Version Description and Installation Guide
Kernel Version 3.0

Judy Bamberger
Timothy Coddington
Robert Firth
Daniel Klein
Dave Stinchcomb
Roger Van Scoy

December 1989

Approved for public release
Distribution unlimited
Version Description and Installation Guide
Kernel Version 3.0

Judy Bamberger
Timothy Coddington
Robert Firth
Daniel Klein
Dave Stinchcomb
Roger Van Scoy

Approved for public release.
Distribution unlimited.

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
# Table of Contents

1. Introduction  
   1.1. Audience  
   1.2. Related Documents  
   1.3. Conventions

2. Acquisition

3. Installation  
   3.1. System Manager Actions  
   3.2. DARK Installation Actions

4. Customization

5. Known Deficiencies

6. Changes from Previous Version

Appendix A. Bug Reporting  
   A.1. Originating Kernel Bug Reports  
   A.2. Usage of DARK Bug Report Form Fields

Appendix B. Bug Report Summary

Appendix C. INSTALL.COM

Appendix D. BUILD.COM

Appendix E. SEI Release Procedure  
   E.1. New Release  
   E.2. Make a Release Tape  
   E.3. RELEASE.COM  
   E.4. GEN_RPTS.COM  
   E.5. MAKE_TAPE.COM

Appendix F. Tape Contents
List of Figures

Figure 3-1: Hardware Configuration 7
Figure 3-2: Software Configuration 7
Version Description and Installation Guide

Abstract: This document characterizes a specific version of the Distributed Ada Real-Time Kernel (DARK) software artifact and supplies documentation for its installation and use. This document is geared toward: the engineer responsible for installing the Kernel, engineers responsible for porting and maintaining the Kernel, and engineers using the Kernel and needing an awareness of changes from the previous release.

1. Introduction

This document characterizes Version 3.0 of the Distributed Ada Real-Time Kernel (Kernel) and supplies documentation for its installation and use. The pertinent pieces of the Kernel environment are:

- The host system is a DEC VAX operating under VMS 5.0 (the specific host used at the SEI is a MicroVAX II operating under MicroVMS 5.0).
- The Kernel exists in two versions:
  1. An implementation in Ada using the TeleSoft Telegen2 V3.22 Ada Development System (of which, the OASYS XA68000 V4.12 Cross-Assembler is a part) targeted to the network of MC68020 processors and peripherals described in the Kernel Architecture Manual.
  2. An implementation in Ada using the VAX Ada V1.5 and VAX Macro targeted to the VAX-11 architecture running VMS V5.0.
- The Kernel User's Manual [KUM 89] provides the detailed information needed to understand and use Version 3.0 of the Kernel.

1.1. Audience

There are two groups of people who need the information in this document:

1. The engineer responsible for installing the Kernel.
2. Kernel users and those who need to know the changes from the previous release.
1.2. Related Documents

[KFD 88] Bamberger, J., Colket, C., Firth, R., Klein, D., Van Scoy, R.  
*Kernel Facilities Definition.*  

[DARK 88] Bamberger, J., Colket, C., Firth, R., Klein, D., Van Scoy, R.  
*Distributed Ada Real-time Kernel.*  

[KUM 89] Bamberger, J., Coddington, T., Firth, R., Klein, D., Stinchcomb, D., Van Scoy, R.  

[KAM 89] Bamberger, J., Colket, C., Firth, R., Klein, D., Van Scoy, R.  

[PORT 89] Bamberger, J., Coddington, T., Firth, R., Klein, D., Stinchcomb, D., Van Scoy, R.  
*Kernel Porting and Extension Guide.*  


[DEC 84] *Guide to Using DCL and Command Procedures on VAX/VMS.*  
Order number: AA-Y501-A-TE.


Digital Equipment Corp., February 1985,  
Order number: AA-EG29A-TE.

[VAX ADA DEV] *Developing Ada Programs on VAX/VMS.*  
Order number: AA-EF86A-TE.

Order number: AA-EF88A-TE.
1.3. Conventions

*Italic* font is used for all Kernel-specific items, such as package and primitive names.
2. Acquisition

The Kernel can be acquired from the Software Engineering Institute. For additional information, please contact:

Judy Bamberger  
Software Engineering Institute  
Pittsburgh PA 15213  
412-268-5795  
ARPANET: bamberg@sei.cmu.edu

The standard Kernel release package consists of:

- 1 Magnetic tape in 6250 bpi Vito Backup format
- 1 each of the following documents:
  - Version Description and Installation Guide (this document, [VDIG 89])
  - Kernel Facilities Definition ([KFD 88])
  - Kernel Architecture Manual ([KAM 88])
  - Kernel User's Manual ([KUM 89])
  - Kernel Porting and Extension Guide ([PORT 89])
3. Installation

This installation procedure will install both the VMS and 68020 versions. To compile, link, and execute the Kernel requires the hardware and software configurations shown in Figures 3-1 and 3-2.

<table>
<thead>
<tr>
<th>Type</th>
<th>Required Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>DEC VAX</td>
</tr>
<tr>
<td></td>
<td>Magnetic tape drive (9 in or TK50)</td>
</tr>
<tr>
<td>Target</td>
<td>68020 configuration described in [KAM89] or another suitable bare-board 68020</td>
</tr>
</tbody>
</table>

**Figure 3-1:** Hardware Configuration

<table>
<thead>
<tr>
<th>Type</th>
<th>Required Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>VMS V5.1</td>
</tr>
<tr>
<td></td>
<td>VAX Ada V1.5</td>
</tr>
<tr>
<td></td>
<td>Backup V5.0</td>
</tr>
<tr>
<td></td>
<td>TeleSoft V3.22, VAX/VMS to Embedded MC680X0 Targets</td>
</tr>
<tr>
<td>Target</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>All needed software contained in the TeleSoft package</td>
</tr>
</tbody>
</table>

**Figure 3-2:** Software Configuration

One optional DEC tool used by the development team is MMS, Module Management System, a standard VMS configuration tool. It is not required to install the Kernel, but a copy of the MMS description (i.e., the "make file") used to recompile the Kernel is included in the release.
The final directory structure generated by installing all three Kernel releases looks like:

```
SEI DARK

+------------------+
|                  |
|                  |
| COMMANDS V1 V2 V3|
|                  |
|                  |
| 00 00            |
|                  |
|                  |
| 00 COMMANDS     |
|                  |
+------------------+

68K COMMON VMS VMS_ADALIB
```
3.1. System Manager Actions

The following actions require system privileges and must be performed by the local system manager.

1. Create a new directory [SEI_DARK]
   a. The creation of this directory occurs only on the first release of DARK installed.
   b. Ensure that all users who need access to the Kernel can access the [SEI_DARK] directory (the required access modes are: w:re and g:re; the installation procedure will take care of the protection of the subdirectories).

2. The VMS version of the Kernel relies on having its own logical name table. To create the needed table:

   ```
   SET PROC/PRIV=SYSPRV
   CREATE/NAM_TABLE/PARENT=LNS$SYSTEM_DIRECTORY/PROT=G:RMED LNS$DARK_NAMES
   SET PROC/PRIV=NOSYSPRV
   ```
   a. This table will need to be created when V3.00 is installed for the first time.
   b. This table will need to be re-created after every system reboot. The above CREATE command should be placed in the appropriate system startup command file to ensure this.

3.2. DARK Installation Actions

The following actions require no special privileges and can be performed anytime after the system manager actions are complete.

1. Log in to the VMS system.
2. Set your default to the SEI release directory:

   ```
   SET DEF [SEI_DARK]
   ```
3. Create a logical name for the local tape drive:

   ```
   DEFINE MT <tape-drive name>
   ```
   The <tape-drive name> must be replaced by the appropriate device specification for the local system (at the SEI for example, this command looks like: DEFINE MT MUBO). This equivalence will then be used by the DCL command scripts that perform the installation.
4. Physically load the release tape on the drive, allocate the tape drive, and mount the tape:

   ```
   ALLOCATE MT:
   MOUNT/NOUNL/FOR MT:
   ```
5. Unload the DCL command scripts and support files from the release tape:

```
BACKUP/LOG/REPLACE MT:SCRIPTS.BAK/SAVE *
```

A log message is displayed on the terminal for each file unloaded from tape.

6. Complete the release-tape processing by invoking INSTALL.COM (shown in Appendix C):

```
@INSTALL V3.00
```

INSTALL.COM will:

a. Create the needed release directories.
b. Move the command and support files into [SEI_DARK.COMMANDS].
c. Unload all the Kernel sources from the release tape.
d. Generate a log message on the terminal for each step of the process.

7. Complete the installation process by building the VMS\(^1\) and TeleSoft\(^2\) compilation libraries from the new Kernel sources. This is done by invoking BUILD.COM (shown in Appendix D):

```
@[SEI_DARK.V3.COMMANDS]BUILD V3.00
```

BUILD.COM will:

a. Create a new VMS compilation library.
b. Assemble all the VMS code and import it into the VMS library.
c. Compile all Ada code into the VMS library.
d. Create a new TeleSoft compilation library.
e. Assemble all the 68020 code and import it into the TeleSoft library.
f. Compile all Ada code into the TeleSoft library.
g. Link the network processor software.
h. Generate a log message on the terminal for each step of the process.

After completing the installation process:

- All the 68020-specific code resides in [SEI_DARK.V3.00.68K].
- All the VMS-specific code resides in [SEI_DARK.V3.00.VMS].
- All the common code resides in [SEI_DARK.V3.00.COMMON].
- The TeleSoft compilation library resides in [SEI_DARK.V3.00.68K].
- The VMS compilation library resides in [SEI_DARK.V3.00.VMS_ADALIB].

---

\(^1\)The VMS build can only be performed if the VAX Ada V1.5 system is installed on the local system.

\(^2\)The 68020 build can only be performed if the TeleSoft Ada V3.22 system is installed on the local system.
Some additional items of note:

1. In the event of a disaster, this same installation process can be used to regenerate any release from its release tape.

2. A listing of all the files contained on the tape is included as Appendix F of this document.

3. A listing of all the differences between V3.00 and V2.00 is not included in this document, but can be found in the configuration directory for the release. [SEI_DARK.Vx.yy.zz]DIFFERENCES.RPT shows the differences between releases, and [SEI_DARK.Vx.yy.zz]NEW_FILES.RPT lists all the files new to this release.

4. An electronic version of the problem reporting form is contained in [SEI_DARK.V3.COMMANDS]BUG.RPT.
4. Customization

Customization of the Kernel to accommodate the local environment and the application's needs are discussed in Chapter 3.3 and Appendix C of [KUM 89]. A complete discussion of the Kernel's target hardware configuration can be found in the *Kernel Architecture Manual*. 
5. Known Deficiencies

The following bug reports document known problems, deficiencies, and issues related to V3.00 of the Kernel (both 68020 and VMS versions). Included is the bug report tracking number, a summary "title" of the issue, and a brief description of the issue. Information about the status of these bug reports is available on request.

- 0013_SEI. Nproc won't permit single node reset
  This is a sporadic problem in the DARK testbed. It cannot be produced on a regular basis.

- 0019_SEI. Process Index Table (PIT) sizing incorrect
  This is related to 0032_SEI. It is an internal issue and should not manifest itself to users of the Kernel.

- 0032_SEI. GPTB maximum number of processes same
  See 0019_SEI.

- 0033_SEI. die and kill must release all resources
  Kernel primitives die and kill do not release semaphores, nor do they cancel any pending timeouts. As a result, the semaphores remain claimed, and the timeouts remain in the time event queue (and will expire silently when they occur).

- 0034_SEI. alarm expiry does not cancel pending claim
  The expiration of an alarm does not cancel any pending claim of a semaphore. As a result, when an alarm expires, the claim is still pending, and it remains up to the user to release as appropriate.

- 0035_SEI. RM.initialize time delays not accounted
  The Kernel currently does not propagate time delays around the network during initialization. These time deltas are to be computed using the logic analyzer. It has not been possible as yet to use the logic analyzer in such a manner as to obtain these time deltas. Therefore, during network initialization, when the Master Main Unit sends out the "go" message containing its view of time by which all subordinate Main Units initialize their clocks, the assumption is that there is zero time delta for each propagation of the "go" message.

- 0047_SEI. more than one claim of same semaphore allowed (deadlock)
  A sure-case deadlock situation is not detected by the Kernel. A process may request to claim a semaphore that it already has claimed. Depending on how MM.claim is called (i.e., with a timeout or not), this could result in a deadlock.

- 0048_SEI. overwrite old messages on queue overflow not implemented
  The only option for handling messages is to drop the newest message. The other required (per [KFD]) option, to overwrite oldest messages, is not provided.

- 0050_SEI. GIG interrupt table defaults misleading
  The current initial state of the Interrupt Table allows an application to bind to any interrupt not previously bound by the Kernel itself, including interrupts required by the Ada runtime. This potentially allows an application to act in a manner destructive of the integrity of the Kernel and Ada runtime. For the present, the application user should take care to bind only those interrupts
explicitly commented in internal Kernel package inames as available to the application, and any interrupts for devices that are part of the application-specific hardware.

- **0059_SEI. check_message, discard_message primitives required**
  To provide a clean interface between the Kernel primitives and the typed message passing described in [KUM 89], these two additional primitives should be provided. This interface would be a nice extension to the Kernel.

- **0072_SEI. size of Kernel datagram not computed correctly**
  The size of a Kernel datagram is tied to the size of an entry in the NCT. In the delivered configuration, with all generic parameters set as provided, this will not cause a problem. The problem will arise when/if the settings are such that another Kernel-to-Kernel message is larger than an NCT entry message. One simple case of this is the "process created" message, which contains the logical (i.e., string-valued) name of the Kernel process being created. If that logical name is sufficiently larger than the NCT entry, it is likely that an internal error may occur. Thus, extreme caution should be used when modifying the values of:

  - `GPMG.maximum_length_of_process_name_value`
  - `GNC/NC.maximum_length_of_processor_name_value`

- **0076_SEI. Time event queue performance improvements**
  Resources did not allow for requisite analysis and ensuing performance improvements of the algorithms that manipulate the time event queue. The key assumptions are identified, along with a number of alternatives to improve performance if required. These improvements would be a nice extension to the Kernel.

- **0077_SEI. CM.alloc_devicereceiver doesn't check if local PID**
  `GCM.allocate_device_receiver` does not check to ensure that the target process ID to be allocated to receive incoming messages from a non-Kernel device is, in fact, a locally created process.

- **0078_SEI. DGM.alloc dg selection of datagrams from large queue**
  **THIS IS A VMS VERSION OF 0069_SEI, WHICH WAS REPORTED AND FIXED IN 68020 VERSION.**
  There is a bug in `DGM.alloc dg`. The behavior is supposed to be as follows: when a small message (i.e., one that fits in a small datagram) is sent, `alloc dg` is supposed to dequeue a small datagram. If there are none available, `alloc dg` is supposed to return a large datagram, and then as a last resort, a Kernel datagram.

- **0079_FER. use of standard.long_integer**
  Use of package `standard long_integer` was detected in the following packages when porting the 68020 Kernel to the Rational: `MZ8305_definitions`, `PIO.toPIOcontrol_ptr`, function `unchecked_conversion` instantiation change to `HI.hw_long_integer`. 
• 0080_FER. long/short name mismatch (to_kernel_time)
  Mismatch of long name and short name in specification and body (respectively) of generic_time_globals.to_kernel_time (elapsed_time parameter).

• 0083_SEI. CM specification exceptions not equal to raised exceptions
  Spec comments for generic_communication_management not complete or incorrect. Note the following inconsistencies:
  1. Message_timed_out is not explicitly raised in gcm_body.
  2. Network_failure is not raised explicitly in gcm_body.
  3. In several places, reference is made to the subprogram allocate_device; however, there is no such subprogram.
  4. Receiver_dead can be raised during a call to send_message.
  5. Replacing_previous_allocation is explicitly raised in allocate_device_receiver and not in allocate_device.

• 0084_SEI. specification documentation not equal to code implementation
  The following specs contain documentation that does not match the code it is supposed to be describing:
  • (generic_)network_configuration:
    • NCT, logical_name: initial value is all blanks (not "none")
  • generic_process_table:
    • semaphore
      • number_of_waiting_processes: may also be decremented by die/kill (once BR 0033_SEI is fixed)
      • queue_head: may also be reset by die/kill (once BR 0033_SEI is fixed)
      • sema_previously_claimed: may also be reset by die/kill (once BR 0033_SEI is fixed); style of documentation not similar to others; missing who sets/resets
  • process_identifier: isn't it set to null now when pruning, per BR 0055_SEI?
  • process_attributes.process_initialization_status: documentation is wrong; only discusses "declared" and "created"; needs "remotely_created"
  • schedule_attributes.next: no documentation about changes
  • communication_attributes.maximum_message_queue_size: modified by die/kill
  • communication_attributes.message_queue: modified by die/kill
  • communication_attributes.current_send_buffer: modified by die/kill
  • pending_activity_attributes.pending_activity: reset by die/kill (once BR 0033_SEI is fixed)
• pending_activity_attributes.current_pending_message: for "safety," should this be reset for die/kill?

• pending_activity_attributes.current_receive_buffer_address: incomplete documentation; need to reset for die/kill

• pending_activity_attributes.current_receive_buffer_size: incomplete documentation; need to reset for die/kill

• pending_activity_attributes.alarm_event_id: reset by die/kill (once BR 0033_SEI is fixed)

• pending_activity_attributes.alarm_resumption_priority: reset by die/kill (once BR 0033_SEI is fixed)

• acknowledged_message_information*: interaction with die/kill (once BR 0033_SEI is fixed)

• semaphore_attributes.sema_last_claimed: set/use information missing

• semaphore_attributes.*: interaction with die/kill (once BR 0033_SEI is fixed)

• datagram_globals:

• comment before type datagram is ...: missing comment about "local"

• comment before header description: missing comment about "local"

• general concern: use of "we"

• generic_interrupt_globals:

• interrupt_table documentation is incomplete

• set/used information is not present.

• 0085_SEI: subset of 0084_SEI that MUST be FIXED

the following specs contain documentation that does not match the code it is supposed to be describing:

• (generic_)network_configuration:

• NCT logical_name: initial value is all blanks (not "none")

• generic_process_table:

• process_attributes.process_initialization_status: documentation is wrong; only discusses "declared" and "created"; needs "remotely_created"

• semaphore_attributes.sema_last_claimed: set/use information missing

• datagram_globals:
• comment before type datagram 's ...: missing comment about "local"
• comment before header description: missing comment about "local"
• general concern: use of "we"

• generic_interrupt_globals:
  • interrupt_table documentation is incomplete
  • set/used information is not present.
6. Changes from Previous Version

A log of all the new files can be found on the release tape in file NEW_FILES.RPT. A line-by-line listing of the differences between V2.00 and V3.00 can also be found on the release tape, in file DIFFERENCES.RPT. These report files are loaded into the configuration directory, along with all the Kernel sources, as part of the installation process.

In addition, the following bug reports document problems detected in V2.00 and corrected in the V3.00 release.

- 0016_SEI. GQM performance is linear. Actual fix exists, but was not completed in time to make it into the V2.00 release.
- 0037_SEI. Tool interface not implemented
  Package tool_interface is not provided with V2.00 of the Kernel. However, all key data structures are potentially visible (the application may import them via appropriate WITH clauses) and use appropriate _debug print routines to view the contents of them. In no case should the application attempt to modify the contents of any internal Kernel data structure, as that would invalidate the assumptions the Kernel makes about its own state.
- 0040_SEI. Check for null process ID as input parameter
  The Kernel does not verify that the input process ID values are not null (e.g., communication_management subprograms, send_message and send_message_and_wait). As a result, if an application does not declare a process ID to the Kernel, a constraint error may be raised when calling these Kernel primitives.
- 0049_SEI. Valid process ID for sender/receiver
  See 0040_SEI.
- 0069_SEI. DGM.alloc dg selection of datagrams from large queue
  Under heavy load situations, it appears that when the Nproc queue of small datagrams is exhausted, the Nproc dies instead of beginning to select datagrams from the queue of large datagrams. This may have been fixed as a side effect of fixes to other communication-related bug reports, but this has not yet been tested to our satisfaction.
- 0081_SEI. GPAR short names
  The modules generic_process_attribute_readers and gpar_body do not follow the Kernel style guideline of using short names.
- 0082_SEI. TI.begin/end to check illegal process ID
  Bug reports 0040_SEI and 0049_SEI are completed with the exception of the changes necessary to TI.begin_collection and TI.cease_collection. In order to close the open reports, this report is being filed to keep track of the remaining items on the list. These remaining items are still in violation of [KFD] 10.1.31.
  All DARK primitives that accept a process identifier as an input parameter should check for an uninitialized process identifier (they needn't check for an illegal one, just uninitialized). The affected primitives still not done are: TI.begin_collection and TI.cease_collection.
- 0085_SEI. VMS problem elaborating generics
Code that previously worked under TeleSoft/MC68020 now fails under VMS with the unhandled exception PROGRAM_ERROR.

The reason is that the Ada initialization code is attempting to perform generic instantiations before elaborating the corresponding generic bodies, this is erroneous under [RM 12.2(3)].

The reason this problem has not been seen before is that the VMS compiler elaborates program units in a different order than does the TeleSoft compiler. This is legitimate—such freedom is granted by [RM 10.5(2)].

- 0086_SEI. AST encaps setting interrupt_nesting wrong
  
  In the VMS version only, the AST encapsulation does not correctly set the interrupt_nesting level. This causes DARK code called from an AST handler to fail.

- 0087_SEI. create/use DARK_text_io (I/O being interrupted may be corrupting the stack)
  
  In the VMS implementation, the Ada I/O routines [RM 14] do not seem to be properly reentrant. Thus, if a DARK process is suspended during a call of an I/O operation, and another process then calls the same or another I/O routine, the Ada run time goes wrong and the DARK program fails in strange ways.

- 0089_SEI. AST encaps must fully save/restore context
  
  The VMS AST encapsulation handler, am.ast_handler_encapsulation, contains an error. The effect of this error is that, if an AST occurs in the middle of an interruptable instruction, the processor state is not fully saved and restored.

The following outstanding bug reports reference changes required to the Kernel User's Manual. These are integrated into the [KUM89], which accompanies the V3.00 release of DARK.

- 0029_SEI. KUM words too weak about set_alarm behavior
- 0038_SEI. text_io blocks; interferes with Kernel (KUM)
- 0040_SEI. check for null process ID as input parameter
- 0045_WIT. KUM initialization_order example confusing
- 0046_WIT. KUM typos
- 0049_SEI. valid process ID for sender/receiver
- 0052_SEI. KUM, 4.13.3, sync not allowed in i/h
- 0061_WIT. KUM.get_clock used but not in Kernel specifications anywhere
- 0062_WIT. KUM tailoring parameter description missing
Appendix A: Bug Reporting

A.1. Originating Kernel Bug Reports

Bugs are reported in the same manner, no matter what their source (e.g., internal SEI, DARK development team, acceptor site). Complete the first page (the first four sections on the figure) of the attached bug report form is completed, either manually or electronically. The bug report is then sent to the correct place at the SEI. For bugs submitted by Email, there is a dark_bugs account, which will be examined daily by the Bug Report Database Maintainer. For bugs submitted via U.S. postal service, there is a dark_bugs mail stop, which will be checked daily by the Bug Report Database Maintainer. For bugs submitted from within the SEI, either of the two methods above are acceptable, as well as Campus Mail to the Bug Report Database Maintainer. In any case, the DARK bug reports will make it to the Bug Report Database Maintainer.

A.2. Usage of DARK Bug Report Form Fields

1. Contact information to be completed by bug report originator when the bug report is generated.
   a. Reported by: Name of the originator of the bug report. This field must be completed.
   b. Phone: Phone number(s) where the originator of the bug report may be reached. This field must be completed.
   c. Email address: The electronic mail address of the originator of the bug report. This field must be completed if the originator has Email access to the SEI.
   d. Full address: Address to which any hard-copy correspondence about the bug or about DARK should be sent. This field must be completed by all non-SEI originators of bug reports.
   e. Date reported: Date (ymmd) when the bug report is generated. This field must be completed.

2. Problem description information to be completed by bug report originator when the bug report is generated.
   a. Severity: The impact the bug has on the bug originator. Options include:
      i. Stopped dead: Use of Kernel artifacts cannot continue; no workarounds, acceptable or not, are available.
      ii. Significant: Use of Kernel artifacts are severely affected; no acceptable workaround found, but can get by for now.
      iii. OK workaround: Use of Kernel artifacts hindered; acceptable workaround found, but would rather have the real capability.
iv. **FYI:** Use of Kernel artifacts unaffected, but this bug was found.

This field must be completed.

b. **Problem description:** A brief yet complete description of the observed problem (i.e., what is "broken").

c. **Proposed solution:** A brief statement of the anticipated fix (used for planning). This field should be completed by the originator of the bug report if the originator has a proposed solution; otherwise, this field need not be completed.

d. **Alternative(s):** A brief statement of any alternatives found by the originator of the bug report to "workaround" the bug. This field should be completed by the originator of the bug report if the originator has found an alternative; otherwise, this field need not be completed.

3. **Discovery information to be completed by the bug report originator when the bug report is generated.**

   a. **How discovered:** Indication of how the problem was discovered. This takes into consideration the strategy used to discover the bug and the diagnostic tools used.

      Strategies include:

      i. **Inspection:** Visual examination of the artifacts.

      ii. **Compilation:** Compiling the Kernel or any piece of code that imports the Kernel specifications.

      iii. **Execution:** Execution of any piece of code that uses Kernel capabilities.

      iv. **Formal proof:** Doing or examining the proof.

      v. **Other:** Provide description.

      Diagnostic tools include:

      i. **Unit test case:** Indication that unit testing caused the bug to surface.

      ii. **System test case:** Indication that system testing caused the bug to surface.

      iii. **Application:** Indication that other code caused the bug to surface.

      iv. **Other:** Provide description.

      This matrix must be completed.

   b. **Diagnostic tool:** Exact description of the tool that caused the bug to surface. Wherever possible, this should include a description or a copy of the program or process by which the problem can be reproduced to facilitate reproduction of the same problem on the DARK testbed at the SEI. This field must be completed.
4. **Identification of DARK artifacts information to be completed by the bug report originator when the bug report is generated.**

   a. **DARK artifact and version:** Full identification of the DARK artifact (e.g., document name, Kernel package name) and version identification (including date of issue). This field must be completed.

   b. **Files:** List of source files to which the bug has been tracked. This field should be completed by the originator of the bug report if the originator has determined this information; otherwise, this field need not be completed.

5. **SEI tracking information to be completed immediately by the maintainer of the bug report database when the bug report is entered into the database.**

   a. **Bug number:** Bug numbers are of the form nnnn_XXX, where:

      i. nnnn is some number in the range 0001 .. 9999.

      ii. XXX is a site-unique, three-letter acronym indicating the organization that originated the bug report. Examples include: SEI for the Software Engineering Institute, WEC for Westinghouse Electric Corporation.

      This field must be completed.

   b. **Date received:** Date (yymmdd) received at the SEI. This field must be completed.

   c. **Related bug reports:** Possibly amended when the bug is closed. Enumeration of any bug reports related to this one or opened via this one, and their status when closing this one; "n/a" if there are none. This field must be completed.

   d. **Related design decisions:** Possibly amended when the bug report is closed. Enumeration of any design decisions related to this one, opened via this one, and their status when closing this one; "n/a" if there are none. This field must be completed.

   e. **Defect category:** Options include:

      i. **Requirements:** There is something wrong (missing, contradictory, incomplete, etc.) with the Kernel requirements. Applicable artifacts: [KFD].

      ii. **Design:** There is something wrong (missing, contradictory, incomplete, etc.) with the Kernel design. Applicable artifacts: [KAM], structure of Kernel code.

      iii. **Code:** There is something wrong (missing, extra, etc.) in the Kernel code (Ada or assembly language). Applicable artifacts: code, test cases.

      iv. **Documentation:** There is something wrong (missing, contradictory, incomplete, etc.) with the in-line documentation of the code (Ada or assembly language). This includes all header commentary. Applicable artifacts: code.

      v. **Document:** There is something wrong (missing,
contradictory, incomplete, etc.) with a Kernel document. Applicable artifacts: [KFD], [KAM], [KUM], [VDIG].

This field must be completed.

6. **DARK work assignment information to be completed within five working days of receipt of the bug report.**

   a. **Assigned to:** Name of DARK team member responsible for making the fix, verifying the fix, and leading the effort to close out the bug report, all related bug reports, and all related design decisions. This field must be completed.

   b. **Date assigned:** Date (yymmdd) when the bug is assigned to a DARK team member for fixing. This field must be completed.

   c. **Priority:** Completed when the bug is assigned to a DARK team member for fixing. Options include:

      i. **High:** Fix these first.

      ii. **Medium:** Fix these next.

      iii. **Low:** Fix these if and only if there is time.

   This field must be completed.

   d. **Due date:** Date (yymmdd) by which the fix must be completed. This field must be completed.

7. **Actual fix information to be completed by the DARK team member assigned to fix the bug.**

   a. **Proposed fix:** Brief description of the anticipated fix and enumeration of any reasonable alternatives and why these alternatives may need to be taken instead (used for planning). This should be completed within five working days of being assigned responsibility for fixing the bug and before the actual fix is applied. This also includes identifying any workarounds that can be taken by the user community to avoid the bug if the fix cannot be applied in time. This field must be completed.

   b. **Actual fix:** Description of the actual fix. This field must be completed.

   c. **Actual fix date:** Date (yymmdd) the DARK team member completed the fix. This field must be completed.

   d. **Directories:** Indication of the DARK artifacts affected by the fix. This field must be completed.

   e. **Files:** Indication of all files affected by the fix. This field must be completed.

   f. **Problem:** The actual problem (which may be identical to the reported problem description). This field must be completed.

   g. **Solution:** The actual solution (which may be identical to the reported proposed solution). This field must be completed.
8. Closing information to be completed by the Configuration Control Board (CCB)\(^3\) when the bug report is ready to be closed.
   
   a. **Close date:** Date (yymmdd) the bug report is verified as being closed. This field must be completed.
   
   b. **Verified by:** Name of the CCB member verifying the bug as being closed. This field must be completed.
   
   c. **How verified:** Indication of what criteria were used to verify the fix was made as reported (e.g., inspection of documentation, examination of test case results). This field must be completed.
   
   d. **Checklist:** Helpful clues for the CCB and the bug fixer to ensure that all propagated and related modifications have, in fact, been made. This field is optional.

9. Available version information to be completed when the next deliverable version of the relevant project artifact is generated.

   a. **Available in DARK version:** Completed by the DARK project leader when planning for the next release. This field must be completed.
   
   b. **Applicable VDIG version and section:** Completed by the individual responsible for generating the VDIG portion of the KUM in preparation for release of the Kernel. This field must be completed.

---

\(^3\)The CCB currently comprises the DARK project leader and the maintainer of the DARK project bug report database.
# DARK Bug Report Form (Part 1)

| Reported by | | |
| Phone | | |
| Email address | | |
| Full address | | |
| (external only) | | |

| Date reported | | |

**Severity**

- Stopped dead()
- Significant()
- OK work-around()
- FYI()

**Problem description**

**Proposed solution**

**Alternative(s)**

**How discovered**

<table>
<thead>
<tr>
<th>STRATEGY</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOOL</td>
</tr>
<tr>
<td>Proof</td>
</tr>
<tr>
<td>Unit Test</td>
</tr>
<tr>
<td>System Test</td>
</tr>
<tr>
<td>Application</td>
</tr>
<tr>
<td>Other</td>
</tr>
</tbody>
</table>

**Diagnostic tool**

- DARK artifact
- and version

- Files
# DARK Bug Report Form (Part 2)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug number</td>
<td></td>
</tr>
<tr>
<td>Date received</td>
<td></td>
</tr>
<tr>
<td>Related bug reports</td>
<td></td>
</tr>
<tr>
<td>Related design decisions</td>
<td></td>
</tr>
<tr>
<td>Defect category</td>
<td>Requirements() Design() Code() Documentation() Document()</td>
</tr>
<tr>
<td>Assigned to</td>
<td></td>
</tr>
<tr>
<td>Date assigned</td>
<td></td>
</tr>
<tr>
<td>Priority</td>
<td>High() Medium() Low()</td>
</tr>
<tr>
<td>Due date</td>
<td></td>
</tr>
<tr>
<td>Proposed fix</td>
<td></td>
</tr>
<tr>
<td>Actual fix</td>
<td></td>
</tr>
<tr>
<td>Actual fix date</td>
<td></td>
</tr>
<tr>
<td>Directories</td>
<td></td>
</tr>
<tr>
<td>Files</td>
<td></td>
</tr>
<tr>
<td>Problem</td>
<td></td>
</tr>
<tr>
<td>Solution</td>
<td></td>
</tr>
<tr>
<td>Close date</td>
<td></td>
</tr>
<tr>
<td>Verified by</td>
<td></td>
</tr>
<tr>
<td>How verified</td>
<td></td>
</tr>
<tr>
<td>Checklist</td>
<td>KFD() KAM() KUM()</td>
</tr>
<tr>
<td></td>
<td>KS(): Module contents() References() History() Notes()</td>
</tr>
<tr>
<td></td>
<td>IKS(): Module contents() References() History() Notes()</td>
</tr>
<tr>
<td></td>
<td>Subprogram header(): Req() Prim() Time() Parameters()</td>
</tr>
<tr>
<td></td>
<td>Precon() Actions() Postcon()</td>
</tr>
<tr>
<td></td>
<td>Errors() Examples() Notes()</td>
</tr>
<tr>
<td></td>
<td>Body(): PDL()</td>
</tr>
<tr>
<td>Available in</td>
<td></td>
</tr>
<tr>
<td>DARK version</td>
<td></td>
</tr>
<tr>
<td>Applicable VDIG</td>
<td>version and section</td>
</tr>
<tr>
<td>Send this form to:</td>
<td>DARK BUGS</td>
</tr>
<tr>
<td></td>
<td>c/o Judy Bamberger</td>
</tr>
<tr>
<td></td>
<td>Software Engineering Institute</td>
</tr>
<tr>
<td></td>
<td>Pittsburgh PA 15213</td>
</tr>
<tr>
<td></td>
<td>412-268-5795</td>
</tr>
<tr>
<td>Email:</td>
<td><a href="mailto:dark-bugs@sei.cmu.edu">dark-bugs@sei.cmu.edu</a></td>
</tr>
</tbody>
</table>

CMU/SEI-89-TR-20 29
Appendix B: Bug Report Summary

This appendix summarizes all known Kernel problems. It provides the following information:

- **BR#:** bug report tracking number.
- **Stat:** status of the bug report:
  - **clo:** closed. The problem has been corrected in the current released version of the Kernel.
  - **asg:** assigned. The problem has been assigned to a project member for further study.
  - **una:** unassigned. The problem has been identified, but no one is currently studying the problem.

- **Assignee:** the user id of the individual assigned to fix the problem.
- **Title:** a brief one-line description of the problem.

A full bug report on any problem listed below is available on request.

<table>
<thead>
<tr>
<th>BR#</th>
<th>stat</th>
<th>assignee</th>
<th>title</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001_SEI</td>
<td>clo</td>
<td>firth</td>
<td>GIM documentation (error checking)</td>
</tr>
<tr>
<td>0002_SEI</td>
<td>clo</td>
<td>firth</td>
<td>GIM disable extra checking</td>
</tr>
<tr>
<td>0003_SEI</td>
<td>clo</td>
<td>rlvs</td>
<td>default timeslice quantum not tailorable</td>
</tr>
<tr>
<td>0004_SEI</td>
<td>clo</td>
<td>dvk</td>
<td>GTSM documentation (quantum is MINimum)</td>
</tr>
<tr>
<td>0005_SEI</td>
<td>clo</td>
<td>rlvs</td>
<td>GTM synchronize without timeout needed</td>
</tr>
<tr>
<td>0006_SEI</td>
<td>clo</td>
<td>bamberg</td>
<td>GST documentation (priority model)</td>
</tr>
<tr>
<td>0007_SEI</td>
<td>clo</td>
<td>bamberg</td>
<td>GPAR documentation (exceptions)</td>
</tr>
<tr>
<td>0008_SEI</td>
<td>clo</td>
<td>dvk</td>
<td>ST documentation (old generics)</td>
</tr>
<tr>
<td>0009_SEI</td>
<td>clo</td>
<td>firth</td>
<td>GTM documentation (read clock)</td>
</tr>
<tr>
<td>0010_SEI</td>
<td>clo</td>
<td>bamberg</td>
<td>GPTB documentation (stack_high_address)</td>
</tr>
<tr>
<td>0011_SEI</td>
<td>clo</td>
<td>firth</td>
<td>GKT documentation (should be 150_000 years)</td>
</tr>
<tr>
<td>0012_SEI</td>
<td>clo</td>
<td>bamberg</td>
<td>GPAM documentation (exceptions)</td>
</tr>
<tr>
<td>0013_SEI</td>
<td>asg</td>
<td>dvk</td>
<td>Nproc won’t permit single node reset</td>
</tr>
<tr>
<td>0014_SEI</td>
<td>clo</td>
<td>tac</td>
<td>GCM print to be removed</td>
</tr>
<tr>
<td>0015_SEI</td>
<td>clo</td>
<td>tac</td>
<td>send_message_and_wait with zero timeout</td>
</tr>
<tr>
<td>0016_SEI</td>
<td>clo</td>
<td>rlvs</td>
<td>GCM performance is linear</td>
</tr>
<tr>
<td>0017_SEI</td>
<td>clo</td>
<td>rlvs</td>
<td>GCM doesn’t remove TEQ pending alarm event</td>
</tr>
</tbody>
</table>
0018 SKI co rlvs unneeded layer of code with timers

0019 SKI asg rlvs PIT sizing incorrect

0020 SKI clo rlvs documentation of decl/create process

0021 SKI clo tac send * to accommodate tag-only msg better

0022 SKI clo rlvs code reorg PIT/GMC

0023 SKI clo rlvs update intro to KFD

0024 SKI clo rlvs update primitive part of KFD

0025 SKI clo tac GCM/bus_ic, local optimizations, ACK/NAK

0026 SKI clo tac priority inversion for send/receive

0027 SKI clo firth send of large message = constraint_error

0028 SKI clo firth provide traceback when process dies

0029 SKI clo bamberg KUM words too weak about set_alarm behavior

0030 SKI clo rlvs implement non-Kernel devices

0031 SKI clo rlvs synchronize doesn't work from Main Unit

0032 SKI asg rlvs GPTB: maximum number of processes same

0033 SKI asg rlvs die/kill must release all resources

0034 SKI asg rlvs alarm expiry does not cancel pending claim

0035 SKI asg rlvs initialize time delays not accounted

0036 SKI clo firth floating point coprocessor state not saved

0037 SKI clo rlvs tool interface not implemented

0038 SKI clo bamberg text_iob blocks; interferes with Kernel (KUM)

0039 SKI clo dvk make Nproc statistics available

0040 SKI clo dvk check for null process ID as input parameter (code) (documentation)

0041 SKI clo tac initial asynchronous send is lost

0042 SKI clo rlvs die called within receive DG i/b

0043 SKI clo rlvs KFD changes: chapters 5, 6, 7

0044 SKI clo rlvs bamberg KFD changes done; KUM must be examined for possible impacts

0045 SKI clo dlrs data lost for longword+1 length messages

0046 SKI clo bamberg KUM initialization_order example confusing
<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0047 SKI asg</td>
<td>firth</td>
</tr>
<tr>
<td>0048 SKI asg</td>
<td>tac</td>
</tr>
<tr>
<td>0049 SKI clo</td>
<td>dvk</td>
</tr>
<tr>
<td>0049 SKI clo</td>
<td>bamberg</td>
</tr>
<tr>
<td>0050 SKI asg</td>
<td>firth</td>
</tr>
<tr>
<td>0051 SKI clo</td>
<td>firth</td>
</tr>
<tr>
<td>0052 SKI clo</td>
<td>bamberg</td>
</tr>
<tr>
<td>0053 SKI clo</td>
<td>rlve</td>
</tr>
<tr>
<td>0053 SKI clo</td>
<td>bamberg</td>
</tr>
<tr>
<td>0054 SKI clo</td>
<td>rlve</td>
</tr>
<tr>
<td>0054 SKI clo</td>
<td>bamberg</td>
</tr>
<tr>
<td>0055 SKI clo</td>
<td>rlve</td>
</tr>
<tr>
<td>0056 SKI clo</td>
<td>rlve</td>
</tr>
<tr>
<td>0057 SKI clo</td>
<td>firth</td>
</tr>
<tr>
<td>0058 SKI clo</td>
<td>firth</td>
</tr>
<tr>
<td>0059 SKI una</td>
<td></td>
</tr>
<tr>
<td>0060 SKI clo</td>
<td>tac</td>
</tr>
<tr>
<td>0061 WIT clo</td>
<td>bamberg</td>
</tr>
<tr>
<td>0062 WIT clo</td>
<td>bamberg</td>
</tr>
<tr>
<td>0063 SKI clo</td>
<td>rlve</td>
</tr>
<tr>
<td>0064 SKI clo</td>
<td>dga</td>
</tr>
<tr>
<td>0065 SKI clo</td>
<td>firth</td>
</tr>
<tr>
<td>0066 SKI clo</td>
<td>firth</td>
</tr>
<tr>
<td>0067 SKI clo</td>
<td>firth</td>
</tr>
<tr>
<td>0068 SKI clo</td>
<td>firth</td>
</tr>
<tr>
<td>0069 SKI clo</td>
<td>tac</td>
</tr>
<tr>
<td>0070 SKI clo</td>
<td>tac</td>
</tr>
<tr>
<td>0071 SKI clo</td>
<td>firth</td>
</tr>
<tr>
<td>0072 SKI asg</td>
<td>tac</td>
</tr>
<tr>
<td>0073 SKI clo</td>
<td>rlve</td>
</tr>
</tbody>
</table>
0074_SEI clo  tac  constraint error/mismatch of data types
0075_SEI clo  rlvs  SCR run queue performance improvements
0076_SEI una  rlvs  T2Q performance improvements
0077_SEI una  CM.alloc_dev_rcvr doesn't check if local PID
0078_SEI una  DCM.alloc_dg selectn of datagrams fm large Q
0079_FER una  use of standard.long_integer
0080_FER una  long/short name mismatch (to Kernel time)
0081_SEI clo  dvk  gpar short names
0082_SEI clo  rlvs  TI.begin/end to check illegal process ID
0083_SEI una  CM.specs.exceptions /= raised exceptions!
0084_SEI una  spec docn /= code impln!
0085_SEI clo  rlvs  VMS problem elaborating generics
0086_SEI clo  firth  AST encaps setting interrupt_nesting wrong
0087_SEI clo  rlvs  create/use DARK_text_io (IO being bamberg interrupted maybe corrupting the stack)
0088_SEI asg  firth  subset of 0084_SEI that MUST be FIXED!
0089_SEI clo  firth  AST encaps must fully save/restore context
Appendix C: INSTALL.COM

$..........................................................$  
$ Module: Install.com  
$ Purpose:  
$ This command procedure is used by a recipient of a DARK release tape to  
$ install the software.  
$ Parameters:  
$ pl: version being installed.  
$ Revision History:  
$ 14-fab-89 rlv created  
$ 12-oct-89 rlv stream lined and modified to account for  
$ vms version  
$ $..........................................................$  
$!!! DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG !!!!!!$  
$!!! define mt ps:[DARK_CM.release.testing]$  
$!!! define mt stub0:$  
$!!! DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG !!!!!!$  
$ set prot=(r:rwed, o:rwed, G:rwed, W:*)/default$  
$ set noon$  
$!!! Obtain the version being installed.$  
$!!!$  
$ version := 'pl$  
$ loop_back$:  
$ write sys$output "$"  
$ if version .eqs. ":" then -  
$ inquire/nopunc version "Which version are you installing [Vx.yy]: "$  
$ if version .eqs. ":" then goto loop_back  
$ command_local = f$parse(""version","","NAME") + ",commands"$  
$!! Check to see if the indicated release currently exists.$  
$!!$  
$ call check_release$  
$!!$  
$!! Create the configuration directory(s) to hold the release.$  
$!!$  
$ create/dir ['version']$  
$ create/dir ['command_local']$  
$ create/dir ['version'.68k]$  
$ create/dir ['version'.common]$  
$ create/dir ['version'.vms]$  
$ create/dir ['version'.vms_adalib]$  
$!! Store the commands and support files in their home (and clean up).$  
$!!$  
$ rename/log *.*.com,*.*.rpt ['command_local']*$  
$ purge ['.'command_local']$  
$!! Move to the configuration directory and copy in all the needed sources$  
$!! from the release tape.$  
$!!$  
$ set default ['.'version'.68k]$  
$ set prot=(w:rwed,o:rwed,g:rwed) *.*
$ backup/log/replace mt:68k_sources_bak/save_set *.*
$ set prot=(w:r,o:rwe,g:rwe) *.*
$!
$ set default [-.common]
$ set prot=(w:rwed,o:rwed,g:rwed) *.*
$ backup/log/replace mt:cmn_sources_bak/save_set *.*
$ set prot=(w:r,o:rwe,g:rwe) *.*
$!
$ set default [-.vms]
$ set prot=(w:rwed,o:rwed,g:rwed) *.*
$ backup/log/replace mt:vm_sources_bak/save_set *.*
$ set prot=(w:r,o:rwe,g:rwe) *.*
$!
$ Cleanup...
$!
$ set default [-]
$ write sys$output ""
$ write sys$output "DARK release 'version' now installed."
$ write sys$output ""
$ dismount/nounl mt:
$ deallocate mt:
$ exit
$!
$ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
$!
$ ! Module: Check_release
$!
$ ! Purpose:
$ ! This procedure checks to see if the release currently exists.
$ ! If it does, it ensures that the release is to be over written
$ ! before allowing the installation process to continue.
$!
$ ! Parameters:
$ ! none.
$!
$ ! Revision History:
$!
$ ! 14-feb-89 rlvs created
$!
$ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
$!
$ ! check_release: Subroutine
$!
$ ! If the version does not exist, then proceed with installation of
$ ! the new release.
$!
$ ! release_version := ['f$parse(version,,"name")']
$ ! release_dir := 'f$extract(1,10, f$parse(version,,"type"))'.dir
$ ! release := 'release_version'.'release_dir'
$ ! if f$search('"'"release""') .eqs. "" then exit
$!
$ ! If the release already exists, then there was either a typo above, or
$ ! the existing version needs to be re-created. In either case, we stop
$ ! the installation process and let the user make the next move.
$!
$ ! existing_release:
$ ! write sys$output ""
$ ! write sys$output "Version 'version' already exists..."
$ ! inquire/nopunc kill "Do really wish to over write it [Y/N]: "
$ ! if kill .eqs. "N" then stop
$!
If an existing release is to be re-created, let's really make sure...

write sys$output ""
inquire/nopunc kill "Are you positive you want to over write it [Y/N]: "
if kill .eqs. "N" then stop

Obliterate the existing release... in a noisy manner, so that user knows what is going on

set default [.'version']
set prot=(w:rwed,o:rwed,g:rwed) *.*
delete/log [...]*..*;
delete/log [...]*..*;
set default [sei_dark]
exit
end subroutine
Appendix D: BUILD.COM

$ Module: Build.com

$ Purpose:
$ This command procedure is used by a recipient of a DARK release tape to
$ build the VMS and TeleSoft compilation libraries.

$ Parameters:
$ pl: the version being built

$ Revision History:
$ 14-feb-89  plvs created
$ 12-oct-89  plvs streamlined and modified to account for
$ vms version

$ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
$ !!! DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG !!!
$ !!! root := "dark.release"
$ !!! DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG DEBUG !!!
$ !!! Set up the needed global symbols
$ !!!
$ !!! root := "sei_dark"
$ !!!
$ !!! Obtain the version being installed
$ !!!
$ set noon
$ version := 'pl'
$ loop_back:
$ write sys$output ""
$ if version .eqs. "" then -
$ inquire/nopunc version "Which version are you building [Vx.yy]: "
$ if version .eqs. "" then goto loop_back
$ !!!
$ !!!
$ call Build_VMS
$ call Build_TeleSoft
$ !!!
$ exit
$ !!!

$ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
$ !
$ VMS build procedure
$ !
$ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
$ !
$ Build_VMS: Subroutine
$ !
$ set def sys$disk:['root'.'version']
$ !
$ Issue some needed advice...
$ !
$ Write sys$output ""
$ Write sys$output "Please ensure that VAX Ada V1.5 and VAX ACS"

CMU/SEI-89-TR-20
Write sys$output "have been installed on this system..."
Write sys$output ""
Write sys$output "Is the VAX Ada system installed [Y/N] "
if junk .eqs. "N" then exit

Create the VMS compilation library

write sys$output ""
write sys$output "Creating VMS compilation library"
write sys$output ""
acs create library [.vms_adalib]
acs set library [.vms_adalib]

Define the needed search paths...
d tells VMS to search as follows:
[SEI_DARK.Vx.yy.vms],[SEI_DARK.Vx.yy.common]
cms tells VMS to search as follows:
[SEI_DARK.Vx.yy.vms]

define d sys$disk:['root'.‘version’.vms],
   sys$disk:['root’.‘version’.common]
define cms sys$disk:['root’.‘version’.vms]

Execute the command file to compile & assemble the sources.
If the MUS utility is available, this may also be done by
issuing the following command:
MUS/action/from_source/descr=[SEI_DARK.Vx.yy.vms]dark.mms DARK

Note that the D and CMS search paths above must still be defined
for MUS to operate correctly

set default [.vms]
@coms:dark

Cleanup...

set def [--]
purge
ext

Cleaning...

TeleSoft build procedure

Build_TeleSoft: Subroutine

set def sys$disk:['root’.‘version’.68k]

Issue some needed advice...

Write sys$output ""
Write sys$output "Please ensure that TeleSoft V3.22 Ada for VAX/VMS to 680X0"
Write sys$output "has been installed on this system..."
Write sys$output ""
inquire/nopunc junk "Is the TeleSoft system installed [Y/N] "
if junk .eqs. "N" then exit
!
Set up the master TeleSoft library; this library can then be referenced
by local or project libraries as the default source of DARK object modules
!
write sys$output ""
write sys$output "Creating TeleSoft compilation library"
write sys$output ""
open/write lib_file liblst.slb
write lib_file "-- Create local library"
write lib_file " name: adalib"
write lib_file "-- 68020 specific libraries"
write lib_file " name: tsada$dir:math68020 -- Math Libraries"
write lib_file " name: tsada$dir:cgs68020 -- Code Generator Support"
write lib_file " name: tsada$dir:tsa68020rtl -- Software Floating Point"
write lib_file ""
close lib_file
write lib_file tsada/e68/create adalib
!
!
Define the needed search paths...
!
d tells VMS to search as follows:
!
[SEI_DARK.Vx.yy.68k], [SEI_DARK.Vx.yy.common]
!
comes tells VMS to search as follows:
!
[SEI_DARK.Vx.yy.v68k]
!
!
define d sys$disk:['root'.'version'.68k], -
sys$disk:['root'.'version'.common]
!
!
Execute the command file to compile & assemble the sources.
!
If the MMS utility is available, this may also be done by
!
issuing the following command:
!
MMS/action/from_source/descr=[SEI_DARK.Vx.yy.68k]dark.mms DARK
!
!
Note that the D and COMS search paths above must still be defined
!
for MMS to operate correctly
!
!
@coma:dark
!
Cleanup...
!
set def [--]
purge
exit
Appendix E: SEI Release Procedure

E.1. New Release

To create a new release of the Kernel, the following steps are needed:

1. Log in to [DARK_CM] on AESTSC.
2. Create the release by executing the release command script (shown in detail in Section E.3).
   
   ```
   @ [DARK_CM.RELEASE]RELEASE
   ```
3. Cut the release tape by executing the maketape command script (shown in detail in Section E.5 and following the instructions it provides).
   
   ```
   @ [DARK_CM.RELEASE]MAKE_TAPE
   ```
4. Bundle the tape with the documents specified in Chapter 2 and ship as needed.

E.2. Make a Release Tape

To create a new release of the Kernel, the following steps are needed:

1. Log in to [DARK_CM] on AESTSC.
2. Cut the release tape by executing the make_tape command script (shown in detail in Section E.5 and following the instructions it provides).
   
   ```
   @ [DARK_CM.RELEASE]MAKE_TAPE
   ```
3. Bundle the tape with the documents specified in Chapter 2 and ship as needed.
E.3. RELEASE.COM

Program: RELEASE.COM
Purpose: This command file configures a new DARK release in a safe place.
Revision History:
20-feb-89 rlve created
12-oct-99 rlve streamlined and modified to account for vms version

Revision History:

WRITE SYSS$OUTPUT ""
INQUIRE/NOPUNC VERSION "What version is this [Vz.yy]: "
WRITE SYSS$OUTPUT ""
IF VERSION .EQS. "" THEN EXIT
MOVE to the configured release directory.
SET DEFAULT [.RELEASE]
CREATE/DIR [. 'VERSION]
CREATE subdirectories to hold the different version of the Kernel.
CREATE/DIR [. 'VERSION.68K]
CREATE/DIR [. 'VERSION.COMMON]
CREATE/DIR [. 'VERSION.VMS]

Populate the 68K release directory with the updated files.
SET DEFAULT [. 'VERSION.68K]
COPY DARK$WORK_68KREP:*.*

Populate the common release directory with the updated files.
SET DEFAULT [. COMMON]
COPY DARK$WORK_REP:*.*.ADA *

Populate the VMS release directory with the updated files.
SET DEFAULT [. VMS]
COPY DARK$WORK_VMSREP:*.* *

Generate the 68K release reports & clean up
WRITE SYSS$OUTPUT ""
WRITE SYSS$OUTPUT "Please edit [DARK_CM.RELEASE]GEN_RPTS.COM to generate"
WRITE SYSS$OUTPUT "generate 68020 reports"
WRITE SYSS$OUTPUT ""
INQUIRE/NOPUNC junk "Return when ready..."
SET DEFAULT [. 68K]
$ [DARK_CM.RELEASE]GEN_RPTS
$!
$ PURGE/LO
$ RENAME *.*; *.*;1
$ SET PROT=(S:R,O:RWE,G:RE,W:RE) *.*
$!
$!
$ Clean up the common files
$!
$ SET DEFAULT [-.COMMON]
$ PURGE/LO
$ RENAME *.*; *.*;1
$ SET PROT=(S:R,O:RWE,G:RE,W:RE) *.*
$!
$ Generate the VMS release reports & clean up
$!
$ WRITE SYS$OUTPUT ""
$ WRITE SYS$OUTPUT "Please edit [DARK_CM.RELEASE]GEN_RPTS.COM to"
$ WRITE SYS$OUTPUT "generate VMS reports"
$ WRITE SYS$OUTPUT ""
$ INQUIRE/HOPUNC junk "Return when ready..."
$!
$ SET DEFAULT [-.VMS]
$ @[DARK_CM.RELEASE]GEN_RPTS
$!
$ PURGE/LO
$ RENAME *.*; *.*;1
$ SET PROT=(S:R,O:RWE,G:RE,W:RE) *.*
$!
$ SET DEFAULT [DARK_CM.RELEASE]
$ EXIT
E.4. GEN_RPTS.COM

Purpose:
This command file generates the difference and new file reports between the current DARK release and its immediate predecessor.

All paths are hard coded in this file. So, to generate the reports, requires that current, previous, and version be modified as needed.

Revision History:
12-oct-89 rlva created

Define current [dark_cm.release.v3.00.vm], [dark_cm.release.v3.00.common]
Define previous [dark_cm.release.v2.00]
Version = "V3.00"

Prepare the new_files report header

open/write new_files new_files.rpt
Write new_files "-------------------------------------"
Write new_files "'version' New File Report"
Write new_files "-------------------------------------"

Prepare the differences report header

open/write diffs differences.rpt
Write diffs "-------------------------------------"
Write diffs "'version' Difference Report"
Write diffs "-------------------------------------"
define sys$output diffs

on error then goto loop
loop:

next_file = $search("current:*,",1)
If next_file .eqs. "" then goto exit_point
next_file = $parse("'next_file',","NAME") + -
   $parse("'next_file',","TYPE")
If $parse("'next_file',","TYPE") .eqs. "RPT" then goto loop
If $search("previous:','next_file',",2) .eqs. ""
Then
Write new_files "'next_file'"
Else
Write diffs ""
Write diffs "------------------- Differences for 'next_file' "
Write diffs ""
diff current:'next_file' previous:'next_file'
Endif

loop back for the next file to process
$!
$ goto loop
$!
$!
$!
Normal exit point, we jump here when the end of list file is reached
$!
$ exit_point:
$!
$ Close new_files
$ Close diffs
$ Exit
E.5. MAKE_TAPE.COM

$ Module: make_tape.com
$ Purpose:
$ This command procedure writes the physical release tape.
$ Notes:
$ This command procedure can only be run from node AESTSC.
$ Revision History:
$ 20-fab-89 rlvs created
$ 12-Oct-89 rlvs updated to accommodate new VMS version
$
$ revision history
$ 20-fab-89 rlvs created
$ 12-Oct-89 rlvs updated to accommodate new VMS version
$ Define:
$ mt
$ version "V3.00"
$ on error then goto error_exit
$ set prot=(G:rewd)/default
$ commands = $parse("'version'," , "NAME") + ".commands"
$ Prepare the tape drive...
$ allocate mt:
$ init mt: DARK
$ mount/nounl/for mt: DARK
$ Save the needed command files, sources, and reports
$ set noon
$ backup/rewind/log/verify/ignore=label/list=tcl -
$داركم. 'commands'. MT:scripts.bak/save_set
$ backup/rewind/log/verify/ignore=label/list=tcl2 -
$داركم. 'version.68k'. MT:68k_sources.bak/save_set
$ backup/rewind/log/verify/ignore=label/list=tcl3 -
$داركم. 'version.common'. MT:cmn_sources.bak/save_set
$ backup/rewind/log/verify/ignore=label/list=tcl4 -
$داركم. 'version.vms'. MT:vms_sources.bak/save_set
$ Prepare the tape listing
$ delete [dark cm.doc.wdig]contents_of_'version';
$ append/new_version tcl1.lis,tcl2.lis,tcl3.lis,tcl4.lis -
$داركم. 'version'
$ delete tcl1.lis;,tcl2.lis;,tcl3.lis;,tcl4.lis;
$ Cleanup...
$ dismount/nounl mt:
$ deallocate mt:
$ exit
$!
$ error_exit:
$!
$ dismount/nounl mt:
$ deallocate mt:
$!
$ write sys$output ""
$ write sys$output ""
$ write sys$output "In case of:
$ write sys$output ""
$ write sys$output " INIT-F-NOPRIV, no privilege for attempted operation"
$ write sys$output ""
$ write sys$output "error, make_tape from account with VOLPRO privs"
$ write sys$output ""
$ write sys$output ""
$!
$ exit
Appendix F: Tape Contents

Listing of save set(s)

Save set: SCRIPTS.BAK
Written by: DARK_CM
UIC: [011015,000000]
Date: 16-OCT-1989 12:05:49.94
Command: BACKUP/REWRITE/LOG/VERIFY/IGNORE=LABEL/LIST=TC1 [DARK_CM.RELEASE.
Operating system: VAX/VMS version V5.1
BACKUP version: V5.0
CPU ID register: 08000000
Node name: _AESTSC::
Written on: _AESTSC$MB0:
Block size: 8192
Group size: 10
Buffer count: 3

[DARK_CM.RELEASE.V3.COMMANDS] BUG.RPT; 1 8 11-OCT-1989 14:07
[DARK_CM.RELEASE.V3.COMMANDS] BUILD.COM; 1 10 13-OCT-1989 08:07

Total of 3 files, 27 blocks
End of save set

Listing of save set(s)

Save set: 68KSOURCES.BAK
Written by: DARK_CM
UIC: [011015,000000]
Date: 16-OCT-1989 12:06:03.19
Command: BACKUP/REWRITE/LOG/VERIFY/IGNORE=LABEL/LIST=TC2 [DARK_CM.RELEASE.
Operating system: VAX/VMS version V5.1
BACKUP version: V5.0
CPU ID register: 08000000
Node name: _AESTSC::
Written on: _AESTSC$MB0:
Block size: 8192
Group size: 10
Buffer count: 3

[DARK_CM.RELEASE.V3.00.68K] BIO_BODY.ADA; 1 121 9-OCT-1989 07:49
[DARK_CM.RELEASE.V3.00.68K] CLOCK.ADA; 1 19 12-SEP-1989 10:25
[DARK_CM.RELEASE.V3.00.68K] C_BODY.ADA; 1 13 25-AUG-1989 09:59
[DARK_CM.RELEASE.V3.00.68K] DARK.COM; 1 13 12-OCT-1989 08:54
[DARK_CM.RELEASE.V3.00.68K] DARK.MMS; 1 81 28-SEP-1989 07:33
[DARK_CM.RELEASE.V3.00.68K] DATAGRAM_GLOBALS.ADA; 1 54 12-SEP-1989 10:48
[DARK_CM.RELEASE.V3.00.68K] DATAGRAM_MANAGEMENT.ADA; 1 32 12-SEP-1989 10:54
[DARK_CM.RELEASE.V3.00.68K] DMN_BODY.ADA; 1 41 25-MAY-1989 16:59
[DARK_CM.RELEASE.V3.00.68K] DIFFERENCES.RPT; 1 1800 12-OCT-1989 11:02
[DARK_CM.RELEASE.V3.00.68K] GENERIC KERNEL_TIME.ADA; 1 56 27-SEP-1989 08:11
[DARK_CM.RELEASE.V3.00.68K] GETA68.COM; 1 1 13-FEB-1989 10:08
[DARK_CM.RELEASE.V3.00.68K] GKT_BODY_MACHINE_CODE.A68; 1 35 25-SEP-1989 13:40
[DARK_CM.RELEASE.V3.00.68K] GTG_BODY.ADA; 1 23 27-SEP-1989 09:24
[DARK_CM.RELEASE.V3.00.68K] HI_BODY.ADA; 1 1 18-AUG-1988 12:53
[DARK_CM.RELEASE.V3.00.68K] INTERPROCESSOR_INTERRUPTS.ADA; 1 11 12-SEP-1989 11:14
[DARK_CM.RELEASE.V3.00.68K] INTERRUPT_PRIORITY_ROUTINES.A68; 1
Total of 60 files, 3103 blocks
End of save set
### Listing of save set(s)

**Save set:** CMS SOURCES.BAK  
**Written by:** DARK CM  
**Date:** 16-OCT-1989 12:09:08.80  
**Command:** BACKUP/NOREWIND/LOG/VERIFY/IGNORE=LABEL/LIST=TC3 [DARK_CM.RELEASE]  
**Operating system:** VAX/VMS version V5.1  
**CPU ID register:** 08000000  
**Node name:** _AESTSC::_  
**Written on:** _AESTSC$M3:_.  
**Block size:** 8192  
**Group size:** 10  
**Buffer count:** 3

<table>
<thead>
<tr>
<th>File Name</th>
<th>Title</th>
<th>Version</th>
<th>Date Created</th>
<th>Time Created</th>
</tr>
</thead>
<tbody>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]ALARM_MANAGEMENT.ADA;1</td>
<td></td>
<td>8</td>
<td>27-SEP-1989 08:42</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]BUS_IO.ADA;1</td>
<td></td>
<td>26</td>
<td>12-SEP-1989 12:45</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]COMMUNICATIONGLOBALS.ADA;1</td>
<td></td>
<td>7</td>
<td>27-SEP-1989 08:43</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]CONTEXT_SAVE_AREA.ADA;1</td>
<td></td>
<td>11</td>
<td>27-SEP-1989 08:44</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]CONTEXT_SWITCHER.ADA;1</td>
<td></td>
<td>24</td>
<td>9-OCT-1989 08:15</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]CONTEXT_SWITCHERGLOBALS.ADA;1</td>
<td></td>
<td>17</td>
<td>12-SEP-1989 10:36</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]CSA_BODY.ADA;1</td>
<td></td>
<td>8</td>
<td>12-SEP-1989 12:47</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]CSA_DEBUG.ADA;1</td>
<td></td>
<td>1</td>
<td>18-AUG-1988 12:53</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]CSA_DEBUG BODY.ADA;1</td>
<td></td>
<td>6</td>
<td>27-SEP-1989 09:24</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]CSC_BODY.ADA;1</td>
<td></td>
<td>10</td>
<td>27-SEP-1989 09:24</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]CS_BODY.ADA;1</td>
<td></td>
<td>1</td>
<td>16-AUG-1988 16:59</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]DCG_BODY.ADA;1</td>
<td></td>
<td>5</td>
<td>12-SEP-1989 10:38</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]DARK TEXT_IO.ADA;1</td>
<td></td>
<td>11</td>
<td>12-SEP-1989 13:03</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]DEBUG.ADA;1</td>
<td></td>
<td>9</td>
<td>11-APR-1989 15:24</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]DDG_BODY.ADA;1</td>
<td></td>
<td>3</td>
<td>27-SEP-1989 09:24</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]DDG_DEBUG.ADA;1</td>
<td></td>
<td>8</td>
<td>27-SEP-1989 09:24</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]DDG_DEBUG BODY.ADA;1</td>
<td></td>
<td>18</td>
<td>27-SEP-1989 09:24</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]DTIO_BODY.ADA;1</td>
<td></td>
<td>5</td>
<td>9-OCT-1989 07:34</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]ER_BODY.ADA;1</td>
<td></td>
<td>12</td>
<td>7-DEC-1988 10:09</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]GCC_BODY.ADA;1</td>
<td></td>
<td>1</td>
<td>17-AUG-1988 06:58</td>
<td></td>
</tr>
<tr>
<td>[DARK_CM.RELEASE.V3.00.COMMON]GCM_BODY.ADA;1</td>
<td></td>
<td>261</td>
<td>25-AUG-1989 08:53</td>
<td></td>
</tr>
</tbody>
</table>

---

**CMU/SEI-89-TR-20**
Total of 109 files, 2364 blocks
End of save set

Listing of save set(s)

Save set: VMS_SOURCES.BAK
Written by: DARK_CM
UIC: [011015,000000]
Date: 16-OCT-1989 12:11:42.71
Command: BACKUP/NOREWRITE/LOG/VERIFY/IGNORE=LABEL/LIST=TC4 (DARK_CM.RELEASE
Operating system: VAX/VMS version V5.1
BACKUP version: V5.0
CPU ID register: 08000000
Node name: _AESTSC::
Total of 37 files, 3040 locks
End of save set
This document characterizes a specific version of the Distributed Ada Real-Time Kernel (DARK) software artifact and supplies documentation for its installation and use. This document is geared toward: the engineer responsible for installing the Kernel, engineers responsible for prototyping and maintaining the Kernel, and engineers using the Kernel and needing an awareness of changes from the previous release.