LIFE SCIENCES FLIGHT EXPERIMENTS MICROCOMPUTER

Final Report

NASA/ASEE Summer Faculty Fellowship Program--1987

Johnson Space Center

Prepared by:	Peter N. Bartram, Ph.D.
Academic Rank:	Associate Professor
University and Department:	Norwich University Computer Science Engineering Division of Engineering Northfield, Vermont 05663
NASA/JSC	
Directorate:	Space and Life Sciences
Division:	Life Sciences Project
Branch:	Project Engineering
JSC Colleague:	James S. Evans
Date:	August 6, 1987
Contract Number:	NGT 44-001-800

ABSTRACT

A promising microcomputer configuration for the Spacelab Life Sciences Laboratory Equipment inventory consists of multiple processors. One processor is reserved for the principal investigator's use, with additional processors dedicated to real-time input and output operations. A simple form of such a configuration, with a processor board for analog-to-digital conversion and another processor board for digital-to-analog conversion, was studied. The system used digital parallel data lines between the boards, operating independently of the system bus. Good performance of individual components was demonstrated: the analog-to-digital converter operated at over 10000 samples per second. The combination of the data transfer between boards with the input or output functions on each board slowed performance, with a maximum throughput of 2800 to 2900 analog samples per second. Any of several techniques, such as use of the system bus for data transfer or the addition of direct memory access hardware to the processor boards, should give significantly improved performance.

INTRODUCTION

The Life Sciences Laboratory Equipment (LSLE) inventory for the Spacelab includes a microcomputer for high-speed acquisition of Life Sciences experimental data (both analog and digital). This computer provides the vital link between Spacelab experiments aboard Shuttle missions and the Science Monitoring Area in the Life Sciences Project Divison building at the Johnson Space Center. The LSLE microcomputer sends the real-time data, in digital form, properly formatted, to the High Rate Multiplexor (HRM), which in turn transmits the data to Earth via communication (TDRSS) satellite. The current LSLE microcomputer is based upon compatibility with the PDP-8, and thus no longer meets the goal of being compatible with systems familiar to principal investigators in the life sciences.

Requirements for a replacement LSLE microcomputer include providing all of the data acquisition, HRM downlink, and experiment control functions of the current machine. Additional capabilities needed include better and more extensive displays in the Spacelab, faster data sampling rates, extensive data storage for use with mid-deck experiments (no HRM downlink available for missions not carrying the Spacelab), greater speed, more memory, compatibility with modern computer systems (such as the IBM PC) familiar to principal investigators, and the ability to be programmed using higher level languages. The design should be based upon commercially available boards and a recognized backplane bus to connect them. Custom hardware should be avoided (HRM interface excepted).

Of the possible system architectures, a multiple processor (CPU) approach seems to offer the greatest promise. By use of separate CPU's for data acquisition and control (for example, analog-to-digital conversion or output to the HRM), a "master" CPU board could be reserved almost entirely for the principal investigator's use. Expansion would have minimal impact on the original parts of the system, as additional processors would handle the additional data acquisition load. This modularization would facilitate software and hardware maintenance. It would allow on-going replacement of obsolete components at the board level, without significantly disturbing other parts of the system.

EQUIPMENT USED

The Intel 8088 processor and boards using this processor were chosen for compatibility with familiar microcomputers (such as the IBM PC) and because of the availability of a wide range of software, including programming languages. The STD bus was selected because of the significant number of processor and other STD bus boards available from several vendors. Boards meeting the STD bus standards are also of relatively small size. To demonstrate the concept of the multiple processor configuration, a simple case was studied. This system consists of a master processor board (with memory), reserved for the principal investigator's use, and additional processor boards (with separate memory) for use with peripheral devices. Each peripheral board also contains an interface for real-time modules attached to the board by means of the Intel iSBX connector. Data transfer from memory on the peripheral boards to other boards can be accomplished using the STD bus (but only by the master processor board, as the peripheral boards cannot control the STD bus), or independently of the master board using a separate external direct parallel interface. One board is equipped with a multi-channel analog-to-digital converter, the other with a digital-to-analog converter. The digital-to-analog converter served as a representative output device, chosen here as a simpler and more readily available output device compared the HRM, while involving the same process of accepting data from another board in the system. In the final system, many additional boards will be needed, including digital-to-analog converters and a custom HRM interface board, as well as digital input and output interfaces. Each would be contained on its own dedicated processor board on the STD bus.

For the small scale demonstration system studied, Table 1 lists the equipment used. The master CPU, not listed in the table, was not actually used, as it was not needed for the peripheral board to peripheral board data transfer (STD bus lines were not used for this). Rather, each peripheral processor board comes with the Intel 8256 parallel input/output chip, which provides a means of independent data transfer, complete with interrupts and "handshake" control signals.

TABLE 1

EQUIPMENT USED

HARDWARE

Ziatech 8862 Card Cage and Power Supply (STD bus)
Ziatech 8830 Intelligent I/O Control Processor Board for STD bus (2 used)
Data Translation DTX311 Analog-to-Digital Converter module
Data Translation DTX328 Digital-to-Analog Converter module
personal computers (2 used) for downloading programs to the Ziatech 8830 boards and debugging them on-board.
oscilloscopes for displaying analog output and parallel transfer handshake signals.
function generator for analog input signals.

SOFTWARE

Ziatech 8830 software (and ROM's) to provide program loading and debugging on-board, using the personal computer and the 8830 serial port.
Microsoft C Language, version 4.00
Microsoft Macro Assembler, version 4.00
Microsoft Linker (loader), version 3.51
Mircosoft MS-DOS operating system, version 3.10
(The Microsoft software was run on the personal computers to develop code for the Intel 8088 processor on each Ziatech 8830 board.)

COMPANY ADDRESSES

Data Translation
100 Locke Drive
Marboro, MA 01752
(617) 481-3700

Microsoft Corporation
10700 Northrup Way
Bellevue, WA 98004
(206) 882-8089

Ziatech Corporation
3433 Roberto Court
San Luis Obispo, CA 93401
(805) 541-0488

SOFTWARE DEVELOPED

Prior to developing a complete system of routines for analog-to-digital conversion and data transfer out from one board, and data transfer in and digital-to-analog conversion on the other, individual functions were checked out. In all cases, software consisted of a main program, in C, calling functions (subprograms) written in 8088 assembler. All software developed is available from either Don Stilwell, mail stop SE3, NASA Johnson Space Center, Houston, Texas 77058, or from Peter Bartram, Division of Engineering, Norwich University, Northfield, Vermont 05663, by sending an MS-DOS formatted diskette.

In testing the Intel 8256 parallel interface chip (part of each board), the external interrupt was successfully used. Routines for the use of the 8256 for parallel transfer with handshake were coded first without use of interrupts by polling the handshake signals. However, the handshake pulse on the receiver side is short, independent of program instructions, and can be missed between execution of polling instructions. Therefore the interrupt signal was polled instead, even though interrupts were not enabled. Routines were also coded to make full use of the 8256 chip interrupts for handshake parallel transfer, and these were used in later work.

The Data Translation DTX328 digital-to-analog (d/a) converter used does not implement any of the interrupt capability available to modules attached to the selected peripheral processor board. Thus, for the d/a, only polling techniques were used. The DTX311 analog-to-digital (a/d) converter does allow for interrupt use to signal end-of-conversion. Separate a/d routines were prepared, one using polling only, the other making full use of the interrupt. When combined with parallel transfer, the a/d interrupt is given higher priority by the interrupt controller in the 8256 chip.

For testing system throughput, three software combinations were prepared. All three used the parallel transfer with handshake implemented with interrupts. All three used two data buffers - while one was being sent (received), the other was being prepared (used), switching back and forth between them.

The first package made no use of the analog-to-digital converter. Instead, the sending board created and sent data which represented 1024 values per cycle of a ramp

function. The receiving side used both the interrupt-based parallel transfer input and digital-to-analog conversion to output the ramp. The parallel transfer portion allowed for "extra" data to be transmitted and received - the first two bytes in each buffer were used for analog output, but additional bytes could be sent and received, though not used. This provided a means to determine how the d/a output rate would be affected by the board to board transfer.

The second software package added analog-to-digital converter support, with polling (matching the d/a on the other board). Interrupts were used for parallel transfer only. To examine the effect of conversion speeds and parallel transfer rates on system throughput, the number of a/d converted values contained in each buffer could be varied, and the number of bytes in the buffer actually transmitted could also be changed. As in the first package, only the first value in the buffer received was converted by the d/a on the other board. For example, ten samples a/d could be taken, two of them (four bytes) sent. After receiving all four bytes, only two bytes (one value) would be used for d/a output. By varying these two parameters, an examination of how the different functions affected total throughput was possible.

The third package differed from the second only in that the analog-to-digital conversion software made full use of the priority interrupt system. While this introduced higher overhead in servicing the a/d, it prevented service of the parallel transfer from locking out a/d service (except for short periods at the beginning of servicing the parallel transfer interrupt).

RESULTS

In evaluating throughput, the d/a output was monitored using an oscilloscope. For runs using both a/d and d/a converters, the analog input (from a function generator) and the parallel transfer handshake signals were also oscilloscope monitored. Conversion rates were computed from waveform periods (handshake signals) and time for "steps" in the d/a "stairstep" output. Figures presented here are accurate only in the first, occasionally second, significant digit, as the reading of the oscilloscope traces was often difficult.

For the digital-to-analog converter combined with parallel input, the maximum rate was 3600 samples output per second, which compares favorably with vendor claims. However, the conversion rate declined significantly with the receiving of additional data via parallel transfer. When four bytes were received (instead of two) and a very brief check for missing data was made using the other two bytes, output dropped to 2500 samples output per second. The rate dropped more with additional data transferred.

For the complete system, with each a/d value transmitted (two-byte buffers) and re-converted with the d/a on the other board, the throughput was 2500 samples per second without use of a/d interrupts, and between 2800 and 2900 samples per second with full use of available interrupts. However, it was shown that the analog-to-digital converter was capable of much faster rates when little parallel output was required. Rates up to about 7500 samples per second with no a/d interrupts, greater than 10000 samples per second with full interrupt use, were obtained. These rates were attained by transmitting only one sample in 100, and calculating the a/d rate from the d/a output. In this case, negligible processor time was spent in parallel transmission.

When more a/d values were transmitted to the d/a processor than were used by that board (though received), a similar decline in throughput was noted. For example, sending three a/d conversions for every one actually d/a converted resulted in an a/d conversion rate of 3700 samples per second without a/d interrupts, 4100 samples per second with a/d interrupts.

CONCLUSIONS AND RECOMMENDATIONS

As component speeds were significantly higher in isolation than when combined into a single system, it is concluded that the processor instruction time necessary to service both the real-time modules (a/d, d/a) and the parallel transfer is limiting. This is also supported by estimating the number of clock cycles needed for the necessary code compared to the number of clock cycles available between samples at desired conversion rates. Some improvement can be achieved by emphasizing timing (rather than good programming techniques) in preparing code. Operation of the parallel transfer without interrupts would save the overhead time of processing the interrupt (register saves and restores, etc.), for example. While using interrupts for the a/d increased throughput, it did so in spite of more instruction clock cycles required, by preventing the parallel transfer from taking priority over the a/d service. Use of real-time hardware modules which do not require as much software to service would also give some improvement. Substitution of a faster processor, such as the NEC V20, which is pin compatible with the Intel 8088 but five to ten per cent faster for the same clock rate, would also give marginal improvement.

More significant gains require removing some of the input/output overhead. For board to board transfer, the STD bus could be used instead of the 8256 interface. This would require the transfer to be programmed through the master processor, as the peripheral processor boards cannot control the STD bus. This would remove much of the parallel transfer code from the peripheral processors, but the master would no longer be totally dedicated to the principal investigator. Because direct memory access (DMA) is available on the master board considered, the overhead for the master would be much less than on the peripheral processor boards. Alternatively, DMA could be added to the peripheral processor boards. The possibility of designing a "piggyback" module adding appropiate DMA hardware is worth investigating. By reducing the involvement of the processor in moving data between memory and the 8256 parallel interface chip for board to board transfer, the processor would have more time to service the real-time modules at design conversion rates.

The potential is great for major gains over the present

LSLE microcomputer. The multiple-processor approach is still the most attractive, because of the conceptual advantages of sharing work among many units. Further study should provide the means to improve significantly the data throughput rates.