FILE COPY

CERN-Data Handling Division
DD/83/13
E.M. Rimmer
May 1983

CERN'S APPROACH TO FASTBUS SOFTWARE

(Paper presented at the 3rd Biennial Conference
on Real-Time Computer Applications in Nuclear and
Particle Physics, Berkeley, USA, 16-19 May, 1983)

CERN'S APPROACH TO FASTBUS SOFTWARE

E.M.Rimmer
DD Division
CERN
1211 Geneva 23
Switzerland

ABSTRACT

Software (and firmware) has been produced to support FASTBUS Processor Interfaces. It is based on a library of Routines callable from high-level languages such as FORTRAN or BASIC. The Routines are implemented by compiling interface-independent Lists. List Elements can be regarded as machine code, called F-Code, for a hypothetical FASTBUS Machine. The Lists are executed by interface-dependent FASTBUS List Processors, implemented in software, firmware or a mixture. The List Processors can also be used by Event Builders, in a data acquisition environment, to execute pre-compiled Lists.

Introduction

FASTBUS

Since many readers will be unfamiliar with FASTBUS, a brief outline is given here. FASTBUS is a new standard parallel bus system for high-speed and/or large-scale data acquisition and processing (1). It is built from back-plane segments, housed in crates, linked by cable segments for inter-crate communication. Handshaked bus protocols are uniform system-wide, enabling communication between a Master and connected Slave(s) whether the Slave(s) is on the local segment or on a remote segment. Standard Segment Interconnects are used to route inter-segment traffic. An arbitration mechanism resolves contention between competing Masters. Segments and non-intersecting routes can operate autonomously, allowing a large degree of parallelism.

FASTBUS Protocols

FASTBUS protocols define mandatory procedures for the fundamental sequences in any FASTBUS transaction. These are :

- Master procedures for participating in the arbitration process
- Master procedures for asserting Slave Addresses
- Slave procedures for connection after address recognition
- Master/Slave procedures for data exchange after connection
- Master/Slave procedures for disconnection
- Master procedures for relinquishing bus mastership

By means of simple signal interlocks, FASTBUS protocols allow an almost limitless variety of transactions between a Master and connected Slave(s) during a single period of bus mastership.

After arbitration, the winning Master establishes an Arbitration Lock, which

is retained until bus mastership is no longer required. During the Arbitration Locked period, the Master can perform one or more Address-Locks, that is, connection, data exchange and disconnection, with one or more Slaves. During an Address Locked period, the Master can perform as many data exchanges as required with the connected Slave(s), in the read direction, the write direction or both.

Software Environment

At CERN, user programs are usually written in FORTRAN or varieties of BASIC and ideally should be portable between several types of mini and microcomputers. A portable Interpreter, PILS, written in PASCAL, is being implemented at CERN (2). The PILS language contains Minimal BASIC as a subset, and allows user access to standard FORTRAN-callable libraries. PILS user code will be immediately portable between computers; differences in BASIC dialects offered by manufacturers presently hampers such portability.

CERN Input/Output software support is based on FORTRAN-callable libraries rather than the development equipment-oriented languages. For example, programmed CAMAC I/O is supported via standard ESONE-NIM Routines (3). System Linker/Loaders create user programs with access to selected facilities, such as CAMAC I/O plus histogramming plus graphics. The same library-based approach has been followed for FASTBUS.

General purpose FASTBUS Interfaces and Controllers must support all allowed Arbitration and Address locked sequences, as well as illegal sequences generated for diagnostic purposes. This has led to the choice of FASTBUS List Processing as a basis for implementing Interface software and firmware. A draft proposal for the use of List Processing has been made by the FASTBUS Software Group (4) and the same approach has been adopted for the software for a VAX 780/FASTBUS Interface (5).

Hardware Environment

At CERN, all supported minicomputers are equipped with standard CAMAC hardware and software. Therefore, to accelerate the penetration of FASTBUS into High Energy Physics experiments, the first two Interfaces for which List Processors have been written drive FASTBUS Controllers via CAMAC, as shown in Figures 1 and 2. They are the FIORI, a FASTBUS-to-Input/Output Register Interface (6), and a prototype of the CFI, a CAMAC-to-FASTBUS Intelligent Interface (7).

Both Interfaces could be driven by data and control channels connected directly to the Host computer. Such a connection may be developed later to adapt the CFI for a limited set of Host computers. This will need a substantial system programming effort, presently avoided by using existing CAMAC software.

A project recently launched at CERN provides support for Motorola 68000 processors within the VME bus system. A FORTRAN cross-compiler will be provided for the M68000; PILS has already been installed on this microcomputer. A VME-based Input-Output register, compatible with the CAMAC I/O register used to drive the FIORI, is under development. By emulating standard CAMAC calls in the M68000/VME system, the CAMAC/FASTBUS List Processors will be portable to the VME/FASTBUS environment. A prototype set-up is expected by the end of 1983.

Software Architecture

Figure 3 illustrates the basic architecture of the FASTBUS software, which is described in detail in reference 8. All FASTBUS transactions, whether primitive or complex, can be decomposed into sequences which are combinations of the protocol procedures listed above. This structure is mirrored in the definition of a FASTBUS 'Machine Code', F-Code. Transactions or sequences of transactions represented in F-Code are called Lists, and individual List Elements, or F-Code Instructions, define a particular protocol procedure or subprocedure. An F-Code Instruction can contain

information such as FASTBUS Address, Data, block length, etc. The Lists, and the List Compiler which generates them, are entirely interface-independent.

Lists are interpreted and executed by interface-dependent List Processors implemented in Host software, Interface firmware or a mixture, depending on the Interface architecture. A specific List Processor can be used over the entire application range from module development to data acquisition Event Builders. Use of the same List Compiler and Processor in both test and production environments eases hardware debugging, often complicated by differences between diagnostic and application software.

If necessary, List Pre-Processors can be used either before or after List storage as indicated in Fig.3, to produce Lists which are better matched to specific List Processor hardware or software. Ideally however, F-Code should be standardised, so that future FASTBUS Interfaces and Controllers can be designed as efficient F-Code Machines.

The List Compiler

Lists are produced by a portable List Compiler, written in FORTRAN IV, a restriction imposed by the lack of FORTRAN-77 or PASCAL on some CERN computers. The List Compiler is invoked by calls made to the proposed standard FASTBUS Routines (4). Work is in progress to attach the List Compiler to a compile-and-store facility to generate Lists for use in the real-time data acquisition of complex physics events.

The FASTBUS Routines proposal defines calling sequences for the common FASTBUS operations such as Read single data word, Write data block, Broadcast single control word, and so on, each of which corresponds to a sequence of F-Code Instructions. In addition, the proposal contains definitions of some List Manipulation Routines but does not suggest a standard F-Code Instruction format for List Elements.

When a FASTBUS Routine is called, the List Compiler generates the

FIGURE 1

FIGURE 2

FIGURE 3

corresponding F-Code List and passes it to a List Processor for immediate execution. Alternatively, the user can specify that Processing is to be delayed until a compound List has been compiled from several Routine calls. In addition, the Compiler allows individual F-Code Instructions to be inserted into Lists. In this way, it is expected that most user needs are covered, from simple test and diagnostic requirements to sophisticated event building.

The List Processors

Three List Processors have been written for the CERN Interfaces; two for the FIORI and one for the CFI.

Portable CAMAC-FIORI List Processor

A software List Processor has been written, in FORTRAN IV, which runs in a CAMAC-equipped Host computer and generates standard CAMAC Routine calls to drive the I/O Register connected to the FIORI. This implementation is by no means efficient. However, it is portable to any system supporting FORTRAN and the standard CAMAC Routines (only 2 of these Routines are used in practice). It is running on PDP-11 and HP21MX machines, and is being installed on Norsk Data and VAX machines. It was relatively painless to produce, modify and debug, which was very important during the development stages of the hardware and software design. Furthermore, it provides 'readable' reference code which describes the operation of the hardware, and was used to ease the implementation of the PDP-11-specific Processor.

PDP-11 FIORI List Processor

The main software overhead in the portable FIORI List Processor is due to the I/O calls made to the operating system to execute CAMAC cycles. On average, 10-20 CAMAC cycles are needed to execute a simple FASTBUS transaction, and in the PDP-11 RSX-11M system, a QIO call takes about 1 millisecond. Therefore a special QIO, to execute an entire FASTBUS List, was incorporated in the standard CAMAC Driver of the CERN-supported PDP-11

software. With this Driver, FASTBUS Block Transfer data exchange is accomplished at around 60 microseconds per 32-bit word on a PDP-11/60.

Portable CFI List Processor

The prototype CAMAC-to-FASTBUS Intelligent Interface has 3 components, shown in Figure 2; a CAMAC I/O Module CAMPORT, a Motorola 68000 KDM Development Module in a stand-alone crate, and a FASTBUS Backplane Module FBPORT. These modules are interconnected by a Motorola KDM-like bus.

The FBPORT, which is the subject of another paper at this Conference (9), drives FASTBUS via an AMD2910 microprogram sequencer with associated microstore and condition logic. The sequencer is controlled from a Transaction Store which is loaded with a set of pointers to micro-code sequences. These sequences (almost) map onto F-Code Instructions.

The M68000 dialogues with the Host computer via CAMPORT for List downloading and data buffer exchange. It does a certain amount of List Pre-Processing before loading the Transaction Store, and dialogues with FBPORT for execution, termination and exception handling of the FASTBUS Lists.

Thus the function of the List Processor proper is distributed amongst these modules, in software and firmware, and the portion which runs in the Host consists of a relatively simple CAMAC dialogue. FASTBUS transactions run at the speed of the FBPORT sequencer plus its local memory, and FASTBUS Block Transfer data exchange is accomplished at around 150 nanoseconds per 32-bit word.

The prototype Host software has been implemented by generating standard CAMAC calls to perform the dialogue with the M68000. The CFI is intended for use in a real-time environment where physics events are described by pre-compiled Lists. The Lists, and any associated output data, are downloaded into M68000 memory (and into the Transaction Store) at initialisation time and execution is triggered by a single CAMAC operation. Read

data are buffered in M68000 memory and extracted at CAMAC speeds after List execution.

The design of this interface has been reviewed recently and the architecture may undergo modification before a Mark 2 version is produced. The main design deficiency is the need to buffer read data before transferring through CAMPORT. This is not a severe handicap when the Host system is based on CAMAC, but would cause undue degradation of throughput if the Host system could itself support fast 32-bit data transfers. Therefore, an alternative fast data path, by-passing the intermediate buffer, is being studied.

F-Code Instructions

F-Code Instructions (List Elements) have a fixed length of 8 bytes. 2 bytes are allocated for the Operation Code, 2 bytes for Operation code extension, and 4 bytes for optional FASTBUS data or other FASTBUS parameter. An Instruction defines either a FASTBUS hardware sub-operation or (software) conditions to be applied during List processing.

bytes 0/1 = OP Code
 bytes 2/3 = OP Code extension
 bytes 4/5 = Optional FASTBUS data
 or other parameter
 bytes 6/7 = Optional FASTBUS data
 or other parameter

For a FASTBUS operation which involves the transfer of 4 bytes of information to or from the Address/Data lines, bytes 4/7 can be the source or sink of this information, or can contain a pointer to the source/sink. For FASTBUS Block Transfer operations, associated output/input buffers are used as the data source/sink, and bytes 4/5 contain the maximum byte transfer count.

The complete set of defined List Elements can be found in Ref.8. An example of a List is given here which contains Elements to define the following sequence; arbitration, connection to the Data Space of Slave Number 1 for a Block Data read, followed by connection to the Control Space of Slave Number 2 for a single data word write;

data associated with the block read are stored in an associated input buffer; the OP codes are given in decimal and the OP code extension bytes are not used.

bytes 0/1 2/3 4/5 6/7

111		LIST I.D.	Start of List
52		ARB.LEVEL	Arbitrate
32		ADDR.1	
4		SEC.ADD.1	Connect, Data Space, Slave 1
58		BLK.LEN.	Read Data Block
54			Disconnect
34		ADDR.2	
4		SEC.ADD.2	Connect, Control Space, Slave 2
0		DATA WORD	Write data word
54			Disconnect
53			Release bus
97		LIST I.D.	End of List

Authorship

The List Compiler, portable FIORI List Processor and portable CFI Host software were written by E.M.Rimmer, CERN.

The CFI M68000 software was written by C.Bizeau, CERN. The FBPORT microcode was written by A.W.Booth, presently at the Department of Electrical Engineering, University of Aston, Birmingham, UK.

The PDP-11 FIORI List Processor was written by D.R.Botterill, presently at the Rutherford Appleton Laboratory, UK, and by D.Burckhart, CERN. D.Burckhart also wrote an early version of the FIORI software which called standard CAMAC Routines but which was not based on List Processing.

Conclusions

The definition and execution of F-Code Instruction Lists as a means of

implementing FASTBUS Interface and Controller software and firmware allows the powerful flexibility of this new data handling system to be fully exploited.

By implementing a portable List Compiler and portable List Processors, support and maintenance loads are kept to a minimum and software homogeneity is established across a range of operating environments, from module debugging to complex data acquisition.

Further benefit could be derived from a formal standardisation of F-Code, to allow FASTBUS Interfaces and Controllers to be designed as efficient F-Code machines.

References

1. FASTBUS, A Modular High-Speed Data Acquisition System for High-Energy Physics and other Applications, U.S. NIM Committee, November 1982
2. PILS, A Portable Interactive Language System, D.O.Williams, R.D.Russell, Proceedings of Real-Time Data '82, pp.157-159, INRIA, Paris

Programmers' Introduction to PILS, R.D.Russell, DD/OC/82/3, 19 pages, DD Division, CERN
3. Subroutines for CAMAC, ESONE/SR/01; DOE/EV-0016; IEEE Std.758
4. Standard Routines for FASTBUS, FASTBUS Software Working Group, R.Pordes (Editor), FERMI National Laboratory
5. E.J.Siskind, Private Communication
6. FIORI, CERN FASTBUS-to-I/O Register Interface, C.F.Parkman, DD Division, CERN, December 1982

The FIORI is a CERN derivative of the IORFI, I/O Register-to-FASTBUS Interface; C.Logg and L.Paffrath, SLAC
7. CFI, CERN CAMAC-to-FASTBUS Interface, Draft Specification, R.MaClaren et al, DD Division, CERN, March 1983
8. CERN Software for FASTBUS/CAMAC, E.M.Rimmer, DD Division, CERN, March 1983
9. FBPORT, An Intelligent Interface between FASTBUS and a Microcomputer Bus, A.W.Booth, these Proceedings