

# CRISP V3.1 MIGRATION GUIDE

Includes Guidance on Migrating from VAX to Alpha

Last revised: 26-Apr-2001

## 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).

V3.2 Database Hot Swap capability will be added to CRISP/32.

V3.x [As necessary] Versions of both Alpha and VAX CRISP, with identical features.

V4.0 Alpha-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

Alpha CRISP requires VMS 7.2-1 or later. The following ECO's from Compaq are required for a VMS 7.2-1 system:

- [Alpha] VMS721\_UPDATE-V0100 is required to solve an issue with RTL installation.
- [Alpha] An ECO (number pending) for the Macro Compiler. Compiling some CRISP programs cause the Alpha Macro compiler to enter an infinite loop.

The VAX version of CRISP requires VMS 5.5-2 or later.

## Configuration Changes

There are several changes to CRISP\_CONFIG to support new features.

One change is that users can now select the maximum number of databases, processes, and logics. Increasing any of these may require AUTOGEN of the system.

## Logic Building

Every version of CRISP/32 requires that logics be rebuilt. This version is no different. However, all the logic functionality has been moved into the Logic RTL (CRISPLOGICRTL). Future versions of CRISP/32 will not require that the user rebuild logics.

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.
LIST	Produce a listing file of the Macro code.
NOOPT	Do not optimize the macro code (Alpha only)

The DEBUG option requires experience in debugging detached processes, and so is not for use by novices.

## New Logical Name

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.

## 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 set up for standalone operation. This allows an administrative computer without CRISP to monitor messages from CRISP systems.

On Alpha, the "secondary output device", CRISP\$EHDOUT, does not exist. CRISP\$EHDOUT is still supported on VAX. However, its use should be unnecessary if the Log Server is used.

The Log Server will be started or not depending on the user's answers during CRISP\_CONFIG. At 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.

## Data Integrity Issues

The use of multi-processor systems had never been supported for CRISP, 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 4-byte longword, two processes modifying those data items, even if they seem 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, currently ignore locking.

In CRISP V3.1, code was added to DBASRV (and possibly IDC) 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 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 could even 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:

```
LOGICAL;      OP_INPUTS(5)
;
FLOAT;       PLC_INPUTS(16)
LOGICAL;     PLC_BITS(16)
```

```

;
LOGICAL;    LOGIC_INPUTS(10)
FLOAT;     LOGIC_VALUES(20)
LOGICAL;    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 LOGICALs. 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:

```

LOGICAL;    OP_INPUTS(5)
LOGICAL;    OP_BIT_DUMMY(4)           ! Data Isolation
;
FLOAT;     PLC_INPUTS(16)
LOGICAL;    PLC_BITS(16)
FLOAT;     PLC_FLOAT_DUMMY(4)        ! Data Isolation
LOGICAL;    PLC_BIT_DUMMY(4)         ! Data Isolation
;
LOGICAL;    LOGIC_INPUTS(10)
FLOAT;     LOGIC_VALUES(20)
LOGICAL;    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 function 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



# 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://www.openvms.compaq.com:8000/72final/6459/6459PRO.HTML>

DECmigrate, the VAX to Alpha migration tool, is not required. All CRISP components and layered products are native Alpha code. DECmigrate can be downloaded from the web at:

<http://www.support.compaq.com/amt/decmigrate/index.html>

## 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. Test 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 VMS 7.1 release notes, section 4.20.1.3.

## Obsolete or Unavailable Components

The following elements of VAX CRISP are unavailable on Alpha. These all require Q-Bus support, which is not available on Alpha. PCWS and remote CWS will continue to operate normally.

- Basic CRT
- Locally-connected CWS
- Arbiter-based I/Onyx
- Arbiter-based active/standby switching

The following elements of VAX CRISP are unavailable on Alpha. These might be supported if customer interest is sufficient.

- 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.

Active/standby pairs of systems are not currently supported on Alpha. The code is present, but has not been tested.

## Layered Products

The following layered products are being adapted for Alpha CRISP:

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 Alpha CRISP.

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 DECmigrate.
- Tightly coupled database access, which has been a deprecated feature of VAX CRISP 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.

### **DECmigrate Issues**

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

- The run-time libraries supplied with Alpha CRISP are not built with the /TIE option, and so are not directly callable from migrated VAX programs.
- DECmigrate could be used to migrate the VAX run-time libraries, but this has not been tested. However, if migrated 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 DEC C. DEC C has a VAX C emulation mode.

To create code that is compatible with CRISP V 3.1, specify the following DEC 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 DEC C.

However, the following inconsistencies are not mentioned in the manual:

- Under VAX C, the value of global variable `vaxc$errno` always held a VMS return code when a C library function was called. In DEC C, `vaxc$errno` is defined by header file `<errno.h>` as a macro. `vaxc$errno` has a VMS status code only when the value `errno` indicates that the error is not translatable to a Unix-style status value.
- Macros `"__DEC_C"` and `"__VAX_C"` can be used to conditionalize code. The DEC C compiler sets the `"__VAX_C"` 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 DEC 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 DEC 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. DEC 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 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**

DEC 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 `*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).