Defence Science Journal, Vol 44, No 2, April 1994, pp 119-129

An Aid for Mechanization of Flight Control Systems on Micro-computers

T.V. Rama Murthy
National Aerospace Laboratories, Bangalore-560 017
and
V. Seshadri
Indian Institute of Technology, Madras-600 036

ABSTRACT

This paper deals with the development of an automated aid to translate the block diagram of flight control system (FCS) to assembly level code. By defining a suitable syntax and by building a matrix of inputs and outputs of the blocks, it becomes easy to translate the block diagram. It is also shown how the process of fault detection can be automated. The results obtained through the automated aid have been validated by ORACL library using the block diagram of Cstar controller of F8 aircraft.

NOMENCLATURE

q, N_z, δ_e	aircraft states
δ_c	elevator command
S	signature camberra metric
S^*_k	geometric moving average of S
T_F	execution time of control law
A_R, B_R, C_R, D_R	state space specifications of regulator
A_P, B_P, C_P, D_P	state space specifications of plant

INTRODUCTION

Although attempts have been made to translate the block diagrams of flight control systems to assembly level code, it appears that issues of detection of faults like sensor faults, hardware/software faults and self-test are not directly addressed as a part of computer-aided tools. An earlier work[1] on the translator has been improved by (i) implementing and verifying fault-detection algorithms,(ii) validating the tool using ORACL library, and (iii) debugging aid has been made easier by employing Turbopascal >=5.0.

2. DEVELOPMENTS IN ACTIVE CONTROL TECHNOLOGY

Fly-by-wire, electrical signaling and control configured vehicles (CCV) have been the major milestones in the development of active control technology (ACT). The applications of ACT have been mainly in the following areas: (i) artificial stability, (ii) ride and handling quality improvements, (iii) load alleviation, (iv) flutter speed enhancement, (v) centre of gravity control, (vi) envelope limiting, (vii) fatigue reduction and (viii) pilot relief-autopilot functions.

2.1 Implementation Aspects of ACT

The phenomenal growth in the area of VLSI architecture and related devices has made possible the execution of control laws through digital means. However, for reliability, one has to go in for redundant computing elements with self-test capabilities. The software of FCS may be divided generally into three modules – (i) executive software including I/O and flight test constituting about 25 per cent, (ii) control law

Received 14 September 1992, revised

computation including gain schedule and logic (about 35%) and (iii) fault-tolerance including redundancy management, self-test, etc. (about 40 per cent)[2].

The executive software deals with the overall supervision of the system. The complete software cycle called 1 major cycle consists of n minor cycles, each minor cycle is typically 10 or 20 ms to meet the real-time requirements. The executive initiates the minor cycle operations from the table of computations. It also communicates with the pilot for auto-pilot and other outerloop options and for status reporting.

There are many benefits due to CCV implementation. For example, in F-8 aircraft, control laws have been designed for improved handling qualities, envelope limiting, gust alleviation, capability to fly with reduced static stability in lateral axis improved dutch roll damping, turn coordination[3], etc. The outerloop modes are: attitude hold, altitude hold, mach hold; auto-pilot functions like approach/land, go around and take off[4]. The executive/utility sets up the computation table depending on options selected by the pilot. Often, the controller has to get gain settings based on function of mach number and/or dynamic pressure or mach/altitude grid. Hence gain scheduling task will also find entry in minor cycle computation table. All tasks which need processing rates of 50, 25, 12.5, 6.25 Hz will be initiated once in each, alternate, once in 4 and once in 8 minor cycles respectively assuming minor cycle of 20 ms.

2.2 Fault Detection Measures

Redundancy management essentially deals with failure monitoring, fault isolation and voting mechanism. Reconfiguration of a failed element may take place by analytic redundancy technique or ignored by continuation with remaining redundant units. The following form part of pre-flight test[5]: limit cycle tests, structural resonance tests, frequency response tests,electromagnetic interference tests (including lightning tests).

The self-test and in-flight monitoring capability required depends entirely on its criticality and assumes major importance and the various techniques that are used are given elsewhere[6].

Sensor monitoring is done through: (i) comparison monitoring of feedback sensor signals, (ii) software limiting of outerloop sensor signals and (iii) monitoring of critical gains in air-data computation. Servo monitoring is performed by comparing actual servo output against digital models. Input/output circuitry check is done by feeding known constants to A/D converter and checking the code. Similarly, a known digital code is fed to DAC and its analog value checked. CPU monitoring is performed by self-test. All known instructions and control crucial to flight safety are checked. Dynamic computation monitoring is performed by an external independent analog element in some cases. Watchdog timer checks whether the computations are completed within the specified time limits. Memory monitoring is performed by checking, (i) all words accessed from memory for parity, (ii) periodic checksums of critical instructions, constants and scratch pad locations and (iii) program flow through critical instructions.

3. NEED FOR AUTOMATED AID

It is reported that assembly level programs have been used for implementing the flight control programs of modern class of fighter aircraft like F/A 18, F-8 and JA-37. Real-time considerations have forced the designers to write assembly level programs to translate flight control laws with the attendant poor productivity and inflexibility in incorporating changes in control laws. Often, hard debugging effort is required to make the assembly level programs. The designer has to keep track of interconnection of blocks to obtain the exact word length of variables. Some errors due to shorter word length of variables, wrong entries, inadequate delay operation of variables in filters may go unnoticed or be corrected after many run-time attempts.

Fault-detection measures like sensor failures, timing faults and value faults are important in FCS software design. It is desirable to automate these processes. Since the assembly level program is generated manually, it is time consuming to design self-test of the instruction set of the processor.

To meet these requirements, the automated aid has been designed and it works in the environment of IBM PC/MSDOS, MASM assembler, LINK and Turbopascal $\geq$ 5.0. The automated aid generates reliable codes at optimum assembly level with FCS specifications being expressed in easy-to-use high level syntax. Any hardware/software fault usually manifests in terms of value fault and/or timing fault.

It is known that important instructions are exercised as a part of self-test routine in in-flight monitoring. It will be very easy to select the set of basic instructions from that used by the automated aid. The code-segment part is generated using only the predefined set of macros. From the macro-bodies, a set of basic instructions which can express the complete macro-set is selected. A program is then exercised which makes use of these basic instructions. By comparing the results obtained with the expected values, it is possible to flag any error that occurs while running the self-test.

4. PARSER

4.1 Syntax Design

Typical FCS block diagrams have been studied to arrive at a suitable syntax for representing the blocks. Each block is defined in terms of a set or subset of input(s), output, constants, control variables, etc. The following syntactic entities are defined to specify the various blocks :

<ALPHA>::=*A/B/C../Z* NUMS::=0/1/2/3../9

<ALPHANUM>::=ALPHA/NUMS

<SYMBOL>::=<ALPHA> $\int_0^5$ [ALPHANUM]

The operator $\int_0^5$ specifies that all syntax items enclosed by square brackets are to be repeated 0 or *m* number of times ($m \leq 5$)[7]. Syntax to handle the numerical constants are defined as follows:

<REALNUM>::=<INTEGER><FRACTION>/ <FRACTION>

<FRACTION>::= <INTEGER>

<INTEGER>::=<NUMS>

$\int_0^5$ [<NUMS>]

<REALNUMS>::=<REALNUM>

$\int_0^{11}$[,<REALNUM>

<SYMBOLS>::=<SYMBOL>

$\int_0^{14}$ [,<SYMBOL>]

One can take advantage of limited number of blocks that are required and select certain alphabets to denote the blocks. For example, an amplifier may be defined by a three-character symbol as shown below :

<AMPLIFIER>::=*A*<*S*/ALPHANUM><ALPHA =NUM>, the character *S* denotes a summing block, where gains are ±1. Other blocks are defined in a similar way.

<GAIN>::=*G*<*C*/*U*/ALPHANUM> <ALPHANUM>, the character *C* indicates a controllable gain-block where output is expressed in terms of variables *Y, X* as (*Y/X*); *U* indicates an adaptive gain-block where gain is obtained through interpolation.

<FILTER>::=*F*<*U*/ALPHANUM> <ALPHANUM>, where *U* indicates an adaptive filter block where the filter coefficients are obtained through interpolation.

<SWITCH>::=*S*<ALPHANUM> <ALPHANUM>; this block is similar to a single-pole, multi-throw switch.

<QUANTIZE>::=*Q*<*X/L*> <ALPHANUM> :*X,L* indicate *A/D* block and *D/A* block respectively.

<DECISION>::=*D*<*O*/ALPHANUM> <ALPHANUM>, where *O* indicates an observer block, (see Sec 6 also).

Now, an amplifier may be completely specified as:

<AMP>::=<AMPLIFIER><SYMBOL><REALN UMS><SYMBOL> representing block-name, inputs, gain-values and ouput. Other blocks are defined in a similar way.

Dialogue procedures have been developed so that the specifications are entered in the required sequence. The sequence of specification entry is determined from the block selected by the user and hence the syntax check can be made based on the definitions given in the preceding paragraphs. Procedures to handle SYMBOL(S), REALNUM(S), BLOCK have been designed as a part of parsing algorithms, which also check and warn the entry of duplicate block-names and output symbols.

4.2 Adaptive Block

The control law is generally obtained by using optimization techniques at typical flight conditions spanning the flight envelope. As a result, at any other flight condition, the scheduling calls for single or double

interpolations. The interpolation formulae[8] are used to estimate the parameters . (gain values or fliter coefficients) for any given value of the grid variable(s). There are two assembly procedures INTRPL1 and INTRPL2 for estimating the interpolated values based on single or double grids. The scheduled values $Y_1, .., Y_i$ .. are assumed to be stored in ROM in short-real format at discrete values X_L, X_L+X_{ND}, X_L+2X_{ND} ... X_L+iX_{ND}.. of selected parameter like dynamic pressure or mach-altitude grid pair. In adaptive gain or filter blocks, the gain value or the filter coefficients are to be interpolated as a function of grid variable(s) X (or X, Y). The adaptive block is mechanized by obtaining the following information: single (or double) interpolation scheme, grid variables, the starting value of X-axis grid X_L (or X and Y axes grids X_L, Y_L), the incremental values of X-axis grid X_{ND} (X_{ND}, Y_{ND}); the total number of bytes/coefficient N and filter order m are required in case of filter block; the total number of X-axis grid points is required in case of double-grid gain-blocks. These details enable the calculation of gain value or filter coefficients based on any given value of X (or X, Y). The automated aid handles the scheduling of blocks by defining the syntax for these blocks in terms of the above entities.

5. CODE GENERATION

The code generator generates 8086/8087 processor-based (i) data-segment where constants and variables with correct size and type are declared and (ii) code-segment where the code for the functional part of each block is generated.

5.1 Filter Discretization

As the controller is implemented in digital domain, the filter blocks use continuous to discrete-domain transformation. Pre-warped Tustin transformation of s to Z domain has been used due to its valid cascaded property and widespread usage in FCS[3,9]. A procedure in the automated aid package obtains digital filter coefficients a_i, b_i from s-plane transfer function, viz. $H(s) = \Sigma^n_{i=0} A_i s^i / \Sigma^m_{i=0} B_i s^i$ and $m > n$ in general and A_i, B_i are real. The Tustin transformation $s <-> 2(Z\text{-}1)/(KT(Z+1))$ where T = sampling interval and K = Pre-warping correction factor, gives output $Y(nT)$ as $Y(nT) = \Sigma^m_{i=0} a_i X_i(n\text{-}i)T - \Sigma^m_{i=1} b_i Y_i(n\text{-}i)T$.

5.2 Algorithms for Obtaining the Size of the Variable

As each block is parsed, the entities — the inputs, outputs and control variables — are entered into an aggregate array INPUTS by invoking a procedure INSERT. INSERT, in turn, invokes a function OKIN passing each entity as an argument. If OKIN finds the entity in the array INPUTS, then it returns false; else it returns true. If OKIN returns a true, then INSERT updates the array INPUTS with the entity

An array FILTORD of integer contains the length of the entity at the index corresponding to same entity in the array INPUTS. Whenever OKIN returns a false, it also gives the index of the array where the entity is stored. The length of the variable gets updated, if the order of the current filter to which it is connected is more. By this procedure, it is possible to keep track of the route of the variable so that its maximum length can be determined exactly.

5.3 Constants in Data-segment

Constants like filter coefficients, filter order, amplifier gains, etc. will have to be defined in the data-segment. The block-name being unique is used by prefixing it to these constants so that duplication errors are avoided. The filter coefficients a_i, b_i of filter F00 are thus defined as FOO _ NUMR _ COEF and FOO _ DENMR _ COEF. This technique is followed while defining other constants.

A procedure in the translator declares all the variables in the data-segment with the correct size based on the contents of the array INPUTS and FILTORD.

5.4 Code-segment Generation

In code-segment, the macro-calls are generated in the order corresponding to each block specified in the block diagram. The inputs, outputs of filters are updated before the termination of the translated assembly program. The macros are all predefined and the macro-assembler[10] takes care of differing lengths of the passed arguments by using EXITM directive. Table 1 lists the macros necessary to generate the codes for the functional part of each block. The automated aid generates the macro-call statement with the appropriate arguments collected during the parsing stage. The F8 Cstar block diagram, sketched in Fig. 1 and Fig. 2(a), shows its specification file EXP.DAT. Now, the automated aid is invoked by calling its command file AA in DOS environment. AA generates the code-segment part in AFCS.ASM and data part with initialization in AFCS.DSG. The data part and the

predefined macros are all integrated by using an 'include' directive. Parts of these files are shown in Fig. 2(b).

6. FAULT DETECTION MEASURES

6.1 Observers for Handling Sensor Failures

Analytic redundancy based on observer has been studied for reconstruction of flight control sensors in aircraft[11]. In this paper, it is shown how the automated aid can handle the sensor failure using observers.

Table 1. List of MACROS and their functions

Macro	Function
UP	$X(n-i)<-X(n-i+1)\ i=0$ to N
UPDATE	
AMPL	$Y=\sum a_i X_i,\ a_i \longrightarrow$ real
SUMPN	$Y=\sum a_i X_i,\ a_i=\pm 1;\ m\leq 12$
GAIN	GAIN BLOCK, $y=gx$
GAINC	$Y=(m/d)\,x$
OBS	Observer
CPM	Value-fault detection
FILTER (ORDER = 1) (ORDER = 2) (ORDER < 6)	$Y(n)=\sum_{i=0}^{m} a_i X(n-i)-\sum_{i1}^{m} b;\ Y(n-i)$
SWITCH	Similar to single-pole, double-throw switch
SCH 1 (1 GRID) GAIN BLOCK FILTER BLOCK	Single-grid interpolation
SCH 2 (2 GIRDS) GAIN BLOCK FILTER BLOCK	Double-grid interpolation

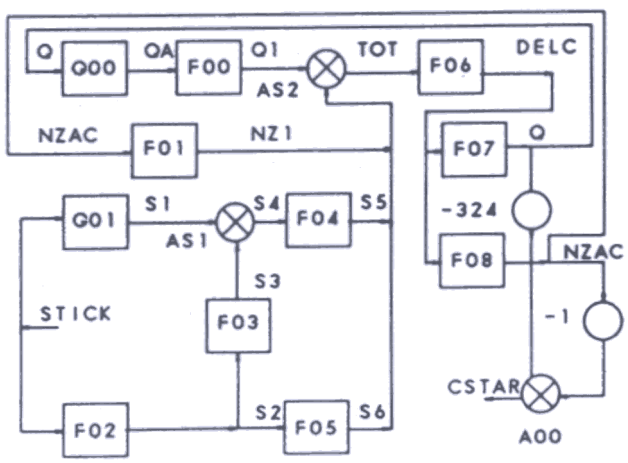


Figure 1 Block diagram of F–8 Cstar controller.

A short-period approximation of longitudinal dynamics of F–8 aircraft has been taken as the plant for the design of the observer. The signature Camberra metric[12] is filtered by a geometric moving average filter.

If the system and observer states are X and $\hat{X}$, then the difference between the outputs of the system and observer will be $C(X-\hat{X})$, so that the observer is given by

$$\dot{\hat{x}} = A\hat{x}(t) + BU(t) + L[Y(t) - C\hat{x}(t)]$$
$$= (A-LC)\hat{x}(t) + BU(t) + LY(t).$$

Where $X=[q\ N_z\ \delta_e]^T$ (1)

where L is chosen such that $(A-LC)$ has stable eigen values placed further away from the eigen values of A. The poles of $(A-LC)$ are placed at $-37.5 \pm j37$ and -37.5. The values of matrices A and B at flight condition (altitude of 20,000 ft and mach no = 0.67[3]) are given in the following after suitable transformation of the axis.

$$A=\begin{bmatrix} -0.616 & 0.008705 & -12.85 \\ 728.1 & -1.05 & -1362 \\ 0 & 0 & -12.5 \end{bmatrix} \quad B=\begin{bmatrix} 0 \\ 1.362+3 \\ 1.25+1 \end{bmatrix} \quad (2)$$

$$C=\begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \quad 3)$$

The output q is assumed to be available through redundant channels. L is found to be

$$[98.33 - 584400\ 146.1]^T \quad (4)$$

$\hat{X}(s) = H(s)U(s) + G(s)Y(s)$, where $H(s) = (sI - A + LC)^{-1} B$ and $G(s) = (sI - A + LC)^{-1} L$ and then the Tustin equivalent of each of the states of X is obtained. By using ORACLS[13] and data given by Eqns 2 to 4, $N_z(s)$ is given by

$$N_z(s) = H(s)\,\delta_c(s) + G(s)q(s) \quad (5)$$

where

$H(s)=1362[s^2+98.972s-70881]/(s^3+112.5s^2+45709s+64730)$

$G(s)=-595500(s^2+13.33s+8.716)/(s^3+112.5s^2+{}^{2}+45709s+64730)$

6.2 N_z Sensor Failure Detection

For detecting N_z sensor failure, the signature Camberra metric. $S = (\hat{N}_z - Nz\text{sense})/(\hat{N}_z + Nz\text{sense})$ is used. A fixed weight W is given to current observation of S and complementary weight $(1 - W)$ is given to geometric average of all the subsequent observations. Then geometric average at kth sampling instant is given by $S^*_K = W S_K + (1-W) S^*_{K-1}$

```
G00     Q; 324; QA;
F00     QA; 0;75; 1; 0;4167; 1; Q1;
F01     NZAC;1;0;104175;0;6667;1;NZ1;
G01     STICK;0;729;S1;
F02     STICK;55;1;13;349;55;S2;
F03     S2;0;071;0;1;S3;
AS1     S1;S3;1;1;S4;
F04     S4;0;4791;0;0;4167;1;S5;
F05     S2;0;4791;1;0;4167;1;S6;
AS2     Q1;NZ1;S5;S6;1;1;1;1;TOT;
F06     TOT;0;000515;0;001185;1,0;DELC;
F07     DELC;-172.5,-168.7;1,14.17,27.81,87.37;Q;
F08     DELC;1362.5,839;28;-117100;1,14;17;27;81;
        87;37;NZAC;
A00     NZAC;Q;-1;-324;CSTAR;
```

Figure 2(a). The specification file exp.dat corresponding to the block diagram sketched in Fig. 1

```
INCLUDE AFCS MAC
SX:
@GAIN Q,QA,G00
@FILTER 1;QA,Q1;F00,1
@FILTER 1,NZAC,NZ1,F01,2
@GAIN STICK,S1,G01
@FILTER 1,STICK,S2,F02,2
@FILTER 1,S2,S3,F03,1
@SUMPN S4,2,S1,1,S3,1
@FILTER 1,S4,S5,F04,1
@FILTER 1,S2,S6,F05,1
@SUMPN TOT,4,Q1,1,NZ1,1,S5,1,S6,1
@FILTER 1,TOT,DELC,F06,1
@FILTER 1,DELC,Q,F07,3
@FILTER 1,DELC,NZAC,F08,3
@AMPL A00,CSTAR,2,NZAC,Q
@PREPUPDATE 4, 4,NZAC,NZ1,STICK,S2
@PREPUPDATE 5, 2,Q,DELC
@UP  7,QA,2,Q1,2,S3,2,S4,2,S5,2,S6,2,TOT,2
INCLUDE AFCS.EPI
```

The code-segment AFCS.ASM generated by the automated-aid

```
@DSG G00_GAIN,<  324.000000>
@DSG F00_NUMR_COEF,<   1.7810149,  -1 7339037>
@DSG F00_DENMR_COEF,<  -0.9528888>
@DSG F01_NUMR_COEF,<   0.0009105,   0.0018211,   0.0009105>
@DSG F01_DENMR_COEF,<  -1.8755781,   0.8792203>
@DSG G01_GAIN,<    0.729000>
@DSG F02_NUMR_COEF,<   0.0048763,   0.0097527,   0.0048763>
@DSG F02_DENMR_COEF,<  -1.7450220,   0.7645274>
@DSG F03_NUMR_COEF,<   7.0629823,  -7.0629823>
@DSG F03_DENMR_COEF,<   1.0000000>
```


(Contd)

```
@DSG F04_NUMR_COEF,<    1.1226650,     .1226650>
@DSG F04_DENMR_COEF,<  -0.9528888>
@DSG F05_NUMR_COEF,<    1.1462206,     .0991094>
@DSG F05_DENMR_COEF,<  -0.9528888>
@DSG F06_NUMR_COEF,<    0.0005269,  -0.0005031>
@DSG F06_DENMR_COEF,<  -1.0000000>
@DSG F07_NUMR_COEF,<  -0.0153689,  -0.0156682,   0.0147704,   0.0150697>
@DSG F07_DENMR_COEF,<  -2.7409867,   2.4927177,  -0.7511111>
@DSGW F07_NUMR_TOT,< 4>
@DSG F08_NUMR_COEF,<  11.9285535, -12.1958828, -12.3439784,  1 .7804579>
@DSG F08_DENMR_COEF,<  -2.7409867,   2.4927177,  -0.7511111>
@DSGW F08_NUMR_TOT,< 4>
@DSG A00_GAIN,<   -1.000000, -324.000000
@DSG Q,<0.0,0.0,0.0,0.0,0.0>
@DSG QA,<0.0,0.0,0.0>
@DSG Q1,<0.0,0.0,0.0>
@DSG NZAC,<0.0,0.0,0.0,0.0>
@DSG NZ1,<0.0,0.0,0.0,0.0>
@DSG STICK,<10.0,0.0,0.0,0.0>
@DSG S1,<0.0>
@DSG S2,<0.0,0.0,0.0,0.0>
@DSG S3,<0.0,0.0,0.0>
@DSG S4,<0.0,0.0,0.0>
@DSG S5,<0.0,0.0,0.0>
@DSG S6,<0.0,0.0,0.0>
@DSG TOT,<0.0,0.0,0.0>
@DSG DELC,<0.0,0.0,0.0,0.0,0.0>
@DSG CSTAR,<0.0>

  data-segment AFCS.DSG generated by the automated-aid
```

Figure 2(b). The code-and data-segement statements generated by the automated aid.

The geometric moving average process is considered equivalent to operation of filter where the output is given by

$y(n) = Wx(n) + (1\text{-}W)\, y(n\text{-}1)$. The transfer function of the equivalent analog filter is $\{s + (2/T)\}/\{(s(2W)/W)(2/T)\}$. The observer forms a subclass of the decision block specified as DO<ALPHANUM>. The full syntax of the observer block is so defined to enable extraction of control input, normal sensor signal, back-up signal, output signal, limits for switching to analytical value, and time after which the observer is to be used. The time factor is to ward off the false alarm which may be present either during the initial asymptotic region or whenever a new model of the plant is inserted due to change in the flight condition. This parameter may not be required if interpolation of the observer can be done at an acceptable rate.

6.3 Timing-fault, Value-fault and Self-test

The automated aid, apart from translating the FCS block diagram, enables detection of two important types of faults as a part of in-flight monitoring, viz. value faults and timing faults. This is done through a set of pre-defined macros. It is possible to obtain a minimal instruction set by which one can express the complete set of predefined macros. This way of organizing the translator enables easy design of processor self-test.

6.3.1 Timing-fault Detection

If the execution time lies between a maximum of t_{max} and a minimum of t_{min} and the current sample of the execution time of the given control law is T_f then if $(t_{min} < T_f < t_{max})$ then timing fault is false; else timing fault is true.

Determination of t_{min} and t_{max} has to be carried out based on macros defined for various blocks. Since the macrocall contains relevant information, it will be easy to obtain the execution time by the translator itself. Table 2 gives the list of macros that have been used and the execution time in terms of number of clock cycles of 8086/8087 pair. The translator obtains the total execution time in terms of number of clock cycles. The execution time found out could set a watch dog timer. Verification of timing-fault detection has been done by an interrupt service routine which makes use of timer ticks available on IBM PC.

Table 2. List of MACROS and their execution time

MACRO	EXECUTION TIME
UP	17 + (21 + 150 * TOT) * ORDER
UPDATE	144 * N
AMPL	268 + (NUM–1) * 266
SUMPN	109 + S * 139
GAIN	268
GAINC	494
OBS	667
CMP	658
FILTER (ORDER = 1)	884
(ORDER = 2)	1410
(ORDER < 6)	113 + 290 * (2 * ORDER + 1)
SWITCH	153 + 436 * I)
SCH 1 (1 GRID)	
GAIN BLOCK	3186
FILTER BLOCK	3798 + 2237 * ORDER
SCH2 (2 GIRD)	
GAIN BLOCK	6963
FILTER BLOCK	7002 + 4750 * ORDER

TOT Numbers of inputs and outputs connected to filter blocks of a given order greater than 2; I : number of throws to the switch block; N : 1 or 2 (filter blocks of order less than 3; NUM : number of inputs to an amplifier; and S : number of inputs to a summing block

6.3.2 Value-fault Detection

The value fault is defined by the condition that the variable V does not satisfy the relation $V_{min} < V_{max}$ where V_{min} and V_{max} refer respectively tc the minimum and maximum excursions of the variable V. The variable V is often a physical parameter like pitch angle, pitch rate and normal acceleration. The parser checks the syntax of the decision block which consists of following entities: control signal, signal to be switched if upper limit is crossed, signal to be switched if lower limit is crossed, output signal and upper and lower limits, V_{max} and V_{min}.

6.3.3 Instruction Set Self-test

The instruction set self-test is often monitored in-flight[6]. It will be possible to obtain the basic set (Fig. 3) which has been used for defining the macros. An efficient test-program containing the basic instructions is then executed to generate certain test results and are checked against the expected values. Any deviations can be used to change the flag conditions.

7. VALIDATION OF THE TRANSLATOR

The model following Cstar controller sketched in Fig. 1 forms the FCS block diagram to be used for validation. Simulation based on the discrete state-space model of Cstar controller of short period dynamics of F–8 aircraft is compared with that generated by the automated aid at different sampling rates. As a part of the validation, the N_z sensor fault-detection is introduced and the performance of the closed loop with reconstructed N_z is compared with that of discrete simulation. Timing-fault and value-fault detection algorithms are next tested.

7.1 Reduction to Standard Form

The problem of discretizing the continuous time state equations $\dot{X} = AX + BU$ and $Y = CX$ consists of finding a discrete-time model $X(k+1)T = MX(k)T + NU(k)T$ and $Y(k)T = C\,X(k)T$, where $X(k)T$ and $X(k+1)T$ represent state vectors at sampling instants kT and $(k+1)T$. The solutions of M and N are well known and are given by $M = \exp(AT)$ and $N = \int_0^T \exp(Av)\,dv\,B$. Figures 4 and 5 represent the standard forms of the regulator-plant cascade. The regulator and plant portions have subscripts r and p respectively. The following are the notations: input and output vectors U_p, Y; feedback error signals U_r and U_c state vectors X_r and X; regulator is specified by A_r, B_r, C_r and D_r and plant specified by A_p, B_p, C_p and D_p.

The F–8 Cstar controller block diagram has to be reduced to the state-space form

$$\begin{bmatrix} \dot{X} \\ \dot{X}_r \end{bmatrix} = A \begin{bmatrix} X \\ X \end{bmatrix} + B\,U_p \qquad (6)$$

It can be shown that[14]

The following are the basic instructions that have been used for designing macros. memb,memw,memr refer to memory byte location,memory word location and short-real memory location respectively. imm refers to immediate value as one of the operands.

```
ADD SI,imm          ADD BX,imm          ADD SI,memw
ADD memw,AX         ADD AX,memw         AND memw,imm

     procedure

    memw,imm        CMP memb,imm        CMP SI,imm

    CX              DEC memw

FADD memr           FADDP ST(1),ST      FCOMP memr
FCOMP memr[SI]      FDIV memr           FILD  memw
FISTP memw          FLD mem             FLD mem[4]
FLD mem[SI][4]      FLD mem[SI][ROM]    FLDZ
FMUL memr           FMUL memr[4]        FMUL mem[SI]
FMULP ST(1),ST      FST memr            FSTP memr[4]
FSTP memr           FSTP memr[SI]       FSTP memr[BX]
FSTSW memw          FSUBP ST(1),ST      FSUB memr

    AX

JBE short label     JE short labe       JMP short  abel
JMP far ptr label   JNE label           LOOP labe

MOV AL,imm          MOV AX,SEG ESEG     MOV AX,memw
MOV BX,imm          MOV CL,imm          MOV CX,imm
MOV DS,AX           MOV ES,AX           MOV SI,imm
MOV SI,memw         MOV SI,AX           MOV SI,CX
MOV memb,imm        MOV memw,imm        MOV memw,SI
MUL memw

                    RET

    AX,CL           SHL SI,             SUB SI,imm
```

Figure 3. The set of instructions of 8086/8087 processors that are used in the macros.

$$A = \begin{bmatrix} (A_p + B_p K_p) & B_p C_r \\ O & A_r \end{bmatrix} \quad (7)$$

$$B = [B_p D_r : B_r]^T \quad (8)$$

$$X = [q\, N_z \delta e\, N_z 1\, N_z 1\, q1\, p]^T \text{ where } p = \delta_c(s)/(K1s + K2)$$

$$X_r = [z \dot{z} \ddot{z}] \text{ and } c(s)/(0.3493s^3 + 6.538s^2 + 45.56s + 55) =$$

$$U(s)/(0.41767s^3 + 6.562s^2 + 36.265s + 55) = Z \quad (9)$$

Equation (6) represents the standard form and one can obtain its discrete time model. ORACL-based programs[14] have been developed which give the discrete

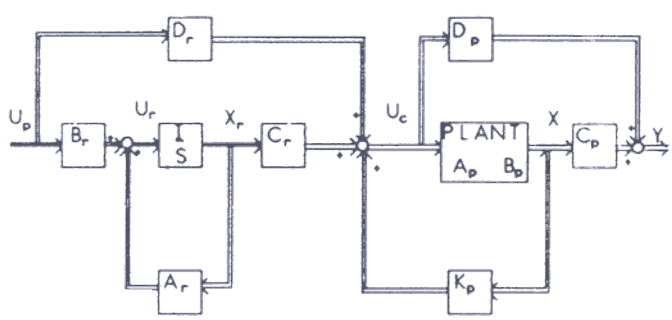
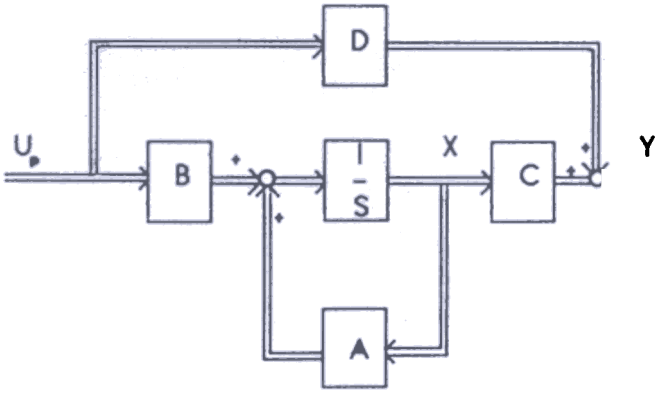


Figure 4. Regulator and plant cascade.



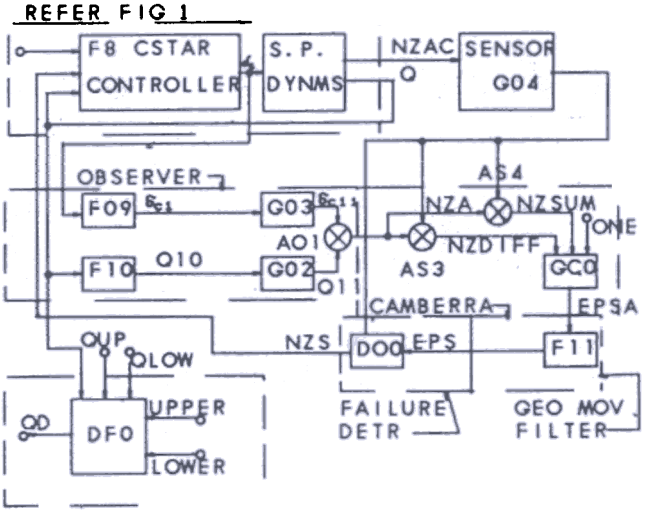


Figure 6. Block diagram of F–8 Cstar controller with sensor failure detection and reconfiguration.

```
G00     Q; 324; QA;
F00     QA; 0.75; 1; 0;4167; 1; Q1;
F01     NZS;1;0;104175;0;6667;1;NZ1;
G01     STICK;0;729;S1;
F02     STICK;55;1;13;349;55;S2;
F03     S2;0.071,0;1;S3;
AS1     S1,S3;1;1;S4;
F04     S4;0;4791;0;0;4167;1;S5;
F05     S2;0;4791;1;0;4167;1;S6;
AS2     Q1;NZ1;S5;S6;1;1;1;1;TOT;
F06     TOT;0;000515;0;001185;1;0;DELC;
F07     DELC;-172;5;-168;7;1;14;17;27;81;87;37;Q;
F08     DELC;1362;5;839;28;-117100;1;14;17;27;81;87;37;NZAC;
A00     NZAC,Q;-1;-324;CSTAR;
F09     DELC;1;98;72;-70881;1;112;5;4570;64730;DELC1;
F10     Q;1;13;33;8;716;1;112;5,4570;64730;Q10;
G02     Q10;-595500;Q11;
G03     DELC1;1362;DELC11;
A01     DELC11;Q11;1;1;NZA;
G04     NZAC;1;NZ;
AS3     NZ;NZA;1;-1;NZDIFF;
AS4     NZ,NZA;1;1;NZSUM;
GC0     ONE; NZDIFF; NZSUM; EPSA;
F11     EPSA;1;100;9;100;EPS;
DO0     EPS;NZ;NZA;NZS;-0;9;0;9;100;
DF0     Q;QUP;QLOW;QD;-0;09;0;09;
```

Figure 7. The specification file corresponding to the block diagram sketched in Fig. 6 block DFO performs value-fault detection.

time model of the Cstar controller and the responses of q, N_z for a step input at various sampling rates. These are stored for comparison with the Tustin equivalent. Figures 6 and 7 show the block diagram with sensor fault-detection and reconfiguration. It can be noticed that the Camberra metric is used as the signature. The filter F–11 implements the geometric moving average of the signature discussed earlier. The results of the

sensor failure and reconfiguration at the sampling rate of 50 Hz is shown in Fig. 8 and it has been compared to the healthy sensor case.

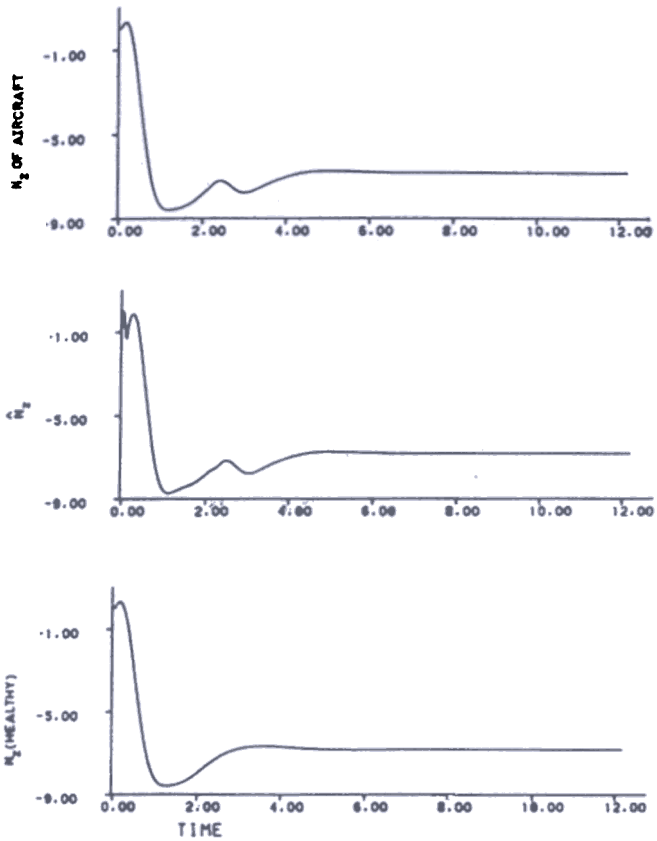


Figure 8. N_z **response with sensor failure at 2 s.**

8. CONCLUSIONS

The automated aid (i) can handle most of the blocks that exist in FCS, (ii) does not need sophisticated work station and can work on IBM PC or INTEL microprocessor development systems and (iii) has potentiality for use in autonomous systems and can aid in generation of N-version program for improved reliability.

ACKNOWLEDGEMENTS

The authors would like to thank the Director, NAL and Head, ALD for permission to publish this paper. The authors would also like to thank Dr A Krishnan, Dr A Pedar, Mr Shyam Chetty and Mr A Shantharam for helpful discussions.

REFERENCES

1. Rama Murthy, T.V. & Seshadri, V. An automated aid for microcomputer implementation of flight control systems. *In* Proceedings of the National Systems Conference, December 1988.
2. Symposium of Microprocessor Applications in Future Aerospace Systems, *IEEE/AESS,* 1981.
3. Hartmann, G.L.; Hauge, J.A. & Hendrick, R.C. F8C digital CCV flight control laws. NASA, Washington, D.C., 1976. NASA CR-2629.
4. Flapper, J.A. & Throndsen, E.O. L-1011 flight control system. AGARD AG-224, Integrity in electronic flight control system, 1977. Paper 22.
5. Carleton, D.L. F-16 flight control system development. AGARD AG-224, Integrity in electronic flight control system, 1977. Paper 19.
6. Bailey, D.G. & Folkesson. JA-37 digital automatic flight control system (DAFCS) self-test development. AGARD AG-224, Integrity in electronic flight control system, 1977. Paper 20.
7. Gauthier, R.L. & Ponto, S.D. Designing systems programs. Prentice Hall, New Jersey, 1970.
8. Schindler, T.M.; Johnston, A.M. & Keith, G.W. AFTI/F-16 DFCS development summary—A report to Industry—Software design/mechanization, *IEEE, NAECON,* 1983. p. 1227.
9. Slater, G.L. A unified approach to digital flight control algorithms. Paper presented at the AIAA Mechanics and Control of Flight Conference, August 1974. Paper 74-884.
10. Microsoft macroassembler for MS-DOS. Microsoft Corp, 1985.
11. Shapiro, E.Y.; Schenk, F.L. & Decarli, H.E. Reconstructed flight control sensor signals via Luenberger observers, *IEEE,* 1979, AES-155, 245-52.
12. Pau, L.F. Failure analysis and performance monitoring. Marcel Decker, 1981.
13. Armstrong, E.S. ORACLS—A design system for linear multivariable control. Marcel Dekker, 1980.
14. Rama Murthy, T.V. An automated aid for microcomputer implementation of flight control systems. Indian Institute of Technology, Madras, 1989. MS Thesis.