FOURTEENTH EUROPEAN ROTORCRAFT FORUM

Paper No. 34

THE SUPER PUMA MKII
AUTOMATIC FLIGHT CONTROL SYSTEM

A. VAISSIERE

SFIM

13 AV. RAMOLFO-GARNIER

91301 MASSY, FRANCE

20-23 September, 1988

MILANO, ITALY

ASSOCIAZIONE INDUSTRIE AEROSPAZIALI

ASSOCIAZIONE ITALIANA DI AERONAUTICA ED ASTRONAUTICA

THE SUPER PUMA MKII AUTOMATIC FLIGHT CONTROL SYSTEM

1 - INTRODUCTION

The SUPER PUMA MKII Automatic Flight Control System (AFCS) has been developped by SFIM since 1985 with the French Government Technical Administration STTE support. SFIM was selected by **AEROSPATIALE** as the SUPER PUMA MKII AFCS supplier in 1986.

The first AFCS computer has been delivered to **AEROSPATIALE** in April 1988 and is currently being evaluated on the **AEROSPATIALE** real time test bench "SISYPHE".

Following a brief review of the AFCS functions, the main points of the AFCS 165 architecture are discussed in the first part of this paper.
The paper continues with a detailed description of the software development. Emphasis is placed on the software support tools developed by SFIM for this application.
As a last point, technology available on the production version, reliability and maintenability will be considered.

2 - AFCS FUNCTIONS

The digital AFCS achieves three main functions :

- automatic flight control after autopilot engagement on basic or optional modes,

- monitoring of sensors immediately after setting power on,

- elaboration of power margin and flight envelope limits immediately after setting power on.

The AFCS is active on all four axis, on the basic version of the AFCS, the following modes are available :

- attitude and heading hold modes,
- cruise modes,
- radionavigation and approach modes.

For all modes, the pilot is provided with the capability of a large variety of fly through actions.

The optional modes refer to SAR or ASW/ASV missions or tactical military missions.

The AFCS presents the flight director capability.

3 - SYSTEM DESCRIPTION

3.1 - Design objectives

The AFCS 165 is designed to meet the operationality and safety objectives :

- the AFCS must be instantaneously and fully operational after first failure, for all AFCS functions, without increasing pilot workload, to achieve high mission succes,

- the AFCS must be fail passive upon second failure.

The dual/dual architecture has been chosen to meet these requirements.

3.2 - AFCS main features

The AFCS 165 System includes two identical AFCS computers and two Control Panels (AFCP).

In nominal operation, both computers are simultaneously engaged. When a computer detects its own failure, it disengages itself automatically while the other computer carries on controlling according to the modes engaged.
The remaining computer is then able to complete the mission throughout the AS 332 MKII flight envelope, keeping the same safety level, thanks to its dual architecture.

The two computers are synchronized and monitor each other through a Cross Talk Link.

3.3 - Peripheral Environment

Each AFCS computer is connected to the whole System Environment (cf figure 1) :

- two basic Control Panels (AFCP),

- two Flight Data Computers (FDC) which provide AHRS and ADC data,

- two Symbol Generator and Concentrator Units (SGCU) which drive the Displays and stand for concentrator unit in regards to the AFCS for radio navigation and radio altitude data,

- a vertical gyroscope for FPC monitoring in case of a FDC failure,

- a set of engine sensors for power margin computations,

- a reconfiguration and test panel,

- control sticks switches through the AFCP,

- series and parallel actuators.

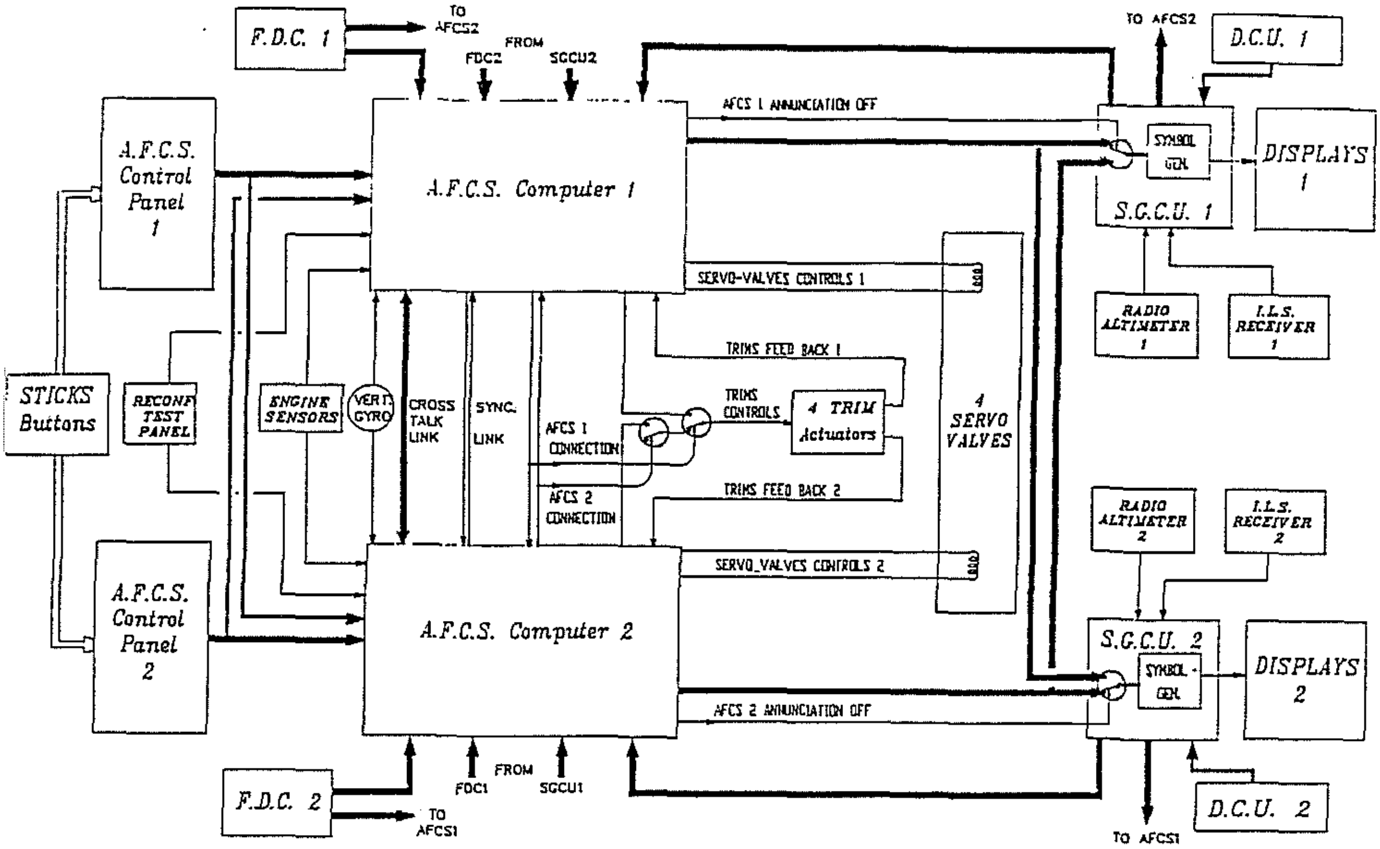


Fig. 1 : SYSTEM ARCHITECTURE

3.4 - System configuration

In normal configuration, each AFCS computer processes the parameters output from both AFCP's, both FDC's and both SGCU's, while each SGCU processes only those data output from the computers of its half-system.
When a reconfiguration occurs, the links handled by the shut down computer are disregarded by the rest of the system.

On every axis (pitch, roll, yaw, collective), the AFCS controls the rotor pitch through a series actuator (dual servo-valve) and a parallel actuator (electrical trim).

Each AFCS computer feeds one of the two inputs of each dual servo-valve, the actuator travel being proportional to the sum of the two inputs. Each servo-valve input authority is sufficient to provide control laws nominal performance within a single computer.

The four trim actuators are always controlled by one of the computer with hot standby in other AFCS computer. The master/slave functionning is fixed by aircraft pinning. When the master computer is disconnected, trim controls shift automatically to the second computer.

3.5 - Description of AFCS 165 computer

Each computer is fail passive and incorporates two redundant and separate processing channels (cf. Figure 2). The critical tasks (e.g. inner loops control laws) are developed in dissymmetrical software).

In order to avoid actuator runaways in case of a processor failure (hardware or software), an analog device votes between actuators control signals output from the two processing channels.

Each processing unit is based on the utmost powerful MC 68020 microprocessor and associated circuits (memory, timer, ...). The cyclic interrupts are produced by a synchronization clocks electronic connected to the second computer.

The two processing units communicate through a specific link, composed of dedicated memories, accessible by the two processors. This feature, associated with synchronization allows very close monitoring and immediate passivation by the voters in case of any computer failure.

The Power Supply is simplex and fail passive and works in a redundant way on both helicopter DC networks. Any failure is detected and causes the computer to be disconnected.

A set of safety logics controls the AFCS computer Disconnection and the Hands-On Recovery Alarm Request.

Each processor unit encodes one duplex sensors side and a part of the simplex sensors. The encoded measures are then exchanged through the inter-processors cross talk.

One processor handles the Annunciation Link to the SGCU, the critical parameters are returned to the other processor for monitoring purpose.

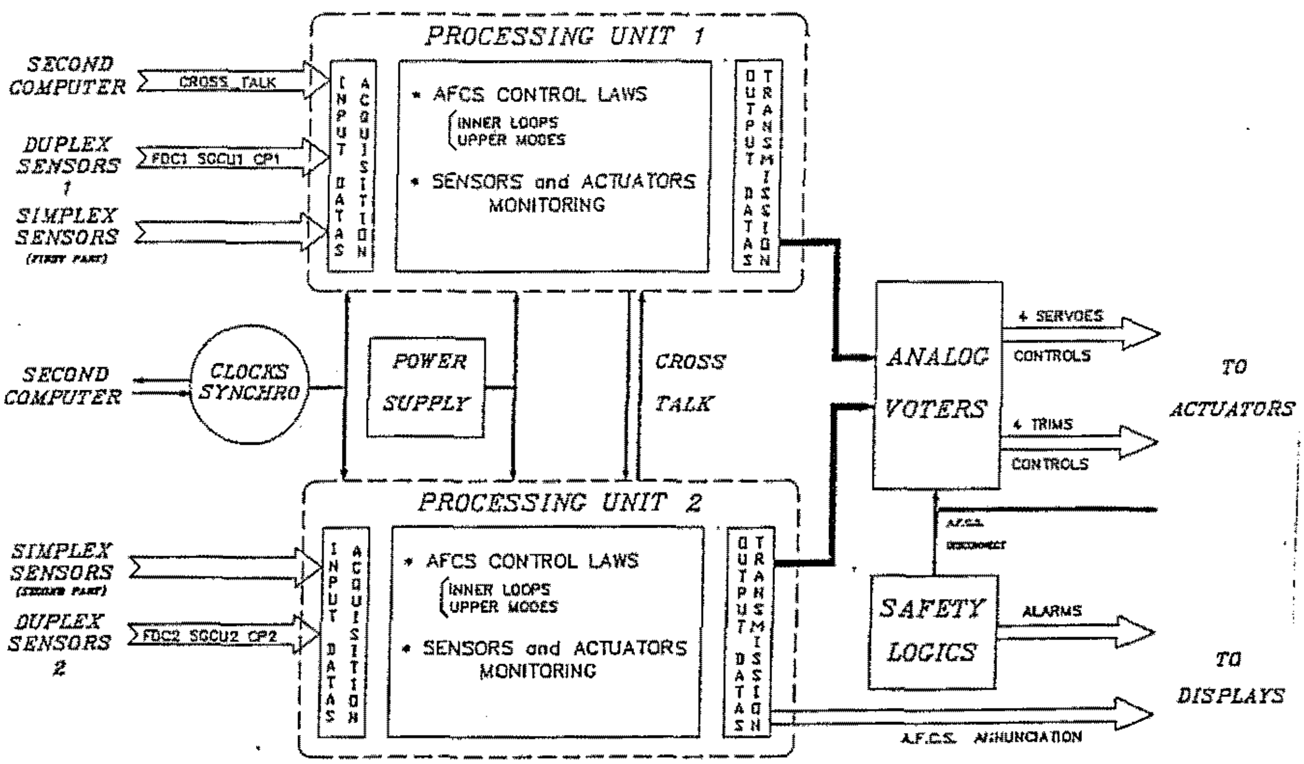


Fig. 2 : A.F.C.S. 165 COMPUTER DESCRIPTION

4 - SOFTWARE DEVELOPMENT

4.1 - Generalities

Most of the AFCS 165 functions are performed by software. The critical functions are developed dissymetrically by two different teams and implemented in each computer in the two 68020 processing unit. Dissymetry between the two software goes from the design phasis to the test phases, both teams using high order language Pascal supported by two different development trains (BSO and OREGON).

According to the criticallity of the AFCS functions developed by use of software, emphasis has been placed on the software development at the very beginning of the project. Thus, all AFCS functions are developed in accordance with Level 1 Procedures DO178A Standards ("Software Considerations in Airborne Systems and Equipment Certification").

In order to ensure the quality of the software all stages of the software life cycle are supported by specific tools, developped by SFIM for this application.

4.2 - Software life cycle

Software life cycle has been stated in the Software Quality Assurance Plan at the beginning of the project.

Software development main phasis are the following :

- System Requirements.
- Software Requirements.
- Software Design.
- Coding and Module Unitary Testing.
- Software Integration.
- Hardware/Software Integration.
- Software and System Validation.

Documentation is delivered at each phase accordingly to the Level 1 DO178A Requirements.

Software reviews are planned at the end of each phase with the participation of **AEROSPATIALE.DH.**
The french regulatory agencies (CEAT and STTE) follow the software development after each review.

4.3 - Software support tools goals

Software support tools has been developed by SFIM to assist the software development teams to ensure software consistency and traceability among the different phases.

Furthermore, to increase the development efficiency of the software, support tools focus on the following aspects :

- interactive data handling,
- automatic software and documentation production.

The main software support tools are presented here after.

4.4 - OSCAR

OSCAR is a design support tool which is used at the design phase in order to ensure the consistency of the data flow described in the design issue and the concrete data flow of the realised software.

Each software module is described by the software designer using standardized guidelines, by the mean of modules forms describing all caracteristic informations (variables, constants, procedures). Variables dictionary and cross reference tables are then produced by OSCAR.

All these formalized informations constitute the basis of the design documentation.

When all modules informations have been compiled without any consistency errors by OSCAR, the declarative parts of the PASCAL modules are automatically generated by OSCAR.

Figure 3 and 4 show a typical module form and the corresponding PASCAL declarative part.

4.5 - PAGOS

PAGOS has been created to ensure the traceability between the code and the Block Diagram Control Laws Specifications (either during the Software Specification phase or Design phase).

PAGOS enables the interactive graphical data acquisition of the Block Diagrams and the automatic code generation from the Block Diagrams.

Figure 5 shows a typical Block Diagram description and the corresponding PASCAL code.

4.6 - ODT

ODT has been created to assist the module testing and the software integration testing on the Host Computer (VAX 11/780 and uVAX 2).
It provides a structured test language which enables a standardized presentation of the test definitions and ensures the consistency between test documentation and test execution.

Test outputs are analyzed graphically in facilitating interpretation of the test results.

Test programs are archived and reexecuted allowing "non regression" testing whenever a modification is made on tested program.

Figure 6 show a test definition using test language and the graphical test results from PAGOS example.

4.7 - GCF

GCF is a software configuration management tool. The whole set of files necessary to the recovery of a software version, as well as test sets are managed by GCF in a library.

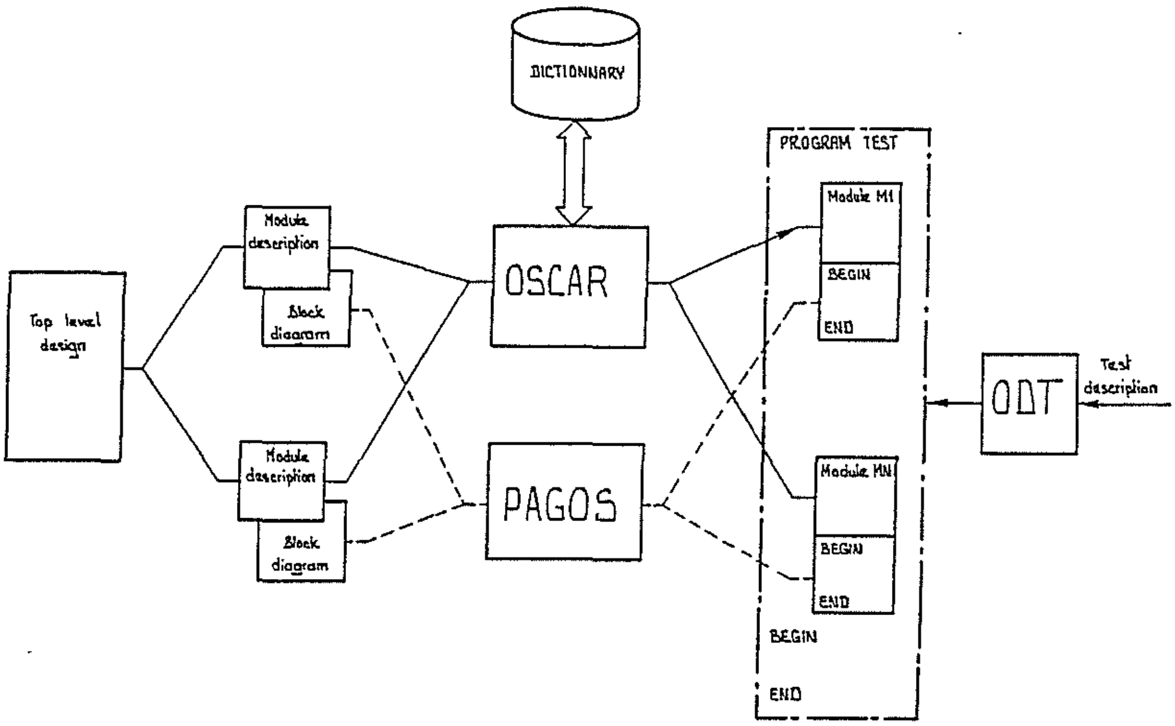
The elements of the library are managed in Version/Revision.

The software evolution is managed by mean of evolution forms. GCF allows to link the evolution of each library element to the evolution forms.

4.8 - Software Workshop

All these tools represent a dedicated software workshop for AFCS software development.

As it is shown in Figure 7, OSCAR and PAGOS allows for Block Diagram Control Laws Specification, to make the first step towards the Automatic Software Production.



Nevertheles the use of these tools, as automatic Software Production Tools, does not reduce the test effort for the AFCS 165 Certifiction Process.

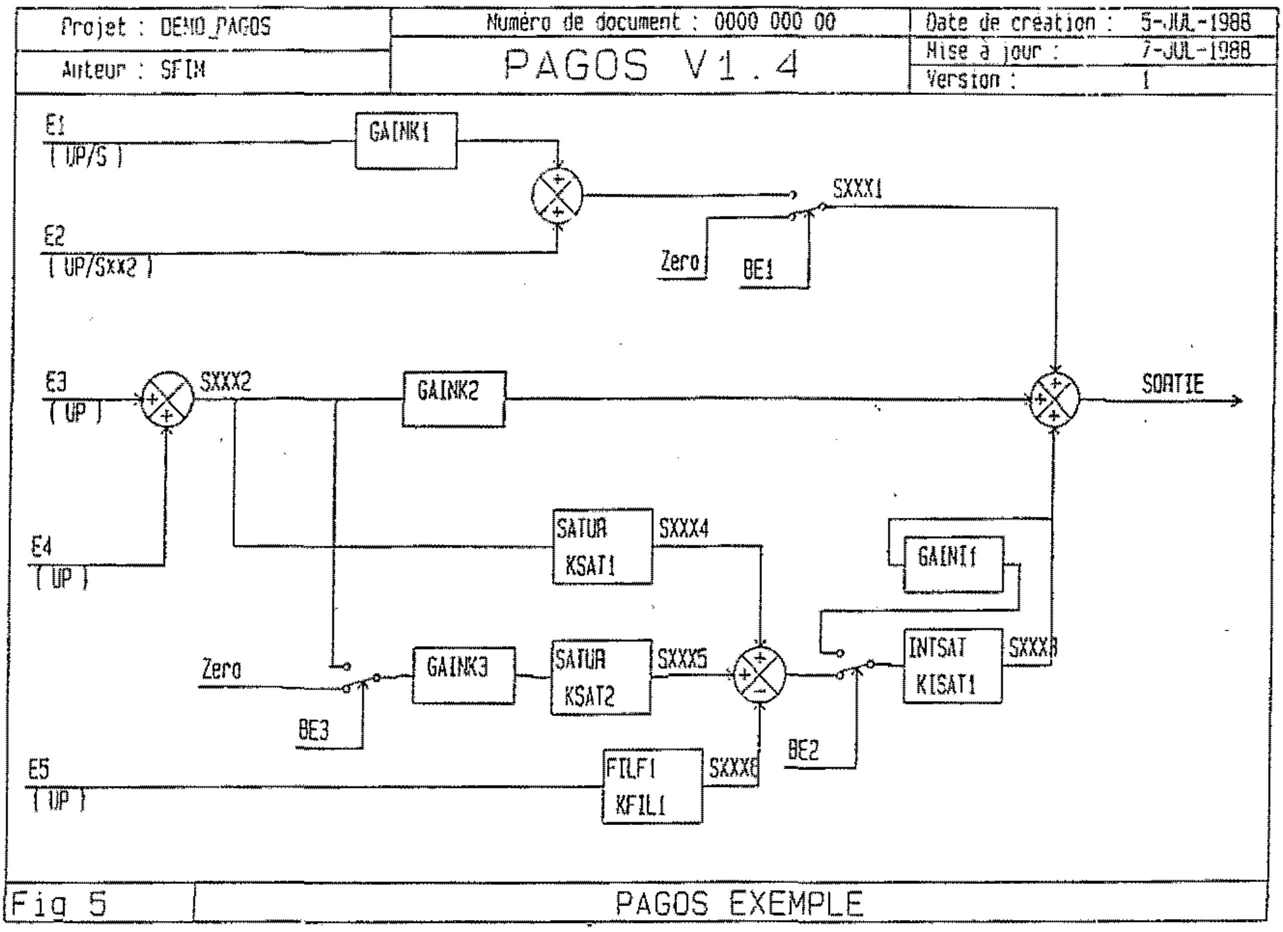
The set of pictures 3 to 6 illustrate, for one standard module, the typical products handled by the software workshop.

```
TITRE : DEMO_PAGOS

cadence : 10 ms

tache : TACHE_PAGOS

fonction :EXEMPLE

REFERENCES :     Specifications logicielles
                   Chapitre :DEMONSTRATION
                 Paragraphe :PAGOS

DESCRIPTION :
.P;This module shows an exemple of the use of PAGOS.

ENTREES
! INPUT FLOW CARACTERISTICS
E1,E2,E3,E4,E5
BE1,BE2,BE3

SORTIES
! OUTPUT FLOW CARACTERISTICS
SORTIE

CONSTANTES
        ZERO, KISAT1

PROCEDURES
        SATUR
        FILF1
        INTSAT
TYPES
        KF1
        XF1
        KINTSAT
        XINTSAT

DEFINITIONS

VARIABLES

SORTIE
Output from exemple
T=REAL
U=mA
P=obs

SXXX1
Intermediate variable
T=REAL

SXXX2
Intermediate variable
T=REAL

SXXX3
Intermediate variable
T=REAL

SXXX4
Intermediate variable
T=REAL

SXXX5
Intermediate variable
T=REAL

SXXX6
Intermediate variable
T=REAL

CONSTANTES

GAINK1
Direct Gain
T=REAL
V=0.5

GAINK2
Direct Gain
T=REAL
V=2.0

GAINK3
Direct Gain
T=REAL
V=1.5

GAINI1
Integral Gain
T=REAL
V=-0.6

KSAT1
Saturation level
T=REAL
V=2.0

KSAT2
Saturation level
T=REAL
V=3.0

KISAT1
Integral saturation level
P=mod
T=KINTSAT
V=2.4

KFIL1
Constant array of filter
T=KF1
V=2.6

PROCEDURES

DEMO_PAGOS;

TYPES
```

Fig. 3 : OSCAR Module Form

```
MODULE DEMO_PAGOS ;

(* 1. GENERALITES
      1.1 DESCRIPTION

This module shows an example of the use of PAGOS.
      1.2 DATE

      Generation de cette en-tete a 17:41 le 07-07-1988 a partir des
   informations incluses dans le dictionnaire a 17:41 le 07-07-1988

      1.3 REFERENCES

      Fichier produit par GENE_DECL version 1.0 du 4 Janvier 1988

   2. CONSTANTES SIMPLES
                                *)
(*      2.1 CONSTANTES SIMPLES IMPORTEES
                                          *)
CONST
   (* null value *)
   ZERO        = 0 ;

(*      2.2 CONSTANTES SIMPLES EXPORTEES
                                          *)
CONST
   (* Integral Gain *)
   GAINI1      = -0.6 ;

   (* Direct Gain *)
   GAINK1      = 0.5 ;

   (* Direct Gain *)
   GAINK2      = 2 ;

   (* Direct Gain *)
   GAINK3      = 1.5 ;

   (* Saturation level *)
   KSAT1       = 2 ;

   (* Saturation level *)
   KSAT2       = 3 ;

(* 3. TYPES
         *)
TYPE
   KF1 = RECORD
        A : REAL ;
        B : REAL ;
          END ;

   XF1 = REAL ;

   KINTSAT = RECORD
        EC : REAL ;
        DT : REAL ;
          END ;

   XINTSAT = REAL ;

(* 4 . CONSTANTES STRUCTUREES
                                *)
(*      4.1 CONSTANTES STRUCTUREES IMPORTEES
                                          *)
(*      4.2 CONSTANTES STRUCTUREES EXPORTEES
                                          *)
VAR
   (* Constant array of filter *)
   KFIL1       : [GLOBAL] KF1             (* tau = 2.6 sec, DT = 0.01 sec. *)
               := ( 0.00383, 0.99617 ) ;

   (* Integral saturation level *)
   KISAT1      : [GLOBAL] KINTSAT
               := ( 2.4, 0.01 ) ;

(* 5 . VARIABLES
                *)
(*      5.1 VARIABLES IMPORTEES
                                   *)
VAR
   (* First Input *)
   E1          : [EXTERNAL] REAL        ;

   (* Second Input *)
   E2          : [EXTERNAL] REAL        ;

   (* Third Input *)
   E3          : [EXTERNAL] REAL        ;

   (* Forth Input *)
   E4          : [EXTERNAL] REAL        ;

   (* Fifth Input *)
   E5          : [EXTERNAL] REAL        ;

   (* Logical input *)
   BE1         : [EXTERNAL] BOOLEAN     ;

   (* Logical input *)
   BE2         : [EXTERNAL] BOOLEAN     ;

   (* Logical input *)
   BE3         : [EXTERNAL] BOOLEAN     ;

(*      5.2 VARIABLES EXPORTEES
                                   *)
   (* Output from exemple *)
   SORTIE      : [GLOBAL] REAL          ;

   (* Intermediate variable *)
   SXXX1       : [GLOBAL] REAL          ;

   (* Intermediate variable *)
   SXXX2       : [GLOBAL] REAL          ;

   (* Intermediate variable *)
   SXXX3       : [GLOBAL] REAL          ;

   (* Intermediate variable *)
   SXXX4       : [GLOBAL] REAL          ;

   (* Intermediate variable *)
   SXXX5       : [GLOBAL] REAL          ;

   (* Intermediate variable *)
   SXXX6       : [GLOBAL] REAL          ;

(* 6 . PROCEDURES ET FONCTIONS
                                *)
(*      6.1 PROCEDURES ET FONCTIONS IMPORTEES
                                               *)
[EXTERNAL] PROCEDURE SATUR ( E : REAL ; K : REAL ; VAR S : REAL ) ;

   EXTERN ;

[EXTERNAL] PROCEDURE FILF1 ( E : REAL ; K : KF1 ; VAR X : XF1 ;
                             VAR S : REAL ) ;
   EXTERN ;

[EXTERNAL] PROCEDURE INTSAT( E : REAL ; K : KINTSAT ;
                             VAR X : XINTSAT ; VAR S : REAL ) ;

   EXTERN ;

(*      6.2 PROCEDURES ET FONCTIONS EXPORTEES
                                               *)
[GLOBAL] PROCEDURE DEMO_PAGOS;

   FORWARD ;

%INCLUDE'DEMO_PAGOS.pag/list'

END .
```

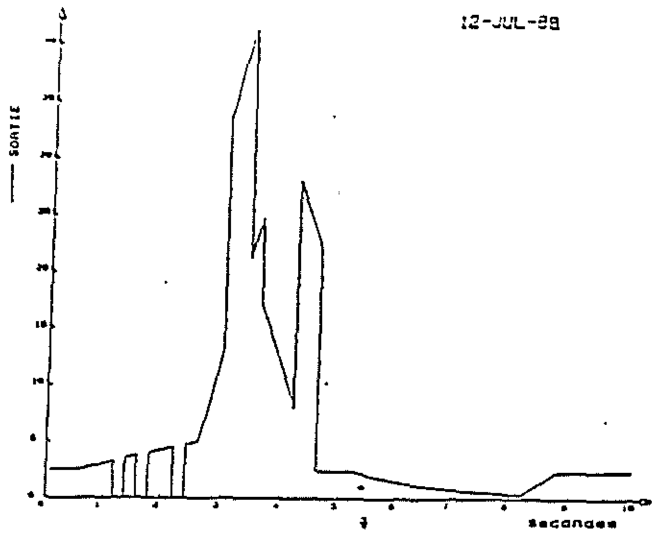
Fig. 4 : Declarative Part generated by OSCAR

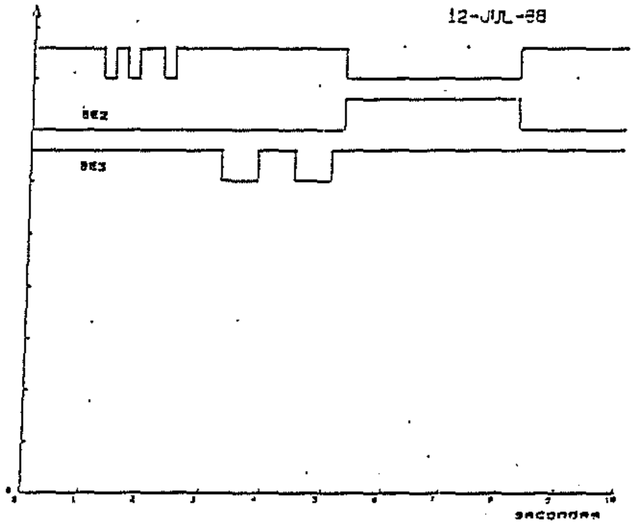


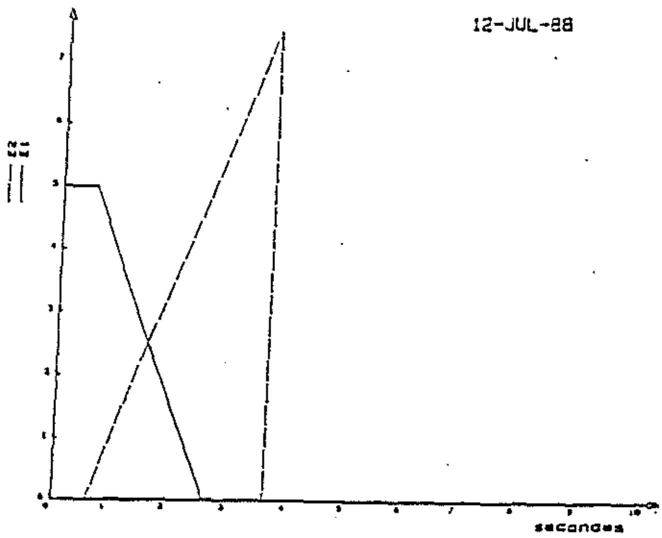

```
              Projet : DEMO_PAGOS
              Auteur : SFIM
               Titre : PAGOS EXEMPLE
             Version : 1
    Date de creation :  5-JUL-1988
Derniere mise a jour :  7-JUL-1988


Voir document numero : 0000 000 00
              Figure : 5

}

VAR
(* Variables internes *)
    XKFIL1 : REAL ;
    XKISAT1 : REAL ;

PROCEDURE demo_pagos ;

    VAR
        WORK1 : REAL ;
        WORK2 : REAL ;
        WORK3 : REAL ;

BEGIN
SXXX2 := (+ E3 + E4) ;
SATUR(SXXX2, KSAT1, SXXX4) ;
IF BE3
    THEN WORK1 := SXXX2
    ELSE WORK1 := Zero ;
WORK2 := (GAINK3 * WORK1) ;
SATUR(WORK2, KSAT2, SXXX5) ;
FILF1(E5, KFIL1, XKFIL1, SXXX6) ;
IF BE2
    THEN WORK3 :=  (GAINI1 * SXXX3)
    ELSE WORK3 :=  (+ SXXX4 + SXXX5 - SXXX6) ;
INTSAT(WORK3, KISAT1, XKISAT1, SXXX3) ;
IF BE1
    THEN SXXX1 :=  (+  (GAINK1 * E1) + E2)
    ELSE SXXX1 := Zero ;
SORTIE := (+ SXXX1 +  (GAINK2 * SXXX2) + SXXX3) ;
END ;
```

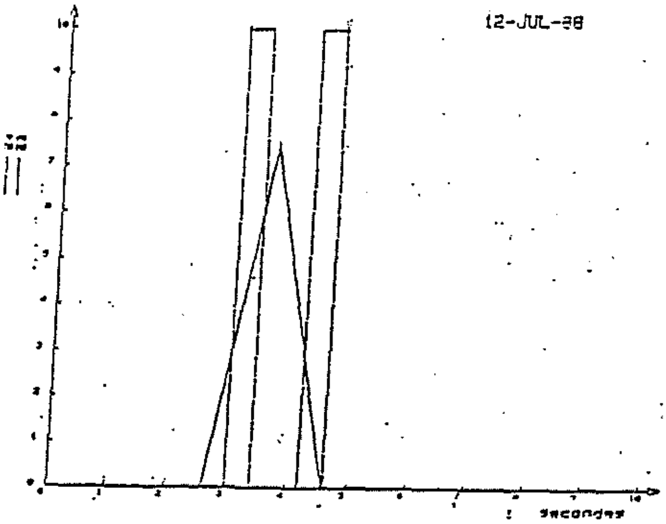
Fig. 5 : Block diagram and corresponding code

```
(* BOUCLE DE SIMULATION *)
DURING (duree_simul) STEP (dt_simul) SIMULATE
    BEGIN
    TEMPS_SIMUL := simul_time ;

    ( valeurs par defaut de toutes les variables d'entree )

    BE1 := true  ;
    BE2 := false ;
    BE3 := true  ;

    E2 := 0 ;
    E3 := 0 ;
    E4 := 0 ;
    E5 := 0 ;

    ( liste des actions )


    t1 := 3*dt ;
    AT TIME = 0.0 DURING (t1+10*dt) DO E2 := 5.0;
    AT TIME = t1 DURING 10*dt DO derive_ (E2,-2.5 ) ;
    AT TIME = t1 DURING 15*dt DO derive_ (E3,+2.5 ) ;
    AT TIME = t1 + 3*dt DURING dt DO BE1 := FALSE;
    AT TIME = t1 + 5*dt DURING dt DO BE1 := FALSE;
    AT TIME = t1 + 8*dt DURING dt DO BE1 := FALSE;

    t2 := t1 + 10*dt ;
    AT TIME = t2 DO E3 := 0;
    AT TIME = t2 DURING 5*dt DO derive_ (E3,+7.5 ) ;
    AT TIME = t2+5*dt DURING 5*dt DO derive_ (E3,-7.5 ) ;
    AT TIME = t2+2*dt DURING 2*dt DO biais_constant_ (E4, 10.0) ;
    AT TIME = t2+6*dt DURING 2*dt DO biais_constant_ (E4, 10.0) ;
    AT TIME = t2+3*dt DURING 3*dt DO BE3 := false ;
    AT TIME = t2+9*dt DURING 3*dt DO BE3 := false ;
    AT TIME = t2+13*dt DURING 13*dt DO
        begin
        BE1 := false ;
        BE2 := true ;
        end ;
    t3 := t1 + 25*dt;
    AT TIME = t3 DURING 5*dt DO derive_ (E5, -10.0 ) ;
    AT TIME = t3+5*dt DURING 30*dt DO biais_constant_ (E5,-5.0) ;

    DEMO_PAGOS;

    EVERY dt_simul DO ecrire_record_sortie;

    END ;
```









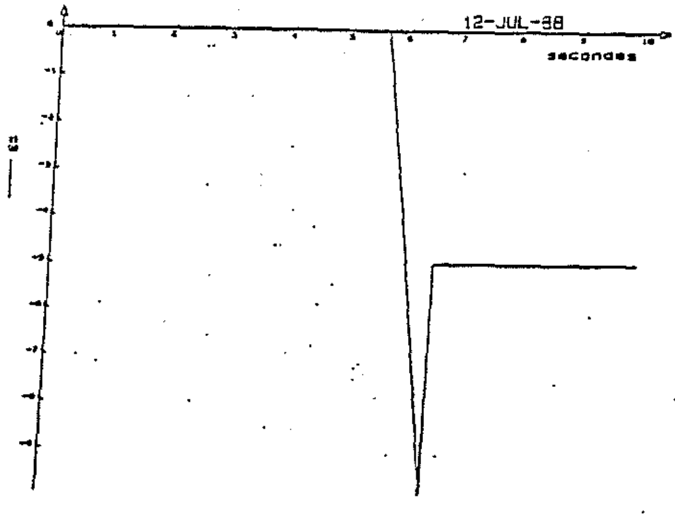


Fig. 6 : Test definition and graphical results

5 - TECHNOLOGY, RELIABILITY, MAINTENABILITY

5.1 - Environmental conditions

The technology used for the SUPER PUMA MKII SFIM AFCS is one of the means to achieve safety, reliability and maintenability objectives, in most severe environmental conditions related to different types of mission.
Therefore, technical design choices to compete with environmental threats has been taken against the following :

- temperature,
- vibration and mechanical damage,
- thunderstorms secondary effects (lightning protection),
- electromagnetic interferences.

These technical choices will be overviewed is the followings parts.

5.2 - AFCS Mechanical and thermal technology

Although the first AFCS identified roles are civil ones, naval and military roles are planned for the future. Therefore, a sealed box has been retained for the AFCS computers. The SUPER PUMA MK II does not provide "clean" forced air for equipment cooling, thus, only natural convection is used. Particular attention has been paid to thermal conduction path between hot sources and the box skin.

The mechanical design has optimized weight distribution. A damped mounting tray has also been retained in order to filter the major part of environmental vibration and mechanical stress, thus enhancing drastically the overall system reliability and availability.

The dimensionnal characteristics for the production version of the AFCS computer are 4 MCU, to ARINC 700 specification, equiped with two ARINC 404 connectors featuring 134 contacts each. This box is able to contain up to six electronic boards in addition to the Power Supply board. Only five boards (of four different types) will be used for the production version, thus offering a spare room for growth capability.

5.3 - AFCS Electronic technology

In order to minimize the power dissipation inside the box, CMOS technology has been used whenever it was possible.

The AFCS computer electronics uses mainly DIL components to achieve fonctionnal requirements in order to keep compatibility between industrial and military components at the Lay out level, which is not the case today for most of the surface mounted parts (PLCC/LCC).
Thus, upgrading AFCS computers to military version is achieved only by components substitution.

Specific input/output functions are implemented in SFIM custom designed VLSI (ASIC 35000 gates), available in all temperature ranges.

5.4 - Lightning protection

The AFCS computers are designed to recover nominal operation after stress due to thunderstorms secondary effects. The stress severity is modelized in SAE-AE4L Comittie Report.
The protections parts have been chosen to deal with levels 3 and 4 as per defined in SAE-AE4L Report.

Those protections parts are surface mounted components for space saving.

5.5 - EMI protection

Basic EMI protection to meet "emission" requirements is achieved by design attention in the field of components choice, boards layout, wiring definition. Non suceptibility to radiated emissions till 50 V/m is also insured basically by use of low pass filters. Protection against stronger fields is offered as an option by replacing "standard" connectors with specific filtering connectors.

5.6 - Maintenability

Specific attention has been placed on the maintenability aspects from the early phase of the development, using the capabilities brought by digital technology in two mains areas :

- system reporting, covering the others equipments of the AFCS system (AFCS control units, trims actuators),

- self test and fault diagnosis within the computers, and communication with maintenance concentrators.

6 - CONCLUSION

Thanks to its dual/dual architecture based on hardware redundancy and software dissimilarity, the AFCS 165 provides failure passivation and is fully operational after first AFCS computer defect.

With regards to its critical functions performed in software and to meet the level 1 DO178A Standards, a rigorous methodology is supported by a wide range of software support tools covering all phases of software development.

The use of these tools ensure :

- consistency of the produced software along with the specifications,

- tracability of the software development,

- increase of efficiency.

The advanced technology insertion satisfy the needs of the multirole SUPER PUMA MK II to complete missions when subjected to severe environmental conditions.