Fermilab

TM-1427
2320.000

Review of the Status of the FASTBUS Standard Routine Specification*

Ruth Pordes

Computing Department
Fermi National Accelerator Laboratory
Batavia, Illinois 60510 USA

November 1986

*(Submitted to the 1986 IEEE Nuclear Science Symposium, Washington, D.C., October 29-31, 1986)

Paper submitted to the 1986 IEE Nuclear Science Symposium:

REVIEW OF THE STATUS OF THE FASTBUS STANDARD ROUTINE SPECIFICATION

Ruth Pordes
Computing Department,
Fermilab, P.O.Box 500,
Batavia, Illinois 60510

Abstract

Within the next few months the FASTBUS Software Working group hopes to distribute the Specification for Standard Routines for FASTBUS. The draft specification will go to the members of the overseeing NIM committee for review. This paper presents the current status of the specification. It includes a list of the goals of the specification; some details of the concepts embedded in it; as well as an overview of the software implementations of the previously distributed draft versions of the specification.

Introduction

In this paper I present the status of the Specification for Standard Routines for FASTBUS. I will address some of the concepts contained in the specification and give an overview of the activities of the FASTBUS Software working group over the past year, and of existing software that implements drafts of the proposed specification.

History

As stated in papers presented at previous Nuclear Science Symposia, one of the goals of the committee that developed the FASTBUS hardware specification was to publish a technical specification on a standard software interface to perform actions on the bus. The goal is to define a common language with which to perform FASTBUS operations. The goal, for example, is to have every person who reads a register from the data space of a FASTBUS slave write the same line of code; He should not have to type FRD on a VAX using a UPI interface to FASTBUS, FRDAT on an IBM PC using a TSC interface to FASTBUS, or FREAD on a 68020 using a Aleph Event Builder interface to FASTBUS, etc and should be able to specify the same parameters on any system.

Work supported by the U.S. Department of Energy, Contract DE-AC-2-76CH0300.

The reasons for trying to achieve this goal are to aid

- portability of code,

- portability of programmers,

- those implementing software for new FASTBUS hardware

- designers of FASTBUS hardware that must be controlled or accessed by software

- in making software to access FASTBUS easier to maintain, understand and share.

Institutions such as Fermilab find that in the long term the definition of and adherence to software standards is beneficial to software implementers and users alike. The Fermilab Computing Department provides and supports software for many different experiments that use different computer systems, different programming languages, different interfaces to their hardware, and who write very different kinds of application programs. The existence of standards greatly ease this task; allows collaboration on and sharing of code; enables more effective education on the use of the code; and provides a common language for the discussion and development of applications of the code.

In the real world, of course, one must allow for implementations to use the specific features provided by the FASTBUS hardware interface being used. The specification for standard routines for FASTBUS must allow for such extensions and implementation specific details without making the whole specification so rigid as to be unusable. This is one of the areas that has made the achievement of the specification such a long process.

Software Needed for FASTBUS Applications

When any computer is used to access modules in a FASTBUS network there are several levels of software. The interface between the computer and the bus may itself contain software (microcode). The interface will be controlled by system level software, whose form and content are specific to the requirements of the interface and the computer system being used. There will be user interface software (to access the system software) which is included in the program of each user of FASTBUS. The users will write independent application specific programs that access FASTBUS. In addition, for large FASTBUS systems, there is clearly a need for system wide resource management and data base software to manage the allocation and use of the bus.

A goal of the Standard Routine specification is to provide a uniform software interface to the applications programmer. It is recognised that the system software and interface microcode will be much more specific to the hardware being used.

Since any microcode and system software must be provided before any applications programmer libraries, and it is these levels of software that dictate the functionality and throughput of the system, some implementations of FASTBUS software have concentrated only on these levels - where standardization is more difficult. These implementations are not mentioned in this paper.

Status of Standard Routine Specification

At the FASTBUS Software working group held in conjunction with the Nuclear Science Symposium last year, the 1985 Draft specification was considered to be in good technical condition when applied to one particular type of FASTBUS interface - list processing interfaces. Much technical editing has been done in the last year. It is expected that following the meeting in November we will be able to make the specification document available for review by members of the overseeing NIM committee.

During the long editing process, the technical specification of the software routines that perform the simple FASTBUS operations, such as reading a single FASTBUS register, has been essentially frozen. The name and parameter sequence of the transaction routines are defined as I reported at the Real Time Data Acquisition Conference in Chicago, eighteen months ago.

It is the meaning and use of the 'non-FASTBUS' entities such as the parameters and error control facilities, required in order to actually implement the routines, that have caused the most problems in arriving at a general standard and I expand on these more below.

Environment and Parameters

These are non-FASTBUS entities that are relevant to any software used to perform operations on FASTBUS.

Environment

It is not, in general, sufficient to provide a FASTBUS address, amount of data to transfer and a destination or source of the data, in order to perform the expected actions on the FASTBUS.

Associated information, such as the FASTBUS arbitration vector the interface should use for the operation, the time the interface should wait before timing out, whether the interface should regard a slave-status response of 2 as an error or not, all affect the actual performance and completion of the FASTBUS actions. This and other information that affect the actual operation of the requested FASTBUS actions are termed the 'Environment' accompanying each request. A well defined interface for initializing, changing, and interrogating the Environment, and for associating an Environment with requested FASTBUS actions, is included in the standard routine specification.

Parameters

Included within the Environment are parameter values that affect any operation performed. These parameters may be set or read by the user of the software. When FASTBUS operations are performed to Slave registers in data space, either one or two addressing cycles may be necessary, depending on whether the slave is addressed geographically, by slot position, or logically, with an assigned address that the slave recognises internally.

In the latter case, where no secondary address cycle should be done as part of the FASTBUS operation, a parameter is set by the user prior to calling the routine to do the FASTBUS operation. This parameter is stored within the Environment associated with the requested FASTBUS operation, and the FASTBUS interface does not do the cycle.

Error Control and Processing

The provision of three status response lines, and several independent timeouts that can be active when an addressed device does not respond make FASTBUS powerful and flexible. The interpretation of the status response to a FASTBUS cycle will vary between applications.

The Standard Routines specification includes mechanisms for the user to selectively define the response of the software or hardware interface to the slave status response. These selections are kept as part of the Environment. As an example, it is often possible for a FASTBUS interface to retry an operation that receives an SS=1 (Busy) response from a slave, without requiring any software intervention. The Environment defines whether this should be done for a particular operation, or whether SS=1 is to be regarded as a fatal error response.

Given the number of possible different error responses, the Standard Routines specification includes definitions of error codes, and routines for selectively decoding and reporting errors. In particular, when executing a list of operations, the user may wish to report only a selected set. The specification includes mechanisms - of necessity quite complicated - to allow for the automatic and selective reporting of errors.

List Processing And Embedded Processor Interfaces

Often the software overhead involved in giving instructions to a FASTBUS interface is much greater than the time taken to perform the actual FASTBUS operation. For this reason, some FASTBUS interfaces are designed to be List Processors; that is they can be given a single instruction to execute a list of FASTBUS operations.

The standard routine specification provides for list processing Environments, where calling a FASTBUS operation routine merely adds the encoded instructions for performing the operation to an existing

list, (but does not immediately tell the interface to perform the operation). Once such a list is built another routine is called to execute the list as a whole.

Alternatively, some FASTBUS interfaces have been built that incorporate a dedicated processor with the bus interface provided as an extension of the processor itself - a so-called 'Embedded Processor'. Such interfaces do not support lists. Whenever the processor encounters a FASTBUS operation instruction, it is sent straight to hardware to be executed.

Thus, list processing cannot be a required part of any general specification - and can be only a well defined extension to a basic standard.

Implementations of Draft Specifications

There have been two clearly identified Draft Versions of the Standard Routine Specifications. Implementations of these are in widespread use in the FASTBUS community. Below I give a short review of those of which I am aware with apologies for those I have omitted.

1983 Draft

The 1983 Draft Standard, otherwise known as the Blue Book, was implemented by the Computing Department at Fermilab and the University of Illinois Physics Department, for the IORFI-II and UNIBUS Processor Interface (UPI) for PDP-11s running the RT-11 or RSX-11M operating system and for the UPI under VAX/VMS; The PDP-11 software is being used by experiments and diagnostic test stands at Fermilab and collaborating institutions. TRIUMF has implemented a subset of the 1983 specification for the IORFI-II interface on a PDP-11 using RSX-11M.

A subset of the 1983 draft has been implemented by CERN for the FIORI, CFI, Fast Sequencer and LeCroy 1821 FASTBUS Interfaces, for VAX, NORD and 68K computers. This software is in use at CERN and at collaborating institutions all over Europe. It has been imported to Los Alamos, TRIUMF and Brookhaven. The software has been used in several experiments at CERN, and is being used by the LEP experiments in diagnostic test stands.

A subset of the 1983 draft has been implemented by LeCroy in the internal microcode of the LeCroy 1821 segment manager. This interface is being used by several fixed target experiments at Fermilab. It provides special features tailored to reading out the LeCroy FASTBUS TDCs and ADCs. Most of the large new fixed target experiments at Fermilab include a few FASTBUS crates among their mostly CAMAC data acquisition systems.

The data from FASTBUS is fed into buffer memories. The event data - spread out among several buffer memories on a single FASTBUS crate - are read out into the host computer using an 1821. This topology is becoming quite common, and in addition Fermilab is building a Smart CAMAC Crate controller which can feed data into the same FASTBUS buffer memories.

A (different) subset of the 1983 draft was also implemented at the Stanford Linear Accelerator on the VAX for the SLAC Scanner Processor; The National Laboratory for High Energy Physics in Japan (KEK) has implemented the 1983 draft for the FASTBUS Processor Interface, a high speed interface between the VAX and a FASTBUS cable segment.

1985 Draft

The 1985 Draft Standard, otherwise known as the FASTBUS Revisions, was implemented by the Collider Detector Facility (CDF) at Fermilab for the UPI (and QPI) interface. It is in heavy use for diagnostics and data acquisition at CDF itself and at institutions collaborating on the experiment. This implementation, based on the kernel of building and executing lists of operations, provides a very useful well defined interface for the coding of Compound Operation routines. These routines internally call many FASTBUS operation routines to execute a particular higher level function, for example to download a segment interconnect route table and initialize a FASTBUS segment.

This same implementation has been been adapted at the SLAC Linear Collider Detector (SLD) for use with the IORFI-II on a MicrovaxII system, and at New York Computing (NYCB) for the FASTBUS-MicrovaxII interface. A subset of the 1985 draft has been implemented at the University of Illinois for the IBM-PC interfaced to FASTBUS through a Test Segment Controller.

1986 Draft Standard

The proposed 1986 standard is now in the process of being implemented by CERN for its newly developed Aleph Event Builder and GPM interfaces, and will eventually be implemented for the new CERN Host Interface. The Fermilab Computing Department intends initially to implement the standard for the LeCroy 1821 interface.

Problems

Differences between Implementations of the Draft Standards

Implementations of the different draft standard are clearly not compatible. They do not satisfy the practical goals of providing common or portable software. In addition to the software mentioned above, there are several implementations of software that allow access

to FASTBUS that do not pretend to follow any standard. Whether, in spite of these obvious problems, a published draft standard routine specification will achieve any of the practical goals we have established remains to be seen. Experiments cannot change software in the middle. The Collider Detector Facility at Fermilab hopes to take a significant amount of data in the next year. It can clearly not change software interfaces at this stage.

At Fermilab, we are attempting to make all computers used for the data acquisition and test stands, those with 32-bit words; this precludes use of PDP-11s. It is therefore unlikely that any of the PDP-11 software would be converted to match a different standard.

On the positive side, in practice none of the draft standards are violently different. It is straightforward to see how to change a user level program from using one to using another. The same concepts are addressed in all drafts; it is the detailed descriptions that vary.

Our hope is that if a software standard is published, in the long term designers and implementers of software controlled FASTBUS Masters will try and adhere to it.

Generality of the Specification

One of the problems facing people writing the technical general specification has been that we are nearly all implementers. It has proved very difficult to separate out the requirements preferred by a specific implementation, from the requirements and details of a general specification. Thus the 1983 Draft specification was designed in conjunction with the hardware and microcode of the Unibus Processor Interface (UPI). The 1985 Draft follows closely the Collider Detector Facility requirements for their VAX/VMS data acquisition system using the UPI. The 1986 specification, which we hope to propose as a standard, tries to address a wider range of potential FASTBUS hardware interfaces.

User Level Documentation

One of the lessons I have learned from writing a general specification has been that one cannot use such a specification as a replacement for a user's guide to a particular implementation. The standard specification is directed to the implementer of software, it does not provide the user of the software all he needs to know to use it, for example that you must call routine A before B, etc. The standard specification cannot provide the level of detail needed by the applications user of the software. The specification now includes the statement that 'All implementations shall be accompanied by documentation describing them; their features that match the general specification and those that differ.'

Bibliography

Interested readers are referred to the proceedings of the Nuclear Science Symposium for the past six years, and the last three proceedings of the Conference on Computer Applications in Real Time. All the existing implementations mentioned in this paper are reported in these proceedings.

The FASTBUS Users Guide, which is being developed by the FASTBUS Users Guide Working Group, will include a bibliography of known publications on FASTBUS. The aim of the guide is to provide information on FASTBUS that supplements the specification, and to aid the hardware designer and the FASTBUS user in the use of FASTBUS systems. The bibliography is available from the working group. The Fermilab Computing Department Library maintains an index of FASTBUS documents available internallly, that includes many entries with authors outside the laboratory.

Acknowledgements

Publication of the FASTBUS software specification will have been made possible by the efforts of all the present and past members of the FASTBUS Software Working group, the Revisions and Editing sub-committees of that group, as well as the organizations that support the individual members of the group. In particular, the editor, Ken Dawson of TRIUMF, has spent many hours in order to arrive at a mutually acceptable and consistent document.

My work has been supported by the Fermilab Computing Department.