A DSP/BIOS AD535 Codec Device Driver for the TMS320C6x11 DSK

David Friedland *Software Development Systems*

ABSTRACT

This document describes the usage and design of a device driver for the TLC320AD535 audio codec on the TMS320C6x11 DSK. This device driver is written in conformance to the DSP/BIOS™ IOM device driver model and uses the generic TMS320C6X1X EDMA McBSP driver to transfer samples to and from the serial port. For details on this generic driver, see the application note *A DSP/BIOS EDMA McBSP Device Driver for TMS320C6x1x DSPs* (SPRA846).

Contents

1 Usage 2
1.1 Configuration 3
1.2 Device Parameters 4
1.3 Channel Parameters 4
1.4 Control Commands 4
2 Architecture 5
3 Constraints 5
4 References 5
Appendix A Device Driver Data Sheet 6
A.1 Device Driver Library Name 6
A.2 DSP/BIOS Modules Used 6
A.3 DSP/BIOS Objects Used 6
A.4 CSL Modules Used 6
A.5 CPU Interrupts Used 6
A.6 Peripherals Used 6
A.7 Interrupt Disable Time 6
A.8 Memory Usage 6

List of Figures

Figure 1 DSP/BIOS IOM Device Driver Model 2
Figure 2 Codec Device Driver Partitioning 3

List of Tables

Table A–1 Device Driver Memory Usage 6

1 Usage

The device driver described here is part of an IOM mini-driver. That is, it is implemented as the lower layer of a 2-layer device driver model. The upper layer is called the class driver and can be either the DSP/BIOS GIO, SIO/DIO, or PIP/PIO modules. The class driver provides an independent and generic set of APIs and services for a wide variety of mini-drivers and allows the application to use a common interface for I/O requests. Figure 1 shows the overall DSP/BIOS device driver architecture. For more information about the IOM device driver model as well as the GIO, SIO/DIO, and PIP/PIO modules, see the *DSP/BIOS Device Driver Developer's Guide* (SPRU616).

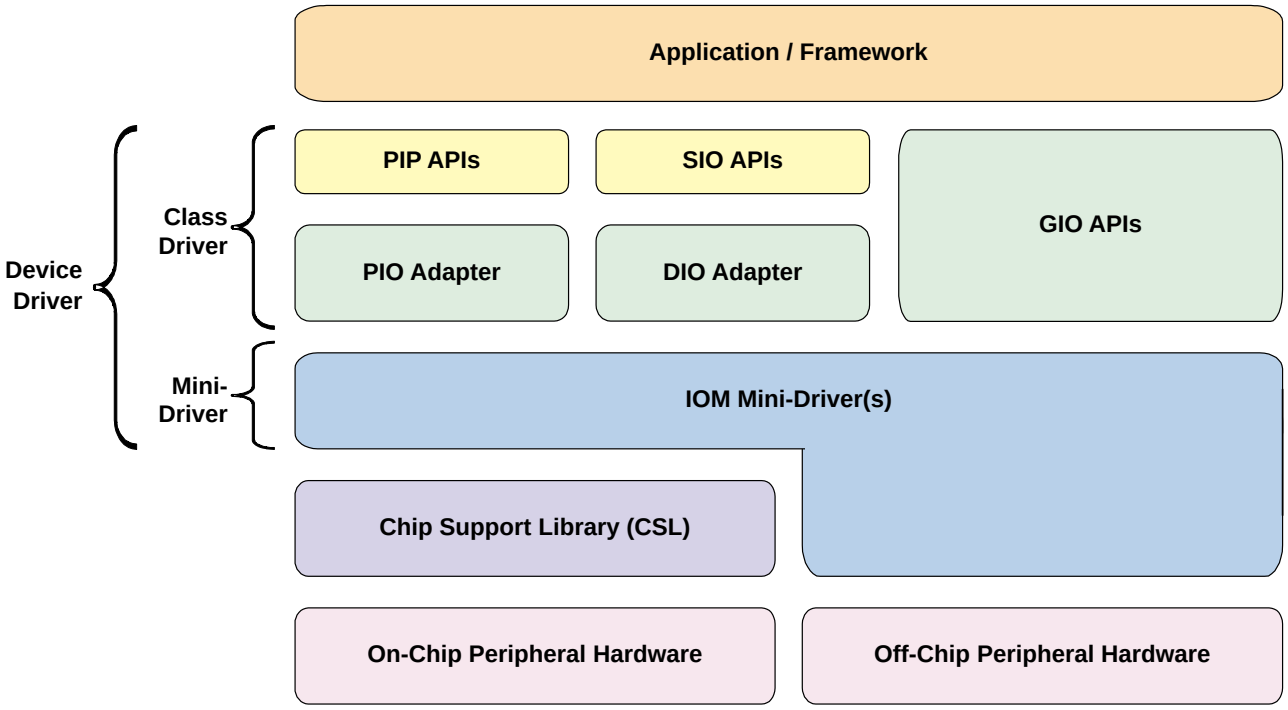

Figure 1. DSP/BIOS IOM Device Driver Model

Many mini-driver implementations split the code into a codec-specific portion and a generic portion that will work across many different codecs. Figure 2 shows the data flow between the components in a system in which the mini-driver is split into a generic part and a codec-specific part. This device driver uses the generic TMS320C6x1x EDMA McBSP device driver to transfer samples to and from the serial port. This means that to use this device driver, an application must not only link with this device driver library (*dsk6x11_edma_ad535.l62*), but also with the generic device driver library (*c6x1x_edma_mcbsp.l62*). Other than this, the use of the generic device driver is hidden from the user. These device driver libraries are compiled for TMS320C621x, but can also be used with TMS320C671x. For example, if you are using TMS320C6711 and additional floating-point optimizations are needed, recompile the libraries for TMS320C6711. Note that this device driver uses McBSP port 0 to communicate with the codec, which means it cannot be used for any other purposes.

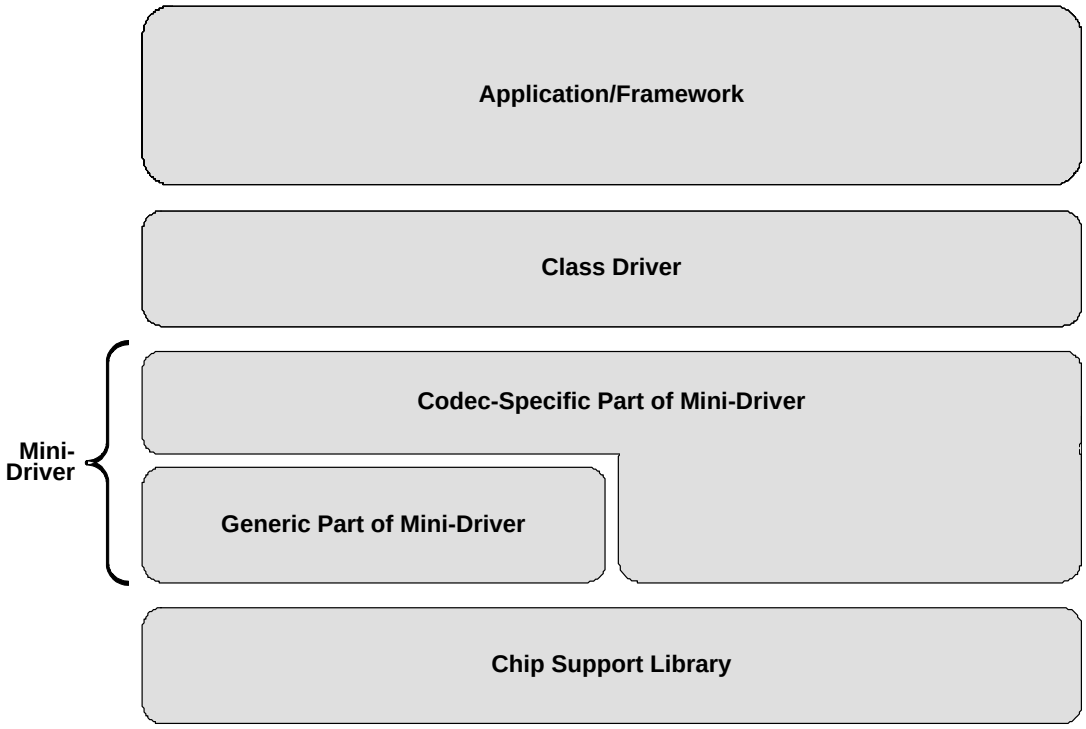

Figure 2. Codec Device Driver Partitioning

1.1 Configuration

To use this driver, a device entry has to be added and configured in the configuration tool. This device driver will set up the generic TMS320C6x1x EDMA McBSP driver to meet its needs.

- **Init function:** Type _DSK6X11_EDMA_AD535_init.
- **Function table ptr:** Type _DSK6X11_EDMA_AD535_Fxns.
- **Function table type:** Select IOM_Fxns.
- **Device id:** This property is ignored by this device driver, since there is only one TLC320AD535 codec on the TMS320C6X11 DSK.
- **Device params ptr:** A pointer to your instance of the device parameter structure. Set this property to 0x0 to use the default parameters. The parameter structure and its defaults are described below.
- **Device global data ptr:** This property must be set to 0x0.

1.2 Device Parameters

```
/* Number of AD535 registers */
#define DSK6X11_EDMA_AD535_NUMREGS 7

typedef struct DSK6X11_EDMA_AD535_DevParams {
    Int versionId;
    Bool cacheCalls;
    Int irqId;
    Int reg[DSK6X11_EDMA_AD535_NUMREGS];
    Uns intrMask;
    Int edmaPriority;
} DSK6X11_EDMA_AD535_DevParams;
```

- **versionId:** Version number of the driver.
- **cacheCalls:** If this parameter is set to TRUE, the device driver will treat buffers issued to any IOM channel associated with the device as if they are in cacheable memory and the L2 data cache is enabled. The default value of this parameter is TRUE.
- **irqId:** This parameter selects which IRQ number to use for the EDMA interrupt. The system default is 8. The default parameter of this value is 8.
- **reg[DSK6X11_EDMA_AD535_NUMREGS]:** The codec register setup. For information on the codec itself and its registers, refer to the TLC320AD535 Data Manual listed in the References section.
  - **reg[0]:** This register is a NOP. Default value is 0x0.
  - **reg[1]:** This register is a data channel register. The DSK6x11 doesn't use the data channel on the TLC320AD535. Default value is 0x0.
  - **reg[2]:** Data channel register (see above). Default value is 0x0.
  - **reg[3]:** This register is a voice channel control register. Default value is 0x6.
  - **reg[4]:** This register is a voice channel control register. Default value is 0x0.
  - **reg[5]:** Voice channel register controlling the input gain. Default value is 0x2 (0 dB).
  - **reg[6]:** Voice channel register controlling the output gain. Default value is 0x0 (0 dB).
- **intrMask:** Interrupt mask, set in the ISR.
- **edmaPriority:** Priority queue, to use, for all EDMA transfers.

1.3 Channel Parameters

This driver does not have any channel parameters. Any values that are passed as channel parameters will be ignored (NULL is suggested).

1.4 Control Commands

This device driver has no run-time control commands.

2 Architecture

This portion of the mini-driver driver inherits the features of the generic TMS320C6x1x EDMA McBSP driver. The only thing the codec-specific part does is to set up the codec and leaves the transfers of samples to the generic device driver. The fact that this device driver uses the generic device driver is hidden from the user in all aspects except that the generic device driver library has to be linked into the application. This device driver uses the McBSP port 0 to set up the codec. After setting up the codec, the device driver closes down McBSP port 0 before calling the generic TMS320C6x1x EDMA McBSP device driver's mdBindDev() function, since otherwise the generic device driver could not allocate this port for data transfer.

It is important to note that every sample sent to the codec that has the LSB set will be interpreted as a command. This device driver doesn't strip the LSB when sending samples to the codec, which means that the application layer has to strip the LSB from every sample it sends to the codec device driver. Also note that this device driver doesn't set the McBSP to "free running during emulation halt", since this is a mono codec and we don't have to worry about external frame sync problems.

3 Constraints

- Inherits the constraints of the generic TMS320C6x1x EDMA McBSP driver.
- If only the input channel is created, the samples received from the TLC320AD535 codec are erroneous. The workaround is to also open the output channel even if only the input channel is used to transfer data.

4 References

All these documents are available on the TI Developer's Village.

1. *A DSP/BIOS EDMA McBSP Device Driver for TMS320C6x1x DSPs* (SPRA846)
2. *AD535 Data Manual*, SLAS202B
3. *DSP/BIOS Device Driver Developer's Guide* (SPRU616)
4. *TMS320C6000 Chip Support Library API Reference Guide* (SPRU401)
5. *TMS320C6000 Peripherals Reference Guide* (SPRU190)
6. *TMS320C6000 DSP/BIOS Application Programming Interface (API) Reference Guide* (SPRU403)

Appendix A Device Driver Data Sheet

A.1 Device Driver Library Name

dsk6x11_edma_ad535.l62

When building an application the generic c6x1x_edma_mcbsp.l62 library is required.

A.2 DSP/BIOS Modules Used

Same as for the generic TMS320C6x1x EDMA McBSP device driver documentation.

A.3 DSP/BIOS Objects Used

Same as for the generic TMS320C6x1x EDMA McBSP device driver documentation.

A.4 CSL Modules Used

Same as for the generic TMS320C6x1x EDMA McBSP device driver documentation.

A.5 CPU Interrupts Used

Same as for the generic TMS320C6x1x EDMA McBSP device driver documentation.

A.6 Peripherals Used

Same as for the generic TMS320C6x1x EDMA McBSP device driver documentation.

A.7 Interrupt Disable Time

Maximum time that hardware interrupts can be disabled by the driver: refer to the generic TMS320C6x1x EDMA McBSP device driver documentation. This measurement is taken using the compiler option –O3.

A.8 Memory Usage

Includes the memory usage of the generic TMS320C6x1x EDMA McBSP device driver documentation.

Table A–1. Device Driver Memory Usage

	Uninitialized Memory	Initialized Memory
CODE	—	1432 (8-bit bytes)
DATA	108 (8-bit bytes)	176 (8-bit bytes)

NOTE: This data was gathered using the sectti command utility.
Uninitialized data: .bss
Initialized data: .cinit + .const
Initialized code: .text + .text:init

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI 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 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. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

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

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265