TCP/IP CONTROL SYSTEM INTERFACE DEVELOPMENT USING MICROCHIP BRAND MICROCONTROLLERS*

Christopher E. Peters[#], Maria A. Power, ANL, Argonne, IL 60439, USA

Abstract

Even as the diversity and capabilities of Single-Board-Computers (SBCs) like the Raspberry Pi and BeagleBoard continue to increase, low level microprocessor solutions also offer the possibility of robust distributed control system interfaces. Since they can be smaller and cheaper than even the least expensive SBC, they are easily integrated directly onto printed circuit boards either via direct mount or pre-installed headers. The ever increasing flash-memory capacity and processing clock speeds has enabled these types of microprocessors to handle even relatively complex tasks such as management of a full TCP/IP software and hardware stack. The purpose of this work is to demonstrate several different implementation scenarios wherein a computer control system can communicate directly with an off-the-shelf Microchip brand microcontroller and its associated peripherals. The microprocessor can act as a Hardware-to-Ethernet communication bridge and provide services such as distributed reading and writing of analog and digital values, webpage serving, simple network monitoring and others to any custom electronics solution.

MOTIVATION AND THEORY

The Argonne Tandem Linac Accelerator System (ATLAS) is located at the United States Department of Energy's Argonne National Laboratory in the suburbs of Chicago, Illinois. It is a National User Facility capable of delivering ions from hydrogen to uranium for low energy nuclear research in order to perform physics analysis of the properties of the nucleus. In support of this goal, the accelerator control system and its electronics hardware have been in a state of continuous upgrade since its transition to computer control in the late 1980s [1]. In addition, the beam transport hardware and support electronics equipment has been in a constant state of change. The interfaces into the control system range from simple 0-10v analog input signals, to 50+ bit binary coded decimal, to all manner of digital communication protocols (RS-232/485, GPIB, USB, SNMP, ModBUS).

These signal types can require inefficient cable solutions, sometimes requiring hundreds of separate conductors to transmit relatively slow changing (sub ~10Hz) and relatively low accuracy requirement (2-4 significant figures) beam transport control channel points. Specifically, these types of channels stand in contrast to the high speed, high accuracy data acquisition and timing channels used by the target and detectors groups.

This work was supported by the U.S. Department of Energy, Office of Nuclear Physics, under Contract No. DE-AC02-06CH11357. This research used resources of ANL's ATLAS facility, which is a DOE Office of Science User Facility. *microchip.com #ChrisPeters@anl.gov

The result is a mismatch between the modest channel speed and accuracy requirements and the high cost of implementation. The goal of this work is not only to define a low cost control system hardware interface, but also to push that platform lower down into the custom hardware components currently developed at ATLAS (Figure 1). This will provide better focus for hardware designers to only work directly on electronics separate from the high level interface, and allows the controls group to rely more on generic and reusable interfaces.

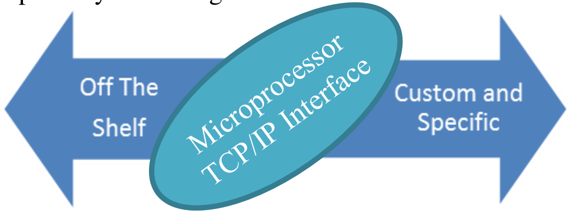


Figure 1: Microprocessor Based Interfaces Exist in the Middle of the Spectrum of Hardware Design Complexity.

MICROPROCESSOR COMPARISON TO SINGLE BOARD COMPUTERS

The goal of this work is to investigate the advantages and disadvantages of using the latest in microprocessor based platforms in a distributed control system environment. Single Board Computers (SBCs) have increasingly been viewed as a way to add various communication protocols [2] to hardware. These devices can interface with simple electronics via on-board ADCs and PWM/DACs, and also act as complex data processing platforms in their own right. This paper specifically focuses on the Microchip brand [3] and the OEM produced PIC32 Ethernet Starter Kit (PIC32 or ESK).

It is often difficult to directly compare the offerings of common SBCs like the BeagleBoard Black [4], the Raspberry Pi [5], and Industrial PCs like PC/104 [6] and microcontrollers, since the selection of each platform is often highly dependent on the application. Our goal is to determine if the more limited and focused capabilities of microprocessors are matched well to implementation into low-level custom built beam transport electronic devices.

PRODUCTS AND INITIAL EXPERIENCES

The initial steps in this research were to determine what possible COTS offerings were available to push TCP/IP communication down into the hardware design level. Initial experimentation was performed using a starter kit, however all components are available for installation on any custom developed PCB hardware.

ISBN 978-3-95450-146-5

Current Baseline Product

Microchip's Ethernet Starter Kit was chosen as the test bed for this work due to the company's reputation for field support and previous experience with the authors. This kit comes in the form of a small form factor PCB board with an Ethernet port and several debugging tools (Figure 2). One such tool is an on-board USB interface for PC-USB debugging. There is also a board-to-board high density plug on the bottom, for the addition of optional expansion and I/O breakout boards.

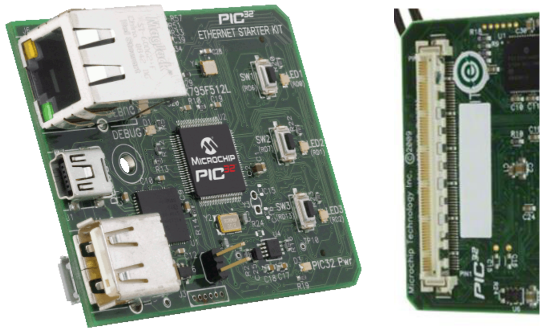


Figure 2: Microchip Ethernet Starter Kit [2] & Connector.

Preloaded Software and Services

One of the major advantages to working with products from large OEMs is in the form of bundled software libraries. In the case of Microchip, these libraries are offered open source, and with a free license. There are also prewritten API routines, similar to those which come with SBC OS distributions, which can operate UARTs, SPI, I2C, and USB busses with the minimum of code implementation from the end user. This code base is bundled along with a compiler and debugging tools into Microchip's MPLAB IDE at no additional cost [7].

IMPLEMENTATION AND SENARIOS

Web Browser/HTTP and AJAX Connections

The starter kit comes preloaded with an HTTP server. Using this server, it is possible to create HTML elements on the client side which connect asynchronously to the PIC32 via Asynchronous Java and XML (AJAX) calls. This way, a very simple but user-friendly interface can be created (Figure 3) for debugging or testing offline. User manuals or schematics can also be stored in non-volatile memory on the chip for documentation purposes.

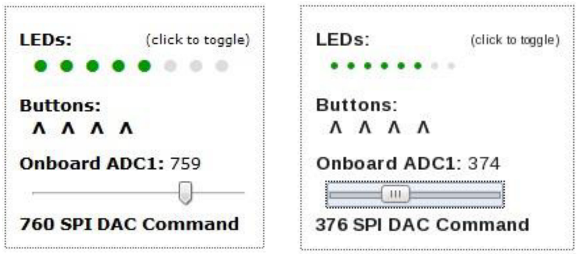
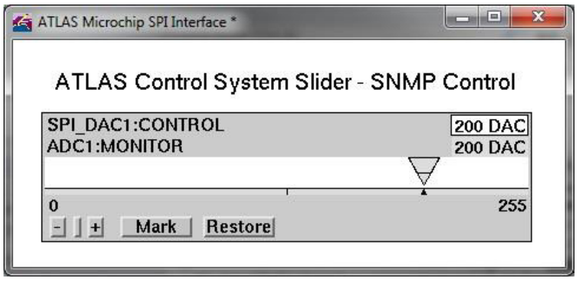


Figure 3: HTML Range Slider communicating via AJAX in (Left) Windows and (Right) Linux Firefox Browsers. The slider sets an external DAC, and reads back the ADC.

Low Level Protocols like SNMP

The starter kit also comes with standard SNMP libraries. SNMP has been demonstrated in the past to be useful in a control system environment [8], and can be additionally leveraged as a common control system protocol. At ATLAS, low level SNMP handlers were implemented into local control system libraries and were then able to communicate reliably with the PIC32. Standard operator display pages were created (Figure 4) with SNMP executing asynchronously in the background and a seamless interface to the operators was presented.



ATLAS Microchip SPI Interface *
ATLAS Control System Slider - SNMP Control
SPI_DAC1:CONTROL 200 DAC
ADC1:MONITOR 200 DAC
0
255
Mark
Restore

Figure 4: ATLAS Standard Slider interfaced to Microchip snmpset:MICROCHIP-MIB::SpiDAC1= INTEGER: 200 snmpget:MICROCHIP-MIB::analogPot=INTEGER: 200.

Higher Level Protocols like Channel Access

In order to demonstrate capabilities similar to those of full SBCs, it was desired to implement some level of EPICS integration into the PIC32. However, EPICS libraries are mostly C++ and contain a significant codebase dedicated to multitasking and thread-safety. Therefore, it would be difficult to directly port standard EPICS libraries to the Microchip compiler. Instead, the EPICS Channel Access Protocol Reference [9] was used to implement a very small subset of low level TCP/UDP commands which the PIC32 can respond to. This has enabled at least a first pass at integration with a higher level control system.

Cost Comparisons

The below (Table 1) provides a cost comparison for assemblies like SBCs and PIC32 development kits compared to individual component parts of a custom microprocessor hardware design. Assuming that the PIC32, physical Ethernet (PHY) chip, and Ethernet socket are in addition to an existing circuit board design, the total additional cost is under 25 USD. This does not include development costs and other electrical components.

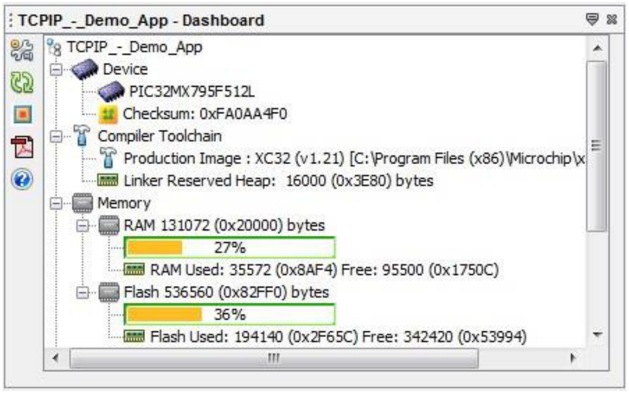
Table 1: Development Kits and Component Costs

Component	Cost
PIC32 Ethernet Starter Kit (ESK)	70 USD
ESK I/O Expansion Board (Optional)	70 USD
PIC32MX795F512 Processor Only	**11 USD**
TI DP83848C PHY Chip	**6 USD**
MagJack Ethernet Plug SI-60000	**5 USD**
Beagleboard Black	55 USD
Raspberry PI Ver-B	35 USD

ISBN 978-3-95450-146-5

PERFORMANCE

Any control logic duties of the microprocessor such as input filtering, I/O transactions, alarming, etc., can only exist in addition to the existing low level interface code. There are several restrictions which must be considered before using a microprocessor for control interfaces. The first restriction is on code size (static, EEPROM) and dynamic (stack, heap) memory sizes. (Figure 5) shows the amount of remaining memory with all HTTP and SNMP services running, in addition to a large amount of test services. This leaves 342kB available for control system logic, and 95kB free for live RAM scratch space. While this would be considered impossibly restrictive in the SBC realm, microprocessor operations are normally focused and hierarchal in nature. A simple Ethernet-to-SPI bridge (TCP/IP to SPI DAC output directly inside the PCB hardware) would only require a few kB of code.



```
TCPIP_-_Demo_App - Dashboard
TCPIP_-_Demo_App
  Device
    PIC32MX795F512L
    Checksum: 0xFA0AA4F0
  Compiler Toolchain
    Production Image : XC32 (v1.21) [C:\Program Files (x86)\Microchip\x
    Linker Reserved Heap:  16000 (0x3E80) bytes
  Memory
    RAM 131072 (0x20000) bytes
      27%
      RAM Used: 35572 (0x8AF4) Free: 95500 (0x1750C)
    Flash 536560 (0x82FF0) bytes
      36%
      Flash Used: 194140 (0x2F65C) Free: 342420 (0x53994)
```

Figure 5: Test Memory Usage with HTTP, SNMP, and EPICS CA Services fully complied, using MPLAB.

The PIC32 tested during this work contains an on-board Ethernet controller which communicates via direct-memory-access (DMA) to an external PHY chip. In this way, moving data to the Ethernet subsystem does not use CPU instructions, making transfers extremely fast. The demo board in this work was able to send TCP data at a rate of over 1 MByte per second to a remote host. This speed was reduced only slightly by actively sending simultaneous HTTP and SNMP requests and responses.

CONCLUSION AND FUTURE WORK

There are several additional steps to be undertaken in order to fully qualify the concept of a generic microcontroller-based control system interface. First and foremost, within this work only a demonstration kit has been tested. The next iteration should be installed directly inside on an existing PCB along with the required oscillators, filter capacitors and power supplies. Another option is to develop an ATLAS specific interface PCB which uses the same PIC32 processor and components, but has custom headers which are already used by the electronics development group. In this way, a control system microprocessor kit specific to ATLAS could be sourced and then re-used on any available hardware project by the Electronics groups.

Table 2: Microprocessor Advantages and Disadvantages

Feature	Advantages	Disadvantages
Low-level on board HW communication	Configurable pin-outs speak a wide variety of HW level protocols.	Challenge to create a single standard interface design.
Small, Low Cost and native real time.	Hardware Engineers can integrate directly into end products.	Initial work for custom PCBs to distribute signals
Few Running Application sections and services	Low security risk & risk of bad behaviour as a result of complex operating system.	Requires some knowledge of Firmware development.
Increases in memory and processor speed	Can support TCP/IP stack, and has OEM software libraries	Less memory and processing power compared to SBCs.

A successful end result would push control system development towards an "Internet of Things" concept [10], whereby every device in the system has an integrated Ethernet plug and its own IP address. This greatly reduces cable overhead cost and increases the capabilities by which all devices can communicate (Table 2). Common security concerns like unintended access and outdated operating system patches can be mitigated by the greatly reduced code base and the targeted Ethernet capabilities of a microprocessor. While these features are not always distinct from a single board computer, the microprocessor as a concept is much more able to be integrated directly into the actual circuit boards of control hardware devices and has the dual advantage of lower component cost and zero unneeded operating system services providing additional networking vulnerabilities.

REFERENCES

[1] F. Munson, D. Quock, B. Chapin, and J. Figueroa, "Argonne's ATLAS Control System Upgrade", ICALEPCS '99, Trieste, Italy, October 4-8, 1999.

[2] S. Cleva, A. Bogani, L. Pivetta, "A Low-Cost High-Performance Embedded Platform for Accelerator Controls", Proceedings of PCaPAC2012, Kolkata, India, 2012.

[3] http://www.microchip.com

[4] http://beagleboard.org/Products/BeagleBone+Black

[5] http://www.raspberrypi.org/

[6] http://www.pc104.org/specifications.php

[7] http://www.microchip.com/pagehandler/en-us/devtools/mla/home.html

[8] C. Peters, M. Power, "Distributed Network Monitoring Made Easy - An Application for Accelerator Control System Process Monitoring", ICALEPCS2013, San Francisco, CA, 2013.

[9] http://www.aps.anl.gov/epics/docs/CAproto.html

[10] Evans, Dave, "The Internet of Things", Internal Cisco Whitepaper, April 2011, http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

ISBN 978-3-95450-146-5