# MODULAR MICROCOMPUTERS FOR TRAINERS

Edward M. Holler

Naval Training Equipment Center
Orlando, FL 32813

## ABSTRACT

There is a critical need for standard lower cost computer systems in Navy trainers. The proliferation of computer hardware and software has kept procurement, maintenance, update, and training costs high on most Navy systems. Microcomputer and computer graphics technology offers an exceptional opportunity to improve the capabilities and reduce the costs of trainers requiring computer systems. However, the benefits of this technology cannot be fully realized unless the proliferation problem is solved. This paper discusses an approach to microcomputer design and development that will take full advantage of microcomputer technology and reduce the computer proliferation problem. The basic premise is as follows: a standard general purpose microcomputer architecture combined with high level language application software can substantially reduce costs of all new trainers. The Navy is currently examining microcomputer architectures and development approaches that can be optimized for a wide range of trainer applications. Applications include flight simulators, Anti-Submarine Warfare (ASW) trainers, and warfare operations trainers, etc. This paper reports on a research project at the Naval Training Equipment Center (NAVTRAEQUIPCEN) which is currently investigating the use of microcomputer technology in Navy training systems. The following topics will be discussed (1) microcomputer architecture; (2) control of multiple microcomputers; (3) software development facility and approach; (4) test and evaluation plan; and (5) conclusions.

## INTRODUCTION

Most Navy trainers utilize mainframe and super minicomputer systems to compute trainer algorithms. Figure 1 illustrates the computer system architecture for the real-time flight simulator located in the Visual Technology Research Simulator (VTRS) at NAVTRAEQUIPCEN. This is just one example of the many computer architectures used in Navy trainers. In fact, there are over 793 computer systems in the Navy inventory using 63 different hardware types and 43 different programming languages. This proliferation of computer architectures and software continues to make procurement, maintenance, and update costs for trainers requiring computer systems extremely high. However, advances in microcomputer technology now offer other alternatives.

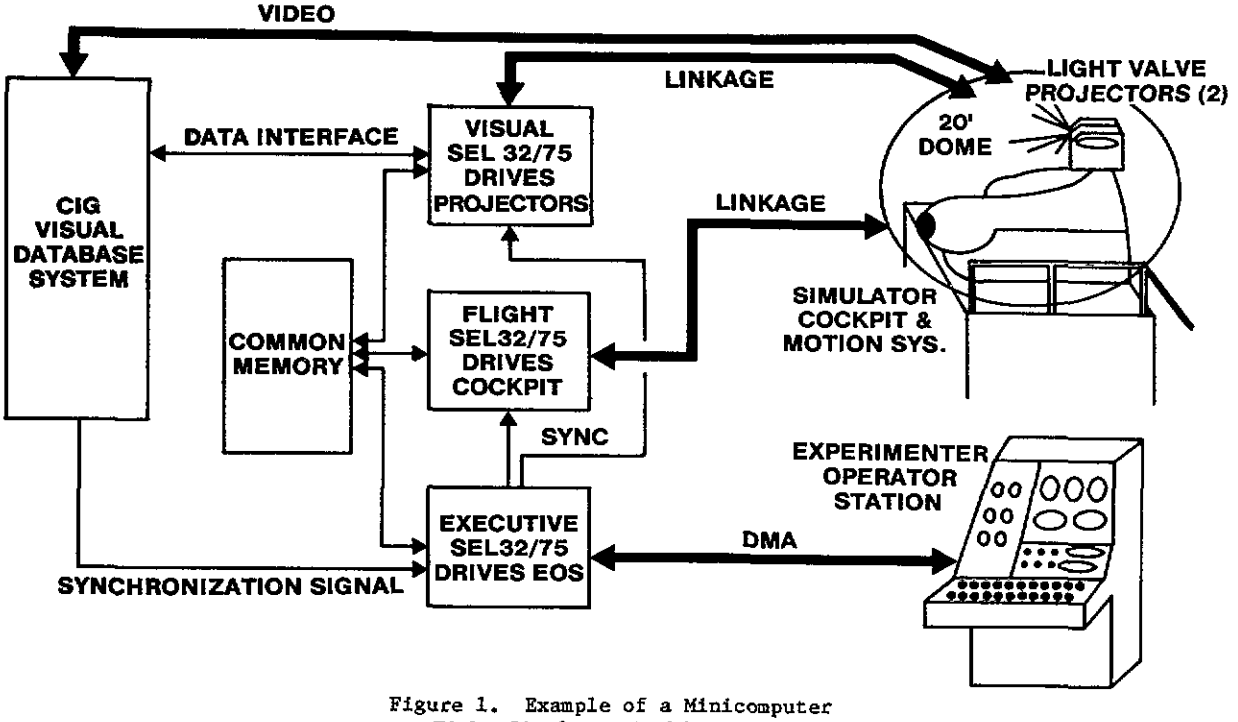


Figure 1. Example of a Minicomputer Fight Simulator Architecture. (NTEC-VTRS SYSTEM)

The reduced costs and increased performance of microcomputer technology has made microcomputers very attractive for trainer applications. It is known that simulator vendors are planning to make extensive use of micrcomputers in the near future. However, industry is not interested or motivated to standardize the application of microcomputer technology in trainers. If no action is taken by the Navy, microcomputers will contribute further to computer proliferation. This paper provides a solution to this problem through a standard microcomputer system architecture that can easily be adapted to Navy trainers. Three outstanding technical problems must be solved before multiple microcomputers can be used effectively: (1) the control of multiple microcomputers; (2) the partitioning of the simulator software into smaller tasks for each applicaton microcomputer and (3) bus contention. This paper will address these issues.

MICROCOMPUTER ARCHITECTURE

Figure 2 shows a multiple microcomputer system that is appropriate for a wide range of trainer applications. This system architecture uses a set of identical microcomputers. It is *the intent of this project to:* (1) evaluate this design and develop standards for the use of microcomputers in training systems; (2) determine microcomputer parameter requirements that will support a multiple microcomputer system architecture and high level languages; (3) input these results into trainer computer specifications that will be used to buy future trainers; (4) eliminate unsuitable microcomputers through these new definite performance specifications (proliferation of microcomputers will be limited due to the process of natural selection); and (5) develop general purpose automated software development tools that can be used to reduce software development costs for multiple microcomputer systems.

The multiple microcomputer system in Figure 2 utilizes high performance microcomputers (Motorola 68000s) to perform functionally modular trainer tasks in parallel. Each of these microcomputers are connected to a high speed 20 megabytes per second bus (VMEBUS 16 bit parallel data path) to allow fast data transfers and the quick interprocessor communication required for real-time trainers. Each microcomputer also has its own dedicated I/O bus which directly connects to the various trainer functions (control loading, etc.). This provides a direct path from each trainer function to its respective microcomputer which allows true parallel processing of trainer tasks. The microcomputer on the left side of Figure 2 is the control microcomputer. It controls the operation of the multiple microcomputer system including trainer initialization, startup tasks; instructor station inputs and requests; control of the application microcomputers, start, stop, bus arbitration; and disk I/O.

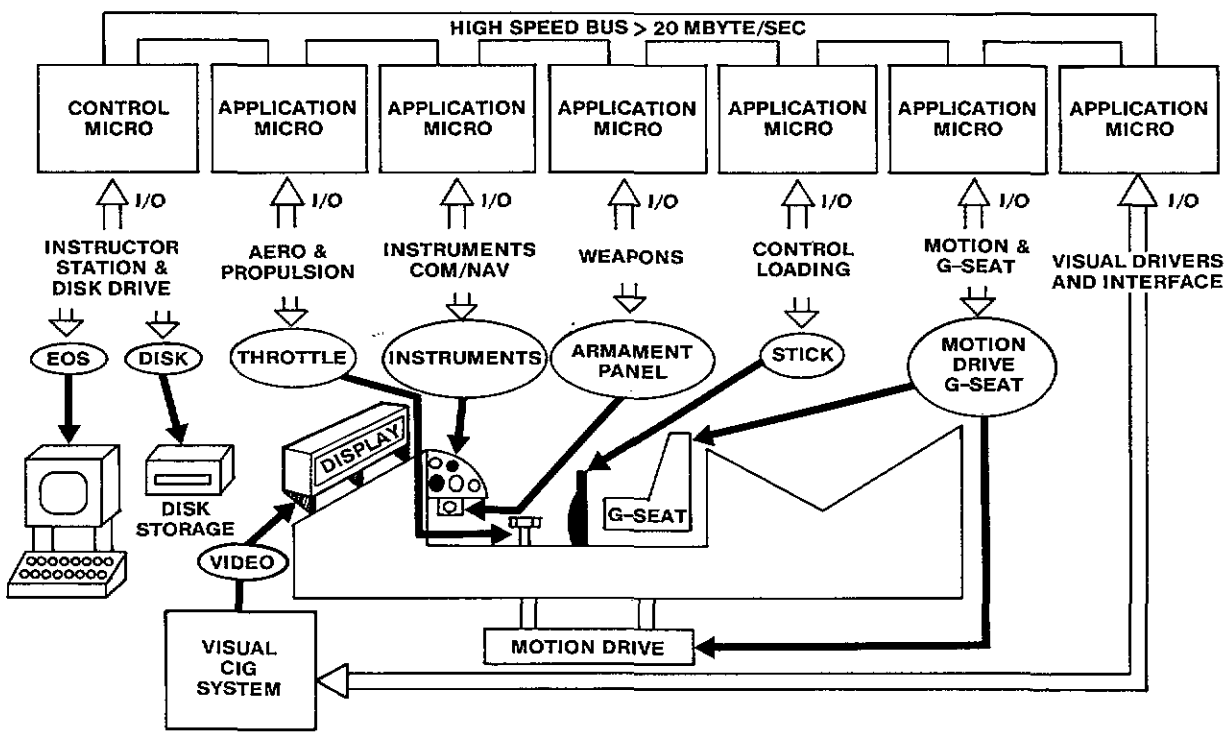


Figure 2. Multiple Microcomputer System.

## CONTROL OF MULTIPLE MICROCOMPUTERS

The control microcomputer directs the operation of the multiple microcomputer system with a small efficient executive program which performs the basic trainer control functions as follows:

a. System Startup and Task Loading. After system power up occurs, the functionally modular simulator programs must be loaded into the RAM (random access memory) of each application microcomputer. These modular programs reside on a disk storage device and are loaded directly into each application microcomputers' dual ported RAM.

b. System Initialization. All simulation parameters must be initialized after startup. These parameters include constants, variables, booleans, etc., which determine the state and status of the trainer at startup time. This initialization data is contained in a file on the disk storage device and is loaded into the RAM of each microcomputer by the control micro. After the system initialization is complete, then the flight initial conditions are performed.

c. Flight Initial Conditions. On each flight (trainer) scenario, initial conditions must be loaded into the RAM of each of the multiple microcomputers. These parameters are a smaller subset of the system parameter and must be reloaded each time a new flight (trainer scenario) is required. To perform this task, the control microcomputer must halt execution of the application microcomputers and then load the flight simulation parameters into the appropriate application microcomputer's RAM.

d. Instructor Station Inputs/Outputs. Special instructor station inputs or requests such as freeze, reset, etc., must be loaded by the control micro into the RAM locations on the appropriate application micro. Also any special request for training parameters must be retrieved by the control microcomputer from the appropriate application microcomputer's RAM.

e. Bus Arbitration. Bus arbitration will be handled by the hardware located on each microcomputer board. The VMEBUS has seven interrupt levels and four priority levels. The highest priority will be assigned to the control microcomputer and a lower priority will be assigned to each of the application microcomputers. This will allow the control microcomputer to gain fast access to the bus to perform any time critical functions such as start, stop, and freeze tasks.

f. Data Transfer. Data transfers between microcomputers will be handled at the beginning of each frame. Data will be organized in blocks and transferred between microcomputers using a block move technique which will reduce the number of bus arbitrations and increase the data transfer rate. The VMEBUS is rated at a 20 megabytes per second transfer rate (16 bit parallel path) but transfers will actually be slower with the overhead (bus arbitration).

g. Peripheral I/O. The control microcomputer will interface with a storage device (disk drive) and perform the I/O processor task. Disk files will be loaded into the system from the disk unit and stored on the disk as required by the instructor station. Both hard disk drives and floppy disk drives will be used to provide the convenience of permanent storage on the hard disk and backup capability on the floppy drive.

## SOFTWARE DEVELOPMENT FACILITY AND APPROACH

Software development for a multiple microcomputer is a tedious and complex task. Microcomputers do not have all the fancy development tools of a mainframe (such as a VAX) and are generally harder to program. Hence, the advantage of low cost microcomputer hardware can soon be offset by higher software development costs if the wrong approach is taken. Several methods were used on this project to reduce software development costs: (1) a high level transportable language was used (Fortran 77); (2) software partitioning was accomplished using automated techniques; and (3) a VAX 11/780 was used for software development. These are discussed below.

This project setup a software development facility utilizing a VAX 11/780 networked to a Motorola VME/10 development system. All initial development work was done on the VAX. Since both systems have Fortran 77 compilers, only minor changes were needed to execute Fortran code on the VME/10. This setup took advantage of the VAX development tools for the partitioning effort (explained below) and allowed us to download simulation software directly to the VME/10 system for testing. The VME/10 also has a card cage in the back for five more microcomputers so that a prototype multiple microcomputer system can be designed and implemented directly on the VME/10. The partitioning process is explained below.

The software development process of breaking up a big simulation program into smaller parts (partitioning) so that they can be loaded easily into the application microcomputers is illustrated in Figure 3. Figure 3 is summarized as follows: the simulator application software (row 1) is divided into smaller functional subprograms (row 2). The functional subprograms are grouped by task (row 3) and loaded as separate tasks into the individual application microcomputers (row 4). Although the partitioning process sounds simple in theory, the actual implementation is complex.

This project is developing automated techniques for dividing and distributing the simulator software on each of the application microcomputers. These techniques are designed for simulator software written in compliance with MIL-STD-1644. The basic requirements for this approach to work are (1) the program must be written in Fortran 77 and (2) the program must utilize functional subroutines which should ideally be no more than 200 lines of executeable code. Programs have been developed to strip off all input and output parameters from each subroutine from the original VTRS T2-C flight simulation source code. After the subroutines are divided and distributed among the partitions, another program checks each input and output

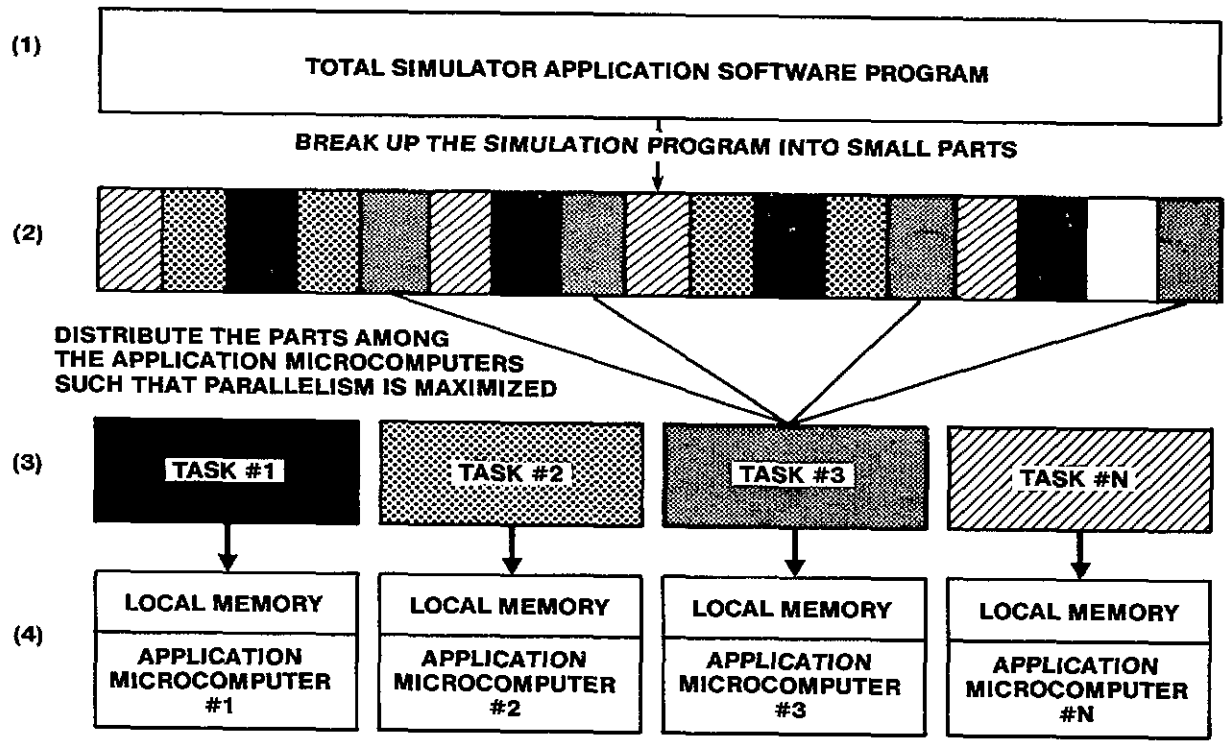


Figure 3. Multiple Microcomputer Software Development.

parameter for each subroutine to ensure that all data transfers are satisfied. Then, a PERT network analysis is done using the subroutines as nodes and data paths as links. The order of execution of the subroutines as well as the time relationship is graphically displayed using this technique. Also, the critical path can be determined if the execution time for each individual subroutine is known.

## TEST AND EVALUATION PLAN

The T2-C flight simulation software will be implemented on the multiple microcomputer system architecture and will then be integrated into the VTRS T2-C real-time flight simulator. Simulation performance parameters will be evaluated and recorded. Evaluation parameters include (1) simulation update rates, (2) precision of flight parameters, (3) startup time, (4) reliability, (5) total systems performance, (6) hardware costs, (7) software costs, and (8) ease of use.

## CONCLUSIONS

Microcomputer technology offers a real opportunity to reduce costs and increase performance for all training systems that utilize computers. However, some potential problem areas still exist for more complex trainers that require multiple microcomputer systems. This is especially true for real-time trainers such as flight simulators which require a large number of tightly coupled microcomputers to perform real-time flight tasks. Four potential problem areas are: (1) control of the microcomputers; (2) bus contention; (3) software partitioning; and (4) software development and update costs.

This project is currently in the software development phase and not all of the issues of multiple microcomputer system architecture have been completely researched. However, it has been determined that an automated software partitioning techique is essential in breaking up a large simulation program into smaller parts that can be easily loaded into the application microcomputers. Also, a high level language such as Fortran 77 or Ada should be used for the applicaton software. Although this may take more memory than assembly language, memory chips are cheap compared to software costs. The software development work should be done on a mainframe (VAX or equivalent) as much as possible, since a mainframe has better software development tools than a microcomputer.

The computer proliferation problem for trainers can be reduced by developing standards for hardware and software that support a modular design approach for trainer systems. Each type of trainer should be specified as a set of standard modules with defined interfaces for both the hardware and the software. This approach would ensure compatibility between the modules of the trainer system. Hardware modularity could be implemented by requiring computer systems to use a non-proprietary bus such as the VMEbus and a standard board size such as the Eurocard board. This would allow computer boards to fit in a standard card cage and allow them to communicate physically and electrically through the standard bus. Software modularity would be implemented by a standard software partitioning of trainer functions into "functional modules." These modules would then execute on modular hardware (microcomputers).

## ABOUT THE AUTHOR

Mr. Edward M. Holler is a Principal Investigator with the Computer Systems Laboratory at the Naval Training Equipment Center, Orlando, Florida. He is currently responsible for a multiple microcomputer system architecture project for real-time trainers. He holds a Masters Degree from the University of California, Santa Barbara, in Electrical Engineering with a specialty in Computer Science and a Bachelor's Degree from California State Polytechnic College, San Luis Obispo, in Electronic Engineering. He was formerly the visual engineer at the VTRS facility where he developed the air-to-ground software for the T2C flight simulator. In earlier associations, he was the Maintenance Engineer for the Harpoon Missile System, Flight Test Engineer for the F-14 AWG-9 weapon control system, and laboratory engineer for the Sparrow 7F missile at the Pacific Missile Test Center, Point Mugu, California.