AVR32120: AVR32 ABDAC audio bitstream DAC driver example

Features

- **16-bit stereo digital-to-analog converter.**
- **20 Hz to 20 kHz frequency range +- 2.5 dB.**
- **High impedance output.**

1 Introduction

This application note describes how to use the ABDAC peripheral on AVR®32 devices. This DAC is suitable for generating audio playback. By using the generic clock interface, the ABDAC is capable of supporting a wide range of playback frequencies.

Also included with this application note is an example ABDAC driver for a standalone development. The example driver outputs a sinus waveform from the ABDAC.

32-bit AVR® Microcontrollers

Application Note

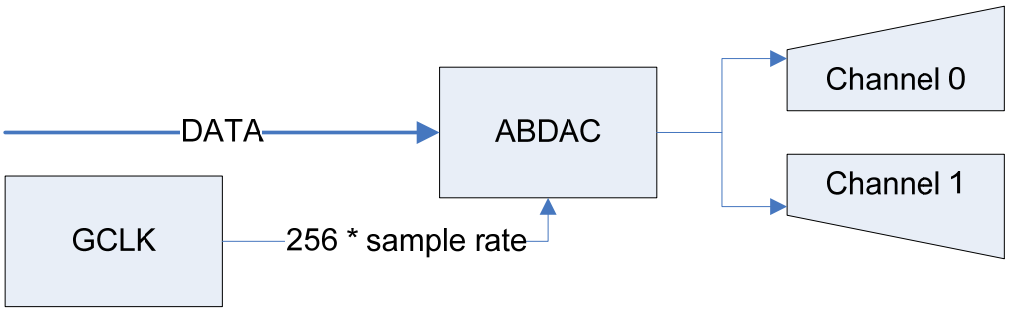
Rev. 32079C-AVR32-03/08

2 Functional description

For detailed description of the ABDAC peripheral see the datasheet of the device.

The ABDAC is a very simple peripheral and its use is straight forward. It needs a clock signal provided from the generic clock system and data input to the channels. See the block diagram in Figure 2-1 to get an overview of the module.

Figure 2-1. Clock and data path block diagram.



2.1 Generic clock

The ABDAC uses a generic clock to provide the sample frequency. This generic clock is hard wired inside the device and must be 256 times the sample frequency.

The generic clock should be configured and enabled before the ABDAC is enabled. For a description of which generic clock is used see the clock section in the datasheet of the device. Further configurations of the generic clock are also described in this section.

The generic clock output range may be limited by its source clock frequency, it is therefore vital to design in an oscillator which is able to provide a base frequency that is dividable by the generic clock divider in order to reach the required output sample rate. See Table 2-1 for examples.

Table 2-1. Base frequencies needed for an output sample rate.

Output sample rates	OSC or PLL frequency	GCLK divider
48000 Hz, 240000 Hz, 12000 Hz	24.576 MHz	2, 4, 8
44100 Hz, 22050 Hz, 11025 Hz	22.5792 MHz	2, 4, 8
32000 Hz, 16000 Hz, 8000 Hz	16.384 MHz	2, 4, 8

2.2 Channels

When the ABDAC is enabled it expects the Sample Data Register (SDR) to be updated at the same interval as the output sample rate. Both channels can be updated with one write instruction, since they are in the same I/O register (SDR).

If the sample data register is not updated within 256 clock cycles from the generic clock input to ABDAC, the UNDERRUN bit will be set in the Interrupt Status Register (ISR). Underruns are a sign of too much CPU load and therefore the application should be implemented by using interrupts, or even better, direct memory access (DMA) if available in the device.

2.3 Interrupts

There are two interrupts available to offload the CPU.

The TX_READY interrupt can be used as a trigger to signal that the next sample for each channel can be written.

The application should also enable the UNDERRUN interrupt to handle underruns when filling the Sample Data Register (SDR). Underruns will cause glitches and noise on the output signals.

If the underrun interrupt trigger, it is a sign of CPU overloading because the application was not able to provide the data in time.

2.4 DMA

The ABDAC may be connected to a DMA controller on the device. This will offload the CPU when transferring data from a buffer in RAM to the ABDAC. The application will only need to fill a buffer and pass the buffer address to the DMA controller.

Triggers for when a buffer is complete will let the application know when to pass a new buffer to the DMA controller.

The underrun interrupt is vital for DMA transfers as it will indicate that the data busses in the device are overloaded or the DMA transfer to the ABDAC does not have enough priority.

3 Electrical connection

The output from the device is not intended for driving headphones or speakers. The pads are limiting the maximum amount of current. In the majority of all practical cases, this will not be enough to drive a low impedance source.

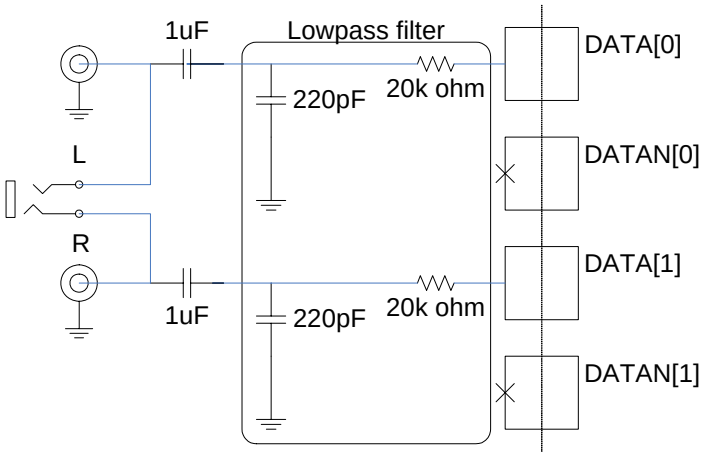
Because of this limitation, an external amplifier should be connected to the output lines to amplify these signals. This amplifier device could also be used to control the volume.

For testing purposes a line in or microphone input on a sound system can be used to evaluate the output signal.

3.1 Passive filter

For connecting the ABDAC to high impedance devices, like line in on an amplifier, a passive filter should be added. See Figure 3-1 for an example schematic.

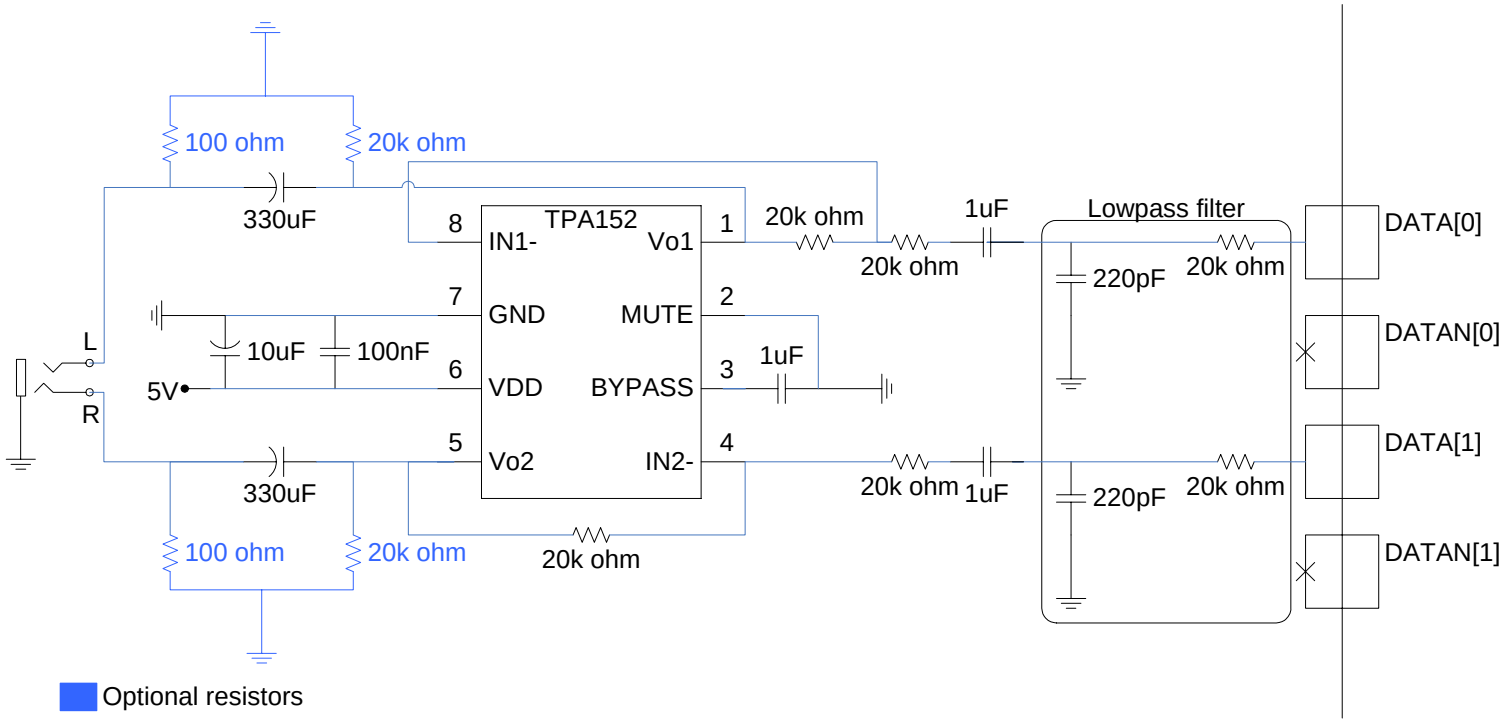
Figure 3-1. Line out with passive filter schematic



3.2 External amplifier

An external amplifier is required if the ABDAC is driving low impedance headphones or speakers directly. See Figure 3-2 for an example schematic using Texas Instruments™ TPA152 stereo audio amplifier.

Figure 3-2. High power output with external amplifier schematic



4 Driver implementation

4.1 Files

The driver consists of two files “abdac.c” and “abdac.h”. Where “abdac.h” declares all functions and “abdac.c” contains the source code. The only change needed in the driver is to specify the target device.

The target device is specified at the top in “abdac.h”.

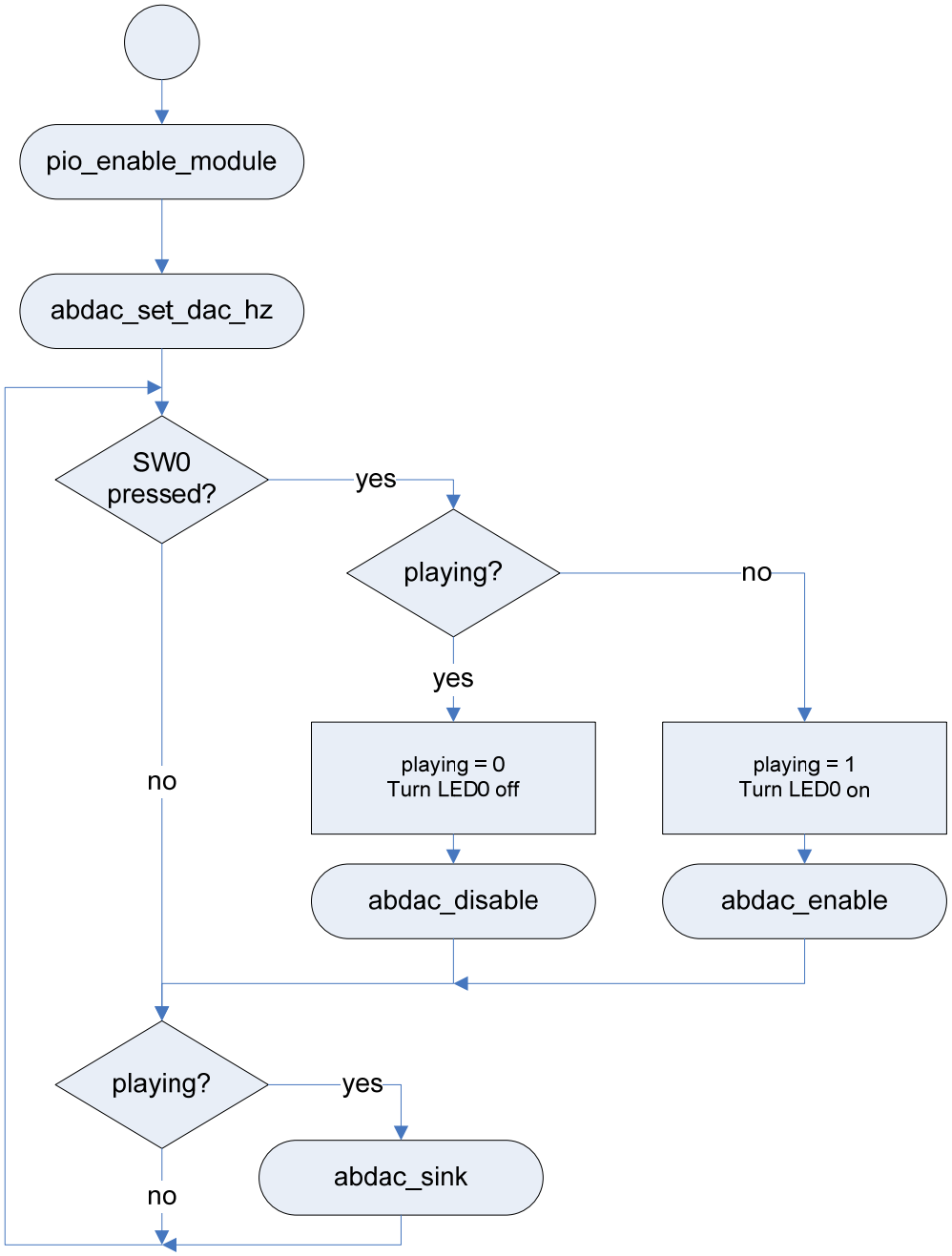
4.2 Example code

The example code for the driver outputs a sinus wave on both DAC channels. This output is enabled by a user input on a GPIO line. The wiring information is included in the Doxygen documentation, see page 6.

The example code is targeted for ATSTK®1000, but can, with some tweaking, work with any AVR32 devices with an ABDAC.

Figure 4-1 shows the flow of the example application. The application is implemented by polled function calls to make it less dependant on other modules.

Figure 4-1. ABDAC example application flowchart.



4.3 Doxygen documentation

All source code is prepared for doxygen automatic documentation generation. Premade doxygen documentation is also available within the source code included with this application note. It is located in doxygen/index.html.

Doxygen is a tool for generating documentation from source code by analyzing the source code and using known keywords. For more details see http://www.stack.nl/~dimitri/doxygen/.

5 Further reading

5.1 DMA Controller

The DMA controller can offload the CPU while outputting data from the ABDAC. For more information about the DMA controller see the appropriate chapter in the datasheet for the AVR32 device.

5.2 Interrupt

The ABDAC interface has an interrupt line connected to the interrupt controller (IC). Handling the ABDAC interrupt requires programming the IC before configuring the ABDAC.

For more information and details about the interrupt controller see application note *AVR32101 – The AVR32 Interrupt Controller*.

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr32@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature