12

B.S

AFAL-TR-77-167

PALEFAC

H. B. Chalstrom, Jr.
The Charles Stark Draper Laboratory, Inc.
555 Technology Square
Cambridge, MA 02139

September 1977

TECHNICAL REPORT AFAL-TR-77-167

Final Report for Period 1 ~~September~~ 1975 to 31 May 1977

Approved for public release; distribution unlimited.

Prepared for
Air Force Avionics Laboratory, AFSC, AFWAL
Wright-Patterson Air Force Base
Ohio 45433

[redacted]

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

James R. Sheffield
Signature
Name
Project Engineer/Scientist

L. Daniel Snyder
Signature
Name
Supervisor

FOR THE COMMANDER

Robert A. Dessert
Signature and Title

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFAL/AA/DAIS *,W-PAFB, OH 45433 to help us maintain a current mailing list".*

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 18 AFAL-TR-77-167 ✓	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER 9
4. TITLE (and Subtitle) 6 PALEFAC.		5. TYPE OF REPORT & PERIOD COVERED Final Report. 1 Mar[redacted] 1975-May [redacted] 77,
		6. PERFORMING ORG. REPORT NUMBER 14 R-1087 / 31
7. AUTHOR(s) 10 H. B./Chalstrom Jr		8. CONTRACT OR GRANT NUMBER(s) 15 F33615-75-C-1206 New
9. PERFORMING ORGANIZATION NAME AND ADDRESS The Charles Stark Draper Laboratory, Inc. 555 Technology Square Cambridge, MA. 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 16 Project 2052 Task 0212 17 63243F
11. CONTROLLING OFFICE NAME AND ADDRESS 12 24p.		12. REPORT DATE 11 Sep[redacted] 1977
		13. NUMBER OF PAGES 19
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) System Avionics Division Air Force Avionics Laboratory Wright-Patterson Air Force Base, Dayton Ohio 45433		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release;
distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Top-down Functional Design, Digital Avionics, Software, Compilers, Higher Order Languages, Software Reliability, Real-time, Executives, Software Partitioning, JOVIAL.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the Palefac system which is part of the non-real-time support software of the Digital Avionics Information System (DAIS) of the Air Force Avionics Laboratory (AFAL). Palefac is a tool which aids in the development of real-time flight software for avionics embedded digital computers. This report deals with three aspects of the relationship of Palefac to the DAIS program.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

408 386 ii

LB

Table of Contents

List of Illustrations

1. Introduction

This report describes the Palefac system which is part of the non-real-time support software of the Digital Avionics Information System (DAIS) of the Air Force Avionics Laboratory (AFAL). Palefac is a tool which aids in the development of real-time flight software for avionics embedded digital computers. This report deals with three aspects of the relationship of Palefac to the DAIS program.

Section 2 gives some background of the DAIS program and its objectives and explains how the need for Palefac arose. Section 3 is a step-by-step description of the procedure for developing mission software using Palefac. This procedure begins with the mission requirements and results in operational flight programs. Sections 4 and 5 deal, respectively, with the development history of the Palefac programs and with their internal structures.

Section 6 offers some conclusions and some recommendations for how to implement Palefac in future DAIS-like systems. It will be shown how useful Palefac is to the DAIS approach to digital avionics software development. Section 7 is a bibliography.

2. Palefac and the DAIS Approach

DAIS is an attempt by AFAL to demonstrate the benefits to be derived from the use of the following in the development of avionics software:

1) digital flight computers,
2) a high order programming language,
3) greater programming rigor, including strict software architecture standards,
4) an automated software development and testing lab,
5) uniform hardware and software interactions,
6) a linkage-editor to produce flight load modules, and
7) fully integrated mission requirements encompassing all mission subsystems.

The anticipated benefits were:

1) more efficient and more capable code produced due to a more formalized and organized approach to the coding and testing of flight code,
2) reduced costs of maintaining software and hardware and of modernizing the avionics system to include new technological developments, again due to better methods of coding and testing,
3) greater software reliability, due to the availablity of more thorough testing techniques and to the more rigorous approach to code writing.

The flight software is composed of two parts: the executive software and the application software. The executive software is the flight computer's operating system. It manages the resources of the flight computer in which it runs and provides service to the applications, such as data moving and task activating. The application software consists of the code for the subsystems, such as navigation and weapon delivery. The executive will get developed early on, will be fairly hardware dependent (though it can be coded primarily in the high order language) and will change little from mission to mission or when new features are added. The application code, on the other hand, will need to be modified and re-tested for each new mission and each new system capability. It is the development of this application code with which Palefac is concerned.

Once a set of flight hardware is defined and an executive is operating on it, the generation of a mission comes down to this loop:

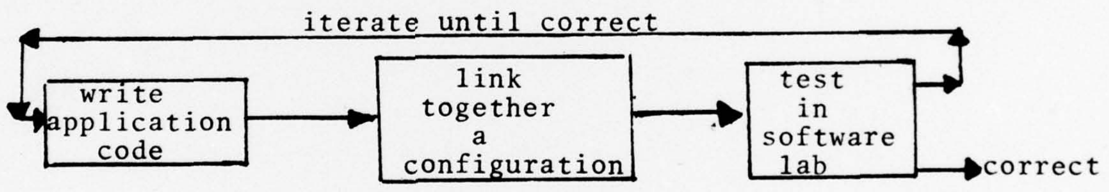


Figure 1 Production of an Operational Flight Program

With the use of a higher order language (HOL) for programming and a simulation test lab for testing, it seemed as though it would be easy to make several iterations of this loop in a short span of time, thereby arriving at better, more efficient code produced in less time. Problems arose, however, partly because of the DAIS federated computer architecture, which has several independent computers communicating over a MIL-STD-1553A bus. Linking together a DAIS configuration involves not only the usual functions of a standard link editor, but also the partitioning of the application software among the several processors and the construction of the tables to drive the executive and to control the data bus. Palefac was conceived to automatically generate the executive and bus control tables and to make the software partitioning transparent to the application code.

3. Mission Generation Using Palefac

As Figure 1 shows, the production of an operational flight program (OFP) is an iterative process involving three steps: application coding, system building, and system verifying. For logical clarity, a different user is designated as the active participant in each step: the application programmer, system designer, and system verifier. In practice, these three users could be the same person.

3.1 The Application Environment

The application programmer writes the application code, which consists of the routines which do navigation, weapon delivery, radar control, communication, pilot display, and all of the other functions which help fly the airplane. Figure 2 shows the environment in which he works. He should be able to code modules without having any knowledge of overall system (i.e. configurations). All that he is concerned with is his inputs, his outputs, the algorithm to use, and any other constraints placed upon the module. (Since any additional constraints degrade both the code produced and the efficiency of code production, they should be kept to a minimum, or eliminated altogether if possible).

The application programmer compiles his module and tests it out as far as is possible. He then runs it through the Palefac Pre-processor which extracts the executive pseudo-instructions used by the module and stores them in the Palefac Module Input (PMI) file. This PMI file is later used as input to Palefac and is the interface between the Application and System Build environments.

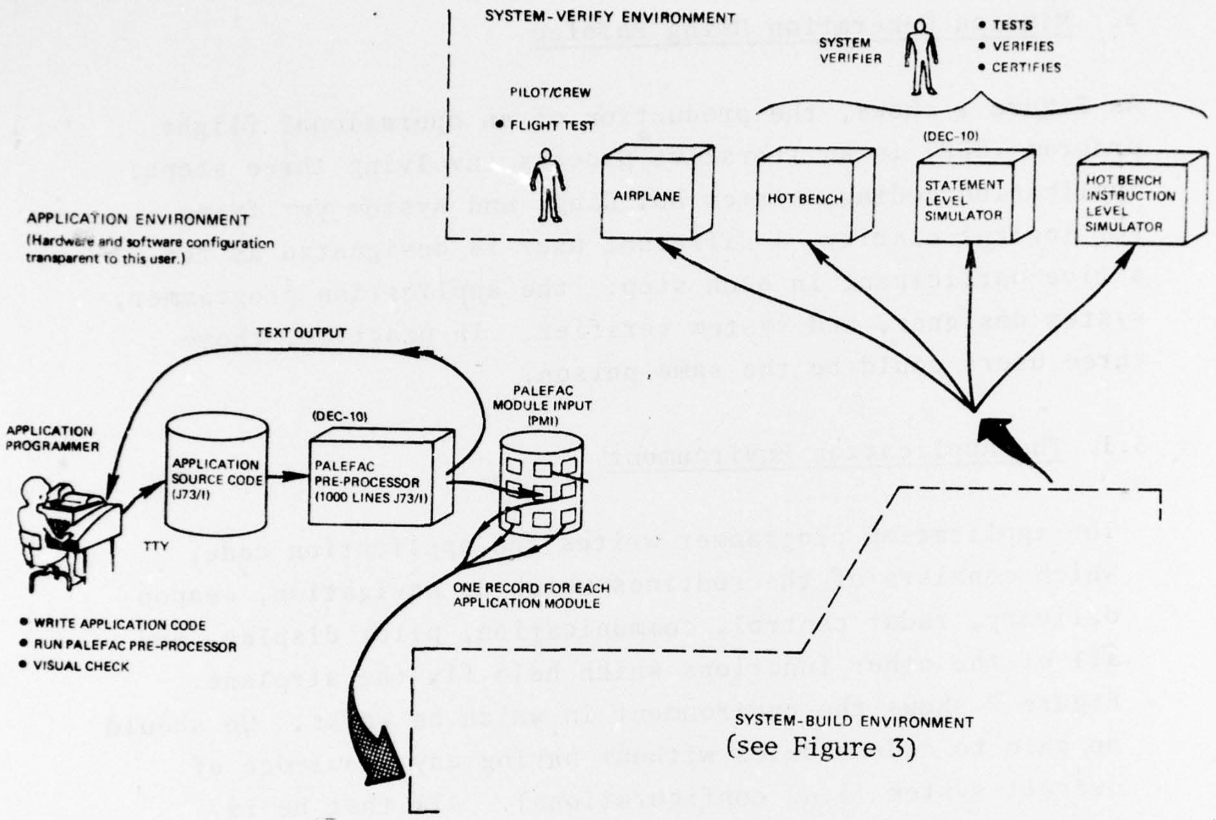
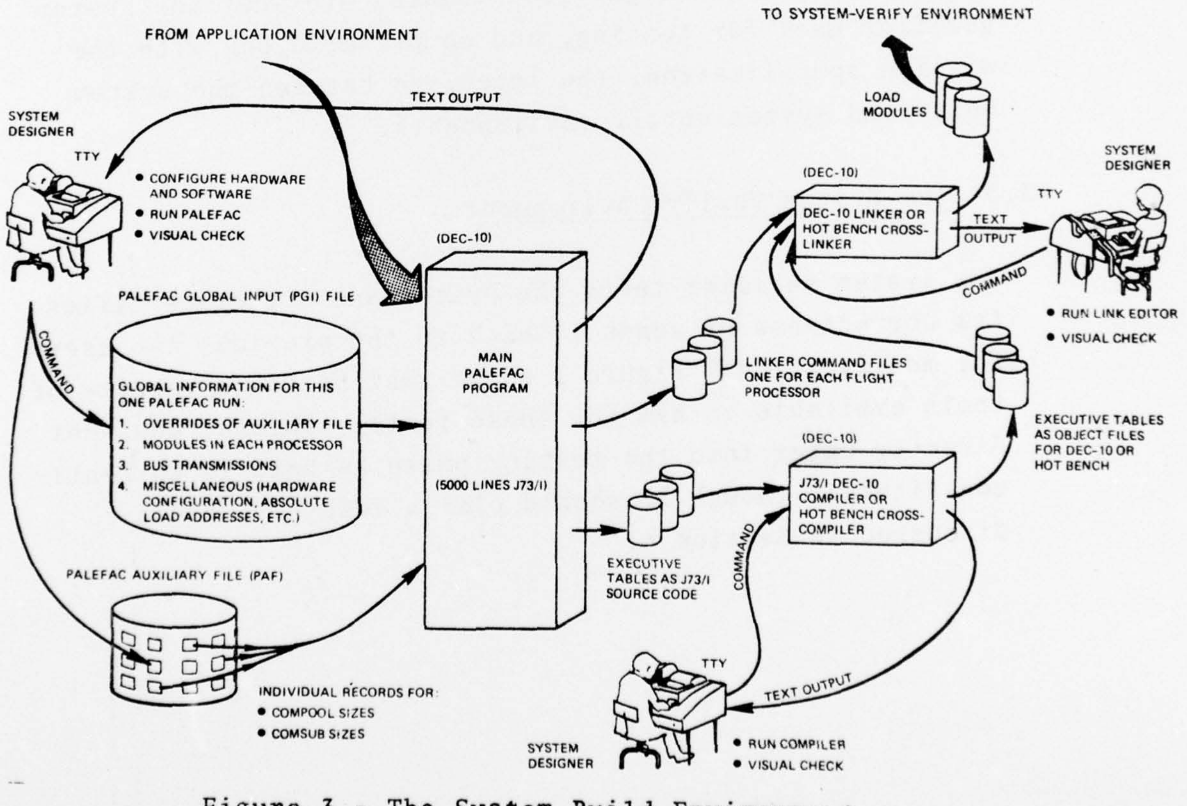

SYSTEM-VERIFY ENVIRONMENT
SYSTEM VERIFIER
• TESTS
• VERIFIES
• CERTIFIES
PILOT/CREW
• FLIGHT TEST
(DEC-10)
AIRPLANE
HOT BENCH
STATEMENT LEVEL SIMULATOR
HOT BENCH INSTRUCTION LEVEL SIMULATOR
APPLICATION ENVIRONMENT
(Hardware and software configuration transparent to this user.)
TEXT OUTPUT
APPLICATION PROGRAMMER
APPLICATION SOURCE CODE (J73/I)
(DEC-10)
PALEFAC PRE-PROCESSOR (1000 LINES J73/I)
PALEFAC MODULE INPUT (PMI)
TTY
ONE RECORD FOR EACH APPLICATION MODULE
• WRITE APPLICATION CODE
• RUN PALEFAC PRE-PROCESSOR
• VISUAL CHECK
SYSTEM-BUILD ENVIRONMENT
(see Figure 3)

Figure 2 - The Application and System Build Environments

3.2 The System Build Environment

The system designer chooses a particular configuration of hardware and software to use for the mission. He must decide how to partition the software among the several federated processors, how many remote device stations (RT's) to have and which devices will be on each one, and where the remote devices get their inputs from and send their outputs to (i.e. which compools). The criteria used in partitioning the software are not well defined at this point. Certainly if two modules communicate with each other a great deal, they should be in the same processor. Likewise, if two modules execute cyclically in phase and at a high rate, they probably should be in different processors so that they do not compete for limited CPU resource.

Figure 3 shows the environment in which the system designer operates. After configuring the system and generating a Palefac Global Input (PGI) file which represents that configuration, he runs Palefac, also using as input the PMI file created by the application programmer. Palefac produces two outputs: a set of linker command file (one for each flight processor) which will generate flight-processor load modules, and the executive and bus command tables as JOVIAL J73/I source code.



FROM APPLICATION ENVIRONMENT
TO SYSTEM-VERIFY ENVIRONMENT
TEXT OUTPUT
SYSTEM DESIGNER
TTY
• CONFIGURE HARDWARE AND SOFTWARE
• RUN PALEFAC
• VISUAL CHECK
LOAD MODULES
(DEC-10)
DEC-10 LINKER OR HOT BENCH CROSS-LINKER
TEXT OUTPUT
SYSTEM DESIGNER
TTY
COMMAND
• RUN LINK EDITOR
• VISUAL CHECK
(DEC-10)
PALEFAC GLOBAL INPUT (PGI) FILE
COMMAND
GLOBAL INFORMATION FOR THIS ONE PALEFAC RUN:
1. OVERRIDES OF AUXILIARY FILE
2. MODULES IN EACH PROCESSOR
3. BUS TRANSMISSIONS
4. MISCELLANEOUS (HARDWARE CONFIGURATION, ABSOLUTE LOAD ADDRESSES, ETC.)
MAIN PALEFAC PROGRAM
(5000 LINES J73/I)
LINKER COMMAND FILES ONE FOR EACH FLIGHT PROCESSOR
EXECUTIVE TABLES AS OBJECT FILES FOR DEC-10 OR HOT BENCH
(DEC-10)
J73/I DEC-10 COMPILER OR HOT BENCH CROSS-COMPILER
PALEFAC AUXILIARY FILE (PAF)
EXECUTIVE TABLES AS J73/I SOURCE CODE
COMMAND
TTY
TEXT OUTPUT
INDIVIDUAL RECORDS FOR:
• COMPOOL SIZES
• COMSUB SIZES
SYSTEM DESIGNER
• RUN COMPILER
• VISUAL CHECK

Figure 3 - The System Build Environment

The fact that the tables are output by Palefac as source code allows them to be targetted to any machine which has a compiler or cross-compiler for the language. Currently only two machines do have JOVIAL J73/I compilers: the DEC-10 and the DAIS flight processor. This feature insures that, while Palefac is executive-specific (it is useful only for the DAIS executive), its outputs can be used for a variety of computers which may be used for DAIS application software development.

The system designer compiles the Palefac executive and bus control table output files (PMD files) using the necessary compiler or cross-compiler to get them targetted properly. He then links together a load module for each flight processor. These flight load modules are what the system verifier uses for testing, and comprise, along with the mission specification, the interface between the system build and system verify environments.

3.3 The System Verify Environment

The system verifier tests the system and either certifies its correctness or sends it back to the previous two users for modification. Figure 2 shows that he has a variety of tools available to him for these tests. Palefac does not directly enter into the testing phase as DAIS is currently constituted, though it should play a role here as is discussed in Section 6.

4. Palefac Development

The Palefac programs (there are two: Palefac and the Palefac Pre-processor) were developed in five phases:

1) definition of interfaces,
2) top-down functional design,
3) detailed design,
4) coding, and
5) integration and testing.

4.1 Definition of Interfaces

Before any design work was done, several interfaces had to be specified. When work was begun on Palefac, the executive design was at a stage where the formats and, in some cases, even the functions of various executive tables were not known. Also, the manner in which application modules were to request executive intervention was not known. Several months of interface discussions were needed for these issues to be resolved, during which time Draper evaluated various design proposals as to their feasibility and how efficiently they could be done automatically by Palefac. The end result of these interface meetings was the Draper produced Interface Control Document between Palefac and the Mission Software. This was the first DAIS publication which described the exact formats of the executive and bus control tables. Other interface documents were written during this time span, but were much more straightforward. These include the SDVS to Palefac and JOVIAL to Palefac interface control documents.

4.2 Top-down Functional Design

Once all of the interfaces were designed, all of the Palefac system inputs and outputs were known. The job of getting from a set of inputs to a set of outputs was viewed as a function, which was decomposed, working from lesser to greater level of detail, into sub-functions. A functional diagram of the two Palefac programs was produced, containing about thirty modules for Palefac and ten for the Pre-processor. The final versions of the programs have forty-three and twenty modules, respectively, which makes it seem as if the original design was rather far off. This is not the case, however.

The modules in the original design retained their intended functions and interactions with each other in the final product. The additional modules were required for two reasons which were not foreseen at design time. First, the JOVIAL J73/I language turned out to be grossly inadequate in the area of input/output. Several FORTRAN modules had to be added to do reading, writing, opening and closing of disk files. The second area which required additional programming was the interface with the DEC-10 operating system. Operation with the SDVS require the special operating system features to be used which are not accessible from JOVIAL. Routines had to be added which were written in MACRO-10, the DEC-10 assembly language.

4.3 Detailed Design

This phase involved examining each module as a separate function with a set of inputs and a set of outputs. The work of this phase was to design for each function an algorithm which will accomplish the transformation from input to output. Occasionally, it became clear that the desired output could not be obtained from the specified input, at which time functional design specification had to be changed.

4.4 Coding

Because each function had been specified in such detail before any coding had been done, the coding phase simply involved translating algorithms from the language in which they were specified in the detailed design document into JOVIAL J73/I. For each module in the Palefac programs, a standalone test module was written. This test module executed the Palefac module on a specific set of inputs and verified the correctness of the results produced. By the end of the code production phase, all modules had been written and successfully standalone tested.

4.5 Integration and Testing

This phase of program development is traditionally the most expensive and the one whose cost can least be accurately estimated. We found integration of Palefac to be astoundingly easy and trouble-free. The almost forty modules of the Palefac program were integrated into a working program in the space of about two weeks from start to finish. We attribute this to two factors. First, the comprehensiveness of the design phase insured that all of the interactions between modules were clearly defined. Once this distinct division of responsibilities was made, coding each function became a relatively small and error-free task.

The second factor which made integration easier was the standalone testing of the modules during coding. The modules not only fit together easily but were nearly error-free internally.

Integration proceeded from the top downward in the program functional tree. The control program was written, but called only dummy sub-functions until those sub-functions themselves had been integrated in a similar fashion.

Several modifications have been made to the Palefac programs since they were first integrated due to changes in the executive and SDVS requirements. During the debugging which was necessary after each program modification, almost none of the bugs found were interface problems, but were instead problems in the algorithm of a particular module. This situation supports the hypothesis that extra time and money spent up front for a good design can save many times its cost in coding and integration.

4.6 Coding Standards

Coding standards helped to make the code produced more uniform in structure from one programmer to the next, as well as more error free. However, standards were not employed as extensively as they should have been nor were they enforced.

The block structuring of JOVIAL helped obviate the use of unconditional and conditional explicit branches in most cases. The GOTO statement was used, however, only as an escape when an error condition was encountered.

5. Program Internal Structure

Detailed descriptions of the modules in both programs and their interfaces can be found in the final specification documents. The present section describes the top-level of program control and sub-function operation.

Both the Palefac and Pre-processor programs were originally written to run from the DEC-10 monitor, with the top-level sequencer module as the main program. In order for the Palefac programs to be executable from the SDVS, three new modules had to be added to each program:

- a MACRO-10 main program (EXEC),
- a JOVIAL module to set up the SDVS return parameter block (SDVSFN), and
- a FORTRAN module to print the SDVS return parameter block (FORSDV).

The MACRO-10 program, EXEC first determines whether it was executed from SDVS or the DEC-10 monitor. If run by SDVS, then error traps are enabled so that errors will not cause the user to return to the monitor, but to SDVS. EXEC then passes control to the sequencer module of the program.

5.1 The Pre-Processor Program

The Pre-processor program has five parts:

1) command string interpret,
2) application code read,
3) PMI output,
4) text output, and
5) decode.

The command string interpret phase first determines whether this command string is for pre-processing or decoding. If it is for decoding, the decode function is called and it does the remaining command string interpretation for itself. If the string is a pre-processor one, then the rest of it is interpreted and the sub-parts of the string are stored in internal tables for later use. For pre-processing, the command string interpreter calls the application code reader.

In this second phase, the JOVIAL J73/I code for the application module is scanned for keywords which indicate executive pseudo-instructions. These are extracted and compacted by the Pre-processor.

The PMI output phase combines all of the information extracted from the application module into a PMI record of proper format. This record is then written into the PMI file specified on the command string.

The text output phase is called only if the verbosity level specified in the command string is 2. This module outputs the information which was extracted from the application module, formatted so as to be easier to read.

The decode function is called when the Pre-processor is being used to decipher PMI records rather than to Pre-process application programs. This function completes interpretation of the command string and then goes into user query mode. The user is asked which records he would like decoded from the PMI file specified in the command string. Then the same text output module as is used for pre-processing is called to decode the specified records and print them.

5.2 The Palefac Program

There are six main components of the Palefac program:

1) initialization,
2) Identifier Table construction,
3) table building,
4) tasking output,
5) datablock output, and
6) other output.

In the initialization phase, the command string is read from the terminal and decomposed into its component file specifications which are then stored in internal tables. The current date and time are determined and saved. Two segments of the PGI file are read into internal storage (the Miscellaneous and Partition segments). Finally, the text output file is opened and some run identification information is written into it.

The Identifier Table construction phase is concerned with filling in most of the primary Palefac internal table called the Identifier Table (IDTAB). In this phase, the PMI and PAF files are read, the comsub local storage areas are calculated, and absolute priorities are assigned to all tasks in the configuration.

The table builder phase accomplishes some additional tasks which must get done before output can begin. The Bus Message segment of the PGI is read. The internal table DMAP, which deals with synchronous transmissions, is sorted and completely filled in. The indices which tasks will have in Task Tables A and B are calculated and saved.

As the names imply, the last three phases are concerned with output. In the tasking output phase, tables concerned with task control are output. These tables include the event, task, and mission cycle event generation tables. Also, the DMA pointer blocks and SYNPTR tables are output here. The datablock output phase writes tables concerned with I/O, like the SIL, RAT, MINR, MIST, RDT, and the compool areas. The other output section has three functions. First, the overlay statements are copied from the temporary disk files into which they have been written into the PMD files. Then the PPI file is written for whatever the target linker is. Finally, the output files are copied into the text output file, if the verbosity value so indicates.

6. Conclusions and Recommendations

Based upon our experience with software development on DAIS using Palefac, we make recommendations for enhancements to the program. Palefac is the information clearing house of the software development. It is the place where information about hardware configuration, software partitioning, task hierarchy, and data flow all come together to form a total picture of the avionics system being produced. There are a number of ways in which this opportunity should be exploited.

First, Palefac should provide more extensive text output to the user to aid in software production. Data-flow maps, control structure maps, priority maps, and a variety of other graphical displays of system operation can be produced here. These things, when used in conjunction with other features of an automated software production laboratory, can make the generation of mission code much simpler and more reliable.

Second, since Palefac is examining both the application software code and the configuration it should determine both feasibility and adherence to standards. Palefac should determine if the software architecture standards are being violated either in code itself or by the particular configuration chosen. Analysis of finite shareable resources (e.g., processor time and data bus) would help the user determine if his configuration is even feasible, given the resources available to it.

Third, Palefac logically belongs as a subfunction of a software production laboratory, like the Software Design and Verification system (SDVS) of the DAIS program. Although the functions of the Palefac are essential to software production in a DAIS-like environment, those functions should appear to the user as part of the software lab. For example, the processing of mission code through the Palefac Pre-processor program should be part of the general translation function where compilation is done.

Most efficient operation would have the Pre-processor function accomplished by the compiler. While the source code was being scanned, the compiler would recognize constructions of the language and the Pre-processor would recognize executive service requests (which are the Pre-processor inputs).

The functions of the Palefac program belong to the phase of the software lab in which the test case is built. The user would specify a configuration, and Palefac would be called on to generate the executive tables and linker commands, and do analysis and output. Then, the system would be linked and tested as specified by the user.

To summarize, it is clear from our experience with software production on DAIS that Palefac is the information clearing house of the software development process. This feature should be taken advantage of to do much more system analysis and output. Also, Palefac should be an invisible part of the software production lab.

7. Bibliography

1. Palefac User's Guide, Charles Stark Draper Lab., Inc., revised May 1977 (DAIS number MA202200).

2. "Palefac and the DAIS Program", H. B. Chalstrom and J. A. Chalstrom, NAECON 1977 proceedings, May 1977. (Draper report number P-430).

3. Palefac Pre-processor Detailed Design Specification - Final, Charles Stark Draper Laboratory, Inc., revised May 1977. (DAIS number 202201).

4. Palefac Detailed Design Specification - Final, Charles Stark Draper Laboratory, Inc., revised May 1977 (DAIS number SA202200).

5. Palefac Pre-processor/Palefac to Mission Software Interface Control Document, revised May 1977 (DAIS number SA802309C).