song that will be played next is usually determined by the Audio Player Recorder Framework software depending the state CYCLE and SHUFFLE. A user also has the option to overwrite the default sequence by writing a differnt value to $play_number before or during the playback. If the value is written before the playback, the next song that will be played will be $play_number. If the value is wriiten during playback, the next song to be played will be $play_number+1.

Note that the user-specified $play_number value is only recongized by Audio Player Recorder Framework SW with PLAY_LIST and PLAY_NEXT commands. When playing back songs using $operation=PLAY_PREV command, user-specified $play_number value is ignored.

$play_number usage examples:

Example 1. $play_number is Set When Playback Has Not Started

```
$op=DIR
dirinfo=$dir_info
display dirinfo

01 - Will I Am 206kbps 44khz .mp3
02 - Do It Again 205kbps 44khz .mp3
03 - Jai Ho 320kbps 44khz.mp3
04 - sample 128kbps 44kHz.mp3

$play_number=2
$op=PLAY_LIST     ----> Plays the song  “02 - Do It Again 205kbps 44khz .mp3”
```

9.29 $play_number is Set During Playback

```
$op=DIR
dirinfo=$dir_info
display dirinfo

01 - Will I Am 206kbps 44khz .mp3
02 - Do It Again 205kbps 44khz .mp3
03 - Jai Ho 320kbps 44khz.mp3
04 - sample 128kbps 44kHz.mp3

$op=PLAY_LIST
$play_number=2
$op=PLAY_NEXT   ----> Plays the file “03 -
 Jai Ho 320kbps 44khz.mp3”. Leaving the playback just to run till the end of the first song will
also have the same effect

$play_number=3
$op=PLAY_PREV   ----> Play number will ignored and the file “02 -
 Do It Again 205kbps 44khz .mp3” will be played
```

9.30 What is the Valid Range for Playback? Will Audio Player Recorder Framework Send Error Code for Invalid play_number?

A: The valid tange depends on the state of playback. play_number can be 1 to maximum file count when playback is stooped. Play_number can be 1 to (maximum file count – 1) when playback is running. Audio Player Recorder Framework sends no error for invalid play_number, it simplily ignores the command.

9.31 How Does Audio Player Recorder Framework Add Time-Stamps to Recoded or Newly Created Folders?

A: There is no support for real-time tracking for the current time and date in Audio Player Recorder Framework. HW RTC module is used to track the time from the system start-up. During Audio Player Recorder Framework initialization RTC module shall be configured to date 12 February 2012 and time 18:00:00. RTC counts from this initial time with real-time precision. Time stamps will be added to the recorded or newly created folders based on this time.

9.32 Where the Files Recorded by Audio Player Recorder Framework Will be Stored on SD Card?

A: Audio Player Recorder Framework stores recorded files on the SD card in a directory with name 'RecDir'. If there is no directory with name 'RecDir', it will be created.

9.33 How Does Audio Player Recorder Framework Assigns Names for the Recoded Files?

A: Audio Player Recorder Framework uses indexed naming format for recoded files as RECxxx, whereas 'xxx' varies from 000 to 049.

9.34 Can a User Configure the Name for Files to be Recoded?

A: Yes, a user can specify record file name by setting the $record_file register each time before starting the record operation. File name in the 'record_file' register will be valid for only one record operation. If user specifies same name twice in the record_file register, Audio Player Recorder Framework ignores the name and assigns file name using default naming format. If user specifies no file name, record file name will be set using default naming RECxxx.

9.35 What is the Maximum Limit for Record File Count? What Happens If the User Tries to Record Beyond the Limit?

A: Audio Player Recorder Framework allows record up to 50 files. When the user tries to record more than 50 files, the record will not be started and Audio Player Recorder Framework sends an error code (0x07) to the host UIF to indicate that the maximum file limit for record has reached. The file count limit is applicable to all the audio files (MP3/WMA/AAC) in RecDir irrespective of whether they are created by Audio Player Recorder Framework or copied by user by some other means.

9.36 If the User Deletes Some Files in 'RecDir' Folder, How Can the Files Will be Named by Audio Player Recorder Framework in Subsequencial Recordings?

A: File naming always be done in sequence starting from REC000 to REC049. If some files are deleted in between, next record operation fills those gaps and continues with the sequence. For example, assume the user does record to create files from REC000.mp3 to REC020.mp3; he deletes files from REC010.mp3 to REC015.mp3 and starts the record. Then the file naming sequence starts from REC010.mp3 continues until REC015.mp3 and then proceeds with the naming from REC021.mp3 till REC049.mp3.

9.37 What are the File Formats Supported for Playback?

A: Audio Player Recorder Framework supports MP3 file format for playback. However Audio Player Recorder Framework framework reads the information of MP3, WMA and AAC files during the file browsing and getting file listing for playback. Audio Player Recorder Framework playback stops when it reads WMA or AAC files for playback. Playback can be again initiated using play command.

9.38 What Happens If I Select an Invalid Configuration for Record?

- When invalid bit rate or sample rate is selected, record will happen at default configurations; 192kbps bit rate, 48 KHz sample rate and stereo mode.
- When invalid channel mode is selected and if bit rate and sample rate combinations are valid, then channel mode supported for that combination will be selected.
- For example:
  - Assume 8 kbps bit rate, 8 KHz sample rate and stereo mode is selected for record. Since stereo mode is not supported at 8 kbps and 8 KHz record happens with mono mode, 8kbps bit rate and 8 KHz sample rate.
  - Assume 320 kbps bit rate, 48 KHz sample rate and mono mode is selected for record. Since mono mode is not supported at 320 kbps and 48 KHz record happens with stereo mode, 320 kbps bit rate and 48 KHz sample rate.

9.39 Do I Get an ACK Error Code When I Select an Invalid Record Configuration?

A: No error ACK shall be sent to the host UIF when invalid record configurations are selected. Record configurations are validated only when record operation is initiated by the host UIF and the corresponding virtual registers shall be modified to valid values. Record configuration virtual registers can be read from host UIF during the record to check at what configurations record is happening.

9.40 Does the PLAY Command Work While in Fast Forward or Rewind Mode?

A: No, the Play command is not supported when the system is in fast forward or rewind mode.

9.41 Does the Pause Command Work While in Fast Forward or Rewind Mode?

A: No, the Pause command is not supported when the system is in fast forward or rewind mode.

9.42 Why Does Record in 320 kbps Have Noise in the Recording?

A: When recording at higher bitrates such as 320 kbps, the SD card may not be able to keep up with the encoded data being written. This is typically the case with SD cards which have no speed classification (for example, they are not explicitly marked as supporting class 4, class 6, class 10, and so forth).

A: Use a faster SD card.

9.43 What are the Baud Rates Supported by Audio Player Recorder Framework?

A: Audio Player Recorder Framework supports the baud rates 9600, 14400, 19200, 38400, 57600, 115200, and 230400 bps.

9.44 How to Do I Configure the Baud Rate from the Host?

A: You could use the host APIs as follows:

- Call the *uif_WriteScalar()* function with parameter 'virtualRegisterAddr' set to 'UIF_CMD_BAUDRATE' and parameter 'value' set to value of the baudrate needs to configured.
- Check the return value of the *uif_WriteScalar()* function. Return value will be '0' if baud rate configuration on Audio Player Recorder Framework is successful. If successful, change the baud rate on the host.

NOTE:

- Audio Player Recorder Framework sends acknowledgment for baud rate command and waits for 100 msecs (to allow the transfer completion of ACK) before changing the baud rate to new value. Host applications should accommodate for this time delay.
- Parameter 'value' of the *uif_WriteScalar()* function should be the actual value of the baud rate to be configured. Valid values are 9600, 14400, 19200, 38400, 57600, 115200, and 230400.

9.45 What Happens If an Invalid Baud Rate Value is Sent to Audio Player Recorder Framework?

A: Audio Player Recorder Framework sends the error code 0x03 along with the acknowledgment for the set baud rate command. This error code will be returned by the *uif_WriteScalar()* function.

9.46 What is play_number Command and How Can It be Used?

A: The play_number command sets the index of the song to be played next

A: Behaviour of the play_number command differs based on the system state:

- Playback not running: play_number is set
- Playback started: Index of the song played will be play_number. Sine play_number is set during playback, the song played next will be play_number +1, whether it is by using play_next command or by leaving the play to run the next song. play_number will be ignored when the previous file is selected using play_prev command.

A: play_number command examples:

- play_number is set when playback is not running

```
$op=DIR
dirinfo=$dir_info
display dirinfo

01 - Will I Am 206kbps 44khz .mp3
02 - Do It Again 205kbps 44khz .mp3
03 - Jai Ho 320kbps 44khz.mp3
04 - sample 128kbps 44kHz.mp3

$play_number=2
$op=PLAY_LIST
```

- The $play_number=2 command plays the song “02 - Do It Again 205kbps 44khz .mp3”
- play_number is set when playback is not running

```
dirinfo=$dir_info
display dirinfo

01 - Will I Am 206kbps 44khz .mp3
02 - Do It Again 205kbps 44khz .mp3
03 - Jai Ho 320kbps 44khz.mp3
04 - sample 128kbps 44kHz.mp3

$op=PLAY_LIST
$play_number=2
$op=PLAY_NEXT   ----> Plays the file “03 -
 Jai Ho 320kbps 44khz.mp3”. Leaving the playback just to run till the end of the first song
will also have the same effect

$play_number=3
$op=PLAY_PREV   ----> Play number will ignored and the file “02 -
 Do It Again 205kbps 44khz .mp3” will be played
```

9.47 What is the Valid Range for Playback and Will There be an Error Code for Invalid play_number?

- Playback NOT running: play_number can be 1 to maximum file count
- Playback running play_number can be 1 to (maximum file count 1)

A: There is no error for invalid play_number, it just ignores the command.

9.48 Why is the Date for Recorded Files Feb 12, 2012? How are the File Dates Calculated?

A: There is no API support for adjusting the RTC at this time. However, the HW RTC module is used to track the time from the system start-up. During initialization, the RTC module is configured to 12 February 2012 and time 18:00:00. From then on the RTC counts from this initial time with real-time precision. Time stamps will be added to the recorded or newly created folders based on this time.

9.49 Is There an API for Setting the RTC?

A: Not at this time. This has been filed as an enhancement request.

9.50 What do the Acknowledge Error Codes Returned From Audio Player Recorder Framework Mean?

A: Error codes are as below:

```
0 - The last operation is successful
1 - Generic failure. Used internally, host never receives this error code
2 -
 Invalid operation command. This error code will be sent when invalid operation command($op=xyz)
is received.
3 -
 Command or command parameters invalid. This error code will be sent if the command requested
will not match with any of the four requests; scalar write, scalar read, array read and array
write.
    This error code is also sent when parameters associated with a command are not valid. For
invalid baud rate values, this error code is sent.
4 -
 Invalid operation. This error code is to be sent when an invalid operation, like writing a read-
only register, is requested. This is not implemented on C55 UIF. Command console takes care of
filtering the invalid operations.
5 - RX buffer overflow - Host sends a array write request of length more than 50 characters.
6 - Command is invalid for the current system state -
 Host sends a command which is not supported in the current system. This is only implemented for
USB MSC to indicate that enumeration is in progress.
7 - File count in the record directory has reached maximum limit.
```

- There are only error codes for selected commands. For most of the commands all the invalid operations will be ignored. In the future, more error codes will be added.
- Format of the acknowledgment packet is as shown here:

ACK ID	RSV0	RSV1	RSV2	Status Byte 1	Status Byte 0

- Status byte 0 and 1 contains the error codes. Since we transfer MSB first, 5th and 6th byte of 6 byte acknowledgment packet on MSP will have this error codes.
- ACKID can be:
  - 4 - Write completion acknowledgment
  - 5 - Read completion acknowledgment

9.51 What Should I Do If I Have Lost, Broken, or Formatted the SD Card That Shipped with the ACTBP Kit?

A: Contact TI for a replacement device.

A: The SD card contains BOOTIMG.bin (bootloader image), MP3 files, and the MP3 CODEC. Only the BOOTIMG.bin file is downloadable due to royalties.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265