Application Report
SLAA213–September 2004

Interfacing the DAC7654 to the MSP430F449

Jojo Parguian
Data Acquisition - Digital/Analog Converters

ABSTRACT

This application report shows how easy it is to interface the DAC7654 digital-to-analog converter to the MSP430F449 mixed signal microcontroller using the SPI mode of serial communication. By using the software program provided in this application report, a simple routine can be created to generate a sinusoidal waveform. Using the DAC7654 EVM along with the HPA449 evaluation system makes developing the interface even easier.

Contents

List of Figures

1 Introduction

The DAC7654 is a quad-channel, low-power, 16-bit resolution, voltage-output, digital-to-analog converter (DAC), which features low-glitch, 16-bit monotonicity with double-buffered serial interface. The double-buffered register architecture is implemented to allow simultaneous updating of all DACs while writing new data to each input registers. The communication port accepts 24-bit serial input data, which is interfaced with the MSP430F449 using the SPI protocol. The DAC's digital circuit is powered by V_{DD} = +5 V, whereas the digital logic is powered by IO_{VDD} = +3.3 V to match the MSP430's logic voltage. The DAC's analog V_{CC} supply can range from a minimum of +4.75 V to a maximum of +5.25 V, whereas V_{SS} ranges from a minimum of –5.25 V to a maximum of –4.75 V for bipolar mode operation. If unipolar mode operation is desired, a single supply of +5 V on V_{CC} should be applied and V_{SS} should be connected to AGND. This application report covers only the bipolar mode of operation.

2 Hardware Setup Configuration

This application report is based on an experiment using the HPA449 platform for the MSP430F449 and the DAC7654 EVM revision A. Once the HPA449 and the DAC7654 EVM are configured properly, they can easily be connected. Figure 1 and Figure 2 show the hardware configuration setup for both the DAC7654 EVM and the HPA449 boards.

The HPA449 comes from the factory configured with the correct jumper settings. The only requirement is to add another jumper on J30 pins 3 and 4 to route the correct signal for the RSTSEL pin function of the DAC7654 EVM (see Figure 1).

The hardware setup configuration for the DAC7654 EVM (shown in Figure 2) depicts the simple diagram of the interface connection between the DAC7654 and the MSP430F449 (shown in Figure 3).

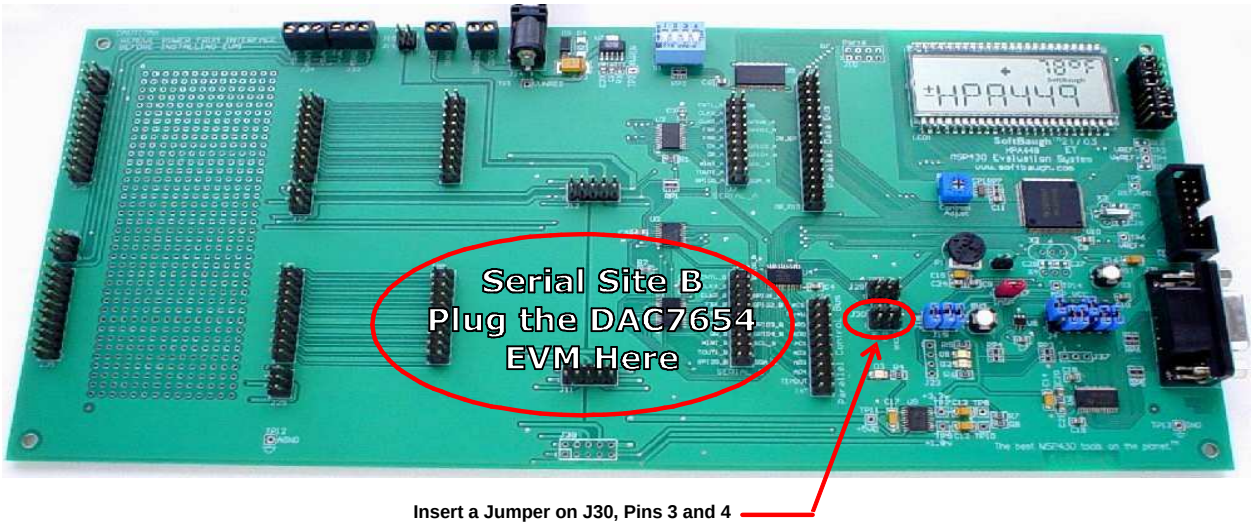

Figure 1. HPA449 Hardware Configuration

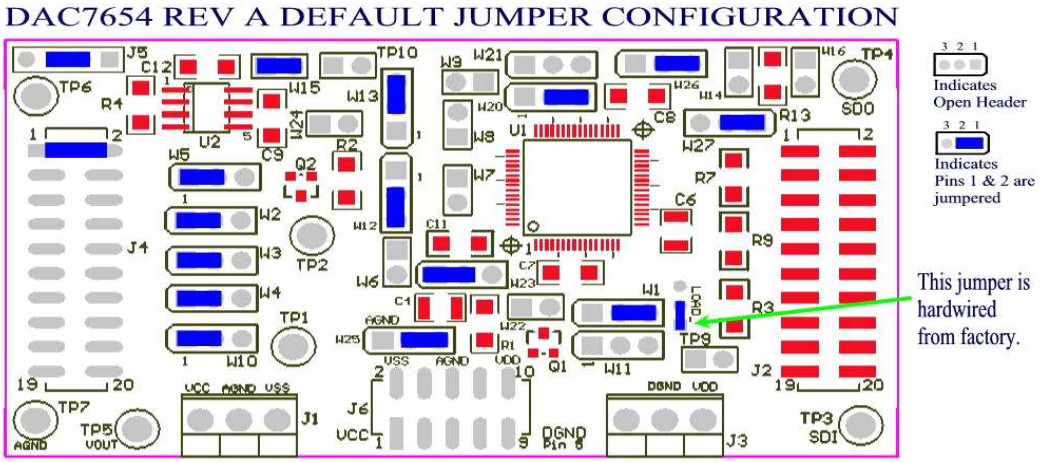

Figure 2. DAC7654 EVM Hardware Configuration

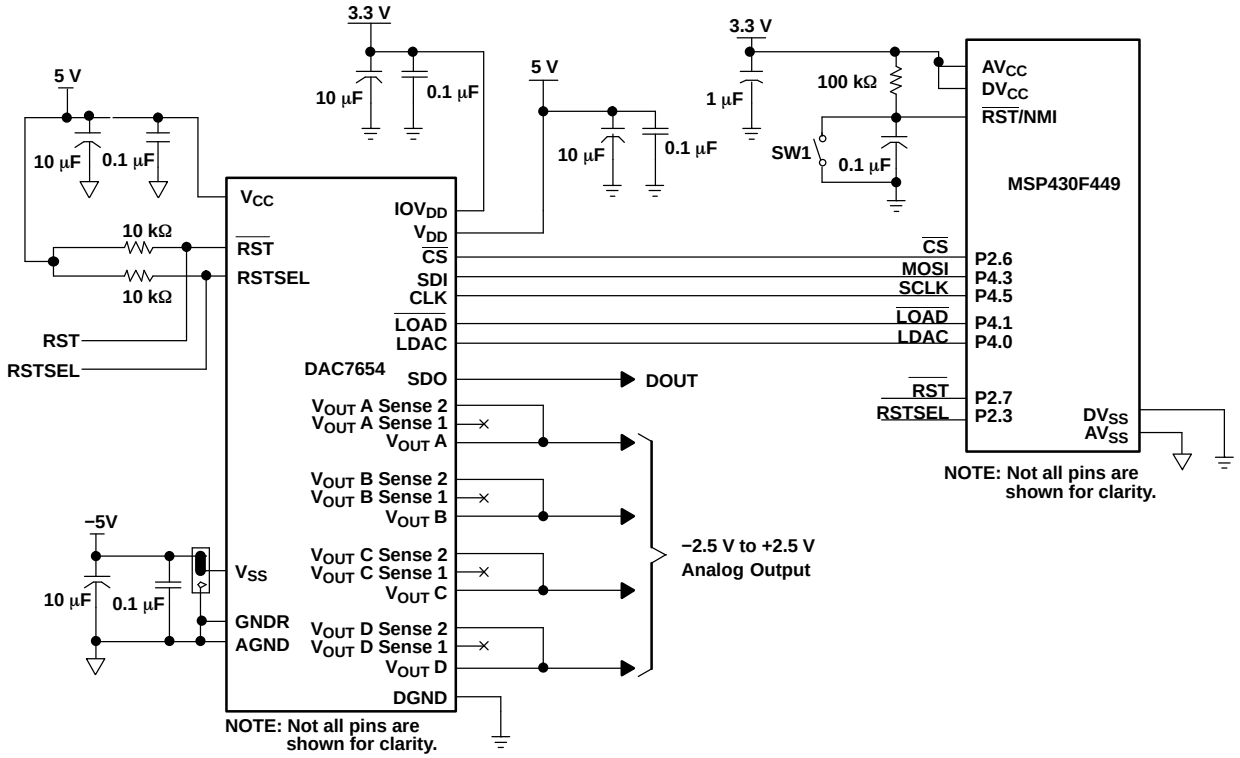

Figure 3. MSP430 and DAC7654 Circuit Diagram

2.1 Principles of Operation

The MSP430F449 microcontroller interfaces with the DAC7654 using the SPI serial data communication protocol via the MSP430 microcontroller's USART1 port. Only two pins of the four-pin SPI mode of configuration are used. This implementation makes it unneccessary to read any data back from the DAC7654 or to have the MSP430 microcontroller be slaved by another host peripheral for SPI purposes. Therefore, the STE and the MISO functions in SPI mode of the USART1 port are not used.

The $\overline{CS}$ function is particularly useful when multiple devices share the SPI bus. Although the DAC7654 is the only device the MSP430 microcontroller interfaces with, the $\overline{CS}$ function is still implemented to show some useful timing relationships. This function is accomplished using a GPIO pin, P2.6, to enable the serial communication. The $\overline{RST}$ and RSTSEL pins are connected to P2.7 and P2.3 of the MSP430F449, respectively. These GPIO pins are set high in the software program at all times so that when the DAC7654 goes into power up, the outputs go to the mid-scale code of 8000h and keep the reset function in non-reset mode. See the DAC7654 data sheet SBAS263 for further information regarding the reset function. The GPIO pins, P4.1 and P4.0, control the $\overline{LOAD}$ and LDAC pins, respectively, to load the data into the DAC input register and to update the DAC registers.

The DAC7654 receives a 24-bit digital input word serially. Because the SPI only provides eight data clocks per transmission, three write cycles are required to give it a burst of eight SCLK cycles grouped into three within a $\overline{CS}$ low period (see Figure 4). Starting from the MSB, the first two bits (B23 and B22) contain the address bits that select one of four DAC input registers. The selected register is updated when the $\overline{LOAD}$ signal goes low. The third bit (B21) following the address bits is the *Quick Load* bit; when this bit is high, the serial data word in the DAC shift register is loaded into all of the DAC input registers when the $\overline{LOAD}$ signal goes low. If this bit is low, only the selected DAC input register is updated as previously described. Five unused bits follow the quick-load bit (B21), and the last sixteen bits [B15:B0] compose the DAC digital word with the most significant bit first.

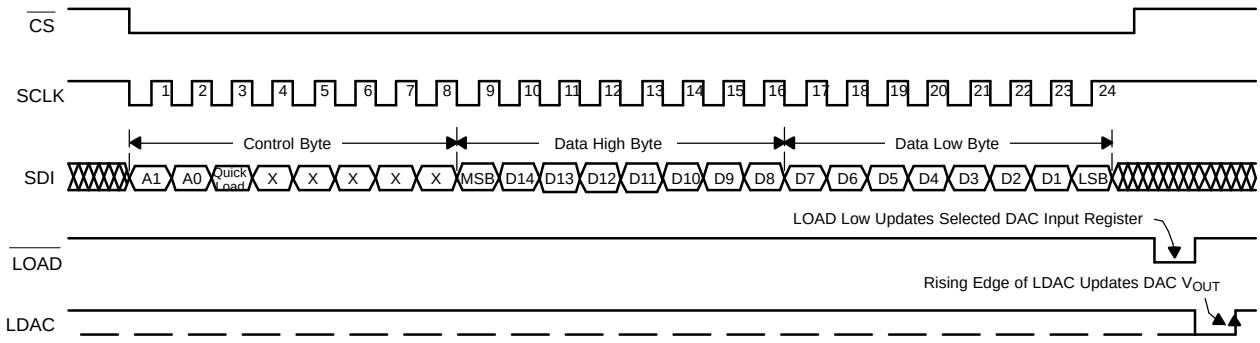

Figure 4. DAC7654 Serial Interface Timing

Because either the rising edge of SCLK or $\overline{CS}$ of the DAC7654 can clock the data in, they are completely interchangeable. Therefore, consider the way the $\overline{CS}$ and SCLK are treated. As previously noted, this application report deals only with how the $\overline{CS}$ signal enables and disables the serial communication. Hence, the SCLK should idle high and only clock in data when $\overline{CS}$ is low (see Figure 4). This requires that the SPI mode be configured for CPOL = 1 and CPHA = 1.

In addition, if the $\overline{CS}$ signal is kept active (i.e., $\overline{CS}$ = 0) all the time, providing it is the only device in the bus, the DAC7654 works properly with the SCLK, generating 24 clock pulses for every data transfer. In this case, the SPI mode needs to be configured for CPOL = 0 and CPHA = 0.

The DAC7654 features a double-buffered architecture to allow new data to be written to each of the DAC registers without disturbing the analog outputs. The first sets of registers are the DAC input registers, which are level-triggered via the $\overline{LOAD}$ signal. If the serial data are clocked into the serial shift register while $\overline{LOAD}$ is low, the DAC registers that are accidentally selected change as the shift register bits flow through A1 and A0. Therefore, the $\overline{LOAD}$ signal must be high during the write cycle to avoid corrupting the DAC input registers that are accidentally selected and to prevent the DAC input registers from becoming transparent to the shift register.

The second sets of registers are the DAC output registers, which are edge-triggered via the LDAC signal. When the LDAC signal is transitioned from low to high, the digital word currently in the DAC input registers is latched into the DAC output registers, and all the DAC outputs are updated simultaneously.

3 Generating the Sine-wave Output

The actual timing diagram of the SPI serial interface is shown in Figure 5. Channel 1 shows the SCLK running at approximately 4 MHz while channel 2 shows the SDI transmitting the 24-bit control and data word with the quick-load bit set for all DAC registers to be loaded with the same data. Channel 3 is the $\overline{CS}$ signal, and channel 4 is the $\overline{LOAD}$ signal. The LDAC signal is not shown in this figure because the oscilloscope channels are limited to four. Figure 6 shows the LDAC asserted following the $\overline{LOAD}$ signal.

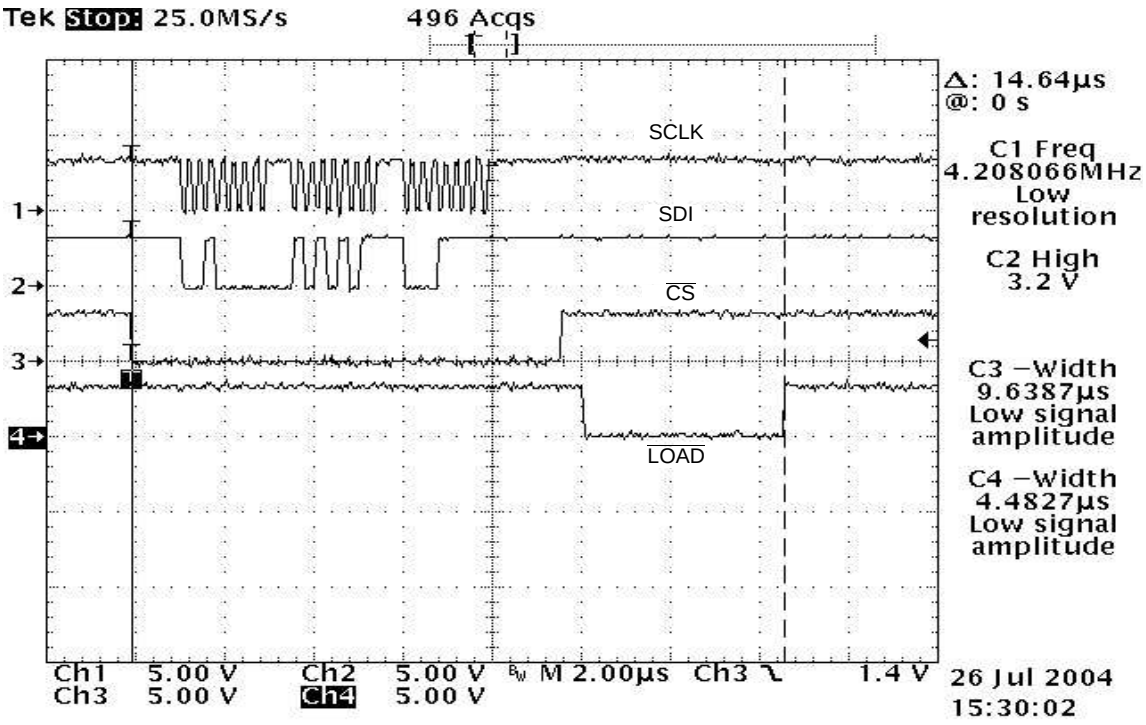

Figure 5. Actual Timing of the DAC7654 SPI Serial Interface

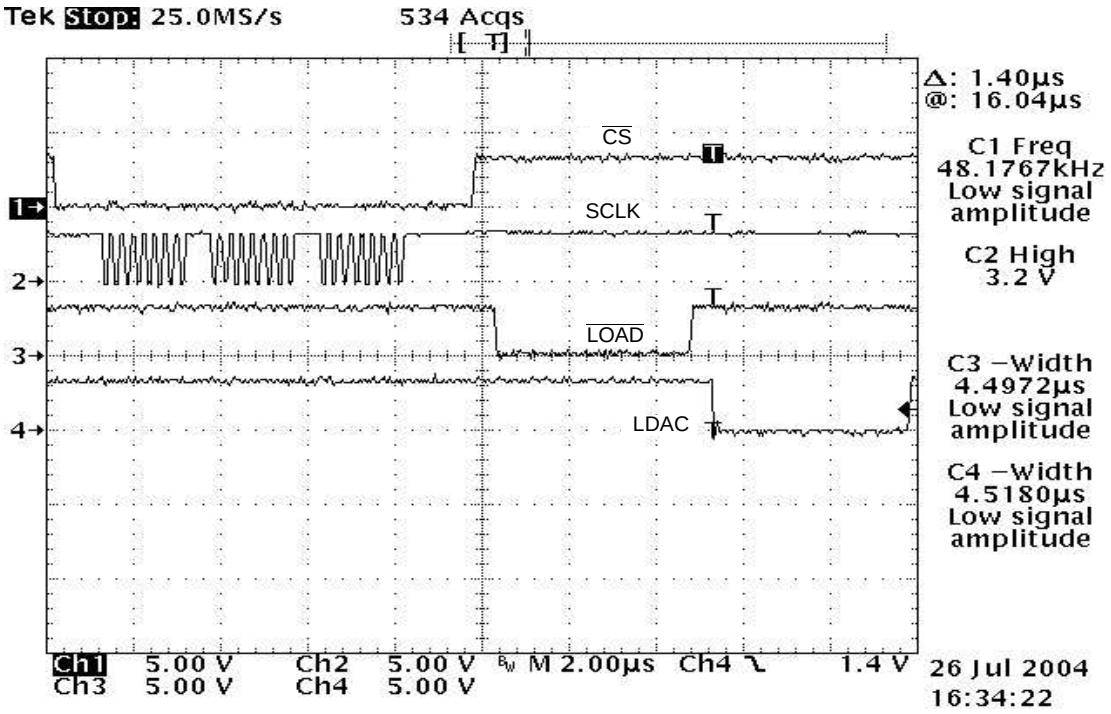

Figure 6. DAC7654 Control Signal Timing With LDAC

If all the serial interface timing for the DAC7654 is met as shown in Figure 5 and Figure 6, the sinusoidal waveform shown in Figure 7 should be observed. The DAC channel A output displays the sine wave with an amplitude of ±5 V while the rest of the DAC output channels are ±2.5 V. The signal amplitude of output A is ±5 V because the DAC A channel output of the DAC7654 is connected to an external output amplifier with a gain of two as shown in Figure 8. Only one DAC output channel at a time can be connected to the external amplifier and evaluated using the DAC7654 EVM board.

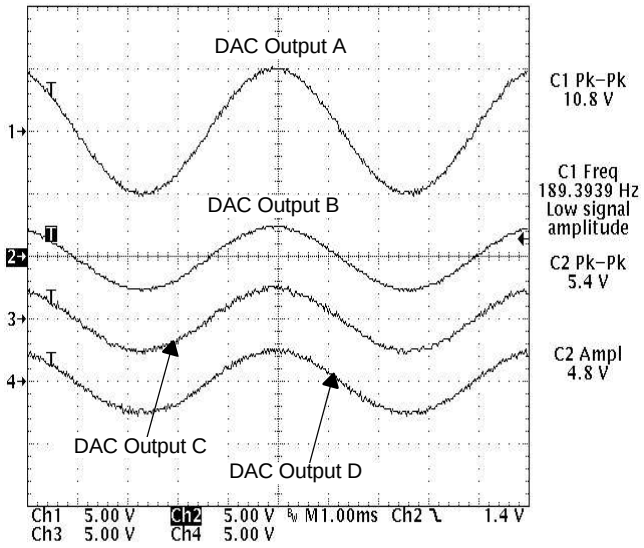

Figure 7. Output Waveform Timing Diagram, Control Byte (0x34)

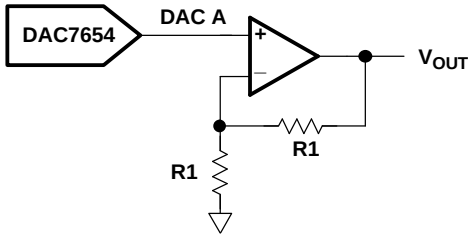

Figure 8. DAC A Output With a Gain of 2 (1 of 4 DACs)

4 Summary

This application note shows how easy it is to interface the DAC7654 to the MSP430F449 microcontroller using the SPI mode of serial communication. By using the software program provided in this application report, a simple routine can be created to generate a sinusoidal waveform. Using the DAC7654 EVM along with the HPA449 evaluation system makes developing the interface even easier. For more detailed information regarding the DAC7654, see the data sheet, SBAS263. For additional support, contact the TI Data Acquisition Product group by sending an e-mail to dataconvapps@list.ti.com.

For questions or information regarding the HPA449 evaluation system, contact SoftBaugh, Inc. at e-mail address info@softbaugh.com, or call the toll-free number (800) 794-5756 or the commercial number (770) 772-8111.

5 References

1. DAC7654 16-Bit, Quad Voltage Output, Digital-to-Analog Converter data sheet (SBAS263)
2. DAC7654 Evaluation Module User's Guide (SLAU130)
3. MSP430F449 data sheet (SLAS344)
4. MSP430X4XX Family User's Guide Manual (SLAU056)

5. MSP430F44X Evaluation System (HPA449) User's Guide (SoftBaugh, Inc)

Appendix A MSP430F449 Software Code

A.1 Header File

```
/*******************************************************************************
*
* FILENAME:    DAC7654.H
* PROJECT:     Application Note
* AUTHOR:      Jojo Parguian
* COMPANY:     Texas Instruments, Incorporated
*              HPA/DAP Applications, Dallas
* HISTORY:     July 26, 2004
*******************************************************************************/

#ifndef __DAC7654
#define __DAC7654
/*********************************************
* Pin Assignment - GPIO Definitions
 *********************************************/
/* J7 Connections */
#define CSa               0x80  /* P3.7 */
#define SCLKa             0x08  /* P3.3 */
#define FSR_A             0x00  /* P3.0 */
#define SDA_A             0x02  /* P2.1 */
#define SDO_A             0x01  /* P3.0 */
#define INTa              0x80  /* P2.7 */
/*********************************************
Defines and Commands********
*************************************/
#define Quick_Load        0x20  /* P2.7*/
/* J2 Connections */
#define RSTSEL           0x08  /* P2.3 (TOUTB)*/
#define CSb              0x40  /* P2.6*/
#define RST              0x80  /* P2.7 (INT_B)*/
#define LOAD_            0x02  /* P4.1*/
#define LDAC             0x01  /* P4.0*/
#define F_SYNC           0x01  /*P1.2*/
/*************************************
* Miscellaneous MSP430 Register Definitions
*************************************/
#define ACLK_5           0x040
#define SmCLK_5          0x020
#define MCLK_5           0x010
#define UCLK_5           0x008
#define LED              0x001
#define SPI              0x038
#endif/* #ifndef_DAC7654 */
```

A.2 Main Code

```
;*******************************************************************************
; MSP430F449 Demo - SPI Communication with DAC7654 SPI function using
; the HPA449 v1.1
 ; Assembled with IAR Embedded Workbench for MSP430 Kickstart
;
; AUTHOR:      Jojo Parguian
; COMPANY:     Texas Instruments, Incorporated
;              HPA/DAP Applications, Dallas
; Used on:     HPA449 V1.1
;              DAC7654EVM Rev 2 and Rev A
;*******************************************************************************

#include "msp430x44x.h"    //Standard Equations
#include "DAC7654.h"             //DAC Equations
#include "legal.asm"
```

```
#include "readme.asm"
#define DATASPI R9
;-------------------------------------------------
; 16-bit Sine Lookup table with 256 steps
 ;-------------------------------------------------
ORG  01000h
;-------------------------------------------------
Sin_tab
DW      32768,33572,34376,35178,35980,36779,37576,38370,39161,39947
DW      40730,41507,42280,43046,43807,44561,45307,46047,46778,47500
DW      48214,48919,49614,50298,50972,51636,52287,52927,53555,54171
DW      54773,55362,55938,56499,57047,57579,58097,58600,59087,59558
DW      60013,60451,60873,61278,61666,62036,62389,62724,63041,63339
DW      63620,63881,64124,64348,64553,64739,64905,65053,65180,65289
DW      65377,65446,65496,65525,65535,65525,65496,65446,65377,65289
DW      65180,65053,64905,64739,64553,64348,64124,63881,63620,63339
DW      63041,62724,62389,62036,61666,61278,60873,60451,60013,59558
DW      59087,58600,58097,57579,57047,56499,55938,55362,54773,54171
DW      53555,52927,52287,51636,50972,50298,49614,48919,48214,47500
DW      46778,46047,45307,44561,43807,43046,42280,41507,40730,39947
DW      39161,38370,37576,36779,35980,35178,34376,33572,32768,31964
DW      31160,30358,29556,28757,27960,27166,26375,25589,24806,24029
DW      23256,22490,21729,20975,20229,19489,18758,18036,17322,16617
DW      15922,15238,14564,13900,13249,12609,11981,11365,10763,10174
DW      9598,9037,8489,7957,7439,6936,6449,5978,5523,5085,4663,4258
DW      3870,3500,3147,2812,2495,2197,1916,1655,1412,1188,983,797,631
DW      483,356,247,159,90,40,11,1,11,40,90,159,247,356,483,631,797,983
DW      1188,1412,1655,1916,2197,2495,2812,3147,3500,3870,4258,4663
DW      5085,5523,5978,6449,6936,7439,7957,8489,9037,9598,10174,10763
DW      11365,11981,12609,13249,13900,14564,15238,15922,16617,17322
DW      18036,18758,19489,20229,20975,21729,22490,23256,24029,24806
DW      25589,26375,27166,27960,28757,29556,30358,31160,31964,32768
;-------------------------------------------------
           ORG   0F000h
;-------------------------------------------------
;*************************************************
; Program Code
;*************************************************
          RSEG CODE
;*************************************************
RESET
     mov.w #0A00h,SP        ;Initialize stack-pointer
     call #Init_Sys         ;Initialize system
     clr.w  R6
Write_Data
     mov.w #0FFh,R6
     move.w #0,R5
     bic.b #0FFh,&P1OUT
     bic.b #LDAC+LOAD_,&P4OUT
Again
     bic.b #CSb,&P2OUT
     move.b #Quick_Load,&U1TXBUF
WaitXMT0
           bit.b   #UTXIFG1,&IFG2               ; TXBUF ready?
           jnc     WaitXMT1
           mov.w   Sin_tab(R5),R9
           swpb    R9                           ; MSB first
           mov.b   R9,&U1TXBUF
           swpb    R9                           ; LSB next
WaitXMT1
           bit.b   #UTXIFG1,&IFG2               ; TXBUF ready?
           jnc     WaitXMT1                     ; MSB first
           mov.b   R9,&U1TXBUF
WaitXMT2
           bit.b   #UTXIFG1,&IFG2               ; TXBUF ready?
           jnc     WaitXMT2
```

```
            incd.w  R5
            sub.w   #1,R6
            mov.w   #05h, R14
Delay0
            dec.w R14                           ;
            jnz   Delay0                        ;
            bis.b #CSb,&P2OUT
            bic.b #LOAD_,&P4OUT
            mov.w #0Ah,R14                       ;Allow 5µsec settling time
Delay1
            dec.w R14                           ;
            jnz   Delay1
            bis.b #LOAD_,&P4OUT
            bic.b #LDAC,&P4OUT
            mov.w #0Ah,R14                       ;Allow 5µsec settling time
Delay2
            dec.w R14                           ;
            jnz   Delay2
            bis.b #LDAC,&P4OUT
            and.w #0FFh,R6
            jnz  Again
            jmp  Write_Data
;*****************************
;Clear TX Flag
;*****************************
CLEAR0
            bit.b #UTXIFG0,&IFG1                ;TXBUF ready?
            jnc CLEAR0                          ;1 = ready
            bic.b #UTXIFG0,&IFG1
            ret
;*****************************
;Clear TX Flag
;*****************************
CLEAR1
            bit.b #UTXIFG1,&IFG2                ;TXBUF ready?
            jnc CLEAR1                          ;1 = ready
            bic.b #UTXIFG1,&IFG2
            ret
;************************************************************
;Init_Sys; Modules and Controls Registers set-up subroutine
;************************************************************
StopWDT
            mov.w         #WDTPW+WDTHOLD,&WDTCTL  ;  Stop Watchdog Timer
 SetupFLL2
            bis.b  #FN_4,&SCFI0                  ; x2 DCO, 8MHz nominal
DCO
            bis.B  #DCOPLUS+XCAP14PF,&FLL_CTL0    ; DCO+, configure load caps
            mov.b  #121,&SCFQCTL                 ;(121+1) x 2 x 32768 = 7.99MHz
SetupPorts
;Port2
            bis.b #CSb+RST+RSTSEL, &P2DIR
            bis.b #CSb+RST+RSTSEL, &P2OUT
;Port4
            bis.b #SPI,&P4SEL                     ;P4.3,4,5 SPI option select
            bis.b #LDAC+LOAD_, &P4DIR
            bis.b #LDAC+LOAD_, &P4OUT
SetupSPI0
            bis.b #USPIE0,&ME1                    ;Enable SPI TX/RX
            mov.b #CHAR+SYNC+SYNC+MM,&U0TCTL
            mov.b #02h,&U0BR0
            mov.b #00h,&U0BR1
            mov.b #00h,&U0MCTL
            bis.b #UTXIE0, &P1IE
SetupSPI1
            bis.b #USPIE1,&ME2                       ; Enable SPI TX/RX
            mov.b #CHAR+SYNC+MM+SWRST,&U1CTL         ; 8-bit SPI Master
```

```
            bis.b #CKPL+SSEL0+SSEL1+STC,&U1TCTL       ; 3-pin SPI mode, SMCLK
            mov.b #002h,&U1BR0              ; CKPL+CKPH gives SCLK idle high and data
            mov.b #000h,&U1BR1              ; sampled on the falling edge of SCLK
            mov.b #000h,&U1MCTL             ; CKPL gives SCLK idle high and data
            bis.b #USPIE1,&ME2              ; sampled on the rising edge of SCLK
            bic.b #SWRST, &U1CTL            ; CKPH gives SCLK idle low and data
                                            ; sampled on the rising edge of SCLK
            ret
;-------------------------------------------------------------
WDT_ISR; Exit LPM0 on reti;
;-------------------------------------------------------------
          xor.b   #01h,&P1OUT               ; monitors functioning of the system
          bic     #LPM0,0(SP)               ; Clear LPM0 from TOS
          reti                              ; return from interrupt
;-------------------------------------------------------------
IRQ1_ISR; Exit LPM0 on reti
;-------------------------------------------------------------
          bic.b   #01h,&P1OUT
          bic.b   #INTa, &P2IFG
          reti                              ; return from interrupt
;-------------------------------------------------------------
          COMMON INTVEC         ;MSP430x44x Interrupt vectors
;-------------------------------------------------------------
             ORG   WDT_VECTORx
WDT_VEC      DW    WDT_ISR                   ; Watchdog/Timer, Timer mode
    ORG      RESET_VECTOR
RESET_VEC    DW   RESET                      ; POR, ext. Reset, Watchdog
    ORG      PORT2_VECTOR
PORT2_VEC    DW   IRQ1_ISR                   ; PORT2, Ext. Int.
    END
```

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.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265