

2

SEP 12 1994

35 Station 21

ENGINEERING DATA TRANSMITTAL

1. EDT 123308

2. To: (Receiving Organization)

3. From: (Originating Organization)

4. Related EDT No:

Distribution

Environmental Risk and Performance Assessment

NA

7. Purchase Order No:

NA

5. Proj/Prog/Dept/Div: Technology Applications

6. Cog/Proj Engr: D.W. Langford

9. Equip/Component No:

NA

8. Originator Remarks:

Supporting Document Approval Request. Comments Welcome

10. System/Bldg/Facility:

NA

12. Major Assm Dwg No:

NA

13. Permit/Permit Application No.

NA

14. Required Response Date:

NA

11. Receiver Remarks:

15. DATA TRANSMITTED

(A) Item No.	(B) Document/Drawing No.	(C) Sheet No.	(D) Rev No.	(E) Title or Description of Data Transmitted	(F) Impact Level	(G) Reason for Transmittal	(H) Originator Disposition	(I) Receiver Disposition
1	WHC-SD-ER-CSWD-004		0	VAM3D-CG Configuration Management Plan	1	1	1	1

SQ/ per Telecon JA Rawlins 9/18/94

16. KEY

Impact Level (F)	Reason for Transmittal (G)	Disposition (H) & (I)
1, 2, 3, or 4 see MRP 5.43 and EP-1.7	1. Approval 2. Release 3. Information 4. Review 5. Post-Review 6. Dist (Receipt Acknow. Required)	1. Approved 2. Approved w/comment 3. Disapproved w/comment 4. Reviewed no/comment 5. Reviewed w/comment 6. Receipt acknowledged

17. SIGNATURE/DISTRIBUTION

(See Impact Level for required signatures)

(G) Reason	(H) Disp	(J) Name	(K) Signature	(L) Date	(M) MSIN	(J) Name	(K) Signature	(L) Date	(M) MSIN	(G) Reason	(H) Disp
1	1	Cog./Proj. Eng DW Langford	<i>David W. Langford</i>	8/12/94	B5-25						
1	1	Cog./Proj. Eng. Mgr JA Rawlins	<i>JA Rawlins</i>	8/17/94	HO-36						
1	1	QA ML Hermanson	<i>ML Hermanson</i>	8/11/94	1-04						
1	1	Safety JC Van Keuren	<i>JC Van Keuren</i>	8/16/94	H4-64						
3	1	RT Hirano	<i>R. Hirano</i>	8/19/94	B5-25						
3	1	Central Files			18-04						
3	1	OSTI			18-07						

Signature of EDT Originator Date

Authorized Representative for Receiving Organization Date

204 N. ... 204 N. ... 9-1-94 Cognizant/Project Engineer's Manager Date

21. DOE APPROVAL (if required) Ltr No. [] Approved [] Approved w/comments [] Disapproved w/comments

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

RELEASE AUTHORIZATION

Document Number: WHC-SD-ER-CSWD-004, REV. 0

Document Title: VAM3D-CG CONFIGURATION MANAGEMENT PLAN

Release Date: 9/1/94

* * * * *

**This document was reviewed following the
procedures described in WHC-CM-3-4 and is:**

APPROVED FOR PUBLIC RELEASE

* * * * *

WHC Information Release Administration Specialist:

N.L. Solis N.L. SOLIS
(Signature)

9-1-94
(Date)

SUPPORTING DOCUMENT		1. Total Pages 23 24	
2. Title VAM3D-CG CONFIGURATION MANAGEMENT PLAN		3. Number WHC-SD-ER-CSWD-004	4. Rev No. 0
5. Key Words Code Custodian Configuration Management		6. Author Name: D. W. Langford <i>David Warren Langford</i> Signature Organization/Charge Code 86910/J812A	
APPROVED FOR PUBLIC RELEASE			
7. Abstract <i>9-1-94 J. Dolin</i> The VAM3D-CG computer code has been licensed for use at Hanford, from HydroGeologic, Inc., of Herndon, VA. Version 2.4b has been installed on the 3200GWW workstations, and is currently under configuration management. The purpose of this report is to describe the installation and configuration management of VAM3D-CG on the Hanford Computer System.			
8. PURPOSE AND USE OF DOCUMENT - This document was prepared for use within the U.S. Department of Energy and its contractors. It is to be used only to perform, direct, or integrate work under U.S. Department of Energy contracts. This document is not approved for public release until reviewed.		10. RELEASE STAMP <div style="border: 1px solid black; padding: 5px; text-align: center;">OFFICIAL RELEASE BY WHC DATE SEP 12 1994 35 Station 21</div>	
PATENT STATUS - This document copy, since it is transmitted in advance of patent clearance, is made available in confidence solely for use in performance of work under contracts with the U.S. Department of Energy. This document is not to be published nor its contents otherwise disseminated or used for purposes other than specified above before patent approval for such release or use has been secured, upon request, from the Patent Counsel, U.S. Department of Energy Field Office, Richmond, VA.			
DISCLAIMER - This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or any third party's use or the results of such use of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof or its contractors or subcontractors. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.			
9. Impact Level <i>X SQ per telecon JA Lawlins 9/8/94</i>			

VAM3D-CG CONFIGURATION MANAGEMENT PLAN

D. W. Langford

Table of Contents

1.0	PROGRAM FILE STORAGE CONFIGURATION	1
1.1	PROGRAM FILE STRUCTURE	1
1.2	FILE ARCHIVAL CONVENTIONS AND TOOLS	1
1.2.1	Source Code	1
1.2.2	Executable code	3
1.2.3	Test Problems	4
1.2.4	Script Files	6
2.0	QA MODIFICATIONS TO THE SOURCE CODE	6
2.1	DATA SET TRACEABILITY	6
2.2	BINARY/ASCII FILE LABEL	7
3.0	VAM3D-CG SOFTWARE IDENTIFICATION SCHEME	7
3.1	VERSION NUMBER	7
3.2	REVISION NUMBER	8
3.3	DATE OF ARCHIVAL	8
4.0	PROGRAM COMPILATION	9
4.1	COMPILATION OPTIONS	9
4.2	COMPILATION SCRIPS	9
5.0	PROJECT FILE	11
5.1	HARD COPY RECORDS (PAPER)	11
5.2	COMPUTER RECORDS (MAGNETIC MEDIA)	11
6.0	PUBLIC DIRECTORY FOR EXECUTABLE CODE	12
7.0	CHANGE CONTROL	13
7.1	NEW VERSIONS AND CORRECTIONS OF VAM3D-CG PROVIDED BY THE VENDOR	13
7.1.2	When the New Code Operates Satisfactorily	14
7.2	LOCALLY REQUESTED ENHANCEMENTS AND MODIFICATIONS	16
7.3	MODIFICATIONS TO THE ARRAY DIMENSIONS	17
7.4	MIGRATION TO A NEW OPERATING SYSTEM	19
8.0	TEST VERSIONS OF THE SOFTWARE	20
9.0	REFERENCES	21

1.0 PROGRAM FILE STORAGE CONFIGURATION

1.1 PROGRAM FILE STRUCTURE

The original version of VAM3D-CG¹ was delivered by the vendor in two source code files. The primary file included all of the FORTRAN common blocks and executable statements, with the second file containing the FORTRAN PARAMETERS. Additional source code files have been added to perform software QA and data traceability functions (see Section 2.0). More files may be created as the code matures.

1.2 FILE ARCHIVAL CONVENTIONS AND TOOLS

The source code is stored directly as text files, rather than using a configuration management tool. This should be adequate, since very few modifications should be made to the source code directly. Any significant changes should be made in bulk, by the software vendor.

One of the difficulties with VAM3D-CG has been the vendor's reluctance to change the version number on the software with each new release of the code. Therefore, each unique copy of the code is also identified by the date at which the source code was archived. This is used both in the source code, and in the directory names for code archival. Code identifiers are discussed in greater detail in Section 3.0.

Common File Storage (CFS) is used to store the program files, executable modules, and test problems. It may be accessed directly from the execution platform, and allows a directory tree structure with multiple levels. A detailed log of this information is maintained within the project file (see Section 5). Figure 1 depicts the storage configuration in CFS.

Four primary sub directories are used for file storing the source code, executable files, test files, and script files. These sub directories are described below.

The project file contains a listing of the files currently stored, and a brief description of each.

1.2.1 Source Code

Five primary sub directories are use to archive the source code, as follows:

¹VAM3D-CG is a proprietary software product of HydroGeologic, Inc.

1. Original code(s), as delivered by the vendor.
 - a. Each new version/revision of the code delivered by the vendor is placed in a separate sub directory. No changes should be made to this code.
 - b. All files used by a given version/revision are stored together, in the same sub directory.
 - c. The third level sub directory names indicate the code version (e.g., "ver2.4d").
 - d. Bottom level directory names indicate the date the code was received by the custodian (e.g., "jan5-94").
 - e. If any errors are subsequently found in the code, a "readme" or "comment" file should be added, indicating the nature of the error.

2. Temporary archival of new versions.
 - a. Newly received source code is archived here, after it has been modified for local use, and only if it differs from the vendor supplied code.
 - b. This sub directory is named "new temp".
 - c. The third level sub directory names indicate the version number (e.g., "ver2.4b").
 - d. Bottom level directory names indicate the date the code was archived (e.g. "aug13-93").
 - e. All files used by that version are stored together, including both source code and script files.
 - f. Once the code has been verified and archived, the lower level directories and their contents should be deleted.

3. Working source code.
 - a. Only versions/revisions of the code which were put into production should be included in this directory. This information archives the actual production source code. It will differ from the original vendor code when:
 - i. The code must be changed for the existing operating environment, or,
 - ii. One or more minor errors have been corrected locally (usually under the direction of the vendor).
 - b. Each version/revision of the code is placed in a separate sub directory.
 - c. All files used by that version are stored together, including both source code and script files.
 - d. The third level sub directory names indicate the version number (e.g., "ver2.4b").
 - e. Bottom level directory names indicate the date the code was archived (e.g. "aug13-93").

The most recent "working version" should duplicate the "current version". This is a backup of the current version, and is a historical record of the source code.

4. Current source code.
 - a. This saves the source code used in creating the working executable. It should duplicate the most recent "working version" of the source code.
 - b. Include all files required by the software, and any script files for compiling and loading the source code.
 - c. The source code should be taken from this directory when local modifications are made.
5. Utility software, which is not under configuration management, is stored in a separate source code directory. When a utility is placed under configuration management, the source code should be removed from this directory.

The PARAMETER file, named "vor.inc", sets the array dimensions for the software. It may exist in multiple versions, all of which may be current (see Section 3.2, Revision Number). When this happens, the file is stored as "vor.inc.xxx", where the "xxx" is a serial number for the file. Project file records indicate the serial number of the 'vor.inc' file which was used to generate each revision of the code.

A backup copy of the current source code is maintained in the project file, in a location separate from the CFS hardware. This ensures the source code is not destroyed by a single disaster. Where practical, test files are also backed up at a location away from the primary storage facility.

1.2.2 Executable code

Three primary sub directories are used, as follows:

1. The current working executable(s) must be archived.

Multiple revisions may be present. If so, they must should all be compiled from the same source code, except:

- a. The PARAMETER files ("vor.inc") may be different.
 - b. The revision numbers (Call to BUILDHDR, top of main routine) must correctly identify the current revisions.
2. Defunct executables are stored in a separate sub directory. These must be available in case a serious error is found in a program upgrade.

Third level directories indicate the version number of the code being archived. If differing executables exist, which have the same version number, bottom level directories indicate the date the corresponding source code was archived (from the 'working' source code directory).

Defunct executables may be deleted after two years of storage.

3. A third sub directory is present for saving utility software executables, as needed.

1.2.3 Test Problems

Several test problems have been acquired for VAM3D-CG. These have been divided into "test sets", which may include one or more problems. Many of these problems were used in verifying and benchmarking VAM3D-CG, and are described in the certification report (Lu et al., 1994).

All of the files used by a given test set are placed into a single sub directory. This includes input files, output files, batch execution files, and graphics files (e.g. postscript printer), where applicable.

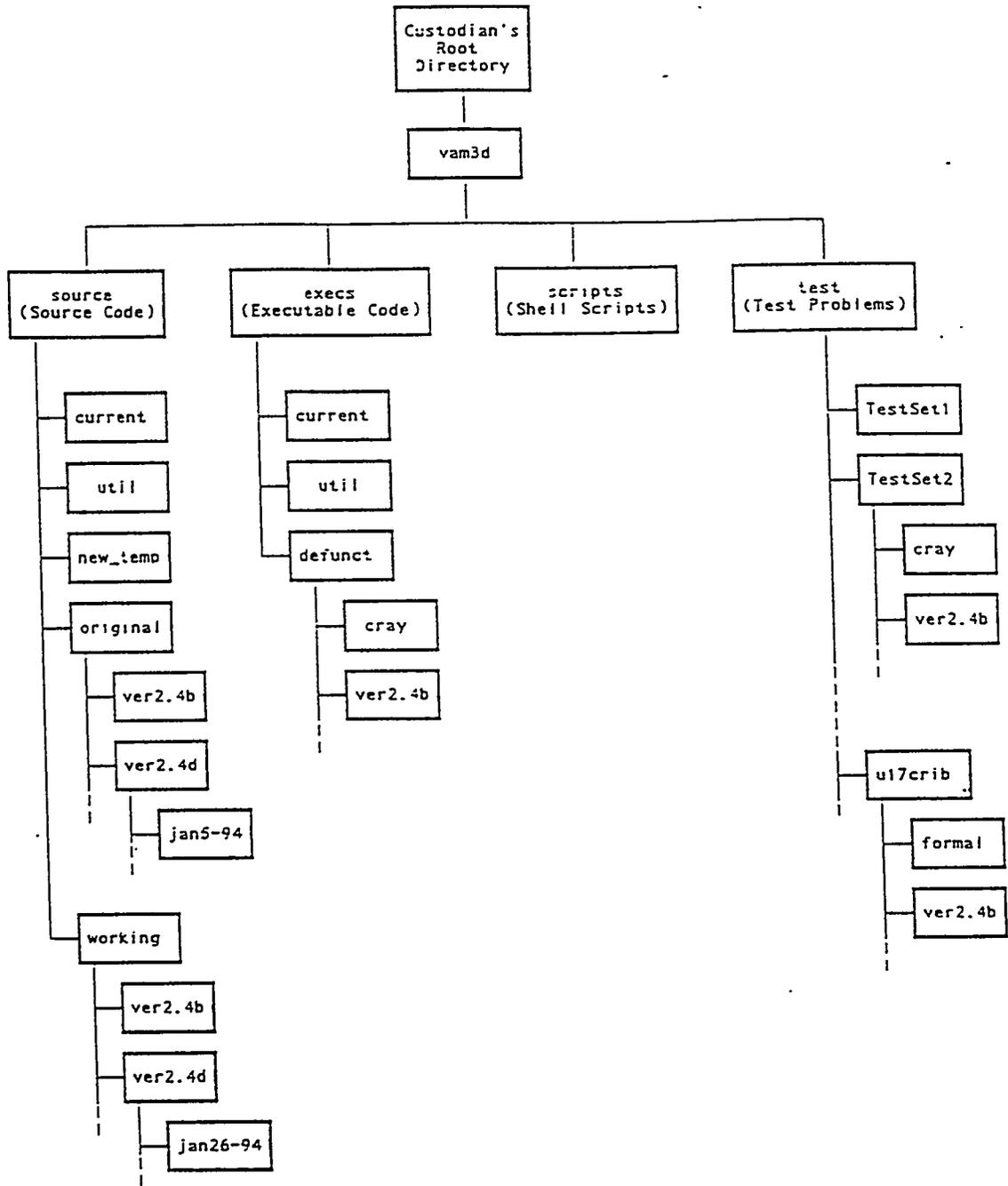
Different sub directories are used to store the results for different versions of the code. The results must be traceable back through as many versions as practical, as some errors may be tracked to their source if the data is available.

Test sub directory names are assigned as follows:

1. The second level test directory names indicate the name of the test set.
2. Third level directory names indicate the version of the software to which the results apply.
3. Bottom level directory names indicate archival date of the version being which was tested. This is taken from the "working code" sub directory name.
4. If a test set has been used for formal, published verification of the software, a bottom level directory, named "formal", is created to contain the I/O files specific to that verification effort. The "formal" sub directory may be further divided, if multiple formal documentations have been performed.

When a set contains several tests, the input and output files are placed into "tar" files. Batch files are created to execute all of the tests from a set in a single execution. Larger problems are stored individually, with I/O files saved directly as text files.

Figure 1. VAM3D-CG CFS Directory Structure



VAM3D-CG users should be granted read access to these files upon request.

1.2.4 Script Files

Various shell scripts for compiling, loading, and/or running the software may be present. These are duplicated in a separate directory.

2.0 QA MODIFICATIONS TO THE SOURCE CODE

As originally delivered, VAM3D-CG lacked some QA and useability features. The needed information is placed in an 80 character record at the beginning of the output file(s). That record is referred to as the QA header. Change Request #93-001, in the project file, describes this record in detail. A description of the required information is given below.

The QA header is designed to satisfy site requirements, in which software output must be traceable to the software which created the data (WHC-CM-4-2, QR 19.0, Rev 1, Section 4.2.5).

When a test version of the software must be created, the QA header must be replaced with wording that clearly indicates the code has not been approved. This insures the results from an experimental version of the code will not be taken as approved data.

New programming was required to implement these features. In particular, machine dependent coding was used for user identification and time/date functions (for which FORTRAN has no standard). The QA header I/O routines are placed in a separate source code file from the vendor supplied files. Machine dependent code is similarly stored in a separate file, to ease conversion of the code to a new operating system.

Future vendor releases of VAM3D-CG may include the QA header. This will eliminate the need to modify the code for this purpose.

2.1 DATA SET TRACEABILITY

1. The time and date of the run. This uniquely identifies the execution which created the data.
2. The name, version, and revision of the code. This identifies the name and version which created the data. The revision number is intended to distinguish between minor variations of the same software (e.g., operating system and dimension limits).
3. User identification, if accessible from the operating system. This identifies the person responsible for creating the data.

4. The most recent EDT and ECN numbers referencing the configuration and modifications of the source code. These identify changes which satisfy local requirements (e.g., QA header information), and are not strictly part of the VAM3D-CG numeric model.
5. The date of the source code archival. Normally, this will be the date when the most recent changes were made. This aids in tracking the changes, since the vendor occasionally changes the code, without modifying the version number.

2.2 BINARY/ASCII FILE LABEL

Executables created by some FORTRAN compilers cannot reliably distinguish between formatted and unformatted I/O files. VAM3D-CG writes a number of files which may be either formatted or unformatted. Therefore, the first character written to these file should be an 'A' for an ASCII (formatted) file, or a 'B' for a binary (unformatted) file.

3.0 VAM3D-CG SOFTWARE IDENTIFICATION SCHEME

Software is usually identified by a name (or mnemonic) and version number. This scheme is inadequate for VAM3D-CG, since the vendor controls the version number, and does not always assign a new version number for each software release. Also, multiple executables of the same version may be required, if the array dimensions must be modified for a particular problem.

However, all unique source code versions of VAM3D-CG must be identifiable, even though the version number is not changed. Coding changes are therefore identified through the version number, a revision number, and the date of archival. These are all within the QA header. The usage of these identifiers is discussed below.

3.1 VERSION NUMBER

VAM3D-CG is proprietary software. The vendor controls the program version number, which cannot be modified locally. In the source code, the version number is stored in four different places:

- a. Comments at the top of the main routine (file vam3d.f).
- b. Banner printout, at the top of the main routine (file vam3d.f). This output goes to the user's screen.
- c. Header line at the top of the main output file.
- d. Call to BUILDHDR, at the top of the main routine, second item in the call list.

3.2 REVISION NUMBER

The revision number was added to the source code locally. It distinguishes between versions with only non functional differences. The revision number is uniquely defined for each:

- a. Operating System (if allowed by license restrictions).
- b. Set of array dimensions.

The revision number usage is documented in the project file. For each revision, the following information is maintained:

- a. The date it was archived to the "working" code sub directory.
- b. The operating system.
- c. The serial number of the PARAMETER file ("vor.inc") used to create that revision, if multiple PARAMETER files are present.

The revision number is stored in one place in the source code:

- a. Call to BUILDHDR, at the top of the main routine, third item in the call list.

Revision numbers are three digits long. The first digit identifies the operating system, while the last two identify the array dimensions. For example, revisions 100-199 are run on the Cray, while revisions 200-299 are run on the SGI.

The serial number of the PARAMETER file, "vor.inc", is left to the discretion of the code custodian. Its primary use is to distinguish between PARAMETER files which must be archived in the same directory. Therefore, with each new release of VAM3D-CG, the serial numbers may start over at zero.

If more than one revision exists, documentation should be readily available to the users, which identifies the differences between revisions. This should be a text file in the executable (public) directory.

3.3 DATE OF ARCHIVAL

The vendor occasionally provides upgrades to the software, without modifying the version number. A scheme is required to uniquely identify each source code file. The date of archival in the "working code" sub directory was selected for this purpose.

This information is the last item on the QA Header. It is stored in one place in the source code:

- a. Initialization of variable QAREC, just prior to the call to BUILDHDR, at the top of the main routine.

4.0 PROGRAM COMPILATION

4.1 COMPILATION OPTIONS

VAM3D-CG is written in standard FORTRAN F77, with the exception of the time and date calls, and user identification (for which F77 has no standard). It has been a relatively easy code to install on the targeted platform. Commonly used compiler options for a model of this nature are as follows:

- PRECISION** All of the real variables in VAM3D-CG have been explicitly defined as REAL*8 by the source code. This accuracy is sufficient for modeling purposes. No compiler options are required to enforce double precision calculations, although these options are still normally used. This is the "-r8" option on the Silicon Graphics Inc. (SGI) FORTRAN compile command line.
- STATIC** Limited testing indicates that static memory allocation is required for proper operation of VAM3D-CG in the workstation environment. This is the "-static" option on the SGI FORTRAN compile command line.
- OPTIMIZATION** The available FORTRAN compilers have numerous optimization options. To date, several different levels of optimization have been used successfully. The default optimization level (the "-O1" option on the SGI compile command line) has been used for current versions of the executable code. The "-O2" option has also been used successfully.

4.2 COMPILATION SCRIPTS

The Unix utility MAKE has been used to build executables for VAM3D-CG. MAKE files exist for a variety of platforms. These are maintained in storage with the configured source code, although the MAKE files themselves are not under configuration management.

Because the bulk of the VAM3D-CG source code is stored in a single file, compilation can be easily done using simple "f77" commands. A simple Unix script file for compiling and loading VAM3D-CG is shown below. Note that the file "vor.inc" must be present with the other source code files.

```
# script to compile and load VAM3D-CG from source code files
#
f77 -c -r8 -O1 -static vam3dcg.f
f77 -c -r8 -O1 -static {...other vendor supplied files, if present..}
f77 -c -r8 -O1 -static id.f
f77 -c -r8 -O1 -static sgisys.f
cc -c idinfo.c
f77 -r8 -O1 -static vam3dcg.o id.o sgisys.o idinfo.o /usr/lib/libsun.a
```

The files "id.f", "idinfo.c", and "sgisys.f" are not required by the numeric model. They can be replaced by dummy routines which return blank filled character strings, without effecting the validity of the numeric results. However, one loses the traceability of the VAM3D-CG output files.

A dummy version of file "id.f" is available, named "idx.f". It replaces the QA header with dummy information, and should be used with versions of the VAM3D-CG source code which have not been formally qualified.

The procedure "idinfo.c" is written in C. Linking C object code with FORTRAN object code must be done according to the conventions of the operating system and compilers. Historically, some systems require the procedure name "idinfo" to be followed by an underscore character. On other systems, "idinfo" must be all capital, or all lower case, letters.

No special libraries are required by VAM3D-CG. However, on some Unix platforms the C procedure "idinfo.c" does not have default access to the required user tables. In particular, on the SGI platform, the code must be linked with the file "/usr/lib/libsun.a". If linked without this file, the code displays a segmentation fault error when run.

Historically, some of the workstation compilers have been unable to compile a large FORTRAN code in a single pass. Often, the finished executable produces faulty data, without any error messages or warnings being given. If this occurs, the Unix utility "fsplit" can be used to split the FORTRAN source code file "vam3dcg.f" into single routines, which can be compiled separately using an "f77 -<options> *.f" command. This has not been necessary for the current version of VAM3D-CG.

5.0 PROJECT FILE

A project file is maintained for the VAM3D-CG source code. This file contains all of the available records concerning the software, and includes both paper and electronic records.

The information maintained in the project file is discussed below.

5.1 HARD COPY RECORDS (PAPER)

1. Configuration Management Plan
2. Copies of the License Agreement(s)
3. User's Guide(s)
4. Correspondence concerning the software. Incoming and outgoing.
5. Change Request/Problem Reports (CR/PR) which are still open. This is a list of the known program deficiencies, which have not yet been resolved.
6. Change History (closed CR/PR's). For each change to the coding, this will include:
 - a. The CR/PR for each change made,
 - b. A description of the change, and
 - c. The programmer responsible for the change.

The original CR/PR for a problem will become part of the Change History file when the problem has been resolved.

7. Test documentation.

5.2 COMPUTER RECORDS (MAGNETIC MEDIA)

1. Backup copy of the source code.
 - a. Vendor original source code.
 - b. Current modified source code.
2. Logs of:
 - a. Change Request/Problem Report. This includes a unique number for each problem report, and the current status of the report (open or closed). For closed CR/PR's, the log also includes the EDT or ECN number which describes the changes, where applicable.

- b. Revision numbers currently assigned, and the unique feature(s) of each revision.
 - c. Document numbers for documents concerning the software.
 - d. Reservation status of the current source code files, when those files are 'checked out' for modification.
3. User List.
 4. Programmer(s) who have modified the code.
 5. Custodian(s).
 6. Related software (identify pre and post processors).
 7. A description of the hardware and system software currently being used to compile, load, and maintain VAM3D-CG.
 8. A list of computer and CFS directories for all files related to configuration management and change control.
 9. A brief description of each file stored.
 10. When multiple revisions exist, a short description of each current revision of the executable code is maintained. This information should be readily available to the users.
 11. Test problems are maintained on magnetic media. These are available for qualification of new versions and revisions of the software.

At least two copies of each magnetic media file are maintained, each on a separate disk. This information is maintained in a spreadsheet, except for the source code files.

6.0 PUBLIC DIRECTORY FOR EXECUTABLE CODE

VAM3D-CG, and its pre and post processors, have been installed on a public directory on the current operating system. This provides a single location for the executable codes, which the user's can access directly.

Users should have execute access to the executable and batch files in the public directory, and read access to any text information (e.g., "readme") files. The code custodian may require read/write access to the text files.

This public directory will change if and when new hardware is brought on line, and old hardware is retired. The current operating system and directory name for this directory is documented in the project file.

7.0 CHANGE CONTROL

Four events may occur which will necessitate change control for VAM3D-CG. These are (1) receiving a new version from the vendor, (2) locally requested modifications, (3) modifications to the array dimensions, and (4) migration to a new operating system. Each event requires separate change control procedures.

The actions required for each change process are described below. These are not intended to be strict procedural instructions. Instead, they are general guidelines which ensure all of the program records and information remain current through the change process.

Users should contact the code custodian when a problem is found in the code, and a written description of the error will be generated. The custodian will assign a Change Request/Problem Report (CR/PR) number to each such request, and will maintain a file on existing problems with the software.

The VAM3D-CG users, as well as the project management, must approve of the modifications before any changes are made to the software.

When a change is completed, an ECN must be generated to notify the users of the change, and describe that change. WHC Data Administration must also be notified of the change.

7.1 NEW VERSIONS AND CORRECTIONS OF VAM3D-CG PROVIDED BY THE VENDOR

Errors in the code are referred to the vendor, HydroGeologic, Inc., for resolution. On occasion, HydroGeologic may also provide upgrades to the source code, without any previous notification of errors in the software.

If the new version corrects an error which effects the integrity of previously computed results, all users must be notified.

7.1.1 Receiving New Source Code Files

1. Archive and document the delivery of the source code.
 - a. Save the code in the directory for "original" code (see Section 1.2), and on floppy disk in the project file.
 - b. Document the code in the project file.
 - i. Update the change history log.
 - ii. Update the "original" code file descriptions.

2. Modify the new source code, according to the local QA and other requirements, as needed. This may include:
 - a. The vendor supplied code file often contains the locally generated code (e.g., QA header and run identification coding), as well as the PARAMETER file(s). These must be separated from that file before compilation and linking.
 - b. The vendor often comments out the initialization of QAREC, and the call to BUILDHDR, at the top of the main routine. Calls to the time and date routines may also be commented out. These must be re-activated.
 - c. The parameters in the file "vor.inc", may not agree with those in common usage locally. These must be reset.
 - d. If multiple revisions of the code are required (See section 3.2), create corresponding versions of the PARAMETER file. Any other changes made to the source code should be carefully documented.
3. If any changes were required, create a new temporary sub directory (CFS) for the modified code ("new-temp" see Section 1.2).
 - a. Use the software version number, and the date of archival, as the lower level directory names.
 - b. Set the variable QAREC, at the top of the main routine, to indicate the date of archival. Use dummy information for the EDT/ECN references in that variable.
 - c. Archive the modified code in that directory. It remains there until it has been properly qualified.
4. Compile and load the new source code. Omit the QA header output in this version of the code.
5. Test the new code to determine that it performs as expected. Errors in the local (e.g., QA Header) modifications are to be corrected immediately. If other coding errors exist, contact HydroGeologic and indicate the nature of the problem. Let them install the corrections.
6. If any changes or corrections were made, write a short discussion of the error which was found (text file). Save that information with the original code provided by the vendor.

7.1.2 When the New Code Operates Satisfactorily

1. Generate an EDT or ECN to explain the changes to the users.
 - a. If possible, explain the differences from the previous version. (The vendor does not always provide this information, and the proprietary source code should not be examined any more than necessary.)
 - b. Document the test results.

- c. Indicate when the new code will be officially installed.
 - d. The EDT/ECN should be approved and routed before installing the new code. If this is not possible, notify all users of the change via a written memo.
 - e. Data Administration must also be notified of the change.
2. Re-compile the source code with correct entries for the revision number, EDT or ECN number, and date of last modification for the QA header.
 - a. If applicable, build multiple revisions of the executable.
 3. Archive the necessary executable files, test files, and source code files.
 - a. Previous executable files(s). This must be stored in the directory for defunct executables. A new sub directory may need to be created (see Section 1.0).
 - b. Test results. Create new test case sub directories, as required. Do not replace any existing results.
 - c. Working source code. Create a new "working" source code sub directory (see Section 1.0).
 - d. Save the new source code files into the directory for the current source code. (Note: the source code previously in that directory should already be archived in an existing "working" code directory, and need not be saved.)
 - e. Backup the current source code to a floppy disk in the project file.
 - f. If the source code has been archived to a temporary sub directory, delete those files.
 4. Replace the executable(s) in the directory for the current executable code(s).
 5. Log the required information in the project file:
 - a. EDT/ECN numbers.
 - b. If a Change Request or Problem Report (CR/PR) was involved, it must be logged and the CR/PR closed.
 - c. Update the change history log and revision logs.
 - d. Update the executable, working source code, and original source code documentation.
 6. Have the new executable moved into the public directory for VAM3D-CG related programs. The system manager may need to do this.

7.2 LOCALLY REQUESTED ENHANCEMENTS AND MODIFICATIONS

These are changes which are not properly the responsibility of the vendor. This would include such features as the QA record at the top of the output files. Changes which might effect the integrity of the computed data should not be handled locally, but should be referred to the vendor for resolution.

Most locally developed changes should be relatively simple, and should not impact the numeric modeling aspects of the software. More complex modifications may be developed locally. However, the vendor should perform the technical review, and implement the modifications into the controlled code. This minimizes the number of people, outside of the vendor's organization, who have access to the proprietary software.

1. When a modification is desired, a Change Request/Problem Report (CR/PR) must be submitted to (or by) the code custodian. This documents the existing problems, or desired enhancements.
2. When a CR/PR has been received, the following actions are taken:
 - a. If the change involves a software error, verify that the error is in the software, rather than the input data.

For errors which might effect the integrity of the computed results, contact the vendor (See Section 7.1).

- b. Save a copy of the CR/PR in the project file (Open CR/PR's).
- c. Distribute the CR/PR to the Project Manager and the Programmer(s). Get a consensus on whether the change should be made locally, by the vendor, or should simply not be made. Get management approval for any changes.

If the changes are not approved, close the CR/PR, or leave a copy in the project file, under 'Known Bugs'.

- d. Retrieve a copy the source code from the "current" source code directory.
 - i. Replace the QA header EDT/ECN/date information with a notice that indicates the code has not been approved.
 - ii. Mark the code as 'reserved' in the project file.
- e. Deliver the required source code to the programmer(s):
 - i. Source code file(s) to be modified.
 - ii. For the source files which will not be modified, and which are not needed, provide only object modules.
 - iii. Shell scripts required to compile/load the code.
 - iv. Any test problems needed and/or requested.

3. After the Changes Have Been Made, the following actions are taken:
 - a. Ensure Proper Verification and Documentation of the Changes. The programmer(s) must provide:
 - i. Corrected source code.
 - ii. Documentation describing the changes which were made. Include the names of the modules and procedures which were modified, the programmer, the date of the changes, and a description of the changes.
 - iii. Discussion of the tests which were run, and the results of those tests.

The documents may be combined into a single document, if appropriate. Place them in the Change History (closed CR/PR's) section of the Project File. They should also become part of the EDT/ECN describing the changes.

- b. Ensure the new source code is properly commented.

A description of the changes must be included in the module header, and in the header of each routine which was modified. This must include the programmer, date, and a (very) brief description of the change(s).

- c. If any changes were made beyond the scope of the original CR/PR, get management approval of the changes.
 - d. Rebuild and test the executable.

At this point, the procedure duplicates that for new source code received from the vendor (Section 7.1.2). However, there are some additional steps, as follows:

- e. In the project file source code directory:
 - i. Unreserve source code file(s).
 - f. Notify the vendor of the changes. In particular, note any modifications to the user manual. The vendor will usually want a copy of the new source code.

7.3 MODIFICATIONS TO THE ARRAY DIMENSIONS

The program dimensions may need modification for particular models. VAM3D-CG uses an external INCLUDE file to set the array size parameters (the PARAMETER file, "vor.inc"). Therefore, only the PARAMETER file needs to be changed.

Because of the design of FORTRAN, changes to the array dimensions are not generally considered modifications to the software. This should be the case for VAM3D-CG. Therefore, no ECN needs to be created for these changes. However, the user must still have access to documentation on the revisions which are available.

The program revision number is used to distinguish between executable codes with differing array sizes (and operating systems). To track these changes, each PARAMETER file is given a serial number. In the project file, the serial number of the PARAMETER file will be documented for each revision of the code. (See Section 3.2)

The code custodian must make the changes in the array size parameters. The steps required are as follows:

1. Determine:
 - a. The new revision number for the modified code.
 - b. The new serial number for the PARAMETER file ("vor.inc").
 - c. Document these numbers in the project file.
 - i. Software revision numbers. Include the serial number of the PARAMETER file in this log.
 - ii. PARAMETER file serial numbers.
2. Make a copy of the PARAMETER file.
 - a. Modify the dimensions to satisfy the problem requirements.
 - b. Update (or create) the serial number of the file in the commenting at the top of the file.
 - c. The modified copy of the include file must be renamed, and archived with the corresponding source code.
 - i. A file name extension should be added which includes the serial number corresponding to the file.
 - ii. Add it to both the "current" and "working" directories.
3. Retrieve a copy of the current VAM3D-CG source code. Modify the revision number (call to BUILDHDR at the top of the main routine).
4. Compile and load a test version of the software, with the desired array dimensions. Omit the QA header for this executable.
5. Run some relevant tests to ensure the software works.
6. Re-build the executable with the QA header restored.
7. Save the executable file in the current executable directory. Notify the user(s) who requested this change that the executable is available.
8. Update the documentation:
 - a. Update (or create) the text file which describes the current revisions of the software (in the public directory).
 - b. Update the revision log in the project file.
9. DO NOT save the modifications to the main source code file. Only the new version of the file "vor.inc" must be archived.

7.4 MIGRATION TO A NEW OPERATING SYSTEM

1. Check the Software Licensing Agreement. The operating environment is part of that agreement. This may require re-negotiation of the license.
2. Determine the new revision number(s) for each executable (See Section 3.2).
 - a. Document the revision number(s) and PARAMETER file serial numbers (if applicable) in the project file.
 - b. Update the change history log and source code documentation in the project file.
 - c. If multiple code revisions are required, update (or create) the text file which explains differences between revisions.
3. Move the source code to the new system. Recompile the code and run the existing test problems, to ensure the results to not change significantly.
 - a. Some small differences should be expected when changes in precision or math libraries are involved. In this case, comparing plot files can sometimes be more informative that comparing printed output data.
 - b. If serious problems occur in any test, notify the vendor.
4. Generate an EDT to document the operation of the software on the new system. This should include discussions of the following:
 - a. Any system dependent changes required to the source code.
 - b. A description of the configuration of the code on the new system. This includes source code files and compilation procedures. Desk instructions for software use should also be made available (e.g., public directories where the executable(s) is located).
 - c. The test cases run, and their results.
 - d. Any other information which the users may need in order to successfully run the code.
5. Build the new, formally verified executable(s).
 - a. For each code revision, set the correct revision number at the top of the code (call to BUILDHDR).

6. Archive the source and executable codes (see Section 1.2):
 - a. Create a new "working code" sub directory. Save the source code and script files in that directory, and to the "current" source code sub directory.
 - b. Move the previously used executable code to the sub directory for defunct codes. (If permitted by license agreement. Otherwise delete that source code.)
 - c. Save the new executable code to the sub directory for "current" executables.

8.0 TEST VERSIONS OF THE SOFTWARE

On occasion, a user will want to make modifications to the software. This may be for quick tests of a perceived problem, or to examine a new model.

Source code may be delivered to an authorized user or programmer, under the following conditions:

1. Do not deliver any more source code than is needed. Provide object (compiled) files for those files which do not need to be modified.
2. Modify the QA header to clearly indicate the program results are not approved.
 - a. In the variable QAREC, in the main routine (just prior to the call to BUILDHDR).
 - b. Replace the routine BUILDHDR with appropriate dummy coding.
3. Notify the user that he or she is responsible for:
 - a. Using any results from non validated versions of the code.
 - b. Protecting the proprietary code from unauthorized uses, and from access by unauthorized users.

9.0 REFERENCES

Lu, A. H., D. W. Langford, "Certification of VAM3D-CG, Version 2.46, for WHC Computer Platforms," WHC-SD-ER-CSWD-005, Westinghouse Hanford Company, Richland, Washington, 1994.

Huyakorn, Peter S., Sorab Panday, "VAM3D-CG - Variable Saturated Analysis Model in Three Dimensions with Preconditioned Conjugate Gradient Matrix Solvers, Documentation and User's Guide", HydroGeologic, Inc., April 1992.

"Quality Assurance Manual" WHC-CM-4-2, QR 19.0, Rev 1, Westinghouse Hanford Company, Richland, Washington, April 1998.