Technical notes on using Analog Devices DSPs, processors and development tools
Contact our technical support at dsp.support@analog.com and at dsptools.support@analog.com
Or visit our on-line resources http://www.analog.com/ee-notes and http://www.analog.com/processors

Interfacing the ADSP-21262 SHARC® EZ-KIT Lite™ Boards to High-Speed Converter Evaluation Boards

Contributed by R. Murphy and R. Gentile

Rev 1 – January 23, 2004

Introduction

This Engineer-to-Engineer Note describes the use of the Parallel Data Acquisition Port (PDAP) on the ADSP-21262 SHARC® DSP and the use of the EZ-KIT Extender card to interface an ADSP-21262 EZ-KIT Lite™ board to evaluation boards of high-speed converters.

Representatively, physical connection to an AD9244-EB Evaluation Board is described, as well as the configuration of the SHARC DSP. This project was implemented using an ADSP-21262 EZ-KIT Lite board, a SHARC EZ-KIT Extender card, an AD9244 High-Speed Converter Evaluation Board, and the VisualDSP++® 3.0 SP1 development environment.

Background

In the 1970's and 80's, high-speed mixed-signal designs were often constrained by digital (not analog) circuitry limitations. Highspeed (>10MSPS) parallel converters, for example, have been available from industry leaders like Analog Devices, Inc., since the 1970's.

More and more applications are demanding intensive real-time algorithms. In addition, higher sample rates (14 bits at greater than 50MSPS) are available from analog-to-digital converters (ADCs) and digital-to-analog converters (DACs). In addition to the higher sample rates are vast decreases in power consumption, exemplified in the AD9244, which consumes only 590 mW at its full sample rate of 65 MSPS. These factors mandate faster programmable general-purpose (GP) digital signal processors (DSPs) to handle the challenges presented by these high-speed designs.

Until recently, most designers were forced to interface high-speed parallel converters to application specific ICs (ASICs) or fast field programmable gate arrays (FPGAs).

Advantages of using a GP DSP

One of the biggest advantages of general-purpose programmable processors is that these solutions typically cost less than their closest digital processing counterparts (FPGAs and ASICs). Additionally, general-purpose DSP design cycles are much shorter, allowing faster time to market. Some companies must hire (or consult with) professionals who specialize in FPGA/ASIC design. Companies may even be forced to send their intellectual property (IP) out of house, involving risks in confidentiality (hardware, firmware, and software). On the other hand, general-purpose DSP code can be placed onto off-chip ROM or masked onto a DSP, like the ADSP-21262, further protecting intellectual property.General-purpose processors are also fully programmable, unlike ASICs, which require costly redesign (time and money). These factors drive many engineers to consider general-purpose DSPs as the solution of choice, even more so as DSPs approach "Pentium class" core rates.

The processing bandwidth needed depends on the processor's interface capabilities, which in turn, are influenced by several factors including: block processing versus sample processing, the existence of a Direct Memory Access (DMA) controller, multi-ported memory, and the use of external FIFOs.

Fortunately, Analog Devices' 32-bit SHARC family of DSPs boasts a zero-overhead DMA controller as well as dual-ported on-chip SRAM. Similar to the

ADSP-2116x DSP family, the ADSP-21262 is a SIMD processor, and it runs at a core instruction rate of 200 MHz (5ns). The SIMD capability effectively doubles the data processing bandwidth of the processor, providing 1200 MFLOPs of processing power. The combination of core speed, an independent DMA controller, and a large dual-ported on-chip memory (up to 2 Mbits of SRAM and 4 Mbits of ROM) enable the ADSP-21262 to perform efficient block processing at high data rates.

Hardware Interface

AD9244 HSC

The high-speed converter in this example is the AD9244 (Figure 1). It is a 14-bit, 65 MSPS, analog-to-digital converter with an on-chip, high-performance sample-and-hold amplifier. At half the power dissipation (only 590 mW) of any competitor's ADC, the AD9244 is highly desirable for power-sensitive applications, such as wireless communications subsystems (Microcell, Picocell), medical imaging systems, ultrasound equipment, portable and battery-operated instrumentation, high-resolution CCD imaging, and IF digitizing applications.

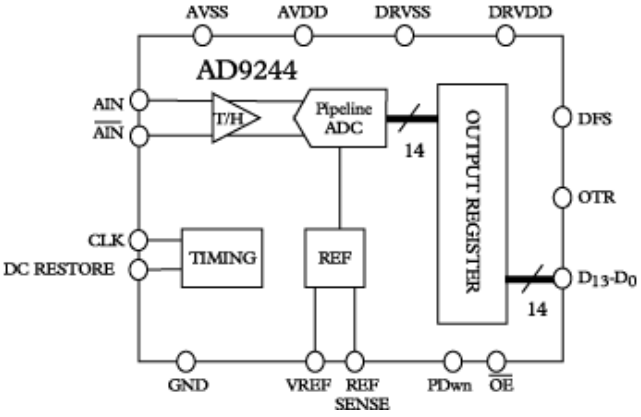

Figure 1. AD9244 Block Diagram

The AD9244 includes an on-board, programmable voltage reference. Alternatively, an external reference can be used to suit the DC accuracy and temperature drift requirements of the application. The AD9244 evalution board has circuitry to satisfy this requirement. A differential or single-ended clock input is used to control all internal conversion cycles, and the AD9244 evaluation board has provisions for several methods of providing this clock source. This clock will also be used to control the PDAP interface on the ADSP-21262. The digital output is hardware-selectable and can be presented in straight binary or in twos complement format. An out-of-range (OTR) signal indicates an overflow condition that can be used with the most significant bit to determine low or high overflow.

The AD9244 evaluation board (AD9244-65PCB) allows designers to validate the AD9244 in their system. The ADSP-21262 SHARC DSP is also available as an evaluation board (ADDS-21262-EZLITE). Analog Devices has developed an EZ-KIT Extender Card (Figure 3) to provide designers with an easy solution to evaluate the ADSP-21262 and AD9244 together,.

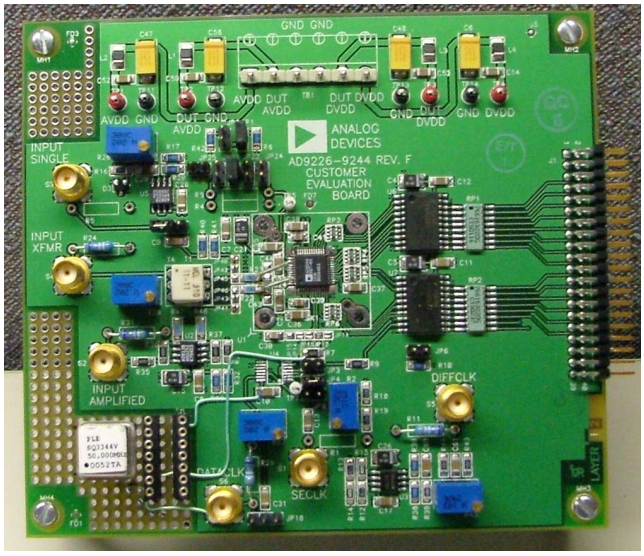

Figure 2. AD9244 ADC Evaluation Board

AD9244-65PCB Configuration

The AD9244 evaluation board provides several options, permitting multiple combinations of data inputs and outputs and clock inputs. The configuration used for this project is included below. See Reference [5] for more information.

DFS: Digital Format Select (JP2) Open = straight binary; Closed = twos complement

DRVDD: Output Drive. The AD9244 output drivers can be configured to interface with 5-V or 3.3-V logic families by setting DRVDD to 5V or 3.3V, respectively.

OEB: Digital Output Enable. (High = digital output enabled; Low = high impedance state)

OTR: Out of Range. High = analog input voltage exceeds analog input range. The out-of-range signal is not used in this project. Optionally, it can be ANDed with the MSB and its complement to detect underrange and overrange conditions.

Clock Input Modes: Several modes of operation for the AD9244 clock interface are available, such asdifferential clock input (AC- or DC-coupled) orsingle-ended clock inputs (AC- or DC-coupled).

Analog Input Configuration

The AD9244 evaluation board allows analog inputs to be driven:

- Differentially through a transformer
- Via an AD8138 high-speed differential amplifier
- Directly (single-ended)

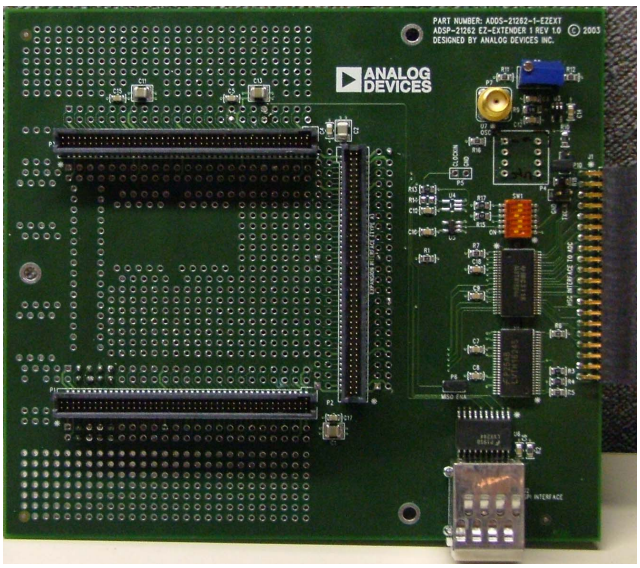

Figure 3. SHARC EZ-KIT Lite Extender Card

EZ-KIT Lite Extender Card

This card, which connects to the EZ-KIT Lite board's Expansion Interface, provides a path to the PDAP for the ADC, with a few extras. The extender card provides an interface between the ADSP-21262 EZ-KIT Lite and a host of high-speed converter ADC and DAC evaluation boards, providing debug access to all signals. The SPI-compatible port of the ADSP-21262 is routed to this card as well, supporting SPI-configurable high-speed converter products (The AD9244 does not use this). To maintain signal integrity over multiple PCBs, a buffer has been used for all data signals; a GPIO pin can enable/disable this buffer. The board also provides an extensive breadboard area as well as clock-squaring circuitry and space to install a crystal oscillator. The PDAP peripheral of the ADSP-21262 SHARC DSP has the capability to use either the DAI pins or the Parallel Port pins (AD15-0) as its data bus. For greater flexibility, the Parallel Port pins are routed to the AD9244 evaluation boards via the Extender Card. This project uses the PDAP with the Parallel Port pins and provides the option to connect the Parallel Port to a parallel device using the extender card.

ADSP-21262 SHARC DSP

The ADSP-21262 SHARC DSP provides two parallel peripherals that are useful when connecting to parallel ADCs: the Parallel Port and the Parallel Data Acquisition Port (PDAP).

The Parallel Port is an asynchronous 16-bit bi-directional interface, providing an interface to inexpensive external SRAM. Its maximum throughput is 66 MB/s. The Parellel Port is used with the SHARC DSP's zero-overhead DMA controller, and the address and data pins are multiplexed., The external DMA address index, count, and modify registers can be configured such that the index register is not incremented, allowing the Parallel Port to receive data from the ADC and place it in memory.

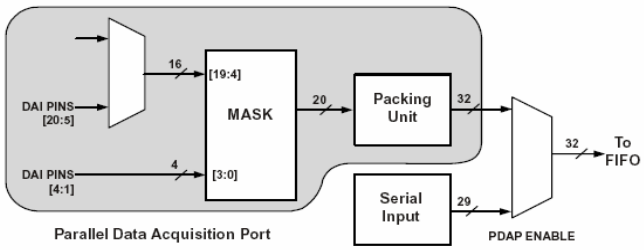

Figure 4. ADSP-21262 PDAP

The PDAP interface (Figure 4) uses one channel of the Input Data Port and the 16 DAI pins or the Parallel Port's address pins to bring data into memory. The PDAP is also used with the DMA controller, but is an input only.In addition to its data pins, the PDAP has two control inputs: PDAP_HOLD and PDAP_CLK. Use PDAP_HOLD to pause the input buffer in situations where the data is invalid, or

use it with an ADC's 'out-of-range signal. The PDAP_CLK signal is an input, and the ADC data must be sent synchronously to this clock. Since the AD9244 requires a clock source, this project will use the PDAP and clock it with the same clock that supplies the AD9244. The data pins can be routed through the Parallel Port or the DAI; if routed through the DAI, they must be used with DAI_P[20:5]. Whether using the data pins on the Parallel Port or the DAI, the control signals (PDAP_CLK, PDAP_HOLD) must be programmed to use one of DAI_P[1:4]. The PDAP_CLK signal has a maximum rate of ¼ core clock, or 50 MHz.

Note: Using the PDAP (with either set of pins) prevents the use of IDP channel 0.

The PDAP can be configured to a specific application. You can mask out (ignore) specific bits, which is useful in applications where particular bits of the data stream communicate status and/or data validity. You can mask out bits like these in real-time as opposed to spending processor time to massage the data after acquisition. The PDAP contains a packing unit, allowing several methods to efficiently correlate the acquired data with the 32-bit internal memory space. Data can be packed in 20-, 32-, 16-, 10-, or 11-bit methods. More information on packing is available in chapter 6 of the ADSP-2126x SHARC DSP Peripherals Manual. [Reference 2].

You can select the clock edge on which the data is sampled. An additional signal (PDAP Output Strobe) on the PDAP is asserted when the packing unit receives a particular number of words (dependent on the packing mode); this signal (within the DAI) is intended to be configured to generate an interrupt. It may also be configured as an output and routed to an external DAI pin. (For more information, see Reference [2].)

Software Example

The attached code example is intended as a starting point. It demonstrates how to configure the DAI, how to use DMA with the PDAP peripheral, and how to manipulate the data. Improvements can be made by implementing DMA chaining, and by implementing a block-processing protocol. The example code is included in the appendix.

A GPIO flag is used to enable and disable the buffer on the extender card. This pin must be enabled as an output and cleared.

DMA will be used to move the data from the PDAP FIFO into internal memory.

PDAP

The PDAP can optionally mask out bits in the data stream. This provides a simple method to remove status and format information from raw data. The AD9244 does not provide these status and format bits, so this project masks them out, allowing only 14 data bits to pass through.

The example code empties the 8-deep FIFO 100 times, allowing enough data to represent the input signal to be brought into memory. Ideally, a robust processing routine might use two or more blocks of memory, allowing the DMA controller to fill one block while the core processes the other block. Below are some basic instructions to display the data in a plot of internal memory.

After loading and running the .DXE file, the memory buffer will be filled with sample data. This can now be displayed graphically in a plot window in VisualDSP++:

1. From the View menu, choose Debug Windows, Plot, and then New.
2. Under 'Data Setting' click 'Browse' and select the label 'OutBuffer' located at 0xC0000 in internal memory.
3. Specify these options:Count = ~200, Stride = 1, Data = char.
4. Click 'Add' and then click 'OK'.

Summary

With slower sampling rates such as 10-20MS/s, the AD9244 high-speed ADC is a very useful accessory for the ADSP-21262. To sample at ADC's full rate, more throughput is needed to budget a realistic number of clock cycles for data processing. The two parts are well paired from a power-consumption perspective, consuming 600 mW (typical) for the ADSP-21262 DSP and 640 mW for the AD9244 ADC. (See EE-216 and the AD9244 ADC data sheet for more information.)

This application will have more effective throughput in SHARC processors in the ADSP-2136x family,

which feature higher core clock rates (300 MHz core). Although the peripheral clock in the new SHARC family is slower, (150 MHz/4 vs. 200 MHz/4), more processing bandwidth is available in the faster core.

References

[1] *ADSP-2126x SHARC DSP Core Manual*, Revision 1.0, November 2003. Analog Devices, Inc.

[2] *ADSP-2126x SHARC DSP Peripherals Manual*, Revision 1.0, December 2003. Analog Devices, Inc.

[3] *ADSP-21262 EZ-KIT Lite® Evaluation System Manual*, Revision 1.0, October 2003. Analog Devices, Inc.

[4] *ADSP-21262 SHARC Processor Preliminary Data Sheet*, Revision PrB, August 2003. Analog Devices, Inc.

[5] *AD9244 A/D Converter Data Sheet*, Revision A, June 2003. Analog Devices, Inc.

[6] *Estimating Power Dissipation for ADSP-21262S SHARC® DSPs (EE-216)*, December 2003, Analog Devices, Inc.

Appendix

HSC.asm

```
/* ********************************************************************************
 *
 * Copyright (c) 2003 Analog Devices Inc.  All rights reserved.
 *
 * ********************************************************************************/

#include "def21266.h"
#include "SRU.h"

#define MAX_STATES 0xFF
#define INTERNAL_MEM_ADDRESS 0xC0000
.global main;
.global IDP_ISR;

.section/dm seg_dmda;
.var OutBuffer[500];

.section/pm seg_pmco;
main:
    IRPTL=0x0;                      // clear all latched interrupts
    bit set IMASK DAIHI;            // enable hi-priority DAI interrupt in core
                                    // interrupt register
    bit set MODE1 CBUFEN;           // enable circular buffering

    r0 = 0x000FFFFF;
    dm(DAI_PIN_PULLUP) = r0;        // pullup un-used DAI pins

    bit set flags FLG2O;            // initialize flag2, which is routed to the /OE
                                    // pin on the buffer on the Ez-Kit-Extender
    bit set FLAGS FLG2;
    ustat2 = dm(IDP_CTL);           // Reset the IDP by enabling...
    bit set ustat2 IDP_EN;
    dm(IDP_CTL) = ustat2;
    bit clr ustat2 IDP_EN;          // ...and then disabling it.
    dm(IDP_CTL) = ustat2;

//setup for DMA-driven data handling FIFO --> Internal memory
r9=INTERNAL_MEM_ADDRESS;

    dm(IDP_DMA_I0)=r9;              //normal-word alias of Outbuffer
    r0= 1;                          dm(IDP_DMA_M0)=r0;
    r0= 8;                          dm(IDP_DMA_C0)=r0;//FIFO is 8-deep x32

    ustat2=
             //IDP_PP_PACKING2|     //2 16-bit words per 32-bit location in fifo
             IDP_PP_PACKING3|       //one 20-bit word per 32-bit location in fifo
                                    //left-aligned
             IDP_PP_SELECT|         // AD[15-0] if set, if cleared use DAI_P[20-5]
             IDP_P20_PPMASK| //Bits in the data buffer can be masked out.
             IDP_P19_PPMASK| //clr=masked
             IDP_P18_PPMASK| //set=unmasked
             IDP_P17_PPMASK|
```

```
            IDP_P16_PPMASK|
            IDP_P15_PPMASK|
            IDP_P14_PPMASK|
            IDP_P13_PPMASK|
            IDP_P12_PPMASK|
            IDP_P11_PPMASK|
            IDP_P10_PPMASK|
            IDP_P09_PPMASK|
            IDP_P08_PPMASK|
            IDP_P07_PPMASK|
            IDP_PP_CLKEDGE;       //latch data in falling edge of the clock that
                                  //is provided to the PDAP
    dm(IDP_PP_CTL) = ustat2;
    ustat2 = IDP_DMA0_INT ;
    dm(DAI_IRPTL_PRI)=ustat2;     //unmask individual interrupt for DMA_INT (PDAP)
                                  //in RIC
    dm(DAI_IRPTL_RE)=ustat2;      //PDAP interrupts latch on the rising edge only

//Following are two macros that setup the Signal Routing Unit (SRU) to configure
//the two pins we'll be using there, PDAP_CLK & PDAP_HOLD. The data pins in this
//case are routed through the parallel ports AD15-0 pins, but could be routed via
//the SRU alternatively.

    //Hold
    SRU(LOW,DAI_PB01_I);
    SRU(DAI_PB01_O, IDP0_FS_I);
    SRU(LOW,PBEN01_I);

    //Clk
    SRU(LOW,DAI_PB02_I);
    SRU(DAI_PB02_O, IDP0_CLK_I);
    SRU(LOW,PBEN02_I);

    i0=0xC0000;//normal-word address of buffer
    m0=2;//2 in SIMD, 1 in SISD
    l0=0;
    i8=0xC0000;//normal-word address of buffer
    m8=2;
    l8=0;

    ustat2 = dm(IDP_PP_CTL);
    bit set ustat2 IDP_PDAP_EN;    //PDAP if set, IDP channel 0 if cleared
    dm(IDP_PP_CTL) = ustat2;

    ustat2 = dm(IDP_CTL);    // Start the IDP
    bit set ustat2 IDP_EN;
    dm(IDP_CTL) = ustat2;

    bit clr FLAGS FLG2;      //flg2 is routed to /OE for the buffer on the Extender
                             //card.
    bit set MODE1 IRPTEN;    //Enable interrupts globally

    lcntr = 0x100, do fill_mem until lce;
    r0= 8; dm(IDP_DMA_C0)=r0;//FIFO is 8-deep x32, re-init count
    ustat2  = dm(IDP_CTL);
    bit set ustat2 IDP_DMA_EN;
    dm(IDP_CTL) = ustat2;    //This will kick off the DMA of the IDP buffer to
                             //Internal memory
```

```
    idle; //wait for dma completion interrupt;
    call shift_bits;
fill_mem: nop;nop;

main.end:
jump main.end;

shift_bits:
bit set Mode1 PEYEN;         //Enable SIMD to process data in PEx & PEy
lcntr=8, do shift until lce; //Each iteration of the loop will process one
                             // buffer's worth of data
/* in packing mode 3, the data is stored in the buffer like this:
    1|HHHHHxxx|
    2|GGGGGxxx|
    3|FFFFFxxx|
    4|EEEEExxx|
    5|DDDDDxxx|
    6|CCCCCxxx|
    7|BBBBBxxx|
    8|AAAAAxxx|
*/
    r0 = dm(i0,m0);
    r0 = rot r0 by -18;      // sample is now in R1, but needs to be
                             // shifted right (rotated)
                //i.e. before r0 = 0x 1111 1111 1111 11xx xxxx (x = don't care)
                //     after  r0 = 0x xxxx xx11 1111 1111 1111 (x = don't care)
shift: pm(i8,m8) = r0;
        bit clr Mode1 PEYEN;
        rts;

IDP_ISR://This interrupt indicates that the current DMA has completed
//test for IDP_DMA0_INT (Read of DAI_IRPTL clears latched interrupt)

        r0=dm(DAI_IRPTL_H);
        btst r0 by 10;
        if not SZ call dma_again;//SZ flag cleared if tested bit =1;
        rti;
dma_again:
    ustat2  = dm(IDP_CTL);
    bit clr ustat2 IDP_DMA_EN; //disable DMA
    dm(IDP_CTL) = ustat2;
    rts;
IDP_ISR.end:
```

Listing 1. main.asm

Document History

Version	Description
Rev 1 – January 23, 2004 *by R. Murphy*	Initial Release