AN10695

Using the LPC288x in audio application

Rev. 01 — 12 February 2008 Application note

Document information

Info	Content
Keywords	LPC2888, LPC2880, Audio, DADC, DDAC, I2S, CGU, design rules, application software
Abstract	This application note provides design rules, application schematics and provides the steps involved in programming the LPC288x peripherals/blocks used for audio applications.

founded by Philips

Revision history

Rev	Date	Description
01	20080212	Initial version

Contact information

For additional information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Introduction

This application note provides details on how to use the LPC288x (LPC2888/LPC2880) in an audio application. This document shows design rules/schematics and the steps involved in programming the LPC288x peripherals used for audio applications.

LPC288x modules covered in this application note are Dual channel 16-bit analog to digital converter, I2S input module (DAI), I2S output module (DAO), and Dual channel 16-bit digital to analog converter.

For each module, the following sections are provided:

1. Overview

2. Application Description

    a. Design rules

    b. Application Schematic

    c. Application Software (code example only for CGU configuration)

2. Dual channel 16-bit analog to digital converter

2.1 Overview

The ADC circuitry consists of two identical 16-bit Sigma-Delta converters. In order to allow use for synchronized sampling applications, such as stereo audio, the converters are synchronized so that the two channels operate on “left” and “right” data which are sampled at the same time.

2.2 Application description

2.2.1 Design rules

AINR and AINL both need to be foreseen with a 1M pull-down resistor to ground. Both resistors need to be as close as possible to the inputs of the IC for the suppression of idle tones.

2.2.2 Application diagram

Fig 1 shows how the ADC should be connected when used. When this part is not used the supplies must be connected and the inputs can be left open. In software this part must be set in power down using the power control registers in the CGU (Clock Generation Unit).

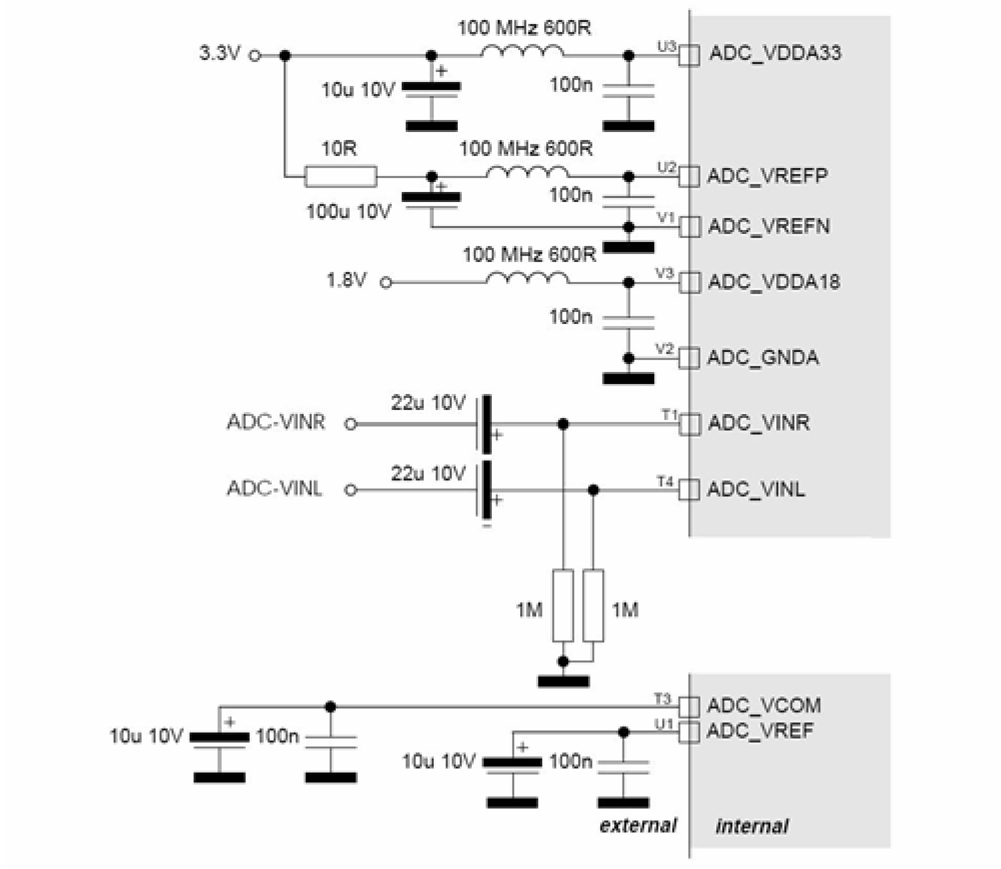

Fig 1. DADC

2.2.3 Application software

Following steps describe how to program the Dual ADC and SAI4:
(System initialization (reset) code should include the following steps if the Dual ADC and SAI4 are used in the application)

1. Write the Stream I/O Configuration register with the prescribed/fixed bits. If the DAI is used for I2S input, be sure that the DAI_OE bit is set properly for the DAI mode (see User Manual, Section 19–4.1 on page 253).
2. Program the CGU to provide 128 times the Nyquist sampling frequency for the Dual ADC and decimator, and route this to its DADC_CLK and DADC_DCLK outputs. For example, if audio with sampled at 44.1 kHz is (or will be) present on AINL and AINR, DADC_CLK and DADC_DCLK should be 5644.8 kHz.

The code example below shows how to program the CGU for the DADC.
In this example, the High Speed PLL uses the fast (12 MHz) oscillator as input and generates the 22.5792MHz. This example uses the “DAIO” selection stage and register DAIOFDCR1 as Fractional Divider register (compare to Table 7–71): The selection stage (DAIO) selects the High Speed PLL and generates the output clock (DAIO base clock, i.e. 22.5792MHz, 512fS) which is fed to all the DAIO spreading stages. The DADC CLK

and the DADC DCLK are programmed as 1/4 of the DAIO base clock by the fractional divider (DAIOFDCR1), i.e. 5.6448MHz, 128fS.

C Code

```
#include "LPC288x.h"

/* Divided by 4 */
#define DAIOFDCR1_MADD  0x48
#define DAIOFDCR1_MSUB  0xE8

/*********************************************************************
* Initialize High Speed PLL
*********************************************************************
/
HPMODE = 0x04;      /* Power down the High Speed PLL */
HPFIN = 0x01;       /* Select main oscillator as PLL's input clock */
HPNDEC = 131;       /* Refer to User Manual Table 7-41 */
HPMDEC = 1408;      /* Refer to User Manual Table 7-41 */
HPPDEC = 23;        /* Refer to User Manual Table 7-41 */
HPSELR = 8;         /* Refer to User Manual Table 7-41 */
HPSELI = 2;         /* Refer to User Manual Table 7-41 */
HPSELP = 31;        /* Refer to User Manual Table 7-41 */
HPMODE = 0x01;      /* Power up the High Speed PLL */
while ((HPSTAT & 1) == 0) {};    /* Wait for PLL to lock */

/*********************************************************************
* Configure Selection Stages
*********************************************************************
/
/* Selects High Speed PLL for “side 2” of selection stage*/
DAIOFSR2 = 0x07;
/* Enables side 2 of the stage */
DAIOSCR = 0x02;

/*********************************************************************
 * Configure Spreading Stages
********************************************************************/
/* Disable Fractional Divider in the Base Control register. */
DAIOBCR = 0;

/* Clear RUN bit in FDCR, fractional divider 1 */
DAIOFDCR1 &= ~0x01;

/* Reset FDR by setting bit 1(FDRES), set FDSTRCH bit,
MADD is 0x48(72), MSUB is 0xE8(232). where m = 0x60 and  n = 0x18.
FDR is divided by 4. */
config = (((DAIOFDCR1_MSUB << 8) | DAIOFDCR1_MADD) << 3) | (0x1 << 2)
| (0x1 << 1);
```

```
DAIOFDCR1 = config;

/* Clear Reset(FDRES) bit */
config &= ~0x2;
DAIOFDCR1 = config;

/* Set RUN bit (FDRUN) */
config |= 0x1;
DAIOFDCR1 = config;

/*********************************************************************
* Enable Select Register setting. ESR with 3 bit fields.
* Bit 0 ESR_EN is 1 causing spreading stage output clock under
* the control of the fractional divider, which runs slower
* than the selection stage clock. ESR_SEL, bits 3:1 is 1, FDR1 is * *
* selected for DDAC ESR2, DADC ESR1, DADC ESR2.
*********************************************************************
/
DADCESR1 = (0x1<<1) | (0x1<<0);
DADCESR2 = (0x1<<1) | (0x1<<0);

/*********************************************************************
* Base Control Register, set bit 0(FDRUN) to start all the FDRs.
*********************************************************************
/
DAIOBCR = 0x1;
```

3. If the PGAs are to be active initially, write the DAINCTRL register to set their starting gain.
4. Write the fixed/specified values to the DADCCTRL register, plus the Dither bits if this feature is desired.
5. Write the Decimator Control register with the desired initial values, including a 1 in the ENTIMER bit. ENTIMER disables the Decimator from sending values to SAI4 until its outputs are valid. (See Table 21–311 in User Manual which shows the delay as a function of whether the two DC blocking filters are enabled.
6. Write the SAI4 Interrupt Request register in the interrupt controller (INT_REQ19 - 0x8030 044C) to enable SAI4 interrupts at the desired priority level (See User Manual Section 9–5.1).
7. Write the SAI4 Mask register with zero (es) in the desired interrupt condition(s). For fully interrupt-driven applications, write a 0 in one of the LNMTMK, LHALFMK, or LFULMK bits. For dedicated DMA, write a 0 to LOVER to allow interrupt for overrun (which indicates an error in DMA operation or programming). For dynamically assigned DMA in Slave mode, write a 0 to LNMTMK.

Since L and R values are always loaded from the decimator into SAI4 together, there is no reason to enable both L and R interrupts. Of course the corresponding R condition(s) can be enabled instead of the L condition(s).

3. I2S input module (DAI) and I2S output module (DAO)

3.1 Overview

The LPC288x can input a single- or dual-channel audio stream from an Inter-IC Sound (I2S) bus. The I2S input module is called the DAI. It can capture serial data in standard Philips IIS format, or in right-justified 16-, 18-, 20-, or 24-bit format.

The LPC288x can output a single- or dual-channel audio stream to an I2S bus. The I2S output module is called the DAO. It can output serial data in standard Philips IIS format, or in right-justified 16-, 18-, 20-, or 24-bit format.

3.2 Application description

3.2.1 Design rules

The BCK and WS can also be used in I2S master mode, meaning that these inputs can also be switched as outputs to generate a bit clock and word select for a slave device. In this case the slave only returns the data.

3.2.2 Application diagram

Fig 2 shows how the DAI and DAO Interface should be connected when used.

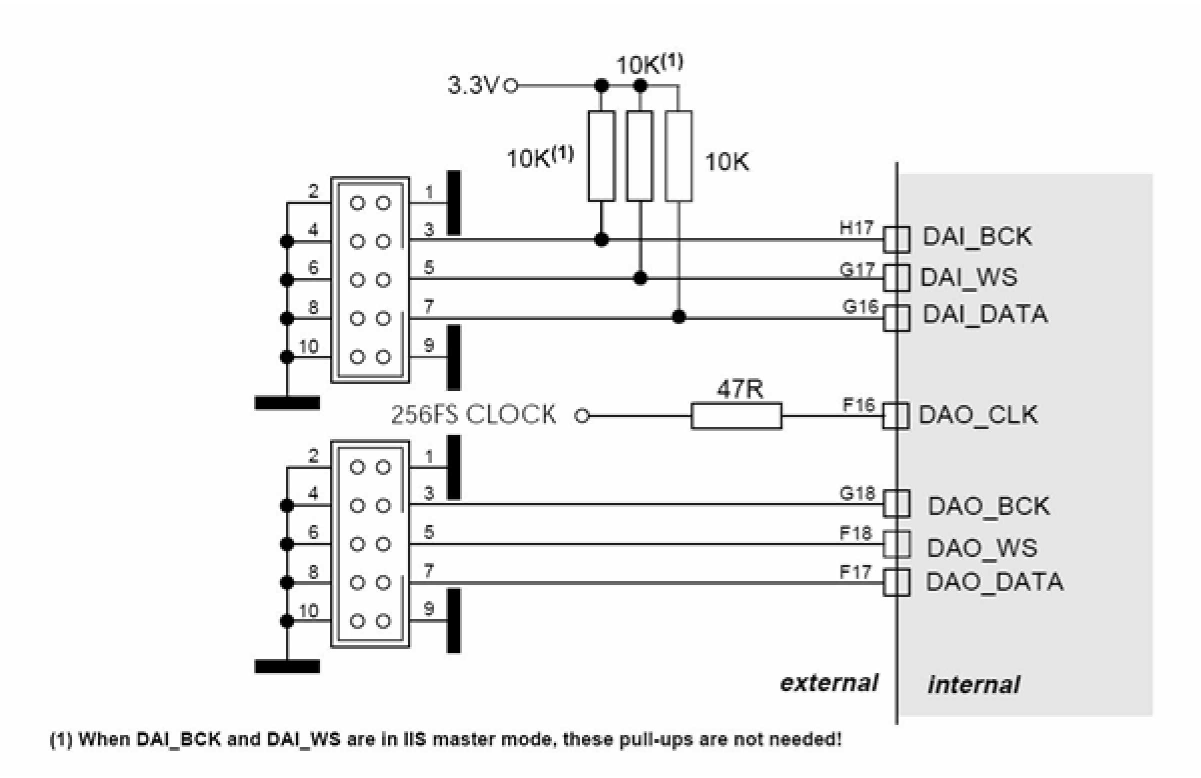

Fig 2. DAI and DAO

3.2.3 Application Software

3.2.3.1 Following steps are needed to program the DAO and SAO1

(System initialization (reset) code should include the following steps if the DAO and SAO1 are used in the application)

1. Write the desired format codes to the I2S Format register.

2. Write the Stream I/O Configuration register with the prescribed/fixed bits. If the DAI is used for I2S input, be sure that the DAI_OE bit is set properly for the DAI mode (See User Manual Section 19–4.1).

3. Program the CGU to provide the desired DAO bit clock and route it to its DAO_BCK output, and program a fractional divider to divide that bit clock by twice the number of bits per word in stretched mode, and route the fractional divider output to its DAO_WS output.

The below example will show how to program the CGU for the DAO
In this example, the High Speed PLL uses the fast (12 MHz) oscillator as input and generates the 22.5792MHz. This example uses the “DAIO” selection stage and registers DAIOFDCR3, DAIOFDCR4 and DAIOFDCR5 as Fractional Divider registers (Compare to Table 7–71 in User Manual):

The selection stage (DAIO) selects the High Speed PLL and generates the output clock (DAIO base clock, i.e. 22.5792MHz, 512fS) which is fed to all the DAIO spreading stages.

The DAO CLK is programmed as 1/2 of the DAIO base clock by the fractional divider (DAIOFDCR3), i.e. 11.2896MHz, 256fS. The DAO WS is programmed as 1/512 of the DAIO base clock by the fractional divider (DAIOFDCR4), i.e. 44.1 KHz, fS. The DAO BCK is programmed as 1/16 of the DAIO base clock by the fractional divider (DAIOFDCR5), i.e. 1.4112MHz, 32fS.

C Code

```c
#include "LPC288x.h"

/* Divide by 2 */
#define DAIOFDCR3_MADD  0x48
#define DAIOFDCR3_MSUB  0xB8

/* Divide by 512 */
#define DAIOFDCR4_MADD  0x1FF
#define DAIOFDCR4_MSUB  0x3FF

/* Divide by 16 */
#define DAIOFDCR5_MADD  0x78
#define DAIOFDCR5_MSUB  0xF8

/*********************************************************************
* Initialize High Speed PLL
*********************************************************************
*/
HPMODE = 0x04;      /* Power down the High Speed PLL */
HPFIN = 0x01;       /* Select main oscillator as PLL's input clock */
HPNDEC = 131;       /* Refer to User Manual Table 7-41 */
HPMDEC = 1408;      /* Refer to User Manual Table 7-41 */
HPPDEC = 23;        /* Refer to User Manual Table 7-41 */
HPSELR = 8;         /* Refer to User Manual Table 7-41 */
HPSELI = 2;         /* Refer to User Manual Table 7-41 */
HPSELP = 31;        /* Refer to User Manual Table 7-41 */
HPMODE = 0x01;      /* Power up the High Speed PLL */
while ((HPSTAT & 1) == 0) {};     /* Wait for PLL to lock */

/**********************************************************************
* Configure Selection Stages
**********************************************************************
/
/* Selects the High Speed PLL for “side 2” of the selection stage */
DAIOFSR2 = 0x07;
/* Enables side 2 of the stage */
DAIOSCR = 0x02;
```

```
/**********************************************************************
* Configure Spreading Stages
**********************************************************************
/
/* Disable Fractional Divider in the Base Control register. */
DAIOBCR = 0;
/* Clear RUN bit in FDCR, fractional divider 3 */
DAIOFDCR3 &= ~0x01;

/* Reset FDR by setting bit 1(FDRES), set FDSTRCH bit,
MADD is 0x48(72), MSUB is 0xB8(184). where m = 0x90 and n = 0x48. FDR
is divided by 2. */
config = (((DAIOFDCR3_MSUB << 8) | DAIOFDCR3_MADD) << 3) | (0x1 << 2)
| (0x1 << 1);
DAIOFDCR3 = config;

/* Clear Reset(FDRES) bit */
config &= ~0x2;
DAIOFDCR3 = config;

/* Set RUN bit(FDRUN). */
config |= 0x1;
DAIOFDCR3 = config;

/**********************************************************************
* Enable Select Register setting. ESR with 3 bit fields.
* Bit 0 ESR_EN is 1 causing spreading stage output clock under
* the control of the fractional divider, which runs slower than the
* selection stage clock.
* ESR_SEL, bits 3:1 is 3, FDR3 is selected for DAO ESR1.
**********************************************************************
/
DAOESR1 = (0x3<<1) | (0x1<<0);

/* Clear RUN bit in FDCR, fractional divider 4 */
DAIOFDCR4 &= ~0x01;

/* Reset FDR by setting bit 1(FDRES), set FDSTRCH bit, MADD is
0x1FF(511), MSUB is 0x3FF(1023). where m = 0x200 and n = 0x001. FDR
is divided by 512. */
config = (((DAIOFDCR4_MSUB << 10) | DAIOFDCR4_MADD) << 3) | (0x1 <<
2) | (0x1 << 1);
DAIOFDCR4 = config;

/* Clear Reset(FDRES) bit */
config &= ~0x2;
```

```
DAIOFDCR4 = config;

/* Set RUN bit(FDRUN). */
config |= 0x1;
DAIOFDCR4 = config;

/*************************************************************************
* Enable Select Register setting. ESR with 3 bit fields.
* Bit 0 ESR_EN is 1 causing spreading stage output clock under
* the control of the fractional divider, which runs slower
* than the selection stage clock. ESR_SEL, bits 3:1 is 4, FDR4 is
* selected for DA1 ESR2, DAO ESR2.
*************************************************************************
/
DAOESR2 = (0x4<<1) | (0x1<<0);
/* Clear RUN bit in FDCR, fractional divider 5 */
DAIOFDCR5 &= ~0x01;

/* Reset FDR by setting bit 1(FDRES), set FDSTRCH bit, MADD is
0x78(120), MSUB is 0xF8(248). where m = 0x80 and n = 0x08. FDR is
divided by 16.*/
config = (((DAIOFDCR5_MSUB << 8) | DAIOFDCR5_MADD) << 3) | (0x1 << 2)
| (0x1 << 1);
DAIOFDCR5 = config;

/* Clear Reset(FDRES) bit */
config &= ~0x2;
DAIOFDCR5 = config;

/* Set RUN bit(FDRUN). */
config |= 0x1;
DAIOFDCR5 = config;

/*************************************************************************
* Enable Select Register setting. ESR with 3 bit fields. bit 0
* ESR_EN is 1 causing spreading stage output clock under the control
* of the fractional divider, which runs slower than the selection
* stage clock. ESR_SEL, bits 3:1 is 5, FDR5 is selected for DAI ESR1.
*************************************************************************
/
DAOESR3 = (0x5<<1) | (0x1<<0);

/* Base Control Register, set bit 0(FDRUN) to start all the FDRs. */
DAIOBCR = 0x1;
```

4. Write the SAO1 Interrupt Request register in the interrupt controller (INT_REQ20 - 0x8030 0460) to enable SAO1 interrupts at the desired priority level (See User Manual Section 9–5.1).

5. Write the SAO1 Mask register with zero (es) in the desired interrupt condition(s). For fully interrupt-driven applications, write a 0 to the LMTMK or LHALFMK bit (or RMTMK or RHALFMK if only the R channel is used). For DMA operation, write a 0 to LUNDER and/or RUNDER to allow interrupt for under run (which indicates an error in DMA operation or programming).

Since DAO always shifts the L and R values together, except for LUNDER and RUNDER when using two DMA channels, there is no reason to enable both L and R interrupts.

3.2.3.2 Following steps are needed to program the DAI and SAI1

(System initialization (reset) code should include the following steps if the DAI and SAI1 are used in the application)

1. Write the desired format codes to the I2S Format register.

2. Write the Stream I/O Configuration register with the prescribed/fixed bits, and 1 in the DAI_OE bit for Master mode, 0 for Slave mode.

3. In Slave mode, program the High Speed PLL to take its input from either the BCKI pin or the WSI pin. If the ratio between the bit clock and the sampling frequency is known, use the BCKI pin. If not, use WSI. Particularly when using WSI, note that the HS PLL has problems locking to a frequency less than 100 kHz. In Master mode, program the HS PLL to take its input from the Main oscillator.

4. Program the CGU to provide the proper DAI clocking. In Slave mode the external I2S bit clock arrives on the BCKI pin -- program the CGU to route this clock to its DAI_XBCK output. In Master mode, program the CGU to generate the bit clock and route it to its DAI_BCKI output, and program a fractional divider to divide that bit clock by twice the number of bits per word in stretched mode, and route the fractional divider output to its DAI_WS output.

The following example shows how to program the CGU for the DAI

In this example, the DAI is programmed to operate in Slave mode. Therefore, this example uses the “DAI” selection stage (compare to Table 7–71 in User Manual):

The selection stage (DAI) selects the DAI BCK pin and generates the output clock (DAI base clock) which is fed to the DAI spreading stage (DAI BCK).

C Code

```
#include "LPC288x.h"
/*********************************************************************
* Configurate Selection Stages
*********************************************************************
/
/* Selects the DAI BCK pin for “side 2” of the selection stage */
DAIFSR2 = 0x03;
/* Enables side 2 of the stage */
DAISCR = 0x02;
```

5. Write the SAI1 Interrupt Request register in the interrupt controller (INT_REQ16 - 0x8030 0440) to enable SAI1 interrupts at the desired priority level (See User Manual Section 9–5.1).

6. Write the SAI1 Mask register with zero (es) in the desired interrupt condition(s). For fully interrupt-driven applications, write a 0 in one of the LNMTMK, LHALFMK, or LFULMK bits. For dedicated DMA, write a 0 to LOVER to allow interrupt for overrun (which indicates an error in DMA operation or programming). For dynamically assigned DMA in Slave mode, write a 0 to LNMTMK.

Since L and R values are always loaded from the DAI into SAI1 together, there is no reason to enable both L and R interrupts. Of course the corresponding R condition(s) can be enabled instead of the L condition(s).

4. Dual channel 16-bit digital to analog converter

4.1 Overview

The dual channel bit stream DAC can be used for stereo audio and other one- or two-channel D-to-A applications, particularly those involving regular, periodic conversion. The basic architecture of the block consists of an input block that receives 24-bit inputs at the Nyquist sample frequency of interest (fs), up-samples and interpolates to 128 fs using 16-bit coefficients and performs noise shaping, after which the digital results are converted to analog voltages.

4.2 Application description

4.2.1 Design rules

Because of the low rejection ratio of VREFP and VREFN the supply to these pins needs to be very clean to prevent unwanted distortion in the audio signal. One of the possibilities to do this is to provide a big capacitor between VREFP and VREFN. If even higher audio performance is needed then it is advised to provide an external LDO which is going to provide the supply for the SDAC.

4.2.2 Application diagram

Fig 3 shows how the DAC should be connected when used. The schematic shows how to connect the DAC with VREFP decoupling. The schematic also shows how to connect the DAC with an external LDO.

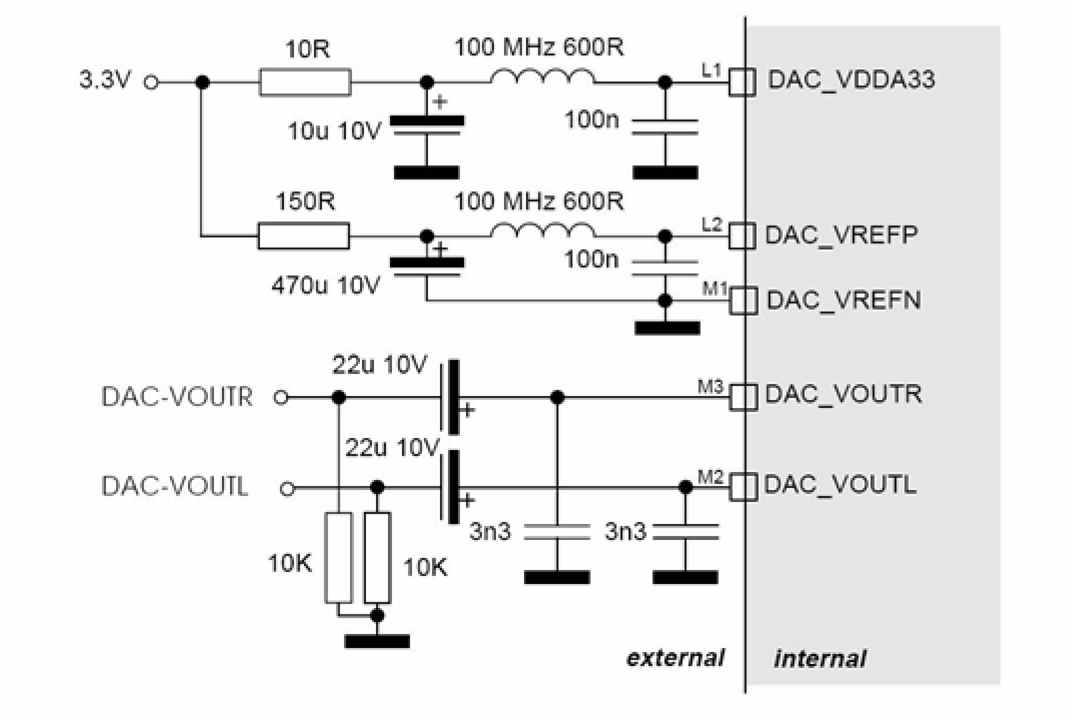

Fig 3. DDAC

4.2.3 Application software

Following steps are needed to program the DAO and SAO2

(System initialization (reset) code should include the following steps if the Dual DAC and SAO2 are used in the application)

1. Write the Stream I/O Configuration register with the prescribed/fixed bits. If the DAI is used for I2S input, be sure that the DAI_OE bit is set properly for the DAI mode (See User Manual Section 19–4.1).

2. Write the DDACCTRL and DDACSET registers with the desired values. Set PD in DDACCTRL to 1 initially, per step 1 of User Manual Section 22–6.2.

3. Program the CGU to provide the following clocks:
a. 128 fs on its DDAC_DCLK output
b. 256 fs on its DDAC_CLK output if 8 kHz ≤ fs ≤ 32 kHz and the MODE field in DDACCTRL is 00, otherwise 128 fs on DDAC_CLK.
c. fs on its DAO_WS output (this signal is used for both the DAO and the dual DAC)

Following example shows how to program the CGU for the DDAC.
In this example, the High Speed PLL uses the fast (12 MHz) oscillator as input and generates the 22.5792MHz. This example uses the “DAIO” selection stage and registers DAIOFDCR0, DAIOFDCR1 and DAIOFDCR4 as Fractional Divider registers (compare to Table 7–71 in User Manual):

The selection stage (DAIO) selects the High Speed PLL and generates the output clock (DAIO base clock, i.e. 22.5792MHz, 512fS) which is fed to all the DAIO spreading stages.

The DDAC CLK is programmed as 1/4 of the DAIO base clock by the fractional divider (DAIOFDCR0), i.e. 5.6448MHz, 128fS. The DDAC DCLK is programmed as 1/4 of the DAIO base clock by the fractional divider (DAIOFDCR1), i.e. 5.6448MHz, 128fS. The DAO WS is programmed as 1/512 of the DAIO base clock by the fractional divider (DAIOFDCR4), i.e. 44.1KHz, fS.

C Code

```
#include "LPC288x.h"

/* Divide by 4 */
#define DAIOFDCR0_MADD  0x48
#define DAIOFDCR0_MSUB  0xE8

/* Divide by 4 */
#define DAIOFDCR1_MADD  0x48
#define DAIOFDCR1_MSUB  0xE8

/* Divide by 512 */
#define DAIOFDCR4_MADD  0x1FF
#define DAIOFDCR4_MSUB  0x3FF

/*********************************************************************
* Initialize High Speed PLL
*********************************************************************
/
HPMODE = 0x04;     /* Power down the High Speed PLL */
HPFIN = 0x01;      /* Select main oscillator as PLL's input clock */
HPNDEC = 131;      /* Refer to User Manual Table 7-41 */
HPMDEC = 1408;      /* Refer to User Manual Table 7-41 */
HPPDEC = 23;       /* Refer to User Manual Table 7-41 */
HPSELR = 8;        /* Refer to User Manual Table 7-41 */

HPSELI = 2;        /* Refer to User Manual Table 7-41 */
HPSELP = 31;       /* Refer to User Manual Table 7-41 */
HPMODE = 0x01;      /* Power up the High Speed PLL */
while ((HPSTAT & 1) == 0) {};    /* Wait for PLL to lock */

/*********************************************************************
* Configure Selection Stages
*********************************************************************
/
/* Selects the High Speed PLL for “side 2” of the selection stage */
DAIOFSR2 = 0x07;
/* Enables side 2 of the stage */
```

```
DAIOSCR = 0x02;

/***********************************************************************
* Configure Spreading Stages
***********************************************************************
/

/* Disable Fractional Divider in the Base Control register. */
DAIOBCR = 0;

/* Clear RUN bit in FDCR, fractional divider 0 */
DAIOFDCR0 &= ~0x01;

/* Reset FDR by setting bit 1(FDRES), set FDSTRCH bit, MADD is
0x48(72), MSUB is 0xE8(232). where m = 0x60 and n = 0x18. FDR is
divided by 4. */
config = (((DAIOFDCR0_MSUB << 8) | DAIOFDCR0_MADD) << 3) | (0x1 << 2)
| (0x1 << 1);
DAIOFDCR0 = config;

/* Clear Reset(FDRES) bit */
config &= ~0x2;
DAIOFDCR0 = config;

/* Set RUN bit(FDRUN). */
config |= 0x1;
DAIOFDCR0 = config;

/***********************************************************************
* Enable Select Register setting. ESR with 3 bit fields.
* Bit 0ESR_EN is 1 causing spreading stage output clock under the
* control of the fractional divider, which runs slower
* than the selection stage clock. ESR_SEL, bits 3:1 is 0, FDR0 is
* selected for DDAC ESR1.

***********************************************************************
/
DDACESR1 = (0x0<<1) | (0x1<<0);

/* Clear RUN bit in FDCR, fractional divider 1 */
DAIOFDCR1 &= ~0x01;

/* Reset FDR by setting bit 1(FDRES), set FDSTRCH bit, MADD is
0x48(72), MSUB is 0xE8(232). where m = 0x60 and n = 0x18. FDR is
divided by 4. */
config = (((DAIOFDCR1_MSUB << 8) | DAIOFDCR1_MADD) << 3) | (0x1 << 2)
```

```
| (0x1 << 1);
DAIOFDCR1 = config;

/* Clear Reset(FDRES) bit */
config &= ~0x2;
DAIOFDCR1 = config;

/* Set RUN bit(FDRUN). */
config |= 0x1;
DAIOFDCR1 = config;

/*********************************************************************
* Enable Select Register setting. ESR with 3 bit fields. Bit 0 ESR_EN
* is 1 causing spreading stage output clock under the control of the
* fractional divider, which runs slower than the selection stage
* clock. ESR_SEL, bits 3:1 is 1, FDR1 is selected for DDAC ESR2, DADC
* ESR1, DADC ESR2.
*********************************************************************
*/
DDACESR2 = (0x1<<1) | (0x1<<0);

/* Clear RUN bit in FDCR, fractional divider 4 */
DAIOFDCR4 &= ~0x01;

/* Reset FDR by setting bit 1(FDRES), set FDSTRCH bit, MADD is
0x1FF(511), MSUB is 0x3FF(1023). where m = 0x200 and n = 0x001. FDR
is divided by 512. */
config = (((DAIOFDCR4_MSUB << 10) | DAIOFDCR4_MADD) << 3) | (0x1 <<
2) | (0x1 << 1);
DAIOFDCR4 = config;

/* Clear Reset(FDRES) bit */
config &= ~0x2;
DAIOFDCR4 = config;


/* Set RUN bit(FDRUN). */
config |= 0x1;
DAIOFDCR4 = config;

/*********************************************************************
* Enable Select Register setting. ESR with 3 bit fields.
* Bit 0 ESR_EN is 1 causing spreading stage output clock under the
* control of the fractional divider, which runs slower than the
* selection stage clock. ESR_SEL, bits 3:1 is 4, FDR4 is
* selected for DA1 ESR2, DAO ESR2.
```

```
*********************************************************************
/
DAOESR2 = (0x4<<1) | (0x1<<0);

/* Base Control Register, set bit 0(FDRUN) to start all the FDRs. */
DAIOBCR = 0x1;
```

4. Perform the power-up procedure described in User Manual Section 22–6.2.

5. Write the SAO2 Interrupt Request register in the interrupt controller (INT_REQ21 - 0x8030 0464) to enable SAO2 interrupts at the desired priority level (see User Manual Section 9–5.1).

6. Write the SAO2 Mask register with zero (es) in the desired interrupt condition(s). For fully interrupt-driven applications, write a 0 to the LMTMK or LHALFMK bit (or RMTMK or RHALFMK if only the R channel is used). For DMA operation, write a 0 to LUNDER and/or RUNDER to allow interrupt for under run (which indicates an error in DMA operation or programming).

Since L and R values are removed from the SAO simultaneously, except for LUNDER and RUNDER when using two DMA channels, there is no reason to enable both L and R interrupts.

1) Legal information

a. Definitions

b. Disclaimers

c. Trademarks

5. Contents