




Test	VAX, database locking disabled	Alpha, database locking enabled	Alpha, database locking disabled	VAX, database locking enabled

**Process Monitor (command PDMON)**

**Test items:**

- While CRISP is running, "PDMON" should show several processes. Cross-check the PID field against the output of "SHOW SYS" to verify that all those processes actually exist. Core CRISP/32 processes name a VMS process name that begins with "CRISP\$" (for example, DBASRV is process CRISP\$DBASRV). The core CRISP/32 processes are: DBCTRL, DBASRV, MON, NETMON, PRINT, FILES, CASRV, and CLE.
- Changing the configuration with CRISP\_CONFIG will change the processes started. Verify that this happens appropriately.
- Each logic started with LGCONFIG will be a process. Verify that these process exist using "SHOW SYS". The process name will be the name of the logic.
- After CRSTOP, there should be no processes. The Log Server, Historian, and IDI remain connected to the Software Bus. Verify that these processes are still connected after CRISP stops, and that they eventually exit.
- Select a process (for example, CASRV). Do a "CRSTOP CASRV". The process should disappear from the PDMON list.
- Select a process. Do a "STOP/ID=xxx", where "xxx" is the process ID. The process should still appear in the PDMON list, but will not appear when "SHOW SYS" is done.
- With the "ghost" entry from the above test, do a "CLN yyy" where "yyy" is the slot number of the process that no longer exists. Use PDMON to verify that the process no longer shows up.

**Logic Monitor (command LDMON)**

**Test items:**

- While CRISP is running, "LDMON" should show all logics configured with LGCONFIG. Cross-check the list of logics against the output of "SHOW SYS" to verify that all those processes actually exist.
- Changing the configuration with LGCONFIG will change the processes started. Verify that this happens appropriately. (Requires CRSTOP followed by CRSTART.)
- If LGCONFIG is used to eliminate all logics, verify that no logics are started after CRSTART.
- After CRSTOP, there should be no logics running.
- Select a logic. Do a "CRSTOP xxx", where "xxx" is the name of the logic. The logic should disappear from the LDMON list.
- Select a logic. Do a "STOP/ID=xxx", where "xxx" is the process ID (as shown by "SHOW SYS"). The logic should still appear in the LDMON list, but will not appear when "SHOW SYS" is done.
- With the "ghost" entry from the above test, do a "LDCLN yyy" where "yyy" is the slot number of the logic that no longer exists. Use LDMON to verify that the logic no longer shows up.
- Select a logic. Do a "LGINSTALL STOP xxx". Use LDMON and "SHOW SYS" to verify that the logic is gone.

**Database Monitor (command DDMON)**

**Test items:**

- Reboot the system. "DDMON" should display an error message that the Software Bus cannot be located.
- Start CRISP with CRSTART. While CRISP is running, the DDMON command should list the CRISP system databases (CRISP, TSKDIR, DBDIR, CRTDIR [if Basic CRT configured], LOGDIR, NETMON, WSDIR) plus any databases configured by LGCONFIG.
- Stop CRISP with CRSTOP. The DDMON command should list the following CRISP system databases: CRISP, TSKDIR, DBDIR, LOGDIR.

**CRISP Access Server (CASRV process)**

The CASRV performs two functions: fetching Historian data, and retrieving CWS display files.

**Test items:**

- Configure the Historian and capture some data. Use CrispView to retrieve the data. If CrispView is not available, a special test program can be provided to retrieve the data.
- Configure CRISP so that the CRISP\$CWS\_DSP\_FILESPEC logical points to a valid CWS display file (this is in the CASRV configuration section). Use a CWS (VAX only) or PCWS to retrieve displays from the display file.
- The message counts for the CASRV in the NETMON database should be increasing. These can be observed using PCWS and the standard display file.

#### CRISP Database Access Server (DBASRV process)

The DBASRV retrieves and modifies real-time data (that is, data from CRISP databases). When the CWS Trend process is configured, it retrieves trend data.

##### Test items:

- Use an operator workstation (PCWS, CWS [VAX only], Window Workstation, or CrispView) to display real-time data. Use CRISP\$EXE:DB\_SETUP to change the data and verify that the workstation updates appropriately.
- Use an operator workstation (PCWS, CWS [VAX only], Window Workstation, or CrispView) to modify real-time data. Use both jog and modify. Enable logging on the PCWS displays, which will cause log messages to be written to CRISP\$TT. Use CRISP\$EXE:DB\_SETUP to verify that the data has changed.
- Configure the CWS Trend process. Include in the trend file a few permanent trends.
- At CRSTART, verify that CWSTND is connected to the Software Bus using PDMON, and that it is a process shown in SHOW SYS.
- Use the CRISP\$UTL:CWSTA program to verify that the trend global section exists. Verify that the permanent trends have data in them.
- Use an operator workstation product to display the trend data (Window Workstation or CrispView).
- The message counts for the DBASRV in the NETMON database should be increasing. These can be observed using PCWS and the standard display file.

#### IDC

##### Test items:

- Create a list of transfer instructions containing both local and remote databases. Configure CRISP/32 to use IDC instead of database transfers.
- Use the IDCCMP command to prepare the transfers.
- Start the system with CRSTART. IDC should start.
- Verify that the transfers work, both to the local databases and the remote databases.
- Modify the transfer list, and re-process with IDCCMP. Verify that the transfers still work properly.

#### Dual system test case 1

- Create a list of transfer instructions containing databases that are on a dual (active/standby) system. Configure CRISP/32 to use IDC instead of database transfers.
- Use the IDCCMP command to prepare the transfers.
- Start the system with CRSTART. IDC should start.
- Verify that the transfers work to the databases on both systems.

#### Dual system test case 2 (VAX only)

- Create a test system that is part of a dual (active/standby) system, and install the same databases on both systems. (One of the systems can be a V3.0 system.)
- Create a list of transfer instructions containing instructions to synchronize data between the two systems. Configure CRISP/32 on each system to use IDC instead of database transfers.
- Use the IDCCMP command on each system to prepare the transfers.
- Start the systems with CRSTART. IDC should start on both systems.
- Verify that the transfers work. Cause a switchover and verify that the transfers continue to work.

#### Database Transfers

##### Test items:

- Create a list of transfer instructions. Configure CRISP/32 to use database transfers instead of IDC.
- Use the DBTCMD command to prepare the transfers.
- Start the system with CRSTART.
- Verify that the transfers work.
- Modify the transfer list, and re-process with DBTCMD. Verify that the transfers still work properly.

#### Monitor CRISP processes (CRMON command)

##### Test items:

- After CRSTART, get a list of processes being monitored with the "CRMON SHOW" command. The following CRISP core processes are always monitored: CLE and DBASRV. If I/Onyx is present, ICM is also monitored.
- Suspend the DBASRV with the command "SET PROC/SUSPEND CRISP\$DBASRV". A message should appear on CRISP\$TT that the process has stopped.
- Resume the DBASRV with the command "SET PROC/RESUME CRISP\$DBASRV". A message should appear on CRISP\$TT that the process has resumed.
- (Dual system) Suspend the DBASRV and verify that it causes a switchover.

## Historian

The Historian system consists of three processes: HISTORIAN, the controller; SEPARATOR, the data separator; and LOGGER, the logger. See the CASRV for the test items.

## I/Onyx

### Test items for ICF:

- Configuring the I/Onyx data should be normal.
- Configure I/Onyx and save a dialog file. Go back into ICF and change the configuration. Then go back into ICF, using the previous dialog file. The original configuration should be restored.
- The ICF "SHOW" command should give sensible results for each module type while CRISP is running.
- The ICF "DISPLAY" command should give sensible results whether or not CRISP is running.
- While CRISP is running, ICF should be able to enable and disable I/Onyx clusters at will.
- If CRISP is stopped, and ICF is at its command prompt, "CRSTART" should return an error.
- Attempting to run ICF if CRISP has not been started since reboot should fail.

### Test items for ICM:

- ICM should start as part of CRSTART.
- In a dual module server system, with CRISP running, reset the standby module server. A message should be displayed on CRISP\$TT, but the module servers should not switch over.
- In a dual module server system, with CRISP running, reset the active module server. A message should be displayed on CRISP\$TT, and the module servers should switch over.
- Resetting a module should cause a new block to be downloaded, and a message on CRISP\$TT should be displayed.

### Test items for inputs:

- The data being read from I/Onyx modules should be written to the correct locations in the database.
- The values should be sensible.
- Test the database variables on either side of the values transferred. They should be unaffected by I/Onyx operations. For example, if array X(10) has a 8-input digital input coming in starting at X(1), values X(0) and X(9) should be unaffected. Use a workstation to enter values for the variables outside the endpoints and verify that they remain unchanged.

### Test items for outputs:

- The data being written to I/Onyx modules should be read from the correct locations in the database.
- The values should be sensible.

## Classic I/O

### Test items for CCF and CCIO\$CCF\_COMPILE (CCC command):

- Configuring the Classic I/O data should be normal.
- The CCF commands should give sensible results for each module type while CRISP is running.
- While CRISP is running, CCF should be able to enable and disable Classic I/O clusters at will.

### Test items for CCM:

- CCM should start as part of CRSTART.
- In a dual CC server system, with CRISP running, reset the standby CC server. A message should be displayed on CRISP\$TT, but the CC servers should not switch over.
- In a dual CC server system, with CRISP running, reset the active CC server. A message should be displayed on CRISP\$TT, and the CC servers should switch over.

### Test items for inputs:

- The data being read from Classic I/O modules should be written to the correct locations in the database.
- The values should be sensible.
- Test the database variables on either side of the values transferred. They should be unaffected by Classic I/O operations. For example, if array X(10) has a 8-input digital input coming in starting at X(1), values X(0) and X(9) should be unaffected. Use a workstation to enter values for the variables outside the endpoints and verify that they remain unchanged.

### Test items for outputs:

- The data being written to Classic I/O modules should be read from the correct locations in the database.
- The values should be sensible.

## IDI

IDI operation should be as expected. There were no changes in IDI functionality for this release.

## Logic Calls

The logic calls in CRISP/32 V3.1 should perform identically to the calls in V3.0. No changes were made to the logic calls to change their operation. Anything "broken" in V3.0 will remain broken the same way in V3.1.

The recommended procedure is to do a side-by-side comparison between a VAX V3.0 system and the system under test. A logic and CWS display file is available to test all calls.(being updated by RJR presently).

The following logic calls are present in CRISP/32.

ABSALM_C16	ABS_C16
ADD	ADD_C16
ADD_TIMES	ALIAS_CREATE
ALIAS_DELETE	ALMDEV_C16
ALMLAT_C16	ALMLHD_C16

ALMLH_C16	AND_ARRAY
AND_C16	AND_NOT_ARRAY
ARRAY_SORT	BITOFF_C16
BITON_C16	BIT_PACK
BIT_TEST_ARRAY	BZERO
BZERO_C16	CHK_HIST_ALIAS
CMPRS	CMPRS_C16
COMPARE_TIMES	CONTROL_C16
CONTROL_P	CONTROL_PI
CONTROL_PID	CONTROL_PID_ADV
CONTROL_PID_PLUS	CONTROL_PI_PLUS
CONTROL_P_PLUS	CRTLOCK
CRTLOCKCHECK	CRTLOGDEST
CVT_INT_STR	CVT_STR_FLT
CVT_TIME_ASC_BIN	CVT_TIME_BIN_ASC
CVT_TIME_BIN_INTEGER	CVT_TIME_VEC_BIN
DACOSINE_C16	DASINE_C16
DATANGENT_C16	DAY_OF_WEEK
DCOSINE_C16	DELTAT_C16
DEVALM_C16	DEV_ALARM_1
DEV_ALARM_2	DISPLAY_ALARM_MSGS
DIVREM_C16	DIV_C16
DSINE_C16	DTANGENT_C16
END_STATS	EXP10_C16
EXP_C16	FAO
FIND_INDEX	FIXDAT
FIXDAT_C16	FIXTIM_C16
GET_HIST_END_TIME	GET_HIST_START_TIME
GET_SYMBOL_INFO	INITOD_C16
INIT_DATE_TIME_CONTEXT	INIT_STATS
ISTRAN	KLCRT_C16
KYLINK	KYLINK_C16
LN_C16	LOG_C16
LOOKUP_C16	MAKZERO
MAKZERO_C16	MULT_C16
NEWPT_STATS	NEW_DESTINATION
NOT_ARRAY	OR_ARRAY
OR_C16	POLY_C16
PRINT_ALARM_MSGS	PRTDSP
PRTDSP_C16	RACOSINE_C16
RANDOM_NUMBER	RASINE_C16
RATANGENT_C16	RCOSINE_C16
RDDSP	RDDSP_C16
READ_NETMON_MSG	REPORT_STATS
RESET_STATS	RSINE_C16
RTANGENT_C16	SAVEDB
SCALE_C16	SCAN_HIST
SCOPY	SCOPY_C16
SCROLL_MSG	SCROLL_MSG_PLUS
SENICC_C16	SETDSP
SETDSP_C16	SETMSG_C16
SETQRY_C16	SITRAN
SMERGE_C16	SQRT10_C16
SQRT_C16	STATUS_STRING
STR_EDIT	STR_MERGE
SUBMIT_BATCH	SUB_C16
SUB_TIMES	SUM
SUM_C16	TABLE_LOOKUP
TIME_AND_DATE	TRANSFER
TRANSFER_C16	TRANSFER_TMR_CNT
TRANSLATE_LOGICAL_NAME	UNITS_C16
VALUE_C16	WAKEUP
WWS_GETDSP	WWS_SETDSP
XOR_ARRAY	YRDAY_C16