To be published in
IEEE Trans. Nucl. Sci.

BNL# 21884

Presented at IEEE Nuclear Science
Symposium, 20-22 October 1976,
Braniff Place Hotel, New Orleans.

CONF-761006--19

A SHARED RANDOM ACCESS MEMORY RESOURCE FOR MULTIPROCESSOR REAL-TIME SYSTEMS*

D. G. Dimmler and W. H. Hardy, II

Brookhaven National Laboratory
Upton, New York 11973

October 1976

*Research carried out under the auspices of Energy Research and Development Administration: Contract No. EY-76-C-02-0016

DISCLAIMER

A SHARED RANDOM ACCESS MEMORY RESOURCE FOR MULTIPROCESSOR REAL-TIME SYSTEMS*

D. G. Dimmler and W. H. Hardy, II

Brookhaven National Laboratory
Upton, New York 11973

Abstract

A shared random-access memory resource is described which is used within real-time data acquisition and control systems with multiprocessor and multibus organizations. Hardware and software aspects are discussed in a specific example where interconnections are done via a UNIBUS. The general applicability of the approach is also discussed.

Background

The recent advent of inexpensive computer processors and the drastic decrease in price of core and semiconductor random access memory make modern architectures for data acquisition and control systems feasible which avoid many of the limitations imposed for economic and other reasons on older structures. Instead of using one computer processor for many different functions as in traditional time sharing concepts, new architectures, such as the distributed function architecture, described elsewhere,[1-9] have the tendency to dedicate a processor and memory to one or a small set of compatible functions. These processor/memory combinations comprise functional nodes. A complete system consists of a network of interconnected functional nodes.

Such an architecture leads to multibus, multinode configurations, as shown in an example in Fig. 1, and as conceptually described elsewhere. A master control node including a PDP11/40 processor controls via an experiment control node a neutron spectrometer including a position sensitive detector.[10-12] The data collected from the detector is processed by the data input node, a special purpose direct memory increment processor. This processor inserts the data into a random access memory, here called shared memory resource, of a potential size of several million words. A display node provides an on-line display function on a cathode ray tube using the data in process of being collected as a source.

Shared Memory Resource

A significant element of such an architecture is a random access memory resource where large common data arrays of up to several million words and common control parameters are stored and can be accessed by several functional nodes. Such a random access memory resource includes:

a. a multiport memory control (MPMC) which provides access from several processors to the memory segments of the shared memory resource;

b. one or several memory segments; and

c. the allocation and access software, which operates at each of the accessing nodes.

Multiport Memory Control

As Fig. 2 shows, the multiport memory control includes:

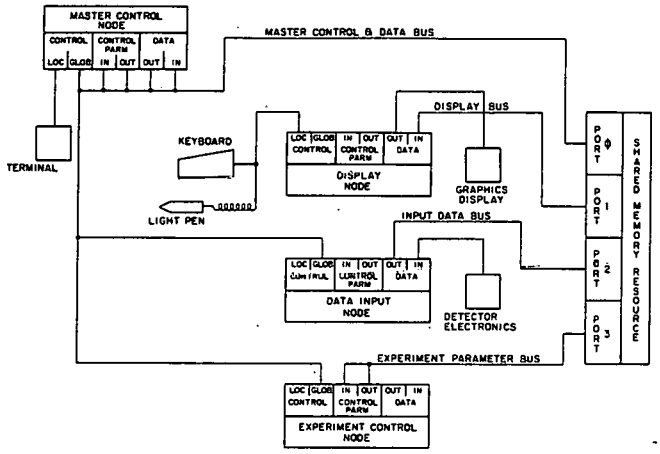


Figure 1

Application Node
Typical Configuration

a. a set of access ports where one functional node is connected to each port;

b. an internal tri-state bus with a bus arbitrator; and

c. a set of memory ports where one memory segment is connected to each port.

Access Ports

Up to 16 functional nodes can be connected to the shared memory resource via up to 16 access ports. An access port communicates with a connected processor either via a standard UNIBUS† or via an extended UNIBUS. An extended UNIBUS resembles two physical bus cables where the additional cable carries the high order address bits A18 to A29.

One of the possible processors connected to an access port is a PDP11. Such a processor has a logical address space of 28K words (1K = 1024). This logical address space is subdivided into regions depending on the function performed by the processor. An example, as described elsewhere,[13] is shown in Fig. 3. A region of 4K words between the addresses 140000_8 and 160000_8 is assigned as a window through which the processor accesses the shared memory resource. This logical address space is mapped as shown in Fig. 3 into a processor physical address space by the memory management of the processor. If such a management is missing

†UNIBUS is a registered trademark of Digital Equipment Corporation.

*This work carried out under the auspices of the Energy Research and Development Administration: Contract No. EY-76-C-02-0016.

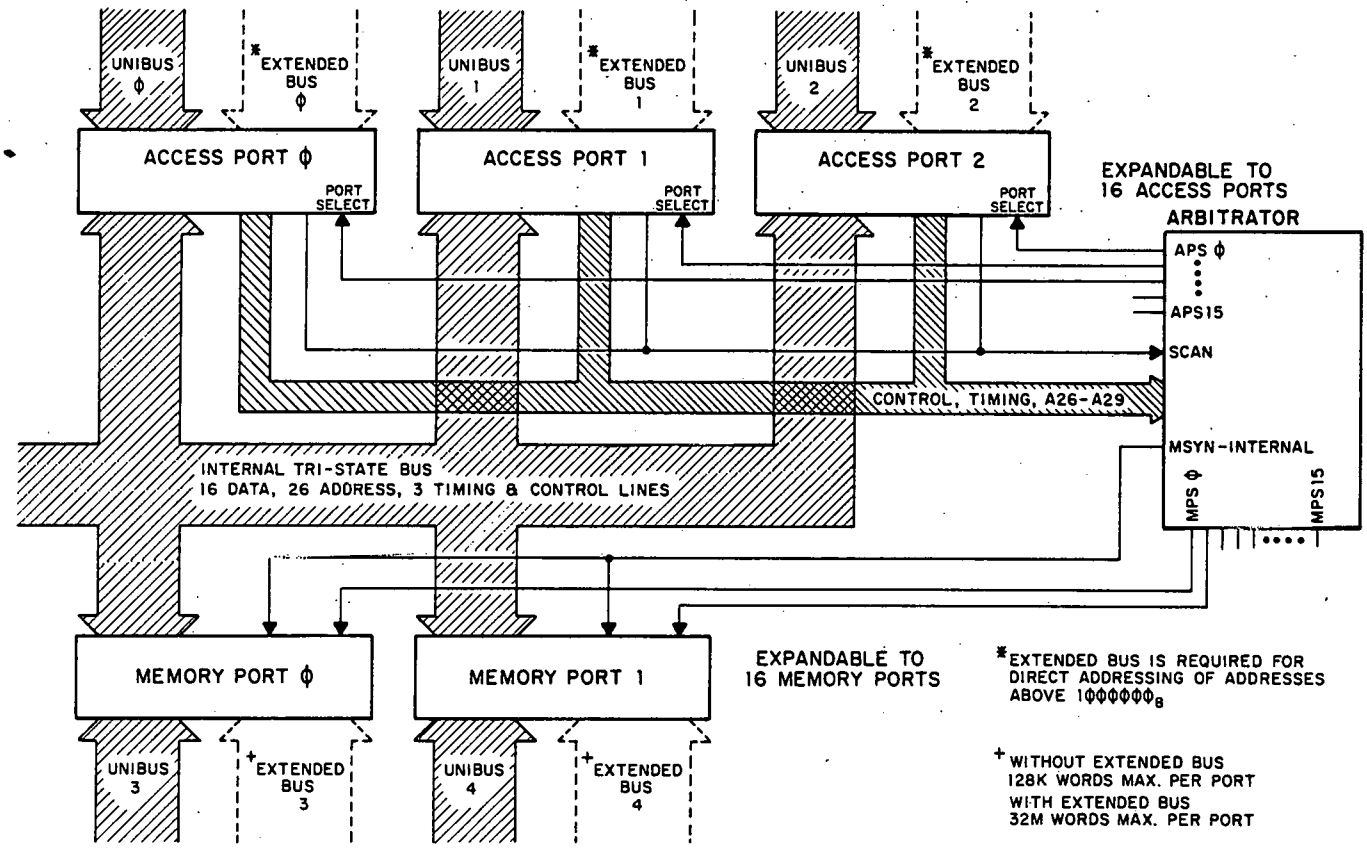
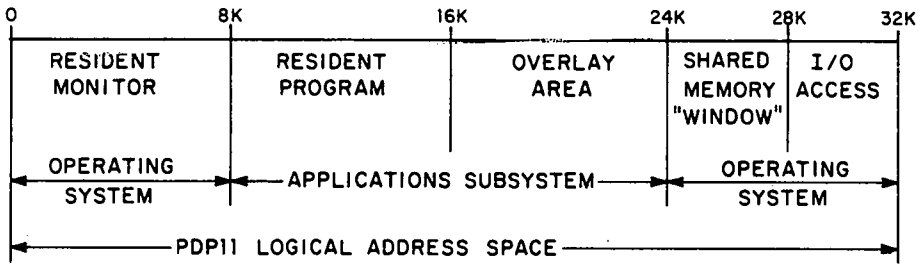


Figure 2 ↑

Multi-port Memory Controller
Typical Configuration



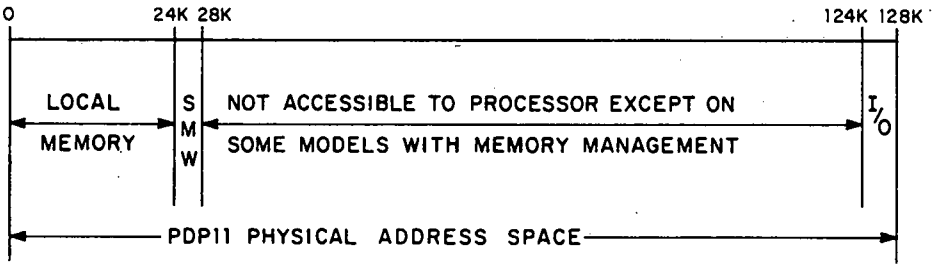
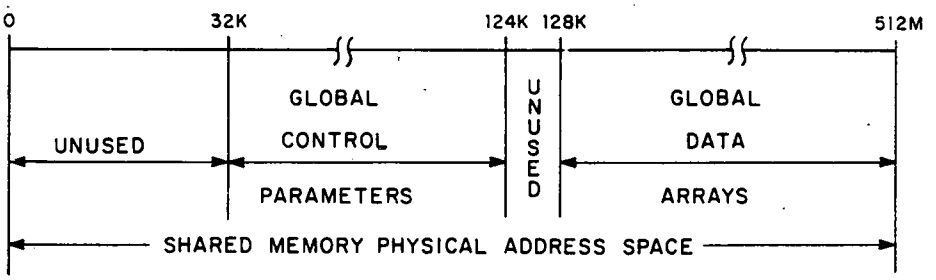


Figure 3 →

Shared Memory System
Address Allocations



as is the case of a small PDP11 processor, then the processor physical address space between 28K and 124K is not accessible and the previous logical and physical address spaces are synonymous. The window region, then, is mapped into the shared memory physical address space by the memory management located in the corresponding access port as shown in Fig. 4. The physical address in the example is composed of the bits 2^0 to 2^{12} provided by the corresponding bits of the UNIBUS and by the bits 2^{13} to 2^{29} provided by the previously filled map address register. The mapping algorithm is access-port-specific. Other ports may be set to different algorithms. If an extended UNIBUS is connected then the content of the mapping address register does not contribute to the physical address. The entire physical address is provided at the input of the access port. This situation often arises when a special purpose node sorts experimental data in real time into large arrays. In the application example given previously the direct memory increment node sorts data collected from the position sensitive detector into the shared memory resource via an extended UNIBUS.

One access port is designated a master access port. All others are considered slave access ports. The processor connected to the master access port may temporarily lock out access to all slave ports. This feature is necessary for memory allocation purposes as explained in a subsequent paragraph.

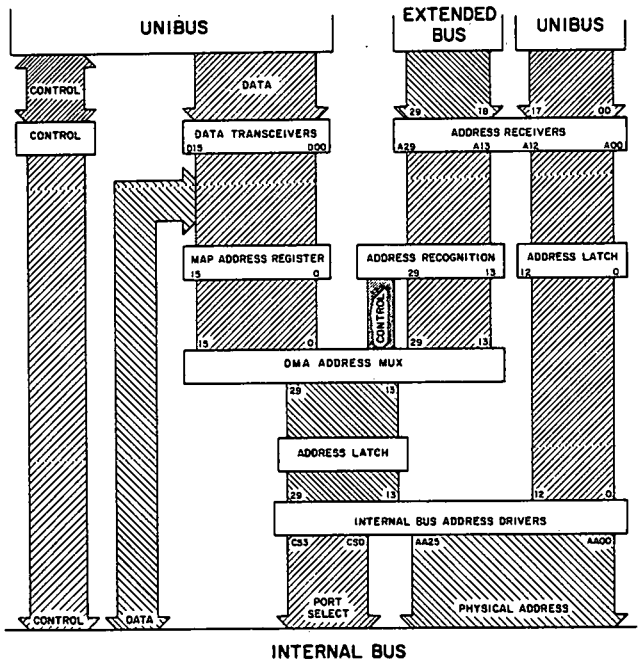


Figure 4

Multi-port Memory
Control Access Port

Five access-port-specific functions and address bit arrangements can be selected by manual switches and jumpers. They are:

a. the port can be protected against a WRITE access;

b. the high order address bits can be provided either by the map address register or by the UNIBUS extension as described previously;

c. the start of the logical address space of the access window can be selected;

d. the length of the above mentioned address space can be selected; and

e. the addresses of the address port device registers can be selected.

An access port contains four device registers. They are:

Register 0: Map Address Register. This register contains the high order bits A13 to A28 for mapped access as described above;

Register 1: Low Physical Address Register. This register contains the low order bits 2^0 to 2^{15} of the last physical address accessed;

Register 2: High Physical Address Register. This register contains the high order bits 2^{16} to 2^{29} of the last physical address accessed;

Register 3: Status Register. This register contains status information as shown in Table 1.

Table 1

Status Register Contents

BIT	
Ø	CØØ UNIBUS state from last memory access.
1	CØ1 UNIBUS state from last memory access.
3	Port locked against access at time of last access attempt (functions on slave ports only).
4	Port is currently locked against access (functions on slave ports only)
5	Slave port access lock out (functional on master port only).
9-11	UNIBUS address bits A13-A15 of last memory access.
14	R/W access indicator
15	No response from memory on last access (invalid address or memory failure).

The access port is designed on one printed circuit module occupying six segments of a standard PDP11 connector block. The contents of registers 1, 2 and 3, as well as internal hardware states, are displayed on light emitting diodes along the top of the module. This allows for simplified diagnosis of errors and for easy observation of system performance.

Memory Ports

Up to 16 commercial UNIBUS compatible memory segments can be connected via the 16 memory ports. One port is assigned for one memory segment. A memory segment communicates with the port either via a standard UNIBUS which restricts the size of the segment to 128K words or it communicates via an extended UNIBUS as described above. In the latter arrangement

the size restriction does not exist; however, a slight modification of the address decoding used in commercially delivered segments is usually necessary.

Internal Bus

Transfers between the access ports and the memory ports are accomplished over an internal tri-state bus under the control of a bus arbitrator. The arbitrator uses a round-robin algorithm for the scheduling of concurrent access requests. The arbitrator scans the ports with a frequency of 5 MHz, hence in a four port system, a given port is polled every 800 ns. If a polled port has a request pending, scanning is halted and the port is allowed one memory access*. Afterwards scanning is resumed. This algorithm was chosen over a port priority system to insure that a port cannot monopolize the shared memory resource. The arbitrator has been constructed as an independent printed circuit module. Other algorithms would require the design of a different arbitrator module.

The total instantaneous memory cycle of a processor to a memory location in the shared memory resource depends on:

a. the number of connected ports;

b. the number of concurrent accesses pending;

c. the type of the access;

d. the speed of the memory used; and

e. the logic delay of approximately 150 ns in the memory port controller.

In the example of a four access port system with a memory of 900 ns read/write time the total cycle time of a read (DATI) instruction is in the best case 900 ns read/write time plus 150 ns logic delay of the controller totalling 1050 ns. The worst case is the situation where a real/modify/write (DATIP) access request arrives just after the polling time and all other ports have pending DATIP requests. The cycle time, then, extends to 900 x 8 + 150 x 4 = 7800 ns.

Shared Memory Allocation and Utilization

The shared memory physical address space is divided into two sections as shown in Fig. 2. The global control parameter section (GCPS) of the memory contains tables which must be accessed by more than one processor, such as the memory management master directories. The GCPS has a minimum size of 4K words dictated by the size of the shared memory directories. In small systems the GCPS may also contain device buffers. In large systems with processor specific memory management these buffers reside in each processor's physical address space in the area between 28K and 120K, and the shared memory "window" is moved to 120K - 124K in the processor physical address space. Allocation of the GCPS is done at the time of system generation. The global data array area is allocated under the control of the node connected to the master access port. This node becomes the master control node of the multiprocessor system. One master control node must be identified within the system. The master control node defines the type of access the other nodes will have to the arrays in the data area. The access definition is array specific. During the deallocation process access from all slave nodes is locked out by hardware. After the deallocation process is complete all ports may utilize the data area within the access description contained in the array allocation directories as outlined in detail below.

The shared memory space of 2^{29} words is divided into allocation units of 4096 words each. Allocation takes place in multiples of allocation units. For the purposes of allocation simplicity the following rules apply:

a. All arrays are assumed to be three dimensional;

b. The linear address is calculated using the following algorithm:

$$A = ((i-1) + I * (j-1) + I * J * (k-1)) * N$$

where

I is the X dimension

J is the Y dimension

K is the Z dimension

N = 1 for byte data
= 2 for word data
= 4 for double word data

i, j, k are the indices of the datum being addressed;

c. The maximum dimensions are defined by:

$I \leq 32767$

$J \leq 32767$

$K * I \leq 32767$

d. Allocation can only take place via the designated master port. All other ports are designated as slave ports.

e. The deallocation function may pack the global data array region of the shared memory in order to avoid scattering. Thus, during the deallocation process all ports except the master port are denied access.

f. All directory information is contained in the directory page located in the global control parameter area.

g. An array is allocated in a contiguous physical space.

Array Allocation Directory Organization

An array allocation directory is maintained within the data memory. The directory has a descending tree structure consisting of:

a. a master memory definition table (MMDT);

b. a logical array allocation table (LAAT);

c. an array allocation directory (AAD) consisting of allocation control blocks (ACB)s. Space for one ACB exists for each potential array; and

d. an array allocation unit map (AAUM) containing a one byte descriptor for each allocation unit.

Master Memory Definition Table (MMDT)

The MMDT contains the basic definitions for the shared memory resource software. It begins at physical address 200000 of the GCPS. The contents of the MMDT are described below:

*If the access is of a DATIP (read-modify-write) type, two accesses are allowed: one read access and one write access.

LABEL	INITIAL VALUES	DESCRIPTION
AALART	140100	Address of beginning of LAAT
ACBSTR	140200	Address of beginning of AAD
AAMWMP	144000	Address of beginning of AAUM
FIRALU	40	First available allocation unit (ALU)
CURALU	*	Current ALU
CURACB	*	Current ACB
NALLRT	40	Number of arrays allowed
MSTFSP	*	temporary variables
MSTDAA	*	" "
MSTCAA	*	" "

Logical Array Allocation Table (LAAT)

The LAAT links the logical array identifier used at the FORTRAN level to an allocation control block ACB which defines the physical attributes of the array.

Array Allocation Directory (AAD)

Each allocated array has an allocation control block (ACB) associated with it. The contents of an ACB are described below:

LABEL	TYPE	
ARYALF	BYTE	Allocation Flag Ø available for allocation 1 failing or otherwise not usable 2 nonexistent 200_8 allocated
ARYNUM	BYTE	FORTRAN logical array identifier $0 \leq N \leq 31_{10}$
DATTYP	WORD	Data type Ø byte 1 word 2 double word
EMXDIM	WORD	X Dimension (I) $Ø \leq I \leq 32767$
EMYDIM	WORD	Y Dimension (J) $Ø \leq J \leq 32767$
EMSDIM	WORD	Z Dimension (K) $Ø \leq I * K \leq 32767$
EMRDAC	WORD	Read access control The bits Ø-15 correspond to ports Ø-15 respectively. A bit = Ø implies no read access. A bit = 1 implies read access.
EMWTAC	WORD	Write Access control The bits Ø-15 correspond to ports Ø-15 respectively. A bit = Ø implies no write access. A bit = 1 implies read access.
ARYSTR	WORD	First allocation unit. The ALU number of the first ALU allocated for this array.
ARYEND	WORD	Last allocation unit. The ALU number of the last ALU allocated for this array.

Array Allocation Unit Map (AAUM)

The AAUM[14] as presently implemented contains 6144 bytes, one byte for each ALU. The byte contains status information for each allocation unit:

a. Ø available for allocation;

b. 1 failing or otherwise not usable;

c. 2 nonexistent;

d. 200_8 allocated.

Fortran Level Access

Two sets of FORTRAN routines are provided for shared memory management. The first set consists of allocation routines which are used by the master processor to allocate the memory. The second set consists of access routines which can be used by any processor to access the shared memory resource.

Memory Allocation Routines

All memory allocation routines make use of a common block MEMMG1. The form of this common block is:

```
COMMON/MEMMG1/IAL,IDT,IX,IY,IZ,IRD,IWT,IFPG,ILPG,ACBAD
```

A one-to-one correspondence exists between the contents of an ACB and common block MEMMG1 except for the last entry ACBAD. ACBAD contains the logical address of a specified ACB. The variable IS is a return parameter containing error information in all shared memory resource routines. The allocation routines are described below in detail.

MLOCKR(IS)	locks out all slave ports from access to the shared memory resource.
MINITR(MSIZE,IS)	initializes all tables and directories. All currently allocated arrays will be deallocated. MSIZE is the number of allocation units available.
MALOCR(IARRAY,IS)	allocates the array designated by IARRAY. IDT, IX, IY, IZ, IRD and IWT must be specified in the common block MEMMG1.
MDALCR(IARRAY,IS)	unconditionally deallocates the array IARRAY. This routine may pack the global data array region of the shared memory.
LARSTS(IARRAY,IS)	sets the contents of the common block MEMMG1 to correspond to the contents of the allocation control block specified by IARRAY.
MUNLKR(IS)	unlocks the memory for access by slave ports.
MSETBD(INUM,IS)	can be used to indicate to the memory management system that a given ALU is failing or otherwise unusable. The specified ALU should not be allocated at the time of the call at MSETBD. INUM is the ALU number of the malfunctioning memory.
ARYCLR(IARRAY,IS)	sets the content of the array specified by IARRAY to zero.

Memory Access Routines

The memory access routines provide access to the shared memory resource. The access and storage for each data type are:

STORAGE TYPE	ACCESS ROUTINE	VALUE RETURNED AS	STORAGE ROUTINE	ARGUMENT TYPE
BYTE	EMRDBY	BYTE	EMWTBY	BYTE
INTEGER	EMRDSI	INTEGER	EMWTSI	INTEGER
32 BIT INTEGER	EMRDDI	REAL*	EMWTDI	REAL*
REAL	EMRDSR	REAL	EMWTSR	REAL

*Double precision integers were implemented due to a need by the experiments being controlled. Since the FORTRAN system being used does not support double precision integers conversion to a real number for use of the FORTRAN level is necessary.

All access routines are FORTRAN function subroutines. As such they may be used essentially the same as array variables. For example:

```
BYTE = EMRDBY(IARRAY,I,J,K,IS)
  IX = EMRDSI(IARRAY,I,J,K,IS)
  XI = EMRDDI(IARRAY,I,J,K,IS)
  XI = EMRDSR(IARRAY,I,J,K,IS)
```

All storage routines are FORTRAN subroutines of the form:

```
CALL EMWTBY(IARRAY,I,J,K,BYTE,IS)
CALL EMWTSI(IARRAY,I,J,K,IX,IS)
CALL EMWTDI(IARRAY,I,J,K,XI,IS)
CALL EMWTSR(IARRAY,I,J,K,XI,IS)
```

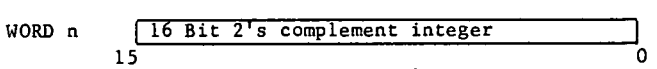
In all of the above the variables have the following usage:

IARRAY	FORTRAN logical array identifier
I	X subscript
J	Y subscript
K	Z subscript
IS	error return
BYTE	a byte variable
IX	an integer variable
XI	a floating point variable

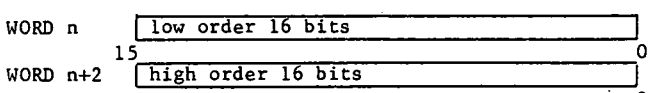
Shared memory data storage Formats

The data storage formats conform where possible to the formats in the FORTRAN system being used. They are:

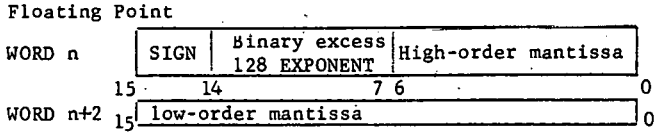
Single Precision Integer (16 bit 2's complement)



Double Precision Integer (32 bit 2's complement)



Floating Point



Array Storage Algorithm

Arrays are stored in physical memory with the first subscript varying most rapidly. For example, an array dimensioned (3,3,3) would be stored as:

1,1,1	Lowest memory location
2,1,1	
3,1,1	
1,2,1	
2,2,1	
3,2,1	
1,3,1	
2,3,1	
3,3,1	
1,1,2	
2,1,2	
.	
.	
.	
3,3,3	Highest memory location

Error Codes

As mentioned before, every shared memory management routine has a return parameter IS, where information is recorded by the routine. The content of IS means:

OCTAL

0	Request successfully completed;
200	Invalid array number;
201	Array is already allocated;
202	IX * IZ too large;
203	System type not valid;
204	Insufficient room to allocate array;
205	Array is not allocated;
206	Invalid byte in byte map on pack operation;
207	Invalid or unallocated array on access request;
210	This port does not have the desired type of access to the specified array;
211	Dimension exceeds allocation dimension.

Mechanical Implementation

The memory port controller is mechanically constructed with hardware compatible in dimensions, connectors and layout with the PDP11 system of the Digital Equipment Corporation. The access port is implemented on a printed circuit module occupying 6 segments and the memory port card occupies 4 segments of a standard 9 slot PDP11 connector block. The arbitrator uses 2 segments. The other segments are used for appropriate connections to the various UNIBUS's. Figure 5 shows a photograph of an implemented memory port controller.

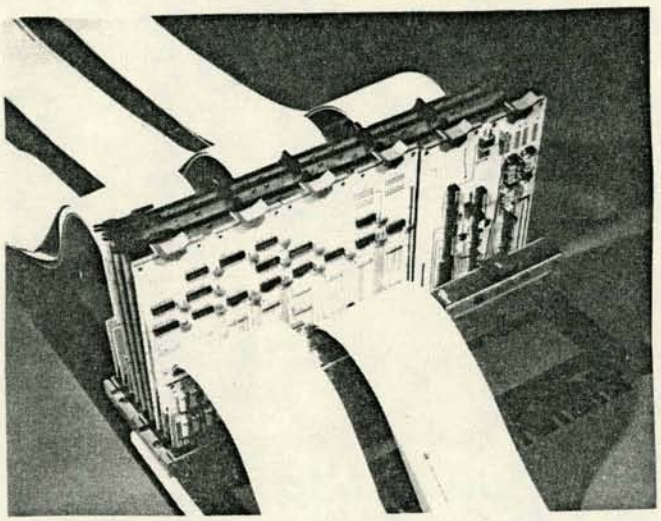


Figure 5

Multi-port Memory Controller
A current implementation

References

1. D. G. Dimmler. Functional Distribution - An Architecture for Multi-User Computer Networks in Instrumentation. IEEE Trans. Nucl. Sci. NS-21 838 (Feb. 1974).
2. F. W. Stubblefield and D. G. Dimmler. Transaction Processing in the Common Node of a Distributed Function Laboratory Computer System. IEEE Trans. Nucl. Sci. NS-22, 473 (Feb. 1975).
3. F. W. Stubblefield and D. G. Dimmler. A Task Scheduler and Service Subsystem for the Common Node of a Distributed Laboratory Computer Network. IEEE Trans. Nucl. Sci. NS-23, 413 (Feb. 1976).
4. F. W. Stubblefield. Continuous Sharing of a Record-Oriented Output Device Within a Distributed Function Laboratory Computer Network. IEEE Trans. Nucl. Sci., NS-23, 423 (Feb. 1976).
5. F. W. Stubblefield. Logical and Physical Resource Management in the Common Node of a Distributed Function Laboratory Computer Network. IEEE Trans. Nucl. Sci., NS-23, 406 (Feb. 1976).
6. F. W. Stubblefield. A File Management of Experiment Control Parameters within a Distributed Function Computer Network. Being presented at IEEE Nucl. Sci. Symp., 19-22 Oct. 1976.
7. D. G. Dimmler, N. Greenlaw, M. A. Kelley, D. W. Potter, S. Rankowitz and F. W. Stubblefield. The Brookhaven Reactor Experiment Control Facility - A Distributed Function Computer Network -. IEEE Trans. Nucl. Sci. NS-23, 398 (Feb. 1976).
8. D. G. Dimmler, D. W. Huszagh, S. Rankowitz and J. Scott. A Controllable Real-Time Data Collection System for Coastal Oceanography. Proc. of OCEAN '76 Conf., Sept. 1976, Washington, D. C.
9. D. G. Dimmler. The ISABELLE Control and Monitoring System Global Overview. Internal Report BNL-CRISP 75-5, April 1975.
10. J. L. Alberi, J. Fischer, V. Radeka, L. C. Rogers, and B. Schoenborn. A Two-Dimensional Position-Sensitive Detector for Thermal Neutrons. BNL 19487. Nucl. Instr. and Meth. 127, 507 (1975).
11. J. L. Alberi andV. Radeka. Position Sensing by Charge Division. BNL 20720. IEEE Trans. Nucl. Sci., Vol. NS-23, 251 (Feb. 1976)
12. J. L. Alberi. Position Readout by Charge Division in Large Two-Dimensional Detectors. Being presented at IEEE Nucl. Sci. Symp., 19-22 Oct., 1976.
13. D. G. Dimmler. Network Specific Software Stuctures In a Private Node of a Distributed Function Laboratory Computer Network (in preparation).
14. D. G. Dimmler. Architecture of a Maintenance Subsystem in a Multi-User Computer Installation. Proc. of Workshop on Fault Detection and Diagnosis in Digital Circuits and Systems, Lehigh University, Allentown, Dec. 1970.