Year 2000 Compliance Concerns

with the

ISA Thermoluminescent Dosimetry Data Processing

(TL-DP) Software System

prepared by

Ken Savitz

External Dosimetry Section
Radiological Health Department
Safe Sites of Colorado
Rocky Flats Environmental Technology Site
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, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness 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. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.
## Table of Contents

**Overview** ........................................................................................................... 1
**Background** ........................................................................................................ 1
**System Concerns** ................................................................................................. 1
**Methodology** ....................................................................................................... 2
**Research** ............................................................................................................... 2
**Data Files** ............................................................................................................ 2
  - ISA TLD Software ......................................................................................... 3
  - ISA System Software .................................................................................. 3
**Source Code** ......................................................................................................... 3
  - ISA Software .............................................................................................. 3

**Testing and Code Modification** ........................................................................... 4
**Testing** ................................................................................................................ 4
**Code Modification** .............................................................................................. 4
**Summary** .............................................................................................................. 5

**Appendix I** ........................................................................................................ 6
**Digital OpenVMS Year 2000 White Paper** ............................................................ 6

**Appendix II** ........................................................................................................ 15
**Digital Guidelines for Year 2000 Code Investigations** ........................................ 15
........................................................................................................................................ 16

**Appendix III** ...................................................................................................... 22
**Digital Testing for the Year 2000 with OpenVMS** ............................................... 22

**Appendix IV** ...................................................................................................... 26
**Digital Results of OpenVMS Year 2000 Analysis** ................................................ 26

**Appendix V** ...................................................................................................... 34
**ISA Code Modifications** .................................................................................... 34
OVERVIEW

The year 2000 is rapidly approaching, and there is a good chance that computer systems that utilize two digit year dates will experience problems in retrieval of date information. The ISA Thermoluminescent Dosimetry Data Processing (TL-DP) software and computer system has been reviewed for Year 2000 compliance issues.

BACKGROUND

The ISA software resides on a computer node named FALCON. FALCON is a MicroVAX 3300 from Digital Equipment Corporation (DEC) located in the External Dosimetry laboratory at the Rocky Flats Environmental Technology Site (RFETS). This computer system is used to capture thermoluminescent dosimetry (TLD) data from Panasonic UD-710 automatic tld readers and then to process the data into meaningful dose equivalent results.

The computer is currently running the VMS version 5.5-2 operating system along with VAX Fortran77, DECNet, Diskkeeper, and other native software programs. The user applications include the TLD processing software from ISA and the bar code scanning software developed at RFETS. Both of these applications were written in FORTRAN using the native VAX Record Management Services (RMS) file handling system.

There are other software products used within and as add-ons to the ISA application. These include the SaturnGraphics graphical utility and the UDMS interactive adhoc report writer.

SYSTEM CONCERNS

To evaluate the Year 2000 compliance issues, there are many parts of the computer that must be examined. The operating system(s) and the application programs must be considered separately as well as any intelligent devices connected to the system. If a component contains a real-time clock or if the computer’s system clock is accessed for date information, there is the potential for a problem.

Generally speaking, the concern for the arrival of the year 2000 is for the use of systems to represent a year as ‘98’ rather than ‘1998’. There is a possibility that a year such as ‘00’ will be appended with ‘19’ and not the correct ‘20’.
METHODOLOGY

An inspection of a software system for year 2000 compliance requires consideration of all aspects of the process including the computer hardware, the operating system, the application language(s), the data file formats, and any other components utilized in the data pathway. For External Dosimetry, this pathway also includes specialized hardware consisting of the Panasonic TLD readers, the National Instruments general purpose interface bus (GPIB), and the Microdesigns glow curve interface (GCI/1).

Appendix II provides the Guidelines for year 2000 Code Investigations from DEC. While this recommended technique describes a methodology which is sound, due to time and manpower constraints, an abbreviated approach will be applied.

The abbreviated approach will be as follows:

1. an investigation of the data file formats using available documentation and the system file editors,
2. a search of the source code for obvious date references (i.e., 'year' or 'date'), and
3. a test of the software with the system clock set beyond the year 2000.

This investigation will be documented with copies of the file formats, listings of the source code, and output reports from the testing.

RESEARCH

DATA FILES

There are a number of data files in the FALCON system. Each type of file has a different file format. As a rule, the ISA software uses a full date representation and would, therefore, not be affected by the year 2000.

The following is a list of files and the date representation based on available information:
ISA TLD Software -

Many of the ISA files store dates by a number of seconds since a point in time and then use an algorithm to convert to a date. These values are generally stored as double precision variables.

1. ecf data bases - double precision variable (8 digit floating point)
2. issue data bases - double precision variable (8 digit floating point)
3. mean ecf data bases - double precision variable (8 digit floating point)
4. raw data files - 16 character values (MMDDYYYhhmm)
5. history data bases - double precision variable (8 digit floating point)

However, there are additional files that are generally used at ISA sites which may cause Year 2000 problems. These files were typically created as data transfer files for ISA to send information to a records management system.

1. dose update file - character 6 variable (YYMMDD)
2. issue update file - character 6 variable (YYMMDD)

It is anticipated that these dates would be created correctly by the ISA software. However, the problem may be in the program that interprets the data, outside of ISA.

ISA System Software -

There are a number of ‘system’ files for the ISA software. These files control such things as the menus, the printers, and the application devices. It is not expected that these files will be affected by the year 2000.

**SOURCE CODE**

ISA Software

A cursory inspection of the source code for keywords such as 'date' and 'year' was performed with the aid of the VMS operating system search command. This search resulted in a large number of 'hits' which were reviewed, but did not reveal any obvious Year 2000 problems.
Testing and Code Modification

TESTING

Since a complete review of the source was not feasible, a second VAX computer running VMS 5.5-2 was configured with a duplicate set of the ISA software and necessary data files. An iterative process of trial and error was initiated.

With the system clock set to the year 2000, testing of the application software commenced. Immediately, the ISA software would not load the first menu screen. Investigation of the source code revealed that a FORTRAN function called ‘idate’\(^1\) was utilized to read the system clock. This function returns a 2 digit year. Fortunately, the function was isolated to a single subroutine called read_clock which was part of the generic library subroutines. A second subroutine was found which also used the FORTRAN function. These two subroutines were called by approximately 40 of the ISA programs. A simple if/then algorithm was inserted to test the year returned by the idate function and add ‘1900’ if greater than 50 or ‘2000’ if less than 50.

Testing resumed and continued successfully. All of the most critical programs were tested including ecf generation, issue which performs a date test for MECF validation, dose which performs a background calculation, and dose update which produces a file containing a six digit date and a two digit year in each record. All programs executed and handled the data without problem.

The data capture programs were tested only in the RS232 mode and without glow curves. It was not possible to test the glow curve capability on the backup VAX computer system.

Code Modification

All of the program modifications were limited to the subroutines using the idate function. Since these subroutines were called by a large number of main programs, the entire software system was rebuilt.

\[^1\]VMS On-Line Help - FORTRAN Intrinsic Functions IDATE

IDATE (month,day,year) - A subroutine that returns three values representing the current date. The arguments must be defined as integers or integer array elements. The month is represented as the number of the month (1 - 12). The day is represented as the day of the month. The year is represented as the last two digits of the year.
SUMMARY

The ISA TL-DP software is not Year 2000 compliant without some rather minor code modifications. The FORTRAN `idate` function is an inherently non-compliant clock function since it only returns a two digit year. It is the correct year based on the system clock; however, any program which uses this function must recognize that the century is not provided and must, therefore, be added to the year.

A Year 2000 investigation must consider every component in a data system. Although assurances have been obtained from the manufacturers, as of this writing the Microdesigns glow curve interface and the National Instruments IEEE interface have not been tested.
Appendix I

DIGITAL OpenVMS Year 2000 White Paper
OpenVMS Year 2000 Initiative

1 What is the Year 2000 Problem?

The Year 2000 problem results from an industry-wide practice of representing years with only two digits instead of four (for example, specifying 96 instead of 1996). This practice was widespread from the 1960s to the 1980s to save disk and memory space when these resources were relatively expensive. To compound the problem, many standards and programming guides promoted 2-digit year formats, and some common specifications, such as ANSI and DOD, also allowed 2-digit year formats.

When used, these 2-digit year representations can cause application problems during the transition to the year 2000 if the system interprets the year 00 as 1900 or interprets any 2-digit year (01, 02, and so forth) as a 1900s date. In fact, many applications in the industry have already encountered problems caused by 2-digit year formats. Some examples of problems currently posed by the 2-digit year method are:

- Bank cards expiring in the year 2000 have been denied transactions by ATMs that interpret 00 as a card that expired in 1900.
- Bank computers performing mortgage calculations can produce negative balances.
- Warehouses tracking date-sensitive inventory can misinterpret a 00 year field as inventory that is 97 years old.

2 Customer Concerns

During 1996, customers started contacting DIGITAL, and OpenVMS in particular, for information about the status of our products with regard to the Year 2000 problem. Requests for information typically centered around the following questions:

- **Status of OpenVMS products**

  "Are OpenVMS products currently Year 2000 ready? Are there any problems or limitations?"

  The OpenVMS operating system is ready for the year 2000 now! We are publishing the results of our investigations on the World Wide Web, including some minor restrictions that have been found.
• Strategy and plans

"What plans are in place to ensure the Year 2000 readiness of OpenVMS products?"

The cornerstones of the OpenVMS Year 2000 Initiative are all discussed in this document: Investigation, Testing, Documentation, and Solutions.

• Schedule

"When will Year 2000 ready releases of OpenVMS products be available?"

Modifications or enhancements for Version 7.1 and Version 6.2 will be ready in 1997. In addition, support for Version 5.5-2 will be ready during the first half of 1998. OpenVMS Year 2000 activities conform to the following schedule, which seems to underlie Year 2000 efforts across the entire information technology (IT) industry:

- 1996: Vendors and users begin assessing IT environments
- 1997: Year 2000 solutions become available
- 1998: Users validate Year 2000 solutions
- 1999: Users install solutions in production environments

3 Current Status of OpenVMS

The OpenVMS operating system is ready and safe for the transition to the year 2000! We can confidently make this claim for a number of reasons:

• **Software uses 4-digit years**
  The OpenVMS operating system has always allowed a 4-digit year format, which is not affected by the transition to the year 2000. Applications that consistently use the 4-digit year format will make a seamless transition into the year 2000.

• **Hardware uses 4-digit years**
  At the hardware level, no changes are required because DIGITAL’s Alpha and VAX system platforms use 4-digit year representational formats and will be totally unaffected by the transition to the year 2000.

• **Testing found no problems**
  OpenVMS is one of the few operating systems that allows users to set the system clock to times in the future, so it is possible to simulate the transition to the year 2000 and beyond. Both DIGITAL and OpenVMS customers have performed such simulations and no problems have been reported.
• Code investigation found only a few, minor limitations
  A complete, formal, and detailed code investigation on OpenVMS Version 7.1 uncovered only a very few, minor limitations in older and rarely used components. We will distribute several modifications or enhancements in a Year 2000 kit, which will be available through the normal service channels and on the Web by November 1997.

Importance of Testing Your Environment

DIGITAL expects that most Year 2000-related problems will occur primarily in layered applications. Therefore, it is important to start evaluating your environments and applications as soon as possible. Even if DIGITAL's products are ready for the year 2000, you must ensure that the environments in which these products operate are also ready. For help in implementing your own Year 2000 initiative, contact your Digital support representative to learn what tools and information they can offer to get you started.

4 How OpenVMS is Addressing Customer Concerns

Our goal for satisfying customer concerns about the year 2000 can be summarized in a single statement:

• "We will ensure that our products work without problems through the transition to the year 2000 and beyond."

We decided that the best way to address customer concerns was to launch a complete and formal initiative that addresses the whole OpenVMS environment --- that is, the operating system and all other products that are usually packaged with it. The goals of the OpenVMS Year 2000 Initiative are as follows:

• To ensure Year 2000 readiness of OpenVMS software
  In the OpenVMS software environment, many interfaces and layered software components operate on the Year 2000-ready core of the OpenVMS operating system. We intend to ensure that these interfaces and layers of software are also ready for the transition to the year 2000.

• To conform to DIGITAL's new Year 2000 Warranty
  DIGITAL stands behind the Year 2000 readiness of its products with a Year 2000 Warranty on its Year 2000-ready products. This warranty has been created specifically to cover Year 2000-ready products. Our goal is to ensure that OpenVMS software and its software...
layered products meet the requirements of this new warranty.

- **To maintain OpenVMS quality**

  OpenVMS and its layered software products have traditional strengths (for example, 24x365 operation and disaster tolerance) and top-in-class quality that we will maintain by ensuring that all our products can smoothly and seamlessly make the transition to the year 2000 and beyond.

## 5 OpenVMS Engineering's Commitment

The highest levels of OpenVMS management have confirmed that the Year 2000 effort commands a high priority. Engineering work has centered around the activities of investigation, testing, documentation, and solutions.

### Investigation

The entire OpenVMS engineering group has been involved in conducting an investigation of the latest OpenVMS product code to evaluate its Year 2000 readiness. First, we created an inventory of all OpenVMS products (including their modules and components) that should be addressed by our due diligence effort. Subsequently, all product sources were analyzed by OpenVMS engineers in a fashion equivalent to a line-by-line inspection.

To ensure that OpenVMS investigations were conducted at the same level of thoroughness, all engineers received guidelines that precisely and formally identified all aspects of the investigation, including the following:

- Methodology
- Criteria for due diligence
- Problem reporting and documentation
- Tools to aid the investigation and reporting of data

### Testing

Testing plays an important and prominent role in our Year 2000 effort. Regression test suites have been used to simulate the transition to the year 2000 and beyond and to validate product modifications and enhancements. Among others, we have found the following transition dates interesting to test:

- September 9, 1999 to September 10, 1999 (to confirm correct translation of 9/9/99)
- December 31, 1998 to January 1, 1999 (to check whether 99 is used to mean "no expiration date")
- December 31, 1999 to January 1, 2000 (to check century transition; January 1 should be a Saturday)
OpenVMS Engineering and several OpenVMS customers have run simulations of the year 2000 and no problems have ever been found. Regression test suites have been run on OpenVMS VAX and Alpha Version 7.1, OpenVMS VAX Version 6.2, and OpenVMS Alpha Version 6.2-1H3; tests will also be run on VMS Version 5.5-2.

In addition to the regression test suites, we have also run other tests, including load testing, cluster tests, the User Environment Test Package (UETP), and ad hoc tests of DECwindows Motif, the Backup utility, the job controller, and the file system exerciser. Test clusters have contained a combination of both DECnet-Plus and DECnet for OpenVMS (Phase IV) nodes.

Based on our tests and investigations, applications that consistently use the 4-digit year representations that OpenVMS produces or accepts as input will not be affected by the transition to the year 2000.

Since OpenVMS is one of the few operating systems that allows users to advance system clocks to times in the future, you can also conduct simulations to test your own software now for potential year 2000 problems.

Documentation

We are providing customers with comprehensive documentation of our investigation results, including a list of any year 2000-related limitations identified by the investigation. This information is posted on the Web as it becomes available. We will also ship release notes with the Year 2000 kits.

Solutions

Because the few restrictions we have found are so minor and limited in scope, we plan to release Year 2000 kits rather than new versions of our products. These kits will be available through the normal service channels and over the Web from the Software Patch (ECO) Access page. The release schedule is as follows:

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>SCHEDULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenVMS Version 7.1 and 7.1-1H1</td>
<td>November 1997</td>
</tr>
<tr>
<td>Layered products for Version 7.1</td>
<td>October to December 1997</td>
</tr>
<tr>
<td>OpenVMS Version 6.2 and 6.2-1H3</td>
<td>December 1997</td>
</tr>
<tr>
<td>OpenVMS Version 5.5-2 and 5.5-2H4</td>
<td>First half 1998</td>
</tr>
</tbody>
</table>

Ken Savitz
x4686/212-4443
We strongly recommend that customers upgrade their environments to our Year 2000-ready releases before the year 2000.

**Future Products' Readiness for Year 2000**

Another goal of the OpenVMS Year 2000 Initiative is to incorporate checkpoints and other verification steps into existing engineering and maintenance quality processes during 1997 to ensure that no new year 2000-related problems are introduced in our new and updated software products.

**6 Scope of the OpenVMS Year 2000 Initiative**

The following sections describe the scope of the OpenVMS Year 2000 Initiative.

**OpenVMS Versions**

To ensure that all components and features of our products are Year-2000 ready, we are conducting a full investigation on the latest release of our software products. However, some earlier versions will also be reviewed to aid customers who still rely on older versions of OpenVMS products.

Our approach is as follows:

- Conduct a full investigation on the latest release of our software products because this release is the most comprehensive.
- Backport modifications and enhancements for the Year 2000 to previous versions whenever possible and appropriate.
- Conduct tests and simulations on configurations of key components of older versions of software.
- Investigate selected code from previous versions of products wherever it differs from that of newer versions.

To be specific, we are conducting a complete due diligence investigation on OpenVMS Version 7.1. Next we will address OpenVMS Versions 6.2 and 5.5-2 by including modifications and enhancements backported from Version 7.1. In addition, we will identify differences in code between Version 7.1 and Versions 6.2 and 5.5-2, investigate them, and then provide any necessary additional modifications or enhancements. The investigation of Version 6.2 is currently underway and no new limitations have been identified so far. When the investigations are complete and any necessary modifications have been applied, we will run our OpenVMS regression test suites on OpenVMS Versions 6.2 and 5.5-2.
The versions that will be supported with Year 2000 kits are Versions 5.5-2, 5.5-2H4, 6.2, and 6.2-1H3. However, the Version 6.2-1H3 kit can be installed on any other variant of Version 6.2 because Version 6.2-1H3 includes them all.

Products

The OpenVMS Year 2000 Initiative addresses not only the OpenVMS operating system, but also several families of layered products. The status of these products is also documented with the results of our investigation.

7 Software Outside the Scope of the OpenVMS Year-2000 Initiative

The following sections describe areas that are outside the scope of the OpenVMS Year 2000 Initiative.

Old Versions

The OpenVMS Year 2000 Initiative may not address some old or obsolete versions of products that are still in use. We recommend that customers upgrade their environments to the more recent Year 2000-ready releases before the year 2000.

Retired Products

Generally speaking, products that will be retired by the end of calendar year 1998 will not be addressed by the OpenVMS Year 2000 Initiative.

Freeware

The OpenVMS Year 2000 Initiative will not address freeware. Freeware is any piece of software that DIGITAL distributes under a disclaimer such as the following:

"The freeware components on this CD are included as prebuilt applications. Please note that this software is provided 'as is'. Digital Equipment Corporation disclaims all warranties with regard to this software, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS."

8 Other DIGITAL Year 2000 Programs

The OpenVMS Year 2000 Initiative is part of a larger DIGITAL Year 2000 effort. Digital Equipment Corporation has implemented a company-wide program to address Year 2000 readiness of its products and services. The DIGITAL Year 2000 Program helps protect customers and business partners against the possibility of serious Year 2000-related issues. The program covers hardware systems, operating systems, layered products, and services.
The corporate program also coordinates efforts among various product groups within DIGITAL and works with DIGITAL's software partners to share information and planning in order to deliver a complete Year 2000-compliant solution to our customers.
Appendix II

DIGITAL Guidelines for Year 2000 Code Investigations
Guidelines for Year 2000 Code Investigations

This document describes the approach OpenVMS Engineering took when investigating the OpenVMS Version 7.1 operating system for Year 2000 readiness. This information is provided in case it is useful to customers who are reviewing their own applications and environments in preparation for the Year 2000.

1 OpenVMS Year 2000 Investigation Process

We investigated the latest code for products in our organization, including all the system integrated products (SIPs): OpenVMS Cluster Systems, Volume Shadowing, RMS Journaling, DECnet for OpenVMS, and DECnet-Plus for OpenVMS. All of these SIPs are Year 2000 ready. See our investigation findings for the status of other layered products.

The only products we did not investigate are freeware and products that are going to be retired. For more information about the goals and scope of the OpenVMS Year 2000 Initiative, see the white paper.

The OpenVMS Year 2000 Initiative entailed a due diligence investigation of the OpenVMS Version 7.1 operating system. Engineers were asked to ensure that the methods and tools used for searches and analyses provide results that were equivalent to the results that would be obtained from a line-by-line code review. We developed a formal investigation process and issued guidelines to be used by all who investigated the base operating system. Those guidelines, which are summarized in this document, were also distributed to layered products that run on OpenVMS.

The following sections describe the OpenVMS Year 2000 investigation process, which included the following steps:

- Defining the scope of the project (see Section 2)
- Finding and analyzing potential problem areas in code (see Section 3)
- Testing the system for Year 2000 problems (see Section 4)
- Documenting the findings (see Section 5)
- Resolving any Year 2000 limitations (see Section 6)

2 Defining the Scope and Process for the Project

We used the following processes to define the scope of the project and break it down into manageable pieces:
1. Made an inventory

First we created an inventory of all the components and modules of code to be investigated. To do this, we inventoried all the files that were used in the build of the OpenVMS VAX and Alpha Version 7.1 release kits.

2. Populated a database

We used the inventory data to populate a database so that we could track investigation findings and chart our progress toward completion.

3. Identified areas of responsibility and assigned resources

Individuals were assigned to investigate the code they own and these assignments were entered in our database. This exercise allowed us to identify unassigned "orphan" modules (usually older code that is not actively maintained), which were subsequently adopted by volunteer investigators.

4. Set investigation boundaries among dependent components

Where the status of one piece of code is dependent upon the status of another piece of code, we established boundaries of responsibility for individual investigators and then coordinated the findings of each to resolve dependencies.

5. Established standard ratings for investigated code

Investigators were asked to rate investigated code (modules, interfaces, routines, and global cells) as follows:

- **N/A**: The investigated piece of code does not contain or manipulate date or time data.
- **PASS**: The investigated piece of code contains date/time-related coding that will continue to function without error as the date changes from December 31, 1999 to January 1, 2000, and well beyond that. Such code meets the following requirements:
  - The code correctly processes, calculates, compares, and sequences date data within and between the 20th and 21st centuries, including leap year calculations.
  - All date calculations, representations, and translations are correct for any date data within the date range covered by the investigation. (The OpenVMS date range is defined later in these guidelines.)
The code provides 4-digit year interfaces and APIs, or, in the case of 2-digit year representations, the code includes an encoding and decoding mechanism that allows unique and unambiguous representation of years over a range that spans the latter portion of the 20th century and extends well into the 21st century. (For example, OpenVMS Alpha allows the use of system global cell EXE$GL_TRANSITION_YEAR.)

NOTE: Displays that use 2-digit fields to display years in dates are not a problem unless an application parses the year value and uses it as a date. To be safe, you may wish to convert all output year fields to 4-digit year fields where possible. You may also need to make corresponding adjustments to any applications that parse these fields.

Any dependencies on external code is on code that has also been investigated and found to be Year 2000 ready.

- **FAIL**: Some aspect of the code is not Year 2000 ready.

  A piece of code must be classified FAIL if it does not function properly with date data beyond December 31, 1999.

  NOTE: We also allowed the FAIL status to be used to flag situations where enhanced documentation would be helpful; for example, 4-digit alternatives might exist for 2-digit year dates, but the alternatives could be better documented.

- **DEPENDS**: The code itself is Year 2000 ready, but it calls an interface or routine whose status is not yet known.

  NOTE: We tried to schedule investigations so that the status of dependencies was publicly available before dependent code was investigated.

- **RETIRED**: Code is for a component that is retired/unsupported or will be retired/unsupported as of January 1, 1999, and the product manager has explicitly stated that the code need not be investigated.

6. Set 2038 as the boundary for achieving Year 2000 readiness

We determined that the Year 2000 Initiative would address any date/time problems up to, but not including, the year 2038. We specified 2038 as the cutoff date for the Year 2000 evaluation because another industry-wide, date-related
problem may occur in 2038. That problem is analogous to the Year 2000 problem, but is caused by limitations in the current C language standards for time and date storage.

NOTE: The 2038 problem is expected to have little, if any, effect on the OpenVMS operating system. Generally speaking, OpenVMS has been designed to function properly through the year 9999.

3 Finding and Analyzing Potential Problem Areas in Code

This section lists steps for finding and analyzing potential problem areas in code.

1. Examine the code and data flow for all time and date related APIs, system global cells, data structure fields, media formats, and message formats for possible Year 2000 problems. You may wish to search component map files to help identify date-related interfaces.

   Be sure to check all dates that are input or output. Watch for the following conditions and coding practices as well as any other time/date conditions that are unique to the code you are investigating:

   - 2-digit representations of years
   - Small binary fields that might represent dates
   - Invalid date differentials
   - Procedures that add 1900 (or another hardcoded century value) to the date or subtract 1900 from the date
   - Code that divides by 100 or 400 (to calculate leap years)
   - Code that results in incorrect sequencing of dates
   - Code that causes incorrect scheduling of system events

2. You may need to make a more detailed analysis than is immediately apparent. Dates might be converted to an identity or format that is no longer obviously a date and then be passed on to other routines.

   For example, a 4-digit year (1997) might be exported to a process that deletes 1900 from the year and passes the result (97) to another process, which then performs another calculation, eventually causing a problem.

3. Make sure that the format of date/time stamps outputs unambiguous date data. For example, you might need to examine the user interface to ensure that the format can accommodate a 4-digit year.
4. After you analyze code found by searching for known date/time APIs and data structures, you must analyze the code flow to see how the date data is interpreted and check whether it is exported.

5. If the code you are investigating has any dependencies on other code, you must find out the Year 2000 readiness status of that code before you can assume that your code is Year 2000 ready. (We used a central clearinghouse to share the status of dependent code.)

Following is a list of questions to use when you review code:

- Does your code import date information in any way --- for example, in APIs, system global cells, data structure fields, media formats, or message formats? If so, does the information contain an inherent or specific year field? If so, is the associated year field represented by two digits or four digits?
- What does your code do with the date information?
- If your code manipulates the year, does it use all four digits of the year?
- Do you store a date in any data structures, either as a date string or a binary integer?
- Are all date data structure fields large enough to contain a 4-digit year?
- Does your code export any date information, either as a date string or a binary integer? If so, are all four digits of the year included?
- Does the code do any time/date arithmetic? If so, are 4-digit years used?

4 Testing Your Code for Year 2000 Readiness

While testing cannot be used as a substitute for investigating code, you can identify potential Year 2000 problems by testing software in an artificial environment that simulates future dates.

The OpenVMS operating system allows you to set the system clock forward to create an appropriate testing environment. This allows you to test a variety of post-year 2000 scenarios, such as whether January 1, 2000 falls on a Saturday and March 1, 2000 falls on a Wednesday (after leap day on February 29).

Note that many system applications and layered products are affected by any substantial modification of system time. The testing procedure described on this website is similar to that used for common maintenance operations (such as system upgrades) and includes steps to minimize the side effects of modifying system time.

You may wish to perform other tests. In addition to regression test suites, we also ran other tests, including load testing, cluster tests, the User Environment Test Package (UETP), and ad hoc tests of DECwindows Motif, the Backup utility, the job controller,
and the file system exerciser. Test clusters contained a combination of both DECnet-Plus for OpenVMS (Phase V) and DECnet for OpenVMS (Phase IV) nodes.

5 Documenting Your Findings

If you put effort into investigating your code, you will want to document your work. We started by creating a database that listed all the modules in the operating system. As engineers completed their investigations, they filed formal reports for all code, using the ratings we established to identify the status of all modules and date-related interfaces.

As this information was entered into the database, we could periodically poll the database to track our status in terms of investigation completion and Year 2000 readiness.

If engineers found Year 2000 limitations in the code they investigated, these were detailed in a separate database along with the proposed solution (either a code change or enhanced documentation). As you can see from the results of our investigations, the problem reports were minimal.

6 Resolving Year 2000 Limitations

If you detect any Year 2000 limitations in your code, you should change the code and/or the data storage if at all possible. In cases where an external standard or convention prohibits changing 2-digit year date formats, provide an alternative or work around the restriction and thoroughly document the condition.

In general, code that works with and stores a 2-digit year should be changed to use a 4-digit year. When this is not possible, you may be able to use a century window to define how 2-digit year values are to be interpreted; for example, OpenVMS Alpha allows the use of global cell EXE$GL_TRANSITION_YEAR, which defines 57 to 99 to be interpreted as 1957 to 1999, and 00 to 56 to be interpreted as 2000 to 2056.

Other data storage options include, but are not limited to, the OpenVMS binary-format time quadword, the OpenVMS text-format time (for example, "10-DEC-1997 11:59:52.61"), the Coordinated Universal Time (UTC) binary-format time, and the UTC text-format time.
Appendix III

DIGITAL Testing for the Year 2000 with OpenVMS

Ken Savitz
x4686/212-4443
kenneth.savitz@rfets.gov
Testing for the Year 2000 with OpenVMS

While testing cannot substitute for investigating code for year 2000 problems, you can identify potential year 2000 problems by testing software in an artificial environment that simulates future dates. The OpenVMS operating system allows you to set the system clock forward and backward, which provides an appropriate environment for system testing. The following procedure describes a technique for protecting your environment while testing applications for the Year 2000.

Many system applications and layered products are affected by any substantial modification of system time, independent of the change to the Year 2000. The following procedure includes guidelines that minimize side-effects that may occur due to these time modifications. Digital recommends that you follow these guidelines to ensure system integrity.

**Caution**

*Do not perform Year 2000 tests or widely vary the system clock on a system in a production environment.*

Follow this procedure to test your system:

1. Ensure that installed Product Authorization Keys (PAKs) will not expire during the testing period. If your system has temporary PAKs, contact your application vendor.

2. Perform a complete backup of the system disk and associated data disks. (If possible, back up to spare disks.)

3. Remove the system to be tested from the production environment to ensure data integrity during the test session. For example, shut down all network connections as well as any local NTP or DECdts processes before changing the system clock. Nodes running NTP or DECdts clients or servers participate in maintaining system time across the network; their presence could either reset the clock to the current time or could serve a future time to all other nodes in the network.

4. To ensure that the system time is reset before any application code is started, use the following commands to perform a conversational boot of your backup copy of the system disk:
On VAX:

>>> b/r5:1

On Alpha:

>>> b -fl 0,1

Also set the SETTIME system parameter to 1:

```
SYSBOOT> SET SETTIME 1
SYSBOOT> CONTINUE
```

Then respond to the time prompt later in the bootstrap operation.

Some VAX or Alpha consoles may require different commands. Consult the OpenVMS installation and upgrade documentation for your specific Alpha or VAX processor.

5. Once the system has bootstrapped, mount the backup copies of any other necessary data disks.

6. Initialize and complete tests for your specific environment. Among other dates, we have found the following transition dates interesting to test:

   - September 9, 1999 to September 10, 1999 (to confirm correct translation of 9/9/99)
   - December 31, 1998 to January 1, 1999 (to check whether 99 is used to mean "no expiration date")
   - December 31, 1999 to January 1, 2000 (to check century transition; January 1 should be a Saturday)
   - February 28, 2000 to February 29, 2000 (to verify leap year calculation; Tuesday, February 29, 2000 is a valid date in the Gregorian calendar)
   - February 29, 2000 to March 1, 2000 (to verify leap year calculation; March 1 should be a Wednesday)

   These dates are only suggestions; more testing of later dates might be required to thoroughly test your applications.

7. When testing is complete, shut down the system. Then mount and/or restore the original system disk and all associated data disks. (If you backed up your disks on tape instead of disk, restore the original system disk and all original data disks from your backup tapes.)
8. Reboot the restored system. To ensure that the rebooted system has the correct time loaded, DIGITAL recommends that you perform a conversational bootstrap operation with the SYSGEN parameter SETTIME set to 1:

On VAX:

```plaintext
>>> b/r5:1
```

On Alpha:

```plaintext
>>> b -fl 0,1
```

```
SYSBOOT> SET SETTIME 1
SYSBOOT> SET WRITESYSPARAMS 0
SYSBOOT> CONTINUE
```
Appendix IV

DIGITAL Results of OpenVMS Year 2000 Analysis
Results of OpenVMS Year 2000 Analysis

Investigation of the OpenVMS Version 7.1, Version 6.2, and Version 5.5-2 operating systems is now complete; no new limitations were found in our investigations of the earlier versions of the operating system. Investigations of OpenVMS layered products are nearing completion. The OpenVMS base operating systems are generally Year 2000 ready except for a few minor limitations in some older components.

Kits with Year 2000 enhancements for OpenVMS Version 7.1, Version 6.2, and Version 5.5-2 are now available. Many layered product kits have already shipped and others should ship in the near future. Scheduling information for Year 2000 kits is included in the OpenVMS Year 2000 overview.

Table 1 alphabetically lists the OpenVMS product families covered by our investigation, gives the status of each product's investigation, and notes whether any limitations have been reported. We will update this information as investigations complete or if limitations are reported. To learn the status of specific products within these product families, check the Year 2000 layered software products database.

Table 1 Results of OpenVMS Product Investigations for the Year 2000

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>INVESTIGATION STATUS</th>
<th>LIMITATIONS REPORTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenVMS Version 7.1</td>
<td>Complete; kit available</td>
<td>See Kit Notes</td>
</tr>
<tr>
<td>OpenVMS Version 6.2</td>
<td>Complete; kit available</td>
<td>See Kit Notes</td>
</tr>
<tr>
<td>OpenVMS Version 5.5-2</td>
<td>Complete; kit available</td>
<td>See Kit Notes</td>
</tr>
<tr>
<td>DECamds</td>
<td>Complete</td>
<td>None</td>
</tr>
<tr>
<td>DECnet for OpenVMS (Phase IV)</td>
<td>Complete</td>
<td>None</td>
</tr>
<tr>
<td>DECnet-Plus for OpenVMS(Phase V)</td>
<td>Complete</td>
<td>None</td>
</tr>
<tr>
<td>DECprint Supervisor (DCPS)</td>
<td>Complete</td>
<td>None</td>
</tr>
<tr>
<td>DECram</td>
<td>Complete</td>
<td>None</td>
</tr>
<tr>
<td>DECthreads</td>
<td>Complete</td>
<td>None</td>
</tr>
<tr>
<td>DECwindows Motif Version 1.2-4</td>
<td>Complete; kit available</td>
<td>See Kit Notes</td>
</tr>
<tr>
<td>DECwindows Motif Version 1.2-3</td>
<td>Complete; kit available</td>
<td>See Kit Notes</td>
</tr>
<tr>
<td>DECwindows Motif Version 1.2-3 (World Wide)</td>
<td>Complete; kit available</td>
<td>See Kit Notes</td>
</tr>
</tbody>
</table>
## Freeware and Retired Products

Regardless of how it is distributed, all freeware is offered to customers on an unsupported basis. As a rule, freeware will not be investigated and supported for the Year 2000.

Products that are retired or unsupported now or that will be retired or unsupported in the near future will not be investigated or supported for the Year 2000. A list of such products will be posted and updated as this information becomes available.
1 DECwindows Motif for OpenVMS

Year 2000 kits with enhancements for DECwindows Motif for OpenVMS are now available for the following versions:

<table>
<thead>
<tr>
<th>VERSION</th>
<th>PLATFORM</th>
<th>KIT NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1.2-4</td>
<td>VAX</td>
<td>VAXMOTF02_U4012</td>
</tr>
<tr>
<td></td>
<td>Alpha</td>
<td>ALPMOTF03_U4012</td>
</tr>
<tr>
<td>Version 1.2-3</td>
<td>VAX</td>
<td>VAXMOTF08_U3012</td>
</tr>
<tr>
<td></td>
<td>Alpha</td>
<td>ALPMOTF08_U3012</td>
</tr>
<tr>
<td>(World Wide)</td>
<td>VAX</td>
<td>VAXDWMW02_U3012</td>
</tr>
<tr>
<td></td>
<td>Alpha</td>
<td>ALPDWWM02_U3012</td>
</tr>
</tbody>
</table>

You can download kits ALPMOTF03_U4012 and VAXMOTF02_U4012 directly from Digital Services. Use the Search and Download Utility to search for keywords VAXMOTF and ALPMOTF.

Customers with service contracts can access the Version 1.2-3 kits. All kits shown in the table above include modifications that address the following Year 2000-related limitations:

- A 2-digit year format is used in several places:
  - The routines `xnl_parsedatetime` and `xnl$parse_date_time` allow the year to be specified with two digits provided the year is between 1970 and 1999.
  - When the `xnl_getlocale` routine returns locale information for some languages, it specifies that years should be displayed using 2-digit years. Specifically, the following locale files define a string containing %y, which denotes a 2-digit year (%Y connotes a 4-digit year):
    - XNL_ES_ES.UID (Spanish)
    - XNL_FR_BE.UID (Belgian French)
    - XNL_FR_CA.UID (Canadian French)
    - XNL_IW_IL.UID (Hebrew)
    - XNL_NO_NO.UID (Norwegian)

The following locale files define an array containing XNL_ABYEAR, which denotes a 2-digit year (XNL_YEAR connotes a 4-digit year):

- XNL_DE_AT.UID (Austrian German)
- XNL_DE_CH.UID (Swiss German)
DECwindows Calendar accepts a 2-digit year in its interchange format, which is used for cut and paste and in included files (that is, files processed by the Include... option in the File menu). The 2-digit years are always considered to be in the twentieth century; that is, Calendar adds 1900 to the 2-digit value.

This behavior has been documented and changed so that 2-digit years are correctly interpreted in the year 2000. Note that Calendar itself always specifies 4-digit years in the interchange format.

- A CDA Converter routine that converts DTIF format documents to DDIF assumes that years start with 19, including the current date, which it writes on every page of the DDIF document. This code has been modified in the Year 2000 kits.

- Paint and Print Screen both write the creation date to the introductory comments of a PostScript file generated by the Print command. To calculate the creation date, the number of years since 1900 is appended to 19. Using this procedure, on January 1, 2000 the code in DECwindows Motif Versions 1.2-3 and 1.2-4 would cause the date to be written as 1-JAN-19100.

The code has been changed so that the number of years since 1900 is now mathematically added to 1900 instead of being appended to 19.

Kits VAXDWMW02_U3012 and ALPDWWM02_U3012 also include an enhancement to correctly display the year in the Show Links dialog box in the Calendar. (Without the enhancement, the year 2001 would display as 201.)

2 Digital Distributed Computing Environment (DCE) for OpenVMS

Kit Numbers: VAXDCE0n_014 and ALPDCE0n_014

These kits contain Year 2000 enhancements for DCE for OpenVMS Version 1.4. You can download these kits directly from Digital Services. Use the Search and Download Utility to search for keywords VAXDCE and ALPDCE. These kits include modifications that address the following Year 2000-related limitations:
In the SYSSCOMMON:[SYSHLP.EXAMPLES.DCE.DTSS] directory, an example of using the Spectracom time provider does not correctly calculate leap years for the year 2000 and beyond.

The DCE example program BOOK.C accepts a 2-digit year for date input. However, date calculations work correctly only if a 4-digit year is input. A code change in the kit causes the example to enforce the entry of 4-digit years and to output the year in 4-digit format.

DCE dates are based on the UNIX time standard whereby 2-digit years from 00 to 68 belong to the 2000's and years from 69 to 99 belong to the 1900's. The code in the chk_date routine is off by one year, using the ranges 00 to 69 and 70 to 99. With the code change in the kit, the date interpretations will conform to the UNIX standard.

In the Year 2000 kit, the DCE ASN1 code correctly calculates the leap year and prints 4-digit dates on output. The DCE ASN1 code has been updated to correct the leap year calculations and to print 4-digit dates on output.

A couple of routines have two date-related limitations:
- DCE dates are based on the UNIX time standard whereby 2-digit years from 00 to 68 belong to the 2000's and years from 69 to 99 belong to the 1900's. The routines are off by one year, using the ranges 00 to 69 and 70 to 99.
- The routines miss the checks for century and quad-century years when they calculate leap years.

Several print routines output the year in 2-digit format. The new kit changes the routines to output a 4-digit year format.

A few prompts for dates accept 2-digit year formats. The new kit changes the prompts to require 4-digit year input.

These changes will also be included in the next major release of the product.

3 PATHWORKS

The status of the PATHWORKS server and client products is summarized in the following sections.

PATHWORKS Server

The status of PATHWORKS server products is as follows:
- **PATHWORKS Version 6.0 for OpenVMS (Advanced Server)**

  This product is fully Year 2000 compliant, with one exception:
  
  - Included in the product is an application called PCDISK, a utility used to manage FAT files. FAT files are not supported in Version 6.0, but PCDISK is shipped with Version 6.0 for customers who need to use it to migrate off the FAT file system. One feature of PCDISK extracts files from the FAT file system to ODS-2. This extract function will not work in year 2000. However, customers are expected to have migrated off the FAT file system by the year 2000.

- **PATHWORKS Version 5.0F for OpenVMS (LAN Manager)**

  **Kit number:** PWRK50F_E01050

  This kit contains several Year 2000 enhancements as well as bugfixes. You can download this kit directly from Digital Services. Use the Search and Download Utility to search for keyword PWRK50F.

**PATHWORKS Client**

This section describes the PATHWORKS client products that are being made ready for the Year 2000. Some kits are already available. You can download PATHWORKS kits directly from Digital Services. Use the Search and Download Utility to search for keyword PW or a variant such as PW32 or JPW.

- **PATHWORKS 32 Version 7.0A**

  **Kit number:** PW3270A_E01

  This kit contains changes for files NFTW.EXE and RFA_DAP.DLL in the Network File Transfer (NFT) utility for Windows. NFT uses a 2-digit date in the log file where logged operations are timed and dated using the format mm/dd/yy. Without this kit, an internal condition related to an industry-wide standard causes years starting in the year 2000 to be handled improperly. This condition has been modified at a higher level in the DECnet File Transfer utility. Once you apply this kit, the NFT/W utility functions correctly.

  The next version of PATHWORKS 32 that ships will include these changes and be Year 2000 ready.

- **PATHWORKS Client for DOS and Windows Version 6.0A**
A kit will be released soon to address limitations in the following areas:

- DECnet Network File Transfer utilities for DOS and Windows
- TCP/IP File Transfer utilities for DOS and Windows
- File Access Listener for DOS and Windows
- LAT Control Program
- Print utility (PRINTQ)
- Broadcast Message Receiver for Windows 3.1
- Mail for DOS
- ManageWorks: LanManager server and user modules, and NetWare user module

All other PATHWORKS client products will be retired before 1999, with the exception of the two versions of PATHWORKS client software that have been translated into Japanese. Our investigation of these products has completed and kits are available as follows:

- **Japanese PATHWORKS for DOS and Windows for Version 5.1 for DOS/V**
- **Japanese PATHWORKS for DOS and Windows for Version 5.1 for PC98**

**Kit numbers:** JPWDOSV_V51_Y2K and JPWPC98_V51_Y2K

These kits resolve Year 2000 limitations in the following areas:

- DECnet Network File Transfer utilities for DOS and Windows
- TCP/IP File Transfer utilities for DOS and Windows
- File Access Listener for DOS and Windows
- LAT Control Program
- Print utility (PRINTQ)
- Broadcast Message Receiver for Windows 3.1
- Mail for DOS

- **Japanese PATHWORKS for Windows 95 Version 1.0A for DOS/V**
- **Japanese PATHWORKS for Windows 95 Version 1.0A for PC98**

**Kit numbers:** JPWPCAT_V10A_Y2K and JPWPC98_V10A_Y2K

These kits address limitations found in the Network File Transfer (NFT) utility.

**NOTE:** There are several instances of 2-digit years in file information or copyright information in some PATHWORKS client software. However, these dates are strictly cosmetic and in no way affect the software's ability to function properly.
Appendix V

ISA Code Modifications
SUBROUTINE: READCLCK-FOR

DESCRIPTION: Reads the system clock.

%Log: DUBO: [DEVELOPMENT.SYSTEM.GENSUBS]READCLCK.FOV $

Rev 1.020 Jun 1990 16:07:16TAL
Initial revision.

SUBROUTINE read_clock (month, day, year, hour, minute, seconds) ! read system clock subroutine

INCLUDE 'ISA$SYS:[INCLUDE]ISA.HED'

INTEGER*2
* month,
* day,
* year,
* hour,
* minute,
* seconds

LOCAL VARIABLES

LOGICAL*1
flag

date_error

CHARACTER*80
message

DOUBLE PRECISION
* secs_since_midnight,
seconds since midnight
* secs_per_day
seconds per day

DATA secs_per_day /86400./
number of seconds per day

DATA INITIALIZATION

* PROGRAM BODY

DATA date_error = .TRUE.
initialize
date error to true
DO WHILE (date_error)
error is true

Ken Savitz
x4686/212-4443
05/26/98
CALL IDATE (month,day,year)  !call FORTRAN
IDATE routine
secs_since_midnight = SECNDS(0.0)  !use SECNDS
function to get seconds since midnight

!* added by krs 4/29/1998 for Year 2000 Compliance

IF (year .LT. 50) THEN
    year = year + 2000
ELSE
    year = year + 1900
ENDIF

!*  
CALL unpack_time (secs_since_midnight,hour,minute,seconds)  !call unpack
time from seconds function
CALL test_date (month,day,year,hour,minute,seconds,date_error)  !call test date routine
END DO  ! end do

RETURN  ! return to
calling routine
END
SUBROUTINE GETDATE.FOR

DESCRIPTION: Returns the date.

******************************************************************************

$Log:
- DUBO:
 [DEVELOPMENT.TLD.SOURCESlGETDATE.FOV$

Rev 1.022 Jun 1990 14:43:44 TAL
Initial revision.

******************************************************************************

SUBROUTINE get-date (ret-date, ret-time)

INCLUDE 'ISA$SYS:[INCLUDE]ISA.HED'

******************************************************************************

LOCAL VARIABLES

******************************************************************************

CHARACTER

* ret-date(*),
* ret-time(*),
* Companyl(1:4)*60,
* Current_time*8

INTEGER*2

* Company_lun,
* Date_format,
* Month,
* Day,
* Year,
* return_status

******************************************************************************

DATA INITIALIZATION

******************************************************************************

CALL get-new-lun (company_lun)

******************************************************************************

PROGRAM BODY

******************************************************************************

OPEN (UNIT=Company_lun, FILE='SITE$SYS:COMPANY.SYS;1', STATUS='OLD',
* ORGANIZATION='SEQUENTIAL', ACCESS='SEQUENTIAL',
* RECl=512, FORM='UNFORMATTED', INITIALIZE=2,
* IOSTAT=Return_status)

IF (Return_status .NE. 0) THEN
  Date_format = 1
  Companyl(1) = ' ' 
ELSE
  READ (Company_lun) Date_format, Company1
  CLOSE (Company_lun)
ENDIF

CALL IDATE (Month, Day, Year)

IF (Date_format .EQ. 0) THEN
  IF (year.GE.50) THEN
    WRITE (Ret_date,100) Year, Month, Day
  ELSE

 asserted by krs for Year 2000 5/11/98

CALL IDATE (Month, Day, Year)

IF (Date_format .EQ. 0) THEN
  IF (year.GE.50) THEN
    WRITE (Ret_date,100) Year, Month, Day
  ELSE

Ken Savitz
x4686/212-4443
kenneth.savitz@rfets.gov
WRITE (Ret_date,110) Year,Month,Day
ENDIF
ELSE
    IF (year.GE.50) THEN
        WRITE (Ret_date,101) Month,Day,Year
    ELSE
        WRITE (Ret_date,111) Month,Day,Year
    ENDIF
ENDIF

CALL TIME (Current_time)
ret_time = current_time (1:5)
RETURN

100 FORMAT ('19',12.2,'-',12.2,'-',12.2)
101 FORMAT (12.2,'-',12.2,'-',19',12.2)
110 FORMAT ('20',12.2,'-',12.2,'-',20',12.2)
111 FORMAT (12.2,'-',12.2,'-',20',12.2)
200 FORMAT (A5)

END