The new utility, APR (Automatic Processor Recovery), will soon go into field beta test at selected sites. With this in mind, the following explanation of operation and functionality is offered to prepare the field for its use:

APR prerequisites are as follow:

1) 10.2 OS
2) FMI 93 (backplane upgrade and copmod)
3) Rev H, or above NSP
4) rev 2.0 SPU Unix
5) rev 3.0.0.6 Diagnostics
6) rev 3.0 SST

The intent of APR is to allow a single C3800 processor to be automatically disabled on a hard error. Other heads in the complex will continue to function uninterrupted.

Because of the data base modifications, it will be possible to run APR on systems where all heads are not at sufficient rev. In this case only heads at rev and with backplanes properly upgraded will be re-enabled on failure.

The user process running on that head will be terminated and a core file created. All other processes will be unaffected. If the user process is executing a system call on the failed head, then the process will be signaled and continue to execute normally. Again, only process executing in ring 4 are recoverable.

If the kernel is executing on the failed head then the system will panic. The panic will be:

ConvexOS:FATAL ERROR:(Sched 8825)irrecoverable hard error: context jump not set

The object of APR will be to deallocate the head, execute SST on the failed head and if it passes, return the head to operation. All options as to number of failures allowed and whether tests are run are fully selectable by the Convex Field personnel.

The heart of APR will be the cpu_monitor daemon, which will monitor the state of the processors on a regular basis. The default will be 60 second intervals. The cpu_monitor daemon will be started in the rc.local file.

The cpu_monitor daemon will be programmable from the file "/etc/cpu_monitor.config file. This file is displayed below:

```
AUTO_REENABLE = 0;    # 0 Indicates no auto re-enable
SST_ENABLE = 1;       # 1 indicates run SST before re-enable. If the head
                        fails SST then the head will not be re-enabled.
FAILURE_COUNT = 0;    # 0 indicates that unlimited errors will result in
                        restart. Any other value will indicate number of
                        retires in the below time period.

FAILURE_COUNT_TIME_LIMIT = 24  # Indicates time limit for failure count.
```

Additions have also been made to the CDB to support APR:

enable_cpu_harderrs - 0 will allow APR functionality, 1 will enable normal error processing. Controlled by xsys_config.

cpu_hard_err_sniff_period - Used bu cpu_monitor daemon to control monitor interval (In seconds).

dynamic_cpu_harderrs_n - Indicates if CPU and backplane for each head support APR.

cpu_automatic_realloc_n - Specifies whether to automatically re-enable failed head.

The backplane modification will consist of a single net from the NVP to the NSP which will inform the NSP in case of a NVP hard error. The backplane must be cop modified to indicate the presence of this net before APR can function.

To determine if a head is APR capable it is, only, necessary to perform:

cdb_get dynamic_cpu_harderrs_n, where n is the CPU number

A value of "1" indicates that the head is hardware capable.

A new switch is available, in xsys_config, to allow the enabling and disabling of APR/CPU hard errors.

The utility cpuconf <-e, -d> can be used to disable a head from the OS complex. When the head is enabled, control store load and sysreset will be automatically performed on the individual head.

Execution of sst will be controlled by the file /diag/hw/cpualloc.test. This file will be used to execute sst on disabled heads, in the form:

cpualloc.test n, Where n is the CPU to be tested.

This will force execution of SST on the effected NSP and NVP and indicate whether the test passed, or failed.

It is also possible to power the indicated head off and replace and execute diaginit on the replaced head while the system is in OS.

APR Procedures

To test APR, perform the following. If the disabled head is not automatically brought back on line, APR is not working properly.

```
flt_6: cpuconf -d 1
[Feb 26 07:22:49 1993] cpualloc: CPU 1 is now off-line
cpu0  enabled
cpu1  DISABLED
cpu2  DISABLED
cpu3  DISABLED
cpu4  DISABLED
cpu5  DISABLED
cpu6  enabled
cpu7  enabled
flt_6: date
Fri Feb 26 07:22:57 CST 1993
flt_6: Feb 26 07:23:15 APR[134]: CPU 1 has gone down.
Feb 26 07:24:58 APR[134]: Attempting to restart CPU 1
Feb 26 07:25:13 APR[134]:  Current CPU allocation: 1 1 0 0 0 0 1 1
flt_6: cpuconf
cpu0  enabled
cpu1  enabled
cpu2  DISABLED
cpu3  DISABLED
cpu4  DISABLED
cpu5  DISABLED
cpu6  enabled
cpu7  enabled
```

If the head is not brought back on line, check the following:

1) /etc/cpu_monitor.config (ConvexOS) should be set up as follows:

```
AUTO_REENABLE = 1;
SST_REENABLE = 1;
FAILURE_COUNT = 4;       # always re-enable the cpu
FAILURE_COUNT_TIME_LIMIT = 24; # time in hours to keep track of cpu failures
```

Standart deafoult should be set

2) /etc/rc.local (ConvexOS) must have the following entry at the bottom of this file.

```
        if [ -f /etc/cpu_monitor -a -f /etc/cpu_monitor.config ]; then
                /etc/cpu_monitor &
                echo -n ' cpu_monitor'
        fi
```

3) Check the following at spu level:

```
spu> cdb_browser
```

let us to go to the configuration database

```
  1. Mode (verbosity level)
  2. Dump database
  3. Item query
  4. Update
  5. Quit
   > 3
   Enter target string > dynamic_cpu_harderrs*
dynamic_cpu_harderrs_0 =   00000001                (Each installed head
dynamic_cpu_harderrs_1 =   00000001                must be set to = 1.
dynamic_cpu_harderrs_2 =   00000000                If not, perform the
dynamic_cpu_harderrs_3 =   00000000                following to set to 1)
dynamic_cpu_harderrs_4 =   00000000
dynamic_cpu_harderrs_5 =   00000000
dynamic_cpu_harderrs_6 =   00000001
dynamic_cpu_harderrs_7 =   00000001

   Enter target string > enable_cpu_harderrs       (This must = 0.)
enable_cpu_harderrs =   00000000

   Enter target string > q

  1. Mode (verbosity level)
  2. Dump database
  3. Item query
  4. Update
  5. Quit
   > 4
   Enter name > dynamic_cpu_harderrs_0            (This command will set
   Enter new integer > 0                           head 0 to 0.)
   Enter name > q

  1. Mode (verbosity level)
  2. Dump database
  3. Item query
  4. Update
  5. Quit
   > 3
   Enter target string > dynamic_cpu_harderrs_*    (Now you can see that
dynamic_cpu_harderrs_0 =   00000000 -> tu powinno być 0 is set to 0.)
dynamic_cpu_harderrs_1 =   00000001   1
dynamic_cpu_harderrs_2 =   00000000   so
dynamic_cpu_harderrs_3 =   00000000
dynamic_cpu_harderrs_4 =   00000000
dynamic_cpu_harderrs_5 =   00000000
dynamic_cpu_harderrs_6 =   00000001
dynamic_cpu_harderrs_7 =   00000001

   Enter target string >q    do głównego menu

  1. Mode (verbosity level)   i ustawiamy cpu0 = 1
  2. Dump database
  3. Item query
  4. Update
  5. Quit
   > 4
   Enter name >    Enter name > dynamic_cpu_harderrs_0 -> cpu0
   Enter new integer > 1                          (This will set head 0
   Enter name > q                                  to a 1.)

  1. Mode (verbosity level) <- sprawdzamy czy się dobrze ustawiło
  2. Dump database
  3. Item query
  4. Update
  5. Quit
   > 3   ->
   Enter target string > dynamic_cpu_harderrs*
dynamic_cpu_harderrs_0 =   00000001 -> OK.         (All installed heads
dynamic_cpu_harderrs_1 =   00000001                are now set to 1).
dynamic_cpu_harderrs_2 =   00000000
dynamic_cpu_harderrs_3 =   00000000
dynamic_cpu_harderrs_4 =   00000000
dynamic_cpu_harderrs_5 =   00000000
```

```
dynamic_cpu_harderrs_6 =   00000001
dynamic_cpu_harderrs_7 =   00000001

   Enter target string > q

  1. Mode (verbosity level)
  2. Dump database
  3. Item query
  4. Update
  5. Quit
   > 5
spu>
```