PREPARED BY
E. Casero

APPROVED BY
E. Maccario

REVIEWED BY
M. Nobile

Honeywell Information Systems Italia

"THIS DOCUMENT AND THE INFORMATION CONTAINED HEREIN ARE CONFIDENTIAL TO AND THE PROPERTY OF HONEYWELL INFORMATION SYSTEMS ITALIA AND ARE MADE AVAILABLE ONLY TO HONEYWELL EMPLOYEES FOR THE SOLE PURPOSE OF CONDUCTING HONEYWELL'S BUSINESS. THIS DOCUMENT, ANY COPY THEREOF AND THE INFORMATION CONTAINED HEREIN SHALL BE MAINTAINED IN STRICTEST CONFIDENCE, SHALL NOT BE COPIED IN WHOLE OR IN PART EXCEPT AS AUTHORIZED BY THE EMPLOYEE'S MANAGER, AND SHALL NOT BE DISCLOSED OR DISTRIBUTED (A) TO PERSONS WHO ARE NOT HONEYWELL EMPLOYEES, OR (B) TO HONEYWELL EMPLOYEES FOR WHOM SUCH INFORMATION IS NOT NECESSARY IN CONNECTION WITH THEIR ASSIGNED RESPONSIBILITIES, UPON REQUEST, OR WHEN THE EMPLOYEE IN POSSESSION OF THIS DOCUMENT NO LONGER HAS NEED FOR THE DOCUMENT FOR THE AUTHORIZED HONEYWELL PURPOSE, THIS DOCUMENT AND ANY COPIES THEREOF SHALL BE RETURNED TO THE EMPLOYEE'S MANAGER. THERE SHALL BE NO EXCEPTIONS TO THE TERMS AND CONDITIONS SET FORTH HEREIN EXCEPT AS AUTHORIZED IN WRITING BY THE RESPONSIBLE HONEYWELL VICE PRESIDENT."
INDEX

1. INTRODUCTION
2. LEVELS OF DIAGNOSIS
3. LEVEL 1 (START)
   3.1 INTEGRATED DIAGNOSTIC ROUTINES (IDR)
   3.2 RESIDENT DIAGNOSTIC ROUTINES (RDR)
      3.2.1 RDR OF CPO BOARD
      3.2.2 RDR OF SPO BOARD
      3.2.3 RDR OF LPO BOARD
   3.3 ERROR MESSAGES
   3.4 BOOTSTRAP
   3.5 DISPLAY CODES
   3.6 DIAGNOSTIC/STANDARD O.S. INTERFACE
4. LEVEL 2 (STAL)
   4.1 INTRODUCTION
   4.2 DIAGX2 STRUCTURE
   4.3 DIRECTORY FILE - P_SECT
   4.4 STAL DIAGNOSTIC PROCESSES
   4.5 PRIMARY (SGM2BT)
   4.6 DIAGNOSTIC MONITOR
   4.7 DIAGNOSTIC MONITOR COMMANDS AND UTILITIES
      4.7.1 USER COMMANDS
      4.7.2 SYSTEM COMMANDS
         4.7.2.1 DG - DIAGX2 FLOPPY GENERATION
         4.7.2.2 DM - DIAGX2 FLOPPY MODIFY
4.7.2.3 DD - DIAGX2 FLOPPY DUP
4.7.2.4 br - RECYCLE ON USER "b" SELECTION
4.7.2.5 (name) - LOAD & EXECUTE A PROCESS FROM DIAGX2
4.7.2.6 LD - LOAD A PROCESS FROM DIAGX2
4.7.2.7 GO - START/RESTART A PROCESS
4.7.2.8 OS - REBOOT SYSTEM STARTING FROM DISK
4.7.2.9 BR - SET/DISPLAY BREAKPOINT
4.7.2.10 NB - CLEAR BREAKPOINT
4.7.2.11 PR - DISPLAY SYSTEM H/W CONFIGURATION
4.7.2.12 DR - DISPLAY ALL PROCESSOR REGISTERS
4.7.2.13 DD..D7 - DISPLAY/MODIFY PROCESSOR DATA REGISTERS
4.7.2.14 AO..A7 - DISPLAY/MODIFY PROCESSOR ADDRESS REG.
4.7.2.15 SR - DISPLAY/MODIFY PROCESSOR STATUS REGISTER
4.7.2.16 PC - DISPLAY/MODIFY PROCESSOR PROGRAM COUNTER
4.7.2.17 MD - DISPLAY MEMORY ON CONSOLE
4.7.2.18 MM - MEMORY DISPLAY/MODIFY
4.7.2.19 FT - FORMAT A FLOPPY
4.7.2.20 LS - LIST DIRECTORY OF DIAGX2 FLOPPY
4.7.2.21 PN - LIST NAMES OF DIAGNOSTIC PROCESSES
4.7.2.22 RL - LIST REVISION LEVELS

4.8 DIAGNOSTIC PROCESSES EXECUTION MODE
4.8.1 AUTOMATIC MODE
4.8.2 TECHNICAL MODE
4.9 DIAGNOSTIC PROCESS OF CPG BOARD (802CPX)
4.9.1 OPERATING RULES
4.9.2 ERROR REPORTING

Honeywell Confidential And Proprietary
4.10 DIAGNOSTIC PROCESS OF DISK (SG2DSK)
4.10.1 OPERATING RULES
4.10.2 FORMAT SELECTION
4.10.3 WRITE SELECTION
4.10.4 DIAGNOSTIC TEST SELECTION
4.10.5 READ SELECTION
4.10.6 DISPLAY BAD BLOCK TABLE SELECTION
4.10.7 DISPLAY DCO REGISTERS & ECA TABLE SELECTION
4.10.8 CHANGE EXECUTION MODE SELECTION
4.10.9 END OF TEST SELECTION
4.10.10 ERROR REPORTING

4.11 DIAGNOSTIC PROCESS OF FLOPPY (SG2FLP)
4.11.1 OPERATING RULES
4.11.2 FORMAT SELECTION
4.11.3 WRITE SELECTION
4.11.4 DIAGNOSTIC TEST SELECTION
4.11.5 TEST ON DIAGX2 SELECTION
4.11.6 READ SELECTION
4.11.7 DISPLAY DCO REGISTERS & ECA TABLE SELECTION
4.11.8 COPY SELECTION
4.11.9 CHANGE EXECUTION MODE SELECTION
4.11.10 END OF TEST SELECTION
4.11.11 ERROR REPORTING

4.12 DIAGNOSTIC PROCESS OF MEMORY (SG2MEM)
4.12.1 OPERATING RULES
4.12.2 ERROR REPORTING
4.13   DIAGNOSTIC PROCESS OF STREAMER (SG2STR)
4.13.1  OPERATING RULES
4.13.2  "A" SELECTION
4.13.3  "B" SELECTION
4.13.4  "C" SELECTION
4.13.5  "D" SELECTION
4.13.6  "X" SELECTION
4.13.7  ERROR REPORTING
4.14   DIAGNOSTIC PROCESS OF CACHE BOARD (SG2CHX)
4.14.1  OPERATING RULES
4.14.2  ERROR REPORTING
4.15   DIAGNOSTIC PROCESS OF SCO BOARD (SG2SCX)
4.15.1  OPERATING RULES
4.15.2  ERROR REPORTING
4.16   DIAGNOSTIC PROCESS OF VMEbus CONTROLLERS (SG2VMX)
4.16.1  OPERATING RULES
4.16.2  ................. CONTROLLER TEST ("A" SELECTION)
4.16.3  ................. CONTROLLER TEST ("B" SELECTION)
4.16.4  TEST OF ALL THE UNITS ("X" SELECTION)
4.16.5  END OF TEST ("K" SELECTION)
4.16.6  ERROR REPORTING
4.16.6.1 ... TEST
4.16.6.2 ... TEST
4.17   PLUGS TEST TOOL (SG2TAP)
4.17.1  OPERATING RULES
4.17.2  ERROR REPORTING
4.18  DISK FLOPPY CONTROLLER TEST TOOL (SG2DFC)

4.18.1 OPERATING RULES
4.18.2 DIAGNOSTIC TEST SELECTION
4.18.3 ACCEPTANCE TEST SELECTION
4.18.4 FLOPPY EMULATION SELECTION
4.18.5 DISPLAY DCON REGISTERS & ECA TABLE SELECTION
4.18.6 CHANGE EXECUTION MODE SELECTION
4.18.7 END OF TEST SELECTION
4.18.8 ERROR REPORTING

4.19  BASIC DEVICE CONTROLLER TEST TOOL (SG2BDC)

4.19.1 OPERATING RULES
4.19.2 ERROR REPORTING

5.  LEVEL 3 (DIAG)

5.1  INTRODUCTION
5.2  INTEGRATION INTO O.S. FILE SYSTEM

5.2.1 DIAG FILE SYSTEM
5.3  FUNCTIONALITIES

5.3.1 MONITOR
5.3.2 TESTS

5.3.2.1 SGCPU
5.3.2.2 SGSYS
5.3.2.3 SGIOB
5.3.2.4 SGIOC
5.3.2.5 SGTTYW
5.3.2.6 SGTTYR
5.3.2.7 SGPRT
5.3.2.8 SGFS

Honeywell Confidential And Proprietary
5.3.2.9  SGCOMM
5.3.3  SERVICE PROCESSES
5.3.3.1  TIMER PROCESS
5.3.3.2  SERVICE PROCESS
5.4  COMMAND FILE FORMAT
5.5  OPERATING RULES
5.5.1  USER REGISTRATION
5.5.2  DIAG USE
5.6  MESSAGES
5.6.1  STATUS MESSAGES
5.6.2  ERROR MESSAGES
1. INTRODUCTION

This manual describes in details the SGM2 diagnostic system use.

Diagnostic System is composed by:

- Resident code
- Test under Diagnostic operating system
- Tests under Standard operating system

2. LEVEL_0E_DIAGNOSIS

The SGM2 diagnosis is made at three levels:

- Level 1 (START) Errom resident tests
- Level 2 (STAL) Tests under Diagnostic stand-alone operating system
- Level 3 (DIAG) Tests under Standard (UNIX) operating system

3. LEVEL_1 (START)

It's always executed during the initialization phase. It's divided into three parts:

Integrated Diagnostic Routines
Resident Diagnostic Routines
Diagnostic Bootstrap
3.1 INTEGRATED_DIAGNOSTIC_ROUTINES (IDR)

These routines (also called SELFTEST) are coded into each standard VMEbus controllers' EPROM. They check integrity of H/W chips on the board, and supply an error code if any failure is found.

The results of IDR's tests are collected and checked by Diagnostic stand-alone Processes: if errors have been found, an error code is send on console.

3.2 RESIDENT_DIAGNOSTIC_ROUTINES (RDR)

These software routines have been developed in order to detect failures into each proprietary Controller board (CPO-DCO-SPO-LPO). They are executed every time the system is switched on or reset. They are coded in EPROM and written in Motorola Assembler 68000.
3.2.1 RDR.DE.CPD_BOARD

The following areas are tested by RDR:

EPROM
- checksum control

PIT
- registers integrity
- interval timer

MMU
- "resisters" integrity
- read/write memory through descriptors
- error management
  - read/write violation
  - operations on disable segment
  - operations out of segment

MEMORY
- addressability (all the present memory)
- integrity (only up to 512 Kbyte)

CPO/SPO INTERRUPTS

SCO CONTROLLER
BIM: registers integrity

DCO CONTROLLER
- check of self-test results
STREAMER: data register integrity
    (interface)

At the end of RDR execution, the following steps are performed:

- disable MMU translations
- clear main memory
- display OK code

Note: RDR's have to run almost for up to 35 seconds, to be sure hard disks have speeded up.
3.2.2  RDR_QE_SPO_BOARD

The following areas are tested by RDR:

- EPROM  
  - checksum control
- STATIC L_RAM  
  - addressability
  - integrity
- STATIC S_RAM  
  - addressability
  - integrity
- PIT  
  - registers integrity
  - interval timer
- SIO  
  - registers integrity
- CPU/SPO interrupts

3.2.3  RDR_QE_LPO_BOARD

The following areas are tested by RDR:

- EPROM  
  - checksum control
- STATIC L_RAM  
  - addressability
  - integrity
- STATIC S_RAM  
  - addressability
  - integrity
- DMA I/O MEM  
  - addressability
  - integrity
- PIT  
  - registers integrity
  - interval timer
- SIO  
  - registers integrity
- CPU/DMA INTERRUPTS
- CPU/LPO INTERRUPTS
3.3 ERROR MESSAGES

If an error occurs during the execution of RDR's, a message is shown on the display. The message consists of a number either blinking or steady, as shown in the following table:

<table>
<thead>
<tr>
<th>CODE</th>
<th>ERROR DESCRIPTION</th>
<th>ORU</th>
<th>CRU</th>
</tr>
</thead>
</table>

* = blinking value
3.4 BOOTSTRAP

In developing this bootstrap sequence, also controller and device operability has been taken into account.

Before starting the physical I/O operation on the device, SELFTEST command is sent to controller, in order to assure the correct functionality of IMDC component.

Boostrap takes place trying to operate on the devices selected in the following sequence:

- diskette (diagnostic or standard O.S.)
- hard disk 0 (standard O.S.)
- hard disk 6 (standard O.S.)

The following command sequence is executed on the selected device:

- RECALIBRATE
- SEEK TO CYLINDER
- READ (sector 0 - head 0 - cylinder 0)

The contents of the loaded sector are tested by means of a pattern control, in order to detect whether they really belong to an operating system load module.
3.5 DISPLAY CODES

The following steady digits can be displayed:

T.B.D.
### 3.6 DIAGNOSTIC/STANDARD O.S. INTERFACE

When the O.S. (diagnostic or standard) is loaded into memory, it can have useful information about the system configuration by reading appropriate locations.

Memory structure, after RDR’s & BOOTSTRAP, is as follows:

<table>
<thead>
<tr>
<th>Addr.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000</td>
<td>DCO/DPO entry table</td>
</tr>
<tr>
<td>000010</td>
<td>DC1/DP1 entry table</td>
</tr>
<tr>
<td>000020</td>
<td>S/LP0 entry table</td>
</tr>
<tr>
<td>000030</td>
<td>S/LP1 entry table</td>
</tr>
<tr>
<td>000040</td>
<td>S/LP2 entry table</td>
</tr>
<tr>
<td>000050</td>
<td>S/LP3 entry table</td>
</tr>
<tr>
<td>000060</td>
<td>S/LP4 entry table</td>
</tr>
<tr>
<td>000070</td>
<td>S/LP5 entry table</td>
</tr>
<tr>
<td>000080</td>
<td>S/LP6 entry table</td>
</tr>
<tr>
<td>000090</td>
<td>S/LP7 entry table</td>
</tr>
<tr>
<td>0000A0</td>
<td>VMEbus controller 0 entry table</td>
</tr>
<tr>
<td>0000B0</td>
<td>VMEbus controller 1 entry table</td>
</tr>
<tr>
<td>0000C0</td>
<td>VMEbus controller 2 entry table</td>
</tr>
<tr>
<td>0000D0</td>
<td>VMEbus controller 3 entry table</td>
</tr>
<tr>
<td>0000E0</td>
<td>CP0 entry table</td>
</tr>
<tr>
<td>0000F0</td>
<td>CP1 entry table</td>
</tr>
<tr>
<td>000100</td>
<td>CHO entry table</td>
</tr>
<tr>
<td>000110</td>
<td>CH1 entry table</td>
</tr>
<tr>
<td>000120</td>
<td>SCO entry table</td>
</tr>
<tr>
<td>000130</td>
<td>R.F.U. entry table</td>
</tr>
</tbody>
</table>

Honeywell Confidential And Proprietary
000140 MEMORY BIT MAP

000144 INITIALIZING DEVICE NUMBER

- Each board entry table contains information about:
  - Board presence/absence
  - Board functionality
  - Slot insertion number
  - Processor type
  - Lines information (only for L/SPO boards)

- Memory bit map gives information about memory configuration, on 32-bit length. Each bit, starting from the least significant bit of a long word, represents 1 memory Mb, as follows:
  - 0 = memory absent or faulted
  - 1 = memory present

- Initializing device number supplies the number of disk from which Operating System has been loaded, starting from 00.
4. LEVEL_2(STAL)

4.1 INTRODUCTION

A stand-alone Diagnostic System (STAL) is available on SGM2 System. STAL is shipped with the system via a floppy, labelled DIAGX2.

The STAL goals are the covered and the localization of failures, at CRU level, in the following areas:

- CPO board
- SCD board
- CHO board
- Memory boards
- DCO/DPO controller board
- SPO/LPO controller board
- VMEbus standard controller boards
- Disk devices
- Floppy device
- Streamer device

4.2 DIAGX2_STRUCTURE

The floppy DIAGX2 has a sequential files structure; files access is made via the parameters TRACK/SECTOR/HEAD of the file start, getted from the DIAGX2 directory file.

The DIAGX2 sector size is 512 bytes, with 9 sectors per track.

The free tracks on DIAGX2 are available for floppy diagnostic Process to perform Format/Write/Read test.

The floppy structure is shown in the following table:
# DIAGX2_Elexex_structure

<table>
<thead>
<tr>
<th>TRK 00</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>08</th>
<th>09</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRIMARY_IP_SECTFILE 1..</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TRK 01</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>08</th>
<th>09</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILE 2..</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TRK 02</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>08</th>
<th>09</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1st</td>
<td>free</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TRK n+1</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>08</th>
<th>09</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AVAILABLE FOR FORMAT/WRIT/READ TESTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TRK 79</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>08</th>
<th>09</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AVAILABLE FOR FORMAT/WRIT/READ TESTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Honeywell Confidential And Proprietary
4.5 PRIMARY_{SGM2BT_}

The Boot file is the 1st DIAGX2 sector (sect. 1 - cyl. 0 - head 0). It is loaded into memory, at address $F5E20, by resident Bootstrap.

SGM2BT target is to load from DIAGX2 floppy the directory file (P_SECT) and the Diagnostic Monitors (CPOMON-L/SPOMON). Correct RDR execution on the other controllers are also checked, reading error information into each board entry table.

Errors during this phase are signaled using STATUS display. The possible displayed values are:

T.B.D.
4.3 DIRECTORY_FILE - P_SECT

The sector 2 of DIAGX2 is used as directory for the files recorded on the floppy.

Each file has an entry in P_SECT, on 16 bytes, with a structure as follows:

<table>
<thead>
<tr>
<th>Bytes count</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-5</td>
<td>file name (must be on 6 character)</td>
</tr>
<tr>
<td>6-7</td>
<td>start cylinder</td>
</tr>
<tr>
<td>8</td>
<td>head</td>
</tr>
<tr>
<td>9</td>
<td>sector</td>
</tr>
<tr>
<td>11-12</td>
<td>length (bytes)</td>
</tr>
<tr>
<td>13-16</td>
<td>entry address of Process</td>
</tr>
</tbody>
</table>

The last 4 bytes of P_SECT contains the parameters (Sector/Cylinder/Head) of the 1st free sector on DIAGX2.

4.4 SIAL_DIAGNOSTIC_PROCESSES

The following Processes are recorded on DIADKT floppy:

- SGM2BT      Primary boot file
- CPOMON      Diagnostic Monitor (CPO)
- SPOMON      "Monitor (SPO)
- LPOMON      "Monitor (LPO)
- SG2CPX      Diagnostic Process of CPO
- SG2SCX      Diagnostic Process of SCO
- SG2CHX      Diagnostic Process of CHO
- SG2SPX      Diagnostic Process of SPO
- SG2LPO      Diagnostic Process of LPO
- SG2MEM      "" "" Memory
- SG2DSK      "" "" Disk
- SG2FLP      "" "" Floppy
- SG2STR      "" "" Streamer
- SG2VMX      "" "" VMEbus cntr
- SG2TAP      Line/Printer/Streamer BUS test
- SG2BDC      Test tool for Disks/Floppy
- SG2DFC      Test tool for DCO controller
4.6 DIAGNOSTIC MONITOR

The STAL Diagnostic Monitor hands the console dialogs with the user, loads and puts in execution the Diagnostic Processes, and hands the tests results. It also contains some utilities (see point 2.2.5.3.3)

STAL console dialogs with user is made in Italian or English language.

The language selection is made at initialization time, using console line testing phase. User is required to input from console a characters string: “abcd” or “dcba”.

- if “abcd” is introduced, console dialogs will be in Italian
- if “dcba” is introduced, console dialogs will be in English

The Diagnostic Monitor is logically divided into two parts:
- System Monitor (CPOMON), running on CPU 68020
- I/O Monitor (L/SPOMON), running on each L/SPO board

The Diagnostic Monitor has also some control keys, to edit input from console. They are:

- BACKSPACE the last character entered from console is deleted
- BREAK all the current running operations are stopped, and the control returns to Monitor
4.7 DIAGNOSTIC_MONITOR_COMMANDS_&_UTILITIES

Diagnostic Monitor is able to hand two types of commands:

- user commands
- system commands

User commands are in lower-case letters, and they are shown on console at the end of initialization phase, as the User Selection Menu.

System commands are in upper-case letters, and they are shown entering from console the command "HE".

4.7.1 USER_COMMANDS

See manual "SYSTEM TESTING GUIDE" for user command operating rules. Follows a brief description of these commands.

a The System H/W configuration is shown on console, and then all the Diagnostic Process relatives to existent H/W resources are automatically runned.

The chain Process order is as follows:

- SG2CPX
- SG2SCX
- SG2CHX
- SG2MEN
- SG2SPX
- SG2LPX
- SG2DSK
- SG2FLP
- SG2STR
- SG2VMX
At the end of last Process, the system is automatically reboot from the first ready disk unit.
The chain execution is stopped if a Diagnostic Process signal an error.

This command performs the same steps as "a" command, except the last.
At the end of last Process execution, the User Selection Menu is shown again on console.
The chain execution is stopped if a Diagnostic Process signal an error.

The Diagnostic Process of CPO board (SG2CPX) is loaded and executed in recycle mode.
The execution is stopped if a "break" is send from console, or if an error is found by Diagnostic Process.

The Diagnostic Process of SCO board (SG2SCX) is loaded and executed in recycle mode.
The execution is stopped if a "break" is send from console, or if an error is found by Diagnostic Process.

The Diagnostic Process of CHO board (SG2CHX) is loaded and executed in recycle mode.
The execution is stopped if a "break" is send from console, or if an error is found by Diagnostic Process.

The Diagnostic Process of Memory (SG2MEM) is loaded and executed in recycle mode.
The execution is stopped if a "break" is send from console, or if an error is found by Diagnostic Process.
The Diagnostic Process of SPO/LPO board (SG2SPX/SG2LPX) is loaded and executed in recycle mode. The execution is stopped if a "break" is send from console, or if an error is found by Diagnostic Process.

The Diagnostic Process of Disk (SG2DSK) is loaded and executed in recycle mode. Read only tests are performed, handling bad disk sectors, on all the media. On diagnostic cylinder (the last device cylinder), error management tests and Format/Write/Read operations are performed. The execution is stopped if a "break" is send from console, or if an error is found by Diagnostic Process.

The Diagnostic Process of Floppy (SG2FLP) is loaded and executed in recycle mode. If floppy is written protected, read only tests are performed on all the media, else also Format/Write/Read verify tests are performed on not used DIADKT floppy cylinders. The execution is stopped if a "break" is send from console, or if an error is found by Diagnostic Process.

The Diagnostic Process of Streamer (SG2STR) is loaded and executed in recycle mode. The execution is stopped if a "break" is send from console, or if an error is found by Diagnostic Process. Read only tests are performed.

The Diagnostic Process of VMEbus Controller boards (SG2VMX) is loaded. Another User Selection Menu is shown on console, and a new selection must be entered, corresponding to the VMEbus Controller board that is to be tested. The selected Diagnostic Process is executed in recycle mode, until a "break" is send from console, or an error is found.
4.7.2 SYSTEM_COMMANDS

The system commands are:

HE  Show list of commands
DG  DIADKT floppy generation
DM  DIADKT floppy modify
DD  DIADKT floppy dup
bR  Recycle on user "b" selection
(name)  Load and execute a Process from DIADKT
OS  Re-boot the system starting from Disk
LD  Load a Process from DIAGX2 into memory
GO  Start/Restart a Process
NR  Set/display breakpoint
NB  Clear breakpoint
PR  Display system H/W configuration
DR  Display all PROCESSOR registers
Dx  Display/modify "x" PROCESSOR data register
Ax  Display/modify "x" PROCESSOR add. register
SR  Display/modify PROCESSOR Status Register
PC  Display/modify PROCESSOR Program Counter
MD  Memory dump
MM  Memory display/modify
LS  List DIAGX2 floppy directory
PN  List names of Diagnostic Processes
FT  Format a floppy
RL  List revision levels

A detailed description of these commands is given in the following sections.

NOTE_1 In the following, optional values will be indicated using [ ], needed values using ()
4.7.2.1 DG==DIAGX2 floppy generation

This command is used to generate DIAGX2 floppy, starting from an already formatted media. The files to be recorded onto DIAGX2 (Diagnostic Processes) are sent from a Motorola system to SGM2 memory, via a line between the two Systems.

Command use: DG

= DG
* Load SGM2BT from MOTOROLA system *
* Loading .... *
* Enter Process name to be loaded ("EXIT" to end)
= CPOMON
* Load CPOMON from MOTOROLA system *
* Loading .... *
* Enter Process name to be loaded ("EXIT" to end)
= ..... .
* = EXIT

4.7.2.2 DM==DIAGX2 floppy modify

The DM command permits to change already recorded Processes onto DIAGX2 floppy, or to add new Processes. The files are sent via a line from a Motorola System to SGM2 memory.

Command use: DM

= DM
* Enter Process name to be loaded ("EXIT" to end)
= SG2DSK
* Load SG2DSK from MOTOROLA system *
* Loading .... *
* Enter Process name to be loaded ("EXIT" to end)
= EXIT

At this point the loaded Process may be executed, giving the corresponding execution command, without re-boot the system.
4.7.2.3 \texttt{DO=\_\_\_DIAGX2\_\_floppy\_\_due}

The \texttt{DO} command makes a logical copy of DIAGX2 floppy onto another already formatted media. Only Processes have an entry in DIAGX2 directory are copied.

Command use: \texttt{DD}

\begin{verbatim}
= DD
\textbackslash w Mount the new floppy. Reply Y when done.
= Y
\end{verbatim}

4.7.2.4 \texttt{bR=\_\_Recycle\_\_on\_\_user\_\_b\_\_selection}

Using this command is possible to recycle on the Diagnostic Processes chain execution.

At the start of first step, the H/W system configuration table is shown on console, requiring a "carriage return" to continue. The table doesn't appear on the other passes start.

The execution is stopped if a "break" is send from console, or if an error is found by a Diagnostic Process.

Command use: \texttt{bR}

4.7.2.5 \texttt{(name)=\_\_Load\_\_and\_\_execute\_\_a\_\_Process\_\_from\_\_DIAGX2}

Is possible to load from DIAGX2 floppy a Process and to put it in execution, giving the Process name as a command.

Command use: \texttt{(name)}

\begin{verbatim}
(name) = Name of Process to be load
\textbackslash w SG2FLP
\textbackslash w Recycle on test ? (Y/N) \textbackslash w
= Y
\end{verbatim}

The Process will be loaded and executed. See point 2.2.6 for more detailed description.
4.7.2.6  LD_=_Load_a_Process_from_Diagx2

It is possible to load from DIAGX2 floppy a Process, at the given address, using LD command.

Command use:  LD <name> <address>

<name> = Name of Process to be loaded
<address> = Loading address

= LD SG2FLP 6000
= LD TEST1 84000

The indicated Processes will be loaded at the given addresses. The control, after each load, is at Monitor level.

4.7.2.7  GO_=_Start/Restart_a_Process

It is possible to start/restart a Process, at the given address, using GO command.

Command use:  GO [address]

[address] = Restart address
Default = Last active breakpoint address.
If no active breakpoints, 0000

= GO

The program execution restarts at the last active break-point address. If no break-points are active, an error message is send on console.

= GO 6840

The program execution restart with the instruction at the given address.
4.7.2.8 OS - Re-boot_the_system_starting_from_Disk

The command OS gives the possibility to re-boot quickly the system, without press RESET pushbutton, starting from any Disk.

A very useful feature offered by this command is by-pass of the Bootstrap initialization chain (Floppy-Disk0-...-Disk6).

This chain automatically boots the system from the first ready System media; if Disk 0 is a System Disk, is not possible to boot automatically from another Disk.

Using OS command, is possible to boot from another Disk without remove first Disk from the system.

Command use: OS [value]
- [value] = Disk device number, between 0 and 6
  Default = 0
  = OS  The system is re-boot starting from Disk 0
  = OS 2  The system is re-boot starting from Disk 2

4.7.2.9 BR - Set/display_breakpoint

Using BR command, is possible to put breakpoints into a Process code, and to display them on console.

When a breakpoint is encountered, CPO or L/SPO stops the Process code execution and go to Monitor command level, giving the possibility to look at processor state, at memory values, at H/W registers, or to change all these values. The Process code execution is restarted using “GO” command.

The breakpoints mechanism is obtained using Illegal Instruction trap. The user code is substituted by $4AFB illegal code, and restored before restarting execution with “GO” command.

Command use: BR <address>

  <address> = Breakpoint address
  = BR 1048D2 (L_RAM address - executed by SPO)
  = BR 6000 (memory address - executed by CPO)
  = BR 7CCA ("" "" "" "")
  = BR 1048D2 6000 7CCA

Honeywell Confidential And Proprietary
4.7.2.10 **NB= Clear breakpoint**

The **NB** command clears breakpoints set by **BR** command, restoring user code.

It is possible to clear all the setted breakpoints, or only one breakpoint.

**Command use:**

```
NB [address]
```

[address] = Breakpoint address

- **NB 6000** (only this breakpoint is cleared)
- **BR 1048D2 7CCA**
- **NB** (all the breakpoints are cleared)
- **BR**
- **=**

4.7.2.11 **PR= Display system H/W configuration**

The **PR** command displays on console the H/W configuration of system at boot time. The following table is shown:

(Y = present resource, / = absent resource)

**Command use:**

```
PR
```

**The System configuration is (VMEbus Units are not solved):**

(P = present Unit, / = absent Unit, P = System Disk)

**MEMORY available:** 08 Mb

<table>
<thead>
<tr>
<th>CP01</th>
<th>CH00</th>
<th>SP00</th>
<th>SP01</th>
<th>SP02</th>
<th>LP03</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
</tbody>
</table>

**TAPE**

<table>
<thead>
<tr>
<th>TAPE</th>
<th>DISK0</th>
<th>DISK1</th>
<th>DISK2</th>
<th>DISK3</th>
<th>DISK4</th>
<th>DISK5</th>
<th>DISK6</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
</tbody>
</table>

Honeywell Confidential And Proprietary
4.7.2.12 \texttt{DR\ldots Display\_all\_PROCESSOR\_registers} \texttt}{

Using DR command, is possible to display on console all CPO or L/SPO registers, compressed into a table.

Command use: 

\begin{verbatim}
DR [pr. name]
\end{verbatim}

\begin{verbatim}
[pr. name] = Processor name
Default = CPO0
\end{verbatim}

\begin{verbatim}
= DR
PC=000068A4 SR=2510
D0=000004F9 D1=FFFFFFFE D2=00000000 D3=00007FFF
D4=00000000 D5=7FFFO000 D6=00007800 D7=000082C8
A0=00803A26 A1=0080E53A A2=00000000 A3=FFFFFFFE
A4=008F7FFA A5=00007FFF A6=00000000 A7=00007AB4
= DR
PC=00102AC2 SR=2300
D0=00000000 D1=FFFFFFFE D2=000A6400 D3=FFFFFFFE
D4=00347FF0 D5=7FFFO000 D6=00000000 D7=00054C0
A0=00344406 A1=0034055A A2=00000000 A3=00000000
A4=F000FFF A5=FFFFFFFE A6=F0FF092A A7=00102FB4
= 
\end{verbatim}

4.7.2.13 \texttt{DO\ldots Display/modify\_PROCESSOR\_data\_registers} \texttt}{

The command Dx displays or changes the value of CPO or L/SPO data registers Dx. If the value is changed, the register Dx will have the new value when the execution of Process code will be resumed, after the command “GO”.

Command use: 

\begin{verbatim}
D(value1) [pr. name] [value2]
\end{verbatim}

\begin{verbatim}
(value1) = Register number
[pr. name] = Processor name
Default = CPO0
[value2] = value to be assigned to register (Hex.)
\end{verbatim}

\begin{verbatim}
= D1
FFFFFFFE
= D1 55A8
= D1
00055A8
= 
\end{verbatim}
4.7.2.14 \texttt{AD\_\_A\_\_}\_Display/modify\_PROCESSOR\_addr\_registers

The command \texttt{AD\_\_A\_\_} displays or changes the value of CPO or L/SPO address register \texttt{Ax}. If the value is changed, the register \texttt{Ax} will have the new value when the execution of Process code will be resumed.

Command use: \texttt{A(value1) [pr. name] [value2]}

\texttt{(value1)} = Register number  
\texttt{[pr. name]} = Processor name  
\texttt{[value2]} = value to be assigned to register (Hex.)

\begin{verbatim}
  = A5 000054E2  
  = A5 00000000  
  = A0 SP2 00C03D02  
  = A0 SP2 C00000  
  = A0 SP2 00C00000  
\end{verbatim}

4.7.2.15 \texttt{SR\_\_Display/modify\_PROCESSOR\_status\_registers}

The command \texttt{SR} displays or changes the value of CPO or L/SPO status register. If the value is changed, the register \texttt{SR} will have the new value when the execution of Process code will be resumed.

Command use: \texttt{SR [pr. name] [value]}

Honeywell Confidential And Proprietary
The command `PC` displays or changes the value of CPO or L/SPO program counter. If the value is changed, giving the command "GO" the execution of Process code will resume starting from new PC value.

Command use: `PC [pr. name] [value]`

- `[pr. name]` = Processor name
  Default = CP00

- `[value]` = value to be assigned to register (Hex.)

- `PC 000078A2`
- `PC 00004000`
- `PC 00006000`
- `PC 00104FF6`
- `PC 00104000`

4.7.2.16 `PC=Display/modify_PROCESSOR_program_counter`

The command `PC` displays or changes the value of CPO or L/SPO program counter. If the value is changed, giving the command "GO" the execution of Process code will resume starting from new PC value.

Command use: `PC [pr. name] [value]`

- `[pr. name]` = Processor name
  Default = CP00

- `[value]` = value to be assigned to register (Hex.)

- `PC 000078A2`
- `PC 00004000`
- `PC 00104FF6`
- `PC 00104000`
4.7.2.17  MD = Display_memory_on_console

Using the command MD, it is possible to display on console the contents of addressed H/W locations, starting from the given address, for the given length.

Command use:

MD [pr. name] [value1] [value2]

[pr. name] = Processor name
  Default = CP00
[value1] = Start address of dump (Hex.)
  Default = 00000000
[value2] = Length of dump (Hex.)
  Default = 16

= MD
  00000000 00 34 05 60 00 34 7F F0 4E B9 00 34 43 F8 4E 71
= MD 4000
  00004000 AA 55 4E FF 65 A3 90 00 00 FF D4 BD DD E5 88 09
= MD 560000A8 90
  560000A8 CC 34 E5 67 CC DD A3 FF FF FF FF 68 C3 80 00 00
  560000B8 22 33 A2 DE 65 BC DD 00 00 00 00 00 00 00 00 00
  560000C8 4E 71 4E 71 BD C5 43 20 89 FF FF FF FF FF FF FF
  560000D8 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
  560000E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  560000F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  560000F9 A5 44 67 BD 00 DD 00 00 00 00 00 00 00 00 00 00
  56000108 4E 71 4E 71 4E 71 3A A9 44 50 00 00 00 00 00 00
  56000118 11 22 33 44 55 66 77 88 99 00 AA BB CC DD EE FF
  56000128 FF FF FF FF FF 00 00 00 00 FF FF FF FF FF FF
= MD SP0 104000 50
  00104000 CC 34 E5 67 CC DD A3 FF FF FF FF FF 68 C3 80 00 00
  00104010 22 33 A2 DE 65 BC DD 00 00 00 00 00 00 00 00 00 00
  00104020 4E 71 4E 71 8D C5 43 20 89 FF FF FF FF FF FF FF
  00104030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
  00104040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

4.7.2.18  MM = Memory_display/modify

The command MM displays on console the value corresponding at the given address, and gives the possibility to change it.

To change the value, the new hex. value must be entered from console, followed by a c/r. Enter c/r to read only.

To exit from MM command, the character ".” must be entered from console.
Command use: \[ \text{MM [pr. name] [value]} \]

[pr. name] = Processor name
Default = CPOO
[value] = Start address (Hex.)
Default = 00000000

= MM 6000
00006000 00?
00006001 11?
00006002 22?FF
00006003 33?00
00006004 44?FF
00006005 55?00
00006006 66?FF
00006007 77?00
00006008 88?.
= MD 6000
00006000 00 11 FF 00 FF 00 FF 00 88 99 AA BB CC DD EE FF
= MM SP1 105800
00105800 00?
00105801 11?
00105802 22?FF
00105803 33?.

4.7.2.19 EL_.Format_a_floppy

Using FT command, is possible to format a floppy. The format parameters are:

- 512 Kb sectors size
- 9 sectors per track
- Interleaving 1

Command use: \[ \text{FT} \]
4.7.2.20 LS = List_directory_of_DIAGX2_floppy

The command LS lists the contents of DIAGX2 directory sector, P_SECT. For each entry into the directory, representing a recorded Process, the following parameters are given:

- Process name
- start position into DIAGX2 (cylinder/head/sector)
- Process length (number of bytes)
- Process entry address

Command use:

= LS

**************************************************************************
#  
# file name  cyls head sect  leng  entry address  
#  
**************************************************************************
# CPOMON  0000  00  03  5000  000014D8  
# SPOMON  0002  01  02  2800  00102690  
# LPOMON  0003  00  04  3400  00103090  
# SG2CPX  0004  00  02  4800  00006000  
# SG2MEM  0005  01  09  4200  00006000  
# SG2DSK  0006  00  01  5600  00008684  
# SG2FLP  0008  01  04  5600  00006E32  
# SG2STR  0009  00  03  6600  0000606E  
# SG2CHO  0009  01  06  3800  000064F2  
# SG2SGD  000A  01  02  3800  00006800  
# SG2VMX  000B  00  07  6400  00006000  
# SG2TAP  000D  01  04  3200  00006500  
# SG2BDC  000E  01  09  4E00  00006000  
# SG2DFC  000F  00  04  6400  0000684A  
=
4.7.2.21  **PN**: List names of Diagnostic Processes

The command **PN** lists on console the names of Diagnostic Processes available in STAL system, for the current revision level. An indication of the tested area is also supplied.

Command use:  

```
= PN
Diagnostic Processes of System SGM2:

SG2CPX - Test of Unit CPO
SG2CHX - Test of Unit CHO
SG2SCX - Test of Unit SCO
SG2MEM - Test of Unit Memory
SG2SPX - Test of Unit SPO
SG2LPX - Test of Unit LPO
SG2DSK - Test of Unit Disk
SG2FLP - Test of Unit Diskette
SG2STR - Test of Unit Streamer
SG2VMX - Test of Units VMEbus
SG2TAP - Test for plugs of Printer/Line/Streamer
SG2BDC - Test tool for Disks/Floppy
SG2DFC - Test tool for DCO controller
```

4.7.2.22  **RL**: List revision levels

The command **RL** lists on console the Revision Level of DIAGX2 floppy and of EPROM code.

Command use:  

```
= RL
* The revision levels are:
DIAGX2  A-01
CP00    A-01     CP01  /
SP00    A-01     xP04  /
SP01    A-01     xP05  /
SP02    A-01     xP06  /
LP03    A-01     xP07  /
```

Honeywell Confidential And Proprietary
4.8 DIAGNOSTIC_PROCESSES_EXECUTION_MODES

It is possible to run STAL Diagnostic Processes in two different modes:

- automatic mode
- technical mode

4.8.1 AUTOMATIC_MODE

This is the Customer user level execution mode. Automatic mode is entered using lower-case Monitor commands (user commands).

Using automatic mode, no other inputs from console are required: all the execution parameters are automatically setted.

The selected Process is executed in recycle mode, until an error is found or a "break" is send from console.

If an error is found, an error message is displayed on console, and control returns to Monitor, that displays the system prompt "=". The error message is as follows:

xERR:xxx  CRU = UNIT yyy

where: xxx = error code
       yyy = CRU to be replaced

If no errors are found, at the end of each pass a PASS count message is displayed on console, until a "break" is received. After "break", the control returns to Monitor, that sends to console the User Selection Menu.

The pass count message is as follows:

SG2XXX: PASS YYY - ERRORS 0000

where: SG2XXX = Process name
       YYY = pass counter
The user selection menu is as follows:

```
SELECT:

a  Full system test
    (test executed both from diskette and disk)
b  Automatic system test
    (test performed from diskette only)
c  Test of Unit CP0
d  Test of Unit SCO
e  Test of Unit Memory
f  Test of Unit CHO
g  Test of Unit Disk
h  Test of Unit Diskette
i  Test of Unit Streamer
l  Test of Unit SPO
m  Test of Unit VMEbus
```
4.8.2 TECHNICAL_MODE

This mode is selected entering from console the Process name.

The following request is made by Monitor:

* Recycle on test? (Y/N) *

The answer defines the execution mode of Process (default, entering c/r, is no recycle).

Control is now given to Diagnostic Process, that displays on console the tests menu. In this mode, is possible to choose the type of test to be runned.

In technical mode, error reporting is more complete than automatic mode: a detailed description of happened error is supplied.

At the end of execution, control returns to Monitor, that displays User Selection Menu on console.
4.9 DIAGNOSTIC_PROCESS_OF_CPO_BOARD_(SG2CPX)

The Diagnostic Process SG2CPX tests the CPO board of SGM2 system.

The following areas are tested:

- 68020
  - H/W registers integrity
  - addressing modes
  - zero divide trap
  - illegal instruction trap
  - privilege violation trap
  - illegal address trap
  - user traps
  - exceptions nesting

- FPU
  - instructions execution
  - exceptions

- PIT
  - registers integrity
  - timers
  - interrupts to CPU

- MMU
  - "registers" integrity
  - read/write mem through descriptor
  - read/write violation error
  - undefined segment access error
  - disable segment error

- Memory addressing

- CACHE

- Bus interrupts to/from CPO
4.9.1 OPERATING RULES

An example of SG2CPX Process load and execution follows:

= SG2CPX
= Recycle on test? (Y/N)
= Y
SG2CPX: PASS 0001 - ERRORS 0000

The Process is loaded from DIAGX2 floppy into system memory, starting from address $6000, and executed in recycle mode until a "break" is send from console. At the end of each pass, the Pass/Error message is updated.

The tests executed in technical mode are the same as user mode. The only differences are in error reporting, more complete that user mode.

4.9.2 ERROR_REPORTING

If any error is found during Process execution, an error message is send to console, as follows:

*ERR:XYY text

where:

X = test number
YY = subtest number

In the following table, the possible error code are shown, with an indication of tested area and error type
<table>
<thead>
<tr>
<th>CODE</th>
<th>ORU</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Honeywell Confidential And Proprietary
4.10 DIAGNOSTIC_PROCESS_OF_DISK_(SG2DSK)

SG2DSK Diagnostic Process is able to detect failures in the following areas:

- DCO controller
- Disk devices

The SG2DSK Process hands bad sectors encountered during tests execution; if sector giving error is into bad block table, no error message is send.

Write tests destroy the media contents, but save bad block table.

Format tests generate a bad block table according to Operating System structure, and write the label "DIAG" at the start of 1st Disk sector (track 00, sector 00, head 00).

4.10.1 OPERATING_RULES

An example of SG2DSK Process load and execution follows:

```
SG2DSK
* Recycle on test ? (Y/N)
```

The Process is loaded from DIAGX2 floppy into system memory, starting from address $6000, and put in execution.
The Process menu is displayed on console:

** ** TESTS FOR CONTROLLER AND DISKS OF SGM2 ** **

The tests accept any configuration and type of device.

The DEFAULT procedure formats all present media, executes diagnostic test, writes and reads not sequentially all media.

To stop a test in execution press "BREAK".

DEVICE NUMBER

DISK0 = 0  DISK1 = 1  DISK2 = 2  DISK(0-2) = A

F = FORMAT AND CERTIFICATION (NO RETRY)
f = FORMAT (NO RETRY)
R = READ
Q = READ (NO RETRY)
r = READ NOT SEQUENTIAL
W = WRITE
w = WRITE WORST PATTERN
V = WRITE RANDOM PATTERN
E = DISPLAY ECA TABLE
Z = DEFAULT (LINK F,w,W,V,R,r)
A = DEFAULT WITHOUT FORMAT

The Process is now ready to accept the displayed commands.
4.10.2  \textbf{EOFORMAI-SELECIION}

Enter the selection corresponding to chosen disk. The following messages will be displayed on console:

\textasteriskcentered *MSG: 01  THIS TEST DESTROY THE MEDIA CONTENTS. DO YOU WANT TO CONTINUE (Y/N)

If N is entered, the test stops, else:

\textasteriskcentered *MSG: 03  START OF PREPARE ON DISK $x$

The selected disk is formatted at 512 sector size, and then bad sectors are searched using several read/write patterns. About 75 minutes are needed to complete a disk format.

The obtained Bad Block Table is according to Operating System structure and disk allocation.

4.10.3  \textbf{WRITE-SELECIION}

Enter the selection corresponding to chosen disk and pattern. The following messages will be displayed on console:

\textasteriskcentered *MSG: 01  THIS TEST DESTROY THE MEDIA CONTENTS. DO YOU WANT TO CONTINUE (Y/N)

If N is entered, the test stops, else:

\textasteriskcentered *MSG: 09  START OF WRITE DISK $x$

All the disk will be written using selected pattern. The bad sectors will be hand according to Bad Block Table present on the media.

At the end of test, or if the execution is interrupted with a "BREAK", the disk Bad Block Table is restored on the media.
4.10.4 DIAGNOSTIC_IESI_SELECTION

This selection doesn’t alter media contents, because Format/Write tests are performed onto diagnostic cylinder.

The following steps are always executed:

- Addressability test

  Different patterns are written into all read/write registers of DCO controller, and then read backwards, to check address logic.

The following steps are executed only if device is present, for all the present disks:

- Seek to last cylinder/Recalibrate

  The capability to reach last cylinder and after home track are checked.

- Error management tests

  - ID not found

    A read to a not existing sector is issued on the presently addressed device.

  - Bad block detect

    A format with all bad sectors is performed onto diagnostic track, and the detection of almost 1 bad sector is checked. The track is then correctly reformatted.

- ECC test

  An error free sector is searched onto diagnostic track, by means of write/read without retries. On this sector, the following tests are performed:
. Check of sector data and ECC field

The critical pattern $EB6DB6DB$ is written on the sector and then reread. Data and ECC are checked.

. Check of long mode operations

The critical pattern is written on the sector with a complemented ECC, and then reread, both in long mode. Data and ECC are checked.

. Check of bits correction capability

The capability to correct until 5 consecutives bits in data or ECC field is checked (both in data and ECC field). The "uncorrectable error" detection is also verified, writing a pattern with 6 bits error.
4.10.5 READ_SELECTION

If R/Q selections are used, the disk is read starting from 1st cylinder to the last; if r selection is used, the disk cylinders are read in the following order:

1st - last - 2nd - last-1 - 3rd - last-2 - ....

Enter the selection corresponding to chosen disk. The following messages will be displayed on console:

**MSG: OC START OF READ DISK x**

All the disk will be read. The bad sectors will be hand according to Bad Block Table present on the media.

4.10.6 DISPLAY_BAD_BLOCK_TABLE_SELECTION

Enter the selection corresponding to chosen disk. The following messages will be displayed on console:

```
****************************************************
LOG. SECTOR CYLINDER HEAD SECTOR

XXXXXX  CCCC  HH  SS
:
:
:
XXXXXX  CCCC  HH  SS
```

where:
XXXXXX = logic bad sector number
CCCC = cylinder \\nHH = head -> physic values
SS = sector /

Honeywell Confidential And Proprietary
4.10.7 DISPLAY_DCO_REGISIERS & ECA_TABLE_SELECTION

Using this selection, it is possible to look at DCO Controller registers and ECAtable contents.

Enter E1 the following messages will be displayed on console:

T.B.D.

4.10.8 CHANGE_EXECUTION_MODE_SELECTION

This selection changes the Process execution mode from recycle to non-recycle, or vice versa.

4.10.9 END_QE_IESI_SELECTION

If X selection is entered, SG20SK Process execution ends. The control returned to Monitor, that displays the User selection menu.
4.10.10 ERROR_REPORTING

If any error is found during test execution, one of following messages is send on console:

*ERR:X01 ERROR: recalibrate not executed
*ERR:X02 ERROR: format not executed
*ERR:X05 ERROR IN SELF-TEST. THE TEST STOPS
*ERR:X06 WRONG ADDRESSING OF DCO REGISTERS. THE TEST STOPS
*ERR:X07 WRONG ACCESS TO DCO REGISTERS. THE TEST STOPS
*ERR:X08 CONTROLLER FAULT. THE TEST STOPS
*ERR:X0B ERROR DURING OPERATION ON DEVICE
*ERR:X0E ERROR DURING FORMAT
*ERR:X0F CONTROLLER FAULT: Bad Block Mark not found
*ERR:X10 DCO controller doesn't found a sector on reformatted track
*ERR:X11 CONTROLLER FAULT: wrong ECC computation
*ERR:X12 CONTROLLER FAULT: unable to perform long mode operations
*ERR:X13 CONTROLLER FAULT: ECC logics
*ERR:X15 CONTROLLER FAULT: not received "ID not found"
*ERR:X18 IMPOSSIBLE TO READ BAD BLOCK TABLE
*WAR:X06 BBT IS FULL! THE TEST STOPS
*ERR:X1A FOUND A BAD BLOCK NOT PRESENT IN BAD BLOCK TABLE
*ERR:X1C INTERRUPT TIMEOUT: CONTROLLER FAULT
*ERR:X1E UNEXPECTED ERROR DURING READ
*ERR:X1F UNIDENTIFIED DEVICE
*ERR:X20 CONTROLLER OR DEVICE FAULT

where: X = test number
4.11 DIAGNOSTIC_PROCESS Điện FLOPPY_(SG2FLP)

SG2FLP Diagnostic Process is able to detect failures in the following areas:

- DCO controller
- Floppy device

The SG2FLP Process works on media having 9 sectors per track, and 512 bytes sector size.

The copy utilities ("b" and "v" selections) need an already formatted media to perform the copy.

"B" selection ("complete test on DIAGX2") performs the same steps as automatic user selection "f". They are:

- format of free DIAGX2 cylinders
- write of worst pattern onto formatted tracks
- read of written pattern, and check of data
- read not sequential of all the media

Write tests destroy all the media contents, except for "B" selection, working onto DIAGX2 floppy free cylinders.

4.11.1 OPERATING_RULES

An example of SG2FLP Process load and execution follows:

```
$ SG2FLP
* Recycle on test? (Y/N)
```

The Process is loaded from DIAGX2 floppy into system memory, starting from address $6000, and put in execution.

The Process menu is displayed on console:
** TEST FOR CONTROLLER AND FLOPPY OF SGM2 **

The DEFAULT procedure formats the present media, reads the formatting pattern, executes the diagnostic test, writes and reads not sequentially all media, checking the data. To stop a test in execution press "BREAK".

Available functionalities:

- F = FORMAT AND CERTIFICATION
- f = FORMAT
- R = READ
- r = READ NOT SEQUENTIAL
- W = WRITE (EB6DB6DB,55AA55AA,E86DB6DB)
- w = WRITE WORST PATTERN
- V = WRITE RANDOM PATTERN
- E = DISPLAY ECA TABLE
- Z = DEFAULT (LINK F,w,W,V,R,r)
- A = DEFAULT WITHOUT FORMAT

The Process is now ready to accept the displayed commands.
4.11.2 FORMAT SELECTION

Mount a scratch floppy, and enter the selection. The following messages will be displayed on console:

*MSG* 01          \///  FORMAT FLOPPY  \///

The floppy is formatted with the following structure:

- 96 tpi
- 512 b sector size
- 9 sect./track (1-9)
- interleaving 1

4.11.3 WRITE SELECTION

Mount a scratch floppy, and enter the selection corresponding to choosed pattern. The following messages will be displayed on console:

*MSG* 04          \///  FLOPPY WRITE  \///

All the floppy will be written using selected pattern.
4.11.4 DIAGNOSTIC TEST SELECTION

This selection doesn’t alter floppy contents.

The following steps are executed:

- Addressability test

  Different patterns are written into all read/write registers of DCO controller, and then read backwards, to check address logic

- Seek to last cylinder/Recalibrate

  The capability to reach last cylinder and after home track are checked

- Error management tests

  . ID not found

    A read to a not existing sector is issued on the presently addressed device
4.11.5  IESI.ON_DIAOX2_SELECTION

This selection performs the same tests as the automatic User menu "f" selection. They are:

- format of free DIAGX2 cylinders
- write of worst pattern onto formatted tracks
- read of written pattern, and check of data
- read not sequential of all the media

The first 3 steps are performed only if the floppy is not in WRITE PROTECT condition.

4.11.6  READ_SELECTION

If "R" selection is used, the floppy is read starting from 1st cylinder to the last; if "r" selection is used, the floppy cylinders are read in the following order:

1st - last - 2nd - last-1 - 3rd - last-2 - ....

Enter the selection corresponding to choosed read type. The following messages will be displayed on console:

MSG: 05     ///  FLOPPY READ     ///

All the floppy will be read, without checking read data.

4.11.7  DISPLAY_DCO_REGISTERS_&_ECA_TABLE_SELECTION

Using E selection, is possible to look at DCO Controller registers and ECA table contents.

Enter E; the following messages will be displayed on console:

---

Honeywell Confidential And Proprietary
4.11.8 COPY_SELECTION

Mount the floppy to be copied, and enter selection corresponding to chosen copy type. The following messages will be displayed on console:

*MSG: 09 Read in execution

The floppy inserted into device will be read into memory. At the end of floppy read, the request to mount an already formatted floppy is made.

*MSG: 0A Take off floppy and insert the new one. Press "RETURN" to start the copy

Mount the new floppy and press "RETURN" key. The following messages will appear on console:

*MSG: 08 Write in execution

The data read from previous floppy will be transfered onto the new one. If "v" selection has been entered, at the end of write also following message will be send on console:

*MSG OD Verify in execution

It is possible to make more than one copy of the same floppy. At the end of write (and verify, if any), the following request is made:

*MSG OC Do you want another copy of this media (y/n)?

If you want, take off the already copied floppy, mount a new one and enter "y"; a new copy will be made.

4.11.9 CHANGE_EXECUTION_MODE_SELECTION

This selection changes the Process execution mode from recycle to non-recycle, or vice versa.
4.11.10 END_OF_TEST_SELECTION

If X selection is entered, SG2FLP Process execution ends. If a scratch floppy has been mounted, remount DIAGX2 and enter "X": the control returns to Monitor, that displays the User selection menu.

4.11.11 ERROR_REPORTING

If any error is found during test execution, one of the following messages is send on console:

*ERR:X01 ERROR: missed recalibrate
*ERR:X02 ERROR: missed format
*ERR:X03 ERROR IN SELF-TESTS. THE TEST STOPS
*ERR:X04 WRONG DCO REGISTERS ADDRESSING. THE TEST STOPS
*ERR:X06 CONTROLLER FAULT. THE TEST STOPS
*ERR:X09 ERROR DURING PHYSICAL OPERATION ON DEVICE
*ERR:X0B CONTROLLER FAULT: missed ID not found
*ERR:X0E Write protected floppy
*ERR:X0F Error during write operations
*ERR:X10 Error during read operations
*ERR:X11 INTERRUPT TIMEOUT: CONTROLLER FAULT
*ERR:X13 Read data different from written data
*ERR:X14 Data compare error

where: X = test number
4.12 DIAGNOSTIC_PROCESS_OF_MEMORY_(SG2MEM)

This Diagnostic Process tests SGM2 memory boards, up to 24 Mb, by means of followings tests:

- Memory addressability, writing address as data
- Memory integrity, using Hartman-Knaizuk algorithm

4.12.1 OPERATING_RULES

An example of SG2MEM Process load and execution follows:

```
= SG2MEM
 Recycle on test? (Y/N)
 = Y
 SG2MEM: PASS 0001 - ERRORS 0000
```

The Process is loaded from DIAGX2 floppy into system memory, starting from address $6000, and executed in recycle mode until a “break” is send from console. At the end of each pass, the Pass/Error message is updated.

The tests executed in technical mode are the same as user mode. The only differences are in error reporting, more complete that user mode.

4.12.2 ERROR_REPORTING

Error reporting of SG2MEM Process is at row-on-board position level, that is position on memory board of fault row is sived in the error message.

The row-on-board message is as in the follows table:
4.13  DIAGNOSTIC_PROCESS_OF_STREAMER(SG2STR)

The Streamer Diagnostic Process SG2STR tests the STREAMER device of SGM2 system.

Write tests destroy tape contents.

4.13.1  OPERATING RULES

An example of SG2STR Process load and execution follows:

```
SG2STR
Recycle on test? (Y/N)
```

The Process is loaded from DIAGX2 floppy into system memory, starting from address $6000, and put in execution.

The Process menu is displayed on console:

**DIAGNOSTIC FOR STREAMER UNIT**

**TEST TYPE SELECTION:**

- A = WRITE, FILE MARK, WRITE, READ, READ STATUS
- B = WRITE DATA, READ DATA
- C = BOT, WRITE, READ STATUS,
  READ DATA, READ STATUS
- D = RETENTION, BOT, READ CARTRIDGE
  WITH STATUS PRINT
- X = END OF TEST

**TEST SELECTION:**

The Process is now ready to accept the displayed commands. The Process menu is shown everytime a "break" is send from console.
4.13.2 A__SELECCION

The following steps are executed:
- write of 256 blocks of "A1A2A3A4"
- write of a file mark
- write of 384 blocks of "12345678"
- read of all the written blocks, with check of data
- read of device status

At the end of test execution, the menu is shown on console.

4.13.3 B__SELECCION

The following steps are executed:
- write of 16 blocks of "12345678"
- read of all the written blocks, with check of data

At the end of test execution, the menu is shown on console.

4.13.4 C__SELECCION

The following steps are executed:
- rewind until BOT
- write of 256 blocks of walking 0 pattern (starting from "FE" to "7F"
- read of device status
- read of all the written blocks, with check of data
- read of device status

At the end of test execution, the menu is shown on console.
4.13.5 D SELECTION

The following steps are executed:
- tape retention
- rewind until BOT
- read of all the cartridge, until "no data" bit is set into device status bytes, or EOT is reached
- read and print of device status

At the end of test execution, the menu is shown on console.

4.13.6 X SELECTION

If X selection is entered, SG2STR Process execution ends. A reset command is send to device, and after the control returns to Monitor, that displays the User selection menu.

4.13.7 ERROR REPORTING

If any error is found during Process execution, an error message is send to console, follows by the print of device Status bits. The message may be one of the following:

*ERR: INTERRUPT NOT RECEIVED
*ERR: RECEIVED INTERRUPT NOT CORRECT
*ERR: DATA COMPARE ERROR
*ERR: NUMBER OF READ BLOCKS # NUMBER OF WRITTEN BLOCKS
*ERR: CARTRIDGE ABSENT
*ERR: ERROR RESETING RDY F/F
*ERR: ERROR RESETING EXC F/F
*ERR: ERROR IN SET-RESET DIRECTION BIT
*ERR: ERROR IN SET-RESET ACKNOLEDGE BIT

Honeywell Confidential And Proprietary
4.14 DIAGNOSTIC_PROCESS_OF_CACHE_(SG2CHX)

SG2CHX Process is able to detect failures into CHO board of SGM2 system, by means of following tests:

TBW

4.14.1 OPERATING_RULES

An example of SG2CHX Process load and execution follows:

= SG2CHX
* Recycle on test? (Y/N)
  Y

SG2CHX: PASS 0001 - ERRORS 0000

The Process is loaded from DIAGX2 floppy into system memory, starting from address $6000, and executed in recycle mode until a "break" is sent from console. At the end of each pass, the Pass/Error message is updated.

The tests executed in technical mode are the same as user mode. The only differences are in error reporting, more complete than that user mode.

4.14.2 ERROR_REPORTING

If any error is found during Process execution, an error message is sent to console, as follows:

*ERRORXXX CRU = UNIT CHO
  text

where:
  XXX = error code
  text = one of following messages

Honeywell Confidential And Proprietary
4.15 DIAGNOSTIC_PROCESS_OE_SCO_(SG2SCX)

SG2SCX Diagnostic Process is able to detect failures into System Controller board (SCO), by means of following tests:

TBW

4.15.1 OPERATING_RULES

An example of SG2SCX Process load and execution follows:

= SG2SCX
* Recycle on test? (Y/N)
= Y
SG2SCX: PASS 0001 - ERRORS 0000

The Process is loaded from DIAGX2 floppy into system memory, starting from address $6000, and executed in recycle mode until a "break" is send from console. At the end of each pass, the Pass/Error message is updated.

The tests executed in technical mode are the same as user mode. The only differences are in error reporting, more complete that user mode.

4.15.2 ERROR_REPORTING

If any error is found during Process execution, an error message is send to console, as follows:

*ERRiXYY  text

where:

X = test number
YY = subtest number

In the following table, the possible error code are shown, with an indication of tested area and error type:
<table>
<thead>
<tr>
<th>CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>

Text may be as follows:
4.16 DIAGNOSTIC_PROCESS_OF_VMEbus_CONTROLLERS_(SG2VMX)

This Diagnostic Process permits to run tests on Controller boards connected to SGM2 system via VMEbus.

The Controller boards up to now supported are:

- 
- 

4.16.1 OPERATING_RULES

An example of SG2VMX Process load and execution follows:

```
= SG2VMX
= Recycle on test ? (Y/N)
```

The Process is loaded from DIAGX2 floppy into system memory, starting from address $6000, and put in execution.

The Process menu is displayed on console:

SELECT:
A    Test of Unit ...
B    Test of Unit ...
X    Test of all the Units
K    End of test

The tests executed in technical mode are the same as user mode. The only differences are in error reporting, more complete that user mode.
4.16.2 **['A'-SELECTION]**

The ... test performs the following operations:

- verify of Controller self tests results

4.16.3 **['B'-SELECTION]**

The ... test performs the following operations:

- verify of Controller self tests results

4.16.4 **IESTI_OF_ALL_UTE_UNITS,['X'-SELECTION]**

If X selection is entered, all the supported Controller tests are sequentially executed, in the Process menu order.

4.16.5 **END_OF_IESTI,['K'-SELECTION]**

If K selection is entered, SG2VMX Process execution ends. The control returned to Monitor, that displays the User selection menu.

Honeywell Confidential And Proprietary
4.16.6  ERROR_REPORTING

4.16.6.1  IESI

If any error is found during Process execution, an error message is send to console, as follows:

```text
#ERR:XYY CRU = UNIT ...
```

where:

- **X** = test number
- **YY** = subtest number
- **text** = one of following messages:

Honeywell Confidential And Proprietary
4.16.6.2 **IESI**

If any error is found during process execution, an error message is sent to console, as follows:

```plaintext
*ERR:XYY  CRU = UNIT ...  text
```

where:
- \( X \) = test number
- \( YY \) = subtest number
- text = one of following messages:

---

Honeywell Confidential And Proprietary
4.17 PLUGS_TEST_TOOL(SG2TAP)

SG2TAP Diagnostic Process permits to test connector interface between SGM2 system and terminals, printer, streamer without connecting physically the devices, but inserting a plug into logic board connector.

4.17.1 OPERATING_RULES

An example of SG2TAP Process load and execution follows:

* SG2TAP
* Recycle on test? (Y/N)
* Y

The Process is loaded from DIAGX2 floppy into system memory, starting from address $6000. The selected test will be executed in recycle mode until a "break" will be sent from console.

The Process menu is displayed on console, as follows:

* TESTS FOR PRINTER-STREAMER-LINE PLUGS OF SGM2 *

To stop a test in execution press "BREAK"

T.B.D.

Note: Is not possible to test the Monitor console line.
4.17.2 ERROR REPORTING

If any error is found during test execution, the following message is sent on console:

```c
\#ERR:OXX  CRU = UNIT YYY
```

where:

- **OOX** = error code
- **YYY** = Unit in error

The possible error codes are as follows:
4.18 DISK_FLOPPY_CONTROLLER_IESI_TOOL_(SG2DFC)

SG2DFC Diagnostic Process performs acceptance tests on DCO disk-floppy Controller.

4.18.1 OPERATINGRULES

An example of SG2DFC Process load and execution follows:

```plaintext
SG2DFC
Recycle on test? (Y/N)
```

The Process is loaded from DIAGX2 floppy into system memory, starting from address $6000, and put in execution.

The Process menu is displayed on console, as follows:

```
T.B.D.
```

The Process is now ready to accept the displayed commands.

4.18.2 DIAGNOSTIC_IESI_SELECTION

T.B.D

4.18.3 ACCEPTANCE_IESI_SELECTION

T.B.D
4.18.4 FLOPPY_EMULATION_SELECTION

T.B.D

4.18.5 DISPLAY_DCO_REGISIERS__ECA_TABLE_SELECTION

T.B.D

4.18.6 CHANGE_EXECUTION_MODE_SELECTION

This selection changes the Process execution mode from recycle to non-recycle, or vice versa.

4.18.7 END_DO_IESI_SELECTION

If K selection is entered, SG2DFC Process execution ends. The control returned to Monitor, that displays the User selection menu.
4.18.8 ERROR REPORTING

If any error is found during test execution, one of the following messages is sent on console:

4.18.8.1 ERROR

[Message content]

4.18.8.2 WARN

[Message content]

4.18.8.3 CRITICAL

[Message content]
4.19  **BASIC DEVICE CONTROLLER TEST TOOL (SG2BDC)**

SG2BDC Diagnostic Process is able to perform single command operations onto disks and floppy devices.

4.19.1  **OPERATING RULES**

An example of SG2BDC Process load and execution follows:

```plaintext
  = SG2BDC
  * Recycle on test? (Y/N)
  =
```

The Process is loaded from DIAGX2 floppy into system memory, starting from address $6000$, and put in execution. The Process menu is displayed on console, as follows:

<table>
<thead>
<tr>
<th>DEVICE TYPE</th>
<th>DISKO 0</th>
<th>DISK1 1</th>
<th>DISK2 2</th>
<th>FLOPPY 3</th>
</tr>
</thead>
</table>

**AVAILABLE OPTIONS**

- D = display wr buffer
- d = display rd buffer
- R = recycle
- M = read in wr buffer
- m = memory modify
- X = end of test

**AVAILABLE COMMANDS**

- R = diagnostic read
- r = normal read
- Z = read without retry
- W = diagnostic write
- w = normal write
- F = format
- S = seek
- A = recalibrate

**READ PARAMETERS:**

```
CCCCHHSS
```

where:

- CCC  = cylinder
- HH   = head
- SS   = sector
- AAAAAAAAA = address

**WRITE PARAMETERS:**

```
CCCCHHSSPPPPPPP
```

**DIAG. WRITE PARAM.:**

```
CCCCHHSS
```

**FORMAT PARAMETERS:**

```
CCCCHH
```

**SEEK PARAMETERS:**

```
CCC
```

**MEMORY MODIFY PAR.:**

```
AAAAAAA
```

**PARAMETERS:**

This menu is also displayed everytime a "break" is send from console.
At this point, it is possible to give the desired command, using a structure as follows:

PARAMETERS: DCPO

where:
- D = DEVICE TYPE
- C = COMMAND
- P = PARAMETERS
- O = OPTIONS

The process will make the following question:

EXECUTION? (Y/N):

If Y is entered, the given command will be executed, else the process menu will be reshow on console.

4.19.2 ERROR_REPORTING

If any error is found during test execution, the following message will be displayed on console:

T.B.D.

- INTERRUPT TIMEOUT: CONTROLLER FAULT
- FLOPPY ABSENT
- MISSED RECALIBRATE
- FLOPPY WRITE-PROTECTED
- READ BUFFER NOT EQUAL TO WRITE BUFFER
5. LEVEL_3 (DIAG)

5.1 INTRODUCTION

DIAG is a test under O.S. designed to exercise central H/W functionalities, peripherals and UNIX kernels, as a user application, but more exactly and controlled.

DIAG can work at system level and/or subsystem level, changing number and types of tests selected by user.

All the informations needed by test (also ones concerning devices operability) are saved or as data into support files, or automatically researched into system s/w areas usable by user programs.

DIAG goals are to assure the affidability of all the system, in user work conditions, by means of automatic procedures (when possible) or manual procedures.
5.2 INTEGRATION_INIQ_OS_FILE_SYSTEM

A directories organization as follows is supplied for an integration more independent as possible by O.S. file system structure:

```
/root directory

bin etc usr
(SYSTEMS DIRS)

dia1 user1 userN
(USERS DIRS)

(DIAG WORKING DIR)

dia9_empty dia9_env dia9_f dia9_tmp
(DIAG DIRS)
```

Using this structure is possible to solve testing problems in I/O block area regarding disk space availability and file system visibility.

Honeywell Confidential And Proprietary
5.2.1 DIAG_FILE_SYSTEM

The following files are present into DIAG file system:

. /usr/dias

  .. DIAG executable modules:
     ... ssstst
     ... sscomm
     ... sscp
     ... ssfs
     ... ssio
     ... ssprt
     ... sgser
     ... ssys
     ... sstimer
     ... ssttyr
     ... ssttyw

  .. diagnostic shell-procedure (DIAG):

. /usr/dias/dias_c

  .. DIAG source modules:
     ... buffcmp.c
     ... buffsen.c
     ... dupbuf.c
     ... mkfile.c
     ... rb0.c
     ... rr0.c
     ... rr1.c
     ... sscomm.c
     ... sscp.c
     ... ssfs.c
     ... ssio.c
     ... sgser.c
     ... ssys.c
     ... sstimer.c
     ... ssttyr.c
     ... ssttyw.c
     ... wb0.c
     ... wr0.c
     ... wr1.c

  .. DIAG compiled modules:

  .. errfile (DIAG error messages files):

  .. DIAG makefile:
. /usr/dias/dias_env

.. all the script files describing current diagnostic running environments of DIAG.

.. Up to now the following script files are present:

- SGCMDAUTO  test "automatic"
- SGCMDCPU   test "cpu and sys"
- SGCMDDKT   test "floppy"
- SGCMDDSK   test "disk"
- SGCMDDFS   test "file system"
- SGCMDLP    test "line printer"
- SGCMDTTYWO-3 test "tty output"
- SGCMDTTYRO-3 test "tty input (keyboard)"
- SGCMDCOMO-3 test "communication"

.. A list of available script files follows:

# dias_env/sgcmdauto

# default script for automatic system test
# selection "a" of DIAG
#
1 20 sscpuc 5000
1 20 sssys 5000
1 20 ssiob s dias_tmp/tmp000 wr 10000 aaddee11
1 20 ssiob s dias_tmp/tmp001 wr 10000 |
1 20 ssloc s /dev/rfd wr 20 11223344
1 20 ssfs /usr/bin
#1 20 ssprt /dev/lp dias_f/lin32
1 20 ssttyw /dev/tty0 dias_f/sswinput
1 20 ssttyw /dev/tty1 dias_f/sswinput
1 20 ssttyw /dev/tty2 dias_f/sswinput
```bash
# diag_env/sscmdddkt
# default script for diskette test
# selection "d" of DIAG
#
# 1 20 ssioc s /dev/rfd wr 300 11223344

# diag_env/sscmdddsk
# default script for disk test
# selection "c" of DIAG
#
# 1 20 ssiob s diag_tmp/tmp000 wr 1000 aaddee11
# 1 20 ssiob s diag_tmp/tmp001 wr 2000 i
# 1 20 ssiob s diag/tmp002 wr 3000 r
# 1 20 ssiob s diag_tmp/tmp003 r 20000 i
# 1 20 ssiob s diag_tmp/tmp004 w 12000 0011eeff

# diag_env/sscmdcpu
# default script for test of cpu & sys
# selection "b" of DIAG
#
# 1 20 ssCPU 1000
# 1 20 ssCPU 10K
# 1 20 ssSYS 1000
# 1 20 ssSYS 100K
```
Honeywell Confidential And Proprietary

```
# diag_env/sscmdlpr
# default script for test of printer 132cqi
# selection "e" of DIAG
#
1 20 ssprt /dev/lp diag_f/132a

# diag_env/sscmdfs
# default script for file system test
# selection "f" of DIAG
#
1 20 ssfs /usr/diag
1 20 ssfs /

# diag_env/sscmdcom3
# default script for test of line 3 in input mode
# selection "h" of DIAG
# test receive data from another computer on line 3
#
1 20 ssconn /dev/tty3
```
Honeywell Confidential And Proprietary
# diag_env/sscmdtty3

# default script for test of line 3 in output mode
#
# selection "s" of DIAG
#
# the printed file sswinput is a default file usable for
# all the tty types. Other pattern are available under
# the directory /usr/dias/dias_f
#
# 1 20 ssttyw /dev/tty3 dias_f/sswinput

# diag_env/sscmdtty2

# default script for test of line 2 in output mode
#
# selection "s" of DIAG
#
# the printed file sswinput is a default file usable for
# all the tty types. Other pattern are available under
# the directory /usr/dias/dias_f
#
# 1 20 ssttyw /dev/tty2 dias_f/sswinput

# diag_env/sscmdtty1

# default script for test of line 1 in output mode
#
# selection "s" of DIAG
#
# the printed file sswinput is a default file usable for
# all the tty types. Other pattern are available under
# the directory /usr/dias/dias_f
#
# 1 20 ssttyw /dev/tty1 dias_f/sswinput
Honeywell Confidential And Proprietary

```bash
# diag_env/sscmdtty0

# default script for test of line 0 in output mode
#
# selection "9" of DIAG
#
# the printed file sswinput is a default file usable for
# all the tty types. Other pattern are available under
# the directory /usr/dia9/dia9_f
#
# 1 20 ssttyw /dev/tty0 dia9_f/sswinput

./usr/dia9/dia9_f

... all the files used as diagnostic patterns by DIAG
into tests for printer and work-station.

./usr/dia9/dia9 tmp

... work directory for tests: sgoob sgfs ....

./usr/dia9/dia9 empty

... directory devoted to file systems mount.
```
5.3 FUNCTIONALITIES

DIAG is composed by following base modules:

- **MONITOR**: father of all the processes and tests coordinator
- **TESTS**: the goals of these processes are to check hardware and software of SOM2
- **SERVICES**: are the processes collecting external monitor requests
- **UTILITY**: is a utility process, called SGSYNTAX, able to execute the syntax analysis of the command file that is to be supplied at DIAG running time

**WARNING**:

As a user depending responsibility, the following may occur:

- "swapper" system process may start if there are a great number of concurrent processes, or for user memory needs.
- unpredictable system work (processes never executed, continuous swapping, ...) if there are a great number of processes, or in case of uncorrect tests priority use.
5.3.1 MONITOR

It executes the following steps:

- check of parameters supplied by Shell (number of parameters, test time value, command file existence).

- read from command file of test to be run structure. One or more processes (as indicated by "nproc" parameter) of selected type (as indicated by "testtype" parameter), are started for each line of command file. The typical test parameters are then supplied to these processes.

- start of TIMER process;

- start of SERVICE process;

- self wait ("wait(2)" for one of following events:

  a) reach of test time parameter, (exit(2) of SGTIMER). The following occur:

    - kill of all the running processes
    - send to console of the message:

      "SGTST END TEST"

    - exit of father process and return to Shell.

  b) termination of SGSERVICE process, (exit(2) of SGSERVICE), due to the send of "STOP" command from console. The following occurs:

    - kill of all the running processes
    - send to console of the message:

      "SGTST END TEST"

    - exit of father process and return to Shell.
c) termination of a test process, \texttt{(exit(2) of SGXXX)}), caused by an error detection.

- if others test processes are running, the counter of end processes is updated and a wait for an event occur.
- if there are no running processes \texttt{(generated processes counter = end processes counter)}, then:
  - kill of all the running processes
  - send to console of the message:
    \texttt{"SGTST END TEST"}
- exit of father process and return to Shell.

\textbf{Limitations:}
- tests generation and execution is not synchronized.
- error management at \texttt{signal()} level is not hand in a complete mode.
5.3.2 IESIS

5.3.2.1 SGCPU

Goal:
- Doesn't make any specific request to kernel.
- It is as an application that executes only internal count at user program level.

Command line format:

NPROC PRIORITY SGCPU MEMSIZE

where:

SGCPU

is cpu test identifier in user mode.

MEMSIZE

is the size of memory that the process requires to alloc in add to own reserved data area.

The process is structured as follows:

1) Dynamic request to kernel to alloc memory via malloc (3)

2) Issue of an error message and process termination if errors during malloc(3)

3) Execution of floating-point arithmetics operations in simple and double precision

4) Write of a pattern into all the allocated area

5) Read and verify of all the allocated area contents.
If any error is found, the process end giving an error message, else recycles starting from point 3.

LIMITATIONS:

- MEMSIZE parameter must be <=65535
5.3.2.2 SGSYS

Goal:
- Makes requests to kernel not requiring any I/O type
- It is as an application floating from user level to kernel level, that stops at PROCESS S/S level, and so executes internal count in system mode.

Command line format:
NPROC PRIORITY SGSYS MEMSIZE

where:
SGSYS is the test identifier
MEMSIZE is the size of memory that the process requires to alloc

The process is structured as follows:

1) Dynamic request to kernel to alloc memory via malloc (3)
2) Issue of an error message and process termination if errors during malloc(3)
3) Request to read system clock. If any error is found, an error message is send and the process ends
4) Write of system clock value into the allocated memory area positions, and recycle starting from point 3.

NOTE:
- stressing of context switching mechanism due to system calls recurrent use

LIMITATIONS:
- MEMSIZE parameter must be <=65535

Honeywell Confidential And Proprietary
5.3.2.3 SGIQB

Goal:
- Perform I/O requests on the specified blocks device,
  using standard I/O buffered structures for the device.
- It is an application that requires standard I/O from
disks, floppies or tapes.

Command line format:
NPROC PRIORITY SGIQB DEVICE R/W/WR DEVSIZE PATTERN

where:
NPROC must be 1
SGIQB is the test identifier
DEVICE is the device pathname
R  read
W  write
WR write plus read with data check
DEVSIZE dimension of the file to be created (number of bytes)
PATTERN may be:  I incremental
    R random
    xxxxxxxxx (hex. string)

- read
1) generation of a file having DEVSIZE dimensions,
   with PATTERN write
2) file contents read in recycle mode, with data check.
   If any error is found, an error message is
   send and process ends
- write

1) write of PATTERN in recycle mode until DEVSIZE. If any error is found, an error message is send and process ends

- write..read

1) write until DEVSIZE

2) read and file contents check. If any error is found, an error message is send and process end

3) repeat from point 1.

ALGORITHM:

The algorithm used by this process try to stress the I/O s/s part devoted to queue I/O requests. The worst parameter is the length of buffer to be transferred.

The algorithm is as follows:

```
BSIZE
\  
BSIZE + i until DEVSIZE
/   
___
i=0
```

NOTES:

- If DEVSIZE is less than BSIZE, the transfer is made for a length equal to DEVSIZE.

LIMITATIONS:

- Only one process is runned for every defined file.
5.3.2.4 SGIQC

Goal:

- Performs I/O requests on specified block device without using standard I/O structures. These types of operations are called "physical I/O's" or "unbuffered mode".

- It is as an application that must hand I/O on block devices synchronously with requests, not in asynchronous and deferred mode as standard I/O.

Command line format:

NPROC PRIORITY SGIQC DEVICE R/W/WR DEVSIZE MIN MAX

where:

NPROC must be 1
SGIQC is test identifier
DEVICE is device pathname
R/W/WR read/write/write plus read
DEVSIZE dimension of the file to be created (number of bytes)

PATTERN may be:  - I incremental
- R random
- xxxxxxxxx (hex. string)

The process has the same SGIOB structure, except the I/O algorithm that is as follows:

DEVSIZE
\  
BSIZE
\___
i=BSIZE

LIMITATIONS:

- Only one process is runned for every defined file.
- Usable only onto floppy and disk test partition.
5.3.2.5 SGIIYW

Goal:

- Performs OUTPUT requests on character type devices, as work stations.
- It is as a user application that leaves to kernel all control characters handling for given device, and so hands only I/O strings.

Command line format:

```
NPROC PRIORITY SGTTYW DEVICE FILE NAME
```

where:

<table>
<thead>
<tr>
<th>NPROC</th>
<th>must be 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGTTYW</td>
<td>test identifier</td>
</tr>
<tr>
<td>DEVICE</td>
<td>device pathname (must be different from console)</td>
</tr>
<tr>
<td>FILE NAME</td>
<td>pathname of the file to be shown</td>
</tr>
</tbody>
</table>

The process is structured as follows:

1) file read into memory
2) read of the file /ETC/TERMCAP, containing the control characters for the terminal under test.
3) send of file contents to the device in recycle mode.

If any error is found, an error message is send and process ends.

**NOIE**

The test will be able to modify the line parameters (BAUD RATE, N.STOP BITS, PARITY, ECHO, ....) of the terminal under test, via manual procedures to be executed on the terminal.
5.3.2.6 SGIIYR

Goal:
- To test character type devices in INPUT mode.
- In this case, the user application has a complete visibility of the device, and so it must hand both I/O strings and device control characters. Typical example is the keyboard function keys.

Command line format:

```
NPROC PRIORITY SGTTYR DEVICE
```

where:

- **NPROC** must be 1
- **SGTTYR** test identifier
- **DEVICE** device pathname (must be different from console)

The process is structured as follows:

1) The process guides operator to introduce all the keyboard characters set, and check the results.
2) If any error is found, an error message is send and process ends

NOTE:

To execute the test on a specified terminal, must run the test starting from another terminal.
5.3.2.7 SGERI

Goal:

- It is as SGTYYB for the printer.

- It is as a user application that leaves to kernel all control characters handling for the printer, and so hands only output strings.

Command line format:

```
NPROC PRIORITY" SGPR T DEVICE FILENAME
```

where:

- NPROC must be 1
- SGPR T test identifier
- DEVICE device pathname
- FILE NAME pathname of the file to be printed

The process is structured as follows:

1) read of the file to be printed.

2) send of file contents to the device in recycle mode.

If any error is found, an error message is send and process ends.
5.3.2.8 SGES

Goal:

- It performs file system access requests, walking through subtrees and starting from a given directory.
  In this mode a great number of requests are made on the device having used file system.

- The kernel part (FILE S/S) that handles updating and requests on regular files and directories is exercised.

The command line is structured as follows:

```
NPROC PRIORITY SGFS DIRECTORY
```

where:

```
DIRECTORY    starting test directory
```

The test, starting from given directory, search all the files under the directory.
5.3.2.9 SGCOMM

**Goal:**

- To test serial lines in INPUT mode.
- A common example is the test of data transmission computer to computer.

**Command line format:**

```
NPROC PRIORITY SGCOMM DEVICE
```

**where:**

- **NPROC** must be 1
- **SGCOMM** test identifier
- **DEVICE** device pathname

**The process is structured as follows:**

1) Read of any input character from selected line, without checking received characters, but updating counter of received characters. The test ends if any error is found or after receiving CRTL D, printing the number of received characters.
5.3.3 SERVICE_PROCESSES

5.3.3.1 TIMER_PROCESS

The process target is to control the test time execution given at running time.

The process executes a sleep(2) of time (parameter given by monitor), and ends when end of time is reached, giving to monitor communication of event.

5.3.3.2 SERVICE_PROCESS

This process has the only target to capture any character introduced from console keyboard. If a control character meaning "break" or "status request" is found, the request is satisfied as follows:

- for status request, the shell command "es__1a" is executed, and then process returns in wait status for others commands.

- for break, exit() is executed, giving to monitor the break request.
5.4 COMMAND_FILE_FORMAT

SGTST is runned as an under shell single command, able to read a file sived as a parameter. Each line of this file describes a test in the format required by SGTST.

"Test" means a single process or a collection of processes having similar particularities, that have the goal to exercise system functionalities.

All the processes generated by the analysis of each command file line are executed concurrently with their "brothers", and concurrently with all others processes generated by "father" process. The "father" process is the monitor of all the runned tests.

Test input is a file, generable using any UNIX text editor (ex. VI), containing the command lines relatives to tests to be runned. This command file (SGCOM) is available on disk file system, and it is the user 0.S interface. This file is formed by n command lines, each having a structure as follows:

NPROC PRIORITY TEST-TYPE test specific informations

where:

NPROC number of concurrently processes

PRIORITY priority of the processes (NICE).

TEST-TYPE one of following:
SGCPU/SGSYS/SGIOB/SGIOC/SGTTYW/SGTTYR
SGPRTB/SGPRTC/SGFS.

The lines starting with # character are ignored, to permit comments insertion.

The fields NPROC and PRIORITY are commons to every test description line.
- NPROC must be an integer number not more high
- PRIORITY may assume values between 0 and 39 if SGTST is runned by super-user, else between 20 and 39.

The total process number may be at max. 18 if SGTST is runned starting from a user different from super-user.
COMMAND FILE example

---

*test n.1 for all the possibility of SGTST*

---

```bash
# test n.1 for all the possibility of SGTST

# nproc priority test-type memsize
# 2 29 SGCPU 12K

# nproc priority test-type memsize
# 2 30 SGSYS 10000

# nproc priority test-type c/s device mode devsize pattern
# 1 25 SGI08 s /dev/wd1 R 10000

# nproc priority test-type c/s device mode devsize pattern
# 1 22 SGI0C c /dev/fd W 10

# nproc priority test-type device file-name
# 1 38 SGTTYW /dev/tty1 /diag-f/path

# nproc priority test-type device
# 1 32 SGTTYR /dev/tty2

# nproc priority test-type device file-name
# 1 32 SGPRT /dev/lp /diag-f/path

# nproc priority test-type directory
# 4 20 SGFS /usr/dia

# nproc priority test-type device
# 1 32 SGCOMM /dev/tty2
```

---

Honeywell Confidential And Proprietary
5.5 OPERATING RULES

5.5.1 USER REGISTRATION

As user_id DIAG needs own environment having following particularities:

- login_id: root.
- password: NOT required.
- working directory: /usr/diag.

SGTST launch is made using a command as follows:

```
SGTST file name,second
```

where:

- file name: pathname of the file containing the commands.
- second: execution time (in seconds) of tests chain.

5.5.2 DIAG_USE

An example of shell-procedure launch for test modules execution (SGTST) follows:

```
# cd /usr/diag
# DIAG
```

The tests menu will be shown on console, giving the possibility to choose the area to be tested and the test duration.

Up to now are available, into diag_env directory, the file scripts to test various system areas (cpu,disk,diskette,fs...), and a script for the automatic testing of all the present resources.

User can modify the file scripts using standard editor (vi()), to create own diagnostic environments.

The DIAG menu is as follows:
Fri May 4 10:23:07 GMT 1986

***************************************************************
* *
* DIAG - Diagnostic System *
* Rev. ... April, 10 1986 *
* (c) Copyright Honeywell Information Systems Italia 1986 *
* *
***************************************************************

SELECT:

a Automatic system test
b Test of cpu
c Test of Disk
d Test of Diskette
e Test of Printer
f Test of File System
g Test of Terminals
h Test of Disk and/or Diskette read
i Test of Streamer
m Test of VMEbus controllers
q End of DIAG

Enter selection:

Note1 Is also possible to introduce the following commands:

l Prova comunicazione tra computer
r a list of current actives processes is shown on console
s all current processes are killed, and DIAG menu is shown on console
h the list of available ghost commands is shown on console

Honeywell Confidential And Proprietary
5.6 MESSAGES

5.6.1 STATUS_MESSAGES

At the end of all the process activation, SGSTST print the following message:

"ACTIVATES # PROCESSES"

Starting from now is possible to obtain a status report relatives to all the activated processes entering from console the character "r".

The obtained status report will be as follows:

<table>
<thead>
<tr>
<th>F</th>
<th>S</th>
<th>UID</th>
<th>PID</th>
<th>PPID</th>
<th>C</th>
<th>PRI</th>
<th>NI</th>
<th>ADDR</th>
<th>SZ</th>
<th>WCHAN</th>
<th>TTY</th>
<th>TIME</th>
<th>COMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>0</td>
<td>35</td>
<td>1</td>
<td>0</td>
<td>26</td>
<td>20</td>
<td>1064</td>
<td>44</td>
<td>8145F8</td>
<td>CO</td>
<td>0:00</td>
<td>errdemon</td>
</tr>
<tr>
<td>1</td>
<td>S</td>
<td>0</td>
<td>43</td>
<td>1</td>
<td>0</td>
<td>39</td>
<td>20</td>
<td>106F</td>
<td>48</td>
<td>FFF800</td>
<td>CO</td>
<td>0:05</td>
<td>cron</td>
</tr>
<tr>
<td>1</td>
<td>S</td>
<td>0</td>
<td>161</td>
<td>49</td>
<td>0</td>
<td>30</td>
<td>20</td>
<td>106C</td>
<td>36</td>
<td>8166C8</td>
<td>CO</td>
<td>0:03</td>
<td>sh</td>
</tr>
<tr>
<td>1</td>
<td>S</td>
<td>0</td>
<td>191</td>
<td>161</td>
<td>0</td>
<td>30</td>
<td>20</td>
<td>106D</td>
<td>64</td>
<td>816740</td>
<td>CO</td>
<td>0:01</td>
<td>ssstst</td>
</tr>
<tr>
<td>1</td>
<td>R</td>
<td>0</td>
<td>192</td>
<td>191</td>
<td>4</td>
<td>62</td>
<td>20</td>
<td>1074</td>
<td>52</td>
<td>C</td>
<td>C</td>
<td>0:29</td>
<td>sgiob</td>
</tr>
<tr>
<td>1</td>
<td>S</td>
<td>0</td>
<td>193</td>
<td>191</td>
<td>0</td>
<td>30</td>
<td>20</td>
<td>1075</td>
<td>44</td>
<td>816830</td>
<td>CO</td>
<td>0:01</td>
<td>ssstst</td>
</tr>
<tr>
<td>1</td>
<td>S</td>
<td>0</td>
<td>194</td>
<td>191</td>
<td>0</td>
<td>39</td>
<td>20</td>
<td>1087</td>
<td>28</td>
<td>FFF800</td>
<td>CO</td>
<td>0:00</td>
<td>sgtimer</td>
</tr>
<tr>
<td>1</td>
<td>S</td>
<td>0</td>
<td>195</td>
<td>191</td>
<td>14</td>
<td>22</td>
<td>20</td>
<td>1092</td>
<td>52</td>
<td>818394</td>
<td>CO</td>
<td>0:31</td>
<td>sgiob</td>
</tr>
<tr>
<td>1</td>
<td>R</td>
<td>0</td>
<td>196</td>
<td>191</td>
<td>41</td>
<td>80</td>
<td>20</td>
<td>10A9</td>
<td>52</td>
<td>C</td>
<td>C</td>
<td>0:33</td>
<td>sgiob</td>
</tr>
<tr>
<td>1</td>
<td>S</td>
<td>0</td>
<td>199</td>
<td>193</td>
<td>0</td>
<td>30</td>
<td>20</td>
<td>10AA</td>
<td>28</td>
<td>816A10</td>
<td>CO</td>
<td>0:00</td>
<td>sh</td>
</tr>
<tr>
<td>1</td>
<td>R</td>
<td>0</td>
<td>200</td>
<td>199</td>
<td>0</td>
<td>60</td>
<td>20</td>
<td>10CC</td>
<td>96</td>
<td>C</td>
<td>C</td>
<td>0:02</td>
<td>ps</td>
</tr>
</tbody>
</table>
where: F (1) Flags (octal and additive) associated with the process:

- 01 in core
- 02 system process
- 04 locked in core (e.g., for physical I/O)
- 10 being swapped
- 20 being traced by another process
- 40 another tracing flag

S (1) The state of the process:

- 0 non-existent
- 1 sleeping
- 3 waiting
- 2 running
- 1 intermediate
- 0 terminated
- 3 stopped
- 4 growing

UID (f,1) The user ID number of the process owner; the login name is printed under the -f option.

PID (all) The process ID of the process; it is possible to kill a process if you know this datum.

PPID (f,1) The process ID of the parent process.

C (f,1) Processor utilization for scheduling.

STIME (f) Starting time of the process.

PRI (1) The priority of the process; higher numbers mean lower priority.

NI (1) Nice value; used in priority computation.

ADDR (1) The memory address of the process, if resident; otherwise, the disk address.

SZ (1) The size in blocks of the core image of the process.

WCHAN (1) The event for which the process is waiting or sleeping; if blank, the process is running.

TTY (all) The controlling terminal for the process.

TIME (all) The cumulative execution time for the process.

CMD (all) The command name; the full command name and its arguments are printed under the -f option.
5.6.2 ERROR_MESSAGES

When a process finds an error condition, send a message to output file (the console, as default) and ends its execution (status terminated).

The error messages have following informations:

PROC.ID  NLINEA  KEYWORD  message text

All the error messages text are into the same file (SGERR), to permit the translation.