WSRC-TR-91-70

ADVANCED SCIENTIFIC COMPUTING ENVIRONMENT GROUP NEW SCIENTIFIC DATABASE MANAGEMENT TASK PROGRAM PLAN (U)

FEBRUARY, 1991

Westinghouse Savannah River Company
Savannah River Laboratory
Aiken, SC 29808

Prepared for the U.S. Department of Energy under Contract DE-AC09-88SR18035

WSRC-TR-91-70

NRTSC
NUCLEAR REACTOR TECHNOLOGY AND SCIENTIFIC COMPUTATIONS

KEYWORDS: SCIENTIFIC COMPUTING
DATABASE MANAGEMENT
GRAPHICS TOOLS
JOSHUA, J70, J80, J90

RETENTION: PERMANENT

ADVANCED SCIENTIFIC COMPUTING ENVIRONMENT GROUP NEW SCIENTIFIC DATABASE MANAGEMENT TASK PROGRAM PLAN (U)

By

J. P. CHURCH

ISSUED: FEBRUARY, 1991

[illegible] 2-7-91
Derivative Classifier Date

SRL SAVANNAH RIVER LABORATORY, AIKEN, SC 29808
WESTINGHOUSE SAVANNAH RIVER COMPANY
Prepared for the U.S. Department of Energy under Contract DE-AC09-88SR18035

DOCUMENT: WSRC-TR-91-70

TITLE: ADVANCED SCIENTIFIC COMPUTING ENVIRONMENT GROUP NEW SCIENTIFIC DATABASE MANAGEMENT TASK PROGRAM PLAN (U)

TASK:

APPROVALS

M. R. Buckner 02/07/91

M. R. BUCKNER, MANAGER DATE:
SCIENTIFIC COMPUTATIONS SECTION

[illegible] 02/07/91

R. R. BECKMEYER, MANAGER DATE:
COMPUTING TECHNOLOGY GROUP

Westinghouse Savannah River Company
Savannah River Laboratory

WSRC-TR-91-70

Advanced Scientific Computing Environment Group New Scientific Database Management Task Program Plan

Objective

The broad, long term, goal of the ASCENT (Advanced Scientific Computing EnvironmenT) Group is to provide a computing environment that will let the scientist/engineer investigate the solution of problems by interacting with conceptual representations drawn directly from the scientific and engineering domains. In this environment the scientist/engineer (i.e., the "problem solver") builds the problem model with reusable virtual objects having associated attributes and behaviors, including any real or artificial constraints. The problem solver could then test the model by perturbing it interactively and observing quantitative (archived experimental measurements; simulated or computed data) and/or qualitative (trends, approximations) responses.

Examples

Some specific examples may help clarify these ideas. A thermalhydraulics analyst should be able to interact with his[1] desktop terminal to build a TRAC model of an SRS reactor by dragging icons of pumps, pipes, vessels, heat structures, etc., to assemble the completed model. The icons would contain the data, correlations, relationships, physics, etc. pertaining to the actual object they represent. The analyst could then specify a flow transient that reproduces the flow changes imposed during one of the recent 'L' reactor tests, compute the results, call up the archived experimental results and display them alongside the computed results, execute other approximate models for the same conditions (e.g., FLOOD code), and display those results to compare with the TRAC and experimental values. The analyst should be able to easily compute correlations and then display and compare trends of both experimental and computed results. Similar analyses could be performed using various reactor physics and charge design codes, Reactor Monitoring System data or Control Computer data, and simulator response.

Another example might be an engineer who needs to know the location of the safety rods in 'P' reactor. He would be able to interact with his desktop terminal to request a representation of a reactor facemap, choose a pull-down menu to select safety rods, and request a printout and/or screen display of the X-Y coordinates of all, or any subset of, the safety rods. If the engineer were analyzing the risk of safety rod failure,

[1] Throughout this document the use of words denoting gender is for convenience only. The reader should substitute the concept of "the problem solver" in such usage.

he could select a number of rods from the facemap display, either singly or by drawing a shape around a group, and choose a pull-down menu to indicate that those rods were assigned 'inoperable' condition. The engineer could then use screen icons representing computer codes and databases to build a procedure that would determine the reactivity worth of the perturbed safety rod complement, and, depending upon the series of codes and recursions specified, could complete the probabilistic analysis of risk. The engineer could interrogate the meteorological database, display a windrose for each stability class, use the mouse to select and perturb a portion of any windrose, and re-do the analysis. Alternatively, the engineer could interrogate the real-time wind data being gathered by the onsite weather towers, and examine alternate evacuation plans by selecting highways with associated attributes of carrying capacity (population source depletion capacity) to minimize both population and maximum individual dose following a postulated major core melt accident.

Proposed System

The proposed capability will require a cadre of hardware and software that trails only slightly the leading edge of technology. It seems clear that implementing the proposed environment at SRS will require:

1. A network environment that comprises supercomputers, graphics boxes, mainframes, clusters, workstations, terminals, and microcomputers (i.e., a full complement of clients and servers)
2. An operating system that is architecturally transparent, thereby permitting the applications code to run on any box that supplies the required computing power (memory, cycles, whatever).
3. A distributed database, with perhaps local caching, with a (perhaps multiple) database managing system that permits use and manipulation of relational, hierarchical, object oriented, GIS, et al, databases.
4. Well specified guidelines for modular software development that facilitate assembling complex problem models from simpler components and software reuse.
5. A critical mass of qualified technical people who can devote the majority of their time to implementing (as opposed to developing) the technology. (Implementation includes QA testing of prototypic and production environments.)

Benefits

The benefit of this proposed environment is that it will provide full flexibility to take advantage of the latest advances in hardware and software and, at the same time, maximize the ability to process scientific information and minimize the time required to develop products . It will let the scientist/engineer (S/E) function at the higher level of abstraction that is his actual area of technical expertise. And the newly hired S/E will be able to contribute much more quickly to SRS research and development programs.

- <u>Program Plan</u>

To reach this long term goal will require a step-by-step progression from the present assemblage of monolithic applications codes running on disparate hardware platforms and operating systems. The initial steps of this progression must acknowledge the existing progress in developing a 'new' JOSHUA system (hereafter called J80) to replace the old system (hereafter called J70) originally developed in the 1970's for scientific code development . The general scope of the project is to initially follow two parallel paths: one, to complete the transition from J70 to J80; and two, to build a J90 version of JOSHUA that is suitable for a distributed computing environment. Upon completion of the first path, personnel will be reassigned to complete and extend the second path. That extension will progress to the full scope described above.

The following is an outline of the initial components of this program:

1. Replace J70 with J80
2. Develop prototype of J90 distributed computing system based on Unix, X Windows, network computing hardware (heterogeneous environment), distributed databases, and distributed file systems.
3. Develop portable graphics tools
4. Initiate training to implement and disseminate above tools and methods.
5. Extend prototypic development of J90 and more advanced systems

These items are discussed in more detail below.

1. Replace J70 with J80

   A. Complete the conversion of all needed codes

      i. Identify codes remaining to be converted.

         a. Reactor Technology
         b. SRL (the latest version of some codes must be converted; e.g. AA3, GRASS, FLOOD, etc.)
         c. Other

      ii. Assess need for, and usefulness of, commercial codes for automatic conversion of old FORTRAN codes to FORTRAN 77 construction (e.g., FOR_STRUCT).

      iii. Convert those codes for which item 1.C.i below is not appropriate.

   B. Integrate J80 into QA procedures

      i. Identify subset of 'old' certified codes. (Not need to consider formal QA for codes not planned to be certified. Only have to get signoff by code owner or custodian that he is satisfied.)

      ii. Get formal agreement of Software QA group and/or code custodians to certify converted code on J80 if results of benchmark problems run on 'new' are within x % or y significant figures of same problems run on 'old' system.

iii. Code custodians to be responsible for pushing paperwork for 'new' versions of certified codes.

C. Initiate Training for Code Custodians and Programmers

i. Issue guidelines for converting J70 codes to J80. Explain the differences in how to code for the two systems. (e.g., "You used to do this in J70; instead, in J80, do this.) Guidelines for coding in J80 are already published.[1] May need to be updated for clarity.
   - Terminal users
   - Code custodians (conversion of J70 codes to J80)

ii. Classes for users.
   a. Lectures at Resource Center for special needs.
   b. Videotape of above lectures for all other needs.
   c. Handholding scheduled at individual's terminal or Resource Center.

iii. Issue programming standards for applications programmers
   a. Portable
   b. FORTRAN 77
   c. Modular coding: number crunching separated from I/O processing
   d. Echo I/O for QA (facilitated by JOSHUA templates)
   e. Include coding to warn if input records are missing.
   f. I/O as JOSHUA and/or ASCII (i.e., both methods in source code as hierarchical 'IF')

D. Documentation of J80

i. User's manual (include special section for former users of J70 to explain differences in coding for the two systems).

ii. Systems Programmer's Manual

iii. Invention disclosure for JED

E. Closeout and decommission the J70 system

i. Have system to identify all usage of J70
(Code: check for J80 conversion. User: check for J80 training.)

ii. Prohibit all usage of J70 for converted codes

iii. Set premature deadline for death of J70 and announce on sign-on screen. ('Premature' means J70 will be kept online for 2-3 months after the announced deadline, but will not be accessible without special authorization.)

2. Develop prototype of network computing system. This system will be architecturally transparent (hence, portable). Components and protocols to be selected based on emerging standards. Hardware dependent performance of the system will be anticipated, if necessary, to facilitate generality, transportability, and flexibility of the resulting system.

[1] DPSTM-GEN-36, "The JOSHUA User's Manual. (Sept., 1987)

This parallels IRM/ATA's development of site standards for distributed computing, graphical user interface, modular programs, and computer aided software development. It moves beyond the relational DBMS considered there and also considers hierarchical, object oriented, and other special databases.

A. First prototype based on current technology (Unix, X Windows, existing distributed file systems) and emerging site standards.

   The key to greatly increasing the computing capability at SRS is to move to a totally distributed computing system (parts of a problem being run on different hardware, accessing information from local and/or global databases). The first step is to focus on an essentially hardware transparent system. A result of this functionality is to be able to immediately take advantage of new compute cycles as soon as they become available. No coding or data changes by applications code users will be necessary.

   i. Use built-in Unix system functions to minimize amount of special coding for the JOSHUA system functions. Possibilities to consider include:
      a. Remote procedure calls
      b. External data references
      c. Unix system file server
   ii. Select alternative distributed file system servers, DBMS.
      a. Try alternative methods
         - hierarchical
         - relational
         - combination of hierarchical and relational (e.g., SIR)
         - object oriented
      b. Resolve file size considerations (block size, record = file, etc.)

B. QA for J90 prototype
   i. Verification and Validation Testing, other testing.
   ii. Documentation

C. Documentation of J90 prototype
   i. System Programmer's manual
   ii. Invention disclosures

D. Facilitate development of production version that provides:
   i. Robustness and reliability
   ii. Error checking

3. Develop graphics tools that are portable with a consistent (standard) user interface
   A. Uniform QA plan
   B. Based on C Language and Motif™ Graphic Tool Kit or X Toolkit
   C. Standard approach to an Applications Programmers Interface (API).
   D. Develop guidelines for interfaces, window management, etc.
   E. Develop tools and/or application codes (following is initial list).

i. Reactor facemap tool
ii. Reactor facemap application (display RMS or CC data)
  - Replace RMS with ARMS (based on standard relational database)
  - QA for ARMS
iii. X,Y plots
  - N dimensional data using any two dimensions to develop X,Y plots
iv. X,Y,Z plots
  - N dimensional data using any three dimensions to develop X,Y,Z plots
v. X,Y,Z,C plots
  - N dimensional data using any four sensible dimensions to develop X,Y,Z,C plots
vi. Vital Safety Function Monitor tool
vii. Vital Safety Function Monitor application
viii. Icon directed I/O
  - Templates for J80/J90

F. Documentation
i. Application's programmers manuals (tell how to use tools to make applications codes)
ii. Invention disclosures for tools
iii. Complete the QA documentation

4. Initiate training to implement and disseminate above tools and methods.

A. Visiting Application Programmer's Chair

i. Technology transfer: members of other SRS/SRL groups work side by side with ASCENT members for one month to develop applications programs using new graphics tools and new computing environments. These visitors, as they return to their regular assignment, accelerate the use of the new technology throughout SRS.

ii. The scope of this program is to accept one or two candidates for one month periods. Qualified candidates with well defined problems will be given guidance in applying the prototypic tools and environments to solve those problems. For example, the reactor facemap tool (discussed in 3.E.i, above) developed to display actual reactor data can also be used to display corresponding computed values during a numerical iteration, or transient, calculation (e.g. FLOOD[2] or ULTRASIM, respectively). Similarly, the X,Y plotting tool (3.E.iii, above) could be incorporated into, and used to update the capabilities of, many existing codes (e.g., ULTRASIM[3], AA3[4,5]). The Visiting

[2] J. P. Church. Computer Program for Calculating Equilibrium Flows and Reactor Damage with Reduced Cooling. DPSTM-130 (April, 1985).
[3] M. V. Gregory. User's Guide to the ULTRASIM Code. DPST-89-371 (March, 1989).
[4] J. A. Smith and J. P. Church. Technical Manual - Accident Analysis Computer Program . DPSTM-120 (February, 1990).
[5] J. P. Church. Supplement to DPSTM-120, Technical Manual - Accident Analysis Computer Program (U). DPSTM-120-1 (February, 1990).

Applications Programmers would make the appropriate changes to the applications codes

After the one month period, or when the task is completed , new candidates will supplant the previously accepted candidates .

iii. Qualified candidates must be knowledgeable about the appropriate technology (e.g., able to program in C and Unix, knowledgeable about VAX and/or IBM systems, etc.) This requirements for particular skills will be determined during a design review of the candidate's problem. The Resource Center will be used to provide the necessary training to bring the candidate up to the minimum level of skills necessary to successfully apply the ASCENT technology developments.

B. Seminars, demonstrations.

-

5. Extend prototypic development of J90 and more advanced system.

A. Extend Object Oriented considerations (I/O, applications kernels, etc.). Achieve uniform usage of object technology in all aspects of scientific computing including database, user interface, computational, and programming models.

B. Select and develop representative applications for the technology (GLASS, GRASS, 'L' Data, Reactor Monitor System data). Strong emphasis on visual representation and interactive access and manipulation of concepts drawn directly from the applications domains.

C. Choose (consistent with emerging site standards) database management systems, programming languages, graphical user interfaces, CASE tools, logic based knowledge tools.

D. Interact with applications programmers to rewrite existing, or develop new, computer codes based on the new technology.

- Structure codes into numerically intensive ('number crunching') parts and data management parts. Applications programmers deal with number crunching parts; ASCENT Group deal with data management.

  Data management can be divided into three interconnected tasks:

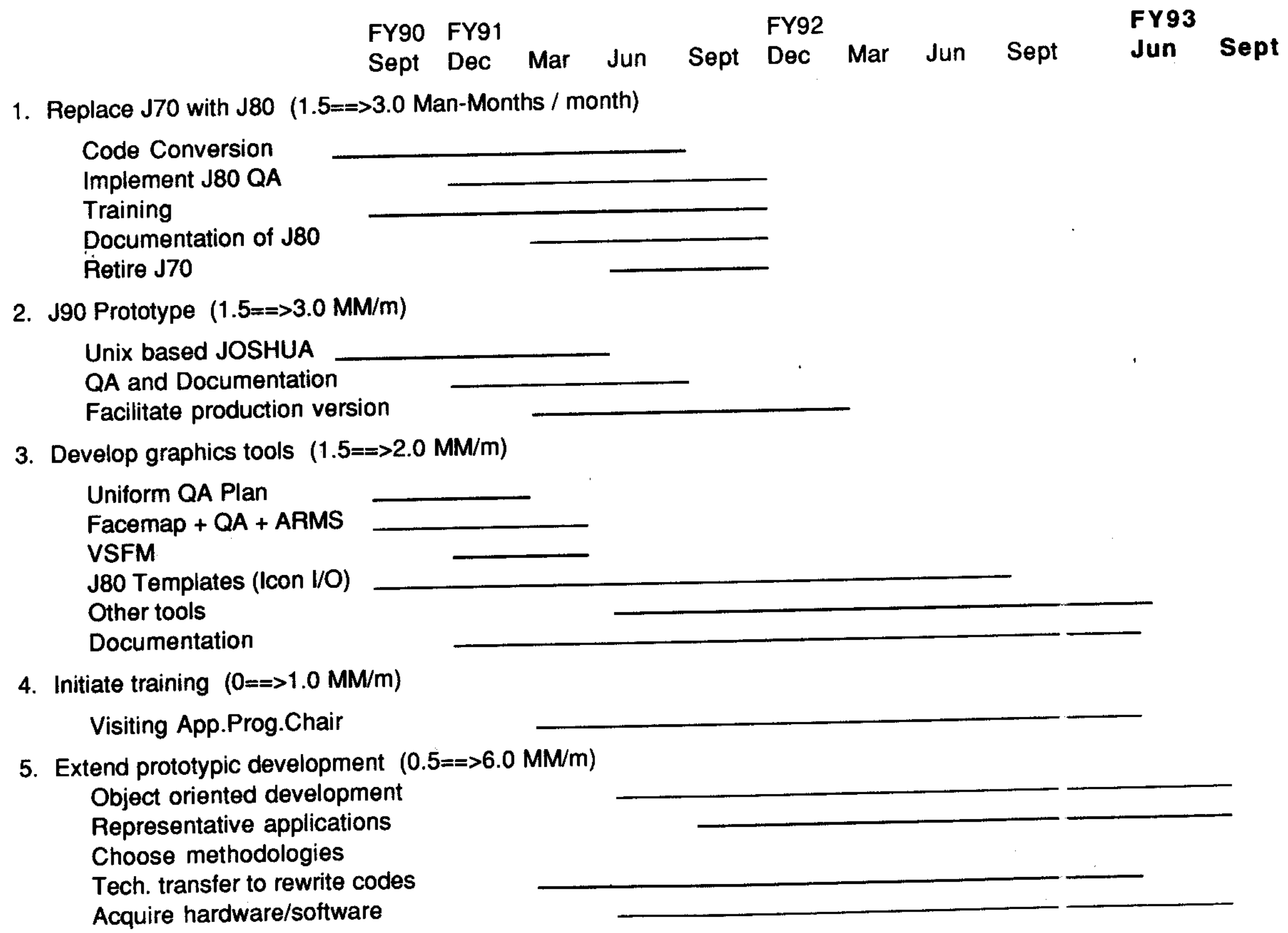
  a. I/O data management or symbol manipulation, which comprises the interface between the user and the I/O data base.

  b. Computational data management, which comprises the interface between the numerically intensive part of the computer code and the corresponding computational, or 'internal', data base.

  c. Interface, or flow of data, between the I/O data base and the computational data base.

E. Acquire latest computer hardware and software to develop above concepts.

- i. Graphics workstations (e.g., Silicon Graphics, Stardent)
- ii. Sun, DEC, IBM, NeXt workstations, with graphics options, capable of running X Windows clients (Unix and VMS workstations/servers with ample power; speeds > 5 MIPS)
- iii. Graphics displays capable of running an X Server. (Mac IIfx's, X terminals, high powered PC's, workstations)
- iv. Secure network capable of providing access to all users.
- v. X Windows and Motif running on all machines.
- vi. Mass storage systems
- vii. Distributed file systems that make mass storage capacity available to any cpu that needs it.
- viii. Develop a blanket procurement contract to lease sole source hardware and software on a rotating basis to evaluate the latest technology and provide recommendations to the scientific computing community at SRS.


FY90 Sept
FY91 Dec
Mar
Jun
Sept
FY92 Dec
Mar
Jun
Sept
FY93 Jun
Sept
1. Replace J70 with J80 (1.5==>3.0 Man-Months / month)
Code Conversion
Implement J80 QA
Training
Documentation of J80
Retire J70
2. J90 Prototype (1.5==>3.0 MM/m)
Unix based JOSHUA
QA and Documentation
Facilitate production version
3. Develop graphics tools (1.5==>2.0 MM/m)
Uniform QA Plan
Facemap + QA + ARMS
VSFM
J80 Templates (Icon I/O)
Other tools
Documentation
4. Initiate training (0==>1.0 MM/m)
Visiting App.Prog.Chair
5. Extend prototypic development (0.5==>6.0 MM/m)
Object oriented development
Representative applications
Choose methodologies
Tech. transfer to rewrite codes
Acquire hardware/software

CC: Ofc. Fin. Mgmt/Prog. Support, DOE, 703-A
J. R. Patterson, DOE, 703-A
R. K. Franks, DOE, 703-41A
T. M Tran, DOE, 703-41A
D. N. Bridges, DOE, 703-41A
R. T. Begley, 773-A
J. D. Spencer, 773-A
F. D. Knight, 703-A
P. R. Katonak, 773-51A
M. J. Hitchler, 773-A
A. J. Garrett, 773-A
W. E. Jule, 773-A
J. M. Stone, 773-41A
F. Beranek, 704-C
H. F. Allen, 707-C
F. D. Benton, 707-C
P. L. Ames, 773-59A
C. E. Apperson, 773-42A
R. R. Beckmeyer, 773-42A
E. L. Funderburk, 773-22A
J. R. Pelfrey, 773-59A
E. C. Randall, 773-22A
N. P. Baumann, 773-41A
J. P. Church, 773-62A
M. V. Gregory, 773-41A
R. L. Postles, 773-42A
T. G. Williamson, 773-41A
J. P. Howell, 773-41A
H. P. Olson, 773-41A
F. J. McCrosson, 773-41A
M. J. Morgan, 773-A
M. D. McKay, 723-A
J. P. Veldman, 723-A
J. D. Bickley, 723-A
M. Whitney, 730-A
T. Motyka, 773-A
R. N. Sims, 773-62A
J. C. Roberts, 773-62A
L. H. Sain, 707-C
D. W. Zimmerman, 707-C
J. W. Hammett, 707-C
J. E. Halverson, 773-63A
W. E. Bickford, 773-41A
D. R. Muhlbaier, 786-5A
L. A. Wooten, 773-41A
M. J. Ades, 773-41A
W. H. Horton, 786-5A
F. D. Ryder, 773-41A
D. C. Losey, 707-C
S. M. Revolinski, 707-C
J. F. Ortaldo, 704-S
H. H. Elder, 704-S
D. C. Iverson, 704-27S
M. W. Barlow, 703-25C
M. E. Maryak, 703-25C
G. A. Antaki, 703-24C
M. Zaslawsky, 703-25C
K. M. Vashi, 703-25C
H. E. Flanders, 703-24C
J. C. Cox, 707-H
E. K. Opperman, 730-A
M. R. Louthan, 773-41A
T. L. Capeletti, 773-A
M. W. Lee, 773-43A
D. A. Wheeler, 773-A
D. F. Bickford, 773-A
D. T. Rankin, 773-41A
S. K. Formby, 730-A
M. J. Plodinec, 773-A
C. R. Wolfe, 773-A
R. L. Adams, 703-A
G. P. Thompson, 703-A
R. T. Walters, 773-A
B. B. Westmoreland, 773-62A
A. O. Smetana, 773-62A
T. R. Arnold, 773-62A
SRL Records, 773-A (4)