NASA Technical Memorandum 104258

A Rule-Based System for Real-Time Analysis of Control Systems

Richard R. Larson and D. Edward Millard

October 1992

National Aeronautics and
Space Administration

NASA Technical Memorandum 104258

A Rule-Based System for Real-Time Analysis of Control Systems

Richard R. Larson
NASA Dryden Flight Research Facility
Edwards, California

D. Edward Millard
Computer Science Corporation
Edwards, California

1992

National Aeronautics and
Space Administration

Dryden Flight Research Facility
Edwards, California 93523-0273

A RULE-BASED SYSTEM FOR REAL-TIME ANALYSIS OF CONTROL SYSTEMS

Richard R. Larson
NASA Dryden Flight Research Facility
Edwards, California 93523-0273

D. Edward Millard
Computer Science Corporation
Edwards, California 93523-0273

Abstract

An approach to automate the real-time analysis of flight critical health monitoring and system status is being developed and evaluated at the National Aeronautics and Space Administration Dryden Flight Research Facility. A software package was developed in house and installed as part of the extended aircraft interrogation and display system. This design features a knowledge-base structure in the form of rules to formulate interpretation and decision logic of real-time data. This technique has been applied for ground verification and validation testing and flight test monitoring where quick, real-time, safety-of-flight decisions can be very critical. In many cases postprocessing and manual analysis of flight system data are not required. This paper describes the processing of real-time data for analysis and the output format, which features a message stack display. The development, construction, and testing of the rule-driven knowledge base, along with an application using the X-31A flight test program, are presented.

Nomenclature

AOA	angle of attack, deg
BIT	built-in test
EHSV	electrohydraulic servo valve
EU	engineering unit
FCC	flight control computer
FCS	flight control system
I/O	input/output
ISV	isolation solenoid valve
LTEFO	left trailing edge flap outboard
LVDT	linear variable displacement transducer
MCC	mission control center
NZ	normal acceleration
OFP	operational flight program
RM	redundancy management
TM	telemetry
XAIDS	extended aircraft interrogation and display system

Introduction

Today's flight control systems are becoming increasingly complex. The ground testing and flight support in the mission control center (MCC) of these systems continues to be more time consuming and costly. Two problem areas are (1) the collection and processing of data and (2) the limited availability of expertise to analyze the information. A direct, raw data conversion in real time into an analyzed result would reduce the human-error element in interpreting the data while also minimizing the amount of postprocessed data. For flight test programs the benefits would result in reduced costs by increasing the sortie rate by minimizing the time to detect and analyze problems. An earlier completion of the test objectives is possible while improving the safety-of-flight monitoring and reducing support personnel. This was a similar goal to support space shuttle flights in the MCC as described in reference 1.

In the MCC traditional real-time displays provide only limited information because of the screen size and number of terminals (fig. 1). Expert knowledge of the system also is required to interpret and analyze the information. An alternative approach would allow a large amount of data to be processed by using a knowledge base which is constructed of rules to formulate conclusions and decision logic. This technique would automatically decide for the user what data to present on a single message display. The term "rule" as defined in this paper is a Boolean expression which may contain any relational or logical operators supported by the C programming language. This concept provides the user the ability to quickly and accurately monitor system parameters such as health, status, configuration, and pilot advisory information. The evolution of this utility is consistent with the findings of the case study reported in reference 2.

The software application that could satisfy this need is a tool such as the extended aircraft interrogation and display system (XAIDS) described in reference 3. The prototype XAIDS was developed in house and demonstrated using the F-18 High Alpha Research Vehicle (HARV) iron bird simulation at NASA Dryden. An improved version of this package was reprogrammed in C language and installed on a UNIX® operating system for continued use by the HARV program. This package was evaluated in the control room for the X-31A, which is described briefly in reference 4.

The XAIDS package is generic; it can be applied to any specific system and is easily portable to any UNIX-based operating system. The conversion from the F-18 HARV to the X-31A program was easily done. The primary differences are the database and the specific knowledge-base logic. This paper describes the development, knowledge-base architecture, testing, and experience using the XAIDS application for the X-31A program. Portions of data from an X-31A flight are presented and analyzed using the XAIDS to demonstrate the tool's capabilities and effectiveness.

XAIDS Messages Design

The XAIDS messages application is a software package which consists of four parts: the knowledge base, parser, database, and message display window. Figure 2 shows these four parts with the bold borders. The database and knowledge-base source files are created for a specific application which is processed by a generic parser. The parser expands the knowledge base into a stand-alone source code which is compiled to form an executable file. This file interfaces with the real-time data input stream and updates the XAIDS messages display at the input data rate. A detailed description of these elements of the system follows.

Knowledge Base

The knowledge base contains the rules which trigger the messages for the message display. Figure 3 shows its general structure. For the application presented in this report the knowledge base consisted of three parts: preprocessing, parameter typing, and rules computation and message generation.

In the preprocessing section the various tests performed before further processing include the following:

1. Verify that the incoming data are live. The live data test requires that the flight control computer (FCC) frame counter be incremented each time the rules file is called. If the counter is constant, a message will be set to indicate a stale data condition and the routine is immediately exited.

2. Test for telemetry (TM) dropouts. The TM data are tested by checking ground station status words to ensure a good signal lock. Data words are tested also to ensure no bits are set in positions that should always be zero. Finally, selected data words are rate checked and compared against a reasonable rate limit threshold. If any of these conditions occur, the messages are not updated.

3. Compute the rule update rate. Determine the difference in the FCC minor frame counter and convert to samples/sec.

4. Determine which FCC channel (1 or 2) is transmitting the data and display that information in a message.

The parameter typing portion of the knowledge file converts desired signals from integer to floating point or vice versa. Scale factors are applied for any raw, fixed point signals to convert to engineering units (EUs). Integers are created from bit masking operations to unpack discrete words for later use in the rule computations.

Arithmetic and Boolean expressions are defined in the rules-computation and message-generation section. Messages are triggered in this section. The rules are written so that they all update with each pass through the knowledge file. Typical C functions are allowed by the parser in these expressions. In addition, customized functions are called in this section.

Parser

The parser is a program written in C language which expands the grammar and structure of the knowledge-base file into additional C code for compilation into an executable module. It reduces the workload of the knowledge-base developer by performing the following tasks:

1. Eliminates explicit data typing

2. Coordinates message ID tags and message on/off logic by appending C code to rules expressions for the "else" path to reset messages

3. Validates references to the data stream

4. Creates logic to permit data to be input from the spreadsheet for testing or from real-time data interfaces

®UNIX is a registered trademark of AT&T Bell Laboratories, Whippany, New Jersey.

Database

A database file contains a symbolic reference of the data words set containing the format type. The database is used with the knowledge-base file to tell the parser how to interpret external data (integer or floating point). The integer type also includes packed discrete words. Scale factors are included for the raw words for conversion to EUs if the scaled words are not available from the database.

Message Display Window

The XAIDS messages are output to a display window in a stack format. Figure 4 shows an example. As new messages are added at the top of the stack, old messages are pushed down. The mouse is used to scroll through the messages should it exceed the window size. All messages are automatically appended with the time of day as they are added to the stack to provide a log of the events. Colored messages help distinguish categories of events. Rescinded messages change to white for 5 sec before removal from the stack. The older messages below are then pushed up to fill the gap in the stack.

The data display stack contains two types of messages. One type is a textual character string that provides interpreted information. This message is typically triggered by single or multiple logical flags. The multiple-flag version is used for common messages applied to different channels such as quad I/O discretes to eliminate duplication. A single message that contains the embedded channel numbers signifies which channels are triggering the message.

The second type of message is used to output data values that continue to be updated. This type is generally used in combination with a textual message which has been triggered to provide additional information. A typical application of this message type is to send a data value to the message stack whenever a particular limit is exceeded. When that signal is less than the limit, the message is removed from the stack.

An important feature of the XAIDS messages window is the option to write a message log file to a disk for later printing. Other menu options are available to (1) freeze the display, (2) prevent the removal of old messages so they can be examined more thoroughly, and (3) print the current message stack.

Knowledge-Base Generation and Testing

The knowledge-base development process consists of four steps as shown in figure 5. These steps are (1) create a real-time database, (2) develop rules logic, (3) test the rules, and (4) install rules with the real-time interfaces.

The database file defines the symbol names and data types for the parser. Any real-time data that can be monitored by the XAIDS can be added to the database file.

The rules logic are developed from documentation, inspection of flight code, system experts, and from system ground and flight testing experience. The logic that triggers the messages was developed by answering the question, "If event x happens, what information do I want to see?" Figure 6 illustrates this logic. A large amount of data is processed, but only limited information needs to be displayed at a given time depending on the display decision criteria.

The rules are verified statically from a spreadsheet as shown in figure 7. This option is selected by clicking the mouse first on the "rules" and then on the "test" boxes. The spreadsheet is automatically loaded with all the parameters used in the rules file. The values for any parameter may be set from the spreadsheet to verify the rules logic. As rules are satisfied, messages appear in the XAIDS messages window.

Finally, the executable XAIDS file is installed on the MCC real-time processors. Dynamic testing is done by playing back a data file through the XAIDS. The messages are compared with known events at specific times on the file. The update rate of the rules can be determined, and the logic to reject data from TM dropouts is tested.

Rules Development Experience

The development of the rules for an MCC application of a program like the X-31A involved a moderate effort. The construction of rules from packed discrete words and flight limit parameters was very mechanical. Since the knowledge-base developer was not previously familiar with the X-31A FCS, however, considerable time was spent learning the system before translation into rules could be done. An inspection of the flight code and FCC data was necessary to learn how the system worked. The multiple-term expressions and nesting of rules such as the actuator redundancy management (RM) logic was more difficult to construct. A custom routine was written to process a table of 430 fail codes from the X-31A data words into a character string. Another 200 messages were added to the knowledge base to monitor the system health, status, flight limits, and pilot advisories. Testing of the rules logic using the spreadsheet was very easy and took less than one day to complete.

Results and Discussion

The test data presented in this section was obtained from a TM tape playback from an X-31A flight. A message file was generated from that playback, and portions of that data are presented from the preflight built-in test (BIT) and events that occurred during flight.

The X-31A preflight BIT program includes an actuator RM test. To understand the actuator command logic for the trailing edge flap logic, refer to figure 8. Basically, isolation valve (ISV) discretes from FCCs 1 and 2 drive actuator 1, and FCC 3 drives actuator 2. If either FCC 3 or hydraulic system A fails, a command path from FCC 2 is opened to actuator 2 to provide redundancy. These paths are all tested for each surface during the actuator portion of preflight BIT. Table 1 shows the results of the preflight BIT for the left trailing edge outboard flap.

The messages indicate which ISV discretes are failed during preflight BIT and whether the actuator or surface is still functioning. The dash (-) preceding the time indicates that the message has been rescinded. This log provides the engineer better insight and visibility into what preflight BIT is doing and ensures confidence that the actuator RM is working as designed.

To verify if some tests are missing or not working properly is easy. The rules are designed as follows. If both paths to a given actuator are failed, a message for a single link fail is replaced with a message indicating that the actuator has failed. If both actuators have totally failed for a given surface, the actuator failed messages are replaced with a single surface fail message. Should any of these paths fail during flight, the appropriate message will immediately be triggered.

Table 2 shows a portion of the XAIDS messages log file from the flight. This segment of the log file contains a record of surface and flight limits that were exceeded. From 14:03:54 to 14:31:05, FCS limits were exceeded four times: (1) NZ @ 14:03:54, (2) VANE #1 @ 14:23:21, (3) AOA @14:31:01, and (4) VANE #1 @ 14:31:05. Messages were triggered showing what limit was exceeded along with the current value of that parameter. Other information contained in the log file indicates that the pilot requested the spin mode at 14:15:22. This mode was not engaged, however, because the airspeed was greater than 200 knots or the airdata was not failed. At 14:27:34 a continuous ignition command to the engine controller from FCC channel 2 was generated because the angle of attack (AOA) exceeded 30 deg.

Table 1. Excerpt from preflight BIT message log file.

XAIDS Message Log File:

–= Message off

11:00:57:252	LTEFO ACT #2 IS FAILED
11:00:57:402	TOTAL LTEFO SURFACE FAIL; ALL ISV'S ARE DEENERGIZED
11:00:58:102	LTEFO ACT #1 FROM C2 DEENERGIZED; C1 STILL FUNCTIONAL
11:00:58:352	LTEFO ACT #2 FROM C3 DEENERGIZED; C2 STILL FUNCTIONAL
11:00:58:352	LTEFO ACT #2 FROM C2 IS ENERGIZED DUE TO FAILURE OF C3
11:00:58:552	LTEFO ACT #1 IS FAILED
11:00:58:902	LTEFO ACT #1 FROM C1 DEENERGIZED; C2 STILL FUNCTIONAL
–11:01:03:152	TOTAL LTEFO SURFACE FAIL; ALL ISV'S ARE DEENERGIZED
–11:01:03:752	LTEFO ACT #1 FROM C2 DEENERGIZED; C1 STILL FUNCTIONAL
–11:01:04:202	LTEFO ACT #2 FROM C3 DEENERGIZED; C2 STILL FUNCTIONAL
–11:01:04:202	LTEFO ACT #2 FROM C2 IS ENERGIZED DUE TO FAILURE OF C3
–11:01:04:602	LTEFO ACT #2 IS FAILED
–11:01:04:812	LTEFO ACT #1 FROM C1 DEENERGIZED; C2 STILL FUNCTIONAL
–11:01:05:312	LTEFO ACT #1 IS FAILED

Table 2. Excerpt from flight message log file.

XAIDS Message Log File:

−= Message off

14:03:54:227	* NZ = 1.7
14:03:54:227	WARNING - NZ LIMIT EXCEEDED IN R3 MODE; > 1.5G
−14:04:02:767	WARNING - NZ LIMIT EXCEEDED IN R3 MODE; > 1.5G
−14:04:02:767	* NZ = 1.7
14:15:22:431	* AIRDATA HAS NOT FAILED
14:15:22:431	* VTAS = 385.6 KNOTS
14:15:22:431	* VTAS > 200 KNOTS
14:15:22:431	SPIN RECOVERY MODE REQUESTED, BUT NOT ENGAGED BECAUSE
14:15:22:431	C1 :SPIN RECOVERY SELECT
14:23:21:683	* VANE #1 CMD = 26.4
14:23:21:683	*** CAUTION *** VANE #1 CMD >= 26 DEG
−14:23:30:133	*** CAUTION *** VANE #1 CMD >= 26 DEG
−14:23:30:133	* VANE #1 CMD = 27.7
14:27:34:965	* AOA > 30 = 30.4
14:27:34:965	C 2 :CONTINUOUS IGNITION BECAUSE
−14:28:08:475	C 2 :CONTINUOUS IGNITION BECAUSE
−14:28:08:475	* AOA > 30 = 30.0
14:31:01:636	* AOA > 30 = 30.5
14:31:01:636	WARNING - AOA LIMIT EXCEEDED IN BASIC MODE OF 30 DEG
14:31:05:286	* VANE #1 CMD = 28.3
14:31:05:186	*** CAUTION *** VANE #1 CMD >= 26 DEG
−14:31:09:636	WARNING - AOA LIMIT EXCEEDED IN BASIC MODE OF 30 DEG
−14:31:09:636	* AOA > 30 = 30.1
−14:31:10:846	*** CAUTION *** VANE #1 CMD >= 26 DEG
−14:31:10:846	* VANE #1 CMD = 28.3

Concluding Remarks

An in-house development of a rule-based, real-time analysis application program for use on a UNIX-based operating system was developed and demonstrated at the NASA Dryden Flight Research Facility. The motivation for this effort was to improve the safety-of-flight systems monitoring and to reduce the amount of postflight data processing required for both flight and ground testing.

A preliminary evaluation of this concept has proven encouraging. Much of the pressure on control room personnel for routine safety-of-flight monitoring probably will be reduced. The XAIDS detected that several flight limits were exceeded from the flight portion presented. The time tagging of the messages has proven usable in providing an automated time log of events during the flight which is printed postflight. This log helps in determining times for postflight analysis. It would be premature to expect to reduce the number of control room personnel, but certainly the types of parameters that are monitored can be modified which more appropriately require human interpretations.

References

[1]Muratore, John F., Troy A. Heindel, Terri B. Murphy, Arthur N. Rasmussen, and Robert Z. McFarland, "Acquisition at Mission Control," *Communication of the ACM*, vol. 33, no. 12, Dec. 1990.

[2]Malin, Jane T., Debra L. Schreckenghost, David D. Woods, Scott S. Potter, Leila Gohannesen, Matthew Holloway, and Kenneth D. Forbus, *Making Intelligent Systems Team Players: Case Studies and Design Issues*, Volumes 1 and 2, NASA TM-104738, 1991.

[3]Glover, Richard D. and Richard R. Larson, *A Knowledge Based Application of the Extended Aircraft Interrogation and Display System*, NASA TM-4327, 1991.

[4]Mackall, Dale, Ken Norlin, Dorothea Cohen, and Gary Kellogg, "Rapid Development of the X-31 Simulation to Support Flight Testing," AIAA Paper 92-4176, 1992.

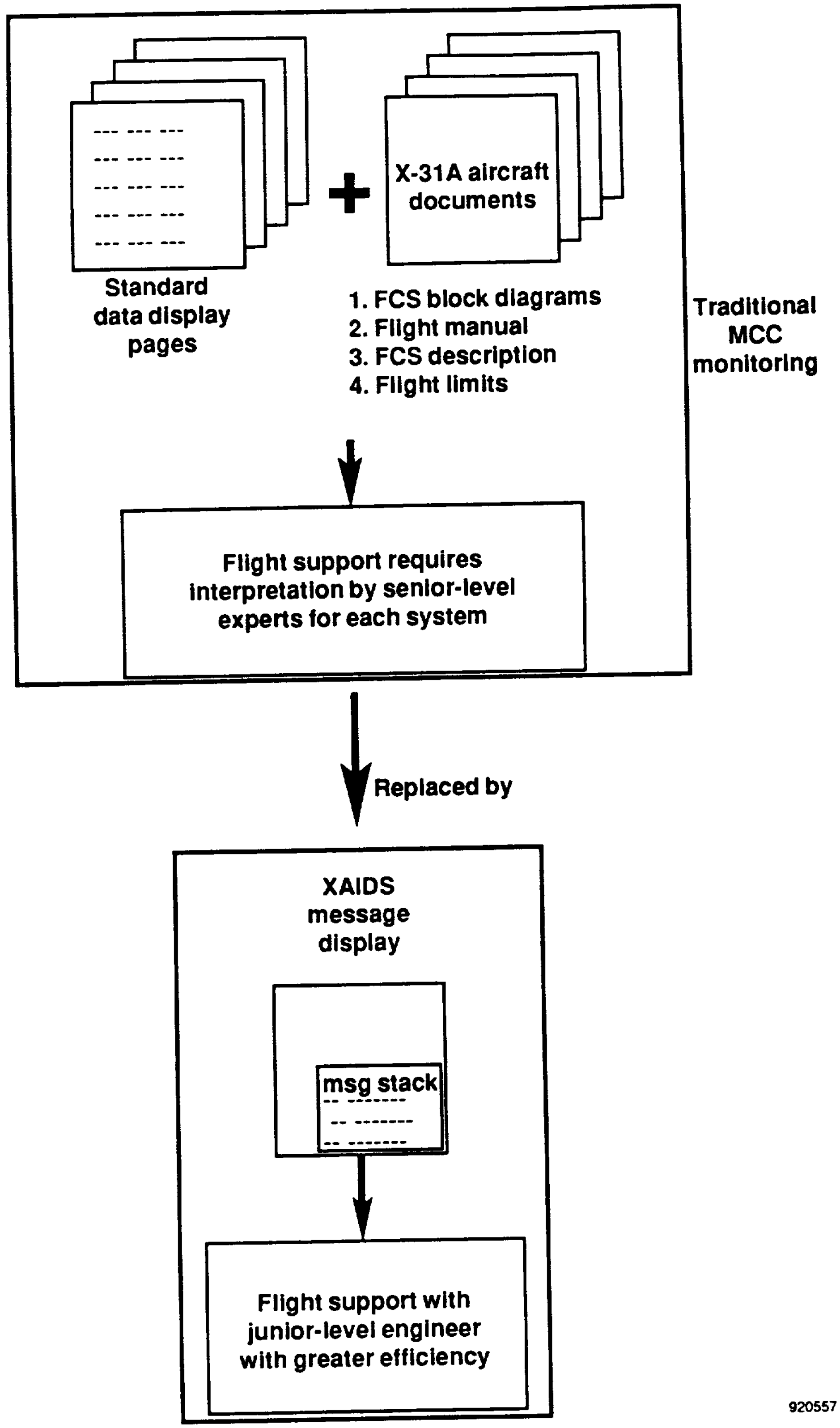

Fig. 1. MCC support using XAIDS application.

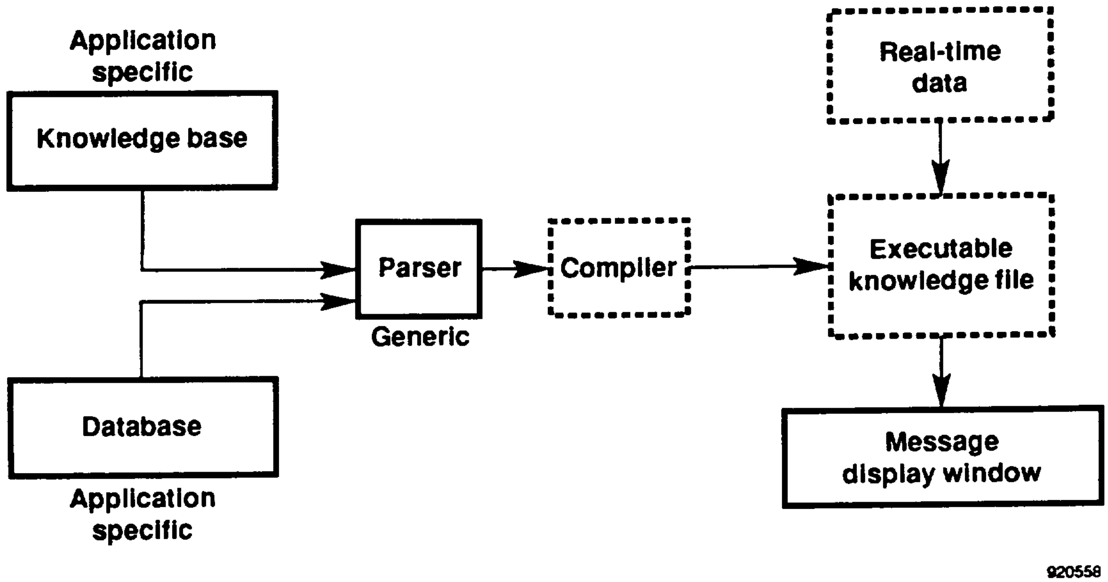

Fig. 2. XAIDS message application design.

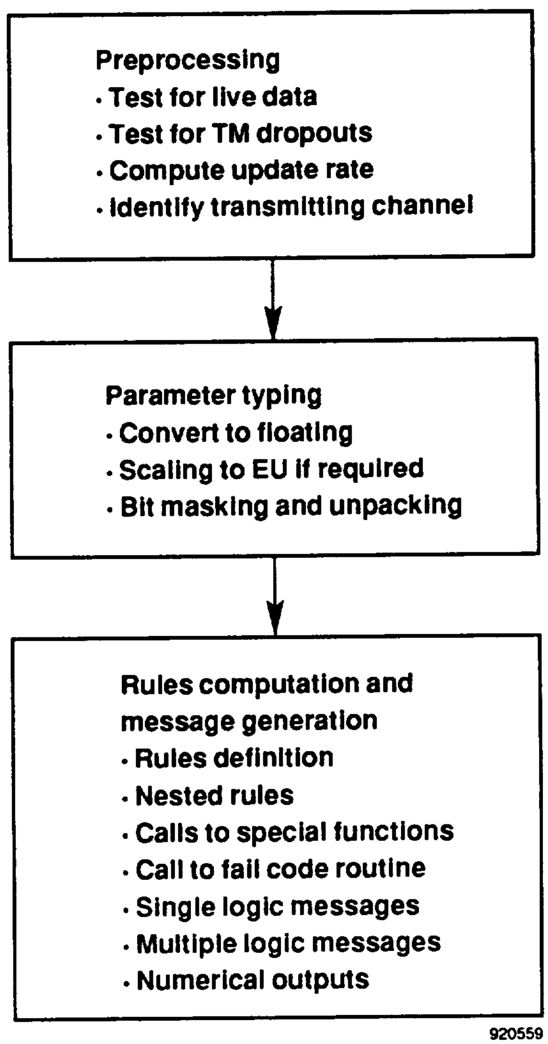

Fig. 3. Knowledge-base execution sequence.

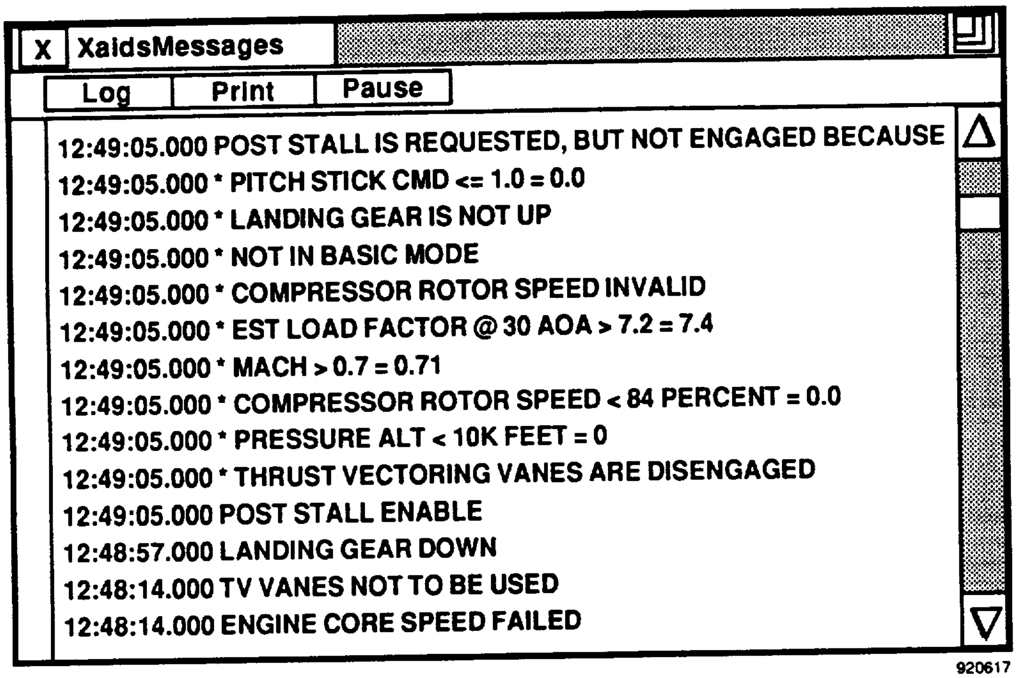

Fig. 4. XAIDS message display window.

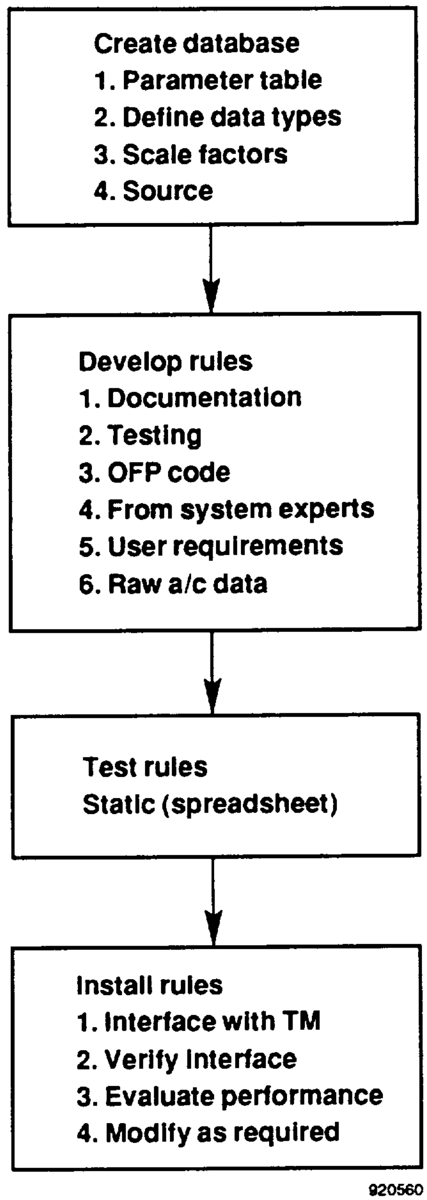

Fig. 5. Knowledge-base development process.

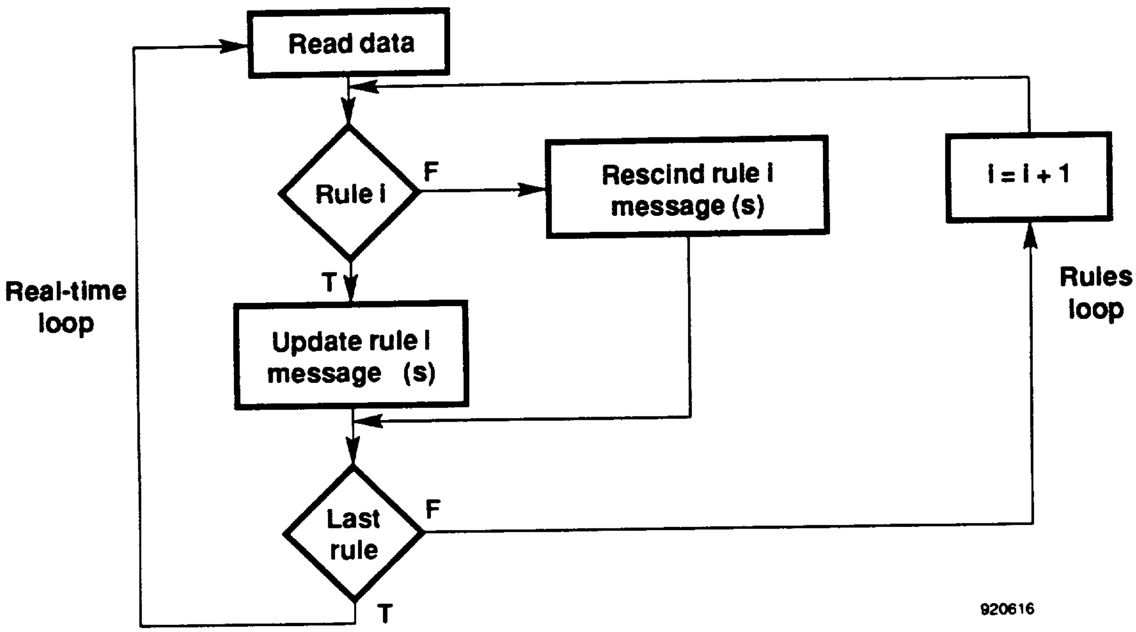

Fig. 6. Real-time loop for rules update.

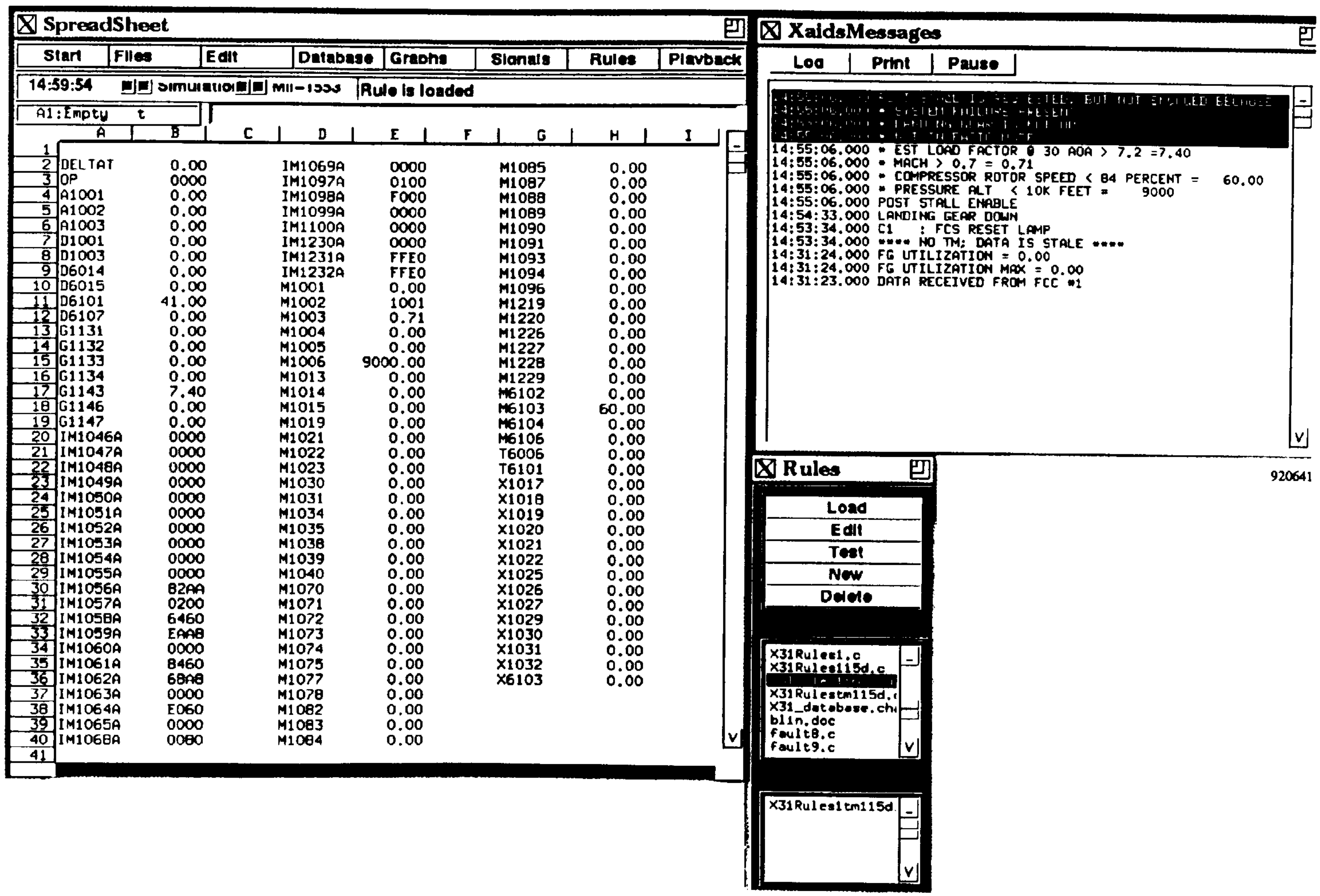

Fig. 7. Rules verification testing.

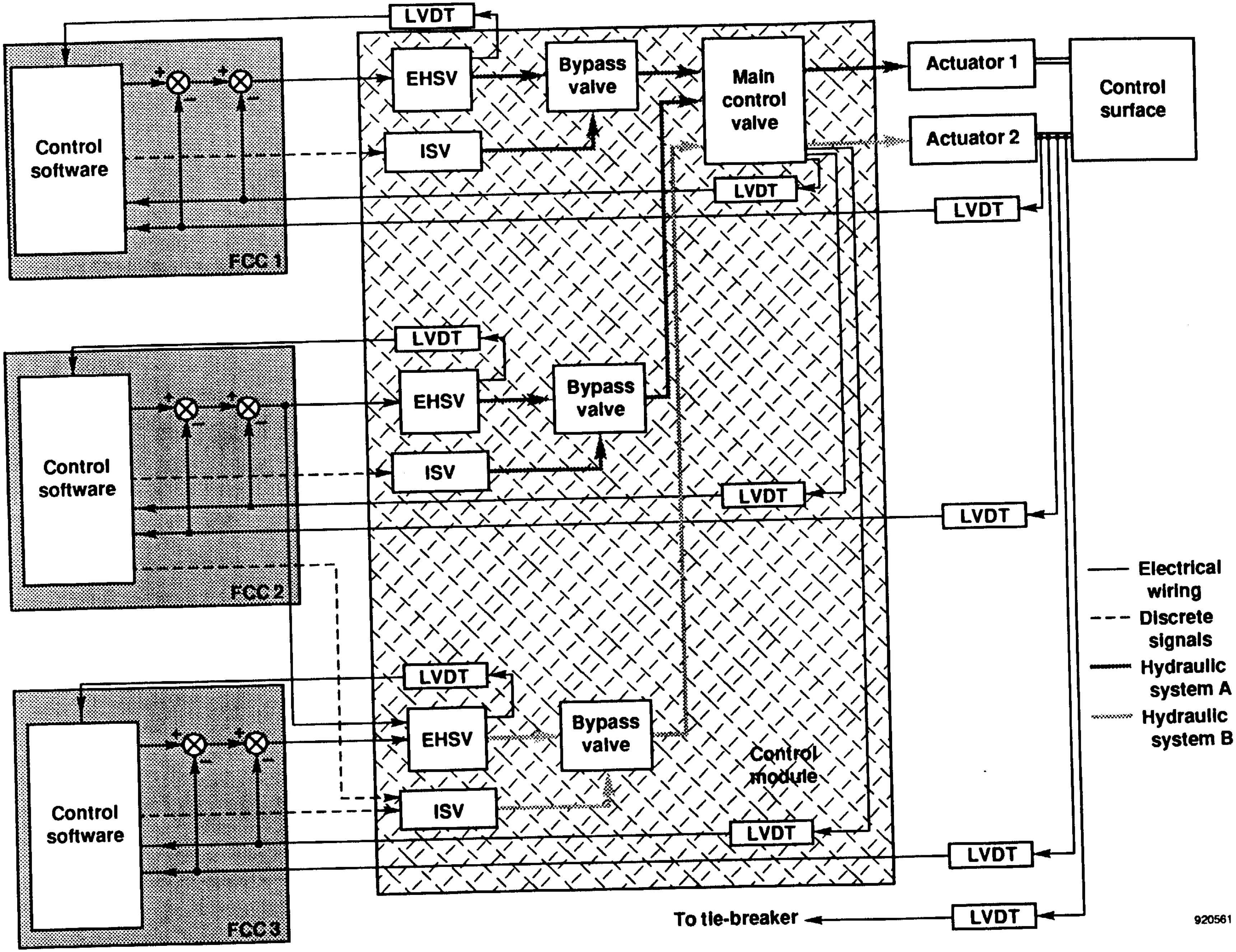

Fig. 8. Trailing edge flap actuation.