CRISP Automation Systems Documentation
Manual Number: MAN-CSP-MIG-031
Subject: CRISP 3.1 Migration Guide
Date Created: 05/22/2002 Revision 1.4 Date Revised: 03/20/2003

Effective Date: **03/20/2003**

Supercedes: **MAN-CSP-MIG-031 v 1.3**

Revision History

Revision	Date Issued	Description	Author	Approved
1.0	05/23/2002	Original Issue	S. Quayle	K. Wild
1.1	07/21/2002	Changes from V3.1-16	S. Quayle	K. Wild
1.2	08/12/2002	Dual Alphas	S. Quayle	K. Wild
1.3	11/29/2002	Fix example	S. Quayle	K. Wild
1.4	03/20/2003	Fix web links	S. Quayle	K. Wild

Reference Documents:

DES-DEV-002 Version 1.0 Software Design
DES-DEV-004 Version 1.0 Coding Standards, Comments and Naming Conventions

ALPHA-Specific Reference Documents:

Document AA-QSBKB-TE: Migrating an Application from OpenVMS VAX to OpenVMS ALPHA. This document can be ordered at:
http://h71000.www7.hp.com/doc/72final/6459/6459PRO.HTML

OpenVMS Migration Software for VAX to Alpha Systems (formerly known as

DECmigrate) translates VAX executables to run on Alpha. It can be downloaded from the web at:
http://h71000.www7.hp.com/openvms/products/omsva/omsva_012_translating_images.html

CRISP Automation Systems Documentation
Manual Number: MAN-CSP-MIG-031
Subject: CRISP 3.1 Migration Guide
Date Created: 05/22/2002 Revision 1.4 Date Revised: 03/20/2003

Introduction

Beginning with V3.1, CRISP/32 works on both VAX and Alpha processors. The Alpha version will work the same as the VAX version of CRISP, with exceptions as detailed in this document.

This document expands on the release notes that come with CRISP/32. You must be familiar with the release notes before using this guide to fine-tune your system.

Version Plans

The current plans for future versions of CRISP are as follows:

V3.0 VAX-only version of CRISP/32.

V3.1 The distribution kit includes both VAX CRISP and Alpha CRISP, with identical features. A new feature of CRISP/32 is that messages sent to CRISP$TT can be logged to disk by the new CRISP Log Server (LOGSRV).

V3.x [As necessary] Versions of both Alpha and VAX CRISP, with identical features. Itanium support may appear in a later member of this series.

V4.0 Alpha and Itanium-only version of CRISP/64. The VAX code on the distribution kit will be the final 3.x version. Features may no longer be identical between VAX and Alpha versions.

Operating System

- CRISP on Alpha requires OpenVMS 7.2 or later. OpenVMS 7.2-2 or later is recommended because of a change to the MACRO compiler. For a OpenVMS 7.2-1 system on Alpha, ECO VMS721_UPDATE-V0100 is required to solve an issue with RTL installation.

- CRISP on VAX version requires OpenVMS 5.5 or later. OpenVMS 5.5-2 is recommended. OpenVMS 5.5-2 is the earliest version for which Y2k support has been guaranteed.

Configuration Changes

There are several changes to CRISP_CONFIG to support new features.

- Users can now select the maximum number of databases, processes, and logics. Increasing any of these may require AUTOGEN of the system.
- Window Workstation V3.0 has a configuration script. This is automatically called by CRISP_CONFIG.
- If the Central Administration product is installed, its configuration script is called by CRISP_CONFIG.

Logic Building

Every version of CRISP/32 requires that logics be rebuilt. This version is no different.

The Logic Build command (LGBUILD) now produces a database in the first step (the CRISP Compiler step). There is no DBO intermediate file.

The Logic Build command (LGBUILD) will now list all available parameters when the filespec is specified as "?". Parameter P7 values that might be useful to users:

/DEBUG	Link the program as DEBUG.
/LIB	Link the program against object libraries

/LIST	Produce a listing file of the Macro code.
/NOOPT	Do not optimize the Macro code (non-VAX only)
/OPT=xx	Optimize the Macro code (non-VAX only)
/SAVE	Use database values saved from CRISP/32 upgrade

The "/DEBUG" option requires experience in debugging detached processes. Logical name CRISP$LOGICS_EHD_DISABLE should be defined before starting the logic (see the "New Logical Names" section).

The "/LIB" option is only usable when the CRISP DEBUG product kit is installed. This product requires a license for the CRISP/32 source code.

Database structures have changed slightly. Databases compiled under older versions of CRISP/32 can still be installed. However, the logics associated with those databases cannot be run.

The "/SAVE" option allows the values in a live database to be imported into the new database structure. At CRISP/32 install time, the values from user databases in CRISP$DB can exported to special files. When the databases are later built, those values can then be imported into the new database.

During a logic build on Alpha, logical name DECC$FILE_SHARING is set to "DISABLE" temporarily. This greatly improves build performance when the system's logical name value is set to "ENABLE".

New Logical Names

When logics are started, the logic is granted a page file quota of PQL_DPGFLQUOTA pages. This may be insufficient, especially for Alpha. In that case, logical name CRISP$LOGIC_PGFLQUOTA can be defined. If it is defined as a number when a logic is started, that value is used as the page file quota for the process.

Logical names may be specified to change the behavior of network adapters.

See topic “FDDI” later in this document.

If logical name CRISP$LOGICS_EHD_DISABLE is defined when a logic is started, the logic does not use its error handler. This is useful when debugging a logic process. Warning: defining this logical name in a production system is not recommended, since disabling the error handler will cause a logic to exit abruptly if an arithmetic exception occurs.

Sometimes, logic call WWS_SETDSP does not drive up the selected display. If logical name CRISP$WWS_SETDSP_RETRY is set to a number greater than zero, the value is treated a number of seconds. The drive-up is automatically retried that many seconds later, over and over, until the correct display is shown.

The CIA_MAINT_WINDOW program, which edits CRISP Security attributes using DECwindows, checks logical name CRISP$CIA_AUTOLOGOUT_MAINT after login. If there is a period of inactivity of that number of minutes, the program will exit. On CRISP/32 systems with WWS V3.0 installed, this logical name is set as a part of the configuration script.

The HISTORIAN_CSV program checks logical names HISTORIAN_CSV_ROOT and HISTORIAN_CSV_LAST. See the “New Component – HISTORIAN_CSV” for additional information.

On Alpha, having logical name DECC$FILE_SHARING defined as “ENABLE” may greatly reduce performance of CRISP/32 and its layered products. Consult the release notes for OpenVMS and Compaq C to determine whether or not this logical name should be set.

New Component – LOGSRV

New for CRISP V3.1 is the CRISP Log Server (LOGSRV). The Log Server is available on both VAX and Alpha.

The primary capability of the Log Server is to capture messages going to CRISP$TT into a disk file. The file name can include the current year, month, day, hour, and minute. When the time changes, the Log Server will close the current file and open a new one with the new name. A new disk file is also opened upon each CRISP start.

The Log Server can also send its messages to other Log Servers via IP. UDP is used to reduce overhead. A Log Server can send its message to a multicast address, which allows several Log Server processes to receive any messages.

The Log Server is also available for standalone operation. The product is known as the "Log Server kit". This allows an administrative computer without CRISP to monitor messages from CRISP systems.

The Log Server will be started or not depending on the user's answers during CRISP_CONFIG. After CRSTOP, the Log Server will remain running for a period of time to capture all other process exit messages. If a CRSTART is attempted before the Log Server exists, a hot start will be performed, and logging will continue.

If customer-written code needs to be captured by the Log Server, it should send output to CRISP$TT_SRV, instead of CRISP$TT. If the Log Server is started with CRISP, CRISP$TT_SRV points into its input mailbox. If the Log Server is not running, CRISP$TT_SRV points directly to CRISP$TT.

New Component – CRISP_MONITOR

A new capability can be configured into CRISP/32. The CRISP_MONITOR job checks the status of disk drives, and can send email to a system administrator when drives exit a shadow set, or when the disks reach an error level.

New Component – HISTORIAN_CSV

A new program, CRISP$UTL:HISTORIAN_CSV, has been added to CRISP/32. This program allows retrieval of one or more Historian point files. The data values are merged by time, and output as comma-separated values that can be directly imported into spreadsheet software.

HISTORIAN_CSV checks logical names as follows:

HISTORIAN_CSV_ROOT The CRISP$HIST_ROOT value to be used on the target system.

HISTORIAN_CSV_LAST The VMS absolute time at which retrieval is to end.

New Capability – Database Hot Swap

User databases can now be deleted. To use this capability, the “CRISP-SWAP” license key must be installed on the system before CRISP is started. The command is “DBINSTALL DELETE <database-name>”. This capability is available on both VAX and Alpha.

When a database is deleted, but a program does not release the database, the program continues to operate, but will never “see” the new database. Other programs which released the old database, or start after the new database is installed, use the new database.

Therefore, there are some important cautions when using this capability:

1. Any logics which use the database should be shut down first (LGINS STOP xxx). If this is not done, CLE will kill each logic. In dual systems, this could cause a switchover.

2. If you use database transfers (DBTCMD), deleting any database will cause the transfer list to re-resolve. Resolve errors will be signaled to CRISP$TT if a needed database no longer exists. After replacing any missing databases, run the DBTCMD command to re-establish those transfers.
3. The Database Access Server (DBASRV) detects when databases have been deleted. Products that use the DBASRV (CWS, PCWS, WWS, CrispView, CWS Trend, remote IDC) will be forced to re-resolve. Because a new database can't be installed instantly, there will be a period of time when the re-resolve fails.
4. Some CRISP products (IDI, IDC, Historian, Chart/II, Basic Workstation) do not respond to a database being deleted. These pieces should be shut down before deleting the database, if they access the database in any way.
5. Customer-written programs need to test from time to time to see if a database is deleted. The 7th reserved LOGICAL (of the 16 reserved LOGICAL's at the start of each database) changing to the true state. When the database is "marked for delete", the customer-written application should call function DBA$RELEASE on the database descriptor. Alternatively, the program could decide to exit.

   A convenience function, DBA$TEST_DELETE_DB, has been provided in the CRISP Software Bus RTL (CRISPSWBRTL.EXE). The function accepts a database descriptor pointer (usually seen as "dbdx" or "dbdx_ptr" in code). It returns 0 if the database has not been marked for delete. If the database has been marked for delete, the function returns a non-zero value.

   The CRISP system databases (DDMON output shows "Sys") cannot be deleted, and so need not be checked by the user's program.

New Product – Central Administration

The CRISP security file, CRISP$CFG:CRISP_UAF.IDX, can now be served from

a central machine to a set of satellite machines. Only the Window Workstation product uses CRISP security at present. The Central Administration product kit integrates fully with CRISP/32.

Data Integrity Issues

The use of multi-processor systems had never been supported for CRISP/32, because of the possibility of two different processes modifying the same element in a database at the same time.

There is a similar problem on Alpha. If two data items are located in the same 8-byte quadword, two processes modifying those data items, even if they are unrelated, can cause loss of data.

There are two ways to handle this issue: Database Locking and Data Isolation.

Database Locking

When a logic runs, it first locks its database for exclusive access. When the logic completes, it unlocks the database. Two processes in CRISP V3.0 do not respect these locks: DBASRV (Database Access Server) and IDC. Changes through these mechanisms can corrupt data values in the database. Most CRISP layered products, including IDI, ignore locking.

In CRISP V3.1, code was added to DBASRV to lock the database before writing. Locking will be disabled by default for single-processor VAX systems. Locking will be enabled by default for Alpha and multi-processor VAX systems.

If database locking is used, it is essential that **all** programs, including customer-written programs, lock the database before writing or modifying any data values.

Note: If IDC or DBT are used to transfer data between databases, logic cycle times and priorities must be examined carefully. A possible race condition

could be established where a low-priority logic, which has its database locked, prevents higher priority processes from executing while waiting for the lock. Customer-written programs can create a deadlock condition if two databases are locked simultaneously.

At CRISP_CONFIG time, the user is offered an opportunity to force IDC to do its local updates at the end of each logic. This option is recommended when database locking is used. However, IDC does not actually lock the database, and so the database could potentially be corrupted by large IDC transfers or slow-running logics. DBT updates are recommended for database transfers within one system.

Data Isolation

A different approach is to isolate parts of a CRISP database to prevent a problem from occurring. Operator inputs would be in one section, data modified by field I/O in another, and the data used by the logic would be in a third.

Warning: Successful data isolation is completely dependent on the customer's database structure. If an "idiot-resistant" solution is desired, use database locking.

The following CRISP logic is an example of data isolation:

```
NUMERIC;   OP_INPUTS(5)
;
FLOAT;     PLC_INPUTS(16)
NUMERIC;   PLC_BITS(16)
;
NUMERIC;   LOGIC_INPUTS(10)
FLOAT;     LOGIC_VALUES(20)
NUMERIC;   LOGIC_OUTPUTS(10)
```

This looks partitioned. However, when the CRISP Compiler processes these declarations, it builds collects each data type into a database section. The FLOATs are all collected together, in the order declared, as are the NUMERICs. Here's what part of the database looks like in memory:

Memory Offset	+2	+0
0	OP_INPUTS(1)	OP_INPUTS(0)
4	OP_INPUTS(3)	OP_INPUTS(2)
8	PLC_BITS(0)	OP_INPUTS(4)
12	PLC_BITS(2)	PLC_BITS(1)
16	PLC_BITS(4)	PLC_BITS(3)

On Alpha, "PLC_BITS(0)" and "OP_INPUTS(4)" are in the same longword. If they are modified by two separate processes at the same time, one of the updates will be lost.

To achieve data isolation, separate the different parts of the database with some "dummy" variables. A minimum of 4 is recommended, although 1 is usually the minimum necessary. For example:

```
NUMERIC;   OP_INPUTS(5)
NUMERIC;   OP_BIT_DUMMY(4)      ! Data Isolation
;
FLOAT;     PLC_INPUTS(16)
```

```
NUMERIC;   PLC_BITS(16)
FLOAT;     PLC_FLOAT_DUMMY(4)   ! Data Isolation
NUMERIC;   PLC_BIT_DUMMY(4)          ! Data Isolation
;
NUMERIC;   LOGIC_INPUTS(10)
FLOAT;     LOGIC_VALUES(20)
NUMERIC;   LOGIC_OUTPUTS(10)
```

Now, the database looks like:

Memory Offset	+2	+0
0	OP_INPUTS(1)	OP_INPUTS(0)
4	OP_INPUTS(3)	OP_INPUTS(2)
8	OP_BIT_DUMMY(0)	OP_INPUTS(4)
12	OP_BIT_DUMMY(2)	OP_BIT_DUMMY(1)
16	PLC_BITS(0)	OP_BIT_DUMMY(3)

This forces the “Operator” and “PLC” values into different longwords.

Warning: If multiple IDI processes are running, isolate the data for each IDI instance separately.

Database Locking

Customer-written programs must call function DBA$GET_CRISP_LOCKING. This returns a non-zero value if database locking was enabled when CRISP was configured.

The following is documentation for functions which implement database locking. They were always in CRISP/32, but have not been documented until now.

DBA$GET_CRISP_LOCKING

Synopsis:

Get the CPU's locking state from the CRISP system database.

Format:

DBA$GET_CRISP_LOCKING (condx)

Arguments:

condx

Usage: Connect descriptor address
Type: Unsigned longword
Access: Read only
Mechanism: By value

An unsigned longword containing the address of the caller's connect descriptor. This is the address returned by the Software Bus connect function, SWB$CONNECT.

Description:

This functions determines whether or not locking is necessary by checking Logical SYSTAT_B_LOCKING in the CRISP database.

Returns:

1 = locking is required
0 = locking is not required

DBA$CVT_DB_LOCK

Synopsis:

Convert a database lock to a different mode.

Format:

DBA$CVT_DB_LOCK (dbdx, mode)

Arguments:

dbdx

Usage: Database descriptor address
Type: Unsigned longword
Access: Read only
Mechanism: By value

An unsigned longword containing the address of the caller's database descriptor. This is the address returned by the CRISP database locate functions, DBA$LOCATE and DBA$LOCATE_AND_RESOLVE_SYMBOL.

mode

Usage: New lock mode
Type: Unsigned Longword
Access: Read only
Mechanism: By value

The lock mode to which the specified database resource should be converted. This should be one of the symbolic names LCK$K_xxMODE defined in the LCKDEF (or $LCKDEF) include module.

Description:

This procedure attempts to synchronously convert the lock on the database resource from its current mode to the specified mode.

Two lock modes are used by CRISP: LCK$K_PWMODE Protected write
LCK$K_NLMODE Null mode

Returns:

Usage: Condition Code
Type: Unsigned Longword
Access: Write only
Mechanism: By value

Condition Values Returned:

Any condition value returned from SYS$ENQW

Success:

SS$_NORMAL = Normal successful completion

DBA$GET_DB_RESOURCE_NAME

Synopsis:

Returns the resource name for a CRISP database.

Format:

DBA$GET_DB_RESOURCE_NAME (dbdx, rname, [rname_len])

Arguments:

dbdx
Usage: Database descriptor address
Type: Unsigned longword
Access: Read only
Mechanism: By value

An unsigned longword containing the address of the caller's database descriptor. This is the address returned by the CRISP database locate functions, DBA$LOCATE and DBA$LOCATE_AND_RESOLVE_SYMBOL.

rname
Usage: Database resource name
Type: Character string
Access: Write only
Mechanism: By descriptor -- fixed-length string descriptor

The address of the (standard VMS) descriptor pointing to the character string that will receive the database resource name. This descriptor should point to a 31-character (or larger) buffer.

[rname_len]
Usage: Resource name length

Type: Unsigned Word
Access: Write only
Mechanism: By reference

The address of a word to receive the length of the string returned in RNAME. This argument is optional.

Description:

This procedure returns the resource name to be used with the VMS lock management facility for gaining exclusive access to the CRISP real-time database specified by DBDX. For further information, see the VMS $ENQ system service.

Returns:

Usage: Condition Code
Type: Unsigned Longword
Access: Write only
Mechanism: By value

Condition Values Returned:

Any condition value returned from SYS$FAO
Any condition value returned from CSP$RESOURCE_NAME

Success:

SS$_NORMAL = Normal successful completion

Example:

```
void *dbdx_ptr;
```

```
char resrc_name[32];
$DESCRIPTOR (resrc_name_dx, resrc_name);


status = dba$get_db_resource_name (dbdx_ptr, &resrc_name_dx,
                                   &resrc_name_dx.dsc$w_length);
```

ALPHA-SPECIFIC ISSUES

References

Information on migrating applications from VAX to Alpha is in the Compaq document “Migrating an Application from OpenVMS VAX to OpenVMS Alpha”, order number AA-QSBKB-TE. This is available on the Web at:

http://h71000.www7.hp.com/doc/72final/6459/6459PRO.HTML

All CRISP/32 components and most layered products are native Alpha code.

Requirements and Limits

There are many changes to SYSGEN parameters necessary in the move to Alpha VMS. This document assumes that you are familiar with those changes.

Compiling large logics may require very large page file quotas during the macro compiler step. Some compiles have taken a peak of 1,086,496 pages.

On Alpha, CRISP_SETUP may generate informational messages when there is insufficient granularity hint space. These can be safely ignored. The MODPARAMS.DAT provided by the CRISP installation includes parameters to increase this space. For more information, see the OpenVMS 7.1 release notes, section 4.20.1.3.

Username “CRISP” may require larger quotas, especially page file quota. These limits are set in the AUTHORIZE utility.

Obsolete or Unavailable Components

The following elements of VAX CRISP are unavailable on Alpha. These all require Q-Bus support, which is available only on some VAX systems. PCWS and remote CWS will continue to operate normally.

Basic CRT
Locally-connected CWS
Arbiter-based I/Onyx
Arbiter-based active/standby switching
CRISPconnect Server for NetDDE
CRISPconnect Server for @aGlance/IT

The CRISP compiler on Alpha does not support translation of version 2.0 logic source files. Translate the logic on a VAX system, and then use the translated code.

Layered Products

The following layered products are available for CRISP on Alpha:

IDI
WORF

The following layered products are currently unavailable. Support for these is planned.

Historian Viewer
Window WorkStation (WWS)

The following layered products are unavailable. These might be supported if customer interest is sufficient.

Chart/II
Reporter

The following layered products are not available, and will not be available in CRISP on Alpha.

CRISP-20/20 Interface
Datagate
VMS/HP 1000 Interface
SFW390 Interface
SPC

Customer-Written Programs

Customer-written programs are an important feature of CRISP. However, the following are not supported:

- Programs ported with OpenVMS Migration Software for VAX to Alpha Systems (formerly known as DECmigrate).
- Tightly coupled database access, which has been a deprecated feature of CRISP/32 for quite some time.
- Beginning with version 4.0, customer-written applications that depend on a NUMERIC or LOGICAL fitting into 2 bytes will have to be modified. Beginning with that version, NUMERIC and LOGICAL variables will be 4 bytes. Each component of a TIMER or COUNTER will be 4 bytes. Any NUMERIC declarations will be treated as LONGWORD declarations.

OpenVMS Migration Software for VAX to Alpha Systems (DECmigrate) Issues

Although use of translated images is not supported, here are some thoughts:

- The run-time libraries supplied with CRISP/32 on Alpha are not built with the /TIE option, and so are not directly callable from migrated VAX programs.
- OpenVMS Migration Software for VAX to Alpha Systems could be used to

migrate the VAX run-time libraries, but this has not been tested. However, if migrated CRISP RTL's (CRISP*RTL_TV.EXE) are present in SYS$LIBRARY, they will be installed automatically by CRISP_SETUP.

• Internal data structures have mostly been longword-aligned on Alpha, including Software Bus communications. Therefore, migrated VAX executables and Alpha executables may not interoperate.

• Migration of programs linked against object libraries has been successful in limited tests. The object libraries are not distributed with CRISP/32, but can be made available to source code licensees.

Language: C

Much of the CRISP code was written in VAX C. The only Alpha C compiler is Compaq C. Compaq C has a VAX C emulation mode.

To create code that is compatible with CRISP V 3.1, specify the following Compaq C compiler flags:

• /nomember_align Disable padding in structures
• /extern=common Use VAX C-compatible external variables
• /gran=longword Use 4-byte access instead of 8-byte accesses

Programs originally written in VAX C will also have to use the compiler flag "/standard=VAXC".

The ***Compaq C User's Guide*** covers many of the important changes from VAX C to Compaq C. However, the following inconsistencies are not mentioned in the manual:

• Under VAX C, the value of global variable **vaxc$errno** always held a VMS return code when a C library function was called. In Compaq C, vaxc$errno is defined by header file <errno.h> as a macro. vaxc$errno has a VMS status code only when the value of **errno** indicates that the error is not translatable to a Unix-style status value.

• Macros "__DECC" and "__VAXC" can be used to conditionalize code.

Warning: the Compaq C compiler sets the "__VAXC" macro when running in VAX C emulation mode.

• Under VAX C, you could open a file for write, rewind it, and then read the contents. This fails under Compaq C.

• Assume the following code fragment:

```
int function2 (DSC$DESCRIPTOR *dx_ptr);  /* prototype */

int function1 (void)
{
     DSC$DESCRIPTOR *p_ptr;
     int retval;

     retval = function2 (&p_ptr);
}
```

In VAX C, the compiler would determine that "p_ptr" is already a pointer to the type needed by "function2", and would ignore the "&" operator. Under Compaq C, the compiler passes the address of "p_ptr".

• Assume the following code fragment:

```
struct dev_struct
        {
        STRING  text[ENTRY_CHAR_MAX];
        UNS32   value;
        SDI_R   (*init)();
        SDI_R   (*read)();
        SDI_R   (*write)();
        SDI_R   (*exit)();
        } ;

struct dev_struct dt00_entry =
{"AAA1",        DT_AAA1,
tidi$aaa1_init,
tidi$aaa1_read,
tidi$aaa1_write,
```

```
tidi$aaa1_exit
};

struct dev_struct dt01_entry =
{"AAA2",        DT_AAA2,
tidi$aaa2_init,
tidi$aaa2_read,
tidi$aaa2_write,
tidi$aaa2_exit
};
```

In VAX C, the compiler would allocate “dt01_entry” adjacent to “dt00_entry” in memory. Compaq C does not.

Language: C++

All header files that ship with CRISP are C++ compatible. The only issue is WORF_DEF_USER_C.H, which ships with the WORF development product.

C++ does not support the VAX C keyword “variant_union”, and so it is replaced by “union” in header file WORF_DEF_USER_C.H. This affects the definitions for typedef SYMBOL_RECORD.

In the real-time sample program, WORF_EXAMPLE_C.C, the following code appears:

```
SYMBOL_RECORD rec;
rec.rt.record_number = 0;
rec.rt.transfer_count = 1;
INIT_DX_PTR (&rec.rt.buffer_dx, (char *)&buff1, sizeof
(buff1));
```

For C++, this must be coded as:

```
SYMBOL_RECORD rec;
rec.t.rt.record_number = 0;
rec.t.rt.transfer_count = 1;
INIT_DX_PTR (&rec.rt.buffer_dx, (char *)&buff1, sizeof
(buff1));
```

".t" is the intervening level in the union, which is hidden from C users.

Language: Macro

Compaq C (both VAX and Alpha) creates all its PSECTs with "NOPIC". VAX C creates all its PSECTS with "PIC". Programs that share code between C and Macro will have to be modified. The header files that ship with CRISP and WORF conditionalize code as follows:

```
.IF DF  IS_ALPHA
      Alpha code goes here
.ENDC
.IF NDF IS_ALPHA
      VAX code goes here
.ENDC
```

For Alpha, create a file called IS_ALPHA.MAR, with the following line:

```
IS_ALPHA = 1
```

Then compile as follows:

```
$ MACRO /MIGRATION/OBJ=module IS_ALPHA.MAR+module.MAR
```

where "module.MAR" is the module to be compiled.

Language: Fortran

To create code that is compatible with CRISP V 3.1 and VAX Fortran, specify the following Fortran 77 compiler flags:

- /old_f77 Use Fortran-77 compiler
- /noalign Disable padding in structures
- /gran=longword Use 4-byte access instead of 8-byte accesses

Because Fortran has no conditional compilation, any padding in files distributed with CRISP are marked as comment lines which begin with "`C**PAD**`". For Alpha, these will have to be manually removed before use.

DBA$UPDATE

DBA$UPDATE automatically forces synchronization to the end of logic when database locking is enabled. No coding changes will be needed for customer-written programs that call DBA$UPDATE. This behavior is exactly the same as if the calling program specified the DBA$M_SYNCH flag in all calls to DBA$UPDATE.

Floating-Point Errors

On Alpha, floating-point errors cause a delayed HPARITH trap. This error can be reported hundreds of Alpha instructions after the error happened. Customer-written programs will have to deal with this as would any other Alpha program.

FDDI

FDDI interfaces work differently on Alpha. If DECnet is using the interface, CRISP processes cannot use the DECnet address (AA-00-04-xx-xx-xx). This is a problem for the server processes (DBASRV, CASRV); I/Onyx and CC Server are unaffected.

Therefore, two optional logical names may be specified before starting

CRISP/32: CRISP$NETxx_AREA and CRISP$NETxx_NODE, where "xx" is the digit(s) of the corresponding CRISP$NETxx logical name.

If one or both of these logical names are specified, the network channels are opened with the AA-00-04 address that corresponds to the specified area and node. If only one is defined, the other defaults to the system's area or node.

If neither is specified, the channels are opened with the hardware MAC address (FDDI) or the DECnet address (non-FDDI).

Probably the easiest place to set these logicals is in a command procedure. At CRISP_SETUP time, directory CRISP$CFG is searched for command procedures that start with “USER_SETUP”. Any procedures found are executed.

CRISP Automation Systems Documentation
Manual Number: MAN-CSP-MIG-031
Subject: CRISP 3.1 Migration Guide
Date Created: 05/22/2002 Revision 1.4 Date Revised: 03/20/2003

Attachments:

Document	Revision	Dated