Application Report
TIDA017–September 2014

Audio Capacitive Touch BoosterPack™ MSP430™ Software

Rafael Mena

ABSTRACT

The LaunchPad™ ACTBP host application is the firmware running on the MSP430 LaunchPad development kit to demonstrate the Audio Player Recorder Framework for the C5000™ Audio Capacitive Touch BoosterPack.

Contents

List of Figures

List of Tables

1 Audio Capacitive Touch BoosterPack (ACTBP) Host Application Overview

The host application was developed for the value-line MSP430G2553 microcontroller (MCU), using its dual timers with capture and compare registers and hardware universal asynchronous receiver/transmitter (UART) peripherals to efficiently perform the task of a host MCU controlling signal processing operations on a DSP. The host application on the MSP430 MCU interfaces to components on the on the ACTBP to accept user inputs from the capacitive touch sensors, drive LEDs and send UART commands to the DSP software to achieve the desired system response.

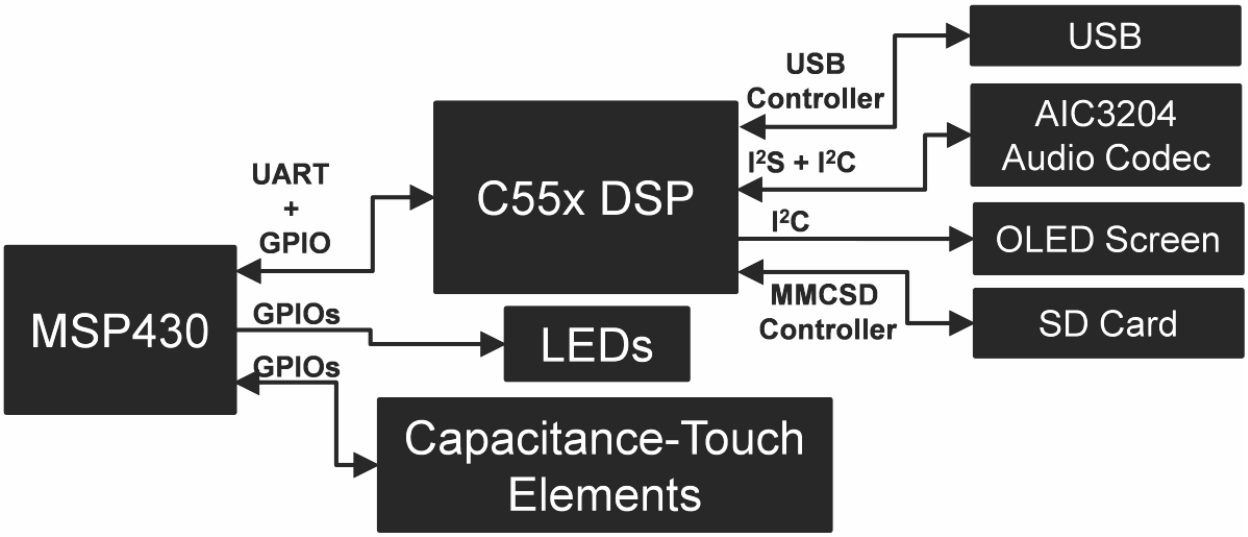

Figure 1. Host Application

The host application also sends message texts displayed on the organic light-emitting diode (OLED) screen to the DSP and enables and disables the USB Mass Storage Connection (MSC) from DSP to the host PC.

2 Host Application User Interface

Upon powering-up the system from the ACTBP USB connector or from the Power-Off mode, the MSP430 MCU on the LaunchPad first boots up the host application from on-chip Flash and initializes its peripherals and other input/output ports. The host application then enables power to the rest of the circuitry on the ACTBP including the C5535 DSP, AIC3204 codec and OLED screen. The host application also drives the LEDs on the ACTBP in a clockwise and then a faster counterclockwise pattern.

Upon DSP initialization, the host application is in the *Browse Directory* mode and the Audio Player Recorder Framework on the DSP is in CPU idle mode waiting for a command on the UART interface. Only valid user inputs for a particular mode result in a UART command being sent to the DSP. The host application enables the CYCLE option in the framework to play MP3 files in a selected folder in a continuous loop.

2.1 Browse Directory Mode

In *Browse Directory* mode, scroll clockwise or counterclockwise on the capacitive-touch scroll wheel to display directory folders in the root directory. Tapping the CENTER location selects the folder displayed on the OLED screen (root directory or any folder in root). The following user inputs are valid in the *Browse Directory* mode:

- Scroll RIGHT: Next directory
- SCROLL LEFT: Previous directory
- Tap CENTER location: Select directory and go to *Browse File* mode
- Double-tap DOWN location: Enable *USB MSC* mode
- Hold UP location: *Power-Down* mode

NOTE:

- The Audio Player Recorder Framework imposes a limit of 50 objects (files or directories) in any folder (including the root folder)
- Folders in the root directory cannot have sub-directories

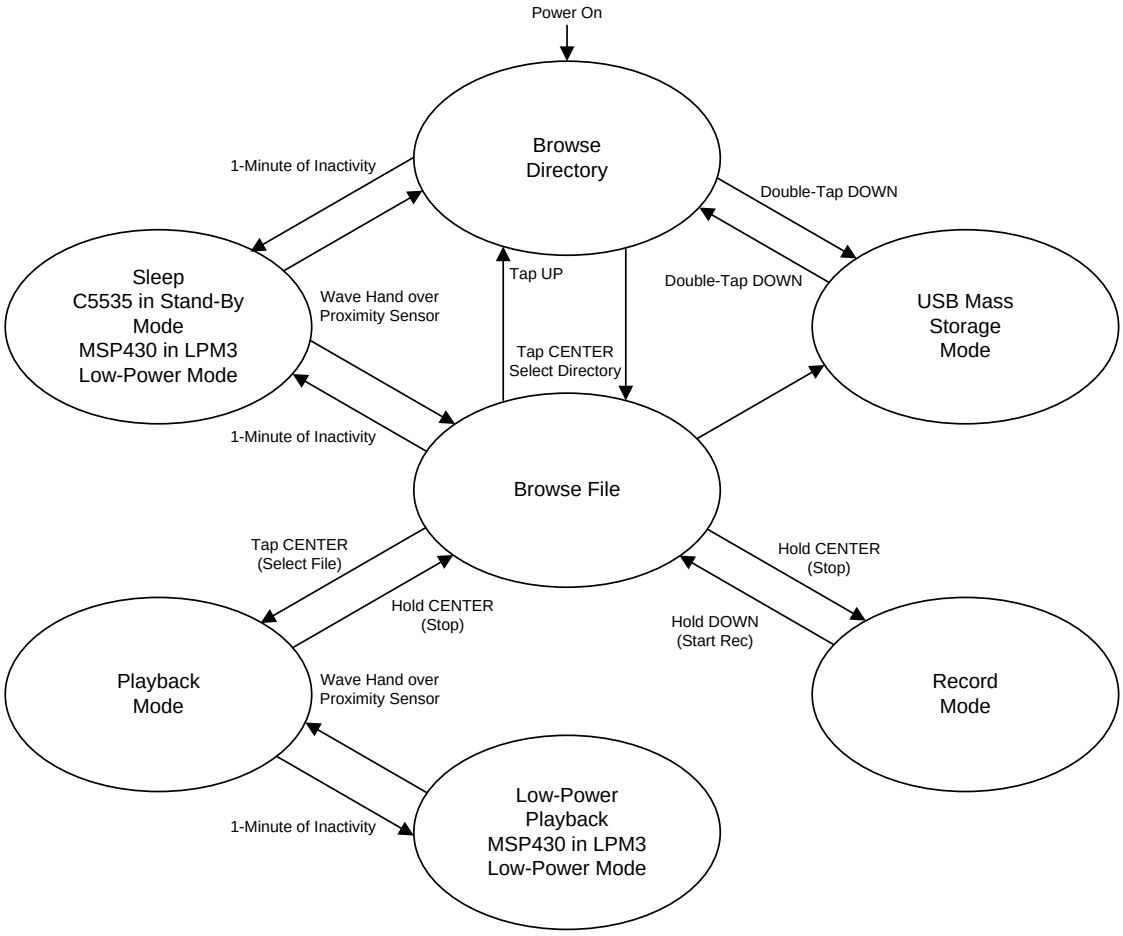

Figure 2. Browse Directory Mode

2.2 Browse File Mode

Upon selecting a folder, the host application changes to the *Browse File* mode. Scrolling now browses through the MP3 files in the chosen folder, both of which are displayed on the OLED screen. Tapping the CENTER location selects the MP3 file and starts playing the file in *Playback* mode. Holding the DOWN location for around 2 seconds sets the Audio Player Recorder Framework into *Record* mode. Currently, the Audio Player Recorder Framework only supports MP3 format files.

The directory folders and files are not sorted in alphabetical order, but in order of its location in the file system on the microSD card. Usually, this is the order in which the folders or files were created or copied on to the microSD card.

The following inputs are valid in the *Browse File* mode as shown in Figure 3:

- Scroll RIGHT: Next file
- Scroll LEFT: Previous file
- Tap UP location: Go to *Browse Directory* mode
- Tap CENTER location: Play selected file in *Playback* mode
- Double-tap DOWN location: Enable USB MSC
- Hold DOWN location: Start recording in *Record* mode
- Hold UP location: *Power-Down* mode

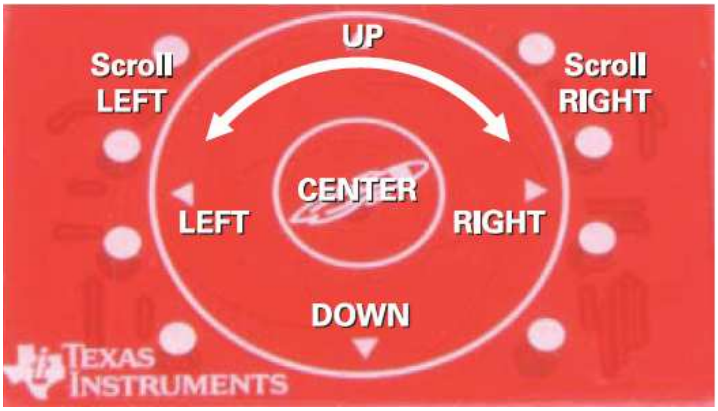

Figure 3. Browse File Mode

2.3 USB Mass Storage Mode

In both the *Browse Directory* and *Browse File* mode, enter the USB Mass Storage mode by double tapping (twice) on the DOWN location. The contents of the microSD card on the ACTBP shows up as a removable drive on the host computer. To come out of USB MSC mode, eject the drive on the host computer and double-tap the DOWN location after the OLED screen displays the *USB Mass Storage Enabled* message. The host application returns back to the *Browse Directory* mode.

2.4 Playback Mode

In *Playback* mode, the Framework plays MP3 files in the selected folder starting from the selected MP3 file in a repeat loop. The following inputs are valid in the *Playback* mode:

- Scroll RIGHT: Volume up
- Scroll LEFT: Volume down
- Tap UP location: Enable or disable shuffle
- Tap CENTER location: Pause or resume *Playback* mode
- Tap RIGHT location: Skip to next file
- Tap LEFT location: Skip to start of current file
- Double-tap LEFT location: Skip to previous file
- Hold RIGHT location: Fast forward
- Hold LEFT location: Rewind
- Hold CENTER location: Stop *Playback* mode and go back to *Browse File* mode
- Hold UP location: *Power-Down* mode

If *Shuffle* is enabled, the play order is randomized and an indicator 'S' is displayed on the OLED screen. During *Fast Forward* or *Rewind*, if the end or start of file is reached, normal playback of the next or previous file in the play order is resumed. The host application reverts back to *Browse File* mode when playback is stopped by holding the CENTER location.

2.5 Record Mode

The host application includes power management logic to put the MSP430 MCU or both MCU and the C5535 DSP into low-power modes for optimal power savings.

- **Power-Down Mode:** From any mode, holding the UP location for two seconds powers down the system. In this state, the ACTBP (C5535 DSP, OLED, and AIC3204) is powered off and the MSP430 MCU is in LPM3 mode. The MSP430 MCU wakes up every 750 ms to scan the capacitive touch sensors for a user input. Turn on the system again by holding the UP location again for two seconds.
- **Standby:** If no user input is detected (no touch activity on the scroll wheel or on the center button), for one minute in the *Browse Directory* or *Browse File* modes, the host application puts the C5535 DSP in low-power RAM-retention standby state. This state is indicated by the center LED being turned on. In this state, the master clock to the DSP components is turned off and the OLED display is turned off. The MSP430 MCU is in LPM3 mode. The MCU wakes up every 750 ms to sample the proximity sensor. Bring the system of *Standby* by waving a hand 3 to 5 cm over the capacitive touch sensors on the ACTBP.
- **Low-Power Playback:** In *Playback* mode, if no user input is detected for one minute, the host application puts the MSP430 MCU in LPM3 mode. The center LED blinks slowly in this state. The MCU wakes up every 750 ms to sample the proximity sensor. Bring the system out of this low-power mode by waving a hand 3 to 5 cm over the capacitive touch sensors on the ACTBP.

3 Developing ACTBP Host Application on the MSP430 LaunchPad

The ACTBP is shipped with a MSP430G2553IN20 MCU programmed with the LaunchPad ACTBP host application firmware. The contents of this section describes steps to develop, customize and debug the firmware using Code Composer Studio™ (CCS) version 4+ Integrated Development Environment (IDE).

3.1 Hardware Preparation

To ensure proper connection to the MSP430 emulation to debug MSP430 host application, it is necessary to do the following steps:

1. Remove jumper 'VCC' from the LaunchPad J3 header. Keep the default jumper settings on the ACTBP.
2. Connect cables from a host computer to micro USB ports on both the LaunchPad and the ACTBP. Only the MSP430 Flash emulation is powered from the LaunchPad USB connection. The ACTBP and the MSP430G2553 MCU on LaunchPad is powered from the ACTBP USB connection.

3.2 Software Download and Preparation

The steps described in the following sections are only required for developing or customizing the LaunchPad ACTBP host application firmware. The ACTBP host application operates on the LaunchPad platform using the MSP430G2553 device and the Audio Capacitive Touch Booster Pack plugin board. The capacitive touch and proximity sensing are enabled by the pin oscillator feature. The application also uses the Capacitive Touch Sense Library to realize and measure the capacitive touch and proximity sensors. The Capacitive Touch Sense Library provides layers of abstraction to generate higher logical outputs such as logical touches and their position (in this hardware, a four-button wheel).

3.2.1 Downloads

Table 1. Download Versions

Link	Version	Release Date	Description
File:ACTBP msp430v1.1.zip	v1.1	13-Apr-12	MSP430G2553 LaunchPad host application for Audio Capacitive Touch Booster Pack with bug fixes
File:ACTBP msp430.zip	v1.0	27-Mar-12	Initial release of MSP430G2553 LaunchPad host application for Audio Capacitive Touch Booster Pack

A zip file of the host application can be downloaded from the link listed here. All software is provided in both binary, executable and source code forms. When this package is installed, all CCS v4+ project files and the source code for the ACTBP host application can be found in the **[INSTALL_PATH]\Source** folder in the selected installation directory. The application firmware binary is installed to the **[INSTALL_PATH]\Software** folder.

3.2.2 Download Link for Code Composer Studio (CCS) IDE

The CCS IDE is required to develop application on the LaunchPad. For more information on how to start developing applications for the LaunchPad and how to install the drivers and IDE that it requires, see http://www.ti.com/ww/en/launchpad/launchpads.html.

3.2.3 Project and File Configuration

```
PROJECT_ROOT  - ACTBP_Host_App
     |   ACTBP_Host_App.c        <C file: Main application code>
     |   ACTBP_HostVars.h        <H file: Variables definition for host application>
     |   C55_APRF_UIF.h          <H file: Environment variables for Audio Player Recorder
Framework (APRF)>
     |   ACTBP_capinput.c        <C file: CapacitanceTouch sensing and Gesture and Event
detection>
     |   ACTBP_capinput.h        <H file: CapacitanceTouch sensing and Gesture and Event
detection>
     |   ACTBP_uart.c            <C file: HW UART ISRs and UART APIs to communicate with DSP>
     |   ACTBP_uart.h            <H file: HW UART ISRs and UART APIs to communicate with DSP>
     |   ACTBP_timer.c           <C file: Timer ISRs and UART communication error timeout reset
mechanism>
     |   ACTBP_timer.h           <C file: Timer ISRs and UART communication error timeout reset
mechanism>
     |   ACTBP_filesys.c         <C file: Functions to send file system navigation commands to
APRF>
     |   ACTBP_filesys.h         <H file: Functions to send file system navigation commands to
APRF>
     |   ACTBP_display.c         <C file: Functions to send OLED display commands to APRF>
     |   ACTBP_display.h         <H file: Functions to send OLED display commands to APRF>
     |   ACTBP_test.c            <C file: Initial test code>
     |   README                  <This file>
     |
     +---[CapTouchLibrary]       <Capacitive Touch Library Code>
     |       Cap_Touch_HAL.c
     |       Cap_Touch_HAL.h
     |       Cap_Touch_Layer.c
     |       Cap_Touch_Layer.h
     |       structure.c         <Created using template_structure.*>
     |       structure.h         <Configured/calibrated specifically for this HW>
     |
     +---[CCS]                   <CCS Project Configuration Files>
     |                           <All listed files are required>
     |                           <The non-listed can be safely removed>
     |   .ccsproject
     |   .cdtbuild
     |   .cdtproject
     |   .project                <Contains links to portable project folders/files>
     |   lnk_msp430g2553.cmd
     |   macros.ini              <Enables portable project>
     |   MSP430G2553.ccxml
```

3.2.4 Import Project in CCS

To import the project into CCS:

1. Open CCS.
2. Select a new project workspace outside of the project folder. (The workspace should be in an independent folder, not containing or contained by the project or package folder).
3. Select *Project* → *Import Existing Project*.
4. Browse to the **[PROJECT_ROOT]\CCS** folder.
5. Make sure that *Copy projects into workspace* is not checked.
6. Click *Finish*.

NOTE: For CCS, while project root is in the outer directory, the CCS project files are located inside CCS. To enable the portability of the project, the *macros.ini* file is created to define the root. Additionally, all project code files (.c, .h) are added as linked resources with their relative path to the project root.

3.3 Capacitive Touch Sense Library

The Capacitive Touch Sense Library CAPSENSELIBRARY is a configurable tool to abstract the various peripheral settings from the application and perform several capacitive touch functions through API calls. The following describes the configuration of the library to support the Capacitive Touch Booster Pack, the methodology to calibrate the different elements, and how the API calls are used in the application.

- **Configuration**

  The first step in the configuration process is identifying the methodology used to measure the capacitance. For the Audio Capacitive Touch Booster Pack, the goal is to highlight the new PinOsc feature; therefore, an RO implementation is chosen, and the relaxation oscillator is implemented with the PinOsc. The RO implementation requires two timers (hardware or software timers): an interval timer (gate) and a frequency counter. The frequency counter is implemented with the Timer_A0 peripheral, and the interval timer is implemented with the WDT+.

  The capacitance sensing elements of the Audio Capacitive Touch Booster Pack is represented in the various structures defined in the *structure.c* file. The element structures define the general-purpose input/output (GPIO) and the performance characteristics of each element. The GPIOs are defined first, the appropriate sensor characteristics are defined, and the performance characteristics are measured and added.

  The sensor structure groups elements as appropriate and identifies the measurement characteristic for that group, namely the interval period. For the RO method, increasing the interval time increases the sensitivity; however, this is at the cost of response time, which is critical for supporting the Audio Player Recorder demo application.

  Unlike the original Capacitive Booster Pack User Experience **430BOOST-SENSE1 <missing link>** in which the proximity sensor and the button and wheel uses an SMCLK of 125 kHz and 1 MHz, respectively, the ACTBP host application uses an SMCLK of 1 MHz for both sensors. The interval count is 8192: 8.192-ms gate time for the button and wheel elements and the proximity element. The wheel is a special kind of sensor in which each element contributes to the sensor performance. The wheel is made up of four elements divided into 64 points or sections and requires that the cumulative response exceed 75 percent. This percentage is based upon the normalized response where meeting the threshold would represent 0% and the maximum response would represent 100%. This is to account for cases when the interaction is near the edges of the wheel instead of the middle.

- **Calibration**

  The calibration of the middle button and the proximity sensor are relatively straight forward, because the desired output is a binary indication of whether or not the threshold is exceeded. Using a controlled test fixture to represent the minimum touch (distance in the case of the proximity sensor), the values are recorded and input as the threshold value in the element structure.

  The calibration for the wheel is more complicated, as several measurements are required at various positions. For a detailed description, see the *Sensor Arrays: Wheels and Sliders* section in the *Capacitive Touch Sense Library Programmer's Guide* (SLAA490). The calibration values for each element are recorded in the element structure in the file structure.c.

4 ACTBP Host Application Architecture

Controlling DSP operations is realized through three main operations in the host application firmware (see Figure 4):

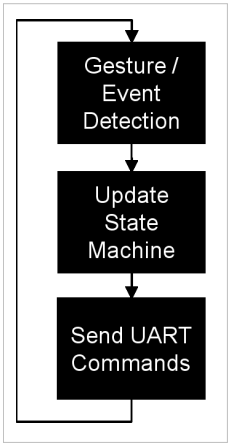

Figure 4. ACTBP Host Application Architecture

- **Gesture and Event Detection**

  The host application periodically scans the capacitance touch sensors to detect user inputs and identify the intended gestures such as scrolling direction, single and double taps and extended hold. These gestures are set to pre-defined events based on the location of these gestures.

- **State Machine and Context Update**

  The expected system behavior of Audio Player Recorder demo is captured in a state machine and the Current and Previous States are updated based on the detected event. The Current context of the Audio Player Recorder Framework is also maintained and updated.

- **UART Communication to the DSP**

  The required UART command or sequence of commands is sent to the DSP based on the Current State and context of the Audio Player Recorder Framework. These commands instruct the Framework to perform the required operations, which include navigating the file system on the SD Card, playing or recording MP3 files, updating the OLED display with appropriate texts or prompts and enabling and disabling USB mass storage connection to a host PC. Details of Audio Player Recorder Framework UART interface are documented on the Audio Capacitive Touch BoosterPack C5535 Software wiki page.

4.1 Gesture and Event Detection

The gesture detection algorithm is adapted from the original Capacitive Touch Booster Pack User Experience demo. The Capacitive Touch Library is used to configure, calibrate, and control the capacitive sensors. The sensor pads are configured in the *structure.h* and *structure.c* files. Specifically, six elements are divided into three sensors: one proximity sensor, one capacitive touch button (center button), and a four-element wheel. After each element is calibrated to determine the touch threshold as well as their maximum values in number of counts (that correlate to the measured capacitances), these calibration values are registered inside *structure.c*.

The *cap_touch_hal.** files provide the hardware abstract layer for the cap touch library, which essentially provides different hardware and peripheral options to drive the capacitive touch functionality. The Pin Oscillator peripheral in the MSP430G2553 MCU is used. The *cap_touch_layer.** files provide the capacitive touch layer, which offers functions that process the raw data into structured format. The main application utilizes this layer for their API calls to setup, measure baseline, detect touches, and detect wheel positions. For more information on the Capacitive Touch Library, see the *Capacitive Touch Sense Library Programmer's Guide* (SLAA490).

The application samples and registers individual finger touches on the 16-position wheel or the CENTER button as well as simple gestures [clockwise and counterclockwise] while the finger moves along and remains on the wheel. Upon wheel position detection, the corresponding LEDs surrounding the wheel light up accordingly. Each individual tap on the CENTER capacitive touch button toggles the center LED.

4.1.1 Proximity Sensor

The proximity sensor is made of the entire top PCB layer surface that covers the capacitive touch wheel. The proximity sensor detection occurs when the measured capacitance increases due to the presence of some conductive object within 1-2 inch from the surface. Generally, a hand wave motion in parallel and 1-2 inch from the ACTBP can trigger the detection.

4.1.2 Individual Touch on Capacitive Touch Sensors

The center button (small round button in the middle of the board) can register an individual touch or press. The capacitive touch wheel consists of four physical "ninja-star" elements arranged in a wheel formation that is calibrated and programed to provide 16-position detection. These positions can be detected individually as a button press. A press by the application's definition is constituted by a separate and single position detection registered continuously (for example, press begins when the position is touched and after no other position was detected, and ends when another position is detected or no further position is detected).

4.1.3 Wheel Gestures on Capacitive Touch Sensors

A gesture on the capacitive touch wheel is formed when a continuous series of touches is detected (no-detection is reported, for example, when the finger never leaves the wheel). A complete continuous finger motion on the wheel might consist of several intermediate gestures, each with one separate direction (clockwise or counterclockwise).

4.1.4 Defined Events

The host application has the following events as defined in the ACTBP_HostVars.h header file in the **[INSTALL_PATH]\Source** folder to reflect the allowed user touch input conditions on the capacitive touch sensors:

Table 2. Allowed User Touch Input Conditions

EVENT	DESCRIPTION	EVENT	DESCRIPTION
UP	Tap detected on UP button	RIGHT	Tap detected on RIGHT button
DOWN	Tap detected on DOWN button	LEFT	Tap detected on LEFT button
MID	Tap detected on CENTER button	MID_HOLD	Hold (extended-touch) detected on CENTER button
UP_HOLD	Hold (extended-touch) detected on UP button	RIGHT_HOLD	Hold (extended-touch) detected on RIGHT button
DOWN_HOLD	Hold (extended-touch) detected on DOWN button	LEFT_HOLD	Hold (extended-touch) detected on LEFT button
DOWNx2	Double-tap detected on DOWN button	LEFTx2	Double-tap detected on LEFT button
SCROLL_RIGHT	Clockwise scroll wheel gesture detected	SCROLL_LEFT	Counterclockwise scroll wheel gesture detected
HOLD_END	No activity detected after Hold Event indicating end of the Hold Event		

4.1.5 Other Events Used in Gesture Detection

A few other events are defined to indicate start or end of detected HOLD events:

Table 3. Other Events Used in Gesture Detection

EVENT	DESCRIPTION	EVENT	DESCRIPTION
THOLD_ON	UP, RIGHT, DOWN or LEFT touch hold event detected	MHOLD_ON	CENTER button touch hold event detected
SCROLL_ON	Scrolling detected	NO_HOLD	End of detected hold event

4.2 State Machine and Context Update

Once an event has been detected, the host application updates a control state machine if the event is a valid user input in the current state as described in Section 2. As defined in the ACTBP_HostVars.h header file in the **[INSTALL_PATH]\Source** folder, the state machine has the following states:

Table 4. State Machine and Context Update

STATE	DESCRIPTION	STATE	DESCRIPTION
BROWSE_DIR	File system navigation to browse directories in root and select directory	BROWSE_FILE	File system navigation to browse and select MP3 files for playback in selected directory
NXT_DIR	Display next directory	NXT_FILE	Display next file in selected directory
PRV_DIR	Display previous directory	PRV_FILE	Display previous file in selected directory
PLAY	Active playback state	PAUSE	Playback pause
FF	Fast forward operation on file currently playing	RWD	Rewind operation on file currently playing
SKIP_F	Skip to next file in play order	SKIP_B	Skip to previous file in play order
SKIP_START	Skip to start of currently playing file	SHUFFLE	Enable/disable shuffle mode
VOL_UP	Volume up operation	VOL_DOWN	Volume down operation
RECORD	Active record state	REC_PAUSE	Record pause
USB_MSC	USB mass storage class enabled	PWR_OFF	Power off state with DSP powered off and MSP430 MCU in LPM3 state

The host application uses the following structure to track the current context of the host application and Audio Player Recorder Framework:

```
<* Structure to store working context */
 struct ContextABP
 {
    enum event current_event;          <* Latest detected event                                */
    enum event_hold last_event;        <* Hold event detected                                  */
    enum state current_state;          <* Current state of state machine                       */
    enum state last_state;             <* Previous state                                       */
    int curr_dir_num;                  <* Directory number of currently selected directory     */
    int curr_file_num;                 <* File number of currently active file                 */

    int curr_dir_count;                <* Total number of directories in currently selected
                                          folder                                              */
    int root_dir_count;                <* Total number of directories in root folder           */
    int curr_file_count;               <* Total number of files in currently selected folder   */
    unsigned char curr_volume;         <* Current volume level                                 */
    unsigned char curr_play_status;    <* Current playback status                              */
    unsigned char curr_record_status;  <* Current record status                                */
    unsigned shuffle_status:1;         <* Current shuffle status                               */
    unsigned cycle_status:1;           <* Current cycle status                                 */
    unsigned USB_cableinsert:1;        <* Current status USB connection to host computer       */
    unsigned disable_command:1;        <* Flag to disable state machine update on detected
                                          event                                               */
 };
```

4.3 UART Communication to the DSP

The host application uses the hardware UART on the MSP430G2553 MCU to communicate with the Audio Player Recorder Framework on the C5535 DSP. Based on the detected user input event, the state machine is updated and the appropriate UART command or sequence of commands is sent to the DSP to achieve the desired system response. The Virtual Registers of the Audio Player Recorder Framework can be written to or read from using scalar (6-byte packet) or array (custom-defined length) UART packets as documented in the Audio Capacitive Touch BoosterPack C5535 Software wiki page.

4.3.1 UART API

Three UART API have been implemented in the host application based on the general UART APIs recommended in C5535 Software documentation wiki. Given the limited on-chip RAM (512 Bytes) on the MSP430G2553 MCU, the API for array reads has not been implemented. An acknowledge (ACK) packet is sent by the DSP for every UART command by the host application. The host application implements a fixed 23-second timeout to receive the ACK. If the ACK is not received in time, the host application flashes the LEDS on the ACTBP and power cycles the DSP. This section documents the UART APIs implemented for the host application.

Title **Page**

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(unsigned int virtualRegisterAddr, unsigned long value,
unsigned int ignore_ack);
```

Arguments

IN	unsigned int	virtualRegisterAddr
	Virtual register name like play_file,time_out, dir_info, and so forth	
	unsigned long	value
	Value to Write	
	unsigned int	ignore_ack
	Flag to ignore ACK	
OUT	None	

Return Value

Status	ACK Error Code

Comments None

Constraints 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(unsigned int virtualRegisterAddr, unsigned int length,
char* arrayToWrite);
```

Arguments

IN	unsigned int	virtualRegisterAddr
	Virtual register name like play_file,time_out, dir_info, and so forth	
	unsigned int	length
	Length of array to write	
	CHAR*	arrayToWrite
	Array to write	
OUT	None	

Return Value

Status	ACK Error Code

Comments None

Constraints Array size is limited to 21 characters.

uif_readScalar

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

Syntax

```
Status uif_readScalar(unsigned int virtualRegisterAddr, unsigned long *value);
```

Arguments

IN	unsigned int	virtualRegisterAddr
	Virtual register name like play_file,time_out, dir_info, and so forth	
OUT	unsigned long*	value
	Read-back value	

Return Value

Status	ACK Error Code

Comments None

Constraints None

5 MSP430 Capacitive Touch Library

- titool:430boost-sense1
- http://www.ti.com/tool/430boost-sense1
- Capacitive Touch Library (CAPSENSELIBRARY)

6 Developing on the MSP430 LaunchPad

- Requirements:
  - CCSv4—download here: {{#tiwikiurl:Download_CCS|Download CCS}}
- Jumper settings:
  - J4—VCC, TEST, and RESET need shorting blocks to program the MSP430 through USB FET on LaunchPad
  - Connect USB to MSP430 USB port
    - If connecting both USBs, remove JP3 from the BoosterPack board
    - Power concern during programming? Need to remove JP3 regardless?

7 FAQ

Q: What MSP430 Value Line products can support this software?

A: The ACTBP ships with the MSP430G2553IN20 device, which has 16KB Flash and 512Byte RAM. The current ACTBP LaunchPad Host App requires xxKB and yy Bytes RAM. The G2xx3 series of the MSP430 Value Line includes a hardware UART on pins X.X and X.Y. UART communication between MSP430 and C5535 requires UART_TX on X.X and UART_RX on X.Y (or refer to J1 and J2 header locations).

Q: How does this capacitive touch software differ from the MSP430 Capacitive Touch BoosterPack?

LED mapping is different. Only one LED lights up at a time instead of two. Hardware UART is used to communicate to the C5535 DSP on the ACTBP board. Both software versions use the same library (available here).

8 References

Capacitive Touch Sense Library Programmer's Guide (SLAA490)

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