S G M 2

DOCUMENT ORDER A 78138664
REVISION
STATUS DRAFT
DATE JUNE 1986
DOCUMENT TREE

DIAGNOSTIC MANUAL

PREPARED BY

E. Casero

REVIEWED BY

M. Nobile

APPROVED BY

E. Maccario

Honeywell

Honeywell Information Systems Italia

Cod. 3897882 W

INDEX

1. INIRODUCIION

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_OE_DIAGNOSIS

The SGM2 diagnosis is made at three levels:

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

3. LEVEL_1_(SIARI)

It's always executed during the initialization phase. It's devided 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_OE_CPO_BOARD

The following areas are tested by RDR:

EPROM	. checksum control
PIT	. registers integrity . interval timer
MMU	. "registers" 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_OE_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

CPO/SPO interrupts

3.2.3 RDR_OE_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

CPO/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:

CODE	ERROR DESCRIPTION	ORU	CRU

* = blinking value

3.4 BOOTSTRAP

In developing this bootstrap sequence, also controller and device operability has been take in account.

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

Bootstrapping 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:

Addr. 000000	DC0/DP0 entry table
000010	DC1/DP1 entry table
000020	S/LP0 entry table
000030	S/LP1 entry table
000040	S/LP2 entry table
000050	S/LP3 entry table
000060	S/LP4 entry table
000070	S/LP5 entry table
000080	S/LP6 entry table
000090	S/LP7 entry table
0000A0	VMEbus controller 0 entry table
0000B0	VMEbus controller 1 entry table
0000C0	VMEbus controller 2 entry table
0000D0	VMEbus controller 3 entry table
0000E0	CP0 entry table
0000F0	CP1 entry table
000100	CH0 entry table
000110	CH1 entry table
000120	SC0 entry table
000130	R.F.U. entry table

000140 MEMORY BIT MAP

000144 INITIALIZING DEVICE NUMBER

- Each board entry table contains informations about:
  - board presence/absence
  - board functionality
  - slot insertion number
  - processor type
  - lines informations (only for L/SPO boards)

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

- Initializing device number supply the number of Disk from which Operating System has been loaded, starting from 00.

4. LEVEL_2_(SIAL)

4.1 INIRODUCIION

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
- SCO 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_SIRUCIURE

The floppy DIAGX2 has a sequential files structure; files access is made via the parameters TRACK/SECTOR/HEAD of the file start, setted 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_Floppy_structure

TRK 00

01	02	03	04	05	06	07	08	09
PRIMARY	P_SECT	FILE 1	....				..	.....

TRK 01

01	02	03	04	05	06	07	08	09
.....	FILE 2	.....			..	.....	FILE 3	.....

TRK 02

.

.

TRK n

01	02	03	04	05	06	07	08	09
			1st free sector					

TRK n+1

01	02	03	04	05	06	07	08	09
AVAILABLE FOR FORMAT/WRITE/READ TESTS								

.

.

TRK 79

01	02	03	04	05	06	07	08	09
AVAILABLE FOR FORMAT/WRITE/READ TESTS								

4.5 PRIMARY_(_SGM2BI_)

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:

Bytes count	Contents
0-5 1-6	file name(must be on 6 character)
6-7 7-8	start cylinder
8 9	" head
9 10	" sector
11-12	length (bytes)
13-16	entry address of Process

The last 4 bytes of P_ SECT contents 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 pluss test
- SG2BDC	Test tool for Disks/Floppy
- SG2DFC	Test tool for DCO controller

4.6 DIAGNOSTIC_MONITOR

The STAL Diagnostic Monitor hands the console dialog 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 dialog 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 dialog will be in Italian

- if "dcba" is introduced, console dialog 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
- SG2MEM
- 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.

b — 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.

c — The Diagnostic Process of CP0 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.

d — The Diagnostic Process of SC0 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.

e — The Diagnostic Process of CH0 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.

f — 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.

g — 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.

h — 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.

i — 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.

l — 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.

m — 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
BR	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 send 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 send 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 DD_=_DIAGX2_floppy_dup

The DD 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: DD

```
= DD
* Mount the new floppy. Reply Y when done.
= Y
=
```

4.7.2.4 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 appare 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: bR

4.7.2.5 *⟨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: ⟨name⟩

⟨name⟩ = Name of Process to be load

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

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

Is possible to load from DIAGX2 floppy a Process, at the sived address, using LD command.

Command use: LD <name> <address>

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

= LD SG2FLP 6000
= LD TEST1 84000
=

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

4.7.2.7 GO_=_Start/Restart_a_Process

Is possible to start/restart a Process, at the sived 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 sived address.

4.7.2.8 OS - Re-boot the system starting from Disk

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

A very usefull feature offered by this command is bypasse 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
=

4.7.2.10 NB_=_Clear_breakpoint

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

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

```
= PR
**********************************************************

The System configuration is (VMEbus Units are not sived):
(P = present Unit     / = absent Unit     P = System Disk)

MEMORY available: 08 Mb                    CP01   /

CH00  P      SP00  P      SP01  P      SP02  P      LP03  P
CH01  /      xP00  /      xP01  /      xP02  /      xP03  /

TAPE      P        DISK0  P        DISK1  P        DISK2  P
DISK3     /        DISK4  /        DISK5  /        DISK6  /
=
```

4.7.2.12 DR_=_Display_all_PROCESSOR_registers

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

Command use: DR [Pr. name]

[Pr. name] = Processor name
Default = CP00

```
= DR
PC=000068A4 SR=2510
D0=0000D4F9 D1=FFFFFFFF D2=00000000 D3=00007FFF
D4=00000000 D5=7FFF0001 D6=00807800 D7=008042C8
A0=00803A26 A1=0080E53A A2=00000000 A3=FFFFFFFF
A4=0088FFFF A5=00007FFF A6=00000000 A7=00007AB4
= DR SP1
PC=00102AC2 SR=2300
D0=00000000 D1=FFFFFFFF D2=000A6400 D3=FFFFFFFF
D4=00347FF0 D5=7FFFFFFF D6=00000000 D7=000054C0
A0=003444D6 A1=0034055A A2=00000000 A3=00000000
A4=F000FFFF A5=FFFFFFFF A6=FFFF09D2 A7=00102FB4
=
```

4.7.2.13 D0_..._DZ_=_Display/modify_PROCESSOR_data_registers

The command Dx displays or changes the value of CPO or L/SPO data register 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: D<value1> [Pr. name] [value2]

<value1> = Register number
[Pr. name] = Processor name
Default = CP00
[value2] = value to be assigned to register (Hex.)

```
= D1
FFFFFFFF
= D1 55A8
= D1
000055A8
=
```

```
= D1 SP0
00000000
= D1 SP0 F499
= D1 SP0
0000F499
=
```

4.7.2.14 A0...AZ - Display/modify PROCESSOR addr. registers

The command Ax displays or changes the value of CP0 or L/SP0 address register Ax. If the value is changed, the register Ax will have the new value when the execution of Process code will be resumed.

Command use: A<value1> [pr. name] [value2]

```
<value1>      = Register number
[pr. name]    = Processor name
                Default = CP00
[value2]      = value to be assigned to register (Hex.)

= A5
000054E2
= A5 0
= A5
00000000
= A0 SP2
00C03DD2
= A0 SP2 C00000
= A0 SP2
00C00000
=
```

4.7.2.15 SR - Display/modify PROCESSOR status registers

The command SR displays or changes the value of CP0 or L/SP0 status register. If the value is changed, the register SR will have the new value when the execution of Process code will be resumed.

Command use: SR [pr. name] [value]

```
[Pr. name]    = Processor name
                Default = CP00
[value]       = value to be assigned to register (Hex.)


= SR
2700
= SR 2300
= SR
2300
= SR SP0
2000
= SR SP0 2700
= SR SP0
2700
=
```

4.7.2.16 PC_=_Display/modify_PROCESSOR_program_counter

The command PC displays or changes the value of CP0 or L/SP0 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 6000
= PC
00006000
= PC SP1
00104FF8
= PC SP1 104000
= PC SP1
00104000
=
```

4.7.2.17 MD_=_Display_memory_on_console

Using the command MD, 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 65 88 09
= MD 560000A8 90
560000A8    CC 34 E5 67 CC DD A3 FF FF FF FF 6B 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
560000F8    A5 44 67 B5 DC 33 00 00 00 0F FF D4 3E 8F BB 00
56000108    4E 71 4E 71 4E 71 8A 9B 44 50 00 0F FF FF 00 FF
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 00 FF FF FF FF FF 00
= MD SP0 104000 50
00104000    CC 34 E5 67 CC DD A3 FF FF FF FF 6B C3 80 00 00
00104010    22 33 A2 DE 65 BC DD 00 00 00 00 00 00 00 00 00
00104020    4E 71 4E 71 BD 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: MM [pr. name] [value]

[pr. name] = Processor name
Default = CP00
[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 EI_=_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: FT

```
= 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

```
= LS
**************************************************************
*                                                            *
* file name   cyls  head  sect    leng     entry address     *
*                                                            *
**************************************************************
*
*  CPOMON     0000   00    03     5C00      00001408        *
*  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      000060EE        *
*  SG2CHO     0009   01    06     3B00      000064F2        *
*  SG2SCO     000A   01    02     3800      00006800        *
*  SG2VMX     000B   00    07     6400      00006000        *
*  SG2TAP     000D   01    04     3200      000065D0        *
*  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 availlables in STAL system, for the current revision level. An indication of the tested area is also supplied.

Command use: PN

```
= 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 list on console the Revision Level of DIAGX2 floppy and of EPROM code.

Command use: RL

```
= 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      /
=
```

4.8 DIAGNOSIIC_PROCESSES_EXECUIION_MODES

Is possible to run STAL Diasnostic Processes in two different modes:

- automatic mode
- technical mode

4.8.1 AUIOMAIIC_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:

*ERR: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 YYYY - ERRORS 0000

where: SG2XXX = Process name
YYYY = pass counter

The user selection menu is as follows:

```
**************************************************************
*                                                            *
*                  STAL - Diagnostic System                  *
*            Revision ....  -----      May, 24 1986          *
*   (c) Copyright Honeywell Information Systems Italia 1986  *
*                                                            *
**************************************************************
```

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 CPO
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 IECHNICAL_MODE

This mode is selected enterins from console the Process name.

The followins request is made by Monitor:

* Recycle on test ? (Y/N) *

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

Control is now sived to Diasnostic 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 reportins 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 DIAGNOSIIC_PROCESS_OE_CPO_BOARD_(SG2CPX)

The Diasnostic Process SG2CPX tests the CPO board of SGM2 system.
The following areas are tested:

- 68020
  - . H/W resisters intesrity
  - . addressins modes
  - . zero divide trap
  - . illesal instruction trap
  - . priviledse violation trap
  - . illesal address trap
  - . user traps
  - . exceptions nestins
- FPU
  - . instructions execution
  - . exceptions
- PIT
  - . resisters intesrity
  - . timers
  - . interrupts to CPU
- MMU
  - . "resisters" intesrity
  - . read/write mem throush descript
  - . read/write violation error
  - . undefined sesment access error
  - . disable sesment error
- Memory addressins
- 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

CODE	ORU	DESCRIPTION

4.10 DIAGNOSIIC_PROCESS_OE_DISK_(SG2DSK)

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

- DCO controller
- Disk devices

The SG2DSK Process hands bad sectors encountered durins tests execution: if sector sivins error is into bad block table, no error message is send.

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

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

4.10.1 OPERAIING_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, startins 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 FORMAT_SELECTION_

Enter the selection corresponding to choosed disk. The followins messages will be displayed on console:

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

If N is entered, the test stops, else:

*MSG: 03 START OF PREPARE ON DISK x

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

The obtained Bad Block Table is accordins to Operatins System structure and disk allocation.

4.10.3 WRITE_SELECTION

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

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

If N is entered, the test stops, else:

*MSG: 09 START OF WRITE DISK x

All the disk will be written usins selected pattern. The bad sectors will be hand accordins 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 TEST 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 complamented 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 choosed disk. The following messages will be displayed on console:

*MSG: 0C 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 choosed disk. The following messages will be displayed on console:

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

   XXXXX               CCCC      HH      SS
     .                  .        .       .
     .                  .        .       .
   XXXXX               CCCC      HH      SS
```

where:

XXXXX = logic bad sector number
CCCC = cylinder \
HH = head) physic values
SS = sector /

4.10.7 DISPLAY_DCO_REGISTERS_&_ECA_TABLE_SELECTION

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

Enter E: 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_OF_TEST_SELECTION

If X selection is entered, SG2DSK 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 DIAGNOSIIC_PROCESS_OE_ELOPPY_(SG2ELP)

SG2FLP Diasnostic Process is able to detect failures in the followins areas:

- DCO controller
- Floppy device

The SG2FLP Process works on media havins 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, workins onto DIAGX2 floppy free cylinders.

4.11.1 OPERAIING_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, startins 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,E86D86DB)
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_DIAGX2_SELECIION

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_SELECIION

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_REGISIERS_&_ECA_IABLE_SELECIION

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:

4.11.8 COPY_SELECTION

Mount the floppy to be copied, and enter selection corresponding to choosed 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 appare 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 0D 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 0C 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 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 PHISYCAL 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 DIAGNOSIIC_PROCESS_OE_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 OPERAIING_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_REPORIING

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 followins 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__SELECTION

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__SELECTION

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__SELECTION

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__SELECIION

The followins steps are executed:

- tape retention
- rewind until BOT
- read of all the cartridse, 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__SELECIION

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_REPORIING

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

*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 RESETTING RDY F/F

*ERR: ERROR RESETTING EXC F/F

*ERR: ERROR IN SET-RESET DIRECTION BIT

*ERR: ERROR IN SET-RESET ACKNOLEDGE BIT

4.14 DIAGNOSIIC_PROCESS_OE_CACHE_(SG2CHX)

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

TBW

4.14.1 OPERAIING_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, startins from address $6000, and executed in recycle mode until a "break" is send from console. At the end of each pass, the Pass/Error messase is updated.

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

4.14.2 ERROR_REPORIING

If any error is found durins Process execution, an error messase is send to console, as follows:

```
*ERR:XXX   CRU = UNIT CHO
  text
```

where: XXX = error code
text = one of followins messases

4.15 DIAGNOSIIC_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 OPERAIING_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_REPORIING

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:

CODE	DESCRIPTION

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 TEST_OF_ALL_THE_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_TEST_("K"_SELECTION)

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

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:

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

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

4.16.6.2 ____IESI

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

*ERR:XYY CRU = UNIT ...
text

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

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 send 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 send on console:

*ERR:OOX CRU = UNIT YYY

where: OOX = error code

YYY = Unit in error

The possible error code are as follows:

4.18 DISK_FLOPPY_CONTROLLER_TEST_TOOL_(SG2DFC)

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

4.18.1 OPERATING_RULES

An example of SG2DFC Process load and execution follows:

```
= 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_TEST_SELECTION

T.B.D

4.18.3 ACCEPTANCE_TEST_SELECTION

T.B.D

4.18.4 ELOPPY_EMULAIION_SELECIION

T.B.D

4.18.5 DISPLAY_DCO_REGISIERS_&_ECA_IABLE_SELECIION

T.B.D

4.18.6 CHANGE_EXECUIION_MODE_SELECIION

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

4.18.7 END_OE_IESI_SELECIION

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 followins messases is send on console:

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:

```
= 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:

```
    DEVICE TYPE
DISK0 = 0     DISK1 = 1     DISK2= 2     FLOPPY = 3


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

    AVAILLABLE 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: CCCC     = cylinder
WRITE PARAMETERS:    CCCCHHSSPPPPPPPP           HH       = head
DIAG. WRITE PARAM.:  CCCCHHSS                   SS       = sector
FORMAT PARAMETERS:   CCCCHH                     AAAAAAAA= address
SEEK PARAMETERS:     CCCC
MEMORY MODIFY PAR.:  AAAAAAAA

PARAMETERS :
```

This menu is also displayed everytime a "break" is send from console.

At this point, 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 gived 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.

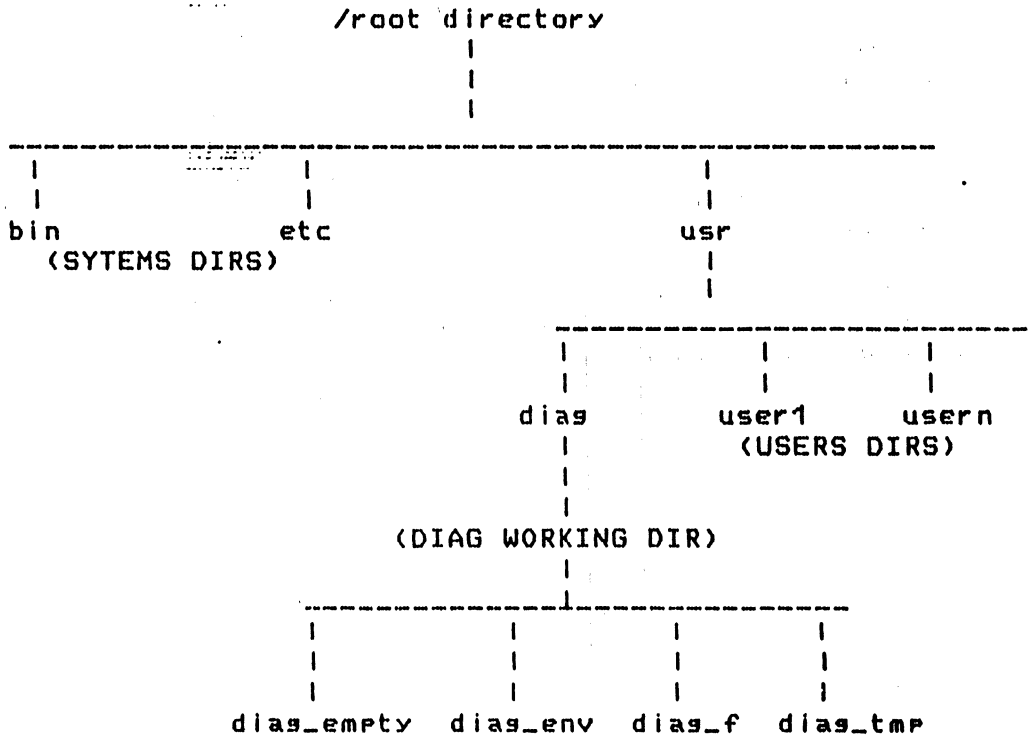
DIAG can work at system level and/or subsystem level,changing number and types of tests selected by user.

All the informations need by test (also ones concerning devices operability) are given 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 INTO O.S. FILE SYSTEM

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



- DIAG DIRS -

Using this structure is possible to solve testing problems in i/o block area resuarding disk space availability and file system visibility.

5.2.1 DIAG_FILE_SYSTEM

The following files are present into DIAG file system:

. /usr/diag

.. DIAG executable modules:

... sgtst
... sgcomm
... sgcpu
... sgfs
... sgiob
... sgprt
... sgservice
... sgsys
... sgtimer
... sgttyr
... sgttyw

.. diagnostic shell-procedure (DIAG):

. /usr/diag/diag_c

.. DIAG source modules:

... buffcmp.c
... buffgen.c
... dupbuf.c
... mkfile.c
... rb0.c
... rr0.c
... rr1.c
... sgcomm.c
... sgcpu.c
... sgfs.c
... sgiob.c
... sgioc.c
... sgmon.c
... sgprt.c
... sgservice.c
... sgsys.c
... sgtimer.c
... sgttyr.c
... sgttyw.c
... wb0.c
... wr0.c
... wr1.c

.. DIAG compiled modules:

.. errfile (DIAG error messages files):

.. DIAG makefile:

. /usr/diag/diag_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"
- SGCMDFS test "file system"
- SGCMDLP test "line printer"
- SGCMDTTYW0-3 test "tty output"
- SGCMDTTYR0-3 test "tty input (keyboard)"
- SGCMDCOM0-3 test "communication"

.. A list of available script files follows:

```
# diag_env/sgcmdauto

    # default script for automatic system test
    #
    # selection "a" of DIAG
    #
    #
    1 20 sgcpu 5000
    #
    1 20 sgsys 5000
    #
    1 20 sgiob s diag_tmp/tmp000 wr 10000 aaddee11
    #
    1 20 sgiob s diag_tmp/tmp001 wr 10000 i
    #
    1 20 sgioc s /dev/rfd wr 20 11223344
    #
    1 20 sgfs /usr/bin
    #
    #1 20 sgprt /dev/lp diag_f/lina32
    #
    1 20 sgttyw /dev/tty0 diag_f/sgwinput
    #
    1 20 sgttyw /dev/tty1 diag_f/sgwinput
    #
    1 20 sgttyw /dev/tty2 diag_f/sgwinput
```

```
# diag_env/sscmddkt

    # default script for diskette test
    #
    # selection "d" of DIAG
    #
    #
    1 20 ssioc s /dev/rfd wr 300 11223344



# diag_env/sscmddsk

    # 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
```

```
# diag_env/sgcmdlpr

    # default script for test of printer 132cgl
    #
    # selection "e" of DIAG
    #
    #
    1 20 sgprt /dev/lp diag_f/132a



# diag_env/sgcmdfs

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



# diag_env/sgcmdcom3

    # 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 sgconn /dev/tty3
```

```
# diag_env/sgcmdcom0

    # default script for test of line 0 in input mode
    #
    # selection "h" of DIAG
    #
    # test receive data from another computer on line 0
    #
    #
    #
    1 20 sgcomm /dev/tty0


# diag_env/sgcmdcom1

    # default script for test of line 1 in input mode
    #
    # selection "h" of DIAG
    #
    # test receive data from another computer on line 0
    #
    #
    1 20 sgcomm /dev/tty1


# diag_env/sgcmdcom2

    # default script for test of line 1 in input mode
    #
    # selection "h" of DIAG
    #
    # test receive data from another computer on line 0
    #
    #
    1 20 sgcomm /dev/tty2
```

```
# diag_env/sgcmdtty3

    # default script for test of line 3 in output mode
    #
    # selection "g" of DIAG
    #
    # the printed file sgwinput is a default file usable for
    # all the tty types. Other pattern are available under
    # the directory /usr/diag/diag_f
    #
    #
    1 20 sgttyw /dev/tty3 diag_f/sgwinput


# diag_env/sgcmdtty2

    # default script for test of line 2 in output mode
    #
    # selection "g" of DIAG
    #
    # the printed file sgwinput is a default file usable for
    # all the tty types. Other pattern are available under
    # the directory /usr/diag/diag_f
    #
    #
    1 20 sgttyw /dev/tty2 diag_f/sgwinput


# diag_env/sgcmdtty1

    # default script for test of line 1 in output mode
    #
    # selection "g" of DIAG
    #
    # the printed file sgwinput is a default file usable for
    # all the tty types. Other pattern are available under
    # the directory /usr/diag/diag_f
    #
    #
    1 20 sgttyw /dev/tty1 diag_f/sgwinput
```

```
# diag_env/sgcmdtty0

    # default script for test of line 0 in output mode
    #
    # selection "g" of DIAG
    #
    # the printed file sgwinput is a default file usable for
    # all the tty types. Other pattern are available under
    # the directory /usr/diag/diag_f
    #
    #
    1 20 sgttyw /dev/tty0 diag_f/sgwinput
```

. /usr/diag/diag_f

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

. /usr/diag/diag_tmp

.. work directory for tests: sglob sgfs ....

. /usr/diag/diag_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 SGM2
- SERVICES: are the processes collecting external monitor requests
- UTILITY : is a utility process, called SGSYNTAX, able to exec the syntax analisys of the command file that is to be supplied at DIAG running time

NOTE:

As a user depending responsability, 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 tipical 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, (exit(2) of SGXXX), caused by an error detection.

- if others test processes are running, the counter of end processes is update and a wait for an event occur.

- if there are no running processes (generated processes counter = end processes counter), then:

- kill of all the running processes

- send to console of the message:

"SGTST END TEST "

- exit of father process and return to Shell.

LIMITATIONS:

- tests generation and execution is not syncronized.

- error management at 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 identifyer 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 identifyer
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.

NOIE:

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

LIMIIAIIONS:

- MEMSIZE parameter must be <=65535

5.3.2.3 SGIOB

Goal:

- Perform I/O requests on the specified blocks device, using standard I/O buffered structures for the device.

- It is as an application that requires standard I/O from disks, floppies or tapes

Command line format:

NPROC PRIORITY SGIOB DEVICE R/W/WR DEVSIZE PATTERN

where:

NPROC	must be 1
SGIOB	is the test identifyer
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 - xxxxxxxx (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:

$$BSIZE + \sum_{i=0}^{BSIZE} i \quad \text{until DEVSIZE}$$

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 SGIOC

Goal:

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

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

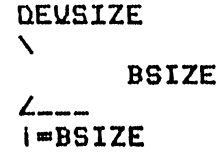
Command line format:

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

where:

NPROC	must be 1
SGIOC	is test identifyer
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 - xxxxxxxx (hex. string)

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



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 sived device, and so hands only I/O strings.

Command line format:

NPROC PRIORITY SGTTYW DEVICE FILE NAME

where:

NPROC	must be 1
SGTTYW	test identifyer
DEVICE	device pathname (must be different from console)
FILE NAME	pathname of the file to be shown

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. Tipical example is the keyboard function keys.

Command line format:

NPROC PRIORITY SGTTYR DEVICE

where:

NPROC	must be 1
SGTTYW	test identifyer
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

NOIE :

To execute the test on a specifyed terminal, must run the test starting from another terminal.

5.3.2.7 SGPRI

Goal:

- It is as SGTTYB 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 SGPRT DEVICE FILENAME

where:

NPROC	must be 1
SGPRT	test identifyer
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 hands updating and requests on regular files and directorys 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 identifyer
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 satisfyed as follows:

- for status request, the shell command "ps___=la" is executed, and then process returs 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 given 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 analisys 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, genarable 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 O.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
#
#
#   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       SGIOB       s    /dev/wd1    R     10000
I
#
#   nproc priority  test-type  c/s  device        mode     devsize
pattern
      1       22       SGIOC       c    /dev/fd         W           10
R
#
#   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/diag
#
#   nproc priority  test-type             device
      1       32       SGCOMM              /dev/tty2
```

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
s	Test ot Terminals
h	Test of Disk and/or Diskette read
l	Test of Streamer
m	Test of VMEbus controllers
q	End of DIAG

Enter selection:

Note: Is also possible to introduce the followins commands:

i	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 shost commands is shown on console
REDE	informations relative to current DIAG release are shown on console

5.6 MESSAGES_

5.6.1 SIAIUS_MESSAGES

At the end of all the process activation, SGTST print the following messase:

"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:

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	COMD
1	S	0	35	1	0	26	20	1064	44	8145F8	C0	0:00	errdemon
1	S	0	43	1	0	39	20	106F	48	FFF800	C0	0:05	cron
1	S	0	161	49	0	30	20	106C	36	8166C8	C0	0:03	sh
1	S	0	191	161	0	30	20	106D	64	816740	C0	0:01	sgtst
1	R	0	192	191	4	62	20	1074	52		C0	0:29	sgiob
1	S	0	193	191	0	30	20	1075	44	816830	C0	0:01	sgservic
1	S	0	194	191	0	39	20	1087	28	FFF800	C0	0:00	sgtimer
1	S	0	195	191	14	22	20	1092	52	81B394	C0	0:31	sgiob
1	R	0	196	191	41	80	20	10A9	52		C0	0:33	sgiob
1	S	0	199	193	0	30	20	10AA	28	816A10	C0	0:00	sh
1	R	0	200	199	0	60	20	10CC	96		C0	0:02	ps

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
- S sleeping
- W waiting
- R running
- I intermediate
- Z terminated
- T stopped
- X growing

UID (f,l) 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,l) The process ID of the parent process

C (f,l) Processor utilization for scheduling

STIME (f) Starting time of the process

PRI (l) The priority ot the process; higher numbers mean lower priority

NI (l) Nice value; used in priority computation

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

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

WCHAN (l) 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 rhe -f option

5.6.2 ERROR_MESSAGES

When a process founds 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.