LTC1099 Enables PC Based Data Acquisition Board to Operate DC-20kHz

A "C" Cruise through Data Acquisition

Richard Markell

Introduction

A complete design for a data acquisition card for the IBM PC is detailed in this application note. Additionally, C language code is provided to allow sampling of data at speeds of more than 20kHz. The speed limitation is strictly based on the execution speed of the "C" data acquisition loop. A "Turbo" XT can acquire data at speeds greater than 20kHz as can machines with 80286 and 80386 processors. The computer that was used as a test bed in this application was an XT running at 4.77MHz and therefore all system timing and acquisition time measurements are based on the 4.77MHz clock speed.

The board may be used, as designed, for data acquisition and/or it may be used as a test vehicle for incorporating an A/D converter into a system. The optical to digital converter described in AN33 could be integrated into the PC plug-in board to build a simple, yet highly accurate, machine vision system. The fast sampling available with the LTC1099 allows voice quality audio to be acquired and stored in PC memory. This could be useful in a PC based voice analysis system.

This note is broken into two parts. First, the hardware will be described. The hardware description is broken into four sections. Figure 1 details the address decoding section. Figure 2 details the A/D converter section. Figure 3 details the timer circuitry. The D/A section is detailed in Figure 4.

The second part of the application note describes the software written in "C" to control the board hardware. These functions and main programs were written and tested using Turbo C available from Borland International. These programs include: 1) The function named "newatod.c." This function is the main function for using the PC plug-in board for data collection. 2) The function named "ntimer.c." This function controls the 8253 hardware timer on the plug-in board. 3) The function called "dacoput.c." This is a debug program for testing the address decoding circuitry on the plug-in board. 4) The function "atodcon.c." is another debug program allowing the user to test the A/D function of the plug-in board.

Hardware

This application note does not pretend to cover all aspects of interfacing to the IBM PC. The intent is to allow the analog circuit designer to get a data collection system up and running with as little pain as possible. Schematic diagrams and explanations are provided in as much depth as possible, but the designer is encouraged to see the additional sources at the end of this application note for a more complete treatment of interfacing to the PC.

Figure 1 details the Address Decoding Section of the plug-in board. Signals from the IBM PC bus are buffered via 74LS244 unidirectional (U2, U3) or 74LS245 bidirectional (U1) buffers to the expansion board. The direction line of the 74LS245 is controlled from the $\overline{\text{BIOR}}$ line (Buffered IO Read) of the IBM Bus. Therefore, when the PC wants to read data from the plug-in board, the 74LS245 lets data flow in only that direction. Figure 5 shows the correct timing of an LTC1099 read operation as viewed on a logic state analyzer. Data in the write mode can only flow in the opposite direction. Since an A/D converter can only provide data to the PC for the PC to read, an LTC1099 write does not mean that the PC writes to it. An LTC1099 write

strobe switches the LTC1099 internal S/H (sample and hold) to hold mode and then starts the A/D conversion. Figure 6 shows the correct timing of the LTC1099 write operation. The $\overline{\text{Gate}}$ input to the 74LS245 is controlled by U10A such that the '245 is active only when a valid PC plug-in board address ($\overline{\text{ENI/O}}$) and a $\overline{\text{BIOR}}$ or $\overline{\text{BIOW}}$ line is asserted. For non-vulcans this means that the data bus to the PC plug-in board can only be accessed when the computer is trying to read or write to the addresses on the board.

U6 (74LS154) and U8 (74LS688) provide address decoding for the on-board devices that must be read from or written to. U8 is an 8-bit digital comparator that allows only hex addresses 0300 to 031F to be accessable on the board. U6 further decodes the addressing so that we may address each address from 300 to 30F. We could, at a later time, add another 74LS154 should we need to decode addresses 310 to 31F hex.

Figure 2 details the Analog to Digital Converter Section of the board. The LTC1099 8-bit A/D is a half flash CMOS converter with a built-in sample and hold. It can sample at rates to 156kHz. In this application, the data acquisition speed is limited by the software. In the circuit of Figure 2, an analog signal is input to the A/D through an anti-alias-

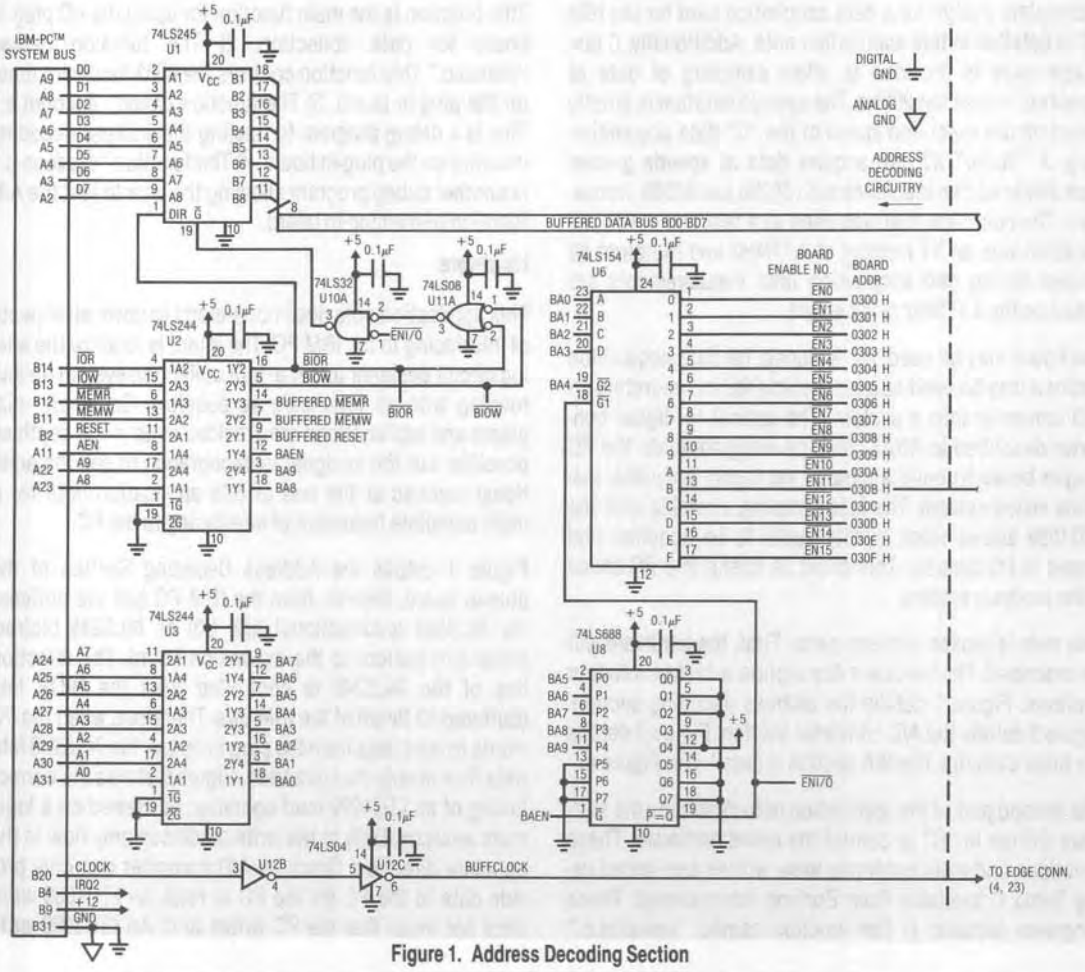


Figure 1. Address Decoding Section

ing low pass filter. U23 (an LT1019-5) is a third generation bandgap reference which supplies the 5V reference to the A/D. The A/D's chip select is asserted when $\overline{EN8}$ and the TIMERCLK' signals are low. U20, a 74LS126, is used to choose which signal starts a conversion. The user can, via the software, choose if the conversion is started by the TIMERCLK' signal or via the $\overline{EN10}$ signal. In a similar manner, the INT select line is used to select whether the TIMERCLK' signal or the INT (Interrupt) signal is used to assert the IRQ2 (Interrupt #2) line of the IBM bus. Carefully note that Figure 2 has both analog and digital ground connections. It is *critically* important to separate these grounds on the board and only bring them together at a single point at the board edge. If this layout is not followed the digital devices will cause switching noise which will appear at the analog device's inputs and cause inaccurate data to be taken.

Figure 3 shows the 8253 timer and associated circuitry. The 8253 is a programmable interval timer with three independent 16-bit counters. The device has six different modes of operation. The mode used to control the A/D in

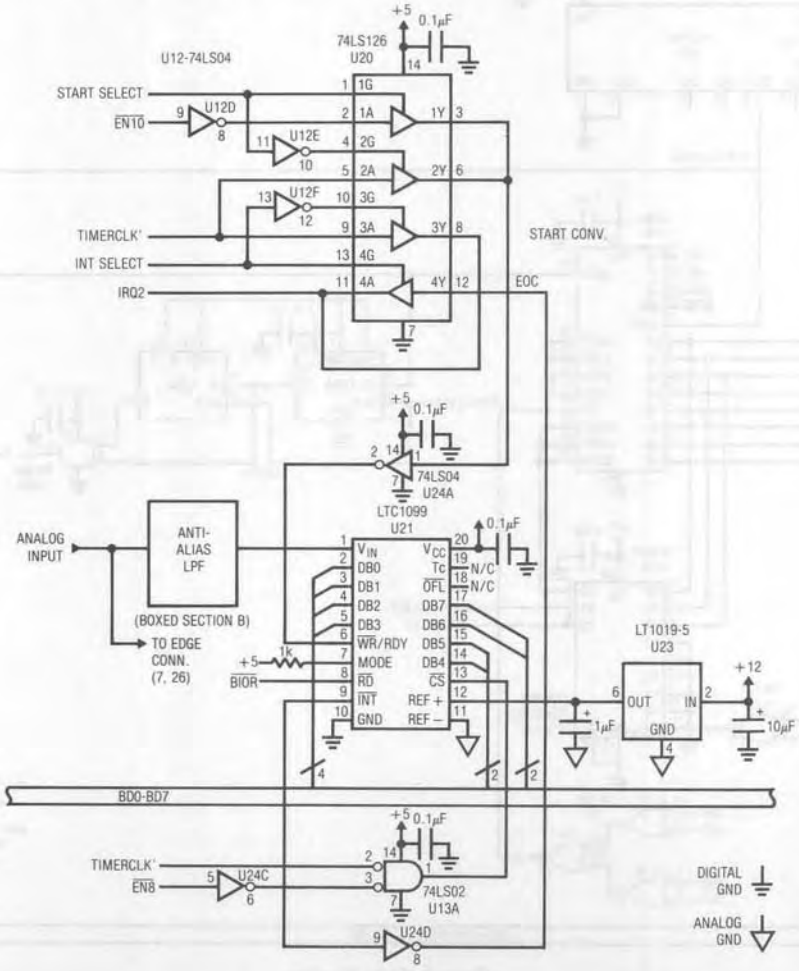


Figure 2. A/D Section

this application note is Mode 2 which produces a string of pulses. These pulses are accurately spaced as they are generated from the 4.77MHz (PC XT) clock signal in the computer. These pulses are used to start the LTC1099 conversion cycles. As seen in Figure 3, U17, a 74LS393 serves to divide the buffered clock signal by 4. This signal, at 1.1925MHz, is then input to the 8253. In this application, the three counters of the 8253 are fed one into the next. This has the effect of providing an output whose period may be divided by up to $2^{48}-1$. Table 1 details the addressing of the counters, the gate ports and the control registers of the 8253. The $\overline{CS}$ must be controllable from the first four addresses shown in Table 1, thus the configuration of U11B, C and D. Figure 7 shows in the top trace, the buffered clock at U12C pin 6 (Figure 1) and, in the bottom trace the 8253 (U15) output at pin 17. The timer was programmed for (50, 'u') or 50 microseconds. Note that this period is correct, as seen in the photo. U18 (a 74LS74) and

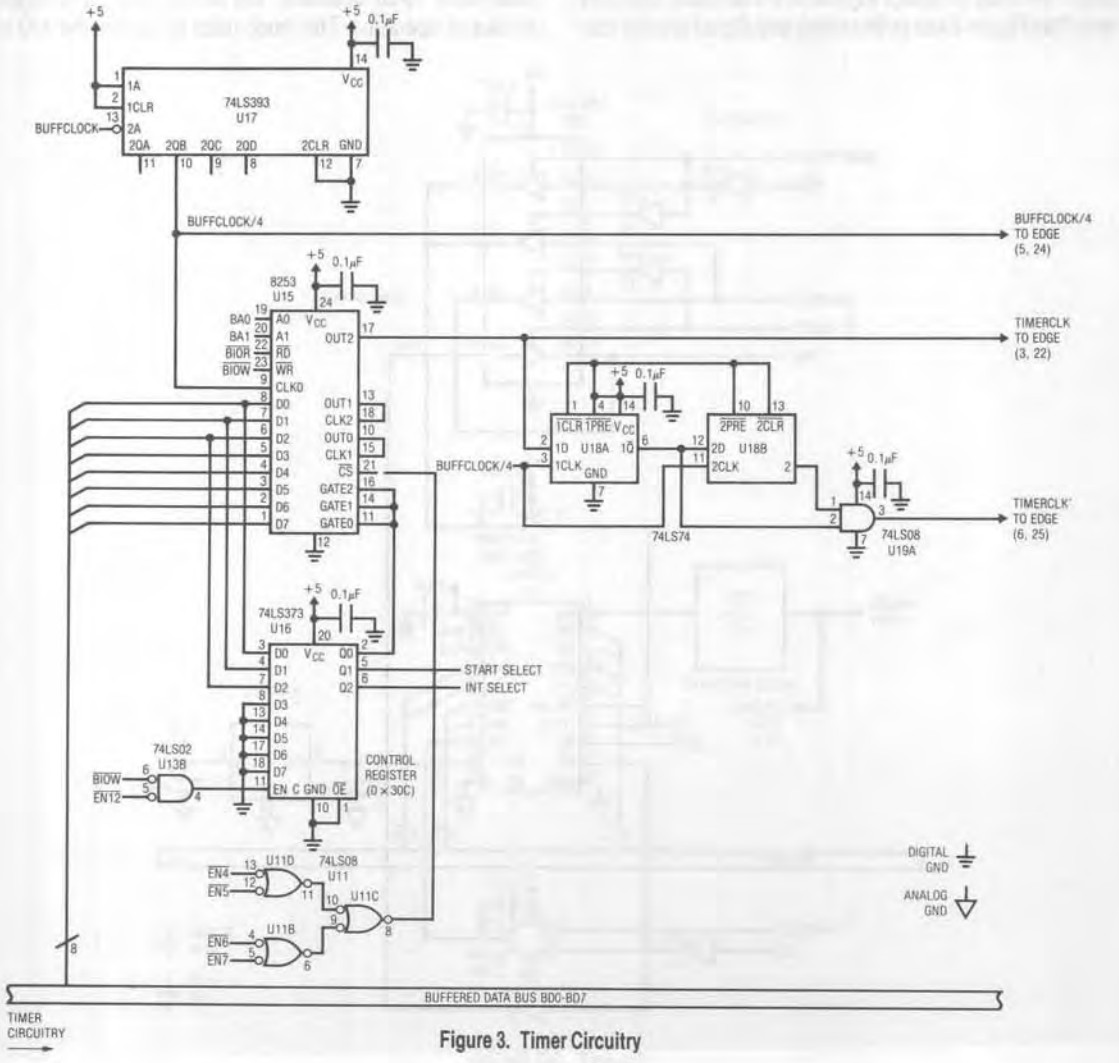


Figure 3. Timer Circuitry

U19A (a 74LS08 gate) are used to process the signal from the timer to produce pulses of pulse width equal to BCLK/4. Figure 8 shows this process. The top trace is the narrowed pulse at pin 3 (U19A) 74LS08. The bottom trace from pin 17 of the 8253 is the unshaped pulse. The next photo, Figure 9, shows more dramatically why the process of pulse narrowing is required. The bottom trace set to 10 microseconds is almost square. Should a digital process start on the falling edge of the clock and continue until the clock rises, this would be almost 5 microseconds for the square wave. This is far too much time for most processors. In the top trace a narrow pulse is only a few hundred nanoseconds wide. This width is what the microprocessor in the PC wants to see. Thus the pulse width processing is necessary for proper circuit operation.

Figure 4 shows the D/A section of the board. It is simply the D/A converter attached to the buffered data bus and the $\overline{CE}$ and $\overline{CS}$ pins connected to $\overline{BIOW}$ and $\overline{EN11}$ respectively. The D/A port is connected to output connector pins to allow analog output from the PC plug-in board.

Table 1. 8253 Timer Port Addresses

PORT ADDRESS	
304 H	8253 Counter 0
305 H	8253 Counter 1
306 H	8253 Counter 2
307 H	Control Register-8253
30C H	Gate Port of 8253 (Bit 0 only)

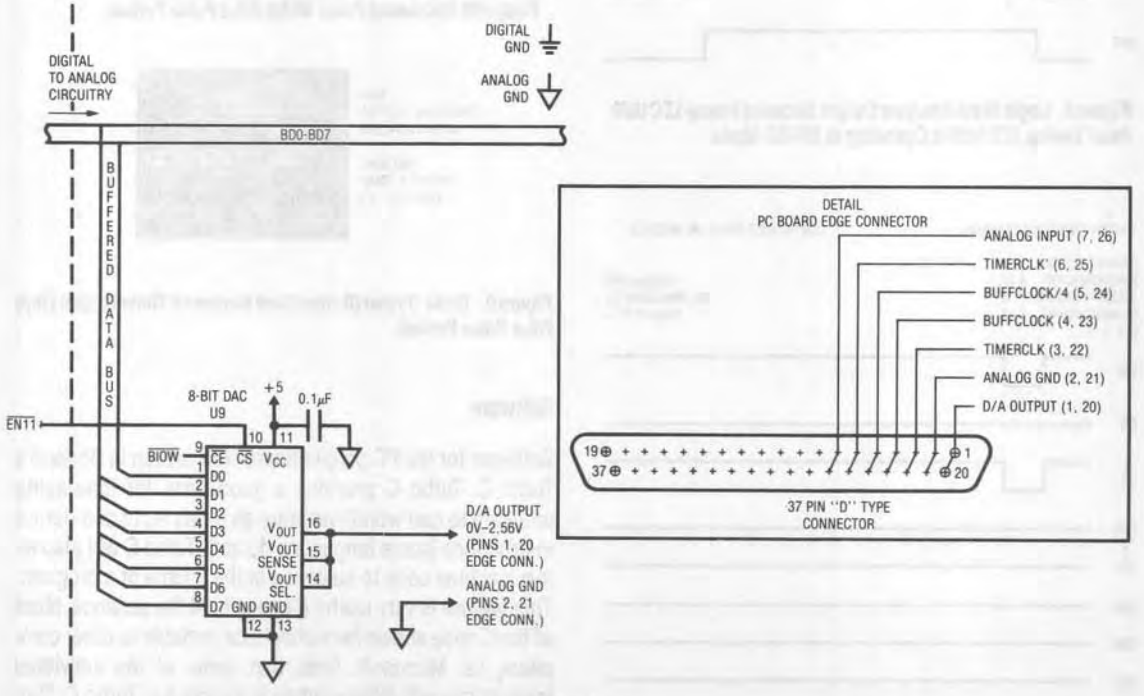


Figure 4. D/A Section

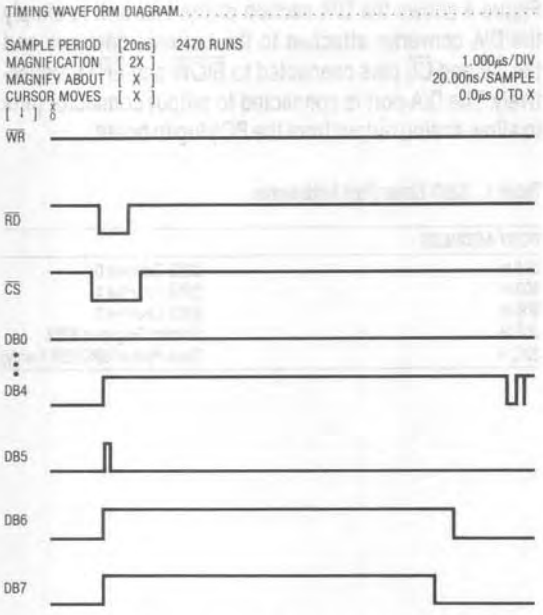


Figure 5. Logic State Analyzer Output Showing Proper LTC1099 Read Timing. LTC1099 is Operating in WR-RD Mode.

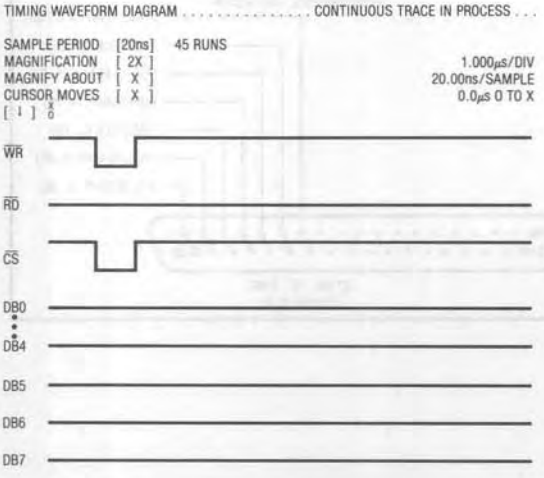


Figure 6. Logic State Analyzer Output Showing Proper LTC1099 Write Timing. LTC1099 is Operating in WR-RD Mode.

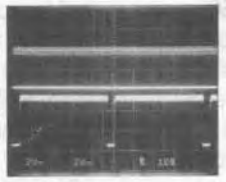


Figure 7. Buffered XT System Clock and 8253 Timer Output

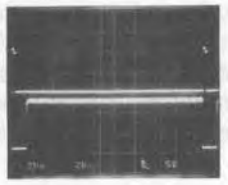


Figure 8. Timer Output (Bottom) and Processed Timer Output (Top) with Decreased Pulse Width (50µs Pulse Period)

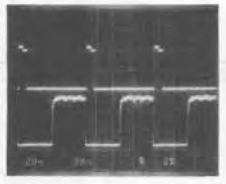


Figure 9. Timer Output (Bottom) and Narrowed Timer Output (Top) (10µs Pulse Period)

Software

Software for the PC plug-in board was written in Borland's Turbo C. Turbo C provides a good base for interfacing software to real world hardware as it has input and output instructions (some languages do not). Turbo C will also allow machine code to be called in the middle of a program. This feature is very useful if speed is of the essence. Most of the C code shown here should be portable to other compilers, i.e. Microsoft. Note that some of the individual instructions will differ slightly in syntax (i.e. Turbo C: Outport; Microsoft: Outp. See either the Microsoft C Bible or another reference for more information).

The individual programs used for data collection and testing of the plug-in board will be described briefly. For more information see the comments interspersed throughout the code and the C references at the end of this article.

Figure 10 details the code called "dacoput.c." The code, when compiled to produce executable code, will continuously output to the D/A converter on the PC plug-in board. This is useful when first testing the board and/or one's knowledge of the C language. The code will produce a ramp which can be viewed on an oscilloscope using the output connector on the PC plug-in board. A photograph of this output is shown in Figure 11.

Figure 12 details the code called "ntimer.c." This is the code used to set up the 8253 timer integrated circuit on the PC plug-in board. The code scans the time unit selected: microseconds, milliseconds, or seconds. Then the division ratio is selected based on the user input. While the code may appear formidable, it really is simple to understand if the potential user calculates the values loaded into the clock[x] registers. Try it!

```
/* This is a test program to test the address decoding circuitry on    */
/* the pc plug in board.  The program outputs continuously to a DAC    */
/* producing a ramp which can be viewed on an oscilloscope.  The program */
/* is called dacoput.c and it must be compiled to an executable file to  */
/* run.                                                                */
/*                                                                     */
/* Richard Markell/Linear Technology Corp./408 432-1900                */
/* Copywrite Linear Technology Corp. 1989.  License to use this        */
/* code is granted when used in conjunction with LTC devices.          */
/* Rev. 1.00/May 8, 1989                                               */
/*                                                                     */
/*  dacoput.c                                                          */

#include<dos.h>
#include<signal.h>
#include<stdio.h>

#define DACO    0x30B   /*This is the DAC port corresponding */
                        /*to address line E11.               */


main()
        {
        int i;

        while(1)
                {               /*Repeat until Control-break hit        */
                for(i=0;i <= 255;i++)           /*Increment i and send */
                dacout(i);                      /*it to DACO.          */
                signal(SIGINT, SIG_DFL);        /*Look for Control-Break */
                }
        }

/*  DACO output  */

dacout(val)
int val;
        {
        outportb(DACO,val);                     /*Send number to DACO  */
        }
```

Figure 10. "dacoput.c" Program Listing

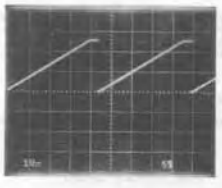


Figure 11. D/A Test Output at Pin 16 U9

```
/* Function to set up Intel 8253 timer to provide timing pulses to  */
/* A to D converter.  Mode 2 of the counter/timer is used to produce  */
/* a string of pulses. In this application the three counters inside  */
/* the 8253 are daisy chained together.  Counter 0's output is input  */
/* to the clock of Counter 1 and Counter 1's output is input to the   */
/* clock of Counter 2.  The output is taken from Counter 2's output.  */
/* See Intel data sheet for more details.                             */
/*                                                                    */
/* Richard Markell/Linear Technology Corp./408 432-1900               */
/* Copywrite Linear Technology Corp. 1989.  License to use this       */
/* code is granted when used in conjunction with LTC devices.         */
/* Rev. 1.00/May 8, 1989                                              */
/* ntimer.c                                                           */

#include<stdio.h>
#include<dos.h>

#define         CNT0    0x304   /*First counter register   */
#define         CNT1    0x305   /*Second counter register  */
#define         CNT2    0x306   /*Third counter register   */
#define         TCNTRL  0x307   /*Timer control word register   */
#define         CNTRL   0x30C   /*Plug in board control register   */

/*main()                        */
/*      {                       */
/*      outportb(CNTRL, 5) ; */ /*Enable 8253 output            */
/*      timer(50, 'u');         */
/*      }                       */


timer(rate, unit)
        int     rate;
        char    unit;
        {
        char    clock[6];
        unsigned int    tocks;

        switch(unit)
        {
        case 'u':       /*Microseconds   */
                {                               /*(1/4.77MHz)=209.5ns */
                tocks = (float) rate / 3.3543 ;  /*209.5ns x 4 x 4 = 3.3543us *
                clock[5] = tocks / 256 ;
                clock[4] = tocks % 256 ;     /* Example: 20 % 256 = 20 */
                clock[3] = 0 ;
                clock[2] = 2;
                clock[1] = 0;
                clock[0] = 2;
                }
                break;

        case 'm':       /*Milliseconds   */
                {
                clock[5] = (rate*2) / 256 ;
                clock[4] = (rate*2) % 256 ;
                clock[3] = 0 ;
                clock[2] = 149 ;
                clock[1] = 0 ;
                clock[0] = 4 ;
                }
                break;

        case 's' :      /*Seconds    */
                {
                clock[5] = (rate*2) / 256 ;
                clock[4] = (rate*2) % 256 ;
                clock[3] = 23 ;
                clock[2] = 75 ;
                clock[1] = 0 ;
                clock[0] = 100 ;
                }
        }
        outportb(TCNTRL, 0x34) ;/* Counter 0, load LSB then MSB, mode 2 */
        outportb(TCNTRL, 0x74) ;/* Counter 1, load LSB then MSB, mode 2 */
        outportb(TCNTRL, 0xB4) ; /* Counter 2, load LSB then MSB, mode 2 */
        outportb(CNT0, clock[0]) ;
        outportb(CNT0, clock[1]) ;
        outportb(CNT1, clock[2]) ;
        outportb(CNT1, clock[3]) ;
        outportb(CNT2, clock[4]) ;
        outportb(CNT2, clock[5]) ;
        }
```

Figure 12. "ntimer.c" Program Listing

Figure 13 is the test program, called "atodcon.c," for testing the analog to digital to analog path through the board. The user inputs data (usually a sine wave) to the A/D via the board edge connector and views the data on an oscilloscope to verify correct operation of the digitization process. Figure 14 shows an oscilloscope photo of a 40Hz sine wave which was input to the A/D and then reconverted to analog via the D/A portion of the board. The speed with which data can be output using this program is not a valid indication of the data acquisition speed of the program called "newatod.c." This is because of the extra time needed to output data with the D/A.

```
/*  This is the main function for testing the A to D, D to A functions    */
/*  on the pc plug in board.  The code below is called atodcon.c.  The    */
/*  user calls the compiled program, pcplug.exe, from the command line    */
/*  while in DOS.  The compiled program contains atodcon.obj and ntimer.  */
/*  obj linked with the various header files.  The program allows the     */
/*  designer to test the hardware by inputting a 0 to 5v signal into      */
/*  the A to D converter and observing the output of the D to A converter */
/*  on an oscilloscope to confirm proper operation of the A to D and      */
/*  D to A functions.                                                     */
/*                                                                        */
/*      Richard Markell/Linear Technology Corp./408 432-1900              */
/*      Copywrite Linear Technology Corp. 1989.  License to use this      */
/*      code is granted when used in conjunction with LTC devices.        */
/*      Rev. 1.00/May 8, 1989                                             */
/*  atodcon.c                                                             */

#include<dos.h>
#include<signal.h>
#include<stdio.h>
#define         STAT    0x309   /*  Protoboard status register  */
#define         CNTRL   0x30C   /*  Control register            */
#define         ADC     0x308   /*  A to D address and data     */
#define         DAC     0x30B   /*  D to A address              */

main()

        {
        char    data;

        outportb        (CNTRL,5);      /* Start conversion; enable 8253  */

        timer (10, 'u');                /*Set timer rate                  */

        while (1 )                      /*Continue until Control-Break hit  */
                        {
                        outportb (ADC,1) ;      /* Start A to D   */
                        data=inportb(ADC) ;     /*Input  */
                        outportb(DAC, data) ;   /*Output */
                        signal (SIGINT, SIG_DFL); /*Looks for Cntrl-Brk  */
                        }
```

Figure 13. "atodcon.c" Program Listing

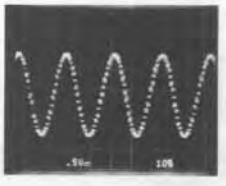


Figure 14. D/A Output for 40Hz Sine Input to A/D (70, 'μ') Timer Setting

```
/*This is the main function for using the PC Plug in board for data     */
/*collection.  This function is named newatod.c.  The user calls the    */
/*compiled program, here titled: npcplug.exe, on the command line while */
/*in DOS.  The compiled progam contains newatod.obj and ntimer.obj linked */
/*with the various header files.  The user must enter on the DOS command*/
/*line:  C:\npcplug dest.xxx.  The destination file may reside on any   */
/*disk in the system, ie dest.xxx may be a:\data5.                       */
/*The program collects SIZE_DATA 8 bit bytes of data. This data is       */
/*then displayed on screen and written to the destination file.          */
/*                                                                       */
/*                                                                       */
/*  Richard Markell/Linear Technology Corp./408 432-1900                 */
/*  Copywrite Linear Technology Corp. 1989.  License to use this code    */
/*  is granted when used in conjunction with LTC devices.                */
/*  Rev. 1.01/June 5, 1989                                               */
/*  newatod.c                                                            */

#include<dos.h>
#include<stdio.h>
#include<stdlib.h>


#define         STAT    0x309   /*  Protoboard status register  */
#define         CNTRL   0x30C   /*  Control register            */
#define         ADC     0x308   /*  A to D address and data     */
#define         DAC     0x30B   /*  D to A address              */
#define         SIZE_DATA       2048    /* Number of samples    */

main(argc, argv)

int     argc;                   /*Command line argument count   */
char    *argv[];                /*Command line arguments        */


        {
        unsigned char   *buffer;        /*Data Buffer               */

        int     outhandle;              /*Destination file handle   */

        int     bytes_written=0, bytes=0, data_points=0;

        unsigned char   data, *ptr;

        if (argc!=2)                    /* Check args               */
          (printf("Format:  C:\\TC\\npcplug dest.xxx "); exit(0););

        if ((outhandle=fopen(argv[1], "wb+" )) == NULL )
          (printf ("Can't open file %s.", argv[1]); exit(0);)

          /*This statement is specific to IBM-PC or clones ( ie non-  */
          /*  portable */


outportb        (CNTRL,5);      /* Start conversion; enable 8253  */

timer (8, 'u');                 /*Set timer rate                  */

if ( (buffer =                  /*Set up data buffer              */
 (unsigned char *)malloc(SIZE_DATA) ) == NULL)
        (
        printf("Memory allocation failed.\n");
        exit(0);
        )

ptr=buffer;

/* Note carefully:  the time between samples set by the timer    */
/* must be greater than the 'while' loop time of execution       */
/* or there may be errors in the acquired data.                  */

while (bytes < SIZE_DATA)    /*Continue until buff full*/

                {
                outportb (ADC,1) ;      /* Start A to D   */
                data=inportb(ADC) ;     /*Input byte      */
                *(ptr++)=data;          /*Put data in buffer  */
                bytes++;                /*Do again        */
                }
printf("\n\n---Contents of file---\n");
ptr=buffer;                             /*Point to start of buffer*/
bytes=0;
while (bytes < SIZE_DATA)

                {
                printf("%4d",*(ptr++));  /*Print buffer to screen */
                bytes++;
                if( bytes % 20 == 0 )
                printf("\n");
                }

ptr=buffer;
bytes=0;
while (bytes < SIZE_DATA)
                {
                if( (bytes_written = fprintf ( outhandle, "%4d",
                        *(ptr++)))==EOF) /*Write data to disk file */
                        perror ("Write failed");
                        else
                        bytes++;
                        data_points = bytes;
                }
                printf("\n\nData Written to File OK. \n");
                printf("\nBytes written = %d ", data_points);

                fclose(outhandle);      /*IBM or clone specific   */
                                        /*statement               */
```

Figure 15. "newatod.c" Program Listing

Figure 15 details the main function for data collection. This function is called "newatod.c." The program collects SIZE_DATA 8-bit bytes of data and writes them to a disk file. The program runs very fast because data is stored to memory until all the data points are collected. Then the data is written to disk. In use the user must type C:\\TC\\npcplug dest.xxx on the DOS command line. "C" recognizes the "\" as a special character, so typing it twice lets the compiler recognize the "\" in the usual DOS manner. If the user is already in the Turbo C directory, TC, then only the command "npcplug dest.xxx" need be typed. (Where npcplug is the compiled program containing the various header files linked with newatod.obj and ntimer.obj.) Note that the data acquisition speed could perhaps be improved somewhat by substituting machine code for the "C" code in the "while" loop starting with the statement:

while (bytes < SIZE_DATA).

The code as written allows data to be collected at maximum sampling rates of approximately 20kHz (with a 4.77MHz XT).

Note also that although all the code has been extensively tested (by an analog hacker), no claims are made as to its optimization. All reader comments are solicited as to writing better, faster, or more analog sounding code.

BOX SECTION A

Analog to Digital Converter Selection

The job of the A/D converter on the PC plug-in board is to digitize voice, transducers and other types of analog system inputs.

The LTC1099 was chosen as the on-board A/D for the PC plug-in board. It is a CMOS part with 8-bits of resolution requiring only 75mW of power. Its total unadjusted error is ±1LSB for the LTC1099 and ±1/2LSB for the "A" part. All timing inputs of the LTC1099 are edge sensitive for easy microprocessor interface.

A major benefit of the LTC1099 is that it contains an integral sample and hold function that allows signals of 156kHz at levels up to 5Vp-p to be directly digitized. This is at least an order of magnitude better than most converters of this type.

The LTC1099 is ideal for Digitial Signal Processing applications such as digital filtering, pattern recognition and FFT signal analysis. In DSP applications, the dynamic characteristics of the converter (be it A/D or D/A) are critical. Dynamic characteristics are usually measured by inputting a spectrally pure sine wave to the A/D and collecting the digitized output data. This data is then analyzed for such things as SNR, Harmonic Distortion and Effective Number of Bits (ENOBs). It is beyond the scope of this application note to discuss every aspect of dynamic signal testing of A/D and D/A converters.

Figures 16 and 17 show FFT plots of the LTC1099 converter. Figure 16 shows the plot for an input signal, f_{IN} = 85.61790kHz, with a dynamically tested SNR of 49.0dB. The next figure, Figure 17, shows the plot for an input signal f_{IN} = 103.898440kHz. Its SNR is calculated to be also 49.0dB. These numbers are remarkable for the fact that the relationship between SNR and resolution is related by the following equation:

SNR = (6.02N + 1.76dB) where N is the resolution of the A/D in bits.

Thus, the tests conclude that the LTC1099 is very close to the theoretical in its dynamic signal to noise ratio. A plot of SNR versus frequency is shown in Figure 18.

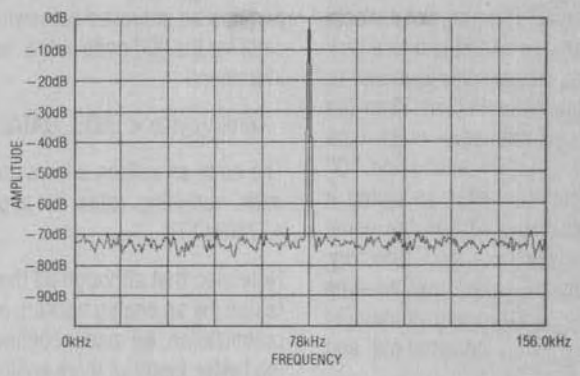


Figure 16. LTC1099 FFT Plot for f_{IN} = 78.144251kHz (0V to 5V), f_{SAMPLE} = 312.000000kHz, S/N = 49.0dB, T_A = 25°C

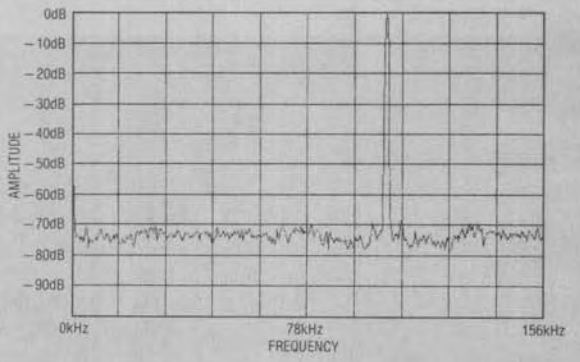


Figure 17. LTC1099 FFT Plot for f_{IN} = 103.898440kHz (0V to 5V), f_{SAMPLE} = 312.000000kHz, S/N = 49.0dB, T_A = 25°C

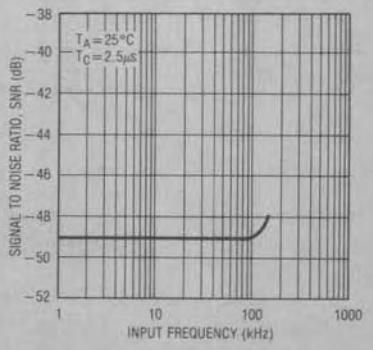


Figure 18. Signal to Noise Ratio (SNR) vs Input Frequency

BOX SECTION B

Anti-Aliasing Lowpass Filter

In all sampled data systems, the phenomenon of aliasing can rear its ugly head. This can, and usually will, happen when the analog input frequency exceeds half the sample frequency. Thus, when we say that we can collect data at 20kHz (and therefore sample at 40kHz), we must have an anti-aliasing filter at the input to the A/D to limit the input frequency range to this limit. Obviously, in the selection of the anti-aliasing filter we must give careful consideration to the input signal's frequency components and the fact that we cannot build a filter with an infinitely steep cutoff slope. Figures 19, 20, 21 and 22 detail some choices for anti-aliasing filters.

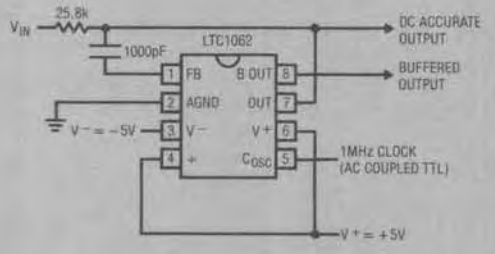


Figure 19. LTC1062 in 10kHz Anti-Aliasing Application

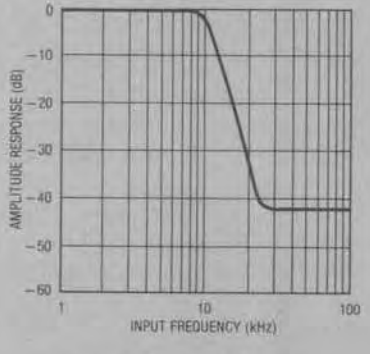
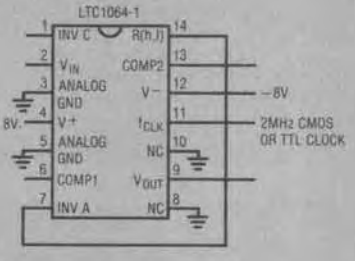
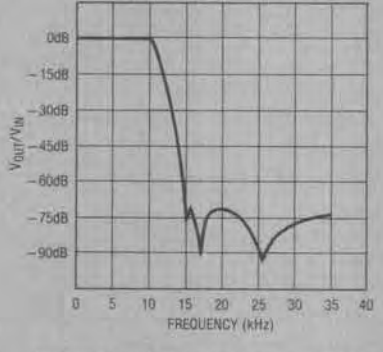


Figure 20. Amplitude Response of Circuit Shown in Figure 19



NOTE: THE POWER SUPPLIES SHOULD BE BYPASSED BY A 0.1μF CAPACITOR CLOSE TO THE PACKAGE.

Figure 21. LTC1064-1 in 20kHz Anti-Aliasing Application



8TH ORDER CLOCK SWEEPABLE LOWPASS ELLIPTIC ANTIALIASING FILTER

Figure 22. Frequency Response of Circuit Shown in Figure 21

REFERENCES

1. "Turbo C," Borland International, 1800 Green Hills Road, PO Box 660001, Scotts Valley, CA 95066-0001
2. "Interfacing Sensors to the IBM PC," Tompkins, Willis J. and Webster, John, Prentice Hall, Englewood Cliffs, NJ 07632, 1988
3. "The Waite Group's Microsoft C Bible," Barkakati, Nabajyoti, Howard W. Sams and Company, Indianapolis, IN, 1988
4. "Interfacing to the IBM Personal Computer," Eggebrecht, Lewis C., Howard W. Sams and Company, Indianapolis, IN, 1983
5. PC Plug-In Prototype Board Manufacturer: JDR Microdevices, 2233 Branham Lane, San Jose, CA 95124 800-538-5000 (This is where you get the prototype board to plug into your computer.)