A MICROPROCESSOR CONTROLLED THREE-PORT INTERFACE TO THE 168/E PROCESSOR

M. Rost, G. Iksal

Department of Physics, University of Siegen, 59 Siegen
W. Germany

ABSTRACT:

The 168/E is a SLAC developed, fast processor, which emulates the IBM 370/168. For many applications one is forced to overlay the processor's memory. We designed an interface, which connects the 168/E to a host-computer and a low cost MOS-memory with an address-space of 4 M-bytes. This three-port interface is controlled by the Motorola MC 68000 microprocessor which also manages the execution of the several overlays. Since all control is performed by the 68000 software, the interface hardware is relatively simple. Seen from the host-computer side it connects the host to a 168/E-processor with a large memory. No restrictions are made to the number of program or data overlays. The maximum time spent overlaying the whole 168/E program and exchanging all data is limited to 70 msec. The connection of additional 168/E processors is possible.

I. Introduction

The 168/E designed and first built at SLAC[1], is a fast processor based on bit-slice chips, to emulate the important op-codes of the IBM 370/168. Fortran and assembler routines can be transferred from the IBM subject to minor restrictions. To work with the processor, access is needed to an IBM-computer to generate the necessary microcode and to a host computer to perform all I/O transfers and to organize the control of the 168/E. Because the size of the processor's memory is limited to 128 k-byte data and 96 k byte 168/E program memory, the host also has to manage overlaying of program and data.

Up to now, there exist several interfaces [2] between 168/E processors and a host, and in case of the SLAC designed Bermuda Triangle also to two large buffer memories to allow overlaying of program and data. Whereas these interfaces have proved to be very powerful in several applications, they all have the limitation that they are designed for a specific environment.

The aim of the three port interface described here is to use it together with one or two 168/E processors as a powerful "stand alone" computing facility which is independent of the host and the kind of buffer memory connected for overlay runs. The interface is controlled by the Motorola 68000 microprocessor, which allows us to reduce the necessary hardware to a minimum, and results in a high flexibility of the whole assembly.

II. Description of the interface

Fig. 1 shows the flow-diagram of the interface hardware. It connects the host (upper left) to a 168/E processor and the buffer memory whose address-space is limited to 4 M bytes. In our design, the clock and status control logic of the 168/E is considered to be part of the 168/E itself, and is installed on a special card.

Because it should be possible to change the host computer or to work completely off-line with a parallel link to a disk-drive, the select and interrupt logic (upper right) as well as the transmitters to the host side are installed on another separate card. In some cases (especially DEC PDP 11) it is possible to work with standard host-computer equipment.

The interface consists of four essential parts : three different busses and the controlling Motorola MC 68000 microprocessor. The micro has its own memory (16 K bytes in our design) which could be either RAM, EPROM or PROM. It also includes two asynchromous serial interfaces to connect it with a TTY and/or a floppy disk if necessary.

Most care has been taken to speed up the data transfer on the data busses. The MC 68000 supports 24 address bits, these are more than we need. By using the spare bits effectively as commands, the microprocessor can generate an address and command together i.e. fast. The command bits are applied to three control PROMS whose outputs control the various transmitters. This technique allows us to steer the interface data flow with very simple and fast macro instructions.

Going more into details, it is only the buffer memeory which needs 20 address bits. So all transfers concerning the buffer are controlled by the upper 3 address bits (23-21) provided from the microprocessor, whereas all other transfers which do not involve the buffer are steered by the next four bits (20-17). Table 1 gives a survey of all transfers which could occur, and the steering address bits of the Motorola processor involved. The lower 16 address bits (16-1) are connected parallel to the buffer and the two 168/E memories.
All transfers concerning the buffer use bits 20-17 additionally, which could be considered as a page address. A normal transfer between the buffer and a 168/E memory will copy the contents of a page or a part or it, always using the same (lower) addresses.

If necessary, a special 4-bit register network allows to set or reset bits 16-13 independently at the 168/E transceiver side (A-register, fig. 1). So we are able to divide a page into up to 16 parts, where each part could be transferred to another region in the corresponding buffer page or 168/E memory.

The page size of the buffer is defined by the control-software. It could vary in powers of two from 2^2 to 2^{16} k. Using some or all address bits 16-13 as enable signals to the corresponding 168/E transceivers, this means, that in principle the interface hardware is prepared to control one 168/E with 64 K-word memories or 2, 4, 8, 16 processors with 32, 16, 8, 4 k-word memories respectively.

The Motorola chip provides seven interrupt levels. Because it is comfortable to have the halt-signals of different 168/E's and the host interrupt connected to separate interrupt inputs (fig. 1), and to work with a still acceptable 168/E memory size, the number of 168/E's to be eventually controlled is restricted to four processors.The normal case will be one or two processors with 32 k-word memories.

The two data busses of the interface may work with different speeds. Whereas the 16 bit bus is responsible for loading the buffer and the Motorola RAM's and for controlling the 168/E's, the 32 bit bus is mainly used for the fast transfer of overlays between the buffer and the 168/E memories. Because only addresses are needed to control this transfer, we can make extended use of the Motorola test-long word instruction, which provides two transfer addresses within three (Motorola) micro-instructions (see Fig. 2). This means, that the fastest transfer speed is determined by the microprocessor cycle time, which is now 500 nsec per microinstruction (8 MHz type) and could be increased to 400 nsec (10 MHz type) later on.[3] So the fastest transfer rate using the interface is now 750 nsec per 32 bit word, which is 3/2 of the microprocessors micro-cycle.

III. Programming the interface

Let us first consider the fast data transfer on the 32 bit bus. The transfer program is very simple and consists of four steps:

- set up the upper address byte to address the interface control PROM
- transfer 2 k-words data performing a loop of TSTL-microinstructions
- compare the actual (16 bit) address to the wanted one and
- jump to the loop or finish.

This program needs about 12400 microinstructions. That means one has to wait 1.54 (1.24) msec for the transfer of 2k words, corresponding to a full word frequency of 1.31 (1.65) MHz.This transfer rate is an upper limit and could only be reached if the mean access time of the buffer falls in the range of 450 ns. If this access time is deferred for any reason (especially refresh-cycles of the memory), the 68000 microprocessors automatically waits, expecting a data acknowledgement signal to be asserted. The transfer time will then be increased in steps of 125 (100) ns, the cycle time of the micro.

This situation illustrates the time-flow diagram of the TSTL-instruction,

fig. 2. During the third microcycle, the buffer (as a slave) do not assert its DACK before the 4th clock-pulse rises, which causes the microprocessor to substitute one clock cycle. The data transfer rate thus takes 125 ns longer. If no data acknowledgement signal is sent, the microprocessor will wait forever.

The organisation of the data transfer on the 16 bit bus is similar to the 32 bit transfer. One may set up the same transfer loop using either MOVE or TST instructions. The difference in the data transfer rates will be 3.0 or 1.5 ms resp. for the transfer of 1 k 32-bit word (125 ns cycle).

Another point concerns the control of the 168/E microprocessors. They can be started and controlled by setting and reading the 168/E control register. In reality there are two registers, one for the input - e.g. the command input of the Motorola, and one for the status report of the 168/E to be read from the controlling system. The contents of the tow registers differ in some points.

A "Halt" on a 168/E causes an interrupt to the Motorola system. After the Motorola has finished its current work, it will respond in first reading four bytes of the 168/E data memory which are designed to contain the intention of the program being executed on the 168/E. The control system will then induce the next steps. Depending on its software, this could be the preparation and start of a next overlay step or a new program run or a simple wait for an operator command etc.

In conclusion to the software it is intended to create a "control library" which includes subroutines for all standard transfer and control tasks to be performed on the interface. These subroutines are installed together with an improved Motorola monitor in the EPROMs of the MC 68000 memory. At this place we also want to point out, that the programming of the Motorola MC 68000 could easily be done on various computers using the appropriate cross assembler.[4)]

IV. Conclusion

The microprocessor controlled three port interface introduced here, provides the use of up to four 168/E microprocessors as a low cost and powerful "stand alone" computing facility, which could be used for a wide range of tasks. The interface allows control by a standard TTY, programs and data could be transferred by either parallel or serial link.

Connected to a host, it could behave like an intelligent (two-port) interface to a 168/E processor with access to an enlarged memory. There is no special host computer type preferred. The buffer memory could be either 16 or 32 bit orientated, also different access times are allowed. The control of the data transfers and the overlay management is completely performed by software, which minimizes hardware effort and provides a very flexible system. Overlaying of the 168/E program and data memories has to be done

sequentially. The transfer rate is limited to a full word frequency in the range of 1.3 - 1.6 MHz. The speed mainly depends on the controlling microprocessor cyclus time and the buffer memory access time. Using 500 nsec MOS memory [5] and a 8 MHz Motorola chip, we measured 1 MHz on our system, which seems to us to be fast enough for all of our applications.

References:

1) P.F. Kunz et al., Nucl. Instruments and Methods 9, 435 (1976)
P.F. Kunz et al., SLAC-Pap. 2198, Sept. 1976

2) D. Bernstein et al., SLAC-Pap. 2413, Oct. 1979
R.F. Kunz et al., SLAC-Pap. 2418, Oct. 1979
CERN-DD, 168/E-project notes

3) Motorola MC 68 000 project notes

4) We also believe that a Fortran compiler will be available in future

5) CDC, PDP 11/70 add on memory 94/170

Acknowledgement:

We would like to thank A.F. Flavell, University of Glasgow, for helpfull discussions and constructive suggestions.

Table 1 Data Transfers

No.	Address-lines <23-21><20-17><16-1>			Function	Bus	Remarks
1	0	Ø	Ø	Reset	/	Reset
2	1	Addr.		Buffer → DMem	D 32	fast data transfer
3	2	Addr.		DMem → Buffer	D 32	
4	3	Addr.		Buffer → PMem	D 32	fast Progr. transfer
5	4	Addr.		PMem → Buffer	D 32	
6	5	Addr.		Host → Buffer	D 16/32	
7	6	Addr.		Buffer → Host	D 32/16	slow Data/Progr. transfer
8	7	Ø		MC → High-Word-Reg.	D 16	
9	7	1		High-Word-Reg → MC	D 16	
10	7	2		PAddr. → MC	D 16	read 168/E PC
11	7	3		MC → PM (Impl.)	D 16	start 168/E
12	7	4		MC → CS-Reg.	D 8	control 168/E
13	7	5		CS-Reg. → MC	D 8	
14	7	6		MC → Host	D 16	transfer of control programs
15	7	7		Host → MC	D 16	
16	7	8		Host → PMem	D 16	direct transf. Progr.
17	7	9		Host → PMem	D 16	direct transf. Data
18	7	10		PMem → Host	D 16	direct transf. Progr.
19	7	11		DMem → Host	D 16	direct transf. Data
20	7	12		Addr. 16-13 → Serv.Reg.	A4	segment.of pages

Three - Port - Interface

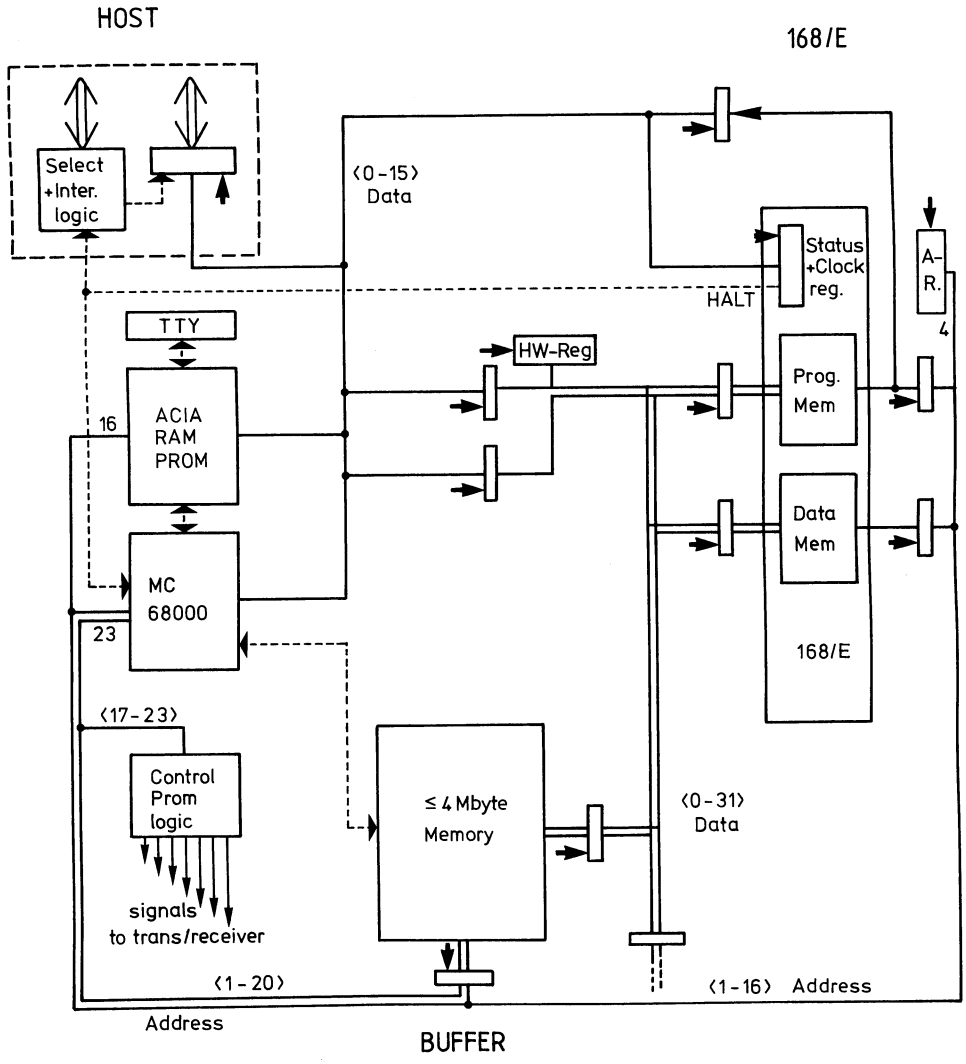

Fig. 1

timing diagram

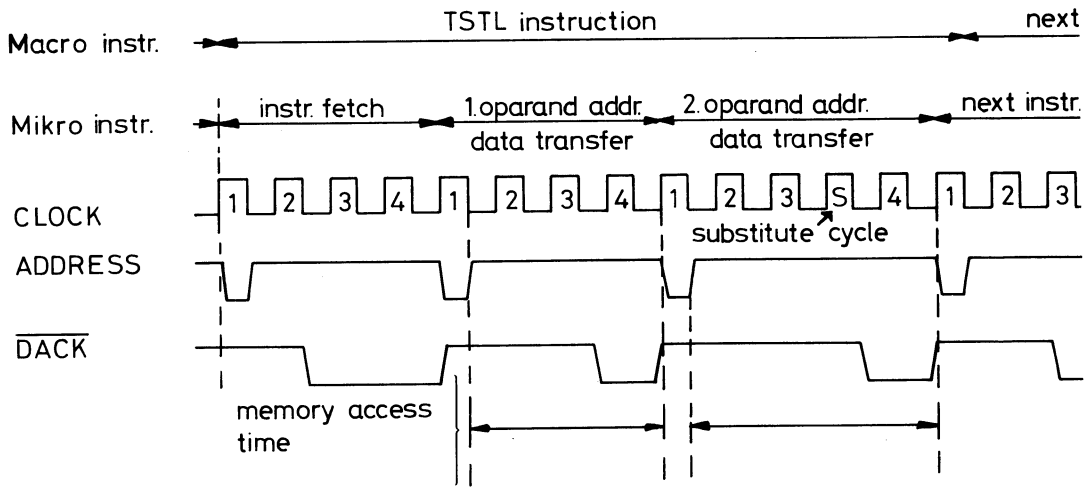

Fig. 2