Audio/Video Configuration Core for DE-Series Boards

For Quartus II 11.0

1 Core Overview

The audio and video peripherals on the DE-series boards, the 1.3 megapixel digital camera (TRDB_DC2), the 5 megapixel digital camera (TRDB_D5M) and the LCD with touchscreen (TRDB_LTM) require configuration via 2-wire and 3-wire serial buses. The Audio/Video Configuration Core provides a convenient way for configuring and initializing these devices.

2 Functional Description

The Audio/Video Configuration IP core is used to set up the audio and video peripherals on the DE-series boards. To set up a peripheral device, the core takes information from a program (via the Avalon bus) or the auto-initialization circuit and sends it our via a serial bus. The block diagram of the core is shown in Figure 1.

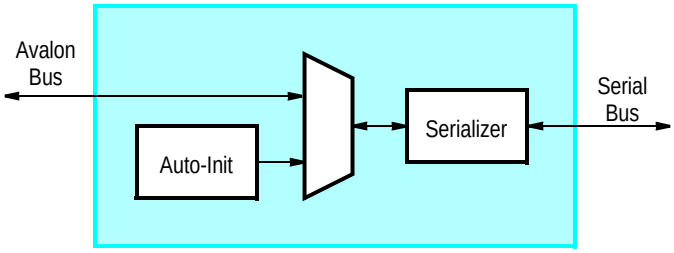

Figure 1. High-level block diagram of the Audio/Video Configuration IP core.

The Audio/Video Configuration should be used to configure the on-board audio and video chips, as well as the 1.3 megapixel digital camera (TRDB_DC2), the 5 megapixel digital camera (TRDB_D5M) and LCD with touchscreen (TRDB_LTM) daughercards. The LCD with touchscreen is configured using a 3-wire serial bus. The other peripherals are configured using 2-wire buses. The on-board audio and video chip share a physical 2-wire serial bus, as shown in Figure 2, while the two daughtercards have separate serial buses. The Audio/Video Configuration core must be instantiated once for on-board peripherals and once for each of the daughter cards used.

3 Instantiating the Core in SOPC Builder

Designers should use the Audio and Video Configuration core's wizard in SOPC Builder to specify its settings. The following configurations are available and shown in Figure 3:

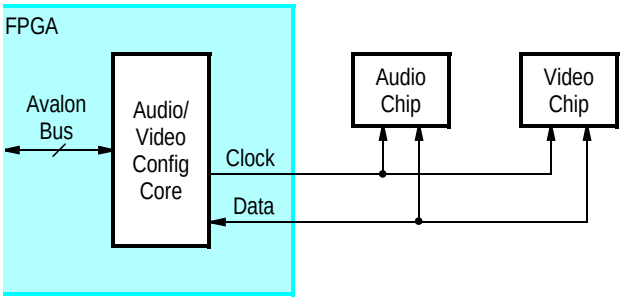

Figure 2. The Audio/Video Configuration core connected to the on-board peripherals.

- **Components**
  - **Audio/Video Device** — allows users to specify the target peripherals. Each option represents a distinct serial bus. The On-Board Peripherals option is for selecting the audio and video chips on the DE-series boards, while the other options are for selecting the various daughtercards.
  - **DE Board** — allows users to specify the target board.
  - **Auto Initialize Device(s)** — allows users to add hardware to the core that will configure the target peripherals for the default operation required by other Altera University Program IP cores, which interact with those peripherals.
- **Auto Initialization Parameters for Audio** — enabled when On-Board Peripherals and Auto Initialize Device(s) are both selected.
  - **Audio In Path** — allows users to choose the microphone or the Line In as the input device for the ADC.
  - **Audio Out - Enable DAC Output** — enables data from the DAC to pass to the Line Out.
  - **Audio Out - Microphone Bypass** — If enabled the signal from the microphone input jack bypasses the ADC and DAC, and goes directly to the output jack.
  - **Audio Out - Line In Bypass** — If enabled the signal from the Line In input jack bypasses the ADC and DAC, and goes directly to the output jack.
  - **Data Format** — allows users to choose the data format as DSP Mode, I2S Format, Left Justified or Right Justified. If using Altera's UP Audio core, then the Left Justified mode must be selected.
  - **Bit Length** — allows the user to specify the number of bits of each audio sample. Valid values include 16, 20 and 24 bits for all modes, and 32 bits for I2S and Left Justified modes only. Both ADC and DAC operate on 24-bit data, so zero-padding or stripping of the least-significant bits happens when the bit length is not 24. Refer to the *Device Operation* Section in the Audio CODEC Datasheet for details.
  - **Sampling Rate** — allows the user to select the number of audio sample per second. Based on the selected rate, the audio chip's Base-Over Sampling Rate and Sample Rate control registers will be set appropriately, as per Wolfson WM8731 Audio CODEC datasheet
- **Auto Initialization Parameters for Video** — enabled when On-Board Peripherals, Auto Initialize Device(s) and a DE-series boards are all selected.
  - **Video Source Format** — allows users to specify the video source as either NTSC or PAL.

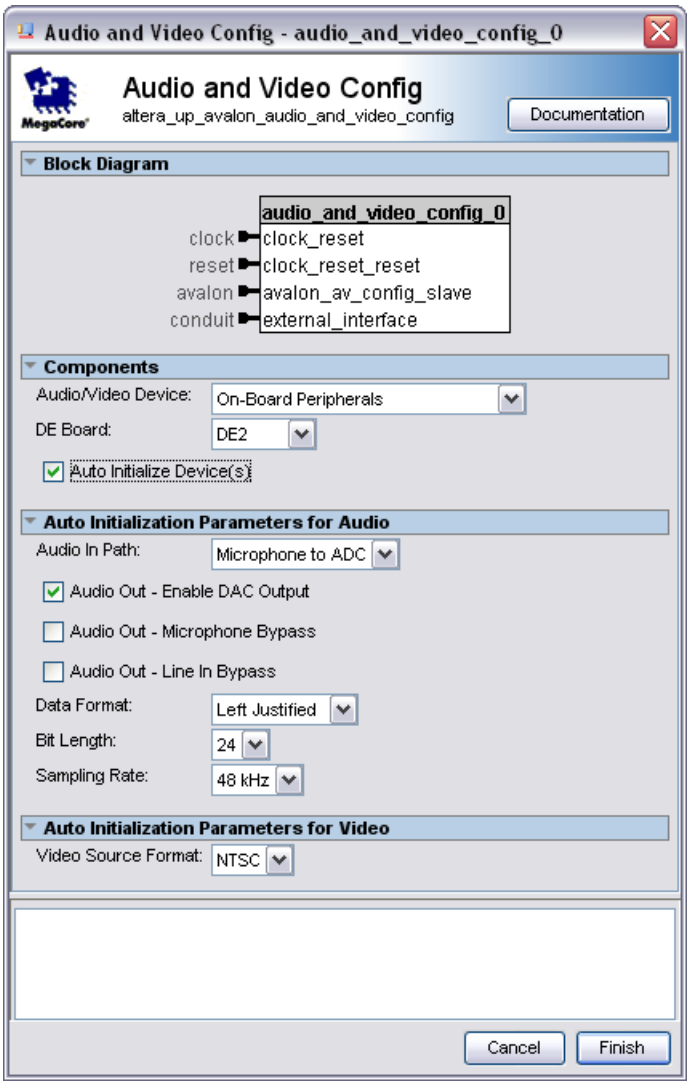

Figure 3. Audio/Video Configuration core's SOPC Builder wizard.

See the Wolfson WM8731 Audio CODEC datasheet for more information about the audio chip's parameters.

Altera strongly recommends that users configure and initialize the audio and video devices with the auto-initialize option, and the default settings. If the audio input comes from the microphone instead of Line In, users should select Microphone to ADC in the Audio-In Path. Also, users may want to adjust the Line In Bypass or Mic Bypass settings, as well as, the Bit Length.

4 Software Programming Model

The Audio/Video Configuration IP core can be controlled by a program running on a Nios II processor. This is accomplished by reading from and writing to the memory-mapped registers in the core. This can be done by directly accessing these registers or using C-language functions accessible through the Hardware Abstraction Layer (HAL) interface.

In the following sections, we describe the memory-mapped registers and how to use them, both directly and using the HAL interface.

4.1 Register Map

The Audio/Video Configuration Core has four 32-bit memory-mapped registers. By writing into these registers, the processor can configure the audio and video devices. Table 1 shows the byte offest, access rights (Read/Write/Clear) and the meaning of each bit of each register in the IP core.

Table 1. Audio/Video Configuration Core register map

Offset in bytes	Register Name	R/W/C	31...24	23...16	15...9	8	7...2	1	0
0	`control`	W	*(1)*	DEV	*(1)*			RIE	R
4	`status`	R/C	*(1)*	CFG	*(1)*	AIS	*(1)*	RDY	ACK
8	`address`	W	*(1)*				Addr *(2)*		
12	`data`	W	*(1)*		Data *(3)*				

Notes on Table 1:

(1) Reserved. Read values are undefined. Write zero.

4.1.1 *Control* Register

Table 2. Control register bits

Bit number(s)	Bit/Field name	Read/Write/Clear	Description
0	R	W	Resets the core and re-initializes the device(s) if auto-initialize was selected.
1	RIE	W	Ready Interrupt Enable - writing a one to this bit will enable an interrupt to occur when the core is ready to start a new serial transfer
23...16	DEV	W	Used to identify the device that is to be configured by the next transfer. Only required when more that one device can be configured by the core. This is the case for on-board device configuration. The valid settings are described in section 4.4.2.

4.1.2 *Status* Register

Table 3. Status register bits

Bit number	Bit name	Read/Write/Clear	Description
0	ACK	R/C	Acknowledge – 1 indicates an error occurred while transmitting the data. This bit is cleared by either writing a 1 or by initiating a new transfer.
1	RDY	R/C	Ready – 1 indicates that the core is ready to start a new transfer. This bit is cleared by initiating a new transfer.
8	AIS	R/C	Autoinitialization successful – 1 indicates that this component has successfully completed initializing all devices. It is cleared by restarting auto-initialization.
23...16	CFG	R	Used to identify the device(s) that can be configured by this component. The value settings are described in section 4.4.1.

4.1.3 *Address* Register

The `address` register is used to transmit the address of the device's control register to be accessed. Table 4 shows the address register's format for the various peripherals.

Table 4. Address Register Formats

Device	31...8	7	6	5...0
On Board Audio	*(1)*		Addr	
On Board Video	*(1)*	Addr		
1.3 megapixel digital camera	*(1)*	Addr		
5 megapixel digital camera	*(1)*	Addr		
LCD with touchscreen	*(1)*			Addr

Notes on Table 4:

(1) Reserved. Read values are undefined. Write zero.

4.1.4 *Data* Register

The `data` register is used to transmit the data to/from the device's control registers. The audio device is write only, therefore this register should not be read. Table 5 shows the data register's format for the various peripherals.

4.2 Configuring Devices

Devices are configured using multiple writes or/and reads to this core. Altera recommends using the auto-initialize option or/and the provided software functions, instead of directly communicating with this core. If this core's software functions are not used, the user can follow the steps below to communicate with devices (refer to Section 4.1

Table 5. Data Register Formats				
Device	**31...16**	**15...9**	**8**	**7...0**
On Board Audio	*(1)*		Data	
On Board Video	*(1)*			Data
1.3 megapixel digital camera	*(1)*	Data		
5 megapixel digital camera	*(1)*	Data		
LCD with touchscreen	*(1)*			Data

Notes on Table 5:

(1) Reserved. Read values are undefined. Write zero.

for details on register maps):

1. If multiple devices are configurable by the core, set the device parameter in the `control` register to the appropriate device.
2. Write the peripheral device's control register address to the `address` register.
3. Write to the `data` register to configure the peripheral's control register.
4. When the `RDY` bit in the `status` register is 1, the transfer is complete.
5. No error occured in the transfer if the acknowledgment (`ACK`) bit in the `status` register is zero.

Note that configuring a device's control registers takes multiple uninterrupted stores or/and loads to the Audio/Video Configuration Core. This implies that if multiple programs are attempting to use this core simultaneously, a hardware mutex should be used to protect access to the core during the above steps.

The clock frequency used by the serial buses is in the kilohertz range. Therefore, data transfers on this bus take a significate amount of time. Using polling to check the ready bit in the `status` register would be very inefficient, and Altera suggest to use of the Audio/Video Configuration core's interrupts to check when it is ready for the next data transfer.

4.3 Software Functions

The Audio/Video Configuration Core is packaged with C-language functions accessible through the hardware abstraction layer (HAL), as listed below. These functions enable users to send configuration data to the devices based on the register address and data given. Users may need to refer to the register map (or register description) section of the following datasheets:

- Wolfson WM8731 Audio CODEC Datasheet — for the audio configuration.
- Video DAC ADV7181 Datasheet — for the video configuration on the DE2 board.
- Video DAC ADV7180 Datasheet — for the video configuration on the DE2-70 and DE2-115 boards.
- LTM Datasheet — for the LTM configuration.

- 5MP CMOS Digital Image Sensor Datasheet — for the D5M camera configuration.

To use the functions, the C code must include the statement:

```
#include "altera_up_avalon_audio_and_video_config.h"
```

4.4 Audio/Video Configuration Functions

4.4.1 CONFIG_DEVICE_TYPE

Prototype:

```
typedef enum {
    UNKNOWN_CONFIG = 0;
    ON_BOARD_AUDIO_ONLY_CONFIG = 1;
    ON_BOARD_DE2_CONFIG = 2;
    ON_BOARD_DE2_70_CONFIG = 3;
    ON_BOARD_DE2_115_CONFIG = 4;
    TRDB_DC2_CONFIG = 8;
    TRDB_D5M_CONFIG = 9;
    TRDB_LTM_CONFIG = 10;
} CONFIG_DEVICE_TYPE;
```

Include: `<altera_up_avalon_audio_and_video_config.h>`

Fields:
`UNKNOWN_CONFIG` — Configuration is unknown.
`ON_BOARD_AUDIO_ONLY_CONFIG` — Configuration is for on-board audio only.
`ON_BOARD_DE2_CONFIG` — Configuration is for both the on-board audio and video on the DE2 board.
`ON_BOARD_DE2_70_CONFIG` — Configuration is for both the on-board audio and video on the DE2-70 board.
`ON_BOARD_DE2_115_CONFIG` — Configuration is for both the on-board audio and video on the DE2-115 board.
`TRDB_DC2_CONFIG` — Configuration is for the 1.3 megapixel digital camera daughtercard.
`TRDB_D5M_CONFIG` — Configuration is for the 5 megapixel digital camera daughtercard.
`TRDB_LTM_CONFIG` — Configuration is for the LCD with touch-screen daughtercard.

4.4.2 SELECTED_ON_BOARD_DEVICE

Prototype:

```
typedef enum {
    AUDIO_DEVICE = 0;
    FIRST_VIDEO_DEVICE = 1;
    SECOND_VIDEO_DEVICE = 2;
} SELECTED_ON_BOARD_DEVICE;
```

Include: `<altera_up_avalon_audio_and_video_config.h>`
Fields: `AUDIO_DEVICE` — Configure the on-board audio device.
`FIRST_VIDEO_DEVICE` — Configure the on-board video device for the DE2 and DE2-115 or the first on-board video device for the DE2-70.
`SECOND_VIDEO_DEVICE` — Configure the second on-board video device for the DE2-70.

4.4.3 alt_up_av_config_open_dev

Prototype: `alt_up_av_config_dev* alt_up_av_config_open_dev(const char *name)`
Include: `<altera_up_avalon_audio_and_video_config.h>`
Parameters: `name` – the Audio/Video Configuration component name in SOPC Builder.
Returns: The corresponding device structure, or NULL if the device is not found.
Description: Opens the Audio/Video Configuration device specified by *name* .

4.4.4 alt_up_av_config_reset

Prototype: `int alt_up_av_config_reset(alt_up_av_config_dev *av_config)`
Include: `<altera_up_avalon_audio_and_video_config.h>`
Parameters: `av_config` – the device structure
Returns: 0 for success
Description: Resets the AV Config core and re-initializes the peripherals device(s) is auto-initialize was enabled.

4.4.5 alt_up_av_config_enable_interrupt

Prototype: `int alt_up_av_config_enable_interrupt(alt_up_av_config_dev *av_config)`
Include: `<altera_up_avalon_audio_and_video_config.h>`
Parameters: `av_config` – the device structure
Returns: 0 for success
Description: Enables the AV Config core's interrupt.

4.4.6 alt_up_av_config_disable_interrupt

Prototype: `int alt_up_av_config_disable_interrupt(alt_up_av_config_dev *av_config)`
Include: `<altera_up_avalon_audio_and_video_config.h>`
Parameters: `av_config` – the device structure
Returns: 0 for success
Description: Disables the AV Config core's interrupt.

4.4.7 alt_up_av_config_read_acknowledge

Prototype: `int alt_up_av_config_read_acknowledge(alt_up_av_config_dev *av_config)`
Include: `<altera_up_avalon_audio_and_video_config.h>`
Parameters: `av_config` – the device structure
Returns: The acknowledge bit or -1 if the core it not ready
Description: Returns the acknowledge bit.

4.4.8 alt_up_av_config_read_ready

Prototype: `int alt_up_av_config_read_ready(alt_up_av_config_dev *av_config)`
Include: `<altera_up_avalon_audio_and_video_config.h>`
Parameters: `av_config` – the device structure
Returns: 1 if the core is ready otherwire 0
Description: Returns the ready bit.

4.4.9 alt_up_av_config_write_audio_cfg_register

Prototype:
```
int alt_up_av_config_write_audio_cfg_register(
alt_up_av_config_dev *av_config, alt_u32 addr,
alt_u32 data)
```
Include: `<altera_up_avalon_audio_and_video_config.h>`
Parameters: `av_config` – the device structure
`addr` – the device's configuration register's address
`data` – the data to be written.
Returns: 0 for success
Description: Writes configuration data to one of the on-board audio device's registers.
Notes: The *av_config* structure should represent a component that does on-board audio configuration only, otherwise an error will occur.

4.4.10 alt_up_av_config_read_video_cfg_register

Prototype: `int alt_up_av_config_read_video_cfg_register( alt_up_av_config_dev *av_config, alt_u32 addr, alt_u32 *data, SELECTED_ON_BOARD_DEVICE video_port)`
Include: `<altera_up_avalon_audio_and_video_config.h>`
Parameters: `av_config` – the device structure
`addr` – the device's configuration register's address
`data` – a pointer to the location where the read data should be stored
`video_port` – the video port to be written to. Should be 1 for the DE2 and DE2-115 boards, and either 1 or 2 for the DE2-70 board
Returns: 0 for success or -1 for failure
Description: Reads configuration data from one of the on-board video device's registers.
Notes: The *av_config* structure should represent a component that does both on-board audio and video configuration, otherwise an error will occur.

4.4.11 alt_up_av_config_write_video_cfg_register

Prototype: `int alt_up_av_config_write_video_cfg_register( alt_up_av_config_dev *av_config, alt_u32 addr, alt_u32 data, SELECTED_ON_BOARD_DEVICE video_port)`
Include: `<altera_up_avalon_audio_and_video_config.h>`
Parameters: `av_config` – the device structure
`addr` – the device's configuration register's address
`data` – the data to be written
`video_port` – the video port to be written to. Should be 1 for the DE2 and DE2-115 boards, and either 1 or 2 for the DE2-70 board
Returns: 0 for success
Description: Writes configuration data to one of the on-board video device's registers.
Notes: The *av_config* structure should represent a component that does both on-board audio and video configuration, otherwise an error will occur.

4.4.12 alt_up_av_config_read_LTM_cfg_register

Prototype: `int alt_up_av_config_read_LTM_cfg_register( alt_up_av_config_dev *av_config, alt_u32 addr, alt_u32 *data)`
Include: `<altera_up_avalon_audio_and_video_config.h>`
Parameters: `av_config` – the device structure
`addr` – the device's configuration register's address
`data` – a pointer to the location where the read data should be stored
Returns: 0 for success or -1 for failure
Description: Reads configuration data from one of the LCD with touchscreen device's registers.
Notes: The *av_config* structure should represent a component that does LTM configuration, otherwise an error will occur.

4.4.13 alt_up_av_config_write_LTM_cfg_register

Prototype: `int alt_up_av_config_write_LTM_cfg_register( alt_up_av_config_dev *av_config, alt_u32 addr, alt_u32 data)`
Include: `<altera_up_avalon_audio_and_video_config.h>`
Parameters: `av_config` – the device structure
`addr` – the device's configuration register's address
`data` – the data to be written.
Returns: 0 for success
Description: Writes configuration data to one of the LCD with touchscreen device's registers.
Notes: The *av_config* structure should represent a component that does LTM configuration, otherwise an error will occur.

4.4.14 alt_up_av_config_read_DC2_cfg_register

Prototype: `int alt_up_av_config_read_DC2_cfg_register( alt_up_av_config_dev *av_config, alt_u32 addr, alt_u32 *data)`
Include: `<altera_up_avalon_audio_and_video_config.h>`
Parameters: `av_config` – the device structure
`addr` – the device's configuration register's address
`data` – a pointer to the location where the read data should be stored
Returns: 0 for success or -1 for failure
Description: Reads configuration data from one of the 1.3 megapixel digital camera device's registers.
Notes: The *av_config* structure should represent a component that does DC2 camera configuration, otherwise an error will occur.

4.4.15 alt_up_av_config_write_DC2_cfg_register

Prototype: `int alt_up_av_config_write_DC2_cfg_register( alt_up_av_config_dev *av_config, alt_u32 addr, alt_u32 data)`
Include: `<altera_up_avalon_audio_and_video_config.h>`
Parameters: `av_config` – the device structure
`addr` – the device's configuration register's address
`data` – the data to be written.
Returns: 0 for success
Description: Writes configuration data to one of the 1.3 megapixel digital camera device's registers.
Notes: The *av_config* structure should represent a component that does DC2 camera configuration, otherwise an error will occur.

4.4.16 alt_up_av_config_read_D5M_cfg_register

Prototype: `int alt_up_av_config_read_D5M_cfg_register( alt_up_av_config_dev *av_config, alt_u32 addr, alt_u32 *data)`
Include: `<altera_up_avalon_audio_and_video_config.h>`
Parameters: `av_config` – the device structure
`addr` – the device's configuration register's address
`data` – a pointer to the location where the read data should be stored
Returns: 0 for success or -1 for failure
Description: Reads configuration data from one of the 5 megapixel digital camera device's registers.
Notes: The *av_config* structure should represent a component that does D5M camera configuration, otherwise an error will occur.

4.4.17 alt_up_av_config_write_D5M_cfg_register

Prototype: `int alt_up_av_config_write_D5M_cfg_register( alt_up_av_config_dev *av_config, alt_u32 addr, alt_u32 data)`
Include: `<altera_up_avalon_audio_and_video_config.h>`
Parameters: `av_config` – the device structure
`addr` – the device's configuration register's address
`data` – the data to be written.
Returns: 0 for success
Description: Writes configuration data to one of the 5 megapixel digital camera device's registers.
Notes: The *av_config* structure should represent a component that does D5M camera configuration, otherwise an error will occur.