n√

# CABPRO: An Expert System for Process Planning Multiwire Cables


Kansas City Division

Robert M. Schaefer



KCP-613-5291

Published April 1994


Final Report



MASTER


Prepared Under Contract Number DE-AC04-76-DP00613 for the

**United States Department of Energy**


AlliedSignal

AEROSPACE

## DISCLAIMER



Printed in the United States of America.

This report has been reproduced from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P. O. Box 62, Oak Ridge, Tennessee 37831; prices available from (615) 576-8401, FTS 626-8401.

Available to the public from the National Technical Information Service, U. S. Department of Commerce, 5285 Port Royal Rd., Springfield, Virginia 22161.

**A prime contractor with the United States Department of Energy under Contract Number DE-ACO4-76-DP00613.**

**AlliedSignal Inc.**
**Kansas City Division**
**P.O. Box 419159**
**Kansas City, Missouri**
**64141-6159**

KCP-613-5291
Distribution Category UC-705



# CABPRO: An Expert System for Process Planning Multiwire Cables

Robert M. Schaefer

Published April 1994

Final Report

Robert M. Schaefer, Project Leader
Project Team:
James S. Colmer
Michelle Miley

Technical Communications
Kansas City Division

AlliedSignal
AEROSPACE

# Contents

*Section* *Page*

Abstract . . . . . 1

Summary . . . . . 1

Discussion . . . . . 2

Scope and Purpose . . . . . 2

Prior Work . . . . . 3

Activity . . . . . 3

Learning AI . . . . . 3

Selection of the Hardware and Software Development Environment . . . . . 4

Engineering Process Planning . . . . . 5

Knowledge Acquisition . . . . . 7

The CABPRO Prototype . . . . . 7

Accomplishments . . . . . 15

Future Work . . . . . 15

Reference . . . . . 16

Appendices

A. User Interface Screens . . . . . 17

B. Sample Cable Description File . . . . . 34

C. Sample Operation Level Plan . . . . . 41

# Illustrations


*Figure* *Page*

1 AI Engineering Workstation Network . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

2 Three Primary Modules of the CABPRO Prototype . . . . . . . . . . . . . . . . . . 8

3 Expert System Architecture . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

4 Hierarchical Rule-Based Network of CABPRO . . . . . . . . . . . . . . . . . . . . . 9

5 Example Cable Sketch . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

# Abstract

*CABPRO (CABle PROcessor) is a set of computer programs using Artificial Intelligence programming to automatically generate process plans and work instructions in support of the manufacture of multiwire cables. Development of these programs required selecting appropriate hardware and software tools, defining engineering process planning activities, acquiring and representing process planning knowledge, and creating a prototype system. A successful prototype was developed and demonstrated.*

# Summary

The CABPRO project was formally initiated in April 1985 to create a set of computer programs using artificial intelligence (AI) programming techniques to perform generative process planning in support of the manufacture of multiwire cables. The primary objective was to use CABPRO to do repetitive and labor-intensive work directions creation activities by emulating the tasks currently being done by cable engineering experts. Process Engineers would use CABPRO to help plan and sequence manufacturing operations and assemble standard sets of text into work directions to be delivered to manufacturing personnel for cable fabrication.

Major activities included learning AI technology, selecting hardware and software tools, defining engineering process planning activities, acquiring and representing process planning knowledge, and creating a prototype system.

Before programming of the prototype could begin, a clear understanding of the engineer's process planning mode of operation had to be gained. This was accomplished by repeated personal interviews with five cable engineering experts. Some interviews were recorded and loosely transcribed. From the interviews came an understanding of formal and informal steps used by cable engineers to select and sequence manufacturing operations. This expert knowledge was represented in the computer program as "rules of thumb" in the form of *if-then* statements.

Using information gained from the interviews, a prototype Expert System, the CABPRO prototype, was developed. Expert systems are AI programs that emulate an expert's methodology for problem solving. The prototype successfully created work directions for three different cable designs. The work directions for all three of the cable designs matched identically or were better than those created by human experts. Accuracy of data in the work directions created by CABPRO, especially in cases where numbers were duplicated in several operations, was superior. The CABPRO prototype provided approximately a 6-to-1 reduction in time required for human experts to perform the same tasks.

Successful demonstration of the prototype system and subsequent evaluation of system architecture provide proof of concept and reason to develop a production version of CABPRO.

# Discussion

## Scope and Purpose

Multiwire cable fabrication is performed by human operators following written instructions. These instructions document and communicate the process used in cable assembly. Cables are manufactured in low volume and come in a multitude of different designs. Many different cables are in production at one time. In addition, design changes often require new written instructions. In this environment, operators seldom become completely familiar with specific cable designs. Therefore, work instructions for each cable design must be accurate, detailed, and current.

The written instructions that the operator follows are called travelers. They "travel"" with the cables to whatever workstations and operators are needed during assembly. All materials, components, tools, machines, and any additional information needed for fabrication must be explicitly defined for each operation. Whether the operation is done by hand (such as twisting wires to improve flexibility of branches) or with the aid of machines (such as transfer molding to insulate connectors), each discrete action is defined in the traveler.

A typical multiwire cable traveler might consist of 80 to 100 pages defining 60 different manufacturing operations. The writing of these travelers represents a significantly large part of the process planning function. To select which operations apply to a specific cable design, to determine what sequence of those operations will produce the best quality cable at the lowest cost, and then to compose the traveler takes an average of 40 engineering manhours.

Using expert system technology to automate the writing of travelers will save many manhours. Travelers will be created with consistency and accuracy. The time required for process planning can be dramatically reduced, freeing engineers for more challenging tasks. It is important to note that a side benefit of using expert system technology will be that the knowledge of experienced cable process engineers can be captured and stored in the system.

CABPRO (for CABle PROcessor) will be an expert system that uses knowledge from experienced cable process engineers to compose travelers directly from design data. There are three primary objectives.

1. ***Automate 60% of the engineering process planning function in support of the manufacture of multiwire cables.***

   Process planning is defined as review of cable designs, selection of manufacturing operations, optimally sequencing the selected operations, writing a draft copy traveler, and creating a process materials list. Sixty percent automation is the goal since special case designs and maintenance of travelers are not planned as part of the system.

2. ***Retain the knowledge of experienced cable engineers***

   As experienced engineers retire or leave, a great store of knowledge is taken with them. Capturing and storing this expertise in a computer knowledge base will preserve that knowledge for use by newer, less experienced engineers. By using a common knowledge base, consistent, accurate, and timely process plans can be created.

3. ***Gain practical expert system development experience***

   Expert system technology is new to the KCD. Only a few projects are currently under way. Success in these projects, including CABPRO, may open possibilities for improved productivity in many areas using expert systems.

The approach to be taken to meet these objectives is:

- learn Artificial Intelligence (AI)/Expert System technology,
- select hardware and software to be used for a development environment,
- define engineering process planning activities,
- acquire and represent expert process planning knowledge, and
- create a prototype system.

The prototype system will be able to accept design data input from the process engineer, create a draft copy traveler, and create a process materials list for each operation. The test cable will be the D55 training cable.

## Prior Work

In addition to defining process planning activities performed by process engineers, an essential part of this expert system is defining available manufacturing techniques and capabilities. Cable fabrication techniques and capabilities have been defined and organized into the "Standard Traveler" for cables. This is a collection of approximately 275 operations available at KCD for the fabrication and inspection of cables. Each operation in the Standard Traveler has associated with it a standard set of words, approved by engineering and manufacturing, to describe actions to be taken to complete the operation. The intent of the Standard Traveler is to give process engineers a tool to start with when creating a traveler specific to a cable design.

The Standard Traveler was developed by and is maintained by cable process engineers. Its development was not part of this project.

## Activity

### Learning AI

One of the major activities during the first eight months of the CABPRO project was to learn AI/Expert System technology. Several different methods were used to acquire the knowledge necessary to begin an expert system development project.

1. Conferences, seminars, and short courses provided information on the state of the art, capabilities, applications, and terminology of expert systems.

2. Professional journals and magazines provided updates of current activities and information about the architecture or workings of expert system.

3. KCD employees involved in Computer Integrated Manufacturing (CIM) and other AI related projects were a valuable source of AI expertise.

4. The personal computer purchased for this project has provided tutorials in Lisp, a common expert system programming language.

5. Discussions held with computer vendors such as Digital Equipment Corp., SUN Microsystems, and Apollo Computers have also proved helpful in learning expert system technology.

## Selection of the Hardware and Software Development Environment

An IBM Personal Computer XT purchased for the CABPRO project was delivered in July 1985. Software purchased with the PC included Gold Common Lisp, Word, and Turbo Pascal. Included in Gold Common Lisp was a Lisp programming tutorial which proved to be a valuable learning tool. The PC, because of its limited memory, processing speed, and available software tools, was not a good choice as the prime computer in developing expert systems.

In August 1985, an implementation plan for the AI Engineering Workstation Network (Figure 1) was approved by the Department of Energy (DOE). The workstation network, with an approximate cost of $395,000, was selected for funding by DOE as a project with high productivity and high risk. The network includes five Apollo engineering workstations connected by a network consisting of coaxial cable and Apollo proprietary network management software. The five workstations are physically located in three different departments at the KCP. The procurement was a joint effort between Electrical Products Engineering and Mechanical Engineering.

The AI Engineering Workstation Network is the foundation for AI/expert system development. Several choices were made regarding other appropriate tools to use in the development of the prototype version of CABPRO. These choices were made based on available technology and applicability to the cable process planning domain. Tools include NIAM information modeling; HERB (an inference engine developed at the KCD); Common Lisp with a FLAVORS extension; the C programming language; and DIALOGUE (an Apollo

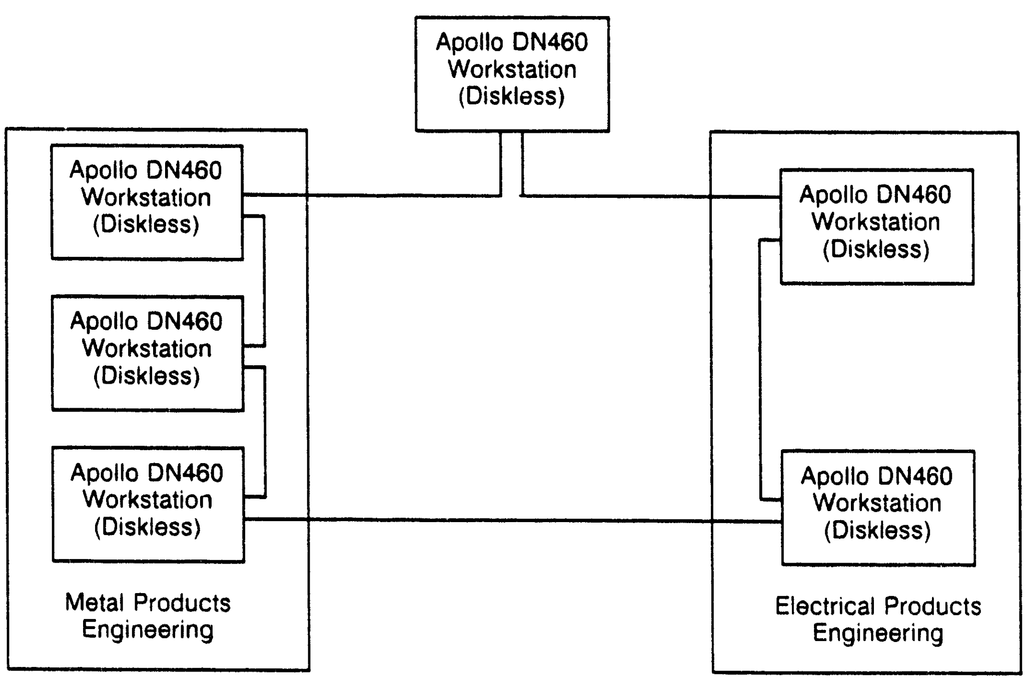



**Figure 1. AI Engineering Workstation Network**

windowing tool kit for user interface development).

The NIAM information analysis methodology is a binary relation method of modeling linguistics/semantics of the static information concerning the problem. It has been shown to provide more facility in data modeling than the traditional Entity Attribute Relationship approach, although not as much accuracy as the interpreted predicate logic approach.[1] NIAM is easy to learn and use. Also, software was available in-house for producing a neutral data model from the binary relation model. The neutral data model resulting from the information modeling was used to aid in design of internal data representations of the system. This approach was found to help in eliminating data access difficulties during the function and rule writing phase of the project. The methodology also proved useful as an interview tool in gathering knowledge from experienced engineers.

HERB is an inference engine/expert system development environment produced as part of the XCUT project at the KCD. HERB supports multiple rule bases, forward and backward chaining, pattern matching, multiple conflict resolution strategies and the execution of Lisp code at any point in a rule. The multiple rule base feature was attractive not only to modularize programming but also to take advantage of the inherent hierarchical nature of process planning. Rules are entered in typical *if-then* form, which requires a working knowledge of the Lisp programming language. The internal HERB database supports FLAVORS representations as well as any Lisp data structure.

Common Lisp was chosen as the programming language to use from within the rules of the CABPRO prototype system because of its compatibility with HERB and FLAVORS. Lisp is a symbolic interpretive programming language that is packaged with editing and debugging tools well suited for expert system development. FLAVORS was used to define data structures designed with the information analysis (NIAM) work. It provided a straightforward approach to all data representations in the prototype and, because it is an extension to Common Lisp, fit easily into the environment.

C was chosen as the language for input/output programming outside the realm of HERB. It is used with the user interface screen management toolkit and DIALOGUE to create output for the traveler text file.

DIALOGUE is an APOLLO window management tool kit used for the user interface. It provided facilities for multi-window management, mouse or keyboard input, and the ability to incorporate user-defined help screens and help files.

## Engineering Process Planning

Process planning is a multi-step process that follows a hierarchical pattern. The initial planning stages are at a macro, or "activity" level, followed by planning at the "operation" level and finally planning is done at the "task" level.

Process planning begins by reviewing a cable design to identify features. A feature is a basic building block of the cable. The two most common cable features are terminals and branches. Terminals are classified by a subtype–junction, pigtail, or connector–and represent cable locations where conductors are terminated. A branch is all of the conductors (wires) and the jacket (outer sleeve) between terminals. Features are usually associated with pieceparts but may be composed of several pieceparts or pieceparts that have process materials added. For example, a connector is a piecepart that is purchased and identified by a part number. The feature–*connector*–is the piecepart plus any encapsulation material that is added during processing.

Once identified, a feature is associated with one or more activities. An activity is a general manufacturing process in cable fabrication such as soldering wires to a connector. Each cable feature that was identified is associated with all appropriate activities. This is done by a process engineer using the cable design drawing set and the engineer's experience. Some engineers make notes or write down the activity-feature list, especially for complicated designs. Most of the time this list is created mentally in a few hours.

At this stage in process planning there is only a list of features associated with activities. For a cable with three connectors, the engineer would associate the activity–*solder*–to each connector. The engineer knows that once connectors have been soldered, they must be insulated to prevent exposure to moisture. Therefore, *encapsulate* is another activity assigned to each connector.

After all activities have been determined, the engineer decides how to sequence those activities for optimum manufacturability. Sometimes natural constraints are used in the decision making. For example, a connector can not be encapsulated before wires are soldered. At other times sequencing decisions are based on design parameters and general knowledge of cable fabrication techniques. This sequenced list of activities and features is a high-level or macro process plan called an *activity level plan*.

When all activity level decisions have been made, the process plan is exploded in detail to the operation level plan. Associated with each activity are one or more manufacturing operations that can be performed by a single operator at a single workstation. The engineer uses the activity level plan, the cable design data, and cable processing knowledge to select specific operations and create the operation level plan.

At this stage a sequenced list of manufacturing operations exists but there are no real instructions for factory personnel. Again, some engineers create written lists of the operation level plans but most do not.

The next stage of process planning is task-level planning. Tasks are very specific actions performed by factory operators. The process engineer uses the operation level plan, cable design data, and cable processing knowledge to select appropriate tasks.

Since the traveler is how a process plan is communicated to factory operators, the engineer must now start the traveler composition process. Significant progress has been made in the cable domain for communicating process plans through what is called a standard traveler. The standard traveler consists of word/graphic instructions agreed upon by engineering and manufacturing to describe each task needed for cable assembly. Standard words for approximately 275 manufacturing operations have been documented.

Organization of the standard traveler takes advantage of the activity-operation-task hierarchy. An operation is the basic unit of the standard traveler consisting of up to three pages of text with optional illustrations. Tasks are represented as separate paragraphs of one to three sentences each. All operations are logically grouped for reference under a given activity.

When the engineer has conceptualized an operation-level plan, specific operations–pages of text–are selected from the standard traveler. These standard operations contain all possible tasks and have blank fields where the engineer must insert cable-specific data such as piecepart or part numbers. Tasks - paragraphs of text - are selected and customized to fit the specific cable design. The conventional process planning methodology is to gather the standard traveler pages from a filing cabinet and customize the text by hand. The completed travelers are given to a typist for input to the KCD traveler information system.

To better describe the activity-operation-task hierarchy of process planning, the following is offered as an example. Solder and encapsulation are general cable fabrication processes (activities) that map to a specific cable feature, such as *connectors*. Mold preparation, material preparation, cure, and cleanup are four operations grouped under the activity - *encapsulation*. Each one of these operations is represented in the standard traveler by 1 to 3 pages of text. Four tasks associated with material preparation are: *weigh material A, weigh material B, combine*, and *mix materials*. A standard traveler task contains 1 to 3 sentences to describe specific actions required to complete each task.

No formal procedure exists for the steps described above as the *activity-operation-task* hierarchy of process planning. However, this methodology is used by all cable engineers to create travelers. One of the objectives of CABPRO is to *retain the knowledge of experienced cable engineers*. Formalizing the process planning methodology by emulating it with software will retain that knowledge for use by less experienced engineers.

## Knowledge Acquisition

Information about how the engineer performs process planning, engineering "rules of thumb," and general cable engineering knowledge had to be gathered. The primary technique for knowledge acquisition from experts was interviewing. Interviews held with five cable engineers consisted of several one-hour question and answer sessions. Many of the sessions were recorded on cassette tape to ensure accuracy of note taking and to allow for easy flow in questioning.

The NIAM information analysis diagraming technique was used to help organize information gathered during interviews. Using this technique helped set or verify terminology and document relationships between the various cable features. The diagrams are used as input to a software tool that automatically generates database record structures. These record structures form the cable description file that supplies all necessary cable data to CABPRO. In addition, the information analysis diagrams became valuable interview tools. Cable engineers were taught to read the diagrams, which helped ensure their accuracy.

## The CABPRO Prototype

For discussion of the prototype architecture refer to Figure 2. There are three primary modules in the prototype system: the user interface, the rule-based network, and the traveler assembly program.

### The User Interface

The purpose of the user interface (UI) is to provide a comfortable means of entering all design information necessary for process planning, then to organize that information into a format usable by the expert system. The UI is perhaps one of the most important modules in CABPRO, since it is the only part of the system seen by the user. If the UI does not present a

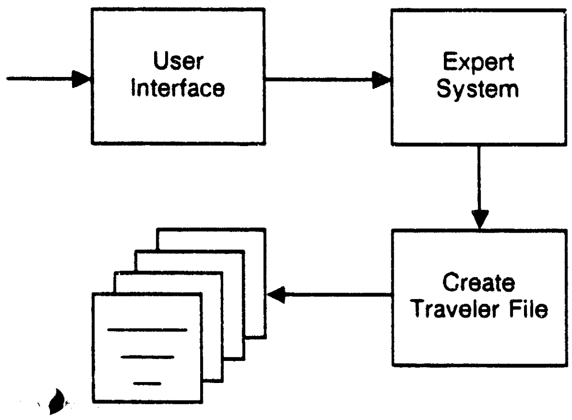



**Figure 2. Three Primary Modules of the CABPRO Prototype**

quick and simple method of interaction, CABPRO will not gain acceptance.

A typical CABPRO user is a process engineer who may or may not have cable domain-specific knowledge. To start a session, the engineer must have a drawing of the cable design. No advance review of the drawing is necessary.

Sample UI screens are included in Appendix A. Screen A-1 is the CABPRO Main Menu which is the first screen that appears after the user types <cabpro> at the operating system prompt. From this menu the user can select any of the seven actions described in the menu in the upper left quadrant of the screen. If the user selects *input cable definition* the menu portion of the screen changes as shown in screen A-2. The *General Information* icon expands to its input window state. In this screen the user inputs requested data. Some fields such as *base part number* require keyboard entry while others such as *Cable type* have pull-down menus to select from a list. The upper right part of the screen will display Help messages when the user points the cursor at a topic and presses the right button. Screen A-2 shows the help message for *Part name*. The UI does a minimum amount of error checking for things like improper data types.

As soon as the user enters a number for connectors, junctions, and pigtails the *terminal information* icon expands for input (Screen A-3). CABPRO will lead the user through the input screens by displaying screens A-2 through A-11 as appropriate for data already gathered. Or the user can expand icons in any sequence desired. Screen A- 12 illustrates nested pull-down windows which make input easy. When all data has been entered the user selects *CLOSE* from Screen A-2 by pressing the left mouse button when the mouse pointer is in the icon. From the *MAIN MENU* the user selects *SAVE DEFINITION* to create the cable description file. A CDF is attached (Appendix B). Each token in the CDF (represented by <token-name>) is replaced with data entered by the user. For example, the token <connpartnumber> corresponds to a connector part number such as "8770605-03." The CDF is a text file of Lisp and Flavors programming code created by a C program using the token replacement procedure. It is stored in the user's home directory just as shown. Generic Flavors definitions used as building blocks for the CDF were designed directly from the NIAM information modeling and interview process. It should be noted that a definition only partially entered may be saved and the user can return later to finish.

When the definition is complete and saved, the user selects *CREATE PROCESS PLAN* from the MAIN MENU to pass the CDF to the expert system portion of CABPRO (see Screen A-13). The CABPRO prototype is a rule-based expert system that incorporates a forward chaining strategy in the inference engine HERB to select and sequence manufacturing operations necessary for the assembly of multiwire cables. Figure 3 illustrates the architecture of the expert system portion of CABPRO.

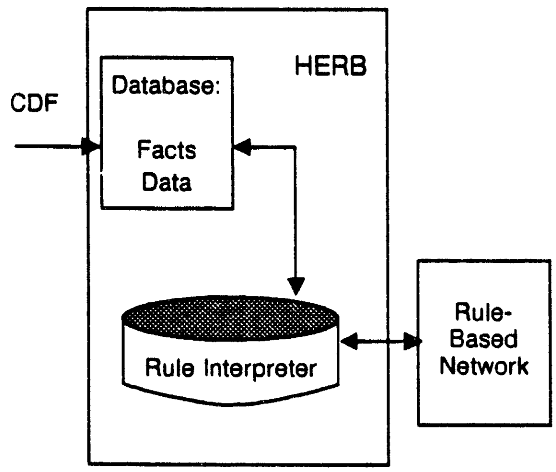



**Figure 3. Expert System Architecture**

HERB (Hierarchical Expert Rule Base system) is the expert system shell used to develop the CABPRO prototype. It supports multiple rule bases, allowing the developer to define and manipulate a rule-base stack. Rule bases can be pushed onto and popped off the stack. Rules in HERB are typical *IF-THEN* statements having a set of *IF* conditions and *THEN* actions. HERB matches conditions with known facts and data with *IF* portions of rules, then selects rules to be executed. When a rule is chosen, the *THEN* actions (usually Lisp commands and/or HERB database manipulation commands) are executed. Five different conflict resolution strategies are offered to break any ties that arise when more than one rule matches. HERB continues the match-select-apply cycle until current facts and data no longer match any rule. Control is then passed back to a Common Lisp environment.

As shown in Figure 4, CABPRO rule bases comprise a hierarchical network operating at two levels for process planning. Sixteen rule bases are grouped into the categories illustrated. At the top level, a rule base is used to decide what rules will be needed to solve the next set of problems and to push and pop the appropriate rule bases as they are needed. Activities are selected and sequenced at one level; then, at a lower level, operations for each activity are selected and mapped to specific standard traveler operation pages. A final rule base takes care of miscellaneous functions like printing the traveler.

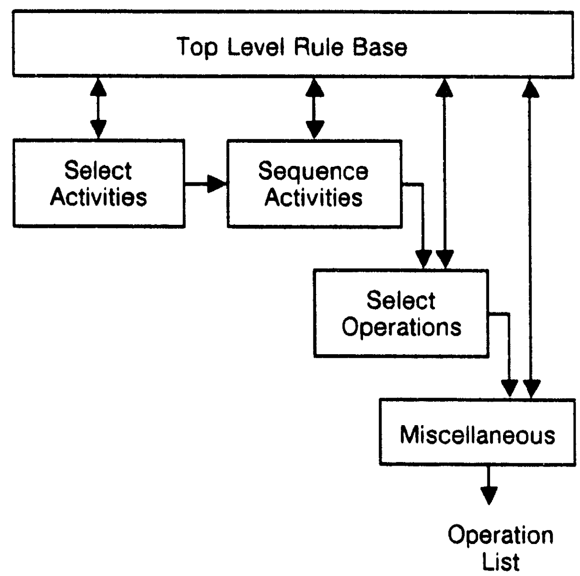



**Figure 4. Hierarchical Rule-Based Network of CABPRO**

The prototype system contains 170 rules distributed among 16 rule bases. All prototype rule bases use the age-directed conflict resolution strategy. The age-directed strategy ensures that rules with conditions matching the newest facts and data in the HERB database are executed first, except rules with a condition designating a GOAL. If the keyword *GOAL* is used to begin a matching condition of a rule and if all other conditions of that rule match, that rule is given special consideration and executed first. This strategy fits the cable process planning problem well because of the goal/subgoal problem solving method used by expert engineers in process planning.

As shown in Figure 4, one top-level rule base is designed to control which rule bases are active at any given time. It does this by controlling the rule base stack. Cable fabrication is done in two stages, a preparatory stage and an assembly stage. In the preparatory stage, all pieceparts are gathered, cut to length, identified, and cleaned. In the assembly stage, the cable is actually formed and pieced together. To take advantage of this constraint, separate rule bases select preparatory and assembly activities.

The sequencing portion of process planning is perhaps the most difficult because of the variety of choices. At the same time, cable fabrication procedures offer several constraints to simplify sequencing. For example, all wires must be soldered to the connectors and all solder joints must be visually inspected before connectors can be encapsulated. These constraints are exploited by the rule-base design. Three rule bases are dedicated to sequencing activities: one for preparatory activities, one for an assembly starting point, and one for the remainder of the assembly activities.

Nine cable fabrication activities can be selected by the CABPRO prototype. A separate rule base exists to select operations for each activity. As operations are selected, a list is created that contains reference numbers mapped to standard traveler text pages.

As shown in Screen A-14, process plans can be viewed or printed. Appendix C contains an example operation level plan. The numbers in the left-most column in the operation level plan are called STMW (Standard Traveler - Multi Wire) numbers and they reference specific operations in the standard traveler library.

Once all of the operations have been selected and sequenced, this operation list is passed to a program that generates a copy of the traveler. The user would select *GENERATE TRAVELER* from the *MAIN MENU* as shown in Screen A-15. CABPRO generates the traveler by comparing STMW numbers in the operation level plan to keys in the standard traveler library database which is built in Flavors and stored in the Lisp environment.

At this point the traveler can be printed as shown in Screen A-16. The user can now exit from CABPRO by selecting *QUIT CABPRO* from the *MAIN MENU*.

The purpose of the prototype is to provide proof of concept for using expert system technology for process planning of multiwire cables. The following example demonstrates that the concept is indeed feasible and that CABPRO will select and sequence operations for a specific cable design.

The previous section presented an overview of the prototype system structure. To better describe selection, sequencing, and other problem solving methods used by CABPRO, an example cable and its generated process plan will be presented. The example will be of a simple, two-connector, one-branch, multiwire cable requiring the use of typical cable fabrication techniques (see Figure 5). This drawing represents a few basic features needed for demonstration. To actually run the prototype a complete design definition is required.

## Describing The Cable

The primary documentation used by the process engineer to describe the cable to CABPRO is the drawing set – the AY and ML. The design parameters are entered through the user interface and organized into a set of Flavors structures called the cable description file. The CDF describes the cable in terms of features and pieceparts. The example cable consists of two connectors identified as features *P1* and *P2*, and one branch identified as

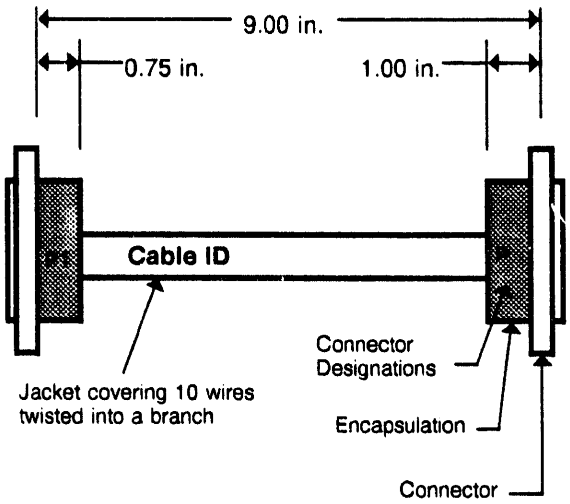



**Figure 5. Example Cable Sketch**

feature *P1-P2*. Pieceparts that make up the features include connectors, jacket, and wire; each piecepart is identified by a part number. Dimensions, diagrams to define wire termination points, general fabrication choices, and guidelines are also specified in the design drawing set and entered into CABPRO. The following is an example of a portion of the Flavors file created for the feature P1 - a connector:

*make-instance 'connector*
*:connector-designation "P1"*
*:connector-type "rack-and-panel"*
*:connector-termination "cup"*
*:connector-insert-type "standard"*
*:connector-part-number "355653-02"*

## Selecting Activities

When the cable description file is complete, it is passed to the rule base network where the first thing to happen is selection of activities. For the demonstration cable, nine activities are selected. The activity list looks like:

*solder p1, solder p2, encapsulate p2, encapsulate p2, form branch p1-p2,*

*marking, prepare branch components, final clean, final inspection.*

CABPRO selects activities by using knowledge stored in rules to look at data in the CDF and decide what processes are needed for fabrication. Here is an example of a rule in the assembly-activity-selection rule base to choose solder for P1:

(DEFRULE "Solder Wires to Connector"
IF
(GOAL select assembly activities)
(GOAL select solder wire activity for P1)
TEST
search CDF to locate connector-termination for P1
match connector-termination value with "cup"
search CDF to locate conductor-type to be terminated at P1
match conductor type with "wire"
THEN
add (solder P1 to the selected activities list)
remove-fact (GOAL(select solder wire activity for P1))

"Solder Wires to Connector" shows the general form and syntax for rules in HERB. The *IF* conditions in this case are both goals. An overall goal to select assembly activities was added to the HERB database by the top-level-control rule base. The second condition, a goal to select a solderwire activity specifically for P1, matches a fact in the HERB database added by a previously executed rule that identified P1 as a feature needing this type of termination activity. The TEST portion of a rule allows the use of Lisp programming code such as function calls or, as in this case, Flavors access/manipulation, to be executed. Design data in the CDF is compared to requirements for needing a solder termination. The TEST portion must return a non-nil value for the rule to be selected for execution. For this rule, all conditions match and the *TEST* is true, so *THEN* statements are executed. Adding <solder P1> to the activities list is done with Lisp and <remove-fact> is a command to manipulate the current state of the HERB database.

Similar activities such as <solder P1> and <solder P2> are selected consecutively because of the age-directed conflict resolution strategy. Facts in the HERB database which match rule conditions similar to the second goal in the example are added for all features needing termination. That of course makes them the newest facts in the database and ensures that rules matching all of the conditions will be selected first. The same scenario is true for other multiple activities, such as encapsulation and the various inspection calls.

## Sequencing The Activities

A rule in the top-level control rule base recognizes when all appropriate activities have been chosen and changes the primary goal from selection to sequencing. The primary goal becomes sequencing preparatory activities starting with separating preparatory and assembly activities. The preparatory activities in the example are:

*marking, prepare branch components.*

The sequencing of these two is trivial. CABPRO knows that the marking will be on the jacket of the P1-P2 branch and, before the jacket can marked, it must be cut to length. Since cutting the jacket to length is in the *prepare branch components* activity, this activity must be sequenced first, making the sequenced preparatory activities list:

*prepare branch components, marking.*

The next step is to select a starting point for assembly. Choosing the first assembly activity is a special case and one rule base is dedicated to that purpose alone. The design of this rule base directs the selection toward *solder* as the first activity, but other activities are considered. Once an activity is chosen, a cable feature must be selected and associated with that activity. With this simple cable containing only connectors and a branch, a termination activity (in this case *solder*) is the preferred way to start. Encapsulation is ruled out because a connector must be soldered before it is encapsulated and no connectors have been soldered yet. *Form branch* is considered and is possible. However, through experience, process engineers have learned that starting at one end - one of the connectors - rather than in the middle with a branch makes overall fabrication easier. *Inspection* and *final clean* activities are not yet appropriate. Solder is then chosen as the first assembly activity. Each one of these decisions corresponds to one or more rules that executed. The rules are *If-Then* Lisp programming representations of knowledge gathered from expert process engineers during the interviews.

Now the feature to be soldered must be selected. Engineers have learned through experience that starting with soldering the most complex feature - connector - makes subsequent fabrication easier. Attributes like connector type, quantity of solder cups, density of solder cups, number of wires to be soldered and approximately 20 other criteria are considered by the engineer when calculating connector complexity. The process engineer reviews known attributes and makes this decision quickly without any real calculation. CABPRO assigns a value to each criteria and calculates the complexity. In the example P1 is chosen because the encapsulation volume is smaller, making it more difficult to keep all the conductors inside. All other criteria compare equally.

The sequence of activities is now:

*prepare branch components, marking, solder p1*

These activities are removed from consideration as control is passed to the rule base designated to sequence the remainder of the activities.

The process by which the remainder cf activities are sequenced relies heavily on constraints that exist in the cable manufacturing domain. The algorithm used in this rule base for sequencing is:

- look at the current state of the process plan
- determine what activities are possible
- determine if any activity should get special consideration
- select the next activity and an associated feature

In the example the current state is *solder p1*. All remaining activities are possible but special consideration is given to *form branch p1-p2* because:

- branch p1-p2 is related to connector p1,
- the branch must be formed before encapsulation,
- another solder cannot be performed before all related branches are formed.

*Form branch p1-p2* is selected and added to the sequenced list that now looks like:

*prepare branch components, marking, solder p1, form branch p1-p2.*

This algorithm is applied with appropriate constraints and rules of thumb until all activities have been sequenced. The final sequenced activities list is:

*prepare branch components, marking, solder p1, form branch p1-p2, solder p2, encapsulate p1, encapsulate p2, final clean, final inspection.*

This list is passed to a rule base that selects operations for the activity represented by the first element.

## Select Operations

To select operations CABPRO must have knowledge of cable fabrication requirements and what operations are available in the standard traveler library. Each rule base for operation selection has one purpose - the activity for which it was created - and therefore need only worry about operations available for that activity. Actual selection of operations is very similar to activity selection. Goals are established indicating possible operations, the CDF is then queried to determine needs, then specific operations are chosen. Each activity is addressed in the order in which it was sequenced. For operations selected in the prototype, sequencing is determined by the order in which it is stored in the standard traveler library. The standard traveler library was developed and organized to make this possible.

Applying the operation selection methodology and sequence strategy to each selected activity, the final process plan for the example cable is:

*Prepare Branch Components*
  *cut wires to length*
  *tin wire ends*
  *cut jacket to length*

*Marking*
*mark jacket*
*apply covercoat over marking*
*Solder P1*
*solder wires to connector*
*spray clean to remove flux residues*
*Form Branch P1-P2*
*twist wires into a bundle*
*install jacket over bundle*
*Solder P2*
*solder wires to connector*
*spray clean to remove flux residues*
*Encapsulate P1*
*prepare mold and install cable*
*mix and pour polyurethane material*
*cure polyurethane*
*remove cable from mold and clean*
*Encapsulate P2*
*prepare mold and install cable*
*mix and pour polyurethane material*
*cure polyurethane*
*remove cable from mold and clean*
*Final Clean*
*mark connector designations*
*clean connectors and install special hardware*
*Final Inspection.*

The final process plan is presented to the engineer for review. If accepted, a copy of the traveler can be requested. Otherwise, the engineer can edit the process plan and alter the order in which operations are performed.

## Create Traveler

Each operation in the process plan corresponds to a set of text pages in the standard traveler library. This module creates a text file of ordered standard traveler pages that can be edited and printed when desired.

## Prototype Results

The purpose of the prototype was to provide proof of concept for using expert system technology for process planning of multiwire cables. The criteria used for measuring the prototype's performance was the comparison of results from CABPRO to process plans developed by engineers. Three cables currently in production with features known to CABPRO were chosen for comparison. One cable consisted of three connectors and two branches while the other two were very similar to the example illustrated above. Design data was entered into CABPRO and in all cases the CABPRO-generated process plans were equivalent to existing plans.

The cables similar to the above example (2 connectors, 1 branch) were chosen for the prototype test because they had similar design but different process plans. Their designs differed in total length and number of conductors in the branch. Following is a summary of differences.

| | Length | Conductors |
|---|---|---|
| Cable A | 9.0 inches | 6 |
| Cable B | 13.0 inches | 15 |

All other design parameters were essentially the same.

For Cable A the chosen assembly activities were sequenced:

*solder, form branch, solder, encapsulate, encapsulate, clean, inspect*

For Cable B:

*solder, form branch, encapsulate, solder, encapsulate, clean, inspect*

The difference in these plans results from a processing rule of thumb that at certain thresholds, solder joints are at risk of damage due to handling, and therefore soldered connectors should encapsulated as soon as possible. Cable B has an encapsulation activity before the 2nd connector is soldered. The advantage to performing solder operations consecutively as in Cable A is that solder and encapsulation activities are performed at geographically separated areas in the factory, causing time delays in the transfer of parts. The process plans created for Cables A and B are equivalent to process plans being used in fabrication.

In addition accuracy, the CABPRO prototype proves a significant time savings in using an expert system to help engineers write travelers. The time required to input data for the test cables averaged 15 minutes and execution time averaged 7 minutes. For an experienced engineer to perform equivalent tasks in a conventional process planning scenario requires approximately 2.5 hours.

## Accomplishments

The objectives for this project as stated above were:

- Automate 60% of the engineering process planning function in support of the manufacture of multiwire cables.
- Retain the knowledge of experienced cable engineers
- Gain practical expert system development experience

The demonstration of the prototype system showed success for all three objectives. It provided proof of the concept that 60% of the process planning function could be automated. The prototype knowledge base (all rule bases) is a collection of cable fabrication techniques, heuristic knowledge from expert process engineers, and a defined methodology for cable process planning. This information is now organized and documented. Practical development experience has certainly been gained.

## Future Work

Successful demonstration of the prototype system and subsequent evaluation of architecture, problem solving methods and project goals should focus efforts on continuing CABPRO development toward a production release. The initial production release should process round-wire polyurethane cables – the same type for which the prototype is designed. This will account for 60% of the cables now in production. Future modules should add capability for other cable types including neoprene, flat-flexible, flat-molded and pigtails.

The selection and sequencing techniques as well as the basic rule base network structure are planned to be incorporated in the initial production release. One change planned in the general architecture is the addition of a database for attributes of individual components. Modules (rule bases) need to be added to implement planning at the task level. Each operation in the standard traveler library could be addressed by a set of rules to manipulate tasks for customizing standard text for a specific cables design. Task level planning will have a significant impact on time savings.

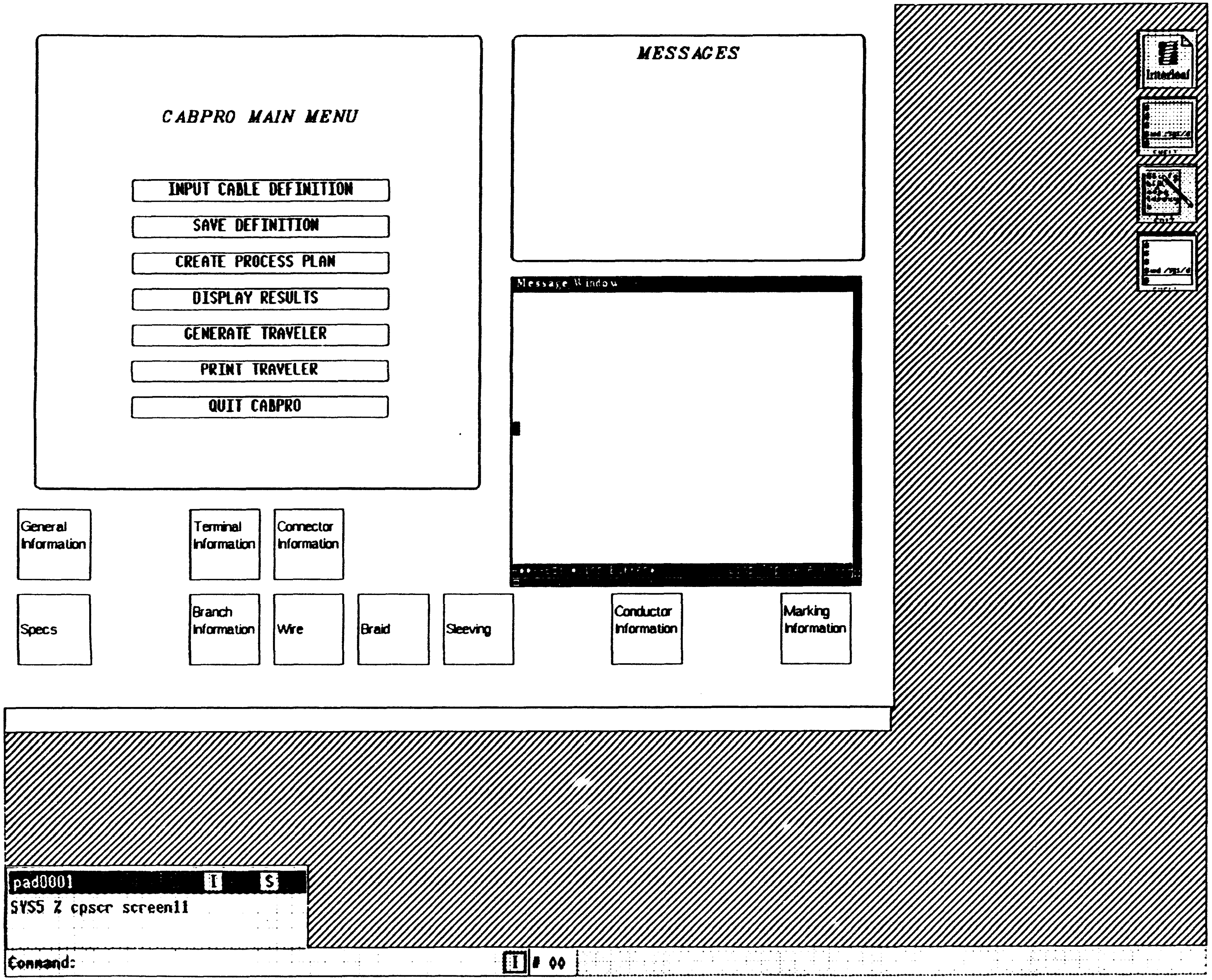
Acceptance of the system is always a concern. Therefore, a continuing effort is made to improve the user interface. Additional error checking, error recovery mechanisms, and additional help information are items for incorporation.

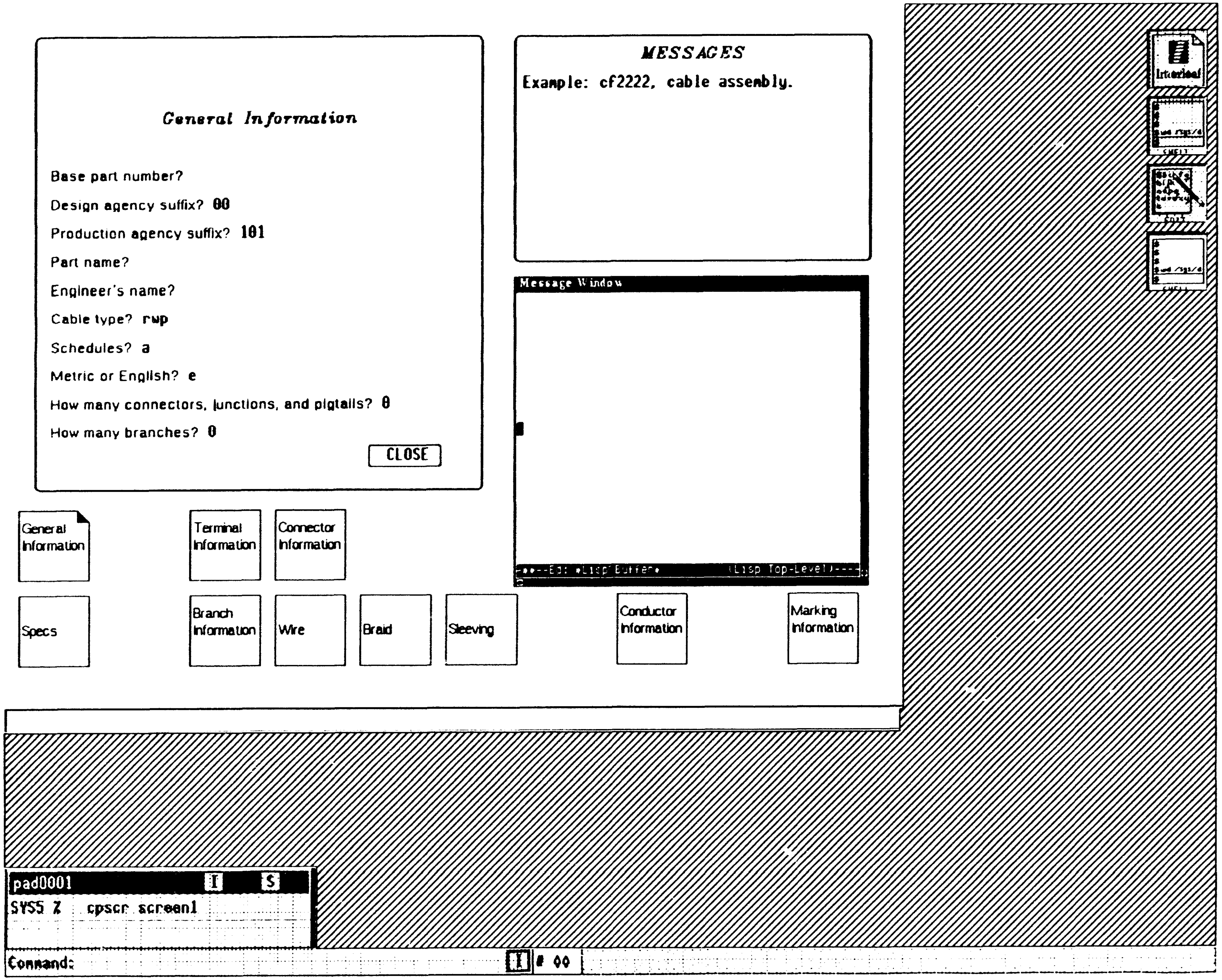
# Reference

[1]Griethuysen, J. J., *Concepts and Terminology for the Conceptual Schema and the Information Base*, ISO TC97/SC5/WG3, American National Standards Institute, New York, 1982

# Appendix A

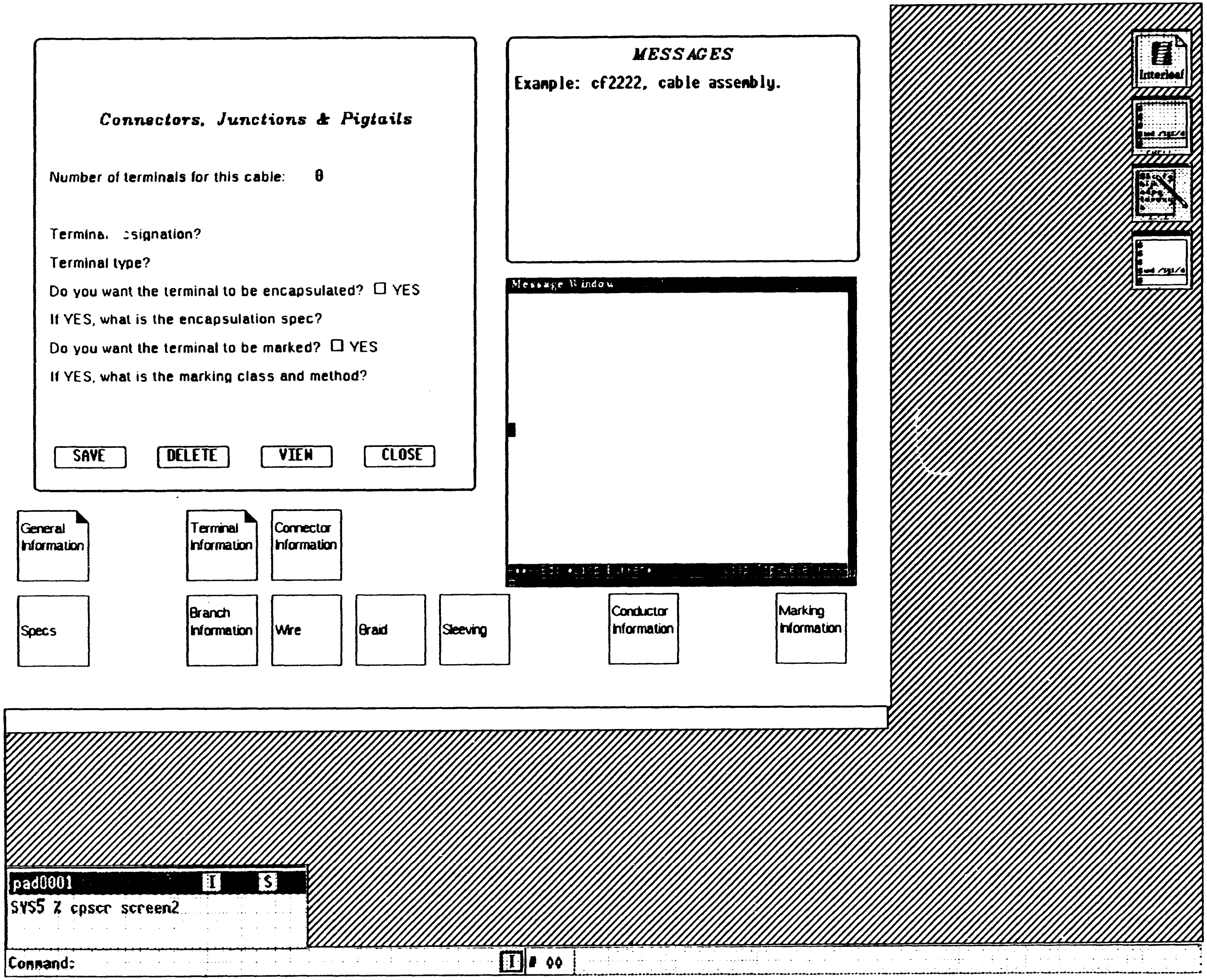
## User Interface Screens

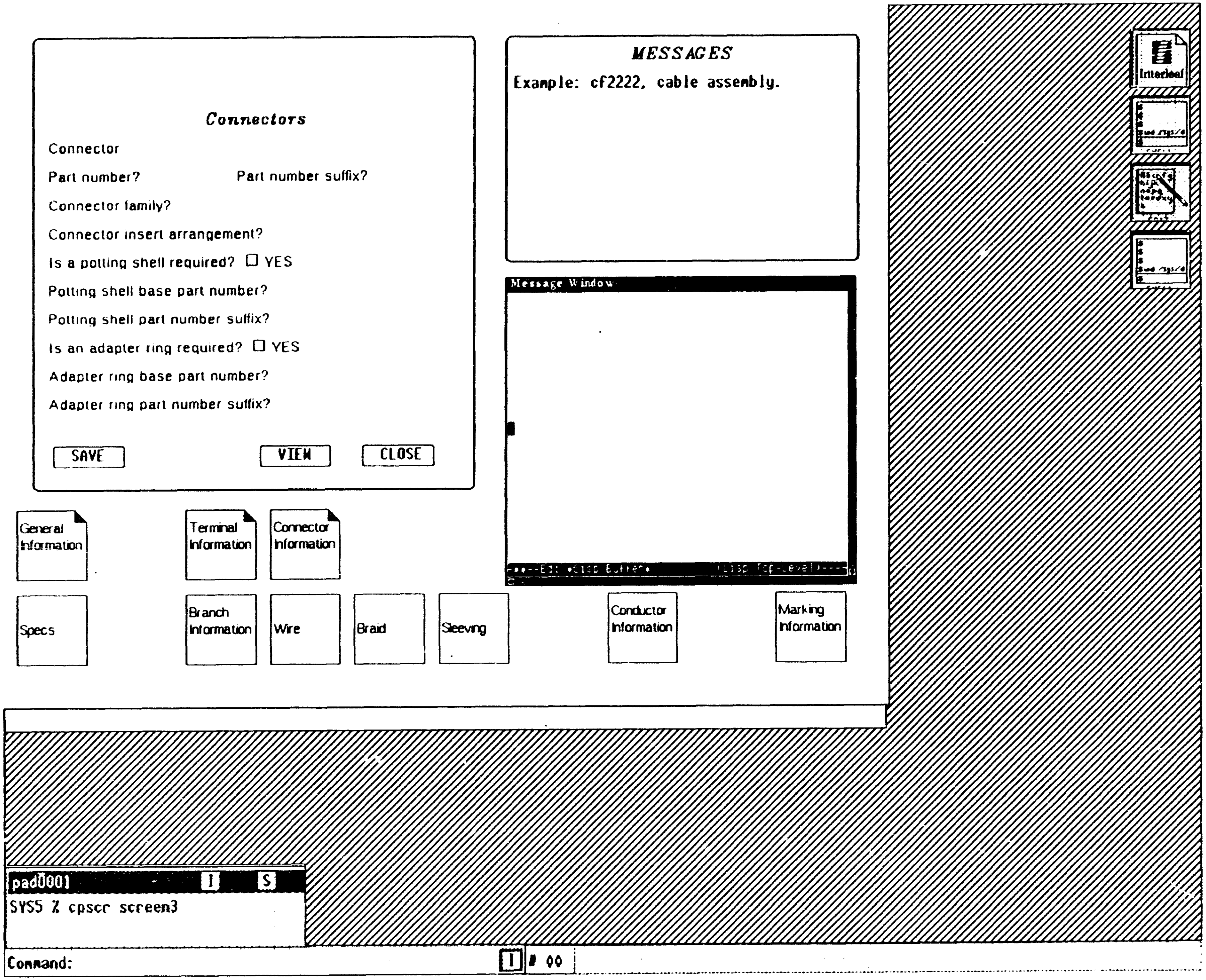

MESSAGES
CABPRO MAIN MENU
INPUT CABLE DEFINITION
SAVE DEFINITION
CREATE PROCESS PLAN
DISPLAY RESULTS
GENERATE TRAVELER
PRINT TRAVELER
QUIT CABPRO
Message Window
General Information
Terminal Information
Connector Information
Specs
Branch Information
Wire
Braid
Sleeving
Conductor Information
Marking Information
pad0001
SYS5 % cpscr screen11
Command:

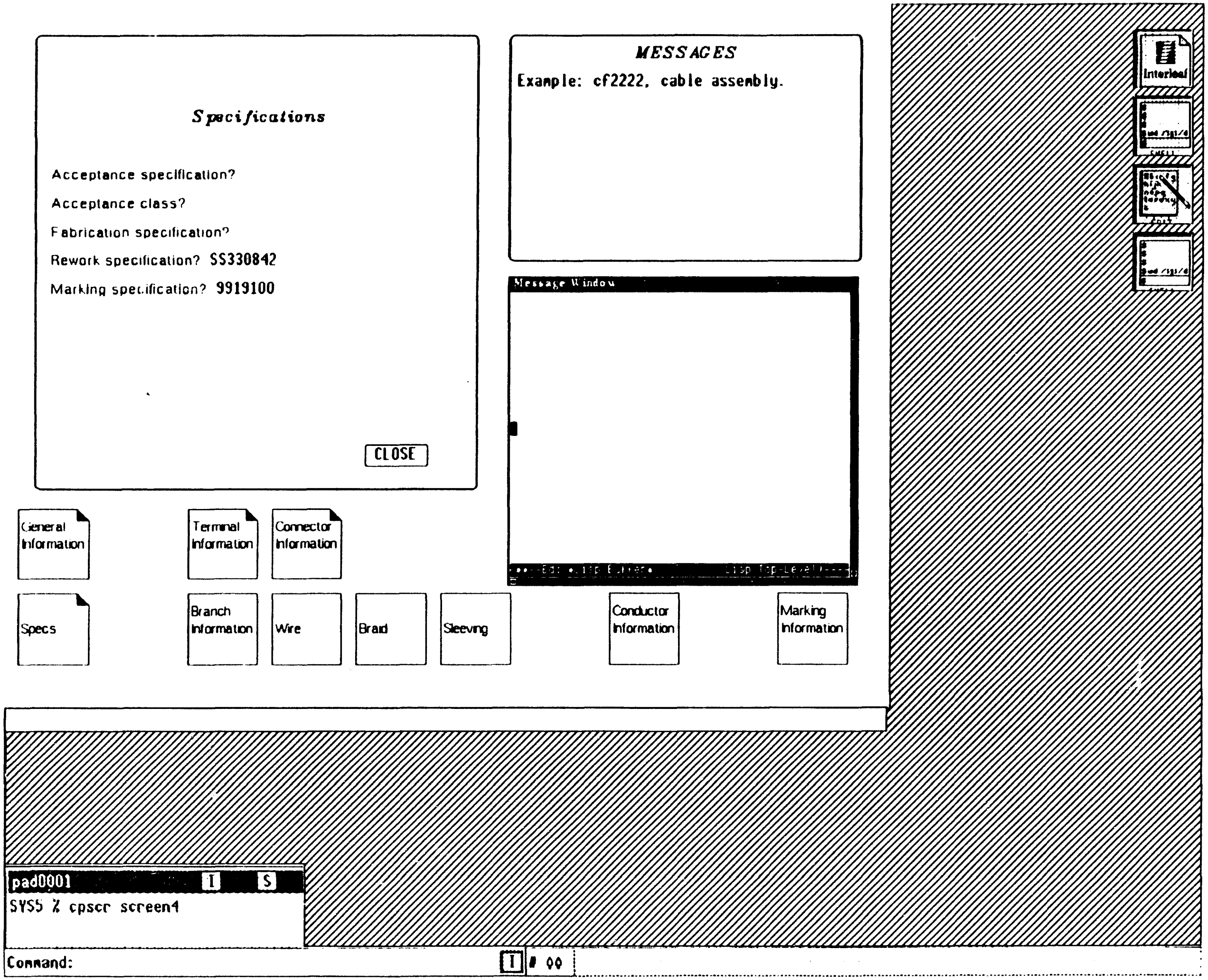


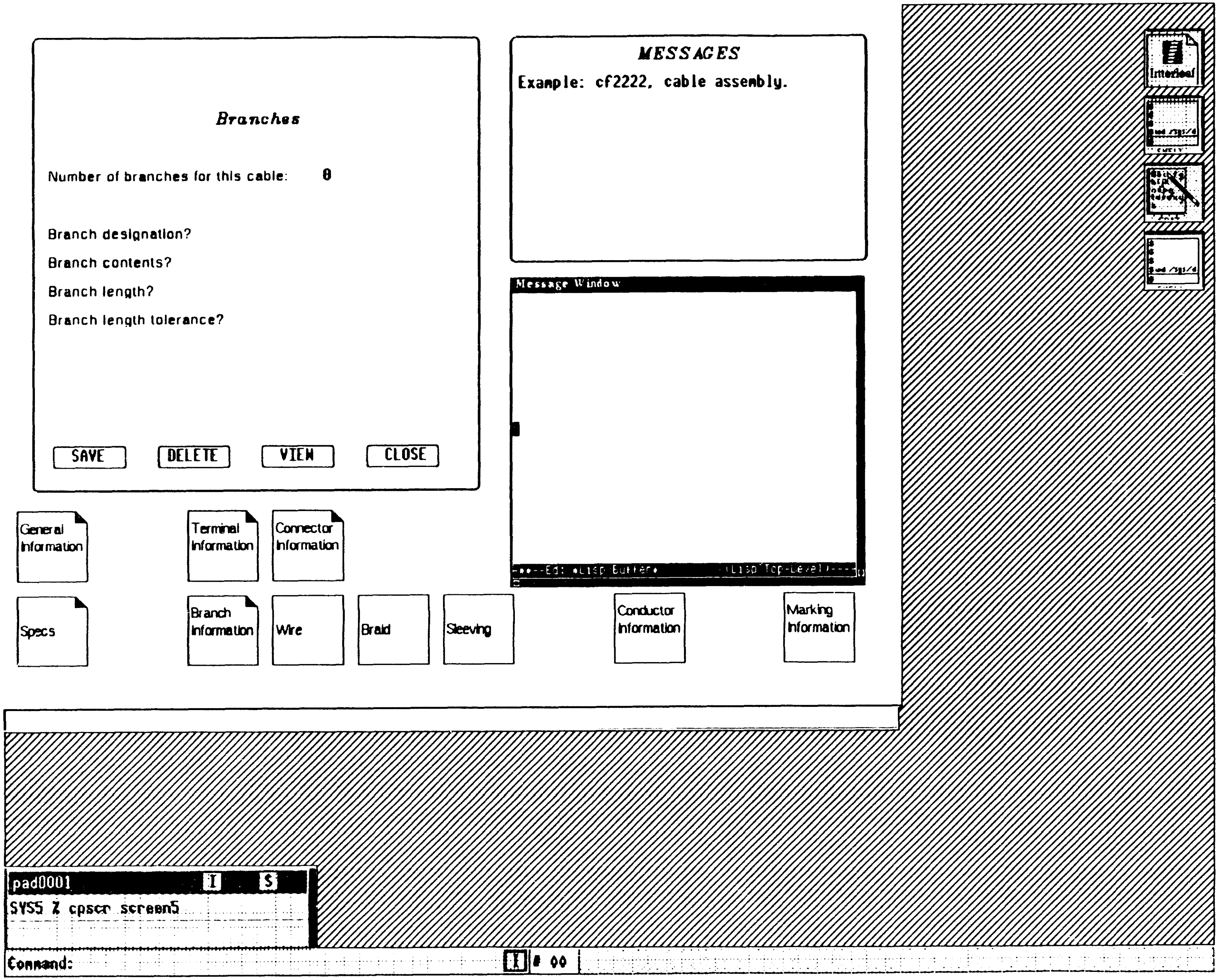

Appendix A

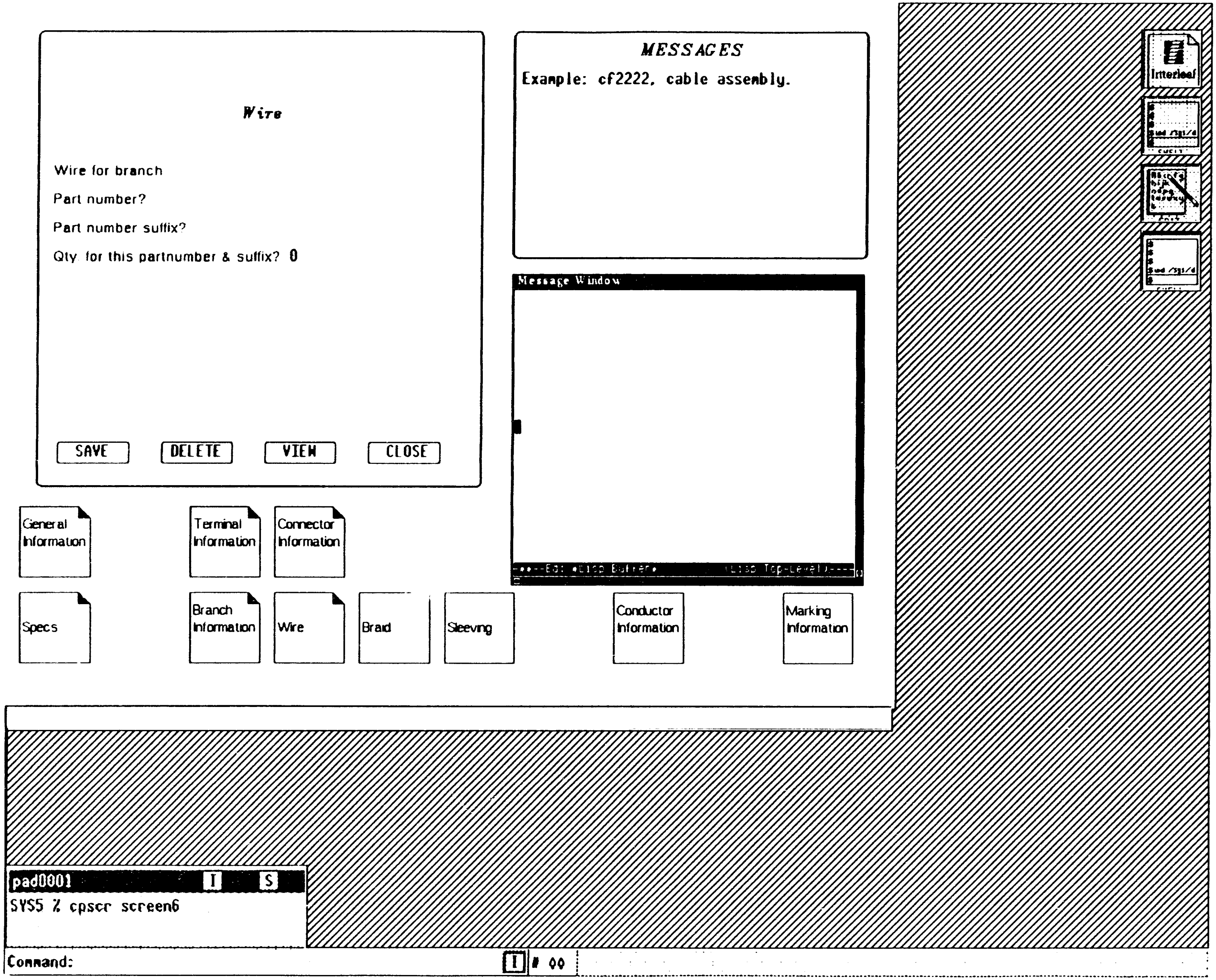
Screen A-2

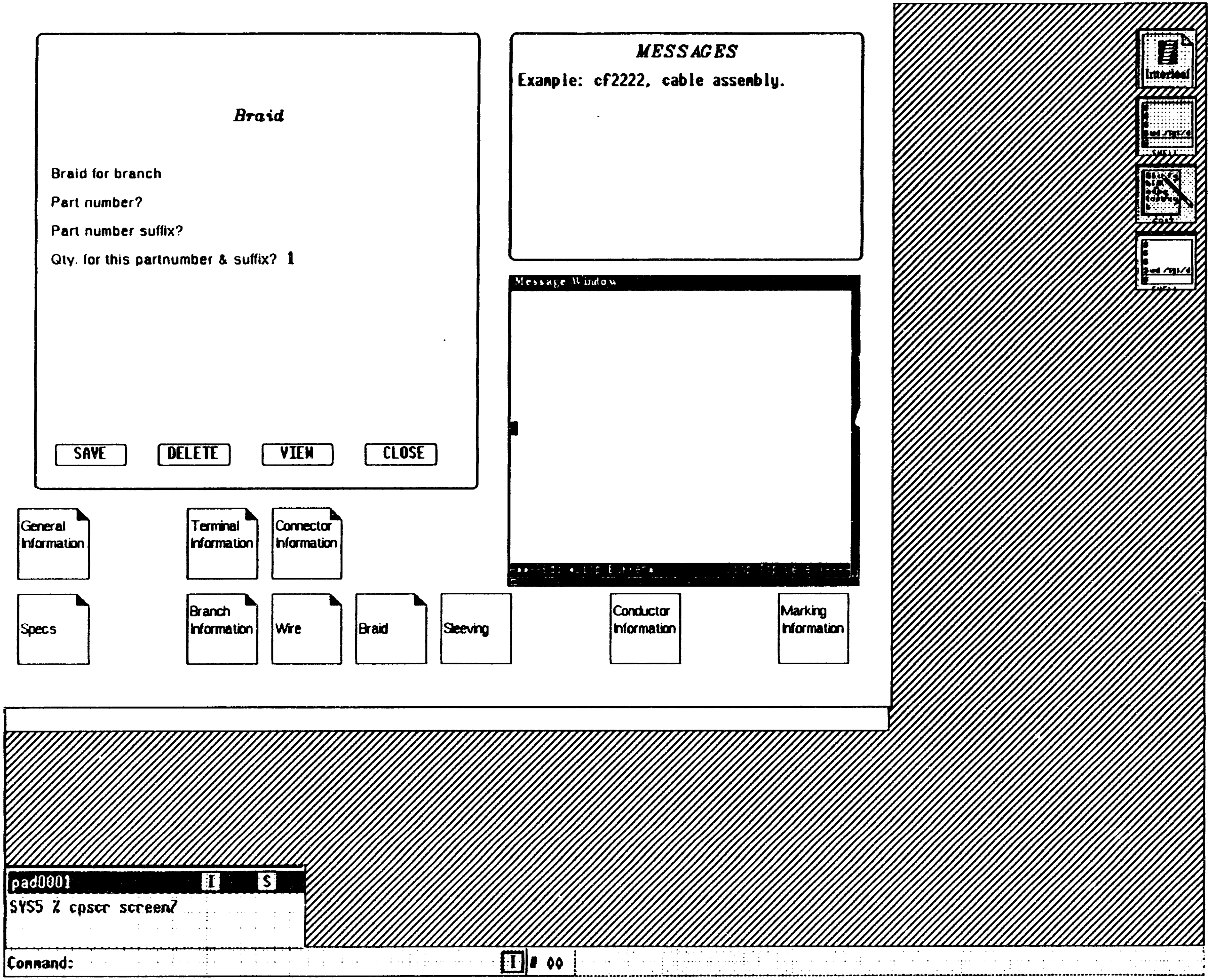

MESSAGES
Example: cf2222, cable assembly.
Connectors, Junctions & Pigtails
Number of terminals for this cable: 0
Termina. :signation?
Terminal type?
Do you want the terminal to be encapsulated? ☐ YES
If YES, what is the encapsulation spec?
Do you want the terminal to be marked? ☐ YES
If YES, what is the marking class and method?
SAVE
DELETE
VIEW
CLOSE
Message Window
General Information
Terminal Information
Connector Information
Specs
Branch Information
Wire
Braid
Sleeving
Conductor Information
Marking Information
Interleaf
pad0001
SYS5 % cpscr screen2
Command:
# 00


MESSAGES
Example: cf2222, cable assembly.
Connectors
Connector
Part number?
Part number suffix?
Connector family?
Connector insert arrangement?
Is a potting shell required? ☐ YES
Potting shell base part number?
Potting shell part number suffix?
Is an adapter ring required? ☐ YES
Adapter ring base part number?
Adapter ring part number suffix?
SAVE
VIEW
CLOSE
Message Window
Interleaf
General Information
Terminal Information
Connector Information
Specs
Branch Information
Wire
Braid
Sleeving
Conductor Information
Marking Information
pad0001
I
S
SYS5 % cpscr screen3
Command:
I
# 00

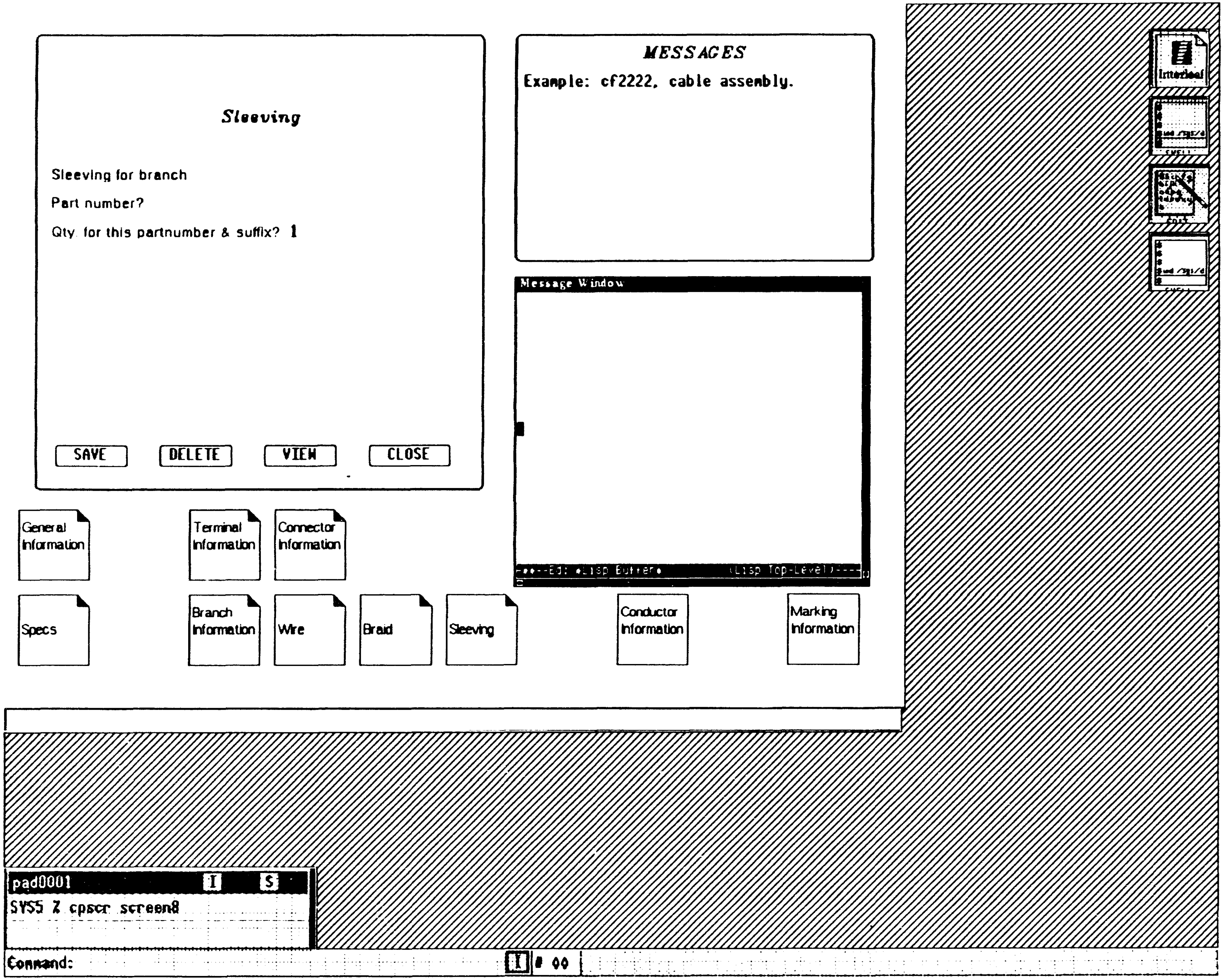

Specifications
Acceptance specification?
Acceptance class?
Fabrication specification?
Rework specification? SS330842
Marking specification? 9919100
CLOSE
MESSAGES
Example: cf2222, cable assembly.
Message Window
Interleaf
General Information
Terminal Information
Connector Information
Specs
Branch Information
Wire
Braid
Sleeving
Conductor Information
Marking Information
pad0001
I
S
SYS5 % cpscr screen4
Command:

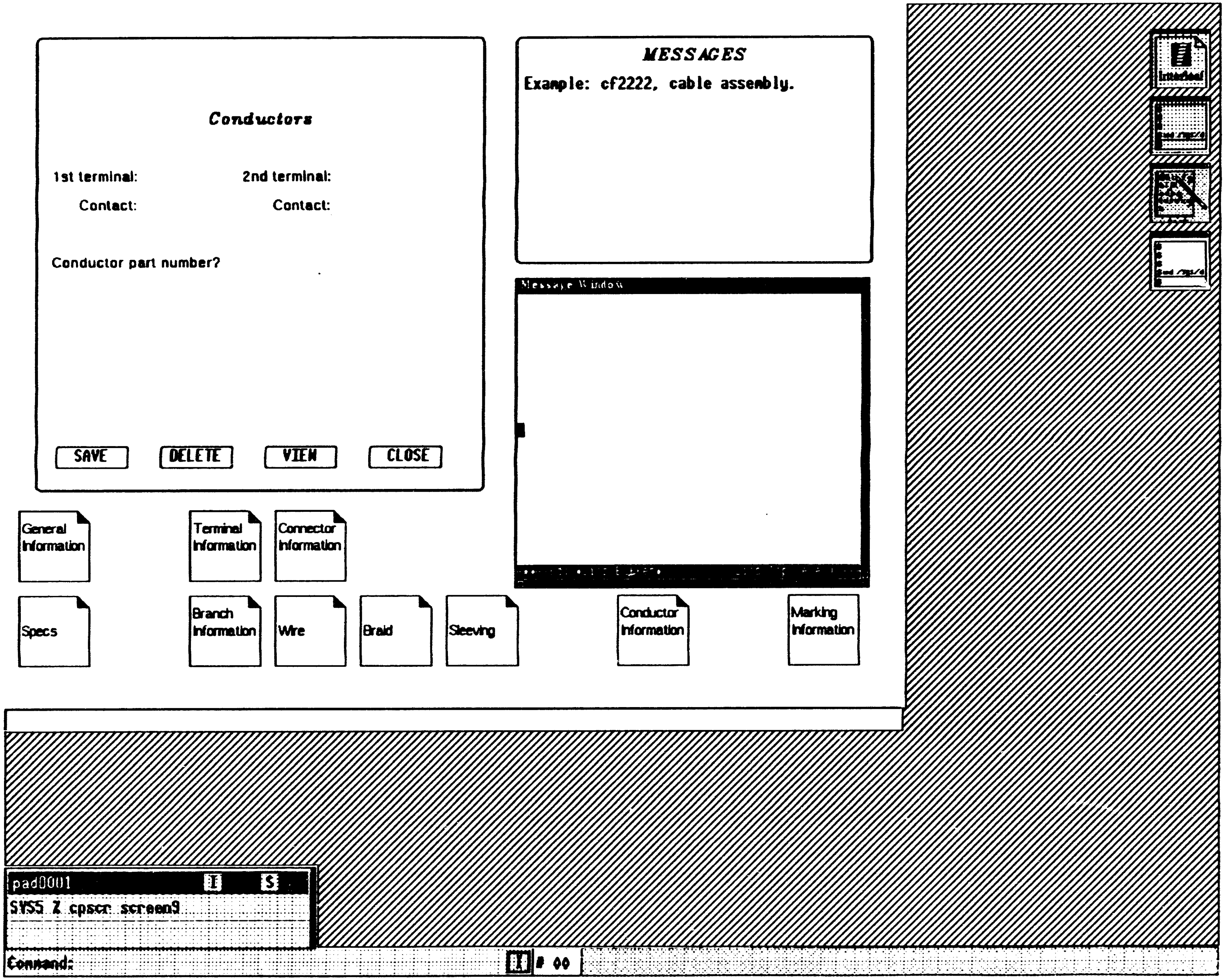

Branches
Number of branches for this cable: 0
Branch designation?
Branch contents?
Branch length?
Branch length tolerance?
SAVE
DELETE
VIEW
CLOSE
MESSAGES
Example: cf2222, cable assembly.
Message Window
General Information
Terminal Information
Connector Information
Specs
Branch Information
Wire
Braid
Sleeving
Conductor Information
Marking Information
Interleaf
pad0001
SYS5 % cpscr screen5
Command:

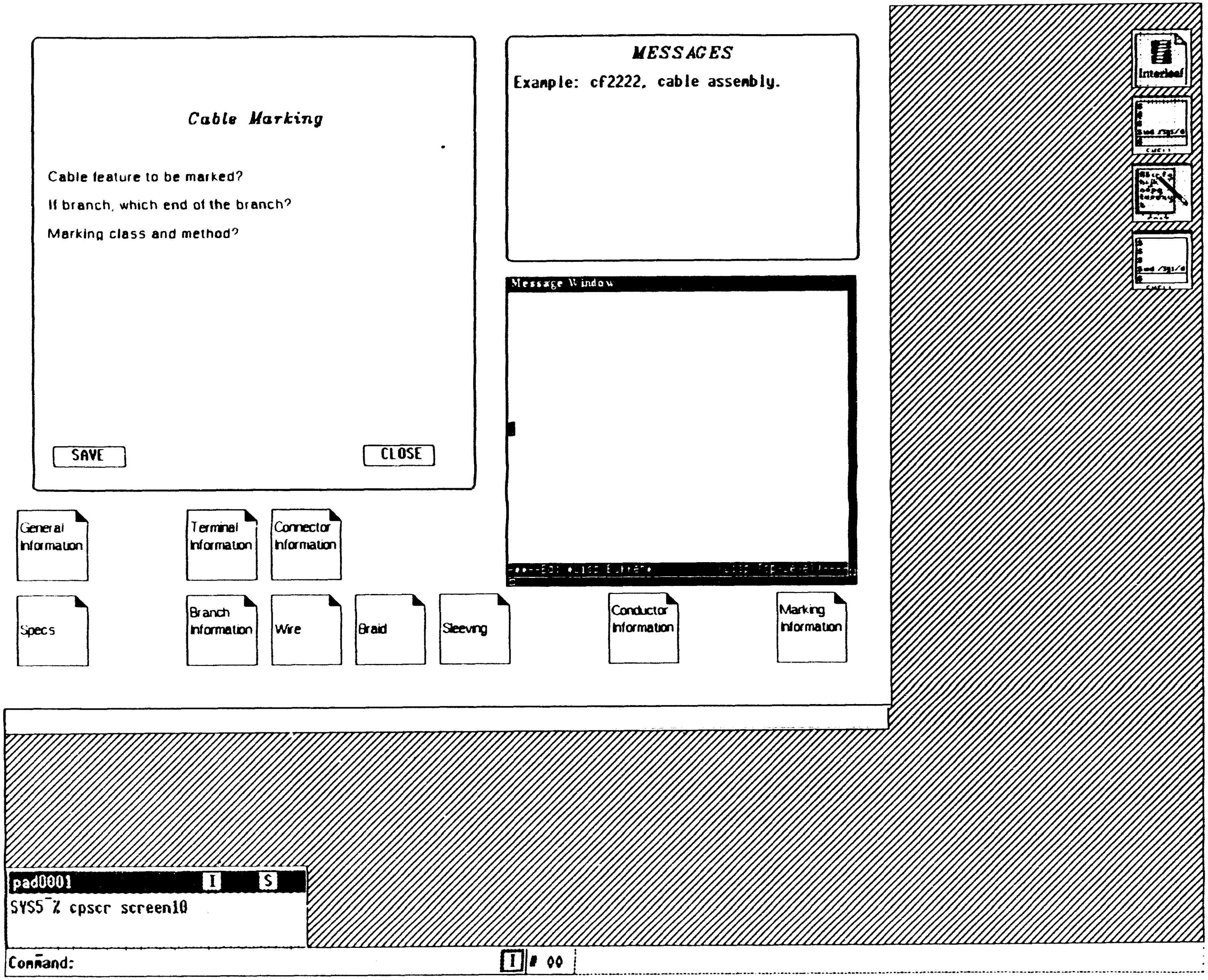

Wire
Wire for branch
Part number?
Part number suffix?
Qty for this partnumber & suffix? 0
SAVE
DELETE
VIEW
CLOSE
MESSAGES
Example: cf2222, cable assembly.
Message Window
General Information
Terminal Information
Connector Information
Specs
Branch Information
Wire
Braid
Sleeving
Conductor Information
Marking Information
Interleaf
pad0001
SYS5 % cpscr screen6
Command:

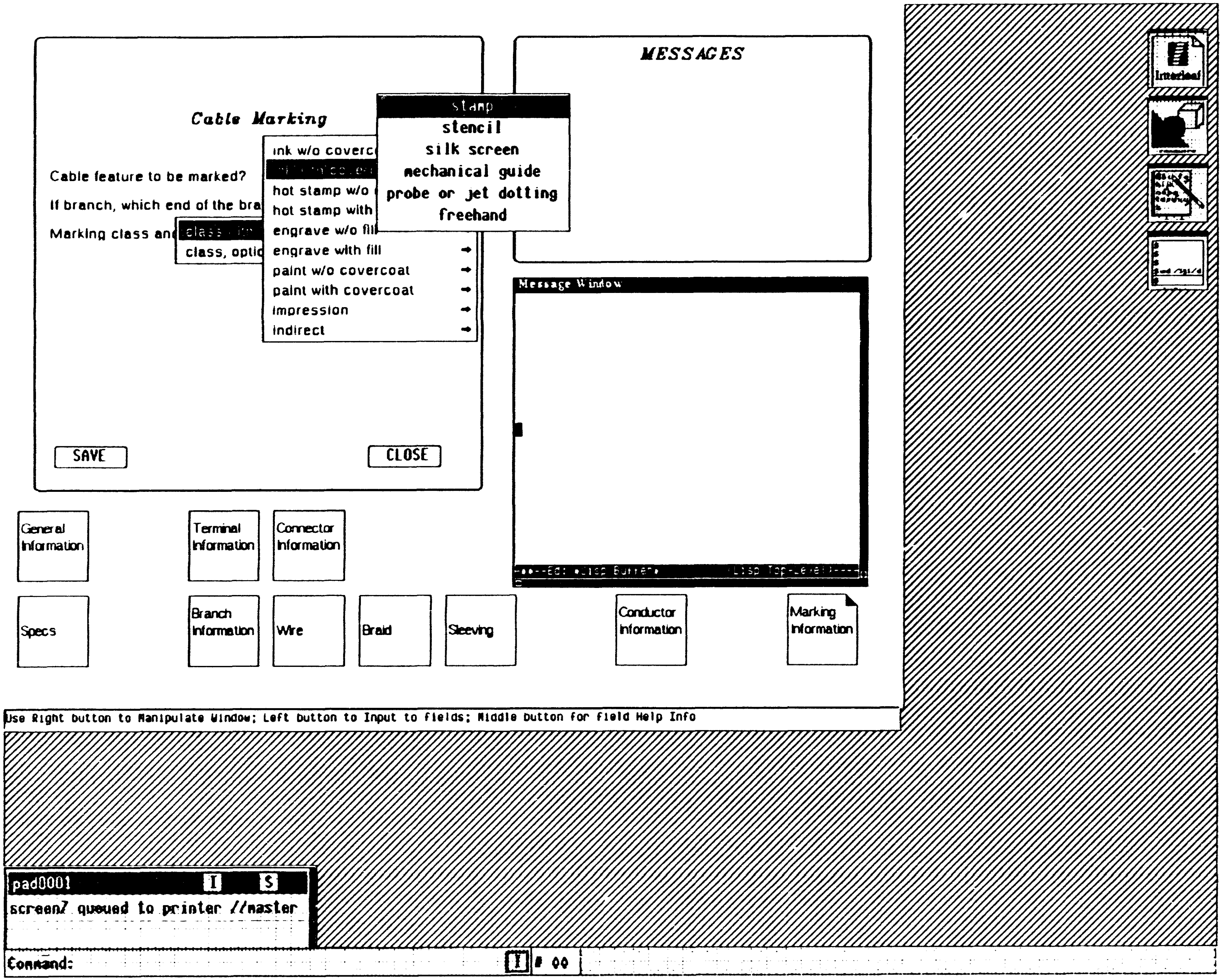


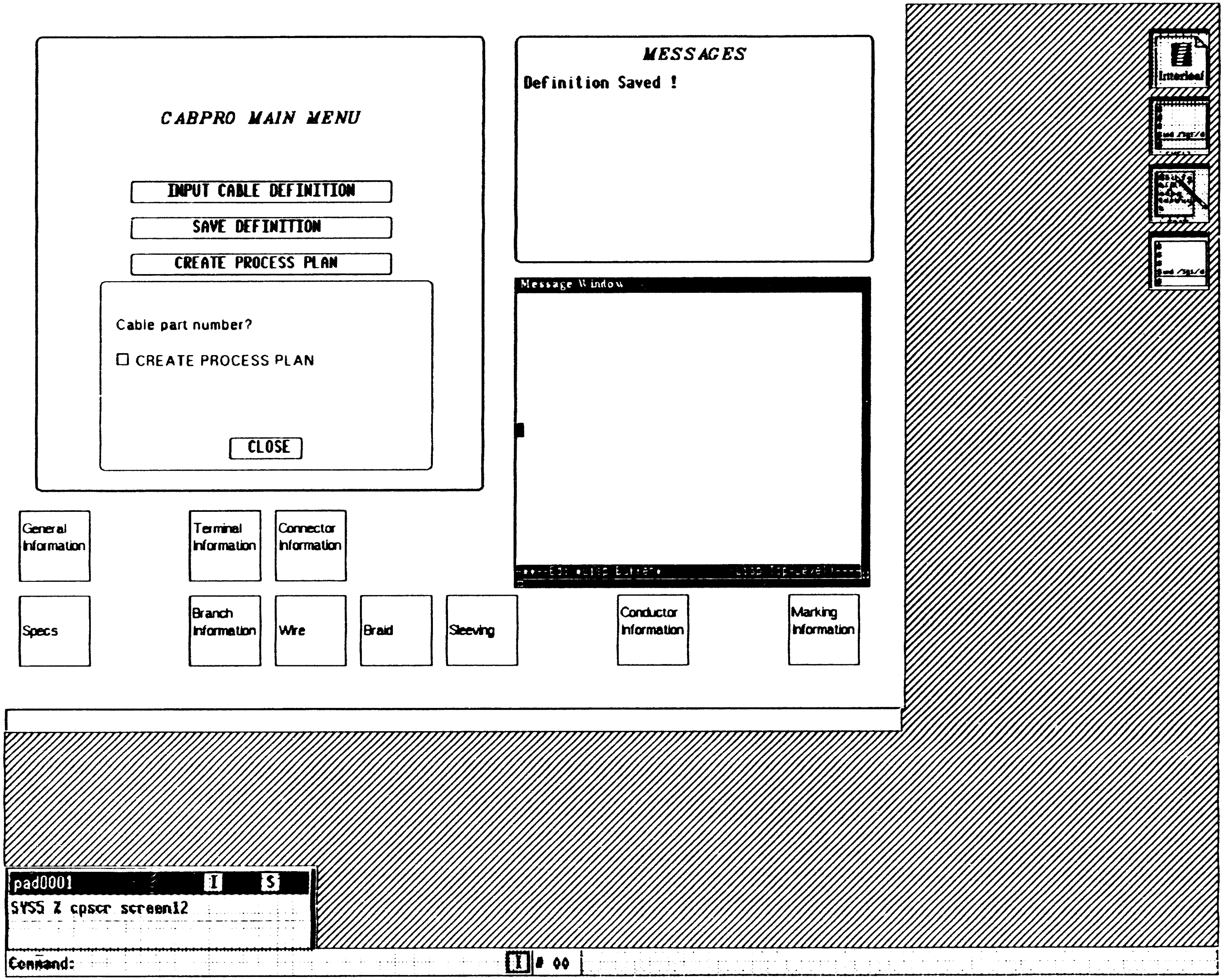
Braid
Braid for branch
Part number?
Part number suffix?
Qty. for this partnumber & suffix? 1
SAVE
DELETE
VIEW
CLOSE
MESSAGES
Example: cf2222, cable assembly.
Message Window
Interleaf
General Information
Terminal Information
Connector Information
Specs
Branch Information
Wire
Braid
Sleeving
Conductor Information
Marking Information
pad0001
SYS5 % cpscr screen7
Command:


Sleeving
Sleeving for branch
Part number?
Qty for this partnumber & suffix? 1
SAVE
DELETE
VIEW
CLOSE
MESSAGES
Example: cf2222, cable assembly.
Message Window
General Information
Terminal Information
Connector Information
Specs
Branch Information
Wire
Braid
Sleeving
Conductor Information
Marking Information
Interleaf
pad0001
SYS5 % cpscr screen8
Command:


Conductors
1st terminal:
2nd terminal:
Contact:
Contact:
Conductor part number?
SAVE
DELETE
VIEW
CLOSE
MESSAGES
Example: cf2222, cable assembly.
General Information
Terminal Information
Connector Information
Specs
Branch Information
Wire
Braid
Sleeving
Conductor Information
Marking Information
pad0001
SYS5 % cpscr screen9
Command:

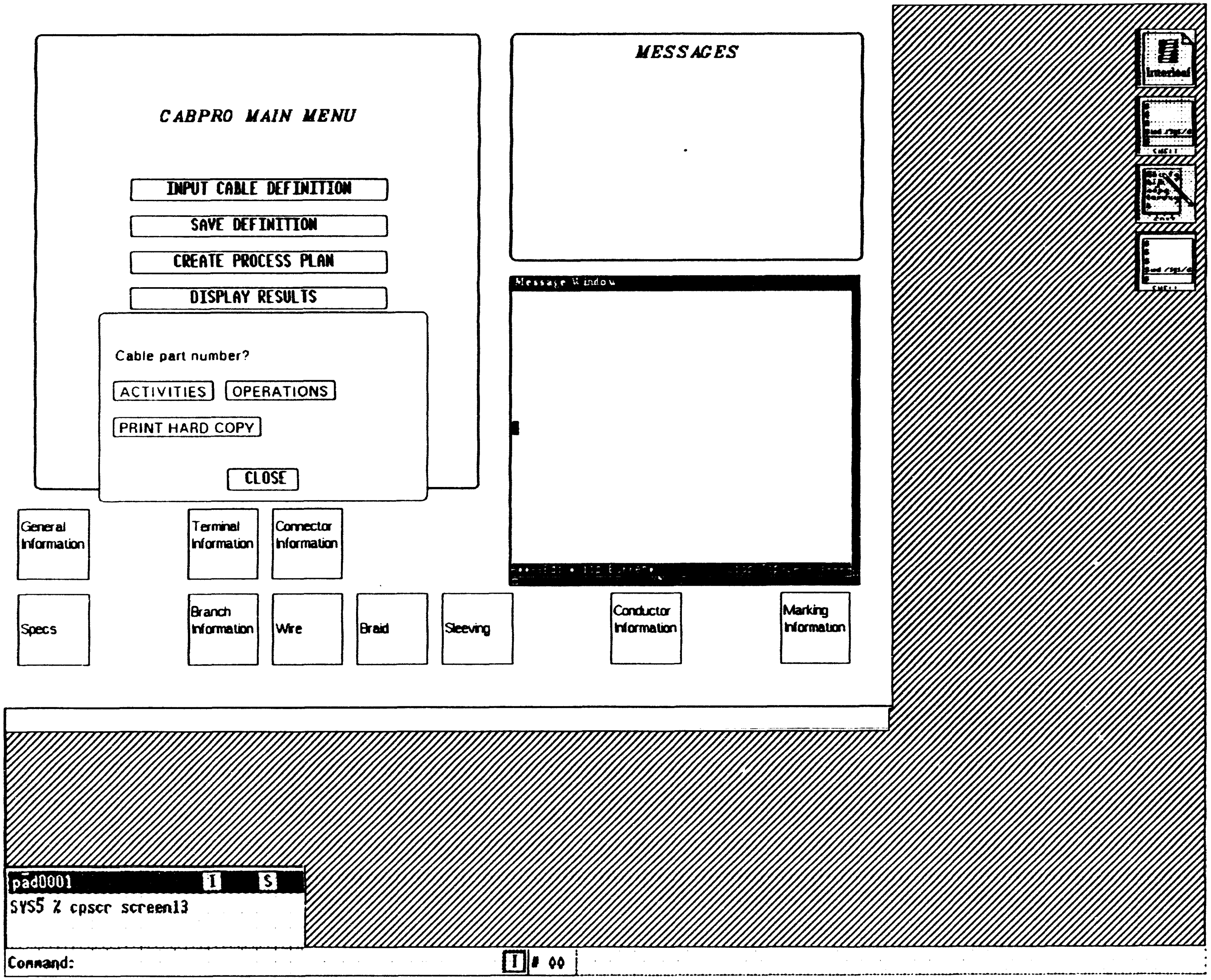

Cable Marking
Cable feature to be marked?
If branch, which end of the branch?
Marking class and method?
SAVE
CLOSE
MESSAGES
Example: cf2222, cable assembly.
Message Window
General Information
Terminal Information
Connector Information
Specs
Branch Information
Wire
Braid
Sleeving
Conductor Information
Marking Information
Interleaf
pad0001
I
S
SYS5 % cpscr screen10
Command:
# 00

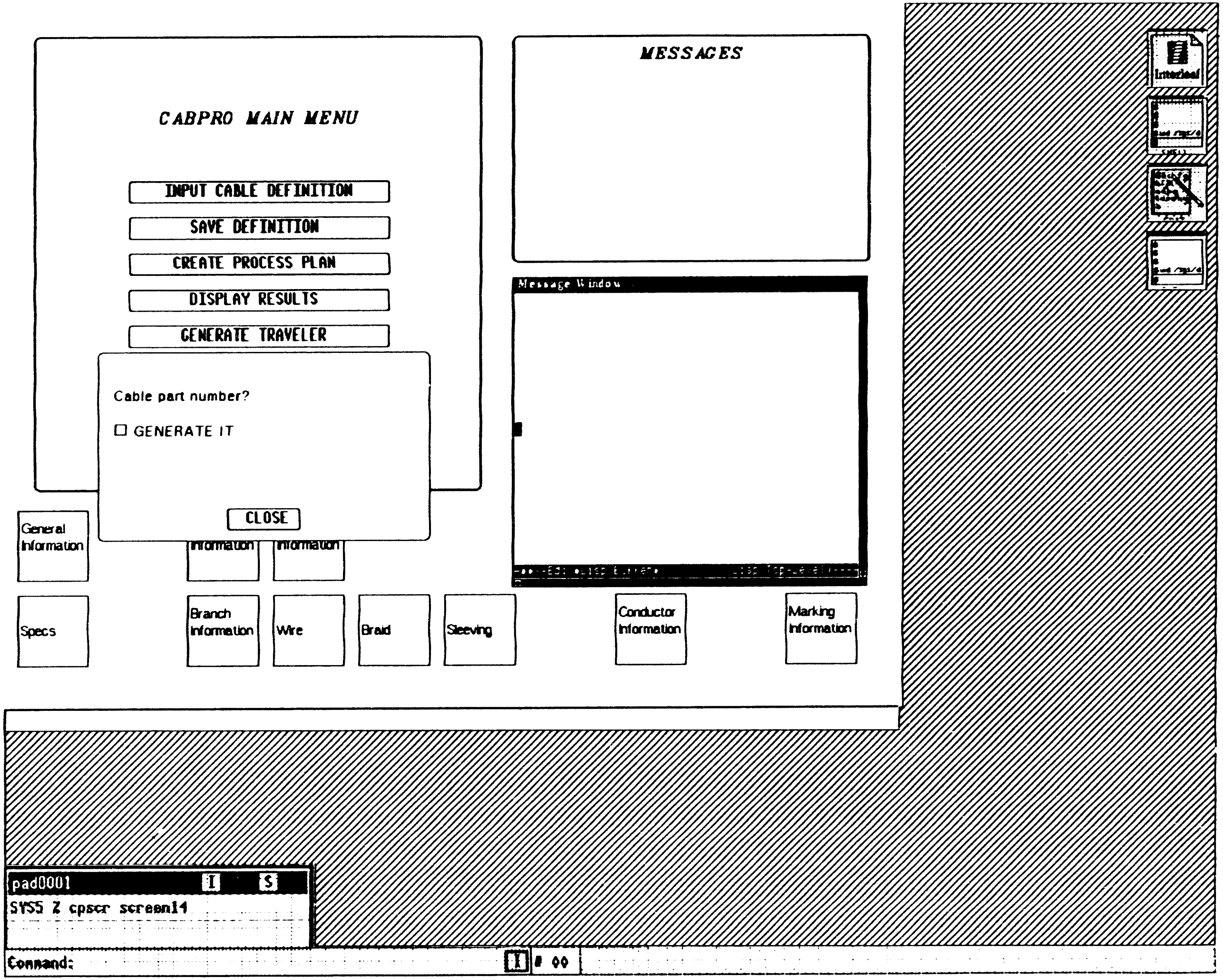

MESSAGES
Cable Marking
Cable feature to be marked?
If branch, which end of the bra
Marking class an
class, optic
ink w/o coverc
hot stamp w/o
hot stamp with
engrave w/o fil
engrave with fill
paint w/o covercoat
paint with covercoat
impression
indirect
stamp
stencil
silk screen
mechanical guide
probe or jet dotting
freehand
SAVE
CLOSE
Message Window
General Information
Terminal Information
Connector Information
Specs
Branch Information
Wire
Braid
Sleeving
Conductor Information
Marking Information
Interleaf
Use Right button to Manipulate Window; Left button to Input to fields; Middle button for field Help Info
pad0001
screen7 queued to printer //master
Command:
# 00

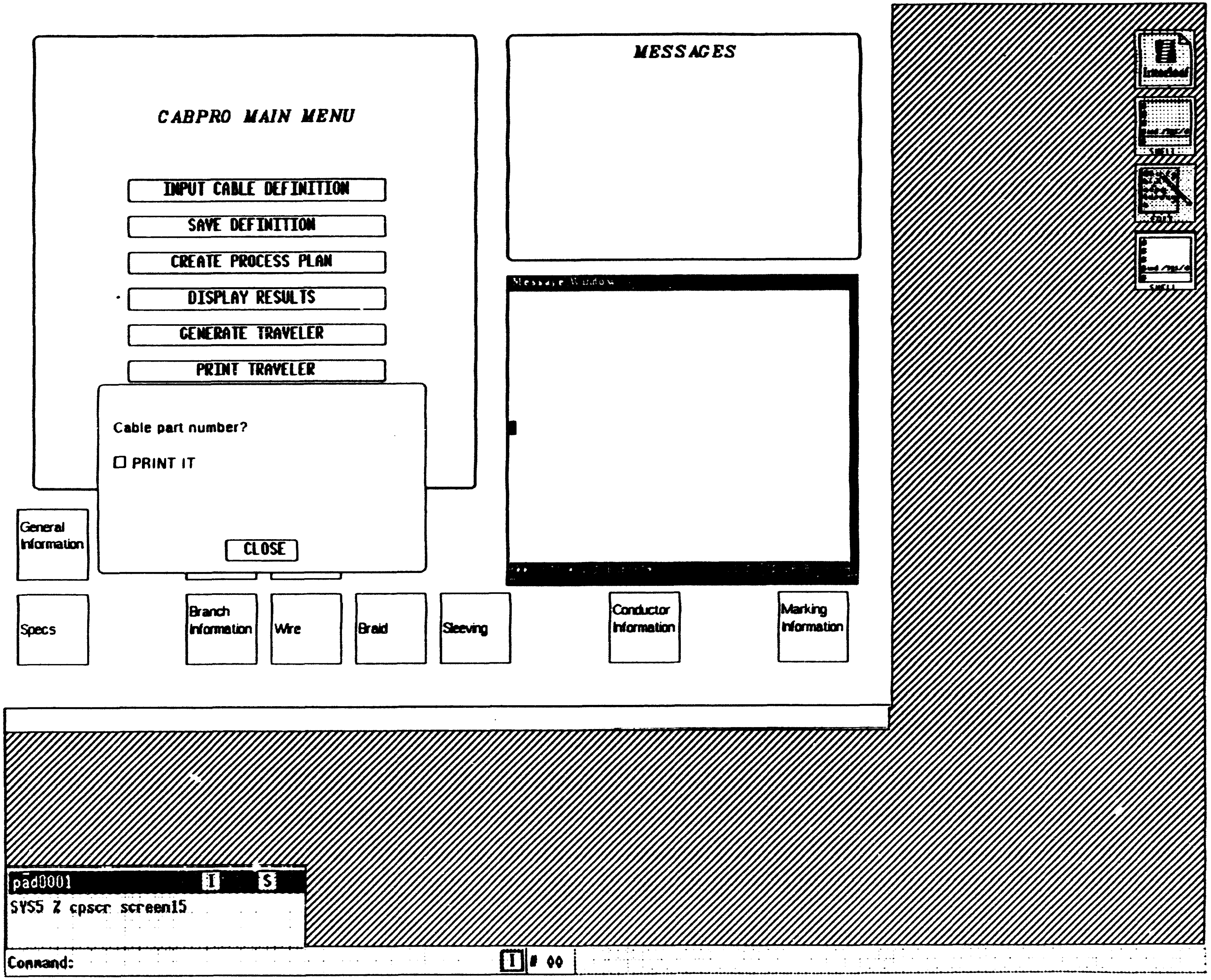



Appendix A

Screen A-13


MESSAGES
CABPRO MAIN MENU
INPUT CABLE DEFINITION
SAVE DEFINITION
CREATE PROCESS PLAN
DISPLAY RESULTS
Cable part number?
ACTIVITIES
OPERATIONS
PRINT HARD COPY
CLOSE
Message Window
General Information
Terminal Information
Connector Information
Specs
Branch Information
Wire
Braid
Sleeving
Conductor Information
Marking Information
Interleaf
pad0001
SYS5 % cpscr screen13
Command:


CABPRO MAIN MENU
INPUT CABLE DEFINITION
SAVE DEFINITION
CREATE PROCESS PLAN
DISPLAY RESULTS
GENERATE TRAVELER
Cable part number?
GENERATE IT
CLOSE
MESSAGES
Message Window
General Information
Information
Information
Specs
Branch Information
Wire
Braid
Sleeving
Conductor Information
Marking Information
Interleaf
pad0001
I
S
SYS5 Z cpscr screen14
Command:
I
# 00




Appendix A

Screen A-16

# Appendix B

## Sample Cable Description File

Sample Cable Description File (CDF)

Following is a sample Cable Description File (CDF). This file is output from the User Interface (UI) and input to the expert system portion of CABPRO. It is an actual Lisp/Flavors program created by a C program attached to the UI code. After creation this file is stored in the current users directory of files in ASCII format.

```
(setq <cable-name>
    (make-instance 'polyurethane-cable
        :specifications
            (list
                (make-instance 'specification-record
                    :fabrication <fabrication>
                    :acceptance <acceptance>
                    :rework <rework>
                    :encapsulation <encapsulation>
                    :marking <marking>
                )
            )
        :connectors
            (list
                (make-instance 'connector-record
                    :conn-desig <conn-desig>
                    :conn-type <conn-type>
                    :conn-termination <cup-pin>
                    :conn-insert-type <conn-insert-type>
                    :conn-pn <conn-pn>
                    :dust-cap-pn <dust-cap-pn>
                    :encaps-type <encaps-type>
                    :marking
                        (make-instance 'connector-marking-record
                            :conn-marking-method <conn-marking-method>
                            :conn-marking-feature-marked <conn-marking-feature>
                            :conn-marking-feature-desig <conn-marking-feature-desig>
                        )
```

```
        :encaps-method
          (list
            (make-instance 'potting-record
              :encaps-pes-no <encaps-pes-no>
              :matl-name <matl-name>
              :potting-shell-pn <potting-shell-pn>
              :adapter-ring-pn <adapter-ring-pn>
            )
          )
    )
(make-instance 'connector-record
        :conn-desig <conn-desig>
        :conn-type <conn-type>
        :conn-termination <cup-pin>
        :conn-insert-type <conn-insert-type>
        :conn-pn <conn-pn>
        :dust-cap-pn <dust-cap-pn>
        :encaps-type <encaps-type>
        :marking
          (make-instance 'connector-marking-record
            :conn-marking-method <conn-marking-method>
            :conn-marking-feature-marked <conn-marking-feature>
            :conn-marking-feature-desig <conn-marking-feature-desig>
          )
        :encaps-method
          (list
            (make-instance 'potting-record
              :encaps-pes-no <encaps-pes-no>
              :matl-name <matl-name>
              :potting-shell-pn <potting-shell-pn>
              :adapter-ring-pn <adapter-ring-pn>
            )
          )
    )
)
```

Sample Cable Description File (CDF)

```
:branches
    (list
        (list
            (make-instance 'lay-in-record
                :lay-in-type <lay-in-type>
                :lay-in-pn <lay-in-pn>
                :lay-in-pn-suffix <lay-in-pn-suffix>
                :lay-in-terminations <lay-in-terminations>
                    (list
                        (make-instance 'lay-in-term-record
                            :lay-in-term-desig <lay-in-term-desig>
                            :lay-in-term-method <lay-in-term-method>
                            :lay-in-term-locations
                                (list
                                    (make-instance 'lay-in-term-points-record
                                        :lay-cup-or-pin <lay-cup-or-pin>
                                    )
                                )
                        )
                        make-instance 'lay-in-term-record
                            :lay-in-term-desig <lay-in-term-desig>
                            :lay-in-term-method <lay-in-term-method>
                            :lay-in-term-locations
                                (list
                                    (make-instance 'lay-in-term-points-record
                                        :lay-cup-or-pin <lay-cup-or-pin>
                                    )
                                )
                        )
                    )
                :physical-route
                    (list
                        (make-instance 'route-record
                            :route-branch-desig <lay-in-route>
                        )
                    )
            )
        )
```

Sample Cable Description File (CDF)

```
(list
    (make-instance 'branch-record
        :branch-desig <branch-desig>
        :sleeve-pn <sleeve-pn>
        :branch-length <branch-length>
        :measure-type <measure-type>
        :no-of-conductors <no-of-conductors>
        :breakouts
            (list
                (make-instance 'breakout-record
                    :breakout-conn-desig <breakout-conn-desig>
                    :breakout-section <breakout-section>
                    :breakout-face <breakout-face>
                )
                (make-instance 'breakout-record
                    :breakout-conn-desig <breakout-conn-desig>
                    :breakout-section <breakout-section>
                    :breakout-face <breakout-face>
                )
            )
        :conductors
            (list
                (make-instance 'conductor-record
                    :cond-type <cond-type>
                    :cond-pn <cond-pn>
                    :cond-pn-suffix <cond-pn-suffix>
                    :cond-qty <cond-qty>
```

```
                    :cond-terminations <cond-terminations>
                        (list
                            (make-instance 'cond-term-record
                                :cond-term-desig <cond-term-desig>
                                :cond-term-method <cond-term-method >
                                :cond-term-locations
                                    (list
                                        (make-instance 'cond-term-points-record
                                            :cond-cup-or-pin <cond-cup-or-pin>
                                        )
                                        (make-instance 'cond-term-points-record
                                            :cond-cup-or-pin <cond-cup-or-pin>
                                        )
                                    )
                            )
                            make-instance 'cond-term-record
                                :cond-term-desig <cond-term-desig>
                                :cond-term-method <cond-term-method >
                                :cond-term-locations
                                    (list
                                        (make-instance 'cond-term-points-record
                                            :cond-cup-or-pin <cond-cup-or-pin>
                                        )
                                        (make-instance 'cond-term-points-record
                                            :cond-cup-or-pin <cond-cup-or-pin>
                                        )
                                    )
                            )
                        )
                    )
                )
            )
        )
    )
  )
)
```

## Sample Cable Description File (CDF)

```
(setq class
    (make-instance
        :part-num <part-nu>
        :part-name <part-name>
        :engineer-name <engineer-name>
        :dimension-scheme <dimension-scheme>
        :schedules <schedules>
        :cable-type <cable-type>
    )
)


(setq cable-identification
    (make-instance 'cable-marking-record
        :cable-feature-marked <cable-feature-marked>
        :cable-feature-desig <cable-feature-desig>
        :cable-marking-method <cable-marking-method>
        :da-pn <da-pn
        :part-name <part-name>
    )
)
```

# Appendix C

## Sample Operation Level Plan

Sample Operation Level Plan

| | |
|---|---|
| 00001 | General-Instructions-For-D55 |
| 00101 | Wire-Cut-and-Strip |
| 00311 | Sleeve-Cut-Hand |
| 02301 | Dip-Tin-Wire-Ends |
| 03101 | Cut-Braid |
| 03121 | Clean-Braid |
| 02501 | Mark-Poly-Sleeve |
| 02531 | Cure-Covercoat |
| 02701 | Mark-Potting-Shell |
| 02703 | Packaged-Marked-Potting-Shells |
| 10301 | Solder-Wires |
| 10303 | Spray-Clean-Trichlo |
| 10501 | Separate-Wires |
| 10503 | Twist-Wires |
| 10701 | Install-Braid |
| 10911 | Install-Poly-Sleeve |
| 10503 | Twist-Wires |
| 10911 | Install-Poly-Sleeve |
| 11701 | Test-Check-Non-LAC |
| 11901 | Inspect-Solder-Joints |
| 00001 | General-Instructions-For-D65 |
| 13101 | Adiprene-Prepare |
| 13102 | Adiprene-Preheat |
| 13103 | Adiprene-Pot |
| 13105 | Adiprene-Cure |
| 13107 | Adiprene-Cool |
| 13109 | Adiprene-Remove |
| 13111 | Adiprene-Clean |
| 13121 | Adiprene-Waybill |
| 10301 | Solder-Wires |
| 10303 | Spray-Clean-Trichlo |
| 10101 | Clean-Parts |
| 10301 | Solder-Wires |
| 10303 | Spray-Clean-Trichlo |
| 11701 | Test-Check-Non-LAC |
| 11901 | Inspect-Solder-Joints |
| 13101 | Adiprene-Prepare |
| 13102 | Adiprene-Preheat |
| 13103 | Adiprene-Pot |
| 13105 | Adiprene-Cure |
| 13107 | Adiprene-Cool |

| | |
|---|---|
| 13109 | Adiprene-Remove |
| 13111 | Adiprene-Clean |
| 13121 | Adiprene-Waybill |
| 19503 | Final-Hot-Stamp |
| 19505 | Cure-Covercoat |
| 19513 | Clean-Lube-Conn |
| 19701 | Final-Inspect |