# Media Player

# Software Design Description

| | |
|---|---|
| **Name** | Media Player |
| **Document Creation Date** | July 23, 2008 |

## Revision Sheet

| Revision Number | Date | Description |
|---|---|---|
| **1.0** | **July 23, 2008** | Initial revision |

## Table of Contents

# 1. Introduction

The Media Player (MP) is an application that utilises several of ADI's standard software modules for decoding and playing audio and image formats including MP3, AAC, WAV and JPEG and BMP. The MP is one of the many applications included in Analog Devices Software Development Kit (SDK). It uses Micrium's uC/OS-II as a real time operating system and takes advantage of ADI's device drivers and system systems to control internal devices and external peripherals onboard the ADSP-BF548 EZ-KIT and ADSP-BF527 EZ-KIT.

## *1.1. Definitions, Acronyms and Abbreviations*

| **Acronyms** | **Definition** |
|---|---|
| AAC | Advanced Audio Coding |
| ADI | Analog Devices Inc |
| BMP | Bit Map |
| DMA | Direct Memory Access |
| HEAAC | High Efficiency Advanced Audio Coding |
| JPEG | Joint Photographic Experts Group |
| LCD | Liquid Crystal Display |
| MP3 | Moving Picture Experts Group Layer-3 Audio |
| MP | Media Player |
| MPEG | Motion Pictures Expert Group |
| OS | Operating System |
| RGB | Red Green Blue |
| SDK | Software Development Kit |
| SPI | Serial Peripheral Interface |
| SPORT | Serial Port |
| WAV | Windows Wave |
| YUV | Luminance-Bandwidth-Chrominance |

## *1.2. References*

All reference material related to the decode modules used in the MP are available in the SDK document directory.

| **Document** | **Name / Location** |
|---|---|
| MP3 Decoder Developer's Guide | DEVMP3-001_MP3 Decoder Developers Guide.pdf |
| MP3 Product Specification Sheet | ADA-171_MP3_Decoder_Spec_Sheet.pdf |
| ID3 TAG Parser Developer's Guide | DEVMP3-002_ID3_Tag_Parser_Developers_Guide.pdf |
| AAC Decoder Developer's Guide | KT-41 AAC Decoder Developers Guide.pdf |
| MPEG-4 HE-AAC v2 Stereo | KT-312_MPEG-4_HE-AAC- |

| Document | Name / Location |
|---|---|
| Decoder Product Specification Sheet | v2_Dec_BF_ProdSpec.pdf |
| MPEG-4 HE-AAC V2 Stereo Decoder (with DAB support) For Blackfin Release Notes | KT-336_MPEG-4_HE-AAC-v2_Dec_BF_Release_Notes.pdf |
| MP4 Parser Developer's Guide | KT-195_MP4_Parser_Developers_Guide.pdf |
| JPEG Decoder Library Developer's Guide | DEVIMG-001-E JPEG Decoder Library Developer's Guide.doc |
| JPEG System IO Buffer Modules Supplementary Developer's Guide | DEVIM1-001-B JPEG System IO Buffer Modules Supplementary Developer's Guide.doc |

# 2. Design Overview

## *2.1. System Objectives*

The objective of the SDK and MP is to provide customers with applications that use ADI's Device Drivers, System Services and code modules. From such applications, customers can learn about Analog's Device Drivers, System Services and code modules and use them as references for application specific software design.

## *2.2. Operating Environment*

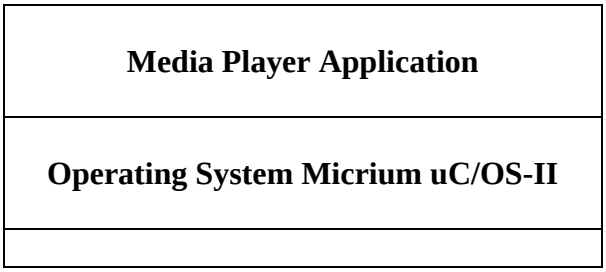
The MP was written using Analog Devices' VDSP 5.0 Update 3. The target hardware platforms are the ADSP-BF548 EZ-KIT and the ADSP-BF527 EZ-KIT. The MP uses Micrium uC/OS-II Kernel Blackfin port release 1.1.

# 3. System Architecture

## *3.1. Architecture Overview*

The MP architecture includes the following components

- Operating System
- Platform Managers
- Control Devices
- Decode Modules



| MP PLATFORM MANAGERS | |
|---|---|
| Application Manager<br>File System Manager<br>Display Manager<br>Player Manager | |
| **DECODERS** | **DEVICES** |
| MP3 (audio)<br>AAC (audio)<br>WAV (audio)<br>JPEG (video)<br>BMP (video) | File System<br>LCD<br>Keypad<br>Rotary Thumbwheel<br>UART<br>Audio Codec |

## *3.2. User Interface*

The MP utilises several internal and external peripherals for its user interface. These include the;

- Rotary thumbwheel
- Keypad
- LCD Display

### 3.2.1. Rotary Thumbwheel

The MP uses the rotary thumbwheel to control audio codec volume levels as well as play/stop control in play mode. In idle mode it also controls file selection. The following table defines the thumbwheel's operation

| Rotary / Thumbwheel | Operation |
|---|---|
| Clockwise | Up Volume (play mode) |
| Anticlockwise | Down Volume (play mode) |
| Press | Play / Pause (play mode)<br>File Selection (idle mode) |

### 3.2.2. Keypad

The MP uses a 4 x 4 matrix keypad. The following table defines the keypads operation

| Keypad Button | Operation |
|---|---|
| 2 | Play/Pause (play mode)<br>File Selection (idle mode) |

| | |
|---|---|
| 3 | Stop (play mode) |
| ↑ | Previous file (idle mode) |
| ↓ | Next file (idle mode) |
| 0 | Down Volume (play mode) |
| HELP | Up Volume (play mode) |
| 2ND | Terminate application |

### 3.2.3. LCD Display

The LCD is 24 bit and has a resolution of 480 x 272 pixels on the ADSP-BF548 EZ-KIT and 320 x 240 on the ADSP-BF527 EZ-KIT.

The LCD is divided into three graphical fields

- Title field
- File listing field
- File tagging field
- Status Information field


# 4. Detailed Design

There are several components types used in the MP design including

- Operating system
- Managers
- Device drivers and system services
- Device controls and
- Decoder libraries

The MP uses Micrium's uC/OS-II as a real time operating system.

The MP uses fours managers that act as the interface between software components including devices drivers, device controls and decoder libraries. The managers are called the application manager, display manager, player manager, and the file system manager. Managers typically contain threads that direct event messages to / from the other software components of the MP.

The device drivers and system services used by the MP are those provided by Analog Devices' VisualDSP++ 5.0. The services cover a collection of functions commonly found in embedded systems covering services such as, Direct Memory Access, Interrupt control, Power management, Flag control, and Timer control.

The VisualDSP devices drivers provide a standardised method of configuring, controlling and communicating with a wide range of internal and external peripherals. The device drivers and system services will not be detailed in this document. Reference material can be found in the VisualDSP++ 5.0 tool suite.

The device control components form the interface between the MP's devices drivers and the MP's managers. The device controls are typically associated with specific device drivers and are responsible for configuration, closure, callbacks, and the sending or receiving of event messages to the managers.

The decoder libraries are responsible for decoding media from a number of different audio and video formats. The decoder libraries used are part of Analog Devices software modules and included

- MP3 Blackfin Decoder, MP3DE-00-00, 13 March 2007, 2.06
- JPEG Blackfin Decoder, JPGDE-00-00, 6 December 2007, 3.08
- HEAAC Blackfin Decoder, AACDE-03-00, 9 October 2007, 3.00a
- WAV decoder, not an official release
- BMP decoder, not an official release

## *4.1. Operating System*

### 4.1.1. Type – Operating System

### 4.1.2. Purpose

The purpose of the OS is to create and manage the threads of the MP's software components.

### 4.1.3. Function

For this application the main function of the OS is to provide a means of posting and receiving event messages between managers, device controls and decoders.

### 4.1.4. Interfaces

Refer to reference documents.

### 4.1.5. Resources

Refer to reference documents.

## *4.2. Application Manager*

### 4.2.1. Type - Manager

### 4.2.2. Purpose

The application manager has two main purposes, to control the MP state machine, and to direct user event messages to the appropriate manager.

### 4.2.3. Function

The MP state machine has three states

- state_play
- state_idle
- state_pause

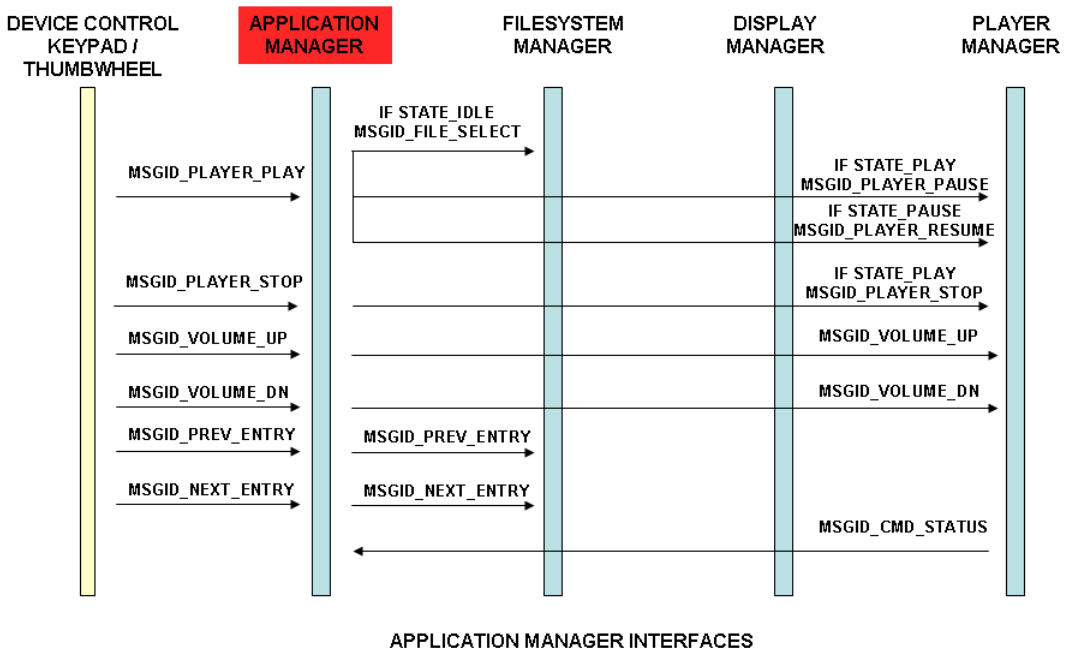
The state is determined by the MP's operational mode and by messages received from the player manager. The operation modes are

- mode_audio
- mode_image

The messages received by the application manager typically originate from a callback from one of the device control software components for example the keypad or thumbwheel. The application manager also directs these messages to the other managers for processing.

### 4.2.4. Interfaces

The application manager interfaces to other managers and device controls via the following messages.




APPLICATION MANAGER INTERFACES

From the interface diagram, it can be seen that for the most part, all messages to the application manager are user event driven and originate from either the keypad or thumbwheel device control. They are such to select/navigate files, start/stop/pause decoding, and control volume levels. The only exception is the MSGID_CMD_STATUS. This message is initiated by the player manager and carries decoder status parameters such as START_DECODE_AUDIO, START_DECODE_VIDEO, STOP_DECODE. These parameters are sent to the application manager and used to control the MP's state machine.

## *4.3. Display Manager*

### 4.3.1. Type - Manager

### 4.3.2. Purpose

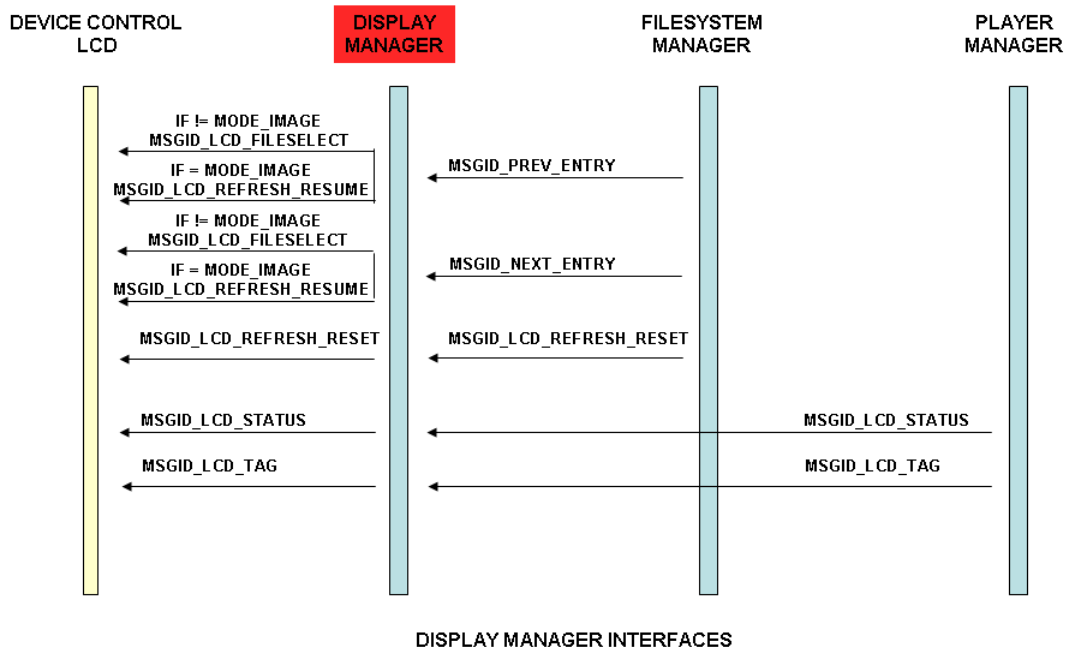
The display manager is the interface between the MP's managers and the LCD control device.

### 4.3.3. Function

The display manager receives messages from other managers and posts them to the LCD control device. It provides graphical controls that are used to display information in each of the LCD fields. Each graphical field is illustrated in the User Interface.

### 4.3.4. Interfaces

The display manager interfaces to other managers and device controls via the following messages.




DISPLAY MANAGER INTERFACES

## *4.4. File System Manager*

### 4.4.1. Type - Manager

### 4.4.2. Purpose

The purpose of the file system manager is to act as an interface between the file system services and the MP's other managers.

### 4.4.3. Function

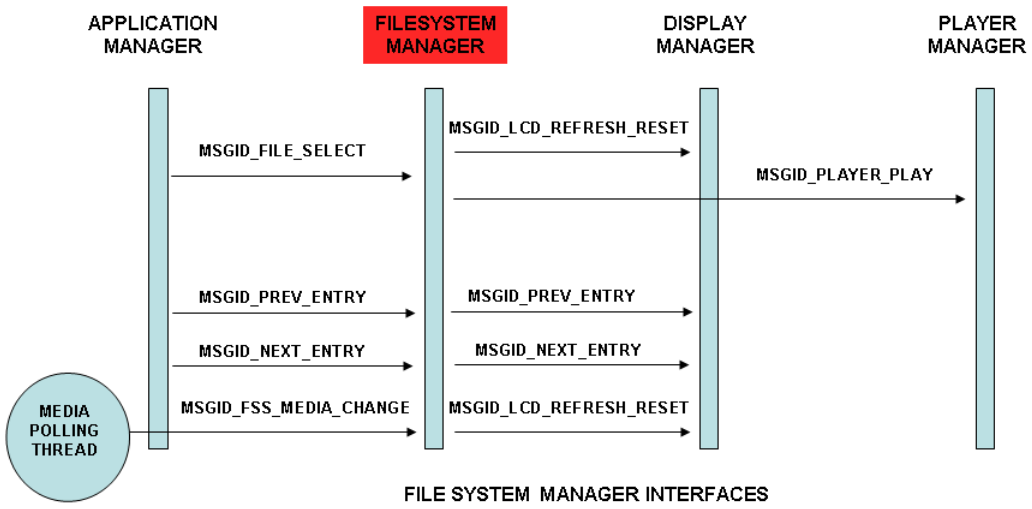
The function of the file system manager is to read the contents of a directory listing and determine whether each entity in the listing is a drive volume, a subdirectory or file. The file system manager provides the capability to navigate through a directory tree via an indexing scheme. The index is used select from a file list and enables the manager to control a graphical scrolling on the LCD display.

The file system manager contains two threads. The primary thread forms the messaging interface to other managers. The secondary thread periodically polls

the file system service so that it can detect if any removable media has been inserted or unplugged from the MP.

### 4.4.4. Interfaces

The file system manager interfaces to the other managers via the following messages.




FILE SYSTEM MANAGER INTERFACES

The media polling thread is a task that periodically polls the file system service via the adi_fss_PollMedia function. This function returns the number of volumes connected to the media player. If the number of volumes changes between consecutive polls, the thread deems that a drive has either been inserted or removed. Upon such an event, a MSID_FSS_MEDIA_CHANGE message is sent to the filesystem manager which then re-logs the drives and updates the LCD directory listing via the display manager.

## *4.5. Player Manager*

### 4.5.1. Type - Manager

### 4.5.2. Purpose

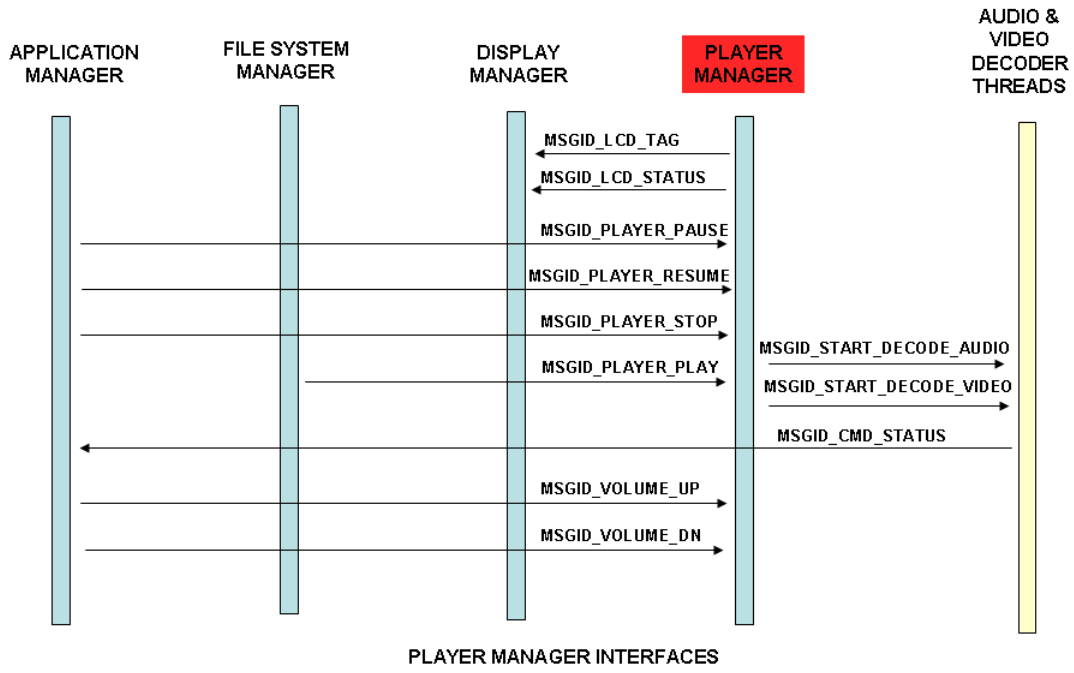
The player manager acts as the interface between the decoder modules (libraries) and other MP managers.

### 4.5.3. Function

The player manager performs several functions including, media decoding control, volume control, posting decode status messages, posting file tag status messages and posting state machine commands.

### 4.5.4. Interfaces

The player manager interfaces to other managers and media decode modules via the following messages.




PLAYER MANAGER INTERFACES

## *4.6. Device Drivers and System Services*

### 4.6.1. Type – Device Drivers and System Services

### 4.6.2. Purpose

All device drivers and system services are documented as part of the Visual DSP release.

### 4.6.3. Function

Refer to VDSP reference documents.

### 4.6.4. Interfaces

Refer to VDSP reference documents.

## *4.7. Rotary Thumbwheel*

### 4.7.1. Type – Device Control

### 4.7.2. Purpose

The purpose of the thumbwheel device control (thumbwheel.c) is to initialise, close and handle the callbacks of the thumbwheel device driver (adi_cnt.c).

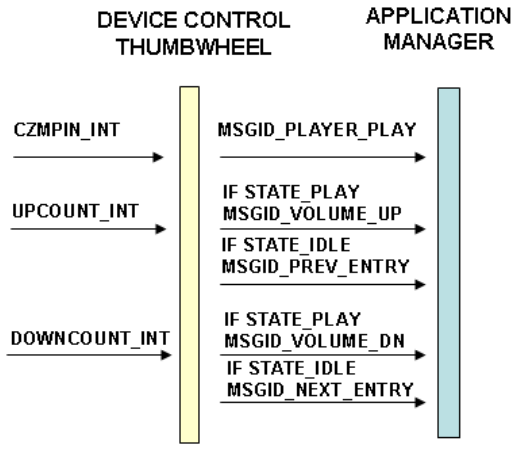
### 4.7.3. Function

There are three functions in the thumbwheel device control; init_thumbwheel(), close_thumbwheel(), and callback_thumbwheel().

init_thumbwheel() and close_thumbwheel() initialise and close the associated device driver. callback_thumbwheel() responds to the callbacks generated by the thumbwheel and post messages to the application depending of the state machine of the MP. A counter up or counter down callback generates a volume up and volume down message to the application manager if the MP is in STATE_PLAY. A counter up or counter down callback generates a next file entry or previous file entry if the MP is in STATE_IDLE. A counter zero callback generates a play message to the application manager.

### 4.7.4. Interfaces

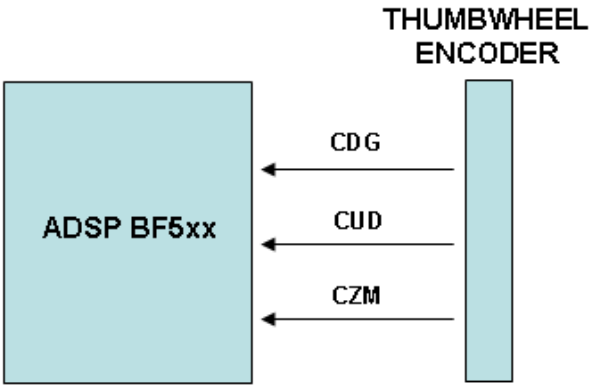
The thumbwheel device control interfaces to thumbwheel device driver callbacks and the application manager via the following messages.




### 4.7.5. Resources

The thumbwheel device control interfaces to an internal peripheral that uses the following resources

- 1 Interrupt
- 3 port flags for counter up, counter down, and counter zero control.




Counter down uses port flag pins PF12 and PH3 on the ADSP-BF527 EZ-KIT and ADSP-BF548 EZ-KIT respectively, counter up uses port flag pins PF13 and PH4 on the ADSP-BF527 EZ-KIT and ADSP-BF548 EZ-KIT respectively, and zero counter uses port flag pins PF11 and PG3 on the ADSP-BF527 EZ-KIT and ADSP-BF548 EZ-KIT respectively.

The interrupt is internal and defined as ADI_INT_COUNTER.

## *4.8. Keypad*

### 4.8.1. Type – Device Control

### 4.8.2. Purpose

The purpose of the keypad device control ( keypad_max1233.c for ADSP-BF527 EZ-KIT and keypad_548.c for ADSP-BF548 EZ-KIT) is to initialise, close, and handle the callbacks of the keypad device driver (adi_max1233.c for the ADSP-ADSP-BF527 EZ-KIT) and (adi_keypad.c for the ADSP-BF548 EZ-KIT). It also posts messages to the application manager via a thread.

### 4.8.3. Function

Due to different keypad hardware implementations, the functionalities of the keypad device control are slightly different between the ADSP-BF527 EZ-KIT and ADSP-BF548 EZ-KIT.

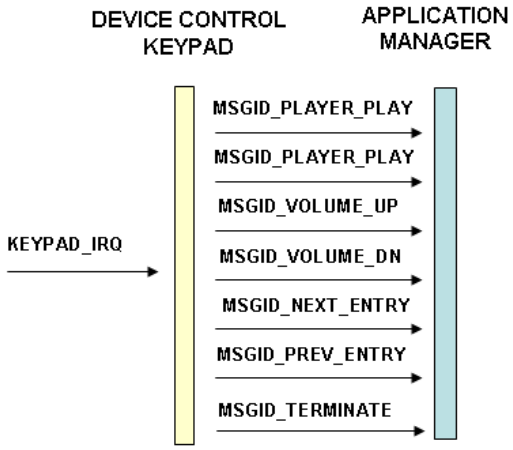
**ADSP-BF548 EZ-KIT**

There are three functions in the keypad_548 device control; init_keypad_548(), close_keypad_548(), and callback_keypad(). init_keypad_548() and close_keypad_548(), initialise and close the associated device driver. callback_keypad() handles the callbacks and posts a messages containing keypad row and column data to a keypad thread. The keypad thread decodes the row and column data and posts keypad event data to the application manager.

**ADSP-BF527 EZ-KIT**
There are three functions in the keypad_548 device control; init_keypad_ MAX1233(), close_keypad_ MAX1233(), and callback_keypad(). init_keypad_ MAX1233() and close_keypad_ MAX1233(), initialise and close the associated device driver. callback_keypad() handles the callbacks and posts a messages containing the decoded keypad data to a keypad thread. The keypad thread posts keypad event data to the application manager.

### 4.8.4. Interfaces

The keypad device control interfaces to keypad device driver callbacks and the application manager via the following messages.

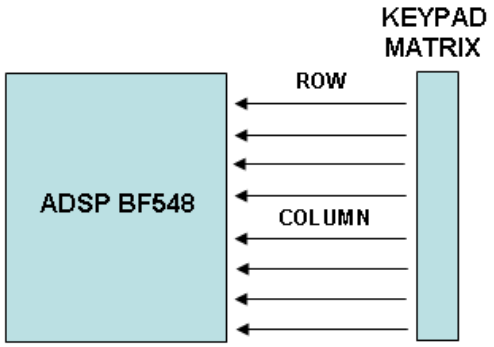



### 4.8.5. Resources

**ADSP-BF548 EZ-KIT**
keypad_548.c configures the keypad device driver on the ADSP-BF548 EZ-KIT. The driver controls interfaces to an internal peripheral that uses the following resources

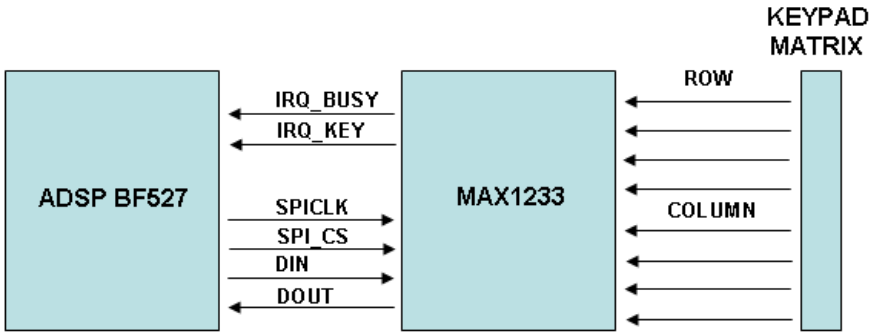
- 1 Interrupt
- 8 port flags including 4 column and 4 row signals

The 8 port flags of the keypad matrix interface to the EPPI1 pins 15 through to 8. The interrupt is internal and defined as ADI_INT_KEYPAD.




**ADSP-BF527 EZ-KIT**
keypad_max1233.c configures the keypad device driver on the ADSP-BF527 EZ-KIT. The driver controls interfaces to an external peripheral called MAX1233 and uses the following resources

- 2 Interrupt
- 1 SPI




The SPI port is configured with SPI_CS #4 together with two interrupts IRQ_BUSY and IRQ_KEY on port pins ADI_FLAG_PG0 and ADI_FLAG_PF9 respectively. IRQ_KEY is used to detect a keypad activity event and IRQ_BUSY is used by the driver to indicate completion of keypad decode scan.

## *4.9. LCD*

### 4.9.1. Type – Device Control

### 4.9.2. Purpose

The purpose of the LCD device control is to initialise, close, and handle the callbacks of the LCD device driver (adi_t350mcqb01.c for the ADSP-BF527 EZ-KIT and adi_lq043t1dg01.c for the ADSP-BF548 EZ-KIT).

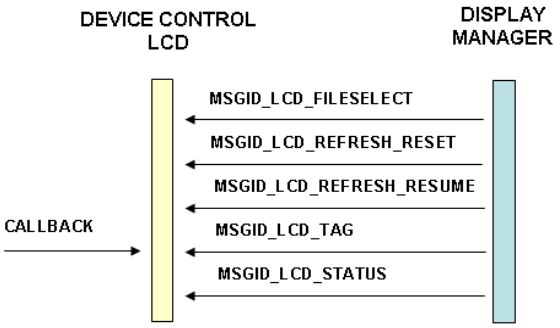
A thread within the LCD device control responds to incoming messages from the display manager and updates the LCD output buffer whenever the contents of the LCD screen needs to be changed.

### 4.9.3. Function

Due to different hardware implementations, the functionalities of the LCD device control are slightly different between the ADSP-BF527 EZ-KIT and ADSP-BF548 EZ-KIT. The LCD on the ADSP-BF548 EZ-KIT uses a 24 bit enhanced PPI whereas on the ADSP-BF527 EZ-KIT an 8 bit PPI is used.
There are three main functions in both versions of the LCD device control; init_lcd_xxx(), close_ lcd_xxx(), and callback_lcd(). init_lcd_xxx() and close_lcd_xxx(), initialise and close the associated device driver. callback_lcd() handles the callbacks and updates the LCD's DMA buffer chain. A thread in the LCD device control receives messages from the display manager that are used to update the LCD output buffer.

### 4.9.4. Interfaces

The LCD device control interfaces to the LCD device driver and to the display manager via the following messages.

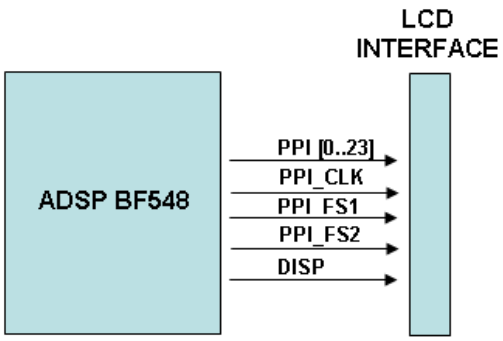



### 4.9.5. Resources

**ADSP-BF548 EZ-KIT**

lcd_nl6448.c configures the LCD device driver with the following resources

- 1 ePPI
- 1 DMA channel
- 1 port flag to control LCD_DISP
- 1 timer

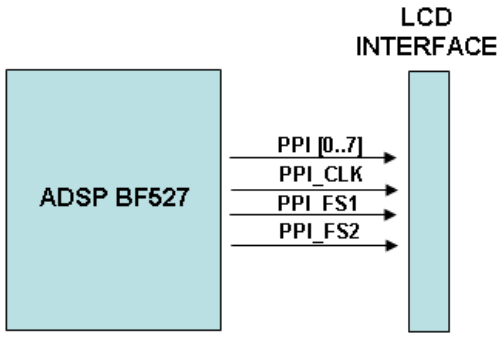



This LCD uses the enhanced PPI port on the ADSP-BF548 EZ-KIT. It is 24 bits wide and connects to PPI0[D0..D17] and PPI1[D0..D5]. A timer is used to generate the LCD frame synchronization signals #1 and #2. A 32 bit DMA data transfer is used to between the PPI data buffer and the LCD output buffer. Port flag pin PE3 is used to control the LCD_DISP signal.

**ADSP-BF527 EZ-KIT**

lcd_t350mcqb01.c configures the LCD device driver with the following resources

- 1 PPI
- 1 DMA channel
- 1 timer




The PPI port is 8 bits wide and supports 24 bit RGB. A timer is used to generate the LCD frame synchronization signals #1 and #2. A 16 bit DMA data transfer is used to between the PPI data buffer and the LCD output buffer.

## *4.10. CODEC*

### 4.10.1. Type – Device Control

### 4.10.2. Purpose

The purpose of the CODEC device control is to initialise, close, and handle the callbacks of the CODEC device driver (adi_bf52xc1.c for the ADSP-BF527 EZ-KIT and adi_ac97.c for the ADSP-BF548 EZ-KIT).

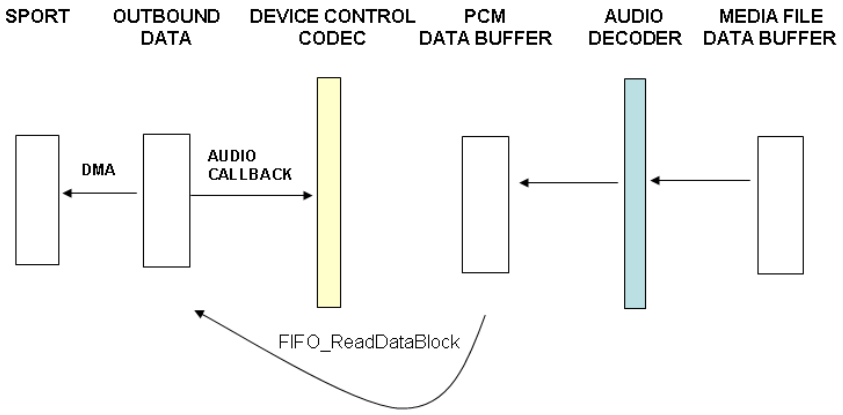
The CODEC device control also regulates volume levels and DMA data flow between the CODEC output buffer and SPORT interface.

### 4.10.3. Function

There are three main functions in both implementations (ADSP-BF548 EZ-KIT and ADSP-BF527 EZ_KIT) of the CODEC device control; init_codec_xxx(), close_codec_xxx(), and AudioCodecCallback(). init_codec_xxx() and close_codec_xxx(), initialise and close the associated device driver. AudioCodecCallback () handles the driver callback and initiates an audio data buffer read.

### 4.10.4. Interfaces

The main interface of the CODEC device control is a DMA generated audio callback that signifies that the outbound data buffer has been processed by the SPORT. Upon receiving this callback the CODEC device control initiates a FIFO buffer read from the PCM data buffer which contains decoded audio.

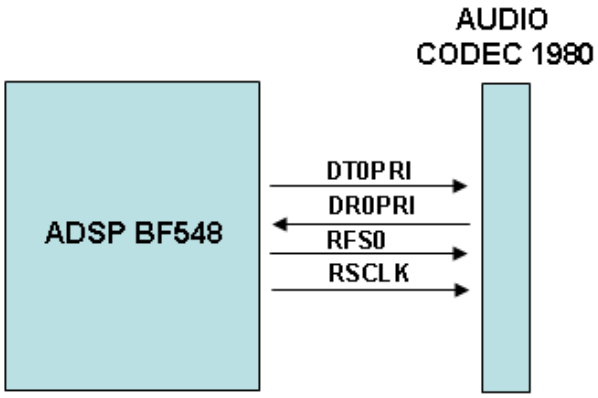



### 4.10.5. Resources

**ADSP-BF548 EZ-KIT**

codec_1980.c configures the codec device driver with the following resources

- 1 SPORT
- 2 DMA channel
- 5 internal interrupts (sport peripheral interrupt, DMA Tx data interrupt, DMA Tx error interrupt, DMA Rx data interrupt, and DMA Rx error interrupt)

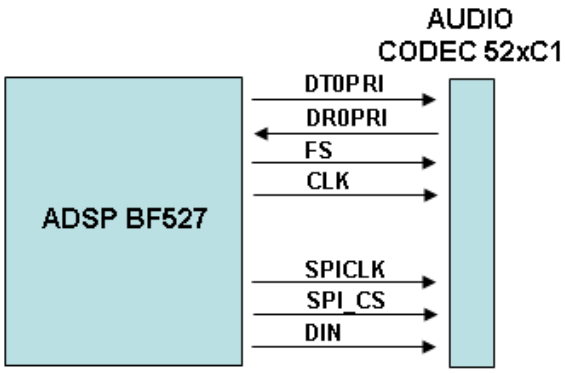



The SPORT is used for both control and data flow on the 1980 codec. It uses two DMA channels one for transmit and the other for receive, although the MP only uses the transmit channel. Five internal interrupts are needed for the 1980 codec.

**ADSP-BF527 EZ-KIT**

codec_52xc1.c configures the codec device driver with the following resources

- 1 SPORT
- 1 DMA channel
- 1 SPI
- 4 internal interrupts (sport peripheral interrupt, DMA Tx data interrupt, DMA Tx error interrupt, and SPI interrupt)




The codec on the ADSP-BF527 EZ-KIT uses the SPORT for data flow and the SPI for control. It uses one DMA channel for transmit, the receive channel is not configured and unused. Four internal interrupts are needed for the 52xC1 codec.

## 4.11. Audio Decoders

### 4.11.1. Type – MP3, AAC, WAV

| Code Module Name | Code Module | Version / Date |
|---|---|---|
| MP3 Decoder Blackfin | MP3DE-00-00 | 2.06, 13th March 2007 |
| HEAACv2 Stereo Decoder | AACDE-005-00 | 3.00a, 9th October 2007 |
| WAV Decoder | Not a controlled release | Not a controlled release |

### 4.11.2. Purpose

The purpose of the audio decoders is to decode audio file bit streams. The MP supports three audio media formats; MP3, AAC and WAV. Both MP3 and AAC decoders require specific code algorithms to decompress the bit streams. Their API's are specified Developers Guides referenced in Section 1.2 of this document.

### 4.11.3. Function

MP3 audio decoder supports MP3 bit streams as specified in ISO/IEC 11172- 3 and ISO/IEC 13818-3. The output format of the MP3 decoder is 16-bit PCM stereo where the left/right PCM channels are interleaved.

The AAC audio decoder supports the following bit streams; MPEG-2 AAC-LC profile (ISO/IEC 13818-7), MPEG-4 AAC-LC audio object (ISO/IEC 14496-3:2005), MPEG-4 SBR audio object (ISO/IEC 14496-3:2005), MPEG-4 PS audio object (ISO/IEC 14496-3:2005). The output format of the AAC decoder is stereo 16-bit PCM with the output buffer format been left/right interleaved.

### 4.11.4. Interfaces

Refer to section 1.2 of this document for the AAC and MP3 reference documents.

### 4.11.5. Resources

Refer to section 1.2 of this document for the AAC and MP3 reference documents.

## 4.12. Video Decoder

### 4.12.1. Type – JPEG, BMP

| Code Module Name | Code Module | Version / Date |
|---|---|---|
| JPEG Decoder Blackfin | JPGDE-00-00 | 3.08, 6th December 2007 |
| BMP Decoder | Not a controlled release | Not a controlled release |

### 4.12.2. Purpose

The purpose of the video decoders is to decode video file bit streams. In the case of the MP, two video media formats are supported; JPEG and Bit Maps. The JPEG decoder requires a specific code algorithm to decompress the bit stream. Their API's are specified Developers Guides referenced in Section 1.2 of this document.

### 4.12.3. Function

The JPEG decoder library used in the MP is configured to decode still images encoded in YUV 420 format. The output image is a YUV 420 planar format.

### 4.12.4. Interfaces

Refer to section 1.2 of this document for the JPEG reference documents.

.

### 4.12.5. Resources

Refer to section 1.2 of this document for the JPEG reference documents.

## *4.13. Video Post Processor*

### 4.13.1. Type – Post Processing

### 4.13.2. Purpose

The video post processor acts as the interface between the video decoder output and the input to the LCD data buffer. The purpose of the video post processing module is to manipulate decoded video outputs to a format that is compatible with the target hardware platform.

### 4.13.3. Function

The output video format required on both the ADSP-BF527 EZ-KIT and ADSP-BF548 EZ-KIT is RGB 888. Only the pixel resolutions differ 480 x 272 for the ADSP-BF548 EZ-KIT and 320 x 240 for the ADSP-BF527 EZ-KIT.

**ADSP-BF548 EZ-KIT**

The video post processing required on the ADSP-BF548 EZ-KIT has three stages

- Decoded video buffer in YUV 420
- Conversion to YUV 422
- Conversion of YUV 422 to RGB 888 via the ADSP-BF548 EZ-KIT pixel compositor

**ADSP-BF527 EZ-KIT**

The video post processing required on the ADSP-BF527 EZ-KIT has two stages

- Decoded video buffer in YUV 420
- Conversion of YUV 420 to RGB 888