Inter-Office Memorandum

To	Audio Interest	Date	January 9, 1981
From	L. Stewart	Location	Palo Alto
Subject	A Simple Audio Terminal	Organization	PARC/CSL

XEROX

Filed on: [Juniper]<Stewart>Audio>DAudioTerm.press

Introduction

This note describes a simple full duplex audio terminal first assembled in December 1979. As currently implemented, the device connects to the Diablo printer port of either an Alto I or an Alto II, or to the printer port of a Dolphin. The design could be easily adapted to other machines having a parallel port and capable of polling at least every 125 microseconds. The audio terminal supports simultaneous input and output streams at 8000 samples per second, 8 bits per sample, m-255 law encoded. This data format is fully compatible with the telephone industry and with the m-law format of the *Auburn* audio board. The audio device currently attached to the audio terminal is a Danray telephone set.

More information

Information about the terminal is stored on [Juniper]<Stewart>Audio>. Relevant files are:

CATFiles.dm	Source files, and Alto microcode for bcpl
CATAudioMC.dm	Alto microcode compatible with *Auburn*
CATIIRun.dm	.run files for the Alto II
CATIRun.dm	.run files for the Alto I

Schematics and operating instructions are not available in machine readable form. Contact the author for more information.

Hardware

The audio terminal as configured for the Dolphin contains 12 DIP packages and around 25 discrete components. A 14 signal (plus grounds) cable goes to the printer port and a 6 wire cable to the telephone set. The device requires about 500 mA of +5 volts, 250 mA of +12 volts, and 150 mA of-5 volts.

The hardware centers around an Intel 2910 Codec (coder-decoder) chip, which contains the Analog to Digital and Digital to Analog converters and m-255 companding circuitry.

On the analog side of the codec, an Intel 2912 PCM line filter chip contains input and output lowpass filters and output amplifier. The 2912 is currently connected directly to a 4-wire telephone set (*Danray*), but through the use of a hybrid circuit or SLIC (*Subscriber Line Interface Circuit*) could be connected to a 2-wire telephone.

The digital side of the codec is designed to connect directly to a 24 channel T1 PCM bus - the chip contains time slot counters and serialization and de-serialization logic. The codec requires a 1.536 MHz clock (T1), a Frame Sync pulse every 192 clock pulses. (192 is 24 8-bit time slots), and time slot assignment information. A special case of the time slot assignment facility is used to assign the codec to time slot 0 - immediately following frame sync. The codec also provides a time slot gate signal described below.

The audio terminal includes a Motorola K1115A crystal oscillator device operating at 6.144 MHz and a divide-by-4 circuit (74163) to generate the 1.536 MHZ clock. A further divide-by-192 curcuit (two more 74163 parts) produces the frame sync signal.

The processor interface of the audio terminal consists of an 8-bit input shift register, an 8-bit output shift register, and a ready flip-flop. Every 125 microseconds, the codec generates and accepts a serial 8-bit PCM sample over its input and output T1 busses. These samples are respectively clocked into and out of the input and output shift registers by a clock generated by gating the main 1.536 MHz clock with the coded time slot gate signal (7400). The end of the time slot gate signal activates the *Ready* flip-flop (7474). The processor must read the input sample and set the new output sample during the approximately 110 microseconds between *Ready* and the next frame sync.

The output shift register is a 74165, which is parallel-loaded from the computer output port by an output port bit used as a strobe. The processor must set up the data bits, turn on the strobe bit, and then turn off the strobe bit before changing the data. *Strobe* also clears *Ready*.

The input shift register is a 74164. Its 8 parallel outputs are valid from end-of-time slot until the next frame sync. Because the Alto I has only 6 input bits available, a 74153 multiplexor is used to select first the most significant four bits and then the least significant four bits of the input shift register. The multiplexor is set to select the high four bits by *Ready* and then set to select the low four bits by *Strobe*. The four multiplexor outputs occupy four bits of the processor input port and the *Ready* signal occupies a fifth. WHhen the Audio terminal is configured for a Dolphin or other processor with an 8-bit input port, the eight parallel data lines are enabled to the port through a 74LS244 octal buffer. (The buffer is required since the Dolphin uses bi-directional data lines.)

In order to reduce synchronization difficulties, the Ready signal is delayed for one half cycle of the 1.536 MHz clock by the other half of the 7474 before connecting to the processor input port. This permits the high order 4 bits of the input sample to be accessed in the same reference that detects *Ready*.

The cables required to connect to an Alto I or an Alto II are slightly different, due to the different printer port connectors on the two machines, but the same bits of the printer port memory locations are used in each case. The cable wiring is different again for the Dolphin, and the port bits are different as well.

Software and Microcode

Alto Microcode

As is implied above, the audio terminal is single buffered; it becomes ready every 125 microseconds and allows at most 110 microseconds or so for service. To meet these requirements, the Alto Memory Refresh Task, which runs roughly every 39 microseconds, has been modified to poll the audio terminal. Each invocation of MRT checks for the presence of *Ready*; if it is not set, the task handles its other duties and blocks. If *Ready* is set, MRT reads one sample and writes one sample to the audio terminal.

Due to the availability of R registers on the Alto, it is necessary to make MRT be RAM-related in order for it to drive the audio terminal. On an Alto II this is done by adding a backplane jumper between MRTActive and TaskA. On an Alto I it is not so easy because MRTActive does not normally come off the control board. A control board patch; however, is fairly easily added to bring MRTActive to a backplane pin.

The current microcode interprets two lists of *Audio Control Blocks*, one each for input and output. Each ACB contains four words: a completion flag, a pointer to the next ACB, a pointer to a main memory data buffer, and a buffer length. Audio data is stored in memory as two samples per word left to right. Emulator microcode routines (JMPRAM) are provided to initialize the microcode, to start or stop the interpretation of ACB chains, and to enqueue a new ACB on the end of an existing chain.

Slightly different microcode is required for the Alto I and the Alto II due to the differences in memory refresh on the two machines. The emulator interfaces are identical.

The Alto microcode appears to use 10 to 15 percent of the machine when it is active, although this has not been precisely measured.

As an experiment, microcode was written which is compatible with the emulator interface of the Alto *Auburn* audio board. This version of the microcode works with all Auburn software - in particular, it works with the Mesa *Audio* program.

Alto Software

At present, there are three bcpl programs which exercise the audio terminal hardware and microcode. ATTester.run uses single input and output ACBs linked to themselves. The two ACBs share the same memory buffer so that the program serves to echo the audio input to the audio output while remembering the last several seconds of audio in its internal ring buffer. The input stream can be disabled so that the output stream will cycle repetitively through the ring buffer.

PlayFile.run and RecFile.run, respectively, play and record audio files from and to the local disk. Both programs accept a file name on the command line, using the GP package. RecFile expects the user to type any character on the keyboard to start recording and another to stop.

D0 Microcode and Software

There is currently no Dolphin microcode for the audio terminal. The general idea would be to use one of the Dolphin *timers* to poll the terminal at approximately 100 microsecond intervals. A mesa program CATEcho.mesa is available for the Dolphin which uses the Mesa bytecodes RPrinter and WPrinter to "manually" operate the terminal.

Due to the differences between the Alto Diablo printer port and the Dolphin printer port, the audio terminal operates somewhat differently on the Dolphin than what was described above for the Alto. Three output-only port bits are used as a *device-address* which conditions the terminal to pay attention to the Dolphin. Three other output-only bits are used to select a function: read data from the terminal, read status from the terminal, or strobe data to the terminal.

Comments and Issues

Control functions

The current design does not have any facilities for detecting off-hook, touch-tones or dialing, and does not provide any facilties for outgoing control functions such as for operating a DAA or audio switching circuits. Such digital inputs and outputs could be added if additional processor port bits were available (and at some cost in complexity).

Multiple codecs

The current design is not directly extensible to multiple codecs operating simultaneously. Buffering and time slot assignment logic would probabnly be needed, which might well change the whole character of the design. Multiple codecs operating one at a time (really a form of audio switching) could be accommodated by adding some digital control functions, as above.