ISTITUTO NAZIONALE DI FISICA NUCLEARE

Sezione di Bologna

INFN/TC-87/1
26 Marzo 1987

I. D'Antone, P. Giacomelli, G. Mandrioli and G. Sanzani:

A MICROVAX-VME INTERFACE

Servizio Documentazione
dei Laboratori Nazionali di Frascati

INFN - Istituto Nazionale di Fisica Nucleare
Sezione di Bologna

INFN/TC-87/1
26 Marzo 1987

A microVAX-VME Interface

(I. D'Antone, P. Giacomelli, G. Mandrioli, G. Sanzani)

ABSTRACT

A simple link between a microVAX and a VME crate has been realized using two commercial interfaces capable of DMA transfer.

A communication software has been developed that allows a reliable block transfer at an effective rate higher than 0.5 Mbytes/sec.

1.-INTRODUCTION

Many modern experiments in the high energy particle physics use the VME standard for their data acquisition systems, because it is cheap and modular and it allows a more hardware/software flexibility in the system design.

On the other hand the Digital Corporation has produced the microVAX machines, which are powerful and software compatible with the family of VAX computers.

The diffusion of these two standard and their contemporary use at different levels in the on-line application requires a fast link between the two systems.

We have realized a link capable of DMA transfer at a rate of 0.7 Mbytes/s with two commercial interface boards.

In the next chapters we describe the busses features and the detail of the interface used; then we give a description of the software developed for the link and the results obtained.

2.-THE HARDWARE SYSTEM.

A schematic view of the installed system is shown in fig.1.

The interface on the VMEbus is a FORCE commercial high-speed parallel I/O board designed to communicate fully asynchronously from the VME bus to the outside world. A master CPU and a memory bank on the VME crate are obviously necessary. We have used in our system a DATASUD CPUA1 with a 8 MHz 68010 and 256 kbytes of RAM.

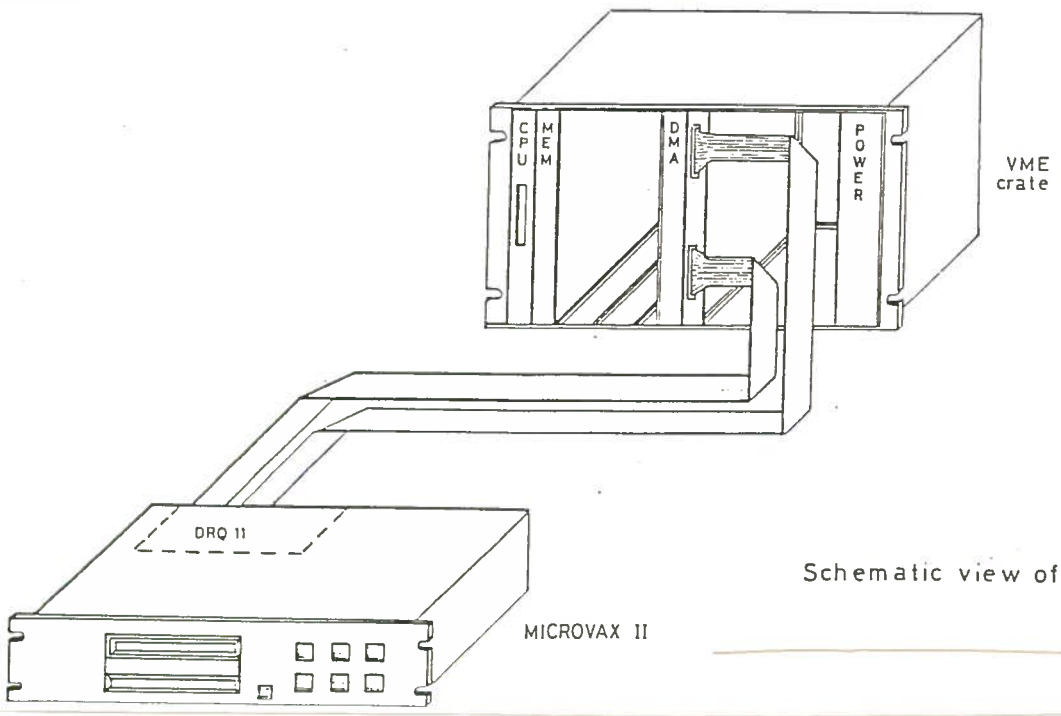


FIG.1

Schematic view of the hardware system used

The Qbus interface is the DRQ11-C delivered by Digital, which is a general purpose DMA interface for high speed data exchange within a user device and a microVAX computer.

3.-DATA BUSSES.

3.1.-VME bus.

The VME bus structure is based on a 16 bit data bus and a 24 bit address space non multiplexed, both expandable to 32 bit.

The VME bus has a master/slave asynchronous data transfer format; this allows multiple bus masters (typically processors and DMA controllers), and besides memories and peripherals of different speeds can be used without slowing the bus.

Data transfer rate as high as 20 Mbytes/s in the expanded 32 bit configuration is possible.

VME bus uses three basic cycle types in its data exchange protocol; these are read, write and a read- modify- write cycles.

Two signals cause the master to close every cycle: the " data transfer ackowledge " signifying that the slave has either read or write data and the "bus error" in response to an on-board error such as a memory parity error.

Any bus allowing multiple mastership must be capable of allocating (arbitrating) which master has the use of the data transfer at a given time. The VMEbus arbitration consists of four prioritized request levels and four prioritized grant levels.

The bus interrupt structure, which is vectorized, has also options which are useful to multiple master systems. There are seven interrupt request lines by which modules may ask for servicing by an interrupt handler (a processor capable of servicing system interrupts).

3.2.-Q22 bus.

The Q22 bus implements 22 bit addressing and 16 bit data transfer. The bus operates asynchronously with a master/slave relationship. The bus master, which is typically the processor or a DMA device, starts a bus transaction. The slave device responds acknowledging the transaction in progress and exchanging data with the bus master.

Seven types of data transfer operations can occur: data word input or output, data byte output, read- modify- write word or byte, data block input or output. In block mode multiple words can be transferred sequentially in both directions starting from a single bus address.

The bus interrupt structure is vectorized. There are four prioritized interrupt levels. When the processor acknowledges the interrupt request it receives from the device a vector address of the service routine.

4.-VME PARALLEL INTERFACE.

4.1.-General features.

The parallel interface FORCE OPIO-1 is a fully VME bus compatible double Eurocard board, providing four opto- isolated 8-bit parallel I/O channels with handshake.

On the board there are three types of programmable devices to provide maximum flexibility in the I/O configuration:

--four Parallel Interface/Timer (PI/T) 68230 supply the I/O ports to the external lines;

--two bus Interrupter Modules (BIM) 68153 provide a software programmable interrupt level structure;

--a Direct Memory Access Controller (DMAC) 68450 performs data transfers between the parallel channels and VMEbus slaves (memory cards), when the board is bus master.

Several channel structures are configurable via hardware and software:

--Bidirectional data port or separate input and output port can be choised;

--Two channels can be grouped together and used as a 16-bit port;

--Each input and output port has additional control lines, configurable as handshake, interrupt or status lines;

--the data transfer can be controlled by polling, interrupt or DMA

--Several handshake modes are software selectable.

The board occupies an address area of 512 bytes. The base address is jumper selectable in the 16 Mbytes address space in 512 bytes increments.

4.2.-Parallel ports.

The parallel I/O is designed with Parallel Interface and Timer modules (PI/T 68230). The clock frequency is 8 MHz.

Each PI/T on the board includes the following features:

--All registers are Read/Write and directly addressable

--One 8 bit output port

--One 8 bit input port

--One 8 bit port with six special function pins

--Selectable handshaking modes with double buffered data transfer.

Each I/O signal is available through two 64 pin I/O connectors or through the 64 pin DIN connector (P2).

The board can operate in a 16-bit parallel mode because two PI/T devices are connected to the lower data bus (D0-D7) and two to the upper data bus (D8-D15). This allows a parallel transfer of 16-bit data with a common handshake.

4.3.-DMA controller.

The 4 channel DMA Controller (DMAC 68450) has a clock frequency of 8 MHz. The maximum data transfer rate is 4 Mbytes/second. Each PI/T device is connected to a request input of the DMA controller. The connections scheme between DMA and PI/Ts are shown in fig.2.

The principal DMAC features are:

--4 independent channels
--17 registers/channel for complete software control
--interface lines for Requesting and Acknowledging
--Programmable channel prioritization
--2 vectored interrupts for each channel
--Auto-request and External-request transfer mode.

The DMA controller can operate on the VMEbus in the following transfer modes:

--Parallel I/O to Memory
--Memory to Parallel I/O
--Memory to Memory.

4.4.-The interrupt structure.

The PI/T devices and the DMA controller can generate interrupts.

The interrupt request signals from the PI/T devices are arbitrated by the two Bus Interrupt Modules (BIM). Each BIM contains 8 registers; they have Read/Write access and define which VMEbus interrupt levels are assigned to the request inputs. This structure allows for each PI/T request a variable interrupt level definition and a vector generation.

The interrupts generated by the DMAC are instead interfaced to the VME with a PAL device.

5.-DRQ11 INTERFACE.

5.1.-General features.

The DRQ11 is a general purpose DMA Interface for high speed data exchange between a user device and Qbus equipped Digital computers.

The interface can operate in two modes: slave or master. The slave mode is essentially the program controlled mode when the interface, as a slave to the processor, responds to its addresses on the bus. In master mode the interface gains control of the processor bus via the DMA request line and (as a bus master) performs a data transfer operation.

The DRQ11 operates directly to or from memory moving 16 bit data between the Qbus and the user device. The maximum block length is 64k words. For larger transfers the interface has an alternate buffer mode option.

The throughput of a DMA device is influenced by many factors, for example the load of the processor bus.

The DRQ11-C maximum data transfer rate is about 400 kwords/s (600 kwords in burst mode).

5.2.-Registers.

The interface operations are supervised by reading/ writing four addressable basic registers:

-- SCR status and command register
-- COR control register
-- ADR address register
-- DBR data buffer register.

The SCR is used for control and monitoring of the interface functions as well as for input and output of the status function bits from or to the user device.

The COR can be seen as an extension of the SCR to control the DMA operation. It also allows multiplexing of the address register ADR.

The ADR is a multiplexed register, which can be addressed to control and monitor a set of address and word count registers.

The DBR can behave as:

--a 16 bit write- only register (ODBR), from which outputs are available to the external device (DATA OUT lines)
--a 16 bit read-only register (IDBR), that reads information by the external device (DATA IN lines).

5.3.-Interrupt structure.

The interrupt structure is vectorized. The interface interrupts when either an error or ready condition exists and interrupt is enabled. An interrupt is caused by applying a pulse to a particular line.

6.-HARDWARE OPERATIONS.

6.1.-DRQ11 lines.

Two groups of equivalent lines are available in two 40 pins connectors of the interface: the transmit lines, marked with a "T", controlled by the DRQ11-C, and the receive lines. marked with an "R", controlled by the external device.

Besides the data lines, two groups of control lines are important:

--- 4 function lines (FUNCT_0 - 3 R or T), which may be used to exchange code with the external device.

In reception a pulse applied to the FUNCT_0_R line causes an interrupt to the microVAX.

In trasmission a pulse is produced by the microVAX setting the function bit FUNCT_0 in the SCR register; it may be used as an interrupt to the user device.

--- 2 lines used to synchronized the DMA transfer:
DAT_AVAI_R or T that means Data Available,
DATA_ACCE_R or T that means Data Accepted.

6.2.-Synchronization of DMA transfer.

Input transfers (user device data to microVAX memory) are synchronized with DATA_AVAI_R and DATA_ACCE_T signals.

A DMA bus cycle is requested with the edge of the DATA_AVAI_R signal (user controlled). The interface DRQ11-C sets the DATA_ACCE_T to acknowledge the current DMA input word.

Output transfers (memory data to user device) are synchronized with DATA_AVAI_T and DATA_ACCE_R signals.

The DATA_AVAI_T is sent out by the microVAX when the output data are valid.

The user latches and acknowledges data by sending back DATA_ACCE_R.

6.3.-VME chips programming.

The chips on the VME board are programmed to match with the DRQ11 protocol.

The PI/T devices, having an 8 bit wide Data bus, are organized as two pairs. Two PI/Ts of a pair are used to input one word and the others two PI/Ts are used to output one word. For this reason four ports are used for the data, one for each PI/T; two of the remaining ports are programmed as control lines.

The PI/Ts can be programmed in the interlocked handshake protocol using two pins (HA and HB on the connector) as control lines (see fig. 2).

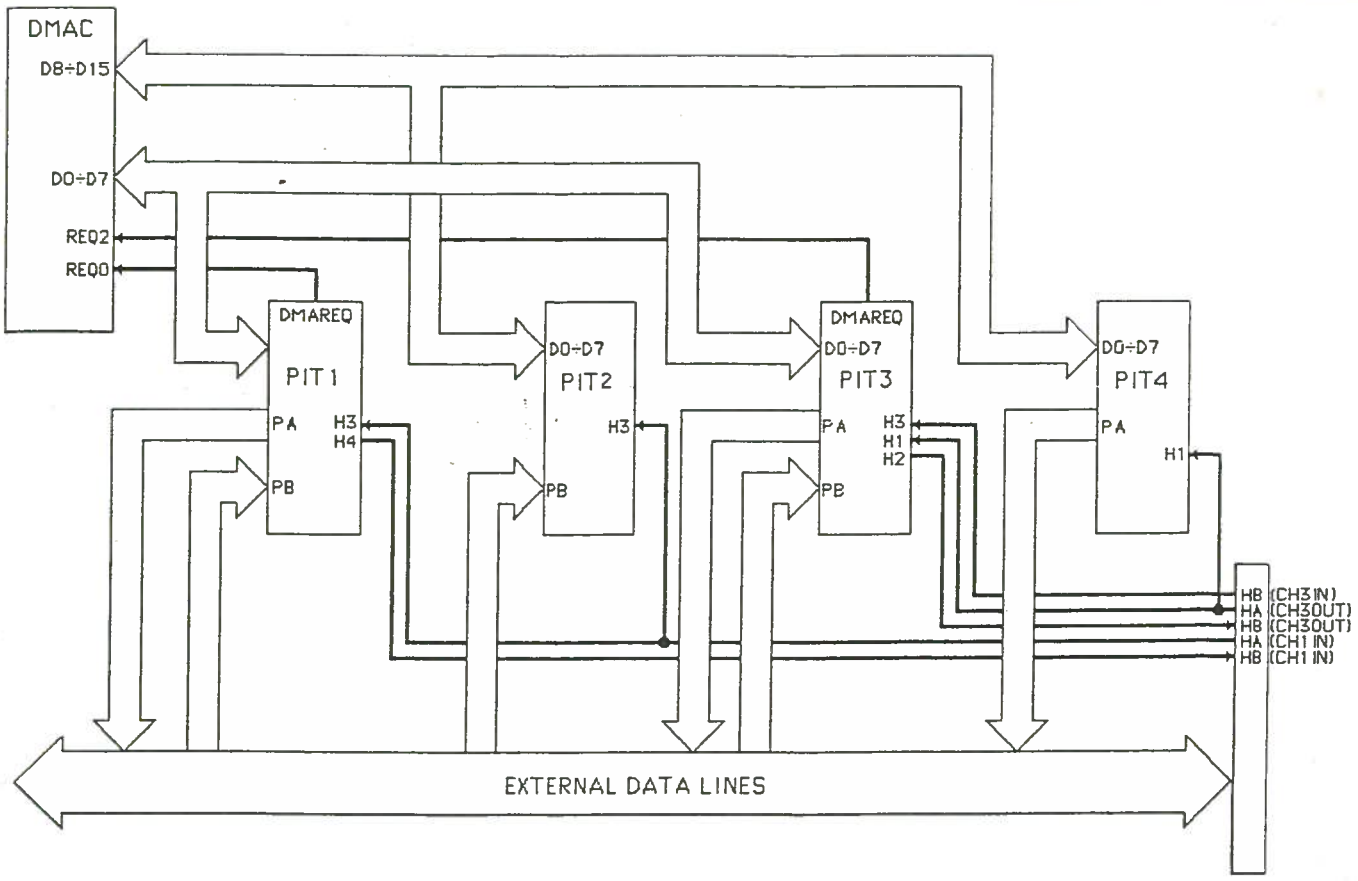


Fig. 2

DMAC-PI/Ts connections.

In PI/T input transfers, data are latched in the double- buffered input latches by asserting HB; at the same time a DMA request 'DMAREQ' is generated by the PI/T to signal the DMAC to empty the input buffer. HA is generated in response to the HB to indicate that new data may be accepted through the input port.

Then the synchronization with the DRQ11 is obtained linking HB with DATA_AVAI_T line and HA with DATA_ACCE_R line.

In the output transfers HB is asserted after data has been transferred to the double- buffered output latches. The data remains stable and HB remains asserted until the edge of the HA input.

The synchronization with the DRQ11 is obtained linking HB with DATA_AVAI_R and HA with DATA_ACCE_T.

As stated before the microVAX is interrupted applying a pulse on the FUNCT_0_R line, while to interrupt the VME CPU the FUNCT_0_T is connected to a special PI/T pin programmed to send an interrupt to the VMEbus when it has a pulse.

The FUNCT_R or T are written or read by the PI/Ts using a simple BIT I/O programming of the lines.

The DMAC programming is done according to the connections on board between PI/T and DMA controller. Particularly the transfers between memory and external device are programmed to work with 16 bit size of the operand.

6.4.-Hardware installing.

The main connections between DRQ11 and the equivalent VME lines have been described in the previous paragraphs.

From the mechanical point of view there is no match between the connectors of the two boards. Therefore we have prepared an "adapter board" to fit the VME interface connectors with the DRQ11 flat cables.

In the figures 3 a) and b) are shown the adapter board connections.

7.-COMMUNICATION SOFTWARE.

7.1.-VME and microVAX programs.

A simple communication software has been written to perform DMA transfers.

The parameter needed to specify the transfer are transmitted by the requester in single word mode using the interrupt capabilities of the interface.

The VME routines are written in assembler language, while the microVAX routines are in FORTRAN using SYS$QIO calls to a standard Digital driver.

To synchronize the two programs we have used the FUNCTi lines.

The FUNCT_0 line is used in both directions as asynchronous interrupt, while the three FUNCT_1-3 lines (read by the microVAX as bits STATi of the CSR register) are interpreted as follows:

--- F1 as READY line, to signal that the receiver is ready to the next transfer
--- F2 as ERROR line, to indicate an error in the previous transfer
--- F3 as STOP line, to indicate no further transfer.

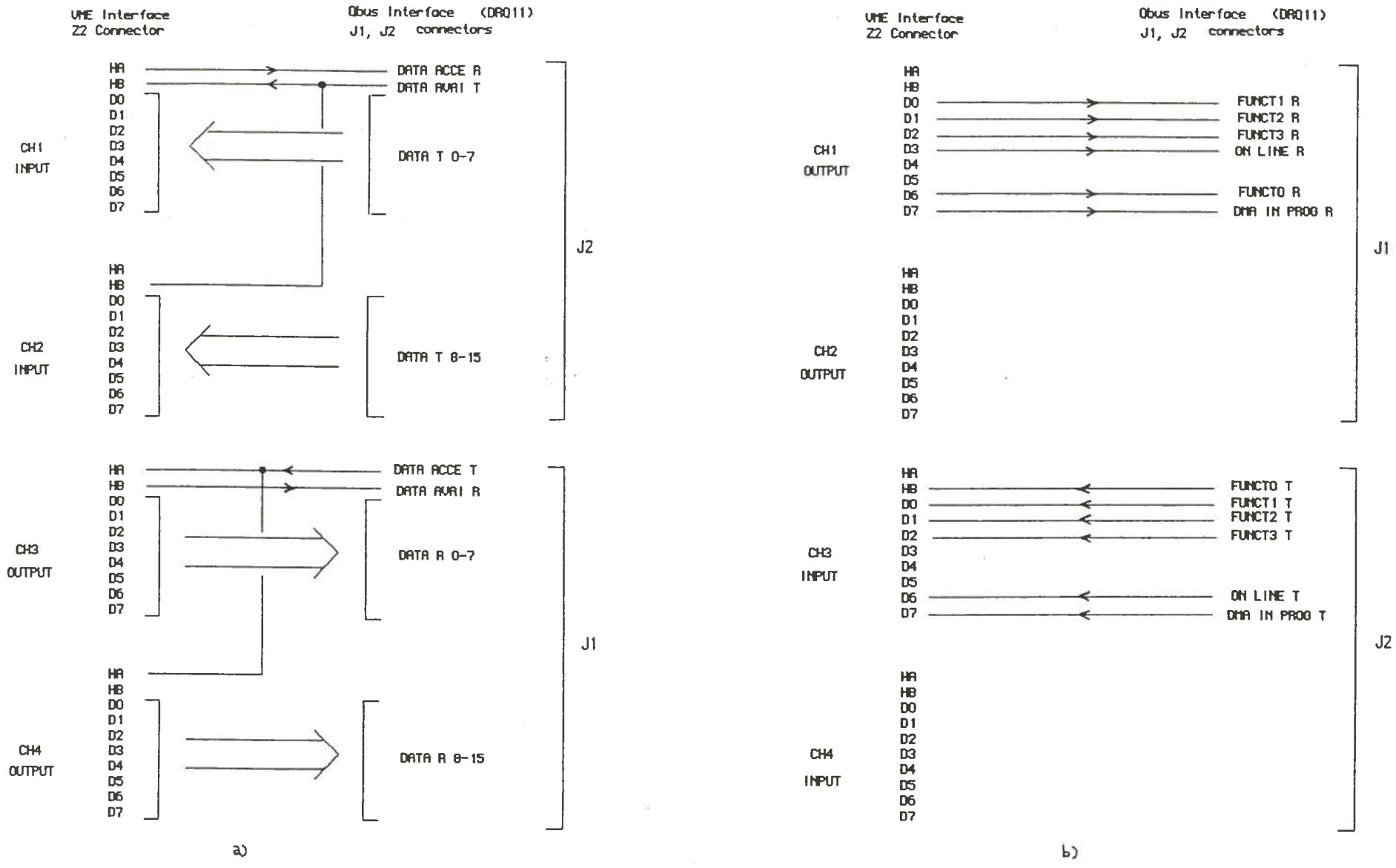


Fig. 3

VME Interface - DRQ11 connections

7.2.-DMA input transfer.

We describe briefly the DMA input transfer protocol between microVAX and VME (from now on the microVAX is considered master although the communication protocol is perfectly symmetrical).

At the beginning of the transfer the microVAX program enables interrupts and asserts the READY line to notify the start of operations sequence, then it waits for interrupt (fig.4).

Once the VME program verifies the READY line, it writes the number of bytes to transfer on the output lines and then asserts the FUNCT_0_R line.

The interrupt service routine on the microVAX reads the Data Buffer Register (DBR) to know the byte number, then it acknowledges the VME with the FUNCT_0_T line and asks a DMA input transfer to the System Service.

When the interrupt is received by the VME, the DMA controller is programmed and the DMA output transfer starts.

When the transfer is completed (the DMAC sends an interrupt to the VME CPU when its counter register, decremented during the transfer, is zeroed) the FUNCTi lines are

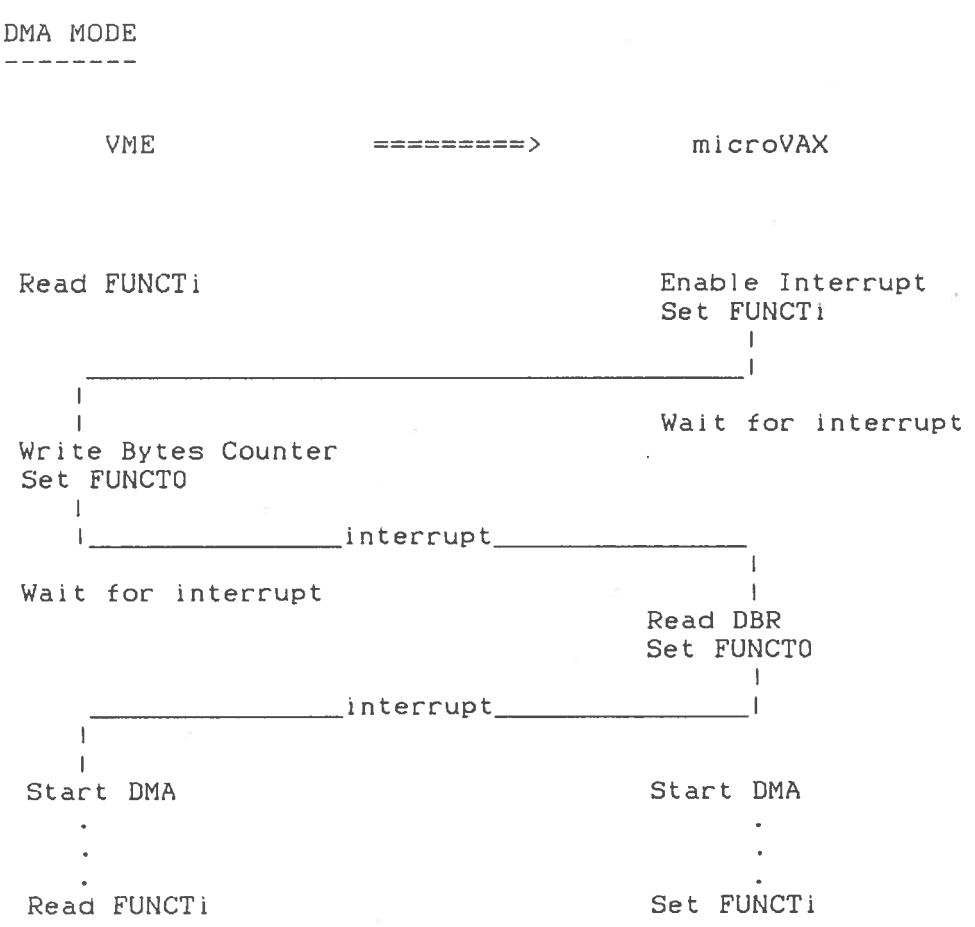


Fig.4

DMA input transfer schematic flowchart

monitored by the VME; these lines are asserted by the microVAX according to the driver status words indicating the operation's completion.

At this time a new transfer cycle can begin.

7.3.-DMA output transfer.

The instructions sequence in DMA output transfer (from microVAX to VME, see fig.5.) is similar to the input transfer.

In this case, at the beginning of the transfer the microVAX program reads the device status; after it writes the byte number on the DBR register and then it interrupts the VME program. Once the VME has read the input lines, FUNCT_0_R line is asserted to signal to the microVAX that the DMAC is ready for the input. After this the microVAX outputs the block of data in DMA mode and reads STATi bits to control the device status.

At the end of the transfer DMA_IN_PROG signal is checked by the VME and when this signal goes off the VME sets FUNCTi lines to acknowledge the transfer completion to the microVAX.

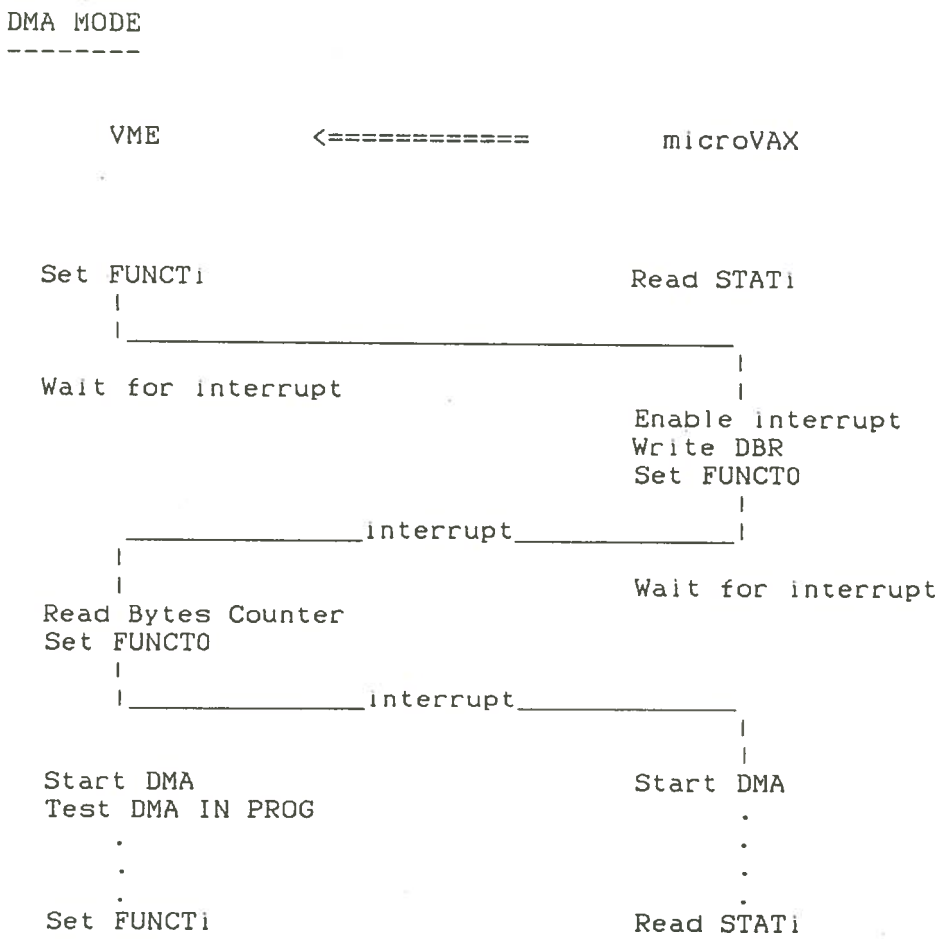


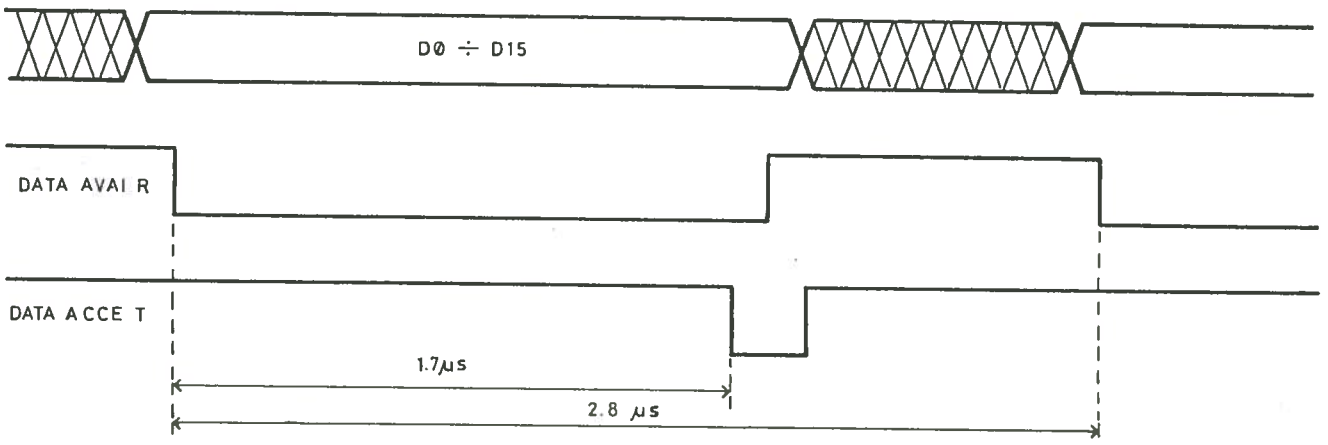
Fig.5

DMA output transfer schematic flowchart

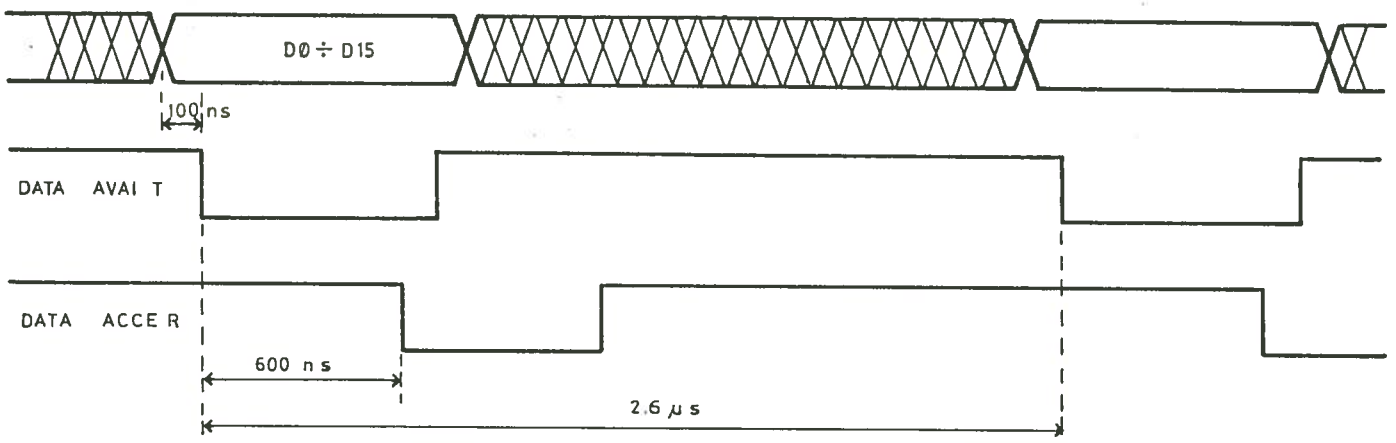
8.-RESULTS.

The routines realizing the sequences described above have been used in a test program to check the system performances.

In fig.6. are shown the timing diagrams for DMA cycles obtained with the system described in par.2. with cables of about 2 meters.



a) VME ⟶ µVAX



b) µVAX ⟶ VME

FIG.6

Timing diagrams for DMA cycles

In a DMA input transfer, DATA_ACCE_T is asserted by the microVAX 1.7 microsec after the data are stable and DATA_AVAI_R is sent out from the VME. The cycle period is 2.8 microsec.

In a DMA output transfer the VME acknowledges the data by sending back DATA_ACCE_R 600 nsec after assertion of DATA_AVAI_T. The cycle period is 2.6 microsec.

Therefore we have a transfer rate of 0.7 Mbytes/sec, that is the maximum hardware rate.

If we consider the time spent in the communication protocol (described in the par.7.) the effective rate obviously decreases. For example to transfer a block of 64 Kbytes our program requires about 105 msec of which 90 msec are spent for the DMA cycles.

Furthermore we did some simple checks to control the system reliability.

A predefined block of data was sent from one computer to the other and verified by the receiver. Moreover random blocks has been sent from the microVAX, sent back by the VME and checked by the microVAX for matching.

After several hours of run (more than 10**7 Kbytes transferred) we have not observed any mistake in the data.

9.-CONCLUSION.

We have realized a straightforward link between a microVAX II and a VME crate using two commercial interfaces with DMA capabilities.

We have obtained a reliable block transfer at an effective rate higher than 0.5 Mbytes/sec.