NINTH EUROPEAN ROTORCRAFT FORUM

Paper n° 42

SOFTWARE RELIABILITY IN DIGITAL HELICOPTER A.F.C.S.

J.C. DERRIEN

S.F.I.M., 13, Avenue Ramolfo Garnier

91301 MASSY, FRANCE

Abstract

Digital techniques provide a system with high flexibility, but this imposes rigorous new procedures, firstly to produce formal specifications against system design, then to verify software against these specifications. When the software has been integrated within the hardware, the whole system must be validated against operational requirements.

These procedures are mostly automatic, to increase software reliability, and occur for software modifications both during flight tests and for certification purposes. On the other hand, the application of these procedures leads to a quite considerable minimization of the flight-test period, whilst at the same time ensuring constant quality of the software.

Basically, to ensure the quality of a digital A.F.C.S. software during all the phases of a project, the S.F.I.M. approach relies on a simulation of the whole system on computer, in closed loop on a quite accurate model of the aircraft.

September 13 - 15, 1983

STRESA, Italy

Associazione Industrie Aerospaziali

Associazione Italiana di Aeronautica ed Astronautica

SOFTWARE RELIABILITY

IN DIGITAL HELICOPTER A.F.C.S.

by J.C. DERRIEN

S.F.I.M., 13, Avenue Ramolfo Garnier

91301 MASSY, France

---

1 - INTRODUCTION :

This paper is related to the development of a digital automatic flight control system on a helicopter. We shall suppose that during the different phases of such a project, a software methodology has been applied according to the state of the art (see DO 178 and GAM T17 standards) in order to produce quality applications software with a priori reliability.

We shall suppose equally that the hardware meets the system requirements (I/O, memories, safety devices, microprocessors), so that the digital computer itself is considered as being completed and tested. Here also, well known standards of production are used to ensure high reliability of the hardware.

The purpose of this paper is mainly to discuss the means of testing the system, dealing firstly with the software according to a given procedure, and then with the system itself (hardware and software) using specific tools and procedures. The latter will be used until the final integration phase of the system and also during the flight test period, in order to increase the a posteriori software reliability.

However, we shall deal first with the production of the formal specifications document related to the system, for which an original approach is presented. This document is the basic milestone for all the procedures used at S.F.I.M. to build digital autopilots and flight director couplers for various types of helicopters.

As a conclusion, an extension in using these procedures and tools for more exhaustive tests will be presented, so that a step in the theory of software reliability could be made. On the other hand, it is possible to foresee an attempt to match the software certification problems.

2 - FORMAL SPECIFICATIONS :

2.1. - General overview and definitions :

Over many years, S.F.I.M. has developed an approach to take into account the various considerations which lead to an accurate definition of an automatic flight control system that corresponds both to a given aircraft and to the operational requirements of the customer.

This approach is based on the system-state definition, which results from the functional and ergonomic characteristics of a system.

Starting from the means of control (action) and the means of observation (annunciation), that are at the human pilot's disposal, we have defined a set of activated-states. We can observe that a correspondence appears between the action and the annunciation. A main hypothesis for producing the formal specifications is that for a given action of the pilot or of the system itself, in automatic modes, there is a set of given annunciations. We are dealing at the moment with the normal operations of a system :

- normal engagement of a mode, by depressing a button on the system control panel.

- setting of a reference value by the pilot on a potentiometer, before a mode is engaged.

- modification of hardware logics (relays), by moving the control sticks, during fly-through handling.

- modification of a reference value by moving a potentiometer when a mode is in operation, or by beep-trim (kind of transparence).

It is noticeable that all these actions are always sequential, and so far, they can be split and analyzed one by one.

However, this idea can be extended to other operations of the system, which are not considered as normal, but which come from hardware safety devices (e.g declutch in case of emergency, or major failures) or from automatic reversionary modes(e.g generally in case of minor failures of sensors, which can be multiple). These actions are mainly automatic, but they are always annunciated through appropriate means (annunciator, dedicated warnings, E.F.I.S. ...).

All these activated-states are alphanumerically encoded at the very beginning of the system analysis and during writing of the formal specification. There is generally quite a large number of such states, according to the capabilities of the human pilot and of the system.At this point, it is important to find an equilibrium between the complexity of the system itself, because of these numerous possibilities, and the ease of implementing the application software.

In fact, we have now proved for six different systems that a good basis is to associate one activated-state to the annunciation of one normal mode, and as far as possible, when a more complex mode can be operated on more than one control axis at the same time, to define the corresponding activated-state with its related annunciations for this two or three axis normal mode.

The next phase concerns the definition of the compatible activated-states matrix (also called static-matrix), which is related to the operational requirements of the system under development (see figure n° 1). By studying the various combinations which are allowed in this static matrix (1 by 1, 2 by 2, ... (n) by (n), (n) being generally less than or equal to 4 for a complete helicopter A.F.C.S.), we are able to define the number of system-states which represent a set of normal modes (see figure n° 2). One main goal is to minimize the number of system-states, according to the complexity of the system itself, and in this way to simplify the implementation and test of the logic part of the whole system. On the other hand this is going to increase within some limits the complexity of internal logical tasks which have to be performed inside each system-state related software.

... / ...

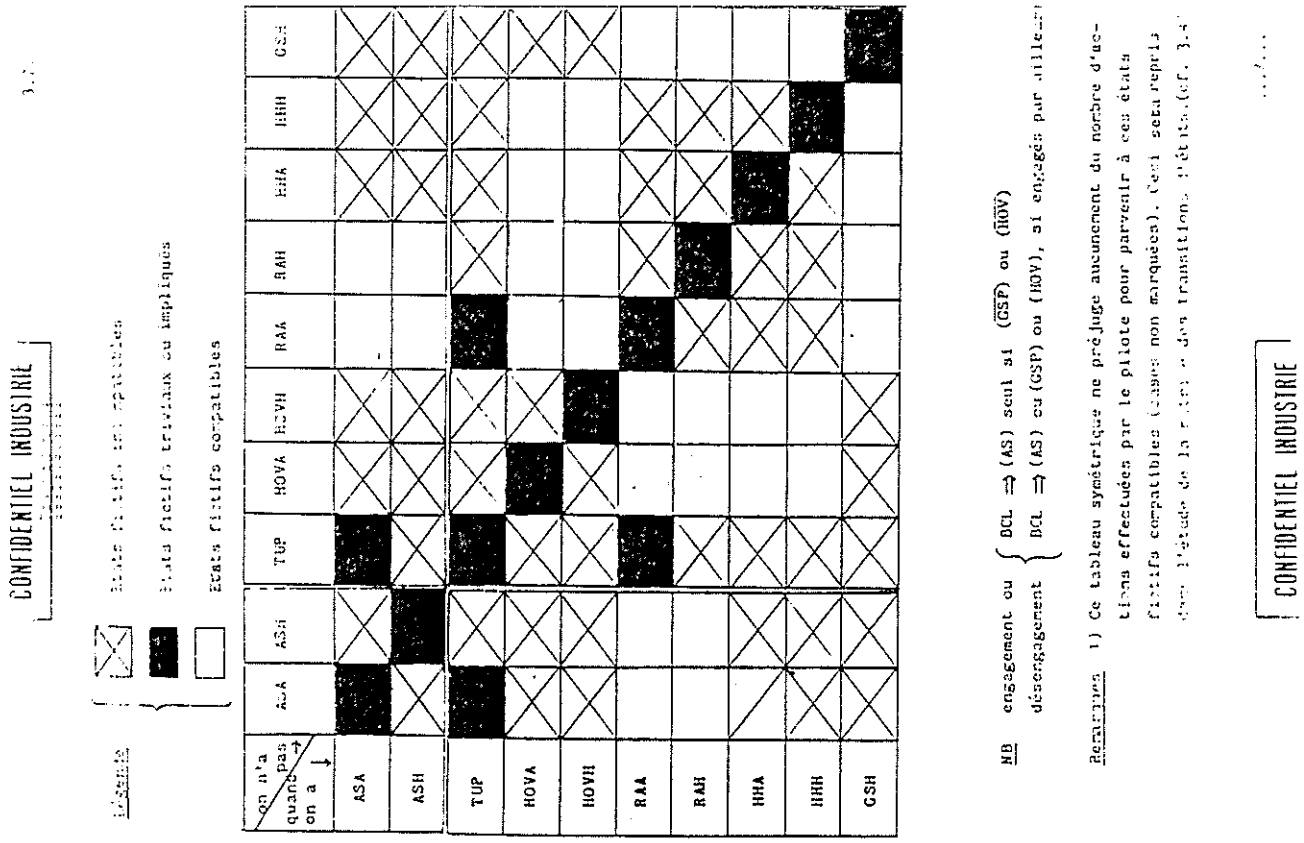
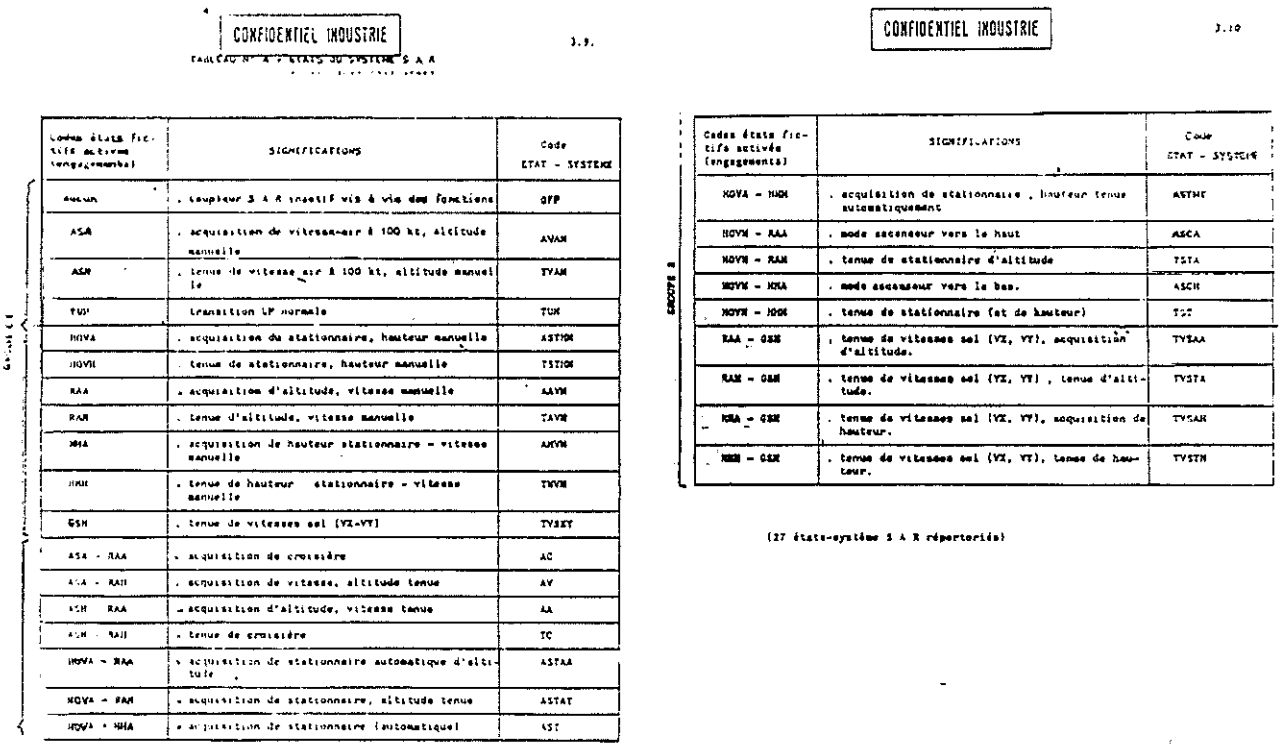


Figure n° 1 : Static matrix of CASM 2100



Codes états fictifs activés (engagements)	SIGNIFICATIONS	Code ETAT - SYSTEME
Aucun	. coupleur S A R inactif vis à vis des fonctions	OFF
ASA	. acquisition de vitesse-air à 100 kt, altitude manuelle	AVAM
ASH	. tenue de vitesse air à 100 kt, altitude manuelle	TVAM
TUP	transition UP normale	TUM
HOVA	. acquisition du stationnaire, hauteur manuelle	ASTHM
HOVH	. tenue de stationnaire, hauteur manuelle	TSTHM
RAA	. acquisition d'altitude, vitesse manuelle	AAVM
RAH	. tenue d'altitude, vitesse manuelle	TAVM
HHA	. acquisition de hauteur stationnaire - vitesse manuelle	AHVM
HHH	. tenue de hauteur stationnaire - vitesse manuelle	THVM
GSH	. tenue de vitesses sol (VX-VY)	TVSXY
ASA - RAA	. acquisition de croisière	AC
ASA - RAH	. acquisition de vitesse, altitude tenue	AV
ASH - RAA	. acquisition d'altitude, vitesse tenue	AA
ASH - RAH	. tenue de croisière	TC
HOVA - RAA	. acquisition de stationnaire automatique d'altitude	ASTAA
HOVA - RAH	. acquisition de stationnaire, altitude tenue	ASTAT
HOVA - HHA	. acquisition de stationnaire (automatique)	AST

Codes états fictifs activés (engagements)	SIGNIFICATIONS	Code ETAT - SYSTEME
HOVA - HHH	. acquisition de stationnaire , hauteur tenue automatiquement	ASTHT
HOVH - RAA	. mode ascenseur vers le haut	ASCA
HOVH - RAH	. tenue de stationnaire d'altitude	TSTA
HOVH - HHA	. mode ascenseur vers le bas.	ASCH
HOVH - HHH	. tenue de stationnaire (et de hauteur)	TST
RAA - GSH	. tenue de vitesses sol (VX, VY), acquisition d'altitude.	TVSAA
RAH - GSH	. tenue de vitesses sol (VX, VY) , tenue d'altitude.	TVSTA
HHA - GSH	. tenue de vitesses sol (VX, VY), acquisition de hauteur.	TVSAH
HHH - GSH	. tenue de vitesses sol (VX, VY), tenue de hauteur.	TVSTH

(27 états-système S A R répertoriés)

Figure n° 2 : System-states set of CASM 2100

The same problem occurs for reversionary modes and their related annunciations, and also for safety and transparent modes. They can be considered as normal modes themselves or as sub-modes included in their respective normal modes. So this notion is relative, but quite useful for simplification of the implementation and also for testing of the system. This goal of minimizing the number of system-states is more or less complicated to achieve, according to the complexity of the system.

For instance, if we observe figure n° 3, we can see that a very simple system can have only 4 system-states, but this number can grow rapidly to 78 for a more sophisticated one. Nevertheless this number of system-states can be taken as a good measurement of the complexity of the system.

Having defined the activated-states and the system-states, it is then possible to define the transition matrix (also called dynamic matrix) which shows all the transitions of which the system is capable from an initial system-state to a final one, when an activated-state is generated (see figure n° 4). It is sometimes possible for a very decoupled system, to have a transition matrix per axis. This simplifies the implementation and the test and reduces the EPROM volume by minimization of tables of labels.

Finally, to each system-state, which has been alphanumerically coded, we can associate an accurate definition of the tasks which have to be performed acquisition of the inputs, logical and analogical conditioning, control laws, transmission of the outputs, annunciations ... This completes the formal specifications document.

We have just presented the way in which we analyze the logical and functional part of an automatic flight control system, in relation to the operational requirements. We are now able to present the way in which we define all the tasks that will be implemented in the system itself for each normal mode, as soon as they have been defined.

2.2. - System specifications related to the normal modes :

S.F.I.M. has developed on a PDP 11/70 computer a whole simulation of a total automatic flight control system for helicopter, which is related to the type of aircraft to be fitted and for which we have all the aerodynamic data. This type of closed-loop simulation is produced for each kind of system. The FORTRAN IV plus, high order language is used. It increases the reliability of this simulation model and on the other hand, it facilitates the optimization of all the control laws, which are an important part of the system itself.

This means that the entire helicopter environment has to be simulated, from the sensors (including their noise characteristics, drifts, bias, ...) to the actuators (with their non-linearities), and finally the flight control system itself (autopilot, coupler, navigation system).

This simulation tool has various capabilities : introduction of failures on sensors and actuators, wind shear and gust simulation, 3 D sea-state for Navy systems. On the other hand, the transition matrix and activated-states, related to the system in development, are programmed to be capable of easy and fast generation of evolution profiles that are compatible with operational requirements (see figure n° 5). It represents the possible operations of the system FDC 155 (modes, sequences of modes).

... / ...

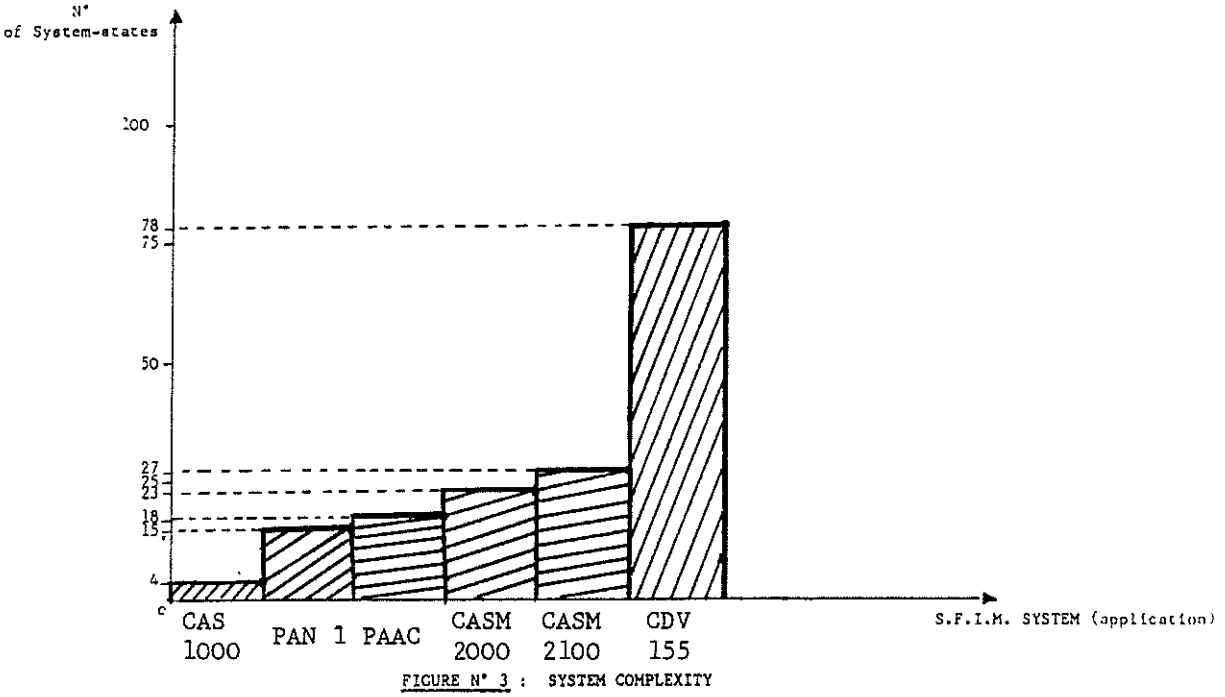


FIGURE N° 3 : SYSTEM COMPLEXITY

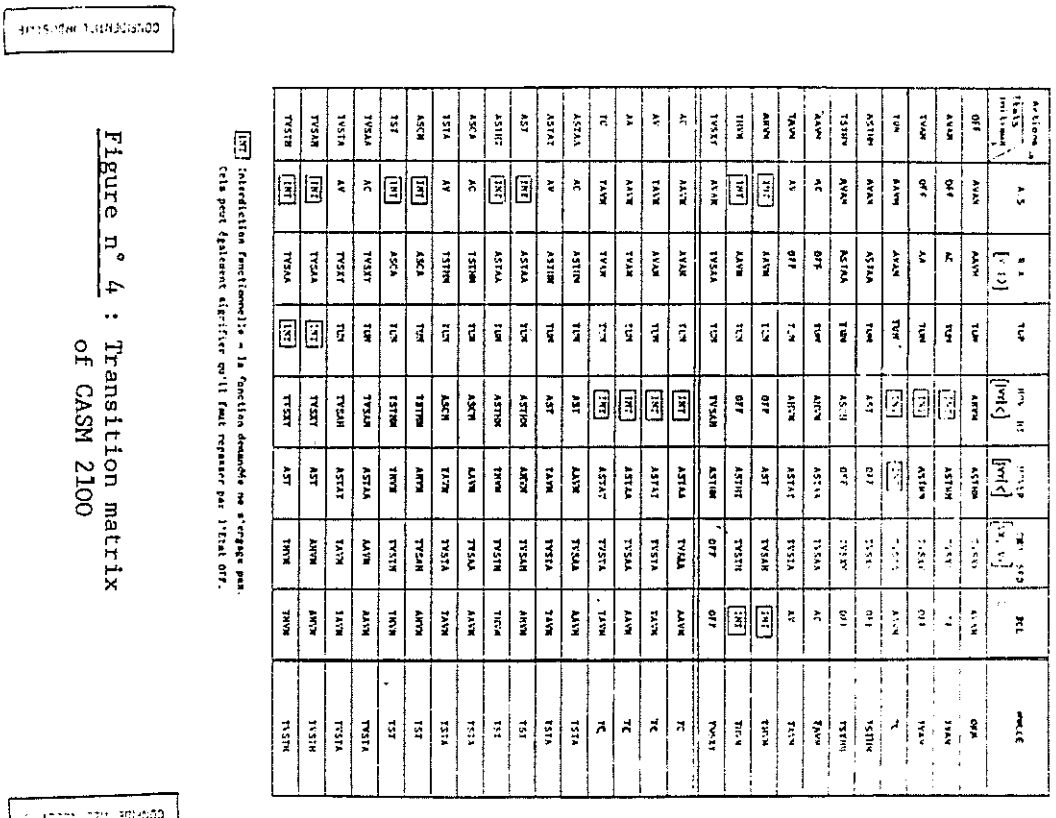
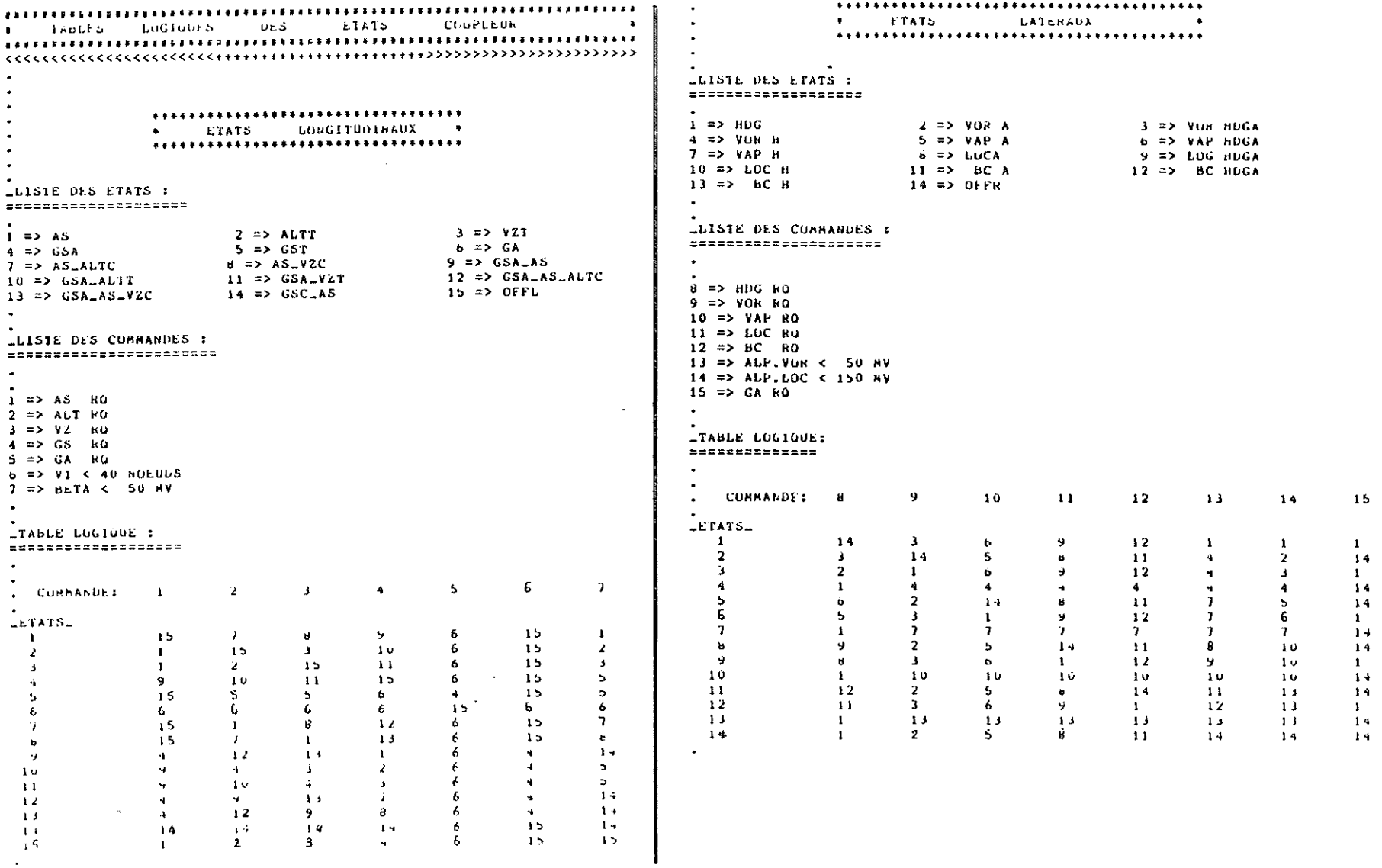


Figure n° 4 : Transition matrix of CASM 2100



```
*****************************************************************
*     TABLES    LOGIQUES     DES      ETATS      COUPLEUR         *
*****************************************************************
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

                  ********************************
                  *     ETATS     LONGITUDINAUX    *
                  ********************************

_LISTE DES ETATS :
====================

1 => AS                  2 => ALTT                3 => VZT
4 => GSA                 5 => GST                 6 => GA
7 => AS_ALTC             8 => AS_VZC              9 => GSA_AS
10 => GSA_ALTT           11 => GSA_VZT            12 => GSA_AS_ALTC
13 => GSA_AS_VZC         14 => GSC_AS             15 => OFFL

_LISTE DES COMMANDES :
========================

1 => AS  RQ
2 => ALT RQ
3 => VZ  RQ
4 => GS  RQ
5 => GA  RQ
6 => V1 < 40 NOEUDS
7 => BETA <  50 MV

_TABLE LOGIQUE :
====================

  COMMANDE:    1     2     3     4     5     6     7
_ETATS_
   1          15     7     8     9     6    15     1
   2           1    15     3    10     6    15     2
   3           1     2    15    11     6    15     3
   4           9    10    11    15     6    15     5
   5          15     5     5     6     4    15     5
   6           6     6     6     6    15     6     6
   7          15     1     8    12     6    15     7
   8          15     7     1    13     6    15     8
   9           4    12    14     1     6     4    14
  10           4     4     3     2     6     4     5
  11           4    10     4     5     6     4     5
  12           4     4    13     7     6     4    14
  13           4    12     9     8     6     4    14
  14          14     3    10    14     6    15    14
  15           1     2     3     4     6    15    15

                  ****************************************
                  *     ETATS          LATERAUX          *
                  ****************************************

_LISTE DES ETATS :
====================

1 => HDG                 2 => VOR A               3 => VOR HDGA
4 => VOR H               5 => VAP A               6 => VAP HDGA
7 => VAP H               8 => LOCA                9 => LOC HDGA
10 => LOC H              11 =>  BC A              12 =>  BC HDGA
13 =>  BC H              14 => OFFR

_LISTE DES COMMANDES :
======================

8 => HDG RQ
9 => VOR RQ
10 => VAP RQ
11 => LOC RQ
12 => BC  RQ
13 => ALP.VOR <  50 MV
14 => ALP.LOC < 150 MV
15 => GA RQ

_TABLE LOGIQUE:
===============

  COMMANDE:    8     9    10    11    12    13    14    15
_ETATS_
   1          14     3     6     9    12     1     1     1
   2           3    14     5     6    11     4     2    14
   3           2     1     6     9    12     4     3     1
   4           1     4     4     4     4     4     4    14
   5           6     2    14     8    11     7     5    14
   6           5     3     1     9    12     7     6     1
   7           1     7     7     7     7     7     7    14
   8           9     2     5    14    11     8    10    14
   9           8     3     6     1    12     9    10     1
  10           1    10    10    10    10    10    10    14
  11          12     2     5     6    14    11    11    14
  12          11     3     6     9     1    12    13     1
  13           1    13    13    13    13    13    13    14
  14           1     2     5     6    11    14    14    14
```

Figure n° 5 : Transition matrix of FDC 155 (simulation)

When the optimization phase for the whole system has been completed on the computer, a complete set of values, which are the nominal adjustments of the automatic flight control system, can be generated and automatically formatted and transferred into a special file, which is used later for generation of DATA EPROM.

It is also possible to estimate the sampling rate of all the algorithms, the format of variables and constants (number of useful bits) compatible with the performances in closed-loop, that are requested by the customer.

At this stage, all the alphanumeric codes of all the dynamic variables, constants, inputs, outputs of each FORTRAN software modules are fixed. Thus, the overall architecture and connections between modules of the application software are fixed and will be found to be the same in the host computer, which is used for real-time software development of the system.

On the other hand, all the constants associated with each module, which are related to the type of helicopter that we intend to fit with a system, are part of the formal specifications. This represents the starting point of the optimization phase that will take place during the flight test period. Moreover, it is possible to associate a set of tests with each module or task that is specified this way. These tests are organized in files which can be easily generated by running the complete non-linear simulation, placing the system in a " real " situation, in so far as our simulation is a closed-loop simulation. These tests are memorized on magnetic media such as floppy disks or magtapes.

It is important to point out that we do not need a very accurate model to produce a formal specification of this kind. If, as we hope, this is the case, our approach will minimize the flight test period, as far as optimization of all the control laws is concerned. For a highly sophisticated system this is a huge advantage.

On the other hand, this formal specification is totally independent from the real-time software language which will be used in the digital A.F.C.S. (assembly, specific macro-assembly, Pascal, ...).

3 - VERIFICATION PLAN :

This phase follows the coding and preliminary testing of the applications software on the software development host computer. It is supposed, at this stage, that this software, which is produced according to a given methodology, meets the formal specifications and it has been debugged using white box test procedures at module level. The verification plan is prepared independently, using the formal specifications described previously, in order to help to increase the reliability of the applications software. No hardware test is involved during this phase.

3.1. - Verification of the logical and functional part of the system :

Given that the listing of the source code has to be understandable and easily readable, it is possible to read through all the tables and labels (system datas, transition matrix, coding of system-states and activated-states,...) to verify that they are totally compatible with the formal specifications.

It is equally possible to verify that each system-state label is correctly associated with its tasks, which have been named, and that the inputs/outputs of each task are all accurately named and connected.

3.2. - Verification of tasks or modules :

In the formal specifications, we have seen that a set of tests was associated with each task or module. These tests have an operational meaning in the way they place each module of the system in a " real " situation, as we have seen previously (§ 2.2). On the other hand, concerning the system, we have implemented the same overall software structure on the PDP 11/70, which provides the closed-loop simulation environment, and on the real-time software development host computer, which, in our case, is a TEXAS 990-10.

Our procedural philosophy is as follows :

a) from the results file of the closed-loop simulation, we create a software input data file through an analog-to-digital conversion model, which organizes these data in the format expected by the real-time applications software, in its acquisition block related to the hardware inputs (aircraft wiring).

b) This file is then transferred onto the software development host computer.

c) The operational software runs using these data, and creates an output data file containing inputs/outputs and some internal variables of each module or task being in the verification process. This is a kind of grey-box test procedure.

d) This output file is then transferred back to the PDP 11/70 computer and decoded from a fixed digital representation into physical units.

e) Finally, the expected results of the tests, which had been previously stored in the PDP 11/70 computer during the test generation,are automatically compared with the respective outputs coming from the real-time applications software. The result is represented on a graphic console.

All the differences are analyzed and errors are corrected. Thus, the reliability of the software increases. To illustrate this procedure, let us take an example (see figure n° 6 : formal specification of the " TRVI " task, part of FDC 155 system).

The most important part of such a procedure is to define the minimum number of tests that is necessary to go through all the paths of a given task. This represents an important part of the formal specifications. For instance, for the " TRVI " task, we have generated a dynamic test on the TEXAS 990-10 host computer. This test can examine several paths (here not all) and many times the same path, but with a lot of different values on the analog inputs (see figure n° 7) within their total variation range (VI $\in$ [42 m/s ; 60 m/s] and GAMAXC $\in$ [- 1.5 ms^{-2} ; + 1.5 ms^{-2}]) and for various combinations of discrete inputs (VLGAXC, CTRAIN).

Such tests are very easy to generate with our automatic procedure and the results can be analyzed according to the simulation results and the formal specifications. Here (see figures n° 7 and 8) we are interest in testing the continuity and the evolution of (VIX) and its derivative (VIP), variables which are used in some control laws, in case of non-validity of the compensated longitudinal accelerometer (GAMAXC, VLGAXC).

3.1.9. Airspeed conditioning = 'TRVI' task generates the following informations, starting from the measured indicated airspeed, the compensated longitudinal acceleration and two logics :

- longitudinal and lateral airspeed with their validity flags
- longitudinal airspeed derivative
- two flags for the reversionary VI sensor
- three logics about the VI, to define some validity thresholds for the control laws of the helicopter.

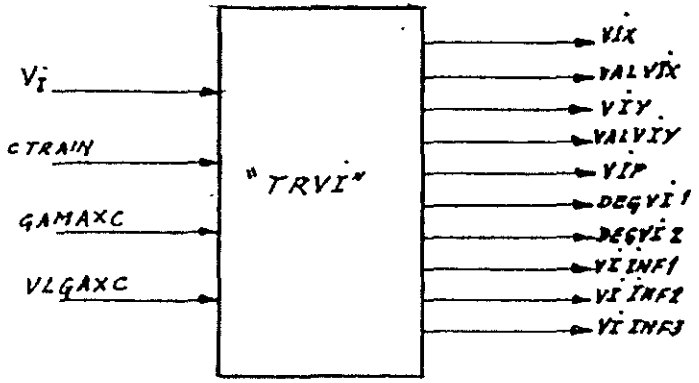


fig: 23.

FIGURE 24 "TRVI" : algorithm

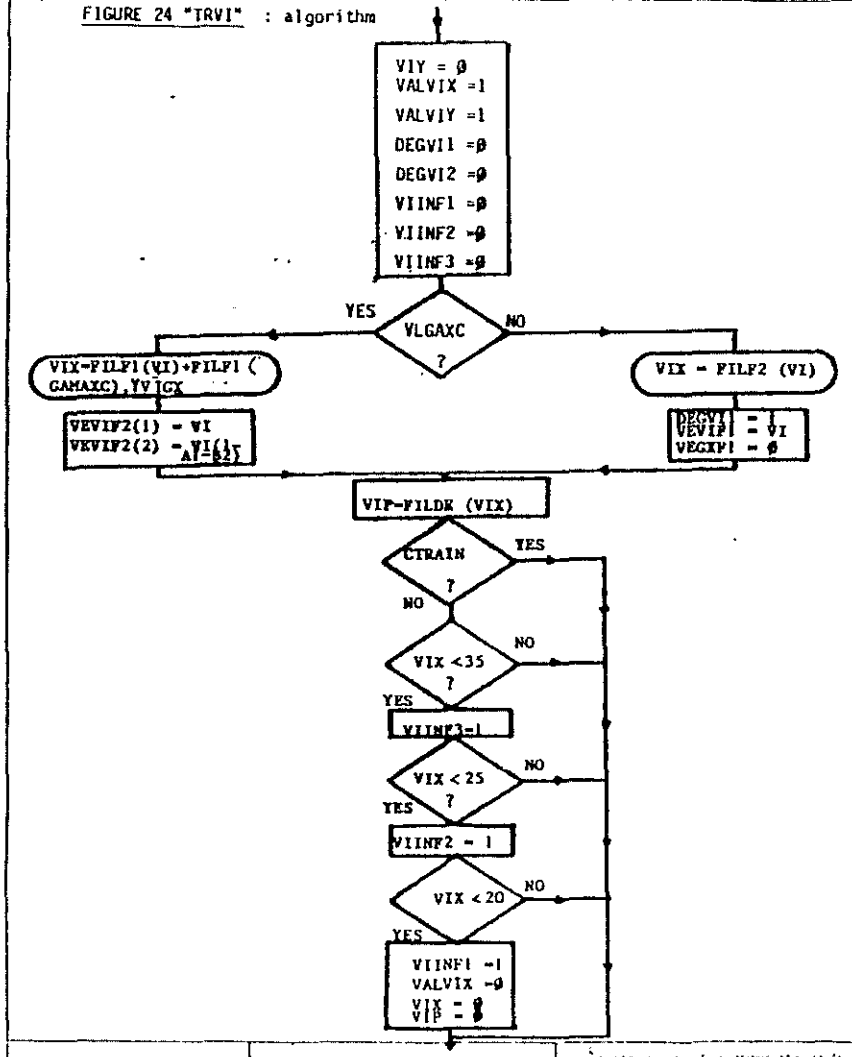


Figure n° 6: Formal specifications (TRVI task)

On the other hand, we have tested the generation of flags which are used for some reversionary modes (see figure n° 8).

3.3. - Verification of modes and sequences of modes :

As we have implemented the transition matrix associated with the system under test in the simulation environment, we are easily able to generate a dynamic test in order to verify the validity of a mode or even of a sequence of modes. The procedure is exactly the same as the one presented before. The test is more a black-box test, as we are at an upper level in the structure of the applications software, but we could make it more precise, like a grey-box test, if we so wish. It would be only a matter of analyzing many more data and dealing with much more information.

To illustrate this comment, we give an example of such a test which has been performed on the PAN 1 system (see figures n° 9, 10, 11, 12). For each of figures n° 9, 10, 11, on the upper part we find the dynamic test issued from the simulation and on the lower part, the result of the same test which has been run on the operational software, in the TEXAS 990-10 host computer.

We can observe the evolution of the attitude references in pitch and roll (see figure n° 9) for a succession of transparent modes. We can also see the evolution of the direct terms of the pitch and roll control laws (see figure n° 10) and the evolution of the derivative terms of the pitch and roll control laws (see figure n° 11), for the same succession of transparent modes. On this last figure, we can notice that a discontinuity problem arises in the calculation of the derivative terms in the operational software, which in this case had no effect on the helicopter itself. Figure n° 12 shows the succession of transparent modes used for this test, and the acquisition of GAMMAY measurement (lateral acceleration), as an example.

4 - VALIDATION PLAN :

Until now, the hardware was not involved in our procedures and dynamic tests were not performed in real-time. On the other hand the hardware and software inputs/outputs interfaces with the outside world (the helicopter wiring itself) had not yet been tested. The validation phase is then necessary to complete integration of the software as well as possible and to improve the operationality of the whole system, implemented within the target computer. However, it is supposed at this stage, that all the hardware resources have been correctly produced and tested independently, above all real-time operation and safety devices.

4.1. - Real time test bench : SILENE

The SILENE configuration (see photo n° 1) has been especially developed by S.F.I.M. to study digital A.F.C.S. for helicopters in real-time. A sharp snalysis of the operational software of the system in test can be made by stimulating compatible evolution profiles of the helicopter in real-time, taking into account sensors, perturbations and servo models. It is then necessary to generate a magnetic tape with all input data files, concerning each mode or each sequence of modes related to the system in test, and coming from the simulation environment.

We use the same procedure as the one presented for the verification plan (cf. § 3), but the number of tests can be greater, in order to stimulate the system automatically with as many operational profiles as possible.

The SILENE configuration is driven by two microprocessors. One is used to stimulate all the specific hardware interfaces to the A.F.C.S. with values from input magtape. The other one is used for acquisition of all the results, formatted on the output magtape. A conversational programme with the human operator has been specially developed, to facilitate the testing operations. The output magtape is then data processed on the PDP 11/70 computer. The results are observed on Benson drawings and statistics can be made automatically. It is possible with such a tool to observe a large set of selected internal variables of the applications software, running in real-time in its digital computer.

4.2. - Application of this validation procedure :

We would now like to present a typical test that has been made for the PAN1 system, in order to show the differences between the verification phase and the validation phase, according to the way these phases have been defined previously.

Figure n° 13 shows one test from the simulation file on PDP 11/70 (input = pitch attitude TETAPA, outputs = pitch attitude command DTETAC, pitch attitude reference TETARF, integration of DTETAC) together with application of the test to the operational software for verification. We can notice the effect of the sampling rate and the resolution on the DTETAC output.

Figure n° 14 shows the same test performed on the target computer through the SILENE real-time test bench, for validation purposes. We can notice the poor evolution of DTETAC and the resultant poor evolution of TETARF, which is due to poor conditioning of one input in the PAN1 hardware interface. On the other hand, we can see that the TETAPA signal generated by SILENE, is a little more noisy than the corresponding signal in the simulation file. This is a normal consequence of introduction of a real-time bench in the test loop.

5 - CONCLUSIONS :

S.F.I.M. has developed all these tools and automatic procedures in order to increase the reliability of the applications software for helicopter A.F.C.S. These procedures, verification and validation plans, are commonly used, even during the flight test period, in order to ensure constant quality of the software. We have proved that, for major modifications in the operational software of the CASM 2000 system during flight testing, very few tiny errors had been pointed out.

By using such procedures, we can ensure a high reliability of the software but, although they are applied through automatic tools, it can take quite a long time to integrate major modifications with sucess. At the moment, we have applied these procedures to five different systems, three of which have already completed all their flight tests, and we have obtained good results.

From the point of view of certification, we think that our procedures are well adapted and can be used to demonstrate critical software in a digital A.F.C.S.

Moreover, we think that it could be possible to use the SILENE bench in order to test the system extensively. A great amount of tests, representing the various configurations in which the system will operate as soon as it is placed in the helicopter, can be easily generated throughout our simulation. This represents a near exhaustive tests method, which is based on a combined utilization of the formal system specifications document and of the simulation environment.

We are working at the moment on this method which could demonstrate the quality class of an application software. The tests are generated throughout our simulation according to the possible evolution range of all the inputs, taking into account the operational profile of the system itself. This notion permits us to elaborate tests proportionally to the ocurrance of the system-states, during a typical mission of the system. The reliability of the software is then defined as :

$$R(n) = \sum_{i=1}^{N} \left\{ P_i \left[\sum_{j=1}^{n_i} \left(\frac{1 - Y_{ij}}{n_i} \right) \right] \right\}$$

$$\left| \begin{array}{l} n = \sum_{i=1}^{N} n_i \\ \sum_{i=1}^{N} P_i = 1 \end{array} \right.$$

Where :

n	=	total number of tests
Pi	=	°/, occurrance of system-state n° i, during the mission.
n_i	=	number of tests, for system-state n° i
N	=	number of system-states, related to the system under evaluation.
index j	=	test n° j is running, $j \in [1, ni]$
index i	=	system-state n° i is tested, $i \in [1, N]$
Yij	=	run characteristic (if test is correct, Yij = 0).

We know that the choice of (n), number of tests, is a major issue. It has to be quite large (χ^2 test) to provide a reliable measurement of R (n). Some more work has to be done in this area but we think that this approach is of considerable value in the field of software reliability.

REFERENCES :

(1) - Estimating software reliability from test data
Eldred Nelson TWR defense and space systems group

(2) - The art of software testing
Glenford I. Hyers ISBN wiley interscience publication

(3) - TRW series of software technology
Thomas Thayer, Myron Lipsow

(4) - Testing software design modeled by finite state machines
TSUN S. CHOW

(5) - The many facets of quantitative assessment of software reliability
J.C. Rault (IRIA)

... / ...

(6) - Quantitative software reliability models - data parameters : a tutorial
Dorothy Swearingen - John Donahoo

(7) - How to measure software reliability and how not to
Bey Littlewood.

(8) - In search of software complexity
Dr Bill Curtis

(9) - Elements of software science
Halstead M. H.

(10)- A complexity measure
Mc Cabe

(11)- Software modeling studies
M.L. Shooman

(12)- Doc SFIM n° 12 468/S.D.-Ed. 3 - Préétude sur l'estimation de la fiabilité des logiciels.

(13)- Doc SFIM n° 12 377-JCD/NC - Système SAR (SAWARI) - Analyse fonctionnelle du projet.

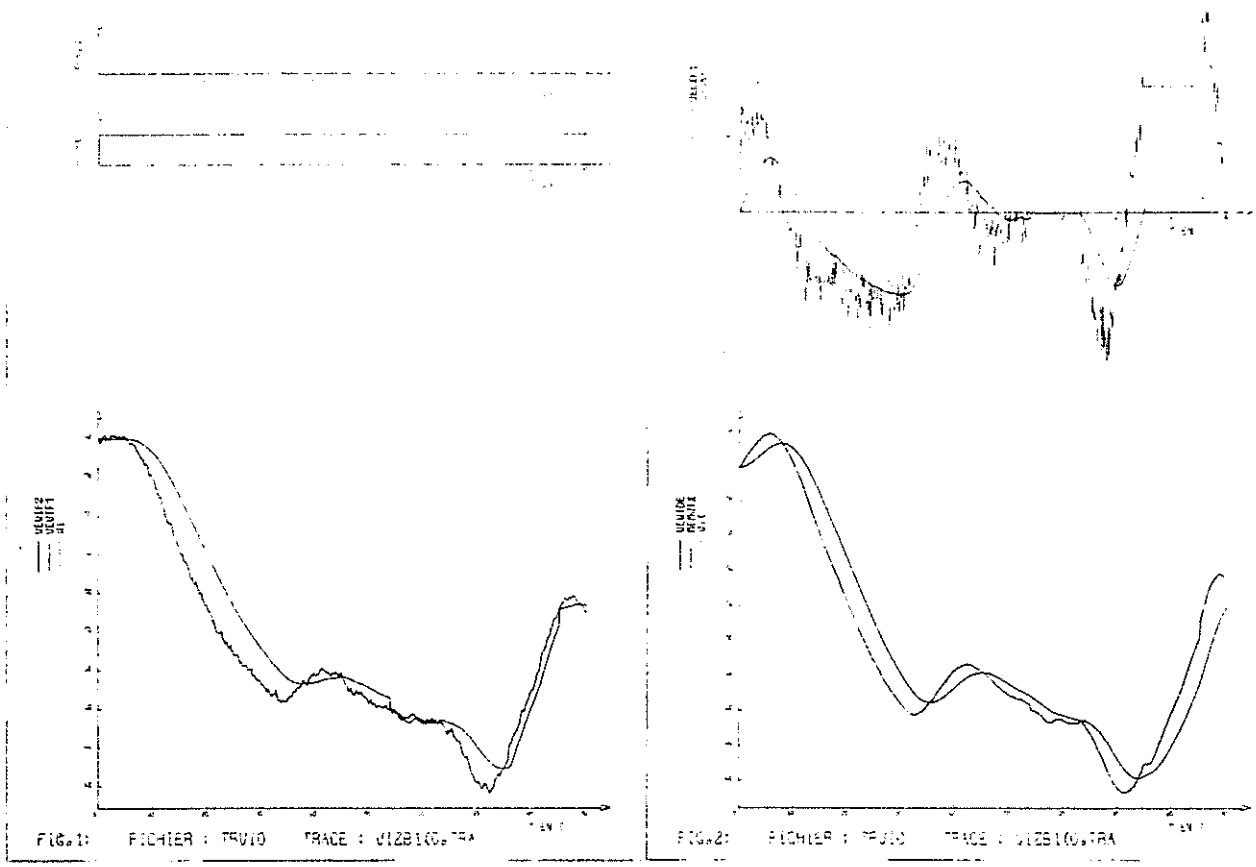


Figure n° 7 : Verification of TRVI Task (FDC 155)

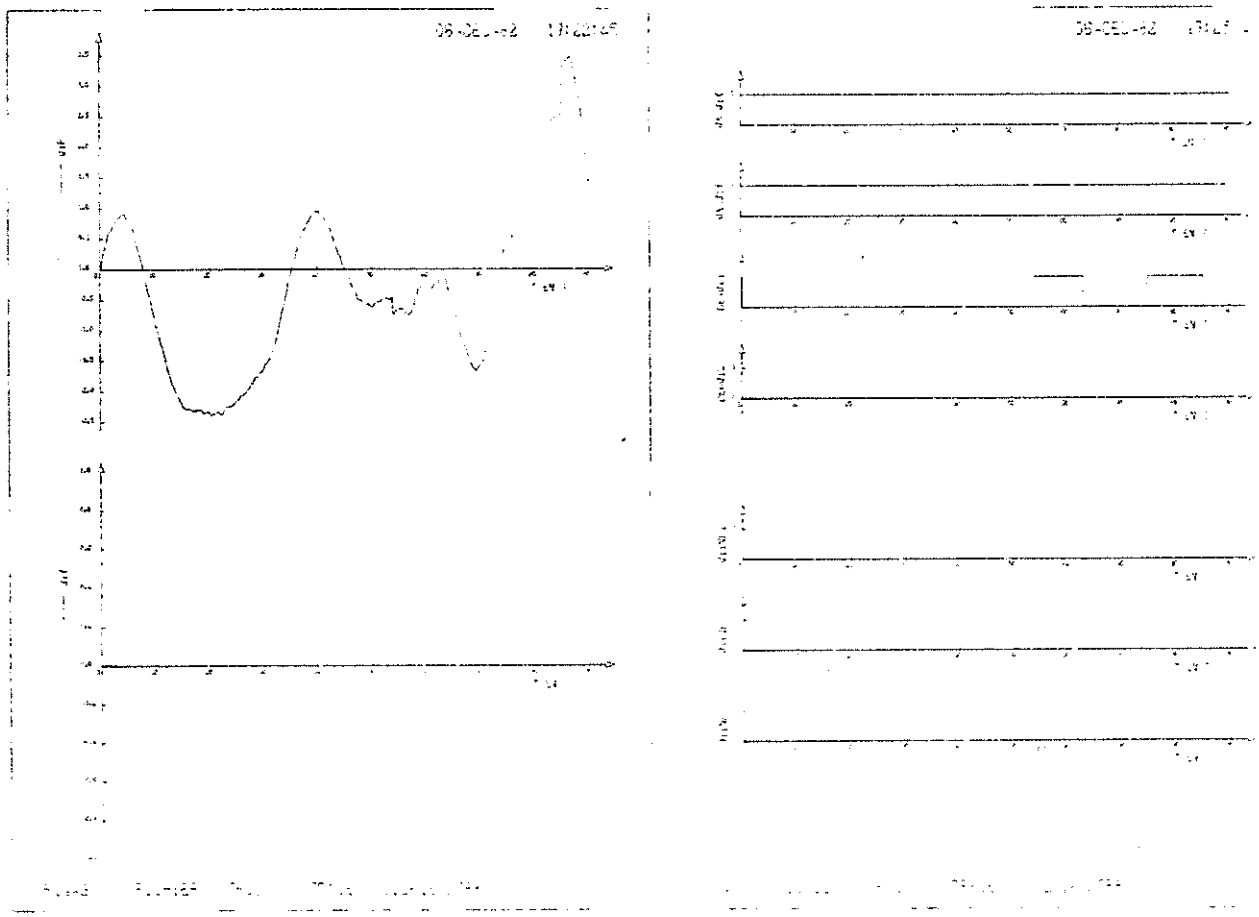


Figure n° 8 : Verification of TRVI Task (FDC 155)

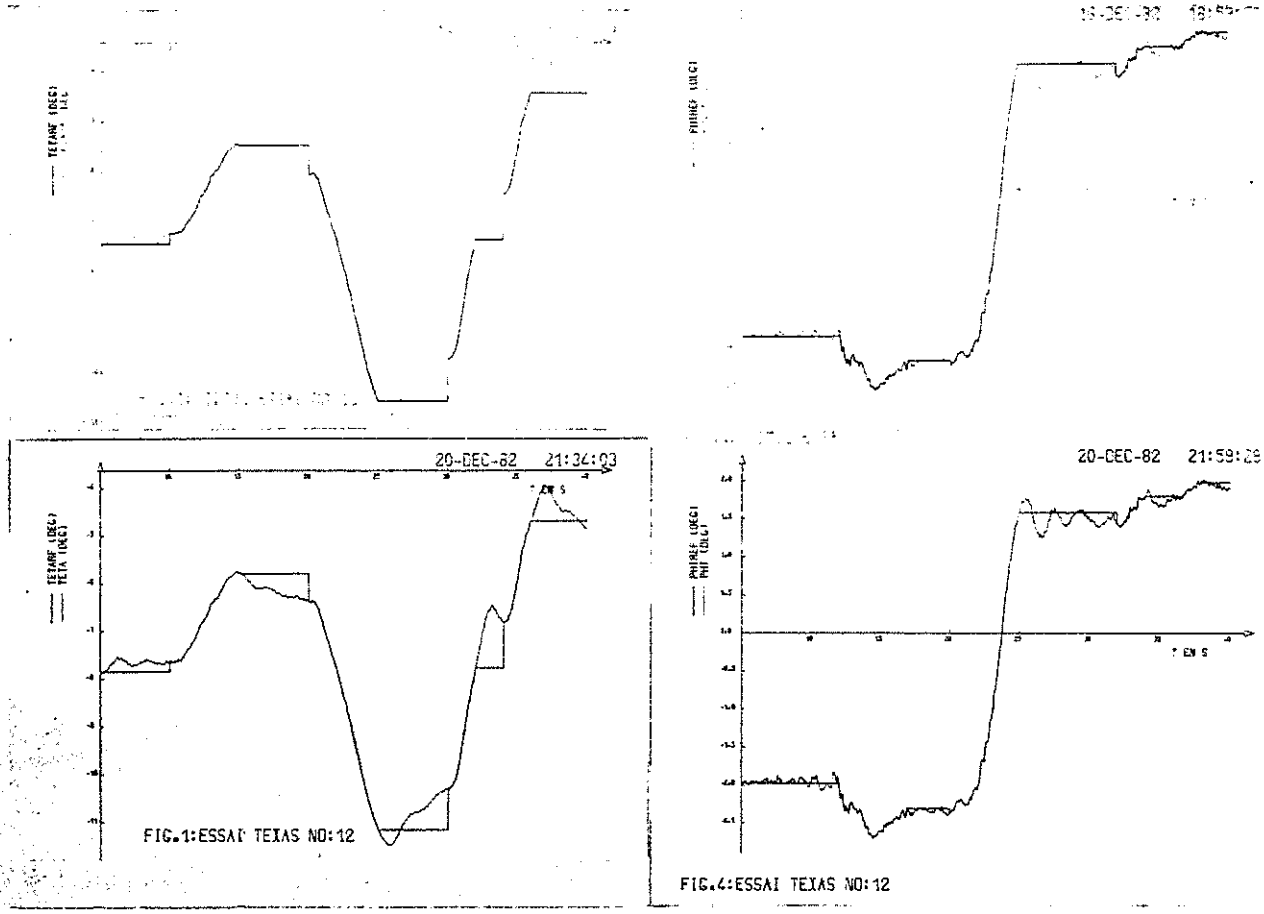


Figure n° 9 : Verification of Sequence of Modes (PAN1)

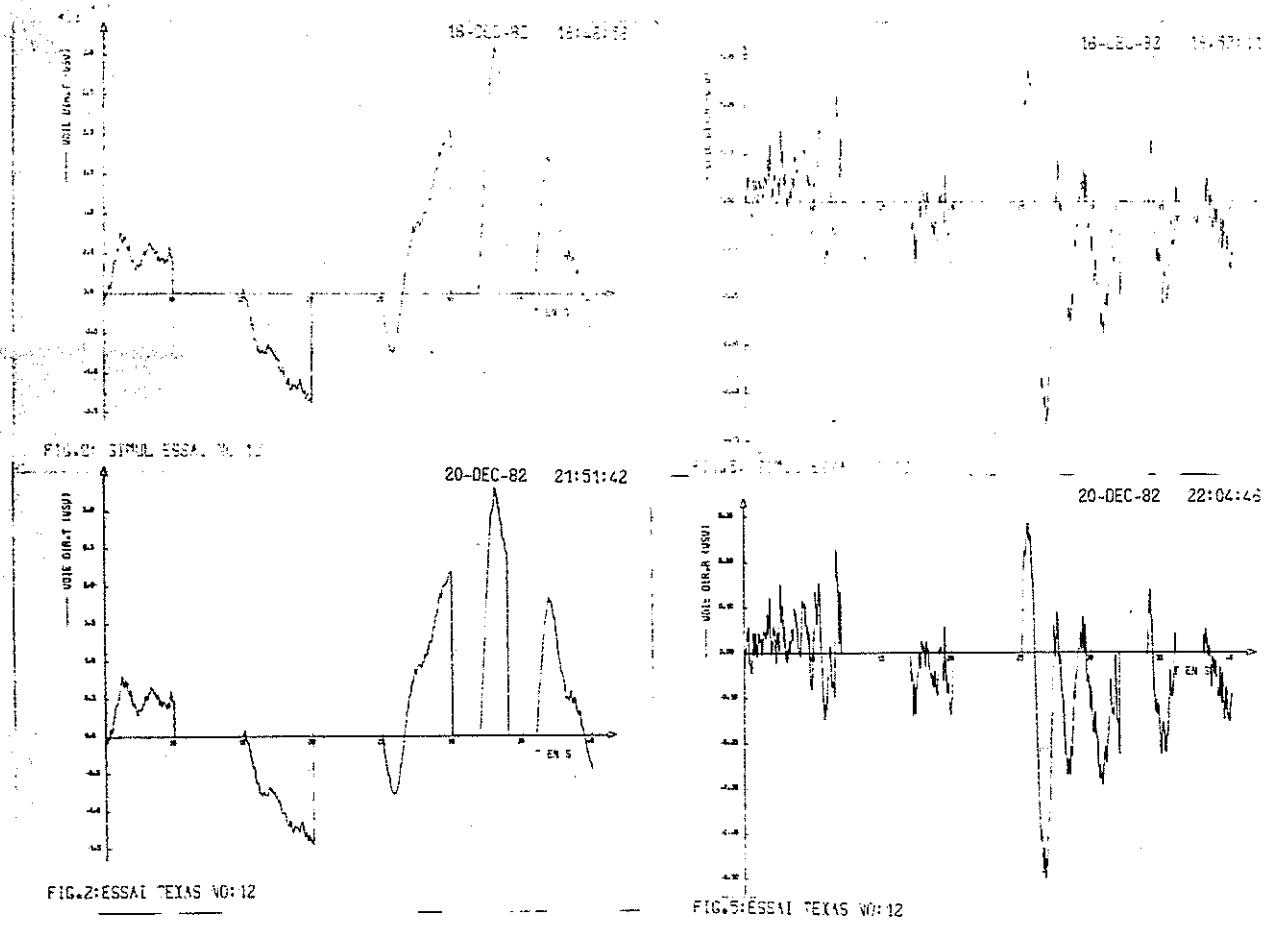


Figure n° 10 : Verification of Sequence of Modes (PAN1)

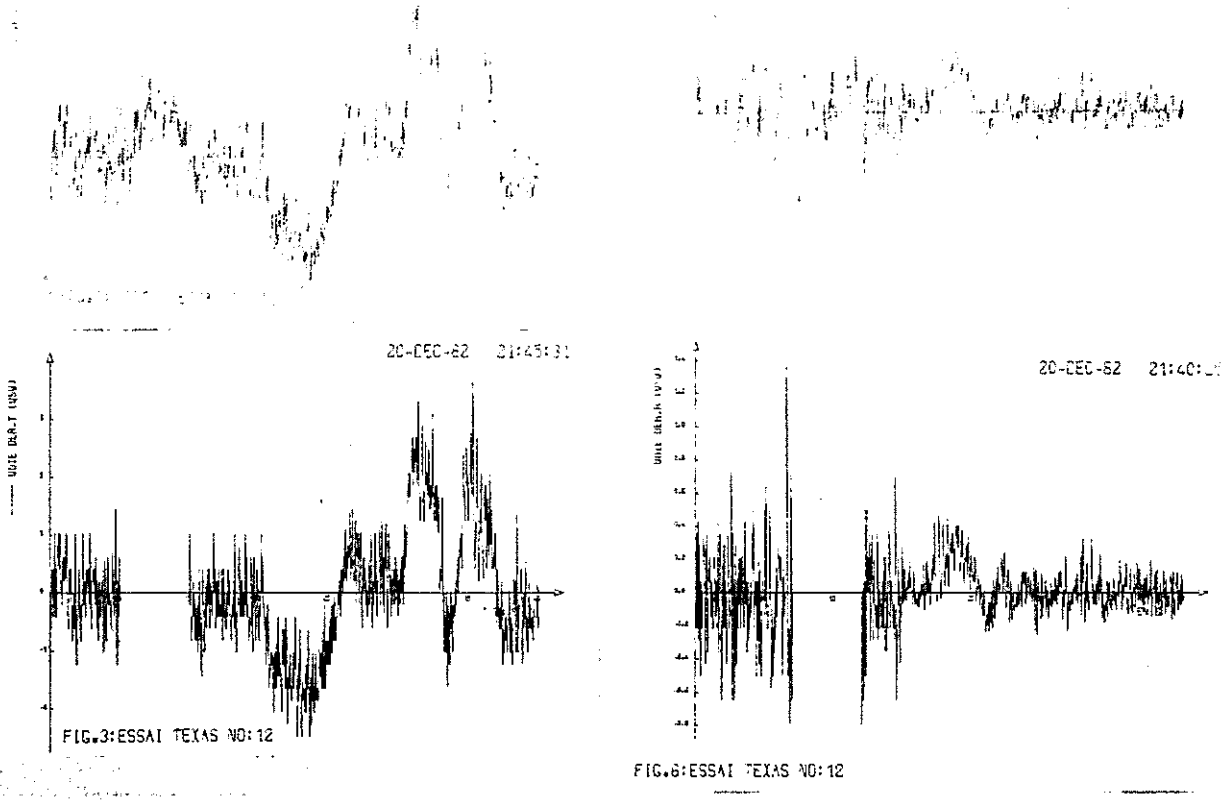


Figure n° 11 : Verification of Sequence of Modes (PAN1)

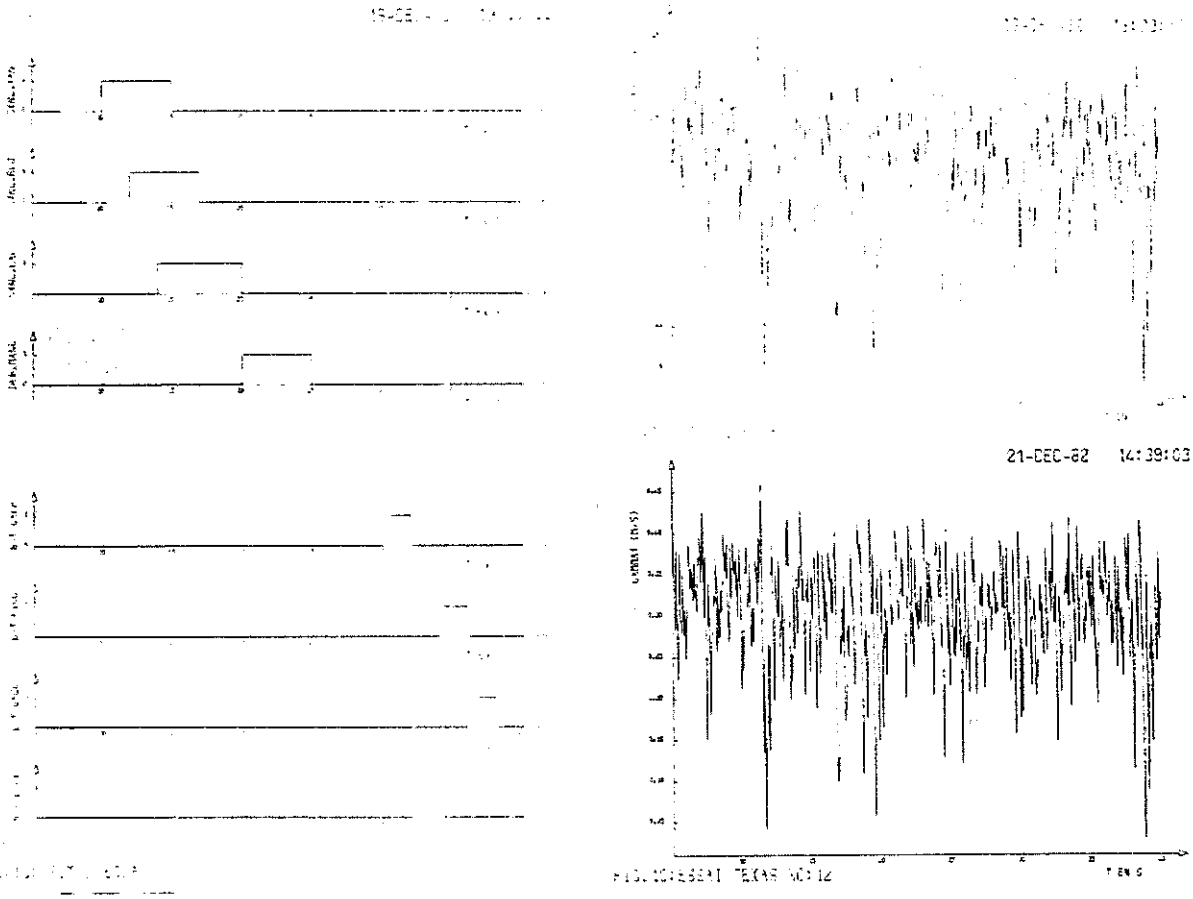


Figure n° 12 : Verification of Sequence of Modes (PAN1)

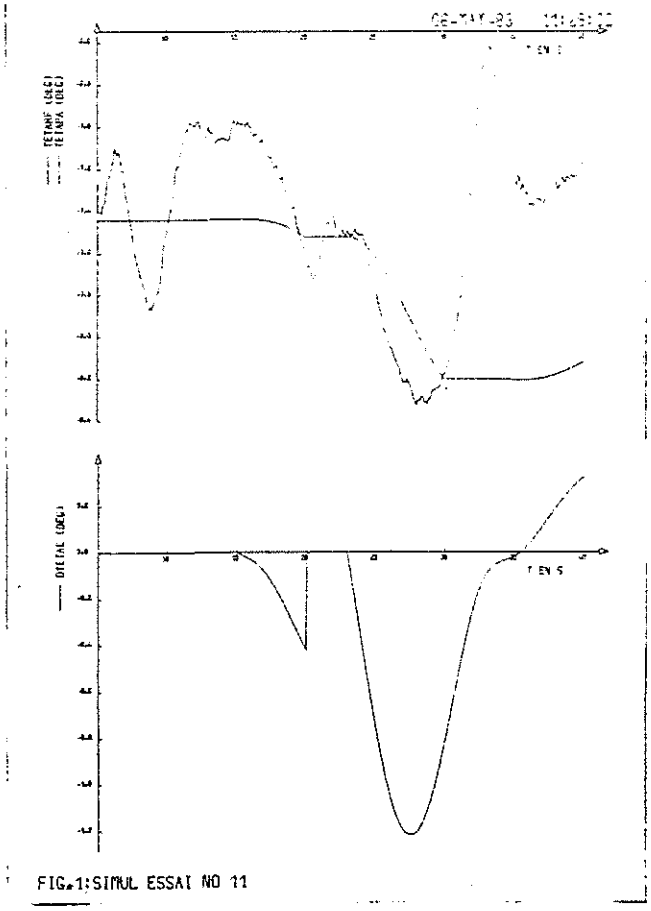


FIG.1:SIMUL ESSAI NO 11

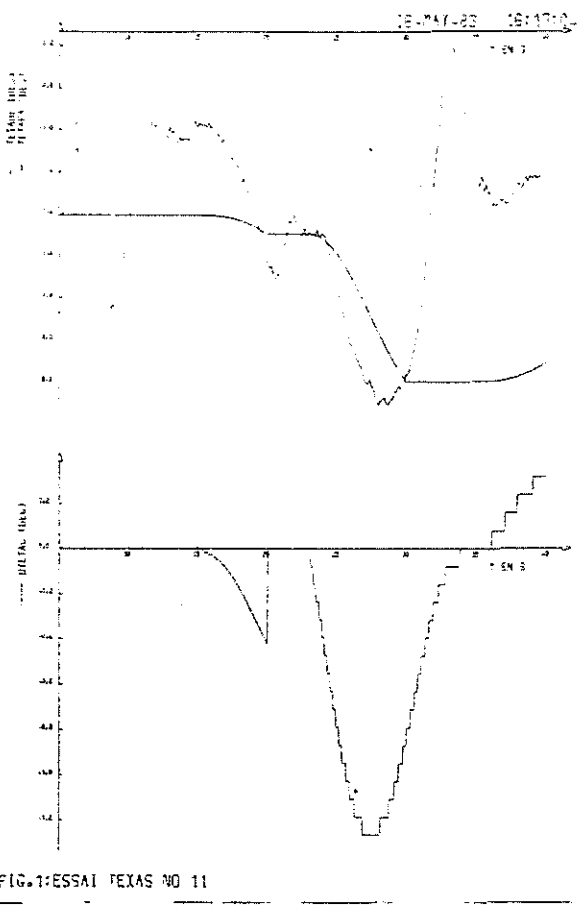


FIG.1:ESSAI TEXAS NO 11

Figure n° 13 : Verification test (PAN1)

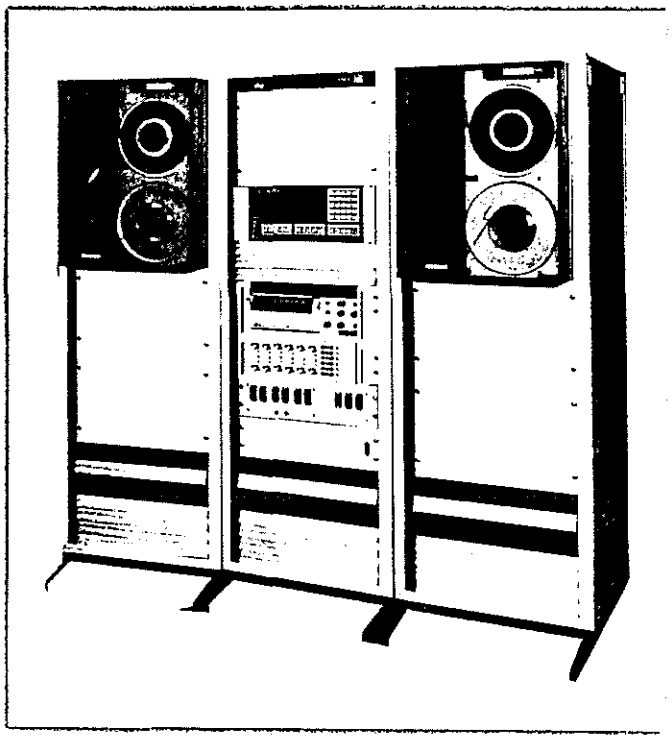


Photo n° 1 : SILENE

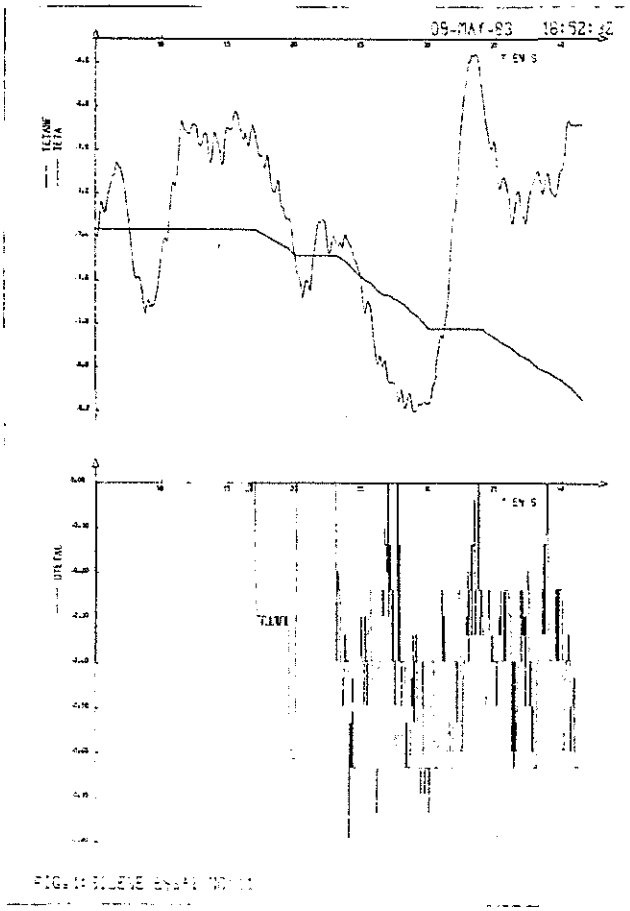


Figure n° 14 : Validation test (PAN1)