AN1730

Digital Amplification Control of an Analog Signal Using the MC68HC705J1A

By Mark Glenewinkel
Consumer Systems Group
Austin, Texas

Introduction

This application note describes the interface between an HC705J1A and a multiplying digital-to-analog converter (MDAC) to digitally control the amplification or attenuation of an operational amplifier circuit. The MDAC used is Analog Device's DAC8043, a 12-bit, 8-pin serial device. By using the MDAC, a mechanical potentiometer can be replaced by a more reliable and robust solution.

The microcontroller unit (MCU) interface must be able to "talk" to the DAC8043 using a serial communication link. The serial peripheral interface (SPI) is one of the most widely used serial transmission methods for communication between an MCU and a peripheral, although not all HC05 Family members have SPI modules. An HC05 MCU without an SPI must interface with the DAC8043 using a software input/output (I/O) driver. This method uses software bit programming to communicate with the DAC8043. Even though it is not as efficient as the hardware SPI method, it provides the MCU with a means to send data to the DAC8043.

The MC68HC705J1A MCU is used here to demonstrate the software driver routine.

The Digital-To-Analog Converter R–2R Network

The digital-to-analog converter (DAC) inside the DAC8043 is based on the R–2R resistor network. Most CMOS DACs are based on the R–2R current steering circuit. **Figure 1** shows a simple 2-bit, R–2R DAC circuit. A reference voltage is applied to the V_{ref} pin and the current, I, is binarily divided throughout the array as shown. These currents are steered in discrete incremental amounts to the I_{Out} and GND nodes. The digital input to the DAC determines the position of the switches used to steer the current. A logic 1 causes the switch to steer the current to I_{Out}, while a logic 0 causes the switch to steer the current to GND. The CMOS DAC multiplies the digital input value by the analog input voltage at the V_{ref} pin.

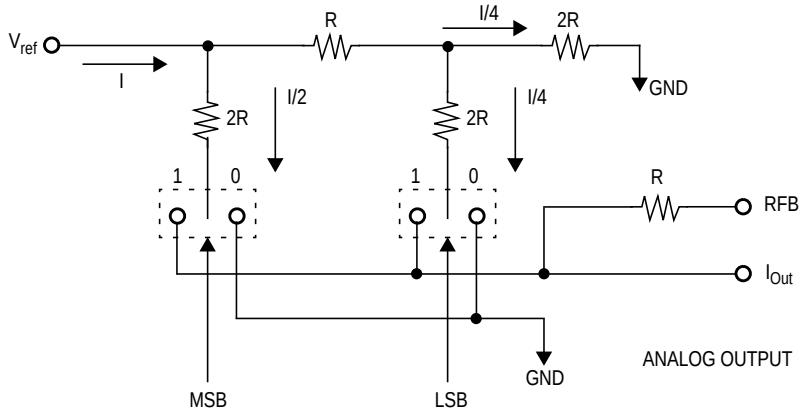


Figure 1. Simple 2-Bit Digital-to-Analog Circuit

In this example, a digital value of 01_2 causes I/4 to flow to I_{Out} and the remainder of the current flows to GND. Therefore, the digital input 01_2 refers to a reading of quarter scale. If the input to the DAC was 11_2, the output current would be full scale minus one LSB (least significant bit). In this example, the full-scale reading would be (3/4)I.

The R–2R DAC output consists of current and can be converted to a voltage by using a current-to-voltage op amp. The feedback resistor R_{FB} is made equal to the resistance of the DAC array of resistors and is connected to the output of the op amp. The current from I_{Out} is

connected to the negative input of the op amp. The maximum output voltage for this configuration is –I(1—2^{-n})R where n is the number of bits of DAC resolution. The minus sign in the output voltage is a result of the op amp in the current-to-voltage configuration.

The resultant voltage output of the DAC then can be defined as

$$V_{Out} = -V_{ref} * D$$

where

$D = (xx/2^n)$

xx is the digital input value

n is the bit resolution of the DAC

Therefore, the input voltage V_{ref} is multiplied by –D. For the 2-bit DAC example shown in **Figure 1**, the available V_{Out} voltages are 0 V_{ref}, –1/4 V_{ref}, –1/2 V_{ref}, and –3/4 V_{ref}.

Digitally Controlled Operational Amplifiers

When combined with an operational amplifier, the R–2R circuit can be configured to either attenuate or amplify a signal by the value of its digital input. When the DAC is configured in this way, it is referred to as a multiplying DAC.

Most analog amplification circuits employ the operational amplifier to either attenuate or amplify a signal. This configuration is shown in **Figure 2**. The gain of the amplifier is set by the potentiometer. By increasing the resistance, R_F, relative to the resistance, R_{In}, the output is amplified. Likewise, decreasing the resistance, R_F, relative to the resistance, R_{In}, the output is attenuated. The output voltage, V_{Out}, is described as

$$V_{Out} = -V_{ref} * (R_F/R_{In})$$

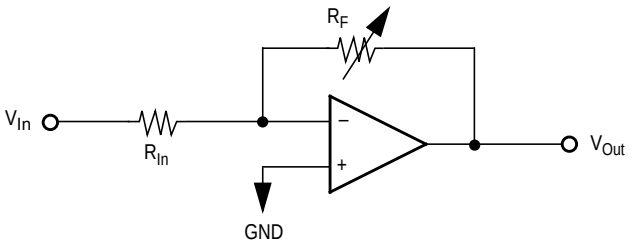


Figure 2. Variable Gain Operational Amplifier

We can configure a CMOS DAC to provide a means of digitally attenuating a signal. **Figure 3** shows a 2-bit DAC in an attenuation configuration. The output voltage, V_{Out}, is defined by

$$V_{Out} = -D * V_{In}$$

where

$D = (xx/2^n)$
xx is the digital input value
n is the bit resolution of the DAC

In this example, a digital input of 01_2 would attenuate the input voltage to produce an output voltage of 1/4 V_{In}.

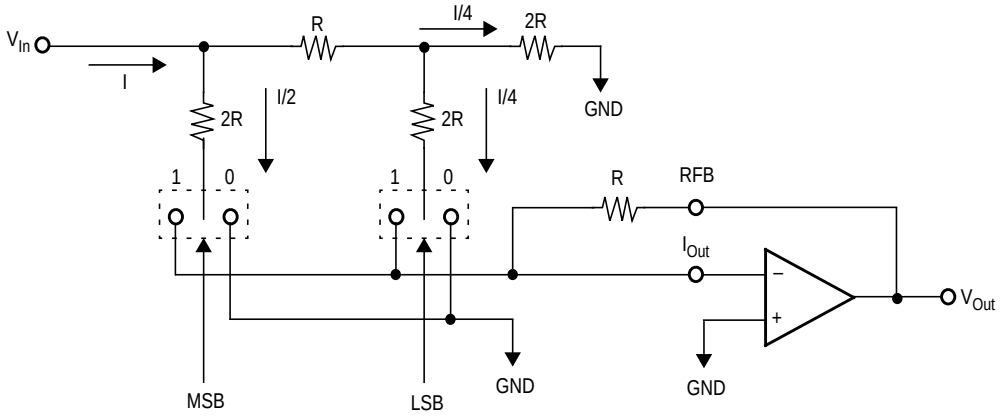


Figure 3. 2-Bit DAC Attenuation Circuit

By rearranging the circuit in **Figure 3**, the CMOS DAC can be configured to provide a means of digitally amplifying a signal. **Figure 4** shows a 2 bit DAC in the amplifying configuration. The output voltage, V_{Out}, is defined by

$$V_{Out} = -(V_{In}/D)$$

where

$D = (xx/2^n)$
xx is the digital input value
n is the bit resolution of the DAC

In this example, a digital input of 01_2 would amplify the input voltage to produce an output voltage of 4 * V_{In}.

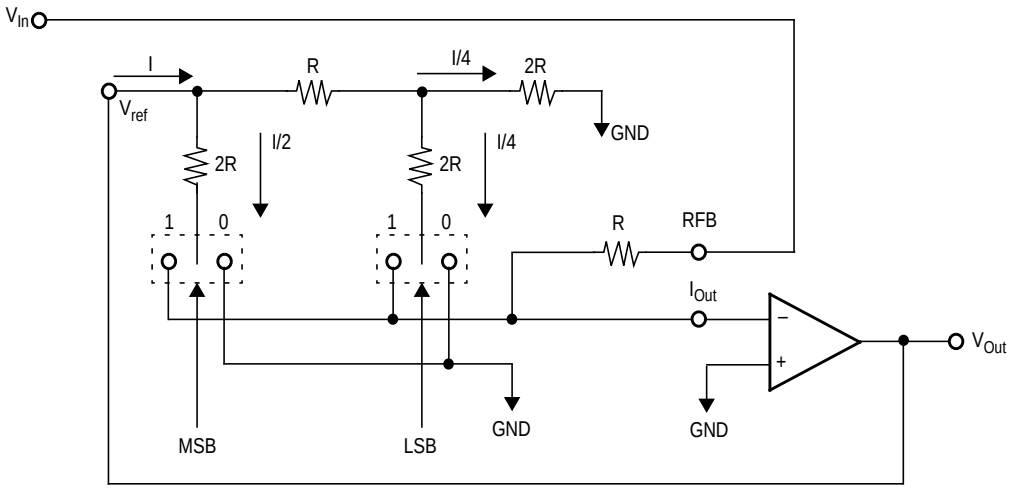


Figure 4. 2-Bit DAC Amplification Circuit

DAC8043 Multiplying Digital-To-Analog Converter

Overview

The DAC8043 is a 12-bit, CMOS, high-accuracy, multiplying DAC. The part is packaged in a space-saving, 8-pin, mini-DIP package ideal for minimum PC board space applications. The DAC features serial data input, double buffering, and excellent analog performance. Separate input clock and DAC loading control lines allow the user full control of data loading and analog output.

Figure 5 shows the functional block diagram with the pinout of the DAC8043. The circuit consists of a 12-bit, serial-in parallel-out shift register, a 12-bit CMOS DAC, a 12-bit DAC register, and control logic.

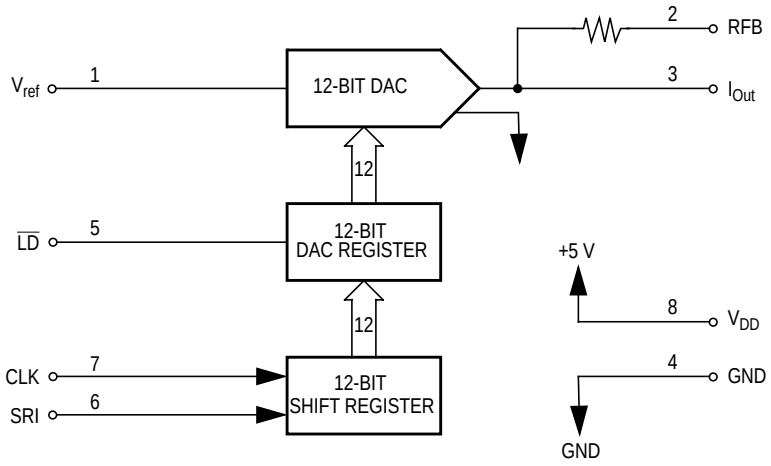


Figure 5. DAC8043 Block Diagram

Digital Interface

The digital interface to the DAC8043 is composed of a serial data port that synchronously receives 12-bit data. Serial data is fed to the DAC8043 MSB (most significant bit) first.

CLK — Serial Data Clock

This pin is an input that clocks in the SRI data on a rising edge.

LD — Load Register

This input pin is used to load the 12-bit serial data into the DAC register.

SRI — Shift Register In

This pin serves as the input data line that receives the 12-bit serial data stream.

Figure 6 shows serial data is clocked into the input register on the rising edge of the clock pulse. When all 12 bits are clocked in, the DAC register is loaded by toggling the $\overline{LD}$ pin low. Data in the DAC register is converted to an output current.

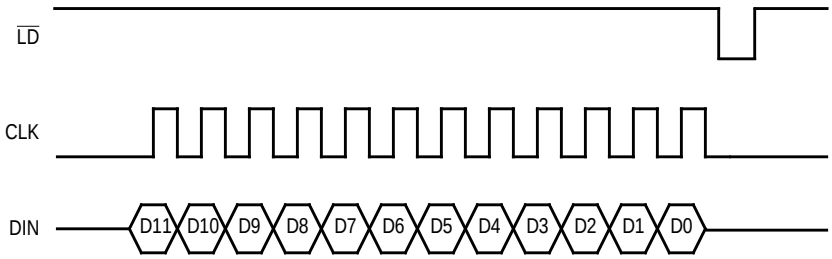


Figure 6. DAC8043 Timing Diagram

Description of the HC705J1A Interface

Hardware

With only 20 pins, the HC705J1A is one of the smaller members of the HC05 Family. It has a total of 1240 bytes of erasable programmable read-only memory (EPROM) and includes 14 I/O pins. The schematic for the HC705J1A-to-DAC8043 interface for amplification is shown in **Appendix A — HC705J1A/DAC8043 Digital Amplifier**. The schematic for the HC705J1A-to-DAC8043 interface for attenuation is shown in **Appendix B — HC705J1A/DAC8043 Digital Attenuator**.

The pins used to drive the DAC8043 on the HC705J1A are:

- Port A, Bit 0— This I/O pin (SER_CLK) is configured as an output to drive the serial clock of the serial transmission bus.
- Port A, Bit 1— This I/O pin (SER_OUT) is configured as an output to drive the serial data out and onto the SRI pin of the DAC8043.
- Port A, Bit 2— This I/O pin ($\overline{CS}$) is configured as an output to drive the $\overline{LD}$ pin on the DAC8043.

For further information on the HC705J1A, consult the *MC68HC705J1A Technical Data*, Freescale order number MC68HC705J1A/D.

Test Software

I/O driving is the process of toggling I/O pins with software instructions to emulate a certain piece of hardware peripheral. The flowchart for the I/O-driven DAC8043 is shown in **Appendix C — HC705J1A/8043 Flowchart**, and the actual HC05 assembly code is given in **Appendix D — HC705J1A/8043 Assembly Code**. This routine was written especially for the DAC8043 and is not a full-featured representation of Motorola's SPI module found on other microcontrollers. Enhancements to the routine were not included to maximize the code's efficiency.

As stated in the preceding hardware section, I/O pins have been used to send out the correct serial transmission protocol to the DAC8043. The HC05 CPU provides special instructions to specifically manipulate single

I/O pins. The DAC8043 serial stream shown in **Figure 6** will be re-created by three I/O pins on the HC705J1A.

This transmission has been put into a subroutine called J8043_TXD. The best way to describe the subroutine is to list each segment of the code to explain the I/O during transmission.

- Initialization
  - Load the X register with 12. Use it as a counter.
- Write the serial output pin.
  - Bit 3 of TEMP is read. If it is high, a 1 is written to SER_OUT. If it is low, a 0 is written to SER_OUT.
- Clock the serial clock pin.
  - The SER_CLK pin is written high and then written low.
- Rotate the data bytes.
  - Arithmetically shift left the TEMP+1 byte (bit 7 ≥ C)
  - Rotate left the TEMP byte (C ≥ bit 0)
- Is the loop done?
  - The X register is decremented and checked to see if it is 0. If X is not 0, the code is executed at the start of writing the SER_OUT pin. This loop continues until 12 transmissions are completed.
- Load the data into the DAC register.
  - The $\overline{CS}$ pin is written low and then written high.
  - Return from subroutine.

The main routine in **Appendix D — HC705J1A/8043 Assembly Code** continuously sends out 12-bit data to the DAC. It cycles through $FFF to $000. Depending on how the hardware is configured, it will attenuate or amplify a signal through different levels. This table shows various examples of digital inputs and their corresponding attenuation or amplification factor.

12-Bit Code	Attenuate	Amplify
$800	1/2	2X
$400	1/4	4X
$200	1/8	8X
$100	1/16	16X
$080	1/32	32X

To test the software routine, follow these steps after programming the HC705J1A with the code in **Appendix D — HC705J1A/8043 Assembly Code** and constructing the digital attenuator schematic in **Appendix B — HC705J1A/DAC8043 Digital Attenuator**. To emulate the design, connect the target pins PA0–PA2 on the emulator to the DAC8043.

1. Check that the oscillator circuit on pin 2 of the HC705J1A is running at 4 MHz.
2. Verify that the $\overline{\text{RESET}}$ pin on the HC705J1A is 5 volts.
3. Generate a 1 V_{PP} signal and feed it into the analog input.
4. After resetting the part, use an oscillator to analyze the signal on the analog output test point. The signal should be continuously attenuated in a periodic fashion.

If you are using an emulator, you can check different DAC values.

1. Stop the program. Memory modify TEMP to $08 and TEMP+1 to $00.
2. Set the PC to J8043_TXD.
3. Create a breakpoint at the RTS within J8043_TXD routine.
4. Run the program until it hits the breakpoint. The output signal should now be 0.5 V_{PP}.

To test for amplification, construct the digital amplifier schematic in **Appendix A — HC705J1A/DAC8043 Digital Amplifier**. The signal should be continuously clipped. With a digital input of $800, the output signal should now be 2.0 V_{PP}.

Layout Considerations

Many considerations apply when laying out mixed signal designs such as the DAC8043 and the HC05 MCU. Analog signal integrity may be greatly affected if proper layout design is not followed.

Use this check list to ensure proper mixed-signal designs.

- Physically separate critical analog circuits from the digital circuits of the MCU. If possible, split the board in half to separate analog and digital circuits. Each half will have its own power and ground system.
- If at all possible, do not let analog input line traces cross digital traces. But if this must happen, make sure they cross at right angles to each other.
- Use power or ground traces to isolate the analog input pins from the digital pins.
- With quality ceramic capacitors, bypass the power supplies to the proper ground at the operational amplifier power pins. Keep the bypass capacitors lead lengths as short as possible.

- To bypass low-frequency power supply noise, use tantalum or aluminum electrolytic capacitors of 5 to 20 F. These should be placed near the point the power supplies enter the board.
- If economically possible, use separate analog and digital ground planes. The two ground planes should be tied together at the low impedance power-supply source.

References/Additional Reading

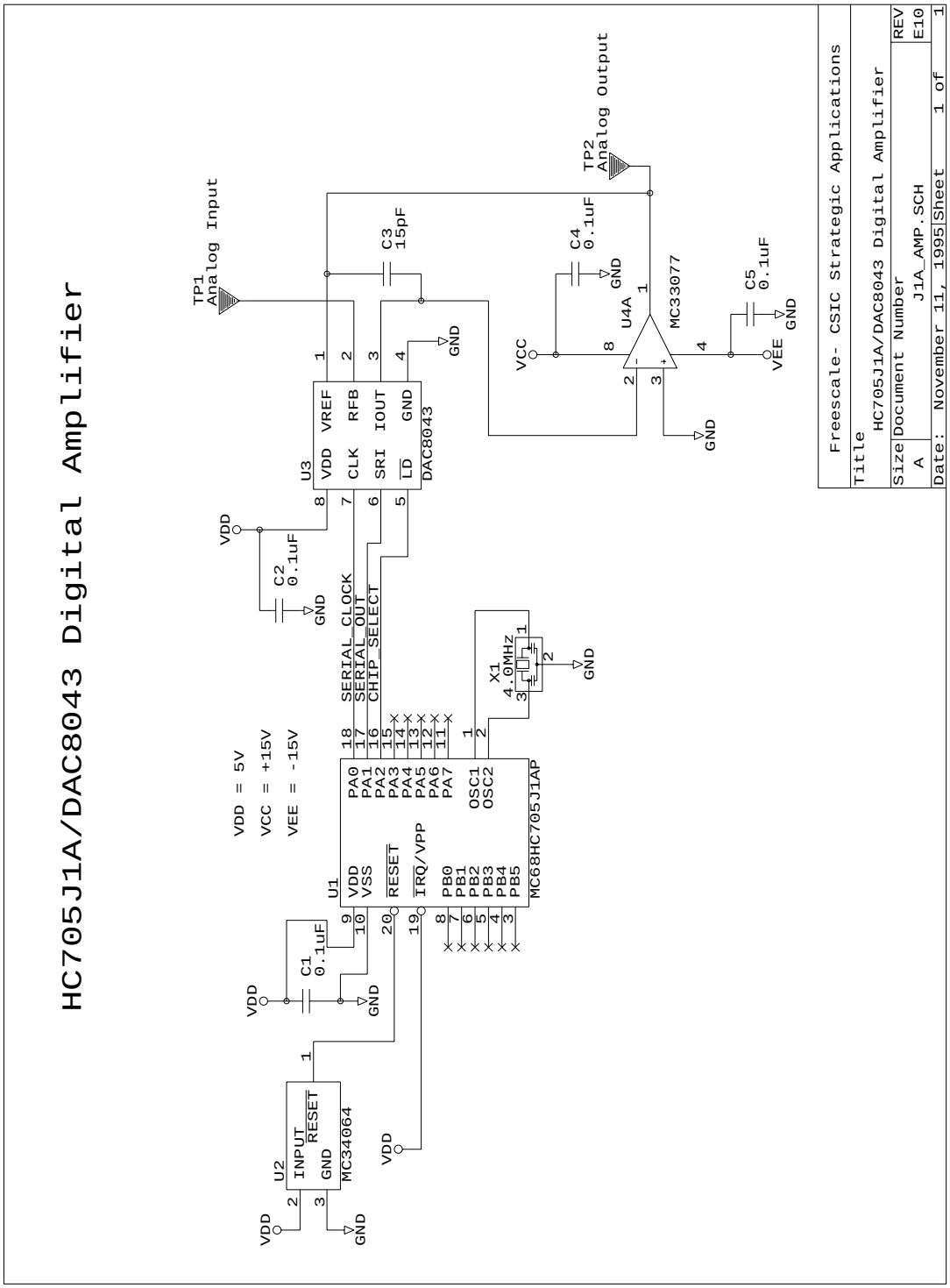
Analog-Digital Conversion Handbook, Third Edition, New York: Prentice-Hall, 1986.

MC68HC05 Applications Guide, (M68HC05AG/AD), Freescale, 1989.

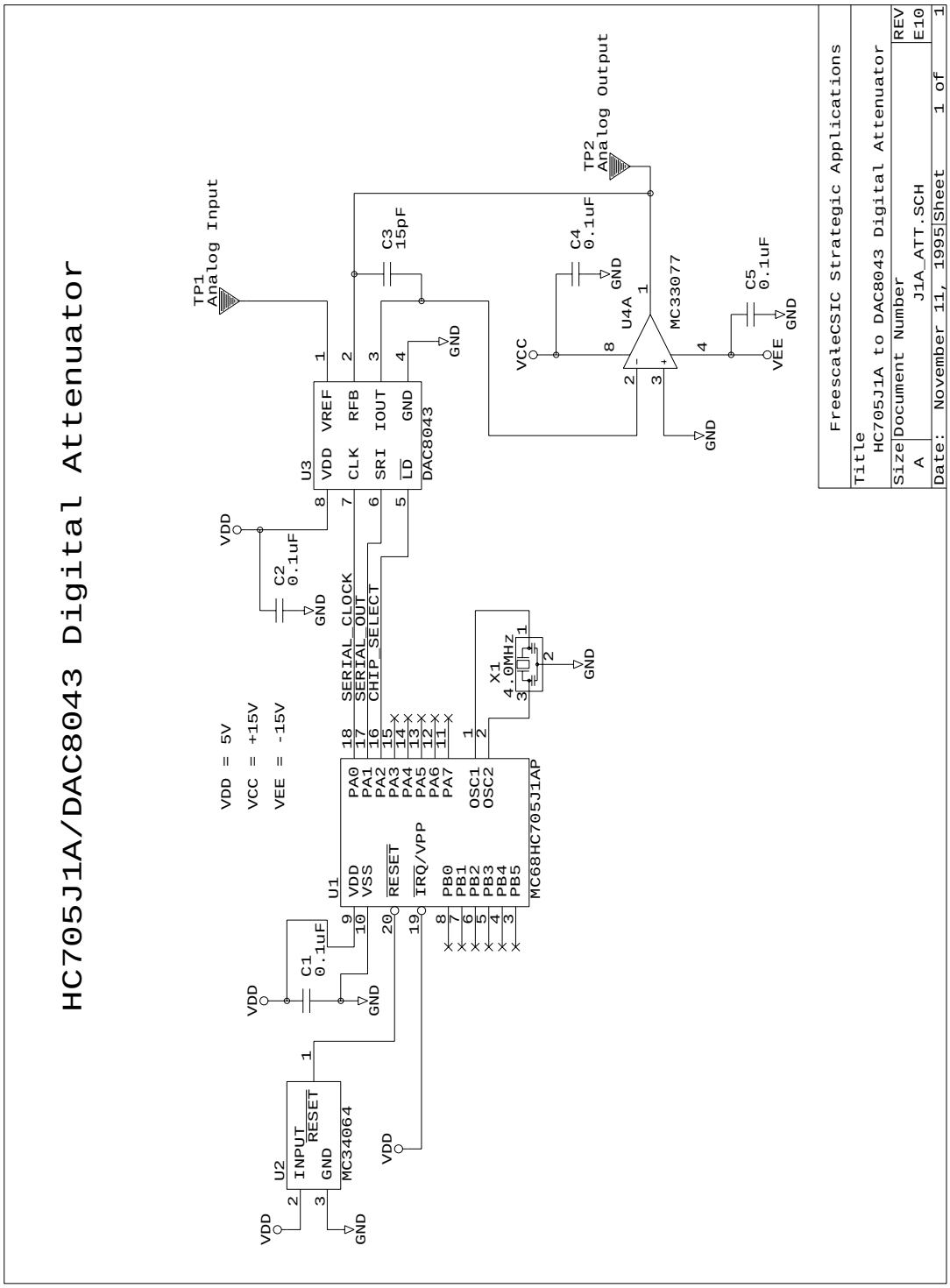
MC68HC705J1A Technical Data, (MC68HC705J1A/D), Freescale, 1995.

DAC8043 Data Sheet, Analog Devices, 1996.

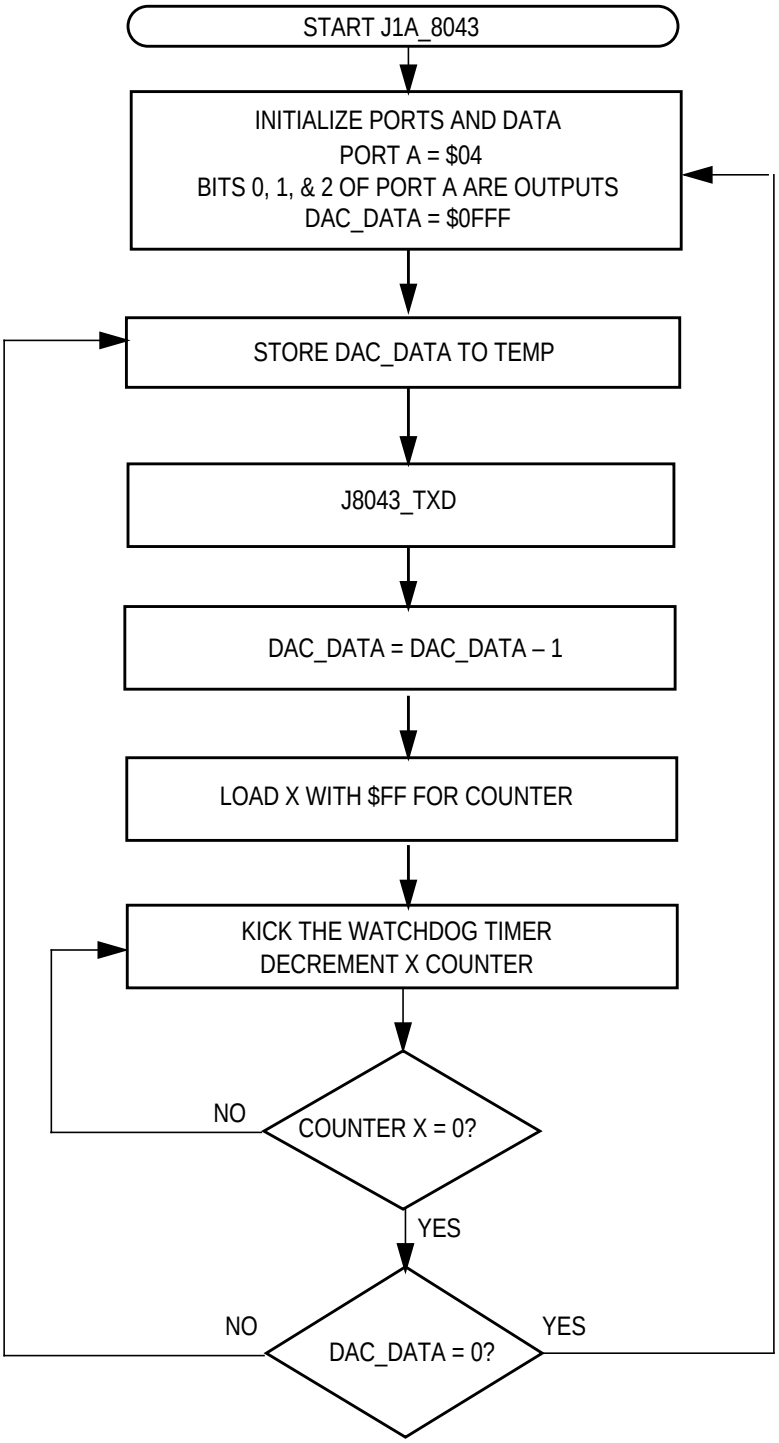
Appendix A — HC705J1A/DAC8043 Digital Amplifier

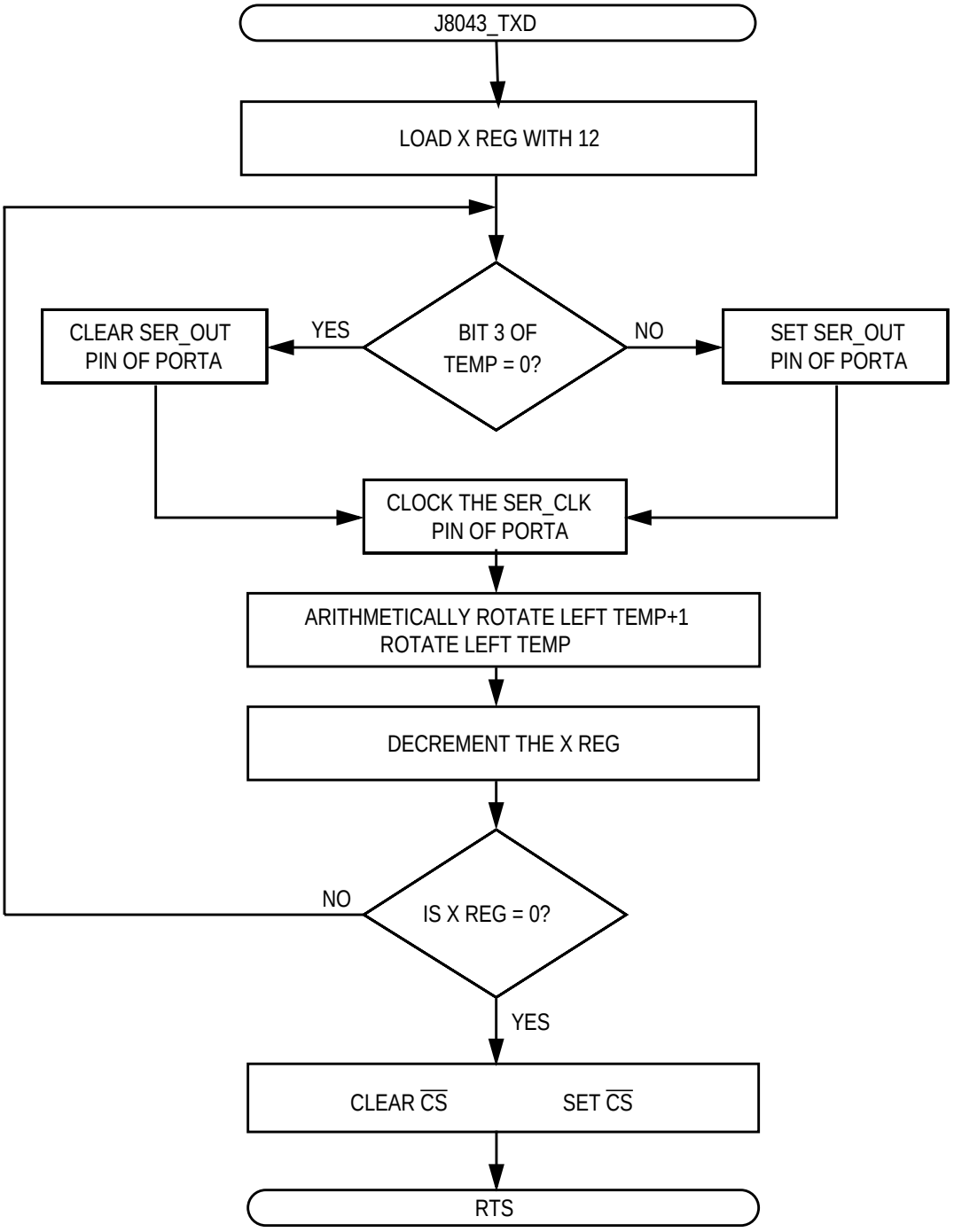


Appendix B — HC705J1A/DAC8043 Digital Attenuator



Appendix C — HC705J1A/8043 Flowchart





Appendix D — HC705J1A/8043 Assembly Code

```
*****************************************************************************
*                                                                           *
*    Main Routine J1A_8043 — 705J1A to Analog Devices DAC8043 MDAC          *
*                                                                           *
*****************************************************************************
*                                                                           *
* File Name: J1A_8043.RTN                  Copyright (c)  1995              *
*                                                                           *
*                                                                           *
* Full Functional Description Of Routine Design:                            *
*                                                                           *
*   Program flow:                                                           *
*                                                                           *
*       Reset:     Initializes ports for bit banging.                       *
*                                                                           *
*                  Initialize DAC data and count for test.                  *
*                                                                           *
*                  Execute continuous loop to create different levels       *
*                    of amplification or attenuation                        *
*       J8043_TXD: Loop 12 times                                            *
*                    Write address data on port pin and clock it            *
*                  Loop done                                                *
*                  Set CS                                                   *
*                  Clear CS                                                 *
*                                                                           *
*****************************************************************************
*                                                                           *
*       Part Specific Framework Includes Section                            *
*                                                                           *
* Place the assembler statement (#INCLUDE) to include the part specific     *
* framework for the target part.                                            *
*                                                                           *
*****************************************************************************

#nolist
#INCLUDE 'H705J1A.FRK'                  ;Include the equates for the HC705J1A
                                        ;so that all labels can be used.
#list
```

```
*****************************************************************************
*                                                                           *
*       MOR Bytes Definitions for Main Routine                              *
*                                                                           *
*****************************************************************************

                org     MOR
                db      $21             ;COP enabled, osc resistor enabled
                                        ;If used on a mask rom part,
                                        ; be sure to specify these options.

*****************************************************************************
*                                                                           *
*             Equates and RAM Storage                                       *
*                                                                           *
*****************************************************************************

SER_CLK         equ     0               ;bit # for serial clock
SER_OUT         equ     1               ;bit # for serial data out
CS              equ     2               ;bit # for chip select

***     RAM storage variables   ***

                org     RAM             ;start of static RAM at $C0
DAC_DATA        rmb     2               ;2 bytes needed for DAC data
TEMP            rmb     2               ;2 bytes for counting

*****************************************************************************
*                                                                           *
*       Program Initialization                                              *
*                                                                           *
* This section sets up the port for bit banging.                            *
*                                                                           *
* To prevent floating inputs and associated high current draw,              *
* the HC705J1A has pulldown devices on all I/O pins. This                   *
* initialization should enable these pulldowns on unused I/O                *
* pins. RESET* enables the pulldowns, so no code is required.               *
*                                                                           *
*****************************************************************************

                org     EPROM           ;start of user eprom at $300
J8043_START     lda     #$04
                sta     PORTA           ;init portA
                lda     #$07
                sta     DDRA            ;init port A dir
```

```
*****************************************************************************
*                                                                           *
*       J1A_8043 Main Program Loop                                          *
*                                                                           *
* The code runs through the routine to check for                            *
* proper serial transmission. It cycles transmission of $FFF to $001 to the *
* DAC8043.                                                                  *
*                                                                           *
*                                                                           *
*****************************************************************************

*       Initialize DAC_DATA for test
J8043_Init      lda     #$0F            ;init for transmit
                sta     DAC_DATA
                lda     #$FF
                sta     DAC_DATA+1

*       Loop to tranmsit and decrement the data counter
J8043_Loop      lda     DAC_DATA        ;store DAC_DATA to TEMP
                sta     TEMP
                lda     DAC_DATA+1      ;store DAC_DATA+1 to TEMP+1
                sta     TEMP+1
                jsr     J8043_TXD       ;jsr to transmit info to DAC8043

*       Decrement DAC_DATA by 1
                lda     DAC_DATA+1      ;work on lower 8 bits
                sub     #$01
                sta     DAC_DATA+1
                lda     DAC_DATA        ;work on upper 4 bits
                sbc     #0
                sta     DAC_DATA

*       Loop to slow down rate of transmission and kick the WatchDog
J8043_WDOG      ldx     #$FF
WAIT            lda     #$00            ;reset COP
                sta     COPR
                decx
                bne     WAIT

*       Check if DAC_DATA = $00
                lda     DAC_DATA
                cmp     #$00
                bne     J8043_Loop
                lda     DAC_DATA+1
                cmp     #$00
                bne     J8043_Loop
                bra     J8043_Init
```

```
*****************************************************************************
*       J8043_TXD SubRoutine                                                *
*                                                                           *
* This subroutine will write 12 bit data to the DAC8043                     *
*                                                                           *
* Conditions: DAC_DATA/+1 has been put in TEMP/+1                           *
* Destroys: X                                                               *
*                                                                           *
*****************************************************************************

*       Send out 12 bit frame   *
J8043_TXD       ldx     #12T            ;load X with 12

***     Write the serial output pin
WRITE           brclr   3,TEMP,J8043_C  ;if temp bit3 = 0,
                                        ;goto j8043_c
                bset    SER_OUT,PORTA   ;ser_out = 1
                bra     J8043_CLOCK     ;goto j8043_clock
J8043_C         bclr    SER_OUT,PORTA   ;ser_out = 0
                brn     J8043_C         ;evens it out

***     Clock the serial clock pin
J8043_CLOCK     bset    SER_CLK,PORTA   ;ser_clk = 1
                bclr    SER_CLK,PORTA   ;ser_clk = 0
                asl     TEMP+1          ;rotate left TEMP+1
                rol     TEMP            ;rotate left TEMP

                decx                    ;decrease counter loop
                bne     WRITE           ;is the count finished?

                bclr    CS,PORTA        ;CS* is low, data is now latched
                bset    CS,PORTA        ;CS* is high
                rts

*****************************************************************************
*                                                                           *
*       Interrupt and Reset vectors for Main Routine                        *
*                                                                           *
*****************************************************************************

                org     RESET
                fdb     J8043_START
```