FASTBUS SOFTWARE PROGRESS*

David B. Gustavson
Computation Research Group
Stanford Linear Accelerator Center

SLAC-PUB-2996
October 1982
(I)

Abstract

The current status of the Fastbus software development program of the Fastbus Software Working Group is reported, and future plans are discussed.

A package of Fastbus interface subroutines has been prepared as a proposed standard, language support for diagnostics and bench testing has been developed, and new documentation to help users find these resources and use them effectively is being written.

Introduction

The Fastbus system relies on computers for initialization, diagnostics, and most normal operations, so computer software is an important part of the system. The Fastbus Software Working Group has been encouraging the production of generally useful software, coordinating software development efforts in the various laboratories, and developing useful standards in an effort to reduce the software burden on the user. Nevertheless, multiprocessor implementations will require some care and more sophistication on the part of the programmer than the single-processor systems we are used to.

Standard Subroutine Package

The primary area of activity of the Fastbus Software Working Group in the last year has been the design and documentation of a package of subroutines which provide a reasonably simple and portable, yet general, interface between the programmer and Fastbus. Though details of the Fastbus interface are certain to vary from installation to installation, just as the applications will vary, the existence of a standard subroutine package will reduce the effort needed to transfer programs and (more importantly) programmers from one system to another. In these days of large collaborations, a commonality of language and notation is more important than ever.

Though it is not possible to standardize the software interface in complete detail without standardizing the hardware of the processor interface as well, at least the names of routines, their general mode of operation, and the significance of their parameters can be specified. This will reduce the number of unproductive variations which otherwise would result at each installation. The package is being defined in a language-independent way, so that it can be applied easily to an implementation for any desired computer language.

Ruth Pordes, of Fermilab, has been the editor for the standard subroutines document, which we hope to present to the NIM Committee soon for approval. She will present these routines in more detail in another paper[1] presented at this symposium, and can provide copies of the current draft to interested parties on request.

Hardware Specification Review

Another major task of the working group has been the review and modification of drafts of the Fastbus Specification[2]. In a system which is so heavily computer oriented, it is important to consider the software effects of proposed changes to the "hardware" specification. Major changes have been made in the areas of status responses, control and status register allocation, interrupt handling, and the handling of reset. These changes were considered by the software working group and sometimes modified at our request. Though the desires of the programmer often conflict with the desire of the device designer for the simplest and cheapest solution, compromises are needed to minimize the real system cost. We think the compromises that have been made are acceptable, and have resulted in a practical system.

Language Support

Diagnostic and test-bench systems need an interactive language so that test programs can be quickly modified for the problem at hand. FDL (Fastbus Diagnostic Language) has been developed at the University of Illinois (UIUC) as one solution to this problem. FDL is implemented in Pascal for portability, but is an interpreter somewhat in the style of Basic, with Fastbus procedures built in. Forth, a self-extensible language which mingles an interpreter, compiler and assembler in a very compact package, has been chosen as SLAC's interactive system, and is also being used as the implementation language for the Snoop diagnostic software[3].

The working group has used Pascal as its publication language for algorithms[4], because it is a good language for the purpose and is widely available. Most of the early users of Fastbus plan to implement their data acquisition systems in Fortran, but in the long run a wide variety of languages will be used. Fermilab is now implementing the system management algorithms in Fortran.

Fastbus User's Handbook

The Fastbus Specification is unusual because of the amount of explanatory material included. This seems appropriate to some degree, because of the relative complexity of Fastbus compared to single-processor systems like CAMAC, but it is not practical to put all the information which users need in that one document. For example, the module designer needs a different kind of explanation and detail than does the system designer[5] or programmer.

Therefore, we decided to remove most of the information which is related to good practice, convention, interfaces or software from the Fastbus Specification document, and to create a new document with this information collected and expanded, called the Fastbus User's Handbook (working title).

* Work supported by the Department of Energy under contract number DE-AC03-76SF00515.

(Invited talk presented at the Nuclear Science Symposium in Washington, D. C., October 20-22, 1982.)

To speed production of this document, we are not laboring over the form and style but are trying to at least get pointers to the relevant articles collected and indexed for easy access. Richard M. Brown of UIUC is the editor of this new document, which is nearing its first draft.

Future Plans

There is a lot of useful work which could be done, but we do not have the resources to do everything. We hope to provide guidance, help eliminate duplication of effort by providing a forum for communication, and develop standards where they are appropriate, but we cannot solve many of the specific problems encountered in a particular implementation.

We plan to start work soon on defining the behavior we want from a Buffered Interconnect. The Fastbus Specification defines a Segment Interconnect, which is intimately tied to the details of the hardware protocols. The Segment Interconnect provides immediate feedback during Fastbus operations on a cycle-by-cycle basis, which allows long-distance communication to act logically the same as local communication. This symmetry simplifies software and allows simple controllers to function effectively. However, the Segment Interconnect approach to system connection can be very inefficient because all the segments joined during an operation are blocked from any other use, degrading the parallelism of the multi-segment system. This problem is reduced somewhat for Fastbus by the ease of installing additional cable segments as needed.

Buffered Interconnects avoid this problem by internally storing a message or operation, waiting to transmit it until the transmission medium is free. However, there is no immediate feedback from the true destination. This means that higher-level protocols, like those used in computer networks, are needed in order to provide reliability by using a system of acknowledgments and time-outs. The transmission could occur in a variety of ways, including serial or parallel lines or cable segments.

It was important to solve the Segment Interconnect problem first, because it involved significant hardware design constraints if it was to work properly, and software is necessary for managing the system's interconnection routes. The Buffered Interconnect, on the other hand, uses the ordinary Fastbus hardware protocols in the same way as any other Master or Slave device, but requires software standardization if a plethora of ad-hoc message formats is to be avoided.

Network protocols for the Fastbus Serial Network also need standardization if that facility is to be widely useable. That system is now in a hardware prototyping stage, but protocols could be developed at any time, perhaps along with the Buffered Interconnect project.

Interrupts in Fastbus are just write operations using the normal protocols, but the content of the data written needs to be standardized to avoid future problems. Standard formats are needed, e.g., for error-reporting interrupts or Host-information-request interrupts.

The Fastbus Parameter Space, part of each device's Control and Status Register space which is normally implemented in PROM, contains a variety of information fields useful for system management. It also has provision for future expansion using a file directory and memory allocation mechanism. Standards for the names and contents of certain files will be needed. For example, it would be useful if the file 'FBPHELP' (if present) always contained information useful for learning or remembering how to use the particular device.

Acknowledgments

The following persons have contributed significantly to the work of the Fastbus Software Working Group, through personal attendance or written communication:

Bob Dobinson, Phil Ponting and E. M. Rimmer of CERN

Jeff Appel, Al Brenner, Steve Gannon, Marvin Johnson, Terry Lagerlund, Ruth Pordes and Lou Taff of Fermilab

Tom Christopher, Martha Evens, W. Kabat, Albert Teng, and Tunghwa Wang of the Illinois Institute of Technology

Richard M. Brown, Bob Downing, Mike Haney, Dave Lesny, Keith Nater and Jerry Wray of the University of Illinois at Urbana (UIUC)

Dwayne Ethridge and Dennis Perry of the Los Alamos National Laboratory

Carl Akerlof of the University of Michigan

John McAlpine of the University of Saskatchewan

Steve Deiss, Dave Gustavson, Terry Holmes, Connie Logg and John Steffani of the Stanford Linear Accelerator Center

Ken Dawson of TRIUMF

References

1. R. Pordes, 'Standard Software Routines for Fastbus and their Implementation for a PDP-11/RT-11 System, Using the Unibus Processor Interface,' paper presented at this symposium.

2. Fastbus Specification, October 1982, U. S. NIM Committee, available from L. Costrell, Department of Commerce, National Bureau of Standards, Washington, DC 20234.

   R. S. Larsen, 'Status of the Fastbus Standard Data Bus,' IEEE Transactions on Nuclear Science, Vol. NS-28, No. 1, February 1981, pp. 322-329.

3. L. Paffrath et al., 'Fastbus Demonstration Systems,' IEEE Transactions on Nuclear Science, Vol. NS-29, No. 1, February 1982, pp. 90-93.

4. S. R. Deiss and D. B. Gustavson, 'Software for Managing Multicrate Fastbus Systems,' paper presented at this symposium.

5. D. B. Gustavson, 'Fastbus Status from a System Designer's Point of View,' IEEE Transactions on Nuclear Science, Vol. NS-28, No. 5, October 1981, pp. 3796-3800.

See also companion papers on Fastbus by R. Downing, H. Verweij and E. Siskind submitted for publication in this issue.