Section 33. Audio Digital-to-Analog Converter (DAC)

HIGHLIGHTS

This section of the manual contains the following major topics:

© 2007 Microchip Technology Inc.

33.1 INTRODUCTION

The Audio Digital-to-Analog Converter (DAC) module is a 16-bit Delta-Sigma signal converter designed for audio applications. Two output channels support stereo operations. Data input is in the form of a 16-bit digital value from the application program via the DMA module or the DAC data and control registers. Data output is an analog voltage which is proportional to the digital input value.

Each output channel provides three voltage outputs:

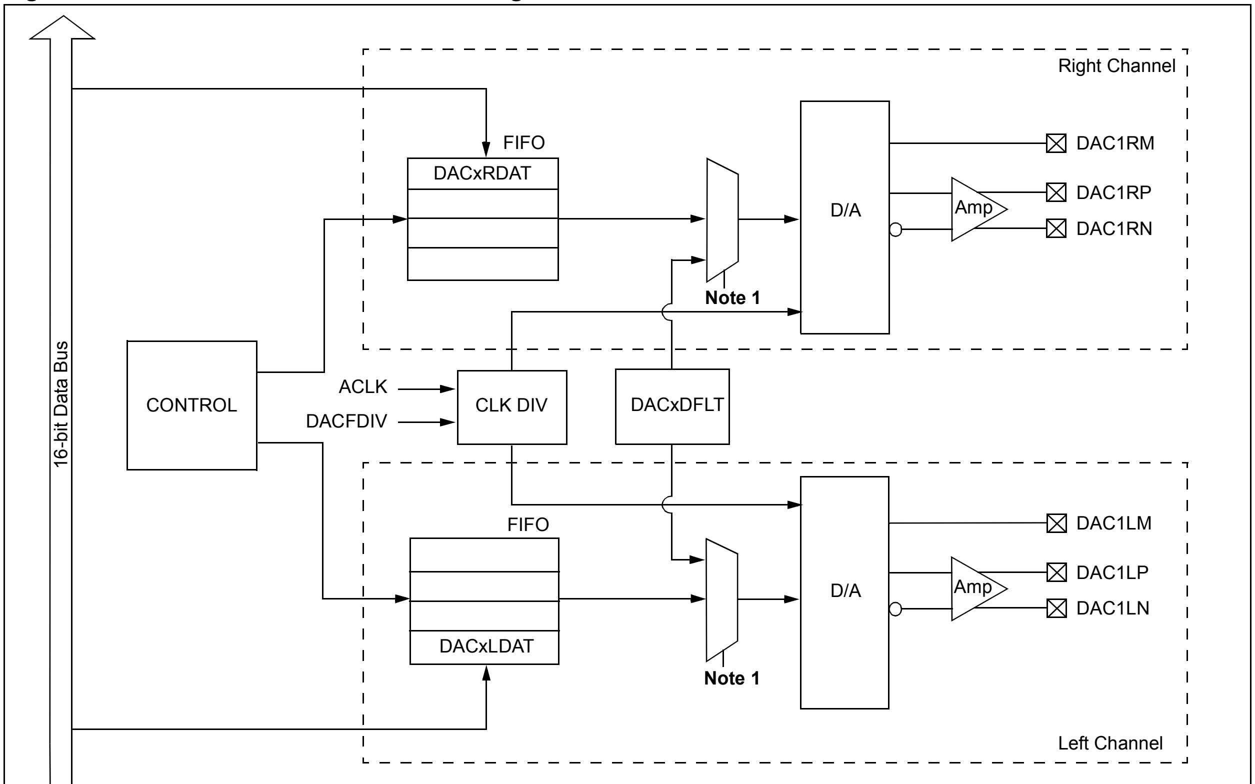
- Positive DAC output
- Negative DAC output
- Midpoint voltage output (not present on all devices)

The positive and negative outputs are differential about a midpoint voltage of approximately 1.65 volts with a voltage swing of approximately ±1 volt. The signals are capable of driving a 1k ohm load. The midpoint output is an offset voltage level that represents the midpoint of the output voltage range.

Figure 33-1 is a simplified block diagram of the Audio DAC. A four-word deep FIFO buffers the data input for each channel. If at any time the FIFO becomes empty (for example, if the DMA module or processor cannot provide data in a timely manner), the DAC accepts alternate data from the DAC Default Data Register (DACxDFLT). This register provides a default input value that represents a “safe” output voltage, which is often the midpoint value or a zero value.

The sample rate of the DAC is established by an integer division of the rate of an auxiliary oscillator or system clock by a divider circuit. The divisor ratio is specified by the DAC Clock Divider (DACFDIV<6:0>) Configuration bits in the DAC Control Register (DACxCON).

Figure 33-1: Audio DAC Module Block Diagram



Note 1: If FIFO is empty, data will be taken from the DACxDFLT register.

© 2007 Microchip Technology Inc.

33.2 KEY FEATURES

The Audio DAC provides these key features:

- 16-bit resolution (14-bit accuracy)
- Second-order digital Delta-Sigma modulator
- 256x over-sampling ratio
- 100 ksps maximum sampling rate
- User controllable sample clock
- Maximum input signal frequency of 45 kHz
- Differential analog outputs
- Signal-to-noise ratio: 90 dB
- Four word deep input buffer
- 16-bit processor I/O and DMA interfaces

33.3 DAC REGISTERS

The Audio DAC module is controlled by five DAC registers.

- **DACxCON: DAC Control Register**

This register configures the corresponding DAC module by enabling/disabling the module and by specifying data format, DAC filter clock divider and operations in Idle/Sleep mode.

- **DACxSTAT: DAC Status and Control Register**

This register specifies which channels are enabled and the status of the data buffer of that channel.

- **DACxDFLT: DAC Default Data Register**

This register specifies a DAC default value to be used as the input when the FIFO is empty.

- **DACxLDAT: DAC Left Channel Data Register**

This register specifies data for the left channel.

- **DACxRDAT: DAC Right Channel Data Register**

This register specifies data for the right channel.

Register 33-1: DACxCON: DAC Control Register

R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	U-0	R/W-0
DACEN	—	DACSIDL	AMPON	—	—	—	FORM
bit 15							bit 8

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1
—	DACFDIV<6:0>						
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15 **DACEN:** DAC Enable bit
1 = Enables module
0 = Disables module

bit 14 **Unimplemented:** Read as ‘0’

bit 13 **DACSIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when device enters Idle mode
0 = Continue module operation in Idle mode

bit 12 **AMPON:** Enable Analog Output Amplifier in Sleep Mode/Stop-in Idle Mode bit
1 = Analog Output Amplifier is enabled during Sleep Mode/Stop-in Idle mode
0 = Analog Output Amplifier is disabled during Sleep Mode/Stop-in Idle mode. All channels are reset

bit 11-9 **Unimplemented:** Read as ‘0’

bit 8 **FORM:** Data Format Select bit
1 = Signed integer
0 = Unsigned integer

bit 7 **Unimplemented:** Read as ‘0’

bit 6-0 **DACFDIV<6:0>:** DAC Clock Divider bit
1111111 = Divide input clock by 128
•
•
•
0000101 = Divide input clock by 6 (default)
•
•
0000010 = Divide input clock by 3
0000001 = Divide input clock by 2
0000000 = Divide input clock by 1 (no divide)

 © 2007 Microchip Technology Inc.

Register 33-2: DACxSTAT: DAC Status and Control Register

R/W-0	U-0	R/W-0	U-0	U-0	R/W-0	R-0	R-0
LOEN	—	LMVOEN	—	—	LITYPE	LFULL	LEMPTY
bit 15							bit 8

R/W-0	U-0	R/W-0	U-0	U-0	R/W-0	R-0	R-0
ROEN	—	RMVOEN	—	—	RITYPE	RFULL	REMPTY
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15 **LOEN:** Left Channel DAC Output Enable bit
1 = Positive and negative DAC outputs are enabled
0 = DAC outputs are disabled

bit 14 **Unimplemented:** Read as ‘0’

bit 13 **LMVOEN:** Left Channel Midpoint DAC Output Voltage Enable bit
1 = Midpoint DAC output is enabled
0 = Midpoint output is disabled

bit 12-11 **Unimplemented:** Read as ‘0’

bit 10 **LITYPE:** Left Channel Type of Interrupt bit
1 = Interrupt if FIFO is empty
0 = Interrupt if FIFO is not full

bit 9 **LFULL:** Status, Left Channel Data input FIFO is Full bit
1 = FIFO is full
0 = FIFO is not full

bit 8 **LEMPTY:** Status, Left Channel Data input FIFO is Empty bit
1 = FIFO is empty
0 = FIFO is not empty

bit 7 **ROEN:** Right Channel DAC Output Enable bit
1 = Positive and negative DAC outputs are enabled
0 = DAC outputs are disabled

bit 6 **Unimplemented:** Read as ‘0’

bit 5 **RMVOEN:** Right Channel Midpoint DAC Output Voltage Enable bit
1 = Midpoint DAC output is enabled
0 = Midpoint output is disabled

bit 4-3 **Unimplemented:** Read as ‘0’

bit 2 **RITYPE:** Right Channel Type of Interrupt bit
1 = Interrupt if FIFO is empty
0 = Interrupt if FIFO is not full

bit 1 **RFULL:** Status, Right Channel Data input FIFO is Full bit
1 = FIFO is full
0 = FIFO is not full

bit 0 **REMPTY:** Status, Right Channel Data input FIFO is Empty bit
1 = FIFO is empty
0 = FIFO is not empty

Register 33-3: DACxDFLT: DAC Default Data Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DACDFLT<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DACDFLT<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-0 **DACDFLT:** DAC Default Value bits

Register 33-4: DACxLDAT: DAC Left Channel Data Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DACLDAT<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DACLDAT<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-0 **DACLDAT:** Left Channel Data bits

Register 33-5: DACxRDAT: DAC Right Channel Data Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DACRDAT<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DACRDAT<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-0 **DACRDAT:** Right Channel Data bits

© 2007 Microchip Technology Inc.

33.4 MODULE OPERATION

Figure 33-2 illustrates the digital-to-analog conversion process.

The digital interpolation filter up-samples the input signal to create additional interpolated data points. The over sampling ratio is 256:1, or 256x the incoming sampling rate. For example, a 100 ksps input signal (the maximum sampling rate) produces 25.6M data points per second.

The interpolation filter also eliminates unwanted noise produced by the up-sampling process.

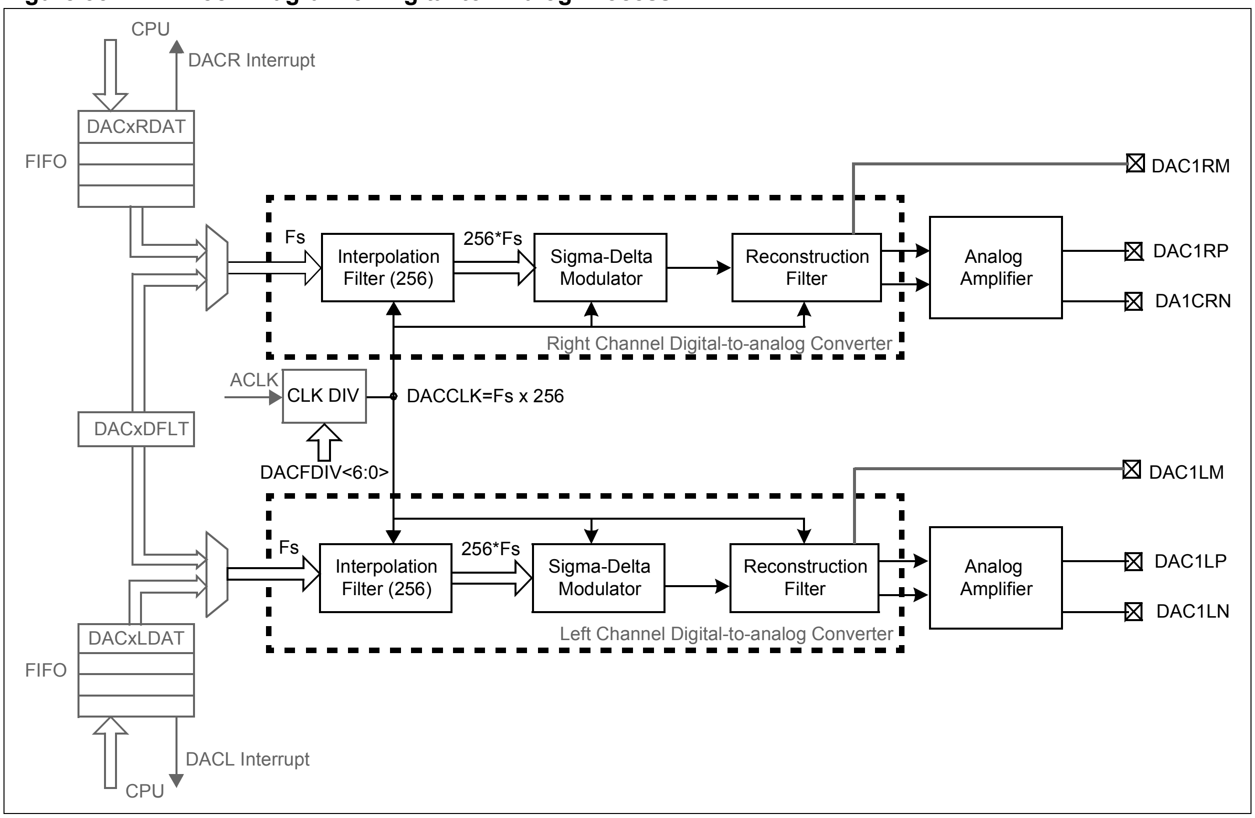
The output of the interpolation filter drives the Sigma-Delta modulator, which converts the word output from the interpolation filter into a serial bit stream.

The bitstream from the modulator is processed by the reconstruction filter to convert the bitstream to an analog signal. It then performs a low-pass filter to yield the desired voltage levels. The reconstruction filter produces two differential voltage outputs and a midpoint reference:

- Positive (voltage level representing the output signal)
- Negative (complement of the positive output signal voltage level)
- Midpoint (offset voltage level representing the midpoint of the output voltage range)

The differential outputs from the reconstruction filter are amplified by analog amplifiers to provide the required 2 volts peak-to-peak voltage swing into a 1 kΩ load.

Figure 33-2: Block Diagram of Digital-to-Analog Process



33.4.1 Data Format

The DAC module accepts 16-bit input data in two formats. Data formatting is controlled by the Data Format Select (FORM) bit in the DACxCON<8> register. The supported formats are:

- `1` = Signed (two's complement)
- `0` = Unsigned

If the FORM bit is configured for unsigned data (FORM = `0`), the user input data yields the following behavior:

- 0xFFFF = Most positive output voltage
- 0x8000 = Midpoint output voltage
- 0x7FFF = Value just below midpoint
- 0x0000 = Minimum output voltage

If the FORM bit is configured for signed data (FORM = `1`), the user input data yields the following behavior:

- 0x7FFF = Most positive output voltage
- 0x0000 = Midpoint output voltage
- 0xFFFF = Value just below midpoint
- 0x8000 = Minimum output voltage

33.4.2 Clock

The DAC clock (DACCLK) must be equal to the sampling rate times 256. Assuming a 100 ksps input, the DAC clock rate must be 25.6 MHz (100,000 x 256 = 25,600,000).

DACCLK is generated by dividing the high-speed oscillator (auxiliary or system clock) by a specified value. The divisor ratio is specified by clock divider bits (DACFDIV <6:0>) in the DAC Control Register (DACxCON). The resulting DACCLK must not exceed 25.6 MHz.

Table 33-1 lists DACCLK rates required to achieve common audio sample rates.

Table 33-1: TABLE 33-1 CLOCK RATE RATIOS

DACCLK Rate (MHz) (Fs x 256)	Sample Data Rate (kHz) (Fs)
1.8432	7.2
2.048	8.0
2.4576	9.6
2.8224	11.025
4.096	16
5.6448	22.05
6.144	24
8.192	32
11.2896	44.1
12.288	48
22.5792	88.2
24.576	96.0
25.6	100

The DACCLK rate is achieved by selecting an auxiliary oscillator with the appropriate DACFDIV<6:0> Configuration bits. For example, a DACCLK rate of 25.6 MHz (for 100 ksps input rate) might be obtained by the following:

Auxiliary Oscillator Frequency	DACFDIV Configuration Bits
25.6 MHz	DACFDIV = `0b0000000`
51.2 MHz	DACFDIV = `0b0000001`
76.8 MHz	DACFDIV = `0b0000010`

For detailed information on oscillator selection, refer to **Section 39. “Oscillator (Part III)”**.

© 2007 Microchip Technology Inc.

33.5 INTERRUPTS AND STATUS

The Audio DAC provides two interrupts, one for each channel. Depending on the setting of the interrupt Configuration bits (LITYPE for the left channel and RITYPE for the right channel) in the DAC Status Register (DACxSTAT), the DAC interrupt is triggered by either a “FIFO Empty” or “FIFO Not Full” condition. The FIFO Empty interrupt can be used with the Audio DAC to maximize throughput while minimizing the impact of interrupts on the CPU. The FIFO Empty interrupt is the simplest and preferred interrupt method for use with DMA.

The FIFO Not Full interrupt is used in applications without DMA to minimize the occurrence of DAC under-run. This interrupt can also be used with DMA, but additional software support is required.

The DAC Interrupt Service Routine (ISR) must clear its corresponding interrupt flags (DAC1LIF and DAC1RIF in the IFS4 register).

> **Note:** DAC reads data from FIFO every 256 DAC clock cycles. DAC interrupts occur once the data is read, depending on the status of the FIFO.

Each channel includes two status bits to indicate the status of its FIFO, shown in the following table:

Channel	Name[1]	Description
Left	LFULL (DAC1STAT<9>)	Left channel FIFO is full
	LEMPTY (DAC1STAT<8>)	Left channel FIFO is empty
Right	RFULL (DAC1STAT<1>)	Right channel FIFO is full
	REMPTY (DAC1STAT<0>)	Right channel FIFO is empty

Note 1: These bits can be read in software to confirm the FIFO status.

33.6 AUDIO DAC OPERATION WITHOUT DMA

Example (33-1) demonstrates a typical configuration for the DAC module. In this example the interrupts for both channels are set to occur every time the corresponding FIFO is not full. When the DAC Enable (DACEN) bit in the DAC Control (DACxCON) register gets set, DAC interrupts are generated for both channels. Since the FIFO is initially empty, the first data value is read from the DAC Default (DACxDFLT) register. For this example, the default is set to a midpoint value of 0x8000.

Example 33-1: DAC Operation Without DMA

```
DAC1STATbits.ROEN = 1;                    /* Right Channel DAC Output Enabled */
DAC1STATbits.LOEN = 1;                    /* Left Channel DAC Output Enabled */

DAC1STATbits.RITYPE = 0;                  /* Right Channel Interrupt if FIFO is not Full */
DAC1STATbits.LITYPE = 0;                  /* Left Channel Interrupt if FIFO is not Full */

DAC1CONbits.AMPON = 0;                    /* Amplifier Disabled During Sleep and Idle Modes */
DAC1CONbits.DACFDIV = 0;                  /* Divide Clock by 1 (Assumes Clock is 25.6MHz) */
DAC1CONbits.FORM = 0;                     /* Data Format is Unsigned */

DAC1DFLT = 0x8000;                        /* Default value set to Midpoint when FORM = 0 */

IFS4bits.DAC1RIF = 0;                     /* Clear Right Channel Interrupt Flag */
IFS4bits.DAC1LIF = 0;                     /* Clear Left Channel Interrupt Flag */

IEC4bits.DAC1RIE = 1;                     /* Right Channel Interrupt Enabled */
IEC4bits.DAC1LIE = 1;                     /* Left Channel Interrupt Enabled */

DAC1CONbits.DACEN = 1;                    /* DAC1 Module Enabled */


void__attribute__((interrupt, no_auto_psv))_DAC1RInterrupt(void)
{
IFS4bits.DAC1RIF = 0;                         /* Clear Right Channel Interrupt Flag */
DAC1RDAT = MyDataR[0];                        /* User Code to Write to FIFO Goes Here */
}

void__attribute__((interrupt, no_auto_psv))_DAC1LInterrupt(void)
{
IFS4bits.DAC1LIF = 0;                         /* Clear Left Channel Interrupt Flag */
DAC1LDAT = MyDataL[0];                        /* User Code to Write to FIFO Goes Here */
}
```

Figure 33-3 illustrates how the Audio DAC interacts with the application program to respond to DAC interrupts and transfer data in a timely manner. This example shows a single-word transfer per interrupt. Depending on your particular application, you could choose to write up to four words per interrupt.

Note:	A write to the FIFO is ignored if the FIFO is full or if a write occurs before the DAC is enabled.

 © 2007 Microchip Technology Inc.

Figure 33-3: Audio DAC Interaction without DMA

A. When DAC is initialized, DAC Enable (DACEN) bit is set. FIFO is not full, so DAC interrupts are generated for both channels. Because FIFOs have no data, contents of DACDFLT register is processed by both DAC channels.

Application program recognizes interrupt request and begins processing ISR.

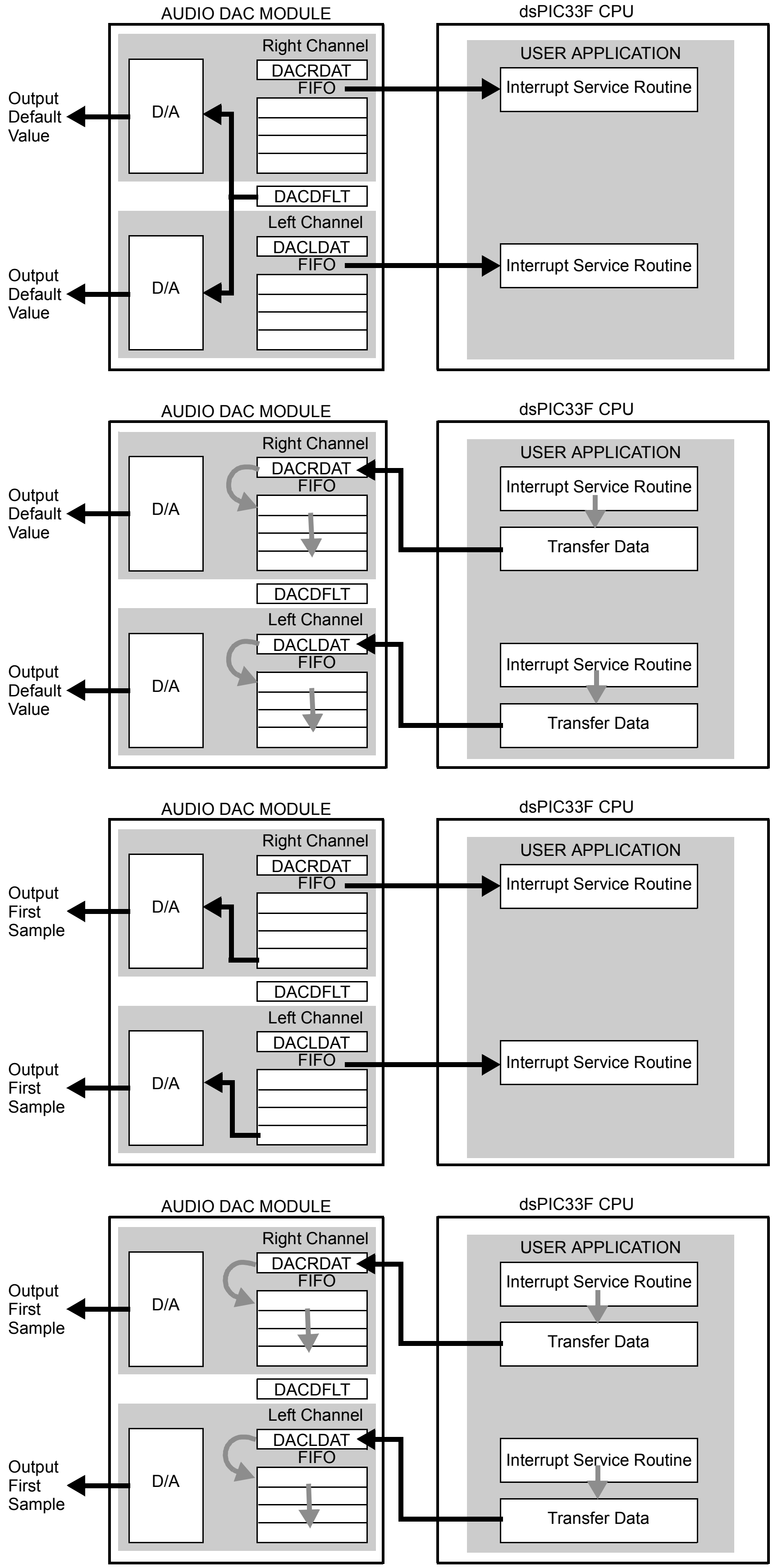
B. Interrupt Service Routine writes to DACRDAT and DACLDAT.

C. After 256 DAC clock cycles, the D/A reads data from FIFO.

As the D/A processing occurs the DAC data register status becomes “not full”, causing another DAC interrupt to be generated.

D. The application program transfers the next word to the left and right data registers. This transfer is completed before the D/A requests the next input word (i.e., before it completes 256 DAC clock cycles).

This process repeats until the application program turns off the DAC module (disables DACEN).



33.7 AUDIO DAC OPERATION WITH DMA

On some of the dsPIC33F devices, the DMA module can transfer data from the CPU to the Audio DAC without CPU assistance. Consult the dsPIC33F device data sheet to determine if DMA is present on your particular device. For more information on the DMA module, refer to **Section 22. “Direct Memory Access (DMA)”**.

When the Audio DAC module uses DMA, it requires one DMA channel for each DAC channel. The DAC channel can be mapped to any available DMA channel. Interaction begins when the DMA receives an interrupt from the Audio DAC. Depending on the setting of the interrupt Configuration bits (LITYPE for the left channel and RITYPE for the right channel) in the DAC Status Register (DACxSTAT), the DMA interrupt is triggered by either a “FIFO Empty” or “FIFO Not Full” condition (see **Section 33.5 “Interrupts and Status”**).

33.7.1 DMA Channel to Peripheral Association Setup

The DMA channel needs to know which interrupt request to respond to and which peripheral target address to write to for the Audio DAC. The interrupt is identified by the Interrupt Select (IRQSEL<6:0>) bits in the DMA Channel x IRQ Request (DMAxREQ) register (in the DMA module). The write address is identified by DMA Channel x Peripheral Address (DMAxPAD) register (also in the DMA module).

Table 33-2 shows which values should be written to associate a particular peripheral with a given DMA channel.

Table 33-2: DMA Channel to Peripheral Association

Peripheral to DMA Association	DMAxREQ Register IRQSEL<6:0> Bits	DMAxPAD Register Values to Write to Peripheral
DAC1 – Right Data Output	`1001110`	0x03F6 – DAC1RDAT
DAC1 – Left Data Output	`1001111`	0x03F8 – DAC1LDAT

33.7.2 DMA Code Setup

Example 33-2 illustrates code for a typical DAC configuration for DMA operation. In this example the interrupts are set to occur every time the FIFO is empty. When the DAC Enable (DACEN) bit gets set, an interrupt is generated because the FIFO is initially empty. Since the FIFO is empty, the first data value is read from the default register. In this case the default is the zero value.

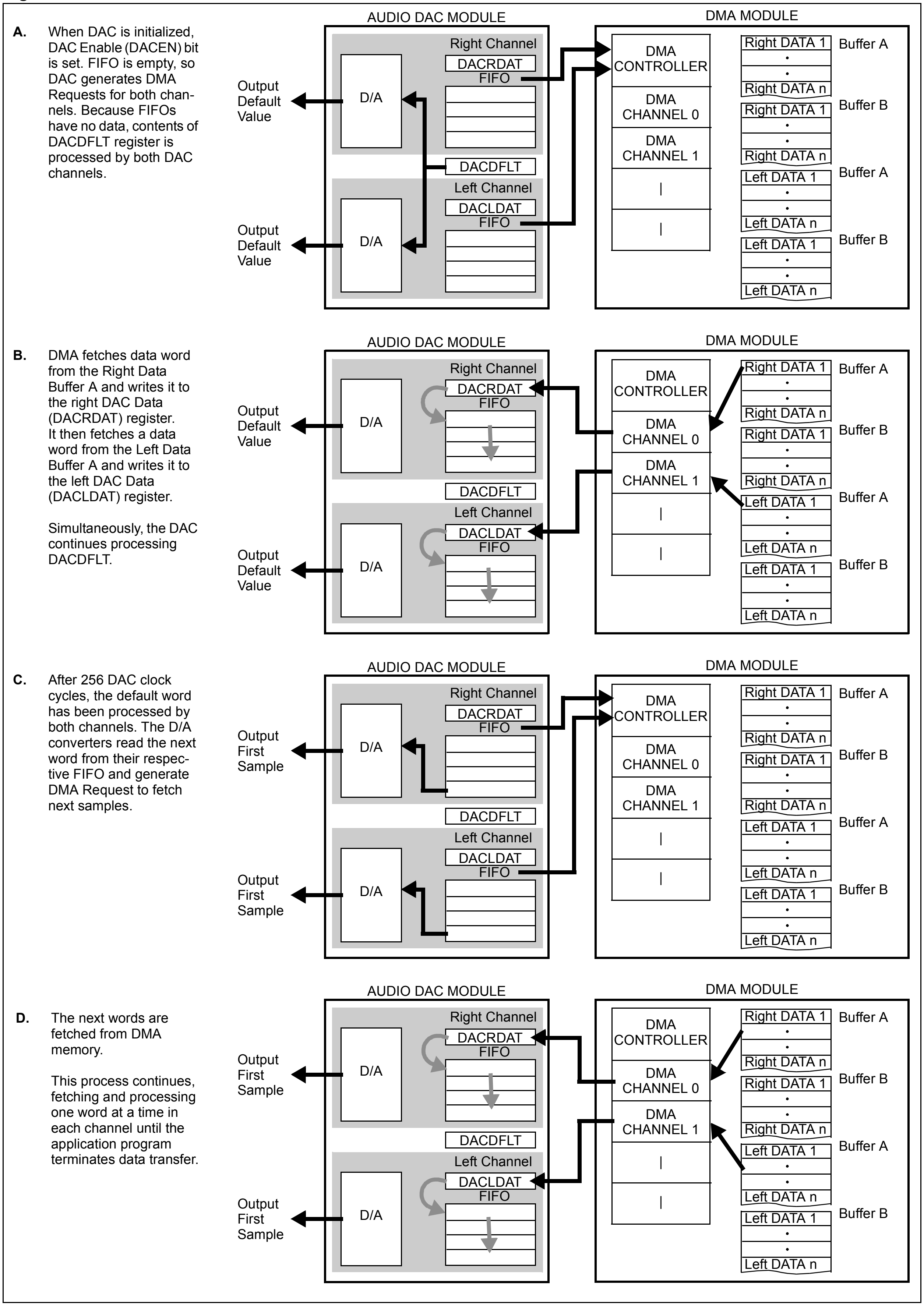
Figure 33-4 illustrates how the Audio DAC interacts with the DMA module to respond to DAC interrupts and transfer data in a timely manner. This example shows a single-word transfer per interrupt.

© 2007 Microchip Technology Inc.

Example 33-2: DAC Operation With DMA

```
/* DMA Code */
unsigned int RightBufferA[32]__attribute__((space(dma)));
unsigned int RightBufferB[32]__attribute__((space(dma)));
unsigned int LeftBufferA[32]__attribute__((space(dma)));
unsigned int LeftBufferB[32]__attribute__((space(dma)));

          /* DMA Channel 0 set to DAC1RDAT */
DMA0CONbits.AMODE = 0;                        /* Register Indirect with Post Increment */
DMA0CONbits.MODE = 2;                         /* Continuous Mode with Ping-Pong Enabled */
DMA0CONbits.DIR = 1;                          /* Ram-to-Peripheral Data Transfer */

DMA0PAD = (volatile unsigned int)&DAC1RDAT;  /* Point DMA to DAC1RDAT */

DMA0CNT = 31;                                 /* 32 DMA Request */
DMA0REQ = 78;                                 /* Select DAC1RDAT as DMA Request Source */

DMA0STA = __builtin_dmaoffset(RightBufferA);
DMA0STB = __builtin_dmaoffset(RightBufferB);

IFS0bits.DMA0IF = 0;                          /* Clear DMA Interrupt Flag */
IEC0bits.DMA0IE = 1;                          /* Set DMA Interrupt Enable Bit */

DMA0CONbits.CHEN = 1;                         /* Enable DMA Channel 0 */

          /* DMA Channel 1 set to DAC1LDAT */
DMA1CONbits.AMODE = 0;                        /* Register Indirect with Post Increment */
DMA1CONbits.MODE = 2;                         /* Continuous Mode with Ping-Pong Enabled */
DMA1CONbits.DIR = 1;                          /* Ram-to-Peripheral Data Transfer */

DMA1PAD = (volatile unsigned int)&DAC1LDAT;  /* Point DMA to DAC1LDAT */

DMA1CNT = 31;                                 /* 32 DMA Request */
DMA1REQ = 79;                                 /* Select DAC1LDAT as DMA Request Source */

DMA1STA = __builtin_dmaoffset(LeftBufferA);
DMA1STB = __builtin_dmaoffset(LeftBufferB);

IFS0bits.DMA1IF = 0;                          /* Clear DMA Interrupt Flag */
IEC0bits.DMA1IE = 1;                          /* Set DMA Interrupt Enable Bit */

DMA1CONbits.CHEN = 1;                         /* Enable DMA Channel 1 */

           /* DAC1 Code */
DAC1STATbits.ROEN = 1;                        /* Right Channel DAC Output Enabled */
DAC1STATbits.LOEN = 1;                        /* Left Channel DAC Output Enabled */

DAC1STATbits.RITYPE = 1;                      /* Right Channel Interrupt if FIFO is Empty */
DAC1STATbits.LITYPE = 1;                      /* Left Channel Interrupt if FIFO is Empty */

DAC1CONbits.AMPON = 0;                        /* Amplifier is Disabled During Sleep/Idle Modes */
DAC1CONbits.DACFDIV = 0;                      /* Divide Clock by 1 (Assumes Clock is 25.6MHz) */
DAC1CONbits.FORM = 0;                         /* Data Format is Unsigned */
DAC1CONbits.DACEN = 1;                        /* DAC1 Module Enabled */
            /* Rest of User Code Goes Here */

void__attribute__((interrupt, no_auto_psv))_DMA0Interrupt(void)
{
IFS0bits.DMA0IF = 0;                          /* Clear DMA Channel 0 Interrupt Flag */
           /* User Code to update Right Buffer in DMA*/
}
void__attribute__((interrupt, no_auto_psv))_DMA1Interrupt(void)
{
IFS0bits.DMA1IF = 0;                          /* Clear DMA Channel 1 Interrupt Flag */
           /* User Code to update Left Buffer in DMA */
}
```

Figure 33-4: Audio DAC Interaction with DMA

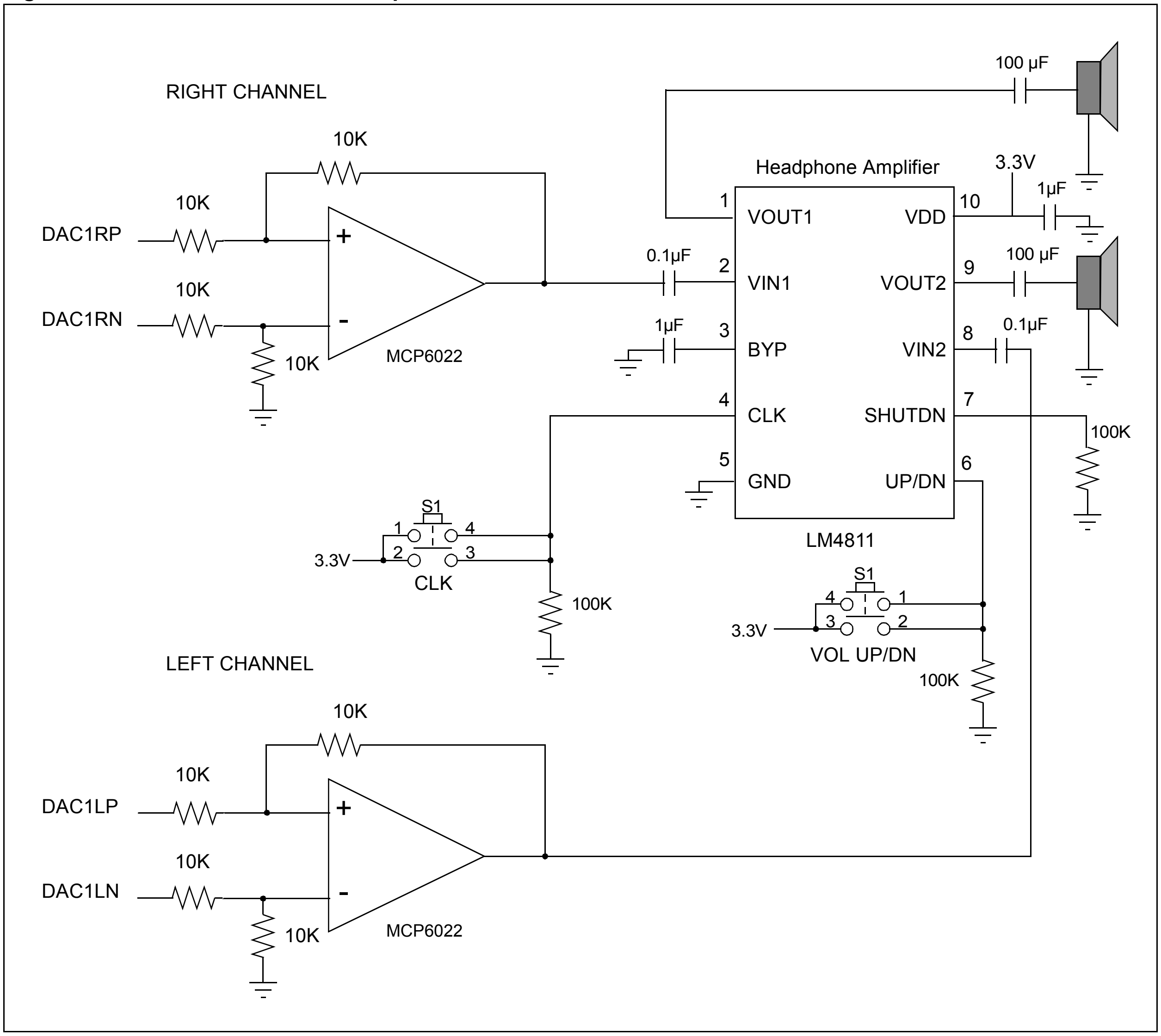


© 2007 Microchip Technology Inc.

33.8 EXTERNAL CIRCUIT EXAMPLE

Figure 33-5 shows a typical configuration for connecting a speaker with a single channel of the DAC module. This example uses the differential outputs of the DAC and produces a single-ended output using op-amp differential amplifiers. The corresponding output is two times the positive input.

Figure 33-5: External Circuit Example



33.9 OPERATIONS IN POWER SAVE MODES

The dsPIC33F family of devices has four power modes comprised of one normal (full-power) mode and three power-saving modes invoked by the `PWRSAV` instruction. Depending on the mode selected, entry into a power-saving mode may also affect the operation of the module.

When the Enable Analog Output Amplifier (AMPON) bit in the DAC Control (DACxCON<12>) register is set, the analog circuitry and the analog output amplifier are powered on during Sleep and Stop-in-Idle mode. This configuration keeps the analog output voltage at a known state during Sleep and Stop-in-Idle modes. If the AMPON bit is cleared and the processor enters Sleep or Stop-in-Idle mode, the analog circuitry is powered off and held in reset to reduce current consumption. When the output amplifier is powered off, the analog output is in a high-impedance state.

In Sleep mode, if the AMPON bit is cleared, the DAC internal state, including the FIFOs (except the SFRs), is reset, the clocks are stopped and the output analog amplifier and associated circuitry are powered down. If the AMPON bit is set, the DAC internal state and the SFRs are maintained, the clocks are stopped and the output analog amplifier and associated circuitry remain powered up.

If the Stop-in-Idle Mode (DACSIDL) bit in the DAC Control (DACxCON<13>) register is set when Idle mode is entered, and if the AMPON bit is not set, the DAC internal state (except the SFRs) is reset, the clocks are stopped and the output analog amplifier and associated circuitry is powered down. If the AMPON bit is set, the DAC internal state and the SFRs are maintained, the clocks are stopped and the output analog amplifier and associated circuitry remains powered up. If DACSIDL is not set when Idle mode is entered, the DAC ignores Idle mode and continues running.

The DACEN bit (DACxCON<15>) is cleared when the device enters Sleep or Stop-in-Idle mode, irrespective of the status of AMPON bit. If the AMPON bit is set before the device enters Sleep or Stop-in-Idle mode, the new data can be placed in the FIFOs upon exiting Sleep or Stop-in-Idle. When the DAC is re-enabled it continues operation where it left off.

Note: If AMPON = `1` and DACEN = `0`, reading the FIFO will empty it.

33.10 AUDIO DAC REGISTER MAPS

Table 33-3 is a bit map of the Audio DAC registers.

© 2007 Microchip Technology Inc.

Table 33-3: Audio DAC Register Maps

File Name	SFR ADDR	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit1	Bit 0	All Resets
DACxCON	03F0	DACEN	—	DACSIDL	AMPON	—	—	—	FORM	—	DACFDIV<6:0>							0000
DACxSTAT	03F2	LOEN	—	LMVOEN	—	—	LITYPE	LFULL	LEMPTY	ROEN	—	RMVOEN	—	—	RITYPE	RFULL	REMPTY	0000
DACxDFLT	03F4	DACDFLT<15:0>																0000
DACxRDAT	03F6	DACRDAT<15:0>																0000
DACxLDAT	03F8	DACLDAT<15:0>																0000

© 2007 Microchip Technology Inc.

33.11 REVISION HISTORY

Revision A (October 2007)

This is the initial release of this document.

© 2007 Microchip Technology Inc.