AC 2007-1152: A CONTROLLER FOR ROBOTICS AND MICROCONTROLLER APPLICATIONS INSTRUCTION

David Livingston, Virginia Military Institute

David L. Livingston is a Professor of Electrical and Computer Engineering at the Virginia Military Institute. He received the B.S.E., M.E. and Ph.D. degrees in Electrical Engineering from Old Dominion University. Dr. Livingston worked as a Staff Engineer in Intelligent Workstations at IBM Endicott Labs, was an Assistant Professor of ECE at Old Dominion University, and a Professor and Program Head of EET at Virginia Western Community College. He is a senior member of IEEE and a member of ASEE, AAAI, ACM and VAS. Dr. Livingston also belongs to Eta Kappa Nu, Tau Beta Pi and Phi Kappa Phi and is a licensed professional engineer in the Commonwealth of Virginia.

A Controller for Robotics and Microcontroller Applications Instruction

Abstract

A controller board inspired by the Handy Board, but based on a pair of Atmel ATmega128's, is discussed. Elements of the hardware design and input/output interfaces are detailed, including parallel and serial I/O, analog I/O, an LCD interface, and dc motor control interfaces. Firmware for hardware drivers written in AVR assembly language and a handshaking protocol for communications between the microcontrollers are also discussed.

The new controller board, designated Koios I, will first be used in our freshman introductory course – in lieu of the Handy Board – which involves learning about electrical and computer engineering through the design and control of robots. Koios I will also be used in our microcontrollers application course, in our senior capstone design course, and in independent research courses requiring a dedicated computer/controller.

Like the Handy Board, Koios I is being developed in an open-source manner. Schematics and firmware code listings are readily available on the web. The firmware for Koios I is written in AVR assembly language; however, user applications can be written in either assembly language or the C programming language via an AVR port of the open source C complier gcc.

Printed-circuit board layouts using surface mount devices are currently in progress; the results of which will be made available in the same manner as the schematics and code. Koios I is intended to be continually evolving and a USB interface for program download and an integrated development environment are planned for the near future.

Introduction

Microcontrollers are found in the implementations of solutions to problems in many sub-disciplines of electrical and computer engineering as well as other fields of engineering and science. In deed, the applications of microcontrollers are taught in a variety of courses: electrical and computer engineering—microcontrollers and microprocessors, mechanical engineering—mechatronics, physics—instrumentation, and chemistry—process control. Within an ECE curriculum, microcontrollers can be applied in digital and computer courses, introductory courses, signal processing and controls courses, robotics courses and capstone design courses. Parten[1] emphasizes the importance of microprocessor education to ECE students that first take a formal course on microprocessors followed by a design projects course where the focus is the application of microprocessors in embedded systems. Microcontrollers also find their way into various robotics courses[2] and design contests such as the Trinity Firefighting Home Robot Contests[3] and the IEEE Region 3 Student Hardware Contest[4].

To serve this variety of applications, a microcontroller is best incorporated into a board that is flexible, provides a variety of interfaces, and is easy to employ. Fred Martin realized the value of such a board when he developed the Handy Board[5, 6] as an outgrowth of the controller used

for MIT's 6.270 course. He also wrote a textbook[7] using the Handy Board and LEGOs Mindstorm parts to teach introductory engineering concepts using robotics.

The remainder of the paper is organized as follows. The motivation behind the work is presented in the next section followed by the hardware development and I/O specifications. Firmware organization is presented followed by a discussion of the controller's benefits to instruction, future work and conclusions.

Motivation

We have used Martin's text, Handy Board, and LEGOs Mindstorm kits to enable the hands-on experience they provide to students in our freshman "Introduction to Electrical and Computer Engineering" course. We discovered, as many others have, that the Handy Board is useful in other courses where an embedded processor is the key element in a design problem such as our capstone design course and in independent study courses.

The Handy Board is based on the Motorola (Freescale) M68HC11, a microcontroller developed in the 70's for use in automotive applications (Martin is currently developing an updated Handy Board using different, up-to-date technology[8]). We decided that an update of the concept of the Handy Board to recent technology that is consistent with the microcontrollers we use in our program was warranted.

Three years ago, we examined different microcontroller families for our "Microcontrollers" course in an effort to make use of recent technology. We chose the Atmel AVR family[9] for the following reasons:

- The family is a true eight-bit RISC architecture with a two-stage instruction fetch/execute pipeline using a system clock of up to 16-20 MHz.
- The straightforward architecture is similar to one that is studied in a prior course in our program on digital systems and computer design.
- The family is device-consistent, inexpensive, and is available in a wide variety of memory sizes, input/output capabilities, and packages.
- An integrated assembler and simulator are available at no cost from Atmel. A port of the open-source gcc compiler is available in a development environment called "WinAVR"[10].
- Other types of support are available at Atmel's website as well as at a user site called AVR Freaks[11].

Furman and Moen[12], in a comparison of microcontrollers for mechatronics, rated the AVR ATmega128 – a high-end member of the AVR family – very high in its applicability to mechatronics education. Experience with the AVR family in our microcontrollers course has been positive. Our students have been quick to learn the family and have successfully used the AVR in design projects.

With our adoption of the AVR family for microcontrollers and a desire to update the technology for the controller used in our introduction course, we set out to develop a new controller board

that uses AVR microcontrollers for consistency while maintaining the interfacing and programming flexibility of the Handy Board. The board has been designated the Koios I after the Greek titan of intelligence and uses the AVR ATmega128 as the base microcontroller.

Hardware

For reasons of flexibility and functionality, the Atmel AVR ATmega128 microcontroller was chosen as the core of Koios I. The ATmega128 provides 128 KB of FLASH memory for program storage and 4 KB of EEPROM for infrequently changed data. It allows for the interfacing of up to 60 KB of directly addressable external SRAM that, with an internal 4 KB, provides for 64 KB of SRAM. By using direct control signal generation and banking, the Koios I is able to access 128 KB of provided SRAM for variable storage.

The intended purpose of Koios I demands that it possess a significant and varied number of input/output (I/O) interfaces. In the original design, special purpose hardware was to be used to create the interfaces. However, it was realized that chip count could be significantly reduced by using a second ATmega128 as an I/O processor. The main microcontroller which is interfaced to the external SRAM is designated as the master processor; the second microcontroller is designated as the I/O processor. Once the decision was made to use two ATmega128's, the I/O interfaces had to be partitioned between the microcontrollers and a communication protocol between the microcontrollers had to be established. Communication is performed via the serial peripheral interface (SPI) which is part of the ATmega128 architecture. Since I/O devices are inherently asynchronous, handshaking is used for time-efficiency and data loss prevention. A block diagram of the Koios I organization is shown in Figure 1.

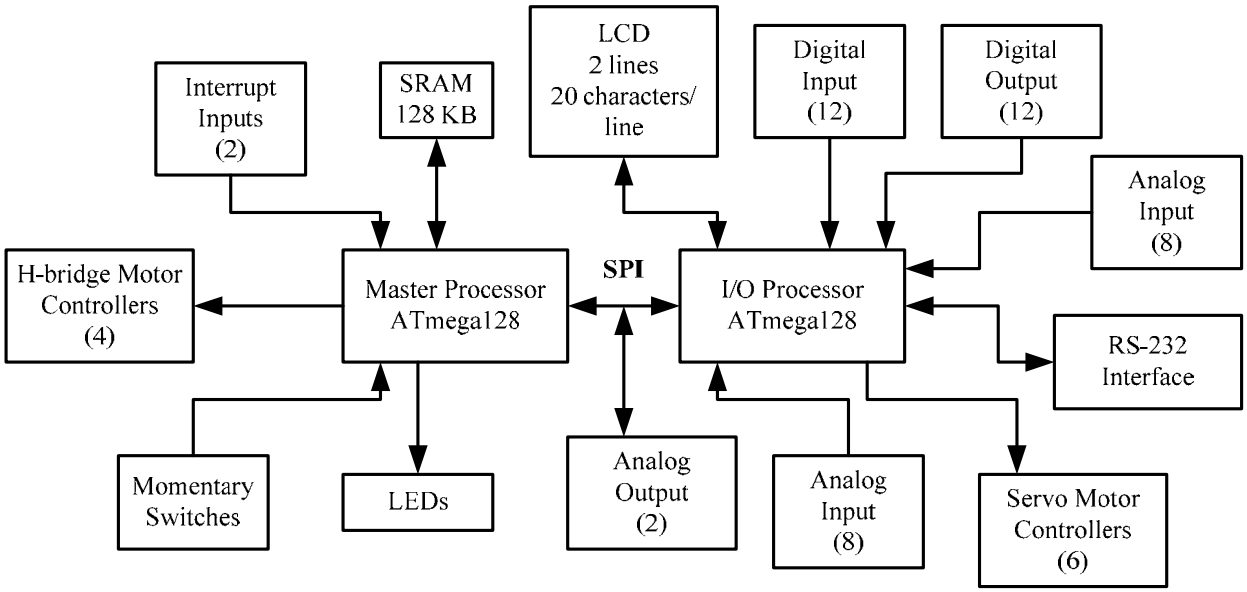

Figure 1. Koios I organization.

Digital I/O is performed in the I/O processor. There are 12 individually addressable digital input signal ports and 12 individually addressable digital output signals. Eight of each 12 can also be addressed as a byte. This layout is particularly useful for implementing and demonstrating full handshaking. In an effort to "student-proof" the system, 1.0 kΩ resistors are placed in series with the digital inputs and socketed discrete buffers are used to isolate output signals. All digital inputs are pulled up using the internal pull-up resistors in the ATmega128.

There are 16 analog-to-digital input ports split evenly between the master processor and the I/O processor. Each port is set up as a 10-bit, single-ended channel capable of converting an input range of 0.00 V to 5.00 V. As with the digital input ports, 1.0 kΩ resistors are placed in series to limit currents due to clipped over-voltages. The ATmega128 is capable of converting at a maximum rate of about 15 kS/s.

Although digital-to-analog conversion can be done at low bandwidths using pulse-width modulation, it was decided to include two 10-bit digital-to-analog converters. The converters are interfaced using the serial peripheral interface. One of the converters can be jumpered to a high-input impedance speaker, providing a mechanism for creating complex sounds.

For serial communications purposes, an RS-232 interface is provided. The ATmega128 provides two asynchronous serial ports one of which on the I/O processor is interfaced to an RS-232 level converter chip which is further connected to a 9-pin D-shell connector. The capabilities of the processor allow for standard serial configurations including bauds of up to 1Mb/s.

Since the controller board is intended to be used for robotics as well as other systems involving the control of dc motors, four pulse-width modulated (PWM) ports are connected to power H-bridges. The master processor supplies the PWM signals along with four direction signals.

Six PWM ports are supplied on the I/O processor for driving servomotors. The ports produce 50 Hz pulses of 1 to 2 ms widths with 9-bit precision. Power for the servomotors is derived from the battery pack powering the entire system.

Direct user input/output is provided by a two-line, 20 characters per line, liquid crystal display (LCD) module, LEDs, and momentary switches. The LCD module is interfaced to the I/O processor while the switches and LEDs are interfaced to the master processor.

To provide for interrupt driven behavior, two interrupt lines are supplied on the master processor. Both lines employ internal pull-up resistors and are configured as active low assertion levels.

Since one of the main functions is to support robotics projects, Koios I needs to be mobile. Based on this requirement, rechargeable NiMH batteries are used to provide power for the controller board and motors that are driven by it. The motor supplies and controller board supplies use the same power base, but are differentially isolated using inductor-capacitor filters followed by active regulators.

Firmware

Koios I is intended to be user programmed using AVR assembly language and the C programming language. Eventually a custom C compiler, in the same vein as Interactive C[6], will be developed to support a user-friendly integrated development environment (IDE) to make programming the device accessible to a wide variety of users. However, regardless of the compiler used, driver code is needed to interface user code to the hardware and to provide a communications mechanism between the two processors.

Since the input/output functionality is partitioned between the processors, a mechanism had to be developed to facilitate communications between the processors, keeping in mind that input/output interfaces possess a variety of timing properties. To manage the communication effectively from the hardware and timing standpoints, it was decided to use the serial peripheral interface (SPI) augmented with bidirectional handshaking. The SPI requires three lines: Master Out/Slave In, Master In/Slave Out, and Serial Clock. To implement the handshaking, each processor provides a digital input line and a digital output line for signaling requests and acknowledgements.

All I/O operations executed by the I/O processor are encoded as byte commands which are sent to the I/O processor using the SPI. Commands are followed by data if warranted. Table 1 contains the commands for the I/O operations implemented on the I/O processor. Since the SPI simultaneously reads and writes to and from the I/O processor, to perform a read operation, a null command (0x00) is sent by the master processor. Remaining I/O operations are implemented as subroutines directly on the master processor.

Since operations vary in the time taken to execute, e.g., LCD operations are much longer than digital output operations, handshaking becomes necessary. The handshaking mechanism used to ensure proper command and data transfer is shown in Figure 2a and 2b. The handshaking routines precede all command and data transfers between the two processors.

Benefits to Instruction, Future Work, and Conclusions

Subsystems have been prototyped and tested. The hardware is currently being laid-out to be implemented using surface-mount logic on printed-circuit boards. As with the schematics and code, the printed-circuit board files will be made available on the project website[12] where their use and modification by all who are interested will be emphasized. By using two ATmega128's, the number of devices has been kept small which in turn will keep costs reasonable.

As Koios I is meant to be a viable replacement of the Handy Board in our curriculum, it will eventually be used in the introduction to ECE course. This will allow for the easy design and implementation of robots and simple embedded systems. To accommodate the probable lack of programming experience by freshmen, somewhat canned routines will have to be written until a user-friendly C programming environment is developed.

Table 1. I/O processor commands.

Command	Code	Comment
LCD Operations	0x1*	
clear LCD	0x10	
write null-terminated string	0x11	followed by string and null (n bytes)
write character at location	0x12	followed by offset and line number
A-to-D	0x2*	
get analog channels 8 to F	0x20 to 0x27	LSB is returned followed by MSB
bit in	0x3*	
get bits 0 to B	0x30 to 0x3B	0x00 or 0x01 is returned
byte in	0x40	byte is returned
servos	0x5*	
control servos 0 to 5	0x50 to 0x55	followed by servo position (2 bytes), LSB first
RS-232	0x6*	
set baud	0x60	followed by baud (2 bytes), LSB first
set size	0x61	followed by size (1 byte)
set parity	0x62	followed by parity (1 byte)
set stop bits	0x63	followed by number of stop bits (1 byte)
write character	0x64	followed by ASCII character
read character	0x65	ASCII character is returned
reset bit	0x7*	
bits 0 to B	0x70 to 0x7B	
set bit	0x8*	
bits 0 to B	0x80 to 0x8B	
byte out	0x90	followed by byte

Koios I will find immediate application in our capstone design course as well as in independent undergraduate research efforts where students are likely to have experience with programming in high-level languages and microcontrollers. The capstone design course typically involves the design and implementation of a robot that competes in a regional contest. Independent undergraduate research projects often are designed around embedded systems and the ease of interfacing and programming that is offered by Koios I will allow students to concentrate on the high-level details of the project.

Our microcontrollers course is project based. Koios I will help simplify hardware implementation details in complex projects allowing for more emphasis on programming issues. The details of the hardware design and firmware routines for Koios I have already been used as example material in the course.

The next hardware subsystem under investigation is a USB interface. This addition will facilitate both programming the board and demonstrating USB operation. Since Koios I is open source, it is expected that it will continue to evolve and remain viable for many years.

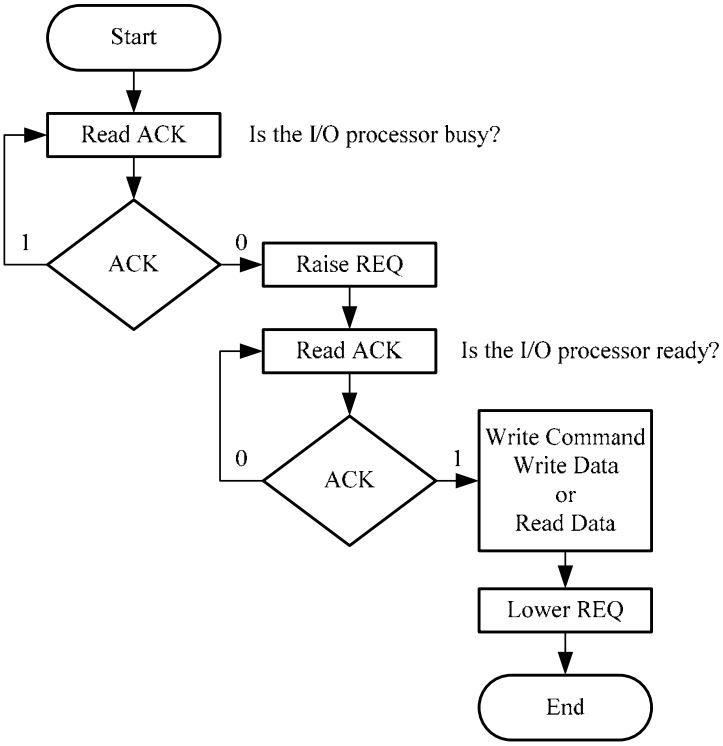

Figure 2a. Handshake, master processor side.

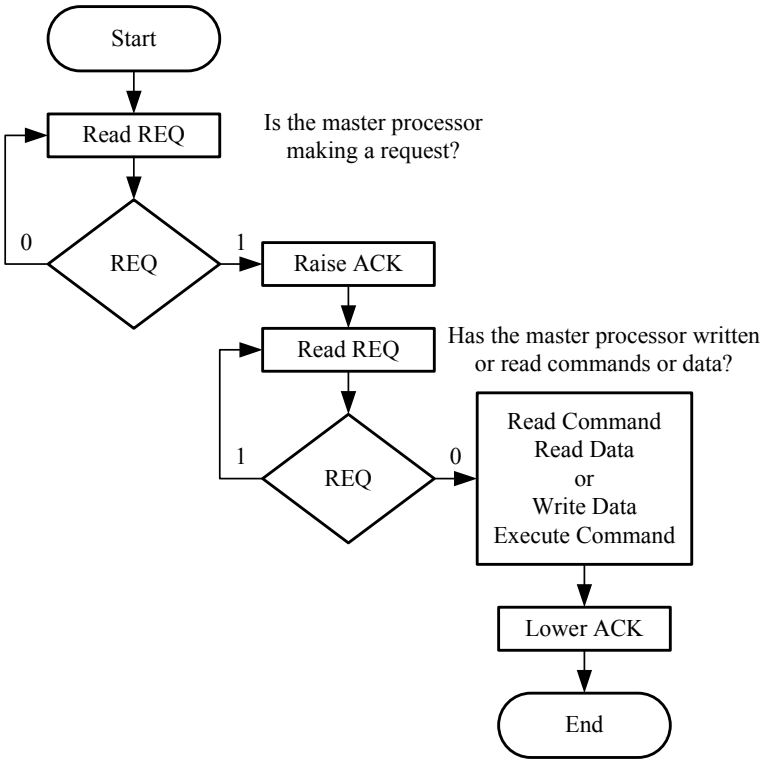

Figure 2b. Handshake, I/O processor side.

Acknowledgements

The author would like to acknowledge support from a Virginia Military Institute Technology, Teaching, and Learning mini-grant.

Bibliography

1. M. Parten, "Embedded Microprocessors in a Project Laboratory," *Computers in Education Journal*, Vol. XVI, No. 1, January – March 2006, pp. 95-101.
2. D. J. Ahlgren, I. M. Verner, D. Pack, and S. Richards, "Strategies and Outcomes in Robotics Education," *Computers in Education Journal*, Vol. XVI, No.1, January – March 2006, pp. 51-65.
3. "Welcome to the 14th International Robot Firefighting Contest," http://www.trincoll.edu/events/robot/.
4. "IEEE SoutheastCon 2007 Student Program," http://www.southeastcon.org/2007/students/#StuHardComp.
5. F. G. Martin, "Fred's World," http://www.cs.uml.edu/~fredm/.
6. F. G. Martin, "The HandyBoard," http://www.handyboard.com/.
7. F. G. Martin, *Robotic Explorations, A Hands-On Introduction to Engineering*. Upper Saddle River, NJ: Prenticel Hall, 2001.
8. F. G. Martin, "The Blackfin HandyBoard," http://www.cs.uml.edu/blackfin/
9. "AVR 8-bit RISC from Atmel," http://www.atmel.com/products/AVR
10. "SourceForge.net: WinAVR," http://sourceforge.net/projects/winavr/http://avrfreaks.net/
11. B. Furman and E. Moen, "Evaluation of Alternative Microcontrollers for Mechatronics Education," *Computers in Education Journal*, Vol. XV, No.3, July – September 2005, pp. 22-33.
12. http://academics.vmi.edu/ee_dl/KoiosI.htm