AN INTEGRATED AUDIO SIGNAL INTERFACE FOR USE IN THE TEACHING LABORATORY

Arthur Lagos and Michael Wagner

Department of Computer Science
University College
University of New South Wales

ABSTRACT- An audio signal interface for the IBM PC-AT is described which was specifically designed for student use in the teaching laboaratory. The interface which is implemented as an IBM PC-AT plug-in board allows the recording and playback of audio signals directly to and from disk files. All functions of signal conditioning, data conversion and the DMA bus interface are integrated on the board which provides for microphone/line inputs and headphone/line outputs.

INTRODUCTION

When a course in *Computer Speech Processing* was first planned at the Computer Science Department of UC/UNSW for 3rd- and 4th-year computer science and engineering students, the question of a suitable laboratory setup needed to be addressed. The open architecture of the IBM Personal Computer in conjunction with the program development environment provided by MS-DOS and C, the graphics facilities of the Extended Graphics Adapter and, very importantly, cost considerations led to the decision to establish a computer speech laboratory based on6 IBM PC-ATs with full analog input and output facilities for acoustic signals.

The specific requirements of a teaching laboratory called for an integrated analog interface with the following characteristics:

- the board is powered by the PC;
- all signal amplification and low-pass filtering in both input and output directions is contained on the interface board without the need for any external devices or cables;
- for analog audio input to the interface, either a microphone or a taperecorder is connected through standard audio connector plugs;
- analog audio output from the interface is provided for headphones, for a taperecorder or both and connections are made through standard audio connector plugs;
- the board has 4 connector jacks for MIKE-IN, LINE-IN, PHONES-OUT, and LINE-OUT;
- the sampling frequency is fixed at 16,000 samples/s, the cut-off frequency of the low-pass filters is 7.6 kHz and quantisation is 12 bits/sample;
- direct memory access (DMA) is used to transfer data to and from the hard disk without restricting the length of the data stream by the size of memory; and
- all amplifier and filter settings are preset using on-board potentiometers that are not accessible to the user.

THE HARDWARE

Overview

The logical functions of the audio interface board are shown in figure 1. For the input direction, the speech signal is conditioned by the input buffer and is then low-pass filtered. Analog-to-digital conversion takes place and the signal samples are transferred to memory using one of the DMA channels.

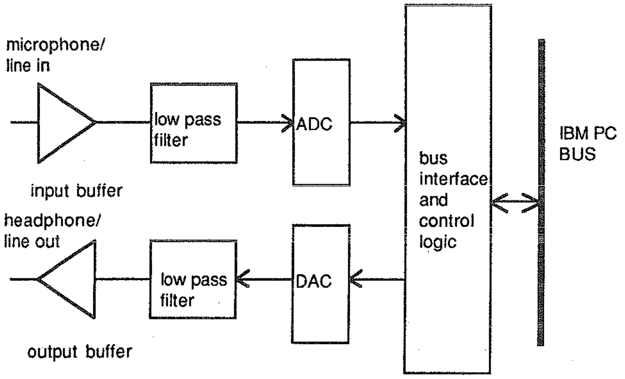

Figure 1. Block Diagram of audio interface board.

In the output direction, samples are transferred from memory to the digital-to-analog converter using DMA. The resulting analog signal is low-pass filtered and amplified to the levels required at the headphones and line jacks by the output buffer.

Input Output Buffers

The input buffer consists of a microphone amplifier with variable gain and a summing amplifier. The summing amplifier mixes the amplified microphone signal with the line-in signal and the resulting signal is fed into the filter stage. The microphone signal gain is set by an on-board potentiometer which is not accessible to the user. Under normal operating conditions, the user will connect either a microphone or a taperecorder for input to the interface. However, the input buffer could also be used in a configuration where microphone and line inputs are mixed.

The output buffer consists of two separate variable-gain amplifiers, one for the headphone signal and one for the line-out signal. The user can therefore listen to the output signal and simultaneously record it on tape. As for the input buffer, the amplifier gains are set by on-board potentiometers inaccessible to the user. A total of four operational amplifiers is used for the input and output buffers.

Low Pass Filters

The anti-aliasing and smoothing filters for the input and output streams are 8th-order Butterworth filters with a cut-off frequency of 7.6 kHz. Each filter is realised in 4 second-order stages where each stage is a biquad with 3 operational amplifiers and an RC network (Wagner & Fulcher, 1986).

Analog-to-Digital and Digital-to-Analog Converters

The converters used in the design were chosen on considerations of performance, cost and availability. For the input stream, the Analog Devices AD574 12-bit analog-to-digital converter is used.

This converter has a number of desirable features such as

- 3-state output buffer circuitry;
- on-chip zener reference for long-term stability; and
- a maximum linearity error of ±LSB.

For the output stream, the Analog Devices AD667 12-bit digital-to-analog converter was used. The features of this converter are characterised by

- a double-buffered input latch;
- on-chip output amplifier and reference; and
- a maximum linearity error of ±LSB.

IBM PC Bus Interface and Control Logic

This section of the circuitry interfaces with the PC bus. In order to allow the transfer of audio data directly to and from the PC hard disk without being limited by memory size, the bus interface and control logic is based on DMA. Since the IBM PC-AT has a 16-bit data bus single-word DMA transfers are used to transfer the 12-bit quantised signal samples to and from memory .

In recording mode, the DMA controller is configured by the software to transfer data from the bus to memory. The 16-kHz sample clock on the interface board is used to initiate an analog-to-digital conversion which, when it is completed, requests a DMA transfer reading the sample from the bus and storing it in memory.

In playback mode, the DMA controller is configured to transfer data from memory to the bus. The sample clock is used to request a DMA transfer reading a sample from memory and making it available on the bus to the digital-to-analog converter.

In addition, the speech board is mapped into the PC I/O address space so that the transfer of data can be enabled or disabled by the software.

THE SOFTWARE

Two drivers have been written for the audio interface board in order to provide basic input and output facilities for audio data. Both drivers are written in assembly language as functions callable from a C program.

For the input of audio data into a disk file, the C-function DAC provides for double-buffered DMA transfer of audio samples to a given file. The calling program provides the file handle and the length of the transfer which is only limited by the amount of free hard disk space.

For the output of audio data from a disk file, the C function ADC provides for double-buffered DMA transfer of audio samples from a given file to the headphones or line-out jacks. The calling program provides the file handle and the start and end pointers for the transfer.

These 2 drivers are designed to be used by the student in the context of a package of graphics and signal processing software which may either be provided as part of the laboratory setup or be developed by the student himself or herself as part of the laboratory assignments and exercises.

CONCLUSION

An audio interface board has been designed and implemented and is currently being used in an IBM PC-AT based computer speech laboratory. The board was specifically designed for student use and provides for analog-to-digital and digital-to-analog conversion of audio signals with a maximum of reliability. This was achieved by deciding on one fixed sampling frequency and filter cut-off frequency, by incorporating all functions of signal conditioning including amplification and filtering on the board, by using student-inaccessible on-board potentiometers for the presetting of all gain and filter parameters, by eliminating all external devices and cabling with the exception of a microphone, headphones and, if required, a taperecorder.

The audio interface board has been used successfully for a course in *Computer Speech Processing* at UC/UNSW.

REFERENCES

Analog Devices (1984) *Data Acquisition Data Book.*

Eggebrecht, L.C., (1983) *Interfacing the IBM Personal Computer.*

IBM (1984) *Technical Reference Manual for the Personal Computer AT.*

Wagner, M., Fulcher, J. (1986) *An IBM-PC Based Speech Research Workstation*, Proc. 1st Australian Conf. on Speech Sc. & Techn., Canberra, 204-209.

A GENERAL PURPOSE SPEECH EDITOR, THE SPEAK LANGUAGE.

H.S.J.Purvis.
Speech Hearing & Language Research Centre
Macquarie University

ABSTRACT- This paper describes a simple computer language known as the speak language that is used in a general purpose speech editor to output words or sentences in a defined sequence with specified timing.

INTRODUCTION

The speech editor is used to record lists of words or sentences in a given sequence and timing. It was decided that the most versatile way of doing this would be to use a simple computer language. The advantage of this method is that it imposes fewer limitations and permits the more complex tasks to be performed easily. The language can be extended as required to meet future needs. This would be difficult if a less flexible method were used. The disadvantage is that A program must be written and this makes the simpler applications more difficult than they might have been if some other method was used.

IDENTIFYING THE SOUNDS

The speech editor is used to divide a speech file into sections that contain words, sentences etc., each section has a beginning and end marker.

A set of these sections is known as a level, and is given a letter from 'A' to 'Z'. Level 'A' consists of one section whose beginning mark is at the start of the file and end mark is at the end of the file. Levels 'B' to 'Z' consist of a number of sections, numbered from 1 to a maximum of 10,000. There is no relationship required between sections at different levels, a level 'C' section may be within a level 'B' section, they may overlap, or they may be unrelated.

Some speech editor commands operate on all sections with a given level, for example the command to adjust the RMS levels of sections to a common value.

The speech editor provides for one or two speech channels, so a channel number is required to complete the identification, the channels are numbered 1 and 2.

The identification of a sound consists of four parts, the data set name, the level, the number, and the channel.

```
Data_Set = Jilly Level = B Number = 1 Channel = 1
```

The term data set refers to the four files that contain the speech and related data. They are the header file containing the fixed information such as sample rates, block sizes, titles etc. the wave file that contains the speech data, the computed data file that contains derived data such as the intensity curve or the pitch curve, the description file that contains the marks used to divide the speech data into sections

THE PROGRAM

The program is written using a text editor. It consists of one to three sections, the variables section, the definitions section, and the procedures section. The procedures section is required but the other two sections are optional. Here is an example of a program that contains all three sections :-

```
Variables
{ All variables are 4 byte integers and MUST be defined. }
{ all statements must end with a semicolon ';' }
Count  Number;
Definitions
{Sound definitions enable a sound to be referred to in a speak command  by its NUMBER or by the use
of a variable. }
Define  Data_Set=Jilly Level=B Number=1 Channel=1;  { number 1 }
Define  Data_Set=Jilly Level=B Number=2:4 Channel=1; { numbers 2 to 4 }
Procedures
Procedure Part_1
[
        Speak Left=3            Spacing=5.0;
        Speak Left=Number       Spacing=1.0;
        Speak Left=2            Spacing=1.0;
]
{ In every program there MUST be a procedure called Main. }
Procedure Main
[
        Left_Data_Set           := Jilly; { These 4 variables are built in }
        Left_Level              := A;
        Left_Number             := 1;
        Left_Channel            := 1;

        Count := 3;             { Assignment }
        Number := 0;

        Repeat Count Times
        [
                speak Left=1                    Spacing=0;
                Inc Number;                     { Number := Number + 1  }
                Part_1;             { call procedure Part_1 }
        ]
        Speak Left=(Data_Set=Jilly Level=B Number=1 Channel=1) Spacing=5.0;
]
        End
```

Comments are enclosed in curly brackets {....} these are ignored by the speech editor.