Cajani R.P.

SGM-2

DOCUMENT ORDER	A78138904
REVISION	AA
STATUS	Draft
DATE	December 9.
DOCUMENT TREE	
COMMESSA	

SGM-2 SYSTEMS SOFTWARE - E.P.S.

PREPARED BY

[illegible]

V. Cajani

REVIEWED BY

APPROVED BY

Antonio Cicu

A. Cicu

Honeywell

Honeywell Information Systems Italia

Loc. Pregnana Milanese, Italia

- S/W to be executed on the front-end and back-end processors.

The objective of the front-end processor is to provide a decentralized execution of the TTY driver, in order to increase the system throughput in terms of Kbs, to increase the CPU througnput (reducing the cpu time required to handle terminal I/O operations) and to optimize the response times.

The objective of the back-end processor is to increase the disk subsystem throughput, reducing the average time required to complete a disc I/O operation requested by the CPU.
The CPU S/W itself will be transparent to the existence of the back-end processor.

The horizontal applications provide facilities in the following areas:

- programming languages

- data base management systems

- communication packages

- office automation utilities.

3.1.1. The system initialization and shutdown

When the system is in stand by status, it is possible to start the initialization sequence, in one of the two ways:
- locally, pushing the "DC POWER ON" switch of the operator pannel
- from remote, via a telephone call.

Having the system already in power on status, the initialization sequence can be entered by pushing the "RESET" switch.

The system initialization consists of three steps:
- run of T&Ds
- primary boot
- secondary boot.

The run of T&Ds guarantees the user that the system is started only after a check that the H/W works properly. The T&Ds are locally executed by each controller/processor via EPROM resident diagnostic routines.
If the execution of a T&D detects an error, an hardware malfunction identification code is displayed on the operator pannel hexadecimal display.
In particular the first megabyte of main memory must be operable in order to allow the system to boot.

The primary boot is executed as CPU (or SCO) EPROM code; it has the task to select the initializing device and to load the secondary boot program from that device into maim memory and give control to it. Any error detected by the primary boot is notified via the operator pannel hexadecimal display.

The secondary boot downloads the code of the VME processors from the init device into the local memories of the VME processors. After that it asks the system console operator if a non "standard" initialization has to be done. If none character is entered from the console within 5 seconds, the secondary boot gives control to the UNIX program. Otherwise, the console operator via an interactive dialogue with the secondary boot can:
- select to run the file system check utility
- select to run a program different from UNIX
- select to download into a VME processor local memory a code different from that one selected by default (i.e. the file stored onto the /system directory and with a name equal to the processor identifier itself).

In the case that the calendar clock battery is out of power, the secondary boot informs of the event the console operator and waits for the correct date to be entered by him.

The console operator is informed via a specific message if the previous system shutdown was not correctly completed (TBD), so initialized via a communication line.

If the operator enters a character in reply to the above question, he can do the following:

- execute/not execute the file system check utility

- select to run a new copy of the UNIX program.

- select to load a processor (like the station processor) local memory with a code different from that one loaded in the proper root file system directory.

If the initialization procedure detects a system configuration which is inconsistent with that defined to UNIX via the configuration facility:

- if more processors/controllers are connected than those declared to UNIX, a warning message is delivered

- if the type of processor/controller connected differs from that declared to UNIX, a message is delivered and the system halts.

3.1.1.2. Initialization from diskette and tape

The inizialization is executed from diskete or tape if an initializing diskette or tape is mounted at power on or reset time.

The reasons for having an initialization from one of these two devices are one of the following:

- execute a recovery procedure

- execute back up copies (or restore back up copies) of the on line disc volumes

- run a stand alone utility

- etc. (see paragraph 3.1.6.).

In case of initialization from diskette or tape, the system/user interactions are usually only via the system console, i.e. the

3.1.1.4. Battery back-up facility.

The SGM2 system allows the optional connection of an external switching battery, auxillary power supply able to communicate with the SGM2 SCO board.

Via this option, as soon as the power fails, all the central H/W is powered by the external battery and the UNIX kernel is signalled of the event so that it is able to do the necessary recovery actions before to power off the system. To be remarked that the system is powered off as soon as possible, in order to save battery power; even when the system is powered off, it will be able to receive from the external switching battery the information that power is back on, so that the system activities can be restarted.

Two distinct AC fail recovery options are supported, according to a system administrator choice:

- soft fail option. It is the default option an does not require any resource in addition to the external battery itself. It provides exactly the same protection as when the power off switch is pressed, i.e. processes are signaled of SIGPOWER (19) and buffers are sync (see above) before to turn off the power.

- non-stop option, which requires a magnetic device (a disc slice or tape), able to store all the memory contents (including main memory, and VME processors local and shared memories) in less than 15 minutes.

The non-stop option provides the saving on the raw device of all the system memories, so that when the system is powered on again the cosole can select to resume the system activities from the point at which AC power failed (optionally the console operator can select to execute a soft fail instead of a resume).

To be remarked that if the power is back again in a short time (being the term short a number of minutes less than 15 and user selected) the system activities are resumed directly from main memory (without asking any question to the console operator), i.e. are resumed in a very short time.

3.1.2. The kernel

The major characteristics of the SGM2 UNIX kernel are to provide an integrated support of the demanded paging and multiprocessor facilities, still maintaining a full compatibility with the standard.

3.1.2.1. The kernel programming interface

All the UNIX System V Release 2 Version 2 system calls are available, providing the same interfaces as those described in the Programmer Reference Manual of the above AT&T release.

To be remarked that the system call interface is compatible with the interface proposed by the /user/group, with the few limitations described by the "System V Interface Definition" published by AT&T.

In addition to that,, the system call interface is also compatible with the interface proposed by the X/Open Group, exceept for the above limitatons.

The run of the System V Verification Suite tests (delivered by AT&T) will guarantee the compliance to the standard. SVVS test relusts will be shipped to AT&T, which is committed to make public the results of the product verification at the beginning of the 1987.

See chapter 6 for wath concerns the portability of AX-Superteam applications to the SGM2 machine.

that is the kernel (which is a process loaded and locked in main memory at boot time) and the shared memory segments, which are entirely allocated in main memory at creation time and locked untill used

- the program pages are shared among concurrent processes (executed on the same processor, see following paragraph):
-- complete sharing of text pages
-- sharing of data pages, untill not modified (see copy-on-write facility), in order to provide a very powerfull support to the "fork" kernel primitive

- main memory page replacement algorithm:

-- based on the LRU (least recently used) policy, selected as a compromise between the FIFO policy and the working set policy defined by P.J. Denning. H/W assist to help the S/W deamon in defining the LRU pages.
To be remarked that the LRU policy states that whenever a fresh page of main memory is needed, the page unreferenced for the longest time (among all the pages of all the processes executed by the CPU, but locked pages) is removed, whereas the working set policy chooses for remove a page of a non-active process or one "non-working-set" page of an active process.

-- pages to be replaced are swapped on the disc device only if necessary: when modified (H/W assist to identify modified pages) or when never swapped (i.e. only loaded from the original device). The reason for swapping unmodified pages is due to the much greather efficency in loading pages from the swapp device instead of loading pages from the original device.

-- none process "page groupping" facility is provided, as for example S/38 provides via the access group facility. The lack of such a facility strongly simplifies the system architecture, but could impact performances, particularly in a transaction processing environment in the case that the number of program pages required to process the transactions is significantly greater than the number of main memory physical pages: an improvement in this area is under evaluation.

define which processing has to be executed on each CPU.
Via this option, a specific CPU executes only the processes related to an administrator defined set of work stations. Via this option, an administrator defined set os resources (cpu and main memory) is allocated to a set of users, so that the performences of the processing executed by this set of user is predictable, being not impacted by the processing executed by the other users.

3.1.2.3.1. Main memory mapping

The mapping of information in the two CPU memories provides to satisfy two major requirements:

- tightly coupled multiprocessor requirement, which implies the sharability of main memory data among the two CPUs

- performance requirement, which implies to keep the process pages loaded as much as possible (i.e. without missing the above requirement) into the memory of the CPU which is running the process. In fact, even if it is possible via the VME bus to access from one CPU the memory of the other CPU, the best performances are obtained accessing CPU "local" memory pages.

From a logical point of view, the main memory data can be divided into two categories:

- GLOBAL: SHARED among processes and/or CPUs

- LOCAL: used only by a process and/or CPU.

The data types allocated in the two types of memory are:

GLOBAL	LOCAL
- kernel tables type S(hared)	- kernel tables type E(xclusive)
- shared memory segments	- kernel stack
	- kernel U-block
	- user program stack

3.1.2.3.2. Process management and event notification

Being the execution of a process statically assigned to a CPU, each CPU manages a specific set of processes.
Consequently, the sharing of the CPU local resources among the processes assigned to the CPU itself is independently decided for each CPU:

- the demand pager operates on CPU local data and local memory pages independently from the other CPU demand pager (exceept for the allocation of shared memory segments) and there is one page daemon per CPU

- the process dispatching routine provides time sharing of the CPU among the processes assigned to the CPU itself, independently from the dispatching of processes on the other CPU

- the interval timer (CPU local device) is locally used to manage what above.

Each CPU can interrupt the other CPU: when a CPU processes an event which changes the status of a process allocated to the other CPU, it interrupts the other CPU, which will rearrange the local process queue status.
What above will typically occur when a process releases a resource, for which a process assigned to the other CPU is waiting or when signals are issued, etc.

On the other hand, the SGM2 architecture is designed to minimize the probability of interCPU notifications required to readdress interrupts, originated by device controllers and processors, from one CPU to the other (in fact in this case the readdressing of interrupts means loss of performances):

- each CPU can execute the device drivers

- each item of the I/O request queue of synchronous devices (like discs) contains the identifier of the CPU which runs the process which has issued the I/O request, so that at each I/O operation the device driver (no matter which CPU is running it) can program the BIM of the controller, specifying which CPU will be interrupted when the I/O is terminated:
CPU to CPU notification is required only to wake-up process(es) of the other CPU waiting for the termination of the same I/O request

- for asynchronous communication lines, the Station Processor interrupts as above the proper CPU (see par. 3.2.1.).

The synchronization of the execution of the semaphore manipulation primitives themself is obtained via the MC68020 and VME bus standard facilities: a semaphore is constituted by a counter and a synchronization byte. The counter is used to change the status of the semaphore resource, the synchronization byte to allow the semaphore manipulation by one CPU at a time. The semaphore manipulation sequences are:

- set CPU not interruptable

- lock the semaphore, via a TAS instruction

- loop untill semaphore available

- operate semaphore (add one or subtract one to/from counter)

- if counter value ...(high propability case):

    - unlock semaphore, via a write of the synchronization byte

    - set CPU interruptable

so that six instructions and four global memory accesses are in general required to manipulate a semaphore (the granularity of synchronized sequence is such that resource contention cases are very unfrequent).

Finally, the multiprocessor code implementation will allow to do not modify the device drivers (in addition to the user applications as explained above) implemented for a monoprocessor UNIX system:

- the device driver will still use the sleep and wake-up primitives (which in the kernel routines are now replaced by the psem and vsem primitives), which are modified to be consisten with the semaphore kernel implementation and without changing the interface

- the kernel itself will lock the device resource (via a semaphore associated to the device tables) before to run driver routines or interrupt handlers.

3.1.4. The subroutines libraries

All the UNIX System V Release 2 Version 2 subroutines will be provided.

The subroutines of the math library will be modified in order to execute floating point operations fully utilizing the MC68881 scientific coprocessor. In the case that a specific SGM2 installation does not have installed the MC68881, an "illegal instruction" trap will be generated and the interrupt handler wille emulate the execution of the instruction (via a S/W routine): this approach will guarantee object program transportability across dishomogeous SGM2 machines.

Some subroutines will be coded in assembly, in order to increase performances: the set of subroutines to be assembly coded will be supplied later.

3.1.5. The S/W factory

The S/W factory products are obtained on the basis of the AT&T source license of the Motorola 68000 Software Generation System (SGS).

Two of the products available via this license (the assembler and the C compiler) are upgraded to produce object code for the MC68020 interior decore, via the acquisition from Motorola of those products.

The linking editor is modified in order to create an object code file with a layout consistent with the demanded paging requirements and an object program virtual memory map consistent with the SGM2 architecture.

It is under evaluation the convenience to integrated the Motorola C compiler with a compatible product, which exhibits better performances in executing some critical sequences, like the subroutine call prologue.

3.2. The S/W to drive the system peripherals

All the peripherals of the system are connected via controllers/processors connected to the system via the VME bus, the only exceptions being the timer (which is installed on the CPU board itself).

When the system is booted, the initializing CPU polls the boards connected to the VME bus, detecting the types of connected boards and running the proper diagnostics. After that the processors local memory code is loaded from the root file system via a processor EPROM/CPU dialogue. At the end of the local memory code loading, control is passed to it by the EPROM.

To be remarked that via the initialization dialogue the user can select to load a processor code different from that one stored on the root file system, so that the user has the flexibility to implement and test his own VME processor code (see paragraph 3.1.1.).

The processor code is obtained running an utility, which takes input from an a.out file and generates a byte stream loadable into the processor local memory with the structure (CRC etc) expected by the processor EPROM.

In case of programming errors, the processor EPROM is able to download on CPU request the processor LOCAL memory contents into the CPU main memory.

The following paragraphs will describe the following processors and controllers:
- the station processor model 0 (SPO)
- the disc controller.

To the SPO are connected up to 8 communication lines and a printer device.

All the communication lines are asynchronous lines.

The printer device interface is the "Centronics" interface.

3.2.1. The TTY driver via SPO

SGM2 provides a "decentralized" execution of the TTY driver, so that the major part of the line character processing is executed by the Station Processor (SPO) S/W, reaching major performance benefits:

- the number of CPU interrupts, originated by the by the terminal I/O operations, is minimized

- the amount of CPU processing required by the TTY drivewr is minimum

- the response times are improved, in particular when the echoing of input characters is executed by the TTY driver itself and not by the application.

What above is provided by the TTY driver delivered by HISI: the application implementors can define their own line drivers, with the limitation that the system interfaces must not be altered.

The interface between the CPU executed TTY driver and the SP executed line driver is via interrupts and via the SP shared memory, which contains a TTY structure plus some buffers for each TTY line connected to the SP.
The buffers consist of a "circular" character list and an header.
The buffer header provides two pointers to the character list: a charcter producer pointer and a character consumer pointer. The buffer management policy allows to avoid synchronization between the producer and the consumer in accessing the buffer.
The maximum number of characters per buffer is 2 Kbyte.

Case of terminal output operations

Caracter producer: CPU executed TTY driver
Caracter cunsumer: SP executed TTY driver, which polls the output buffer, so that the CPU executed TTY driver must not notify the SP of the transition of the buffer from the empty to the not-empty status.
In the case that the application generates characters in output with a speed greather than the line or terminal speed (or the SP is overloaded), the producer pointer reaches the consumer pointer: one character before that, i.e. when the high-water mark is reached, the CPU executed TTY driver lets the application sleep untill the SP will notify (via an interrupt) the reaching of the low-water mark.

Case of terminal input operations

The interrupt level processing has the following characteristics:

- it is not interruptable

- it must complete the interrupt processing in less than 60 microsec, otherwise the SPO throughput is decreased (either in terms of Kb/s/line either in terms of total throughput)

- it may require to the SPO executive the dispatching of background processing in order to complete the processing of the event(s) originated by the interrupt

- the dispatching of interrupt processing routines is based on the vectored interrupt MC68000 facility.

The background level processing has the following characteristics:

- it is always interruptable

- it does not have the same time constrains of the interrupt level processing, but in any case the background processing will decrease the SPO throughput in terms of cocked character processing

- the background processing is dispatched by the SPO executive, when none interrupt is pending, with the following priority criteria:
-- firstable is run the backgound processing requested by the interrupt processing routines
-- after that is run the background processing requested by the CPU.
-- priorities among lines are handled in a round-robin basis.

To be remaked that in general the processing executed for the "first" line has a greather priority than the processing executed for the second line, etc.

Fortran which is compatible with, and identical to, American National Standard FORTRAN, X3.9-1978.

- PASCAL. PHILON FAST/Pascal follows the ANSI/IEEE Standard Specifications for Pascal, as per the document "ANSI/IEEE770X3.97-1983" .

- BASIC. PHILON FAST/BASIC-M is an advanced dialect of BASIC that is most closely compatible in format with Microsoft's MBASIC.

3.3.1.2. SVS BASIC

3.3.2. DATA BASE

3.3.2.1. C-ISAM

3.3.2.2. UNIFY

3.3.3. Office Automation

3.3.3.1. UNIPLEX II+

3.3.3.2. ALICE

3.3.4. Communication

3.3.4.1. BSC2780/3780

Functionality tests will be executed, connecting SGM2 to an IBM host, to verify the following:

- the package can transmit and receive BSC in ASCII and EBCDIC

3.3.4.3. X.25/X.29/X.28/X.3

Provided via THOMSON-TITN telecommunication S/W products.
A specific VME controller is required (see H/W EPS).

X.25 applicable standard:
    CCITT reccomendations X.40-X.180, november 1980

X.25 provides the following facilities, according to the above CCITT reccomendations:

- permanent and switched circuits
- flow control parameter negotiation (window size and packet size)
- throughput class (assigned at initialization time, not negotiable)
- 7 and 127 window size
- call barring (incoming and outgoing at initialization time)
- closed user groups
- fast select
- one way logial channels at initialization time
- received REJ packet properly processed without application implication, REJ neither generated
- Q and M bits (passed to the application)
- D bit
- DTE to DTE communication without network
- LAP and LAPB at frame level with automatic detection
- secondary X.25 address.

DATAGRAMS SERVICE IS NOT HANDLED.

The X.29 S/W supported facilities (according to the X.3 CCITT recommendation):

3.3.4.5. LAN baseband

The LAN is Ethernet based. The Ethernet cable is connected to the system via a VME controller provided by EXCELAN and running the EXOS 8010 package. The protocols followed by EXOS 8010 are:

Interface level	Corresponding EXOS 8010 Protocol(s)
Application	rlogin, rcp, rsh, rwho (UNIX 4.2BSD) FTP, TELNET (Internet)
Transport	TCP, UDP
Network	IP, ICMP
Data link	Ethernet

Ethernet version 1.0 (sept 80) and 2.0 (nov 82) compatible with IEEE 802.3.

The EXOS 8010 implements protocols defined by the following ARPA documents:

- Transmission Control Protocol - DARPA Internet Program Protocol Specification, RFC 793, Sept 1981

- Internet Protocol - DARPA Internet Program Protocol Speficication, RFC 791, Sept 1981

- Internet Control Message Protocol - DARPA Internet Program Protocol Specification, RFC 792, Sept 1981

- User Datagram Protocol, RFC 768, Aug 1980.

EXOS 8010 provides the following utilities:

- file transfer, according to the "File Transfer Protocol, RFC 765, IEN 149, June 1980

- the virtual terminal utility, telnet, according to "Telnet Protocol", RFC 764, IEN 148, June 1980

4. H/W supported/configurations

4.1. H/W supported

The SGM2 is open to connect every type of device connectable via VME standard controller (providing that the specific S/W drivers are linked to the UNIX kernel via the configuration utility), but specific considerations must be done for the devices required to initialize the system (see following paragraph) and for the hard disks (see paragraph 4.1.1.).

The paragraph 4.1.2. provides the list of controllers and devices fully supported at SGM2 first customer shipment. Fully supported means that the device drivers are distributed and qualified by HISI and that all the information and/or facilities required to fully support the device connection (like the terminfo for terminals) is provided by HISI with the SGM2 system.

4.1.1. The devices to initialize the system and the hard disks.

Assume that the driver routines and the command to format the device are available, the connection of a new type of device able to initialize the system requires extra code:

- programming of the System Controller EPROM.
  The system controller code (the "first boot" program) must do the following:
  -- select the initializing device (see paragraph 3.1.1. for the criteria to be followed in the selection of the init device - an init device is recognized by a specific magic number entered on the first block)
  -- load from the file system the executable program /hisi/loadable and give control to it
- link the new driver routines to the /hisi/loadable program and to the UNIX kernel, including the UNIX kernel loaded when the system is booted from diskette
- Stand Alone utilities: T.B.D.

For what concerns the hard disks themself, the format utility must (in addition to the physical initialization of the disc volume) create the "VTOC" and optionally flag the disc as

6. Transferrability

The transferrability objective is to allow the execution on the SGM2 machine of applications developped for the AX-Superteam machine.

The transfer of applications can be at source level, i.e. object code is not transferrable and a source recompilation is required. In fact, even if the MC68020 guarantees upward compatibility with the MC68010 (the CPU chip used by the AX-Superteam), the format of the a.out file is different on the two systems.

AX-Superteam offers two operating systems:
- Xenix
- UNIPLUS System V Release 0.

The transferrability objective is limited to applications developped for the UNIPLUS environment.

Due to the fact that the UNIPLUS S/W is not a coherent subset of the S/W offered on SGM2 (i.e. UNIX System V Release 2 Version 2), to transfer a AX-Superteam application and compile and run it without any modification on SGM2, it must be recompiled with the following option: T.B.S.

In addition to that, all the commands which are available via UNIPLUS R1.9 but are not included in System V 2.2., are distributed via a specific command directory.

7.2. Data structures

7.2.1. Disc volume layout

- first sector:
  - -- volume description (128 bytes)
  - -- flag of boot disc
    information to load the S/W boot program
- second sector: superblock

* TO BE COMPLETED LATER