THIRTEENTH EUROPEAN ROTOCRAFT FORUM

5.8

PAPER N° 108

AVIONIC DEVELOPMENT MEANS

A COMPLETE INTEGRATED OPERATIONAL SOLUTION

J.P. QUEMARD
ELECTRONIQUE SERGE DASSAULT
FRANCE

September 8 to 11, 1987
ARLES, FRANCE

ASSOCIATION AERONAUTIQUE ET ASTRONAUTIQUE DE FRANCE

AVIONIC DEVELOPMENT MEANS

A COMPLETE INTEGRATED OPERATIONAL SOLUTION

J.P. QUEMARD
ELECTRONIQUE SERGE DASSAULT
FRANCE

ABSTRACT

Advanced avionic systems have reached a very high level of complexity by the large number of functions they offer, the extensive use of new technologies and their high level of integration.

The use of a rigorous methodology supported by wide range of tools, including standardized hardware solutions, is essential to manage the development of such systems.

ESD has proposed an integrated solution based on a wide range of powerful tools.

1 - SUMMARY OF REQUIREMENTS

The development of avionic systems has brought to light a certain number of characteristics over the last few years.

- Considerable growth in computing capacity and memory volume

  Memory volumes have increased more than ten fold in ten years.

  Computing capacity has been multiplied by nearly 8 times during the same period.

- Wide variety of coupling and input-output systems

  Systems have progressed from inter-equipment transfers by means of a few discrete signals and an ARINC bus to complex links involving high-speed digital buses, mass memories and a wide variety of couplers.

- Ever growing necessity of controlling cost and quality

  The development of an avionic systems is becoming a preponderant factor in the cost of a program. Similarly, because of its ever increasing critical nature, it is necessary to guarantee high quality of the resulting product.

- Development means must be flexible, being capable for example of accepting a large number of modifications during development and allowing the reuse of complete software functions.

- Hardware must be based on high-performance technologies in order to guarantee high performance with regard to dimensions, weight and reserve capacity. Hardware must also be as standard as possible to take advantage of quantity production for reasons of cost and to facilitate the reuse of equipment.

2 - SITUATION OF THE PROBLEM

The solution to this problem involves the following three points :

- The implementation of precise and rigorous methodology allowing the control of cost, lead times and quality of the system produced.

- A compact hardware solution covering a wide range of applications and also allowing easy system expansion.

- Software production means organized around high-order languages facilitating the development and reuse of software, these means being based on the methodology mentioned in the first point.

It is by means of these three aspects that a better level can be reached for the production of avionic systems. Following definition at the component level, then at the integrated circuit level and finally at the hybrid circuit level, it is possible to progress to the hardware/software module concept allowing the constitution of the basic building bricks for avionic system construction (for example, navigation module, missile fire-control module, gun fire-control module, etc.)

3 - IMPLEMENTATION OF A METHODOLOGY

The rigorous application of a methodology results in improved product quality and better cost and lead-time control. It also decreases the cost of integration and final debugging as well as later the cost of maintenance. On the other hand, it defines highly formalized tasks which are then amenable to automation. Here, as in other areas of activity, the purpose of automation is to increase productivity, minimize tedious and repetitive tasks and facilitate the achievement of constant quality.

MINERVE Methodology (Formalized since 1976) (PER 79)

This methodology is based on the following three principles :

- Work is divided into phases and stages characterized by clearly defined activities, products and responsibilities.

- The quality of products as well as their cost and lead-time are monitored continuously.

- Modifications are accepted irrespective of the degree of completion in accordance with a unique procedure, the purpose of which is to avoid any possible degradation of software quality.

MINERVE constitutes ESD's software quality handbook. For each project, specific rules are added to this handbook, the whole constituting the project software quality plan.

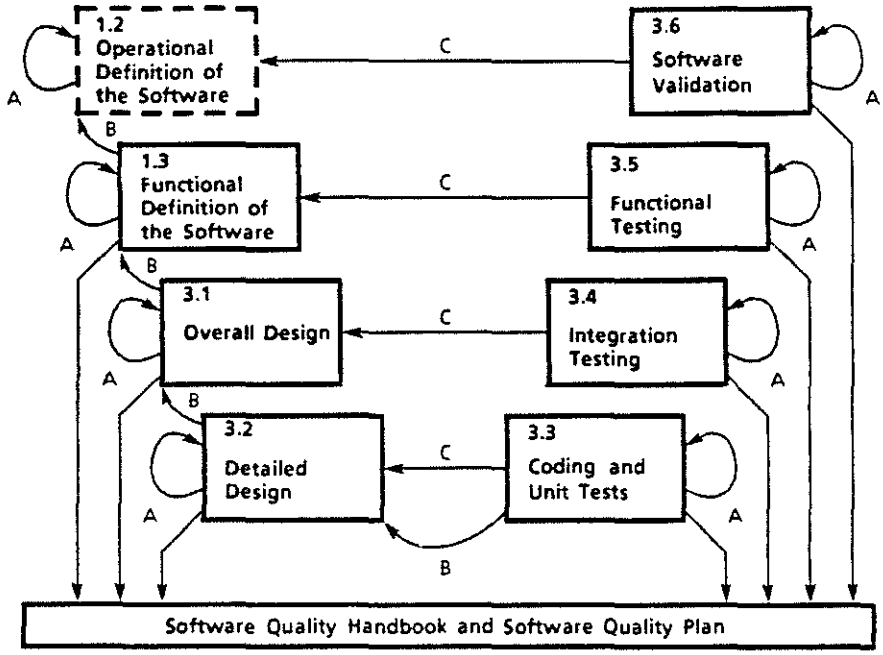
MINERVE makes a distinction between the following three type of check :

- Type A checks : these are checks internal to the product of a stage. Performed by rereading and analyzing documents or code, they consist in checking that the product respects the rules defined in the quality plan (including in particular the application of the standards of documentation, coding, use of tools, etc.). The internal coherence, completeness, readability and accuracy of each product are also checked.

- Type B checks : these are coherence checks between a given level of software description and the former level (code following conception, conception following definition). As in the previous case, this type of check is performed by rereading and reviewing.

- Type C checks : these are program tests performed in four successive stages and with reference to the various software descriptions.

The progress from one stage to the next is subject to obtaining satisfactory results for these three types of check. These various checks are illustrated by the following figure :



Modification control under Minerve constitutes a novelty : by its procedure, it integrates maintenance in the same process as development.

The modification procedure is executed in the following two stages :

- Noting of the modification : the observation of an anomaly or a modification request is formalized by completing a "Modification Sheet" which must be sufficiently precise to determine all the products involved (code, concept, definition) and to evaluate the effect on cost and lead time.

- Execution of the modification : decided in the light of full knowledge, the modification can then be executed, respecting the sequence of the stages concerned, thereby guaranteeing the coherence of all the products. Execution of the modification is recorded on a "Follow-Up Sheet".

4 - HARDWARE

The concerned hardware is divided in the following two categories :

- On-board hardware constituting the processing units of the avionic system.

- Implementation and test tools. This essential hardware allows avionic system debugging and validation.

4.1 - On-Board Hardware

The proposed architecture has been developed with the objective of satisfying several criteria :

- Compatibility with the French standard PMF (Processeur Militaire Français - French Military Processor).
- Choice of a central unit based on standard components.
- Very large-scale integration of components.
- Wide range of standard couplers allowing connection to a large variety of peripherals.
- Allowance for implementation problems from the start of hardware design.

All these characteristics have led to the definition of a set of boards answering the problem.

4.1.1 - Central Unit

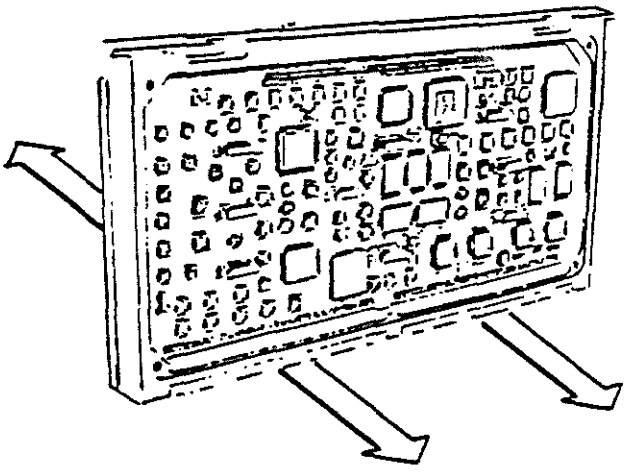
One of the essential characteristics of the central unit adopted is the association of CPU resources and program memory on the same board in order to obtain a completely autonomous calculation capability. In addition, in order to achieve optimum calculation capacity, the following two buses are available :

- an overall bus VME.M,
- a high-speed private bus BPR allowing local extensions (memory, inputs-outputs, etc.).

Assembly on the same board of private domain functions is made possible by the use of large-scale integration packaging technologies (macrohybrid UCMH*).

In addition, the dialoguing system is provided by memories divided between the various resources in the form of data-transfer memories, thereby eliminating the bottleneck of common memories. Finally, the discrete-signal interrupt system is extended by the use of reserved memory spaces resulting in a dynamically configurable interrupt system.

(*) UCMH : Unité Centrale Macro-Hybride (Macrohybrid Central Unit), an ESD registered trademark.



- BPR : PRIVATE RESOURCE EXTENSION BUS
- VME.M : RESERVED FOR THE DIALOGUING SYSTEM

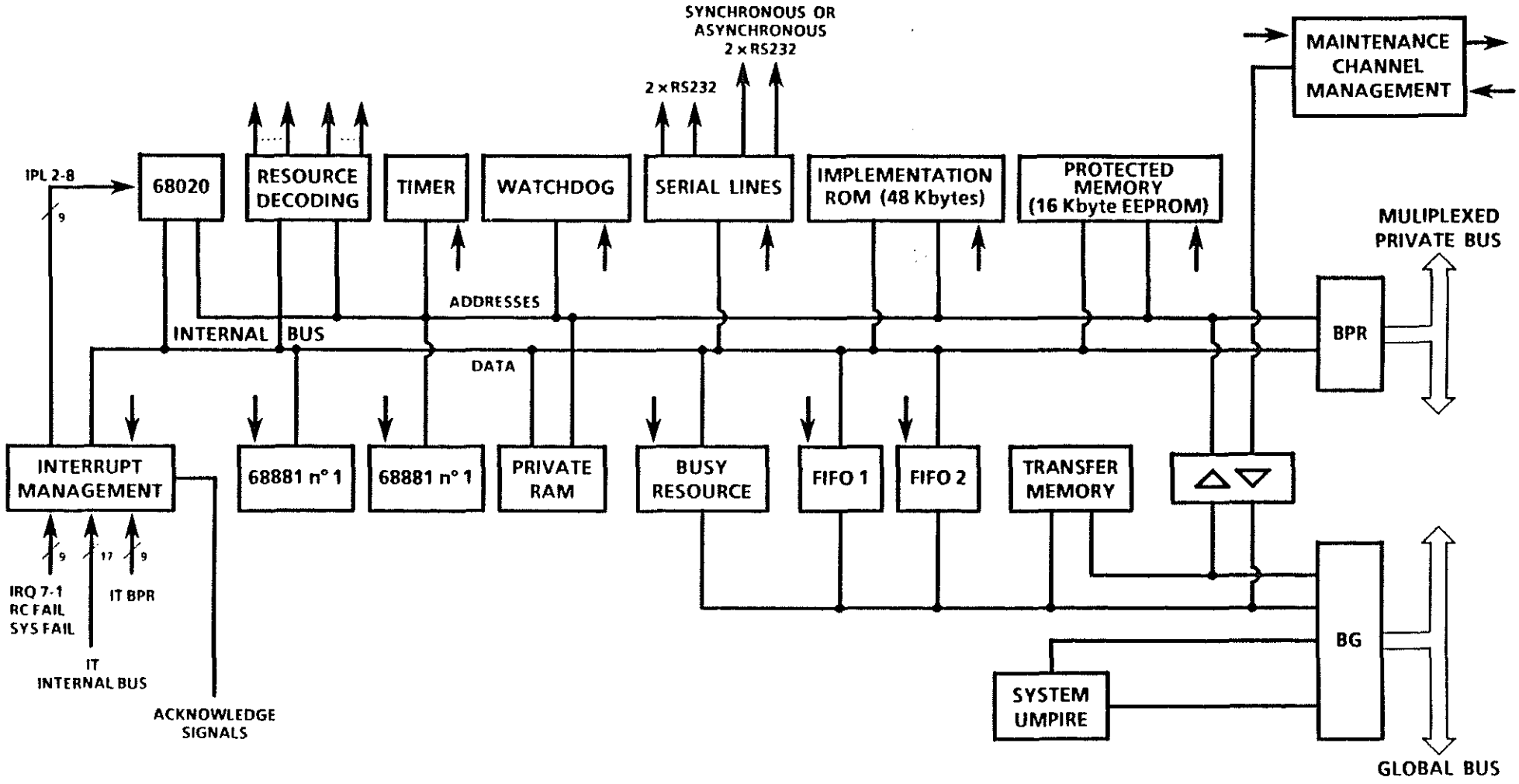
This central unit architecture allows operation in a multimaster or multiprocessor mode without the necessity of modifying the configuration of the central unit board.

The following elements are assembled on the 1/2 ATR central unit board measuring 163 mm × 92 mm × 12 mm :

- a 16.67 MHz type 68020 microprocessor,
- two 16.67 MHz type 68881 coprocessors,
- an implementation 48 Kbyte ROM,
- a protected 16 Kbyte EEPROM,
- a private 768 Kbyte RAM with parity error detection,
- a 256 Kbyte data-transfer RAM,
- four asynchronous lines (RS232 and RS422),
- a real-time clock,
- an interrupt management system,
- inter-resource signalling (FIFO register),
- software execution monitoring by a watchdog system,
- a system umpire,
- a maintenance channel management system.

The capacity of this central unit is evaluated at 1.2 Mips or 300 Kflops with an electrical power consumption of 25 W.

UCMH
FUNCTIONAL DIAGRAM



4.1.2 - Peripheral Boards

A large number of boards may be associated with this central unit board for creating custom configurations :

- Memory extensions
  on private and/or global buses

- Low-speed remote terminal coupler board (BVA)
  types RS232, RS422 and V24

- Digital ARINC bus coupling board

- Multiplexed digital bus coupling boards :
  . DIGIBUS (GAMT 101/GAMT 103)
  . DIGIBUS II (with high-speed link)
  . 1553-B

- Mass memory coupling board (with an HDLC)

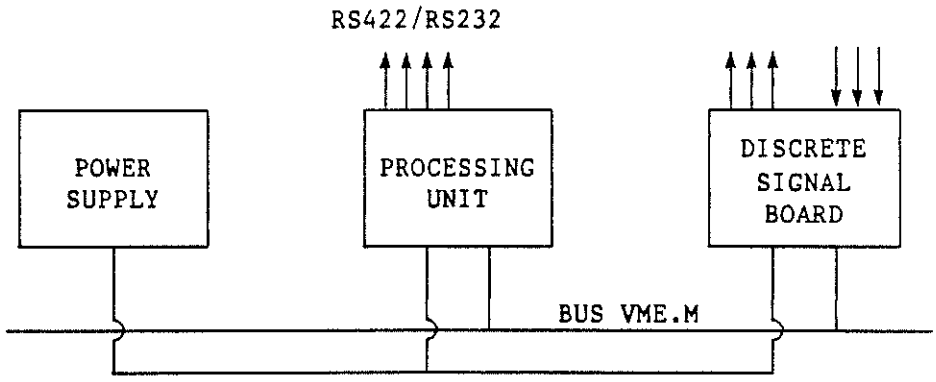
- Winchester disc coupling board

- etc.

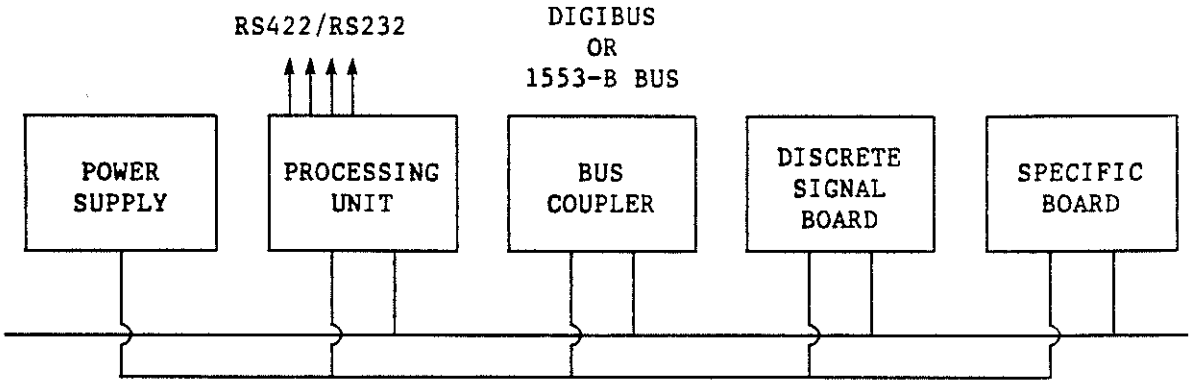
4.1.3 - Possible Configurations

With this set of boards, a wide variety of on-board equipment can be obtained :

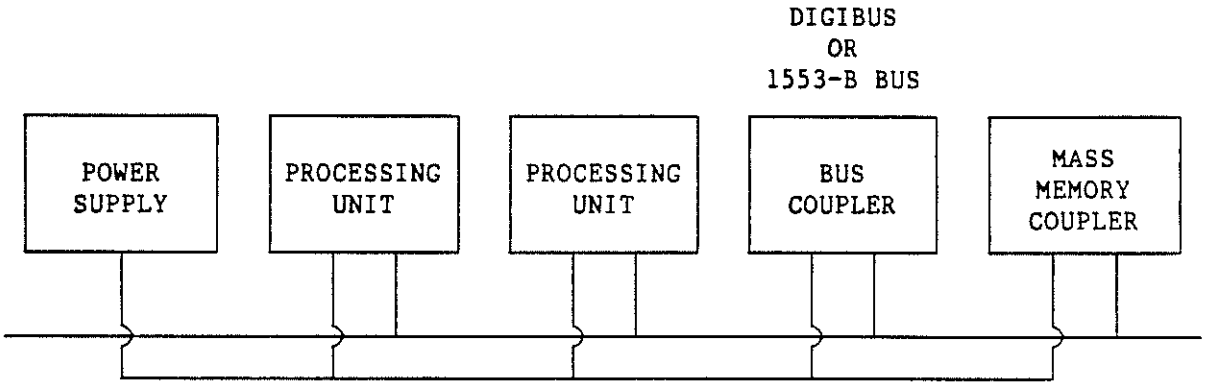
Basic Equipment (Data concentration type)



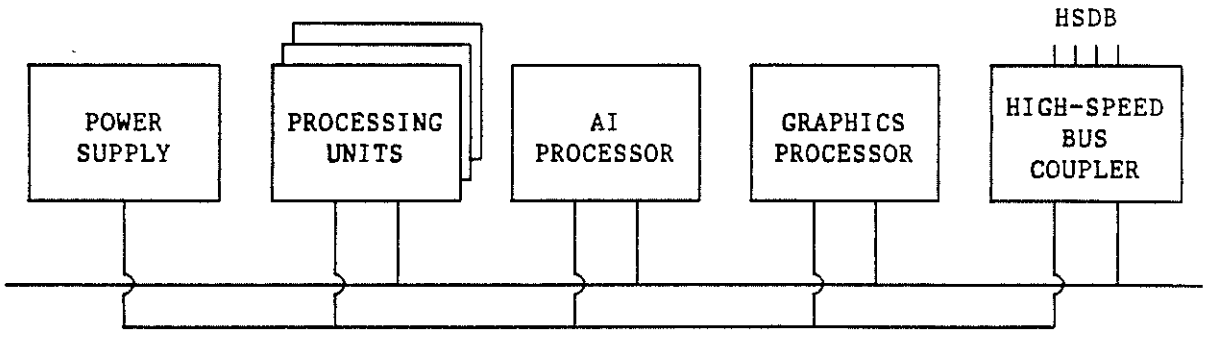
Medium-Size Equipment



Mission computer (Biprocessor)



Advanced Multiprocessor Mission Computer



4.2 - Debugging and Test Tools

In parallel with on-board hardware is proposed a modular range of debugging and test hardware.

- Minimum on-site debugging tool allowing the following elementary operations by means of a console connected directly to the processing unit and embedded debugging monitor :

  . register read/write,
  . memory read/write,
  . program initiation,
  . insertion of program breakpoints.

- The SACRE system is obtained by adding a compatible PC-AT and a logic analyzer to the console, it then being possible to perform the following dynamic debugging operations :

  . trace,
  . dump,
  . software load/unload.

- Finally, by loading IDAS software into the SACRE system, a veritable test tool is obtained by means of the IDAS system.

IDAS [LAM 82] is a software test system which enables its users to automate the test process including testing in real-time environments. In this context, testing implies running a tested program, observing its behaviour and checking its conformity against expected behaviour. IDAS covers a large range of applications since it can be easily adapted to any programming language and to any processor supporting execution of the tested program.

IDAS relies on a high-level **test language** which is used to formalize test operations in terms of test programs. These test programs can be archived and reexecuted allowing **"regression" testing** whenever a modification is made to the tested program.

These test programs can be either compiled or interpreted. They allow the tester to handle in the test programs variables and statements of the tested programs at a symbolic level. Breakpoints and tracepoints are provided. They specify :

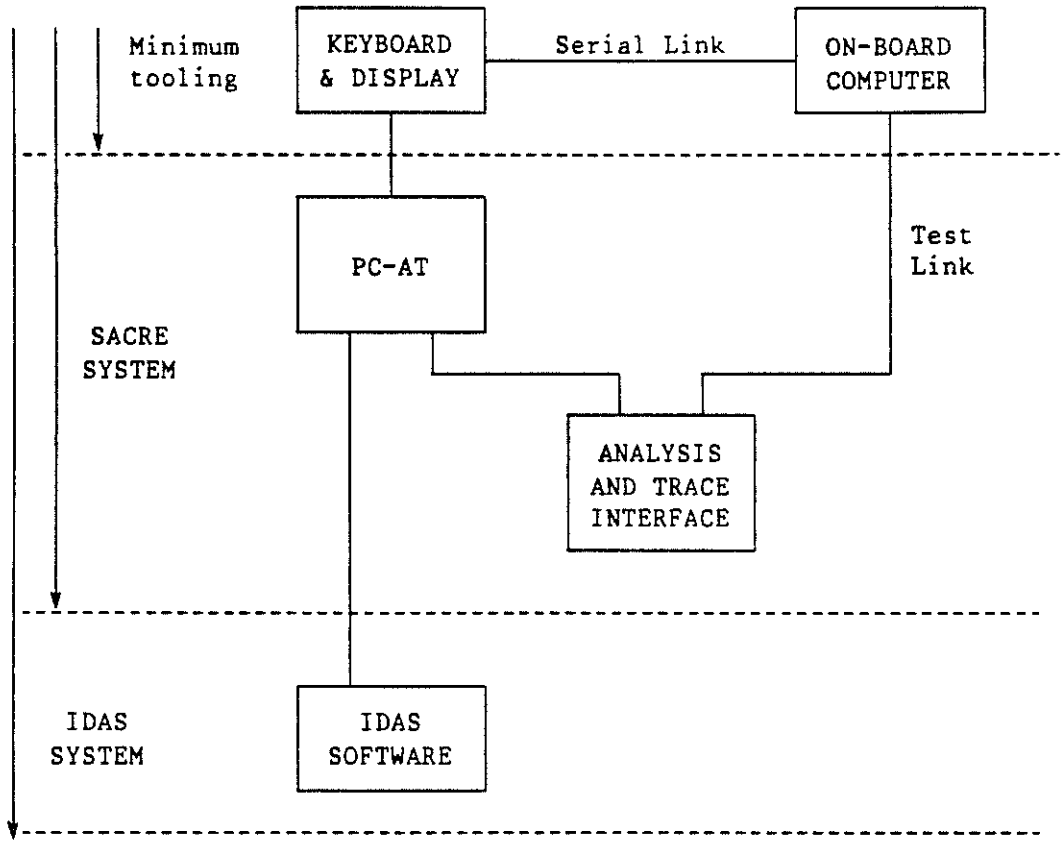
- simple events which occur when a tested program is run (running a statement, manipulating a variable),

- complex events made up of single events by logic combinations.

Instead of being inserted in the code, these breakpoints and tracepoints are loaded into a **hardware interface**, the logical analysis of which is part of the IDAS system.

IDAS is linked to the development software (compilers, link editors, etc.) through a post-processor which provides the IDAS compiler and interpreter with the necessary information (data and statement addresses, data types, etc.). A **symbolic debugger** (SACRE) making use of this information also forms part of the IDAS system.

IDAS is dedicated to real-time application testing and therefore ensures that execution of the tested program **is not disturbed in any way**. This is achieved by providing the SACRE system with a separate machine which supports the test programs and with a dedicated hardware interface which provides the observation capability.



TEST AND DEBUGGING TOOL BLOCK DIAGRAM

5 - SOFTWARE PRODUCTION ENVIRONMENT

A certain number of software tools and components supporting the various activities of software production and test are already available around the PMF. In addition, existing tool extensions are being designed and produced.

5.1 - Existing Tools

5.1.1 - DLAO : Definition de Logiciel Assistée par Ordinateur (Computer-Aided Software Definition [CHE 82])

DLAO is a language and a set of tools constituting a software definition aid system. The DLAO language is based on an entity-relation model. It exhibits a certain number of objects possessing attributes and entering in typical relations. If the language can express the essential backbone of the specification, it also allows formulations in natural language which explain certain semantic aspects. Various interactive tools allow the writing, consultation, modification, filing and reuse of the constituents of a specification. This powerful tool operates on workstations under UNIX System V.

5.1.2 - ENTREPRISE

ENTREPRISE is a set of tools operating on workstations under UNIX, allowing the production of software in language LTR3. It comprises :

- an object manager,
- an editor,
- a paragrapher,
- a syntax editor,
- various compilers (68000, 68020, CMF, etc.),
- an application generator producing the executable binary code.

5.1.3 - *Basic Software*

The following standard basic software is also available for the PMF :

- a modular and high-performance real-time monitor adapted to the 68020 and interfaced for high-order languages (LTR3, ADA).

  This monitor has been specially developed for applications demanding high performance as is the case for all on-board avionic applications.

- software for testing the various hardware components :

  . on-board test (CPU, memory, coupler, etc.),
  . computer acceptance test (functional testing, overall testing).

  This software allows validation of the chosen hardware configuration.

5.1.4 - Test Tools

By means of the IDAS system, testing in a homogeneous and progressive manner is possible :

- on the development means (68020-based work station under UNIX) where IDAS is available and where it uses the 68020 microprocessor of the workstation as the target processor for covering the phases of unit and integration testing.

- for performing integration and validation tests, for which IDAS on SACRE hardware with the real computer is used (see § 4.2).

By its formalization, IDAS tests also check for non-regression, allow overall validation and by the same facility the reuse of software in its hardware environment.

5.2 - Current Developments

5.2.1 - ITI

Within ITI, a group of eight companies in the aeronautical sector (AMD-BA, AEROSPATIALE, CROUZET, ESD, SAGEM, SFENA, SFIM, THOMSON) the following two environments are being produced :

- a system environment allowing system definition and monitoring of system production,

- an advanced software environment based on ENTREPRISE concepts.

All these tools in whose production ESD is participating allow system prime management in the avionic sector with maximum efficiency.

5.2.2 - ENTREPRISE 2

The following extension studies of the present ENTREPRISE environment are also under way :

- multilanguage :
  addition of languages C, PASCAL, ADA

- multiobject :
  complete integration of the DLAO and IDAS tools in ENTREPRISE and addition of a tool for the design of project management tools.

All these current studies will allow the construction in progressive steps of a coherent set of tools while conserving the existing set.

6 - CONCLUSION

The complexity of avionic system development fully justifies the systematic use of hardware building blocks and appropriate data-processing tools for supporting all activities of system life cycles.

Improved productivity can thus be achieved :

- by acquiring homogeneous and efficient tools now,

- by adopting a modular and extensible set of standard hardware for producing solutions which meet requirements,

- by facilitating the reuse of complete hardware/software assemblies in order to minimize the design and integration load of future avionic systems.

BIBLIOGRAPHY

[CHE 82] S. CHENUT-MARTIN and F. DOLADILLE
"DLAO" : An avionic software definition aid system"
AGARD, THE HAGUE, September 1982

[LAM 82] G. LAMARCHE and P. TAILLIBERT
"IDAS" : Test language and associated tools"
AGARD, THE HAGUE, September 1982

[PER 79] J. PERIN
"Avionic software : practical experience of a methodology"
AGARD, OTTAWA, April 1979