NASA Technical Memorandum 85942

Software Modifications to the Demonstration Advanced Avionics System (DAAS)

Bill Nedell and Gordon Hardy

August 1984

National Aeronautics and
Space Administration

```
 33     54     54 AU/RAJU, I. S.
 34  18871  23063 ATL/STRESS
 35   7768   8989 ATL/INTENSITY
    SELECT     ATL/FACTOS
    TERM IN SELECT COMMAND NOT IN DICTIONARY
      You have attempted to SELECT a term which is not a valid
      term in the Thesaurus.  Please try EXPANDing that term to look
      for valid alternatives.
================================================================================
 36   1105   2536 34*35



        INVALID COMMAND-NO HELP SPECIFIED
        INVALID COMMAND-NO HELP SPECIFIED
  1     12     12 AU/NICHOLLS, S.
        INVALID COMMAND-NO HELP SPECIFIED
  2      1      1 RN/NASA-TM-79903
  3      1      1 RN/NASA-TM-85942
          DISPLAY 03/2/1
     84N31105*#   ISSUE 21   PAGE 3327   CATEGORY 4   RPT#: NASA-TM-85942
     A-9713 NAS 1.15:85942   84/08/00  36 PAGES   UNCLASSIFIED DOCUMENT
UTTL: Software modifications to the Demonstration Advanced Avionics Systems
      (DAAS)
AUTH: A/NEDELL, B. F.;  B/HARDY, G. H.
CORP: National Aeronautics and Space Administration. Ames Research Center,
      Moffett Field, Calif.    AVAIL.NTIS   SAP: HC A03/MF A01
MAJS: /*AVIONICS/*DESIGN ANALYSIS/*SOFTWARE ENGINEERING/*SYSTEMS INTEGRATION
MINS: / CHANNELS (DATA TRANSMISSION)/ EVALUATION/ FLIGHT TESTS/ MICROPROCESSORS
ABA:  M.A.C.
ABS:  Critical information required for the design of integrated avionics
      suitable for generation aviation is applied towards software modifications
      for the Demonstration Advanced Avionics System (DAAS). The program
      emphasizes the use of data busing, distributed microprocessors, shared
      electronic displays and data entry devices, and improved functional
      capability. A demonstration advanced avionics system (DAAS) is designed,
      built, and flight tested in a Cessna 402, twin engine, general aviation
      aircraft. Software modifications are made to DAAS at Ames concurrent with
      the flight test program. The changes are the result of the experience
      obtained with the system at Ames, and the comments of the pilots who
      evaluated the system.

ENTER:
```

NASA Technical Memorandum 85942

Software Modifications to the Demonstration Advanced Avionics System (DAAS)

Bill Nedell
Gordon Hardy, Ames Research Center, Moffett Field, California

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

SUMMARY

Ames Research Center initiated a program in 1975 to provide the critical information required for the design of integrated avionics suitable for general aviation. The program emphasized the use of data busing, distributed microprocessors, shared electronic displays and data entry devices, and improved functional capability. A demonstration advanced avionics system (DAAS) was designed, built, and flight tested in a Cessna-402, twin-engine, general aviation aircraft. Software modifications were made to DAAS at Ames concurrent with the flight test program and are documented in this paper. The changes were the result of the experience obtained with the system at Ames, and the comments of the pilots who evaluated the system.

INTRODUCTION

The demonstration advanced avionics system (DAAS) is an integrated, multimicroprocessor-based flight management and control system. DAAS is the result of a program initiated in 1975 by Ames Research Center to demonstrate the feasibility of developing a fully integrated avionics system that would provide the pilot with improved capability and that would be modular, reliable, and easy to maintain. The system has been evaluated in a flight test program at Ames in a Cessna 402B (figs. 1 and 2).

An overview of the program with a summary of the results that led to the DAAS specification is contained in reference 1. A preliminary functional description of DAAS is provided in reference 2, and a detailed description of DAAS is provided in references 3 and 4. Preliminary results of the flight test are contained in reference 5, a complete analysis of the flight test results is contained in reference 6, and a summary of the program results is contained in reference 7.

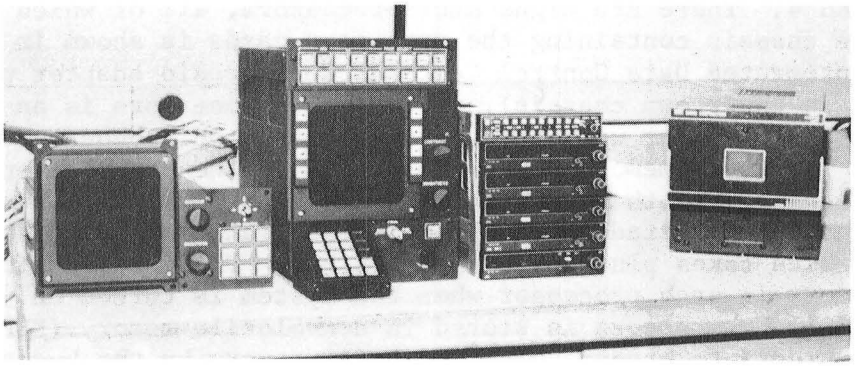


Figure 1.- DAAS equipment on test bench.

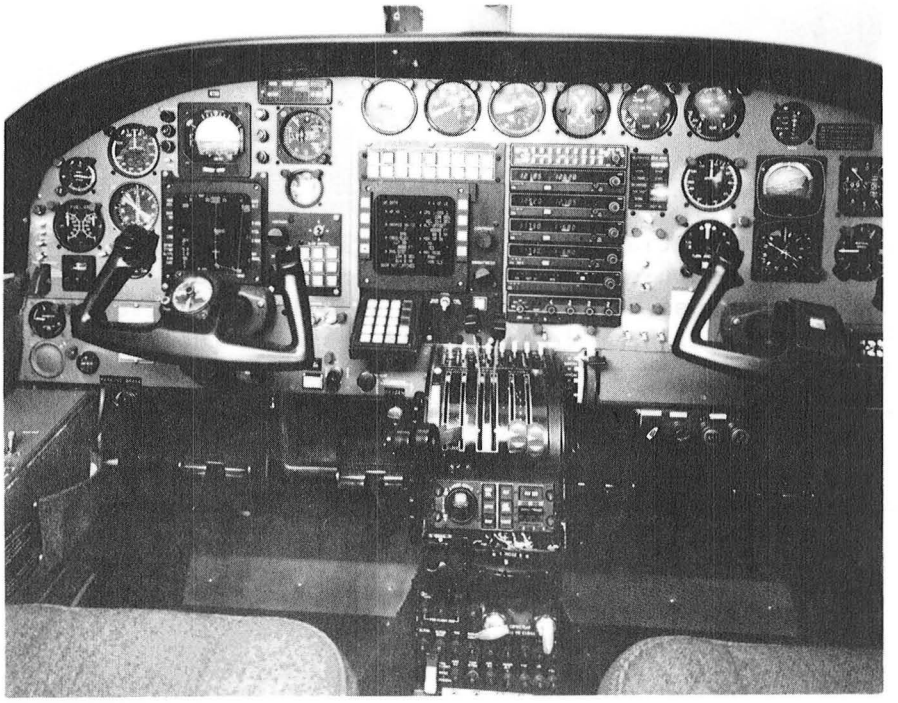


Figure 2.- DAAS cockpit.

As a result of the flight evaluation, several changes have been made to the system software and incorporated into DAAS. The purpose of this paper is to document these changes, and to discuss the characteristics and impact of software modifications to integrated, distributed processing systems. The software modifications are documented fully in appendix A. A brief description of the system architecture and software is provided as an overview.

HARDWARE ORGANIZATION

DAAS is based on a distributed microprocessor network that is linked via an IEEE-488 Standard parallel data bus as shown in figure 3, and is described in detail in references 3 and 4. There are eight microprocessors, all of which communicate via the data bus. The chassis containing the processor cards is shown in figure 4 (except for the Integrated Data Control Center and the radio adapter processors, which are located in their own chassis). One of the processors is an Intel-8048 dedicated to the radio adapter unit built by King Radio on subcontract with Honeywell. The 8048 accesses read-only memory (ROM) exclusively. The other seven microprocessors are Intel-8086s and access both read-only and read/write memory. Several of the processors have circuitry dedicated to the input/output function being performed in addition to that which takes place over the data bus. The data bus initially loads the read/write memory in each processor when the system is turned on. The software for the seven loadable processors is stored in nonvolatile memory (EEPROM), and loaded into the appropriate processor's read/write memory by the bus controller processor (via the data bus). Once the processors are loaded, the data bus is used to transfer information between the eight processors. Prior to releasing the other processors to begin program execution, the bus controller checks to see if a maintenance console/cassette drive is connected to its serial port. If it is, software is run

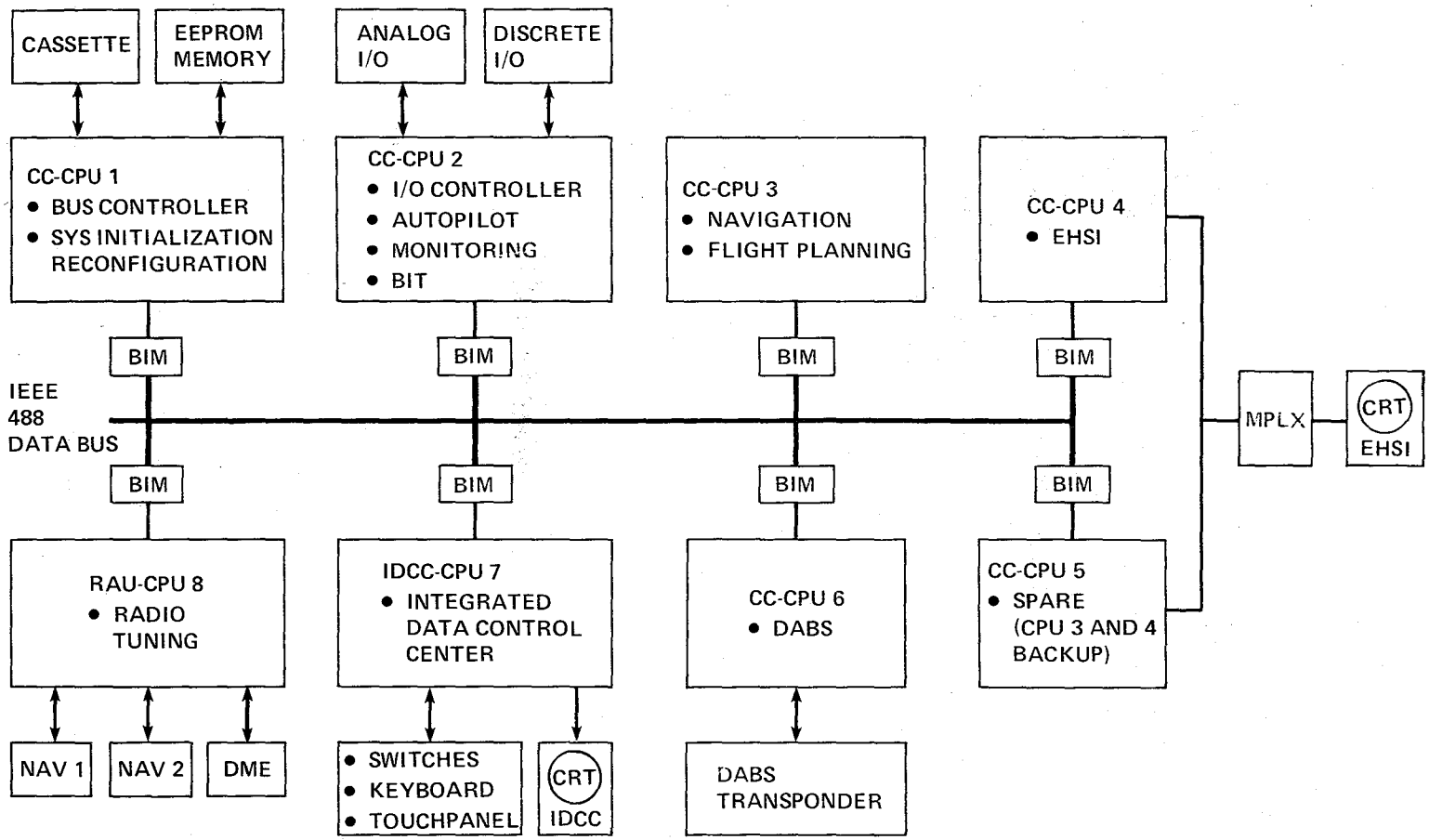


Figure 3.- DAAS processors.

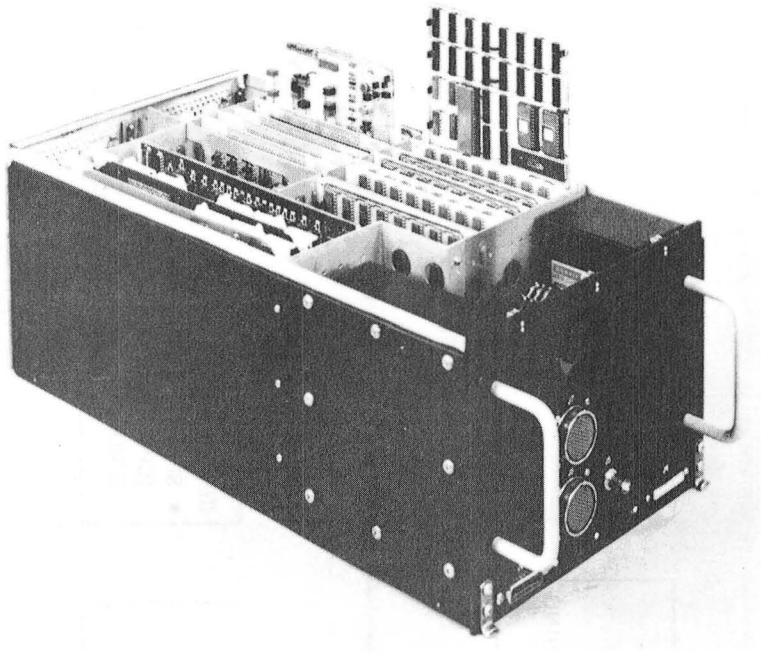


Figure 4.- DAAS processor chassis with top removed.

that allows programs and data from cassette tapes to be loaded into electrically erasable, programmable, read-only memory (EEPROM) or directly into each processor's read/write memory. In addition, several other functions such as software patching and verification can be performed and stored-fault data readout can be run.

SOFTWARE ORGANIZATION

The DAAS software is functionally partitioned with specific and related groups of functions residing within each of the eight processors (fig. 3). The bus controller processor arbitrates all of the interprocessor data transfers in addition to performing the data bus initialization and software transfer to all the processors from EEPROM memory. The downloading procedure is necessary to implement a reconfiguration capability of the DAAS system. If reconfiguration had not been necessary, each processor could have stored its own software in read-only memory which it would execute directly when power was applied to the system. In the approach used with DAAS, a spare processor is available which can take over the functions of either the navigation processor (which does not perform any nondata bus input or output) or the electronic horizontal situation indicator (EHSI) processor, which is multiplexed with the spare processor to the graphics display. A failure detected in either the EHSI or navigation processors causes the bus controller to download the appropriate software into the spare processor's read/write memory and initiate execution.

The navigation processor performs flight planning as well as navigation functions and has the capacity to store ten way points and ten navigation aids. The autopilot processor implements a digital version of the King KFC-200 series analog autopilot, with some modifications that are specific to the DAAS navigation capability. It also performs most of the built-in test functions, configuration and status monitoring, and warning functions. Servo and sensor data (except for the radios) are

interfaced through the autopilot processor. The integrated data control center (IDCC) processor handles pilot requests and provides status annunciation via a CRT display, and is the primary means for the pilot to input and output data and to control DAAS. Map control and function page selection are also handled by the IDCC. The radio adapter unit (RAU) processor performs computer-controlled radio tuning. The discrete address beacon system (DABS) processor interfaces with a Bendix mode-S transponder and includes a simulation of the data link capabilities expected in the future.

Extensive built-in test (BIT) capability is included in the DAAS system. Most of the BIT functions are performed by the autopilot processor into which the servos and the analog, and the discrete input and output circuitry are interfaced. Other BIT tests, such as read/write memory test, processor self test by sample program execution, and bus hardware tests, are distributed throughout the system. The bus controller, autopilot, and IDCC processors include timing circuitry used to implement watchdog timing tests. The bus controller records failures, with the time of their occurrence in EEPROM memory for later recall during maintenance.

SOFTWARE MODIFICATIONS

Eleven changes have been implemented on the DAAS system at Ames. The modifications to the DAAS software have been documented with the same engineering change notice (ECN) form used by Honeywell, Incorporated, to describe their own internal changes. The changes made at Ames are identified as Ames engineering change notice AECN-001 through AECN-011, and are described in detail in appendix A. These descriptions assume some familiarity with the DAAS Functional Description document (ref. 4). The modifications range from changes to the values of program constants and data page formats, to new algorithm implementations and display feature additions, as explained below.

The nature of the software changes made to DAAS can be divided into six categories. These categories may be viewed as representative of those encountered in distributed processing systems in general: alphanumeric constants, algorithms, hardware interfacing, interprocessor communication, pan-processor modifications, and format. All of these categories (except format) have been encountered in performing the changes described in this paper. In general a given modification may require effort in two or more of the categories given above.

Several changes have been made which are representative of the first category, alphanumeric constants. This category would probably generate the most changes in a typical system. The changes do not generally require changes to program logic and are thus easily accomplished.

An example of this type of change is AECN-003, Pretaxi/Taxi Checklist addition. A line has been inserted in a pilot checklist as displayed on the IDCC. Unless a major change in the format of the data being modified is required (described separately) the change can usually be implemented by following the pattern established in the existing software. Other modifications that fall in this category are AECN-004, -005, -006, and -011.

The next category is algorithms, an example of which is AECN-002, percent horsepower computation. Changes in this category involve both the algorithm development

and software coding. The difficulty of the former is in general unbounded and presents the major challenge in implementing a change of this kind. Once the algorithm has been specified, well-established techniques can usually be used to write the necessary software.

When algorithms require the use of floating point arithmetic, it is obviously easiest to code them using processors that can perform floating point arithmetic using hardware. The DAAS system processors do not make use of a math processor with floating point capability (a math processor for the Intel processor was not available when the system was designed). Consequently, all floating point operations were done with fixed-point arithmetic and scaling. The bookkeeping associated with these techniques adds a significant measure of complexity to the coding which would otherwise not be encountered. Debugging the ensuing software can also be more difficult as tracking changes in a given variable requires unraveling the scaling. In the future, it will be increasingly common for processors to be equipped with floating point hardware, thus eliminating the need for scaling in all but the extreme cases.

The third category is hardware interface. These changes are usually very application-specific. Difficulties encountered in this category frequently involve timing considerations. This is particularly evident in distributed processing systems such as DAAS which tend to have several types of timing constraints. The code for hardware operations is usually written in assembly code and involves the rudimentary computer instruction set, posing no significant difficulties other than familiarization with the system.

Two modifications in the hardware category have been made to DAAS. The first involves a system test of the discrete input/output wraparound word test as described in AECN-009. This software runs as a background process. That is, it executes (along with other software) when the processor is not executing higher priority tasks in an interrupt or scheduled mode. Because very little software has been added, the impact on system timing is minimal. The second modification, AECN-010, uses the existing timing structure to generate its own timing used in making warning/caution lights flash. Thus, the existing timing framework was not modified. However, because of the delays introduced by the new software, timing difficulties could still be encountered. If these delays are long enough, system performance could be affected. Unlike a background process which is simply interrupted to maintain the integrity of the timing structure, the function of a time-dependent task such as the one described above would be affected if it were to be interrupted and not allowed to run to completion. If this problem is encountered, the only solutions are to restructure the system software, or to increase the processing speed. The latter is often not feasible in a fully developed system because of the cost involved; thus it is usually the software structure that is modified (though still at great cost in both money and time). Neither solution is particularly desirable; thus, it becomes important to accurately estimate the processor power required in the early design stages, and to allow for error since this estimate is uncertain.

The next category, interprocessor communications, is essentially a subcategory of the previous one. The methodology of tying processors together is a burgeoning field that is widely known as networking. In DAAS the various processors communicate via a shared data bus. One of the DAAS processors is dedicated to controlling the data flow on the bus. This process includes timing and handshake initiation. As might be expected, establishing the framework of the interprocessor communication is a major undertaking in systems of this kind. Both software and hardware considerations are involved; therefore, effort must be expended to ensure that the requirements

of the various parts of the system are met, and that adequate room is left for expansion.

In DAAS a multiple timing structure is implemented to satisfy the various data rates that are required. The data are transferred in variable length buffers. To be able to modularize the interprocessor communication, each processor must maintain the size of the buffer to be transferred in memory. To add data to a buffer, the stored buffer size can be increased in the appropriate processor, and the additional storage can be allocated. As discussed in the previous section, timing constraints must be considered as adding data to a buffer increases the time required to send the buffer. In DAAS the data bus load runs only 50% which allows room for expansion.

The pan processor category refers to software changes that require simultaneous modifications to more than one processor. It is desirable to avoid structures that require changes to be made to more than a single processor as this defeats one of the purposes of functionally subdividing the system by processor. Frequently the only way to avoid these situations is by passing more data between processors. In this case a decision must be made as to whether the derived benefits are worth the overhead of the additional data transfer. An example of this occurs with AECN 001. A change in the equipment complement aboard the test aircraft requires changing the weight and balance constants. Two processors carry this information, the IDCC which displays the current weight and balance figures, and the flight control processor which actually does the weight and balance computations. This is a case where it would seem to be advantageous to add the weight and balance information to a data buffer and then send the current information from the flight control processor to the IDCC. This is particularly true since weight and balance information is changed relatively often.

The last category to be discussed is format, which refers to the large scale software structuring that is required in integrated, multiprocessor systems such as DAAS. The structured layout must be accomplished early in the system design since the software will be written within the constraints and layout that are imposed by the format. This is one of the most important phases of the total system design. Since the nature of the system is largely decided at this stage, philosophical issues must be settled by this point. However, room for expansion and some modifications must be incorporated into the format as the need for changes will be inevitable. Strong demands on flexibility are made by the pilot system interface. In DAAS this is largely accomplished through the EHSI and the IDCC. Several changes were made to the IDCC pages and EHSI display as discussed above without the need for format modifications which is a strong indication of the flexibility that is inherent in the DAAS design. Hardware and format constraints are also imposed on interprocessor communication and flexibility and expansion capability for future modifications must be taken into account. A multiprocessor system can increase its functional capability by adding processors to the system. A limit will then be reached when the interprocessor communication data bus becomes saturated.

CONCLUDING REMARKS

Several software modifications have been made to the DAAS at Ames Research Center. These modifications fall into several categories whose characteristics and impact have been discussed. The flexibility built into DAAS has made changes relatively easy to accomplish. Format changes rank as the most difficult to make,

followed by changes that involve the hardware timing structure of the system (including interprocessor communication). To avoid modification to the existing structure, both of these areas should incorporate flexibility and expansion capability at the initial design stage so that changes can be made within the existing structure rather than by modifying it. Whenever possible it is best to avoid the additional difficulties that are caused when simultaneous changes are required in more than one processor.

APPENDIX A

AMES ENGINEERING CHANGE NOTICES

AECN 001 — Aircraft weight correction
AECN 002 — Percent horsepower computation
AECN 003 — Pretaxi/taxi checklist addition
AECN 004 — Waypoint data page modification
AECN 005 — Flight status page change
AECN 006 — Nav aid data format
AECN 007 — EHSI lubberline addition
AECN 008 — EHSI heading select arrow
AECN 009 — I/O sysbit discrete wraparound
AECN 010 — Flash warning lights
AECN 011 — Weight and balance summary page format

The first five AECNs that are described involve changes to the data entry pages of the IDCC. Each page is described in a separate software module that uses a standard format (fig. 5). Pushing a page select button or toggling the forward/back switch invokes software routines that locate the appropriate page-format software-definition module and that use the information contained there to display the appropriate data on the IDCC. The standard format that is used for the layout of all the pages, minimizes the amount of information that must be contained in the page-format software-definition module. Thus to change or add to the information on a given IDCC page, one changes only the corresponding page table software.

AECN 003 — A line item has been added to the pretaxi/taxi checklist to remind the pilot to engage the autopilot prior to the system test (initiated from the KCI-310). Figure 6(a) shows the pretaxi/taxi checklist prior to the change. Figure 6(b) shows the addition. The change was proposed because the pilot and the experimenter frequently did not engage the autopilot prior to system test initiation which caused the system to halt its operation (caused by system-perceived autopilot failure) and required the pilot to restart the system and to reenter lost initialization data. One way to eliminate this type of failure was to add a line to the checklist.

This change has been accomplished by modifying a constant that indicates the number of lines in the specific checklist, and by simply inserting the required line of text at the appropriate point in the subroutine. The 8086 assembly code used to define the pretaxi/taxi checklist is listed in appendix B. The checklist routine software is included as an example of the assembly language that is used in programming much of DAAS. Some parts of DAAS have been programmed in Intel's higher order language, PL/M, most notably the navigation processor software. For reference, an example of PL/M is also listed in appendix C, which shows the software used in the weight and balance computation described below.

AECN 011 — The maximum gross weight and fuel weight have been added to the weight and balance data output page (fig. 7). These numbers are unavailable on the original page, which makes it difficult to verify aircraft loading limits prior to transferring the weight and balance figures to the system when making flight performance calculations. Previously, the aircraft's maximum gross weight and fuel figures could only be obtained from the initialization page shown in figure 8. With the

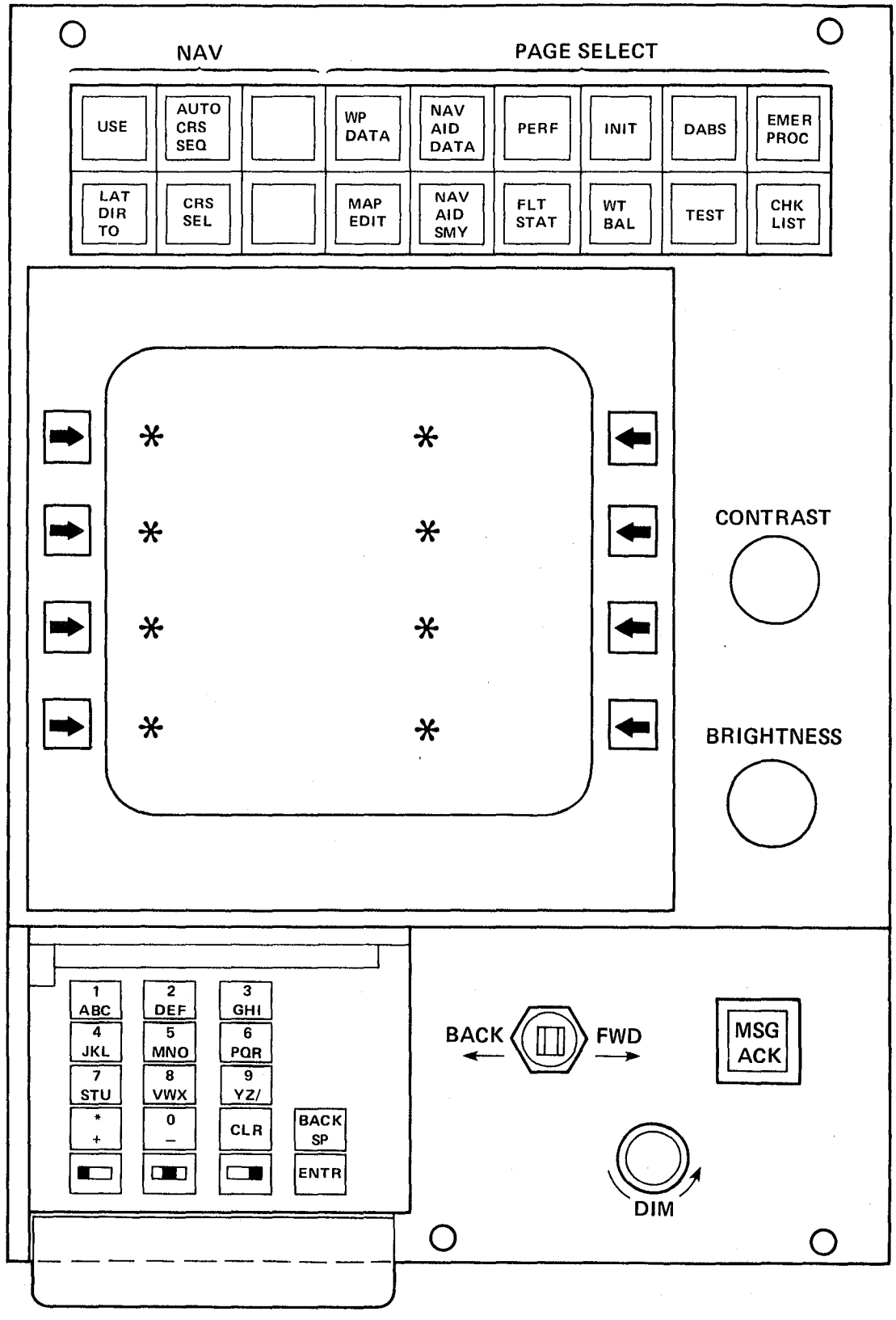
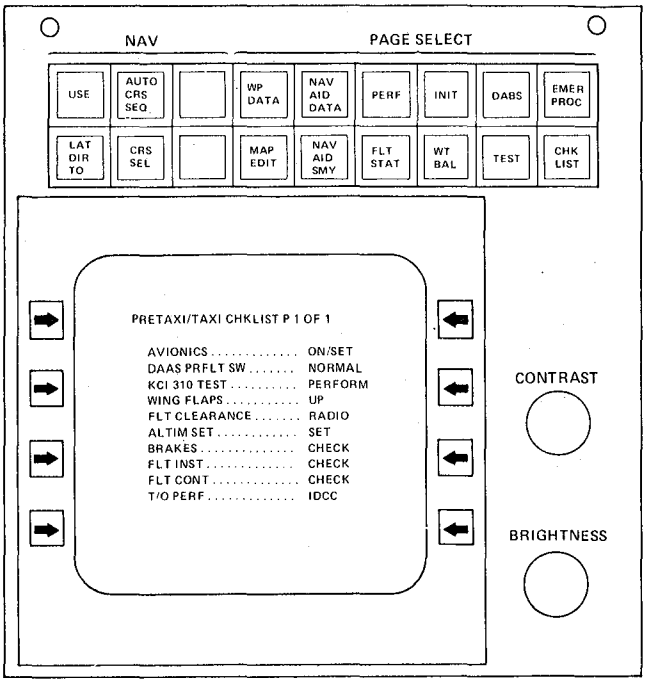
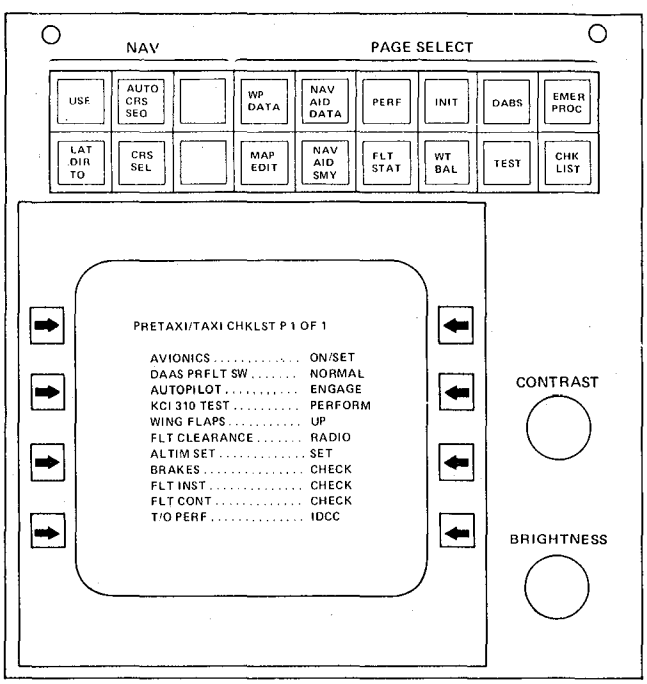


Figure 5.- Page format from functional description.

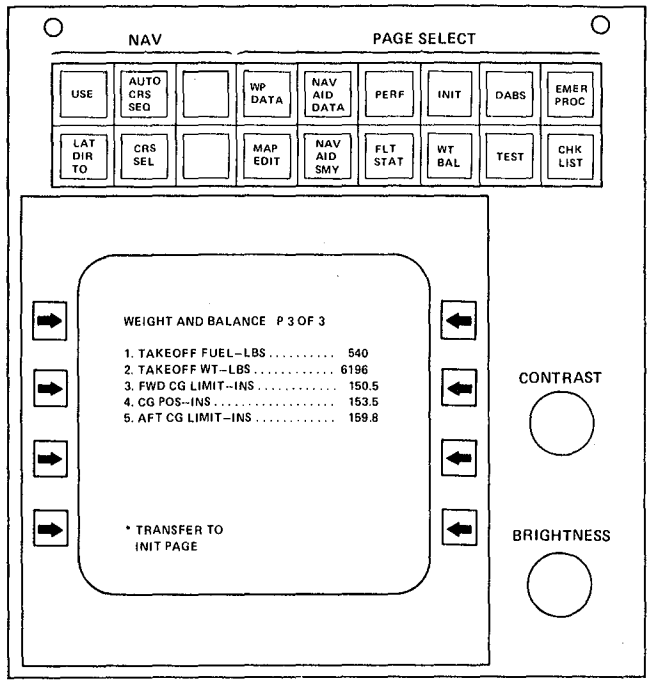


(a) BEFORE

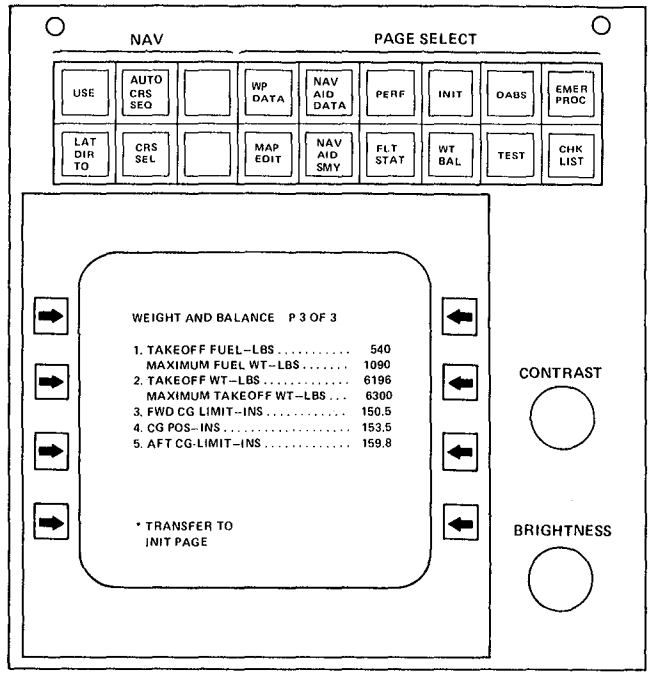


(b) AFTER

Figure 6.- PRETAXI Checklist page.



(a) BEFORE



(b) AFTER

Figure 7.- Weight and balance summary page.

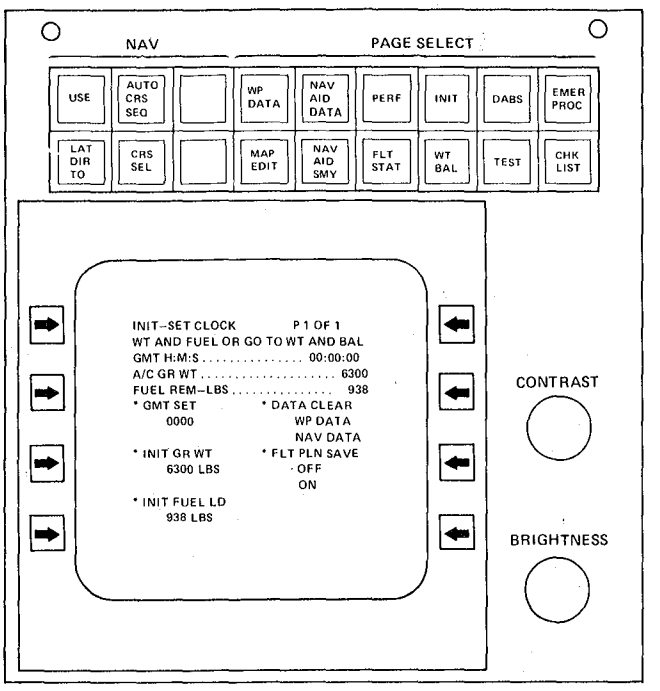


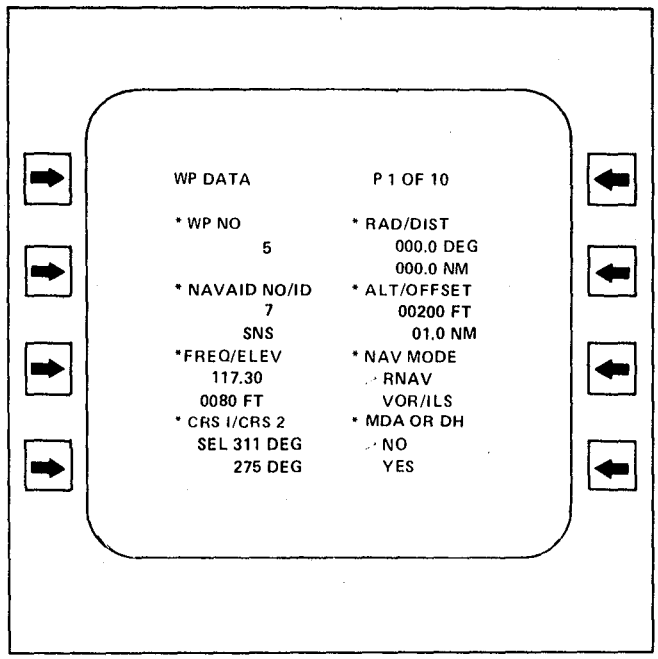
Figure 8.- Initialization page.

change, the maximum gross weight appears next to the calculated takeoff weight (fig. 7(b)) and the maximum allowable fuel weight appears next to the calculated takeoff fuel.

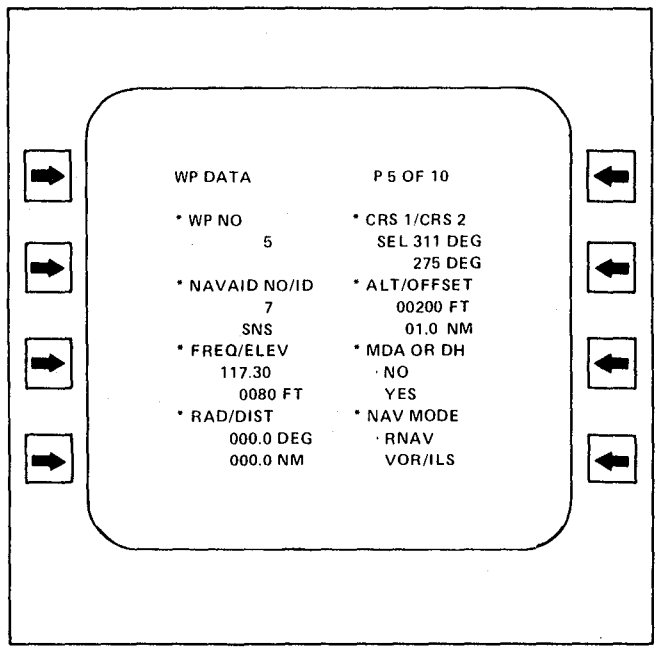
AECN 004 — Some changes have been made to the layout of the way point data entry page. The data entry associated with four of the touch points have been interchanged to comply with various pilot suggestions for system improvement. Their functions are left unchanged. Figure 9 shows the way point data page before and after the changes. The radial/distance data entry has been moved to the lower-left touch point so that all the data defining the way point's horizontal data would fall into one column, the assumption being that such logical grouping makes the system easier to learn and use. The navigational mode selection has been moved to the touch point at the bottom of the page so that it would not be hit accidentally by a pilot who intended to input altitude data.

AECN 006 — A rearrangement of the entire navigational-aid data entry page is shown in figures 10(a) and 10(b). The data entry points are now relocated into one column and grouped to conform to the presentation found on route navigation (RNAV) charts with the pairing on the navigational-aid data page which corresponds to the pairing on the RNAV charts. The purpose was to ease the entry of navigational-aid data which tends to be a time consuming chore and therefore is particularly critical in flight.

AECN 005 — This is a change to the flight status page. Originally the wind magnitude was displayed above the wind direction (fig. 11(a)). This has been reversed so that it conforms with the more common form of giving the wind direction first, followed by the wind magnitude (fig. 11(b)).

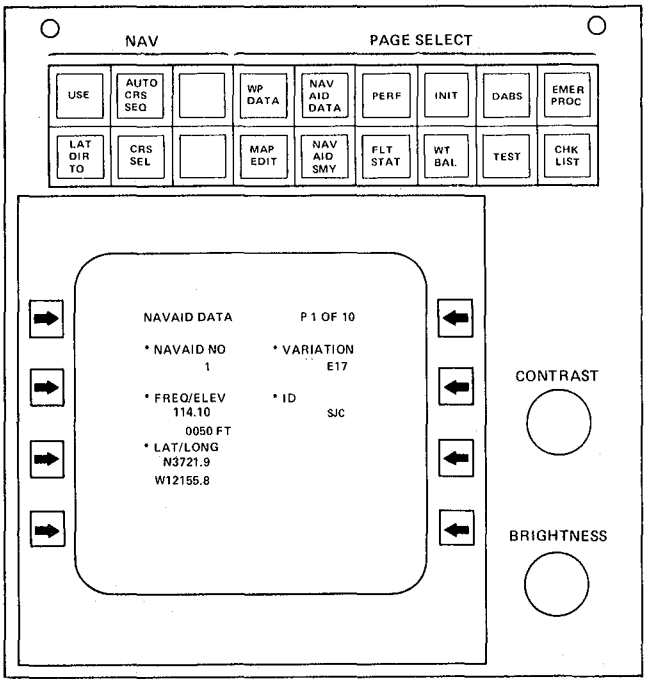


(a) BEFORE

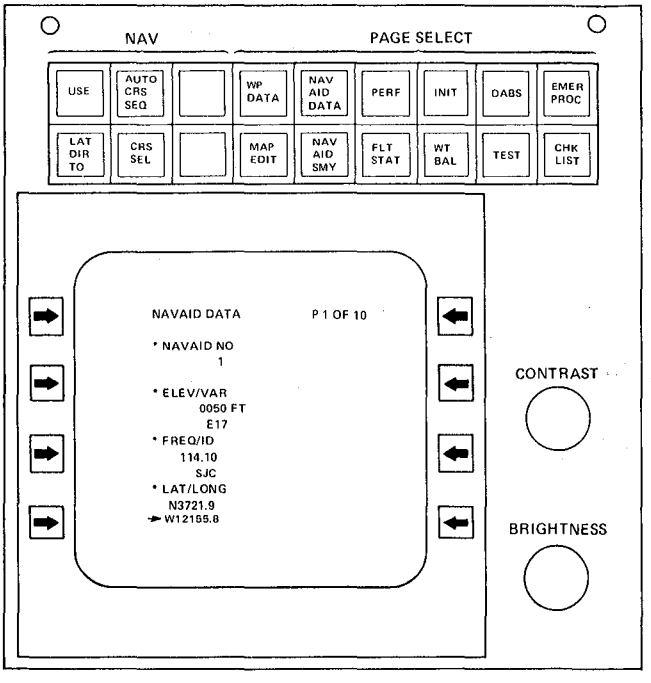


(b) AFTER

Figure 9.- Way point data page.

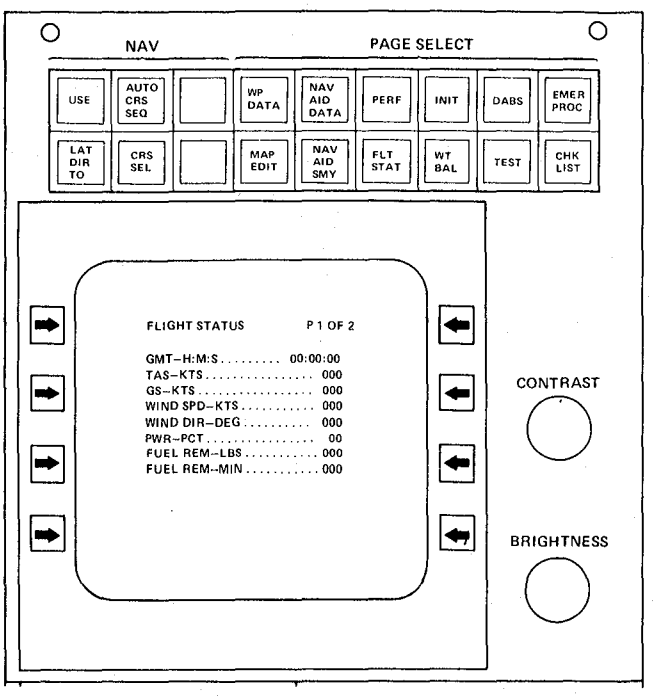


(a) BEFORE

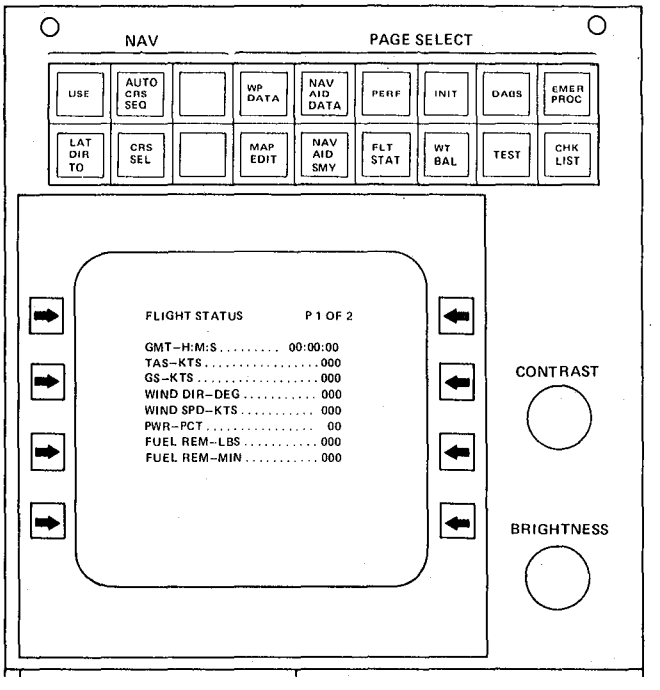


(b) AFTER

Figure 10.- Navigational-aid data page.



(a) BEFORE



(b) AFTER

Figure 11.- Flight status, page 1.

The five changes just described involve the software in only one processor, the IDCC, and do not involve changes to the interprocessor data exchange via the IEEE-488 bus. The following AECN entails a simultaneous change to the software in two processors.

AECN 001 — Because of changes to the equipment complement on the aircraft, empty weight and empty center of gravity have been updated. The displayed empty weight and center of gravity (fig. 12) are changed by modifying the appropriate IDCC page table. However, this action does not change the empty weight and center of gravity figures used in the weight and balance computations. These constants are kept in the flight control processor which actually performs the weight and balance computations. Both changes involve simple changes to the empty weight and center of gravity constants in each respective processor. A potential improvement, which has not been completed because of time constraints, would be to modify the software so that changes to empty weight and center of gravity could be accomplished by changing the defining constants in only one processor.

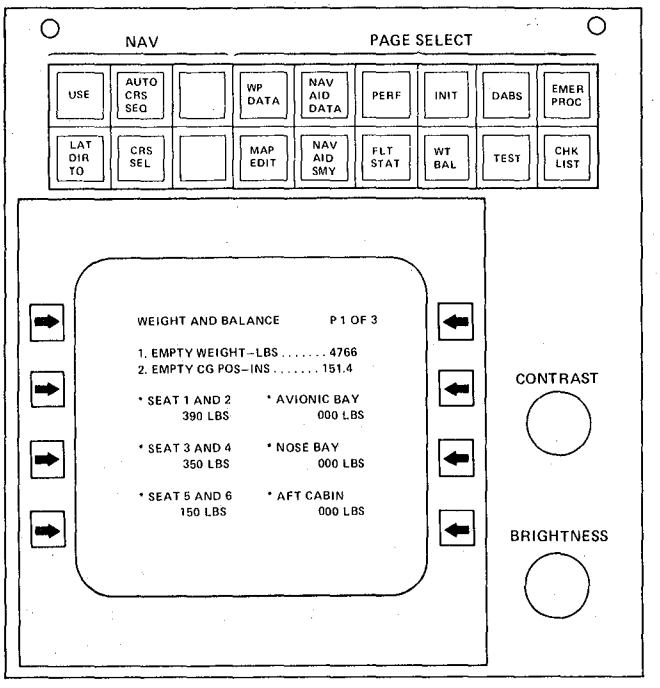


Figure 12.- Weight and balance, page 1.

The following two AECNs require modifications to the flight control processor modules that access the analog and discrete inputs and outputs.

AECN 009 — A correction has been made to the testing of the discrete output wraparound word. The previous version of the system test sets all 16 bits of the discrete output word (table 1) to all ones and then to all zeroes, and checks the discrete input word (wraparound with inversion) for all zeroes and all ones, respectively. Setting both output bits 7 and 8 to ones causes simultaneous auto-trim-up and trim-down commands to the pitch-trim servo's transistors and, with time, causes them to fail. The modified software now separately tests auto-trim up and auto-trim down. No failures of the trim transistors have occurred since the changes have been made.

TABLE 1.- DISCRETE INPUT/OUTPUT WORD DEFINITIONS

BIT	INPUT WORD 1 08800	INPUT WORD 2 08802	INPUT WORD 3 08804	INPUT WORD 4 08806	COMPLIMENT	OUTPUT WORD 1 08200	OUTPUT WORD 2 08202	OUTPUT WORD 3 08204
15	HDGSSLW (1 = ON)	APENG (1 = ENG)	AUXPMP 1 (1 = ON)	APCLENG	←	APCLENG	NAV ARM A	MSTR WRN A
14	HDG SEL (1 = ON)	YDENG (1 = ENG)	AUXPMP 2 (1 = ON)	AP SOL	←	AP SOL	NAVCPLD A	MSTR CTN A
13	APPR (1 = ON)	APYDDC (1 = DISC)	DROPEN (1 = OPEN)	YD SOL	←	YD SOL	REV LOC A	TRM FAIL A
12	NAV (1 = ON)	FLT DIR (1 = ON)	WOW (0 = WOW)	ALT ENG	←	ALT ENG	YAW DMP A	ALTALRT
11	VNAV (1 = ON)	MNTRM (1 = ON)	GR DN LK (0 = DOWN)	VNAVSHTR	←	VNAVSHTR	HDGSEL A	DH ANN
10	ALT (1 = ON)	FLTT 1 (1 = ON)	VG VAL (1 = VAL)	CMDBRRET	←	CMD BR RET	VNAVARM A	MDA ANN
9	ALT ARM (1 = ON)	TRMPWR (1 = ON)	DG VAL (1 = VAL)	REV LOC	←	REV LOC	VNAVCPLD A	DABS ANN
8	SYSINTLK (0 = INTERLOCK)	TRM UP M (1 = ON)	BALT VAL (0 = VAL)	ATRM UP	←	ATRM UP	GS CPLD A	PTRMTST
7	G DUMP (0 = DUMP)	TRM DN M (1 = ON)	FLT T 2 (1 = ON)	ATRM DN	←	ATRM DN	ALTARM A	R ALT TST
6	CWS (0 = ON)	PATL UP (0 = UP lim)	IAS VAL (0 = VAL)	CMPTRVAL	←	CMPTRVAL	ALTHLD A	ADC TEST
5	GOARND (0 = ON)	PATL DN (0 = DN lim)	ADCALTV (0 = VAL)	X		X	GOARD A	WRN HRN
4	MMSNS (1 = SENSED)	PFLTT (1 = TEST)	ALT VAL (1 = VAL)	X		X	FLTDIR A	TEST UP
3						X	APRARM A	TEST DN
2						X	APRCPLD A	X
1						X	AP ANN	X
0						X		X

NOTE: X = SPARE

AECN 010 — Pilots frequently have difficulty detecting warning or caution conditions as annunciated by the red and amber warning lights, although these lights are placed alongside the ADI directly in the pilot's view (fig. 13). An attempt has been made to make the lights more visible by causing them to blink. The red warning light now flashes at a 2-Hz rate whereas the amber light flashes at 1.25 Hz. Although these values are empirical and no formal study has been conducted, it appears that the blinking lights bring about a significant improvement in the timely detection of a warning or caution condition.

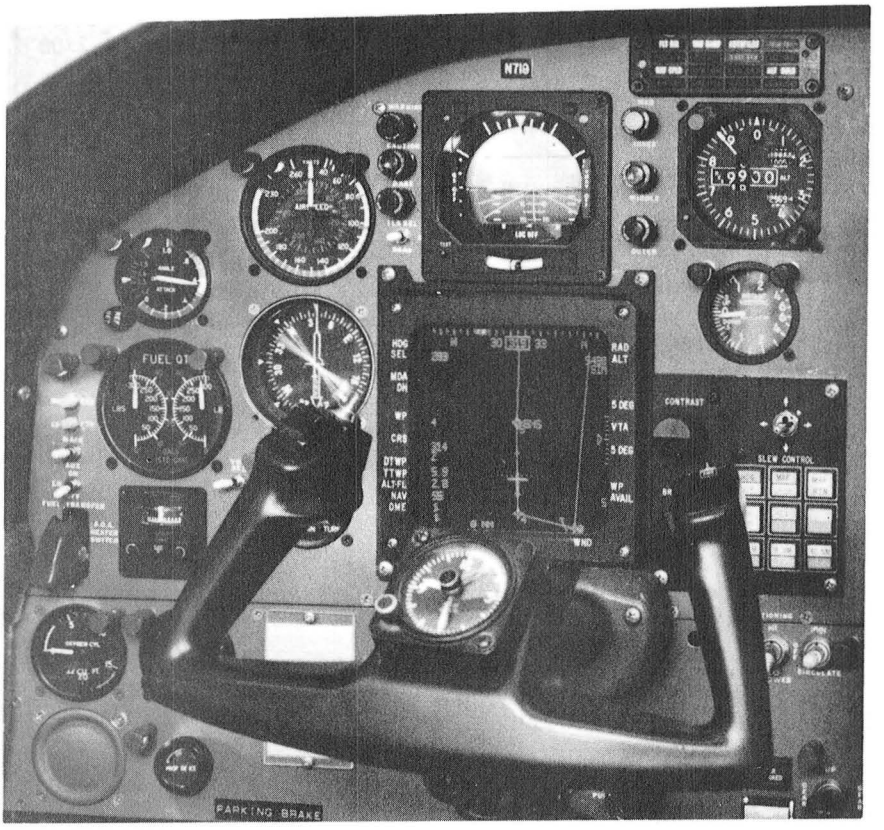


Figure 13.- ADI with warning lights.

The following AECN involves a change to an internal algorithm with no change to any input or output parameters.

AECN 002 — The DAAS system computes and displays the percentage of maximum available horsepower and displays this to the pilot on the flight status page (fig. 11(b)). Previously this calculation was made using fuel flow. This technique gives reasonably accurate readings of engine power at full rich mixtures, but if the mixture is leaned the fuel-flow derived-horsepower figure will be in error. In addition there is no compensation for nonstandard temperature and altitude/pressure effects. An alternate algorithm (table 2) has been proposed which computes percent horsepower directly from engine rpm and manifold pressure using engine data from the Continental operations manual for the TSO-520E engine used in the Cessna 402B. Compensation for nonstandard temperature and altitude/pressure effects is included.

The algorithm has been tested by flying at various power settings varying from 25% to 90% of total power and by comparing the DAAS-computed percent horsepower to

TABLE 2.- PERCENT HORSEPOWER ALGORITHMS

% HP FOR IDCC FLIGHT STATUS

$$1\,HP_{SL} = \left[\left(\frac{RPM-1900}{2500-1900}\right)80 + 155\right]\frac{MAP-6.7}{30-6.7} \quad ; \quad \begin{matrix} 20.0'' < MAP < 30.0 \\ 1900 < RPM < 2500 \end{matrix}$$

$$2\,HP_{SL} = \left[\left(\frac{RPM-1900}{2500-1900}\right)80 + 155\right]\frac{MAP-4.2}{30-4.2} \quad ; \quad \begin{matrix} 30.0 < MAP < 34.5 \\ 1900 < RPM < 2500 \end{matrix}$$

$$3\,HP_{SL} = \left[\left(\frac{RPM-2500}{2700-2500}\right)20 + 235\right]\frac{MAP-6.7}{30-6.7} \quad ; \quad \begin{matrix} 20.0 < MAP < 30.0 \\ 2500 < RPM < 2700 \end{matrix}$$

$$4\,HP_{SL} = \left[\left(\frac{RPM-2500}{2700-2500}\right)20 + 235\right]\frac{MAP-4.2}{30-4.2} \quad ; \quad \begin{matrix} 30.0 < MAP < 34.5 \\ 2500 < RPM < 2700 \end{matrix}$$

$$\Delta HP_h = \frac{h}{20000}(8) \quad ; \quad HP_{SL} < 180$$

$$= \frac{220-HP_{SL}}{220-180}\,\frac{h}{20000}(8) \quad ; \quad HP_{SL} > 180$$

$$HP_h = HP_{SL} + \Delta HP_h$$

$$HP = \left[1 - \frac{T-T_{STD}}{10/3}(0.01)\right] HP_h \quad ; \quad T\ C$$

$$\%\ HP = \frac{HP}{300} * 100 = \frac{HP}{3}$$

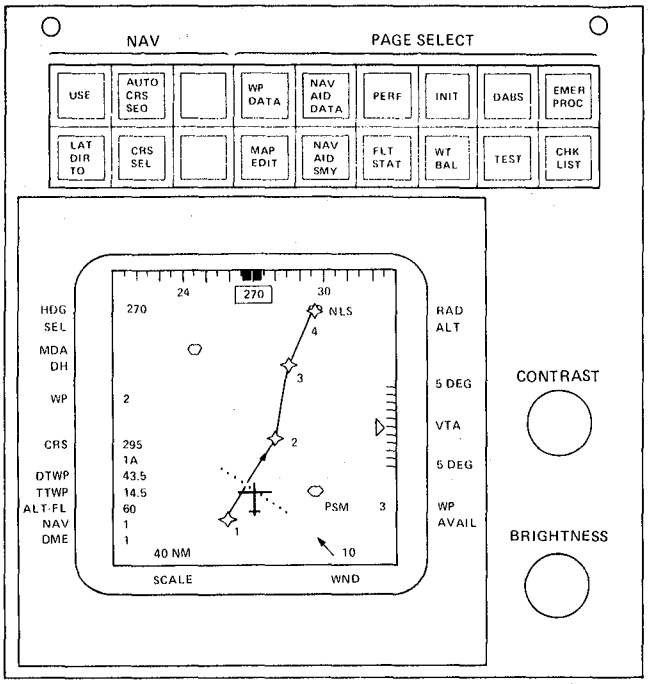
the value given by the operator's manual. The correlation between the two is excellent at all power settings.

The final two completed AECNs are changes to the electronic horizontal situation indicator (EHSI) processor's software.

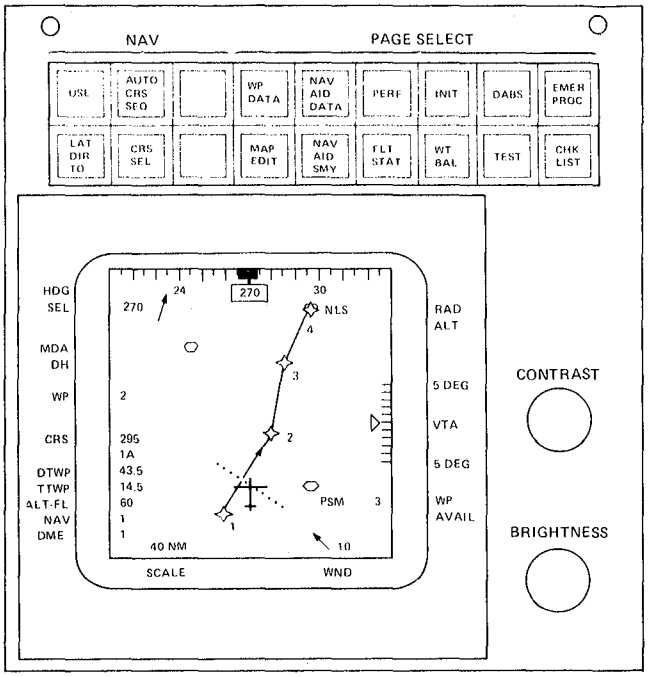
AECN 007 — This change involves a simple change to the module that draws the heading box. It was decided that a lubber line would be useful in determining the heading. The heading box has been lowered and a small lubber line has been added to the top of the box (fig. 14).

AECN 008 — This is a more substantial change which involves adding an analog heading select arrow to the EHSI positioned alongside the digital heading select (HDG SEL) readout at the upper left of the EHSI (fig. 14). With the exception of a few minor changes, the software used to draw the heading select arrow is similar to that used to draw the wind vector arrow (located at the lower right of the EHSI).

The addition of a heading select arrow considerably eases the task of selecting a heading with respect to a course that is visible on the map. In general one is concerned with the angle between a desired heading and the selected course. This angle is readily visible by using the analog heading select arrow. Many favorable comments have been received on this feature. One of the drawbacks of the DAAS-EHSI presentation is the lack of a full compass rose with which to reference courses or compass directions that are based on a current reading. The addition of the analog heading select arrow is seen as a partial solution to the lack of a full compass rose.



(a) BEFORE



(b) AFTER

Figure 14.- EHSI.

Two proposed software changes have been tested but are not permanently incorporated into the DAAS system and should be mentioned briefly. These two changes involve adding data to the packets that are exchanged between processors over the IEEE-488 data bus. Each change requires modifications to the software in all the processors that receive information from the changed packet.

The first change entails the addition of the bearing and distance to the VOR that defines the current way point to the EHSI data packet. This information is sent from the navigation processor to the EHSI processor to allow the display of a navigation-aid bearing arrow that is analogous to the way point bearing arrow. Bearing and distance to a nearby VOR are frequently requested by air traffic control (ATC) and there is currently no way of determining them from the DAAS electronic displays.

A second change involves reserving space in the memory word packet to allow each processor to receive two 16-bit words that are entered from the IDCC. The first word is a location in the memory of a processor in which to place the contents of the second word. This change would allow dynamic altering of programs or data in flight. The change is intended to aid in modifying or debugging the system, but it is not a user mode.

APPENDIX B

8086 ASSEMBLY CODE FOR PRETAXI/TAXI CHECKLIST MODULE

```
ISIS-II MCS-86 MACRO ASSEMBLER V2.0 ASSEMBLY OF MODULE SRCN23
OBJECT MODULE PLACED IN :F1:SRCN23.OBJ
ASSEMBLER INVOKED BY: ASM86 :F1:SRCN23.SRC DEBUG TITLE(SRCN23) DATE(1/18/82) XREF

LINE     SOURCE

   1                        ;        SCREEN 23       PRE TAXI/TAXI CHK LIST
   2                        ;
   3                        ;REVISION HISTORY:
   4                        ;        REV 0    09/05/80
   5                        ;                 SOFWARE RELEASED
   6                        ;
   7                        ;        REV 1    07/28/81
   8                        ;                 ADD PERFORM KCI 310 TEST TO CHECK LIST.
   9                        ;
  10                        ;        REV 2    01/18/82
  11                        ;                 ADD AUTOPILOT ENGAGE TO CHECK LIST.
  12                        ;                 AECN 003
  13                        ;
  14                        ;
  15                        ;
  16     DGROUP  GROUP DATA
  17                        ;
  18                        ;
  19     DATA SEGMENT PUBLIC 'DATA'
  20              PUBLIC SCRN23
  21     SCRN23  LABEL BYTE
  22                        ;
  23     SCREENTABLE      DB       0                          ;BUFFER ACTIVE FLAG
  24     SCREENNUMBER     DB       23                         ;SCREEN NUMBER 23
  25                      DW       VARDATA-SCREENTABLE        ;POINTER TO VARIABLE DATA
  26                      DW       TPDATA-SCREENTABLE         ;POINTER TO TOUCHPOINT RESPONSE DATA
  27                      DB       24                         ;NEXT PAGE- PRE TAKEOFF P1
  28                      DB       22                         ;BACK PAGE- START ENG
  29                      DB       OFFH                       ;CHECKLIST FLAG (FF= CHECKLIST)
  30                      DB       11                         ;NUMBER OF ITEMS
  31
  32                        ;
  33                        ;
  34                        ;        FIXED TEXT
  35                        ;
  36     FIXEDDATA        DB       ' PRETAXI/TAXI CHKLST'

  37                      DB       80H+0,22

  38                      DB       'P 1 OF  1'

  39                      DB       80H+2,2

  40                      DB       'AVIONICS'

  41                      DB       0A0H,13

  42                      DB       'ON/SET'
```

```
LINE    SOURCE
  43                    DB      80H+3, 2
  44                    DB      'DAAS PRFLT SW'
  45                    DB      0A0H, 8
  46                    DB      'NORMAL'
  47                    DB      80H+4, 2
  48                    DB      'AUTOPILOT'             ; AECN 003
  49                    DB      0A0H, 12                ;
  50                    DB      'ENGAGE'                ;
  51                    DB      80H+5, 2
  52                    DB      'KCI 310 TEST'
  53                    DB      0A0H, 9
  54                    DB      'PERFORM'
  55                    DB      80H+6, 2
  56                    DB      'WING FLAPS'
  57                    DB      0A0H, 11
  58                    DB      'UP'
  59                    DB      80H+7, 2
  60                    DB      'FLT CLEARANCE'
  61                    DB      0A0H, 8
  62                    DB      'RADIO'
  63                    DB      80H+8, 2
  64                    DB      'ALTIM SET'
  65                    DB      0A0H, 12
  66                    DB      'SET'
  67                    DB      80H+9, 2
  68                    DB      'BRAKES'
  69                    DB      0A0H, 15
  70                    DB      'CHECK'
  71                    DB      80H+10, 2
  72                    DB      'FLT INST'
  73                    DB      0A0H, 13
```

```
LINE    SOURCE
  74                    DB      'CHECK'
  75                    DB      80H+11,2
  76                    DB      'FLT CONT'
  77                    DB      0AOH,13
  78                    DB      'CHECK'
  79                    DB      80H+12,2
  80                    DB      'T/O PERF'
  81                    DB      0AOH,13
  82                    DB      'IDCC'
  83                    DB      OFFH            ;END OF FIXEDDATA FLAG
  84                    ;
  85                    ;
  86                    ;       VARIABLE DATA
  87                    ;
  88    VARDATA         DB      0               ;NO VARIABLE DATA
  89                    ;
  90                    ;
  91    TPDATA          DB      0,0,0,0         ;TP 1 ACTION - NOT USED
  92                    ;
  93                    DB      0,0,0,0         ;TP 2 ACTION - NOT USED
  94                    ;
  95                    DB      0,0,0,0         ;TP 3 ACTION - NOT USED
  96                    ;
  97                    DB      0,0,0,0         ;TP 4 ACTION - NOT USED
  98                    ;
  99                    DB      0,0,0,0         ;TP 5 ACTION - NOT USED
 100                    ;
 101                    DB      0,0,0,0         ;TP 6 ACTION - NOT USED
 102                    ;
 103                    DB      0,0,0,0         ;TP 7 ACTION - NOT USED
```

```
LINE     SOURCE

 104                    ;
 105                    DB      0,0,0,0         ;TP 8 ACTION - NOT USED

 106                    ;
 107     DATA ENDS
 108                    ;
 109                    END
```

APPENDIX C

PLM CODE FOR WEIGHT AND BALANCE

```
ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE WGTBAL
NO OBJECT MODULE REQUESTED
COMPILER INVOKED BY:  PLM86 :F1:WGTBAL.SRC OPTIMIZE(3) NOOBJECT XREF SYMBOLS PRINT(:LP:) IXREF(:F2:WGTBAL.IXI) DATE(11/2
                      -0/81)


                /*
                2.4 WEIGHT AND BALANCE

                REVISION:   02/17/81        KEN BROEN
                        .11/06/81   AIRCRAFT WEIGHT CORRECTION (ECN 076)

                PURPOSE:
                    TO COMPUTE THE TAKEOFF FUEL WEIGHT, TAKEOFF WEIGHT, FORWARD CARGO LIMIT,
                    AFT CARGO LIMIT AND THE CENTER OF GRAVITY FOR THE AIRPLANE, USING PILOT ENTERED DATA.

                CONNECTIONS:

                    CALLED BY:  BACKGROUND PROCESSING

                    CALLS TO:   MULDIV

                ADDITIONAL INFORMATION:
                    NONE
                */
```

```
            $EJECT
  1         WGTBAL:DO;
  2  1        DECLARE DCL LITERALLY 'DECLARE';
  3  1        DCL ENDDO LITERALLY 'END';
  4  1        DCL ENDIF LITERALLY ' ';
  5  1        DCL SEAT1$2 INTEGER EXTERNAL;
  6  1        DCL SEAT3$4 INTEGER EXTERNAL;
  7  1        DCL SEAT5$6 INTEGER EXTERNAL;
  8  1        DCL AV$BAY INTEGER EXTERNAL;
  9  1        DCL NOSE$BAY INTEGER EXTERNAL;
 10  1        DCL AFT$CABIN INTEGER EXTERNAL;
 11  1        DCL WING$LOCK INTEGER EXTERNAL;
 12  1        DCL OTHER$WGT INTEGER EXTERNAL;
 13  1        DCL DIST$SEATS INTEGER EXTERNAL;
 14  1        DCL MAIN$FUEL INTEGER EXTERNAL;
 15  1        DCL AUX$FUEL INTEGER EXTERNAL;
 16  1        DCL TO$FUEL INTEGER EXTERNAL;
 17  1        DCL TO$AC$WGT INTEGER EXTERNAL;
 18  1        DCL CG$POS INTEGER EXTERNAL;
 19  1        DCL FWD$CG INTEGER EXTERNAL;
 20  1        DCL AFT$CG INTEGER EXTERNAL;
 21  1        DCL EMPTY$WGT INTEGER DATA (4839);
 22  1        DCL EMPTY$CG INTEGER DATA (4832)          /* CG IN INCHES TIMES 32 */  /*ECN 076*/;

 23  1        MULDIV:PROCEDURE (A,B,C) INTEGER EXTERNAL;
 24  2            DCL (A,B,C) INTEGER;
 25  2        END MULDIV;

 26  1        WEIGHT$AND$BALANCE:PROCEDURE PUBLIC;

 27  2            /* TAKEOFF FUEL WEIGHT */;

 28  2            TO$FUEL = MAIN$FUEL + AUX$FUEL;

                  /* TAKEOFF WEIGHT */

 29  2            TO$AC$WGT =       EMPTY$WGT
                                  + TO$FUEL
                                  + SEAT1$2
                                  + SEAT3$4
                                  + SEAT5$6
                                  + AV$BAY
                                  + NOSE$BAY
                                  + WING$LOCK
                                  + AFT$CABIN
                                  + OTHER$WGT;

                  /* FORWARD CARGO LIMIT */

 30  2            IF TO$AC$WGT <= 5000
                  THEN
 31  2                DO;
 32  3                    FWD$CG = 4720          /* 147.5 SCALED 10 */;
 33  3                ENDDO;
                  ELSE
 34  2                DO;
 35  3                    /* FORWARD LIMIT = 147.49 + .00252 (TAKEOFF WEIGHT - 5000) */;
```

```
36   3                  FWD$CG = 4720 + 2 * (TO$AC$WGT - 5000) / 25 ;
37   3              ENDDO;
38   2          ENDIF /* TO$AC$WGT <= 5000 */;

                /* AFT CARGO LIMIT */

39   2          IF TO$AC$WGT <= 5900
                THEN
40   2              DO;
41   3                  AFT$CG = 5126           /* 160.187 SCALED 10 */;
42   3              ENDDO;
                ELSE
43   2              DO;
44   3                  /* AFT LIMIT = 160.20 - .00125 (TAKOFF WEIGHT - 5900) */;
45   3                  AFT$CG = 5126 - (( TO$AC$WGT - 5900) / 25 );
46   3              ENDDO;
47   2          ENDIF /* TO$AC$WGT <= 5900 */;

                /* CENTER OF GRAVITY */

                /* CENTER OF GRAVITY = [  EMPTY CENTER OF GRAVITY (EMPTY WEIGHT)
                                        + 152 (FUEL MAIN)
                                        + 164 (FUEL AUX)
                                        + 186 (WING LOCK)
                                        + 137 (SEATS 1 & 2)
                                        + 175 (SEATS 3 & 4)
                                        + 218 (SEATS 5 & 6)
                                        + 32  (AVIONICS BAY)
                                        + 71  (NOSE BAY)
                                        + 266 (AFT CABIN)
                                        + (DIST BEHIND SEAT 1 + 137) (OTHER WEIGHT) ]
                                        ------------------------
                                        DIVIDED BY TAKEOFF WEIGHT
                */

48   2          CG$POS =((MULDIV ( EMPTY$CG, EMPTY$WGT, TO$AC$WGT ))
                          +(MULDIV ( 4864, MAIN$FUEL, TO$AC$WGT ))
                          +(MULDIV ( 5248, AUX$FUEL, TO$AC$WGT ))
                          +(MULDIV ( 5952, WING$LOCK, TO$AC$WGT ))
                          +(MULDIV ( 4384, SEAT1$2, TO$AC$WGT ))
                          +(MULDIV ( 5600, SEAT3$4, TO$AC$WGT ))
                          +(MULDIV ( 6976, SEAT5$6, TO$AC$WGT ))
                          +(MULDIV ( 1024, AV$BAY, TO$AC$WGT ))
                          +(MULDIV ( 2272, NOSE$BAY, TO$AC$WGT )),
                          +(MULDIV ( 8512, AFT$CABIN, TO$AC$WGT ))
                          +(MULDIV ( DIST$SEAT$1 + 4384, OTHER$WGT, TO$AC$WGT )));

49   2          END WEIGHT$AND$BALANCE;
50   1      END WGTBAL;
```

CROSS-REFERENCE LISTING

```
  DEFN  ADDR   SIZE  NAME, ATTRIBUTES, AND REFERENCES

    23  0000H     2  A. . . . . .    INTEGER PARAMETER          24
    10  0000H     2  AFTCABIN . . .  INTEGER EXTERNAL(5)        29  48
    20  0000H     2  AFTCG. . . . .  INTEGER EXTERNAL(15)       41  45
    15  0000H     2  AUXFUEL. . . .  INTEGER EXTERNAL(10)       28  48
     8  0000H     2  AVBAY. . . . .  INTEGER EXTERNAL(3)        29  48
    23  0000H     2  B. . . . . .    INTEGER PARAMETER          24
    23  0000H     2  C. . . . . .    INTEGER PARAMETER          24
    18  0000H     2  CGPOS. . . . .  INTEGER EXTERNAL(13)       48
     2              DCL. . . . . .  LITERALLY
    13  0000H     2  DISTSEAT1. . .  INTEGER EXTERNAL(8)        48
    22  0002H     2  EMPTYCG. . . .  INTEGER DATA          48
    21  0000H     2  EMPTYWGT . . .  INTEGER DATA          29  48
     3              ENDDO. . . . .  LITERALLY
     4              ENDIF. . . . .  LITERALLY
    19  0000H     2  FWDCG. . . . .  INTEGER EXTERNAL(14)       32  36
    14  0000H     2  MAINFUEL . . .  INTEGER EXTERNAL(9)        28  48
    23  0000H        MULDIV . . . .  PROCEDURE INTEGER EXTERNAL(16) STACK=0000H  48
     9  0000H     2  NOSEBAY. . . .  INTEGER EXTERNAL(4)        29  48
    12  0000H     2  OTHERWGT . . .  INTEGER EXTERNAL(7)        29  48
     5  0000H     2  SEAT12 . . . .  INTEGER EXTERNAL(0)        29  48
     6  0000H     2  SEAT34 . . . .  INTEGER EXTERNAL(1)        29  48
     7  0000H     2  SEAT56 . . . .  INTEGER EXTERNAL(2)        29  48
    17  0000H     2  TOACWGT. . . .  INTEGER EXTERNAL(12)       29  30  36  39  45  48
    16  0000H     2  TOFUEL . . . .  INTEGER EXTERNAL(11)       28  29
    26  0000H   342  WEIGHTANDBALANCE  PROCEDURE PUBLIC STACK=000CH
     1  0000H        WGTBAL . . . .  PROCEDURE STACK=0000H
    11  0000H     2  WINGLOCK . . .  INTEGER EXTERNAL(6)        29  48
```

MODULE INFORMATION:

```
CODE AREA SIZE     = 0156H    342D
CONSTANT AREA SIZE = 0004H      4D
VARIABLE AREA SIZE = 0000H      0D
MAXIMUM STACK SIZE = 000CH     12D
125 LINES READ
0 PROGRAM ERROR(S)
```

END OF PL/M-86 COMPILATION

REFERENCES

1. Denery, D. G.; Jackson, C. T.; Callas, G. P.; Berkstresser, B. K.; and Hardy, G. H.: Integrated Avionics for Future General Aviation Aircraft. AIAA Paper 78-1482, 1978.

2. Denery, D. G.; Callas, G. P.; Jackson, C. T.; Berkstresser, B. K.; and Hardy, G. H.: A Demonstration Advanced Avionics System for General Aviation, SAE Paper 790569, Business Aircraft Meeting and Exposition, Century 11, Wichita, Kans., Apr. 1979.

3. Honeywell, Inc.; and King Radio Corp.: Demonstration Advanced Avionics System (DAAS) Final Report. Contract NAS2-10021 Phase 1, NASA CR-166281, 1982.

4. Honeywell, Inc.; and King Radio Corp.: Demonstration Advanced Avionics System (DAAS) Functional Description. Contract NAS2-10021 Phase 1, NASA CR-166282, 1982.

5. Hardy, G. H.; Callas, G. P.; and Denery, D. G.: Preliminary Results from the NASA General Aviation Demonstration Advanced Avionics System Program. Proceedings of the Society of Experimental Test Pilots 26th Symposium, Beverly Hills, Calif., Sept. 1982.

6. Callas, G. P.; Denery, D. G.; Hardy, G. H.; and Nedell, B. F.: Flight Evaluation Results from the General Aviation Advanced Avionics System Program. NASA TM-84397, 1983.

7. Denery, D. G.; Callas, G. P.; Hardy, G. H.; and Nedell, W.: Design, Development, and Flight Test of a Demonstration Advanced Avionics System. AGARD 45th Symposium of the Avionics Panel on Advanced Concepts for Avionics/Weapon System Design, Development, and Integration, Ottawa, Canada, Apr. 18-27, 1983.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-85942		

4. Title and Subtitle	5. Report Date
SOFTWARE MODIFICATIONS TO THE DEMONSTRATION ADVANCED AVIONICS SYSTEMS (DAAS)	August 1984
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
Bill Nedell and Gordon Hardy	A-9713
	10. Work Unit No.
9. Performing Organization Name and Address	T-5126
Ames Research Center Moffett Field, CA 94035	11. Contract or Grant No.
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Technical Memorandum
National Aeronautics and Space Administration Washington, DC 20546	14. Sponsoring Agency Code
	532-01-11

15. Supplementary Notes

Point of Contact: Bill Nedell, Ames Research Center, MS 210-9, Moffett Field, CA 94035 (415) 965-5454 or FTS 448-5454

16. Abstract

Ames Research Center initiated a program in 1975 to provide the critical information required for the design of integrated avionics suitable for generation aviation. The program emphasized the use of data busing, distributed microprocessors, shared electronic displays and data entry devices, and improved functional capability. A demonstration advanced avionics system (DAAS) was designed, built, and flight tested in a Cessna-402, twin-engine, general aviation aircraft. Software modifications were made to DAAS at Ames concurrent with the flight test program and are documented in this paper. The changes were the result of the experience obtained with the system at Ames, and the comments of the pilots who evaluated the system.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Distributive system	Unlimited Subject Category - 4

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	36	A02

*For sale by the National Technical Information Service, Springfield, Virginia 22161