4

Interfacing a Digital-to-Analog Converter

This chapter describes how you can build a Digital-to-Analog Converter (DAC). With this circuit you can design your own function generator.

Introduction and Motivation:

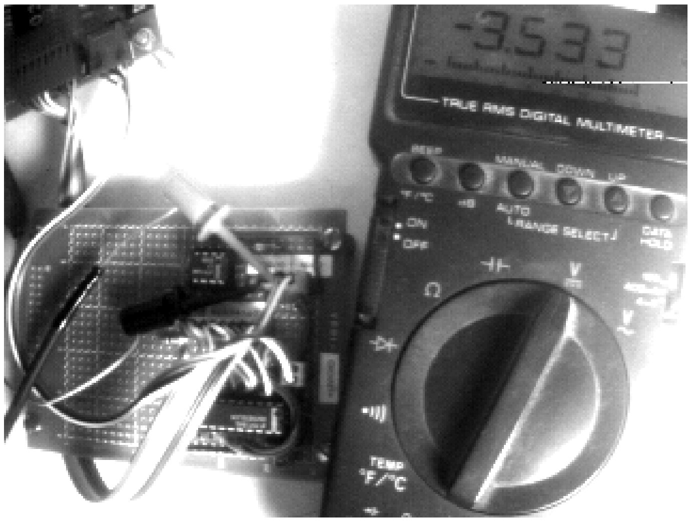

Figure 8: The voltmeter shows how the DAC generates analog voltages (-3.533 V in this photo).

Figure 8 shows the voltmeter reading -3.533 Volts and is the output from a Digital-to-Analog Converter (DAC). This is an example of how the 8255 PC Interface Card can be interfaced with DAC chips to generate analog voltages. There are PC cards dedicated to DAC on the market. However, these are often very expensive (over $100). This interface provides an economical yet effective means of generating analog voltages.

Analog voltage generation opens the door to many more exciting possibilities. For example if you interface the DAC to a power amplifier, you can have DC motor speed control. Perhaps you might want a wave function generator. In software you could program a sine, ramp or square wave, and output the corresponding voltages through this DAC interface.

Parts List, Numbers and Sources:

Part Description	Supplier and Number	QTY	Cost($)
DAC0832LCN (20-pin DIP)	Jameco 128186	1	4.95
LF353N Op amp (8-pin DIP)	Jameco 22939	1	0.59
10 K resistor		5	

Table 5: Parts list for DAC interface

Schematic Wiring Diagram:

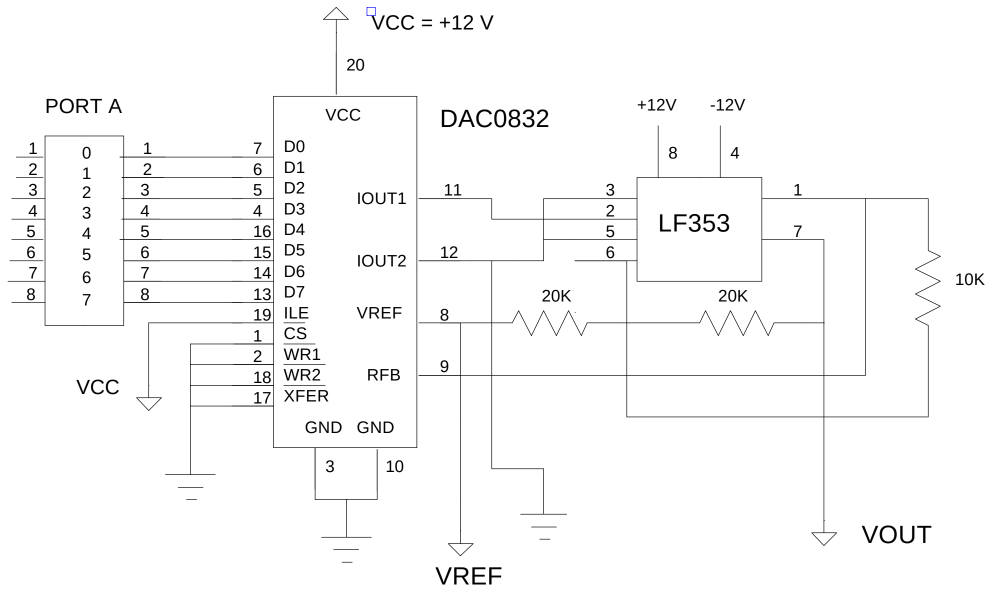

Figure 9: Schematic wiring diagram for DAC Interface. The numbers 0-7 in the box labeled Port A refers to lines A0-A7 on the Terminal Expansion Board. Note: 20 K resistors can be formed by connecting two 10 K resistors in series.

Construction

The DAC0832 is a 20-pin "skinny" DIP chip made by National Semiconductor and works as follows. An 8-bit data word[7] is loaded onto lines D0-D7. The DAC0832 converts this bit data into analog currents IOUT1 and IOUT2. The LF353 then converts these currents into analog voltages[8]. The resulting analog voltage is:

where is the decimal equivalent of the 8-bit data word. As a result, your DAC circuit will give you analog voltages that range from to . Typically DAC circuits set to 5 volts. National's tech sheet says that the maximum is 25 volts.

The DAC has 8-bit resolution. This means that the difference in a single bit is . For example, if you set to +5 volts you will have a single bit resolution of 0.0391 volts. That is, the difference from 0000000 to 00000001 will be 0.0391 volts. Because of the DAC0832's 8-bit resolution, you can never have a voltage of say 0.02 volts. If

[7] An 8-bit data word ranges from 00000000 to 11111111 binary (which is 0 to 255 in decimal)
[8] The LF353N is configured as a current-to-voltage converter

resolution is critical to your application you would have to find a higher resolution DAC but for many purposes 8-bit resolution is sufficient.

In the schematic the DAC0832's Chip Select (CS), Write Current 1 (WR1), Write Current 2 (WR2), and data word Transfer (XFER) are all set to ground. By doing this, digital data on lines D0-D7 is converted as soon as data arrives at D0-D7. This data is automatically latched since the Input Latch Enable (ILE) has been set. This latching feature of the DAC0832 makes DAC very simple to do. If you require very high-speed digital conversion you can use software driven timing routines to control latching using ILE more precisely.

QBasic Program Listing for DAC Interfacing:

```
100 REM DAC.BAS
110 REM PROMPT USER FOR DECIMAL WORD. RETURNS ASSOCIATED ANALOG VOLTAGE
120 CLS
130 LOCATE 2, 1: PRINT "ENTER BASE ADDRESS IN DECIMAL (E.G. 608):"
140 INPUT BASEADDR
150 PORTA = BASEADDR
160 PORTB = BASEADDR + 1
170 PORTC = BASEADDR + 2
180 CNTRL = BASEADDR + 3
190 OUT CNTRL, 128: ' ALL PORTS ARE OUTPUT

200 LOCATE 4, 1: PRINT "SELECT PORT"
210 LOCATE 5, 1: PRINT "(1) PORT A"
220 LOCATE 6, 1: PRINT "(2) PORT B"
230 LOCATE 7, 1: PRINT "(3) PORT C"
240 INPUT CHOICE
250 IF CHOICE < 1 OR CHOICE > 3 THEN 800
255 IF CHOICE = 1 THEN port = PORTA
260 IF CHOICE = 2 THEN port = PORTB
270 IF CHOICE = 3 THEN port = PORTC
275 LOCATE 18, 1: PRINT "ENTER 999 TO QUIT"
280 LOCATE 9, 1: PRINT "Input Decimal word (0 = -5 V and 255 is +5V =>"
300 WHILE VOLT <> 999
305    INPUT VOLT
310    OUT port, VOLT
320 WEND
900 CLS : PRINT "QUITTING"
999 END

800 CLS
810 LOCATE 2, 1: PRINT "ENTER 1, 2 OR 3"
820 GOTO 200
```

Program Description:

Lines 100-190 asks the user for the 8255 Card’s base address and assigns addresses to Ports A, B, C and the Control Word. All three ports are configured with output roles. Lines 200-270 then prompt the user for which output port the DAC circuit is connected to which is Port A in the schematic. Lines 275-320 ask the user to enter a decimal number between 0 and 255. If the user types 999 then the program ends. If you attach a voltmeter across and ground you should read the equivalent voltage. For example, if you entered 255 the voltmeter should read +5 volts. Entering a 0 will read -5 volts. Entering 128 the voltmeter will read close to 0 volts. The Turbo C program operation is identical and its listing follows:

Turbo C Program Listing for DAC Interface:

```
/*
    FILE: DAC.C
    DESC: Prompt user for decimal word. Returns associated analog
voltage
*/

#include<stdio.h>
#include<stdlib.h>
#include<dos.h>          /* outportb, inportb defined here        */
#include<conio.h>        /* formatted text functions defined here */

void main(void) {

   int Volt;
   int BASEADDR;
   int PORTA, PORTB, PORTC;
   int CNTRL;
   int Choice;
   int PORT;

   clrscr();            /* clear screen */
   window(5,5,75,30);   /* set up text window */

   gotoxy(1,1);
   cprintf("Enter Base Address (decimal) e.g. 608\n");
   gotoxy(1,2); scanf("%d", &BASEADDR);
   PORTA = BASEADDR;
   PORTB = BASEADDR + 1;
   PORTC = BASEADDR + 2;
   CNTRL = BASEADDR+3;

   outportb(CNTRL, 128);   /* configure all ports for output */


   do {
      gotoxy(1,4); cprintf("Enter Port Choice\n");
      gotoxy(1,5); cprintf("(1) Port A\n");
      gotoxy(1,6); cprintf("(2) Port B\n");
      gotoxy(1,7); cprintf("(3) Port C\n");
      gotoxy(1,8); scanf("%d", &Choice);
      if(Choice < 1 || Choice > 3) {
          clrscr();
          gotoxy(1,1); cprintf("Enter 1, 2 or 3\n");
      };
   } while (Choice < 1 || Choice > 3);
   switch (Choice) {
       case 1 : PORT = PORTA; break;
       case 2 : PORT = PORTB; break;
       case 3 : PORT = PORTC; break;
   };

   gotoxy(1,18); cprintf("Enter 999 to quit\n");
   gotoxy(1,10); cprintf("Input Decimal word (0 = -5V, 255= +5V \n");
   gotoxy(1,11);

   while(1) {
     scanf("%d", &Volt);
     if(Volt == 999) {
          clrscr(); gotoxy(1,1); cprintf("Goodbye!\n");
          outportb(PORT, 0);  /* quitting */
          exit(0);
     };
     outportb(PORT, Volt);
   };

} /* end of main */_
```

This circuit and your Boondog 8255 PC Interface Card provides an economical yet effective means to do digital-to-analog conversion. You can write code for a function with timing routines and implement your own wave generator. Dedicated DAC boards for the PC cost over $100. Because of your Boondog 8255 PC Interface Card's programmability, you can implement one for less than $10.