SignalMaster – Concept Prototype

OHSSPDS

Open Hardware & Software Signal Processing Development System

Specification Sheet:

Applications: Speech and Biosignal processing capabilities. Potential applications include speech processing for hearing aid processing algorithms and acquisition of biosignals such as evoked potentials and otoacoustic emissions.

Processing Family: Texas Instrument TMS floating point family.
Version 1 Prototype based on TMS320vc33 processor.

Processing Speeds and Memory: 256K bytes of internal memory and addressing lines providing access to 256M Words of external asynchronous memory. Real-time processing capabilities with sample-by-sample input and output capabilities. Latencies depends on processing delay line for filters and FFT processing buffer (typically under 10 ms).

Data Sampling Rate: Programmable with rates of up to 128kHz.

Communication Speeds with programming and interface PC computer:
Version 1 Prototype: USB 2.0 at 480 Mbps

Analog Input: Microphone inputs with drivers
Version 1 Prototype: Two 24 bit A/D convertors with direct input with 144.49 dB range..

Analog Output: Stereo headphones with drivers
Version 1 Prototype: Two 24 bit D/A convertor channels with headphone drivers.

Hardware Controls:
Version 1 Prototype: Digital I/O to memory mapped components, 24 bit programmable register for user add-on hardware control and Serial Bus Interface.

Power Supply: Options to run off battery or USB based power while connected to PC.

User Hardware Expandability: User provided hardware interface region with fused power supply lines with +/- 15 volts and +3.3 volts (1 amp each) and ground signals. Serial bus I^2C and I^2S (Phase II), and 24 bit Digital I/O port will provided for additional controls and triggers to other devices.

Isolation and Safety: Fully isolated medical grade fulfilling IEC 60601 safety testing requirements.

Programmability Options:

Basic-Level: Use of provided applications while connected to a PC.

High-Level: User developed PC applications with interface to DSP hardware using provided dll components with any user selected development language (including MatLab, LabView and other programming languages that support dll interfaces such as Delphi (Visual Pascal), Visual C++).

Low Level: c language based programming with direct control of all hardware aspects.

For additional information, visit: http://www.ihsys.com/ohsspds/index.asp

General System Information:

The use of a DSP allows the proposed system to process data on a sample-by-sample basis while implementing various real-time processing algorithms such as Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filtering and Adaptive Noise Cancellation

(ANC) (Widrow & Steams, 1985; Ferrara & Widrow, 1982). The proposed architecture has been previously used to successfully implement a multi-microphone ANC system for otoacoustic measurements in noisy nursery environments (Ozdamar et al., 1998; Delgado et al., 2000). Using additional microphones, different sources of noise contaminants can be acquired and the corresponding signals eliminated using an ANC algorithm. This algorithm is similar to that used in feedback noise elimination and active noise cancellation. The algorithm automatically adjusts the weights associated with a signal delay network used to adjust for amplitude and phase difference between the noise at the primary microphone and other recording microphones.

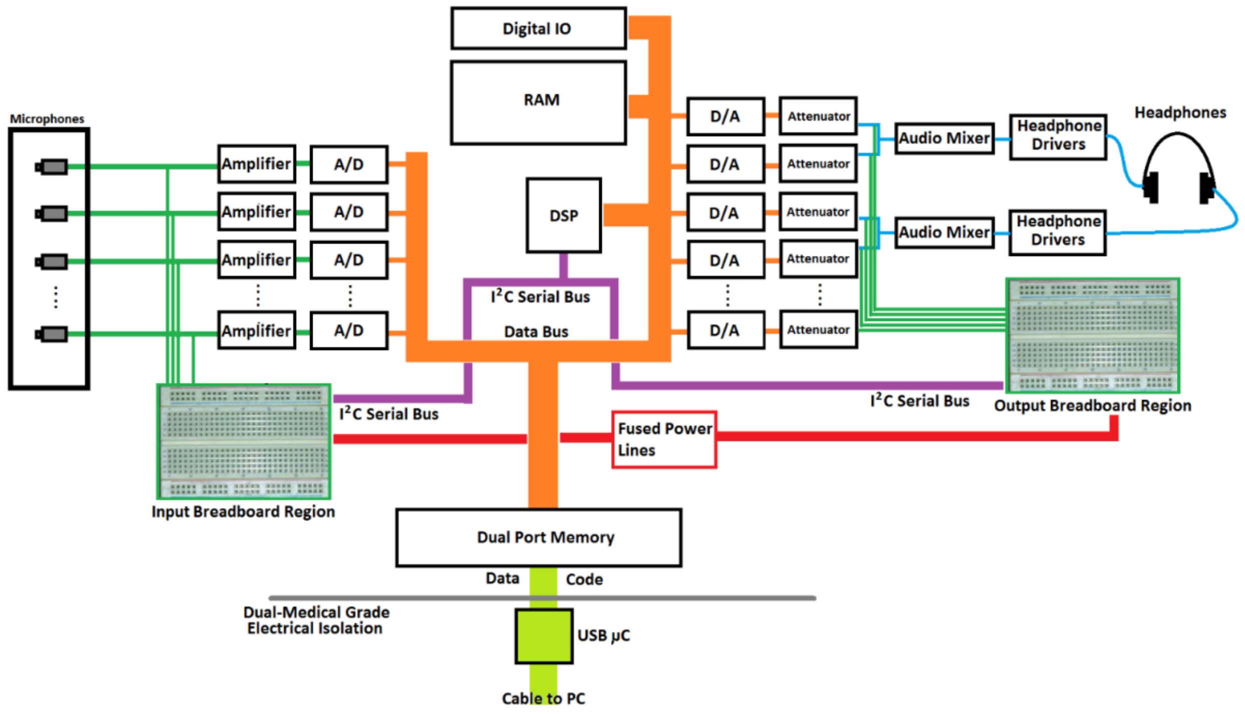

Figure 1: Ultimate system configuration showing multiple input and output channels and control lines.

Innovation:

Hardware: The general hardware configuration is shown in **Figure 1**. During Phase I, the hardware design will be based on the Texas Instrument (TI) 320C6727B 32/64 bit floating point 350 MHz DSP processor, capable of performing 2100 million floating point operations per second (MFLOPS) with a 2.8 ns cycle time, 256K bytes of internal memory and addressing lines providing access to 256M Words of external asynchronous memory (Texas Instruments, 2014). Unlike other data acquisition systems that multiplex Analog-to-Digital (A/D) converters for multiple channels generating a sweeping time difference among channels, the proposed hardware will provide an independent A/D converter and sample and hold circuitry for each input channel allowing for synchronous sampling of all channels. In addition, a programmable amplifier will also be provided for each channel with various gain stages. The hardware is capable of outputting and recording signals at rates of 128 kHz allowing for high frequency acoustic research. The circuit board will also provide sufficient space to incorporate user designed add-on circuitry with access to the I^2C and I^2S serial buses in order to control other external devices. In order to facilitate research in a wide range of areas such as hearing aid processing (Hickson, 1994; Moore et al., 2010, 2011), the proposed hardware will initially provide options for up to four output channels and eight input channels. The output channels can be combined into two acoustic channels, in order to mix stimuli with noise ipsilaterally through headphones or independently to four individual speakers. In addition, the output channels will provide analog programmable attenuator with a 150 dB range. Most other commercial devices can only provide digital attenuation with an effective range limited by the bit resolution of the A/Ds which also eliminates the fine details of the output signal as bit resolution is reduced during the attenuation process. Subsequent designs will provide for additional outputs in order to conduct localization studies in multi-speaker surround sound applications. The use of memory mapping of A/Ds, D/A, amplifier gain registers, and attenuators in the hardware architecture will remain consistent across future hardware versions in order to facilitate compatibility and future software development across platforms.

Software: The software is composed of two primary components: 1) the DSP software and 2) the main PC system software. The DSP is a “slave” to the PC, indicating that it provides specific services and follows the commands of the PC, by executing code downloaded from the PC in order to acquire data, output signals and process signals in real-time, and interface with software modules on the PC. The DSP software is also composed of two components: 1) an interrupt driven routine responsible of data output, acquisition and initial level of processing and 2) the main DSP program responsible for monitoring of PC commands, data transfers to and from the PC, and performing data block based processing algorithms simultaneously. This architecture provides a wide range of flexibility by allowing various applications and processing strategies to be uploaded at any time. By uploading different DSP modules, either made available from IHS, shared libraries or user developed, the processing strategies implemented can be customized for any application. After programming, the system can be disconnected from the PC and the DSP will continue to operate from battery power.

1) Basic Data Acquisition and Processing Option: At the most basic level, the system will provide the ability to generate and record data for either offline processing using other programs such as MatLab or in real-time using the TF32 speech processing module developed by Dr. Paul Milenkovic from the University of Winconsin, Madison, WI. TF32 provides real-time 32 channel scrolling data displays with spectrographic displays, pitch analysis, format tracking, RMS/dB trace, LPC inverse filter with glottal parameter measurement, time-frequency grayscale spectrogram with editable formant track overlay, text labels, time-slice spectrogram (Fourier, LPC, moments), and voice perturbation (jitter, shimmer, periodicity SNR). TF32 is provided as a stand-alone software package or an OCX library package that can be incorporated into any high level programming language. Intelligent Hearing Systems has been collaborating with Dr. Milenkovic for more than ten years in the development of the speech analysis aspect of AACT, a behavioral coding and analysis software package (see Integration of Speech Analysis Tools section below for details).

2) High Level Programming Option: At the next level, the user will be able to interface directly to the system using a dynamically linked library (DLL) providing control functions for direct control of stimulus generation and data acquisition using a wide range of programming languages. Languages that support dlls include most high level programming languages such as C++, Delphi (Visual Pascal) and Visual Basic. Other development packages that also support the use of dlls include MatLab and LabView. In addition, a TF32 Visual Component Library (VCL) will provide full TF32 data processing options as previously described that can also be incorporated into any user developed application. TF32 also provides for user developed code that can be incorporated into the data processing routines to customize different aspect of the provided speech processing routines.

3) Low Level Programming Option: At the highest level of sophistication, the user will be able to directly program the internal DSP options in order to manage the real-time processing capabilities of the system. The system DSP provides a very powerful tool that allows for the manipulation of input and output data on a sample-by-sample basis. The DSP is programmed in C, using a chip set specific TI C language compiler, which TI provides for free for instructors and students. IHS will provide an application to setup the required compilation command files specific for the hardware architecture being used facilitating the software development process. Although this level of programming requires more software development experience, the benefits and level of control achieved cannot obtained using other options. During Phase I, examples will be provided to common data acquisition and analysis problems associated with general signal and speech processing. The users will be able to simply cut and paste the provided code into their own code to obtain the necessary performance functions.

Table I – Data Processing Levels:		
Sample-by-Sample DSP Processing:	**Sample-by-Sample Preprocessing & Block-Based Higher Level DSP Processing:**	**Higher Level PC Based Processing:**
DSP Interrupt Routine: **(Clock Dependent)** Acquire Data Process Data Output Data	**Interrupt Routine:** **(Clock Dependent)** Acquire Data into Buffer(n) Pre-Process Data Output Data from Buffer(n-2) **Main DSP Routine:** **(Block Size Dependent)** Read Filled Data Buffer(n-1) Process Current Buffer (FFT and other high level functions) Store Processed Data Buffer for output Repeat with next buffer	DSP Data Input | Data output to PC through USB Connection | PC Data Processing | Data return to DSP through USB Connection | DSP Data Output
The signal processing latency is limited to the sampling period. Data is acquired, processed and the corresponding output generated within the interrupt cycle. Applications include FIR & IIR filtering, ANC and other sample-by sample applications.	The signal processing latency is limited to the length of the processing buffer queue. For example, at a rate of 40kHz and using a 512 data point buffers, the latency would be 25.6 msec using this particular dual buffer scheme. The length of the buffer can be adjusted to conform with the latency that is appropriate and acceptable for the specific application. Applications include spectral signal	The signal processing latency is increased and is limited by the USB transfer rates and PC processing speed. Although these processing latencies might not be optimal for all applications, it provides for a fast initial prototyping requiring less software development and implementation of canned processing routines as those available in TF32, MatLab and other packages.

	modulation as in frequency and amplitude compression (Stone and Moore, 2008; Souza, 2002).	

Processing Time: One major consideration of the proposed design is that of signal processing latency or in speech processing applications, audio latency, the time required between a signal input and the processed signal output. The High-Level programming interface allows for substantial amounts of processing to be conducted at the PC side, with an increase in latency limited by the processing speed of the PC, but primarily to the transfer rates. The Low-Level programming interface allows for the development of processing algorithms that can be executed on a sample-by-sample basis. Data sampling rates of 40kHz provide for a 25us sampling rate period. This period of time is sufficient to conduct filtering, noise cancellation algorithms and artifact rejection. The sampling rate can also be programmed and reduced to 10kHz, still sufficient for most speech processing applications, and therefore increasing the sample rate period to 100us, further increasing the allowable processing time. Because of the system's interrupt driven architecture, data can be processed on a sample-by-sample basis while blocks of data can be processed simultaneously. Fast Fourier Transfer (FFT) routines can be implemented on such blocks of data as well as other more advanced processing routines. The signal processing latency can be adjusted by selecting the duration of such buffers as required to accomplish the desired processing functions. **Table I** shows the various level of data processing available under the proposed design. Each one of the options shown in Table I can be implemented using either user developed code or code provided by IHS as source code or precompiled, using the three programming strategies highlighted above.

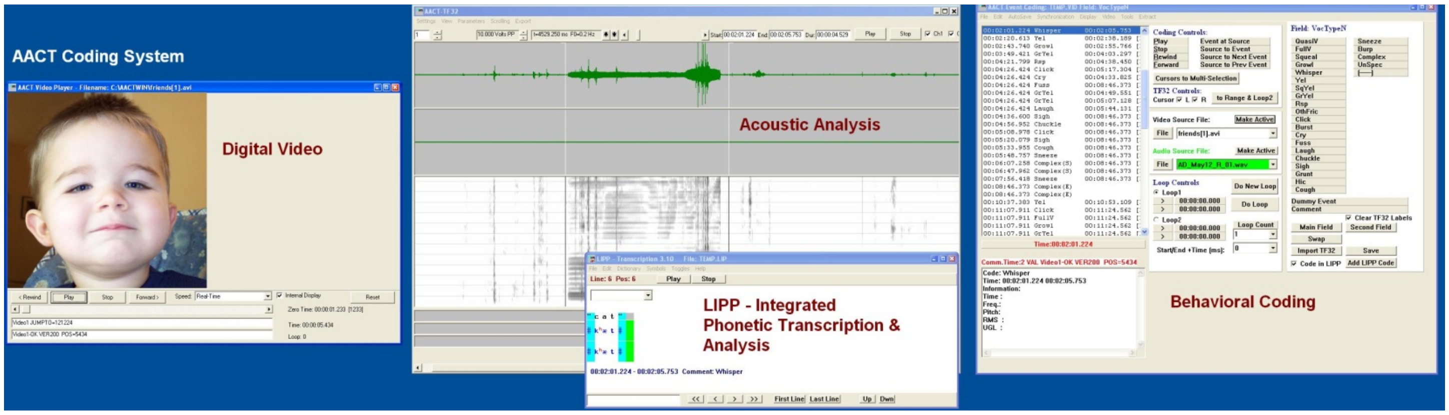

Figure 2: Complete AACT System display with digital video display control (**Left**), TF32 acoustic analysis and LIPP Phonetic coding (**Center**) and behavioral coding screen (**Right**).

Integration of Behavioral Coding and Speech Analysis Tools: The proposed complete package will also provide an option to integrate important behavioral coding and speech analysis tools such as TF32, AACT (Action, Analysis, Coding and Training) software and LIPP (Logical International Phonetics Program). AACT, shown in **Figure 2,** is a flexible behavioral coding system that allows for the coding over a wide range of domains as defined by a user. These can include gestures, affect, gaze as well as different speech elements such vocalizations and intonation. The behaviors and speech elements can be coded using precise time codes extracted from a frame accurate digital video control module. The sound tracks are displayed in real-time along with the video. The user can select from a wide range of analysis display options using an integrated speech analysis package, TF32. In addition, a comprehensive phonetic transcription and analysis package**,** LIPP, can also be used for additional data entry. The integration of these tools will further facilitate research in areas involving other behaviors that need to be analyzed with precise timing in conjunction with speech data.