

NO-A176 855

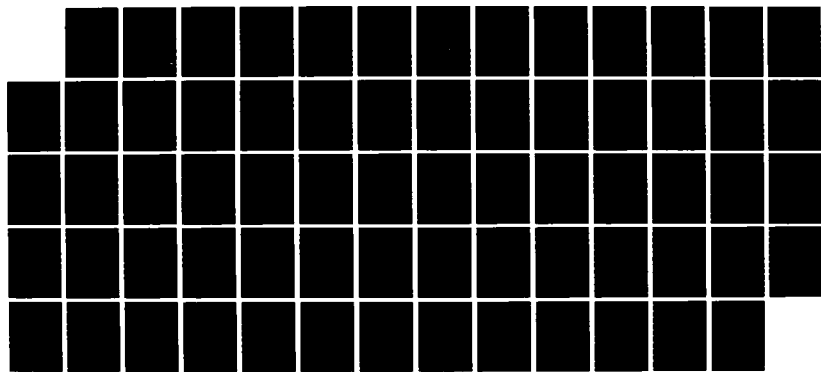
SOFTWARE DEVELOPMENT PLAN FOR THE SH-2F HELICOPTER
WEAPON SYSTEM TRAINER (U) EYRING RESEARCH INST INC
PROVO UT APPLIED TECHNOLOGIES GROUP.. 02 NOV 84
500-0015 N62269-84-C-0424

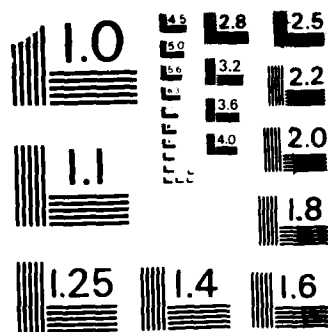
1/1

UNCLASSIFIED

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(2)

500-0015

SOFTWARE DEVELOPMENT PLAN
FOR THE SH-2F HELICOPTER WEAPON SYSTEM
TRAINER (DEVICE 2F106) SOFTWARE CONVERSION
MODIFICATION

Contract No. N62269-84-C-0424 (Competitive Award)
Contract Value: \$468,000

AD-A176 855

Prepared for:

Naval Air Development Center
Warminster, Pa. 18974

Sponsor:

Cmdr. Bateman
NAVAIRSYSCOM
AIR (413), Washington, D.C.

Accession For	
FTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	<i>per</i>
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

DTIC
COPY
INSPECTED
8

Prepared by:

Eyring Research Institute, Inc.
Applied Technologies Group
411 West 7200 South, Suite 100
Midvale, Utah 84047

(801) 566-5628

November 2, 1984

DTIC
ELECTED
FEB 19 1987
E

DTIC FILE COPY

This document has been approved
for public release and sale in
distribution is unlimited

57 1 0 070

SOFTWARE DEVELOPMENT PLAN

Table of Contents:

<u>Section</u>	<u>Description</u>	<u>Page</u>
1.0	INTRODUCTION	1
1.1	Scope	1
1.2	Purpose and Application	1
1.3	Definitions	1
2.0	PROJECT ORGANIZATION	2
2.1	Corporate, Group, and Project Organization	2
2.2	Project Staffing	2
2.3	Staff Assignments	5
3.0	PROGRAM DESIGN APPROACH	9
3.1	Incorporation of Trainer Services Program	9
3.1.1	Process Controller	9
3.1.2	Services	10
3.1.3	Interrupt Handler	10
3.2	Executive Module Modification	13
3.3	Modification of HQA Module	13
3.4	Disk File Modification	15
3.5	Other Modifications	15
3.6	Support Software	18
4.0	IMPLEMENTATION APPROACH	20
4.1	Programmatic Conversion Process	20
5.0	RESOURCE UTILIZATION CONTROL	22
5.1	Timing Analysis Tools	22
5.1.1	Spare CPU Time	22
5.1.2	Spare Time Measurement	23
5.1.3	Module Timing	23
5.2	Memory Resources	24
6.0	CERTIFICATION TEST PHILOSOPHY AND PLANS	27
6.1	Unit Test	27
6.1.1	Code Walk-Through	28
6.1.2	Code Execution	28
6.2	Integration Test	28
6.2.1	Linkage Test	28

SOFTWARE DEVELOPMENT PLAN

Table of Contents (cont.)

<u>Section</u>	<u>Description</u>	<u>Page</u>
6.2.2	Input/Output Test	28
6.2.3	Operational Test	29
6.2.4	Relocatability	29
6.3	Acceptance Test	29
7.0	PROGRAMMING SUPPORT CENTER	30
7.1	Programming Facilities	30
7.2	Programming Tools	30
7.2.1	Emulator Test System	30
7.2.1.1	Hardware Implementation	31
7.2.1.2	Software Implementation	31
7.2.2	System Test Driver	33
7.2.3	REBUG	34
8.0	QUALITY ASSURANCE	35
9.0	PROGRAMMING STANDARDS	36
9.1	Programming Guidelines	36
9.2	Modularity Guidelines	37
10.0	CONFIGURATION MANAGEMENT	38
10.1	Identification	38
10.2	Tracking Changes to the Software	38
10.3	Documentation	39
10.4	VISTA	41
11.0	GOVERNMENT-FURNISHED EQUIPMENT AND SERVICES	43
12.0	SOFTWARE INTEGRATION	43
12.1	Integration of Phase II Software	46
12.2	Pre-Installation Testing	46
13.0	AREAS OF RISK	47
13.1	Input/Output Testing	47
13.2	Integration of Phase 2 Software	47
13.3	Delivery of GFE Items	47
14.0	SCHEDULES AND MILESTONES	48
15.0	RESOURCE ALLOCATION	62

1.0 INTRODUCTION

1.1 Scope

This Software Development Plan (SDP) describes Eyring Research Institute's comprehensive approach for implementing the software conversion/integration tasks of Navy contract N62269-84-C-0424. The scope of this contract includes the following tasks to be accomplished by Eyring:

- a. Development/conversion of software required to facilitate the transition of the SH-2F Helicopter Weapon System Trainer (Device 2F106) Main Trainer Program for use on the trainer's upgraded Harris computer system, under control of the Harris standard operating system (VOS).
- b. Software integration support for inclusion of NAVAIRDEVCON Phase 2 changes into the SH-2F Operational Software VOS version cited in item a.
- c. Software integration support associated with the Flight Control Loader upgrade objective of the SH-2F WST Improvement Program.
- d. Documentation and training related to items a. through c.

1.2 Purpose and Application

The objective of this SDP, and the contract under which it is specified, is to accomplish NAVAIRDEVCON modifications to the SH-2F WST to provide a more efficient platform to train LAMPS H-2 aircrew in all aspects of procedures and tactics.

This document has been written and formatted in accordance with Contract Data Requirements List (CDRL) Item A004, Data Item Description DI-A-2176A, and applicable requirements of DOD-STD-1679A(NAVY). The following sections provide detailed information regarding Eyring's project organization; design, implementation, and resource utilization approach; testing plan and philosophy; and software integration plan. Other topics addressed include areas of risk, project schedules and milestones, and resource allocation.

1.3 Definitions

Terms used in this document generally comply with the definitions of such terms given in DOD-STD-1679A(NAVY). Terms used in this document which are not defined in DOD-STD-1679A(NAVY) are defined at the point of usage.

2.0 PROJECT ORGANIZATION

2.1 Corporate, Group, and Project Organization

In accordance with corporate policy, decision responsibility for performance of this contract has been delegated to the appropriate Eyring group and division; in this instance, Eyring's Salt Lake City Division, which is a part of Eyring's Applied Technology Group. Figure 2-1 depicts Eyring's corporate organization to the division level.

Technical and financial management of the project is conducted at the division/project manager level, with the project manager directly responsible for technical, schedule, and budgetary performance. The project manager is given broad management latitude to ensure efficient, timely performance of contract tasks. Project managerial performance is, in turn, monitored and reviewed by means of periodic formal presentations to upper management personnel, and by detailed, frequent budget reviews and expenditure reports generated by Eyring's computerized accounting system.

2.2 Project Staffing

The project is staffed and structured so as to achieve maximum efficiency (refer to Figure 2-2 for project organization). The project staff is composed of qualified, trained individuals who are proficient and experienced in Eyring's standards and techniques related to projects of this nature. Where possible, the project's conversion and development tasks are divided into modular groups, and each task group is then assigned to a specific development team. Each team member is required to interact with other team members for purposes of conducting design and code walk-throughs and other related reviews and evaluations, with the ultimate result being an integrated, well-designed product that complies with Eyring and contractual standards and requirements.

The individuals assigned to this project are:

- a. Rosalee E. Millard, Project Mgr. and Sr. Project Engineer
- b. System/Software Analyst 1
- c. Lynn Hancock, System/Software Analyst 2
- d. Russell A. Carter, Data Manager and Technical Writer
- e. John Spencer, Staff Engineer 1
- f. Martin T. Jakub, Staff Engineer 2
- g. Diane Ogden, Technical Typist

Divisional and group managerial responsibilities will be fulfilled by Clyde M. Stauffer IV and David K. Sorenson, respectively.

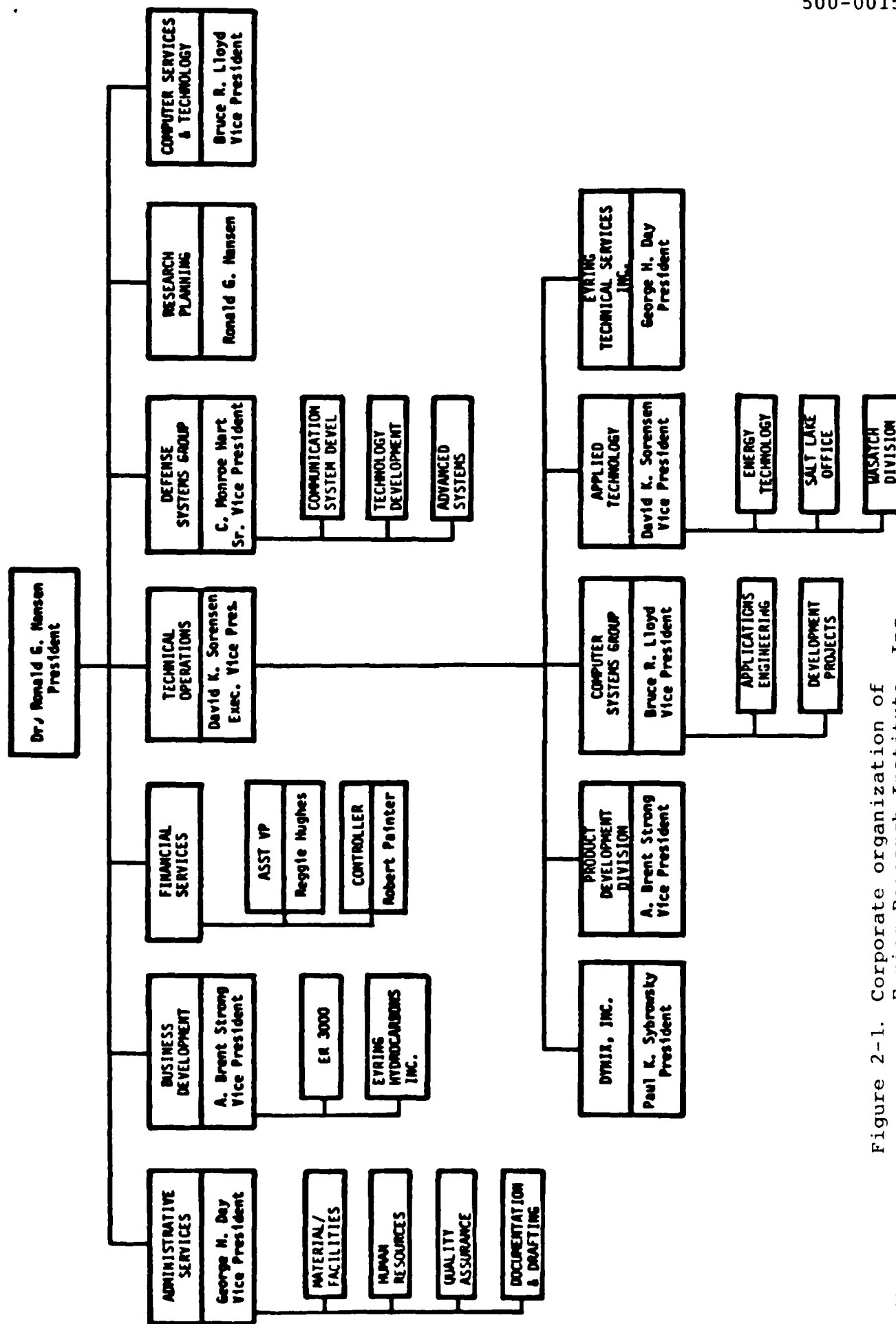


Figure 2-1. Corporate organization of Eyring Research Institute, Inc.

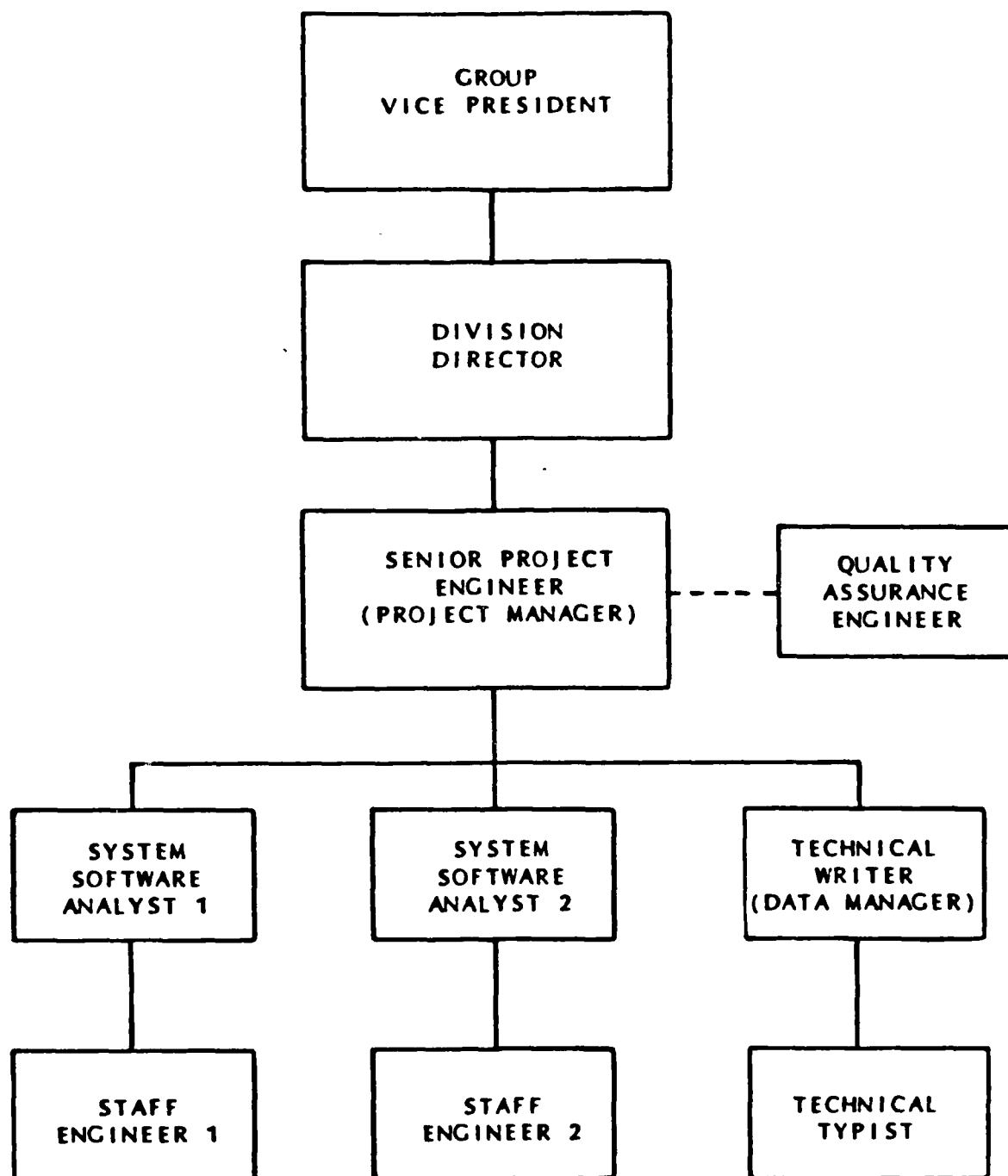


Figure 2-2. SH-2F software conversion project organization.

2.3 Staff Assignments

The Project Manager/Senior Project Engineer has prime responsibility for the technical management and team performance/execution of the overall project. Functional duties include:

- a. Develop the Project Plan (PP).
- b. Monitor and evaluate project status and prepare and submit performance and cost reports.
- c. Prepare Design Approach and Software Development Plan using the PP and software baseline data as guides.
- d. Schedule and conduct design reviews.
- e. Monitor progress of project personnel to ensure compliance with design approach.
- f. Coordinate with QA on the development of Software QA Plan and Computer Program Test Plan, Procedures and Report.
- g. Perform conversion of the Executive Routine (HAA), Trainer Initialization Routines, and the Timer Interrupt Handler.
- h. Provide Technical Support Services for the flight control loader upgrade at Norfolk and North Island NAS.
- i. Supervise integration of new government-furnished software into the trainer baseline.
- j. Monitor and participate in verification testing.
- k. Provide integration support at Norfolk and North Island NAS.
- l. Monitor and participate in Acceptance Test Procedures at Norfolk and North Island NAS.
- m. Supervise and monitor progress of documentation development.
- n. Supervise and monitor timely and accurate CDRL compliance.

The System/Software Analysts will function under the direct supervision of the Project Engineer/Manager and will be responsible for planning and executing the conversion/integration requirements. There will be two software development teams, each headed by a System/Software Analyst. Both teams will share the following functional duties:

- a. Review and become familiar with the GFE software baseline.
- b. Participate in analysis and development of the Software Development Plan.
- c. Participate in design reviews.
- d. Participate in formulating Software Development Guidelines.
- e. Prepare the applicable software documentation.
- f. Perform technical support services for the Flight Control Loader upgrade at Norfolk and North Island NAS.
- g. Participate in verification testing.
- h. Perform integration support at Norfolk and North Island NAS.
- i. Participate in Acceptance Testing at Norfolk and North Island NAS.

Functional requirements specific to Team 1 include:

- a. Convert System Drivers (I/O handlers), Put/Get file handlers, and Trainer Data Files.
- b. Help prepare the Program Package.

Functional requirements specific to Team 2 include:

- a. Modify the Emulator Test System and Development Tools Package.
- b. Convert the CDB, Trainer Simulation Modules, Trainer Utilities, and Diagnostics.
- c. Perform integration of government-furnished software into the trainer program baseline.

Staff Engineers will function under the supervision of

a System/Software Analyst. They will perform conversion/integration in conformance with specified team assignments. Their primary responsibilities include participation in the design, conversion, integration, testing, and documentation of trainer software.

The Quality Assurance Engineer will function under corporate control to assure QA independence. The primary responsibilities of the QA Engineer are:

- a. Assist in the preparation of the Software Development Plan.
- b. Participate in design reviews.
- c. Develop the Software Quality Assurance Plan.
- d. Develop the Computer Program Test Plan, Procedures, and Reports.
- e. Provide configuration management for:
 - 1) Software baseline
 - 2) Government furnished software
 - 3) The new Flight Control Loader software
- f. Monitor verification testing.
- g. Provide necessary QA support and oversight for project execution.

The Technical Writer/Data Manager will be responsible for revision, development and production of all technical documentation and data preparation and delivery during project execution. His functional duties include:

- a. Assist in preparation/publication of the Project Plan (PP).
- b. Review GFE documentation and establish the documentation baseline.
- c. Assist in the preparation of the monthly Performance and Cost Reports.
- d. Assist in the preparation/publication of the Software Development Plan.
- e. Assist in the preparation/publication of the Quality Assurance Plan, and Computer Test Plan, Procedures, and Reports.

- f. Prepare and publish applicable page changes for all converted/integrated software documentation.
- g. Assist in preparation/publication of the Software Development Guidelines.
- h. Develop and publish the Operator's Manual.
- i. Modify the Trainer Programming Report.
- j. Assist in preparation/publication of the Verification and Acceptance Test documentation.
- k. Assist in preparation/delivery of the Program Package.
- l. Develop training course and associated course materials in collaboration with subject matter experts.

The Technical Typist will assist in the preparation of all reports, plans, and software documentation.

3.0 PROGRAM DESIGN APPROACH

This section describes Eyring's overall approach for providing a software system that is efficient, easy to maintain, and uses the numerous capabilities of the Harris H800 computer and VOS operating system to provide improved SH-2F WST performance and expandability. It should be noted that Eyring considers this project to be primarily a conversion effort rather than a design effort, and has formulated its technical recommendations based on that consideration. The project will involve a limited amount of design to replace or rewrite obsolete or unusable software components; however, this effort must be conducted within the constraints posed by the requirement to conform with pre-specified hardware and software (operating system) capabilities and configurations. The majority of existing SH-2F modules will undergo modification (conversion) requiring only module syntax changes and removal of HOLD pseudo-operations. These changes will be accomplished globally and programmatically, and are described in detail in Section 4.0. Changes to software components that may involve restructuring or rewriting of a module, or modules, are described below.

3.1 Incorporation of Trainer Services Program

System services provide the interface between user programs and the Harris VOS operating system. This section addresses and describes the special services, or routines, that must be made available to the SH-2F Main Trainer Program to enable real-time performance and Direct Memory Access (DMA) I/O under VOS. The group of routines designed to provide this capability has been designated the Trainer Services program, and comprises three functional areas: 1) the process controller, 2) the services, and 3) the interval timer (real time clock) and I/O interrupt handlers.

3.1.1 Process Controller

The process controller provides the loading/unloading mechanism for Trainer Services. This is accomplished by changing memory BLU location '37 to point to the Trainer Services main routine and then putting the process controller in wait mode while the Main Trainer Program (MTP) executes. When the process controller is removed from the wait mode, it restores the original contents of location '37 and exits the system. This method of loading and unloading the Trainer Services program provides a means whereby only the MTP can access Trainer Service routines, and eliminates the possibility of another program inadvertently accessing one of these services or a spurious interrupt being processed by the special interrupt handlers.

3.1.2 Services

The MTP has several needs that cannot be met by the existing system services. These include the capability to: a) log error messages from a user application to the OPCOM and the OPCOM log, b) initialize the interface between VOS and the MTP, c) change interrupt vectors, and d) place the MTP in wait state so that other processes may execute. The services area provides a) trainer interface initialization, b) trainer reset, c) trainer wait, d) OPCOM messages, e) interval timer start and stop control, and f) I/O stacking.

The trainer interface initialization routine performs such functions as setting up addresses for the interval timer interrupt handler and the I/O interrupt handler, initializing variables for timing routines, and setting the interval timer counter. Figure 3-1 presents the interface relationship between the MTP, VOS, and the System Service Directory described below.

The trainer reset function performs restoration of the original contents of the interval timer and the interrupt locations for the interval timer and I/O, and removes the process controller from the wait mode.

The trainer wait routine is used to put the MTP in wait mode after the MTP has completed execution of a given frame.

The OPCOM message routine is used to provide communications from the MTP, such as sending a status message to the OPCOM terminal and OPCOM log.

The interval timer control routines perform interval timer start and stop functions required by the Emulator Test System described in Section 7.

The stacking routine performs functions required to operate the I/O stack so that it can be initialized by the I/O interrupt handler.

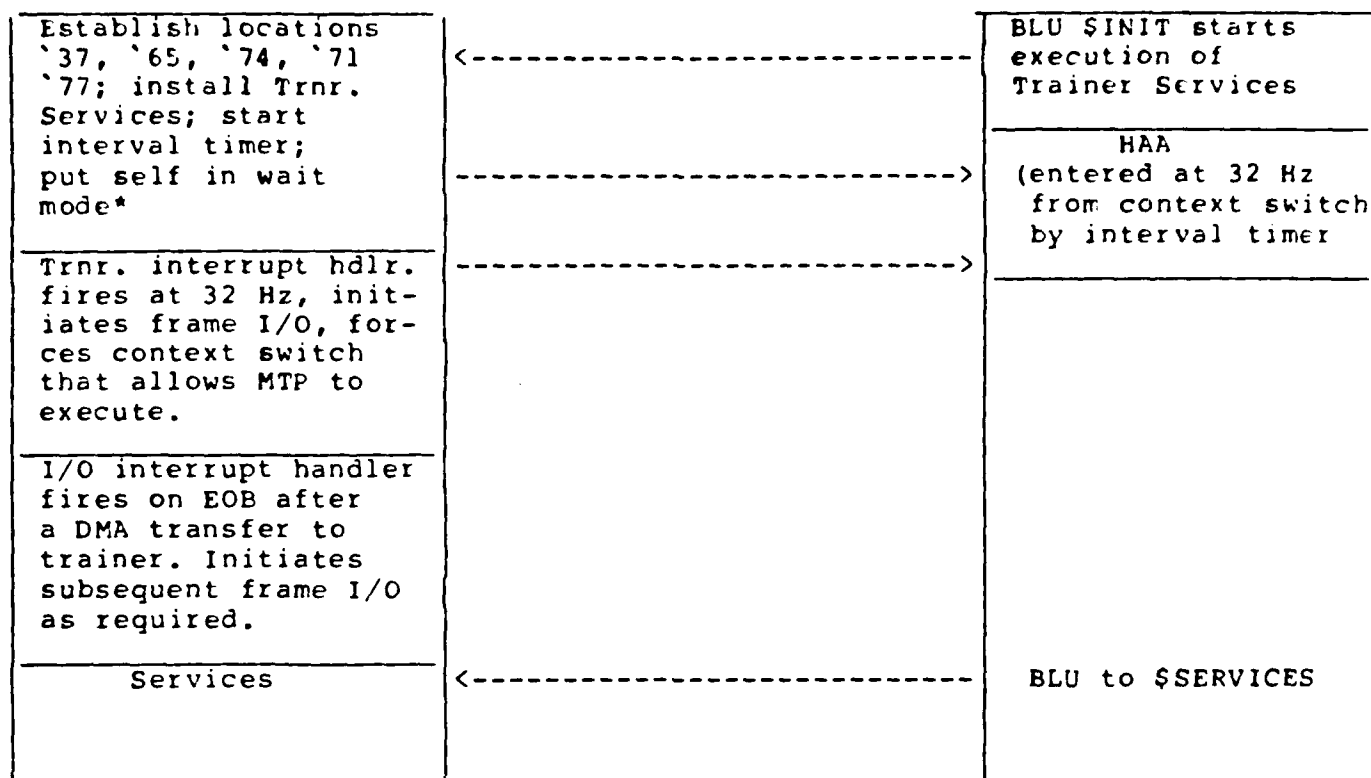
To facilitate incorporation of user services such as Trainer Services, VOS uses a System Service Directory (SSD). User services will be incorporated by adding the name of the service routine, the BLU vector, and the sequence number of the routine to the SSD as shown in Figure 3-1.

3.1.3 Interrupt Handler

The addressing mode capabilities of VOS require that all I/O interrupt handling functions reside in physical address space (monitor mode); i.e., at system level. Thus, those software functions -- including the I/O interrupt handler and interval

TRAINER SERVICES

MAIN TRAINER PROGRAM



SYSTEM SERVICE DIRECTORY

TRINIT	`0037	`0001
TRRESET	`0037	`0002
TRWAIT	`0037	`0003
TRSTOP	`0037	`0004
TRSTART	`0037	`0005
O/M	`0037	`0006
HAMSTRY	`0037	`0007

*When putting Trainer Services in wait mode, the system performs a context switch to allow control to return to the Main Trainer Program.

Figure 3-1. VOS/Main Trainer Program Interface block diagram.

timer interrupt handler -- will be incorporated into the Trainer Services program. Each time the interval timer counter reaches zero, an interrupt is generated. The interrupt handler resets the counter, processes the MTP cycle counters, sets up I/O for the frame, processes mode flags, and initiates frame execution. The interval timer interrupt handler will execute at 32 Hz, the rate at which I/O must be initiated. After initiating I/O, the handler will then, by context switching, return control to the Executive module HAA. HAA will determine if the MTP should return to a wait state, execute high-priority simulation routines, or, if it determines that an execution lag condition exists, execute the necessary routines to eliminate the lag during the remaining frame time.

The I/O interrupt handler will execute at the completion of each DMA transfer, and will initiate the next block I/O transfer and return to the point of interruption. As explained in Section 3.2, all I/O routines will be removed from the Executive module HAA and placed under Trainer Services. Input/output rates and order of transfer will remain intact and unchanged.

Refer to the table below for a listing of Trainer Service routines and functions.

Table 3-1. Trainer Services Routines

<u>Name</u>	<u>Description</u>
BLU Vector Definition	-Entry into Trainer Services
Timer Interrupt Handler	-Handles all bookkeeping functions Initiates new I/O Wakes MTP if in wait state Forces context switch Returns to point of interruption or to HAA
I/O Interrupt Handler	-Initiates further I/O Returns to point of interruption
O/M	-Outputs message to OPCOM, log
TRINIT	-Initializes Trainer Services
TRRESET	-Resets Trainer Services
TRSTART	-Starts interval timer
TRSTOP	-Stops interval timer
TRWAIT	-Puts MTP in wait state
HADSTKA	-Stacks I/O routines

3.2 Executive Module Modification

The Executive module (HAA) of the Main Trainer Program (MTP) will be completely rewritten. Present HAA I/O functions will be removed from the program and placed in the Trainer Services program. The HAA module will become the event scheduler for the MTP on a real-time basis -- HAA will, in operation, pass control to a series of other MTP modules which are scheduled for a specific time frame. No other program will execute at a priority equal to or higher than the MTP, thus ensuring that when the interval timer interrupt handler forces a context switch, HAA will be the first routine executed after the interrupt.

Upon execution, HAA will check a frame counter updated by the interval timer interrupt handler to determine whether the new frame is even or odd numbered. MTP modules are executed during even frames; HAA will exit immediately during an odd frame to provide other users with processing time. If an execution lag is detected during an odd-numbered frame, MTP module execution will continue.

The current system of tables to define execution rates will be modified to use a bit mapping scheme. This approach uses a table containing a four-word entry for each module. The first word is the actual bit map, the second and third words are the instruction (extended addressing branch) to the module associated with that bit map, and the fourth word will contain the execution time of the module when the module timer is activated. The bit map will have the structure shown in Table 3-2.

The code will scan this table, checking bits in the bit maps for each of the modules and comparing them with MTP conditions to determine if the individual modules should execute. This system makes it much easier to streamline the performance of the MTP by allowing the programmer to use the RMM device to activate module timing and manipulate execution rates and groupings to find the optimal configuration.

As HAA completes execution of the modules, it will check to see if a lag condition exists. Under normal conditions (no execution lag), HAA will call Trainer Services, which, in turn, will put the MTP in a wait state to allow other processes to run on the system. The MTP will be taken out of the wait state by the interval timer interrupt handler.

3.3 Modification of HQA Module

The Virtual Memory features of VOS make implementation of overlays complex and inefficient. Therefore, the present overlay routines will be integrated directly into the MTP and made memory-resident. All HQA calls to overlays will be converted to direct calls. This will not affect the logic flow of HQA.

Table 3-2. HAA Bit Map Structure

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F	T	H	M	L	S	S	S	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
R	I	I	E	O	P	P	P	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
E	M	G	D	W	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
E	E	H	I		R	R	R	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
Z	R		U		E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
E			M																				
								1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1
								6	5	4	3	2	1	0									

BIT 23 - IF SET THIS MODULE WILL EXECUTE WHEN MTP IS IN FREEZE MODE

BIT 22 - IF SET, ENGAGE TIMER ROUTINES TO TIME EXECUTION OF MODULE

BIT 21 - SET IF A HIGH MODULE

BIT 20 - SET IF A MEDIUM MODULE

BIT 19 - SET IF A LOW MODULE

BIT 18 - SPARE

BIT 17 - SPARE

BIT 16 - SPARE

BIT 15 - IF SET, EXECUTE IN FRAME 16

BIT 14 - IF SET. EXECUTE IN FRAME 15

BIT 13 - IF SET, EXECUTE IN FRAME 14

BIT 12 - IF SET, EXECUTE IN FRAME 13

BIT 11 - IF SET, EXECUTE IN FRAME 12

BIT 10 - IF SET, EXECUTE IN FRAME 11

BIT 9 - IF SET, EXECUTE IN FRAME 10

BIT 8 - IF SET, EXECUTE IN FRAME 9

BIT 7 - IF SET, EXECUTE IN FRAME 8

BIT 6 - IF SET, EXECUTE IN FRAME 7

BIT 5 - IF SET, EXECUTE IN FRAME 6

BIT 4 - IF SET, EXECUTE IN FRAME 5

BIT 3 - IF SET, EXECUTE IN FRAME 4

BIT 2 - IF SET, EXECUTE IN FRAME 3

BIT 1 - IF SET, EXECUTE IN FRAME 2

BIT 0 - IF SET, EXECUTE IN FRAME 1

3.4 Disk File Modification

The structure of the Main Trainer Program (MTP) will remain the same with regard to disk I/O. The Put/Get routines will be rewritten to: a) use relative records in unblocked files (instead of sector addressing), b) initiate parameter lists, and c) call standard VOS I/O services to place the request on the system disk queue.

All changes to file structure will be transparent to the user. At present, the only modification envisioned is removal from the PPM file of all directories used to access other files. These directories will be placed in the last record of the file they refer to, permitting quicker access, modification, and replacement.

3.5 Other Modifications

In addition to the modifications required to interface the MTP with VOS, Eyring has determined that the following modifications will be required to provide memory expansion capability, convenient FORTRAN interface methods, and relocatability for the SH-2F WST software. These modifications include:

1) Incorporating extended addressing -- The Main Trainer Program will be converted to take advantage of the extended addressing capabilities of the Harris H800 computer and VOS operating system. The primary benefit achieved by this change is increased memory address space in the MTP (to 1 MW), providing ample expansion room. Also, a new module can then be written without concern or regard to which map it will reside in (map independence). In the current version of the MTP, modules in the upper map can only access variables in the lower map via pointers and indirect instructions. These indirect memory access go through pointers in Direct Address Constant (DAC) tables. By using extended instructions, these pointers, and their dummy names, can be eliminated. As an example, the code

```
DFKYBDP  DAC      CDFKYBD  Floor Keyboard
...
TMA*      DFKYBDP      Load Keyboard Indirect(*)
```

will be replaced by

```
TMA                      %CDFKYBD  Floor Keyboard
```

The replacement code is preferable since it is slightly faster in execution, and is easier to understand.

The change described above will be done programmatically. A list of DAC's (pointers) will be collected, and all indirect instructions through the pointers will be replaced by extended

(direct) instructions. A cross-reference utility will later be used to find DAC's that are not used by the program. These unreferenced DAC's will be removed from the code.

In general, any instruction that accesses non-local variables must be extended, since the variables may be in a distant map. Thus, Eyring will to extend all direct or indexed instructions that access non-local data. A software tool will be used to accumulate a list of local data names, which will then allow another program to do the necessary changes to the executable code.

2) Removing executable code from the CDB -- To facilitate easier maintenance and provide for cleaner partitioning of data and executable code, subroutines currently contained within the CDB will be extracted, thus making the CDB a true data repository.

3) Consolidating CDB data blocks -- Data in the CDB, currently split into "upper" and "lower" map sections because of constraints imposed by the Datacraft 6024/5 architecture, will be combined into one contiguous block. Data in the CDB will also be placed into monitor common. Data declared as common provides the means for all modules in the MTP, whether written in assembly language or FORTRAN, to globally access the same variables. In addition to standard common data, VOS supports an enhanced version of common known as monitor common. Common data further declared as monitor common data is accessible not only to all modules within a program, but to multiple programs as well. By placing the CDB in monitor common, both the Trainer Services and MTP can directly access the same memory, although each will be running as individual programs. Test programs which monitor CDB variables may also be written and executed as separate programs, running at lower priority than the MTP, without impinging on trainer time or memory resources.

In order for assembly language programs to access the CDB in monitor common, each module must be prefaced with a "COMM" and "MCOM" declaration. The symbolic names of all of the CDB variables need not be explicitly listed in the COMM statement, however, since a "STAB" statement can be used to obtain the CDB symbol table image from previous assembly of the CDB.

In the current version of the CDB, many variables are initialized at assembly time (using "DATA," "RDAT," "EQIV," pseudo-operations, etc.). Initialization of monitor common data at assembly or compilation time is not supported, however, so a utility program to generate a disk image of the initialized CDB will be created. When the MTP is started, an initialization module will read the disk image and initialize the CDB. The disk image of the initialized CDB need be regenerated only if the CDB changes. The utility program that creates the CDB initialization disk image will use a CDB definition file as its input.

4) CDB Files -- In addition to the CDB initialization disk image described above, several other files will be programmatically generated from a single CDB definition file (CDBDEF). Table 3-3 lists these auxiliary files derived from CDBDEF. Whenever the CDB is modified, these auxiliary files will be regenerated as well.

Although monitor common data cannot be initialized at assembly or compilation time, this restriction does not apply to data declared simply as common data. Thus, the CDBDEF file will use standard assembly language versions of common declaration and initialization statements, as shown by the example formats on the following page:

Table 3-3. Auxiliary Files Generated from CDBDEF

<u>File Name</u>	<u>Description</u>	<u>How Generated</u>
CDB	CDB in monitor common	Batch program with input from CDBDEF
CDBSTAB	Symbol table (STAB file) of CDB variables	Standard VOS assembler option used to save the symbol table image (STAB file)
CDBINI	Initialization values for CDB	Directly from CDBDEF, by utility program
CDBEQV	EQIVs to CDB variables for system-level programs	Batch program with input from CDBDEF
CDBCOM	FORTTRAN common declarations for CDB variables	Batch program with input from CDBDEF
(IAT)	Initializes address tables for various modes.	ITBGEN with input from CDBDEF

```

COMM /CDB/ VAR1(SIZE) (Comments) (Letter in Col 73)
CORG VAR1
DATA (INITIAL VALUE(S) OF VAR1, IF SUCH EXISTS)

COMM /CDB/ VAR2(SIZE) (Comments) (Letter in Col 73)
CORG VAR2
DATA (INITIAL VALUE(S) OF VAR2, IF SUCH EXISTS)

*
*
*
END$

```

This format is similar to that used in the existing disk-based "TRNRIC" file as described in Table 6 of "SH-2F Trainer Software Information On Initial Conditions." The letter in Column 73 is optional and describes those mode(s) in which the variable must be loaded with initial condition values (this is implemented in the present CDB).

From Table 3-3, note that the CDB itself is generated from CDBDEF. This will be accomplished by commenting-out "CORG" and "DATA" statements, and adding an "MCOM" statement to the front of the file. The STAB file, CDBSTAB, is generated by assembly of the CDB. CDBINI is a file of initialization values copied to the CDB monitor common block when the MTP is started-up. The fourth file in the table, CDBEQV, is an EQIV file for CDB variables. It is necessary for monitor mode (physical addressing) assembly-language routines to access CDB variables. The fifth file has a similar function; i.e., allowing FORTRAN files to access the CDB. Two batch programs will be provided to create these files. The last file in the table contains the initialization address tables for the IC routines. These are tables of the addresses of CDB variables to be saved when cutting a new IC, or to be loaded when restoring an old IC. The batch program that performs this function currently exists (ITBGEN). The batch programs listed in Table 3-3 will be written in FORTRAN 77 and delivered with the new MTP.

3.6 Support Software

All support software will be analyzed to determine whether specific functions may be replaced by a standard VOS utility or diagnostic, or whether the existing program can be retained and converted for use under VOS 3.1.

All DOS utilities modified by Reflectone, Inc. will be replaced by standard VOS utilities that perform similar functions. Many Reflectone-generated subroutines will also be replaced by VOS routines. All hardware diagnostic programs will be replaced by standard Harris diagnostic programs.

The trainer diagnostic programs will be converted to be compatible with VOS. Some of the utilities that were used with the existing system will be retained, but will be modified to become interactive or to use VOS capabilities.

In cases where an existing vendor support program is deemed inadequate, Eyring will create a special-purpose support program rather than modify the vendor program. For example, rather than modifying the standard assembler to obtain a global cross-reference listing of symbols, a special cross-reference program will be used. Although such special-purpose programs may require additional work beyond that required to modify the operating system or its support software, the long-term benefits of continued compatibility with new releases of the standard vendor software more than compensate for any short-term gains.

The changes described above will provide a trainer software system that can be more easily expanded, tested, and maintained.

4.0 IMPLEMENTATION APPROACH

Eyring's implementation approach has been designed to provide an efficient and reliable conversion process for SH-2F WST software. The implementation plan includes the following objectives and guidelines:

- a. Eyring will retain as much of the existing software design and control structure as possible.
- b. Eyring will perform the conversion work on the same type of computer as the present SH-2F WST system computer, to insure compatibility.
- c. Eyring will perform global syntax changes to the software programmatically, to reduce the chance of human error.
- d. Project personnel will be organized into teams with specific areas of responsibility.

The existing control structure of the SH-2F WST software will remain basically the same throughout the software conversion process. The executive module (HAA) of the Main Trainer Program is the event scheduler responsible for keeping the training program on a real-time basis. As stated in the Program Design Approach description (Section 3), the interrupt handlers for the interval timer, channel 2 I/O completion, and channel 7 I/O completion will be separate from HAA and will execute in monitor mode.

4.1 Programmatic Conversion Process

To reduce the chance of human error when making global changes to the software, Eyring has developed software that performs repetitive conversion operations automatically. The current DOS-based software baseline will be frozen and programmatically examined to identify those modules which may require change. A text-searching computer program will scan all source code associated with the simulator system for lines of code which may require module alteration. The following criteria will be used to identify such modules:

- a. Scan source for operators that end in W (_W). This identifies all I/O instructions.
- b. Scan for any interrupt control instructions.
- c. Scan for T-register instructions. Operators with T in the second or third position (_T_ or _T), except for SRT, will qualify. All T-register instructions reference the system clock.
- d. Scan for DOS, DOSB, or DOSS in the comment field. Instructions with this type of comment may be coupling directly into DOS.
- e. Scan for \$ at the beginning of an operand. This

- references an external subroutine. A list of externals will be created and each external will be scanned for modules that may require change.
- f. The programs will be test-run in unprivileged mode to detect any privileged instructions that are not detected by the preceding scans. Privileged instructions will be isolated in a known set of external subroutines to segregate them from the main simulator program. This will allow quick and convenient debugging of those routines that can potentially hang the computer.
 - g. Any instruction that makes a memory reference into the old operating system (DOS) will be detected by VOS as an illegal memory reference when executed. This will identify any instruction (and module) for change that the scans failed to identify.

After the modules requiring change have been identified, fixes that use "standard" solutions will be generated. Such standard solutions may involve the use of standard system service calls or, if needed, special "added" services which may be added to the services already provided by the standard operating system. Neither of these requires the modification of the operating system or any vendor-supplied support software.

To ensure that the conversion work is compatible with the SH-2F WST system, VOS version 3.1 will be used on Eyring's Harris H800-2C system throughout the conversion process. Refer to Section 7 for a complete list of contractor hardware and software programming resources.

As described in the project organization plan (Section 2), two programming teams -- each comprising a system/software analyst and a staff engineer -- will perform the conversion/integration tasks.

5.0 RESOURCE UTILIZATION CONTROL

Because this project principally entails conversion of an existing software design (rather than design and development of new software) that uses pre-specified hardware, processing time reserves and memory requirements are consistent with the existing system. The processing time reserves will increase from the original time reserves on the Datacraft 6024/5 because of the increased speed of the Harris H800. The increased addressing capability of the H800 will be utilized as indicated in the Program Design Approach (Section 3) to permit future code expansion.

5.1 Timing Analysis Tools

The timing analysis tools described below provide the capabilities for determining which modules may need performance enhancement and whether modules need to be redistributed to different time frames. These tools will be used primarily during the conversion and integration of the software. During the integration of new and modified modules in particular, the timing analysis tools will identify modules that may be too large and require further modification or redistribution among different time frames.

5.1.1 Spare CPU Time

Frame time is defined as the period of time necessary to accomplish a set amount of work and still maintain real-time performance. There are two types of frame time employed in the simulator.

- 1) I/O frame time, which is the length of time allowed for the completion of one I/O sequence (1/32 of a second).
- 2) Execution frame time, which is the length of time available for the completion of one call list (1/16 of a second).

Cycle time is defined as the length of one major cycle (1/4 second). A cycle is composed of 8 I/O frames or 4 execution frames.

Spare time is the actual amount of time available for use in a given execution frame after all I/O and module execution has been completed. Eyring's calculation for spare time includes a factor for the amount of system overhead that occurs during normal trainer operation.

Frame and cycle times can be altered dynamically by modifying specific constants stored in the timer interrupt handler and trainer executive module.

5.1.2 Spare Time Measurement

Accurate measurement of the spare time remaining in each time frame can be difficult. Simplistic measurement algorithms that essentially read elapsed real time fail to accurately quantify "real" spare CPU power on modern computers that feature concurrent DMA transfers, accelerated instruction pipelining, cache memory, etc. A more powerful means of analyzing CPU usage is required.

In response to stringent requirements for measuring spare time in previous real-time simulator work, Eyring has perfected an algorithm that recognizes state-of-the-art processing techniques and which fulfills the requirements of MIL-D-83468. This algorithm will be incorporated in a new routine named TlT2, which will replace the current HAHSTRY routine and provide significantly enhanced capabilities.

When activated, TlT2 will receive control at the end of each execution frame and continually execute a small loop of instructions. The elapsed "clock" time for each loop will be measured. A value for the fastest loop time encountered during a run will be continually updated while the other loop times will be averaged for each frame and cycle. The fastest loop time corresponds to the case when no cycle-stealing I/O or system overhead occurs to slow CPU execution. By taking the ratio of the fastest loop time to the average loop time for each frame and cycle, weighting factors may be obtained. Multiplying these factors by the elapsed "clock" spare time yields the effective spare CPU time.

The TlT2 routine will specifically identify the following:

- Time used in worst-case frame (T2)
- Percent of spare time in worst-case frame (TS2)
- Time used in worst-case cycle (T1)
- Percent of spare time in worst-case cycle (TS1)
- Average cycle execution time over a test run
- The maximum frame execution time for each frame
- The average frame execution time for all frames over a test run
- The identity of the frame number and encompassing cycle of any frame exceeding available frame time (lagging frame).

5.1.3 Module Timing

Individual MTP modules may be timed to provide performance

information necessary to evaluate a new or modified module. This timing measurement can be activated by a software engineer using the RMM device to set the timer bit in the module entry of the HAA bit map table (described in Section 3.2). The actual execution time of the module is stored in the fourth word of the module entry in the execution table contained in HAA. Using the load map, the engineer can determine the address of the storage location and set the RMM device to monitor the execution time of a module.

5.2 Memory Resources

The memory required by the current version of the SH-2F MTP is approximately 70 kilowords (KW), including the Common Data Base and overlay programs. This value will increase as new code is added to improve the fidelity of the MTP. Converting the MTP to execute on the Harris H800 computer under VOS provides the potential for greatly increasing the memory address space available to the MTP.

Programs may be written to run in one of three addressing modes on the H800. Compatibility mode, which emulates the addressing of the Datacraft 6024 series of computers, constrains executable code to a maximum size of 64 KW, with access by indirect and indexed addressing to 256 KW of data (not executable). Much larger executable code and data address space is obtainable by using extended instructions, supported by the H800 in either the extended or full addressing modes. Executable code can increase to a maximum of 1 megaword (MW) in both modes while a data address space of 1 MW for extended addressing or 3 MW for full addressing is available.

As part of the conversion process, Eyring will perform the changes required to allow the MTP to execute in the extended addressing mode, providing a maximum virtual address space for the MTP of 1 MW. Using extended addressing rather than full addressing will allow integration of any modules written in FORTRAN (the Harris FORTRAN compiler currently supports only the extended addressing and compatibility modes). Because extended instructions use two words of memory -- one for the operation code and one for the address of the memory reference -- only instructions which reference the global variables in the CDB will be converted into their extended form. The use of extended instructions, while requiring an extra word for those instructions which reference CDB variables, obviates the need for DAC tables and their associated memory and allows all modules to be completely map-independent. Approximately 890 KW of spare virtual memory address space will be available for future MTP expansion needs by using extended addressing.

The Common Data Base will be placed in monitor common to make all of the CDB variables accessible by all of the modules,

whether written in FORTRAN or assembly language, in both the Trainer Services Program and the MTP. All monitor common blocks are loaded in high virtual memory, after all executable code, by the Harris Vulcanizer program.

Figure 5-1 illustrates the memory maps for the Trainer Services monitor-mode program and the MTP user-mode program after conversion is completed. The amount of memory, approximately 100 KW, for the MTP includes additional memory for extended instructions as well as modules for the System Test Driver. Added together, the amount of physical memory required for VOS and all resident trainer software totals approximately 177 KW. For a physical memory size of 1536 kilobytes, this leaves 335 KW, or 189%, as spare physical memory.

1 Megaword

Spare Virtual Memory
Common Data Base (approx. 12 KW)
Trainer Services (approx. 1 KW)
TRAINER SERVICES VIRTUAL MEMORY

1 Megaword

Spare Virtual Memory
Common Data Base (approx. 12 KW)
Main Trainer Program (approx. 200 KW)
MAIN TRAINER PROGRAM VIRTUAL MEMORY

512 Kilowords

VOS (approx. 64 KW)
Spare Physical Memory (approx. 335 KW)
Common Data Base (approx. 12 KW)
Main Trainer Program (approx. 100 KW)
Trainer Services (approx. 1 KW)
PHYSICAL MEMORY

6.0 CERTIFICATION TEST PHILOSOPHY AND PLANS

This section describes the general philosophy and plans developed by Eyring to ensure that software delivered under this contract satisfies Statement of Work requirements. The overall purpose of the tests described below is to assist the programmer/analyst in determining whether specific software components meet these requirements within the guidelines and policies of Eyring's conversion approach scheme, quality assurance standards, and configuration management methods. The stated purpose of each test must be accomplished before further conversion, development, or testing can be undertaken.

To accomplish the objectives of this contract, a majority of the SH-2F software components (modules) will require conversion, rather than design and development. A converted module is that which has only undergone a syntax change or similar minor modification; its logic flow will remain intact. A developed module is that which has been designed by Eyring to satisfy the functions of an existing module that is to be replaced. Eyring's test philosophy has been designed to incorporate the testing needs of both module types.

Deficiencies or flaws discovered during testing will be sequentially recorded on a single Software Trouble Report (STR). The STR number will be entered in the VISTA change record as the deficiencies are analyzed and corrected. This approach will provide dual paper-and-programmatic tracking of software deficiencies and corrections. Following correction by the cognizant programmer/analyst, the module will undergo another full test.

Upon successful completion of unit testing, the module will be "configured" and will proceed to the integration testing phase. (Refer to Section 10 for a discussion of Eyring's configuration management plan.)

6.1 Unit Test

The first structured test applied under this plan to a software module is the unit test. The purpose of a unit test is to ascertain the module satisfies the following requirements:

- a. Successfully passes a code walk-through
- b. Satisfies quality assurance requirements
- c. Compiles/assembles without error
- d. Executes correctly, as designed
- e. Satisfies design requirements.

6.1.1 Code Walk-Through

A successful code walk-through will verify that the source code complies with programming and quality assurance standards. The walk-through will be performed by the cognizant programmer/analyst and the project quality assurance representative. In the case of converted modules, the walk-through will serve to verify the correctness of the module's changed syntax, accomplished by a line-by-line comparison of the original and converted module. If the walk-through indicates a logic change to the converted module, that module will then be considered a developed module and will be subject to all test requirements applied to developed modules. A walk-through review of developed modules will focus closely on the module's logic flow, in addition to its adherence to the above-mentioned quality and programming standards.

6.1.2 Code Execution

Module execution will be verified through the use of test driver programs developed by the cognizant programmer/analyst for specific modules, or by use of the Harris debugger program. Converted modules which successfully pass the code walk-through will not be tested for execution, as the module's inherent logic has not been changed. Test drivers will be designed to ensure that all source code lines in developed modules are executed. Successful execution using the test driver or the Harris debugger will verify that the design objectives and requirements of the developed module have been met.

6.2 Integration Test

Configured modules, both converted and developed, will undergo integration testing which has the following objectives:

- a. Verify error-free linkage of modules
- b. Verify proper input/output execution
- c. Verify proper interface design and program operation and initialization
- d. Verify relocatability

6.2.1 Linkage Test

Configured modules will be linked to their appropriate programs. Converted modules will be linked as a group; developed modules will be linked individually.

6.2.2 Input/Output Test

Developed-module I/O will be tested to verify compliance

with design requirements, including the proper handling of erroneous inputs. Programs that require user inputs will be subjected to test inputs that exercise the full range of input parameters up to and beyond boundaries. Programs not requiring user inputs will be "snapshot" during execution, using a debugger or special program.

6.2.3 Operational Test

Operational testing will examine the interface between the user, the hardware, and the software. User interfaces -- such as input prompts, error messages, and help messages -- will be reviewed to ensure their clarity and convenience. Methods of correcting improper inputs will be examined for similar attributes. Hardware interfaces (device channels) will be examined to ensure that proper data and commands are being transmitted to the channel, and to determine that the device functions properly.

6.2.4 Relocatability

Software relocatability will be verified by checking the program's load address over several iterations of program execution.

6.3 Acceptance Test

After the converted software system is delivered and installed at each facility, the system will undergo an acceptance test in accordance with Eyring's Acceptance Test Plan and Procedures documents. Test procedures will verify trainer performance within the scope of the SOW. The Acceptance Test Plan and Procedures will be largely based upon previous contractor experience with similar installations. Deficiencies found during acceptance testing will be corrected and retested; or if acceptable to NADC representatives, will be waived. A final test report will formally summarize the results of the acceptance test.

7.0 PROGRAMMING SUPPORT CENTER

7.1 Programming Facilities

Software module conversion, development, and system integration will be performed using Eyring's Harris H800 computer system. Configured with 3.0 megabytes of physical memory, cache memory option, and a 474 megabyte disk drive, the system provides more than adequate processing time and storage capability. Since the system is not used for day-to-day office automation and administrative support, dedicated computer time for intensive project development is available. Each member of the programming team has a separate CRT terminal linked to the H800 computer.

Hardware and software included in the system are listed below.

<u>Hardware</u>	<u>Software</u>
-Harris H800-2CP computer -Integral floating-point processor -3 MB of high-density memory -6 KB of high-speed cache memory -32 priority interrupts -System console terminal and Maintenance Aid Processor -474-MB Winchester disk drive with IDC -Streaming magnetic tape unit -600-LPM line printer -16-port CNP -Buffered Block Channel (BBC) -Programmable Input/Output Channel -Interactive CRT, Models 8686/8685	VOS Operating System, Version 3.1, including: -Cross Referencer -System Debugger -Symbolic Debugger -Macro Assembler -FORTRAN 77 -Text Editor -VISTA

7.2 Programming Tools

7.2.1 Emulator Test System

Eyring will use an enhanced version of its Emulator Test System to fly and test the simulator software in real-time without a trainer cockpit. Joystick controls will replicate the cyclic stick and collective lever, and yaw pedal controls will also be used. Additional analog inputs will be available for configuration as required for specific tests. Discrete inputs will be replicated using keyboard inputs and switch inputs. A low-resolution graphics display and additional alphanumeric CRT display will indicate aircraft track, altitude, horizontal and vertical

velocities, aircraft attitude, etc. All CDB variables will be accessible by the Emulator Test System. Figure 7-1 is a block diagram of the Emulator Test System.

7.2.1.1 Hardware Implementation. The Emulator Test System will consist of an IBM XT Personal Computer connected to the Harris H800 computer over an RS-232C serial communications line. A standard CRT terminal will also be used. Communications between the H800 and IBM XT will be full duplex at 19,200 bps. The IBM XT will contain an I/O board providing 16 analog input channels, 2 analog output channels, and 24 DI/DO lines. Of the 16 analog input channels, four will be dedicated to the collective, cyclic, and pedal controls, leaving 12 channels available for configuration for specific tests.

The analog output channels and DI/DO lines will not be used initially but the software will be written to accommodate their possible use. The IBM XT keyboard will be used to replicate cockpit switches; each key can be configured as a "one-shot" switch, a toggle on/off switch, or an increment/decrement button for a CDB analog input variable. Certain "hardwired" outputs, such as aircraft track, attitude, altitude, and velocity will be displayed on the low-resolution graphics monitor connected to the IBM XT. The numeric values of other variables will be displayed on the standard CRT screen. The value of any CDB analog or digital input variable may be altered by entering its offset and desired value using the CRT keyboard.

7.2.1.2 Software Implementation. The Emulator Test System software in the H800 will run as a separate real-time program with a priority lower than the Main Trainer Program. It will receive control during the idle time at the end of each frame. With full access to the CDB via Monitor Common, it will use the frame cycle counters relocated from the HAA module to the CDB to synchronize its operation with the Main Trainer Program.

Communications with the IBM XT will be master/slave. The IBM XT will transmit a block of input data upon reception of a block of output data from the H800.

As much of the computational work as possible will be performed by the program in the H800; the software in the IBM XT will primarily handle communications, emulator input, and graphics display. An input file containing the symbolic names of CDB variables, the type of variable, biasing and scaling factors, and associated IBM XT inputs will enable the H800 program to insert input values from the IBM XT into the appropriate CDB variables. Similarly, another file will contain the symbolic names of the CDB variables to be displayed on the emulator Test System CRT.

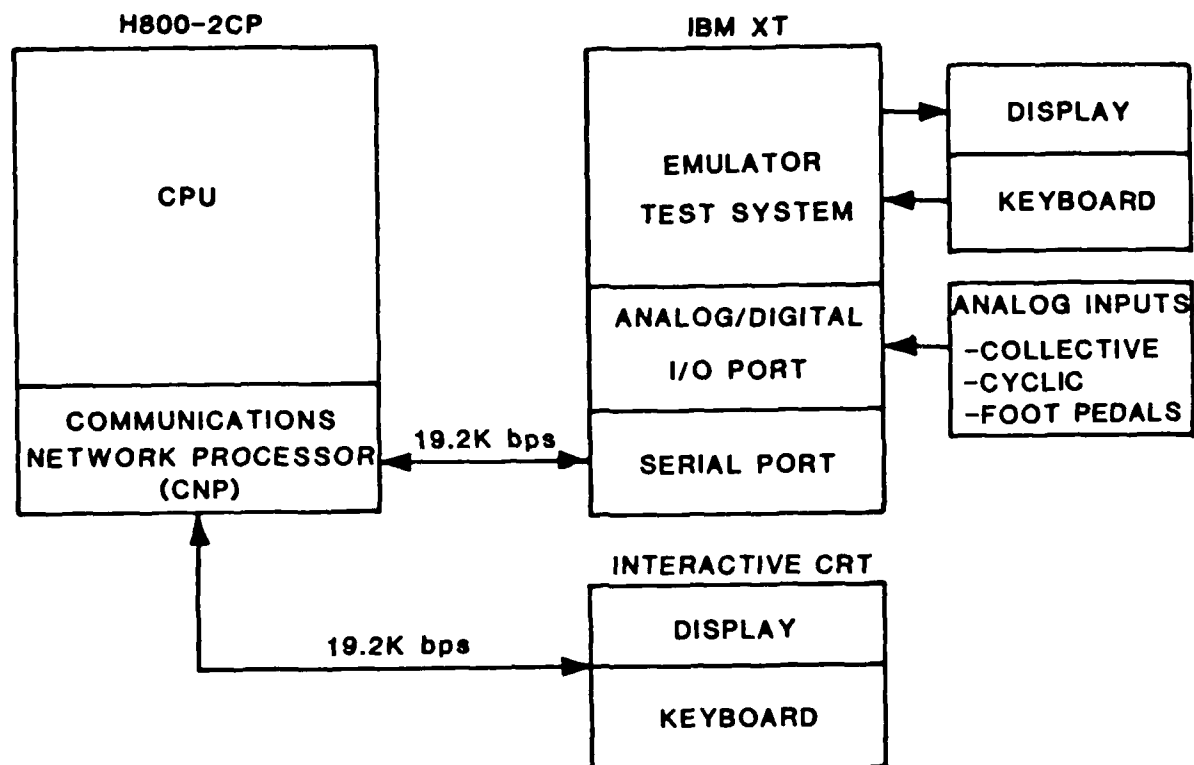
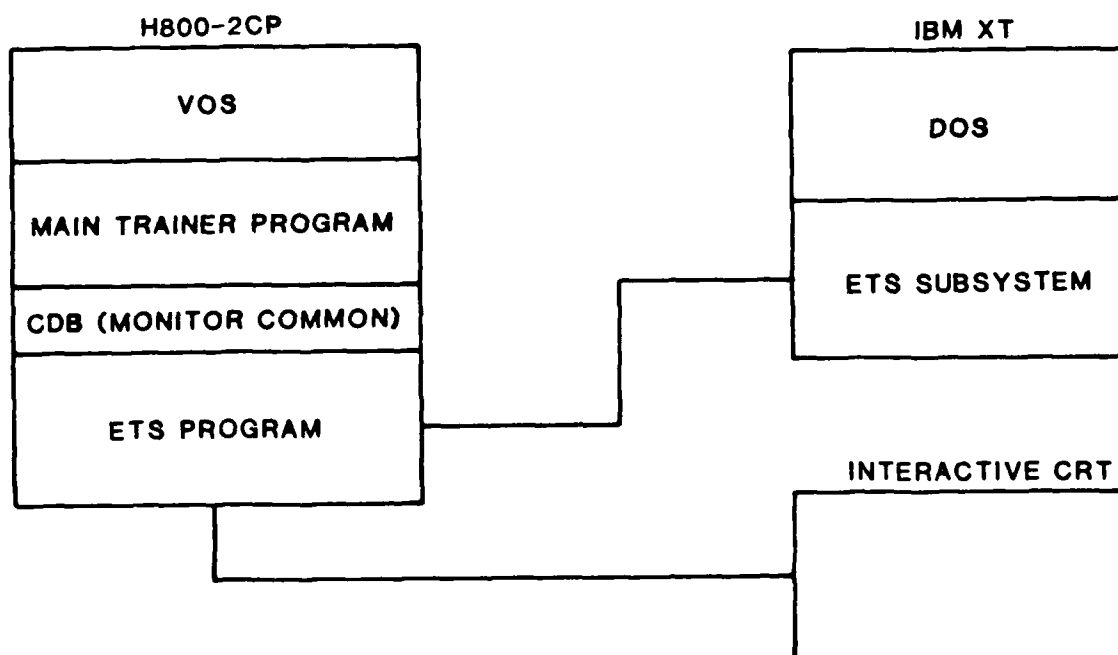
HARDWARE CONFIGURATION**SOFTWARE CONFIGURATION**

Figure 7-1. Emulator Test System block diagram.

7.2.2 System Test Driver

The System Test Driver is a tool designed to enable real-time testing of new or modified simulator software. It automatically "flies" the simulator, recording values of user-specified CDB variables as the flight progresses, for analysis or comparison to other flights. Identical flights may be executed at any time, allowing regression testing to determine the effect changes to simulator software have on simulator response.

There are three operational modes associated with the System Test Driver: Profile Generation, Rerun, and Analysis.

During profile generation the values of the CDB variables used to drive the simulator software during later reruns are recorded. These stimulus variables will be obtained by flying the simulator using the trainer cockpit or the Emulator Test Stand. As the simulator is flown, the values of the stimulus variables, usually the positions of the primary flight controls, will be recorded. Although associated primarily with the Rerun Mode, a list of response variables (CDB variables) may also be specified during profile generation. The values of these response variables will simultaneously be recorded for later analysis and displayed on the CRT.

In the Rerun Mode, the System Test Driver reads the previously recorded stimulus values to drive the simulator software. Values of up to 20 response variables are simultaneously recorded and displayed to the CRT.

The Analysis Mode is performed offline. The System Test Driver analysis tools enable printing and comparison of response files.

7.2.3 REBUG

The REBUG tool greatly eases the debugging of real-time programs. It provides all of the commands of the standard Harris DEBUG utility, including the capability to set breakpoints, examine and modify memory and CPU registers, and dump memory. It also provides additional commands to conveniently enter and examine data as well as the ability to interface to user-written debugging subroutines.

REBUG is activated by toggling a switch register (on the Harris H800 computer system, this is accomplished by entering a command at the MAP terminal). Called during each frame by HQA, the module HQDEBUG will check the switch register to determine whether to call REBUG. REBUG, in turn, may call user-written debugging subroutines, named REBUG1 through REBUG4.

REBUG1 is invoked whenever REBUG is entered -- either as a result of toggling the switch register or when a breakpoint is encountered. The default version of REBUG1 stops the real-time clock, providing the capability to freeze program execution and examine or modify the contents of any memory location between frames. REBUG2 is invoked just before leaving REBUG, usually by entering the "CONTINUE" command. The default version of REBUG2 causes the real-time clock to resume counting from the point just before REBUG was entered. If, while in REBUG, the user enters the "EXIT" command, REBUG3 will be executed and both REBUG and the Main Trainer Program are aborted. The default version of REBUG3 does an immediate return.

Entering the "U" command will invoke REBUG4. The default version of REBUG4 simply performs an immediate subroutine return, but it may be written, for example, to format and output various CDB variables.

Although not an intrinsic part of REBUG, the subroutine REBUG5 is also called by HQDEBUG during each frame execution if the flag DEBUG2 is set (set or cleared by the RMM or REBUG). The default version of REBUG5 is an immediate subroutine return, but it may be written to output status reports or to perform other similar functions, if desired.

8.0 QUALITY ASSURANCE

Contract No. N62269-84-C-0424 requires that the contractor prepare a separate Quality Assurance Plan, in accordance with CDRL Item A0006, and Data Item Description DI-R-2174A. As specified by Form 1423, this item will be delivered 150 days after contract award.

9.0 PROGRAMMING STANDARDS

9.1 Programming Guidelines

The following programming guidelines have been established to facilitate software maintenance and ensure functional operation. During the SH-2F WST conversion project, the programming standards of the original code will be matched in the new code to the greatest extent possible. All modules will have a header containing the following information:

- a. Module name, IDEN statement
- b. Description of module purpose and function
- c. Author's name and date of development
- d. Modification reason; problem and solution
- e. Modifying author's name and date of modification.

Comment lines will explain module logic. All modules will be structurally designed and concise, to facilitate module maintenance.

Software to be developed and integrated as part of the SH-2F conversion program will adhere to DOD-STD-1679A and the following standards:

- a. Code will be of a structured, hierarchic, top-down, modular design.
- b. Code procedures will remain fixed during execution. Self-modifying code is not allowed.
- c. Subprograms or subroutines will be coded as in-line procedures only when execution time or storage is critical.
- d. When possible, the same symbolic names and entry points used in the original system will be retained to increase clarity, uniformity, and ease of internal documentation.
- e. Where possible, all implementation-dependent parameters or compile parameter options should be grouped and identified.
- f. In a program, subprogram, or subroutine where a data storage location has more than one name, use only one of the names throughout the program.
- g. When doing jumps or branches, observe the following rules:
 - 1) Jumps will be permitted to named locations only
 - 2) Jumps to current location references are not permitted
 - 3) Jumps or branches into conditional blocks are not permitted.

- h. Define named constants and compiler parameters to indicate purpose, rather than value. Use separate constants (parameter names) for different purposes, even if some of these have the same values. Avoid the use of numbers whose meanings are not implied by their value.
- i. Use the logical FORTRAN constants TRUE and FALSE for true/false tests.
- j. Avoid using program labels as arguments passed from one program to another.
- k. Identify which function is being satisfied in the code by comments, even when a module satisfies only one function, for Q.A. audit traceability.
- l. Conform development/conversion coding to top-down, structured testing phases.
- m. Code stubs for real-time checkout that consume proper time durations (where timing is critical).
- n. Avoid in-code documentation redundancy.
- o. Declare variable mode (i.e., integer, real, double precision).
- p. General-purpose modules shall have data lists included in header comments.
- q. The initials of the cognizant programmer will appear in columns 71-73.

Deviations from these standards shall be approved by the project manager. Deviations shall be described by comments in the code.

9.2 Modularity Guidelines

The following modularity guidelines apply to executable code:

- a. A module is a separate compilation (subprograms, programs.)
- b. A module must have a name.
- c. A module must return to its caller.
- d. A module may invoke other modules.
- e. A module must have a single entry and exit point.
- f. A module is relatively small in size (maximum, 200 lines of code; average, 100 lines.)
- g. A module should not keep a history for the purpose of modifying its action or logic path.
- h. A module must have a single function.
- i. The number of possible paths through a module should be minimal.

10.0 CONFIGURATION MANAGEMENT

This section discusses the configuration management policy that will apply to the conversion of SH-2F WST software. This policy covers identification of software modules, tracking of changes made to the software, and associated documentation.

10.1 Identification

Because this project principally involves the conversion of existing software, the identification scheme used for the existing software modules will be retained by Eyring. The current scheme uses the "IDEN" statement of the Harris assembler language. This statement identifies the SH-2F simulator system (2F106), the module category (program category designator), the number of the module, the revision (Rxx) and the date, the module's language, the mnemonic name, and provides a short description of the module's function.

The "IDEN" statement will be included in all new modules developed by Eyring for the simulator. To identify these modules, the letters "ERI" will be inserted into columns 71-73 of the "IDEN" statement. The remainder of the statement will retain the same format. If a new module is developed to replace an old module, the "IDEN" statement will be carried forward to the new module. If a new module is developed which does not replace an old module, a new "IDEN" statement will be created in the established format.

Prior to system delivery and installation, the revision number of all system modules will be standardized; i.e. the new revision number will be the highest current revision number rounded up to the next multiple of ten. For example, if the highest revision is R34, R34 will be changed to R40 and R40 will then appear on all modules. This will indicate that a conversion of all the software modules has been completed.

10.2 Tracking Changes to the Software

This section discusses the procedures to establish and protect a baseline for the simulator software provided by NADC, make global changes to existing modules, develop new modules to replace existing modules, and make changes to modules. To keep track of changes, the modules will be placed under the control of VISTA (VOS Integral Tracking and Analysis program). For a more detailed description of VISTA, refer to paragraph 10.4 and Harris manual 0869007-000.

The "baseline" software provided by NADC on magnetic tape will be copied to disk files in a protected area, each module configured as a separate file. To protect the baseline files,

all files will be designated public "read only." The files may, under supervision of the configuration specialist, be copied from the baseline area to a work area where all modification work will take place. The work-area files will have public read and write access but only the owner of the files, the configuration specialist, will be able to delete the files.

Because this project is principally a conversion rather than a development effort, most changes to existing software will be global; i.e., many modules will undergo identical modification such as changes in syntax and removal of "HOLD" statements. This will be accomplished by developing a program to perform these changes in a batch-type operation. When all global changes have been made the converted modules will be placed under the control of VISTA. Programs used to effect global changes will be maintained in a "tools" area that will contain the program file and a comment file. The comment file will contain information about the function of the associated program and instructions on how to run the program.

If it becomes necessary to rewrite or replace an existing module, Eyring project personnel will complete a Software Change Proposal (SCP) form to document which modules are being rewritten and why. New modules will be developed in the programmer's work area. When the module successfully completes unit testing, it will be copied from the programmer's area to a configured area and placed under VISTA change control. This will constitute the first configured release of the new module.

As a method of tracking changes to individual lines of source code, programmers will insert their initials in columns 71-73 of the specific code lines that have been modified. This requirement will be enforced for both converted and developed modules following configuration of the specific module.

Refer to Figure 10-1 for a schematic depiction of the configuration process.

10.3 Documentation

The Software Change Proposal (SCP) and the Software Trouble Report (STR) are the two documents associated with configuration management.

The SCP will be used to identify those modules replaced by new modules. The configuration specialist will maintain a file of SCP's submitted by project personnel.

The STR will be used to identify programming errors or problems found during unit and integration testing. The STR's will be sequentially numbered, and this number will concurrently

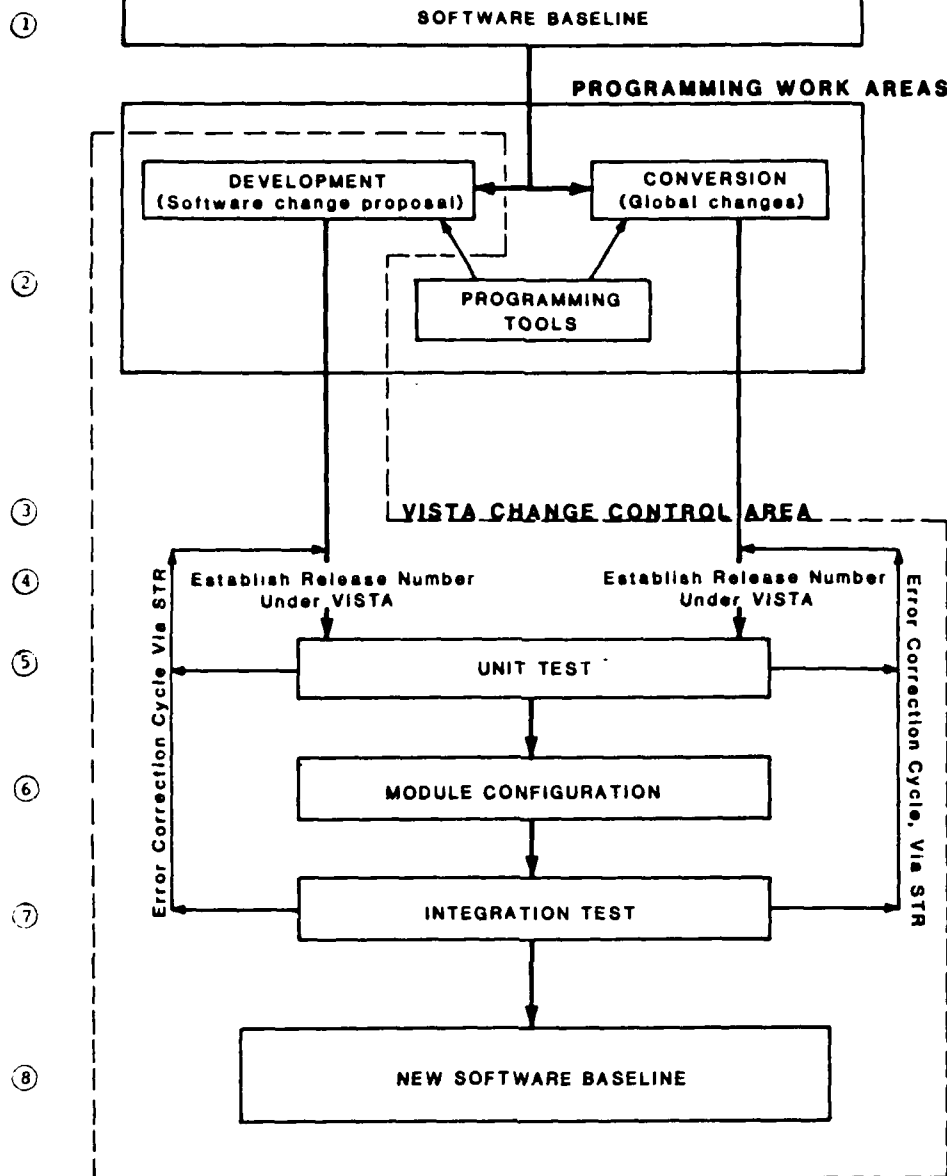


Figure 10-1. The Configuration Process

1. Baseline software is received from NADC. Magnetic tape is copied to disk and given public read-only access. Each module is treated as a separate file.
2. Project configuration specialist copies selected modules to programming work area(s). Programmer/analysts can read from and write to work-area files, but cannot delete these files. Global changes are performed on selected modules, using programs from "Tools" area. Redesign or rewriting of obsolete modules will be accomplished via Software Change Proposal and controlled by VISTA.
3. VISTA (VOS Integral Source Tracking & Analysis) will maintain a record of changes to converted and developed modules by requiring that specific change information be entered following each editing session.
4. When configured under VISTA, a file (module) is assigned a variant number; for example, VE=1:0:0. The variant number and file name uniquely identify the file. The first number is changed when it becomes necessary to customize software. The second number is changed following successful completion of testing. The third number is changed after each TX editing session.
5. Unit test is performed when programmer/analyst determines the module satisfies design or functional requirements. Deficiencies found during unit testing are recorded on Software Trouble Report (STR).
6. Following unit test, module is copied from programming work area to configuration area. Converted module will have release level incremented by one (VE=1:1:0); developed module will be considered as first release at this point (VE=1:0:0).
7. Integration test determines relocatable code links properly with other system modules.
8. Modules successfully passing integration test become part of new, deliverable software baseline.

be used as the change code for recording changes under VISTA. A description of the problem(s) will appear on the report, as well as in the VISTA file. The STR will be used to ensure that all problems have been corrected before the module is retested. The configuration specialist will maintain a file of the STR's, divided into sections for unresolved and resolved STR's. This file will also be used to generate any needed status reports.

10.4 VISTA

VISTA is a Harris CSD commercial program designed to maintain a history of changes and releases made to specific files of source code. VISTA files may only be modified using the Harris Text Editor (TX). When an editing session is completed and before the file is updated, VISTA requests that a six-character change code and one-line change description be entered. The description will contain the initials of the programmer making the change. When the change involves a Software Trouble Report (STR), the change code will match the number of the STR and the description will match the error description on the STR.

Because only TX can be used to edit VISTA files, and TX cannot be used in batch mode, global changes must be made before putting the files under VISTA control. The use of TX to make extensive, repetitive changes of a global nature would be a time consuming, error-prone, and tedious process. New modules, however, will be under VISTA control from the outset of development.

After a module passes unit testing, a new release will be made using VISTA; i.e., the module will be "configured." This will occur whenever unit testing on a module is successfully completed. When an error is detected, either during unit testing or integration testing, an STR will be filled out. VISTA will then be used to track the changes made to the module, as explained above, and to generate a new release when unit testing is completed. This approach will provide a paper audit trail, as well as a programmatic source, for detecting, tracking, and resolving software changes.

11.0 GOVERNMENT-FURNISHED EQUIPMENT AND SERVICES

The following Government Furnished Equipment will be required during performance of this project. (Source code must be supplied on 1600-bpi, nine-track magnetic tape reels.)

<u>ITEM DESCRIPTION</u>	<u>DATE REQUIRED</u>
1) SH-2F WST system source code.....	5 DAC
2) Trainer Programming Report (Volumes 1 through 6) and Mathematical Model Report (Volumes 1 through 4).....	5 DAC
3) Initial Condition dump - two sets of IC's are needed to allow system testing of the trainer software. These sets are defined as:	
a. Aircraft on the ground, running	
b. Aircraft in flight @ approx. 5000-ft altitude.....	30 DAC
4) Scenario Data Bases - Includes the source code for each of the appropriate Problem World Data Bases.....	30 DAC
5) Trainer Test Pilots - A dedicated trainer test pilot, available for the duration of the Norfolk system integration, and for the duration of North Island system integration. Pilots will fly actual trainer missions so that the converted software can be verified and prior to government acceptance.....	Dates to be determined

12.0 SOFTWARE INTEGRATION

Integration of the SH-2F operational software VOS version into the actual SH-2F trainers at Norfolk and North Island will be performed by Eyring personnel. The converted operational software will be installed on the Trainer's H800 computer system running under VOS 3.1. To accomplish the integration, the following sequence of steps will be followed:

- a. Install the software on the trainer system disk
- b. Define the Trainer Services program
- c. Validate operation of RTP I/O
- d. Validate interface to graphic CRT controller
- e. Validate interface to alphanumeric CRT controller
- f. Validate interface to motion platform
- g. Execute Main Trainer Program
- h. Quantitative data collection
- i. Qualitative pilot testing
- j. Acceptance test
- k. Integrate FFI modules
- l. Quantitative data collection
- m. Qualitative pilot testing.

Eyring's tentative system integration schedule is presented in Figure 12-1.

Eyring will supply the SH-2F operational software VOS version on 1600-BPI, 9-track magnetic tape. The software source and job streams will be installed on the trainer system disk under appropriate accounts and access protection. The job streams will then be run to assemble or compile and link the Main Trainer Program and stand-alone utilities in order to validate the compatibility of the system used for the conversion effort with the operational trainer system.

To provide the interface between the Main Trainer Program and VOS, special system services are required and will be referred to as the Trainer Services program. Trainer Services will be interfaced to VOS by updating the System Service Director as

Eyring Research Institute, Inc.		PROJECT NO. N62269-84-C-0424		TITLE NADC - SOFTWARE INTEGRATION		SHEET 1 OF 1	
DATE 1 NOV 1984		ROSAEE MILLARD		PROJECT MANAGER			
MONTH		NORFOLK		NORTH ISLAND			
DAY		APRIL		MAY		JUNE	
-- SOFTWARE INTEGRATION --		1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17		15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31			
INSTALL SOFTWARE ON DISK							
DEFINE TRAINER SERVICES							
VALIDATE I/O							
VALIDATE GRAPHIC INTERFACE							
VALIDATE ALPHANUMERIC INTERFACE							
VALIDATE MOTION INTERFACE							
EXECUTE MTP							
QUANTITATIVE DATA COLLECTION							
QUALITATIVE PILOT TESTING							
ACCEPTANCE TEST							
INTEGRATE FFL MODULES							
QUANTITATIVE DATA COLLECTION							
QUALITATIVE PILOT TESTING							
PRELIMINARY ACCEPTANCE TEST							

Figure 12-1. Integration Schedule

detailed in Section 3.1.2 and by executing the OPCOM command "ADD SERVICE" to set up the internal pointers so that the service can be loaded from disk. These steps define the trainer services as user services to VOS, but as stated in Section 3.1.1, do not actually install them. They are not installed until the Main Trainer Program is executed, to prevent them from being accessed inadvertently.

Operation of the I/O interface will be validated by running stand-alone utilities DORT, BIT, and IOTST3. Since the RMM module is linked with DORT, RMM device operation will also be validated while executing DORT.

Interface to the graphic CRT controller will be validated by running the stand-alone utility PUTDSP to transfer data from a PPM file to the graphic CRT screen.

Interface to the alphanumeric CRT controller will be validated by running the stand-alone utility PUTALF to transfer data from a PPM file to the alphanumeric CRT screen.

Interface to the motion platform will be validated by running the stand-alone utility MOTST.

After completing I/O, display, and motion interface validation, the Main Trainer Program will be executed. Government-furnished test pilots will be required to operate the trainer for testing. To ensure accurate operation of the software, quantitative data collection will be done using the System Test Driver discussed in Section 7.2.2. REBUG may also be used for data collection as discussed in Section 7.2.3. Qualitative pilot testing will then be performed.

To complete the Eyring validation process, Eyring will conduct a preliminary execution of the computer test procedures that will be submitted as a separate document in accordance with CDRL item A009 and Data Item Description DI-T-2144A. These tests will include execution of all stand-alone utilities and diagnostics as well as the Main Trainer Program.

At any time during the validation process, if a discrepancy is discovered in the operational software, the appropriate correction will be made and the area retested.

When Eyring has completed the validation process, the Acceptance Test will be performed following the Computer Test Plan and Computer Test Procedures. Eyring personnel will conduct the Acceptance Test in association with Navy personnel.

12.1 Integration of Phase II Software

Government-furnished Flight Fidelity Improvement (FFI) software modules will be integrated into the SH-2F operational software VOS version. It is assumed that these modules will be supplied to Eyring for preliminary integration and testing prior to the on-site installation. To ensure accurate operation of the operational software Phase II version, quantitative data collection and qualitative pilot testing will be performed. When this validation process is completed, the Acceptance Test will be performed following the Computer Test Plan and Computer Test procedures.

12.2 Pre-Installation Testing

In order to eliminate as many trainer operational variables or anomalies as possible before software system installation, Eyring will require that the functional status of both SH-2F WST's be certified by the appropriate Navy personnel prior to Eyring's installation effort. The certification process will require that all known trainer operational deficiencies be listed and the list delivered to Eyring.

13.0 AREAS OF RISK

Eyring has determined that the following areas pose risk to contract schedule performance.

13.1 Input/Output Testing

Due to the unavailability of a complete SH-2F WST system for validation purposes prior to installation, it will be impossible to check all aspects of the various I/O interrupt handler programs before the converted software is installed at the trainer facility. Following installation, software or hardware anomalies or malfunctions may become apparent with subsequent adverse impact on the proposed installation schedule. Although Eyring will make every effort to validate I/O operations before installation, and will draw upon its past experience in resolving installation-related problems, this area remains a prime risk to schedule conformance.

13.2 Integration of Phase 2 Software

If it is found that Phase 2 software modules provided by NADC to Eyring for integration are deficient or do not conform with programming guidelines/requirements stated in CDRL Item A008, Eyring may elect to return those modules to NADC for modification. This may result in adverse schedule impact prior to or during system installation.

13.3 Delivery of GFE Items

As of the date of this document, Eyring has not received the Initial Conditions and Problem World data as requested in our Technical Proposal. In addition, the SH-2F WST software baseline was delivered to Eyring two weeks later than requested. Cumulative delivery delays of this nature may result in development/conversion schedule adjustments, with consequent slippage in contractual delivery schedules.

14.0 SCHEDULES AND MILESTONES

Eyring's overall schedule for obtaining, converting, and delivering the SH-2F WST software system is presented in Figure 14-1. This chart depicts performance of Statement of Work line items, broken down into project tasks and subtasks as shown in accompanying Table 14-1. This chart and table should be used in conjunction with the Contract Deliverables List presented in Table 14-2, and the resource allocation charts in Section 15.0 to obtain an overview of Eyring's planned project performance timetable.

[illegible]

Figure 14-1 (a), Sheet 1 - Line Item 0001

[illegible]

 Eyring Research Institute Inc.		PROJECT NO. N62269-84-C-0424		TITLE		NADC - Line Item 0001		Sheet 3 of 3 Date Sept. 24, 1984 Rogalene Millard Project Manager														
MONTH		OCT		NOV		DEC		JAN		FEB		MAR										
Pay period or week		4	15	22	29	5	12	19	26	3	10	17	24	31	7	14	21	28	4	11	18	
Beginning		15	12	19	26	3	10	17	24	31	7	14	21	28	4	11	18	25	4	11	18	
Milestones Tasks																						
Deliverables																						
Line Item 0001 - Task 5 Software Conversion																						
EMULATOR TEST SYSTEM		T15A																				
TEST DRIVER		T15B																				
CONVERSION TOOLS		T15C																				
EXECUTIVE MODULES		T15D																				
TIMER INTERPRET HANDLER		T15E																				
INITIALIZATION		T15F																				
CONCEPT CIB		T15G																				
CONVERT SIMULATION MODULES		T15H																				
I/O HANDLERS		T15I																				
FILE HANDLER		T15J																				
UTILITY PROGRAMS		T15K																				
DIAGNOSTIC PROGRAMS		T15L																				
CONFIGURATION OF SOFTWARE		T15M																				
VERIFICATION TESTING		T15N																				
START UNIT TESTING		M6																				
COMPLETE UNIT TESTING		M7																				
START INTEGRATION TESTING		M8																				
COMPLETE INTEGRATION TEST		M9																				
START VERIFICATION TEST		M10																				
START PERFORMANCE ANALYSES		M11																				
COMP. VERIFICATION TESTING		M12																				
COMP. PERFORMANCE ANALYSES		M13																				

Figure 14-1 (a), Sheet 3 - Line Item 0001

[illegible]

Figure 14-1 (b) - Line Item 0002

[illegible]

[illegible]

Table 14-1. Description of Project Tasks

Line Item 1.	SH-2F Weapon System Trainer Software Conversion.
Task 1.	<u>Line Item Management.</u> This task is used to manage line item 0001.
a.	Line item 0001 management.
Task 2.	<u>Project Startup.</u> This task is used to initiate the project. Subtasks include:
a.	Organize project team and make assignments. Establish project accounting system. Prepare project management plan.
b.	Establish programming baseline standards. Define conversion standards.
c.	Obtain baseline from NAVY and review. Obtain and review applicable documents referenced in SOW. Perform on-site review of current system.
d.	Prepare Software Development Plan (SDP). Prepare project performance schedule.
Task 3.	<u>Conversion Design.</u> This task is dedicated to the design of the trainer conversion software. Subtasks include:
a.	Prepare design approach.
b.	In-house design review.
c.	Prepare Software Development guidelines.
Task 4.	<u>Quality Assurance Support.</u> This task is used to prepare the Quality Assurance deliverables pertaining to SOW Item 0001. Subtasks include:
a.	Prepare Software Quality Assurance plan.
b.	Prepare Computer Program Test Plan.
c.	Prepare Computer Program Test Procedures.
d.	Prepare Computer Program Test Report.

Table 14-1. Description of Project Tasks (continued)

Task 5.	<u>Software Conversion/Integration.</u> This task involves the actual conversion and testing of the SH-2F trainer systems. Subtasks include:
a.	Modify emulator test system.
b.	Develop system test driver.
c.	Develop and refine conversion tools.
d.	Convert trainer executive modules.
e.	Develop timer interrupt handler.
f.	Convert trainer initialization routine.
g.	Convert Common Data Base.
h.	Convert trainer simulation modules.
i.	Develop system drivers (I/O handlers).
j.	Convert Put/Get file handler.
k.	Convert trainer utility programs.
l.	Convert trainer diagnostic programs.
m.	Software configuration.
n.	QA and verification.
o.	Norfolk integration.
p.	Norfolk acceptance test.
q.	North Island integration.
r.	North Island acceptance test.
Line Item 2.	Integrate Phase 2 -- Flight Fidelity Improvement (FFI).
Task 1.	<u>Line Item Management.</u> This task is used to manage line item 0002.
a.	Line item 0002 management.
Task 2.	<u>FFI integration.</u> This task includes the integration of NAVAIRDEVCON developed software changes and

Table 14-1. Description of Project Tasks (continued)

	pending corresponding updates to the quality assurance support documents. Subtasks include:
a.	Obtain and integrate FFI GFE Software.
b.	QA support of FFI.
c.	Update Software Quality Assurance plan to include FFI.
d.	Update Computer Program Test Plan to include FFI.
e.	Update Computer program test procedures to include FFI.
f.	Update Computer program test report to include FFI.
Line Item 3.	Integration Support for new Flight Control Loader (FCL).
Task 1.	<u>Line Item Management</u> . This task is used to manage line item 0003.
a.	Line item 0003 management.
Task 2.	<u>FCL integration support</u> . This task is to provide technical support for FCL software updates. Subtasks include:
a.	Technical Support to integrate FCL Software.
b.	QA and verification of FCL.
Line Item 4.	Prepare technical data and software program package.
Task 1.	<u>Documentation</u> . This task includes updating documentation affected by the software conversion. Subtasks include:
a.	Prepare Operator's Manual.
b.	Modify trainer programming report.
c.	Prepare verification and acceptance test documentation.
d.	Prepare program package.

Table 14-1. Description of Project Tasks (continued)

-
- e. Finalize program package.

Line Item 5. Prepare and Conduct Training Course.

Task 1. Train facility personnel. This task is to prepare and conduct facility personnel training. Subtasks include:

- a. Conduct needs analysis/assessment.
- b. Prepare course design.
- c. Develop training materials.
- d. Conduct training at Norfolk.
- e. Conduct training at North Island.

Line Item 6. CDRL Technical Data

Task 1. Generate Reports. This task is to generate the status reports, agendas, and minutes. Subtasks include:

- a. Performance and cost reports, agendas, and minutes.
-

Table 14-2. Contract Deliverable Items List

<u>Line Item</u>	<u>Task</u>	<u>Item</u>	<u>Scheduled Date</u>
6	1	Performance and Cost Report (A001)	18 Oct 1984
1	1	Software Development Plan (A004)	2 Nov 1984
6	1	Performance and Cost Report (A001)	15 Nov 1984
6	1	Performance and Cost Report (A001)	15 Dec 1984
6	1	Performance and Cost Report (A001)	15 Jan 1985
1	2	Software Development Guidelines (A005)	31 Jan 1985
1	3	Software Quality Assurance Plan (A006)	31 Jan 1985
6	1	Performance & Cost Report (A001)	15 Feb 1985
2	1	Software Quality Assurance Plan for SOW Item 0002 (A006)	4 Mar 1985
1	3	Computer Program Test Plan for Item 0001 (A008)	4 Mar 1985
2	1	Computer Program Test Plan for SOW Item 0002 (A008)	4 Mar 1985
1	3	Computer Program Test Procedures for SOW Item 0001 (A009)	4 Mar 1985
2	1	Computer Program Test Procedures for SOW Item 0002 (A009)	4 Mar 1985
6	1	Performance & Cost Report (A001)	15 Mar 1985
4	1	Program Package Document (A007)	1 Apr 1985
6	1	Performance & Cost Report (A001)	15 Apr 1985
6	1	Performance & Cost Report (A001)	15 May 1985
6	1	Performance & Cost Report (A001)	15 Jun 1985
1	3	Computer Program Test Report for SOW Item 0001 (A00A)	25 Jun 1985
2	1	Computer Program Test Report for SOW Item 0002 (A00A)	25 Jun 1985

Table 14-2. Contract Deliverable Items List (continued)

<u>Line Item</u>	<u>Task</u>	<u>Item</u>	<u>Scheduled Date</u>
4	1	Final Program Package Document (A007)	25 Jun 1985
4	1	Operator's Manual (A00B)	10 Jul 1985
4	1	Trainer Programming Report Updates (A00C)	10 Jul 1985
6	1	Performance & Cost Report (A001)	15 Jul 1985
6	1	Performance & Cost Report (A001)	15 Aug 1985

15.0 RESOURCE ALLOCATION

Eyring's personnel resource allocation projection for the duration of this contract is presented in Figure 15-1.

Figure 15-2 depicts Eyring's planned financial expenditures for the duration of this contract.

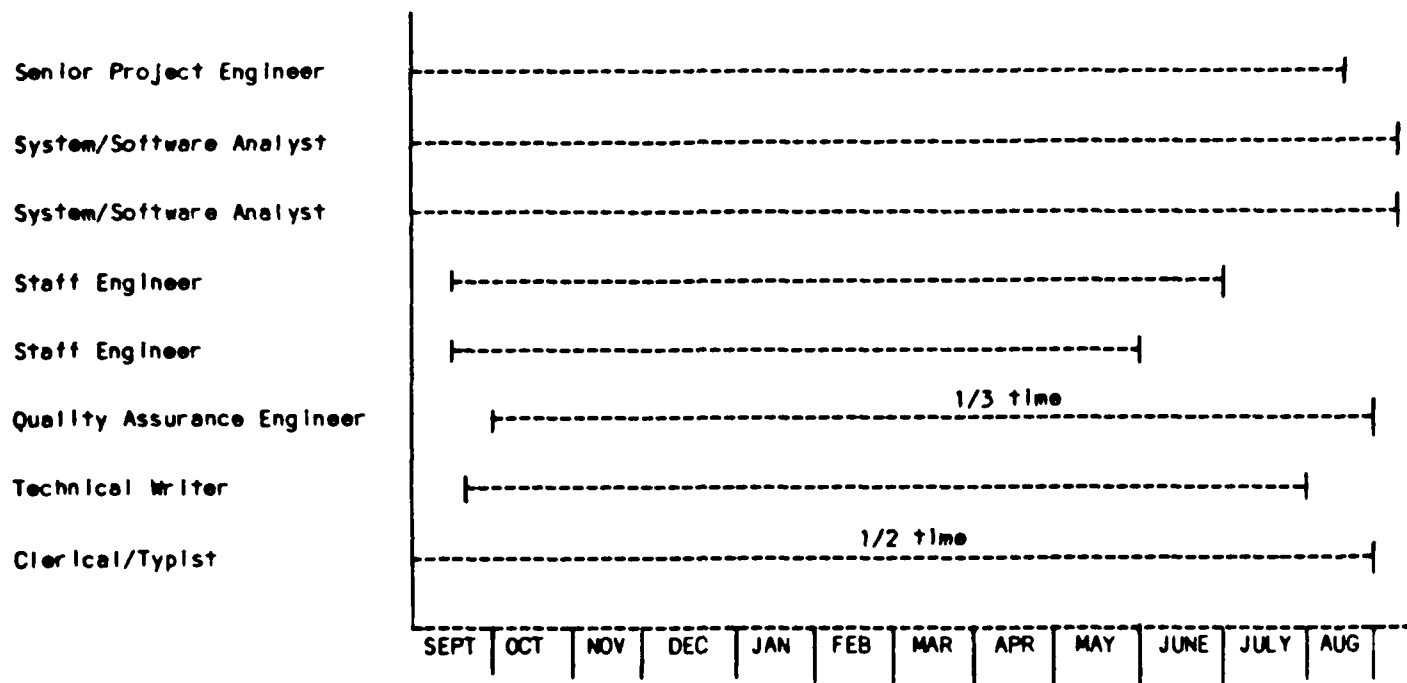


Figure 15-1. Resource allocation: manpower requirements vs. time

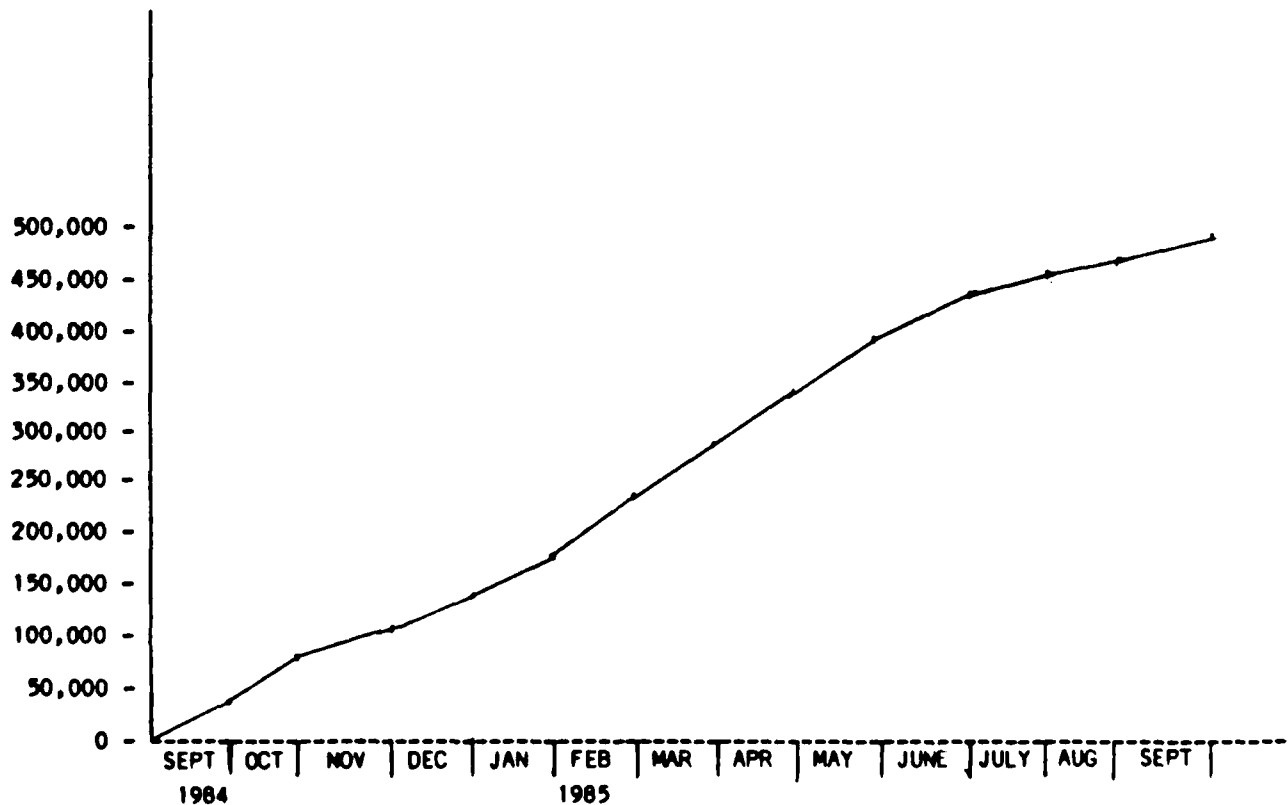


Figure 15-2. Resource allocation: expenditure vs. time

END

3-87

DTIC