

uEAC Firmware Technical Profile

Bryce Himebaugh, Indiana University Computer Science Department

1.0) Abstract

This document describes a technical framework that will be used by three IUPUI senior design students to develop research software used in the Extended Analog Computer project at Indiana University. As a technical profile, this document is intended to communicate the features that are significant to the research utility of the developed software while leaving much latitude in the implementation detail. The hope is that this flexibility will leave the design space open for the students to explore.

2.0) Background

The micro Extended Analog Computer (uEAC) is a device that has been designed at the Indiana University Computer Science Department to allow research into continuous valued computation. This machine is a hybrid computer that couples the flexibility of a digital machine with the computational properties of a gradient manifold system.

The digital portion of the uEAC is managed by the TI MSP430F169 microcontroller. The MSP430 is responsible for bridging the gap between the analog hardware and a host computer. The host computer communicates with the MSP430 through a virtual serial port that physically runs over USB. An FTDI FT2232 device along with host side drivers provide the virtual serial port functionality. The virtual serial port abstraction allows the host to communicate with the MSP430 in the same manner as a standard serial port.

The analog portion of the uEAC includes a 5x5 grid of pins that are physically connected through a single piece of conductive foam. Each pin in the grid can be individually setup by the MSP430 as a current source, current sink, voltage sense, or current sense. The MSP430 uses SPI serial DACs to control the magnitude of the current flowing at each pin defined as a current source or sink. The MSP430 uses banks of analog multiplexers to read the voltage at the pin circuitry when the pin is defined as a voltage sense or current sense. The current sense functionality causes a 1k resistor to be connected to ground at the pin of interest. The voltage across the 1k resistor is measured to determine the current flowing into the pin.

Software running in the MSP430 interprets commands from the host computer to allow the host to specify pin configurations. This interpreter takes ASCII strings that represent a simple command language and translate them into actions related to the analog pin matrix. This simple command language will be designed as part of this project to provide a clear, concise method of specifying pin architectures. While the details of this language are largely left as an implementation detail, human usability should be considered in the design as in some cases the interaction with the uEAC will be through a simple terminal program on the host. However, in most cases the interaction will be through a script or application running on the host.

While this project centers on designing the command language and implementing the interpreter software that runs on the MSP430, there are two other software components that are needed to make the uEAC useful when the machines are deployed at research facilities beyond IU or IUPUI.

The first piece is a bootloader. This is a low level piece of software that stays resident in the MSP430's flash memory. The function of this bootloader is to allow the host to download updated interpreter software to MSP430 flash memory as the interpreter development will progress over time. The bootloader prevents the researcher at external facilities from needing to install any of the MSP430 tools that are used to download via JTAG connector. The only host side application required to download new interpreter software should be a simple terminal program (example: Hyperterminal).

The second piece of software is an bridge that allows users to make remote connections to the uEAC through TCP/IP. This would allow a user to make a TCP/IP socket based connection to a host machine running the bridge software. The bridge software would pass the uEAC interpreter commands through the virtual serial port. This would allow remote users to access a particular uEAC by knowing the IP address of the host computer and the port that the uEAC bridge software is listening on.

This document will describe the key functionality that is expected from the three software components 1.) Interpreter 2.) Bootloader 3.) Bridge. The expectation for documentation and communication will also be outlined. This document should be considered a development guide, but not an absolute standard. If there are areas where the design can be improved if the requirements are changed, these areas should definitely be highlighted and considered. Communication throughout the development process will be critical to a successful outcome.

3.0) Interpreter

The interpreter is a piece of software running on the MSP430 which provides a way for the host computer to instantiate analog configurations along with reading results. This software communicates with the host computer through a virtual serial port. From the point of view of the MSP430, commands are received through a standard UART as ASCII strings. The design and implementation of this command language is the focus of this section.

The Interpreter needs to provide the primitive capabilities of setting matrix pins as current source/sinks, or current/voltage measurement points. The interpreter also needs to provide the capability to instantiate Lukaseiwcz Logic Arrays (LLA). In this context, when the term LLA is used, this refers to a transfer function that has one input and one output. The input will always be a pin that is defined as a current measurement point. The output can either be a source pin, a sink pin, or no pin. The case where an output pin is not used, the LLA output data should be transmitted to the host.

The LLA transfer function will be a lookup table that is indexed by the input data. The transfer function should be implemented as a 256 word (16 bit) table. Note that this table size is the exact size of one flash memory block in the MSP430. There will be 27 pre-defined functions that can be selected. These 27 functions have been defined by Jonathan Mills in his research related to using the Extended Analog Computer to implement solutions to partial differential equations. It is likely that researchers will desire functions beyond these 27. Flash space should be reserved to allow users to download their own transfer functions. 25 user-defined transfer functions spaces should be provided.

uEAC Firmware Technical Profile
Indiana University Computer Science Department

The following is a table that outlines the categories of interpreter commands that need to be implemented.

| <i>Command Type</i> | <i>Description</i> |
|---------------------------------|---|
| Set Pin Function | <u>Allows the host to define the pin type for a specified pin along with the magnitude if it defined as an output. A pin can be defined as a voltage sense current sense, current source, or current sink.</u> Voltage sense should be the default state of the pins. |
| Set LLA | Allows the host to create an LLA connection between two pins. This command will need to know the input pin, output pin (or “send to host”), LLA function number, and output period. The LLA function number will be 1-27 for the pre-defined functions or 28-52 for the user defined functions. The output period will be 1-256 where the number represents an integer multiple of the base system sample period. For sample periods greater than 1, the input should be averaged over the desired number of samples. |
| <u>Define Transfer Function</u> | Allows the host to download a function table to a specified transfer memory slot in flash. The function data should go into one of the 25 slots in the range of 28-52. |
| <u>Report Transfer Function</u> | This command should allow the host to request the data stored in a particular LLA (referenced by number) slot to be sent back to the host. |
| Report Pin Voltage | This command should provide the host with the voltage at a specified pin. |

4.0) Bootloader

The bootloader is a program written for the MSP430 that allows the host to download a new interpreter software through a simple terminal such as Hyperterminal. When the uEAC is powered on, the bootloader should issue a message to the host. If the host responds back with a specific, pre-defined keyboard accessible character, then the bootloader is entered. If this sequence is not entered in a period of 3 seconds, then the bootloader should try to execute the interpreter that is already resident. The bootloader should have the following base operations:

| <i>Command Type</i> | <i>Description</i> |
|----------------------------|--|
| Download | Allows the Host to download an MSP430 binary file to flash. The expectation is that this binary will be the interpreter application. |
| Verify | Allows the flash memory to verified against a downloaded MSP430 binary. |
| Version | Reports the version of the bootloader and the interpreter if one is loaded. |

5.0) Bridge

The TCP/IP bridge is an application that allows users to connect to the host computer and pass commands from a TCP/IP socket to the uEAC through the virtual serial port. Similarly, this application should pass information from the uEAC back to the remote client when it is available. The goal of the bridge is to provide remote access to uEAC resources from simple sockets based clients. It should be possible to rebuild the bridge to run on either Windows (Cygwin) or Linux. An example client should also be provided to demonstrate the capabilities of the bridge.

6.0) Documentation

Documentation of the software design will be a very critical element in making the software useful and maintainable. The project code repository should be documented using the Doxygen Tools. A software design description should also be provided that explains the detailed functionality of the code. This document will be used to allow others to come up to speed on the software to continue development in the future.

7.) Project Communication

I expect that there will be regular communication through phone, email and visits as development progresses. Beyond the adhoc communication, I would like the following other communication mechanisms.

| <i>Mechanism</i> | <i>Information Requested</i> |
|------------------------|--|
| Weekly Status Report | This can be a simple email outlining the progress from the week and the work plan for the following week. This should also include any major obstacles that are being confronted. |
| Concept Design Review | Block diagrams and a description of the planned approach. We should set a date for this review if there is not already a similar milestone in the curriculum. <u>Scheduled as part of the second oral review Senior Design milestone to be delivered 10/21/05.</u> |
| Detailed Design Review | This should be a detailed code review along with a preliminary handoff of the software documentation and code. |
| Final Handoff | Any issues from the Detailed Design Review should have been evaluated and corrected if possible. It would be best if this handoff occurs at least 2 weeks after the Detailed Design Review to give the development team a chance to respond to review issues. |

8.)Change History

R001 9/28/05 Document Creation, Bryce Himebaugh

R002 9/30/05 Modifications per discussion with Russ Eberhart, Bryce Himebaugh

- a.)Sec. 3, added language to clarify that “Set Pin Function” should also set magnitude
- b.)Sec 3, “Define LLA Function” renamed to “Define Transfer Function” as these downloadable functions are arbitrary.
- c.) Sec 7, Added a date for Concept Review to align with the 10/21 IUPUI oral review previously scheduled.
- d.) Sec 8, Added "Change History" section