A "FRONT PANEL" HUMAN INTERFACE FOR FASTBUS*

SLAC-PUB-2640
October 1980
(I)

David B. Gustavson, Terry L. Holmes,
Leo Paffrath and John P. Steffani
Stanford Linear Accelerator Center
Stanford University, Stanford, California 94305

ABSTRACT

A human interface based on the Snoop<1> diagnostic module has been designed to facilitate checkout of FASTBUS<2> devices, diagnosis of system faults, and monitoring of system performance. This system, which is a generalization of the usual computer front panel or control console, includes logic analyzer functions, display and manual-control access to other modules, a microprocessor which allows the user to create and execute diagnostic programs and store them on a mini-floppy disk, and a diagnostic network which allows remote console operation and coordination of information from multiple segments' Snoops.

THE FASTBUS ENVIRONMENT

FASTBUS is a powerful and flexible data acquisition and general purpose modular bus system which supports multiple processors and multiple busses (segments). Such systems can be quite complex, so FASTBUS has been designed from the beginning with the problems of fault avoidance, detection and diagnosis in mind. A FASTBUS diagnostic module, called the Snoop Module, has been designed to facilitate the process of detection and diagnosis.

Because large or complex systems are difficult to initialize correctly by manual operations such as setting switches, FASTBUS avoids switches. Initialization must therefore be performed by a Host computer in the system, which loads registers in the modules in order to specify addresses, priorities, and options. However, for the service technician or user who deals with only a few modules at a time such an automated system is an annoying and wasteful overhead. For these simple situations, a convenient interface equivalent to manually operated switches is needed. Displays and controls similar to those provided by the "front panel" or console of a small computer are needed for the module repair technician. The Snoop Module contains a microprocessor and the necessary hardware to become a Master on the FASTBUS, along with an interface to a display, keyboard and diskette storage device in order to provide these functions.

In normal large-system operation it is desirable to run occasional non-destructive tests at low priority, such as writing data to an unused register and reading it back, or reading known test patterns. The Snoop contains enough hardware to allow its microprocessor to simulate a simple slave with module type identifier, test patterns, switch registers, display registers, and scratchpad registers for this purpose.

---

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

Contributed to the 1980 IEEE Nuclear Science and Nuclear Power Systems Symposium, Orlando, Florida Nov. 5-7, 1980.

When a FASTBUS system has failed or is being exercised by the Snoop, it cannot be used simultaneously for communication with the diagnostic equipment. Furthermore, diagnosis may require comparison of bus signals on several segments in order to localize interconnection failures. FASTBUS provides a Serial Diagnostic Network (FSDN) to solve this problem. The Snoop provides a convenient access point for human interface to the FSDN when its supporting terminal is connected. The Snoop can also be controlled through the FSDN from a remote terminal. Snoops can communicate with their neighbors and with the system Host processor to set up test conditions and coordinate test results for interconnect fault isolation.

Complex multiprocessor multisegment systems like FASTBUS need performance monitoring tools in order to discover the location of bottlenecks or overloaded segments. The Snoop can take statistical samples and snapshots of FASTBUS activity periodically, providing the needed information. Bottlenecks can frequently be removed without moving modules or changing addresses by merely adding a direct interconnection between two overly communicative segments, since FASTBUS allows an arbitrary interconnect topology. Predicting traffic loads in advance is not always easy, so measurements and subsequent optimizations will be needed occasionally in most large systems.

IMPLEMENTATION STRATEGY

In a FASTBUS system, there is too much information to be displayed on any module front panel. A compact terminal is needed which can be easily moved to a location convenient for the current problem. The package also should include a small floppy disk for storage of diagnostic routines to be executed in the Snoop. We selected the Heath (now Zenith) H89, an inexpensive but compact, rugged and convenient package which includes a Z80 microprocessor with 48 kilobytes of random access memory and two serial ports. One serial port communicates with the Snoop, and the other is available for connection of an optional hardcopy device.

The general structure of the software is shown in Figure 1.

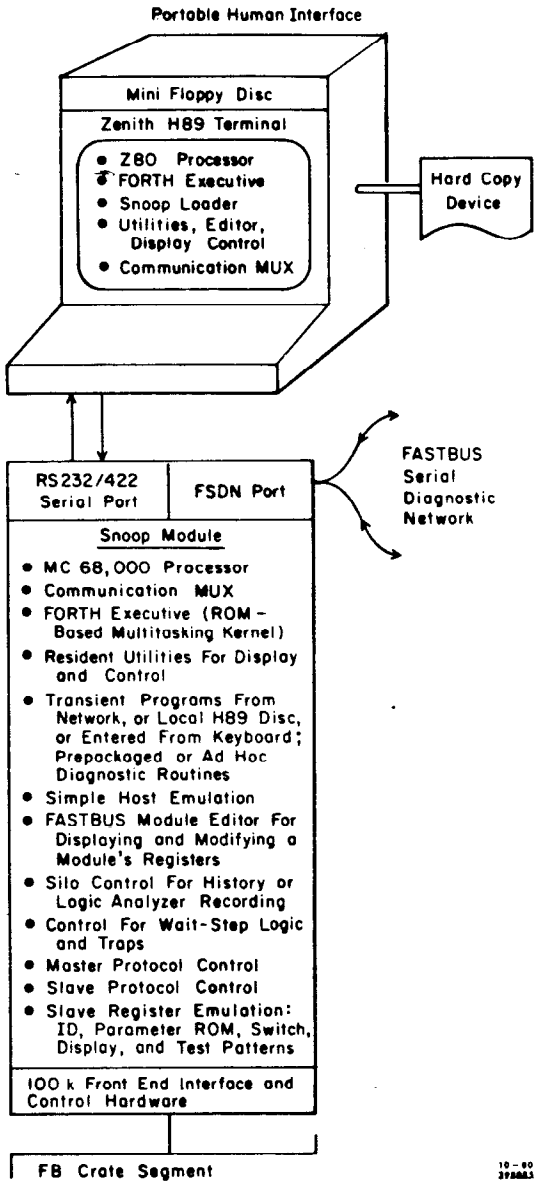

Figure 1. Snoop Software Organization

FORTH<3> was selected as the system implementation language because we needed multitasking, interrupt handling, full processor speed for certain critical code sequences, interactive diagnostic program development and execution by the ultimate user, and an extensible language to support a wide variety of applications conveniently. No other single language covers this broad spectrum from machine code to structured high-level constructs in such a natural way.

The FORTH version used as a starting point was the public domain version distributed by the FORTH Interest Group<4>. Though we have enhanced and optimized it for our needs and have rewritten it for the MC68000, it remains faithful to the FIG model and the proposed FORTH standard. Its public domain nature will facilitate distribution of our work to other Snoop users. Its primary disadvantages lie in its novelty and its potential unreadability. There are not many experienced FORTH programmers as yet, and FORTH is not much like any other language. Its unfamiliarity, Reverse Polish Notation and extensibility conspire to make FORTH programs hard for the novice to read. It is easy to write obscurely in any language, but easier in FORTH than in most others. Nevertheless, disciplined programmers can and do write intelligible programs in FORTH, and many technicians have mastered FORTH at the test bench. When extensions appropriate to the application and environment are provided, the system can be quickly learned and efficiently adapted and used.

Because it was not practical to include a floppy disk in each Snoop, it was decided to simulate one by serial communication with the supporting terminal, whether connected directly or through the FSDN. Disk traffic thus shares the communication path with display and keyboard traffic. In most applications envisioned for this system, disk traffic is infrequent and light. FORTH is very compact, so programs are short. The worst case will probably be data gathering for FASTBUS traffic measurements.

The Snoop contains all the essential software for responding to FSDN messages and responding as a FASTBUS slave in read-only memory (ROM). Thus Snoops power up in a useful state even if no disk is available. If ROM storage should prove insufficient, an automatic bootstrap via the FSDN is easily implemented with help from a bootstrap server on the network, perhaps the Host computer.

Since program creation and editing are directly related to the disk contents, the program editor can run in the H89 with no communication with the Snoop. Another task of the H89 is the multiplexing of communications over the single serial line to the Snoop so that the one line can be shared by terminal command string traffic and by binary disk block images. Part of the work of formatting displays is also done by the H89. Command lines for the Snoop are also collected and edited in the H89.

A similar multiplexer is needed in the Snoop, but it must take care of traffic from the FSDN as well as from the H89. It must also provide a transparent mode which routes traffic from H89 to FSDN enroute to a remote Snoop or other station, while also supporting Snoop to Snoop communication through the FSDN.

Since FORTH runs interpretively at its higher (machine independent) levels, it is easy to do exotic things like have one program create another and send it to a remote Snoop for execution. One simple example of such behavior is the loading of a diagnostic program into the Snoop by the FORTH loader which is running on the support terminal.

THE HUMAN INTERFACE

The effectiveness of a tool such as the Snoop is largely determined by the appropriateness of the mechanisms provided for displaying information and controlling the hardware, and their ease of use.

Display formats are particularly important, as there is so much information available that it can be difficult to find the relevant items. Formats appropriate to the more common applications will be provided, but the format-specifying primitives will also be

available. This allows the user to easily modify a standard display temporarily in order to tailor it to his current interests, or to create new display types and save their descriptors for future use.

A typical display, for example, shows the main control registers of the module at address A. The contents of the registers are changed by editing the display with the screen editor running on the H89 and then typing an update command which causes the new values to be written into the module.

Periodic displays can be requested which take a coherent snapshot of the module every few seconds. Display formats can be changed interactively by the user. The lower part of the screen is normally treated as an independent display used for displaying command lines, but can be used for other purposes as well.

The logic analyzer part of the Snoop has several modes of operation, and several corresponding display formats. Control of the analyzer is usually by means of screen-editing parameter fields in the display, but may also be by typed keyword commands. Commonly used setups can be stored and recalled as needed. One display uses a timing diagram format, with hexadecimal information accompanying each cycle as appropriate, and a scrolling mechanism to allow searching through the whole history silo of the Snoop. Comparing histories on two different segments requires a dual split display with independent scrolling, and some software assistance for finding matching entries in the two Snoop silos.

An initializer for small systems is driven by a hierarchy of displays, one showing which modules are to be initialized, and others showing the initialization actions needed for each module. The initialization specification is stored on the diskette and can be changed or executed as needed. This capability is especially useful on the test bench, where modules are frequently removed and replaced.

Convenient tools for describing FASTBUS operations will also be included. For example, to read a single word from a given address and leave the result on the stack:

```
<address> DRD
```

To write to a particular control register:

```
<data> <register number> <module address> XCWR
```

Individual FASTBUS cycles can be described and combined into arbitrarily complex operations by using a more primitive level of descriptors. For example, XCWR above could be described in terms of an address cycle, extended address cycle and data write cycle.

The performance measurement system is usually set up and coordinated by the Host, the computer which has all the system configuration information. The H89 will receive the display information from the Host through the FSDN. If it is necessary, this information can also be gathered by the H89 through the FSDN, using configuration information previously obtained from the Host or manually entered. This avoids the problem of the measurement changing the behavior being measured.

The H89 can also communicate with processors through the FSDN in order to act as a remote terminal, if the processors support such remote terminals. Equipment other than Snoops is likely to use the FSDN in a shared manner in the future, with interesting possibilities including gateway connections to other networks and perhaps telephone ports which could permit remote diagnosis from the expert's home terminal.

In extensive systems, multiple H89's and other processors may be active, and communication among multiple persons cooperating in the use of the diagnostic system may be important. The FSDN should make this straightforward.

CURRENT STATUS

As of October, 1980, the H89 system is running the FORTH multitasking operating system developed for this application. Evolutionary improvements are still occurring, but the basic system is reliable and stable.

The real Snoop hardware is not yet available, but experience is being gained by connecting the H89 to a small Z80 processor which is also running FORTH. A low-speed network interface on the Z80 has been built and is beginning tests involving three Z80 "Snoops".

Support functions for the display system are being developed, but design of the format controls is not complete.

The FORTH kernel has been written for the MC68000, and testing on a KDM development board<5> will begin shortly.

SUMMARY

The Snoop/H89 system promises to be a flexible and convenient tool for a wide range of problems, from diagnosis in large systems to single-module systems on the test bench. It could even serve as the main computer for small systems. Though its full capability will be reached only through time and an evolution based on practical experience, the underlying system has been designed to provide a sound basis for that evolution, and the essential capabilities should be available as soon as the hardware is ready.

REFERENCES

1. FASTBUS Snoop Diagnostic Module, R. Downing and H. Walz, contributed to the 1980 IEEE Nuclear Science and Nuclear Power Systems Symposium, Orlando, Florida Nov. 5-7, 1980.

2. FASTBUS Tentative Specification, July 1980, and updates September 1980, U. S. NIM Committee.

   Status of the FASTBUS Standard Data Bus, R. S. Larsen, invited paper presented at the 1980 IEEE Nuclear Science and Nuclear Power Systems Symposium, Orlando, Florida Nov. 5-7, 1980.

3. A good introduction to FORTH can be found in the August 1980 FORTH theme issue of Byte magazine.

4. FORTH Interest Group, P. O. Box 1105, San Carlos, CA 94070

5. MC68000 Design Module User's Guide, MEX68KDM. A publication of Motorola Semiconductor Products, P. O. Box 20912, Phoenix, AZ 85036.