AN ABL SOFTWARE ENVIRONMENT FOR A MINI-COMPUTER

Murray N. Kronick

A Thesis

in

The Department

of

Computer Science

Presented in partial fulfillment of the requirements
for the degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada.

March, 1983.

ABSTRACT

AN ABL SOFTWARE ENVIRONMENT FOR A MINI-COMPUTER

Murray N. Kronick

The development of a software environment supporting ABL (an Alternative Based Language) as an important tool within the software development process is described.

This ABL philosophy was originally developed by Dr. W. Jaworski at Concordia University [HINT81, JAWO81].

The ABL approach places an emphasis on structured design techniques, modularization, and a step-wise refinement methodology. This work presents the design and implementation of an interactive editor and an interactive interpreter for this concept.

The application of this methodology and the software environment is illustrated through the design of a computer-based automatic dialing system.

CR Categories: D.2 Software Engineering
D.3.3 Language Constructs
J.3 Life and Medical Sciences - Medical Information Systems
K.6.3 Software Management

ACKNOWLEDGEMENTS

I wish to express my deepest gratitude to my thesis supervisor, Dr. Wojciech M. Jaworski, for his direction and support throughout the project.

My thanks are also extended to Tyme Systems Ltd., for allowing me the unrestricted use of their facilities.

Finally, my sincerest appreciation to my wife, Judy Field, for her unending encouragement throughout the duration of the project.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

I. INTRODUCTION

"To learn to dissociate our reasoning from underlying computational models, and to get rid of our operational models, and to get rid of our operational thinking habits, that is what I regard as computing science's major task". [DIJK79].

Scope and Purpose. Software design and the ensuing programming effort have traditionally been carried out in environments distinct from one another.

There has been a decided lack of complete formal notations for the specification of a design. Conventional flowcharts become too unwieldy for a non-trivial system; decision tables often do not give the reader a global appreciation of the problem at hand.

On the other hand, there is an overabundance of programming languages currently in existence. Indeed, there has been a proliferation of new languages, versions, supersets, and special-purpose languages, such that committees charged with maintaining programming standards have been unable to keep up [ACM81].

It is the frequent lack of communication between the design phase and the programming/implementation phase that calls for the need of a complete software environment.

This environment must support the Software Development Process.

The Software Development Process should be considered a continuum, and not just a series of discrete events. The concensus at the National Bureau of Standards Programming Environment Workshop [ACM81] was that "most participants viewed software development as a series of refinements of objects from the general requirement specification to the concrete realization of the program. The important research lies with discovering the transformations and increasingly automating application of the transformation".

Responding to these needs, this thesis has defined and implemented a software environment in order to satisfy the following objectives:

1) To promote structured techniques and modularization throughout the software development process. This allows maximum benefit of using the step-wise refinement methodology.

2) To escape from the conventional, sequential approach to software development, and to promote programming by logical segments.

3) To develop an operational, "user friendly" system

on a mini-computer; that demands little technical skill and no conventional programming expertise to understand and use.

4) To provide a software tool that maintains the components of a system (or program) separately from the flow connecting those components.

Basis of a Software Environment. A Software Environment can be defined as "the methods, techniques and tools which are used during the development of a software system" [ACM81]. Figure 1 is a model of a typical software environment.

The diagram illustrates how the user (designer, programmer or end-user) interfaces primarily with those software tools and programs available to him/her. These tools communicate with the database management system.

The database management system (DBMS), regardless of its level of sophistication, is meant to support high-level data- and file- manipulation commands, which in turn use primitives available in the supervisor. At this stage, commands become implementation dependent.

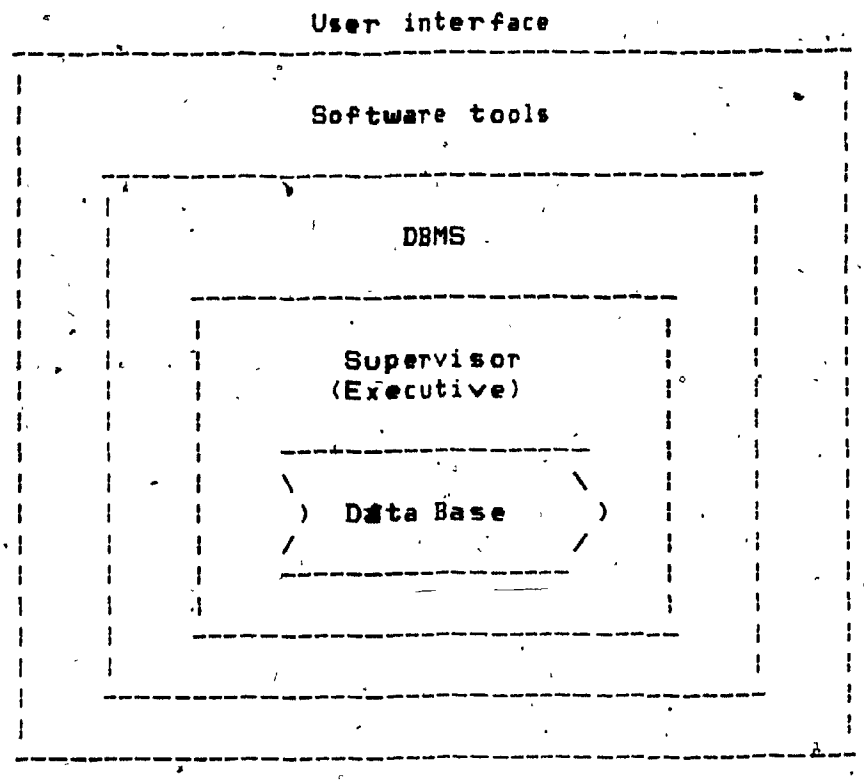

Figure 1. Model of a Software Environment

The supervisor itself may perform actual operations on the data in the database, along with device control instructions in the machine language.

This approach, although complex, allows for very high-level commands to be invoked by the user, and the underlying levels to remain transparent to that user.

The Selection of ABL. In accordance with the stated objectives, it was endeavoured to find a new software tool with an appropriate approach.

A design / programming concept known as ABL (an Alternative Based Language), developed by Dr. W. Jaworski, and currently undergoing research at Concordia University, [HINT81, HORV82, JAWO81, LEBE82a, LINA82, MORG81], was selected and further developed to serve as the kernel for the software environment.

It must be emphasized that the invention here was not to invent another programming language. It was to define a software environment, in order to provide automated assistance at each stage of the software development life cycle. This is intended to increase the potential of each computer professional to improve both the quality and the quantity of software produced.

II. THE SOFTWARE DEVELOPMENT PROCESS

Much research has been done in enumerating the steps of the software development process, their boundaries and interactions [BOEH78, MILL79, OSTE80, PETE78].

The steps are listed in Figure 2, showing the sequence of events and their principal feedback loops.

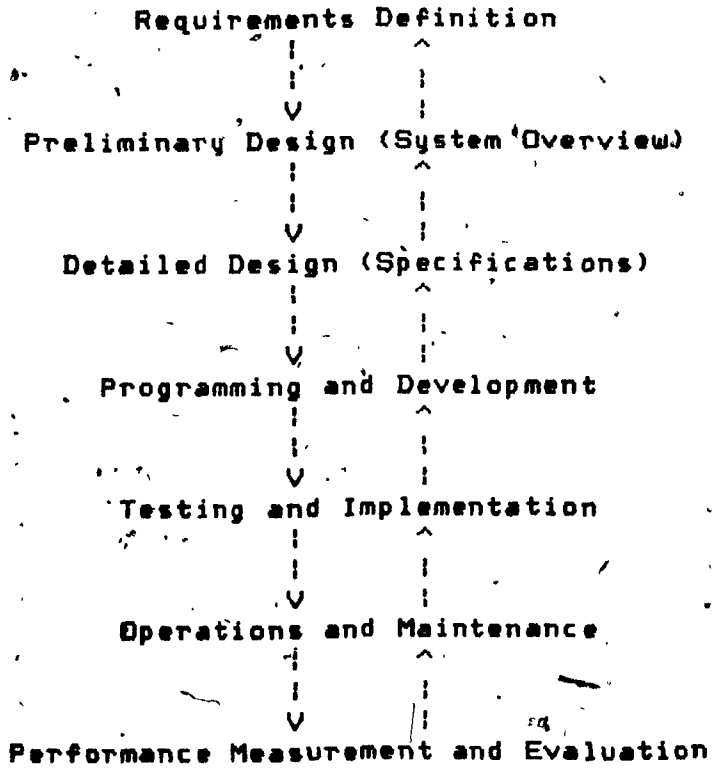

Figure 2. The Stages of Software Development

This figure illustrates software development as a sequential process, each step being reviewed and approved prior to proceeding to the next level.

Each of these seven steps is explained below:

Requirements Definition. The very first step in problem solving is to succinctly define the problem. That is, to establish the problem requirements and objectives, and form a mental model of the system. There is usually little written material at this stage, and what exists is presented in the form of a narrative. The functional and performance characteristics of the system are described, but should avoid the suggestion of a possible algorithm or solution to the problem.

An overall cost-justification based on a first approximation of time and dollar estimates is typically done at this stage, as well.

Preliminary Design. The formulation of plausible strategies to satisfy the system requirements are developed now. This step is performed at a very high level of abstraction, for later decomposition of the problem into distinct modules including all their interactions. In addition, the initial database requirements and structures are determined.

The software environment reports can be used as an

integral part of the System Overview document, to represent control flows in addition to overall data flows. This document is in non-technical terms, and can be easily reviewed by the prospective user for errors and omissions. The designer/analyst should not advance to the next step until the user gives approval to the System Overview.

The general software requirements, indicating the possible types of software tools needed in the upcoming stages of the project, are also determined at this stage.

Detailed Design. Refinement at this level is accomplished through analysis of each module, changing the orientation from problem comprehension to that of software construction planning. Contraction / expansion in the level of abstraction must be facilitated by the software tool and general flow can be confirmed using module "walk-through" and prototyping techniques.

Complex problems are then decomposed into modules, which are groups of actions to be executed in discrete sections and under mutually exclusive conditions.

Once these modules have been defined, they are synthesized into a detailed specifications document, for further approval by the end user prior to proceeding to the software production step. This decision will also be based upon the estimates of the magnitude of the programming

and implementation effort involved in the project.

One other task must be accomplished at this point - to prepare a validation test plan for comparison of future test results with recently defined system objectives and specifications. These should be on a high level, testing both assertions and boundary conditions within modules.

Programming and Development. Each module and each instruction within the module must eventually be broken down into something that the host computer can operate on. This programming and development function consists of successive stages of refinement of earlier models until the final (lowest) level is reached. It is here that actions, previously written as natural language descriptions, must be translated into executable code in the appropriate implementation language.

As a result of this approach, the programmer should not have to make assumptions based upon imperfect information: the higher levels of abstraction, if complete, contain all necessary analysis and design criteria.

Implicit in the completion of this phase is the assumption that all modules have been fully implemented and debugged, that standards have been followed, and that the accompanying documentation and user manuals are complete.

Testing and Implementation. The nature of the testing phase is to ensure that the "finished product" corresponds to the system's original functional and performance characteristics, and the specifications. These tests must be performed on a module level, as well as on the system level. The latter ensures that the integration between modules is correct.

The objective of "bug-free" software is an elusive one, but one that must be pursued. There are many methods of determining the robustness, reliability and completeness of software, but they must all adhere to this philosophy: "Testing is the process of executing a program with the intent of finding errors" [MYER80].

Software Quality Assurance Committees have been established with the mandate of producing standards, recommending development and testing tools and methodologies, improving documentation quality, and developing implementation strategies [GUST82].

The multiplicity of testing and verification tools in a software environment facilitate the Quality Assurance Inspector's job.

The implementation strategy must include budgets for user training, initial system loading and startup, system configuration and conversion, and a parallel run.

Operations and Maintenance. The most important ingredient for the long-term success of a software project is good documentation. The users should be supplied with a copy of the System Overview, and the appropriate sections of the detailed specifications, according to need. These, of course, were produced with the aid of the software environment.

These documents will give operators and users a global appreciation of the system's capabilities and it's boundaries, as well as details of individual application modules.

The software environment can also be used as a documentation tool to describe actual operation sequences and conditions. Conventionally, these procedures have been written in either unstructured natural language, or perhaps using decision tables. These can be replaced by prototypes, which offer the advantage of being able to test (simulate) these operating procedures prior to implementation, and then use one of the software environment printouts as both documentation and operations manual.

Maintenance can be thought of as accomodating functional changes to the system (major or minor), the need for which may be initiated internally or externally.

Internal impetus would include bugs and other weak-

nesses, or a change in the user's (management's) needs. External factors would include legal or governmental considerations, or the opportunity to implement a new technology.

It is imperative that the software environment encourage the maintenance of systems, thereby allowing the software to keep up with the dynamics of its surrounding environment.

Performance Measurement and Evaluation. The performance of an operational system may be measured by a number of methods, and must be compared to the performance characteristics called for in the requirement definition phase.

A prototype lends itself to the simulation of a given environment, so that an estimate of a performance metric may be arrived at prior to the determination of the actual figure in an operational system.

A post-implementation evaluation of an existing system is crucial to the quality of feedback given to the software people who worked on the project. This may also unearth suggestions that could be added onto the maintenance schedule now, or be incorporated into future phases of the project.

III. ABL- AN ALTERNATIVE BASED LANGUAGE

The ease and efficiency of software modifiability is maximized through localization of the effect of changes to that software.

This concept is referred to as "information hiding", the "virtual machine concept", or "data abstraction" [BELABO, PARN79]. Dijkstra remarks that "there is ... an abstraction involved in naming an operation and using it on account of what it does, while completely disregarding how it works" [DIJK72].

In ABL the abstract machine and the abstract program have been patterned after these concepts.

The Abstract Program, Abstract Machine. In ABL, the abstract machine is defined as a set of actions, represented by a set of operations and a set of data objects. These are established to abstract both the operations (actions) and their data elements (data objects).

Here, the ABL user need not distinguish between "real" instructions and data types actually available in the implementation language and on the host machine from

those that he has defined in his abstract machine. Furthermore, the system designer may invent new software instructions and develop new data types, without regard to those that are previously hardware or programming language implemented.

Another strong feature of this aspect of ABL is that the transformation from abstract machine to hardware machine need not be done in a single step. A step-by-step refinement process will break down a large problem into a set of smaller ones, and, indeed, aid in the determination of the appropriate subsets toward the eventual solution. This again helps in the machine transportability, or "mobility", by remaining essentially independent from the implementation level until the lowest level of abstraction.

The abstract program in ABL is defined as a set of interconnected alternatives, or steps. An abstract program describes the current level of abstraction of the program modelling process. Successive refinements of the program model will eventually result in the program model being equivalent to the program [LEBE82b].

Within each abstract program, the set of interconnected steps define the program flow.

Steps are constructed out of alternatives. These alternatives consist of groups of actions, which are

sequences of instructions, each with one entry and one exit. These actions are to be executed in discrete sections, and under mutually exclusive conditions.

These alternatives are "guarded" by specific conditions giving us the opportunity to choose between alternatives, depending upon the instantaneous values in the state vector.

These conditions may be thought of as a special case of an action, with only a Boolean value returned, that of "TRUE" or "FALSE".

With the actions "executing" the available instructions from the abstract machine, and the conditions controlling the execution of these actions, we have a viable software tool.

Backus-Naur Form of ABL. The most widely accepted formal notation to describe the syntax of programming languages and software tools is Backus-Naur Form (BNF) [JENS74, KORF74]. Figure 3 uses BNF to define ABL and its components.

NOTE: The following five symbols are Meta-symbols belonging to the BNF formalism, and are not symbols of ABL.

::=	denotes "IS DEFINED AS"
|	denotes 'OR' condition between symbols on either side of the | sign.
{ }	Denotes possible repetition of the enclosed symbol(s) zero or more times.
[ ]	Denotes possible repetition of the enclosed symbol(s) one or more times.
(BLANK)	Delimiter, to separate different objects.

```
<ABL>          ::=     { <ABSTRACT MACHINE> }  { <ABSTRACT PROGRAM> }

<ABSTRACT MACHINE> ::= <MACHINE SECTION> <PREDICATE SECTION>
                       <ACTION SECTION> <DATA OBJECT SECTION>

<MACHINE SECTION> ::=   <MACHINE IDENTIFIER>
                        <MACHINE DESCRIPTION>

<MACHINE IDENTIFIER> ::=   <IDENTIFIER>

<IDENTIFIER>         ::=   [ <CHARACTER> ]

<MACHINE DESCRIPTION> ::=   <DESCRIPTION LINE>

<DESCRIPTION LINE> ::=   [ <CHARACTER> ]

<CHARACTER> ::=   <CHAR>  |  <DIGIT>  |  <SPECIAL CHARACTER>

<CHAR>      ::=    A | B | C | ... | Z

<SPECIAL CHARACTER>   ::=  ** SEE NOTE 1

<PREDICATE SECTION> ::=       <PREDICATE LIST>
                              <COMPOUND PREDICATE LIST>

<PREDICATE LIST> ::=   { <PREDICATE> }

<PREDICATE> ::=  <PREDICATE NUMBER>  <PREDICATE DESCRIPTION>
                 <ARGUMENT LIST>  <EXECUTABLE CODE>  <QUANTUM>

<PREDICATE NUMBER> ::=   <NUMBER>

<NUMBER>           ::=   [ <DIGIT> ]

<DIGIT>          ::=   0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<PREDICATE DESCRIPTION> ::=   <DESCRIPTION LINE>

<ARGUMENT LIST>    ::=   <INPUT ARGUMENT LIST>
                         <OUTPUT ARGUMENT LIST>

<INPUT ARGUMENT LIST> ::=   { <ARGUMENT> }

<OUTPUT ARGUMENT LIST> ::=   { <ARGUMENT> }

<ARGUMENT> ::=     <DATA OBJECT NUMBER>

<EXECUTABLE CODE> ::= { <IMPLEMENTATION LANGUAGE STATEMENT>}

<IMPLEMENTATION LANGUAGE STATEMENT> ::=   ** SEE NOTE 2
```

```
<QUANTUM> ::=      <ESTIMATED EXECUTION>

<ESTIMATED EXECUTION> ::=    <NUMBER>

<COMPOUND PREDICATE LIST> ::=  { <COMPOUND PREDICATE> }

<COMPOUND PREDICATE> ::=  <PREDICATE>
                          <ABSTRACT PROGRAM IDENTIFIER>
                          <MACHINE IDENTIFIER>

<ABSTRACT PROGRAM IDENTIFIER> ::=  <PROGRAM IDENTIFIER>

<ACTION SECTION> ::=    <ACTION LIST>  <COMPOUND ACTION LIST>

<ACTION LIST>     ::=   { <ACTION> }

<ACTION> ::=    <ACTION NUMBER>  <ACTION DESCRIPTION>
                <ARGUMENT LIST>  <EXECUTABLE CODE>  <QUANTUM>

<ACTION NUMBER> ::=    <NUMBER>

<ACTION DESCRIPTION> ::=    <DESCRIPTION LINE>

<COMPOUND ACTION LIST>    ::=    { <COMPOUND ACTION> }

<COMPOUND ACTION> ::= <ACTION> <ABSTRACT PROGRAM IDENTIFIER>
                      <MACHINE IDENTIFIER>

<DATA OBJECT SECTION> ::=    {  <DATA OBJECT NUMBER>
                                <DATA OBJECT DESCRIPTION> }

<DATA OBJECT NUMBER> ::=    <NUMBER>

<DATA OBJECT DESCRIPTION> ::=    <DESCRIPTION LINE>

<ABSTRACT PROGRAM> ::=   <PROGRAM SECTION>  <PROGRAM BODY>

<PROGRAM SECTION> ::=    <PROGRAM IDENTIFIER>
                         <MACHINE IDENTIFIER>
                         <PROGRAM DESCRIPTION>

<PROGRAM IDENTIFIER> ::=    <IDENTIFIER>

<PROGRAM DESCRIPTION> ::=    { <DESCRIPTION LINE> }

<PROGRAM BODY> ::=    [ <CLUSTER> ]

<CLUSTER> ::=     <CLUSTER IDENTIFIER>  <CLUSTER DESCRIPTION>
                  <ACTION LIST>    { <ALTERNATIVE> }
```

```
<CLUSTER IDENTIFIER> ::=   <CLUSTER NUMBER> . 0

<CLUSTER NUMBER> ::=   <NUMBER>

<CLUSTER DESCRIPTION> ::=   <DESCRIPTION LINE>

<ALTERNATIVE> ::=  <ALTERNATIVE IDENTIFIER>
                   <ALTERNATIVE DESCRIPTION>
                   <ARGUMENT LIST>  <ALTERNATIVE VECTOR>

<ALTERNATIVE IDENTIFIER> ::=   <CLUSTER NUMBER> .
                               <ALTERNATIVE NUMBER>

<ALTERNATIVE NUMBER> ::=   <DIGIT>

<ALTERNATIVE DESCRIPTION> ::=   <DESCRIPTION LINE>

<ALTERNATIVE VECTOR> ::=  <PRECONDITION LIST>
                          <ACTION SEQUENCING LIST>
                          <POSTCONDITION LIST>
                          <NEXT STEP>  <EXCEPTION STEP>

<PRECONDITION LIST>   ::=   <CONDITION LIST>

<CONDITION LIST> ::= { <PREDICATE NUMBER> <PREDICATE VALUE>}

<PREDICATE VALUE> ::=   Y | N

<ACTION SEQUENCING LIST> ::=
                { <ACTION NUMBER> <ACTION SEQUENCE NUMBER> }

<ACTION SEQUENCE NUMBER> ::=   <NUMBER>

<POSTCONDITION LIST> ::=   <CONDITION LIST>

<NEXT STEP> ::=    <CLUSTER NUMBER>

<EXCEPTION STEP> ::=   <CLUSTER NUMBER>


** NOTE 1: SPECIAL CHARACTERS ARE DEFINED BY THE IMPLEMENTA-
          TION CHARACTER SET.

** NOTE 2: THE IMPLEMENTATION LANGUAGE STATEMENTS ARE DEFIN-
          ED BY THE IMPLEMENTATION LANGUAGE BNF.
```

Figure 3. Backus-Naur Form of ABL.

The ABL Process. In order to actually develop software using ABL, a methodology has been developed whereby all components and their inter-relationships may be determined in a logical, step-wise fashion.

Software development using ABL is composed of problem design and decomposition, followed by coding in the implementation language, of a multitude of small linear segments of code [BELK76, FANC76].

In order to illustrate this ABL Process, a sample problem of the functional design of a simple portable electronic calculator is used (Figure 4). Here, the calculator operands are entered in postfix fashion from a keypad, and results are shown on an LED display. This straightforward example could easily be extended to the design of a full-fledged multi-function calculator; this abstract program would form one of the building blocks at a lower level of abstraction.

In our opinion, this is a very flexible method needed to design, build, execute and test the logic of the calculator before a single printed circuit board is constructed.

The ABL Process to produce the design and solution of a given project is enumerated in these guidelines:

1.0 Name the project and perform a primary investigation.

2.0 Define the project boundaries.

3.0 Divide the project into a set of discrete named modules, or abstract programs {Pj}.

(Program 01 in the example)

4.0 For each Pj, define the set of sub-tasks within the module, called steps or clusters {Cj}.

(labelled C1, C2, C3)

5.0 For each Cj, define the set of alternatives {ALj} within the cluster.

(columns 1-9 in the example. The downward arrows 'V' indicate belonging of an alternative to a particular cluster. Note also the cluster and alternative descriptions at the top).

6.0 For each ALj,

6.1 Define the set of data input objects {DIj}.

These are the data elements used in evaluating the preconditions and the data elements referenced in the actions.

(the asterisks * for D1 and D2 indicate input references to relevant data objects).

6.2 Define the set of preconditions {PRj}.

These are similar to Dijkstra's "guard", in that all alternatives of the current cluster have their preconditions evaluated, and only the one

alternative whose guard is true (or empty) proceeds to the action execution stage. Each of these preconditions are equivalent to a logical 'IF' statement, or each may be a compound predicate. These represent a more complex evaluation of the current state, but still return a single Boolean function. To avoid any non-determinism, only one precondition may be true. (The precondition entries are labelled P1 to P7. They indicate the Boolean values "guarding" the alternative:

Y = Yes (True)

N = No (False)

- = Don't Care)

6.3 Define the set of actions to be executed {AC$_j$}. An action may be simple or compound. A simple action is a sequence of one or more statements, i.e., code segments with one entry and one exit. A compound action invokes the next lower level of abstraction by transferring control to another abstract program and associated abstract machine. This is the equivalent of a sub-module or sub-routine, each starting the ABL process again.

(The actions are labelled A1 to A10. The action

entries indicate the sequence of appropriate actions to be executed.)

6.4 Define the set of postconditions {POj}. Similar in form to preconditions, they enable validity checks to be done after all the actions of the alternative are completed. This is important when proving programs, and to add criteria of certainty during the execution of a program. These are most commonly used to test boundary conditions.
(The postconditions are labelled S11 and S12).

6.5 Define the set of data output objects {DOj}. These are the data elements modified by the actions in the given alternative.
(The asterisks * for fields labelled D1, D2, D3 indicate modification of relevant data objects.)

6.6 Define the Next Step indicator Nj.
If all relevant postconditions in the alternative are satisfied, then control is transferred to the cluster number indicated by Nj.
If Nj=0, then we terminate the current abstract program, resume the next higher level of abstraction by transferring control back to the invoking abstract program, reinstating it's state vector, and continuing execution from the

subsequent action.

(The Next Step vector is labelled 'NEXT').

6.7 Define the Exception Step indicator Ej. If any of the postconditions in the alternative are not satisfied, then control is transferred to the cluster number indicated by Ej. This is commonly an error or other extraordinary situation. If Ej=0, then we terminate the current abstract program, as described in 6.6.

(The Exception Step vector is labelled 'EXCP').

The chronology of building ABL programs is sufficiently flexible, so that the sequence of six steps listed above need not necessarily correspond to the formation of an ABL module.

The example in Figure 4 is not meant to be an exhaustive solution of the fictitious calculator problem; it only attempts to illustrate the form of ABL by means of a simple example.

The above ABL Process has been defined using structured conventional techniques. This ABL Process is also shown in a simple ABL abstract program in Figure 5.

The reader can see a distinct difference between the conventional form and the ABL form.

```
MACHINE: CA  PORTABLE ELECTRONIC CALCULATOR MACHINE
         PROGRAM: 01  BASIC POSTFIX CALCULATOR WITH +,-,*,/,=  OPERANDS

CLUSTER  1.0  INITIALIZE ON POWER-UP OR CLEAR
              1.1  ( 1)  INITIALIZE PROCEDURE
CLUSTER  2.0  DETERMINE CHARACTER ENTERED AND DECODE
              2.1  ( 2)  NUMERIC ENTERED
              2.2  ( 3)  '+' OPERAND ENTERED
              2.3  ( 4)  '-' OPERAND ENTERED
              2.4  ( 5)  '*' OPERAND ENTERED
              2.5  ( 6)  '/' OPERAND ENTERED, NON-ZERO BUFFER
              2.6  ( 7)  '/' OPERAND ENTERED, BUT DIVISION BY ZERO
              2.7  ( 8)  '=' OPERAND ENTERED
CLUSTER  3.0  ERROR CONDITION
              3.1  ( 9)  PRINT ERROR MESSAGE


     1 2 3 4 5 6 7 8 9  CLUSTERS / ALTERNATIVES
C1   V                  INITIALIZE ON POWER-UP OR CLEAR
C2   . V V V V V V V .  DETERMINE CHARACTER ENTERED AND DECODE
C3   . . . . . . . V   ERROR CONDITION

                        DATA INPUTS
D2   . * * * * * * .    BUFFER
D1   . . * * * * . .    ACCUMULATOR

                        PRECONDITIONS
P1   - Y - - - - - - -  CHAR = 0..9
P2   - - Y - - - - - -  CHAR = '+'
P3   - - - Y - - - - -  CHAR = "-"
P4   - - - - Y - - - -  CHAR = "*"
P5   - - - - - Y Y - -  CHAR = "/"
P6   - - - - - N Y - -  BUFFER = 0
P7   - - - - - - - Y -  CHAR = "="

                        ACTIONS
A1   1 . . . . . . . .  ACCUMULATOR = 0    O1
A2   2 . 2 2 2 2 . . .  CLEAR INPUT BUFFER & L.E.D. DISPLAY    O2
A3   3 2 3 3 3 3 . . .  ACCEPT NEXT CHAR FROM KEYPAD    I3
A4   . 1 . . . . . . .  PLACE DIGIT OR DECIMAL POINT INTO BUFFER    I3  O2
A5   . . 1 . . . . . .  ADD CONTENTS OF BUFFER INTO ACCUMULATOR    I2  O1
A6   . . . 1 . . . . .  SUBTRACT CONTENTS OF BUFFER FROM ACCUMULATOR    I2  O1
A7   . . . . 1 . . . .  MULTIPLY CONTENTS OF BUFFER TIMES ACCUMULATOR    I2  O1
A8   . . . . . 1 . . .  DIVIDE CONTENTS OF BUFFER INTO ACCUMULATOR    I2  O1
A9   . . . . . . 1 . .  PRINT CONTENTS OF ACCUMULATOR    O1
A10  . . . . . . . . 1  PRINT ERROR MESSAGE

                        POSTCONDITIONS
S11  - N - - - - - - -  BUFFER VALUE > 999999.99        (OVERFLOW)
S12  - - N N N N - - -  ACCUMULATOR VALUE > 999999.99   (OVERFLOW)

                        DATA OUTPUTS
D1   * . * * * * . * .  ACCUMULATOR
D2   * * * * * * . . .  BUFFER
D3   * * * * * * . . .  CHARACTER INPUT FROM KEYPAD

NEXT 2 2 2 2 2 2 3 1 1

EXCP 0 3 3 3 3 3 3 3 0
```

Figure 4. The ABL Process- Calculator Example

```
MACHINE: PR  THE ABL PROCESS - ABSTRACT MACHINE
             PROGRAM: 01  THE ABL PROCESS- GUIDELINES IN ABL FORM

CLUSTER   1.0  NAME THE PROJECT AND PERFORM A PRELIMINARY INVESTIGATION
               1.1  ( 1)  NAME THE PROJECT AND PERFORM A PRELIMINARY INVESTIGATION
CLUSTER   2.0  DEFINE THE PROJECT BOUNDARIES
               2.1  ( 2)  DEFINE THE PROJECT BOUNDARIES
CLUSTER   3.0  DIVIDE THE PROJECT INTO DISCRETE MODULES (ABSTRACT PROGRAMS)
               3.1  ( 3)  DIVIDE THE PROJECT INTO DISCRETE MODULES (ABSTRACT PROGRAMS)
CLUSTER   4.0  FOR EACH ABSTRACT PROGRAM, DEFINE THE STEPS (CLUSTERS)
               4.1  ( 4)  FOR EACH ABSTRACT PROGRAM, DEFINE THE STEPS (CLUSTERS)
CLUSTER   5.0  FOR EACH CLUSTER, DEFINE THE ALTERNATIVES WITHIN THE CLUSTER
               5.1  ( 5)  FOR EACH CLUSTER, DEFINE THE ALTERNATIVES WITHIN THE CLUSTER
CLUSTER   6.0  FOR EACH ALTERNATIVE, DEFINE THE FOLLOWING COMPONENTS:
               6.1  ( 6)  DEFINE THE DATA INPUT OBJECTS
               6.2  ( 7)  DEFINE THE PRECONDITIONS TO BE EVALUATED
               6.3  ( 8)  DEFINE THE ACTIONS TO BE EXECUTED
               6.4  ( 9)  DEFINE THE POSTCONDITIONS TO BE EVALUATED
               6.5  (10)  DEFINE THE DATA OUTPUT OBJECTS
               6.6  (11)  DETERMINE THE NEXT STEP AND THE EXCEPTION STEP

      1 2 3,4 5 6 7 8 9 10 11  CLUSTERS / ALTERNATIVES
C1    V . . . . . . . . . .    NAME THE PROJECT AND PERFORM A PRELIMINARY INVESTIGATION
C2    . V . . . . . . . . .    DEFINE THE PROJECT BOUNDARIES
C3    . . V . . . . . . . .    DIVIDE THE PROJECT INTO DISCRETE MODULES (ABSTRACT PROGRAMS)
C4    . . . V . . . . . . .    FOR EACH ABSTRACT PROGRAM, DEFINE THE STEPS (CLUSTERS)
C5    . . . . V . . . . . .    FOR EACH CLUSTER, DEFINE THE ALTERNATIVES WITHIN THE CLUSTER
C6    . . . . . V V V V V V    FOR EACH ALTERNATIVE, DEFINE THE FOLLOWING COMPONENTS:

                               PRECONDITIONS
P1    - - - - - Y - - - - N    ANY MORE DATA INPUT OBJECTS ?
P2    - - - - - - Y - - - N    ANY MORE PRECONDITIONS ?
P3    - - - - - - - Y - - N    ANY MORE ACTIONS ?
P4    - - - - - - - - Y - N    ANY MORE POSTCONDITIONS ?
P5    - - - - - - - - - Y N    ANY MORE DATA OUTPUT OBJECTS ?

                               ACTIONS
A1    1 . . . . . . . . . .    NAME THE PROJECT
A2    2 . . . . . . . . . .    PERFORM A PRELIMINARY INVESTIGATION
A3    . 1 . . . . . . . . .    DEFINE THE PROJECT BOUNDARIES
A4    . . 1 . . . . . . . .    DIVIDE THE PROJECT INTO DISCRETE MODULES (ABSTRACT PROGRAMS)
A5    . . . 1 . . . . . . .    DIVIDE THE ABSTRACT PROGRAMS INTO STEPS (CLUSTERS)
A6    . . . . 1 . . . . . .    DIVIDE THE CLUSTERS INTO ALTERNATIVES
A7    . . . . . 1 . . . . .    DEFINE THE DATA INPUT OBJECTS
A8    . . . . . . 1 . . . .    DEFINE THE PRECONDITIONS
A9    . . . . . . . 1 . . .    DEFINE THE ACTIONS
A10   . . . . . . . . 1 . .    DEFINE THE POSTCONDITIONS
A11   . . . . . . . . . 1 .    DEFINE THE DATA OUTPUT OBJECTS
A12   . . . . . . . . . . 1    DETERMINE THE NEXT STEP
A13   . . . . . . . . . . 2    DETERMINE THE EXCEPTION STEP

                               POSTCONDITIONS
S11   Y - - - - - - - - - -    AFTER PRELIMINARY INVESTIGATION: CONTINUE THE PROJECT ?
S12   - Y Y Y Y - - - - - -    IS THIS STEP COMPLETE ?
S13   - - - - - - - - - - Y    ANY FURTHER PROJECTS ?

NEXT  2 3 4 5 6 6 6 6 6 6 1

EXCP  0 2 3 4 5 6 6 6 6 6 0
```

Figure 5. The ABL Process in ABL form

ABL Templates. Templates are defined in order to provide an unalterable framework for the addition of abstract machines and programs. This is consistent with the statement that "templates reinforce the view that a program is a hierarchical composition of syntactic objects, rather than a sequence of characters" [TEITB1].

These templates can serve as building blocks for program development, and also as an alternative method of formally describing the structure of ABL.

There are two templates in ABL:
Figure 6a contains the abstract machine template;
Figure 6b contains the abstract program template.

```
M.0   <ABSTRACT MACHINE TEMPLATE>
      M.1  <MACHINE IDENTIFIER>  <MACHINE DESCRIPTION>

      M.2  <PREDICATE NUMBER>  <PREDICATE DESCRIPTION>
           <INPUT ARGUMENT LIST>  <OUTPUT ARGUMENT LIST>
           <EXECUTABLE CODE>  <QUANTUM>

      M.3  <COMPOUND PREDICATE NUMBER>
           <COMPOUND PREDICATE DESCRIPTION>
           <PROGRAM IDENTIFIER>  <MACHINE IDENTIFIER>

      M.4  <ACTION NUMBER>  <ACTION DESCRIPTION>
           <INPUT ARGUMENT LIST>  <OUTPUT ARGUMENT LIST>
           <EXECUTABLE CODE>  <QUANTUM>

      M.5  <COMPOUND ACTION NUMBER>
           <COMPOUND ACTION DESCRIPTION>
           <PROGRAM IDENTIFIER>  <MACHINE IDENTIFIER>

      M.6  <DATA OBJECT NUMBER>  <DATA OBJECT DESCRIPTION>
```

Figure 6a. ABL Abstract Machine Template.

```
P.0   <ABSTRACT PROGRAM TEMPLATE>

      P.1  <PROGRAM IDENTIFIER>  <MACHINE IDENTIFIER>
           <PROGRAM DESCRIPTION>

      P.2  <CLUSTER NUMBER>  <CLUSTER DESCRIPTION>
           <ACTION LIST>

      P.3  <ALTERNATIVE NUMBER>  <ALTERNATIVE DESCRIPTION>
           <INPUT ARGUMENT LIST>  <OUTPUT ARGUMENT LIST>
           <PRECONDITION LIST>  <ACTION LIST>
           <POSTCONDITION LIST>
           <NEXT STEP>  <EXCEPTION STEP>
```

Figure 6b. ABL Abstract Program Template.

Characteristics and Principles of ABL. There exists a number of basic principles to be applied in order to ensure the quality of software tools and environments. Some focus on intangible properties such as "be easily understood" [FREEBO], and others have been developed with the intention of collecting software metrics, such as "structuredness" or "efficiency" [BOEH78].

ABL stresses the use of structured techniques and modularization in its design. This approach allows the software engineering process to proceed in a natural, systematic way. It is also critical to the validity of many of the following properties, all of which were design goals of ABL.

The self-documenting aspect of ABL is maximized by allowing as much use of natural language descriptors as is necessary, and by also permitting unlimited levels of refinement.

ABL is an interactive computer system that is designed to be visually appealing to the user (using aesthetically pleasing CRT displays and legible printouts).

This human engineering consideration is succinctly stated by Mitchell, "An (interactive system) can be thought of as a small, closed society of man and system with mutual feedback" [MITC79].

A "user friendly" system must match the capabilities of

the computer automaton to the needs of the human component by recognizing certain traits of human psychology. Criteria such as flexibility, the use of full names for descriptors, on-line help for clarifications, unambiguous responses and error messages, satisfactory response time, and general simplicity [GOODB1] are all attempted by ABL.

A likely result of this approach is that "users of a flexible interface will perceive their system to be more benevolent than will users of a rigid, unadaptable interface" [WALT73]. This is a critical factor in the user acceptance and therefore the long-term success of a software tool.

The conventional programming methodology provides very little experience in software design and an abundance of coding skill. The low level of complexity, simplicity of decomposition of the design problem and the apparent ease of operation of the ABL tool can result in reduced demands in program design expertise.

The need for ease of program modifiability recognizes "the evolutionary nature of software, namely that programs are not static objects but undergo continuous modification to cope with the everchanging environment. That is, in addition to dynamic execution on a machine, programs display dynamics of their own evolution and growth while execution is at rest" [BELA80]. In order to accomplish

this, ABL has a powerful software update facility (to be shown later), which may be used in a static fashion (just the editor), or dynamically, (through the interactive interpreter) during simulation, testing, or execution.

IV. ABL AS A TOOL IN THE SOFTWARE DEVELOPMENT PROCESS

Applicability to the Seven Development Stages. ABL, as a complete software tool, has utility at every stage in the software development process- from conception and definition through to implementation and maintenance.

In the problem definition and requirements stage, Analytical Modelling and Dynamic Environment Simulation [REIF79] is facilitated in ABL by virtue of its prototyping capabilities and ease of use. This is important for the time / dollar estimates being developed.

In the preliminary design stage, refinement of prototyping into a more detailed level of modularization is done with the aid of ABL. The outputs are to be included in the System Overview document.

The detailed design stage continues the refinement of models and modules with ABL. Use of ABL as a data description language, by listing data elements and their interrelationships would be done at this stage, as well.

In the programming and development stage, use of post-conditions provide testing and debugging aids. The subsequent inclusion of the 'quantum' fields as defined in the BNF would facilitate the analysis of timing characteristics.

Simulation, flow tracing, breakpoint setting, dynamic program modification, error trapping, dynamic variable assignment and execution monitoring have been implemented in the ABL interactive interpreter, providing a powerful means of dynamic software development and testing. These techniques may be applied at any level of the hierarchy, so that high-or medium-levels of refinement of a problem may be "walked-through" and documented before proceeding to the next lower level. It also means that modules can be developed and tested separately, and interfaced at a common higher level later on.

By virtue of its structured approach, ABL is also very supportive of machine independence, or "migration" [MCIN78]. The implementation language(s) and host machine(s) need not be committed to until many steps into the refinement process, so that transportability from one machine or one language to another can be achieved easily.

For testing and implementation, ABL lends itself to powerful testing and analysis tools, both static and dynamic.

In the static category, there exist syntax analyzers and other error scans for unreferenced elements, and noncorrespondance of data types [OSTE80]. Control-flow graphs may be used for loop optimization and complexity measurement [MCCA76]. Data-flow analysis indicates relationships between variables, specifically the dependence between variables (to indicate side-effects, etc.), and the mutually exclusive or independent variables, to aid in the partitioning of modules. Symbolic execution, deriving symbolic expressions for outputs in terms of inputs, are the extensions of these data-flow diagrams [CLAR76, CLAR80].

The combinatorial nature of these techniques is nontrivial, and is beyond the scope of this work.

Operations and maintenance are facilitated by the modular, structured programming techniques employed by ABL, and by the natural, self-documenting approach.

Performance evaluation in ABL is primarily accomplished by measuring efficiency of execution. Each ABL primitive construct (predicate, action) has an estimated execution figure associated with it. The sum of execution times (quantum) for different program models may be compared in order to determine the most efficient structure.

Features of the ABL Software tool. The ABL on-line development process is unlike conventional programming techniques, in that it need not be done in a sequential fashion.

Even when simulating the execution of a program, pitfalls such as unsatisfied references are designed to indicate to the programmer immediately, who will then have the choice of suspending execution, or merely "plugging in" a value and resuming execution.

Thus, the programmer can monitor the program through his "window" into the computer. During flow tracing, as control passes outside the confines of the window, or upon entering or exiting another level of abstraction, the display can be automatically redrawn to accomodate the new environment [TEIT81].

With this powerful development tool, reverse execution of a program as a debugging method becomes feasible as well. This is accomplished by maintaining a history of data objects that have been modified (and their values), and successively restoring these values and the associated control flow step-by-step, thereby giving the illusion of the program executing backwards [WILC76].

Control Flow diagrams in ABL are defined by the precondition and postcondition sections and the Next and Exception Step vectors.

Data Flow diagrams are defined by the input and output data objects associated with the actions.

A consideration for future development is to add an action section to each cluster.

This would mean defining an action or series of actions to affect all alternatives within the cluster. A prime case is to implement semaphores for concurrent processing in this section, to inhibit the execution of the cluster if it were a critical process. Schema such as the Communication Port (CP) [MADBO] have already been designed, and could be used as the synchronization primitives here.

The constructs of ABL have all been designed to facilitate software updating and modification in the design, implementation and maintenance phases of a software project.

Many display formats are available in ABL to show all relevant conditions, actions and relationships within each alternative of the module. These will be shown in the Software Environment Chapter.

A natural language description of each component of the module (clusters, alternatives, predicates, actions, etc.) provide self-documentation.

The table or "Vertical" format of ABL tends to force a clear problem statement of the abstract program, and shows

which elements of the abstract machine are referenced.

This form augments the neatness, compactness, and readability of the program.

A consideration for limiting the complexity of a particular module could be such that the table size not be greater than one page in length or width. As a rule of thumb, anything larger should be segmented into a more manageable table size.

The Comput format of ABL gives a conventional or "Horizontal" representation of ABL. This resembles the 'case' construct found in many conventional programming languages.

V. THE SOFTWARE ENVIRONMENT FOR ABL

Implementation: The Host System. The ABL Software Environment was implemented on a Tyme System 80 / 160 minicomputer.

The system configuration consists of:

Hardware:
- Data General Nova 4 central processor, with 256K bytes of main memory.
- Control Data 9762 Storage Module Disk Drives, with 130 megabytes of random access storage.
- Printronix P600 Matrix Printer/Plotter, with OCR-A font.
- Cybernex XL-87H Video Terminals, with twin intensities and full cursor addressing.

Software:
- MICOS Release 11.2 Operating System
- EXTENSIVE BASIC Language (Interpretive, with ISAM)
- Tyme Systems Utility programs and program development macros.

This system allows for powerful interactive access, and facilitates "user-friendly" programming.

The ABL system requires approximately .5 mbytes of disk storage for programs, documentation, and workfiles.

The Tyme utilities provide for the user's choice of language (English or French) by terminal, by keeping screen layouts, report masks and all messages stored in utility files. Though the ABL software environment is written in English only, it could quickly and easily become bilingual.

Description of Implemented System. The ABL Software Environment consists of two main modules: the editors and the interactive interpreter. The six functions implemented to support these modules are:

1) ABL Program Editor
2) ABL Program Listings
3) ABL Machine Editor
4) ABL Machine Listings
5) ABL Program and Machine Library
6) ABL Interactive Interpreter

These functions are shown in the ABL Environment System Diagram in Figure 7.

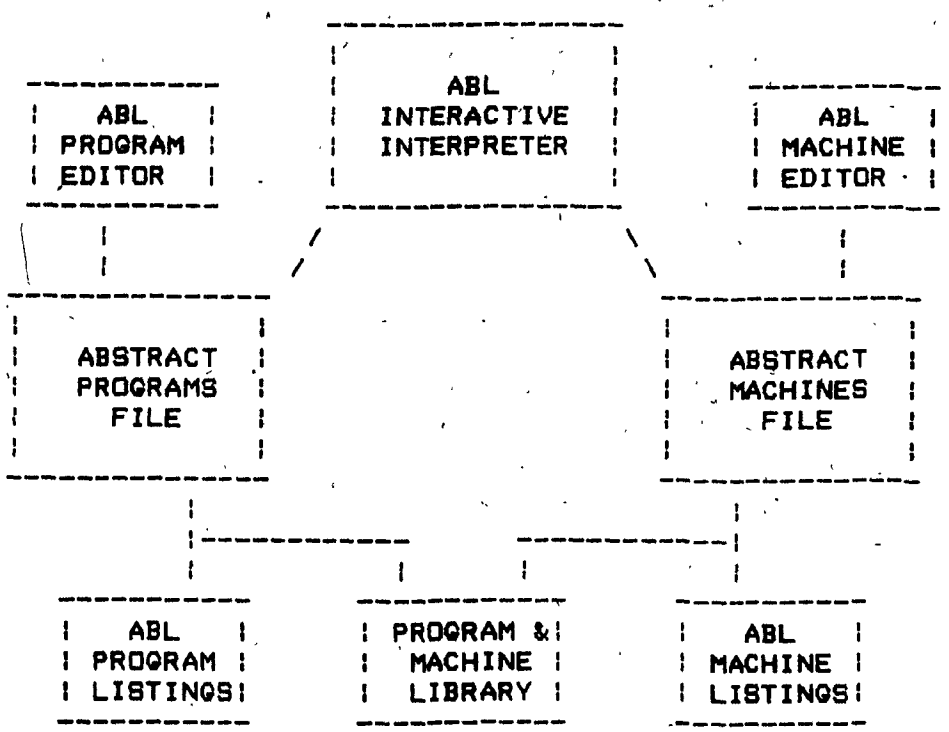

Figure 7. ABL Environment System Diagram

The ABL Program Editor allows the entry, modification or deletion of abstract programs from the master program library.

A hard-copy of these programs may currently be obtained in three formats:

i) the Vertical Transform, which is the most complete listing available. An example has already been presented in Figure 4, the Calculator example.

ii) the Comput form, which is shown in Figure 8, using the same electronic calculator example as in Figure 4. A Comput may be thought of as a "building block" of a module. It is equivalent to a cluster, in that it is composed of a collection of alternatives, selected by a "guard". The computs are labelled 0, 1, 2, and 3 in the example.

iii) the Compact forms of the ABL programs. Figure 9a shows the compact abstract program of alternatives, a "shorthand" method of defining each alternative and its contents. Figure 9b, the Compact Abstract Program Table, is the internal representation of the abstract program in the form of a two-dimensional matrix, as maintained by the ABL software environment.

The ABL Machine Editor allows the entry, modification or deletion of abstract machines. These comprise predicates (pre- and postconditions), actions and data objects.

The hard-copy listing of the portable electronic calculator machine is shown in Figure 10.

A general listing of all machines and all programs that may be run on those machines is shown on the Program and Machine Library Listing (Figure 11).

```
MACHINE: CA  PORTABLE ELECTRONIC CALCULATOR MACHINE
         PROGRAM: 01  BASIC POSTFIX CALCULATOR WITH +,-,*,/,= OPERANDS

0 :  <ENTRY - EXIT POINT>

1 :  COMPUT   INITIALIZE ON POWER-UP OR CLEAR
   [ ]     ACCUMULATOR = 0
           CLEAR INPUT BUFFER & L.E.D. DISPLAY
           ACCEPT NEXT CHAR FROM KEYPAD
           [ ]    ---> 2   EXCEPTION ---> 0

2 :  COMPUT   DETERMINE CHARACTER ENTERED AND DECODE
           [ CHAR = 0..9
             CHAR = '+'
             CHAR = "-"
             CHAR = "*"
             CHAR = "/"
             BUFFER = 0
             CHAR = "="                                          ]
   [Y,-,-,-,-,-,-]  :   PLACE DIGIT OR DECIMAL POINT INTO BUFFER
                        ACCEPT NEXT CHAR FROM KEYPAD
                        [ BUFFER VALUE > 999999.99       (OVERFLOW)          ]
                        [N]    ---> 2   EXCEPTION ---> 3
   [-,Y,-,-,-,-,-]  :   ADD CONTENTS OF BUFFER INTO ACCUMULATOR
                        CLEAR INPUT BUFFER & L.E.D. DISPLAY
                        ACCEPT NEXT CHAR FROM KEYPAD
                        [ ACCUMULATOR VALUE > 999999.99   (OVERFLOW)          ]
                        [N]    ---> 2   EXCEPTION ---> 3
   [-,-,Y,-,-,-,-]  :   SUBTRACT CONTENTS OF BUFFER FROM ACCUMULATOR
                        CLEAR INPUT BUFFER & L.E.D. DISPLAY
                        ACCEPT NEXT CHAR FROM KEYPAD
                        [ ACCUMULATOR VALUE > 999999.99   (OVERFLOW)          ]
                        [N]    ---> 2   EXCEPTION ---> 3
   [-,-,-,Y,-,-,-]  :   MULTIPLY CONTENTS OF BUFFER TIMES ACCUMULATOR
                        CLEAR INPUT BUFFER & L.E.D. DISPLAY
                        ACCEPT NEXT CHAR FROM KEYPAD
                        [ ACCUMULATOR VALUE > 999999.99   (OVERFLOW)          ]
                        [N]    ---> 2   EXCEPTION ---> 3
   [-,-,-,-,Y,N,-]  :   DIVIDE CONTENTS OF BUFFER INTO ACCUMULATOR
                        CLEAR INPUT BUFFER & L.E.D. DISPLAY
                        ACCEPT NEXT CHAR FROM KEYPAD
                        [ ACCUMULATOR VALUE > 999999.99   (OVERFLOW)          ]
                        [N]    ---> 2   EXCEPTION ---> 3
   [-,-,-,-,Y,Y,-]  :   [ ]    ---> 3   EXCEPTION ---> 3
   [-,-,-,-,-,-,Y]  :   PRINT CONTENTS OF ACCUMULATOR
                        [ ]    ---> 1   EXCEPTION ---> 3

3 :  COMPUT   ERROR CONDITION
   [ ]     PRINT ERROR MESSAGE
           [ ]    ---> 1   EXCEPTION ---> 0
```

Figure 8. ABL Comput Form Listing

MACHINE: CA PORTABLE ELECTRONIC CALCULATOR MACHINE
PROGRAM: 01 BASIC POSTFIX CALCULATOR WITH +,-,*,/,= OPERANDS

COMPACT ABSTRACT PROGRAM OF ALTERNATIVES

ALTERNATIVE FORMAT IS: (HORIZONTALLY)
CLUSTER (ALTERNATIVE) / PRECONDITIONS / ACTIONS / POSTCONDITIONS / NEXT STEP / EXCEPTION STEP

1.1/0/1,2,3/0/2/0	1 INITIALIZE PROCEDURE
2.1/1/4,3/-11/2/3	2 NUMERIC ENTERED
2.2/2/5,2,3/-12/2/3	3 '+' OPERAND ENTERED
2.3/3/6,2,3/-12/2/3	4 '-' OPERAND ENTERED
2.4/4/7,2,3/-12/2/3	5 '*' OPERAND ENTERED
2.5/5,-6/8,2,3/-12/2/3	6 '/' OPERAND ENTERED, NON-ZERO BUFFER
2.6/5,6/0/0/3/3	7 '/' OPERAND ENTERED, BUT DIVISION BY ZERO
2.7/7/9/0/1/3	8 '=' OPERAND ENTERED
3.1/0/10/0/1/0	9 PRINT ERROR MESSAGE

Figure 9a. ABL Compact Abstract Program of Alternatives Listing

COMPACT ABSTRACT PROGRAM TABLE

TABLE FORMAT IS: (HORIZONTALLY)
CLUSTER (ALTERNATIVE) / PRECONDITIONS / 0 / ACTIONS / 0 / POSTCONDITIONS / 0 / NEXT STEP / EXCEPTION STEP

11	0	1	2	3	0	0	2	0	0	0	0	0
21	1	0	4	3	0	-11	0	2	3	0	0	0
22	2	0	5	2	3	0	-12	0	2	3	0	0
23	3	0	6	2	3	0	-12	0	2	3	0	0
24	4	0	7	2	3	0	-12	0	2	3	0	0
25	5	-6	0	8	2	3	0	-12	0	2	3	0
26	5	6	0	0	0	3	3	0	0	0	0	0
27	7	0	9	0	0	1	3	0	0	0	0	0
31	0	10	0	0	1	0	0	0	0	0	0	0

Figure 9b. ABL Compact Abstract Program Table Listing

CA PORTABLE ELECTRONIC CALCULATOR MACHINE

PREDICATES

1 CHAR = 0..9
CODE: IF CH$< 9 TIME (MSEC): 10 DATA OBJECT INPUTS: 3 OUTPUTS:

2 CHAR = "+"
CODE: IF CH$= "+" TIME (MSEC): 10 DATA OBJECT INPUTS: 3 OUTPUTS:

3 CHAR = "-"
CODE: IF CH$= "-" TIME (MSEC): 10 DATA OBJECT INPUTS: 3 OUTPUTS:

4 CHAR = "*"
CODE: IF CH$= "*" TIME (MSEC): 10 DATA OBJECT INPUTS: 3 OUTPUTS:

5 CHAR = "/"
CODE: IF CH$= "/" TIME (MSEC): 10 DATA OBJECT INPUTS: 3 OUTPUTS:

6 BUFFER = 0
CODE: IF BUF= 0 TIME (MSEC): 10 DATA OBJECT INPUTS: 3 OUTPUTS:

7 CHAR = "="
CODE: IF CH$= "=" TIME (MSEC): 10 DATA OBJECT INPUTS: 3 OUTPUTS:

11 BUFFER VALUE > 999999.99 (OVERFLOW)
CODE: IF BUF> 999999.99 TIME (MSEC): 18 DATA OBJECT INPUTS: 2 OUTPUTS:

12 ACCUMULATOR VALUE > 999999.99 (OVERFLOW)
CODE: IF ACC> 999999.99 TIME (MSEC): 18 DATA OBJECT INPUTS: 1 OUTPUTS:

ACTIONS

1 ACCUMULATOR = 0 PROGRAM/MACHINE: /
CODE: LET ACC=0 TIME (MSEC): 8 DATA OBJECT INPUTS: OUTPUTS: 1

2 CLEAR INPUT BUFFER & L.E.D. DISPLAY PROGRAM/MACHINE: /
CODE: LET BUF=0 TIME (MSEC): 8 DATA OBJECT INPUTS: OUTPUTS: 2

3 ACCEPT NEXT CHAR FROM KEYPAD PROGRAM/MACHINE: /
CODE: ACC 10,CH$ TIME (MSEC): 20 DATA OBJECT INPUTS: 3 OUTPUTS:

4 PLACE DIGIT OR DECIMAL POINT INTO BUFFER PROGRAM/MACHINE: /
CODE: CHG CH$,BUF TIME (MSEC): 18 DATA OBJECT INPUTS: 3 OUTPUTS: 2

5 ADD CONTENTS OF BUFFER INTO ACCUMULATOR PROGRAM/MACHINE: /
CODE: LET ACC=ACC+BUF TIME (MSEC): 15 DATA OBJECT INPUTS: 2 OUTPUTS: 1

6 SUBTRACT CONTENTS OF BUFFER FROM ACCUMULATOR PROGRAM/MACHINE: /
CODE: LET ACC=ACC-BUF TIME (MSEC): 15 DATA OBJECT INPUTS: 2 OUTPUTS: 1

7 MULTIPLY CONTENTS OF BUFFER TIMES ACCUMULATOR PROGRAM/MACHINE: /
CODE: LET ACC=ACC*BUF TIME (MSEC): 25 DATA OBJECT INPUTS: 2 OUTPUTS: 1

8 DIVIDE CONTENTS OF BUFFER INTO ACCUMULATOR PROGRAM/MACHINE: /
CODE: LET ACC=ACC/BUF TIME (MSEC): 25 DATA OBJECT INPUTS: 2 OUTPUTS: 1

9 PRINT CONTENTS OF ACCUMULATOR PROGRAM/MACHINE: /
CODE: PRT ACC TIME (MSEC): 40 DATA OBJECT INPUTS: OUTPUTS: 1

10 PRINT ERROR MESSAGE PROGRAM/MACHINE: /
CODE: PRT "ERROR- OVER/UNDER FLOW" TIME (MSEC): 40 DATA OBJECT INPUTS: OUTPUTS:

DATA OBJECTS

1 ACCUMULATOR
2 BUFFER
3 CHARACTER INPUT FROM KEYPAD

Figure 10. ABL Machine Listing

```
ABL MASTER LIBRARY OF MACHINES AND PROGRAMS


MACHINE: CA  PORTABLE ELECTRONIC CALCULATOR MACHINE

         PROGRAM  01
         BASIC POSTFIX CALCULATOR WITH +,-,*,/,= OPERANDS


MACHINE: DA  AUTOMATIC DIALER MACHINE- MINI-COMPUTER & SD-192 EPABX PHONE

         PROGRAM  01
         AUTO-DIALER DESIGN EXAMPLE  (LEVEL 00)

         PROGRAM  02
         AUTO-DIALER MODULE- GET-NEXT-PATIENT SUB-PROGRAM (LEVEL 01)


MACHINE: ED  ABL INTERPRETER EDIT MACHINE

         PROGRAM  01
         ABL INTERACTIVE INTERPRETER- EDIT PROGRAM (LEVEL 01)


MACHINE: EX  ABL PROGRAM EXECUTION MACHINE

         PROGRAM  01
         ABL INTERACTIVE INTERPRETER- EXECUTE PROGRAM (LEVEL 01)


MACHINE: IN  ABL EXECUTIVE INTERPRETER MACHINE

         PROGRAM  00
         ABL INTERACTIVE INTERPRETER- EXECUTIVE (LEVEL 00)
         THIS PROGRAM IS THE HIGHEST LEVEL OF ABSTRACTION OF THE
         ABL INTERACTIVE INTERPRETER


MACHINE: PA  ABL GET PARAMETERS MACHINE

         PROGRAM  01
         ABL INTERACTIVE INTERPRETER- PARAMETERS PROGRAM (LEVEL 01)


 6 MACHINES AND  7 PROGRAMS PRINTED
```

Figure 11. ABL Program and Machine Library Listing

The ABL Interactive Interpreter. At the heart of the ABL software environment is the interactive interpreter. This major program allows for the on-line design, development, simulation and testing of modules on an interactive basis.

In order to accomodate these needs, the interpretive route was selected, as opposed to compilation. This allows the user to be imprecise about certain bindings over control- or data-flows, and to use those bindings that become available when the objects are actually referenced. This flexibility is best achieved through interpretation [MITC79].

The ABL interpreter was itself designed using the software environment, and the basic control flow is shown in the following Vertical Transforms:

Level 00.01	Executive (main) Program	- Figure 12a.
Level 01.01	Edit Module	- Figure 12b.
Level 01.02	Monitor Parameters Module	- Figure 12c.
Level 01.03	Execute Module	- Figure 12d.

The Executive selects the program to be operated upon, and determines the subsequent module to be run, according to the user selection.

The edit module allows creation or modification of a complete alternative, and its components and descriptions.

The monitor parameters module sets or clears traces and breakpoints for any alternative. This allows user control of the detail of tracing and speed of execution [TEIT81].

There is also an option to dump the trace to the

printer. The execute module actually "runs" the program by stepping through the alternatives, evaluating predicates, and parsing and executing actions in the implementation language.

This Interactive Interpreter, like all the components in the ABL software environment, have been implemented using the EXTENSIVE BASIC language. The BASIC listing of the interpreter, with appropriate file layouts, may be found in the Appendix.

```
MACHINE: IN  ABL EXECUTIVE INTERPRETER MACHINE
             PROGRAM: 00  ABL INTERACTIVE INTERPRETER- EXECUTIVE (LEVEL 00).
                          THIS PROGRAM IS THE HIGHEST LEVEL OF ABSTRACTION OF THE
                          ABL INTERACTIVE INTERPRETER.

CLUSTER  1.0  DISPLAY SCREEN, GET MACHINE CODE
               1.1  ( 1)  DISPLAY SCREEN, GET MACHINE CODE
CLUSTER  2.0  GET PROGRAM CODE, VERIFY THAT IT EXISTS
               2.1  ( 2)  GET PROGRAM CODE, VERIFY THAT IT EXISTS
CLUSTER  3.0  DETERMINE DESIRED SUB-PROGRAM OR QUIT (END)
               3.1  ( 3)  EDIT SUB-PROGRAM
               3.2  ( 4)  SET PARAMETERS SUB-PROGRAM
               3.3  ( 5)  EXECUTE SUB-PROGRAM
               3.4  ( 6)  QUIT (END) INTERPRETER
               3.5  ( 7)  INVALID CHOICE OF OPTIONS
CLUSTER  4.0  LOAD PROGRAM, DISPLAY IT
               4.1  ( 8)  LOAD PROGRAM, DISPLAY IT
CLUSTER  5.0  PRINT QUESTION, ASK USER FOR CHOICE
               5.1  ( 9)  PRINT QUESTION, ASK USER FOR CHOICE


     1 2 3 4 5 6 7 8 9  CLUSTERS / ALTERNATIVES
C1   V . . . . . . . .  DISPLAY SCREEN, GET MACHINE CODE
C2   . V . . . . . . .  GET PROGRAM CODE, VERIFY THAT IT EXISTS
C3   . . V V V V V . .  DETERMINE DESIRED SUB-PROGRAM OR QUIT (END)
C4   . . . . . . . V .  LOAD PROGRAM, DISPLAY IT
C5   . . . . . . . . V  PRINT QUESTION, ASK USER FOR CHOICE

                        PRECONDITIONS
P1   - - Y - - - N - -  'E'DIT ENTERED ?
P2   - - - Y - - N - -  SET 'P'ARAMETERS ENTERED ?
P3   - - - - Y - N - -  E'X'ECUTE PROGRAM ENTERED ?
P4   - - - - - Y N - -  'Q'UIT ENTERED ?

                        ACTIONS
A1   1 . . . . . . . .  DISPLAY INTERACTIVE SCREEN
A2   2 . . . . . . . .  GET MACHINE CODE FROM USER
A3   3 . . . . . . . .  SEE IF MACHINE CODE EXISTS
A4   . 1 . . . . . . .  GET PROGRAM CODE FROM USER
A5   . 2 . . . . . . .  SEE IF MACHINE/PROGRAM EXISTS
A9   . . 1 . . . . . .  ** CALL EDIT PROGRAM  (01/ED) **
A10  . . . 1 . . . . .  ** CALL MONITOR PARAMETERS PROGRAM  (01/PA) **
A11  . . . . 1 . . . .  ** CALL EXECUTE PROGRAM  (01/EX) **
A6   . . . . . . . 1 .  LOAD EXISTING PROGRAM
A7   . . . . . . . 2 .  PRINT PROGRAM DESCRIPTION AND ALTERNATIVES
A8   . . . . . . . . 1  PRINT "EDIT, SET PARAMETERS, EXECUTE OR QUIT ? [EPXQ]"
A12  . . . . . . . . 2  GET RESPONSE FROM USER

                        POSTCONDITIONS
S11  Y - - - - - - - -  DOES MACHINE CODE EXIST ?
S12  - Y - - - - - - -  DOES MACHINE/PROGRAM CODE EXIST ?

NEXT 2 4 5 5 5 0 5 5 3

EXCP 1 2 5 5 5 0 5 5 5
```

Figure 12a. Executive Interpreter Program

```
MACHINE: ED  ABL INTERPRETER EDIT MACHINE
             PROGRAM: 01  ABL INTERACTIVE INTERPRETER- EDIT PROGRAM (LEVEL 01)

CLUSTER  1.0  GET ALTERNATIVE # FROM USER
              1.1  ( 1)  GET ALTERNATIVE # FROM USER
CLUSTER  2.0  PREPARE FOR INPUT OF ONE COLUMN (ALTERNATIVE)
              2.1  ( 2)  PREPARE FOR INPUT OF ONE COLUMN (ALTERNATIVE)
CLUSTER  3.0  ENTER THE FIVE DIFFERENT TYPES
              3.1  ( 3)  ENTER PRECONDITIONS
              3.2  ( 4)  ENTER ACTIONS
              3.3  ( 5)  ENTER POSTCONDITIONS
              3.4  ( 6)  ENTER NEXT STEP NUMBER
              3.5  ( 7)  ENTER EXCEPTION STEP NUMBER
              3.6  ( 8)  END OF INPUT OF THE ALTERNATIVE
CLUSTER  4.0  CHANGE (INCREMENT) TO NEXT INPUT TYPE
              4.1  ( 9)  CHANGE (INCREMENT) TO NEXT INPUT TYPE


     1 2 3 4 5 6 7 8 9   CLUSTERS / ALTERNATIVES
C1   V . . . . . . . .   GET ALTERNATIVE # FROM USER
C2   . V . . . . . . .   PREPARE FOR INPUT OF ONE COLUMN (ALTERNATIVE)
C3   . . V V V V V V .   ENTER THE FIVE DIFFERENT TYPES
C4   . . . . . . . . V   CHANGE (INCREMENT) TO NEXT INPUT TYPE


                         PRECONDITIONS
P1   - - Y - - - - - -   INPUT TYPE = 1 ?  (PRECONDITIONS)
P2   - - - Y - - - - -   INPUT TYPE = 2 ?  (ACTIONS)
P3   - - - - Y - - - -   INPUT TYPE = 3 ?  (POSTCONDITIONS)
P4   - - - - - Y - - -   INPUT TYPE = 4 ?  (NEXT STEP #)
P5   - - - - - - Y - -   INPUT TYPE = 5 ?  (EXCEPTION STEP #)
P6   - - - - - - - Y -   INPUT TYPE = 6 ?  (END OF INPUT)

                         ACTIONS
A1   1 . . . . . . . .   PRINT "ENTER ALTERNATIVE #"
A2   2 . . . . . . . .   GET ALTERNATIVE #
A3   . 1 . . . . . . .   DETERMINE CORRESPONDING COLUMN NUMBER IN TABLE & ON SCREEN
A4   . 2 . . . . . . .   START INPUT OF ALTERNATIVE
A5   . 3 . . . . . . .   INPUT ROW = 1
A6   . 4 . . . . . . .   INPUT TYPE = 1
A7   . . 1 . . . . . .   STORE PRECONDITION ENTERED
A12  . . 2 2 2 2 2 . .   INCREMENT INPUT ROW #  (ROW # = ROW # + 1)
A8   . . . 1 . . . . .   STORE ACTION
A9   . . . . 1 . . . .   STORE POSTCONDITION
A10  . . . . . 1 . . .   STORE NEXT STEP #
A14  . . . . . 3 3 . 1   INCREMENT INPUT TYPE  (TYPE = TYPE + 1)
A11  . . . . . . 1 . .   STORE EXCEPTION STEP #
A13  . . . . . . . 1 .   SAVE ALTERNATIVE

                         POSTCONDITIONS
S11  Y - - - - - - - -   DOES ALTERNATIVE EXIST ?
S12  - - N N N N N - -   END OF INPUT TYPE ? (' ' ENTERED, ADVANCE TO NEXT TYPE)

NEXT 2 3 3 3 3 3 3 0 3

EXCP 1 3 4 4 4 3 3 0 3
```

Figure 12b. Edit Program

```
MACHINE: PA  ABL SET PARAMETERS MACHINE
         PROGRAM: 01  ABL INTERACTIVE INTERPRETER- PARAMETERS PROGRAM (LEVEL 01)

CLUSTER  1.0  INITIALIZE, ASK USER FOR CHOICE
              1.1  ( 1)  INITIALIZE, ASK USER FOR CHOICE
CLUSTER  2.0  INPUT QUESTION RESPONSE
              2.1  ( 2)  SET TRACE
              2.2  ( 3)  SET BREAKPOINT
              2.3  ( 4)  CLEAR TRACE
              2.4  ( 5)  CLEAR BREAKPOINT
              2.5  ( 6)  QUIT (END)
              2.6  ( 7)  INVALID RESPONSE TO QUESTION
CLUSTER  3.0  SET TRACE
              3.1  ( 8)  SET TRACE
CLUSTER  4.0  SET BREAKPOINT
              4.1  ( 9)  SET, BREAKPOINT
CLUSTER  5.0  CLEAR TRACE
              5.1  (10)  CLEAR TRACE
CLUSTER  6.0  CLEAR BREAKPOINT
              6.1  (11)  CLEAR BREAKPOINT


     1 2 3 4 5 6 7 8 9 10 11   CLUSTERS / ALTERNATIVES
C1   V . . . . . . . . . . .   INITIALIZE, ASK USER FOR CHOICE
C2   . V V V V V V . . . .     INPUT QUESTION RESPONSE
C3   . . . . . . . V . . .     SET TRACE
C4   . . . . . . . . V . .     SET BREAKPOINT
C5   . . . . . . . . . V .     CLEAR TRACE
C6   . . . . . . . . . . V     CLEAR BREAKPOINT

                               PRECONDITIONS
P1   - Y - - - - N - - - -     SET 'T'RACE ON ?
P2   - - Y - - - N - - - -     SET 'B'REAKPOINT ON ?
P3   - - - Y - - N - - - -     'C'LEAR TRACE ?
P4   - - - - Y - N - - - -     C'L'EAR BREAKPOINT ?
P5   - - - - - Y N - - - -     'E'ND OF SETTING PARAMETERS ?

                               ACTIONS
A1   1 . . . . . . . . . .     INITIALIZE FOR SET PARAMETERS ROUTINE
A2   2 . . . . . . . . . .     PRINT "SET OR CLEAR TRACE OR BREAKPOINT ? [T/B/C/L/E]
A3   3 . . . . . . . . . .     GET RESPONSE FROM USER
A4   . 1 1 1 1 . . . . . .     ENTER ALTERNATIVE #
A5   . . . . . . 1 . . . .     SET TRACE FOR SPECIFIED ALTERNATIVE
A6   . . . . . . 2 . . . .     PRINT "T" IN ALTERNATIVE COLUMN ON SCREEN
A7   . . . . . . . 1 . . .     SET BREAKPOINT FOR SPECIFIED ALTERNATIVE
A8   . . . . . . . 2 . . .     PRINT "B" IN ALTERNATIVE COLUMN ON SCREEN
A9   . . . . . . . . 1 . .     CLEAR TRACE FOR SPECIFIED ALTERNATIVE
A10  . . . . . . . . 2 . .     ERASE "T" FROM ALTERNATIVE COLUMN ON SCREEN
A11  . . . . . . . . . 1 .     CLEAR BREAKPOINT FOR SPECIFIED ALTERNATIVE
A12  . . . . . . . . . 2 .     ERASE "B" FROM ALTERNATIVE COLUMN ON SCREEN

                               POSTCONDITIONS
S11  - Y Y Y Y - - - - - -     DOES ALTERNATIVE NUMBER EXIST ?

NEXT 2 3 4 5 6 0 1 1 1 1 1

EXCP 2 2 2 2 2 0 1 1 1 1 1
```

Figure 12c. Monitor Parameters Program

```
MACHINE: EX  ABL PROGRAM EXECUTION MACHINE
             PROGRAM: 01  ABL INTERACTIVE INTERPRETER- EXECUTE PROGRAM (LEVEL 01)

CLUSTER   1.0  INITIALIZE, START EXECUTION AT CLUSTER # 1
                  1.1  ( 1)  INITIALIZE, START EXECUTION AT CLUSTER # 1
CLUSTER   2.0  EVALUATE PRECONDITIONS FOR ALL ALTERNATIVES IN THE CLUSTER
                  2.1  ( 2)  EVALUATE PREDICATES FOR ALL ALTERNATIVES IN THE CLUSTER
CLUSTER   3.0  FETCH NEXT ACTION
                  3.1  ( 3)  FETCH NEXT ACTION
CLUSTER   4.0  EXECUTE REGULAR OR COMPOUND ACTION
                  4.1  ( 4)  EXECUTE COMPOUND ACTION
                  4.2  ( 5)  EXECUTE REGULAR ACTION
CLUSTER   5.0  EVALUATE POSTCONDITIONS
                  5.1  ( 6)  EVALUATE POSTCONDITIONS
CLUSTER   6.0  IF POSTCONDITIONS ARE SATISFIED, NEXT CLUSTER=NEXT STEP
                  6.1  ( 7)  IF POSTCONDITIONS ARE SATISFIED, NEXT CLUSTER = NEXT STEP
CLUSTER   7.0  IF POSTCONDITIONS NOT SATISFIED, NEXT CLUSTER=EXCEPTION STEP
                  7.1  ( 8)  IF POSTCONDITIONS NOT SATISFIED, NEXT CLUSTER=EXCEPTION STEP
CLUSTER   8.0  NO ALTERNATIVES SELECTED IN CLUSTER, TRAP ERROR
                  8.1  ( 9)  NO ALTERNATIVES SELECTED IN CLUSTER, TRAP ERROR
```

Figure 12d. continues...

MACHINE: EX ABL PROGRAM EXECUTION MACHINE
PROGRAM: 01 ABL INTERACTIVE INTERPRETER- EXECUTE PROGRAM (LEVEL 01)

	1	2	3	4	5	6	7	8	9	CLUSTERS / ALTERNATIVES
C1	V	.	.	.	.	.	.	.	.	INITIALIZE, START EXECUTION AT CLUSTER # 1
C2	.	V	.	.	.	.	.	.	.	EVALUATE PRECONDITIONS FOR ALL ALTERNATIVES IN THE CLUSTER
C3	.	.	V	.	.	.	.	.	.	FETCH NEXT ACTION
C4	.	.	.	V	V	.	.	.	.	EXECUTE REGULAR OR COMPOUND ACTION
C5	.	.	.	.	.	V	.	.	.	EVALUATE POSTCONDITIONS
C6	.	.	.	.	.	.	V	.	.	IF POSTCONDITIONS ARE SATISFIED, NEXT CLUSTER=NEXT STEP
C7	.	.	.	.	.	.	.	V	.	IF POSTCONDITIONS NOT SATISFIED, NEXT CLUSTER=EXCEPTION STEP
C8	.	.	.	.	.	.	.	.	V	NO ALTERNATIVES SELECTED IN CLUSTER, TRAP ERROR
										PRECONDITIONS
P1	-	-	-	Y	N	-	-	-	-	IS ACTION A COMPOUND ACTION ?
										ACTIONS
A1	1	.	.	.	.	.	.	.	.	CURRENT CLUSTER # = 1
A2	.	1	.	.	.	.	.	.	.	FOR ALL ALTERNATIVES IN CLUSTER, EVALUATE PRECONDITIONS
A3	.	2	.	.	.	.	.	.	.	IF IN TRACE MODE, PRINT PREDICATE # AND INPUT/OUTPUT VALUES
A4	.	3	.	.	.	.	.	.	.	SELECT THE ALTERNATIVE WITH ALL PRECONDITIONS SATISFIED
A5	.	4	.	.	.	.	.	.	.	CURRENT ACTION # = 1
A6	.	.	1	.	.	.	.	.	.	FETCH NEXT ACTION
A7	.	.	1	.	.	.	.	.	.	STORE (PUSH) STATE VECTOR ONTO STACK
A8	.	.	2	.	.	.	.	.	.	RECORD CALLING MACHINE/PROGRAM, ALTERNATIVE #
A9	.	.	3	.	.	.	.	.	.	TRANSFER CONTROL TO SUB-MODULE
A10	.	.	4	.	.	.	.	.	.	RETURN FROM SUB-MODULE, RESTORE (POP) STATE VECTOR
A13	.	.	5	3	.	.	.	.	.	IF TRACE MODE, PRINT ACTION & VARIABLES & VALUES
A14	.	.	6	4	.	.	.	.	.	INCREMENT ACTION # (ACTION # = ACTION # + 1)
A11	.	.	.	1	.	.	.	.	.	PARSE TO DECODE OP-CODE IN IMPLEMENTATION LANGUAGE
A12	.	.	.	2	.	.	.	.	.	PARSE OPERANDS & EXECUTE ACTION IN IMPLEMENTATION LANGUAGE
A15	.	.	.	.	1	.	.	.	.	EVALUATE POSTCONDITIONS FOR THE ALTERNATIVE
A16	.	.	.	.	.	1	.	.	.	CURRENT CLUSTER = NEXT STEP #
A18	.	.	.	.	.	2	2	.	.	IF BREAKPOINT, PAUSE FOR USER INTERACTION
A17	.	.	.	.	.	.	1	.	.	CURRENT CLUSTER = EXCEPTION STEP #
A19	.	.	.	.	.	.	.	.	1	PRINT ERROR MESSAGE, WAIT FOR USER, ALLOW CORRECTIVE ACTION
										POSTCONDITIONS
S11	-	Y	-	-	-	-	-	-	-	DOES ONE ALTERNATIVE HAVE ALL ITS PREDICATES SATISFIED ?
S12	-	-	Y	-	-	-	-	-	-	ANY MORE ACTIONS LEFT IN THIS ALTERNATIVE ?
S13	-	-	-	-	-	Y	-	-	-	ARE ALL POSTCONDITIONS SATISFIED FOR THIS ALTERNATIVE ?
S14	-	-	-	-	-	-	N	N	-	IS NEW CURRENT CLUSTER # = 0 ? (END-OF-MODULE ?)
S15	-	-	-	-	-	-	-	-	Y	DOES USER WISH TO RESUME EXECUTION ?
NEXT	2	3	4	3	3	6	2	2	2	
EXCP	2	8	5	3	3	7	0	0	0	

Figure 12d. Execute Program

Figure 13 is a photograph of the Video Screen while the interactive interpreter is executing the electronic calculator program.

In the photograph, the upper section shows descriptions of the program, current cluster and alternative. Due to screen limitations of 80 columns, there is a maximum of 20 alternatives per module. Above each alternative, a "T" or "B" code indicates that a trace or breakpoint has been set for that alterntiative. The codes "P", "A", "S", "N", and "E" in the body of the alternatives section correspond to precondition, action, postcondition, next step, and exception step, respectively. A pre- or postcondition followed by a minus sign (-) indicates a 'N'o value.

The current state in the photograph is the very beginning of execution of the electronic calculator program. The current alternative is 1.1, as indicated by the alternative description, and the arrows indicating column one. There were no preconditions to be evaluated, and the first action to be executed is number one. At the bottom of the screen, we can see the natural language description of the action (ACCUMULATOR=0), followed by the action number (1). Below this, is the corresponding executable code in the implementation language (LET ACC=0, in BASIC), and a partial state vector of all data objects affected by this action (VALUES: ACC = 0).

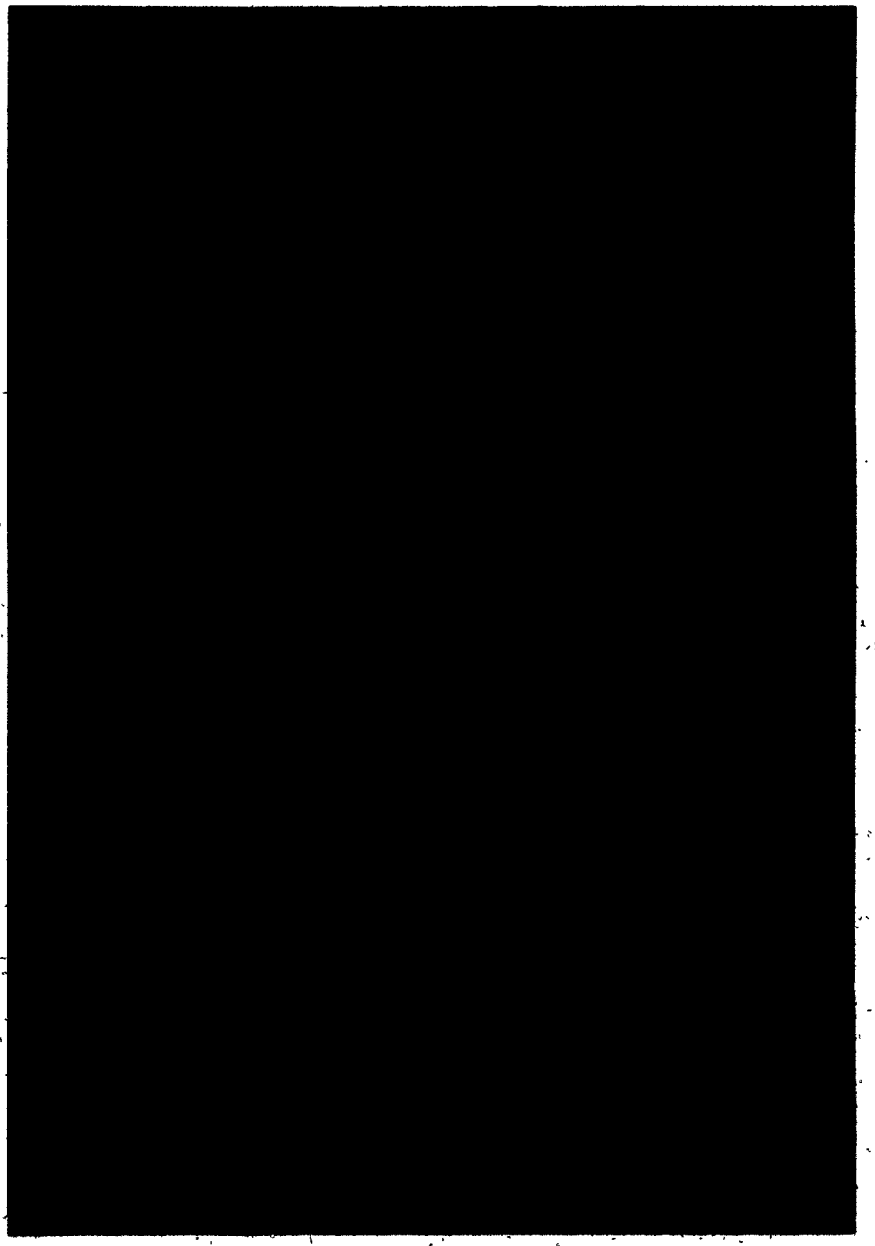

Figure 13. ABL Interactive Interpreter Screen

VI. APPLICATION: A MINI-COMPUTER BASED AUTOMATIC DIALING SYSTEM

"An Office Information System (OIS) is made up of a collection of highly interactive autonomous tasks that execute in parallel, ..., including forms, document preparation, and management, communication and decision-making aids" [ELLI80].

The Existing Environment. The application chosen to illustrate the ABL software environment is found in a medical office, such as a doctor's or dentist's office.

These offices keep patient history files on cards, forms, notes, memorized, and using other informal and quasi-formal methods of information storage.

Each office maintains an appointment book for scheduled visits during each working day. In the case of many dentists, as an example, appointments are frequently booked six months in advance; i.e. as the patient completes the current visit, an appointment is immediately made for the next six-month checkup.

Therefore, one of the duties of the office receptionist is to remind the patient and confirm the scheduled appoint-

ment, about a week to ten days in advance.

This confirmation process has benefit to the patient as a reminder mechanism, and also to the medical professional, to determine case loads well in advance for better planning. Any negatively confirmed appointments can be rescheduled, and the ensuing gaps can be filled with emergency or other short-notice work.

In order to perform these confirmation telephone calls, the receptionist must:

1) consult the appointment book for a given day of the following week, to retrieve relevant patient names.
2) "pull" relevant patient file cards with name, phone number, and last visit information on it.
3) call these patients. This includes delays such as busy lines, no answer, wrong numbers, etc., during which time the phone line is tied up.
4) record the confirmation in the appointment book as successful or unsuccessful.
5) refile all the patient cards and pull them out again next week, or keep several piles of cards in anticipation of next week's appointments.
6) after the visit is complete, record treatment, observations and the next appointment date and time, and refile the patient history cards.

Problem Definition. The design problem to be dealt with is to automate steps 1 through 5, above. Step 6, an ABL representation of therapeutic strategies, is currently being developed [WILL82].

The appointment confirmation module would be one of many in a complete medical office information system, and assumes the existence of the patient master file, and an automated "appointment book".

Other modules in the total system could include billing of work performed, collections, governmental (medicare) reporting, supplies inventory, and productivity analysis of personnel and equipment.

The objectives of automating these tasks are:

i) to eliminate much of the lost time in dialing, etc.

ii) to provide exception reports of unconfirmed appointments to the office manager or practitioner, so that action can be taken to prevent "no-shows"

iii) to maintain a central, up-to-date patient database, so that inquiries to a patients' medical or dental record can be handled quickly and easily. These inquiries can be from governments, insurance companies, other health-care professionals interested in knowing about past treatments, and pharmacists checking for possible dangerous conflicts in prescription medicines.

Specification of the Automatic Dialing System. The basic steps involved in the automated confirmation system are similar to the manual system described above, but with no paper-shuffling and far less effort.

1) To initiate the process, first set up the list of patients to be called. This is done by specifying the following parameters:

   STARTING DATE/TIME (eg. Monday, 09:00)

   ENDING DATE/TIME (eg. Tuesday, 17:00)

   UP TO n PHONE CALLS (eg. 100 calls)

   The system will then form the auto-dialer queue.

2) The computer will automatically dial the next number in the queue. If a connection is made, the appropriate patient information will appear on the Video Screen next to the telephone, and the receptionist will speak, and record the result of the conversation on the terminal. If it is an unsuccesful connection, then the call is placed into the retry queue, for a later attempt. Figure 14a is the detailed specification of this process. Figure 14b is a sub-program at the next lower level of abstraction. Its purpose is to select the next patient to be called.

```
MACHINE: DA  AUTOMATIC DIALER MACHINE- MINI-COMPUTER & SD-192 EPABX PHONE
             PROGRAM: 01  AUTO-DIALER DESIGN EXAMPLE  (LEVEL 00)

CLUSTER   1.0  SYSTEM STARTUP
                  1.1  ( 1)  SYSTEM STARTUP
CLUSTER   2.0  SELECT OFFICE OR HOME PHONE NUMBER
                  2.1  ( 2)  PREFERENCE IS TO CALL PATIENT'S OFFICE NUMBER
                  2.2  ( 3)  PREFERENCE IS TO CALL PATIENT'S HOME NUMBER
CLUSTER   3.0  DETERMINE THE STATUS OF THE TELEPHONE AFTER AUTO-DIALING
                  3.1  ( 4)  LINE IS BUSY
                  3.2  ( 5)  THERE IS NO ANSWER
                  3.3  ( 6)  A CONNECTION HAS BEEN MADE (SOMEONE ANSWERS THE PHONE)
CLUSTER   4.0  RECORD THE RESULT OF THE CONVERSATION
                  4.1  ( 7)  SUCCESSFUL CONFIRMATION OF THE APPOINTMENT
                  4.2  ( 8)  PATIENT WAS NOT REACHED, CONSIDERED AS 'NO ANSWER'
                  4.3  ( 9)  OTHER CHANGE IN THE PATIENT'S FILE IS NEEDED
CLUSTER   5.0  NO MORE PATIENTS TO CALL
                  5.1  (10)  BOTH QUEUES ARE EMPTY, ALL COMPLETE
                  5.2  (11)  RETRY QUEUE IS NOT EMPTY, PRINT IT FOR LATER REFERENCE

      1  2  3  4  5  6  7  8  9 10 11   CLUSTERS / ALTERNATIVES
C1    V  .  .  .  .  .  .  .  .  .  .    SYSTEM STARTUP
C2    .  V  V  .  .  .  .  .  .  .  .    SELECT OFFICE OR HOME PHONE NUMBER
C3    .  .  .  V  V  V  .  .  .  .  .    DETERMINE THE STATUS OF THE TELEPHONE AFTER AUTO-DIALING
C4    .  .  .  .  .  .  V  V  V  .  .    RECORD THE RESULT OF THE CONVERSATION
C5    .  .  .  .  .  .  .  .  .  V  V    NO MORE PATIENTS TO CALL

                                         PRECONDITIONS
P4    -  Y  N  -  -  -  -  -  -  -  -    SHOULD WE CALL THE PATIENT'S OFFICE PHONE NUMBER ?
P5    -  -  -  Y  N  N  -  -  -  -  -    IS THE LINE BUSY ?
P6    -  -  -  N  Y  N  -  -  -  -  -    IS THERE NO ANSWER (6 RINGS) ?
P7    -  -  -  N  N  Y  -  -  -  -  -    IS THERE A CONNECTION (SOMEONE AT OTHER END PICKS UP PHONE)?
P8    -  -  -  -  -  -  Y  N  N  -  -    HAVE WE HAD A SUCCESSFUL CONFIRMATION OF THE APPOINTMENT ?
P9    -  -  -  -  -  -  Y  N  -  -  -    WAS THE PATIENT HIMSELF CONTACTED ?
P10   -  -  -  -  -  -  N  N  Y  -  -    ARE THERE OTHER CHANGES IN THE PATIENT FILE NEEDED ?
P2    -  -  -  -  -  -  -  -  -  Y  Y    IS AUTO DIALER QUEUE EMPTY ?
P3    -  -  -  -  -  -  -  -  -  Y  N    IS RETRY QUEUE EMPTY ?

                                         ACTIONS
A1    1  .  .  .  .  .  .  .  .  .  .    INITIALIZE, STARTUP, ETC.
A2    2  .  .  3  3  .  3  3  3  .  .    ** GET-NEXT-PATIENT SUB-PROGRAM (02/DA) **
A5    .  1  .  .  .  .  .  .  .  .  .    SELECT OFFICE PHONE NUMBER TO CALL
A7    .  2  2  .  .  .  .  .  .  .  .    DIAL TELEPHONE NUMBER THROUGH EPABX PHONE SYSTEM
A6    .  .  1  .  .  .  .  .  .  .  .    SELECT HOME PHONE NUMBER TO CALL
A8    .  .  .  1  .  .  .  .  .  .  .    LAST RETRY TYPE = BUSY
A10   .  .  .  2  2  .  .  2  .  .  .    INCREMENT THE NUMBER OF RETRIES AND INSERT INTO RETRY QUEUE
A9    .  .  .  .  1  .  .  1  .  .  .    LAST RETRY TYPE = NO ANSWER
A12   .  .  .  .  .  1  .  .  .  .  .    RECEPTIONIST TALKS TO PATIENT (OR OTHER PERSON)
A13   .  .  .  .  .  .  1  .  .  .  .    RECORD DATE & TIME OF CONFIRMATION
A14   .  .  .  .  .  .  2  .  2  .  .    WRITE BACK THE UPDATED PATIENT RECORD
A15   .  .  .  .  .  .  .  .  1  .  .    CHANGE THE APPROPRIATE DATA IN THE PATIENT'S RECORD
A3    .  .  .  .  .  .  .  .  .  1  2    PRINT "COMPLETED" ON SCREEN
A4    .  .  .  .  .  .  .  .  .  .  1    PRINT CONTENTS OF RETRY QUEUE ON PRINTER FOR LATER REFERENCE

                                         POSTCONDITIONS
S11   Y  -  -  Y  Y  -  Y  Y  Y  -  -    ARE THERE ANY MORE PATIENTS TO CALL ?

NEXT  2  3  3  2  2  4  2  2  2  0  0

EXCP  5  3  3  5  5  4  5  5  5  0  0
```

Figure 14a. Automatic Dialing System for Appointment Confirmations

```
MACHINE: DA  AUTOMATIC DIALER MACHINE- MINI-COMPUTER & SD-192 EPABX PHONE
          PROGRAM: 02  AUTO-DIALER MODULE- GET-NEXT-PATIENT SUB-PROGRAM (LEVEL 01)

CLUSTER  1.0  GET NEXT ELEMENT IN RETRY QUEUE
              1.1  ( 1)  GET NEXT ELEMENT IN RETRY QUEUE
CLUSTER  2.0  GET THE CURRENT REAL TIME
              2.1  ( 2)  GET THE CURRENT REAL TIME
CLUSTER  3.0  GET THE NEXT ELEMENT IN THE AUTO-DIALER QUEUE
              3.1  ( 3)  GET THE NEXT ELEMENT IN THE AUTO-DIALER QUEUE
CLUSTER  4.0  TEST FOR ELIGIBLILITY OF RETRY
              4.1  ( 4)  LAST RETRY WAS BUSY- ELIGIBLE AGAIN
              4.2  ( 5)  LAST RETRY WAS NO ANSWER- ELIGIBLE
              4.3  ( 6)  INSUFFICIENT TIME HAS ELAPSED- INELIGIBLE FOR RETRY
CLUSTER  5.0  CHOOSE THIS PATIENT FROM AUTO-DIALER QUEUE
              5.1  ( 7)  CHOOSE THIS PATIENT
CLUSTER  6.0  NO MORE PATIENTS TO CALL
              6.1  ( 8)  NO MORE PATIENTS TO CALL

     1 2 3 4 5 6 7 8   CLUSTERS / ALTERNATIVES
C1   V . . . . . . .   GET NEXT ELEMENT IN RETRY QUEUE
C2   . V . . . . . .   GET THE CURRENT REAL TIME
C3   . . V . . . . .   GET THE NEXT ELEMENT IN THE AUTO-DIALER QUEUE
C4   . . . V V V . .   TEST FOR ELIGIBLILITY OF RETRY
C5   . . . . . . V .   CHOOSE THIS PATIENT FROM AUTO-DIALER QUEUE
C6   . . . . . . . V   NO MORE PATIENTS TO CALL

                       PRECONDITIONS
P20  - - - Y N N - -   IF THE LAST RETRY WAS BUSY- HAS 5 MINUTES ELAPSED ?
P21  - - - N Y N - -   IF THE LAST RETRY WAS NO ANSWER- HAS 1 HOUR ELAPSED ?

                       ACTIONS
A20  1 . . . . . . .   FOLLOW THE RETRY QUEUE POINTER TO THE NEXT NODE
A21  . 1 . . . . . .   GET THE CURRENT REAL TIME
A24  . . 1 . . . . .   FOLLOW THE AUTO-DIALER QUEUE POINTER TO THE NEXT NODE
A22  . . . 1 1 . 1 .   CHOOSE THIS PATIENT TO CALL NEXT
A23  . . . 2 2 . 2 .   READ PATIENT MASTER RECORD
A25  . . . . . . . 1   NO MORE PATIENTS TO CALL

                       POSTCONDITIONS
S3   N - - - - - - -   IS RETRY QUEUE EMPTY ?
S2   - - N - - - - -   IS AUTO DIALER QUEUE EMPTY ?

NEXT 2 4 5 0 0 1 0 0

EXCP 3 4 6 0 0 1 0 0
```

Figure 14b. Get-Next-Patient Sub-program for Automatic Dialing System

Host System for Implementation. This automatic dialing system necessitates the use of three different electronic devices to implement the design.

The host mini-computer system described in Part V has sufficient storage to handle the patient files for a medium-to-large sized medical office.

Through an RS232C interface, the appropriate control signals and data (phone number) are sent to an automatic dialer, such as the Racal-Vadic VA811 Multimode Automatic Calling Unit (ACU) [RACA81], which will return the status of the connection, successful or otherwise.

This ACU must be interfaced with the telephone equipment itself. Since the recent Canadian Radio and Television Commission ruling, the "interconnect" phone system market has flourished. One such system which is very adaptable to this type of interface is the Siemens SD-192 Electronic Private Automatic Branch Exchange (EPABX) [SIEM78].

It must be emphasized that whatever hardware is used, the software design is completely machine transportable, one of the major features of the ABL approach.

VII. CONCLUSION

Summary and Conclusion. In evaluating how well the implemented software environment meets its original objectives, it has been shown that modularization and structured techniques are strongly encouraged, along with a design method that is more suited towards the human approach, and not constrained by the implementation-dependent sequential method.

The ABL tool, at the center of the environment, is highly communicative with its users, and therefore, the label of "user friendly" applies.

The software environment does furnish the facility to segregate the control-flows from the data-flows. It provides powerful dynamic and static testing and analysis tools, and lends general clarity by breaking down difficult problems into their individual components.

The last stage in the software development process described within is to measure the performance and evaluate the software environment.

The field of software metrics is still in its infancy,

as so many of the properties are difficult to quantify. It has been seen that the efficiency, modifiability, transportability, and quality of documentation are all enhanced by the software environment, however intangible these characteristics may be. This has proven effective in improving the performance of the software design and programming examples included herein.

The performance of the ABL interactive interpreter was not measured formally. The degradation during execution was certainly acceptable, considering that the abstract programs are interpreted into BASIC, which is itself interpreted.

In conclusion, ABL is a useful and important tool in the software development process, in a minicomputer environment.

Future Research. Future developments to the software environment could include further testing aids mentioned earlier, such as symbolic execution analysis, data-flow analysis and reverse execution of programs.

The ability to accomodate concurrent processing has been designed into the software environment, along with the possiblity of parallel execution. A prime location for this parallelism would be in the evaluation of preconditions for

all alternatives in a cluster, particularly since many of the precondition resultants are considered in more than one alternative.

This would call for the design of a specialized host machine architecture, one that could still be implemented on a mini-computer.

Observations. The intent of this thesis was to develop a software design tool along with an application that would have relevance and utility in a practical environment.

The ABL software environment and the design example of the automatic dialing system are currently under study at Tyme Systems Ltd., for possible application in industry.

The foundation for the ABL software environment was initially developed as far back as 1970 [JAW070].

Only now, in 1983, is ABL at the point of serious consideration by industry, now that the project has reached fruition.

This ABL development experience has shown that any software tool development is a long process.

It is hoped that these developments will encourage more research into the methods of augmenting automated design

tools, in keeping with the applicable software environments.

This can lead to advancements in both the quality and quantity of designs and programs developed by software professionals.

```
PN=ABLA-06A   SAVED UNDER ABLA-06A          MAR 21,82   14:36   PAGE 1

                    "ABLA-06A" ABL INTERACTIVE INTERPRETER- PREPARE SCREEN

0010 REM "ABLA-06A" ABL INTERACTIVE INTERPRETER- PREPARE SCREEN
0020                                        MURRAY KRONICK 7428037
0022                                        THESIS WORK- MASTER OF COMPUTER SCIENCE
0024                                        CONCORDIA UNIVERSITY
0026                                        PROPRIETARY INFORMATION
0028                                        WRITTEN 26/12/81
0029                                        REVISED
0100 SHORT V
0110 LET V=0
1000 GOSUB 9000
1010 OPEN FILE (1,5),"ABLMACHI"
1020 OPEN FILE (2,6),"DUMMY"
1030 OPEN FILE (3,5),"ABLMLIBR"
1090 CHAIN "ABLA-06B",V

6990                                        DATA FOR CRT SETUP
7000 DATA 19,0,"*** ABL INTERACTIVE INTERPRETER ***"
7005 DATA 0,1,"PROGRAM      MACHINE"
7010 DATA 2,2,"1. "
7015 DATA 2,3,"2. "
7025 DATA 4,4,"CLUSTER:    .0"
7030 DATA 0,5,"ALTERNATIVE: "
7045 DATA 1,7,"1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18  19  20"
7995 DATA 99
9000 ON ERROR GOTO 9020
9002 DIM X1,Y1,Z$(80)
9004 INHIBIT
9006 PRINT CRT(1),CRT(4);
9008 READ X1
9010 IF X1=99 THEN GOTO 9020
9012 READ Y1,Z$
9014 IF X1<0 THEN LET X1=1-X1-LEN(Z$)
9016 PRINT PC(X1,Y1);Z$;
9018 GOTO 9008

9020 ON ERROR STOP
9022 PRINT CRT(3);
9024 ALLOW
9026 RETURN

                    TOTAL # LINES OF TEXT =  37
                  TOTAL # CHARACTERS OF TEXT =  983
                TOTAL # CHARACTERS IN 'REM' =  272
                            # LINES PRINTED =  37
```

```
PN=ABLA-06B   'SAVED UNDER ABLA-06B          MAR 21,82   14:36     PAGE 1

                    "ABLA-06B" ABL INTERACTIVE INTERPRETER- LOAD & EDIT PROGRAM, SET PARAMETERS

0010 REM "ABLA-06B" ABE INTERACTIVE INTERPRETER- LOAD & EDIT PROGRAM, SET PARAMETERS
0020                                        MURRAY KRONICK  7428057
0022                                        THESIS WORK- MASTER OF COMPUTER SCIENCE
0024                                        CONCORDIA UNIVERSITY
0026                                        PROPRIETARY INFORMATION
0028                                        WRITTEN 26/12/81
0029                                        REVISED
0030
0049                                        --- CONSTANTS ---
0050 LET R=30
0055 LET C=20
0099                                        --- LOCAL VARIABLES ---
0101                                        C9  CURRENT CLUSTER NUMBER
0102                                        R,C MAXIMUM NUMBER OF ROWS, COLUMNS IN T TABLE
0104                                        T   COMPACTED ABSTRACT PROGRAM TABLE (SEE FILE LAYOUT ABLTXXYY)
0106                                        I,J LOOP CONTROL
0108                                        D   DISPLACEMENT IN T TABLE (USED TO TRANSFER TO & FROM M7 IN FILE)
0110                                        P   POSITION IN T TABLE (COLUMN NUMBER)  (0..C-1)
0112                                        T1  TYPE OF INPUT  (1=PRECONDITION   2=ACTION)
0113                                                           (3=POSTCONDITION  4=NEXT STEP#  5=EXCEPTION STEP #)
0114                                        T1$ TYPE OF INPUT CODES [P/A/S/N/E]
0120                                        B1$ BREAKPOINT VECTOR [Y/N]
0122                                        B2$ TRACE VECTOR [Y/N]
0124                                        V   =0 COMING FROM SCREEN PHASE, =1 COMING FROM EXECUTE PHASE
0150 DIM T1$(5),B1$(20),B2$(20),B3$(1)
0160 SHORT V,C9,T(R-1,C-1),S
0170 LONG I,J,D,P,T1
0200 LET T1$="PASNE"
0299                                        --- FILE VARIABLES ---
0300 DIM M$(4),M1$(94),M2$(2),M3$(60),M5$(70),M6(4)
0320 SHORT M1,M2(10),M3,M4(9),M5(10),M7(9,19)
0910 LET M5$=" ",M5$
0920 LET M3$=" ",M3$
0900 GOSUB 9500
0920 SCOM V
0930 IF V THEN COM V,B1$,B2$,B3$,S
0940 IF V THEN IF B3$="Y" THEN CLOSE FILE (15)
0950 IF V THEN GOTO 1400
```

"ABLA-068" ABL INTERACTIVE INTERPRETER- LOAD & EDIT PROGRAM, SET PARAMETERS

```
0999                                        *** MAIN PROGRAM ***
1000 PRINT CRT(2);
1002 ON ERROR STOP
1004 ON ERROR GOTO 1008
1005 CLOSE FILE (5)
1006 CLOSE FILE (6)
1008 ON ERROR STOP
1009                                        GET PROGRAM NUMBER
1010 LET X1=9
1015 LET X2=1
1020 LET X3=2
1030 GOSUB 9540
1040 ON Z2 THEN GOTO 1030,2050,1030,1030
1045 IF Z1<2 THEN GOTO 1030
1050 LET M5$(3,4)=Z$
1099                                        GET MACHINE NUMBER
1100 ON ERROR STOP
1110 LET X1=21
1130 GOSUB 9540
1140 ON Z2 THEN GOTO 1130,1000,1160,1130
1150 LET M5$(1,2)=Z$
1160 LET M$=M5$(1,2),"  "
1165 LET M1=0
1170 ON ERROR GOTO 1100
1175 SEARCH INDEX (1,2),M$,M1 GIVING M1$,M2
1180 ON ERROR STOP
1190 PRINT PC(21,1);M$(1,3);M1$(1,55);
1199                                        SEE IF PROGRAM ALREADY EXISTS
1200 LET Z$="ABLP".M5$(1,4)
1210 ON ERROR GOTO 1000
1220 OPEN FILE (5,3),Z$
1230 LET Z$(1,4)="ABLT"
1240 OPEN FILE (6,512),Z$
1260 ON ERROR STOP
1299                                        PROGRAM EXISTS, PRINT FIRST TWO LINES OF DESCRIPTION
1300 LET M2$="  "
1310 FOR M3=1 TO 2
1330  ON ERROR GOTO 1370
1340  SEARCH INDEX (5,6),M2$,M3 GIVING M3$
1360  PRINT PC(1,M3+1);M3;".  ";M3$;
1370  ON ERROR STOP
1380 NEXT M3
1399                                        READ IN STORED COMPACTED TABLE INTO T
1400 FOR Z=1 TO 3
1410  READ FILE (6,Z),M5,M5$,M6,M7$
1420  LET D=(Z-1)*10
1430  FOR I=0 TO 9
1440    FOR J=0 TO C-1
1450      LET T(I+D,J)=M7(I,J)
1460    NEXT J
1470  NEXT I
1480 NEXT Z
```

"ABLA-06B" ABL INTERACTIVE INTERPRETER- LOAD & EDIT PROGRAM, SET PARAMETERS

```
1500 IF V THEN GOTO 2000
1510 LET B1$="N",B1$
1520 LET B2$="N",B2$
1530 LET S=0
1540 GOSUB 6000
                        EDIT [E],SET PARAMETERS [P],EXECUTE [X],SAVE [S],OR QUIT [Q]
2000 LET Z=103
2010 GOSUB 9800
2030 ON -Z THEN GOTO 3000,7000,2500,2100,2050
2050 CHAIN "TYU-810"

2099                                    SAVE PROGRAM
2100 FOR Z=1 TO 3
2110  LET M5(1)=Z
2120  LET D=(Z-1)*10
2130  FOR I=0 TO 9
2140    FOR J=0 TO C-1
2150      LET M7(I,J)=T(I+D,J)
2160    NEXT J
2170  NEXT I
2180  WRITE FILE (6,Z),M5,M5$,M6,M7
2190 NEXT Z
2200 GOTO 1000

2499                                --- CHAIN TO E'X'ECUTE MODULE ---
                        TRACE OUTPUT TO PRINTER ? [Y/N]  ^YN
2500 LET Z=125
2510 GOSUB 9800
2520 LET B3$="N"
2530 ON -Z THEN GOTO 2550,2600
2550 LET B3$="Y"
2560 OPEN FILE (15,1),"$SYO"
2600 CHAIN "ABLA-06C",B1$,B2$,B3$,S
```

"ABLA-068" ABL INTERACTIVE INTERPRETER- LOAD & EDIT PROGRAM, SET PARAMETERS

```
2998                                        !--- EDIT MODULE ---
2999                                        GET CLUSTER NUMBER
3000 ON ERROR STOP
3010 LET X1=13
3015 LET X2=4
3020 LET X3=2
3030 GOSUB 9620
3040 ON Z2 THEN GOTO 3030,2000,3030,3100
3050 LET M3,C9=Z
3060 ON ERROR GOTO 3000
3070 SEARCH INDEX (5,6),"A ",M3 GIVING M3$,M4
3080 ON ERROR STOP
3090 GOTO 3115
3099                                        GENERATE NEXT CLUSTER NUMBER
3100 LET M5(2)=M5(2)+1
3110 LET C9,M3=M5(2)
3115 PRINT PC(13,4);
3117 PRINT USING "##.#",M3;
3119                                        GET CLUSTER DESCRIPTION
3120 LET X1=19
3130 LET X3=60
3140 GOSUB 9540
3150 ON Z2 THEN GOTO 3140,3170,3170,3140
3160 LET M3$=Z$
3170 ON ERROR GOTO 3180
3175 DELETE INDEX (5,6),"A ",M3
3180 ON ERROR STOP
3190 INSERT INDEX (5,6),"A ",M3 USING M3$,M4
3200 PRINT PC(19,4);M3$;
3499                                        GET ALTERNATIVE NUMBER
3500 LET X1=13
3510 LET X2=5
3520 LET X3=4
3530 GOSUB 9600
3540 ON Z2 THEN GOTO 3530,3000,3530,3530
3550 LET Y9,Z3=1
3560 GOSUB 9720
3565 IF Y9=0 THEN GOTO 3500
3570 IF INT(Z/10)><C9 THEN GOTO 3500
3575 IF INT(Z/10)=Z/10 THEN GOTO 3500
3580 LET M3=Z
3590 FOR I=0 TO 9
3592  LET M4(I)=0
3594 NEXT I
3600 ON ERROR GOTO 3620
3610 SEARCH INDEX (5,6),"B ",M3 GIVING M3$,M4
3620 ON ERROR STOP
3625 PRINT PC(13,5);
3627 PRINT USING "##.#",M3/10;
3699                                        GET ALTERNATIVE DESCRIPTION
3700 LET X1=19
```

"ABLA-06B" ABL INTERACTIVE INTERPRETER- LOAD & EDIT PROGRAM, SET PARAMETERS

```
3710 LET X3=60
3730 GOSUB 9540
3740 ON Z2 THEN GOTO 3730,3500,3760,3730
3750 LET M3$=Z$
3760 PRINT PC(19,5);M3$;
3900 ON ERROR GOTO 3920
3910 DELETE INDEX (5,6),"B ",M3
3920 ON ERROR STOP
3930 INSERT INDEX (5,6),"B ",M3 USING M3$;M4
3998                                    --- GET ONE ALTERNATIVE (ONE COMPLETE COLUMN) ---
3999                                    SCAN ALTERNATIVE NUMBERS, DETERMINE P (COLUMN IN T TABLE)
4000 FOR P=0 TO M5(3)
4020  IF T(0,P)=0 THEN GOTO 4120
4040  IF T(0,P)=M3 THEN GOTO 4200
4060  IF T(0,P)>M3 THEN GOTO 4100
4080 NEXT P
4100 GOSUB 5000
4120 LET M3(3)=M5(3)+1
4130 LET T(0,P)=M3
4160 CHANGE M3/10 TO Z$ USING "#.#"
4170 PRINT PC(P*4,8);Z$;
4199                                    INPUT COLUMN P (ALTERNATIVE)
4200 LET X1=P*4
4220 LET X2=9
4240 LET X3=3
4260 LET T1=1
4300 FOR I=1 TO R-1
4320  ON T1 THEN GOSUB 9610,9620,9610,9620,9620
4340  ON Z2+1 THEN GOTO 4350,4320,4345,4345,4500
4345  LET Z=T(I,P)
4350  IF Z=0 THEN IF T1<4 THEN GOTO 4500
4355  LET T(I,P)=Z
4360  CHANGE Z TO Z$ USING " ##-"
4370  LET Z$(1,1)=T1$(T1,T1)
4380  PRINT PC(X1,X2);Z$;
4385  IF T1=5 THEN GOTO 4700
4390  IF Z2=2 THEN GOTO 4700
4400  LET X2=X2+1
4410  IF X2=22 THEN LET X2=9
4415  IF T1=4 THEN GOTO 4540
4420  GOTO 4680
4500  IF T1=>4 THEN GOTO 4320
4520  LET T(I,P)=Z
4540  LET T1=T1+1
4680 NEXT I
4700 GOTO 3500

4999                                    SHIFT T TABLE ONE COLUMN TO THE RIGHT TO INSERT AN ALTERNATIVE
5000 FOR J=M5(3) TO P+1 STEP -1
5020  LET T1=1
5040  LET X1=J*4
5060  LET X2=8
```

"ABLA-068" ABL INTERACTIVE INTERPRETER- LOAD & EDIT PROGRAM, SET PARAMETERS

```
5100  FOR I=0 TO R-1
5120    LET T(I,J)=T(I,J-1)
5130    IF X2=22 THEN GOTO 5190
5135    IF T1>5 THEN GOTO 5190
5140    IF T(I,J)=0 THEN IF T1<4 THEN GOTO 5180
5150    CHANGE T(I,J) TO Z$ USING " ##-"
5155    LET Z$(1,1)=T1$(T1,T1)
5160    IF I=0 THEN CHANGE T(I,J)/10 TO Z$ USING "#.#"
5165    PRINT PC(X1,X2);Z$;
5170    LET X2=X2+1
5175    IF T1<4 THEN GOTO 5190
5180    LET T1=T1+1
5190  NEXT I
5200  IF X2=22 THEN GOTO 5250
5210  FOR Z=X2 TO 22
5220    PRINT PC(X1,Z);"    ";
5230  NEXT Z
5250 NEXT J
5299                                    ZERO OUT INSERTED COLUMN
5300 FOR I=0 TO R-1
5310  LET T(I,P)=0
5320 NEXT I
5350 LET Z=P+4
5360 FOR I=8 TO 22
5370  PRINT PC(Z,I);"    ";
5380 NEXT I
5390 RETURN

5999                                    PRINT EXISTING ALTERNATIVES
6000 FOR J=0 TO M5(3)-1
6020  LET T1=1
6040  LET X1=J+4
6060  LET X2=8
6100  FOR I=0 TO R-1
6120    IF X2=22 THEN GOTO 6260
6130    IF T1>5 THEN GOTO 6260
6140    IF T(I,J)=0 THEN IF T1<4 THEN GOTO 6240
6160    CHANGE T(I,J) TO Z$ USING " ##-"
6180    LET Z$(1,1)=T1$(T1,T1)
6185    IF I=0 THEN CHANGE T(I,J)/10 TO Z$ USING "#.#"
6190    PRINT PC(X1,X2);Z$;
6200    LET X2=X2+1
6220    IF T1<4 THEN GOTO 6260
6240    LET T1=T1+1
6260  NEXT I
6280 NEXT J
6290 RETURN
```

"ABLA-068" ABL INTERACTIVE INTERPRETER- LOAD & EDIT PROGRAM, SET PARAMETERS

```
6998                                    --- SET PARAMETERS MODULE ---
                                SET/CLEAR TRACE [T/C], SET/CLEAR BREAKPOINT [B/L] OR END [E]
7000 LET Z=104
7020 GOSUB 9800
7040 ON -Z THEN GOTO 7100,7200,7300,7400,2000
7099                                    SET TRACE
7100 GOSUB 7500
7110 LET B2$(P,P)="Y"
7120 PRINT PC(P*4-4,6);"T";
7150 GOTO 7000

7199                                    CLEAR TRACE
7200 GOSUB 7500
7210 LET B2$(P,P)="N"
7220 PRINT PC(P*4-4,6);" ";
7250 GOTO 7000

7299                                    SET BREAKPOINT
7300 GOSUB 7500
7310 LET B1$(P,P)="Y"
7320 PRINT PC(P*4-2,6);"B";
7350 GOTO 7000

7399                                    CLEAR BREAKPOINT
7400 GOSUB 7500
7410 LET B1$(P,P)="N"
7420 PRINT PC(P*4-2,6);" ";
7450 GOTO 7000

7499                                    GET ALTERNATIVE NUMBER TO SET PARAMETER
7500 PRINT PC(2,23);"ENTER ALTERNATIVE NUMBER";
7510 LET X1=30
7520 LET X2=23
7525 LET X3=4
7530 GOSUB 9600
7540 ON Z2 THEN GOTO 7530,7530,7530,7530
7550 LET Y9,Z3=1
7560 GOSUB 9720
7570 IF Y9=0 THEN GOTO 7530
7599                                    DETERMINE P (COLUMN IN T TABLE)
7600 LET P=0
7610 FOR I=0 TO M5(3)-1
7620   IF T(0,I)=Z THEN LET P=I+1
7630 NEXT I
7640 IF P=0 THEN GOTO 7530
7670 PRINT PC(0,23);CRT(5);CRT(6);
7680 RETURN
```

"ABLA-06B" ABL INTERACTIVE INTERPRETER- LOAD & EDIT PROGRAM, SET PARAMETERS

TOTAL # LINES OF TEXT = 309

TOTAL # CHARACTERS OF TEXT = 7857

TOTAL # CHARACTERS IN 'REM' = 1995

LINES PRINTED = 309

"ABLA-06C" ABL INTERACTIVE INTERPRETER- EXECUTE MODULE

```
0010 REM "ABLA-06C" ABL INTERACTIVE INTERPRETER- EXECUTE MODULE
0020                                    MURRAY KRONICK  7428057
0021                                    THESIS WORK- MASTER OF COMPUTER SCIENCE
0024                                    PROPRIETARY INFORMATION
0026                                    WRITTEN 26/12/81
0028                                    REVISED
0049                                    --- CONSTANTS ---
0050 LET R=30
0055 LET C=20
0099                                    --- LOCAL VARIABLES ---
0100                                    A1,A2,A3 COLUMN # FOR ALTERNATIVES
0101                                    C9  CURRENT CLUSTER NUMBER
0102                                    R,C MAXIMUM NUMBER OF ROWS, COLUMNS IN T TABLE
0104                                    T   COMPACTED ABSTRACT PROGRAM TABLE (SEE FILE LAYOUT ABLTXXYY)
0107                                    P,P1,L,L1 CHARACTER POINTERS IN P$
0108                                    P$  INSTRUCTION TO PARSE IN HOST LANGUAGE
0109                                    C8$ INSTRUCTION MNEMONICS IN HOST LANGUAGE
0110                                    P   POSITION IN T TABLE (COLUMN NUMBER)  (0..C-1)
0111                                    P9  PREDICATE VALUE 0=FALSE, 1=TRUE
0113                                    R2  CURRENT ROW # IN T TABLE
0114                                    T9,T9$ VALUE OF SECOND VARIABLE IN EXPRESSION
0115                                    W   LENGTH OF ALPHA VARIABLE
0116                                    W$  CURRENT VARIABLE NAME
0117                                    W1,W2 SUBSCRIPT IN ARRAYS OF VARIABLES V$,V5$
0119                                    B$  BLANKS
0120                                    B1$ BREAKPOINT VECTOR [Y/N]
0122                                    B2$ TRACE VECTOR [Y/N]
0123                                    B3$ PRINT FLAG FOR TRACE
0125                                    S   STACK POINTER (FOR COMPOUND ACTIONS)
0140 ON ERROR GOTO 8790
0150 DIM B$(40),C8$(40*4),P$(30),T1$(5),T9$(30),W$(3),Z$(128),Z2$(5)
0160 SHORT A1,A2,A3,C9,L,L1,P,P1,P9,S,T(R-1,C-1),V,W,W1,W2,R2,T1,Y9(1),Z3,Z4
0170 LONG I,J,T9,X,X1,X2,Z
0200 LET C8$(1,80)="ACC\CHG\CHN\CLF\CLO\COM\DAT\DIM\DLX\DO \FOR\FUN\GOS\GOT\IF \INX\LET\LON\NXT\ONB\"
0210 LET C8$(81,160)="ONG\ONR\ONS\OPF\PRF\PRT\RD \RDF\RET\SCF\SFX\SHT\SLF\SNX\SRX\WRF\   \   \   \   \"
0220 LET B$=" ",B$
0299                                    --- FILE VARIABLES ---
0300 DIM M1$(94),M5$(70),M6(4)
0320 SHORT M2(10),M3(10),M7(9,19)
0330 DIM V$(60),V2$(372),V5$(120),V8$(4),V9$(23),B1$(20),B2$(20),B3$(1)
0340 SHORT V1(19),V2(19),V8(1)
0350 LONG V5(39)
0400 LET V2$=" ",V2$
0410 LET V$=" ",V$
0420 LET V5$=" ",V5$
0900 COM B1$,B2$,B3$,S
```

"ABLA-06C" ABL INTERACTIVE INTERPRETER- EXECUTE MODULE

```
0998                                        *** MAIN PROGRAM ***
1000 IF S=0 THEN GOSUB 2000
1020 IF S THEN GOSUB 6500
1050 IF S THEN GOTO 1020
1500 LET V=1
1520 CHAIN "ABLA-06B",V,B1$,B2$,B3$,S

1999                                        COMMENCE EXECUTION AT CLUSTER NUMBER 1
2000 GOSUB 2900
2020 LET C9=1
2099                                        DETERMINE ALTERNATIVES BELONGING TO CURRENT CLUSTER
2100 LET A1,A2,A3=-1
2110 IF C9=0 THEN RETURN
2120 FOR I=0 TO M5(3)-1
2130  IF INT(T(0,I)/10)><C9 THEN GOTO 2160
2140  IF A1=-1 THEN LET A1=I
2150  LET A2=I
2160 NEXT I
2170 IF A1=-1 THEN GOTO 2285
2199                                        EVALUATE PRECONDITIONS TO SELECT AN ALTERNATIVE
2200 FOR J=A1 TO A2
2210  LET R2=1
2220  GOSUB 7000
2240  IF P9=0 THEN GOTO 2270
2250  LET A3=J
2260  LET J=100
2270 NEXT J
2280 IF A3><-1 THEN GOTO 2400
                    NO ALTERNATIVES SATISFIED IN CLUSTER.  EXECUTION SUSPENDED.
2285 LET C9=0
2290 GOSUB 3800
2295 RETURN

2399                                        FETCH FIRST ACTION
2400 FOR I=1 TO R-1
2410  IF T(I,A3) THEN GOTO 2450
2420  LET R2=I+1
2430  LET I=100
2450 NEXT I
2499                                        ANY MORE ACTIONS ?
2500 IF T(R2,A3)=0 THEN GOTO 2700
2550 SEARCH INDEX (1,2),M5$(1,2),"B ",T(R2,A3) GIVING M1$,M2
2560 LET P$=M1$(61,90)
2570 IF B2$(A3+1,A3+1)="N" THEN GOTO 2600
2580 PRINT PC(1,21);M1$(1,60);PC(1,22);P$;
2590 IF B3$="Y" THEN PRINT FILE (15),"EXECUTING ACTION",M1$
2599                                        IS ACTION REGULAR OR COMPOUND ?
2600 IF M1$(91,94)><"    " THEN GOTO 6000
2620 GOTO 3000
```

"ABLA-06C" ABL INTERACTIVE INTERPRETER- EXECUTE MODULE

```
2699                                        EVALUATE POSTCONDITIONS
2700 LET J=A3
2710 LET R2=R2+1
2720 GOSUB 7000
2749                                        NEXT CLUSTER IS NEXT STEP # IF POSTCONDITIONS WERE SATISFIED
2750 IF P9=1 THEN LET C9=T(R2+1,A3)
2760 IF P9=0 THEN LET C9=T(R2+2,A3)
2799                                        IF BREAKPOINT SET, PAUSE FOR USER INTERACTION
2800 IF B1$(A3+1,A3+1)="N" THEN GOTO 2100
                    BREAKPOINT ENCOUNTERED. 'RETURN' TO CONTINUE, '.' TO STOP.
2810 LET Z=120
2815 GOSUB 9800
2820 ON -Z THEN GOTO 2100,1500
2899                                        READ IN STORED COMPACTED TABLE INTO T
2900 FOR Z=1 TO 3
2910  READ FILE (6,Z),M5,M5$,M6,M7
2920 'LET X=(Z-1)*10
2930  FOR I=0 TO 9
2940    FOR J=0 TO C-1
2950      LET T(I+X,J)=M7(I,J)
2960    NEXT J
2970  NEXT I
2980 NEXT Z
2990 RETURN
2995                                        *** END OF MAIN PROGRAM ***
```

"ABLA-06C" ABL INTERACTIVE INTERPRETER- EXECUTE MODULE

```
2998                                                  ---REGULAR ACTION---
2999                                                  PARSE ACTION OP-CODE IN HOST LANGUAGE
3000 LET L=(STR(CB$,P$(1,3))+3)/4
3020 ON L THEN GOTO 3200,3350,3800,3500,3520,3800,3800,3800,5300,3340
3030 ON L-10 THEN GOTO 3800,3800,3800,3800,3800,5300,4000,3800,3800,3800
3040 ON L-20 THEN GOTO 3800,3800,3800,3550,3650,3700,3800,5100,3800,5050
3050 ON L-30 THEN GOTO 5400,3800,5000,5300,5300,5100
3060 GOTO 3800

3199                                                  'ACCEPT'
3200 LET P=STR(P$,",",5)
3210 CHANGE P$(4,P-1) TO W
3220 ACCEPT (W),Z$
3230 LET W$=P$(P+1)
3240 IF W$="   " THEN GOTO 3810
3250 IF STR(W$,"$")=0 THEN GOTO 3300
3260 GOSUB 8000
3270 LET V2$(V1(W1),V2(W1))=Z$,B$
3280 GOTO 3900

3300 GOSUB 8200
3310 CHANGE Z$ TO V5(W1)
3320 GOTO 3900

3339                                                  'DO'
3340 DO P$(4)
3345 GOTO 3900

3349                                                  'CHANGE'
3350 LET P=STR(P$,",",")
3360 LET W$=P$(P-4)
3370 IF W$="   " THEN GOTO 3810
3380 IF STR(W$,"$") THEN GOTO 3450
3390 GOSUB 8200
3400 LET W2=W1
3405 LET W$=P$(P+1)
3410 GOSUB 8000
3420 LET P$(P,P)=" "
3430 LET Z$=P$(STR(P$,",")+1)
3440 CHANGE V5(W2) TO V2$(V1(W1),V2(W1)) USING Z$
3445 GOTO 3900

3450 GOSUB 8000
3460 LET W2=W1
3465 LET W$=P$(P+1)
3470 GOSUB 8200
3480 CHANGE V2$(V1(W2),V2(W2)) TO V5(W1)
3490 GOTO 3900

3499                                                  'CLOSEFILE'
3500 CHANGE P$(4) TO X
3505 IF X<7 THEN GOTO 3820
```

"ABLA-06C" ABL INTERACTIVE INTERPRETER- EXECUTE MODULE

```
3507 IF X>14 THEN GOTO 3820
3510 CLOSE FILE (X)
3515 GOTO 3900

3519                                        'CLOSE'
3520 FOR X=7 TO 14
3525  CLOSE FILE (X)
3530 NEXT X
3540 GOTO 3900

3549                                        'OPENFILE'
3550 LET P=STR(P$,",")
3555 CHANGE P$(4,P-1) TO X
3560 IF X<7 THEN GOTO 3820
3565 IF X>14 THEN GOTO 3820
3570 LET P$(P,P)=" "
3575 LET P1=STR(P$,",")
3580 CHANGE P$(P+1,P1-1) TO Z
3590 OPEN FILE (X,Z),P$(P1+1)
3595 GOTO 3900

3649                                        'PRINTFILE'
3650 LET P1=STR(P$,",")+1
3660 CHANGE P$(4,P1-2) TO X
3699                                        'PRINT'
3700 LET P=STR(P$,"<34>")
3705 IF P THEN GOTO 3760
3707 IF P$(1,3)="PRT" THEN LET P1=5
3710 LET W$=P$(P1)
3720 IF STR(W$,"$") THEN GOTO 3740
3725 GOSUB 8200
3730 PRINT PC(40,23),W$;"=";V5(W1);
3732 IF P$(1,3)="PRF" THEN PRINT FILE (X),W$," = ",V5(W1)
3735 GOTO 3900

3740 GOSUB 8000
3745 PRINT PC(40,23);W$;"=";V2$(V1(W1),V2(W1));
3747 IF P$(1,3)="PRF" THEN PRINT FILE (X),W$," = ",V2$(V1(W1),V2(W1))
3750 GOTO 3900

3760 LET P$(P,P)=" "
3770 LET P1=STR(P$,"<34>")
3780 PRINT PC(0,23);P$(P+1,P1-1);
3785 IF P$(1,3)="PRF" THEN PRINT FILE (X),P$(P+1,P1-1)
3790 GOTO 3900

                    ACTION'S HOST OPCODE INVALID OR NOT IMPLEMENTED - ACTION IGN
3800 LET Z=115
3805 GOTO 3880

                    VARIABLE MISSING ACCORDING TO HOST LANGUAGE SYNTAX - ACTION
3810 LET Z=116
```

"ABLA-06C" ABL INTERACTIVE INTERPRETER- EXECUTE MODULE

```
3815 GOTO 3880

                        CHANNEL # MUST BE BETWEEN 7 AND 14. FILE COMMAND IGNORED.
3820 LET Z=117
3825 GOTO 3880

                        INVALID OPERAND OR VARIABLE TYPE - ACTION IGNORED.
3830 LET Z=118
3880 GOSUB 9800
3900 LET R2=R2+1
3911 ACCEPT (9),Z$
3920 GOTO 2500

3999                                    'LET'
4000 LET P=STR(P$,"=")
4010 IF P=0 THEN GOTO 3830
4020 LET W$=P$(P+1)
4030 IF STR(P$(P-3,P-1),"$") THEN GOTO 4100
4040 GOSUB 8200
4045 LET W2=W1
4050 LET W$=P$(P+5)
4054 LET T9=0
4055 IF W$="    " THEN GOTO 4150
4060 IF W$(1,1)=>"A" THEN IF W$(1,1)=<"Z" THEN GOTO 4080
4070 CHANGE P$(P+5) TO T9
4075 GOTO 4150

4080 GOSUB 8200
4085 LET T9=V5(W1)
4090 GOTO 4150

4100 GOSUB 8000
4110 LET W2=W1
4115 LET P1=STR(P$(P),"<34>")
4118 IF P1=0 THEN GOTO 4130
4120 LET P$(P1,P1)=" "
4125 LET Z$=P$(P1+1,STR(P$(P1),"<34>")-1)
4128 GOTO 4150

4130 LET W$=P$(P+5)
4133 LET Z$=" ",Z$
4135 IF W$="    " THEN GOTO 4150
4140 GOSUB 8000
4145 LET Z$=V2$(V1(W1),V2(W1))
4150 LET W$=P$(P-3)
4160 IF STR(W$,"$")=0 THEN GOSUB 8200
4170 IF STR(W$,"$") THEN GOSUB 8000
4180 ON STR(" ^+-*/,",P$(P+4,P+4))+1 THEN GOTO 3830,4200,4210,4220,4230,4240,4250,4260
4200 IF STR(W$,"$") THEN GOTO 4260
4205 LET V5(W1)=V5(W2)
4207 GOTO 3900
```

"ABLA-06C" ABL INTERACTIVE INTERPRETER- EXECUTE MODULE

```
4210 LET V5(W1)=V5(W2)^T9
4215 GOTO 3900

4220 LET V5(W1)=V5(W2)+T9
4225 GOTO 3900

4230 LET V5(W1)=V5(W2)-T9
4235 GOTO 3900

4240 LET V5(W1)=V5(W2)*T9
4245 GOTO 3900

4250 LET V5(W1)=V5(W2)/T9
4255 GOTO 3900

4260 LET V2$(V1(W1),V2(W1))=V2$(V1(W2),V2(W2)),Z$
4265 GOTO 3900

4999                                    'SLOTFILE'
5000 CHANGE P$(4) TO X
5010 SLOT FILE (X)
5020 GOTO 3900

5049                                    'SCRATCHFILE'
5050 CHANGE P$(4) TO X
5060 SCRATCH FILE (X)
5070 GOTO 3900

5099                                    'READFILE','WRITEFILE'
5100 LET P=STR(P$," ",")
5105 LET P$(P,P)=" "
5110 LET P1=STR(P$," ",")
5120 CHANGE P$(4,P-1) TO X
5122 IF X<7 THEN GOTO 3820
5123 IF X>14 THEN GOTO 3820
5125 CHANGE P$(P+1,P1-1) TO L1
5127 IF P$(1,3)="WRF" THEN GOTO 5170
5130 READ FILE (X),Z$(1,L1)
5140 LET P$=P$(P1+1)
5150 GOSUB 5800
5160 GOTO 3900

5170 LET P$=P$(P1+1)
5175 GOSUB 5900
5180 WRITE FILE (X),Z$(1,L1)
5190 GOTO 3900

5299                                    'INSERTINDEX','DELETEINDEX','SEARCHINDEX','SEARCHNEXT'
5300 LET P=STR(P$," ",")
5310 LET P$(P,P)=" "
5315 LET P1=STR(P$," ",")
5320 LET P$(P1,P1)=" "
```

"ABLA-06C" ABL INTERACTIVE INTERPRETER- EXECUTE MODULE

```
5325 CHANGE P$(4,P-1) TO X
5330 CHANGE P$(P+1,P1-1) TO X1
5335 LET P=STR(P$,",")
5340 CHANGE P$(P1+1,P-1) TO L1
5345 LET W$=P$
5350 LET P$=P$(P+1)
5355 IF W$><"SNX" THEN GOSUB 5900
5360 IF W$="SRX" THEN SEARCH INDEX (X,X1),Z$(1,L1)
5370 IF W$="INX" THEN INSERT INDEX (X,X1),Z$(1,L1)
5380 IF W$="DLX" THEN DELETE INDEX (X,X1),Z$(1,L1)
5385 IF W$="SNX" THEN SEARCH NEXT (X,X1),Z$(1,L1)
5387 IF W$="SNX" THEN GOSUB 5800
5390 GOTO 3900

5399                                    'SEARCHFIRST'
5400 LET P=STR(P$,",")
5405 LET P$(P,P)=" "
5410 LET P1=STR(P$,",")
5420 CHANGE P$(4,P-1) TO X
5422 IF X<7 THEN GOTO 3820
5423 IF X>14 THEN GOTO 3820
5425 CHANGE P$(P+1,P1-1) TO L1
5440 LET P$=P$(P1+1)
5450 GOSUB 5900
5460 SEARCH FIRST (X),Z$(1,L1)
5470 GOTO 3900

5799                                    TRANSFER Z$ INTO VARIABLES FOR READS
5800 LET P9=0
5810 GOTO 5905

5899                                    TRANSFER VARIABLES INTO Z$ FOR WRITES
5900 LET P9=1
5905 LET X=1
5910 FOR I=1 TO STR(P$,",")/4+1
5915  LET W$=P$(I*4-3)
5920  IF STR(W$,"$") THEN GOTO 5950
5925  GOSUB 8200
5930  IF P9=0 THEN MOVE Z$(X,X+7) TO V5(W1)
5932  IF P9 THEN MOVE V5(W1) TO Z$(X,X+7)
5935  LET X=X+8
5940  GOTO 5970
5950  GOSUB 8000
5955  LET W=V2(W1)-V1(W1)
5960  IF P9=0 THEN MOVE Z$(X,X+W) TO V2$(V1(W1),V2(W1))
5962  IF P9 THEN MOVE V2$(V1(W1),V2(W1)) TO Z$(X,X+W)
5965  LET X=X+W+1
5970  IF X>L1 THEN LET I=100
5980 NEXT I
5990 RETURN
```

"ABLA-06C" ABL INTERACTIVE INTERPRETER- EXECUTE MODULE

```
5998                                        ---COMPOUND ACTION---
5999                                        STORE STATE VECTOR, TRANSFER CONTROL TO SUB-PROGRAM
6000 LET V8$=M5$(1,4)
6010 LET V8=A3
6020 LET V8(1)=R2+1
6030 OPEN FILE (0,512),"ABLSTACK"
6049                                        PUSH TWO RECORDS ONTO CONTROL STACK
6050 LET S=S+1
6060 WRITE FILE (0,S),V$,V1,V2$,V2
6070 LET S=S+1
6080 WRITE FILE (0,S),V5$,V5,V8$,V8,V9$,B1$,B2$,B3$
6090 CLOSE FILE (0)
6100 CLOSE FILE (5)
6110 CLOSE FILE (6)
6120 LET Z$="ABLP",M1$(93,94),M1$(91,92)
6130 OPEN FILE (5,5),Z$
6140 LET Z$(1,4)="ABLT"
6150 OPEN FILE (6,512),Z$
6155 LET Z$="ENTERING NEXT LOWER LEVEL OF ABSTRACTION- PGM/MACH ",M1$(91,92),"/",M1$(93,94)
6160 PRINT PC(0,23);Z$;
6165 IF B3$="Y" THEN PRINT FILE (15);SKP(2)
6167 IF B3$="Y" THEN PRINT FILE (15),Z$
6180 GOSUB 2900
6190 GOSUB 6200
6195 GOTO 2020

6199                                        PRINT EXISTING ALTERNATIVES
6200 FOR J=0 TO M5(3)-1
6210  LET T1=1
6220  LET X1=J*4
6230  LET X2=8
6240  FOR I=0 TO R-1
6250    IF X2=22 THEN GOTO 6350
6260    IF T1>5 THEN GOTO 6350
6270    IF T(I,J)=0 THEN IF T1<4 THEN GOTO 6340
6280    CHANGE T(I,J) TO Z$ USING " ##-"
6290    LET Z$(1,1)=T1$(T1,T1)
6300    IF I=0 THEN CHANGE T(I,J)/10 TO Z$ USING "#.#"
6310    PRINT PC(X1,X2);Z$;
6320    LET X2=X2+1
6330    IF T1<4 THEN GOTO 6350
6340    LET T1=T1+1
6350  NEXT I
6360 NEXT J
6400 PRINT PC(0,6);TAB(75);
6410 FOR I=0 TO 20
6420  IF B2$(I,I)="Y" THEN PRINT PC(I*4-4,6);"T";
6430  IF B1$(I,I)="Y" THEN PRINT PC(I*4-2,6);"B";
6440 NEXT I
6480 RETURN

6499                                        RESTORE STATE VECTOR, TRANSFER CONTROL BACK TO CALLING PROGRAM
```

"ABLA-06C" ABL INTERACTIVE INTERPRETER- EXECUTE MODULE

```
6500 OPEN FILE (0,512),"ABLSTACK"
6509                                   POP TWO RECORDS FROM CONTROL STACK
6510 READ FILE (0,S),V5$,V5,V8$,V8,V9$,B1$,B2$,B3$
6520 LET S=S-1
6530 READ FILE (0,S),V$,V1,V2$,V2
6540 LET S=S-1
6550 CLOSE FILE (0)
6560 LET A3=V8
6570 LET R2=V8(1)
6600 CLOSE FILE (5)
6610 CLOSE FILE (6)
6620 LET Z$="ABLP",V8$
6630 OPEN FILE (5,5),Z$
6640 LET Z$(1,4)="ABLT"
6650 OPEN FILE (6,512),Z$
6700 LET Z$="RESUMING NEXT HIGHER LEVEL OF ABSTRACTION- PGM/MACH ",V8$(3,4),"/",V8$(1,2)
6710 PRINT PC(0,23);Z$;
6720 IF B3$="Y" THEN PRINT FILE (15),SKP(2)
6730 IF B3$="Y" THEN PRINT FILE (15),Z$
6750 GOSUB 2900
6760 GOSUB 6200
6770 GOTO 2500
```

```
PN=ABLA-06C   SAVED UNDER ABLA-06C        MAR 21,82  14:37    PAGE 11

                "ABLA-06C" ABL INTERACTIVE INTERPRETER- EXECUTE MODULE

6999                                    .EVALUATE PREDICATES (PRE + POST CONDITIONS)
7000 LET P9=1
7010 FOR I=R2 TO R-1
7020  IF T(I,J)=0 THEN GOTO 7790
7049                                    DECODE PREDICATE AND EVALUATE
7050  SEARCH INDEX(1,2),M5$(1,2),"A ",T(I,J) GIVING M1$,M2
7060  LET P$=M1$(61,90)
7070  IF B2$(J+1,J+1)="N" THEN GOTO 7080
7075  PRINT PC(1,21);M1$(1,60);PC(1,22);P$;
7077  IF B3$="Y" THEN PRINT FILE (15),"EVALUATING PREDICATE",M1$
7080  IF P$(1,3)<>"IF " THEN GOTO 7700
7090  IF P$(4,6)<>"EOF" THEN GOTO 7100
7092  CHANGE P$(8,8) TO X
7094  ON EOF(X)+1 THEN GOTO 7780,7800
7100  IF STR(P$(4,6),"$") THEN GOTO 7400
7109                                    FIND VALUE OF NUMERIC VARIABLE
7110  LET L=(STR(V5$,P$(4,6))-1)/3
7120  IF L<0 THEN GOTO 7710
7130  IF P$(9,9)=>"A" THEN IF P$(9,9)=<"Z" THEN GOTO 7150
7140  CHANGE P$(9) TO T9
7145  GOTO 7200
7150  LET L1=STR(V5$,P$(9,11))
7160  IF L1=0 THEN GOTO 7710
7170  LET T9=V5((L1-1)/3)
7200  IF B2$(J+1,J+1)="N" THEN GOTO 7300
7210  PRINT PC(35,22);V5(L);P$(7,8);T9;
7220  IF B3$="Y" THEN PRINT FILE (15),"NUMERIC VALUES",V5(L),P$(7,8),T9
7300  ON STR("= ><> < >=<=",P$(7,8))+1 THEN GOTO 7720,7310,7320,7330,7340,7350,7360
7310  IF V5(L)=T9 THEN GOTO 7800
7315  GOTO 7780
7320  IF V5(L)><T9 THEN GOTO 7800
7325  GOTO 7780
7330  IF V5(L)>T9 THEN GOTO 7800
7335  GOTO 7780
7340  IF V5(L)<T9 THEN GOTO 7800
7345  GOTO 7780
7350  IF V5(L)=>T9 THEN GOTO 7800
7355  GOTO 7780
7360  IF V5(L)=<T9 THEN GOTO 7800
7365  GOTO 7780
7399                                    FIND VALUE OF ALPHA VARIABLE
7400  LET L=(STR(V$,P$(4,6))-1)/3
7410  IF L<0 THEN GOTO 7710
7420  IF P$(9,9)="<34>" THEN GOTO 7500
7430  LET L1=(STR(V$,P$(9,11))-1)/3
7440  IF L1<0 THEN GOTO 7710
7450  LET T9$=V2$(V1(L1),V2(L1))
7460  GOTO 7550
7500  LET T9$=P$(10,STR(P$(11),"<34>")-1)
7550  IF B2$(J+1,J+1)="N" THEN GOTO 7600
7570  PRINT PC(35,22);V2$(V1(L),V2(L));P$(7,8);V2$(V1(L1),V2(L1));
7580  IF B3$="Y" THEN PRINT FILE (15),"ALPHA VALUES",V2$(V1(L),V2(L)),P$(7,8),V2$(V1(L1),V2(L1))
```

"ABLA-06C" ABL INTERACTIVE INTERPRETER- EXECUTE MODULE

```
7600  ON STR("= ><> < >=<=",P$(7,8))+1 THEN GOTO 7720,7610,7620,7630,7640,7650,7660
7610  IF V2$(V1(L1),V2(L1))=T9$ THEN GOTO 7800
7615  GOTO 7780
7620  IF V2$(V1(L1),V2(L1))><T9$ THEN GOTO 7800
7625  GOTO 7780
7630  IF V2$(V1(L1),V2(L1))>T9$ THEN GOTO 7800
7635  GOTO 7780
7640  IF V2$(V1(L1),V2(L1))<T9$ THEN GOTO 7800
7645  GOTO 7780
7650  IF V2$(V1(L1),V2(L1))=>T9$ THEN GOTO 7800
7655  GOTO 7780
7660  IF V2$(V1(L1),V2(L1))=<T9$ THEN GOTO 7800
7665  GOTO 7780
                        INVALID PREDICATE - CANNOT BE EVALUATED
7700  LET Z=111
7705  GOTO 7770
                        VARIABLE HAS NOT BEEN GIVEN A VALUE. PREDICATE NOT EVALUATE
7710  LET Z=112
7715  GOTO 7770
                        INVALID OPERAND- PREDICATE NOT EVALUATED
7720  LET Z=113
7770  GOSUB 9800
7780  LET P9=0
7790  LET R2=1
7795  LET I=100
7800 NEXT I
7850 RETURN
```

"ABLA-06C" ABL INTERACTIVE INTERPRETER- EXECUTE MODULE

```
7999                                    .FIND THE ALPHANUMERIC LOCAL VARIABLE
8000 IF W$="    " THEN RETURN
8010 LET W1=(STR(V$,W$)-1)/3
8020 IF W1=>0 THEN RETURN
8050 LET X=(STR(V$,"    ")-1)/3
8060 LET V1(X)=1
8070 IF X THEN LET V1(X)=V2(X-1)+1
8080 LET V2(X)=V1(X)+W-1
8090 LET V$(X*3+1,X*3+3)=W$
8100 GOTO 8010

8199                                    FIND THE NUMERIC LOCAL VARIABLE
8200 IF W$="    " THEN RETURN
8210 LET W1=(STR(V5$,W$)-1)/3
8220 IF W1=>0 THEN RETURN
8250 LET X=(STR(V5$,"    ")-1)/3
8260 LET V5$(X*3+1,X*3+3)=W$
8270 GOTO 8210

8790 ON ERROR STOP
8792 LET Y9(0)=ERS(1)
8794 LET Y9(1)=ERS(0)
8796 ON ERROR GOTO 8790
8798 IF Y9=3210 THEN GOTO 3810
8800 IF Y9=3440 THEN GOTO 3900
8805 IF Y9=3480 THEN GOTO 3900
8810 IF Y9=3525 THEN IF Y9(1)=18 THEN GOTO 3530
8820 IF Y9=3590 THEN GOTO 3900
8875 IF Y9=9838 THEN GOTO 9842
8880 PRINT PC(0,23);"          UNEXPECTED ERROR ";Y9(1);" AT <7>";Y9;PC(0,20);
8900 STOP
```

TOTAL # LINES OF TEXT = 514

TOTAL # CHARACTERS OF TEXT = 14529

TOTAL # CHARACTERS IN 'REM' = 2559

LINES PRINTED = 514

FILE LAYOUT FOR: ABLMACHI PROJECT=AB
AS OF MAR 21,82 AT 14:38 PAGE 1

DESCRIPTION: ABL ABSTRACT MACHINE TABLE
FORMAT CODE:
MEMO:
POINTS TO: 256

TYPE:	I				
SHARED:	Y	VOLUME ID:	0	PREPARED BY:	MK
DYNAMIC:	Y	OPTIMIZE FACTOR:	0%	DATE PREPARED:	OCT 12,81
CATALOG:	300	TOTAL NO RECORDS:	400	REVISION NO:	00
RECORD SIZE:	122	ACTIVE NO RECORDS:	145	REVISED BY:	
U D SIZE:	116	FILE SIZE (BLOCKS):	104	DATE REVISED:	

FIELDS

#	FIELD DESCRIPTION	POS	SIZE	NAME/MASK	C	R	P
1	MACHINE NUMBER	1	2A	M$ (1,2)	0		
2	SECTION CODE (1)	3	1A	M$ (3,3)	0		
3	BUCKET	4	1A	M$ (4,4)	0		
4	ELEMENT NUMBER	5	1S	M1	0		
	USER DATA						
5	DESCRIPTION	7	60A	M1$(1,60)	0		
6	CORRESPONDING EXECUTABLE CODE	67	30A	M1$(61,90)	0		
7	PROGRAM IDENTIFIER (COMPOUND ACTIONS)(2)	97	2A	M1$(91,92)	0		
8	MACHINE IDENTIFIER (COMPOUND ACTIONS)(2)	99	2A	M1$(93,94)	0		
9	ESTIMATED EXECTION TIME (MILLISECONDS)	101	1S	M2	0		
10	INPUT ARGUMENTS	103	5S	M2 (1-5)	0		
11	OUTPUT ARGUMENTS	113	5S	M2 (6-10)	0		

NOTES

(1) SECTION CODE

SECTION CODE	NUMBER	PURPOSE
(BLANK)	0	MACHINE DESCRIPTION
A	1 - N	PREDICATE DESCRIPTION
B	1 - N	ACTION DESCRIPTION
C	1 - N	DATA OBJECT DESCRIPTION

...CONTINUED

(2) THESE IDENTIFIERS ARE USED ONLY FOR COMPOUND ACTIONS. IN ORDER TO 'CALL' SUB-PROGRAMS, USUALLY TO GO ONE LEVEL OF ABSTRACTION LOWER. THESE FIELDS ARE BLANK IF NOT USED.

FILE LAYOUT FOR: ABLHLIBR PROJECT=AB
AS OF MAR 21,82 AT 14:38 PAGE 3

DESCRIPTION: ABL MASTER LIBRARY OF MACHINES & PROGRMS
FORMAT CODE:
MEMO:
POINTS TO: 256

TYPE:	I				
SHARED:	Y	VOLUME ID:	0	PREPARED BY:	MK
DYNAMIC:	Y	OPTIMIZE FACTOR:	0%	DATE PREPARED:	OCT 22,81
CATALOG:	315	TOTAL NO RECORDS:	100	REVISION NO:	00
RECORD SIZE:	4	ACTIVE NO RECORDS:	13	REVISED BY:	
V D SIZE:		FILE SIZE (BLOCKS):	6	DATE REVISED:	

FIELDS

#	FIELD DESCRIPTION	POS	SIZE	NAME/MASK	C	R	P
1	MACHINE NUMBER	1	2A	M8$(1,2)	0		
2	PROGRAM NUMBER	3	2A	M8$(3,4)	0		

FILE LAYOUT FOR: ABLPXXYY PROJECT=AB
AS OF MAR 21,82 AT 14:38 PAGE 4

DESCRIPTION: ABL ABSTRACT PROGRAM
FORMAT CODE:
MEMO: XX=MACHINE YY=PROGRAM
POINTS TO: 256

TYPE:	I				
SHARED:	Y	VOLUME ID:	0	PREPARED BY:	MK
DYNAMIC:	Y	OPTIMIZE FACTOR:	0%	DATE PREPARED:	OCT 12,81
CATALOG:	305	TOTAL NO RECORDS:	200	REVISION NO:	00
RECORD SIZE:	84	ACTIVE NO RECORDS:	0	REVISED BY:	
U D SIZE:	80	FILE SIZE (BLOCKS):	0	DATE REVISED:	

FIELDS

#	FIELD DESCRIPTION	POS	SIZE	NAME/MASK	C	R	P
1	SECTION CODE (1)	1	1A	M2$(1,1)	0		
2	BUCKET	2	1A	M2$(2,2)	0		
3	ELEMENT NUMBER	3	1S	M3	0		
	USER DATA						
4	DESCRIPTION	5	60A	M3$(1,60)	0		
5	INPUT ARGUMENTS	65	5S	M4 (0-4)	0		
6	OUTPUT ARGUMENTS	75	5S	M4 (5-9)	0		

NOTES

(1)

SECTION CODE	NUMBER	PURPOSE
(BLANK)	1 - N	PROGRAM DESCRIPTION
A	1 - N	CLUSTER DESCRIPTION
B	1 - N	ALTERNATIVE DESCRIPTION

FILE LAYOUT FOR: ABLSTACK PROJECT=AB
AS OF MAR 21,82 AT 14:38 PAGE 5

DESCRIPTION: ABL CONTROL STACK FOR EXECUTION
FORMAT CODE: 01
MEMO: RECORD 01
POINTS TO:

TYPE:	R				
SHARED:	N	VOLUME ID:	0	PREPARED BY:	MK
DYNAMIC:	N	OPTIMIZE FACTOR:	0%	DATE PREPARED:	JAN 04,82
CATALOG:	300	TOTAL NO RECORDS:	20	REVISION NO:	00
RECORD SIZE:	512	ACTIVE NO RECORDS.	0	REVISED BY:	
U D SIZE:		FILE SIZE (BLOCKS):	0	DATE REVISED:	

FIELDS

#	FIELD DESCRIPTION		POS	SIZE	NAME/MASK	C	R	P
1	VARIABLE NAMES FOR ALPHA VARS.	20 * 3A	1	60A	V$ (1,60)	0		
2	START POSITION FOR ALPHA VARS.	20 * 1S	61	20S	V1 (0-19)	0		
3	ALPHA VARIABLES		101	372A	V2$(1,372)	0		
4	END POSITION FOR ALPHA VARS.	20 * 1S	473	20S	V2 (0-19)	0		

FILE LAYOUT FOR: ABLSTACK PROJECT=AB
AS OF MAR 21,82 AT 14:38 PAGE 6

DESCRIPTION: ABL CONTROL STACK FOR EXECUTION -02
FORMAT CODE: 02
MEMO: RECORD 02
POINTS TO:

TYPE:	R				
SHARED:	N	VOLUME ID:	0	PREPARED BY:	MK
DYNAMIC:	N	OPTIMIZE FACTOR:	0%	DATE PREPARED:	JAN 04,82
CATALOG:	305	TOTAL NO RECORDS:	20	REVISION NO:	00
RECORD SIZE:	512	ACTIVE NO RECORDS:	0	REVISED BY:	
U D SIZE:		FILE SIZE (BLOCKS):	0	DATE REVISED	

FIELDS

#	FIELD DESCRIPTION	POS	SIZE	NAME/MASK	C	R	P
1	VARIABLE NAMES FOR NUMERIC VARS. 40 * 3A	1	120A	V5$(1,120)	0		
2	NUMERIC VARIABLES	121	40L	V5 (0-39)	0		
3	CALLING MACHINE / PROGRAM	441	4A	V8$(1,4)	0		
4	CALLING ALTERNATIVE (COLUMN) NUMBER	445	1S	V8	0		
5	CALLING ACTION (ROW) NUMBER	447	1S	V8 (1)	0		
6	BUCKET - V9$(1,23)	449	23A	V9$(1,23)	0		
7	BREAKPOINT VECTOR - B1$(1,20)	472	20A	V9$(24,43)	0		
8	TRACE VECTOR - B2$(1,20)	492	20A	V9$(44,63)	0		
9	PRINT TRACE ON PRINTER ? [Y/N] B3$(1,1)	512	1A	V9$(64,64)	0		

FILE LAYOUT FOR: ABLTXXYY PROJECT=AB
AS OF MAR 21,82 AT 14:38 PAGE 7

DESCRIPTION: ABL COMPACTED ABSTRACT PROGRAM TABLE
FORMAT CODE:
MEMO: XX=MACHINE YY=PROGRAM
POINTS TO: ABLPXXYY

TYPE:	R				
SHARED:	Y	VOLUME ID:	0	PREPARED BY:	MK
DYNAMIC:	Y	OPTIMIZE FACTOR:	0%	DATE PREPARED:	OCT 12,81
CATALOG:	310	TOTAL NO RECORDS:	100	REVISION NO:	00
RECORD SIZE:	512	ACTIVE NO RECORDS:	0	REVISED BY:	
U D SIZE:		FILE SIZE (BLOCKS):	0	DATE REVISED:	

FIELDS

#	FIELD DESCRIPTION	POS	SIZE	NAME/MASK	C	R	P
1	RECORD MARK (=255)	1	1S	M5	0		
2	RECORD NUMBER [1/2/3]	3	1S	M5 (1)	0		
3	TOTAL NUMBER OF CLUSTERS	5	1S	M5 (2)	0		
4	TOTAL NUMBER OF ALTERNATIVES	7	1S	M5 (3)	0		
5	TOTAL NUMBER OF PRECONDITIONS (2)	9	1S	M5 (4)	0		
6	TOTAL NUMBER OF ACTIONS (2)	11	1S	M5 (5)	0		
7	TOTAL NUMBER OF POSTCONDITIONS (2)	13	1S	M5 (6)	0		
8	TOTAL NUMBER OF DATA OBJECTS (2)	15	1S	M5 (7)	0		
9	BUCKET	17	3S	M5 (8-10)	0		
10	MACHINE NUMBER	23	2A	M5$(1,2)	0		
11	PROGRAM NUMBER	25	2A	M5$(3,4)	0		
12	BUCKET	27	66A	M5$(5,70)	0		
13	BUCKET	93	5F	M6 (0-4)	0		
14	COMPACTED MATRIX (1)	113	200S	M7 (0-199)	0		

NOTES

(1) THE 30 ROW X 20 COLUMN COMPACTED MATRIX IS SPLIT INTO THREE RECORDS.
THE FIRST RECORD HAS THE FIRST THIRD OF THE CONTROL FLOW TABLE.
THE SECOND RECORD HAS THE SECOND THIRD OF THE CONTROL FLOW TABLE.
THE THIRD RECORD HAS THE LAST THIRD OF THE CONTROL FLOW TABLE.

...CONTINUED

THE VARIABLE M7 IS ACTUALLY DIMENSIONED AS SHORT M7(9,19).

THE STRUCTURE OF THE TABLE IS, FOR EACH COLUMN.

- CLUSTER (ALTERNATIVE) NUMBER	(1.1..99.9) (MULTIPLIED X 10)
- PRECONDITIONS	(-P OR +P, P=PREDICATE NUMBER)
- 0	
- ACTIONS	(1..A , A=ACTION NUMBER)
- 0	
- POSTCONDITIONS	(-P OR +P , P=PREDICATE NUMBER)
- 0	
- NEXT STEP NUMBER	(0..HIGHEST CLUSTER NUMBER)
- EXCEPTION STEP NUMBER	(0..HIGHEST CLUSTER NUMBER)

NOTE THAT THE TABLE IS ALWAYS SORTED IN ORDER OF CLUSTER (ALTERNATIVE) NUMBER.
EG. 11 12 21 31 32 33 41 51

(2) FOR FUTURE USE.

INFERIEURE

LIST OF REFERENCES

ACM81 ACM SIGSOFT Special Issue, "NBS Programming Environment Workshop Report", Software Engineering Notes, Vol. 6, No. 4, August, 1981, pp. 3-13.

BELA80 Belady, L.A., and Leavenworth, B., "Program Modifiability", in "Software Engineering", Freeman, H., and Lewis, P.M., eds., Academic Press, Toronto, 1980, pp. 25-35.

BELK76 Belkin, G., and Jaworski, W.M., "Towards Logic and Performance Analysis of Unwritten Programs", Proceedings, Canadian Computer Conference, Session '76, Montreal, May, 1976.

BOEH78 Boehm, B.W., et al, "Characteristics of Software Quality", North-Holland Publishing, New York, 1978.

CLAR76 Clarke, L.A., "A System to Generate Fast Data and Symbolically Execute Programs", IEEE Transactions on Software Engineering, Vol. SE-2, 1976, pp. 215-222.

CLAR80 Clarke, L.A., and Richardson, D.J., "Symbolic Evaluation Methods for Program Analysis", Program Flow Analysis: Theory and Applications, Muchnick, S.S., and Jones, N.D., eds., Prentice-Hall, Englewood Cliffs, 1980, pp. 264-300.

DIJK72 Dijkstra, E.W., "Notes on Structured Programming", in "Structured Programming", cited by Belady, L.A., and Leavenworth, B., "Program Modifiability", in "Software Engineering", Freeman, H., and Lewis, P.M., eds., Academic Press, Toronto, 1980, pp. 25-35.

DIJK79 Dijkstra, E.W., "My Hopes of Computing Science", reprinted in "Software Engineering", Freeman, H., and Lewis, P.M., eds., Academic Press, Toronto, 1980, pp. 95-109.

ELLI80 Ellis, C.A., and Nutt, G.J., "Office Information Systems and Computer Science", Computing Surveys, Vol. 12, No. 1, March, 1980, pp. 27-60.

FANC76 Fancott, T., and Jaworski, W.M., "Primitive Logical Constructs Considered Harmful in Structured Programs", Proceedings, Canadian Computer Conference, Session '76, Montreal, May, 1976.

FREE80 Freedman, M.D., "Tools for the Efficient Design of Software", in "Software Engineering", Freeman, H., and Lewis, P.M., eds., Academic Press, Toronto, 1980, pp. 111-118.

GOOD81 Good, M., "Etude and the Folklore of User Interface Design", Communications of the ACM, Vol. 24, No. 5, 1981, pp. 34-43.

GUST82 Gustafson, G.G., and Kerr, R.J., "Some Practical Experience with a Software Quality Assurance Program", Communications of the ACM, Vol. 25, No. 1, 1982, pp. 4-12.

HINT81 Hinterberger, H., and Jaworski, W.M., "Controlled Program Design by use of the ABL Programming Concept", in "Angewandte Informatik", (Applied Informatics), Weisbaden, West Germany, July, 1981, pp. 302-310.

HORV82 Horvath, A., "Modelling and Implementation of an Information System for the Control of Truancy in the Quebec High Schools", Master of Computer Science Thesis, Concordia University, Montreal, 1982.

JAWO70 Jaworski, W.M., "An Interactive System for the Generation of Programs from Decision Tables", Computer Aided System Simulation, Analysis & Design (CASSAD '70), University of Houston, 1970.

JAWO81 Jaworski, W.M., unpublished notes, Concordia University, Montreal, 1981.

JENS74 Jensen, K., and Wirth, N., "PASCAL: User Manual and Report", Springer-Verlag, New York, 1974, pp. 110-115.

KORF74 Korfhage, R.R., "Discrete Computational Structures", Academic Press, New York, 1974, pp. 168-178.

LEBE82a Lebensold, J., "A Language for Describing Office Information Systems", Master of Computer Science Thesis, Concordia University, Montreal, 1982.

LEBE82b Lebensold, J., Radhakrishnan, T., and Jaworski, W. M., "A Modelling Tool for Office Automation Systems", a paper presented at ACM-SIGOA conference on Office Information Systems, Philadelphia, Pennsylvania, June 21-23, 1982.

LINA82 Linares, J., "A Comprehensive Support System for Microcode Generation", Master of Computer Science Thesis, Concordia University, Montreal, 1982.

MAO80 Mao, T.W., and Yeh, R.T., "Communication Port: A Language Concept for Concurrent Programming", IEEE Transactions on Software Engineering, Vol. SE-6, No. 2, March, 1980, pp. 194-204.

MCCA76 McCabe, T.J., "A Complexity Measure", IEEE Transactions on Software Engineering, Vol. SE-2, No. 4, December, 1976, pp. 308-320.

MCIN78 McIntire, T.C., "Software Interpreters for Microcomputers", John Wiley & Sons, Toronto, 1978, pp. 1-13, 127-227.

MILL79 Miller, E., "Tutorial: Automated Tools for Software Engineering", IEEE Computer Society Publication, IEEE Catalog No. EHO 150-3, 1979, pp. 1-5.

MITC79 Mitchell, J.G., "The Design and Construction of Flexible and Efficient Interactive Programming Systems", Garland Publishing, New York, 1979.

MORG81 Morgan, A.H., "An Engineering Approach to Problem Analysis", Master of Computer Science Thesis, Concordia University, Montreal, 1981.

MYER80 Myers, G.J., "The Art of Software Testing", John Wiley & Sons, Toronto, 1980.

OSTE80 Osterweil, L., "Using Data Flow Tools in Software Engineering", in "Program Flow Analysis: Theory and Applications", Muchnick, S.S., and Jones, N.D., eds., Prentice-Hall, Englewood Cliffs, 1980, pp. 237-263.

PARN79 Parnas, D.L., "Designing Software for Ease of Extension and Contraction", IEEE Transactions on Software Engineering, Vol. SE-5, No. 2, March, 1979, pp. 128-137.

PETE78 Peters, L.J., and Tripp, L.L., "A Model of Software Engineering", Proceedings; Third International Conference on Software Engineering, May, 1978, pp. 63-70.

RACA81 Racal-Vadic Inc., "VA811 Multimode Automatic Calling Unit", (Technical Brochure), March, 1981.

REIF79 Reifer, D.J., and Trattner, S., "A Glossary of Software Tools and Techniques", in "Tutorial: Automated Tools for Software Engineering", Miller, E., ed., IEEE Computer Society Publication, IEEE Catalog No. EHO 150-3, 1979, pp. 6-14.

SIEM78 Siemens Inc., "SD-192 Electronic Private Automatic Branch Exchange Features Manual", (Systems Engineering and Operations Manual), October, 1978.

TEIT81 Teitelbaum, T., and Reps, T., "The Cornell Program Synthesizer: A Syntax-Directed Programming Environment", Communications of the ACM, Vol. 24, No. 9, September, 1981, pp. 563-573.

WALT73 Walther, G.H., "The On-Line User-Computer Interface: The Effects of Interface, Flexibility, Experience, and Terminal-Type on User Satisfaction and Performance", cited by Good, M., "Etude and the Folklore of User Interface Design", Communications of the ACM, Vol. 24, No. 5, 1981, pp. 34-43.

WILC76 Wilcox, T.R., Davis, A.M., and Tindall, M.H., "The Design and Implementation of a Table Driven, Interactive Diagnostic Programming System", Communications of the ACM, Vol. 19, No. 11, November, 1976, pp. 609-616.

WILL82 Williams, A.J., and Jaworski, W.M., "Representation of Therapeutic Strategies in Dermatology", presented at the Canadian Dermatological Association Annual Meeting, Edmonton, Alberta, July 2-6, 1982.

ANNEX I

ABL TABLES TO SUPPORT THE LOGICAL DESCRIPTION IN THE CHAPTER 3

4. ABL-ROUTINE MODELING;

```
*****************************************************************************
*                                                                           *
*     MODELING # OR THE OVERALL SYSTEM DESIGN AND MODELING OF THIS SYSTEM    *
*     ------------------------------------------------------------------    *
*                                                                           *
*  THIS TABLE PROVIDES AN OVERVIEW IN THE MODELING STAGE OF THE SYSTEM.     *
*  MODELING  HAS 44 ACTION PROCEDURES AND 44 PREDICATES.  THESE ACTIONS     *
*  ARE GROUPED INTO FUNCTIONS AND SUBFUNCTIONS AND FORM THE HIERARCHICALLY  *
*  INTERRELATED BUILDING BLOCKS OF THE WHOLE SYSTEM. THE PREDICATES ACT AS  *
*  CRITERIONS FOR A PARTICULAR SUBFUNCTION, THEY DEFINE THE CONSTRAINTS     *
*  THE ACTIONS IN THE PARTICULAR SUBFUNTION MUST OBEY.  UNLESS THESE        *
*  CRITERIONS ARE SATISFIED, THAT PARTICULAR SUBFUNCTION IS NOT ADEQUATELY  *
*  DESIGNED TO MEET ITS PURPOSE.                                            *
*                                                                           *
*  ONCE ALL THE ACTIONS ARE DESIGNED, THESE SAME ACTIONS WILL BE OPERATIONALLY *
*  SEQUENCED AND INTERFACED IN THE NEXT TABLE CALLED  #SYSTALL              *
*                                                                           *
*****************************************************************************
```

```
1.=== ORGANIZE THE BODY OF THE SYSTEM                          ===#          < 1, 1, 1, 1, 1, 1, 1, 1, 2 >
    1.1==DECENTRELIZED PROCESSING IS NEEDED                 ==   --THEN-GO-TO-CLUSTER-> 1
    1.2==SET UP RANDOM ACCESS DATA-BASE                     ==   --THEN-GO-TO-CLUSTER-> 1
    1.3==MODULARIZE SYSTEM                                  ==   --THEN-GO-TO-CLUSTER-> 1
    1.4==SET UP INTERACTIVE #DB UPDATING NAIVE USER INTERFACE==   --THEN-GO-TO-CLUSTER-> 1
    1.5==SET UP INTERACTIVE #DB QUERRY NAIVE USER INTERFACE  ==   --THEN-GO-TO-CLUSTER-> 1
    1.6==SET UP PSEUDOBATCH #DB QUERRY POSSIBILITY          ==   --THEN-GO-TO-CLUSTER-> 1
    1.7==SET UP ACCESSORIES FOR #ATS                        ==   --THEN-GO-TO-CLUSTER-> 1
    1.8==SET UP PRODUCTION CYCLE PROGRAMS                   ==   --THEN-GO-TO-CLUSTER-> 1
    1.9==BODY OF THE BASIC SYSTEM IS ORGANIZED              ==   --THEN-GO-TO-CLUSTER-> 2
2.=== ORGANIZE THE REGULAR RUN OF THE SYSTEM                   ===#          < 2, 2, 2, 2, 2, 2, 3 >
    2.1==#ATS DAILY DATA USAGE PARAMETER DEFINITION         ==   --THEN-GO-TO-CLUSTER-> 2
    2.2==#ATS DAILY TRANSACTION DATA GENERATION             ==   --THEN-GO-TO-CLUSTER-> 2
    2.3==#ATS TRANSACTION DATA COLLECTION                   ==   --THEN-GO-TO-CLUSTER-> 2
    2.4==#ATS TRANSACTION DATA PROCESSING                   ==   --THEN-GO-TO-CLUSTER-> 2
    2.5==#ATS DAILY DATA DISTRIBUTION                       ==   --THEN-GO-TO-CLUSTER-> 2
    2.6==#ATS DAILY OUTPUT USAGE                            ==   --THEN-GO-TO-CLUSTER-> 2
    2.7==#ATS SYSTEM IS ORGANIZED                           ==   --THEN-GO-TO-CLUSTER-> 3
```

CONTINUED ON THE NEXT PAGE

3.=== DATA UPDATE ORGANIZATION ===$ < 3, 3, 3, 4 >

3.1==UPDATE THE COURSE #NF == --THEN-GO-TO-CLUSTER-> 3

3.2==UPDATE THE STUDENT #NF == --THEN-GO-TO-CLUSTER-> 3

3.3==UPDATE ATTENDANCE RELATED DATA == --THEN-GO-TO-CLUSTER-> 3

3.4==UPDATE ORGANIZATION IS COMPLETE == --THEN-GO-TO-CLUSTER-> 4

4.=== DATA-BASE QUERRY ORGANIZATION ===$ < 4, 4, 4, 4, 4, 4, 0 >

4.1==QUERRY THE COURSE #NF INTERACTIVELY == --THEN-GO-TO-CLUSTER-> 4

4.2==QUERRY THE STUDENT #NF INTERACTIVELY == --THEN-GO-TO-CLUSTER-> 4

4.3==QUERRY THE CLASS #NF INTERACTIVELY == --THEN-GO-TO-CLUSTER-> 4

4.4==QUERRY THE COURSE #NF PSEUDOBATCH MODE == --THEN-GO-TO-CLUSTER-> 4

4.5==QUERRY THE STUDENT #NF PSEUDOBATCH MODE == --THEN-GO-TO-CLUSTER-> 4

4.6==QUERRY THE CLASS #NF PSEUDOBATCH MODE == --THEN-GO-TO-CLUSTER-> 4

4.7==DATA-BASE QUERRY ORGANIZATION IS COMPLETE == --THEN-GO-TO-CLUSTER-> 0

4. AAL-ROUTINE MODELING;

```
*************************************************************************
*                                                                       *
*     MODELING #0 OR THE OVERALL SYSTEM DESIGN AND MODELING OF THIS SYSTEM *
*     ------------------------------------------------------------------ *
*                                                                       *
*  THIS TABLE PROVIDES AN OVERVIEW IN THE MODELING STAGE OF THE SYSTEM.  *
*  MODELING  HAS 44 ACTION PROCEDURES AND 64 PREDICATES.  THESE ACTIONS  *
*  ARE GROUPED INTO FUNCTIONS AND SUBFUNCTIONS AND FORM THE HIERARCHICALLY *
*  INTERRELATED BUILDING BLOCKS OF THE WHOLE SYSTEM. THE PREDICATES ACT AS *
*  CRITERIONS FOR A PARTICULAR SUBFUNCTION. THEY DEFINE THE CONSTRAINTS  *
*  THE ACTIONS OF THE PARTICULAR SUBFUNTION MUST OBEY.  UNLESS THESE     *
*  CRITERIONS ARE SATISFIED, THAT PARTICULAR SUBFUNCTION IS NOT ADEQUATELY *
*  DESIGNED TO MEET ITS PURPOSE.                                        *
*                                                                       *
*  ONCE ALL THE ACTIONS ARE DESIGNED, THESE SAME ACTIONS WILL BE OPERATIONALLY *
*  SEQUENCED AND INTERFACED IN THE NEXT TABLE CALLED  #SYSTALL          *
*                                                                       *
*************************************************************************
```

1.=== ORGANIZE THE BODY OF THE SYSTEM ===# < 1, 1, 1, 1, 1, 1, 1, 1, 2 >

1.1==DECENTRELIZED PROCESSING IS NEEDED == --THEN-GO-TO-CLUSTER-> 1
- 1.1 IF PRODUCTION CYCLE PERIOD IS DEFINED AS 1-2 DAYS LONG AND IF
  DATA COLLECTION PERIOD IS DEFINED AS 30-40 MINUTES LONG AND IF
  INEXPENSIVE TERMINAL IS AVAILABLE AND IF
  INEXPENSIVE DATA COMMUNICATION IS AVAILABLE AND IF
  REMOTE TIME-SHARING COMPUTER IS AVAILABLE

1.2==SET UP RANDOM ACCESS DATA-BASE == --THEN-GO-TO-CLUSTER-> 1
- 1.2 IF ON-LINE SYSTEM ACCESS IS NEEDED AND IF
  DISK-STORAGE IS AVAILABLE AND IF
  INTERACTIVE QUERRY AND UPDATE IS NEEDED AND IF
  SYSTEM USAGE IS STUDIED

1.3==MODULARIZE SYSTEM == --THEN-GO-TO-CLUSTER-> 1
- 1.3 IF NOT SYSTEM OPERATING SPECIALIST IS AVAILABLE IN SCHOOL AND IF
  SYSTEM OPERATING TASK HAS TO BE SHARED BY REGULAR STAFF AND IF
  USER SYSTEM ACCESS ORGANIZATION IS DEFINED AND IF
  CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED AND IF
  SYSTEM MODULARIZATION IS DEFINED

1.4==SET UP INTERACTIVE #DB UPDATING NAIVE USER INTERFACE== --THEN-GO-TO-CLUSTER-> 1
- 1.4 IF HIGH DEGREE OF DATA-BASE ACCURACY IS NEEDED AND IF
  HIGH VOLUME OF CHANGES OCCUR IN THE DATA-BASE AND IF
  DATA-BASE CHANGES HAVE TO BE IMPLEMENTED IN 1-2 DAYS AND IF
  NOT SYSTEM OPERATING SPECIALIST IS AVAILABLE IN SCHOOL AND IF
  SYSTEM OPERATING TASK HAS TO BE SHARED BY REGULAR STAFF AND IF
  USER SYSTEM ACCESS ORGANIZATION IS DEFINED AND IF
  CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED AND IF
  INTERACTIVE UPDATING LANGUAGE FOR THE COURSE #MF. IS NEEDED AND IF
  INTERACTIVE UPDATING LANGUAGE FOR THE STUDENT #MF. IS NEEDED AND IF
  CLASS MASTER FILE UPDATING CAN BE DONE AUTOMATICALLY

CONTINUED ON THE NEXT PAGE

```
1.5==SET UP INTERACTIVE #DB QUERRY NAIVE USER INTERFACE ==     --THEN-GO-TO-CLUSTER-> 1
     1.5 IF  IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING  AND IF
             OUTPUT VOLUME IS SMALL  AND IF
             NOT  SYSTEM OPERATING SPECIALIST IS AVAILABLE IN SCHOOL  AND IF
             SYSTEM OPERATING TASK HAS TO BE SHARED BY REGULAR STAFF  AND IF
             USER SYSTEM ACCESS ORGANIZATION IS DEFINED  AND IF
             CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED  AND IF
             INTERACTIVE QUERRY LANGUAGE FOR THE COURSE #NF. IS NEEDED  AND IF
             INTERACTIVE QUERRY LANGUAGE FOR THE STUDENT #NF. IS NEEDED  AND IF
             INTERACTIVE QUERRY LANGUAGE FOR THE CLASS #NF. IS NEEDED

1.6==SET UP PSEUDOBATCH #DB QUERRY POSSIBILITY   ==           --THEN-GO-TO-CLUSTER-> 1
     1.6 IF  NOT  IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING  AND IF
             NOT  OUTPUT VOLUME IS SMALL  AND IF
             NOT  SYSTEM OPERATING SPECIALIST IS AVAILABLE IN SCHOOL  AND IF
             SYSTEM OPERATING TASK HAS TO BE SHARED BY REGULAR STAFF  AND IF
             USER SYSTEM ACCESS ORGANIZATION IS DEFINED  AND IF
             CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED  AND IF
             INTERACTIVE QUERRY LANGUAGE FOR THE COURSE #NF. IS NEEDED  AND IF
             INTERACTIVE QUERRY LANGUAGE FOR THE STUDENT #NF. IS NEEDED  AND IF
             INTERACTIVE QUERRY LANGUAGE FOR THE CLASS #NF. IS NEEDED

1.7==SET UP ACCESSORIES FOR #ATS                ==     --THEN-GO-TO-CLUSTER-> 1
     1.7 IF  HIGH VOLUME OF TRUANCY TRANSACTION OCCUR (100-250 A LECTURE) AND IF
             DATA COLLECTION PERIOD IS DEFINED AS 30-40 MINUTES LONG  AND IF
             DATA GENERATION PERIOD IS 30-60 SECONDS  AND IF
             ERROR GENERATION POSSIBILITY IN TRANSACTION DATA IS MINIMAL  AND IF
             INEXPENSIVE OPTICAL MARK READER IS FOUND FOR THE TERMINAL

1.8==SET UP PRODUCTION CYCLE PROGRAMS           ==     --THEN-GO-TO-CLUSTER-> 1
     1.8 IF  REGULAR #ATS OPERATOR IS AVAILABLE FROM STAFF  AND IF
             PRODUCTION CYCLE PERIOD IS DEFINED AS 1-2 DAYS LONG  AND IF
             DATA COLLECTION PERIOD IS DEFINED AS 30-40 MINUTES LONG  AND IF
             DATA TRANSMISSION IS BAD SOMETIMES  AND IF
             OUTPUT PRINTING ON THE TERMINAL SOMETIMES HAS TO WAIT

1.9==BODY OF THE BASIC SYSTEM IS ORGANIZED       ==     --THEN-GO-TO-CLUSTER-> 2
     1.9 IF  ALL SUBFUNCTIONS IN THIS FUNCTION (CLUSTER) ARE MET
2.=== ORGANIZE THE REGULAR RUN OF THE SYSTEM                ===$          < 2, 2, 2, 2, 2, 2, 3 >

2.1==#ATS DAILY DATA USAGE PARAMETER DEFINITION  ==     --THEN-GO-TO-CLUSTER-> 2
     2.1 IF  VARIABLE LENGTH OUTPUT IS NEEDED

2.2==#ATS DAILY TRANSACTION DATA GENERATION      ==     --THEN-GO-TO-CLUSTER-> 2
     2.2 IF  DATA GENERATION PROCESS IS ACCEPTED BY THE TEACHERS  AND IF
             ERROR GENERATION POSSIBILITY IN TRANSACTION DATA IS MINIMAL  AND IF
             DATA GENERATION PERIOD IS 30-60 SECONDS

2.3==#ATS TRANSACTION DATA COLLECTION            ==     --THEN-GO-TO-CLUSTER-> 2
     2.3 IF  DATA COLLECTION PROCESS IS ACCEPTED BY THE ADMINISTRATION  AND IF
             DATA COLLECTION ROUTE IS ORGANIZED  AND IF
             DATA COLLECTION PERIOD IS DEFINED AS 30-40 MINUTES LONG  AND IF
             DATA COLLECTION ERRORS ARE MINIMAL  AND IF
             REGULAR #ATS OPERATOR IS AVAILABLE FROM STAFF
```

CONTINUED ON THE NEXT PAGE

```
2.4==BATS TRANSACTION DATA PROCESSING                    ==    --THEN-GO-TO-CLUSTER-> 2
     2.4 IF  EXTENSIVE TRANSACTION STORAGE AND ANALYSIS IS REQUIRED  AND IF
             REGULAR BATS OPERATOR IS AVAILABLE FROM STAFF  AND IF
             PROCEDURE IS SIMPLE  AND IF
             PROCEDURE IS SHORT  AND IF
             OUTPUT CAN BE SPOOLED  AND IF
             PRINTING OF SPOOLED OUTPUT IS NOT MORE THAN 1-2 HOURS

2.5==BATS DAILY DATA DISTRIBUTION                        ==    --THEN-GO-TO-CLUSTER-> 2
     2.5 IF  PROCEDURE IS SIMPLE

2.6==BATS DAILY OUTPUT USAGE                             ==    --THEN-GO-TO-CLUSTER-> 2
     2.6 IF  OUTPUT VOLUME IS ADEQUATE TO DAILY USAGE  AND IF
             FALSE NEGATIVE ERROR RATE IS BELOW 10% RATE  AND IF
             FALSE POSITIVE ERROR RATE IS BELOW 0.1% RATE  AND IF
             PROVIDED INFORMATION CAN BE READILY USED IN DECISION MAKING AND IF
             INFORMATION VALUE IS MAXIMUM OPERATIONAL VS. OVERALL DELAY

2.7==BATS SYSTEM IS ORGANIZED                            ==    --THEN-GO-TO-CLUSTER-> 3
     2.7 IF  ALL SUBFUNCTIONS IN THIS FUNCTION (CLUSTER) ARE MET
3.== DATA UPDATE ORGANIZATION                            ===$                  < 3, 3, 3, 4 >

     3.1==UPDATE THE COURSE #NF                          ==    --THEN-GO-TO-CLUSTER-> 3
          3.1 IF  HIGH DEGREE OF DATA-BASE ACCURACY IS NEEDED  AND IF
                  USER SYSTEM ACCESS ORGANIZATION IS DEFINED  AND IF
                  #NP IN CHARGE OF THE SYSTEM HAS ACCESS  AND IF
                  REDUNDANT DATA IN THE DATA-BASE ARE AUTOMATICALLY UPDATED

     3.2==UPDATE THE STUDENT #NF                         ==    --THEN-GO-TO-CLUSTER-> 3
          3.2 IF  HIGH DEGREE OF DATA-BASE ACCURACY IS NEEDED  AND IF
                  USER SYSTEM ACCESS ORGANIZATION IS DEFINED  AND IF
                  RECORDS OFFICE OF THE SCHOOL HAS ACCESS  AND IF
                  REDUNDANT DATA IN THE DATA-BASE ARE AUTOMATICALLY UPDATED

     3.3==UPDATE ATTENDANCE RELATED DATA                 ==    --THEN-GO-TO-CLUSTER-> 3
          3.3 IF  HIGH DEGREE OF DATA-BASE ACCURACY IS NEEDED  AND IF
                  USER SYSTEM ACCESS ORGANIZATION IS DEFINED  AND IF
                  SECTOR #NP, OF THE SCHOOL HAS ACCESS  AND IF
                  CLASS RELATED DATA CHANGED OR SOME #ABS-CARDS GOT LOST  AND IF
                  DATA-BASE DATA IS UPDATED  AND IF
                  CARD PUNCHING AND INTERPRETING IS AVAILABLE

     3.4==UPDATE ORGANIZATION IS COMPLETE                ==    --THEN-GO-TO-CLUSTER-> 4
          3.4 IF  ALL SUBFUNCTIONS IN THIS FUNCTION (CLUSTER) ARE MET
4.== DATA-BASE QUERRY ORGANIZATION                       ===$                  < 4, 4, 4, 4, 4, 4, 0 >

     4.1==QUERRY THE COURSE #NF INTERACTIVELY            ==    --THEN-GO-TO-CLUSTER-> 4
          4.1 IF  CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED  AND IF
                  #NP IN CHARGE OF THE SYSTEM HAS ACCESS  AND IF
                  OBJECTIVE IS AN EFFICIENT SCHOOL MANAGEMENT  AND IF
                  IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING  AND IF
                  OUTPUT VOLUME IS SMALL
```

CONTINUED ON THE NEXT PAGE

```
4.2==QUERRY THE STUDENT #NF INTERACTIVELY                    ==    --THEN-GO-TO-CLUSTER-> 4
    4.2 IF  CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED  AND IF
            #VP IN CHARGE OF THE SYSTEM HAS ACCESS  AND IF
            SECTOR #VP. OF THE SCHOOL HAS ACCESS  AND IF
            RECORDS OFFICE OF THE SCHOOL HAS ACCESS  AND IF
            TEACHERS HAVE ACCESS  AND IF
            STUDENTS AND PARENTS HAVE INDIRECT ACCESS  AND IF
            OBJECTIVE IS AN EFFICIENT STUDENT AND PARENT COUNSELLING  AND IF
            IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING  AND IF
            OUTPUT VOLUME IS SMALL

4.3==QUERRY THE CLASS #NF INTERACTIVELY                      ==    --THEN-GO-TO-CLUSTER-> 4
    4.3 IF  CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED  AND IF
            #VP IN CHARGE OF THE SYSTEM HAS ACCESS  AND IF
            SECTOR #VP. OF THE SCHOOL HAS ACCESS  AND IF
            TEACHERS HAVE ACCESS  AND IF
            OBJECTIVE IS AN EFFICIENT CLASS MANAGEMENT  AND IF
            IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING  AND IF
            OUTPUT VOLUME IS SMALL

4.4==QUERRY THE COURSE #NF PSEUDOBATCH MODE                  ==    --THEN-GO-TO-CLUSTER-> 4
    4.4 IF  OBJECTIVE IS AN EFFICIENT SCHOOL MANAGEMENT  AND IF
            COURSE INFORMATION IS NEEDED  AND IF
            #VP IN CHARGE OF THE SYSTEM HAS ACCESS  AND IF
            NOT  IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING  AND IF
            NOT  OUTPUT VOLUME IS SMALL

4.5==QUERRY THE STUDENT #NF PSEUDOBATCH MODE                 ==    --THEN-GO-TO-CLUSTER-> 4
    4.5 IF  OBJECTIVE IS AN EFFICIENT SCHOOL MANAGEMENT  AND IF
            STUDENT INFORMATION IS NEEDED  AND IF
            #VP IN CHARGE OF THE SYSTEM HAS ACCESS  AND IF
            NOT  IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING  AND IF
            NOT  OUTPUT VOLUME IS SMALL

4.6==QUERRY THE CLASS #NF PSEUDOBATCH MODE                   ==    --THEN-GO-TO-CLUSTER-> 4
    4.6 IF  OBJECTIVE IS AN EFFICIENT SCHOOL MANAGEMENT  AND IF
            OBJECTIVE IS AN EFFICIENT CLASS MANAGEMENT  AND IF
            CLASS INFORMATION IS NEEDED  AND IF
            #VP IN CHARGE OF THE SYSTEM HAS ACCESS  AND IF
            NOT  IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING  AND IF
            NOT  OUTPUT VOLUME IS SMALL

4.7==DATA-BASE QUERRY ORGANIZATION IS COMPLETE               ==    --THEN-GO-TO-CLUSTER-> 0
    4.7 IF  ALL SUBFUNCTIONS IN THIS FUNCTION (CLUSTER) ARE MET
```

4. ABL-ROUTINE MODELING;

```
*************************************************************************
*                                                                       *
*    MODELING  # OR THE OVERALL SYSTEM DESIGN AND MODELING OF THIS SYSTEM *
*    ------------------------------------------------------------------ *
*                                                                       *
*  THIS TABLE PROVIDES AN OVERVIEW IN THE MODELING STAGE OF THE SYSTEM.  *
*  #MODELING  HAS 44 ACTION PROCEDURES AND 64 PREDICATES.  THESE ACTIONS *
*  ARE GROUPED INTO FUNCTIONS AND SUBFUNCTIONS AND FORM THE HIERARCHICALLY *
*  INTERRELATED BUILDING BLOCKS OF THE WHOLE SYSTEM. THE PREDICATES ACT AS *
*  CRITERIONS FOR A PARTICULAR SUBFUNCTION. THEY DEFINE THE CONSTRAINTS  *
*  THE ACTIONS IN THE PARTICULAR SUBFUNTION MUST OBEY.  UNLESS THESE     *
*  CRITERIONS ARE SATISFIED, THAT PARTICULAR SUBFUNCTION IS NOT ADEQUATELY *
*  DESIGNED TO MEET ITS PURPOSE.                                        *
*                                                                       *
*  ONCE ALL THE ACTIONS ARE DESIGNED, THESE SAME ACTIONS WILL BE OPERATIONALLY *
*  SEQUENCED AND INTERFACED IN THE NEXT TABLE CALLED  #SYSTALL          *
*                                                                       *
*************************************************************************
```

1.=== ORGANIZE THE BODY OF THE SYSTEM ===* < 1, 1, 1, 1, 1, 1, 1, 1, 2 >

1.1==DECENTRELIZED PROCESSING IS NEEDED == --THEN-GO-TO-CLUSTER-> 1
1.1 IF PRODUCTION CYCLE PERIOD IS DEFINED AS 1-2 DAYS LONG AND IF
DATA COLLECTION PERIOD IS DEFINED AS 30-40 MINUTES LONG AND IF
INEXPENSIVE TERMINAL IS AVAILABLE AND IF
INEXPENSIVE DATA COMMUNICATION IS AVAILABLE AND IF
REMOTE TIME-SHARING COMPUTER IS AVAILABLE
A1> INSTALL THE TERMINAL IN SCHOOL

1.2==SET UP RANDOM ACCESS DATA-BASE == --THEN-GO-TO-CLUSTER-> 1
1.2 IF ON-LINE SYSTEM ACCESS IS NEEDED AND IF
DISK-STORAGE IS AVAILABLE AND IF
INTERACTIVE QUERRY AND UPDATE IS NEEDED AND IF
SYSTEM USAGE IS STUDIED
A2> SET UP THE COURSE MASTER FILE
A3> SET UP THE STUDENT MASTER FILE
A4> SET UP THE CLASS MASTER FILE

1.3==MODULARIZE SYSTEM == --THEN-GO-TO-CLUSTER-> 1
1.3 IF NOT SYSTEM OPERATING SPECIALIST IS AVAILABLE IN SCHOOL AND IF
SYSTEM OPERATING TASK HAS TO BE SHARED BY REGULAR STAFF AND IF
USER SYSTEM ACCESS ORGANIZATION IS DEFINED AND IF
CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED AND IF
SYSTEM MODULARIZATION IS DEFINED
A2> SET UP THE COURSE MASTER FILE
A3> SET UP THE STUDENT MASTER FILE
A4> SET UP THE CLASS MASTER FILE
A5> SET UP COURSE MASTER FILE UPDATER PROGRAM
A6> SET UP STUDENT MASTER FILE UPDATER PROGRAM
A7> SET UP COURSE MASTER FILE QUERRY PROGRAM
A8> SET UP STUDENT MASTER FILE QUERRY PROGRAM
A9> SET UP CLASS MASTER FILE QUERRY PROGRAM
A21> SET UP DAILY INITIALIZATION INTERACTIVE PROGRAM
A22> SET UP DATA COLLECTION PSEUDOBATCH PROGRAM
A23> SET UP #AIS, DATA PROCESSING PSEUDOBATCH PROGRAM
A24> SET UP PROCEDURE TO PRINT SPOOLED OUTPUT

CONTINUED ON THE NEXT PAGE

1.4==SET UP INTERACTIVE #08 UPDATING NAIVE USER INTERFACE== ? --THEN-GO-TO-CLUSTER-> 1
1.4 IF HIGH DEGREE OF DATA-BASE ACCURACY IS NEEDED AND IF
HIGH VOLUME OF CHANGES OCCUR IN THE DATA-BASE AND IF
DATA-BASE CHANGES HAVE TO BE IMPLEMENTED IN 1-2 DAYS AND IF
NOT SYSTEM OPERATING SPECIALIST IS AVAILABLE IN SCHOOL AND IF
SYSTEM OPERATING TASK HAS TO BE SHARED BY REGULAR STAFF AND IF
USER SYSTEM ACCESS ORGANIZATION IS DEFINED AND IF
CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED AND IF
INTERACTIVE UPDATING LANGUAGE FOR THE COURSE #MF, IS NEEDED AND IF
INTERACTIVE UPDATING LANGUAGE FOR THE STUDENT #MF, IS NEEDED AND IF
CLASS MASTER FILE UPDATING CAN BE DONE AUTOMATICALLY
A5> SET UP COURSE MASTER FILE UPDATER PROGRAM
A6> SET UP STUDENT MASTER FILE UPDATER PROGRAM
A10> UPDATE COURSE MASTER FILE INTERACTIVELY
A11> UPDATE STUDENT MASTER FILE INTERACTIVELY
A12> UPDATE CLASS MASTER FILE INTERACTIVELY

1.5==SET UP INTERACTIVE #06 QUERRY NAIVE USER INTERFACE == --THEN-GO-TO-CLUSTER-> 1
1.5 IF IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING AND IF
OUTPUT VOLUME IS SMALL AND IF
NOT SYSTEM OPERATING SPECIALIST IS AVAILABLE IN SCHOOL AND IF
SYSTEM OPERATING TASK HAS TO BE SHARED BY REGULAR STAFF AND IF
USER SYSTEM ACCESS ORGANIZATION IS DEFINED AND IF
CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED AND IF
INTERACTIVE QUERRY LANGUAGE FOR THE COURSE #MF, IS NEEDED AND IF
INTERACTIVE QUERRY LANGUAGE FOR THE STUDENT #MF, IS NEEDED AND IF
INTERACTIVE QUERRY LANGUAGE FOR THE CLASS #MF, IS NEEDED
A7> SET UP COURSE MASTER FILE QUERRY PROGRAM
A8> SET UP STUDENT MASTER FILE QUERRY PROGRAM
A9> SET UP CLASS MASTER FILE QUERRY PROGRAM
A33> QUERRY COURSE MASTER FILE INTERACTIVELY
A34> QUERRY STUDENT MASTER FILE INTERACTIVELY
A35> QUERRY CLASS MASTER FILE INTERACTIVELY

1.6==SET UP PSEUDOBATCH #08 QUERRY POSSIBILITY == --THEN-GO-TO-CLUSTER-> 1
1.6 IF NOT IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING AND IF
NOT OUTPUT VOLUME IS SMALL AND IF
NOT SYSTEM OPERATING SPECIALIST IS AVAILABLE IN SCHOOL AND IF
SYSTEM OPERATING TASK HAS TO BE SHARED BY REGULAR STAFF AND IF
USER SYSTEM ACCESS ORGANIZATION IS DEFINED AND IF
CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED AND IF
INTERACTIVE QUERRY LANGUAGE FOR THE COURSE #MF, IS NEEDED AND IF
INTERACTIVE QUERRY LANGUAGE FOR THE STUDENT #MF, IS NEEDED AND IF
INTERACTIVE QUERRY LANGUAGE FOR THE CLASS #MF, IS NEEDED
A36> WRITE A FILE IN COURSE MASTER FILE QUERRY LANGUAGE
A37> QUERRY COURSE MASTER FILE PSEUDOBATCH MODE
A39> WRITE A FILE IN STUDENT MASTER FILE QUERRY LANGUAGE
A40> QUERRY STUDENT MASTER FILE PSEUDOBATCH MODE
A41> WRITE A FILE IN CLASS MASTER FILE QUERRY LANGUAGE
A42> QUERRY CLASS MASTER FILE PSEUDOBATCH MODE
A38> PRINT SPOOLED OUTPUT

CONTINUED ON THE NEXT PAGE

```
1.7==SET UP ACCESSORIES FOR $ATS                    ==    --THEN-GO-TO-CLUSTER-> 1
    1.7 IF  HIGH VOLUME OF TRUANCY TRANSACTION OCCUR (100-250 A LECTURE) AND IF
            DATA COLLECTION PERIOD IS DEFINED AS 30-40 MINUTES LONG  AND IF
            DATA GENERATION PERIOD IS 30-60 SECONDS  AND IF
            ERROR GENERATION POSSIBILITY IN TRANSACTION DATA IS MINIMAL  AND IF
            INEXPENSIVE OPTICAL MARK READER IS FOUND FOR THE TERMINAL
              A42> QUERRY CLASS MASTER FILE PSEUDOBATCH MODE
              A43> PRODUCE # ABS-CARDS FOR THE CLASS
              A16> PRODUCE TEACHERS DATA GENERATION KITS
              A17> SET UP DATA COLLECTION ROUTES IN SCHOOL
              A18> INSTALL OPTICAL MARK READER ON TERMINAL
              A19> VERIFY DATA DISTRIBUTION ROUTES
              A20> DETERMINE DATA USAGE PARAMETERS

1.8==SET UP PRODUCTION CYCLE PROGRAMS               ==    --THEN-GO-TO-CLUSTER-> 1
    1.8 IF  REGULAR $ATS OPERATOR IS AVAILABLE FROM STAFF  AND IF
            PRODUCTION CYCLE PERIOD IS DEFINED AS 1-2 DAYS LONG  AND IF
            DATA COLLECTION PERIOD IS DEFINED AS 30-40 MINUTES LONG  AND IF
            DATA TRANSMISSION IS BAD SOMETIMES  AND IF
            OUTPUT PRINTING ON THE TERMINAL SOMETIMES HAS TO WAIT
              A21> SET UP DAILY INITIALIZATION INTERACTIVE PROGRAM
              A22> SET UP DATA COLLECTION PSEUDOBATCH PROGRAM
              A23> SET UP  $ATS. DATA PROCESSING PSEUDOBATCH PROGRAM
              A24> SET UP PROCEDURE TO PRINT SPOOLED OUTPUT

1.9==BODY OF THE BASIC SYSTEM IS ORGANIZED          ==    --THEN-GO-TO-CLUSTER-> 2
    1.9 IF  ALL SUBFUNCTIONS IN THIS FUNCTION (CLUSTER) ARE MET
              A44> NO ACTION- NEXT CLUSTER

2.=== ORGANIZE THE REGULAR RUN OF THE SYSTEM              ===$          < 2, 2, 2, 2, 2, 2, 3 >

2.1==$ATS DAILY DATA USAGE PARAMETER DEFINITION      ==    --THEN-GO-TO-CLUSTER-> 2
    2.1 IF  VARIABLE LENGTH OUTPUT IS NEEDED
              A25> PERFORM DAILY SYSTEM INITIALIZATION

2.2==$ATS DAILY TRANSACTION DATA GENERATION          ==    --THEN-GO-TO-CLUSTER-> 2
    2.2 IF  DATA GENERATION PROCESS IS ACCEPTED BY THE TEACHERS  AND IF
            ERROR GENERATION POSSIBILITY IN TRANSACTION DATA IS MINIMAL  AND IF
            DATA GENERATION PERIOD IS 30-60 SECONDS
              A26> PERFORM TRANSACTION DATA GENERATION

2.3==$ATS TRANSACTION DATA COLLECTION                ==    --THEN-GO-TO-CLUSTER-> 2
    2.3 IF  DATA COLLECTION PROCESS IS ACCEPTED BY THE ADMINISTRATION  AND IF
            DATA COLLECTION ROUTE IS ORGANIZED  AND IF
            DATA COLLECTION PERIOD IS DEFINED AS 30-40 MINUTES LONG  AND IF
            DATA COLLECTION ERRORS ARE MINIMAL  AND IF
            REGULAR $ATS OPERATOR IS AVAILABLE FROM STAFF
              A27> PERFORM TRANSACTION DATA COLLECTION

CONTINUED ON THE NEXT PAGE
```

2.4==#ATS TRANSACTION DATA PROCESSING == --THEN-GO-TO-CLUSTER-> 2
2.4 IF EXTENSIVE TRANSACTION STORAGE AND ANALYSIS IS REQUIRED AND IF
REGULAR #ATS OPERATOR IS AVAILABLE FROM STAFF AND IF
PROCEDURE IS SIMPLE AND IF
PROCEDURE IS SHORT AND IF
OUTPUT CAN BE SPOOLED AND IF
PRINTING OF SPOOLED OUTPUT IS NOT MORE THAN 1-2 HOURS
A28> PERFORM DATA PROCESSING (REPORTING AND STORAGE)

2.5==#ATS DAILY DATA DISTRIBUTION == --THEN-GO-TO-CLUSTER-> 2
2.5 IF PROCEDURE IS SIMPLE
A29> PERFORM DATA DISTRIBUTION

2.6==#ATS DAILY OUTPUT USAGE == --THEN-GO-TO-CLUSTER-> 2
2.6 IF OUTPUT VOLUME IS ADEQUATE TO DAILY USAGE AND IF
FALSE NEGATIVE ERROR RATE IS BELOW 10% RATE AND IF
FALSE POSITIVE ERROR RATE IS BELOW 0.1% RATE AND IF
PROVIDED INFORMATION CAN BE READILY USED IN DECISION MAKING AND IF
INFORMATION VALUE IS MAXIMUM OPERATIONAL VS. OVERALL DELAY
A30> VERIFY OUTPUT USAGE IN DECISION MAKING

2.7==#ATS SYSTEM IS ORGANIZED == --THEN-GO-TO-CLUSTER-> 3
2.7 IF ALL SUBFUNCTIONS IN THIS FUNCTION (CLUSTER) ARE MET
A44> NO ACTION- NEXT CLUSTER

3.=== DATA UPDATE ORGANIZATION ===$ < 3, 3, 3, 4 >

3.1==UPDATE THE COURSE #MF == --THEN-GO-TO-CLUSTER-> 3
3.1 IF HIGH DEGREE OF DATA-BASE ACCURACY IS NEEDED AND IF
USER SYSTEM ACCESS ORGANIZATION IS DEFINED AND IF
#VP IN CHARGE OF THE SYSTEM HAS ACCESS AND IF
REDUNDANT DATA IN THE DATA-BASE ARE AUTOMATICALLY UPDATED
A10> UPDATE COURSE MASTER FILE INTERACTIVELY

3.2==UPDATE THE STUDENT #MF == --THEN-GO-TO-CLUSTER-> 3
3.2 IF HIGH DEGREE OF DATA-BASE ACCURACY IS NEEDED AND IF
USER SYSTEM ACCESS ORGANIZATION IS DEFINED AND IF
RECORDS OFFICE OF THE SCHOOL HAS ACCESS AND IF
REDUNDANT DATA IN THE DATA-BASE ARE AUTOMATICALLY UPDATED
A11> UPDATE STUDENT MASTER FILE INTERACTIVELY

3.3==UPDATE ATTENDANCE RELATED DATA == --THEN-GO-TO-CLUSTER-> 3
3.3 IF HIGH DEGREE OF DATA-BASE ACCURACY IS NEEDED AND IF
USER SYSTEM ACCESS ORGANIZATION IS DEFINED AND IF
SECTION #VP, OF THE SCHOOL HAS ACCESS AND IF
CLASS RELATED DATA CHANGED OR SOME #ABS-CARDS GOT LOST AND IF
DATA-BASE DATA IS UPDATED AND IF
CARD PUNCHING AND INTERPRETING IS AVAILABLE
A30> QUERRY CLASS MASTER FILE INTERACTIVELY
A31> PUNCH NEW # ABS-CARDS
A32> UPDATE TEACHERS DATA GENERATION KITS

CONTINUED ON THE NEXT PAGE

3.4=UPDATE ORGANIZATION IS COMPLETE == --THEN-GO-TO-CLUSTER-> 4
3.4 IF ALL SUBFUNCTIONS IN THIS FUNCTION (CLUSTER) ARE MET
A44> DO ACTION- NEXT CLUSTER

4.=== DATA-BASE QUERRY ORGANIZATION ===# < 4, 4, 4, 4, 4, 4, 0 >

4.1=QUERRY THE COURSE #MF INTERACTIVELY == --THEN-GO-TO-CLUSTER-> 4
4.1 IF CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED AND IF
#VP IN CHARGE OF THE SYSTEM HAS ACCESS AND IF
OBJECTIVE IS AN EFFICIENT SCHOOL MANAGEMENT AND IF
IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING AND IF
OUTPUT VOLUME IS SMALL
A33> QUERRY COURSE MASTER FILE INTERAGTIVELY

4.2=QUERRY THE STUDENT #MF INTERACTIVELY == --THEN-GO-TO-CLUSTER-> 4
4.2 IF CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED AND IF
#VP IN CHARGE OF THE SYSTEM HAS ACCESS AND IF
SECTOR #VP. OF THE SCHOOL HAS ACCESS AND IF
RECORDS OFFICE OF THE SCHOOL HAS ACCESS AND IF
TEACHERS HAVE ACCESS AND IF
STUDENTS AND PARENTS HAVE INDIRECT ACCESS AND IF
OBJECTIVE IS AN EFFICIENT STUDENT AND PARENT COUNSELLING AND IF
IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING AND IF
OUTPUT VOLUME IS SMALL
A34> QUERRY STUDENT MASTER FILE INTERACTIVELY

4.3=QUERRY THE CLASS #MF INTERACTIVELY == --THEN-GO-TO-CLUSTER-> 4
4.3 IF CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED AND IF
#VP IN CHARGE OF THE SYSTEM HAS ACCESS AND IF
SECTOR #VP. OF THE SCHOOL HAS ACCESS AND IF
TEACHERS HAVE ACCESS AND IF
OBJECTIVE IS AN EFFICIENT CLASS MANAGEMENT AND IF
IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING AND IF
OUTPUT VOLUME IS SMALL
A35> QUERRY CLASS MASTER FILE INTERACTIVELY

4.4=QUERRY THE COURSE #MF PSEUDOBATCH MODE == --THEN-GO-TO-CLUSTER-> 4
4.4 IF OBJECTIVE IS AN EFFICIENT SCHOOL MANAGEMENT AND IF
COURSE INFORMATION IS NEEDED AND IF
#VP IN CHARGE OF THE SYSTEM HAS ACCESS AND IF
NOT IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING AND IF
NOT OUTPUT VOLUME IS SMALL
A36> WRITE A FILE IN COURSE MASTER FILE QUERRY LANGUAGE
A37> QUERRY COURSE MASTER FILE PSEUDOBATCH MODE
A38> PRINT SPOOLED OUTPUT

CONTINUED ON THE NEXT PAGE

```
4.5==QUERRY THE STUDENT MF PSEUDOBATCH MODE              ==     --THEN-GO-TO-CLUSTER-> 4
    4.5 IF  OBJECTIVE IS AN EFFICIENT SCHOOL MANAGEMENT  AND IF
            STUDENT INFORMATION IS NEEDED  AND IF
            AVP IN CHARGE OF THE SYSTEM HAS ACCESS  AND IF
            NOT  IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING  AND IF
            NOT  OUTPUT VOLUME IS SMALL
               A37> WRITE A FILE IN STUDENT MASTER FILE QUERRY LANGUAGE
               A40> QUERRY STUDENT MASTER FILE PSEUDOBATCH MODE
               A38> PRINT SPOOLED OUTPUT

4.6==QUERRY THE CLASS MF PSEUDOBATCH MODE                ==     --THEN-GO-TO-CLUSTER-> 4
    4.6 IF  OBJECTIVE IS AN EFFICIENT SCHOOL MANAGEMENT  AND IF
            OBJECTIVE IS AN EFFICIENT CLASS MANAGEMENT  AND IF
            CLASS INFORMATION IS NEEDED  AND IF
            AVP IN CHARGE OF THE SYSTEM HAS ACCESS  AND IF
            NOT  IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING. AND IF
            NOT  OUTPUT VOLUME IS SMALL
               A41> WRITE A FILE IN CLASS MASTER FILE QUERRY LANGUAGE
               A42> QUERRY CLASS MASTER FILE PSEUDOBATCH MODE
               A38> PRINT SPOOLED OUTPUT


4.7==DATA-BASE QUERRY ORGANIZATION IS COMPLETE           ==     --THEN-GO-TO-CLUSTER-> 0
    4.7 IF  ALL SUBFUNCTIONS IN THIS FUNCTION (CLUSTER) ARE MET
               A44> NO ACTION- NEXT CLUSTER
```

4. ABL-TRANSFORM MODELING;

```
*************************************************************************
*                                                                       *
*     MODELING  # OR THE OVERALL SYSTEM DESIGN AND MODELING OF THIS SYSTEM *
*     ------------------------------------------------------------------ *
*                                                                       *
*  THIS TABLE PROVIDES AN OVERVIEW IN THE MODELING STAGE OF THE SYSTEM.  *
*  #MODELING  HAS 44 ACTION PROCEDURES AND 64 PREDICATES.  THESE ACTIONS *
*  ARE GROUPED INTO FUNCTIONS AND SUBFUNCTIONS AND FORM THE HIERARCHICALLY *
*  INTERRELATED BUILDING BLOCKS OF THE WHOLE SYSTEM. THE PREDICATES ACT AS *
*  CRITERIONS FOR A PARTICULAR SUBFUNCTION, THEY DEFINE THE CONSTRAINTS  *
*  THE ACTIONS IN THE PARTICULAR SUBFUNTION MUST OBEY.  UNLESS THESE     *
*  CRITERIONS ARE SATISFIED, THAT PARTICULAR SUBFUNCTION IS NOT ADEQUATELY *
*  DESIGNED TO MEET ITS PURPOSE.                                        *
*                                                                       *
*  ONCE ALL THE ACTIONS ARE DESIGNED, THESE SAME ACTIONS WILL BE OPERATIONALLY *
*  SEQUENCED AND INTERFACED IN THE NEXT TABLE CALLED  $SYSTALL          *
*                                                                       *
*************************************************************************
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	(ALTERNATIVES)
																	CLUSTERS -
C1	1	1	1	1	1	1	1	1	2	.	.	.	.	.	.	.	=== ORGANIZE THE BODY OF THE SYSTEM
C2	.	.	.	.	.	.	.	.	.	2	2	2	2	2	2	3	=== ORGANIZE THE REGULAR RUN OF THE SYSTEM
C3	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	=== DATA UPDATE ORGANIZATION
C4	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	=== DATA-BASE QUERRY ORGANIZATION
																	PREDICATES -
P1	Y	-	-	-	-	-	-	Y	-	-	-	-	-	-	-	-	PRODUCTION CYCLE PERIOD IS DEFINED AS 1-2 DAYS LONG
P2	Y	-	-	-	-	-	Y	Y	-	-	-	Y	-	-	-	-	DATA COLLECTION PERIOD IS DEFINED AS 30-40 MINUTES LONG
P3	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	INEXPENSIVE TERMINAL IS AVAILABLE
P4	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	INEXPENSIVE DATA COMMUNICATION IS AVAILABLE
P5	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	REMOTE TIME-SHARING COMPUTER IS AVAILABLE
P6	-	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	ON-LINE SYSTEM ACCESS IS NEEDED
P7	-	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	DISK-STORAGE IS AVAILABLE
P8	-	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	INTERACTIVE QUERRY AND UPDATE IS NEEDED
P9	-	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	SYSTEM USAGE IS STUDIED
P10	-	-	N	N	N	N	-	-	-	-	-	-	-	-	-	-	SYSTEM OPERATING SPECIALIST IS AVAILABLE IN SCHOOL
P11	-	-	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	SYSTEM OPERATING TASK HAS TO BE SHARED BY REGULAR STAFF
P12	-	-	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	USER SYSTEM ACCESS ORGANIZATION IS DEFINED
P13	-	-	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED
P14	-	-	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	SYSTEM MODULARIZATION IS DEFINED
P15	-	-	-	Y	-	-	-	-	-	-	-	-	-	-	-	-	HIGH DEGREE OF DATA-BASE ACCURACY IS NEEDED
P16	-	-	-	Y	-	-	-	-	-	-	-	-	-	-	-	-	HIGH VOLUME OF CHANGES OCCUR IN THE DATA-BASE
P17	-	-	-	Y	-	-	-	-	-	-	-	-	-	-	-	-	DATA-BASE CHANGES HAVE TO BE IMPLEMENTED IN 1-2 DAYS
P18	-	-	-	Y	-	-	-	-	-	-	-	-	-	-	-	-	INTERACTIVE UPDATING LANGUAGE FOR THE COURSE #MF. IS NEEDED
P19	-	-	-	Y	-	-	-	-	-	-	-	-	-	-	-	-	INTERACTIVE UPDATING LANGUAGE FOR THE STUDENT #MF. IS NEEDED
P20	-	-	-	Y	-	-	-	-	-	-	-	-	-	-	-	-	CLASS MASTER FILE UPDATING CAN BE DONE AUTOMATICALLY
P21	-	-	-	-	Y	N	-	-	-	-	-	-	-	-	-	-	IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING
P22	-	-	-	-	Y	N	-	-	-	-	-	-	-	-	-	-	OUTPUT VOLUME IS SMALL
P23	-	-	-	-	Y	Y	-	-	-	-	-	-	-	-	-	-	INTERACTIVE QUERRY LANGUAGE FOR THE COURSE #MF. IS NEEDED
P24	-	-	-	-	Y	Y	-	-	-	-	-	-	-	-	-	-	INTERACTIVE QUERRY LANGUAGE FOR THE STUDENT #MF. IS NEEDED
P25	-	-	-	-	Y	Y	-	-	-	-	-	-	-	-	-	-	INTERACTIVE QUERRY LANGUAGE FOR THE CLASS #MF. IS NEEDED

CONTINUED ON THE NEXT PAGE

P26	-	-	-	-	-	-	Y	-	-	-	-	-	-	-	-	HIGH VOLUME OF TRUANCY TRANSACTION OCCUR (100-250 A LECTURE)
P27	-	-	-	-	-	-	Y	-	-	-	Y	-	-	-	-	DATA GENERATION PERIOD IS 30-60 SECONDS
P28	-	-	-	-	-	-	Y	-	-	-	Y	-	-	-	-	ERROR GENERATION POSSIBILITY IN TRANSACTION DATA IS MINIMAL
P29	-	-	-	-	-	-	Y	-	-	-	-	-	-	-	-	INEXPENSIVE OPTICAL MARK READER IS FOUND FOR THE TERMINAL
P30	-	-	-	-	-	-	-	Y	-	-	-	Y	Y	-	-	REGULAR #ATS OPERATOR IS AVAILABLE FROM STAFF
P31	-	-	-	-	-	-	-	Y	-	-	-	-	-	-	-	DATA TRANSMISSION IS BAD SOMETIMES
P32	-	-	-	-	-	-	-	Y	-	-	-	-	-	-	-	OUTPUT PRINTING ON THE TERMINAL SOMETIMES HAS TO WAIT
P33	-	-	-	-	-	-	-	-	-	Y	-	-	-	-	-	VARIABLE LENGTH OUTPUT IS NEEDED
P34	-	-	-	-	-	-	-	-	-	-	Y	-	-	-	-	DATA GENERATION PROCESS IS ACCEPTED BY THE TEACHERS
P35	-	-	-	-	-	-	-	-	-	-	-	Y	-	-	-	DATA COLLECTION PROCESS IS ACCEPTED BY THE ADMINISTRATION
P36	-	-	-	-	-	-	-	-	-	-	-	Y	-	-	-	DATA COLLECTION ROUTE IS ORGANIZED
P37	-	-	-	-	-	-	-	-	-	-	-	Y	-	-	-	DATA COLLECTION ERRORS ARE MINIMAL
P38	-	-	-	-	-	-	-	-	-	-	-	-	Y	-	-	EXTENSIVE TRANSACTION STORAGE AND ANALYSIS IS REQUIRED
P39	-	-	-	-	-	-	-	-	-	-	-	-	Y	Y	-	PROCEDURE IS SIMPLE
P40	-	-	-	-	-	-	-	-	-	-	-	-	Y	-	-	PROCEDURE IS SHORT
P41	-	-	-	-	-	-	-	-	-	-	-	-	Y	-	-	OUTPUT CAN BE SPOOLED
P42	-	-	-	-	-	-	-	-	-	-	-	-	Y	-	-	PRINTING OF SPOOLED OUTPUT IS NOT MORE THAN 1-2 HOURS
P43	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	DISTRIBUTION DELAY IS DEFINED AS 2 HOURS
P44	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	-	OUTPUT VOLUME IS ADEQUATE TO DAILY USAGE
P45	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	-	FALSE NEGATIVE ERROR RATE IS BELOW 10% RATE
P46	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	-	FALSE POSITIVE ERROR RATE IS BELOW 0.1% RATE
P47	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	-	PROVIDED INFORMATION CAN BE READILY USED IN DECISION MAKING
P48	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	-	INFORMATION VALUE IS MAXIMUM OPERATIONAL VS. OVERALL DELAY
P49	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	#VP IN CHARGE OF THE SYSTEM HAS ACCESS
P50	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	REDUNDANT DATA IN THE DATA-BASE ARE AUTOMATICALLY UPDATED
P51	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RECORDS OFFICE OF THE SCHOOL HAS ACCESS
P52	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	SECTOR #VP. OF THE SCHOOL HAS ACCESS
P53	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	CLASS RELATED DATA CHANGED OR SOME #ABS-CARDS GOT LOST
P54	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	DATA-BASE DATA IS UPDATED
P55	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	CARD PUNCHING AND INTERPRETING IS AVAILABLE
P56	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	OBJECTIVE IS AN EFFICIENT SCHOOL MANAGEMENT
P57	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	TEACHERS HAVE ACCESS
P58	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	STUDENTS AND PARENTS HAVE INDIRECT ACCESS
P59	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	OBJECTIVE IS AN EFFICIENT STUDENT AND PARENT COUNSELLING
P60	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	OBJECTIVE IS AN EFFICIENT CLASS MANAGEMENT
P61	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	COURSE INFORMATION IS NEEDED
P62	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	STUDENT INFORMATION IS NEEDED
P63	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	CLASS INFORMATION IS NEEDED
P64	-	-	-	-	-	-	-	-	Y	-	-	-	-	-	Y	ALL SUBFUNCTIONS IN THIS FUNCTION (CLUSTER) ARE MET

CONTINUED ON THE NEXT PAGE

																	ACTIONS -
A1	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A1⟩	INSTALL THE TERMINAL IN SCHOOL
A2	.	1	1	.	.	.	.	.	.	.	.	.	.	.	.	A2⟩	SET UP THE COURSE MASTER FILE
A3	.	2	2	.	.	.	.	.	.	.	.	.	.	.	.	A3⟩	SET UP THE STUDENT MASTER FILE
A4	.	3	3	.	.	.	.	.	.	.	.	.	.	.	.	A4⟩	SET UP THE CLASS MASTER FILE
A5	.	.	4	1	.	.	.	.	.	.	.	.	.	.	.	A5⟩	SET UP COURSE MASTER FILE UPDATER PROGRAM
A6	.	.	5	2	.	.	.	.	.	.	.	.	.	.	.	A6⟩	SET UP STUDENT MASTER FILE UPDATER PROGRAM
A7	.	.	6	.	1	.	.	.	.	.	.	.	.	.	.	A7⟩	SET UP COURSE MASTER FILE QUERRY PROGRAM
A8	.	.	7	.	2	.	.	.	.	.	.	.	.	.	.	A8⟩	SET UP STUDENT MASTER FILE QUERRY PROGRAM
A9	.	.	8	.	3	.	.	.	.	.	.	.	.	.	.	A9⟩	SET UP CLASS MASTER FILE QUERRY PROGRAM
A10	.	.	.	3	.	.	.	.	.	.	.	.	.	.	.	A10⟩	UPDATE COURSE MASTER FILE INTERACTIVELY
A11	.	.	.	4	.	.	.	.	.	.	.	.	.	.	.	A11⟩	UPDATE STUDENT MASTER FILE INTERACTIVELY
A12	.	.	.	5	.	.	.	.	.	.	.	.	.	.	.	A12⟩	UPDATE CLASS MASTER FILE INTERACTIVELY
A13	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A13⟩	VERIFY COURSE MASTER FILE DATA
A14	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A14⟩	VERIFY STUDENT MASTER FILE DATA
A15	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A15⟩	VERIFY CLASS MASTER FILE DATA
A16	.	.	.	.	.	3	.	.	.	.	.	.	.	.	.	A16⟩	PRODUCE TEACHERS DATA GENERATION KITS
A17	.	.	.	.	.	4	.	.	.	.	.	.	.	.	.	A17⟩	SET UP DATA COLLECTION ROUTES IN SCHOOL
A18	.	.	.	.	.	5	.	.	.	.	.	.	.	.	.	A18⟩	INSTALL OPTICAL MARK READER ON TERMINAL
A19	.	.	.	.	.	6	.	.	.	.	.	.	.	.	.	A19⟩	VERIFY DATA DISTRIBUTION ROUTES
A20	.	.	.	.	.	7	.	.	.	.	.	.	.	.	.	A20⟩	DETERMINE DATA USAGE PARAMETERS
A21	.	.	9	.	.	.	1	.	.	.	.	.	.	.	.	A21⟩	SET UP DAILY INITIALIZATION INTERACTIVE PROGRAM
A22	.	.	10	.	.	.	2	.	.	.	.	.	.	.	.	A22⟩	SET UP DATA COLLECTION PSEUDOBATCH PROGRAM
A23	.	.	11	.	.	.	3	.	.	.	.	.	.	.	.	A23⟩	SET UP DATS. DATA PROCESSING PSEUDOBATCH PROGRAM
A24	.	.	12	.	.	.	4	.	.	.	.	.	.	.	.	A24⟩	SET UP PROCEDURE TO PRINT SPOOLED OUTPUT
A25	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	A25⟩	PERFORM DAILY SYSTEM INITIALIZATION
A26	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	A26⟩	PERFORM TRANSACTION DATA GENERATION
A27	.	.	.	.	4	.	.	.	.	.	1	.	.	.	.	A27⟩	PERFORM TRANSACTION DATA COLLECTION
A28	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	A28⟩	PERFORM DATA PROCESSING (REPORTING AND STORAGE)
A29	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	A29⟩	PERFORM DATA DISTRIBUTION
A30	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	A30⟩	VERIFY OUTPUT USAGE IN DECISION MAKING
A31	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A31⟩	PUNCH NEW & ABS-CARDS
A32	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A32⟩	UPDATE TEACHERS DATA GENERATION KITS
A33	.	.	.	.	4	.	.	.	.	.	.	.	.	.	.	A33⟩	QUERRY COURSE MASTER FILE INTERACTIVELY
A34	.	.	.	.	5	.	.	.	.	.	.	.	.	.	.	A34⟩	QUERRY STUDENT MASTER FILE INTERACTIVELY
A35	.	.	.	.	6	.	.	.	.	.	.	.	.	.	.	A35⟩	QUERRY CLASS MASTER FILE INTERACTIVELY
A36	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	A36⟩	WRITE A FILE IN COURSE MASTER FILE QUERRY LANGUAGE
A37	.	.	.	.	.	2	.	.	.	.	.	.	.	.	.	A37⟩	QUERRY COURSE MASTER FILE PSEUDOBATCH MODE
A38	.	.	.	.	.	7	.	.	.	.	.	.	.	.	.	A38⟩	PRINT SPOOLED OUTPUT
A39	.	.	.	.	.	3	.	.	.	.	.	.	.	.	.	A39⟩	WRITE A FILE IN STUDENT MASTER FILE QUERRY LANGUAGE
A40	.	.	.	.	.	4	.	.	.	.	.	.	.	.	.	A40⟩	QUERRY STUDENT MASTER FILE PSEUDOBATCH MODE
A41	.	.	.	.	.	5	.	.	.	.	.	.	.	.	.	A41⟩	WRITE A FILE IN CLASS MASTER FILE QUERRY LANGUAGE
A42	.	.	.	.	.	6	1	.	.	.	.	.	.	.	.	A42⟩	QUERRY CLASS MASTER FILE PSEUDOBATCH MODE
A43	.	.	.	.	.	.	2	.	.	.	.	.	.	.	.	A43⟩	PRODUCE & ABS-CARDS FOR THE CLASS
A44	.	.	.	.	.	.	.	.	1	.	.	.	.	.	1	A44⟩	NO ACTION- NEXT CLUSTER

4. ABL-TRANSFORM MODELING#

```
*****************************************************************************
*                                                                           *
*     MODELING  # OR THE OVERALL SYSTEM DESIGN AND MODELING OF THIS SYSTEM   *
*                                                                           *
*  THIS TABLE PROVIDES AN OVERVIEW IN THE MODELING STAGE OF THE SYSTEM.     *
*  #MODELING  HAS 44 ACTION PROCEDURES AND 64 PREDICATES.  THESE ACTIONS    *
*  ARE GROUPED INTO FUNCTIONS AND SUBFUNCTIONS AND FORM THE HIERARCHICALLY  *
*  INTERRELATED BUILDING BLOCKS OF THE WHOLE SYSTEM. THE PREDICATES ACT AS  *
*  CRITERIONS FOR A PARTICULAR SUBFUNCTION. THEY DEFINE THE CONSTRAINTS     *
*  THE ACTIONS IN THE PARTICULAR SUBFUNTION MUST OBEY.  UNLESS THESE        *
*  CRITERIONS ARE SATISFIED, THAT PARTICULAR SUBFUNCTION IS NOT ADEQUATELY  *
*  DESIGNED TO MEET ITS PURPOSE.                                            *
*                                                                           *
*  ONCE ALL THE ACTIONS ARE DESIGNED, THESE SAME ACTIONS WILL BE OPERATIONALLY *
*  SEQUENCED AND INTERFACED IN THE NEXT TABLE CALLED  #SYSTALL              *
*                                                                           *
*****************************************************************************
```

	17	18	19	20	21	22	23	24	25	26	27	28	(ALTERNATIVES)
													CLUSTERS -
C1	.	.	.	.	.	.	.	.	.	.	.	0	=== ORGANIZE THE BODY OF THE SYSTEM
C2	.	.	.	.	.	.	.	.	.	.	.	0	=== ORGANIZE THE REGULAR RUN OF THE SYSTEM
C3	3	3	3	4	.	.	.	.	.	.	.	0	=== DATA UPDATE ORGANIZATION
C4	.	.	.	.	4	4	4	4	4	4	0	0	=== DATA-BASE QUERRY ORGANIZATION
													PREDICATES -
P1	-	-	-	-	-	-	-	-	-	-	-	-	PRODUCTION CYCLE PERIOD IS DEFINED AS 1-2 DAYS LONG
P2	-	-	-	-	-	-	-	-	-	-	-	-	DATA COLLECTION PERIOD IS DEFINED AS 30-40 MINUTES LONG
P3	-	-	-	-	-	-	-	-	-	-	-	-	INEXPENSIVE TERMINAL IS AVAILABLE
P4	-	-	-	-	-	-	-	-	-	-	-	-	INEXPENSIVE DATA COMMUNICATION IS AVAILABLE
P5	-	-	-	-	-	-	-	-	-	-	-	-	REMOTE TIME-SHARING COMPUTER IS AVAILABLE
P6	-	-	-	-	-	-	-	-	-	-	-	-	ON-LINE SYSTEM ACCESS IS NEEDED
P7	-	-	-	-	-	-	-	-	-	-	-	-	DISK-STORAGE IS AVAILABLE
P8	-	-	-	-	-	-	-	-	-	-	-	-	INTERACTIVE QUERRY AND UPDATE IS NEEDED
P9	-	-	-	-	-	-	-	-	-	-	-	-	SYSTEM USAGE IS STUDIED
P10	-	-	-	-	-	-	-	-	-	-	-	-	SYSTEM OPERATING SPECIALIST IS AVAILABLE IN SCHOOL
P11	-	-	-	-	-	-	-	-	-	-	-	-	SYSTEM OPERATING TASK HAS TO BE SHARED BY REGULAR STAFF
P12	Y	Y	Y	-	-	-	-	-	-	-	-	-	USER SYSTEM ACCESS ORGANIZATION IS DEFINED
P13	-	-	-	-	Y	Y	Y	-	-	-	-	-	CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED
P14	-	-	-	-	-	-	-	-	-	-	-	-	SYSTEM MODULARIZATION IS DEFINED
P15	Y	Y	Y	-	-	-	-	-	-	-	-	-	HIGH DEGREE OF DATA-BASE ACCURACY IS NEEDED
P16	-	-	-	-	-	-	-	-	-	-	-	-	HIGH VOLUME OF CHANGES OCCUR IN THE DATA-BASE
P17	-	-	-	-	-	-	-	-	-	-	-	-	DATA-BASE CHANGES HAVE TO BE IMPLEMENTED IN 1-2 DAYS
P18	-	-	-	-	-	-	-	-	-	-	-	-	INTERACTIVE UPDATING LANGUAGE FOR THE COURSE #MF, IS NEEDED
P19	-	-	-	-	-	-	-	-	-	-	-	-	INTERACTIVE UPDATING LANGUAGE FOR THE STUDENT #MF, IS NEEDED
P20	-	-	-	-	-	-	-	-	-	-	-	-	CLASS MASTER FILE UPDATING CAN BE DONE AUTOMATICALLY
P21	-	-	-	-	Y	Y	Y	N	N	N	-	-	IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING
P22	-	-	-	-	Y	Y	Y	N	N	N	-	-	OUTPUT VOLUME IS SMALL
P23	-	-	-	-	-	-	-	-	-	-	-	-	INTERACTIVE QUERRY LANGUAGE FOR THE COURSE #MF, IS NEEDED
P24	-	-	-	-	-	-	-	-	-	-	-	-	INTERACTIVE QUERRY LANGUAGE FOR THE STUDENT #MF, IS NEEDED
P25	-	-	-	-	-	-	-	-	-	-	-	-	INTERACTIVE QUERRY LANGUAGE FOR THE CLASS #MF, IS NEEDED

CONTINUED ON THE NEXT PAGE

P26	-	-	-	-	-	-	-	-	-	-	-	HIGH VOLUME OF TRUANCY TRANSACTION OCCUR (100-250 A LECTURE)
P27	-	-	-	-	-	-	-	-	-	-	-	DATA GENERATION PERIOD IS 30-60 SECONDS
P28	-	-	-	-	-	-	-	-	-	-	-	ERROR GENERATION POSSIBILITY IN TRANSACTION DATA IS MINIMAL
P29	-	-	-	-	-	-	-	-	-	-	-	INEXPENSIVE OPTICAL MARK READER IS FOUND FOR THE TERMINAL
P30	-	-	-	-	-	-	-	-	-	-	-	REGULAR #ATS OPERATOR IS AVAILABLE FROM STAFF
P31	-	-	-	-	-	-	-	-	-	-	-	DATA TRANSMISSION IS BAD SOMETIMES
P32	-	-	-	-	-	-	-	-	-	-	-	OUTPUT PRINTING ON THE TERMINAL SOMETIMES HAS TO WAIT
P33	-	-	-	-	-	-	-	-	-	-	-	VARIABLE LENGTH OUTPUT IS NEEDED
P34	-	-	-	-	-	-	-	-	-	-	-	DATA GENERATION PROCESS IS ACCEPTED BY THE TEACHERS
P35	-	-	-	-	-	-	-	-	-	-	-	DATA COLLECTION PROCESS IS ACCEPTED BY THE ADMINISTRATION
P36	-	-	-	-	-	-	-	-	-	-	-	DATA COLLECTION ROUTE IS ORGANIZED
P37	-	-	-	-	-	-	-	-	-	-	-	DATA COLLECTION ERRORS ARE MINIMAL
P38	-	-	-	-	-	-	-	-	-	-	-	EXTENSIVE TRANSACTION STORAGE AND ANALYSIS IS REQUIRED
P39	-	-	-	-	-	-	-	-	-	-	-	PROCEDURE IS SIMPLE
P40	-	-	-	-	-	-	-	-	-	-	-	PROCEDURE IS SHORT
P41	-	-	-	-	-	-	-	-	-	-	-	OUTPUT CAN BE SPOOLED
P42	-	-	-	-	-	-	-	-	-	-	-	PRINTING OF SPOOLED OUTPUT IS NOT MORE THAN 1-2 HOURS
P43	-	-	-	-	-	-	-	-	-	-	-	DISTRIBUTION DELAY IS DEFINED AS 2 HOURS
P44	-	-	-	-	-	-	-	-	-	-	-	OUTPUT VOLUME IS ADEQUATE TO DAILY USAGE
P45	-	-	-	-	-	-	-	-	-	-	-	FALSE-NEGATIVE ERROR RATE IS BELOW 10% RATE
P46	-	-	-	-	-	-	-	-	-	-	-	FALSE-POSITIVE ERROR RATE IS BELOW 0.1% RATE
P47	-	-	-	-	-	-	-	-	-	-	-	PROVIDED INFORMATION CAN BE READILY USED IN DECISION MAKING
P48	-	-	-	-	-	-	-	-	-	-	-	INFORMATION VALUE IS MAXIMUM OPERATIONAL VS. OVERALL DELAY
P49	Y	-	-	-	Y	Y	Y	Y	Y	Y	-	-	#VP IN CHARGE OF THE SYSTEM HAS ACCESS
P50	Y	Y	-	-	-	-	-	-	-	-	-	-	REDUNDANT DATA IN THE DATA-BASE ARE AUTOMATICALLY UPDATED
P51	-	Y	-	-	-	Y	-	-	-	-	-	-	RECORDS OFFICE OF THE SCHOOL HAS ACCESS
P52	-	-	Y	-	-	Y	Y	-	-	-	-	-	SECTOR #VP. OF THE SCHOOL HAS ACCESS
P53	-	-	Y	-	-	-	-	-	-	-	-	-	CLASS RELATED DATA CHANGED OR SOME #ABS-CARDS GOT LOST
P54	-	-	Y	-	-	-	-	-	-	-	-	-	DATA-BASE DATA IS UPDATED
P55	-	-	Y	-	-	-	-	-	-	-	-	-	CARD PUNCHING AND INTERPRETING IS AVAILABLE
P56	-	-	-	-	Y	-	-	Y	Y	Y	-	-	OBJECTIVE IS AN EFFICIENT SCHOOL MANAGEMENT
P57	-	-	-	-	-	Y	Y	-	-	-	-	-	TEACHERS HAVE ACCESS
P58	-	-	-	-	-	Y	-	-	-	-	-	-	STUDENTS AND PARENTS HAVE INDIRECT ACCESS
P59	-	-	-	-	-	Y	-	-	-	-	-	-	OBJECTIVE IS AN EFFICIENT STUDENT AND PARENT COUNSELLING
P60	-	-	-	-	-	-	Y	-	-	Y	-	-	OBJECTIVE IS AN EFFICIENT CLASS-MANAGEMENT
P61	-	-	-	-	-	-	-	Y	-	-	-	-	COURSE INFORMATION IS NEEDED
P62	-	-	-	-	-	-	-	-	Y	-	-	-	STUDENT INFORMATION IS NEEDED
P63	-	-	-	-	-	-	-	-	-	Y	-	-	CLASS INFORMATION IS NEEDED
P64	-	-	-	Y	-	-	-	-	-	-	Y	-	ALL SUBFUNCTIONS IN THIS FUNCTION (CLUSTER) ARE MET

CONTINUED ON THE NEXT PAGE

													ACTIONS -
A1	.	.	.	.	.	.	.	.	.	.	.	.	A1> INSTALL THE TERMINAL IN SCHOOL
A2	.	.	.	.	.	.	.	.	.	.	.	.	A2> SET UP THE COURSE MASTER FILE
A3	.	.	.	.	.	.	.	.	.	.	.	.	A3> SET UP THE STUDENT MASTER FILE
A4	.	.	.	.	.	.	.	.	.	.	.	.	A4> SET UP THE CLASS MASTER FILE
A5	.	.	.	.	.	.	.	.	.	.	.	.	A5> SET UP COURSE MASTER FILE UPDATER PROGRAM
A6	.	.	.	.	.	.	.	.	.	.	.	.	A6> SET UP STUDENT MASTER FILE UPDATER PROGRAM
A7	.	.	.	.	.	.	.	.	.	.	.	.	A7> SET UP COURSE MASTER FILE QUERRY PROGRAM
A8	.	.	.	.	.	.	.	.	.	.	.	.	A8> SET UP STUDENT MASTER FILE QUERRY PROGRAM
A9	.	.	.	.	.	.	.	.	.	.	.	.	A9> SET UP CLASS MASTER FILE QUERRY PROGRAM
A10	1	.	.	.	.	.	.	.	.	.	.	.	A10> UPDATE COURSE MASTER FILE INTERACTIVELY
A11	.	1	.	.	.	.	.	.	.	.	.	.	A11> UPDATE STUDENT MASTER FILE INTERACTIVELY
A12	.	.	.	.	.	.	.	.	.	.	.	.	A12> UPDATE CLASS MASTER FILE INTERACTIVELY.
A13	.	.	.	.	.	.	.	.	.	.	.	.	A13> VERIFY COURSE MASTER FILE DATA
A14	.	.	.	.	.	.	.	.	.	.	.	.	A14> VERIFY STUDENT MASTER FILE DATA
A15	.	.	.	.	.	.	.	.	.	.	.	.	A15> VERIFY CLASS MASTER FILE DATA
A16	.	.	.	.	.	.	.	.	.	.	.	.	A16> PRODUCE TEACHERS DATA GENERATION KITS
A17	.	.	.	.	.	.	.	.	.	.	.	.	A17> SET UP DATA COLLECTION ROUTES IN SCHOOL
A18	.	.	.	.	.	.	.	.	.	.	.	.	A18> INSTALL OPTICAL MARK READER ON TERMINAL
A19	.	.	.	.	.	.	.	.	.	.	.	.	A19> VERIFY DATA DISTRIBUTION ROUTES
A20	.	.	.	.	.	.	.	.	.	.	.	.	A20> DETERMINE DATA USAGE PARAMETERS
A21	.	.	.	.	.	.	.	.	.	.	.	.	A21> SET UP DAILY INITIALIZATION INTERACTIVE PROGRAM
A22	.	.	.	.	.	.	.	.	.	.	.	.	A22> SET UP DATA COLLECTION PSEUDOBATCH PROGRAM
A23	.	.	.	.	.	.	.	.	.	.	.	.	A23> SET UP DATS. DATA PROCESSING PSEUDOBATCH PROGRAM
A24	.	.	.	.	.	.	.	.	.	.	.	.	A24> SET UP PROCEDURE TO PRINT SPOOLED OUTPUT
A25	.	.	.	.	.	.	.	.	.	.	.	.	A25> PERFORM DAILY SYSTEM INITIALIZATION
A26	.	.	.	.	.	.	.	.	.	.	.	.	A26> PERFORM TRANSACTION DATA GENERATION
A27	.	.	.	.	.	.	.	.	.	.	.	.	A27> PERFORM TRANSACTION DATA COLLECTION
A28	.	.	.	.	.	.	.	.	.	.	.	.	A28> PERFORM DATA PROCESSING (REPORTING AND STORAGE)
A29	.	.	.	.	.	.	.	.	.	.	.	.	A29> PERFORM DATA DISTRIBUTION
A30	.	.	.	.	.	.	.	.	.	.	.	.	A30> VERIFY OUTPUT USAGE IN DECISION MAKING
A31	.	.	2	.	.	.	.	.	.	.	.	.	A31> PUNCH NEW # ABS-CARDS
A32	.	.	3	.	.	.	.	.	.	.	.	.	A32> UPDATE TEACHERS DATA GENERATION KITS
A33	.	.	.	.	1	.	.	.	.	.	.	.	A33> QUERRY COURSE MASTER FILE INTERACTIVELY
A34	.	.	.	.	.	1	.	.	.	.	.	.	A34> QUERRY STUDENT MASTER FILE INTERACTIVELY
A35	.	.	1	.	.	.	1	.	.	.	.	.	A35> QUERRY CLASS MASTER FILE INTERACTIVELY
A36	.	.	.	.	.	.	.	1	.	.	.	.	A36> WRITE A FILE IN COURSE MASTER FILE QUERRY LANGUAGE
A37	.	.	.	.	.	.	.	2	.	.	.	.	A37> QUERRY COURSE MASTER FILE PSEUDOBATCH MODE
A38	.	.	.	.	.	.	.	3	3	3	.	.	A38> PRINT SPOOLED OUTPUT
A39	.	.	.	.	.	.	.	.	1	.	.	.	A39> WRITE A FILE IN STUDENT MASTER FILE QUERRY LANGUAGE
A40	.	.	.	.	.	.	.	.	2	.	.	.	A40> QUERRY STUDENT MASTER FILE PSEUDOBATCH MODE
A41	.	.	.	.	.	.	.	.	.	1	.	.	A41> WRITE A FILE IN CLASS MASTER FILE QUERRY LANGUAGE
A42	.	.	.	.	.	.	.	.	.	2	.	.	A42> QUERRY CLASS MASTER FILE PSEUDOBATCH MODE
A43	.	.	.	.	.	.	.	.	.	.	.	.	A43> PRODUCE # ABS-CARDS FOR THE CLASS
A44	.	.	.	1	.	.	.	.	.	.	1	.	A44> NO ACTION- NEXT CLUSTER

3. ABL-ROUTINE SYSTALL;

```
*****************************************************************************
*                                                                           *
*          SYSTALL  # (OR THE OVERALL DESCRIPTION OF THE SYSTEM)            *
*          ------------------------------------------------------           *
*                                                                           *
*  THIS SEQUENCE OF PROCEDURES PROVIDES AN OVERVIEW OF THE ENTIRE           *
*  ATTENDANCE MONITORING SYSTEM  (#  ATS.  ).  IT SPECIFIES HOW TO          *
*  SET UP, RUN AND USE THIS SYSTEM.  IT SHOWS THE LOGICAL INTER-            *
*  RELATION AND OPERATIONAL SEQUENCE OF THE SUBSYSTEMS AMONG EACH           *
*  OTHERS.  FIRST WE SET UP THE BASIC SYSTEM; THAT IS THE DATA-BASE (#DB)   *
*  WITH ITS QUERRY (#QUS.) AND UPDATE (#UPS.) UTILITY SUBSYSTEMS.           *
*  SECOND WE GRAFT TO IT THE ATTENDANCE MONITORING SUBSYSTEM (#ATS.).       *
*  FROM THEN ON WE CONCURRENTLY MAINTAIN THE EXACTNESS OF THE DATA-BASE     *
*  WHILE ROUTINELY RUNNING THE ATTENDANCE MONITORING SUBSYSTEM              *
*  IN ORDER TO EXTRACT THE RELATED INFORMATION FROM THE DATA-BASE.          *
*                                                                           *
*                    SYSTALL ACTION(1..44)                                  *
*  SYSTALL HAS 44 ACTIONS.  EACH OF THESE IS A COMPLEX ACTION               *
*  PROCEDURE HAVING THE POSSIBILITY OF BEING DECOMPOSED INTO                *
*  SUB-PROCEDURES.  IN ORDER TO BE ABLE TO ESTABLISH CLEARLY                *
*  THE HIERARCHIAL RELATIONSHIP BETWEEN THE SUB-ACTION AND                  *
*  THE ACTIONS OF SYSTALL THE FOLLOWING NOTATION WILL BE USED:              *
*                                                                           *
*          SYSTALL ACTION(N) = ACTION(N) = AN> XXXXXXXXX                    *
*                                                                           *
*          SYSTALL SUB-ACTION(M) OF ACTION(N) =                             *
*                = SYSTALL ACTION(N-M) = AN-M> XXXXXXXXX                    *
*                                                                           *
*          SYSTALL SUB-SUBACTION(L) OF SUBACTION(M) OF ACTION(N) =          *
*                = SYSTALL ACTION(N-M-L) = AN-M-L> XXXXXXXXX                *
*                                                                           *
*    #  FOR ALL CONTRACTIONS PLEASE REFER TO THE TABLE ON FIGURE 27.         *
*                                                                           *
*          THE PRODUCTION CYCLE OF # ATS.  IS DESCRIBED IN DETAIL           *
*          IN THE FOLLOWING ACTIONS:                                        *
*                                                                           *
*        SYSTALL A(25) = DAILY SYSTEM INITIALIZATION                        *
*        SYSTALL A(26) = DAILY TRANSACTION DATA GENERATION                  *
*        SYSTALL A(27) = DAILY TRANSACTION DATA COLLECTION                  *
*        SYSTALL A(28) = DAILY TRANSACTION DATA PROCESSING                  *
*        SYSTALL A(29) = DAILY DATA DITRIBUTION                             *
*        SYSTALL A(30) = DAILY DATA USAGE VERIFICATION                      *
*                                                                           *
*                                                                           *
*           AN EXAMPLE OF THE COMPLEXITY OF SOME ACTIONS                    *
*           IS ILLUSTRATED IN SYSTALL A(11); WHOSE LOGICAL                  *
*           DESCRIPTION IS 3 LEVELS DEEP.                                   *
*                                                                           *
*        SYSTALL A(11) = UPDATE THE STUDENT MASTER FILE                     *
*                                                                           *
*****************************************************************************
```

CONTINUED ON THE NEXT PAGE

```
1.1 SET UP THE BASIC SYSTEM ( # UTS. AND # DB. )          ===#          < 1, 1, 1, 2 >
    1.1= INSTALL TERMINAL IN SCHOOL                       ==    --THEN-GO-TO-CLUSTER-> 1
    1.2= SET UP EMPTY DATA-BASE                           ==    --THEN-GO-TO-CLUSTER-> 1
    1.3= SET UP ALL #QUS. AND #UPS. INTERACTIVE PROGRAMS. ==    --THEN-GO-TO-CLUSTER-> 1
    1.4= UPDATE ALLOCATED FIELDS IN DATA-BASE             ==    --THEN-GO-TO-CLUSTER-> 2
2.2 INITIAL DATA-BASE VERIFICATION. ( # QUS. AND # UPS. ) ===#          < 2, 3, 3 >
    2.1= VERIFY INFORMATION IN DATA-BASE                  ==    --THEN-GO-TO-CLUSTER-> 2
    2.2= UPDATE INCORRECT INFORMATION IN DATA-BASE        ==    --THEN-GO-TO-CLUSTER-> 3
    2.3= NO UPDATE IS NECESSARY                           ==    --THEN-GO-TO-CLUSTER-> 3
3.3 SET UP THE ATTENDANCE MONITORING SUBSYSTEM ( # UTS. ) ===#          < 3, 4 >
    3.1= GIVE OUT ACCESSORIES FOR ATTEND.MONIT.SUBSYSTEM. ==    --THEN-GO-TO-CLUSTER-> 3
    3.2= SET UP #ATS.INTERACT.PROGS.FOR DAILY PRODUCTION CYCLE ==  --THEN-GO-TO-CLUSTER-> 4
4.4 DO REGULAR RUN OF THE ATTENDANCE MONITORING SUBSYSTEM (# ATS.)===#  < 4, 4, 4, 4, 4, 4, 5 >
    4.1= #ATS. DAILY DATA USAGE PARAMETER DEFINITION      ==    --THEN-GO-TO-CLUSTER-> 4
    4.2= #ATS. DAILY TRANSACTION DATA GENERATION          ==    --THEN-GO-TO-CLUSTER-> 4
    4.3= #ATS. DAILY TRANSACTION DATA COLLECTION          ==    --THEN-GO-TO-CLUSTER-> 4
    4.4= #ATS. DAILY TRANSACTION DATA PROCESSING          ==    --THEN-GO-TO-CLUSTER-> 4
    4.5= #ATS. DAILY DATA DISTRIBUTION                    ==    --THEN-GO-TO-CLUSTER-> 4
    4.6= #ATS.,DAILY OUTPUT USAGE                         ==    --THEN-GO-TO-CLUSTER-> 4
    4.7= NO ACTION,NEXT CLUSTER                           ==    --THEN-GO-TO-CLUSTER-> 5
5.5 MAINTAIN THE EXACTNESS OF THE SYSTEMS DATA ( # UPS. ) ===#          < 4, 4, 6 >
    5.1= UPDATE NON ATTENDANCE RELATED DATA               ==    --THEN-GO-TO-CLUSTER-> 4
    5.2= UPDATE ATTENDANCE RELATED DATA                   ==    --THEN-GO-TO-CLUSTER-> 4
    5.3= NO ACTION, NEXT CLUSTER                          ==    --THEN-GO-TO-CLUSTER-> 6
6.6 INFORMATION EXTRACTING AND REPORTING ( # QUS. )       ===#          < 4, 4, 4, 4, 4, 4 >
    6.1= QUERRY COURSE MASTER FILE INTERACTIVELY          ==    --THEN-GO-TO-CLUSTER-> 4
    6.2= QUERRY STUDENT MASTER FILE INTERACTIVELY         ==    --THEN-GO-TO-CLUSTER-> 4
    6.3= QUERRY CLASS MASTER FILE INTERACTIVELY           ==    --THEN-GO-TO-CLUSTER-> 4
    6.4= QUERRY COURSE MASTER FILE PSEUDOBATCH            ==    --THEN-GO-TO-CLUSTER-> 4
    6.5= QUERRY STUDENT MASTER FILE PSEUDOBATCH           ==    --THEN-GO-TO-CLUSTER-> 4
    6.6= QUERRY CLASS MASTER FILE PSEUDOBATCH             ==    --THEN-GO-TO-CLUSTER-> 4
```

3. ABL-ROUTINE SYSTALL;

```
*****************************************************************
*                                                               *
*        SYSTALL # (OR THE OVERALL DESCRIPTION OF THE SYSTEM)   *
*        ----------------------------------------------------   *
*                                                               *
* THIS SEQUENCE OF PROCEDURES PROVIDES AN OVERVIEW OF THE ENTIRE*
* ATTENDANCE MONITORING SYSTEM (# ATS. ). IT SPECIFIES HOW TO   *
* SET UP, RUN AND USE THIS SYSTEM. IT SHOWS THE LOGICAL INTER-  *
* RELATION AND OPERATIONAL SEQUENCE OF THE SUBSYSTEMS AMONG EACH*
* OTHERS. FIRST WE SET UP THE BASIC SYSTEM, THAT IS THE DATA-BASE (#DB)
* WITH ITS QUERRY (#QUS.) AND UPDATE (#UPS.) UTILITY SUBSYSTEMS.*
* SECOND WE GRAFT TO IT THE ATTENDANCE MONITORING SUBSYSTEM (#ATS.).
* FROM THEN ON WE CONCURRENTLY MAINTAIN THE EXACTNESS OF THE DATA-BASE
* WHILE ROUTINELY RUNNING THE ATTENDANCE MONITORING SUBSYSTEM   *
* IN ORDER TO EXTRACT THE RELATED INFORMATION FROM THE DATA-BASE.
*                                                               *
*                  SYSTALL ACTION(1..44)                        *
* SYSTALL HAS 44 ACTIONS. EACH OF THESE IS A COMPLEX ACTION     *
* PROCEDURE HAVING THE POSSIBILITY OF BEING DECOMPOSED INTO     *
* SUB-PROCEDURES. IN ORDER TO BE ABLE TO ESTABLISH CLEARLY      *
* THE HIERARCHIAL RELATIONSHIP BETWEEN THE SUB-ACTION AND       *
* THE ACTIONS OF SYSTALL THE FOLLOWING NOTATION WILL BE USED:   *
*                                                               *
*        SYSTALL ACTION(N) = ACTION(N) = AN> XXXXXXXXX          *
*                                                               *
*        SYSTALL SUB-ACTION(M) OF ACTION(N) =                   *
*              = SYSTALL ACTION(N-M) = AN-M> XXXXXXXXX          *
*                                                               *
*        SYSTALL SUB-SUBACTION(L) OF SUBACTION(M) OF ACTION(N) =*
*              = SYSTALL ACTION(N-M-L) = AN-M-L> XXXXXXXXX      *
*                                                               *
*   # FOR ALL CONTRACTIONS PLEASE REFER TO THE TABLE ON FIGURE 27.
*                                                               *
*        THE PRODUCTION CYCLE OF # ATS. IS DESCRIBED IN DETAIL  *
*        IN THE FOLLOWING ACTIONS:                              *
*                                                               *
*       SYSTALL A(25) = DAILY SYSTEM INITIALIZATION             *
*       SYSTALL A(26) = DAILY TRANSACTION DATA GENERATION       *
*       SYSTALL A(27) = DAILY TRANSACTION DATA COLLECTION       *
*       SYSTALL A(28) = DAILY TRANSACTION DATA PROCESSING       *
*       SYSTALL A(29) = DAILY DATA DITRIBUTION                  *
*       SYSTALL A(30) = DAILY DATA USAGE VERIFICATION           *
*                                                               *
*          AN EXAMPLE OF THE COMPLEXITY OF SOME ACTIONS         *
*          IS ILLUSTRATED IN SYSTALL A(11), WHOSE LOGICAL       *
*          DESCRIPTION IS 3 LEVELS DEEP.                        *
*                                                               *
*       SYSTALL A(11) = UPDATE THE STUDENT MASTER FILE          *
*                                                               *
*****************************************************************
```

1.1 SET UP THE BASIC SYSTEM (# UTS. AND # DB.) ===# < 1, 1, 1, 2 >

1.1= INSTALL TERMINAL IN SCHOOL == —THEN-GO-TO-CLUSTER-> 1

1.1 IF NOT TERMINAL IS SET UP IN SCHOOL

CONTINUED ON THE NEXT PAGE

```
     1.2= SET UP EMPTY DATA-BASE                              ==    --THEN-GO-TO-CLUSTER-> 1
          1.2 IF  TERMINAL IS SET UP IN SCHOOL  AND IF
                  NOT  ALL DATA-BASE FILES ARE SET UP ON-LINE

     1.3= SET UP ALL #QUS. AND #UPS. INTERACTIVE PROGRAMS.    ==    --THEN-GO-TO-CLUSTER-> 1
          1.3 IF  ALL DATA-BASE FILES ARE SET UP ON-LINE  AND IF
                  NOT  QUERRY AND UPDATE SUBSYSTEMS ARE SET UP

     1.4= UPDATE ALLOCATED FIELDS IN DATA-BASE                ==    --THEN-GO-TO-CLUSTER-> 2
          1.4 IF  ALL DATA-BASE FILES ARE SET UP ON-LINE  AND IF
                  QUERRY AND UPDATE SUBSYSTEMS ARE SET UP  AND IF
                  NOT  ALLOCATED FIELDS IN THE DATA-BASE ARE FILLED
2.2 INITIAL DATA-BASE VERIFICATION. ( # QUS. AND # UPS. )    ===#                    < 2, 3, 3 >

     2.1= VERIFY INFORMATION IN DATA-BASE                     ==    --THEN-GO-TO-CLUSTER-> 2
          2.1 IF  ALL DATA-BASE FILES ARE SET UP ON-LINE  AND IF
                  NOT  DATA-BASE CONTENT IS  VERIFIED

     2.2= UPDATE INCORRECT INFORMATION IN DATA-BASE           ==    --THEN-GO-TO-CLUSTER-> 3
          2.2 IF  DATA-BASE CONTENT IS  VERIFIED  AND IF
                  ERROR IN THE DATA-BASE

     2.3= NO UPDATE IS NECESSARY                              ==    --THEN-GO-TO-CLUSTER-> 3
          2.3 IF  DATA-BASE CONTENT IS  VERIFIED  AND IF
                  NOT  ERROR IN THE DATA-BASE
3.3 SET UP THE ATTENDANCE MONITORING SUBSYSTEM ( # UTS. )    ===#                  < 3, 4 >

     3.1= GIVE OUT ACCESSORIES FOR ATTEND.MONIT.SUBSYSTEM.    ==    --THEN-GO-TO-CLUSTER-> 3
          3.1 IF  NOT  ATTENDANCE MONITORING SUBSYSTEM IS SET UP  AND IF
                  DATA-BASE CONTAINS CORRECT NON TRUANCY INFORM.  AND IF
                  NOT  TRANSACTION DATA GENERATION KITS ARE READY

     3.2= SET UP #ATS.INTERACT.PROGS.FOR DAILY PRODUCTION CYCLE  ==    --THEN-GO-TO-CLUSTER-> 4
          3.2 IF  NOT  ATTENDANCE MONITORING SUBSYSTEM IS SET UP  AND IF
                  TRANSACTION DATA GENERATION KITS ARE READY
4.4 DO REGULAR USE OF THE ATTENDANCE MONITORING SUBSYSTEM (# ATS.)===#          < 4, 4, 4, 4, 4, 4, 5 >

     4.1= #ATS. DAILY DATA USAGE PARAMETER DEFINITION         ==    --THEN-GO-TO-CLUSTER-> 4
          4.1 IF  TASK IS ATTENDANCE MONITORING  AND IF
                  ATTENDANCE MONITORING SUBSYSTEM IS SET UP  AND IF
                  NOT  PRODUCTION CYCLE IS STARTED  AND IF
                  USAGE PARAMETERS ARE DETERMINED

     4.2= #ATS. DAILY TRANSACTION DATA GENERATION             ==    --THEN-GO-TO-CLUSTER-> 4
          4.2 IF  TASK IS ATTENDANCE MONITORING  AND IF
                  PRODUCTION CYCLE IS STARTED  AND IF
                  PRODUCTION DAY IS INITIALIZED TO THE COMPUTER

     4.3= #ATS. DAILY TRANSACTION DATA COLLECTION             ==    --THEN-GO-TO-CLUSTER-> 4
          4.3 IF  TASK IS ATTENDANCE MONITORING  AND IF
                  PRODUCTION CYCLE IS STARTED  AND IF
                  TRANSACTION DATA IS GENERATED
```

CONTINUED ON THE NEXT PAGE

4.4= OATS. DAILY TRANSACTION DATA PROCESSING == —THEN-GO-TO-CLUSTER-> 4
4.4 IF TASK IS ATTENDANCE MONITORING AND IF
PRODUCTION CYCLE IS STARTED AND IF
TRANSACTION DATA IS COLLECTED

4.5= OATS. DAILY DATA DISTRIBUTION == —THEN-GO-TO-CLUSTER-> 4
4.5 IF TASK IS ATTENDANCE MONITORING AND IF
PRODUCTION CYCLE IS STARTED AND IF
DATA IS PROCESSED

4.6= OATS. DAILY OUTPUT USAGE == —THEN-GO-TO-CLUSTER-> 4
4.6 IF TASK IS ATTENDANCE MONITORING AND IF
PRODUCTION CYCLE IS STARTED AND IF
OUTPUT IS DISTRIBUTED AND IF
NOT USAGE PARAMETERS ARE DETERMINED

4.7= NO ACTION,NEXT CLUSTER == —THEN-GO-TO-CLUSTER-> 5
4.7 IF NOT TASK IS ATTENDANCE MONITORING

5.5 MAINTAIN THE EXACTNESS OF THE SYSTEMS DATA (# UPS.) ==# < 4, 4, 6 >

5.1= UPDATE NON ATTENDANCE RELATED DATA == —THEN-GO-TO-CLUSTER-> 4
5.1 IF TASK IS UPDATING THE DATA-BASE AND IF
CHANGE INVOLVES NON OATS. RELATED DATA

5.2= UPDATE ATTENDANCE RELATED DATA == —THEN-GO-TO-CLUSTER-> 4
5.2 IF TASK IS UPDATING THE DATA-BASE AND IF
CHANGE INVOLVES OATS. RELATED DATA

5.3= NO ACTION, NEXT CLUSTER == —THEN-GO-TO-CLUSTER-> 6
5.3 IF NOT TASK IS UPDATING THE DATA-BASE

6.6 INFORMATION EXTRACTING AND REPORTING (# QUS.) ==# < 4, 4, 4, 4, 4, 4 >

6.1= QUERRY COURSE MASTER FILE INTERACTIVELY == —THEN-GO-TO-CLUSTER-> 4
6.1 IF TASK IS QUERRY THE DATA-BASE AND IF
COURSE INFORMATION IS NEEDED AND IF
SMALL AMOUNT OF OUTPUT IS NEEDED

6.2= QUERRY STUDENT MASTER FILE INTERACTIVELY == —THEN-GO-TO-CLUSTER-> 4
6.2 IF TASK IS QUERRY THE DATA-BASE AND IF
STUDENT INFORMATION IS NEEDED AND IF
SMALL AMOUNT OF OUTPUT IS NEEDED

6.3= QUERRY CLASS MASTER FILE INTERACTIVELY == —THEN-GO-TO-CLUSTER-> 4
6.3 IF TASK IS QUERRY THE DATA-BASE AND IF
CLASS INFORMATION IS NEEDED AND IF
SMALL AMOUNT OF OUTPUT IS NEEDED

6.4= QUERRY COURSE MASTER FILE PSEUDOBATCH == —THEN-GO-TO-CLUSTER-> 4
6.4 IF TASK IS QUERRY THE DATA-BASE AND IF
COURSE INFORMATION IS NEEDED AND IF
LARGE AMOUNT OF OUTPUT IS NEEDED

CONTINUED ON THE NEXT PAGE

6.5= QUERRY STUDENT MASTER FILE PSEUDOBATCH == --THEN-GO-TO-CLUSTER-> 4
6.5 IF TASK IS QUERRY THE DATA-BASE AND IF
STUDENT INFORMATION IS NEEDED AND IF
LARGE AMOUNT OF OUTPUT IS NEEDED

6.6= QUERRY CLASS MASTER FILE PSEUDOBATCH == --THEN-GO-TO-CLUSTER-> 4
6.6 IF TASK IS QUERRY THE DATA-BASE AND IF
CLASS INFORMATION IS NEEDED AND IF
LARGE AMOUNT OF OUTPUT IS NEEDED

3. ABL-ROUTINE SYSTALL;

```
*****************************************************************
*                                                               *
*        SYSTALL # (OR THE OVERALL DESCRIPTION OF THE SYSTEM)   *
*                                                               *
*                                                               *
* THIS SEQUENCE OF PROCEDURES PROVIDES AN OVERVIEW OF THE ENTIRE*
* ATTENDANCE MONITORING SYSTEM  (# ATS. ).  IT SPECIFIES HOW TO *
* SET UP, RUN AND USE THIS SYSTEM.  IT SHOWS THE LOGICAL INTER- *
* RELATION AND OPERATIONAL SEQUENCE OF THE SUBSYSTEMS AMONG EACH*
* OTHERS.  FIRST WE SET UP THE BASIC SYSTEM, THAT IS THE DATA-BASE (#DB)
* WITH ITS QUERRY (#QUS.) AND UPDATE (#UPS.) UTILITY SUBSYSTEMS.*
* SECOND WE GRAFT TO IT THE ATTENDANCE MONITORING SUBSYSTEM (#ATS.).
* FROM THEN ON WE CONCURRENTLY MAINTAIN THE EXACTNESS OF THE DATA-BASE
* WHILE ROUTINELY RUNNING THE ATTENDANCE MONITORING SUBSYSTEM   *
* IN ORDER TO EXTRACT THE RELATED INFORMATION FROM THE DATA-BASE.*
*                                                               *
*                   SYSTALL ACTION(1..44)                       *
* SYSTALL HAS 44 ACTIONS.  EACH OF THESE IS A COMPLEX ACTION    *
* PROCEDURE HAVING THE POSSIBILITY OF BEING DECOMPOSED INTO     *
* SUB-PROCEDURES.  IN ORDER TO BE ABLE TO ESTABLISH CLEARLY     *
* THE HIERARCHIAL RELATIONSHIP BETWEEN THE SUB-ACTION AND       *
* THE ACTIONS OF SYSTALL THE FOLLOWING NOTATION WILL BE USED:   *
*                                                               *
*         SYSTALL ACTION(N) = ACTION(N) = AN> XXXXXXXXXX        *
*                                                               *
*         SYSTALL SUB-ACTION(M) OF ACTION(N) =                  *
*               = SYSTALL ACTION(N-M) = AN-M> XXXXXXXXXX        *
*                                                               *
*         SYSTALL SUB-SUBACTION(L) OF SUBACTION(M) OF ACTION(N) =
*               = SYSTALL ACTION(N-M-L) = AN-M-L> XXXXXXXXXX    *
*                                                               *
*    # FOR ALL CONTRACTIONS PLEASE REFER TO THE TABLE ON FIGURE 27.
*                                                               *
*         THE PRODUCTION CYCLE OF # ATS.  IS DESCRIBED IN DETAIL*
*         IN THE FOLLOWING ACTIONS:                             *
*                                                               *
*        SYSTALL A(25) = DAILY SYSTEM INITIALIZATION            *
*        SYSTALL A(26) = DAILY TRANSACTION DATA GENERATION      *
*        SYSTALL A(27) = DAILY TRANSACTION DATA COLLECTION      *
*        SYSTALL A(28) = DAILY TRANSACTION DATA PROCESSING      *
*        SYSTALL A(29) = DAILY DATA DITRIBUTION                 *
*        SYSTALL A(30) = DAILY DATA USAGE VERIFICATION          *
*                                                               *
*                                                               *
*           AN EXAMPLE OF THE COMPLEXITY OF SOME ACTIONS        *
*           IS ILLUSTRATED IN SYSTALL A(11), WHOSE LOGICAL      *
*           DESCRIPTION IS 3 LEVELS DEEP.                       *
*                                                               *
*        SYSTALL A(11) = UPDATE THE STUDENT MASTER FILE         *
*                                                               *
*****************************************************************
```

1.1 SET UP THE BASIC SYSTEM (# UTS. AND # DB.) ===# < 1, 1, 1, 2 >

1.1= INSTALL TERMINAL IN SCHOOL == —THEN-GO-TO-CLUSTER-> 1

1.1 IF NOT TERMINAL IS SET UP IN SCHOOL

A1> INSTALL THE TERMINAL IN SCHOOL

CONTINUED ON THE NEXT PAGE

1.2= SET UP EMPTY DATA-BASE == --THEN-GO-TO-CLUSTER-> 1
1.2 IF TERMINAL IS SET UP IN SCHOOL AND IF
NOT ALL DATA-BASE FILES ARE SET UP ON-LINE
A2> SET UP THE COURSE MASTER FILE
A3> SET UP THE STUDENT MASTER FILE
A4> SET UP THE CLASS MASTER FILE

1.3= SET UP ALL #QUS. AND #UPS. INTERACTIVE PROGRAMS. == --THEN-GO-TO-CLUSTER-> 1
1.3 IF ALL DATA-BASE FILES ARE SET UP ON-LINE AND IF
NOT QUERRY AND UPDATE SUBSYSTEMS ARE SET UP
A5> SET UP COURSE MASTER FILE UPDATER PROGRAM
A6> SET UP STUDENT MASTER FILE UPDATER PROGRAM
A7> SET UP COURSE MASTER FILE QUERRY PROGRAM
A8> SET UP STUDENT MASTER FILE QUERRY PROGRAM
A9> SET UP CLASS MASTER FILE QUERRY PROGRAM

1.4= UPDATE ALLOCATED FIELDS IN DATA-BASE == --THEN-GO-TO-CLUSTER-> 2
1.4 IF ALL DATA-BASE FILES ARE SET UP ON-LINE AND IF
QUERRY AND UPDATE SUBSYSTEMS ARE SET UP AND IF
NOT ALLOCATED FIELDS IN THE DATA-BASE ARE FILLED
A10> UPDATE COURSE MASTER FILE INTERACTIVELY
A11> UPDATE STUDENT MASTER FILE INTERACTIVELY
A12> UPDATE CLASS MASTER FILE INTERACTIVELY

2.2 INITIAL DATA-BASE VERIFICATION. (# QUS. AND # UPS.) ==* < 2, 3, 3 >

2.1= VERIFY INFORMATION IN DATA-BASE == --THEN-GO-TO-CLUSTER-> 2
2.1 IF ALL DATA-BASE FILES ARE SET UP ON-LINE AND IF
NOT DATA-BASE CONTENT IS VERIFIED
A37> QUERRY COURSE MASTER FILE PSEUDOBATCH MODE
A13> VERIFY COURSE MASTER FILE DATA
A40> QUERRY STUDENT MASTER FILE PSEUDOBATCH MODE
A14> VERIFY STUDENT MASTER FILE DATA
A42> QUERRY CLASS MASTER FILE PSEUDOBATCH MODE
A15> VERIFY CLASS MASTER FILE DATA

2.2= UPDATE INCORRECT INFORMATION IN DATA-BASE == --THEN-GO-TO-CLUSTER-> 3
2.2 IF DATA-BASE CONTENT IS VERIFIED AND IF
ERROR IN THE DATA-BASE
A10> UPDATE COURSE MASTER FILE INTERACTIVELY
A11> UPDATE STUDENT MASTER FILE INTERACTIVELY
A12> UPDATE CLASS MASTER FILE INTERACTIVELY

2.3= NO UPDATE IS NECESSARY == --THEN-GO-TO-CLUSTER-> 3
2.3 IF DATA-BASE CONTENT IS VERIFIED AND IF
NOT ERROR IN THE DATA-BASE
A44> NO ACTION- NEXT CLUSTER

CONTINUED ON THE NEXT PAGE

3.3 SET UP THE ATTENDANCE MONITORING SUBSYSTEM (4 UTS.) ===$ < 3, 4 >

3.1= GIVE OUT ACCESSORIES FOR ATTEND.MONIT.SUBSYSTEM. == --THEN-GO-TO-CLUSTER-> 3
3.1 IF NOT ATTENDANCE MONITORING SUBSYSTEM IS SET UP AND IF
DATA-BASE CONTAINS CORRECT NON TRUANCY INFORM. AND IF
NOT TRANSACTION DATA GENERATION KITS ARE READY
A42> QUERY CLASS MASTER FILE PSEUDOBATCH MODE
A43> PRODUCE # ABS-CARDS FOR THE CLASS
A16> PRODUCE TEACHERS DATA GENERATION KITS
A17> SET UP DATA COLLECTION ROUTES IN SCHOOL
A18> INSTALL OPTICAL MARK READER ON TERMINAL
A19> VERIFY DATA DISTRIBUTION ROUTES
A20> DETERMINE DATA USAGE PARAMETERS

3.2= SET UP #ATS.INTERACT.PROGS.FOR DAILY PRODUCTION CYCLE == --THEN-GO-TO-CLUSTER-> 4
3.2 IF NOT ATTENDANCE MONITORING SUBSYSTEM IS SET UP AND IF
TRANSACTION DATA GENERATION KITS ARE READY
A21> SET UP DAILY INITIALIZATION INTERACTIVE PROGRAM
A22> SET UP DATA COLLECTION PSEUDOBATCH PROGRAM
A23> SET UP #ATS. DATA PROCESSING PSEUDOBATCH PROGRAM
A24> SET UP PROCEDURE TO PRINT SPOOLED OUTPUT

4.4 DO REGULAR RUN OF THE ATTENDANCE MONITORING SUBSYSTEM (# ATS.)===$ < 4, 4, 4, 4, 4, 4, 5 >

4.1= #ATS. DAILY DATA USAGE PARAMETER DEFINITION == --THEN-GO-TO-CLUSTER-> 4
4.1 IF TASK IS ATTENDANCE MONITORING AND IF
ATTENDANCE MONITORING SUBSYSTEM IS SET UP AND IF
NOT PRODUCTION CYCLE IS STARTED AND IF
USAGE PARAMETERS ARE DETERMINED
A25> PERFORM DAILY SYSTEM INITIALIZATION

4.2= #ATS. DAILY TRANSACTION DATA GENERATION == --THEN-GO-TO-CLUSTER-> 4
4.2 IF TASK IS ATTENDANCE MONITORING AND IF
PRODUCTION CYCLE IS STARTED AND IF
PRODUCTION DAY IS INITIALIZED TO THE COMPUTER
A26> PERFORM TRANSACTION DATA GENERATION

4.3= #ATS. DAILY TRANSACTION DATA COLLECTION == --THEN-GO-TO-CLUSTER-> 4
4.3 IF TASK IS ATTENDANCE MONITORING AND IF
PRODUCTION CYCLE IS STARTED AND IF
TRANSACTION DATA IS GENERATED
A27> PERFORM TRANSACTION DATA COLLECTION

4.4= #ATS. DAILY TRANSACTION DATA PROCESSING == --THEN-GO-TO-CLUSTER-> 4
4.4 IF TASK IS ATTENDANCE MONITORING AND IF
PRODUCTION CYCLE IS STARTED AND IF
TRANSACTION DATA IS COLLECTED
A28> PERFORM DATA PROCESSING (REPORTING AND STORAGE)
A38> PRINT SPOOLED OUTPUT

CONTINUED ON THE NEXT PAGE

4.5= DATS. DAILY DATA DISTRIBUTION == --THEN-GO-TO-CLUSTER-> 4
4.5 IF TASK IS ATTENDANCE MONITORING AND IF
PRODUCTION CYCLE IS STARTED AND IF
DATA IS PROCESSED
A29> PERFORM DATA DISTRIBUTION

4.6= DATS. DAILY OUTPUT USAGE == --THEN-GO-TO-CLUSTER-> 4
4.6 IF TASK IS ATTENDANCE MONITORING AND IF
PRODUCTION CYCLE IS STARTED AND IF
OUTPUT IS DISTRIBUTED AND IF
NOT USAGE PARAMETERS ARE DETERMINED
A30> VERIFY OUTPUT USAGE IN DECISION MAKING
A20> DETERMINE DATA USAGE PARAMETERS

4.7= NO ACTION;NEXT CLUSTER == --THEN-GO-TO-CLUSTER-> 5
4.7 IF NOT TASK IS ATTENDANCE MONITORING
A44> NO ACTION- NEXT CLUSTER

5.5 MAINTAIN THE EXACTNESS OF THE SYSTEMS DATA (4 UPS.) ===4 < 4, 4, 6 >

5.1= UPDATE NON ATTENDANCE RELATED DATA == --THEN-GO-TO-CLUSTER-> 4
5.1 IF TASK IS UPDATING THE DATA-BASE AND IF
CHANGE INVOLVES NON DATS. RELATED DATA
A10> UPDATE COURSE MASTER FILE INTERACTIVELY
A11> UPDATE STUDENT MASTER FILE INTERACTIVELY
A12> UPDATE CLASS MASTER FILE INTERACTIVELY

5.2= UPDATE ATTENDANCE RELATED DATA == --THEN-GO-TO-CLUSTER-> 4
5.2 IF TASK IS UPDATING THE DATA-BASE AND IF
CHANGE INVOLVES DATS. RELATED DATA
A10> UPDATE COURSE MASTER FILE INTERACTIVELY
A11> UPDATE STUDENT MASTER FILE INTERACTIVELY
A12> UPDATE CLASS MASTER FILE INTERACTIVELY
A33> QUERY COURSE MASTER FILE INTERACTIVELY
A31> PUNCH NEW 4 ARG-CARDS
A32> UPDATE TEACHERS DATA GENERATION KITS

5.3= NO ACTION; NEXT CLUSTER == --THEN-GO-TO-CLUSTER-> 6
5.3 IF NOT TASK IS UPDATING THE DATA-BASE
A44> NO ACTION- NEXT CLUSTER

CONTINUED ON THE NEXT PAGE

6.6 INFORMATION EXTRACTING AND REPORTING (# QUS.) ===* < 4, 4, 4, 4, 4, 4 >

6.1= QUERRY COURSE MASTER FILE INTERACTIVELY == --THEN-GO-TO-CLUSTER-> 4
6.1 IF TASK IS QUERRY THE DATA-BASE AND IF
COURSE INFORMATION IS NEEDED AND IF
SMALL AMOUNT OF OUTPUT IS NEEDED
A33> QUERRY COURSE MASTER FILE INTERACTIVELY

6.2= QUERRY STUDENT MASTER FILE INTERACTIVELY == --THEN-GO-TO-CLUSTER-> 4
6.2 IF TASK IS QUERRY THE DATA-BASE AND IF
STUDENT INFORMATION IS NEEDED AND IF
SMALL AMOUNT OF OUTPUT IS NEEDED
A34> QUERRY STUDENT MASTER FILE INTERACTIVELY

6.3= QUERRY CLASS MASTER FILE INTERACTIVELY == --THEN-GO-TO-CLUSTER-> 4
6.3 IF TASK IS QUERRY THE DATA-BASE AND IF
CLASS INFORMATION IS NEEDED AND IF
SMALL AMOUNT OF OUTPUT IS NEEDED
A35> QUERRY CLASS MASTER FILE INTERACTIVELY

6.4= QUERRY COURSE MASTER FILE PSEUDOBATCH == --THEN-GO-TO-CLUSTER-> 4
6.4 IF TASK IS QUERRY THE DATA-BASE AND IF
COURSE INFORMATION IS NEEDED AND IF
LARGE AMOUNT OF OUTPUT IS NEEDED
A36> WRITE A FILE IN COURSE MASTER FILE QUERRY LANGUAGE
A37> QUERRY COURSE MASTER FILE PSEUDOBATCH MODE
A38> PRINT SPOOLED OUTPUT

6.5= QUERRY STUDENT MASTER FILE PSEUDOBATCH == --THEN-GO-TO-CLUSTER-> 4
6.5 IF TASK IS QUERRY THE DATA-BASE AND IF
STUDENT INFORMATION IS NEEDED AND IF
LARGE AMOUNT OF OUTPUT IS NEEDED
A39> WRITE A FILE IN STUDENT MASTER FILE QUERRY LANGUAGE
A40> QUERRY STUDENT MASTER FILE PSEUDOBATCH MODE
A38> PRINT SPOOLED OUTPUT

6.6= QUERRY CLASS MASTER FILE PSEUDOBATCH == --THEN-GO-TO-CLUSTER-> 4
6.6 IF TASK IS QUERRY THE DATA-BASE AND IF
CLASS INFORMATION IS NEEDED AND IF
LARGE AMOUNT OF OUTPUT IS NEEDED
A41> WRITE A FILE IN CLASS MASTER FILE QUERRY LANGUAGE
A42> QUERRY CLASS MASTER FILE PSEUDOBATCH MODE
A38> PRINT SPOOLED OUTPUT

3. ABL-TRANSFORM SYSTALL;

```
#
#        SYSTALL # (OR THE OVERALL DESCRIPTION OF THE SYSTEM)
#
#
# THIS SEQUENCE OF PROCEDURES PROVIDES AN OVERVIEW OF THE ENTIRE
# ATTENDANCE MONITORING SYSTEM (# ATS. ). IT SPECIFIES HOW TO
# SET UP, RUN AND USE THIS SYSTEM. IT SHOWS THE LOGICAL INTER-
# RELATION AND OPERATIONAL SEQUENCE OF THE SUBSYSTEMS AMONG EACH
# OTHERS. FIRST WE SET UP THE BASIC SYSTEM, THAT IS THE DATA-BASE (#DB)
# WITH ITS QUERRY (#QUS.) AND UPDATE (#UPS.) UTILITY SUBSYSTEMS.
# SECOND WE DRAFT TO IT THE ATTENDANCE MONITORING SUBSYSTEM (#ATS.).
# FROM THEN ON WE CONCURRENTLY MAINTAIN THE EXACTNESS OF THE DATA-BASE
# WHILE ROUTINELY RUNNING THE ATTENDANCE MONITORING SUBSYSTEM
# IN ORDER TO EXTRACT THE RELATED INFORMATION FROM THE DATA-BASE.
#
#                  SYSTALL ACTION(1..44)
# SYSTALL HAS 44 ACTIONS. EACH OF THESE IS A COMPLEX ACTION
# PROCEDURE HAVING THE POSSIBILITY OF BEING DECOMPOSED INTO
# SUB-PROCEDURES. IN ORDER TO BE ABLE TO ESTABLISH CLEARLY
# THE HIERARCHIAL RELATIONSHIP BETWEEN THE SUB-ACTION AND
# THE ACTIONS OF SYSTALL THE FOLLOWING NOTATION WILL BE USED:
#
#        SYSTALL ACTION(N) = ACTION(N) = AN> XXXXXXXXX
#
#        SYSTALL SUB-ACTION(M) OF ACTION(N) =
#              = SYSTALL ACTION(N-M) = AN-M> XXXXXXXXX
#
#        SYSTALL SUB-SUBACTION(L) OF SUBACTION(M) OF ACTION(N) =
#              = SYSTALL ACTION(N-M-L) = AN-M-L> XXXXXXXXX
#
#   # FOR ALL CONTRACTIONS PLEASE REFER TO THE TABLE ON FIGURE 27.
#
#        THE PRODUCTION CYCLE OF # ATS. IS DESCRIBED IN DETAIL
#        IN THE FOLLOWING ACTIONS:
#
#        SYSTALL A(25) = DAILY SYSTEM INITIALIZATION
#        SYSTALL A(26) = DAILY TRANSACTION DATA GENERATION
#        SYSTALL A(27) = DAILY TRANSACTION DATA COLLECTION
#        SYSTALL A(28) = DAILY TRANSACTION DATA PROCESSING
#        SYSTALL A(29) = DAILY DATA DITRIBUTION
#        SYSTALL A(30) = DAILY DATA USAGE VERIFICATION
#
#
#           AN EXAMPLE OF THE COMPLEXITY OF SOME ACTIONS
#           IS ILLUSTRATED IN SYSTALL A(11), WHOSE LOGICAL
#           DESCRIPTION IS 3 LEVELS DEEP.
#
#        SYSTALL A(11) = UPDATE THE STUDENT MASTER FILE
#
```

CONTINUED ON THE NEXT PAGE

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	(ALTERNATIVES)
CLUSTERS -																	
C1	1	1	1	2	.	.	.	.	.	.	.	.	.	.	.	.	1 SET UP THE BASIC SYSTEM (# UTS. AND # DB.) ===>
C2	.	.	.	.	2	3	3	.	.	.	.	.	.	.	.	.	2 INITIAL DATA-BASE VERIFICATION. (# QUS. AND # UPS.() ===>
C3	.	.	.	.	.	.	.	3	4	.	.	.	.	.	.	.	3 SET UP THE ATTENDANCE MONITORING SUBSYSTEM (# UTS.) ===>
C4	.	.	.	.	.	.	.	.	.	4	4	4	4	4	4	5	4 DO REGULAR RUN OF THE ATTENDANCE MONITORING SUBSYSTEM (# ATS.)===>
C5	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	5 MAINTAIN THE EXACTNESS OF THE SYSTEMS DATA (# UPS.) ===>
C6	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	6 INFORMATION EXTRACTING AND REPORTING (# QUS.) ===>
PREDICATES -																	
P1	N	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	TERMINAL IS SET UP IN SCHOOL
P2	-	N	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-	ALL DATA-BASE FILES ARE SET UP ON-LINE
P3	-	-	N	Y	-	-	-	-	-	-	-	-	-	-	-	-	QUERRY AND UPDATE SUBSYSTEMS ARE SET UP
P4	-	-	-	N	-	-	-	-	-	-	-	-	-	-	-	-	ALLOCATED FIELDS IN THE DATA-BASE ARE FILLED
P5	-	-	-	-	N	Y	Y	-	-	-	-	-	-	-	-	-	DATA-BASE CONTENT IS VERIFIED
P6	-	-	-	-	-	Y	N	-	-	-	-	-	-	-	-	-	ERROR IN THE DATA-BASE
P7	-	-	-	-	-	-	-	N	N	Y	-	-	-	-	-	-	ATTENDANCE MONITORING SUBSYSTEM IS SET UP
P8	-	-	-	-	-	-	-	Y	-	-	-	-	-	-	-	-	DATA-BASE CONTAINS CORRECT NON TRUANCY INFORM.
P9	-	-	-	-	-	-	-	N	Y	-	-	-	-	-	-	-	TRANSACTION DATA GENERATION KITS ARE READY
P10	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	Y	Y	N	TASK IS ATTENDANCE MONITORING
P11	-	-	-	-	-	-	-	-	-	N	Y	Y	Y	Y	Y	-	PRODUCTION CYCLE IS STARTED
P12	-	-	-	-	-	-	-	-	-	-	Y	-	-	-	-	-	PRODUCTION DAY IS INITIALIZED TO THE COMPUTER
P13	-	-	-	-	-	-	-	-	-	-	-	Y	-	-	-	-	TRANSACTION DATA IS GENERATED
P14	-	-	-	-	-	-	-	-	-	-	-	-	Y	-	-	-	TRANSACTION DATA IS COLLECTED
P15	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	-	-	DATA IS PROCESSED
P16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	-	OUTPUT IS DISTRIBUTED
P17	-	-	-	-	-	-	-	-	-	Y	-	-	-	-	N	-	USAGE PARAMETERS ARE DETERMINED
P18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	TASK IS UPDATING THE DATA-BASE
P19	-	-	-	-	-	-	-	-	-	-	)	-	-	-	-	-	CHANGE INVOLVES NON #ATS. RELATED DATA
P20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	CHANGE INVOLVES #ATS. RELATED DATA
P21	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	TASK IS QUERRY THE DATA-BASE
P22	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	STUDENT INFORMATION IS NEEDED
P23	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	COURSE INFORMATION IS NEEDED
P24	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	CLASS INFORMATION IS NEEDED
P25	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	SMALL AMOUNT OF OUTPUT IS NEEDED
P26	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	LARGE AMOUNT OF OUTPUT IS NEEDED
P27	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	END OF THE YEAR
ACTIONS -																	
A1	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A1> INSTALL THE TERMINAL IN SCHOOL
A2	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A2> SET UP THE COURSE MASTER FILE
A3	.	2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A3> SET UP THE STUDENT MASTER FILE
A4	.	3	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A4> SET UP THE CLASS MASTER FILE
A5	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	A5> SET UP COURSE MASTER FILE UPDATER PROGRAM
A6	.	.	2	.	.	.	.	.	.	.	.	.	.	.	.	.	A6> SET UP STUDENT MASTER FILE UPDATER PROGRAM
A7	.	.	3	.	.	.	.	.	.	.	.	.	.	.	.	.	A7> SET UP COURSE MASTER FILE QUERRY PROGRAM
A8	.	.	4	.	.	.	.	.	.	.	.	.	.	.	.	.	A8> SET UP STUDENT MASTER FILE QUERRY PROGRAM
A9	.	.	5	.	.	.	.	.	.	.	.	.	.	.	.	.	A9> SET UP CLASS MASTER FILE QUERRY PROGRAM
A10	.	.	.	1	.	1	.	.	.	.	.	.	.	.	.	.	A10> UPDATE COURSE MASTER FILE INTERACTIVELY
A11	.	.	.	2	.	2	.	.	.	.	.	.	.	.	.	.	A11> UPDATE STUDENT MASTER FILE INTERACTIVELY
A12	.	.	.	3	.	3	.	.	.	.	.	.	.	.	.	.	A12> UPDATE CLASS MASTER FILE INTERACTIVELY
A13	.	.	.	.	2	.	.	.	.	.	.	.	.	.	.	.	A13> VERIFY COURSE MASTER FILE DATA
A14	.	.	.	.	4	.	.	.	.	.	.	.	.	.	.	.	A14> VERIFY STUDENT MASTER FILE DATA
A15	.	.	.	.	6	.	.	.	.	.	.	.	.	.	.	.	A15> VERIFY CLASS MASTER FILE DATA

CONTINUED ON THE NEXT PAGE

A16	.	.	.	.	.	.	.	3	.	.	.	.	.	.	.	.	A16>	PRODUCE TEACHERS DATA GENERATION KITS
A17	.	.	.	.	.	.	.	4	.	.	.	.	.	.	.	.	A17>	SET UP DATA COLLECTION ROUTES IN SCHOOL
A18	.	.	.	.	.	.	.	5	.	.	.	.	.	.	.	.	A18>	INSTALL OPTICAL MARK READER ON TERMINAL
A19	.	.	.	.	.	.	.	6	.	.	.	.	.	.	.	.	A19>	VERIFY DATA DISTRIBUTION ROUTES
A20	.	.	.	.	.	.	.	7	.	.	.	.	.	.	2	.	A20>	DETERMINE DATA USAGE PARAMETERS
A21	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	A21>	SET UP DAILY INITIALIZATION INTERACTIVE PROGRAM
A22	.	.	.	.	.	.	.	.	2	.	.	.	.	.	.	.	A22>	SET UP DATA COLLECTION PSEUDOBATCH PROGRAM
A23	.	.	.	.	.	.	.	.	3	.	.	.	.	.	.	.	A23>	SET UP SAIS. DATA PROCESSING PSEUDOBATCH PROGRAM
A24	.	.	.	.	.	.	.	.	4	.	.	.	.	.	.	.	A24>	SET UP PROCEDURE TO PRINT SPOOLED OUTPUT
A25	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	A25>	PERFORM DAILY SYSTEM INITIALIZATION
A26	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	A26>	PERFORM TRANSACTION DATA GENERATION
A27	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	A27>	PERFORM TRANSACTION DATA COLLECTION
A28	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	A28>	PERFORM DATA PROCESSING (REPORTING AND STORAGE)
A29	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	A29>	PERFORM DATA DISTRIBUTION
A30	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	A30>	VERIFY OUTPUT USAGE IN DECISION MAKING
A31	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A31>	PUNCH NEW $ ABS-CARDS
A32	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A32>	UPDATE TEACHERS DATA GENERATION KITS
A33	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A33>	QUERRY COURSE MASTER FILE INTERACTIVELY
A34	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A34>	QUERRY STUDENT MASTER FILE INTERACTIVELY
A35	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A35>	QUERRY CLASS MASTER FILE INTERACTIVELY
A36	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A36>	WRITE A FILE IN COURSE MASTER FILE QUERRY LANGUAGE
A37	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	A37>	QUERRY COURSE MASTER FILE PSEUDOBATCH MODE
A38	.	.	.	.	.	.	.	.	.	.	.	.	2	.	.	.	A38>	PRINT SPOOLED OUTPUT
A39	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A39>	WRITE A FILE IN STUDENT MASTER FILE QUERRY LANGUAGE
A40	.	.	.	.	3	.	.	.	.	.	.	.	.	.	.	.	A40>	QUERRY STUDENT MASTER FILE PSEUDOBATCH MODE
A41	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A41>	WRITE A FILE IN CLASS MASTER FILE QUERRY LANGUAGE
A42	.	.	.	.	5	.	.	1	.	.	.	.	.	.	.	.	A42>	QUERRY CLASS MASTER FILE PSEUDOBATCH MODE
A43	.	.	.	.	.	.	.	2	.	.	.	.	.	.	.	.	A43>	PRODUCE $ ABS-CARDS FOR THE CLASS
A44	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	1	A44>	NO ACTION- NEXT CLUSTER

```
*****************************************************************
*                                                               *
*       SYSTALL  # (OR THE OVERALL DESCRIPTION OF THE SYSTEM)   *
*                                                               *
*  THIS SEQUENCE OF PROCEDURES PROVIDES AN OVERVIEW OF THE ENTIRE *
*  ATTENDANCE MONITORING SYSTEM  (#  ATS,  ).  IT SPECIFIES HOW TO*
*  SET UP, RUN AND USE THIS SYSTEM.  IT SHOWS THE LOGICAL INTER-  *
*  RELATION AND OPERATIONAL SEQUENCE OF THE SUBSYSTEMS AMONG EACH *
*  OTHERS.  FIRST WE SET UP THE BASIC SYSTEM, THAT IS THE DATA-BASE (#DB)
*  WITH ITS QUERRY (#QUS.) AND UPDATE (#UPS.) UTILITY SUBSYSTEMS. *
*  SECOND WE GRAFT TO IT THE ATTENDANCE MONITORING SUBSYSTEM (#ATS.).
*  FROM THEN ON WE CONCURRENTLY MAINTAIN THE EXACTNESS OF THE DATA-BASE
*  WHILE ROUTINELY RUNNING THE ATTENDANCE MONITORING SUBSYSTEM    *
*  IN ORDER TO EXTRACT THE RELATED INFORMATION FROM THE DATA-BASE.*
*                                                               *
*                    SYSTALL ACTION(1..44)                      *
*  SYSTALL HAS 44 ACTIONS.  EACH OF THESE IS A COMPLEX ACTION   *
*  PROCEDURE HAVING THE POSSIBILITY OF BEING DECOMPOSED INTO    *
*  SUB-PROCEDURES.  IN ORDER TO BE ABLE TO ESTABLISH CLEARLY    *
*  THE HIERARCHIAL RELATIONSHIP BETWEEN THE SUB-ACTION AND      *
*  THE ACTIONS OF SYSTALL THE FOLLOWING NOTATION WILL BE USED:  *
*                                                               *
*          SYSTALL ACTION(N) = ACTION(N) = AN> XXXXXXXXX        *
*                                                               *
*          SYSTALL SUB-ACTION(M) OF ACTION(N) =                 *
*                = SYSTALL ACTION(N-M) = AN-M> XXXXXXXXX        *
*                                                               *
*          SYSTALL SUB-SUBACTION(L) OF SUBACTION(M) OF ACTION(N) =
*                = SYSTALL ACTION(N-M-L) = AN-M-L> XXXXXXXXX    *
*                                                               *
*    #  FOR ALL CONTRACTIONS PLEASE REFER TO THE TABLE ON FIGURE 22.
*                                                               *
*          THE PRODUCTION CYCLE OF # ATS.  IS DESCRIBED IN DETAIL
*          IN THE FOLLOWING ACTIONS:                            *
*                                                               *
*         SYSTALL A(25) = DAILY SYSTEM INITIALIZATION           *
*         SYSTALL A(26) = DAILY TRANSACTION DATA GENERATION     *
*         SYSTALL A(27) = DAILY TRANSACTION DATA COLLECTION     *
*         SYSTALL A(28) = DAILY TRANSACTION DATA PROCESSING     *
*         SYSTALL A(29) = DAILY DATA DISTRIBUTION               *
*         SYSTALL A(30) = DAILY DATA USAGE VERIFICATION         *
*                                                               *
*            AN EXAMPLE OF THE COMPLEXITY OF SOME ACTIONS       *
*            IS ILLUSTRATED IN SYSTALL A(11), WHOSE LOGICAL     *
*            DESCRIPTION IS 3 LEVELS DEEP.                      *
*                                                               *
*         SYSTALL A(11) = UPDATE THE STUDENT MASTER FILE        *
*                                                               *
*****************************************************************
```

CONTINUED ON THE NEXT PAGE

```
                17 18 19 20 21 22 23 24 25 26 _ (ALTERNATIVES)

                                                              CLUSTERS -
     C1  .  .  .  .  .  .  .  .  .  0   1 SET UP THE BASIC SYSTEM ( # UTS. AND # DB. )                    ===
     C2  .  .  .  .  .  .  .  .  .  0   2 INITIAL DATA-BASE VERIFICATION. ( # QUS. AND # UPS. )          ===
     C3  .  .  .  .  .  .  .  .  .  0   3 SET UP THE ATTENDANCE MONITORING SUBSYSTEM ( # UTS. )          ===
     C4  .  .  .  .  .  .  .  .  .  0   4 DO REGULAR RUN OF THE ATTENDANCE MONITORING SUBSYSTEM (# ATS.)===
     C5  4  4  6  .  .  .  .  .  .  0   5 MAINTAIN THE EXACTNESS OF THE SYSTEMS DATA  ( # UPS. )        ===
     C6  .  .  .  4  4  4  4  4  4  0   6 INFORMATION EXTRACTING AND-REPORTING ( # QUS. )               ===

                                                              PREDICATES -
     P1  -  -  -  -  -  -  -  -  -  -   TERMINAL IS SET UP IN SCHOOL
     P2  -  -  -  -  -  -  -  -  -  -   ALL DATA-BASE FILES ARE SET UP ON-LINE
     P3  -  -  -  -  -  -  -  -  -  -   QUERRY AND UPDATE SUBSYSTEMS ARE SET UP
     P4  -  -  -  -  -  -  -  -  -  -   ALLOCATED FIELDS IN THE DATA-BASE ARE FILLED
     P5  -  -  -  -  -  -  -  -  -  -   DATA-BASE CONTENT IS  VERIFIED
     P6  -  -  -  -  -  -  -  -  -  -   ERROR IN THE DATA-BASE
     P7  -  -  -  -  -  -  -  -  -  -   ATTENDANCE MONITORING SUBSYSTEM IS SET UP
     P8  -  -  -  -  -  -  -  -  -  -   DATA-BASE CONTAINS CORRECT NON TRUANCY INFORM.
     P9  -  -  -  -  -  -  -  -  -  -   TRANSACTION DATA GENERATION KITS ARE READY
     P10 -  -  -  -  -  -  -  -  -  -   TASK IS ATTENDANCE MONITORING
     P11 -  -  -  -  -  -  -  -  -  -   PRODUCTION CYCLE IS STARTED
     P12 -  -  -  -  -  -  -  -  -  -   PRODUCTION DAY IS INITIALIZED TO THE COMPUTER
     P13 -  -  -  -  -  -  -  -  -  -   TRANSACTION DATA IS GENERATED
     P14 -  -  -  -  -  -  -  -  -  -   TRANSACTION DATA IS COLLECTED
     P15 -  -  -  -  -  -  -  -  -  -   DATA IS PROCESSED
     P16 -  -  -  -  -  -  -  -  -  -   OUTPUT IS DISTRIBUTED
     P17 -  -  -  -  -  -  -  -  -  -   USAGE PARAMETERS ARE DETERMINED
     P18 Y  Y  N  -  -  -  -  -  -  -   TASK IS UPDATING THE DATA-BASE
     P19 Y  -  -  -  -  -  -  -  -  -   CHANGE INVOLVES NON #ATS. RELATED DATA
     P20 -  Y  -  -  -  -  -  -  -  -   CHANGE INVOLVES #ATS. RELATED DATA
     P21 -  -  -  Y  Y  Y  Y  Y  Y  -   TASK IS QUERRY THE DATA-BASE
     P22 -  -  -  -  Y  -  -  Y  -  -   STUDENT INFORMATION IS NEEDED
     P23 -  -  -  Y  -  -  Y  -  -  -   COURSE INFORMATION IS NEEDED
     P24 -  -  -  -  -  Y  -  -  Y  -   CLASS INFORMATION IS NEEDED
     P25 -  -  -  Y  Y  Y  -  -  -  -   SMALL AMOUNT OF OUTPUT IS NEEDED
     P26 -  -  -  -  -  -  Y  Y  Y  -   LARGE AMOUNT OF OUTPUT IS NEEDED
     P27 -  -  -  -  -  -  -  -  -  -   END OF THE YEAR

                                                              ACTIONS -
     A1  .  .  .  .  .  .  .  .  .  .   A1>  INSTALL THE TERMINAL IN SCHOOL
     A2  .  .  .  .  .  .  .  .  .  .   A2>  SET UP THE COURSE MASTER FILE
     A3  .  .  .  .  .  .  .  .  .  .   A3>  SET UP THE STUDENT MASTER FILE
     A4  .  .  .  .  .  .  .  .  .  .   A4>  SET UP THE CLASS MASTER FILE
     A5  .  .  .  .  .  .  .  .  .  .   A5>  SET UP COURSE MASTER FILE UPDATER PROGRAM
     A6  .  .  .  .  .  .  .  .  .  .   A6>  SET UP STUDENT MASTER FILE UPDATER PROGRAM
     A7  .  .  .  .  .  .  .  .  .  .   A7>  SET UP COURSE MASTER FILE QUERRY PROGRAM
     A8  .  .  .  .  .  .  .  .  .  .   A8>  SET UP STUDENT MASTER FILE QUERRY PROGRAM
     A9  .  .  .  .  .  .  .  .  .  .   A9>  SET UP CLASS MASTER FILE QUERRY PROGRAM
     A10 1  1  .  .  .  .  .  .  .  .   A10> UPDATE COURSE MASTER FILE INTERACTIVELY
     A11 2  2  .  .  .  .  .  .  .  .   A11> UPDATE STUDENT MASTER FILE INTERACTIVELY
     A12 3  3  .  .  .  .  .  .  .  .   A12> UPDATE CLASS MASTER FILE INTERACTIVELY
     A13 .  .  .  .  .  .  .  .  .  .   A13> VERIFY COURSE MASTER FILE DATA
     A14 .  .  .  .  .  .  .  .  .  .   A14> VERIFY STUDENT MASTER FILE DATA
     A15 .  .  .  .  .  .  .  .  .  .   A15> VERIFY CLASS MASTER FILE DATA
CONTINUED ON THE NEXT PAGE
```

A16	.	.	.	.	.	.	.	.	.	A16>	PRODUCE TEACHERS DATA GENERATION KITS
A17	.	.	.	.	.	.	.	.	.	A17>	SET UP DATA COLLECTION ROUTES IN SCHOOL
A18	.	.	.	.	.	.	.	.	.	A18>	INSTALL OPTICAL MARK READER ON TERMINAL
A19	.	.	.	.	.	.	.	.	.	A19>	VERIFY DATA DISTRIBUTION ROUTES
A20	.	.	.	.	.	.	.	.	.	A20>	DETERMINE DATA USAGE PARAMETERS
A21	.	.	.	.	.	.	.	.	.	A21>	SET UP DAILY INITIALIZATION INTERACTIVE PROGRAM
A22	.	.	.	.	.	.	.	.	.	A22>	SET UP DATA COLLECTION PSEUDOBATCH PROGRAM
A23	.	.	.	.	.	.	.	.	.	A23>	SET UP $ATS. DATA PROCESSING PSEUDOBATCH PROGRAM
A24	.	.	.	.	.	.	.	.	.	A24>	SET UP PROCEDURE TO PRINT SPOOLED OUTPUT
A25	.	.	.	.	.	.	.	.	.	A25>	PERFORM DAILY SYSTEM INITIALIZATION
A26	.	.	.	.	.	.	.	.	.	A26>	PERFORM TRANSACTION DATA GENERATION
A27	.	.	.	.	.	.	.	.	.	A27>	PERFORM TRANSACTION DATA COLLECTION
A28	.	.	.	.	.	.	.	.	.	A28>	PERFORM DATA PROCESSING (REPORTING AND STORAGE)
A29	.	.	.	.	.	.	.	.	.	A29>	PERFORM DATA DISTRIBUTION
A30	.	.	.	.	.	.	.	.	.	A30>	VERIFY OUTPUT USAGE IN DECISION MAKING
A31	.	5	.	.	.	.	.	.	.	A31>	PUNCH NEW $ ABS-CARDS
A32	.	6	.	.	.	.	.	.	.	A32>	UPDATE TEACHERS DATA GENERATION KITS
A33	.	4	.	1	.	.	.	.	.	A33>	QUERRY COURSE MASTER FILE INTERACTIVELY
A34	.	.	.	.	1	.	.	.	.	A34>	QUERRY STUDENT MASTER FILE INTERACTIVELY
A35	.	.	.	.	.	1	.	.	.	A35>	QUERRY CLASS MASTER FILE INTERACTIVELY
A36	.	.	.	.	.	.	1	.	.	A36>	WRITE A FILE IN COURSE MASTER FILE QUERRY LANGUAGE
A37	.	.	.	.	.	.	2	.	.	A37>	QUERRY COURSE MASTER FILE PSEUDOBATCH MODE
A38	.	.	.	.	.	.	3	3	3	A38>	PRINT SPOOLED OUTPUT
A39	.	.	.	.	.	.	.	1	.	A39>	WRITE A FILE IN STUDENT MASTER FILE QUERRY LANGUAGE
A40	.	.	.	.	.	.	.	2	.	A40>	QUERRY STUDENT MASTER FILE PSEUDOBATCH MODE
A41	.	.	.	.	.	.	.	.	1	A41>	WRITE A FILE IN CLASS MASTER FILE QUERRY LANGUAGE
A42	.	.	.	.	.	.	.	.	2	A42>	QUERRY CLASS MASTER FILE PSEUDOBATCH MODE
A43	.	.	.	.	.	.	.	.	.	A43>	PRODUCE $ ABS-CARDS FOR THE CLASS
A44	.	.	1	.	.	.	.	.	.	A44>	NO ACTION— NEXT CLUSTER

9. ABL-ROUTINE INITIALIZATION:

```
SYSTALL A(25) = A25> = PERFORM DAILY DATA USAGE DEFINITION
                      AND INITIALIZATION
                    = $INITIALIZATION WITH 18 ACTIONS
                    = SYSTALL A25>-(1..18)

AT THE BEGINNING OF THE DAY WHEN THE ATTENDANCE MONITORING
SUBSYSTEM IS TO BE RUN, THE DAY HAS TO BE INITIALIZED BY THE
OPERATOR TO THE SYSTEM. THAT MEANS ENTERING IN THE SYSTEM
CERTAIN SPECIFICATIONS AND OUTPUT SELECTION PARAMETERS.
THESE VALUES ARE STORED IN A SPECIAL BUFFER RECORD OF THE
STUDENT MASTER FILE UNTIL DATA PROCESSING TIME.

1.=== START                                          ===:          < 2 >

    1.1==ENTER PROCEDURE                        ==   --THEN-GO-TO-CLUSTER-> 2
2.=== EVICT OLD DATA, ENTER NEW PARAMETERS           ===:          < 0, 2, 2, 3 >

    2.1==WARNING, ATTEMPT TO INITIALIZE TWICE A DAY ==   --THEN-GO-TO-CLUSTER-> 0

    2.2==ENTER NEW PRODUCTION CYCLE SPECIFICATIONS  ==   --THEN-GO-TO-CLUSTER-> 2

    2.3==SPECIFICATION ERROR RECOVERY               ==   --THEN-GO-TO-CLUSTER-> 2

    2.4==OUTPUT SELECTION PARAMETER DEFINITION      ==   --THEN-GO-TO-CLUSTER-> 3
3.=== STORE PARAMETERS                               ===:          < 3, 0 >

    3.1==OUTPUT SELECTION PARAMETER ERROR RECOVERY  ==   --THEN-GO-TO-CLUSTER-> 3

    3.2==FINALIZE INITIALIZATION                    ==   --THEN-GO-TO-CLUSTER-> 0
```

9. ABL-ROUTINE INITIALIZATION;

```
SYSTALL A(25) = A25> = PERFORM DAILY DATA USAGE DEFINITION
                      AND INITIALIZATION
                    = #INITIALIZATION WITH 18 ACTIONS
                    = SYSTALL A25>-(1..18)

AT THE BEGINNING OF THE DAY WHEN THE ATTENDANCE MONITORING
SUBSYSTEM IS TO BE RUN, THE DAY HAS TO BE INITIALIZED BY THE
OPERATOR TO THE SYSTEM. THAT MEANS ENTERING IN THE SYSTEM
CERTAIN SPECIFICATIONS AND OUTPUT SELECTION PARAMETERS.
THESE VALUES ARE STORED IN A SPECIAL BUFFER RECORD OF THE
STUDENT MASTER FILE UNTIL DATA PROCESSING TIME.

1.=== START                                                    ===#          < 2 >

    1.1==ENTER PROCEDURE                                 ==    --THEN-GO-TO-CLUSTER-> 2
         1.1 IF START
2.=== EVICT OLD DATA, ENTER NEW PARAMETERS                     ===#          < 0, 2, 2, 3 >

    2.1==WARNING, ATTEMPT TO INITIALIZE TWICE A DAY      ==    --THEN-GO-TO-CLUSTER-> 0
         2.1 IF PRODUCTION CYCLE DAY IS ALREADY INITIALIZED

    2.2==ENTER NEW PRODUCTION CYCLE SPECIFICATIONS       ==    --THEN-GO-TO-CLUSTER-> 2
         2.2 IF NOT PRODUCTION CYCLE DAY IS ALREADY INITIALIZED AND IF
                NOT ERROR RECOVERY OF ENTERED DATA

    2.3==SPECIFICATION ERROR RECOVERY                    ==    --THEN-GO-TO-CLUSTER-> 2
         2.3 IF NOT PRODUCTION CYCLE DAY IS ALREADY INITIALIZED AND IF
                ERROR RECOVERY OF ENTERED DATA AND IF
                NOT VALID SPECIFICATIONS

    2.4==OUTPUT SELECTION PARAMETER DEFINITION           ==    --THEN-GO-TO-CLUSTER-> 3
         2.4 IF NOT PRODUCTION CYCLE DAY IS ALREADY INITIALIZED AND IF
                NOT ERROR RECOVERY OF ENTERED DATA AND IF
                VALID SPECIFICATIONS
3.=== STORE PARAMETERS                                         ===#          < 3, 0 >

    3.1==OUTPUT SELECTION PARAMETER ERROR RECOVERY       ==    --THEN-GO-TO-CLUSTER-> 3
         3.1 IF NOT PRODUCTION CYCLE DAY IS ALREADY INITIALIZED AND IF
                ERROR RECOVERY OF ENTERED DATA AND IF
                NOT VALID OUTPUT PARAMETERS

    3.2==FINALIZE INITIALIZATION                         ==    --THEN-GO-TO-CLUSTER-> 0
         3.2 IF NOT ERROR RECOVERY OF ENTERED DATA AND IF
                VALID OUTPUT PARAMETERS
```

9. ABL-ROUTINE INITIALIZATION:

```
SYSTALL A(25) = A25> = PERFORM DAILY DATA USAGE DEFINITION
                      AND INITIALIZATION
                    = #INITIALIZATION WITH 18 ACTIONS
                    = SYSTALL A25>-(1..18)

    AT THE BEGINNING OF THE DAY WHEN THE ATTENDANCE MONITORING
    SUBSYSTEM IS TO BE RUN, THE DAY HAS TO BE INITIALIZED BY THE
    OPERATOR TO THE SYSTEM. THAT MEANS ENTERING IN THE SYSTEM
    CERTAIN SPECIFICATIONS AND OUTPUT SELECTION PARAMETERS.
    THESE VALUES ARE STORED IN A SPECIAL BUFFER RECORD OF THE
    STUDENT MASTER FILE UNTIL DATA PROCESSING TIME.
```

1.=== START ===: < 2 >

1.1==ENTER PROCEDURE == --THEN-GO-TO-CLUSTER-> 2
1.1 IF START
A25-1> ENTER COMMAND = CALL,ABSCON

2.=== EVICT OLD DATA, ENTER NEW PARAMETERS ===: < 0, 2, 2, 3 >

2.1==WARNING, ATTEMPT TO INITIALIZE TWICE A DAY == --THEN-GO-TO-CLUSTER-> 0
2.1 IF PRODUCTION CYCLE DAY IS ALREADY INITIALIZED
A25-6> DISPLAY LAST PRODUCTION CYCLE SPECIFICATIONS AND PARAMETERS
A25-17> WRITE ERROR MESSAGE
A25-18> INTERRUPT EXECUTION

2.2==ENTER NEW PRODUCTION CYCLE SPECIFICATIONS == --THEN-GO-TO-CLUSTER-> 2
2.2 IF NOT PRODUCTION CYCLE DAY IS ALREADY INITIALIZED AND IF
NOT ERROR RECOVERY OF ENTERED DATA
A25-6> DISPLAY LAST PRODUCTION CYCLE SPECIFICATIONS AND PARAMETERS
A25-2> PURGE PREVIOUS VALID-TRANSACTION FILE
A25-3> SET UP FLAG
A25-4> OPEN STUDENT MASTER FILE
A25-5> RETRIEVE DABS-RECORD FROM STUDENT MASTER FILE
A25-7> ENTER PRESENT DAYS SPECIFICATIONS
A25-8> ENTER SCHOOL DAY
A25-9> ENTER SEMESTER
A25-10> ENTER SCHOOL WEEK
A25-11> GET DATE FROM SYSTEM
A25-12> DISPLAY AND VERIFY ENTERED SPECIFICATIONS

CONTINUED ON THE NEXT PAGE

```
2.3=SPECIFICATION ERROR RECOVERY                    ==    --THEN-GO-TO-CLUSTER-> 2
    2.3 IF  NOT  PRODUCTION CYCLE DAY IS ALREADY INITIALIZED  AND IF
         ERROR RECOVERY OF ENTERED  DATA  AND IF
         NOT  VALID SPECIFICATIONS
            A25-5> DISPLAY LAST PRODUCTION CYCLE SPECIFICATIONS AND PARAMETERS
            A25-7> ENTER PRESENT DAYS SPECIFICATIONS
            A25-8> ENTER SCHOOL DAY
            A25-9> ENTER SEMESTER
            A25-10> ENTER SCHOOL WEEK
            A25-11> GET DATE FROM SYSTEM
            A25-12> DISPLAY AND VERIFY ENTERED SPECIFICATIONS


2.4=OUTPUT SELECTION PARAMETER DEFINITION           ==    --THEN-GO-TO-CLUSTER-> 3
    2.4 IF  NOT  PRODUCTION CYCLE DAY IS ALREADY INITIALIZED  AND IF
         NOT  ERROR RECOVERY OF ENTERED  DATA  AND IF
         VALID SPECIFICATIONS
            A25-13> ENTER OUTPUT SELECTION PARAMETERS
            A25-14> DISPLAY AND VERIFY ENTERED PARAMETERS

3.== STORE PARAMETERS                                ==*          < 3, 0 >

3.1=OUTPUT SELECTION PARAMETER ERROR RECOVERY       ==    --THEN-GO-TO-CLUSTER-> 3
    3.1 IF  NOT  PRODUCTION CYCLE DAY IS ALREADY INITIALIZED  AND IF
         ERROR RECOVERY OF ENTERED  DATA  AND IF
         NOT  VALID OUTPUT PARAMETERS
            A25-13> ENTER OUTPUT SELECTION PARAMETERS
            A25-14> DISPLAY AND VERIFY ENTERED PARAMETERS


3.2=FINALIZE INITIALIZATION                         ==    --THEN-GO-TO-CLUSTER-> 0
    3.2 IF  NOT  ERROR RECOVERY OF ENTERED  DATA  AND IF
         VALID OUTPUT PARAMETERS
            A25-15> STORE ENTERED VALUES IN $ABS-REC OF $STUD-MAST-FILE
            A25-16> CLOSE STUDENT MASTER FILE
```

9. ABL-TRANSFORM INITIALIZATION:

```
SYSTALL A(25) = A25> = PERFORM DAILY DATA USAGE DEFINITION
                AND INITIALIZATION
              = INITIALIZATION WITH 18 ACTIONS
              = SYSTALL A25>-(1..18)

AT THE BEGINNING OF THE DAY WHEN THE ATTENDANCE MONITORING
SUBSYSTEM IS TO BE RUN, THE DAY HAS TO BE INITIALIZED BY THE
OPERATOR TO THE SYSTEM. THAT MEANS ENTERING IN THE SYSTEM
CERTAIN SPECIFICATIONS AND OUTPUT SELECTION PARAMETERS.
THESE VALUES ARE STORED IN A SPECIAL BUFFER RECORD OF THE
STUDENT MASTER FILE UNTIL DATA PROCESSING TIME.
```

	1	2	3	4	5	6	7	8	(ALTERNATIVES)
									CLUSTERS -
C1	2	.	.	.	.	.	.	0	=== START ===
C2	.	0	2	2	3	.	.	0	=== EVICT OLD DATA; ENTER NEW PARAMETERS ===
C3	.	.	.	.	.	3	0	0	=== STORE PARAMETERS ===
									PREDICATES -
P1	Y	-	-	-	-	-	-	-	START
P2	-	Y	N	N	N	N	-	-	PRODUCTION CYCLE DAY IS ALREADY INITIALIZED
P3	-	-	N	Y	N	Y	N	-	ERROR RECOVERY OF ENTERED DATA
P4	-	-	-	N	Y	-	-	-	VALID SPECIFICATIONS
P5	-	-	-	-	-	N	Y	-	VALID OUTPUT PARAMETERS
									ACTIONS -
A1	1	.	.	.	.	.	.	.	A25-1> ENTER COMMAND = CALL,ABSCOM
A2	.	.	2	.	.	.	.	.	A25-2> PURGE PREVIOUS VALID-TRANSACTION FILE
A3	.	.	3	.	.	.	.	.	A25-3> SET UP FLAG
A4	.	.	4	.	.	.	.	.	A25-4> OPEN STUDENT MASTER FILE
A5	.	.	5	.	.	.	.	.	A25-5> RETRIEVE #ABS-RECORD FROM STUDENT MASTER FILE
A6	.	1	1	1	.	.	.	.	A25-6> DISPLAY LAST PRODUCTION CYCLE SPECIFICATIONS AND PARAMETERS
A7	.	.	6	2	.	.	.	.	A25-7> ENTER PRESENT DAYS SPECIFICATIONS
A8	.	.	7	3	.	.	.	.	A25-8> ENTER SCHOOL DAY
A9	.	.	8	4	.	.	.	.	A25-9> ENTER SEMESTER
A10	.	.	9	5	.	.	.	.	A25-10> ENTER SCHOOL WEEK
A11	.	.	10	6	.	.	.	.	A25-11> GET DATE FROM SYSTEM
A12	.	.	11	7	.	.	.	.	A25-12> DISPLAY AND VERIFY ENTERED SPECIFICATIONS
A13	.	.	.	.	1	1	.	.	A25-13> ENTER OUTPUT SELECTION PARAMETERS
A14	.	.	.	.	2	2	.	.	A25-14> DISPLAY AND VERIFY ENTERED PARAMETERS
A15	.	.	.	.	.	.	1	.	A25-15> STORE ENTERED VALUES IN #ABS-REC OF #STUD-MAST-FILE
A16	.	.	.	.	.	.	2	.	A25-16> CLOSE STUDENT MASTER FILE
A17	.	2	.	.	.	.	.	.	A25-17> WRITE ERROR MESSAGE
A18	.	3	.	.	.	.	.	.	A25-18> INTERRUPT EXECUTION

10. ABL-ROUTINE GENERDATA;


```
*****************************************************************************
*                                                                           *
*        SYSTALL A(26) = A26> = GENERATE TRANSACTION DATA                    *
*                       ------------------------------                      *
*                                                                           *
*                           = #GENERDATA WITH 13 ACTIONS                    *
*                           = SYSTALL A26>-(1..13)                          *
*                                                                           *
*    THIS PROCEDURE PERFORMS THE FOLLOWING=                                 *
* 1  TRANSACTION DATA IS GENERATED BY THE TEACHERS AT EVERY LECTURE         *
*    WITH THE HELP OF THE DATA GENERATION KIT.                              *
* 2  TEACHERS TAKE THE ROLL CALL WITH THE CLASS LIST                        *
* 3  EACH STUDENT ABSENT HAS HIS #ABS-CARD REMOVED FROM THE CLASS' DECK     *
*    AND PUT IN THE POUCH RESERVED FOR THE ABSENT STUDENTS ON THE DATA      *
*    GENERATION KIT. THESE CARDS ARE PREIDENTIFIED AND THE ENTIRE           *
*    ROLLCALL TAKES ONLY 30-60 SECONDS OF THE TEACHER'S TIME                *
* 4  DATA GENERATION KIT IS READIED FOR COLLECTION                          *
*                                                                           *
*****************************************************************************
```

```
1.=== START                                          ===#          < 2 >
    1.1==PICK UP NECCESSARY MATERIAL                  ==   --THEN-GO-TO-CLUSTER-> 2
2.=== DATA GENERATION                                ===#          < 2, 0 >
    2.1==UPDATE INDIVIDUAL DATA GENERATION KIT        ==   --THEN-GO-TO-CLUSTER-> 2
    2.2==DATA GENERATION                              ==   --THEN-GO-TO-CLUSTER-> 0
```

```
10.  ABL-ROUTINE  GENERDATA;
```

```
*************************************************************************
*                                                                       *
*        SYSTALL A(26) = A26> = GENERATE TRANSACTION DATA                *
*                                                                       *
*                        = %GENERDATA WITH 13 ACTIONS                   *
*                        = SYSTALL A26>-(1..13)                         *
*                                                                       *
*    THIS PROCEDURE PERFORMS THE FOLLOWING=                             *
*  1   TRANSACTION DATA IS GENERATED BY THE TEACHERS AT EVERY LECTURE    *
*      WITH THE HELP OF THE DATA GENERATION KIT.                        *
*  2   TEACHERS TAKE THE ROLL CALL WITH THE CLASS LIST                  *
*  3   EACH STUDENT ABSENT HAS HIS ABS-CARD REMOVED FROM THE CLASS' DECK *
*      AND PUT IN THE POUCH RESERVED FOR THE ABSENT STUDENTS ON THE DATA *
*      GENERATION KIT. THESE CARDS ARE PREIDENTIFIED AND THE ENTIRE      *
*      ROLLCALL TAKES ONLY 30-60 SECONDS OF THE TEACHER'S TIME          *
*  4   DATA GENERATION KIT IS READIED FOR COLLECTION                    *
*                                                                       *
*************************************************************************
```

```
1.=== START                                                  ===#          < 2 >

    1.1==PICK UP NECCESSARY MATERIAL                         ==    --THEN-GO-TO-CLUSTER-> 2
        1.1P  NO CONDITIONS

2.=== DATA GENERATION                                        ===#          < 2, 0 >

    2.1==UPDATE INDIVIDUAL DATA GENERATION KIT               ==    --THEN-GO-TO-CLUSTER-> 2
        2.1 IF  NOT  DATA GENERATION KIT IS UPDATED

    2.2==DATA GENERATION                                     ==    --THEN-GO-TO-CLUSTER-> 0
        2.2 IF  DATA GENERATION KIT IS UPDATED  AND IF
                THE PROCESS IS SIMPLE AND ACCEPTED BY TEACHERS  AND IF
                FAULTY DATA GENERATION POSSIBILITY IS MINIMAL AND ACCEPTABLE  AND IF
                IT TAKES LESS THAN A MINUTE
```

10. ABL-ROUTINE GENERDATA;


```
*************************************************************************
*                                                                       *
*        SYSTALL A(26) = A26> = GENERATE TRANSACTION DATA                *
*                                                                       *
*                        = $GENERDATA WITH 13 ACTIONS                   *
*                        = SYSTALL A26>-(1..13)                         *
*                                                                       *
*    THIS PROCEDURE PERFORMS THE FOLLOWING-                             *
* 1  TRANSACTION DATA IS GENERATED BY THE TEACHERS AT EVERY LECTURE      *
*    WITH THE HELP OF THE DATA GENERATION KIT.                          *
* 2  TEACHERS TAKE THE ROLL CALL WITH THE CLASS LIST                    *
* 3  EACH STUDENT ABSENT HAS HIS $ABS-CARD REMOVED FROM THE CLASS' DECK  *
*    AND PUT IN THE POUCH RESERVED FOR THE ABSENT STUDENTS ON THE DATA   *
*    GENERATION KIT. THESE CARDS ARE PREIDENTIFIED AND THE ENTIRE        *
*    ROLLCALL TAKES ONLY 30-60 SECONDS OF THE TEACHER'S TIME             *
* 4  DATA GENERATION KIT IS READIED FOR COLLECTION                       *
*                                                                       *
*************************************************************************

1.=== START                                                  ===$        < 2 >

    1.1=PICK UP NECCESSARY MATERIAL                      ==   --THEN-GO-TO-CLUSTER-> 2
        1.1P  NO CONDITIONS

              A26-1>  PICK UP DATA GENERATION KIT FROM MAILBOX
              A26-2>  PICK UP NEW CLASS LISTS
              A26-3>  PICK UP NEW $ABS-CARDS
              A26-4>  GO TO CLASS

2.=== DATA GENERATION                                        ===$        < 2, 0 >

    2.1=UPDATE INDIVIDUAL DATA GENERATION KIT            ==   --THEN-GO-TO-CLUSTER-> 2
        2.1 IF  NOT  DATA GENERATION KIT IS UPDATED
              A26-10>  INSERT NEW STUDENTS $ABS-CARDS IN THE DECK
              A26-11>  THROW AWAY DELETED STUDENTS $ABS-CARDS
              A26-12>  INSERT NEW CLASSLIST IN THE BIG POUCH


    2.2=DATA GENERATION                                  ==   --THEN-GO-TO-CLUSTER-> 0
        2.2 IF  DATA GENERATION KIT IS UPDATED  AND IF
             THE PROCESS IS SIMPLE AND ACCEPTED BY TEACHERS  AND IF
             FAULTY DATA GENERATION POSSIBILITY IS MINIMAL AND ACCEPTABLE  AND IF
             IT TAKES LESS THAN A MINUTE
              A26-5>  USE CLASS LIST IN BIG POUCH
              A26-6>  TAKE $ABS-CARDS FOR THE CLASS FROM POUCH
              A26-7>  MAKE ROLL CALL
              A26-8>  PUT $ABS-CARDS OF PRESENT STUDENTS IN RIGHT POUCH
              A26-9>  PUT $ABS-CARDS OF ABSENT STUDENTS IN LEFT POUCH
              A26-13>  READY THE DATA GENERATION KIT FOR COLLECTION PICK-UP
```

10. ABL-TRANSFORM GENERDATA;

```
*****************************************************************************
*                                                                           *
*        SYSTALL A(26) = A26> = GENERATE TRANSACTION DATA                    *
*                                                                           *
*                        = GENERDATA WITH 13 ACTIONS                        *
*                        = SYSTALL A26>-(1..13)                             *
*                                                                           *
*    THIS PROCEDURE PERFORMS THE FOLLOWING=                                 *
* 1  TRANSACTION DATA IS GENERATED BY THE TEACHERS AT EVERY LECTURE          *
*    WITH THE HELP OF THE DATA GENERATION KIT.                              *
* 2  TEACHERS TAKE THE ROLL CALL WITH THE CLASS LIST                        *
* 3  EACH STUDENT ABSENT HAS HIS #ABS-CARD REMOVED FROM THE CLASS' DECK     *
*    AND PUT IN THE POUCH RESERVED FOR THE ABSENT STUDENTS ON THE DATA      *
*    GENERATION KIT. THESE CARDS ARE PREIDENTIFIED AND THE ENTIRE           *
*    ROLLCALL TAKES ONLY 30-40 SECONDS OF THE TEACHER'S TIME                *
* 4  DATA GENERATION KIT IS READIED FOR COLLECTION                          *
*                                                                           *
*****************************************************************************
```

	1	2	3	4	(ALTERNATIVES)
					CLUSTERS -
C1	2	.	.	0	== START ==
C2	.	2	0	0	== DATA GENERATION ==
					PREDICATES -
P1	-	N	Y	-	DATA GENERATION KIT IS UPDATED
P2	-	-	Y	-	THE PROCESS IS SIMPLE AND ACCEPTED BY TEACHERS
P3	-	-	Y	-	FAULTY DATA GENERATION POSSIBILITY IS MINIMAL AND ACCEPTABLE
P4	-	-	Y	-	IT TAKES LESS THAN A MINUTE
					ACTIONS -
A1	1	.	.	.	A26-1> PICK UP DATA GENERATION KIT FROM MAILBOX
A2	2	.	.	.	A26-2> PICK UP NEW CLASS LISTS
A3	3	.	.	.	A26-3> PICK UP NEW #ABS-CARDS
A4	4	.	.	.	A26-4> GO TO CLASS
A5	.	.	1	.	A26-5> USE CLASS LIST IN BIG POUCH
A6	.	.	2	.	A26-6> TAKE #ABS-CARDS FOR THE CLASS FROM POUCH
A7	.	.	3	.	A26-7> MAKE ROLL CALL
A8	.	.	4	.	A26-8> PUT #ABS-CARDS OF PRESENT STUDENTS IN RIGHT POUCH
A9	.	.	5	.	A26-9> PUT #ABS-CARDS OF ABSENT STUDENTS IN LEFT POUCH
A10	.	1	.	.	A26-10> INSERT NEW STUDENTS #ABS-CARDS IN THE DECK
A11	.	2	.	.	A26-11> THROW AWAY DELETED STUDENTS #ABS-CARDS
A12	.	3	.	.	A26-12> INSERT NEW CLASSLIST IN THE BIG POUCH
A13	.	.	6	.	A26-13> READY THE DATA GENERATION KIT FOR COLLECTION PICK-UP

16. ABL-ROUTINE DATACOLLECT;


```
*****************************************************************************
*                                                                           *
*       SYSTALL A(27)= A27>  = TRANSACTION DATA COLLECTION                  *
*                            = @DATACOLLECT WITH 14 ACTIONS                 *
*                            = SYSTALL A27>-(1..14)                         *
*                                                                           *
*  DATA COLLECTION IS A VERY CRITICAL OPERATION OF THE @ATS SYSTEM.  IF IT DOES*
*  NOT RESPECT THE PREVIOUSLY DETERMINED CRITERIONS SYSTEM REJECTION WILL OCCUR.*
*                                                                           *
*                                                                           *
*     THIS ROUTINE DOES THE FOLLOWING TASKS=                                *
*  1  DATA GENERATION KITS ARE COLLECTED BY THE CORRIDOR SUPERVISOR         *
*     FROM EACH TEACHER TEN MINUTES AFTER THE LECTURE STARTED               *
*  2  ABSENT STUDENTS CARDS ARE EXTRACTED FROM THE POUCHES                  *
*  3  EACH SUPERVISOR SENDS HIS CARDS TO THE TERMINAL SITE                  *
*  4  FRAGMENTED DATA COLLECTION PROCESSING IS DONE THROUGH THE             *
*     CARD READER INTO THE SYSTEM AT EVERY LECTURE  ++                      *
*  5  TRANSACTION DATA IS VERIFIED BY THE SYSTEM                            *
*  6  TRANSACTION DATA IS COLLECTED IN A BUFFER FILE WITH THE PREVIOUS      *
*     LECTURES' TRANSACTION DATA                                            *
*  7  OUTPUT IS PRINTED WITH OPTIONAL SUMMARY REPORT OF ABSENT              *
*     STUDENT LISTING TO EACH ASSISTANT PRINCIPAL (REPORT A)                *
*                                                                           *
*                                                                           *
*         THE LOGICAL DESCRIPTION OF @DATACOLLECT IS 3 LEVELS DEEP          *
*     AT SYSTALL A27-13> = PERFORM FRAGMENTED DATA INPUT                    *
*                       = CALL @FRAGINPUT WITH 13 ACTIONS                   *
*                       = SYSTALL A27-13>-(1..13)                           *
*                                                                           *
*               AT SYSTALL A27-13-5> = PERFORM DATA VERIFICATION             *
*                                   = @VERIFDATA WITH 12 ACTIONS            *
*                                   = SYSTALL A27-13-5>-(1..12)             *
*                                                                           *
*  ++ THIS FEATURE IS ONE OF THE TECHNICAL HIGHLIGHTS OF THIS SYSTEM        *
*     (SEE CHAPTER FOUR FOR THE IMPLEMENTATION OF THIS TASK). WE HAVE       *
*     BEEN ABLE TO ACHIEVE AN INPUT RATE OF NEARLY 3000 TRANSACTIONS        *
*     PER HOUR WITH AN EXTREMELY LOW ERROR RATE.                            *
*                                                                           *
*****************************************************************************

1.=== ENTER TRANSACTION DATA COLLECTION PROCESS                    ===*    < 0, 1, 2 >
    1.1==DESIGN PROBLEM                    ==   --THEN-GO-TO-CLUSTER-> 0
    1.2==ORGANIZATION PROBLEM              ==   --THEN-GO-TO-CLUSTER-> 1
    1.3==INITIATE DATA COLLECTION          ==   --THEN-GO-TO-CLUSTER-> 2
2.=== DATA COLLECTION                                         ===*    < 2, 2, 0 >
    2.1==TRANSACTION DATA IS NOT READY     ==   --THEN-GO-TO-CLUSTER-> 2
    2.2==TRANSACTION DATA IS READY         ==   --THEN-GO-TO-CLUSTER-> 2
    2.3==LAST LECTURES TRANSACTION DATA IS READY  ==   --THEN-GO-TO-CLUSTER-> 0
```

16. ABL-ROUTINE DATACOLLECT;

```
        SYSTALL A(27) = A27>  = TRANSACTION DATA COLLECTION
                              = $DATACOLLECT WITH 14 ACTIONS
                              = SYSTALL A27>-(1..14)

DATA COLLECTION IS A VERY CRITICAL OPERATION OF THE $ATS SYSTEM.  IF IT DOES
NOT RESPECT THE PREVIOUSLY DETERMINED CRITERIONS SYSTEM REJECTION WILL OCCUR.


   THIS ROUTINE DOES THE FOLLOWING TASKS=
1  DATA GENERATION KITS ARE COLLECTED BY THE CORRIDOR SUPERVISOR
   FROM EACH TEACHER TEN MINUTES AFTER THE LECTURE STARTED
2  ABSENT STUDENTS CARDS ARE EXTRACTED FROM THE POUCHES
3  EACH SUPERVISOR SENDS HIS CARDS TO THE TERMINAL SITE
4  FRAGMENTED DATA COLLECTION PROCESSING IS DONE THROUGH THE
   CARD READER INTO THE SYSTEM AT EVERY LECTURE  ++
5  TRANSACTION DATA IS VERIFIED BY THE SYSTEM
6  TRANSACTION DATA IS COLLECTED IN A BUFFER FILE WITH THE PREVIOUS
   LECTURES' TRANSACTION DATA
7  OUTPUT IS PRINTED WITH OPTIONAL SUMMARY REPORT OF ABSENT
   STUDENT LISTING TO EACH ASSISTANT PRINCIPAL (REPORT A)


      THE LOGICAL DESCRIPTION OF $DATACOLLECT IS 3 LEVELS DEEP
   AT SYSTALL A27-13> = PERFORM FRAGMENTED DATA INPUT
                     = CALL $FRAGINPUT WITH 13 ACTIONS
                     = SYSTALL A27-13>-(1..13)

            AT SYSTALL A27-13-5> = PERFORM DATA VERIFICATION
                                 = $VERIFDATA WITH 12 ACTIONS
                                 = SYSTALL A27-13-5>-(1..12)

++ THIS FEATURE IS ONE OF THE TECHNICAL HIGHLIGHTS OF THIS SYSTEM
   (SEE CHAPTER FOUR FOR THE IMPLEMENTATION OF THIS TASK). WE HAVE
   BEEN ABLE TO ACHIEVE AN INPUT RATE OF NEARLY 3000 TRANSACTIONS
   PER HOUR WITH AN EXTREMELY LOW ERROR RATE.
```

CONTINUED ON THE NEXT PAGE

1.=== ENTER TRANSACTION DATA COLLECTION PROCESS ===* < 0, 1, 2 >

1.1==DESIGN PROBLEM == --THEN-GO-TO-CLUSTER-> 0
1.1 IF NOT DATA GENERATION PROCESS IS ACCEPTABLE TO THE TEACHERS AND IF
NOT ACCEPTABLE ERROR RATE

1.2==ORGANIZATION PROBLEM == --THEN-GO-TO-CLUSTER-> 1
1.2 IF DATA GENERATION PROCESS IS ACCEPTABLE TO THE TEACHERS AND IF
NOT DATA COLLECTION ROUTE IS ORGANIZED AND IF
NOT ACCEPTABLE DELAY IS 15 MINUTES FOR DATA GENERATION

1.3==INITIATE DATA COLLECTION == --THEN-GO-TO-CLUSTER-> 2
1.3 IF DATA GENERATION PROCESS IS ACCEPTABLE TO THE TEACHERS AND IF
DATA COLLECTION ROUTE IS ORGANIZED AND IF
ACCEPTABLE ERROR RATE AND IF
ACCEPTABLE DELAY IS 15 MINUTES FOR DATA GENERATION AND IF
ACCEPTABLE DELAY IS 30 MINUTES FOR DATA COLLECTION

2.=== DATA COLLECTION ===* < 2, 2, 0 >

2.1==TRANSACTION DATA IS NOT READY == --THEN-GO-TO-CLUSTER-> 2
2.1 IF NOT TEACHERS FINISHED DATA GENERATION FOR THE CURRENT LECTURE

2.2==TRANSACTION DATA IS READY == --THEN-GO-TO-CLUSTER-> 2
2.2 IF TEACHERS FINISHED DATA GENERATION FOR THE CURRENT LECTURE AND IF
ACCEPTABLE DELAY IS 15 MINUTES FOR DATA GENERATION AND IF
ACCEPTABLE DELAY IS 30 MINUTES FOR DATA COLLECTION AND IF
MORE LECTURES AFTER THIS

2.3==LAST LECTURES TRANSACTION DATA IS READY == --THEN-GO-TO-CLUSTER-> 0
2.3 IF TEACHERS FINISHED DATA GENERATION FOR THE CURRENT LECTURE AND IF
ACCEPTABLE DELAY IS 15 MINUTES FOR DATA GENERATION AND IF
ACCEPTABLE DELAY IS 30 MINUTES FOR DATA COLLECTION AND IF
NOT MORE LECTURES AFTER THIS

16. ABL-ROUTINE DATACOLLECT;

```
*****************************************************************************
*                                                                           *
*        SYSTALL A(27) = A27>  = TRANSACTION DATA COLLECTION                *
*                                                                           *
*                              = #DATACOLLECT WITH 14 ACTIONS               *
*                              = SYSTALL A27>-(1..14)                       *
*                                                                           *
*  DATA COLLECTION IS A VERY CRITICAL OPERATION OF THE #ATS SYSTEM.  IF IT DOES*
*  NOT RESPECT THE PREVIOUSLY DETERMINED CRITERIONS SYSTEM REJECTION WILL OCCUR*.
*                                                                           *
*                                                                           *
*     THIS ROUTINE DOES THE FOLLOWING TASKS=                                *
*  1  DATA GENERATION KITS ARE COLLECTED BY THE CORRIDOR SUPERVISOR         *
*     FROM EACH TEACHER TEN MINUTES AFTER THE LECTURE STARTED               *
*  2  ABSENT STUDENTS CARDS ARE EXTRACTED FROM THE POUCHES                  *
*  3  EACH SUPERVISOR SENDS HIS CARDS TO THE TERMINAL SITE                  *
*  4  FRAGMENTED DATA COLLECTION PROCESSING IS DONE THROUGH THE             *
*     CARD READER INTO THE SYSTEM AT EVERY LECTURE  ++                      *
*  5  TRANSACTION DATA IS VERIFIED BY THE SYSTEM                            *
*  6  TRANSACTION DATA IS COLLECTED IN A BUFFER FILE WITH THE PREVIOUS      *
*     LECTURES' TRANSACTION DATA                                            *
*  7  OUTPUT IS PRINTED WITH OPTIONAL SUMMARY REPORT OF ABSENT              *
*     STUDENT LISTING TO EACH ASSISTANT PRINCIPAL (REPORT A)                *
*                                                                           *
*                                                                           *
*        THE LOGICAL DESCRIPTION OF #DATACOLLECT IS 3 LEVELS DEEP           *
*     AT SYSTALL A27-13> = PERFORM FRAGMENTED DATA INPUT                    *
*                       = CALL #FRAGINPUT WITH 13 ACTIONS                   *
*                       = SYSTALL A27-13>-(1..13)                           *
*                                                                           *
*              AT SYSTALL A27-13-5> = PERFORM DATA VERIFICATION              *
*                                  = #VERIFDATA WITH 12 ACTIONS             *
*                                  = SYSTALL A27-13-5>-(1..12)              *
*                                                                           *
*  ++ THIS FEATURE IS ONE OF THE TECHNICAL HIGHLIGHTS OF THIS SYSTEM        *
*     (SEE CHAPTER FOUR FOR THE IMPLEMENTATION OF THIS TASK). WE HAVE       *
*     BEEN ABLE TO ACHIEVE AN INPUT RATE OF NEARLY 3000 TRANSACTIONS        *
*     PER HOUR WITH AN EXTREMELY LOW ERROR RATE.                            *
*                                                                           *
*****************************************************************************
```

1.=== ENTER TRANSACTION DATA COLLECTION PROCESS ===$ < 0, 1, 2 >

1.1==DESIGN PROBLEM == --THEN-GO-TO-CLUSTER-> 0
1.1 IF NOT DATA GENERATION PROCESS IS ACCEPTABLE TO THE TEACHERS AND IF
NOT ACCEPTABLE ERROR RATE
A27-2> REVIEW SYSTEM DESIGN OR FACE SYSTEM REJECTION

1.2==ORGANIZATION PROBLEM == --THEN-GO-TO-CLUSTER-> 1
1.2 IF DATA GENERATION PROCESS IS ACCEPTABLE TO THE TEACHERS AND IF
NOT DATA COLLECTION ROUTE IS ORGANIZED AND IF
NOT ACCEPTABLE DELAY IS 15 MINUTES FOR DATA GENERATION
A27-3> ORGANIZE DATA COLLECTION ROUTE TO THE CENTER

CONTINUED ON THE NEXT PAGE

```
1.3==INITIATE DATA COLLECTION                          ==     --THEN-GO-TO-CLUSTER-> 2
     1.3 IF  DATA GENERATION PROCESS IS ACCEPTABLE TO THE TEACHERS  AND IF
             DATA COLLECTION ROUTE IS ORGANIZED  AND IF
             ACCEPTABLE ERROR RATE  AND IF
             ACCEPTABLE DELAY IS 15 MINUTES FOR DATA GENERATION  AND IF
             ACCEPTABLE DELAY IS 30 MINUTES FOR DATA COLLECTION
                A27-1>  NO ACTION - NEXT CLUSTER

2.===DATA COLLECTION                                                ===:        < 2, 2, 0 >

     2.1==TRANSACTION DATA IS NOT READY                ==     --THEN-GO-TO-CLUSTER-> 2
          2.1 IF  NOT  TEACHERS FINISHED DATA GENERATION FOR THE CURRENT LECTURE
                A27-4>  WAIT 10-15 MINUTES


     2.2==TRANSACTION DATA IS READY                    ==     --THEN-GO-TO-CLUSTER-> 2
          2.2 IF  TEACHERS FINISHED DATA GENERATION FOR THE CURRENT LECTURE  AND IF
                  ACCEPTABLE DELAY IS 15 MINUTES FOR DATA GENERATION  AND IF
                  ACCEPTABLE DELAY IS 30 MINUTES FOR DATA COLLECTION  AND IF
                  MORE LECTURES AFTER THIS
                     A27-5>  LET #VP COLLECT #DAT-GEN-KITS FROM CLASSES
                     A27-6>  EXTRACT TRUANT STUDENTS' #ABS-CARDS FROM THE KITS
                     A27-7>  KEEP #DAT-GEN-KITS IN #VP'S OFFICE
                     A27-8>  SEND #ABS-CARDS TO THE COMPUTER CENTER OF THE SCHOOL
                     A27-4>  WAIT 10-15 MINUTES
                     A27-9>  COLLECT ALL #ABS-CARDS FOR THE SCHOOL
                     A27-10>  ACTIVATE TERMINAL, CARD READER AND LOG IN
                     A27-11>  READ IN ALL #ABS-CARDS FOR THE CURRENT LECTURE
                     A27-12>  MAKE A TEMPORARY TRANSACTION DATA FILE ON DISK
                     A27-13>  PERFORM FRAGMENTED DATA INPUT - CALL #FRAGINPUT
                     A27-14>  WAIT FOR THE NEXT LECTURE

     2.3==LAST LECTURES TRANSACTION DATA IS READY      ==     --THEN-GO-TO-CLUSTER-> 0
          2.3 IF  TEACHERS FINISHED DATA GENERATION FOR THE CURRENT LECTURE  AND IF
                  ACCEPTABLE DELAY IS 15 MINUTES FOR DATA GENERATION  AND IF
                  ACCEPTABLE DELAY IS 30 MINUTES FOR DATA COLLECTION  AND IF
                  NOT  MORE LECTURES AFTER THIS
                     A27-5>  LET #VP COLLECT #DAT-GEN-KITS FROM CLASSES
                     A27-6>  EXTRACT TRUANT STUDENTS' #ABS-CARDS FROM THE KITS
                     A27-7>  KEEP #DAT-GEN-KITS IN #VP'S OFFICE
                     A27-8>  SEND #ABS-CARDS TO THE COMPUTER CENTER OF THE SCHOOL
                     A27-4>  WAIT 10-15 MINUTES
                     A27-9>  COLLECT ALL #ABS-CARDS FOR THE SCHOOL
                     A27-10>  ACTIVATE TERMINAL, CARD READER AND LOG IN
                     A27-11>  READ IN ALL #ABS-CARDS FOR THE CURRENT LECTURE
                     A27-12>  MAKE A TEMPORARY TRANSACTION DATA FILE ON DISK
                     A27-13>  PERFORM FRAGMENTED DATA INPUT - CALL #FRAGINPUT
```

16. ABL-TRANSFORM DATACOLLECT;

```
*****************************************************************************
*                                                                           *
*    SYSTALL A(27) = A27>  = TRANSACTION DATA COLLECTION                    *
*                          = #DATACOLLECT WITH 14 ACTIONS                   *
*                          = SYSTALL A27>-(1;.14).                          *
*                                                                           *
*  DATA COLLECTION IS A VERY CRITICAL OPERATION OF THE #ATS SYSTEM.  IF IT DOES*
*  NOT RESPECT THE PREVIOUSLY DETERMINED CRITERIONS SYSTEM REJECTION WILL OCCUR*.
*                                                                           *
*                                                                           *
*     THIS ROUTINE DOES THE FOLLOWING TASKS=                                *
*  1  DATA GENERATION KITS ARE COLLECTED BY THE CORRIDOR SUPERVISOR         *
*     FROM EACH TEACHER TEN MINUTES AFTER THE LECTURE STARTED               *
*  2  ABSENT STUDENTS CARDS ARE EXTRACTED FROM THE POUCHES                  *
*  3  EACH SUPERVISOR SENDS HIS CARDS TO THE TERMINAL SITE                  *
*  4  FRAGMENTED DATA COLLECTION PROCESSING IS DONE THROUGH THE             *
*     CARD READER INTO THE SYSTEM AT EVERY LECTURE   ++                     *
*  5  TRANSACTION DATA IS VERIFIED BY THE SYSTEM                            *
*  6  TRANSACTION DATA IS COLLECTED IN A BUFFER FILE WITH THE PREVIOUS      *
*     LECTURES'  TRANSACTION DATA                                           *
*  7  OUTPUT IS PRINTED WITH OPTIONAL SUMMARY REPORT OF ABSENT              *
*     STUDENT LISTING TO EACH ASSISTANT PRINCIPAL (REPORT A)                *
*                                                                           *
*                                                                           *
*          THE LOGICAL DESCRIPTION OF #DATACOLLECT IS 3 LEVELS DEEP         *
*     AT SYSTALL A27-13> = PERFORM FRAGMENTED DATA INPUT                    *
*                       = CALL #FRAGINPUT WITH 13 ACTIONS                   *
*                       = SYSTALL A27-13>-(1..13)                           *
*                                                                           *
*               AT SYSTALL A27-13-5> = PERFORM DATA VERIFICATION             *
*                                   = #VERIFDATA WITH 12 ACTIONS            *
*                                   = SYSTALL A27-13-5>-(1..12)             *
*                                                                           *
*  ++ THIS FEATURE IS ONE OF THE TECHNICAL HIGHLIGHTS OF THIS SYSTEM        *
*     (SEE CHAPTER FOUR FOR THE IMPLEMENTATION OF THIS TASK). WE HAVE       *
*     BEEN ABLE TO ACHIEVE AN INPUT RATE OF NEARLY 3000 TRANSACTIONS        *
*     PER HOUR WITH AN EXTREMELY LOW ERROR RATE.                            *
*                                                                           *
*****************************************************************************
```

	1	2	3	4	5	6	7	(ALTERNATIVES)	
								CLUSTERS -	
C1	0	1	2	.	.	.	0	=== ENTER TRANSACTION DATA COLLECTION PROCESS	==
C2	.	.	.	2	2	0	0	== DATA COLLECTION	==
								PREDICATES -	
P1	-	N	Y	-	-	-	-	DATA COLLECTION ROUTE IS ORGANIZED	
P2	N	Y	Y	-	-	-	-	DATA GENERATION PROCESS IS ACCEPTABLE TO THE TEACHERS	
P3	N	-	Y	-	-	-	-	ACCEPTABLE ERROR RATE	
P4	-	N	Y	-	Y	Y	-	ACCEPTABLE DELAY IS 15 MINUTES FOR DATA GENERATION	
P5	-	-	Y	-	Y	Y	-	ACCEPTABLE DELAY IS 30 MINUTES FOR DATA COLLECTION	
P6	-	-	-	N	Y	Y	-	TEACHERS FINISHED DATA GENERATION FOR THE CURRENT LECTURE	
P7	-	-	-	-	Y	N	-	MORE LECTURES AFTER THIS	

CONTINUED ON THE NEXT PAGE

								ACTIONS -
A1	.	.	1	.	.	.	.	A27-1> NO ACTION - NEXT CLUSTER
A2	1	.	.	.	.	.	.	A27-2> REVIEW SYSTEM DESIGN OR FACE SYSTEM REJECTION
A3	.	1	.	.	.	.	.	A27-3> ORGANIZE DATA COLLECTION ROUTE TO THE CENTER
A4	.	.	.	1	5	5	.	A27-4> WAIT 10-15 MINUTES
A5	.	.	.	.	1	1	.	A27-5> LET #VP COLLECT #DAT-GEN-KITS FROM CLASSES
A6	.	.	.	.	2	2	.	A27-6> EXTRACT TRUANT STUDENTS' #ABS-CARDS FROM THE KITS
A7	.	.	.	.	3	3	.	A27-7> KEEP #DAT-GEN-KITS IN #VP'S OFFICE
A8	.	.	.	.	4	4	.	A27-8> SEND #ABS-CARDS TO THE COMPUTER CENTER OF THE SCHOOL
A9	.	.	.	.	6	6	.	A27-9> COLLECT ALL #ABS-CARDS FOR THE SCHOOL
A10	.	.	.	.	7	7	.	A27-10> ACTIVATE TERMINAL, CARD READER AND LOG IN
A11	.	.	.	.	8	8	.	A27-11> READ IN ALL #ABS-CARDS FOR THE CURRENT LECTURE
A12	.	.	.	.	9	9	.	A27-12> MAKE A TEMPORARY TRANSACTION DATA FILE ON DISK
A13	.	.	.	.	10	10	.	A27-13> PERFORM FRAGMENTED DATA INPUT - CALL #FRAGINPUT
A14	.	.	.	.	11	.	.	A27-14> WAIT FOR THE NEXT LECTURE

```
11.  ABL-ROUTINE  FRAGINPUT;
```

```
*************************************************************************
*                                                                       *
*         SYSTALL A(27-13) = A27-13> = PERFORM FRAGMENTED TRANSACTION    *
*                                      DATA INPUT ON THE TERMINAL       *
*                                    = #FRAGINPUT WITH 13 ACTIONS       *
*                                    = SYSTALL A27-13>-(1..13)          *
*                                                                       *
*  THIS ROUTINE IS CALLED BY THE DATA COLLECTION PROCEDURE AT SYSTALL A27-13>
*  IT PERFORMS FRAGMENTED INPUT OF THE TRANSACTION DATA.  THIS MEANS THAT
*  ONLY A PART OF THE TOTAL TRANSACTION DATA NECESSARY FOR THE FINAL
*  DATA PROCESSING OF THE DAY IS BEING COLLECTED INTO AN INPUT-FILE
*  AT ANY GIVEN LECTURE.
*                                                                       *
*  ESSENTIALLY THIS ROUTINE PERFORMS THE FOLLOWING=                     *
*                                                                       *
*  1 IT TRANSFERS EACH TRANSACTION DATA RECORD FROM THE DUMMY INPUT FILE
*    TO A VALIDATED TRANSACTION DATA FILE CALLED $DADCR
*  2 CHECKS THE VALIDITY OF EACH TRANSACTION RECORD $TRANS-REC BY WAY
*    OF REFERENCE TO THE DATA-BASE $DB.
*  3 REJECTS ALL NON VALID TRANSACTION DATA WITH AN APPROPRIATE ERROR
*    MESSAGE.
*  4 PRODUCES AN OPTIONAL LISTING OF TRUANT STUDENTS FOR THE CURRECT
*    LECTURE  (REPORT 'A').
*                                                                       *
*     THE LOGICAL DESCRIPTION OF THIS ROUTINE IS 2 LEVELS DEEP
*      AT SYSTALL A27-13-5> IT CALLS TRANSACTION DATA VERIFICATION SUBROUTINE
*                           = #VERIFDATA WITH 12 ACTIONS
*                           = SYSTALL A27-13-5>-(1.12)
*                                                                       *
*          AT THE END RETURN TO $DATACOLLECT                            *
*                                                                       *
*************************************************************************
```

```
   1.== START WITH OUTPUT OPTION SELECTION                    ==>          < 2, 2 >
       1.1==CALL PROCEDURE WITH REPORTING              ==   --THEN-GO-TO-CLUSTER-> 2
       1.2==CALL PROCEDURE WITH NO REPORTING           ==   --THEN-GO-TO-CLUSTER-> 2
   2.== TRANSACTION DATA VERIFICATION                         ==>          < 0, 0, 2 >
       2.1==ERROR -NO INITIALIZATION DONE-             ==   --THEN-GO-TO-CLUSTER-> 0
       2.2==ERROR -NO DATA-                            ==   --THEN-GO-TO-CLUSTER-> 0
       2.3==COLLECT VALID TRANSACTION                  ==   --THEN-GO-TO-CLUSTER-> 2
   3.== DATA COLLECTION SUMMARY REPORTING OPT I OR 2          ==>          < 0, 0 >
       3.1==WRITE TRUANCY SUMMARY                      ==   --THEN-GO-TO-CLUSTER-> 0
       3.2==WRITE ONLY MESSAGES                        ==   --THEN-GO-TO-CLUSTER-> 0
```

```
*************************************************************************
*                                                                       *
*        SYSTALL A(27-13) = A27-13> = PERFORM FRAGMENTED TRANSACTION     *
*                                     DATA INPUT ON THE TERMINAL        *
*                                   = #FRAGINPUT WITH 13 ACTIONS        *
*                                   = SYSTALL A27-13>-(1..13)           *
*                                                                       *
*  THIS ROUTINE IS CALLED BY THE DATA COLLECTION PROCEDURE AT SYSTALL A27-13>
*  IT PERFORMS FRAGMENTED INPUT OF THE TRANSACTION DATA.  THIS MEANS THAT
*  ONLY A PART OF THE TOTAL TRANSACTION DATA NECESSARY FOR THE FINAL
*  DATA PROCESSING OF THE DAY IS BEING COLLECTED INTO AN INPUT-FILE
*  AT ANY GIVEN LECTURE.                                                *
*                                                                       *
*  ESSENTIALLY THIS ROUTINE PERFORMS THE FOLLOWING=                     *
*                                                                       *
*  1 IT TRANSFERS EACH TRANSACTION DATA RECORD FROM THE DUMMY INPUT FILE
*    TO A VALIDATED TRANSACTION DATA FILE CALLED #DADCR                 *
*  2 CHECKS THE VALIDITY OF EACH TRANSACTION RECORD #TRANS-REC BY WAY
*    OF REFERENCE TO THE DATA-BASE #DB.                                 *
*  3 REJECTS ALL NON VALID TRANSACTION DATA WITH AN APPROPRIATE ERROR
*    MESSAGE.                                                           *
*  4 PRODUCES AN OPTIONAL LISTING OF TRUANT STUDENTS FOR THE CURRECT
*    LECTURE  (REPORT "A").                                             *
*                                                                       *
*     THE LOGICAL DESCRIPTION OF THIS ROUTINE IS 2 LEVELS DEEP          *
*      AT SYSTALL A27-13-5> IT CALLS TRANSACTION DATA VERIFICATION SUBROUTINE
*                             = #VERIFDATA WITH 12 ACTIONS              *
*                             = SYSTALL A27-13-5>-(1,12)                *
*                                                                       *
*          AT THE END RETURN TO #DATACOLLECT                            *
*                                                                       *
*************************************************************************
```

```
  1.== START WITH OUTPUT OPTION SELECTION                          ==#          < 2, 2 >

      1.1=CALL PROCEDURE WITH REPORTING                      ==    —THEN-GO-TO-CLUSTER-> 2
          1.1 IF  START  AND IF
                  OPTION WITH LISTING  AND IF
                  NOT  OPTION WITH NO LISTINGS

      1.2=CALL PROCEDURE WITH NO REPORTING                   ==    —THEN-GO-TO-CLUSTER-> 2
          1.2 IF  START  AND IF
                  NOT  OPTION WITH LISTING  AND IF
                  OPTION WITH NO LISTINGS
  2.== TRANSACTION DATA VERIFICATION                               ==#          < 0, 0, 2 >

      2.1=ERROR -NO INITIALIZATION DONE-                     ==    —THEN-GO-TO-CLUSTER-> 0
          2.1 IF  NOT  EXECUTION DAY IS INITIALIZED

      2.2=ERROR -NO DATA-                                    ==    —THEN-GO-TO-CLUSTER-> 0
          2.2 IF  EXECUTION DAY IS INITIALIZED  AND IF
                  NOT  INPUT TRANSACTION DATA IS COPIED FROM CARD TO DISK

      2.3=COLLECT VALID TRANSACTION                          ==    —THEN-GO-TO-CLUSTER-> 2
          2.3 IF  EXECUTION DAY IS INITIALIZED  AND IF
                  INPUT TRANSACTION DATA IS COPIED FROM CARD TO DISK  AND IF
                  NOT  TRANSACTION DATA IS VERIFIED
```

CONTINUED ON THE NEXT PAGE

```
3.=== DATA COLLECTION SUMMARY REPORTING OPT I OR 2                    ===$          < 0, 0 >

    3.1==WRITE TRUANCY SUMMARY                                  ==    —THEN-GO-TO-CLUSTER-> 0
        3.1 IF  EXECUTION DAY IS INITIALIZED  AND IF
                INPUT TRANSACTION DATA IS COPIED FROM CARD TO DISK  AND IF
                TRANSACTION DATA IS VERIFIED  AND IF
                OPTION WITH LISTING

    3.2==WRITE ONLY MESSAGES                                    ==    —THEN-GO-TO-CLUSTER-> 0
        3.2 IF  EXECUTION DAY IS INITIALIZED  AND IF
                INPUT TRANSACTION DATA IS COPIED FROM CARD TO DISK  AND IF
                TRANSACTION DATA IS VERIFIED  AND IF
                OPTION WITH NO LISTING
```

11. ABL-ROUTINE FRAGINPUT;

```
*************************************************************************
*                                                                       *
*        SYSTALL A(27-13) = A27-13> = PERFORM FRAGMENTED TRANSACTION     *
*                                     DATA INPUT ON THE TERMINAL        *
*                                   = #FRAGINPUT WITH 13 ACTIONS        *
*                                   = SYSTALL A27-13>-(1..13)           *
*                                                                       *
*  THIS ROUTINE IS CALLED BY THE DATA COLLECTION PROCEDURE AT SYSTALL A27-13>
*  IT PERFORMS FRAGMENTED INPUT OF THE TRANSACTION DATA.  THIS MEANS THAT
*  ONLY A PART OF THE TOTAL TRANSACTION DATA NECESSARY FOR THE FINAL
*  DATA PROCESSING OF THE DAY IS BEING COLLECTED INTO AN INPUT-FILE
*  AT ANY GIVEN LECTURE.
*
*  ESSENTIALLY THIS ROUTINE PERFORMS THE FOLLOWING=
*
*  1 IT TRANSFERS EACH TRANSACTION DATA RECORD FROM THE DUMMY INPUT FILE
*    TO A VALIDATED TRANSACTION DATA FILE CALLED #DABCR
*  2 CHECKS THE VALIDITY OF EACH TRANSACTION RECORD #TRANS-REC BY WAY
*    OF REFERENCE TO THE DATA-BASE #DB.
*  3 REJECTS ALL NOW VALID TRANSACTION DATA WITH AN APPROPRIATE ERROR
*    MESSAGE.
*  4 PRODUCES AN OPTIONAL LISTING OF TRUANT STUDENTS FOR THE CURRECT
*    LECTURE  (REPORT 'A').
*
*     THE LOGICAL DESCRIPTION OF THIS ROUTINE IS 2 LEVELS DEEP
*      AT SYSTALL A27-13-5> IT CALLS TRANSACTION DATA VERIFICATION SUBROUTINE
*                           = #VERIFDATA WITH 12 ACTIONS
*                           = SYSTALL A27-13-5>-(1.12)
*
*          AT THE END RETURN TO #DATACOLLECT
*                                                                       *
*************************************************************************
```

1.== START WITH OUTPUT OPTION SELECTION ===# < 2, 2 >

1.1==CALL PROCEDURE WITH REPORTING == --THEN-GO-TO-CLUSTER-> 2
1.1 IF START AND IF
OPTION WITH LISTING AND IF
NOT OPTION WITH NO LISTING
A27-13-1> ENTER COMMAND = CALL,ABKPLST #

1.2==CALL PROCEDURE WITH NO REPORTING == --THEN-GO-TO-CLUSTER-> 2
1.2 IF START AND IF
NOT OPTION WITH LISTING AND IF
OPTION WITH NO LISTING
A27-13-2> ENTER COMMAND = CALL,ABKPNLT #

CONTINUED ON THE NEXT PAGE

```
2.=== TRANSACTION DATA VERIFICATION                                    ===$          < 0, 0, 2 >

    2.1==ERROR -NO INITIALIZATION DONE-                          ==    --THEN-GO-TO-CLUSTER-> 0
        2.1 IF  NOT  EXECUTION DAY IS INITIALIZED
                A27-13-6>  WRITE ERROR MESSAGE


    2.2==ERROR -NO DATA-                                         ==    --THEN-GO-TO-CLUSTER-> 0
        2.2 IF  EXECUTION DAY IS INITIALIZED  AND IF
            NOT  INPUT TRANSACTION DATA IS COPIED FROM CARD TO DISK
                A27-13-6>  WRITE ERROR MESSAGE
                A27-13-11>  INTERRUPT EXECUTION
                A27-13-13>  COPY, TRANSACTION DATA FROM CARDS TO A DUMMY DISKFILE


    2.3==COLLECT VALID TRANSACTION                               ==    --THEN-GO-TO-CLUSTER-> 2
        2.3 IF  EXECUTION DAY IS INITIALIZED  AND IF
            INPUT TRANSACTION DATA IS COPIED FROM CARD TO DISK  AND IF
            NOT  TRANSACTION DATA IS VERIFIED
                A27-13-3>  OPEN STUDENT MASTER FILE READ MODE
                A27-13-4>  OPEN COURSE MASTER FILE READ MODE
                A27-13-5>  PERFORM INPUT TRANSACTION DATA VERIFICATION ,CALL $VERIFDATA

3.=== DATA COLLECTION SUMMARY REPORTING OPT I OR 2                      ===$          < 0, 0 >

    3.1==WRITE TRUANCY SUMMARY                                   ==    --THEN-GO-TO-CLUSTER-> 0
        3.1 IF  EXECUTION DAY IS INITIALIZED  AND IF
            INPUT TRANSACTION DATA IS COPIED FROM CARD TO DISK  AND IF
            TRANSACTION DATA IS VERIFIED  AND IF
            OPTION WITH LISTING
                A27-13-7>  SORT STUDENT OUTPUT RECORDS ON $VP. KEY
                A27-13-8>  SORT STUDENT OUTPUT RECORDS ON STUDENT NAME
                A27-13-9>  WRITE MORNING TRUANCY REPORT 'A' FROM STUDENT OUTPUT RECORDS
                A27-13-10>  WRITE ERROR SUMMARY FROM INVALID TRANSACTIONS AND ERROR MESSAGES
                A27-13-12>  APPEND VALID $TRANS-REC TO PREVIOUSLY VALIDATED $TRANS-REC ($DADCR)


    3.2==WRITE ONLY MESSAGES                                     ==    --THEN-GO-TO-CLUSTER-> 0
        3.2 IF  EXECUTION DAY IS INITIALIZED  AND IF
            INPUT TRANSACTION DATA IS COPIED FROM CARD TO DISK  AND IF
            TRANSACTION DATA IS VERIFIED  AND IF
            OPTION WITH NO LISTING
                A27-13-10>  WRITE ERROR SUMMARY FROM INVALID TRANSACTIONS AND ERROR MESSAGES
                A27-13-12>  APPEND VALID $TRANS-REC TO PREVIOUSLY VALIDATED $TRANS-REC ($DADCR)
```

11. ABL-TRANSFORM FRAGINPUT;


```
*****************************************************************************
*                                                                           *
*         SYSTALL A(27-13) = A27-13> = PERFORM FRAGMENTED TRANSACTION        *
*                                      DATA INPUT ON THE TERMINAL           *
*                                    = #FRAGINPUT WITH 13 ACTIONS           *
*                                    = SYSTALL A27-13>-(1..13)              *
*                                                                           *
*  THIS ROUTINE IS CALLED BY THE DATA COLLECTION PROCEDURE AT SYSTALL A27-13> *
*  IT PERFORMS FRAGMENTED INPUT OF THE TRANSACTION DATA.  THIS MEANS THAT    *
*  ONLY A PART OF THE TOTAL TRANSACTION DATA NECESSARY FOR THE FINAL         *
*  DATA PROCESSING OF THE DAY IS BEING COLLECTED INTO AN INPUT-FILE          *
*  AT ANY GIVEN LECTURE.                                                    *
*                                                                           *
*  ESSENTIALLY THIS ROUTINE PERFORMS THE FOLLOWING=                         *
*                                                                           *
*  1 IT TRANSFERS EACH TRANSACTION DATA RECORD FROM THE DUMMY INPUT FILE     *
*    TO A VALIDATED TRANSACTION DATA FILE CALLED $DABCR                     *
*  2 CHECKS THE VALIDITY OF EACH TRANSACTION RECORD $TRANS-REC BY WAY       *
*    OF REFERENCE TO THE DATA-BASE $DB.                                     *
*  3 REJECTS ALL NON VALID TRANSACTION DATA WITH AN APPROPRIATE ERROR       *
*    MESSAGE.                                                               *
*  4 PRODUCES AN OPTIONAL LISTING OF TRUANT STUDENTS FOR THE CURRECT        *
*    LECTURE  (REPORT 'A').                                                 *
*                                                                           *
*     THE LOGICAL DESCRIPTION OF THIS ROUTINE IS 2 LEVELS DEEP              *
*      AT SYSTALL A27-13-5> IT CALLS TRANSACTION DATA VERIFICATION SUBROUTINE*
*                             = #VERIFDATA WITH 12 ACTIONS                  *
*                             = SYSTALL A27-13-5>-(1.12)                    *
*                                                                           *
*          AT THE END RETURN TO $DATACOLLECT                                *
*                                                                           *
*****************************************************************************
```

CONTINUED ON THE NEXT PAGE

	1	2	3	4	5	6	7	8	(ALTERNATIVES)
									CLUSTERS -
C1	2	2	.	.	.	.	.	0	=== START WITH OUTPUT OPTION SELECTION ====
C2	.	.	0	0	2	.	.	0	=== TRANSACTION DATA VERIFICATION . ====
C3	.	.	.	.	.	0	0	0	=== DATA COLLECTION SUMMARY REPORTING OPT I OR 2 ====
									PREDICATES -
P1	Y	Y	-	-	-	-	-	-	START
P2	-	-	N	Y	Y	Y	Y	-	EXECUTION DAY IS INITIALIZED
P3	-	-	-	N	Y	Y	Y	-	INPUT TRANSACTION DATA IS COPIED FROM CARD TO DISK
P4	-	-	-	-	N	Y	Y	-	TRANSACTION DATA IS VERIFIED
P5	Y	N	-	-	-	Y	-	-	OPTION WITH LISTING
P6	N	Y	-	-	-	-	Y	-	OPTION WITH NO LISTING
									ACTIONS -
A1	1	.	.	.	.	.	.	.	A27-13-1> ENTER COMMAND = CALL,ABKPLST #
A2	.	1	.	.	.	.	.	.	A27-13-2> ENTER COMMAND = CALL,ABKPMLT #
A3	.	.	.	.	1	.	.	.	A27-13-3> OPEN STUDENT MASTER FILE READ MODE
A4	.	.	.	.	2	.	.	.	A27-13-4> OPEN COURSE MASTER FILE READ MODE
A5	.	.	.	.	3	.	.	.	A27-13-5> PERFORM INPUT TRANSACTION DATA VERIFICATION ,CALL #VERIFDATA
A6	.	.	1	1	.	.	.	.	A27-13-6> WRITE ERROR MESSAGE
A7	.	.	.	.	.	1	.	.	A27-13-7> SORT STUDENT OUTPUT RECORDS ON #VP, KEY
A8	.	.	.	.	.	2	.	.	A27-13-8> SORT STUDENT OUTPUT RECORDS ON STUDENT NAME
A9	.	.	.	.	.	3	.	.	A27-13-9> WRITE MORNING TRUANCY REPORT "A" FROM STUDENT OUTPUT RECORDS
A10	.	.	.	.	.	4	1	.	A27-13-10> WRITE ERROR SUMMARY FROM INVALID TRANSACTIONS AND ERROR MESSAGES
A11	.	.	.	2	.	.	.	.	A27-13-11> INTERRUPT EXECUTION
A12	.	.	.	.	.	5	2	.	A27-13-12> APPEND VALID #TRANS-REC TO PREVIOUSLY VALIDATED #TRANS-REC (#DABCR)
A13	.	.	.	3	.	.	.	.	A27-13-13> COPY, TRANSACTION DATA FROM CARDS TO A DUMMY-DISKFILE

12. ABL-ROUTINE VERIFDATA;

```
*****************************************************************************
*                                                                           *
*     SYSTALL A(27-13-5) = A27-13-5> PERFORM TRANSACTION DATA VERIFICATION   *
*                        = #VERIFDATA WITH 11 ACTIONS                       *
*                        = SYSTALL A27-13-5>-(1..11)                        *
*                                                                           *
* THIS ROUTINE IS CALLED BY #FRAGINPUT AT SYSTALL A27-13-5>.                *
* IT VERIFIES THE TRANSACTION DATA RECORD STRINGS FOR ERRORS                *
* BEFORE IT APPENDS THEM TO THE DAILY VALID TRANSACTION DATA FILE (#DABCR). *
* IT ALSO PRODUCES AN ERROR LISTING FOR THE REJECTED #TRANS-RECS.           *
*                                                                           *
*       AT THE END RETURN TO #FRAGINPUT                                     *
*                                                                           *
*****************************************************************************
```

1.=== ENTER #VERIFDATA ROUTINE ===$ < 2 >

1.1==START VERIFICATION ROUTINE == --THEN-GO-TO-CLUSTER-> 2

2.=== CHECK FOR INPUT ERROR ===$ < 2, 2, 3, 2, 0 >

2.1==GET TRANSACTION RECORD == --THEN-GO-TO-CLUSTER-> 2

2.2==CHECK DATA COMMUNICATION VALIDITY == --THEN-GO-TO-CLUSTER-> 2

2.3==NO DATA COMMUNICATION ERROR DETECTED == --THEN-GO-TO-CLUSTER-> 3

2.4==DATA COMMUNICATION ERROR DETECTED == --THEN-GO-TO-CLUSTER-> 2

2.5==NO MORE TRANSACTION DATA, RETURN == --THEN-GO-TO-CLUSTER-> 0

3.=== TRANSACTION DATA ANALYSIS ===$ < 4, 2 >

3.1==CHECK TRANSACTION RECORD VALIDITY == --THEN-GO-TO-CLUSTER-> 4

3.2==EMPTY TRANSACTION RECORD DETECTED == --THEN-GO-TO-CLUSTER-> 2

4.=== DATA COLLECTION ERROR AND SUMMARY REPORTING ===$ < 2, 2, 2 >

4.1==ERROR -DUPLICATE TRANSACTION RECORD DETECTED- == --THEN-GO-TO-CLUSTER-> 2

4.2==ERROR -STUDENT WHICH DROPPED COURSE REPORTED ABSENT- == --THEN-GO-TO-CLUSTER-> 2

4.3==ERROR -SCHEDULE CONFLICT WITH OTHER COURSE- == --THEN-GO-TO-CLUSTER-> 2

12. ABL-ROUTINE VERIFDATA;


```
*****************************************************************************
*                                                                           *
*       SYSTALL A(27-13-5) = A27-13-5> PERFORM TRANSACTION DATA VERIFICATION *
*                          = #VERIFDATA WITH 11 ACTIONS                     *
*                          = SYSTALL A27-13-5>-(1..11)                      *
*                                                                           *
*  THIS ROUTINE IS CALLED BY #FRAGINPUT AT SYSTALL A27-13-5>.               *
*  IT VERIFIES THE TRANSACTION DATA RECORD STRINGS FOR ERRORS               *
*  BEFORE IT APPENDS THEM TO THE DAILY VALID TRANSACTION DATA FILE (#DAYCR). *
*  IT ALSO PRODUCES AN ERROR LISTING FOR THE REJECTED #TRANS-RECS.          *
*                                                                           *
*          AT THE END RETURN TO #FRAGINPUT                                  *
*                                                                           *
*****************************************************************************
```

```
1.=== ENTER #VERIFDATA ROUTINE                                  ===#        < 2 >
    1.1==START VERIFICATION ROUTINE                          ==   --THEN-GO-TO-CLUSTER-> 2
        1.1P  NO CONDITIONS
2.=== CHECK FOR INPUT ERROR                                     ===#        < 2, 2, 3, 2, 0 >
    2.1==GET TRANSACTION RECORD                              ==   --THEN-GO-TO-CLUSTER-> 2
        2.1 IF  NOT  THERE IS A TRANSACTION RECORD IN BUFFER
    2.2==CHECK DATA COMMUNICATION VALIDITY                   ==   --THEN-GO-TO-CLUSTER-> 2
        2.2 IF  THERE IS A TRANSACTION RECORD IN BUFFER
    2.3==NO DATA COMMUNICATION ERROR DETECTED                ==   --THEN-GO-TO-CLUSTER-> 3
        2.3 IF  THERE IS A-TRANSACTION RECORD IN BUFFER  AND IF
                NOT  DATA TRANSMISSION ERROR IN TRANSACTION RECORD
    2.4==DATA COMMUNICATION ERROR DETECTED                   ==   --THEN-GO-TO-CLUSTER-> 2
        2.4 IF  THERE IS A TRANSACTION RECORD IN BUFFER  AND IF
                DATA TRANSMISSION ERROR IN TRANSACTION RECORD
    2.5==NO MORE TRANSACTION DATA; RETURN                    ==   --THEN-GO-TO-CLUSTER-> 0
        2.5 IF  END OF INPUT-TRANSACTION DATA DUMMY FILE
3.=== TRANSACTION DATA ANALYSIS                                 ===#        < 4, 2 >
    3.1==CHECK TRANSACTION RECORD VALIDITY                   ==   --THEN-GO-TO-CLUSTER-> 4
        3.1 IF  THERE IS A TRANSACTION RECORD IN BUFFER  AND IF
                NOT  DATA TRANSMISSION ERROR IN TRANSACTION RECORD
    3.2==EMPTY TRANSACTION RECORD DETECTED                   ==   --THEN-GO-TO-CLUSTER-> 2
        3.2 IF  #TRANS-REC IS EMPTY LINE
4.=== DATA COLLECTION ERROR AND SUMMARY REPORTING               ===#        < 2,.2, 2 >
    4.1==ERROR -DUPLICATE TRANSACTION RECORD DETECTED-        ==   --THEN-GO-TO-CLUSTER-> 2
        4.1 IF  #TRANS-REC IS DUPLICATE ENTRY
    4.2==ERROR -STUDENT WHICH DROPPED COURSE REPORTED ABSENT- ==   --THEN-GO-TO-CLUSTER-> 2
        4.2 IF  STUDENT IS DELETED FROM CORRESPONDING COURSE
    4.3==ERROR -SCHEDULE CONFLICT WITH OTHER COURSE-          ==   --THEN-GO-TO-CLUSTER-> 2
        4.3 IF  #TRANS-REC COURSE HAS THE WRONG SCHEDULE
```

12. ABL-ROUTINE VERIFDATA;


```
*****************************************************************************
*                                                                           *
*       SYSTALL A(27-13-5) = A27-13-5> PERFORM TRANSACTION DATA VERIFICATION *
*                          = #VERIFDATA WITH 11 ACTIONS                     *
*                          = SYSTALL A27-13-5>-(1,,11)                      *
*                                                                           *
*  THIS ROUTINE IS CALLED BY #FRAGINPUT AT SYSTALL A27-13-5>.               *
*  IT VERIFIES THE TRANSACTION DATA RECORD STRINGS FOR ERRORS               *
*  BEFORE IT APPENDS THEM TO THE DAILY VALID TRANSACTION DATA FILE (#DAYCR). *
*  IT ALSO PRODUCES AN ERROR LISTING FOR THE REJECTED #TRANS-RECS.          *
*                                                                           *
*         AT THE END RETURN TO #FRAGINPUT                                   *
*                                                                           *
*****************************************************************************
```

1.== ENTER #VERIFDATA ROUTINE ==* < 2 >

1.1==START VERIFICATION ROUTINE == --THEN-GO-TO-CLUSTER-> 2

1.1P NO CONDITIONS.

A27-13-5-11> SORT ALL RECORDS IN TRANSACTION FILE.

2.== CHECK FOR INPUT ERROR ==* < 2, 2, 3, 2, 0 >

2.1==GET TRANSACTION RECORD == --THEN-GO-TO-CLUSTER-> 2

2.1 IF NOT THERE IS A TRANSACTION RECORD IN BUFFER

A27-13-5-10> READ NEXT #TRANS-REC INTO BUFFER FROM DUMMY INPUT FILE

2.2==CHECK DATA COMMUNICATION VALIDITY == --THEN-GO-TO-CLUSTER-> 2

2.2 IF THERE IS A TRANSACTION RECORD IN BUFFER

A27-13-5-1> ANALYSE CHECK-SUM OF THE INPUT-TRANSACTION STRING

2.3==NO DATA COMMUNICATION ERROR DETECTED == --THEN-GO-TO-CLUSTER-> 3

2.3 IF THERE IS A TRANSACTION RECORD IN BUFFER AND IF
NOT DATA TRANSMISSION ERROR IN TRANSACTION RECORD

A27-13-5-12> NO ACTION, NEXT CLUSTER

2.4==DATA COMMUNICATION ERROR DETECTED == --THEN-GO-TO-CLUSTER-> 2

2.4 IF THERE IS A TRANSACTION RECORD IN BUFFER AND IF
DATA TRANSMISSION ERROR IN TRANSACTION RECORD

A27-13-5-8> WRITE APPROPRIATE ERROR MESSAGE TO ERROR FILE

A27-13-5-9> COPY #TRANS-REC TO ERROR FILE

A27-13-5-10> READ NEXT #TRANS-REC INTO BUFFER FROM DUMMY INPUT FILE

2.5==NO MORE TRANSACTION DATA, RETURN == --THEN-GO-TO-CLUSTER-> 0

2.5 IF END OF INPUT TRANSACTION DATA DUMMY FILE

A27-13-5-12> NO ACTION, NEXT CLUSTER

CONTINUED ON THE NEXT PAGE

```
3.=== TRANSACTION DATA ANALYSIS                                    ===$          < 4, 2 >

   3.1==CHECK TRANSACTION RECORD VALIDITY                          ==    --THEN-GO-TO-CLUSTER-> 4
        3.1 IF  THERE IS A TRANSACTION RECORD IN BUFFER  AND IF
              NOT  DATA TRANSMISSION ERROR IN TRANSACTION RECORD .
                   A27-13-5-2>  DECOMPOSE $TRANS-REC INTO TRANSACTION ITEMS
                   A27-13-5-3>  RETIEVE CORRESPONDING STUDENT RECORD
                   A27-13-5-4>  RETRIEVE CORRESPONDING COURSE RECORD
                   A27-13-5-5>  ANALYSE TRANSACTION ITEMS FOR VALIDITY

   3.2==EMPTY TRANSACTION RECORD DETECTED                          ==    --THEN-GO-TO-CLUSTER-> 2
        3.2 IF  $TRANS-REC IS EMPTY LINE
                   A27-13-5-8>  WRITE APPROPRIATE ERROR MESSAGE TO ERROR FILE
                   A27-13-5-9>  COPY $TRANS-REC TO ERROR FILE
                   A27-13-5-10>  READ NEXT $TRANS-REC INTO BUFFER FROM DUMMY INPUT FILE

4.=== DATA COLLECTION ERROR AND SUMMARY REPORTING                   ===$          < 2, 2, 2 >

   4.1==ERROR -DUPLICATE TRANSACTION RECORD DETECTED-              ==    --THEN-GO-TO-CLUSTER-> 2
        4.1 IF  $TRANS-REC IS DUPLICATE ENTRY
                   A27-13-5-8>  WRITE APPROPRIATE ERROR MESSAGE TO ERROR FILE
                   A27-13-5-9>  COPY $TRANS-REC TO ERROR FILE
                   A27-13-5-10>  READ NEXT $TRANS-REC INTO BUFFER FROM DUMMY INPUT FILE

   4.2==ERROR -STUDENT WHICH DROPPED COURSE REPORTED ABSENT-       ==    --THEN-GO-TO-CLUSTER-> 2
        4.2 IF  STUDENT IS DELETED FROM CORRESPONDING COURSE
                   A27-13-5-8>  WRITE APPROPRIATE ERROR MESSAGE TO ERROR FILE
                   A27-13-5-9>  COPY $TRANS-REC TO ERROR FILE
                   A27-13-5-10>  READ NEXT $TRANS-REC INTO BUFFER FROM DUMMY INPUT FILE .

   4.3==ERROR -SCHEDULE CONFLICT WITH OTHER COURSE-                ==    --THEN-GO-TO-CLUSTER-> 2
        4.3/IF  $TRANS-REC COURSE HAS THE WRONG SCHEDULE
                   A27-13-5-8>  WRITE APPROPRIATE ERROR MESSAGE TO ERROR FILE
                   A27-13-5-9>  COPY $TRANS-REC TO ERROR FILE
                   A27-13-5-10>  READ NEXT $TRANS-REC INTO BUFFER FROM DUMMY INPUT FILE
```

12. ABL-TRANSFORM VERIFDATA;

```
*****************************************************************************
*                                                                           *
*       SYSTALL A(27-13-5) = A27-13-5> PERFORM TRANSACTION DATA VERIFICATION *
*                          = #VERIFDATA WITH 11 ACTIONS                     *
*                          = SYSTALL A27-13-5>-(1..11)                      *
*                                                                           *
*  THIS ROUTINE IS CALLED BY #FRAGINPUT AT SYSTALL A27-13-5>.               *
*  IT VERIFIES THE TRANSACTION DATA RECORD STRINGS FOR ERRORS               *
*  BEFORE IT APPENDS THEM TO THE DAILY VALID TRANSACTION DATA FILE (#DABCR). *
*  IT ALSO PRODUCES AN ERROR LISTING FOR THE REJECTED #TRANS-RECS.          *
*                                                                           *
*         AT THE END RETURN TO #FRAGINPUT                                   *
*                                                                           *
*****************************************************************************
```

	1	2	3	4	5	6	7	8	9	10	11	12	(ALTERNATIVES)
													CLUSTERS -
C1	2	.	.	.	.	.	.	.	.	.	.	0	=== ENTER #VERIFDATA ROUTINE ===
C2	.	2	2	3	2	0	.	.	.	.	.	0	=== CHECK FOR INPUT ERROR ===
C3	.	.	.	.	.	.	4	2	.	.	.	0	=== TRANSACTION DATA ANALYSIS ===
C4	.	.	.	.	.	.	.	.	2	2	2	0	=== DATA COLLECTION ERROR AND SUMMARY-REPORTING ===
													PREDICATES -
P1	-	N	Y	Y	Y	-	Y	-	-	-	-	-	THERE IS A TRANSACTION RECORD IN BUFFER
P2	-	-	-	N	Y	-	N	-	-	-	-	-	DATA TRANSMISSION ERROR IN TRANSACTION RECORD
P3	-	-	-	-	-	-	-	Y	-	-	-	-	#TRANS-REC IS EMPTY LINE
P4	-	-	-	-	-	-	-	-	Y	-	-	-	#TRANS-REC IS DUPLICATE ENTRY
P5	-	-	-	-	-	-	-	-	-	Y	-	-	STUDENT IS DELETED FROM CORRESPONDING COURSE
P6	-	-	-	-	-	-	-	-	-	-	Y	-	#TRANS-REC COURSE HAS THE WRONG SCHEDULE
P7	-	-	-	-	-	-	-	-	-	-	-	-	#TRANS-REC IS VALID
P8	-	-	-	-	-	Y	-	-	-	-	-	-	END OF INPUT-TRANSACTION DATA DUMMY FILE
													ACTIONS -
A1	.	.	1	.	.	.	.	.	.	.	.	.	A27-13-5-1> ANALYSE CHECK-SUM OF THE INPUT-TRANSACTION STRING
A2	.	.	.	.	.	.	1	.	.	.	.	.	A27-13-5-2> DECOMPOSE #TRANS-REC INTO TRANSACTION ITEMS
A3	.	.	.	.	.	.	2	.	.	.	.	.	A27-13-5-3> RETIEVE CORRESPONDING STUDENT RECORD
A4	.	.	.	.	.	.	3	.	.	.	.	.	A27-13-5-4> RETRIEVE CORRESPONDING COURSE RECORD
A5	.	.	.	.	.	.	4	.	.	.	.	.	A27-13-5-5> ANALYSE TRANSACTION ITEMS FOR VALIDITY
A6	.	.	.	.	.	.	.	.	.	.	.	.	A27-13-5-6> COPY #TRANS-REC TO VALID TRANSACTION FILE (#DABCR)
A7	.	.	.	.	.	.	.	.	.	.	.	.	A27-13-5-7> COPY STUDENT DISPLAY ITEMS ON STUDENT OUTPUT RECORD
A8	.	.	.	.	1	.	.	1	1	1	1	.	A27-13-5-8> WRITE APPROPRIATE ERROR MESSAGE TO ERROR FILE
A9	.	.	.	.	2	.	.	2	2	2	2	.	A27-13-5-9> COPY #TRANS-REC TO ERROR FILE
A10	.	1	.	.	3	.	.	3	3	3	3	.	A27-13-5-10> READ NEXT #TRANS-REC INTO BUFFER FROM DUMMY INPUT FILE
A11	1	.	.	.	.	.	.	.	.	.	.	.	A27-13-5-11> SORT ALL RECORDS IN TRANSACTION FILE
A12	.	.	.	1	.	1	.	.	.	.	.	.	A27-13-5-12> NO ACTION, NEXT CLUSTER

```
13.  ABL-ROUTINE  ABSPROC;
```

```
*************************************************************************
*                                                                       *
*        SYSTALL A(28) = A28> = PERFORM DAILY $ATS DATA PROCESSING.      *
*                                                                       *
*                            = $ABSPROC WITH 25 ACTIONS                 *
*                            = SYSTALL A28>-(1..25)                     *
*                                                                       *
*  THIS PROCESS ANALYSES AND STORES THE COLLECTED TRANSACTION DATA AT THE *
*  END OF THE DAY.  IT DIRECTLY STORES ALL TRUANCY DATA IN THE DATA-BASE  *
*  FOR FURTHER USE. AFTER THIS DATA REDUCTION PHASE THIS PROCEDURE        *
*  WRITES A CONCIZE TRUANCY REPORT (REPORT "B" ) IN ACCORDANCE TO THE     *
*  FORMAT AND DEGREE OF DETAIL PREVIOUSLY SPECIFIED BY A SET OF           *
*  OUTPUT SELECTION PARAMETERS.  ALL OUTPUT IS SPOOLED AND SAVED ON DISK  *
*  UNTIL NEXT MORNING.                                                    *
*                                                                       *
*     THE LOGICAL DESCRIPTION OF THIS PROCEDURE IS 2 LEVELS DEEP         *
*                                                                       *
*          AT SYSTALL A28-11> STORE TRANSACTION DATA IN THE DATA-BASE    *
*                            = CALL $ABSTUP9 WITH 17 ACTIONS            *
*                            = SYSTALL A28-11>-(1..17)                  *
*                                                                       *
*************************************************************************
```

```
1.=== START ABSPROC                                          ===$          < 2 >
    1.1==CALL PROCEDURE                              ==   --THEN-GO-TO-CLUSTER-> 2
2.=== PREPARE ALLFILES                                       ===$          < 3 >
    2.1==PREPARE ALL FILES                           ==   --THEN-GO-TO-CLUSTER-> 3
3.=== FILE UPDATING                                          ===$          < 3, 3, 3, 4 >
    3.1==UPDATE FILES, WITH OUTPUT PARAMETERS MATCH  ==   --THEN-GO-TO-CLUSTER-> 3
    3.2==UPDATE FILES, WITH OUTPUT PARAMETERS DONT MATCH ==   --THEN-GO-TO-CLUSTER-> 3
    3.3==ERROR -DELETED $TUDENT-                     ==   --THEN-GO-TO-CLUSTER-> 3
    3.4==NO MORE UPDATING                            ==   --THEN-GO-TO-CLUSTER-> 4
4.=== REPORTING                                              ===$          < 0 >
    4.1==REPORTING, FINALIZATION, STOP               ==   --THEN-GO-TO-CLUSTER-> 0
```

13. ABL-ROUTINE ABSPROC;


```
*************************************************************************
*                                                                       *
*        SYSTALL A(28) = A28> = PERFORM DAILY $ATS DATA PROCESSING.      *
*                                                                       *
*                            = $ABSPROC WITH 25 ACTIONS                 *
*                            = SYSTALL A28>-(1..25)                     *
*                                                                       *
* THIS PROCESS ANALYSES AND STORES THE COLLECTED TRANSACTION DATA AT THE *
* END OF THE DAY.  IT DIRECTLY STORES ALL TRUANCY DATA IN THE DATA-BASE  *
* FOR FURTHER USE. AFTER THIS DATA REDUCTION PHASE THIS PROCEDURE        *
* WRITES A CONCIZE TRUANCY REPORT (REPORT 'B' ) IN ACCORDANCE TO THE     *
* FORMAT AND DEGREE OF DETAIL PREVIOUSLY SPECIFIED BY A SET OF           *
* OUTPUT SELECTION PARAMETERS.  ALL OUTPUT IS SPOOLED AND SAVED ON DISK  *
* UNTIL NEXT MORNING.                                                   *
*                                                                       *
*    THE LOGICAL DESCRIPTION OF THIS PROCEDURE IS 2 LEVELS DEEP         *
*                                                                       *
*        AT SYSTALL A28-11> STORE TRANSACTION DATA IN THE DATA-BASE     *
*                           = CALL $ABSTUPD WITH 17 ACTIONS             *
*                           = SYSTALL A28-11>-(1..17)                   *
*                                                                       *
*************************************************************************
```

```
1.== START ABSPROC                                              ===$          < 2 >
    1.1==CALL PROCEDURE                                    ==    --THEN-GO-TO-CLUSTER-> 2
        1.1 IF  ALL TRANSACTION DATA IS COLLECTED AND VERIFIED IN ($DABCR)
2.== PREPARE ALLFILES                                           ===$          < 3 >
    2.1==PREPARE ALL FILES                                 ==    --THEN-GO-TO-CLUSTER-> 3
        2.1 IF  NOT  ALL TRANSACTION DATA IS SORTED IN $DABCR
3.== FILE UPDATING                                              ===$          < 3, 3, 3, 4 >
    3.1==UPDATE FILES, WITH OUTPUT PARAMETERS MATCH        ==    --THEN-GO-TO-CLUSTER-> 3
        3.1 IF  ALL TRANSACTION DATA IS SORTED IN $DABCR  AND IF
                NOT  STUDENT CORRESPONDING TO $TRANS-REC IS DELETED FROM SCHOOL  AND IF
                STUDENTS ABSENCES MATCH OUTPUT SELECTION PARAMETERS SET UP  AND IF
                NOT  END OF VALID TRANSACTION FILE ($DABCR)
    3.2==UPDATE FILES, WITH OUTPUT PARAMETERS DONT MATCH   ==    --THEN-GO-TO-CLUSTER-> 3
        3.2 IF  ALL TRANSACTION DATA IS SORTED IN $DABCR  AND IF
                NOT  STUDENT CORRESPONDING TO $TRANS-REC IS DELETED FROM SCHOOL  AND IF
                NOT  STUDENTS ABSENCES MATCH OUTPUT SELECTION PARAMETERS SET UP  AND IF
                NOT  END OF VALID TRANSACTION FILE ($DABCR)
    3.3==ERROR -DELETED STUDENT-                           ==    --THEN-GO-TO-CLUSTER-> 3
        3.3 IF  ALL TRANSACTION DATA IS SORTED IN $DABCR  AND IF
                STUDENT CORRESPONDING TO $TRANS-REC IS DELETED FROM SCHOOL  AND IF
                NOT  END OF VALID TRANSACTION FILE ($DABCR)
    3.4==NO MORE UPDATING                                  ==    --THEN-GO-TO-CLUSTER-> 4
        3.4 IF  END OF VALID TRANSACTION FILE ($DABCR)
4.== REPORTING                                                  ===$          < 0 >
    4.1==REPORTING, FINALIZATION, STOP                     ==    --THEN-GO-TO-CLUSTER-> 0
        4.1P  NO CONDITIONS
```

13. ABL-ROUTINE ABSPROC;


```
*************************************************************************
*                                                                       *
*        SYSTALL A(28) = A28> = PERFORM DAILY $ATS DATA PROCESSING.      *
*                                                                       *
*                             = $ABSPROC WITH 25 ACTIONS                *
*                             = SYSTALL A28>-(1,,25)                    *
*                                                                       *
*  THIS PROCESS ANALYSES AND STORES THE COLLECTED TRANSACTION DATA AT THE *
*  END OF THE DAY.  IT DIRECTLY STORES ALL TRUANCY DATA IN THE DATA-BASE  *
*  FOR FURTHER USE. AFTER THIS DATA REDUCTION PHASE THIS PROCEDURE        *
*  WRITES A CONCIZE TRUANCY REPORT (REPORT 'B' ) IN ACCORDANCE TO THE     *
*  FORMAT AND DEGREE OF DETAIL PREVIOUSLY SPECIFIED BY A SET OF          *
*  OUTPUT SELECTION PARAMETERS.  ALL OUTPUT IS SPOOLED AND SAVED ON DISK *
*  UNTIL NEXT MORNING.                                                   *
*                                                                       *
*    THE LOGICAL DESCRIPTION OF THIS PROCEDURE IS 2 LEVELS DEEP.        *
*                                                                       *
*        AT SYSTALL A28-11> STORE TRANSACTION DATA IN THE DATA-BASE     *
*                           = CALL $ABSTUPD WITH 17 ACTIONS             *
*                           = SYSTALL A28-11>-(1,,17)                   *
*                                                                       *
*************************************************************************
```

1.=== START ABSPROC ===: < 2 >

1.1==CALL PROCEDURE == —THEN-GO-TO-CLUSTER-> 2
1.1 IF ALL TRANSACTION DATA IS COLLECTED AND VERIFIED IN ($DABCR)
A28-1> ENTER COMMAND = CALL,ABSOIR $

2.=== PREPARE ALLFILES ===: < 3 >

2.1==PREPARE ALL FILES == —THEN-GO-TO-CLUSTER-> 3
2.1 IF NOT ALL TRANSACTION DATA IS SORTED IN $DABCR
A28-2> SORT $TRANS-RECS. BY STUDENT ID.NO. (MAJOR KEY)
A28-3> SORT $TRANS-RECS. BY CLASS ID.NO. (MINOR KEY)
A28-4> ELIMINATE DUPLICATE ENTRIES OF $TRANS-RECS.
A28-5> REWIND TRANSACTION FILE ($DABCR)
A28-6> OPEN STUDENT MASTER FILE
A28-7> OPEN COURSE MASTER FILE, CLASS MASTER FILE
A28-8> UPDATE PRODUCTION POINTERS IN $ABS-RECORD
A28-9> READ ALL $TRANS-RECS. FOR ONE STUDENT INTO BUFFER (1-6 RECORDS)

3.=== FILE UPDATING ===: < 3, 3, 3, 4 >

3.1==UPDATE FILES, WITH OUTPUT PARAMETERS MATCH == —THEN-GO-TO-CLUSTER-> 3
3.1 IF ALL TRANSACTION DATA IS SORTED IN $DABCR AND IF
NOT STUDENT CORRESPONDING TO $TRANS-REC IS DELETED FROM SCHOOL AND IF
STUDENTS ABSENCES MATCH OUTPUT SELECTION PARAMETERS SET UP AND IF
NOT END OF VALID TRANSACTION FILE ($DABCR)
A28-10> RETRIEVE STUDENT RECORD
A28-11> PERFORM DATA-BASE TRUANCY STORAGE UPDATE -CALL $ABSTUP$-
A28-12> WRITE STUDENT TRUANCY RELATED DATA TO STUDENT OUTPUT FILE
A28-13> INCREMENT STATISTICS
A28-16> READ ALL $TRANS-RECS INTO BUFFER FOR NEXT STUDENT (1-6 RECORDS)

CONTINUED ON THE NEXT PAGE

```
3.2==UPDATE FILES, WITH OUTPUT PARAMETERS DONT MATCH        ==    --THEN-GO-TO-CLUSTER-> 3
    3.2 IF  ALL TRANSACTION DATA IS SORTED IN #DABCR  AND IF
            NOT  STUDENT CORRESPONDING TO #TRANS-REC IS DELETED FROM SCHOOL  AND IF
            NOT  STUDENTS ABSENCES MATCH OUTPUT SELECTION PARAMETERS SET UP  AND IF
            NOT  END OF VALID TRANSACTION FILE (#DABCR)
                A28-10>  RETRIEVE STUDENT RECORD
                A28-11>  PERFORM DATA-BASE TRUANCY STORAGE UPDATE -CALL #ABSTUP9-
                A28-13>  INCREMENT STATISTICS
                A28-16>  READ ALL #TRANS-RECS INTO BUFFER FOR NEXT STUDENT (1-6 RECORDS)


3.3==ERROR -DELETED DTUDENT-                                ==    --THEN-GO-TO-CLUSTER-> 3
    3.3 IF  ALL TRANSACTION DATA IS SORTED IN #DABCR  AND IF
            STUDENT CORRESPONDING TO #TRANS-REC IS DELETED FROM SCHOOL  AND IF
            NOT  END OF VALID TRANSACTION FILE (#DABCR)
                A28-10>  RETRIEVE STUDENT RECORD
                A28-14>  WRITE ERROR MESSAGE
                A28-15>  DISPLAY STUDENT IN ERROR FILE
                A28-16>  READ ALL #TRANS-RECS INTO BUFFER FOR NEXT STUDENT (1-6 RECORDS)


3.4==NO MORE UPDATING                                       ==    --THEN-GO-TO-CLUSTER-> 4
    3.4 IF  END OF VALID TRANSACTION FILE (#DABCR)
                A28-25>  NO ACTION, NEXT CLUSTER

4.== REPORTING                                                  ==:$          < 0 >

    4.1==REPORTING, FINALIZATION, STOP                      ==    --THEN-GO-TO-CLUSTER-> 0
        4.1P  NO CONDITIONS

                A28-17>  SORT ERROR FILE ON $VP KEY,
                A28-18>  SORT STUDENT OUTPUT FILE ON $VP KEY
                A28-19>  WRITE STUDENT REPORT 'B' FROM STUDENT OUTPUT FILE
                A28-20>  WRITE ERROR REPORT FROM ERROR FILE
                A28-21>  COMPUTE STATISTICS FOR THE DAY
                A28-22>  WRITE STATISTICS SUMMARY REPORT
                A28-23>  STORE STATISTICS
                A28-24>  CLOSE ALL FILES
```

13. ABL-TRANSFORM ABSPROC;

```
SYSTALL A(28) = A28> = PERFORM DAILY ATS DATA PROCESSING.
                    = $ABSPROC WITH 25 ACTIONS
                    = SYSTALL A28>-(1..25)

THIS PROCESS ANALYSES AND STORES THE COLLECTED TRANSACTION DATA AT THE
END OF THE DAY.  IT DIRECTLY STORES ALL TRUANCY DATA IN THE DATA-BASE
FOR FURTHER USE. AFTER THIS DATA REDUCTION PHASE THIS PROCEDURE
WRITES A CONCISE TRUANCY REPORT (REPORT "B" ) IN ACCORDANCE TO THE
FORMAT AND DEGREE OF DETAIL PREVIOUSLY SPECIFIED BY A SET OF
OUTPUT SELECTION PARAMETERS.  ALL OUTPUT IS SPOOLED AND SAVED ON DISK
UNTIL NEXT MORNING.

   THE LOGICAL DESCRIPTION OF THIS PROCEDURE IS 2 LEVELS DEEP

      AT SYSTALL A28-11> STORE TRANSACTION DATA IN THE DATA-BASE
                        = CALL $ABSTUPD WITH 17 ACTIONS
                        = SYSTALL A28-11>-(1..17)
```

CONTINUED ON THE NEXT PAGE

```
     1  2  3  4  5  6  7  8  (ALTERNATIVES)

                                  CLUSTERS -
C1   2  .  .  .  .  .  .  0   === START ABSPROC                                    ===
C2   .  3  .  .  .  .  .  0   === PREPARE ALLFILES                                ===
C3   .  .  3  3  3  4  .  0   === FILE UPDATING                                   ===
C4   .  .  .  .  .  .  0  0   === REPORTING                                       ===

                                  PREDICATES -
P1   Y  -  -  -  -  -  -  -   ALL TRANSACTION DATA IS COLLECTED AND VERIFIED IN (#DABCR)
P2   -  N  Y  Y  Y  -  -  -   ALL TRANSACTION DATA IS SORTED IN #DABCR
P3   -  -  N  N  Y  -  -  -   STUDENT CORRESPONDING TO #TRANS-REC IS DELETED FROM SCHOOL
P4   -  -  Y  N  -  -  -  -   STUDENTS ABSENCES MATCH OUTPUT SELECTION PARAMETERS SET UP
P5   -  -  N  N  N  Y  -  -   END OF VALID TRANSACTION FILE (#DABCR)
P6   -  -  -  -  -  -  -  -   START

                                  ACTIONS -
A1   1  .  .  .  .  .  .  .   A28-1>  ENTER COMMAND = CALL,ABSOIR #
A2   .  1  .  .  .  .  .  .   A28-2>  SORT #TRANS-RECS. BY STUDENT ID.NO. (MAJOR KEY)
A3   .  2  .  .  .  .  .  .   A28-3>  SORT #TRANS-RECS. BY CLASS ID.NO. (MINOR KEY)
A4   .  3  .  .  .  .  .  .   A28-4>  ELIMINATE DUPLICATE ENTRIES OF #TRANS-RECS.
A5   .  4  .  .  .  .  .  .   A28-5>  REWIND TRANSACTION FILE (#DABCR)
A6   .  5  .  .  .  .  .  .   A28-6>  OPEN STUDENT MASTER FILE
A7   .  6  .  .  .  .  .  .   A28-7>  OPEN COURSE MASTER FILE, CLASS MASTER FILE
A8   .  7  .  .  .  .  .  .   A28-8>  UPDATE PRODUCTION POINTERS IN #ABS-RECORD
A9   .  8  .  .  .  .  .  .   A28-9>  READ ALL #TRANS-RECS. FOR ONE STUDENT INTO BUFFER (1-6 RECORDS)
A10  .  .  1  1  1  .  .  .   A28-10>  RETRIEVE STUDENT RECORD
A11  .  .  2  2  .  .  .  .   A28-11>  PERFORM DATA-BASE TRUANCY STORAGE UPDATE -CALL #ABSTUPD-
A12  .  .  3  .  .  .  .  .   A28-12>  WRITE STUDENT TRUANCY RELATED DATA TO STUDENT OUTPUT FILE
A13  .  .  4  3  .  .  .  .   A28-13>  INCREMENT STATISTICS
A14  .  .  .  .  2  .  .  .   A28-14>  WRITE ERROR MESSAGE
A15  .  .  .  .  3  .  .  .   A28-15>  DISPLAY STUDENT IN ERROR FILE
A16  .  .  5  4  4  .  .  .   A28-16>  READ ALL #TRANS-RECS INTO BUFFER FOR NEXT STUDENT (1-6 RECORDS)
A17  .  .  .  .  .  .  1  .   A28-17>  SORT ERROR FILE ON #VP KEY.
A18  .  .  .  .  .  .  2  .   A28-18>  SORT STUDENT OUTPUT FILE ON #VP KEY
A19  .  .  .  .  .  .  3  .   A28-19>  WRITE STUDENT REPORT "B" FROM STUDENT OUTPUT FILE
A20  .  .  .  .  .  .  4  .   A28-20>  WRITE ERROR REPORT FROM ERROR FILE
A21  .  .  .  .  .  .  5  .   A28-21>  COMPUTE STATISTICS FOR THE DAY
A22  .  .  .  .  .  .  6  .   A28-22>  WRITE STATISTICS SUMMARY REPORT
A23  .  .  .  .  .  .  7  .   A28-23>  STORE STATISTICS
A24  .  .  .  .  .  .  8  .   A28-24>  CLOSE ALL FILES
A25  .  .  .  .  .  1  .  .   A28-25>  NO ACTION, NEXT CLUSTER
```

14. ABL-ROUTINE ABSTUPD;

```
$        SYSTALL A(28-11) = A28-11> = STORE TRANSACTIONS IN THE DATA-BASE
$                                  = #ABSTUPD WITH 17 ACTIONS
$                                  = SYSTALL A28-11>-(1..17)
$
$  THIS ROUTINE WILL PERFORM UPDATING ON ALL TRUANCY RELATED VARIABLES
$  IN THE DATA-BASE.
$
$        THE FOLLOWING TRUANCY VARIABLES ARE UPDATED
$     TOTAL-DAYS-ABSENT FOR THE YEAR IN THE #STUD-REC
$     TOTAL-LECTURES-MISSED FOR THE YEAR IN THE #STUD-REC
$     TOTAL-DAYS-ABSENT FOR THE SEMESTER IN THE #STUD-REC
$     TOTAL-LECTURES-MISSED FOR THE SEMESTER IN THE #STUD-REC
$     LAST-DATE-ABSENT-FROM-SCHOOL IN THE #STUD-REC
$     LAST-DATE-ABSENT-FROM-LECTURE-COURSENO. IN THE #STUD-REC AND #CLASS-REC
$     SUBTOTAL-OF-LECTURES-MISSED-IN-COURSENO. FOR THE YEAR
$                              IN THE #STUD-REC AND THE #CLASS-REC
$     SUBTOTAL-OF-LECTURES-MISSED-IN-COURSENO. FOR THE SEMESTER
$                              IN THE #STUD-REC AND THE #CLASS-REC
$     BINARY CALENDAR ABSENCE PATTERN IN THE #STUD-REC
$     STATISTICS FOR THE DAY
$
```

```
  1.=== ENTER #ABSTUPD                                    ===$          < 2 >

      1.1==ENTER ROUTINE                          ==    --THEN-GO-TO-CLUSTER-> 2
  2.=== UPDATE GENERAL TOTALS OF THE #STUD-REC            ===$          < 3 >

      2.1==INCREMENT #STUD-REC  TOTALS            ==    --THEN-GO-TO-CLUSTER-> 3
  3.=== UPDATE COURSE DATA OF THE #STUD-REC AND #CLASS-REC.  ===$       < 3, 3, 0 >

      3.1==UPDATE COURSE RELATED DATA-BASE        ==    --THEN-GO-TO-CLUSTER-> 3

      3.2==ERROR -DELETED STUDENT FROM THIS COURSE ==    --THEN-GO-TO-CLUSTER-> 3

      3.3==STORE BINARY MASK IN BINARY CALENDAR, RETURN ==  --THEN-GO-TO-CLUSTER-> 0
```

```
14.  ABL-ROUTINE  ABSTUPD;
```

```
*************************************************************************
*                                                                       *
*        SYSTALL A(28-11) = A28-11> = STORE TRANSACTIONS IN THE DATA-BASE *
*                                   = #ABSTUPD WITH 17 ACTIONS          *
*                                   = SYSTALL A28-11>-(1..17)           *
*                                                                       *
*  THIS ROUTINE WILL PERFORM UPDATING ON ALL TRUANCY RELATED VARIABLES   *
*  IN THE DATA-BASE.                                                    *
*                                                                       *
*        THE FOLLOWING TRUANCY VARIABLES ARE UPDATED                    *
*     TOTAL-DAYS-ABSENT FOR THE YEAR IN THE #STUD-REC                   *
*     TOTAL-LECTURES-MISSED FOR THE YEAR IN THE #STUD-REC               *
*     TOTAL-DAYS-ABSENT FOR THE SEMESTER IN THE #STUD-REC               *
*     TOTAL-LECTURES-MISSED FOR THE SEMESTER IN THE #STUD-REC           *
*     LAST-DATE-ABSENT-FROM-SCHOOL IN THE #STUD-REC                     *
*     LAST-DATE-ABSENT-FROM-LECTURE-COURSENO. IN THE #STUD-REC AND #CLASS-REC *
*     SUBTOTAL-OF-LECTURES-MISSED-IN-COURSENO. FOR THE YEAR             *
*                                 IN THE #STUD-REC AND THE #CLASS-REC   *
*     SUBTOTAL-OF-LECTURES-MISSED-IN-COURSENO. FOR THE SEMESTER         *
*                                 IN THE #STUD-REC AND THE #CLASS-REC   *
*     BINARY CALENDAR ABSENCE PATTERN IN THE #STUD-REC                  *
*     STATISTICS FOR THE DAY                                            *
*                                                                       *
*************************************************************************
```

```
   1.=== ENTER #ABSTUPD                                    ===#          < 2 >

      1.1==ENTER ROUTINE                                   ==    --THEN-GO-TO-CLUSTER-> 2
         1.1P  NO CONDITIONS

   2.=== UPDATE GENERAL TOTALS OF THE #STUD-REC            ===#          < 3 >

      2.1==INCREMENT #STUD-REC  TOTALS                     ==    --THEN-GO-TO-CLUSTER-> 3
         2.1 IF  BUFFER CONTAINS N TRANSACTIONS FOR A STUDENT (N= 1 TO 6)
   3.=== UPDATE COURSE DATA OF THE #STUD-REC AND #CLASS-REC.  ===#       < 3, 3, 0 >

      3.1==UPDATE COURSE RELATED DATA-BASE                 ==    --THEN-GO-TO-CLUSTER-> 3
         3.1 IF  BUFFER CONTAINS N TRANSACTIONS FOR A STUDENT (N= 1 TO 6)  AND IF
              NOT  STUDENT IS DELETED FROM COURSE CORRESPONDING TO #TRANS-REC.  AND IF
              NOT  ALL #TRANS-RECS ARE ENTERED FOR THIS STUDENT ON HIS #STUD-REC

      3.2==ERROR -DELETED STUDENT FROM THIS COURSE         ==    --THEN-GO-TO-CLUSTER-> 3
         3.2 IF  BUFFER CONTAINS N TRANSACTIONS FOR A STUDENT (N= 1 TO 6)  AND IF
              STUDENT IS DELETED FROM COURSE CORRESPONDING TO #TRANS-REC.  AND IF
              NOT  ALL #TRANS-RECS ARE ENTERED FOR THIS STUDENT ON HIS #STUD-REC

      3.3==STORE BINARY MASK IN BINARY CALENDAR, RETURN    ==    --THEN-GO-TO-CLUSTER-> 0
         3.3 IF  ALL #TRANS-RECS ARE ENTERED FOR THIS STUDENT ON HIS #STUD-REC
```

14. ABL-ROUTINE ABSTUPD;

```
*****************************************************************************
*                                                                           *
*        SYSTALL A(28-11) = A28-11> = STORE TRANSACTIONS IN THE DATA-BASE    *
*                                   = #ABSTUPD WITH 17 ACTIONS              *
*                                   = SYSTALL A28-11>-(1..17)               *
*                                                                           *
*  THIS ROUTINE WILL PERFORM UPDATING ON ALL TRUANCY RELATED VARIABLES      *
*  IN THE DATA-BASE.                                                        *
*                                                                           *
*        THE FOLLOWING TRUANCY VARIABLES ARE UPDATED                        *
*     TOTAL-DAYS-ABSENT FOR THE YEAR IN THE #STUD-REC                       *
*     TOTAL-LECTURES-MISSED FOR THE YEAR IN THE #STUD-REC                   *
*     TOTAL-DAYS-ABSENT FOR THE SEMESTER IN THE #STUD-REC                   *
*     TOTAL-LECTURES-MISSED FOR THE SEMESTER IN THE #STUD-REC               *
*     LAST-DATE-ABSENT-FROM-SCHOOL IN THE #STUD-REC                         *
*     LAST-DATE-ABSENT-FROM-LECTURE-COURSEN#. IN THE #STUD-REC AND #CLASS-REC *
*     SUBTOTAL-OF-LECTURES-MISSED-IN-COURSEN#. FOR THE YEAR                 *
*                                  IN THE #STUD-REC AND THE #CLASS-REC      *
*     SUBTOTAL-OF-LECTURES-MISSED-IN-COURSEN#. FOR THE SEMESTER             *
*                                  IN THE #STUD-REC AND THE #CLASS-REC      *
*     BINARY CALENDAR ABSENCE PATTERN IN THE #STUD-REC                      *
*     STATISTICS FOR THE DAY/                                               *
*                                                                           *
*****************************************************************************

  1.== ENTER #ABSTUPD                                          ==:    < 2 >.

     1.1==ENTER ROUTINE                                  ==    --THEN-GO-TO-CLUSTER-> 2
        1.1P  NO CONDITIONS
              A28-11-17>  NO ACTION; NEXT CLUSTER

  2.== UPDATE GENERAL TOTALS OF THE #STUD-REC                  ==:          < 3 >

     2.1==INCREMENT #STUD-REC  TOTALS                    ==    --THEN-GO-TO-CLUSTER-> 3
        2.1 IF  BUFFER CONTAINS N TRANSACTIONS FOR A STUDENT (N= 1 TO 6)
              A28-11-1>  ENTER DATE ON #STUD-REC FOR LAST-DAY-ABSENT
              A28-11-2>  INCREMENT ON #STUD-REC. ALL TOTAL DAYS ABSENT BY 1
              A28-11-3>  INCREMENT ON #STUD-REC. ALL TOTAL NUMBERS OF LECTURES ABSENT BY N
              A28-11-12>  TAKE NEXT #TRANS-REC IN THE BUFFER

  3.== UPDATE COURSE DATA OF THE #STUD-REC AND #CLASS-REC.     ==:    < 3, 3, 0 >

     3.1==UPDATE COURSE RELATED DATA-BASE                ==    --THEN-GO-TO-CLUSTER-> 3
        3.1 IF  BUFFER CONTAINS N TRANSACTIONS FOR A STUDENT (N= 1 TO 6)  AND IF
           NOT  STUDENT IS DELETED FROM COURSE CORRESPONDING TO #TRANS-REC.  AND IF
           NOT  ALL #TRANS-RECS ARE ENTERED FOR THIS STUDENT ON HIS #STUD-REC
              A28-11-4>  FOR THE COURSE IN #TRANS-REC ENTER DATE FOR #LAST-DAY-ABS-IN-CR IN #STUD-REC
              A28-11-5>  FOR THE COURSE IN THE #TRANS-REC, INCREMENT SUBTOTALS(DAYS AND PERIODS ABSENT) IN #STUD-REC
              A28-11-6>  RETRIEVE COURSE RECORD FOR SCHEDULE FROM THE COURSE MASTER FILE
              A28-11-7>  RETRIEVE CLASS RECORDS FROM THE CLASS MASTER FILE
              A28-11-8>  INCREMENT CLASS RECORD WITH STUDENTS CORRESPONDING ABS-SUBTOTALS-IN-CR
              A28-11-9>  WRITE ON CLASS RECORD STUDENTS LAST-DATE-ABS-IN-CR
              A28-11-10>  REWRITE CLASS RECORD
              A28-11-11>  PREPARE BINARY MASK FOR THE DAYS SCHEDULE PATTERN( ABSENT BY STUDENT
              A28-11-12>  TAKE NEXT #TRANS-REC IN THE BUFFER
```

CONTINUED ON THE NEXT PAGE

```
3.2==ERROR -DELETED STUDENT FROM THIS COURSE                    ==     --THEN-GO-TO-CLUSTER-> 3
     3.2 IF  BUFFER CONTAINS N TRANSACTIONS FOR A STUDENT (N= 1 TO 6)  AND IF
         STUDENT IS DELETED FROM COURSE CORRESPONDING TO #TRANS-REC.  AND IF
         NOT  ALL #TRANS-RECS ARE ENTERED FOR THIS STUDENT ON HIS #STUB-REC
             A28-11-13> WRITE ERROR MESSAGE
             A28-11-14> WRITE STUDENTS NAME AND COURSE ON ERROR FILE
             A28-11-12> TAKE NEXT #TRANS-REC IN THE BUFFER .


3.3==STORE BINARY MASK IN BINARY CALENDAR, RETURN               ==     --THEN-GO-TO-CLUSTER-> 0
     3.3 IF  ALL #TRANS-RECS ARE ENTERED FOR THIS STUDENT ON HIS #STUB-REC
             A28-11-15> STORE BINARY MASK (DAILY ABSENCE PATTERN) IN THE #STUB-REC.
             A28-11-16> REWRITE UPDATED #STUB-REC. IN STUDENT MASTER FILE
```

14. ABL-TRANSFORM ABSTUPD;

```
*****************************************************************
*                                                               *
*       SYSTALL A(28-11) = A28-11> = STORE TRANSACTIONS IN THE DATA-BASE
*                                  = $ABSTUPD WITH 17 ACTIONS   *
*                                  = SYSTALL A28-11>-(1..17)    *
*                                                               *
*  THIS ROUTINE WILL PERFORM UPDATING ON ALL TRUANCY RELATED VARIABLES
*  IN THE DATA-BASE.                                            *
*                                                               *
*        THE FOLLOWING TRUANCY VARIABLES ARE UPDATED            *
*     TOTAL-DAYS-ABSENT FOR THE YEAR IN THE $STUD-REC           *
*     TOTAL-LECTURES-MISSED FOR THE YEAR IN THE $STUD-REC       *
*     TOTAL-DAYS-ABSENT FOR THE SEMESTER IN THE $STUD-REC       *
*     TOTAL-LECTURES-MISSED FOR THE SEMESTER IN THE $STUD-REC   *
*     LAST-DATE-ABSENT-FROM-SCHOOL IN THE $STUD-REC             *
*     LAST-DATE-ABSENT-FROM-LECTURE-COURSENO. IN THE $STUD-REC AND $CLASS-REC
*     SUBTOTAL-OF-LECTURES-MISSED-IN-COURSENO. FOR THE YEAR     *
*                                  IN THE $STUD-REC AND THE $CLASS-REC
*     SUBTOTAL-OF-LECTURES-MISSED-IN-COURSENO. FOR THE SEMESTER *
*                                  IN THE $STUD-REC AND THE $CLASS-REC
*     BINARY CALENDAR ABSENCE PATTERN IN THE $STUD-REC          *
*     STATISTICS FOR THE DAY                                    *
*                                                               *
*****************************************************************
```

```
      1  2  3  4  5  6   (ALTERNATIVES)

                            CLUSTERS -
C1    2  .  .  .  .  0    == ENTER $ABSTUPD                                  ==
C2    .  3  .  .  .  0    == UPDATE GENERAL TOTALS OF THE $STUD-REC          ==
C3    .  .  3  3  0  0    == UPDATE COURSE DATA OF THE $STUD-REC AND $CLASS-REC.  ==

                            PREDICATES -
P1    -  Y  Y  Y  -  -     BUFFER CONTAINS N TRANSACTIONS FOR A STUDENT (N= 1 TO 6)
P2    -  -  N  Y  -  -     STUDENT IS DELETED FROM COURSE CORRESPONDING TO $TRANS-REC.
P3    -  -  N  N  Y  -     ALL $TRANS-RECS ARE ENTERED FOR THIS STUDENT ON HIS $STUD-REC

                            ACTIONS -
A1    .  1  .  .  .  .     A28-11-1>  ENTER DATE ON $STUD-REC FOR LAST-DATE-ABSENT
A2    .  2  .  .  .  .     A28-11-2>  INCREMENT ON $STUD-REC. ALL TOTAL DAYS ABSENT BY 1
A3    .  3  .  .  .  .     A28-11-3>  INCREMENT ON $STUD-REC. ALL TOTAL NUMBERS OF LECTURES MISSED BY N
A4    .  .  1  .  .  .     A28-11-4>  FOR THE COURSE IN $TRANS-REC ENTER DATE FOR $LAST-DAY-ABS-IN-CR IN $STUD-REC
A5    .  .  2  .  .  .  A28-11-5 FOR THE COURSE IN THE $TRANS-REC. INCREMENT SUBTOTALS(DAYS AND PERIODS ABSENT) IN $STUD-REC
A6    .  .  3  .  .  .     A28-11-6>  RETRIEVE COURSE RECORD FOR SCHEDULE FROM THE COURSE MASTER FILE
A7    .  .  4  .  .  .     A28-11-7>  RETRIEVE CLASS RECORDS FROM THE CLASS MASTER FILE
A8    .  .  5  .  .  .     A28-11-8>  INCREMENT CLASS RECORD WITH STUDENTS CORRESPONDING ABS-SUBTOTALS-IN-CR
A9    .  .  6  .  .  .     A28-11-9>  WRITE ON CLASS RECORD STUDENTS LAST-DATE-ABS-IN-CR
A10   .  .  7  .  .  .     A28-11-10>  REWRITE CLASS RECORD
A11   .  .  8  .  .  .     A28-11-11>  PREPARE BINARY MASK FOR THE DAYS SCHEDULE PATTERN( ABSENT BY STUDENT-)
A12   .  4  9  3  .  .     A28-11-12>  TAKE NEXT $TRANS-REC IN THE BUFFER
A13   .  .  .  1  .  .     A28-11-13>  WRITE ERROR MESSAGE
A14   .  .  .  2  .  .     A28-11-14>  WRITE STUDENTS NAME AND COURSE ON ERROR FILE
A15   .  .  .  .  1  .     A28-11-15>  STORE BINARY MASK (DAILY ABSENCE PATTERN) IN THE $STUD-REC.
A16   .  .  .  .  2  .     A28-11-16>  REWRITE UPDATED $STUD-REC. IN STUDENT MASTER FILE
A17   1  .  .  .  .  .     A28-11-17>  NO ACTION, NEXT CLUSTER
```

17. ABL-ROUTINE DISTRIBUTION;

```
SYSTALL A(29) = A29> = INFORMATION AND DATA DISTRIBUTION
                     = #DISTRIBUTION WITH 26 ACTIONS.
                     = SYSTALL A29>-(1..26)

INFORMATION AND DATA REDISTRIBUTION IS A COMPLEX TASK IN THIS SYSTEM.
DATA COLLECTION IS A FRAGMENTED PROCEDURE WHICH CONTINUES THROUGHOUT THE
DAY AT EVERY LECTURE.  IN ORDER TO RESPECT THE CRITERIONS ESTABLISHED
FOR THE INFORMATION TO REACH ITS TARGET USER AND IN ORDER TO PREVENT
A QUEW FORMATION OF THE DATA TO BE DISTRIBUTED AT THE END OF THE DAY,
WE HAVE TO BEGIN TO SEND BACK WHATEVER DATA IS AVAILABLE EVEN
BEFORE THE FINAL DATA PROCESSING TAKES PLACE. THIS ALSO HAS TO BE
ACHIEVED IN A CRITICAL TIME DELAY.

THIS REGULAR REDISTRIBUTION PROCESS HAS ALSO TO ACCOMODATE A FLOW
OF VARIOUS KIND OF INFORMATION REQUESTS ABOUT THE STORED TRUANCY DATA
IN THE DATA-BASE.
```

```
1.=== REGULAR DAILY DATA DISTRIBUTION                          ===# < 1, 1, 1, 2 >
    1.1==DAILY MORNING DISTRIBUTION  ==  --THEN-GO-TO-CLUSTER-> 1
    1.2== #ABS-CARD REDISTRIBUTION AFTER EACH USE  ==  --THEN-GO-TO-CLUSTER-> 1
    1.3==FINAL INFORMATION DISTRIBUTION  ==  --THEN-GO-TO-CLUSTER-> 1
    1.4==OCCASIONAL TRUANCY INFORMATION REQUESTED  ==  --THEN-GO-TO-CLUSTER-> 2
2.=== OCCASIONAL IRREGULAR INFORMATION DISTRIBUTION             ===# < 1, 1, 1, 1, 1, 0, 1 >
    2.1==INDIVIDUAL REMEDIAL CONSULTATION  ==  --THEN-GO-TO-CLUSTER-> 1
    2.2==PLANNED FAMILY CONSULTATION  ==  --THEN-GO-TO-CLUSTER-> 1
    2.3==UNPLANNED PARENTAL VISIT  ==  --THEN-GO-TO-CLUSTER-> 1
    2.4==TEACHER INFORMATION  ==  --THEN-GO-TO-CLUSTER-> 1
    2.5==SCHOOL MANAGEMENT  ==  --THEN-GO-TO-CLUSTER-> 1
    2.6==EVERYTHING IS DISTRIBUTED  ==  --THEN-GO-TO-CLUSTER-> 0
    2.7==PAUSE  ==  --THEN-GO-TO-CLUSTER-> 1
```

```
*****************************************************************************
*                                                                           *
*        SYSTALL A(29) = A29> = INFORMATION AND DATA DISTRIBUTION            *
*                        ------------------------------------               *
*                                                                           *
*                                = $DISTRIBUTION WITH 26 ACTIONS            *
*                                = SYSTALL A29>-(1..26)                     *
*                                                                           *
*  INFORMATION AND DATA REDISTRIBUTION IS A COMPLEX TASK IN THIS SYSTEM.     *
*  DATA COLLECTION IS A FRAGMENTED PROCEDURE WHICH CONTINUES THROUGHOUT THE  *
*  DAY AT EVERY LECTURE.  IN ORDER TO RESPECT THE CRITERIONS ESTABLISHED     *
*  FOR THE INFORMATION TO REACH ITS TARGET USER AND IN ORDER TO PREVENT      *
*  A QUEU FORMATION OF THE DATA TO BE DISTRIBUTED AT THE END OF THE DAY,     *
*  WE HAVE TO BEGIN TO SEND BACK WHATEVER DATA IS AVAILABLE EVEN             *
*  BEFORE THE FINAL DATA PROCESSING TAKES PLACE. THIS ALSO HAS TO BE         *
*  ACHIEVED IN A CRITICAL TIME DELAY.                                        *
*                                                                           *
*  THIS REGULAR REDISTRIBUTION PROCESS HAS ALSO TO ACCOMODATE A FLOW        *
*  OF VARIOUS KIND OF INFORMATION REQUESTS ABOUT THE STORED TRUANCY DATA     *
*  IN THE DATA-BASE.                                                        *
*                                                                           *
*****************************************************************************
```

1.=== REGULAR DAILY DATA DISTRIBUTION ===$ < 1, 1, 1, 2 >

1.1==DAILY MORNING DISTRIBUTION == —THEN-GO-TO-CLUSTER-> 1
1.1 IF OBJECTIVE IS AN EFFICIENT PARENT CONSULTING AND IF
ACCEPTABLE DELAY IS 2 HOURS AND IF
DATA COLLECTION FOR LECTURE 1 IS COMPLETE AND IF
NOT DATA COLLECTION FOR LECTURE 2 OR 3 ....OR 7 IS COMPLETE AND IF
NOT DATA COLLECTION FOR THE DAY IS COMPLETE AND IF
NOT DATA PROCESSING IS EMBED

1.2== $ABS-CARD REDISTRIBUTION AFTER EACH USE == —THEN-GO-TO-CLUSTER-> 1
1.2 IF DATA COLLECTION FOR LECTURE 2 OR 3 .....OR 7 IS COMPLETE AND IF
NOT DATA COLLECTION FOR THE DAY IS COMPLETE AND IF
NOT DATA PROCESSING IS EMBED AND IF
ACCEPTABLE DELAY IS 2 HOURS

1.3==FINAL INFORMATION DISTRIBUTION == —THEN-GO-TO-CLUSTER-> 1
1.3 IF OBJECTIVE IS AN EFFICIENT STUDENT CONSULTING AND IF
ACCEPTABLE DELAY IS 1-2 DAYS AND IF
DATA COLLECTION FOR THE DAY IS COMPLETE AND IF
DATA PROCESSING IS EMBED

1.4==OCCASIONAL TRUANCY INFORMATION REQUESTED == —THEN-GO-TO-CLUSTER-> 2
1.4 IF INFORMATION IS REQUIRED ON AN IRREGULAR BASIS

CONTINUED ON THE NEXT PAGE

2.=== OCCASIONAL IRREGULAR INFORMATION DISTRIBUTION ===$ < 1, 1, 1, 1, 1, 0, 1 >

2.1==INDIVIDUAL REMEDIAL CONSULTATION == --THEN-GO-TO-CLUSTER-> 1
2.1 IF INFORMATION IS REQUIRED ON AN IRREGULAR BASIS AND IF
OBJECTIVE IS AN EFFICIENT STUDENT CONSULTING AND IF
IMMEDIATE INFORMATION ACCES IS POSSIBLE AND IF
ACCEPTABLE DELAY IS 2 HOURS

2.2==PLANNED FAMILY CONSULTATION == --THEN-GO-TO-CLUSTER-> 1
2.2 IF INFORMATION IS REQUIRED ON AN IRREGULAR BASIS AND IF
OBJECTIVE IS AN EFFICIENT STUDENT AND PARENT CONSULTING AND IF
ACCEPTABLE DELAY IS ONE MONTH

2.3==UNPLANNED PARENTAL VISIT == --THEN-GO-TO-CLUSTER-> 1
2.3 IF INFORMATION IS REQUIRED ON AN IRREGULAR BASIS AND IF
OBJECTIVE IS AN EFFICIENT STUDENT AND PARENT CONSULTING AND IF
IMMEDIATE INFORMATION ACCES IS POSSIBLE

2.4==TEACHER INFORMATION == --THEN-GO-TO-CLUSTER-> 1
2.4 IF INFORMATION IS REQUIRED ON AN IRREGULAR BASIS AND IF
OBJECTIVE IS TO PROVIDE EFFICIENT CLASS MANAGEMENT AND IF
IMMEDIATE INFORMATION ACCES IS POSSIBLE AND IF
ACCEPTABLE DELAY IS ONE MONTH

2.5==SCHOOL MANAGEMENT == --THEN-GO-TO-CLUSTER-> 1
2.5 IF INFORMATION IS REQUIRED ON AN IRREGULAR BASIS AND IF
OBJECTIVE IS AN EFFICIENT SCHOOL MANAGEMENT AND IF
ACCEPTABLE DELAY IS 2-3 DAYS

2.6==EVERYTHING IS DISTRIBUTED == --THEN-GO-TO-CLUSTER-> 0
2.6 IF ALL REGULAR DATA IS DISTRIBUTED AND IF
NOT INFORMATION IS REQUIRED ON AN IRREGULAR BASIS

2.7==PAUSE == --THEN-GO-TO-CLUSTER-> 1
2.7 IF NOT ALL REGULAR DATA IS DISTRIBUTED AND IF
NOT INFORMATION IS REQUIRED ON AN IRREGULAR BASIS

```
SYSTALL A(29) = A29> = INFORMATION AND DATA DISTRIBUTION
                     = #DISTRIBUTION WITH 26 ACTIONS
                     = SYSTALL A29>-(1..26)

INFORMATION AND DATA REDISTRIBUTION IS A COMPLEX TASK IN THIS SYSTEM.
DATA COLLECTION IS A FRAGMENTED PROCEDURE WHICH CONTINUES THROUGHOUT THE
DAY AT EVERY LECTURE.  IN ORDER TO RESPECT THE CRITERIONS ESTABLISHED
FOR THE INFORMATION TO REACH ITS TARGET USER AND IN ORDER TO PREVENT
A QUEU FORMATION OF THE DATA TO BE DISTRIBUTED AT THE END OF THE DAY,
WE HAVE TO BEGIN TO SEND BACK WHATEVER DATA IS AVAILABLE EVEN
BEFORE THE FINAL DATA PROCESSING TAKES PLACE. THIS ALSO HAS TO BE
ACHIEVED IN A CRITICAL TIME DELAY.

THIS REGULAR REDISTRIBUTION PROCESS HAS ALSO TO ACCOMODATE A FLOW
OF VARIOUS KIND OF INFORMATION REQUESTS ABOUT THE STORED TRUANCY DATA
IN THE DATA-BASE.
```

1.=== REGULAR DAILY DATA DISTRIBUTION ===# < P, 1, 1, 2 >

1.1=DAILY MORNING DISTRIBUTION == —THEN-GO-TO-CLUSTER-> 1
1.1 IF OBJECTIVE IS AN EFFICIENT PARENT CONSULTING AND IF
ACCEPTABLE DELAY IS 2 HOURS AND IF
DATA COLLECTION FOR LECTURE 1 IS COMPLETE AND IF
NOT DATA COLLECTION FOR LECTURE 2 OR 3 .....OR 7 IS COMPLETE AND IF
NOT DATA COLLECTION FOR THE DAY IS COMPLETE AND IF
NOT DATA PROCESSING IS ENDED
A29-1> AT THE CENTER SEPARATE ALL DATA TO CORRESPONDING #VP
A29-2> -SEPARATE INPUT #ABS-CARD DECK FOR EACH #VP
A29-3> -SEPARATE OUTPUT FOR EACH #VP
A29-4> -SEND BACK EACH #VP HIS #ABS-CARDS
A29-5> -SEND BACK EACH #VP HIS ERROR LISTING
A29-6> -SEND BACK EACH #VP HIS TRUANCY REPORT "A"
A29-9> AT THE #VP'S OFFICE PREPARE NEXT PRODUCTION CYCLE
A29-10> -CHECK ERROR LISTING FOR DAMAGED #ABS-CARDS
A29-11> -TAKE OUT FAULTY CARDS FROM EACH DECK
A29-12> -REPLACE #ABS-CARDS TO THEIR RESPECTIVE #DAT-GEN-KIT FOR REUSE
A29-13> -DISTRIBUTE #DAT-GEN-KITS TO TEACHERS' MAILBOXES
A29-14> PERFORM TRUANCY RELATED HUMAN COMMUNICATION
A29-15> -PARTITION TRUANCY REPORT "A" INTO SMALL SECTIONS
A29-16> HAND OUT EACH SECTION TO PRE-ASSIGNED PERSONNAL
A29-17> PHONE PARENTS OF TRUANT STUDENTS
A29-20> -RELATE INFORMATION ON HAND

CONTINUED ON THE NEXT PAGE

1.2== #ABS-CARD REDISTRIBUTION AFTER EACH USE == —THEN-GO-TO-CLUSTER-> 1
1.2 IF DATA COLLECTION FOR LECTURE 2 OR 3 ....,OR 7 IS COMPLETE AND IF
NOT DATA COLLECTION FOR THE DAY IS COMPLETE AND IF
NOT DATA PROCESSING IS ENDED AND IF
ACCEPTABLE DELAY IS 2 HOURS
A29-1> AT THE CENTER SEPARATE ALL DATA TO CORRESPONDING #VP
A29-2> -SEPARATE INPUT #ABS-CARD DECK FOR EACH #VP
A29-3> -SEPARATE OUTPUT FOR EACH #VP
A29-4> -SEND BACK EACH #VP HIS #ABS-CARDS
A29-5> -SEND BACK EACH #VP HIS ERROR LISTING
A29-9> AT THE #VP'S OFFICE PREPARE NEXT PRODUCTION CYCLE
A29-10> -CHECK ERROR LISTING FOR DAMAGED #ABS-CARDS
A29-11> -TAKE OUT FAULTY CARDS FROM EACH DECK
A29-12> -REPLACE #ABS-CARDS TO THEIR RESPECTIVE #DAT-GEN-KIT FOR REUSE
A29-13> -DISTRIBUTE #DAT-GEN-KITS TO TEACHERS' MAILBOXES

1.3==FINAL INFORMATION DISTRIBUTION == —THEN-GO-TO-CLUSTER-> 1
1.3 IF OBJECTIVE IS AN EFFICIENT STUDENT CONSULTING AND IF
ACCEPTABLE DELAY IS 1-2 DAYS AND IF
DATA COLLECTION FOR THE DAY IS COMPLETE AND IF
DATA PROCESSING IS ENDED
A29-1> AT THE CENTER SEPARATE ALL DATA TO CORRESPONDING #VP
A29-3> -SEPARATE OUTPUT FOR EACH #VP
A29-7> -SEND BACK EACH #VP HIS TRUANCY REPORT "B"
A29-8> -SEND PRINCIPAL TRUANCY STATISTICS REPORT
A29-14> PERFORM TRUANCY RELATED HUMAN COMMUNICATION
A29-18> -MEET TRUANT STUDENT
A29-20> -RELATE INFORMATION ON HAND

1.4==OCCASIONAL TRUANCY INFORMATION REQUESTED == —THEN-GO-TO-CLUSTER-> 2
1.4 IF INFORMATION IS REQUIRED ON AN IRREGULAR BASIS
A29-25> NO ACTION - NEXT CLUSTER

2.=== OCCASIONAL IRREGULAR INFORMATION DISTRIBUTION ===# < 1, 1, 1, 1, 1, 0, 1 >

2.1==INDIVIDUAL REMEDIAL CONSULTATION == —THEN-GO-TO-CLUSTER-> 1
2.1 IF INFORMATION IS REQUIRED ON AN IRREGULAR BASIS AND IF
OBJECTIVE IS AN EFFICIENT STUDENT CONSULTING AND IF
IMMEDIATE INFORMATION ACCES IS POSSIBLE AND IF
ACCEPTABLE DELAY IS 2 HOURS
A29-22> -QUERRY #DB. FOR RETROACTIVE STUDENT TRUANCY INFORMATION
A29-18> -MEET TRUANT STUDENT
A29-20> -RELATE INFORMATION ON HAND

2.2==PLANNED FAMILY CONSULTATION == —THEN-GO-TO-CLUSTER-> 1
2.2 IF INFORMATION IS REQUIRED ON AN IRREGULAR BASIS AND IF
OBJECTIVE IS AN EFFICIENT STUDENT AND PARENT CONSULTING AND IF
ACCEPTABLE DELAY IS ONE MONTH
A29-22> -QUERRY #DB. FOR RETROACTIVE STUDENT TRUANCY INFORMATION
A29-18> -MEET TRUANT STUDENT
A29-19> -MEET PARENTS OF TRUANT STUDENT
A29-20> -RELATE INFORMATION ON HAND

CONTINUED ON THE NEXT PAGE

```
2.3==UNPLANNED PARENTAL VISIT                    ==    "—THEN-GO-TO-CLUSTER-> 1
    2.3 IF  INFORMATION IS REQUIRED ON AN IRREGULAR BASIS  AND IF
        OBJECTIVE IS AN EFFICIENT STUDENT AND PARENT CONSULTING  AND IF
        IMMEDIATE INFORMATION ACCES IS POSSIBLE
            A29-22>  -QUERRY #DB. FOR RETROACTIVE STUDENT TRUANCY INFORMATION
            A29-19>  -MEET PARENTS OF TRUANT STUDENT
            A29-21>  -MEET STAFF


2.4==TEACHER INFORMATION                         ==    —THEN-GO-TO-CLUSTER-> 1
    2.4 IF  INFORMATION IS REQUIRED ON AN IRREGULAR BASIS  AND IF
        OBJECTIVE IS TO PROVIDE EFFICIENT CLASS MANAGEMENT  AND IF
        IMMEDIATE INFORMATION ACCES IS POSSIBLE  AND IF
        ACCEPTABLE DELAY IS ONE MONTH
            A29-23>  -QUERRY #DB. FOR RETROACTIVE CLASS TRUANCY INFORMATION
            A29-21>  -MEET STAFF
            A29-20>  -RELATE INFORMATION ON HAND


2.5==SCHOOL MANAGEMENT                           ==    —THEN-GO-TO-CLUSTER-> 1
    2.5 IF  INFORMATION IS REQUIRED ON AN IRREGULAR BASIS  AND IF
        OBJECTIVE IS AN EFFICIENT SCHOOL MANAGEMENT  AND IF
        ACCEPTABLE DELAY IS 2-3 DAYS
            A29-24>  -QUERRY #DB. FOR OVERALL TRUANCY STATISTICS
            A29-21>  -MEET STAFF
            A29-20>  -RELATE INFORMATION ON HAND


2.6==EVERYTHING IS DISTRIBUTED                   ==    —THEN-GO-TO-CLUSTER-> 0
    2.6 IF  ALL REGULAR DATA IS DISTRIBUTED  AND IF
        NOT  INFORMATION IS REQUIRED ON AN IRREGULAR BASIS
            A29-25>  NO ACTION - NEXT CLUSTER


2.7==PAUSE                                       ==    —THEN-GO-TO-CLUSTER-> 1
    2.7 IF  NOT  ALL REGULAR DATA IS DISTRIBUTED  AND IF
        NOT  INFORMATION IS REQUIRED ON AN IRREGULAR BASIS
            A29-26>  WAIT
```

17. ABL-TRANSFORM DISTRIBUTION;


```
SYSTALL A(29) = A29> = INFORMATION AND DATA DISTRIBUTION

                     = #DISTRIBUTION WITH 26 ACTIONS
                     = SYSTALL A29>-(1..26)

INFORMATION AND DATA REDISTRIBUTION IS A COMPLEX TASK IN THIS SYSTEM.
DATA COLLECTION IS A FRAGMENTED PROCEDURE WHICH CONTINUES THROUGHOUT THE
DAY AT EVERY LECTURE.  IN ORDER TO RESPECT THE CRITERIONS ESTABLISHED
FOR THE INFORMATION TO REACH ITS TARGET USER AND IN ORDER TO PREVENT
A QUEU FORMATION OF THE DATA TO BE DISTRIBUTED AT THE END OF THE DAY,
WE HAVE TO BEGIN TO SEND BACK WHATEVER DATA IS AVAILABLE EVEN
BEFORE THE FINAL DATA PROCESSING TAKES PLACE. THIS ALSO HAS TO BE
ACHIEVED IN A CRITICAL TIME DELAY.

THIS REGULAR REDISTRIBUTION PROCESS HAS ALSO TO ACCOMODATE A FLOW
OF VARIOUS KIND OF INFORMATION REQUESTS ABOUT THE STORED TRUANCY DATA
IN THE DATA-BASE.
```

CONTINUED ON THE NEXT PAGE

	1	2	3	4	5	6	7	8	9	10	11	12	(ALTERNATIVES)
													CLUSTERS -
C1	1	1	1	2	.	.	.	.	.	.	.	0	== REGULAR DAILY DATA DISTRIBUTION ==
C2	.	.	.	.	1	1	1	1	1	0	1	0	== OCCASIONAL IRREGULAR INFORMATION DISTRIBUTION ==
													PREDICATES -
P1	-	-	-	-	-	-	-	-	-	Y	N	-	ALL REGULAR DATA IS DISTRIBUTED
P2	-	-	-	Y	Y	Y	Y	Y	Y	N	N	-	INFORMATION IS REQUIRED ON AN IRREGULAR BASIS
P3	-	-	-	-	-	Y	Y	-	-	-	-	-	OBJECTIVE IS AN EFFICIENT STUDENT AND PARENT CONSULTING
P4	-	-	-	-	-	Y	-	Y	-	-	-	-	ACCEPTABLE DELAY IS ONE MONTH
P5	-	-	Y	-	Y	-	-	-	-	-	-	-	OBJECTIVE IS AN EFFICIENT STUDENT CONSULTING
P6	-	-	Y	-	-	-	-	-	-	-	-	-	ACCEPTABLE DELAY IS 1-2 DAYS
P7	Y	-	-	-	-	-	-	-	-	-	-	-	OBJECTIVE IS AN EFFICIENT PARENT CONSULTING
P8	Y	Y	-	-	Y	-	-	-	-	-	-	-	ACCEPTABLE DELAY IS 2 HOURS
P9	-	-	-	-	-	-	-	Y	-	-	-	-	OBJECTIVE IS TO PROVIDE EFFICIENT CLASS MANAGEMENT
P10	-	-	-	-	-	-	-	-	-	-	-	-	ACCEPTABLE DELAY IS 2 MONTHS
P11	-	-	-	-	-	-	-	-	Y	-	-	-	OBJECTIVE IS AN EFFICIENT SCHOOL MANAGEMENT
P12	-	-	-	-	-	-	-	-	Y	-	-	-	ACCEPTABLE DELAY IS 2-3 DAYS
P13	-	-	-	-	Y	-	Y	Y	-	-	-	-	IMMEDIATE INFORMATION ACCES IS POSSIBLE
P14	Y	-	-	-	-	-	-	-	-	-	-	-	DATA COLLECTION FOR LECTURE 1 IS COMPLETE
P15	N	Y	-	-	-	-	-	-	-	-	-	-	DATA COLLECTION FOR LECTURE 2 OR 3 ....,OR 7 IS COMPLETE
P16	N	N	Y	-	-	-	-	-	-	-	-	-	DATA COLLECTION FOR THE DAY IS COMPLETE
P17	N	N	Y	-	-	-	-	-	-	-	-	-	DATA PROCESSING IS ENDED
													ACTIONS -
A1	1	1	1	.	.	.	.	.	.	.	.	.	A29-1> AT THE CENTER SEPARATE ALL DATA TO CORRESPONDING #VP
A2	2	2	.	.	.	.	.	.	.	.	.	.	A29-2> -SEPARATE INPUT #ABS-CARD DECK FOR EACH #VP
A3	3	3	2	.	.	.	.	.	.	.	.	.	A29-3> -SEPARATE OUTPUT FOR EACH #VP
A4	4	4	.	.	.	.	.	.	.	.	.	.	A29-4> -SEND BACK EACH #VP HIS #ABS-CARDS
A5	5	5	.	.	.	.	.	.	.	.	.	.	A29-5> -SEND BACK EACH #VP HIS ERROR LISTING
A6	6	.	.	.	.	.	.	.	.	.	.	.	A29-6> -SEND BACK EACH #VP HIS TRUANCY REPORT "A"
A7	.	.	3	.	.	.	.	.	.	.	.	.	A29-7> -SEND BACK EACH #VP HIS TRUANCY REPORT "B"
A8	.	.	4	.	.	.	.	.	.	.	.	.	A29-8> -SEND PRINCIPAL TRUANCY STATISTICS REPORT
A9	7	6	.	.	.	.	.	.	.	.	.	.	A29-9> AT THE #VP'S OFFICE PREPARE NEXT PRODUCTION CYCLE
A10	8	7	.	.	.	.	.	.	.	.	.	.	A29-10> -CHECK ERROR LISTING FOR DAMAGED #ABS-CARDS
A11	9	8	.	.	.	.	.	.	.	.	.	.	A29-11> -TAKE OUT FAULTY CARDS FROM EACH DECK
A12	10	9	.	.	.	.	.	.	.	.	.	.	A29-12> -REPLACE #ABS-CARDS TO THEIR RESPECTIVE #DAT-GEN-KIT FOR REUSE
A13	11	10	.	.	.	.	.	.	.	.	.	.	A29-13> -DISTRIBUTE #DAT-GEN-KITS TO TEACHERS' MAILBOXES
A14	12	.	5	.	.	.	.	.	.	.	.	.	A29-14> PERFORM TRUANCY RELATED HUMAN COMMUNICATION
A15	13	.	.	.	.	.	.	.	.	.	.	.	A29-15> -PARTITION TRUANCY REPORT "A" INTO SMALL SECTIONS
A16	14	.	.	.	.	.	.	.	.	.	.	.	A29-16> HAND OUT EACH SECTION TO PRE-ASSIGNED PERSONNAL
A17	15	.	.	.	.	.	.	.	.	.	.	.	A29-17> PHONE PARENTS OF TRUANT STUDENTS
A18	.	.	6	.	2	2	.	.	.	.	.	.	A29-18> -MEET TRUANT STUDENT
A19	.	.	.	.	.	3	2	.	.	.	.	.	A29-19> -MEET PARENTS OF TRUANT STUDENT
A20	16	.	7	.	3	4	.	3	3	.	.	.	A29-20> -RELATE INFORMATION ON HAND
A21	.	.	.	.	.	.	3	2	2	.	.	.	A29-21> -MEET STAFF
A22	.	.	.	.	1	1	1	.	.	.	.	.	A29-22> -QUERRY #DB. FOR RETROACTIVE STUDENT TRUANCY INFORMATION
A23	.	.	.	.	.	.	.	1	.	.	.	.	A29-23> -QUERRY #DB. FOR RETROACTIVE CLASS TRUANCY INFORMATION
A24	.	.	.	.	.	.	.	.	1	.	.	.	A29-24> -QUERRY #DB. FOR OVERALL TRUANCY STATISTICS
A25	.	.	.	1	.	.	.	.	.	1	.	.	A29-25> NO ACTION - NEXT CLUSTER
A26	.	.	.	.	.	.	.	.	.	.	1	.	A29-26> WAIT

15. ABL-ROUTINE DATAUSE;

```
*************************************************************************
*                                                                       *
*        SYSTALL A(30) = A30> = DATA USAGE DETERMINATION                *
*                        ------------------------------                 *
*                            = $DATAUSE WITH 14 ACTIONS                 *
*                            = SYSTALL A30>-(1..14)                     *
*                                                                       *
*                                                                       *
* DATA USAGE DETERMINATION IS THE MOST IMPORTANT FEATURE IN THE SUCCESFULL*
* MODELING AND IMPLEMENTATION OF THIS $ATS SYSTEM.  INITIALLY BY STUDYING*
* DATA USAGE, WE CAN DETECT MAJOR FLAWS IN THE LOGICAL DESIGN OF THE     *
* SYSTEM, LONG BEFORE THE USER BECOMES FRUSTRATED BY ITS SHORTCOMINGS    *
* AND REJECTS THE SYSTEM ALLTOGETHER.                                   *
*                                                                       *
* SUBSEQUENT DATA USAGE STUDIES WILL PERMIT TO CLOSELY TUNE IN THE       *
* REPORTING PART OF THE SYSTEM TO THE REAL AND OFTEN UNEXPRESSED NEED    *
* OF THE USER IN HIS DECICION MAKING TASK.  IN THIS WAY WE CAN OPTIMIZE  *
* OUTPUT USAGE, WHILE  MINIMIZING OUTPUT VOLUME AND THE INHERENT         *
* DATA PROCESSING EXPENSES.                                             *
*                                                                       *
* WHILE MEASURING OUTPUT USAGE DIFFERS WITH EACH TYPE OF OUTPUT, BASICALLY*
* WE HAVE TO ESTABLISH A COMPARATIVE STANDARD OF HOW MUCH ATTENTION      *
* THEY SHOULD GET TO BE CONSIDERED USEFULL.                             *
*                                                                       *
*************************************************************************
```

```
1.=== OBSERVATION                                        ===$        < 2 >
    1.1==USAGE VERIFICATION STUDY          ==  --THEN-GO-TO-CLUSTER-> 2
2.=== OUTPUT OPTIMIZATION                                ===$        < 2, 2, 2, 2, 2, 0, 0 >
    2.1==REPORT A IS TOO LONG - LOW USAGE  ==  --THEN-GO-TO-CLUSTER-> 2
    2.2==REPORT B IS TOO LONG - LOW USAGE  ==  --THEN-GO-TO-CLUSTER-> 2
    2.3==REPORT A IS MAYBE TOO SHORT       ==  --THEN-GO-TO-CLUSTER-> 2
    2.4==REPORT B IS  MAYBE TOO SHORT      ==  --THEN-GO-TO-CLUSTER-> 2
    2.5==OPTIMUM OUTPUT LEVEL              ==  --THEN-GO-TO-CLUSTER-> 2
    2.6==SYSTEM LOGIC PROBLEM              ==  --THEN-GO-TO-CLUSTER-> 0
    2.7==DATA USAGE STUDY IS COMPLETE      ==  --THEN-GO-TO-CLUSTER-> 0
```

```
15.  ABL-ROUTINE  DATAUSE;

*************************************************************************
*                                                                       *
*         SYSTALL A(30) = A30> = DATA USAGE DETERMINATION               *
*                                ------------------------               *
*                               = $DATAUSE WITH 14 ACTIONS              *
*                               = SYSTALL A30>-(1..14)                  *
*                                                                       *
*                                                                       *
*  DATA USAGE DETERMINATION IS THE MOST IMPORTANT FEATURE IN THE SUCCESFULL *
*  MODELING AND IMPLEMENTATION OF THIS $ATS SYSTEM.  INITIALLY BY STUDYING *
*  DATA USAGE, WE CAN DETECT MAJOR FLAWS IN THE LOGICAL DESIGN OF THE   *
*  SYSTEM, LONG BEFORE THE USER BECOMES FRUSTRATED BY ITS SHORTCOMINGS  *
*  AND REJECTS THE SYSTEM ALLTOGETHER.                                  *
*                                                                       *
*  SUBSEQUENT DATA USAGE STUDIES WILL PERMIT TO CLOSELY TUNE IN THE     *
*  REPORTING PART OF THE SYSTEM TO THE REAL AND OFTEN UNEXPRESSED NEED  *
*  OF THE USER IN HIS DECICION MAKING TASK.  IN THIS WAY WE CAN OPTIMIZE *
*  OUTPUT USAGE, WHILE  MINIMIZING OUTPUT VOLUME AND THE INHERENT       *
*  DATA PROCESSING EXPENSES.                                            *
*                                                                       *
*  WHILE MEASURING OUTPUT USAGE DIFFERS WITH EACH TYPE OF OUTPUT, BASICALLY *
*  WE HAVE TO ESTABLISH A COMPARATIVE STANDARD OF HOW MUCH ATTENTION     *
*  THEY SHOULD GET TO BE CONSIDERED USEFULL.                            *
*                                                                       *
*************************************************************************
   1.== OBSERVATION                                                ==$        < 2 >

      1.1==USAGE VERIFICATION STUDY                          ==    --THEN-GO-TO-CLUSTER-> 2
          1.1 IF  ALL OUTPUT IS REDISTRIBUTED IN THE PREDETERMINED DELAY  AND IF
                  NOT  OBSERVATION IS COMPLETE
   2.== OUTPUT OPTIMIZATION                                                ==$        < 2, 2, 2, 2, 2, 0, 0 >

      2.1==REPORT A IS TOO LONG - LOW USAGE                  ==    --THEN-GO-TO-CLUSTER-> 2
          2.1 IF  'A' RATIO IS SMALL  AND IF
                  NOT  DESIGN LOGIC PROBLEM DETECTED

      2.2==REPORT B IS TOO LONG - LOW USAGE                  ==    --THEN-GO-TO-CLUSTER-> 2
          2,2 IF  'B' RATIO IS SMALL  AND IF
                  NOT  DESIGN LOGIC PROBLEM DETECTED

      2.3==REPORT A IS MAYBE TOO SHORT                       ==    --THEN-GO-TO-CLUSTER-> 2
          2.3 IF  'A' RATIO IS HIGH

      2.4==REPORT B IS  MAYBE TOO SHORT                      ==    --THEN-GO-TO-CLUSTER-> 2
          2.4 IF  'B' RATIO IS HIGH

      2.5==OPTIMUM OUTPUT LEVEL                              ==    --THEN-GO-TO-CLUSTER-> 2
          2.5 IF  NOT  'A' RATIO IS SMALL  AND IF
                  NOT  'B' RATIO IS SMALL  AND IF
                  NOT  'A' RATIO IS HIGH  AND IF
                  NOT  'B' RATIO IS HIGH

      2.6==SYSTEM LOGIC PROBLEM                              ==    --THEN-GO-TO-CLUSTER-> 0
          2.6 IF  'A' RATIO IS SMALL  AND IF
                  'B' RATIO IS SMALL  AND IF
                  DESIGN LOGIC PROBLEM DETECTED

      2.7==DATA USAGE STUDY IS COMLETE                       ==    --THEN-GO-TO-CLUSTER-> 0
          2.7 IF  DATA USAGE STUDY IS COMPLETE
```

15. ABL-ROUTINE DATAUSE;


```
SYSTALL A(30) = A30> = DATA USAGE DETERMINATION
                     = $DATAUSE WITH 14 ACTIONS
                     = SYSTALL A30>-(1..14)

DATA USAGE DETERMINATION IS THE MOST IMPORTANT FEATURE IN THE SUCCESFULL
MODELING AND IMPLEMENTATION OF THIS DATA SYSTEM.  INITIALLY BY STUDYING
DATA USAGE, WE CAN DETECT MAJOR FLAWS IN THE LOGICAL DESIGN OF THE
SYSTEM, LONG BEFORE THE USER BECOMES FRUSTRATED BY ITS SHORTCOMINGS
AND REJECTS THE SYSTEM ALLTOGETHER.

SUBSEQUENT DATA USAGE STUDIES WILL PERMIT TO CLOSELY TUNE IN THE
REPORTING PART OF THE SYSTEM TO THE REAL AND OFTEN UNEXPRESSED NEED
OF THE USER IN HIS DECICION MAKING TASK.  IN THIS WAY WE CAN OPTIMIZE
OUTPUT USAGE, WHILE  MINIMIZING OUTPUT VOLUME AND THE INHERENT
DATA PROCESSING EXPENSES.

WHILE MEASURING OUTPUT USAGE DIFFERS WITH EACH TYPE OF OUTPUT, BASICALLY
WE HAVE TO ESTABLISH A COMPARATIVE STANDARD OF HOW MUCH ATTENTION
THEY SHOULD GET TO BE CONSIDERED USEFULL.
```

1.== OBSERVATION ===$ < 2 >

1.1==USAGE VERIFICATION STUDY == —THEN-GO-TO-CLUSTER-> 2
1.1 IF ALL OUTPUT IS REDISTRIBUTED IN THE PREDETERMINED DELAY AND IF
NOT OBSERVATION IS COMPLETE
A30-1> DETERMINE NUMBER OF PHONE CALLS TO PARENTS ($NOPH.)
A30-2> DETERMINE NUMBER OF STUDENTS NOT BY AVP. ($NONT.)
A30-3> DETERMINE NUMBER OF STUDENTS ON REPORT 'A' ($NA.)
A30-4> DETERMINE NUMBER OF STUDENTS ON REPORT 'B' ($NB.)
A30-5> DETERMINE RATIO 'A' = $NOPH./$NA.
A30-6> DETERMINE RATIO 'B' = $NONT./$NB.

2.== OUTPUT OPTIMIZATION ===$ < 2, 2, 2, 2, 2, 0, 0 >

2.1==REPORT A IS TOO LONG - LOW USAGE == —THEN-GO-TO-CLUSTER-> 2
2.1 IF 'A' RATIO IS SMALL AND IF
NOT DESIGN LOGIC PROBLEM DETECTED
A30-7> SHORTEN REPORT 'A' FOR THE NEXT PRODUCTION CYCLE
A30-12> ADJUST OUTPUT SELECTION PARAMETERS FOR THE NEXT PRODUCTION CYCLE

2.2==REPORT B IS TOO LONG - LOW USAGE == —THEN-GO-TO-CLUSTER-> 2
2.2 IF 'B' RATIO IS SMALL AND IF
NOT DESIGN LOGIC PROBLEM DETECTED
A30-9> SHORTEN REPORT 'B' FOR THE NEXT PRODUCTION CYCLE
A30-12> ADJUST OUTPUT SELECTION PARAMETERS FOR THE NEXT PRODUCTION CYCLE

CONTINUED ON THE NEXT PAGE

```
2.3==REPORT A IS MAYBE TOO SHORT                    ==    --THEN-GO-TO-CLUSTER-> 2
     2.3 IF  "A" RATIO IS HIGH
             A30-8>  LEGTHEN REPORT "A" FOR THE NEXT PRODUCTION CYCLE
             A30-12> ADJUST OUTPUT SELECTION PARAMETERS FOR THE NEXT PRODUCTION CYCLE


2.4==REPORT B IS  MAYBE TOO SHORT                   ==    --THEN-GO-TO-CLUSTER-> 2
     2.4 IF  "B" RATIO IS HIGH
             A30-10> LEGTHEN REPORT "B" FOR THE NEXT PRODUCTION CYCLE
             A30-12> ADJUST OUTPUT SELECTION PARAMETERS FOR THE NEXT PRODUCTION CYCLE


2.5==OPTIMUM OUTPUT LEVEL                           ==    --THEN-GO-TO-CLUSTER-> 2
     2.5 IF  NOT  "A" RATIO IS SMALL  AND IF
             NOT  "B" RATIO IS SMALL  AND IF
             NOT  "A" RATIO IS HIGH   AND IF
             NOT  "B" RATIO IS HIGH
             A30-11> MAINTAIN OUTPUT LEVEL FOR THE NEXT PRODUCTION CYCLE
             A30-12> ADJUST OUTPUT SELECTION PARAMETERS FOR THE NEXT PRODUCTION CYCLE


2.6==SYSTEM LOGIC PROBLEM                           ==    --THEN-GO-TO-CLUSTER-> 0
     2.6 IF  "A" RATIO IS SMALL  AND IF
             "B" RATIO IS SMALL  AND IF
             DESIGN LOGIC PROBLEM DETECTED
             A30-7>  SHORTEN REPORT "A" FOR THE NEXT PRODUCTION CYCLE
             A30-9>  SHORTEN REPORT "B" FOR THE NEXT PRODUCTION CYCLE
             A30-14> REVIEW SYSTEM LOGIC


2.7==DATA USAGE STUDY IS COMLETE                    ==    --THEN-GO-TO-CLUSTER-> 0
     2.7 IF  DATA USAGE STUDY IS COMPLETE
             A30-13> NO ACTION - NEXT CLUSTER
```

15. ABL-TRANSFORM DATAUSE;

```
*****************************************************************
*                                                               *
*        SYSTALL A(30) = A30> = DATA USAGE DETERMINATION         *
*                                                               *
*                      = $DATAUSE WITH 14 ACTIONS               *
*                      = SYSTALL A30>-(1..14)                   *
*                                                               *
*                                                               *
* DATA USAGE DETERMINATION IS THE MOST IMPORTANT FEATURE IN THE SUCCESFULL *
* MODELING AND IMPLEMENTATION OF THIS $ATS SYSTEM.  INITIALLY BY STUDYING *
* DATA USAGE, WE CAN DETECT MAJOR FLAWS IN THE LOGICAL DESIGN OF THE *
* SYSTEM, LONG BEFORE THE USER BECOMES FRUSTRATED BY ITS SHORTCOMINGS *
* AND REJECTS THE SYSTEM ALLTOGETHER.                           *
*                                                               *
* SUBSEQUENT DATA USAGE STUDIES WILL PERMIT TO CLOSELY TUNE IN THE *
* REPORTING PART OF THE SYSTEM TO THE REAL AND OFTEN UNEXPRESSED NEED *
* OF THE USER IN HIS DECICION MAKING TASK.  IN THIS WAY WE CAN OPTIMIZE *
* OUTPUT USAGE, WHILE  MINIMIZING OUTPUT VOLUME AND THE INHERENT *
* DATA PROCESSING EXPENSES.                                     *
*                                                               *
* WHILE MEASURING OUTPUT USAGE DIFFERS WITH EACH TYPE OF OUTPUT, BASICALLY *
* WE HAVE TO ESTABLISH A COMPARATIVE STANDARD OF HOW MUCH ATTENTION *
* THEY SHOULD GET TO BE CONSIDERED USEFULL.                     *
*                                                               *
*****************************************************************
```

	1	2	3	4	5	6	7	8	9	(ALTERNATIVES)
										CLUSTERS -
C1	2	:	:	:	:	:	:	:	0	=== OBSERVATION =====
C2	.	2	2	2	2	2	0	0	0	=== OUTPUT OPTIMIZATION =====
										PREDICATES -
P1	Y	-	-	-	-	-	-	-	-	ALL OUTPUT IS REDISTRIBUTED IN THE PREDETERMINED DELAY
P2	-	Y	-	-	-	N	Y	-	-	"A" RATIO IS SMALL
P3	-	-	Y	-	-	N	Y	-	-	"B" RATIO IS SMALL
P4	-	-	-	Y	-	N	-	-	-	"A" RATIO IS HIGH
P5	-	-	-	-	Y	N	-	-	-	"B" RATIO IS HIGH
P6	-	-	-	-	-	-	-	Y	-	DATA USAGE STUDY IS COMPLETE
P7	N	-	-	-	-	-	-	-	-	OBSERVATION IS COMPLETE
P8	-	N	N	-	-	-	Y	-	-	DESIGN LOGIC PROBLEM DETECTED
										ACTIONS -
A1	1	.	.	.	.	.	.	.	.	A30-1> DETERMINE NUMBER OF PHONE CALLS TO PARENTS ($NOPH.)
A2	2	.	.	.	.	.	.	.	.	A30-2> DETERMINE NUMBER OF STUDENTS MET BY $VP. ($NOMT.)
A3	3	.	.	.	.	.	.	.	.	A30-3> DETERMINE NUMBER OF STUDENTS ON REPORT "A" ($NA.)
A4	4	.	.	.	.	.	.	.	.	A30-4> DETERMINE NUMBER OF STUDENTS ON REPORT "B" ($NB.)
A5	5	.	.	.	.	.	.	.	.	A30-5> DETERMINE RATIO "A" = $NOPH./$NA.
A6	6	.	.	.	.	.	.	.	.	A30-6> DETERMINE RATIO "B" = $NOMT./$NB.
A7	.	1	.	.	.	.	1	.	.	A30-7> SHORTEN REPORT "A" FOR THE NEXT PRODUCTION CYCLE
A8	.	.	.	1	.	.	.	.	.	A30-8> LEGTHEN REPORT "A" FOR THE NEXT PRODUCTION CYCLE
A9	.	.	1	.	.	.	2	.	.	A30-9> SHORTEN REPORT "B" FOR THE NEXT PRODUCTION CYCLE
A10	.	.	.	.	1	.	.	.	.	A30-10> LEGTHEN REPORT "B" FOR THE NEXT PRODUCTION CYCLE
A11	.	.	.	.	.	1	.	.	.	A30-11> MAINTAIN OUTPUT LEVEL FOR THE NEXT PRODUCTION CYCLE
A12	.	2	2	2	2	2	.	.	.	A30-12> ADJUST OUTPUT SELECTION PARAMETERS FOR THE NEXT PRODUCTION CYCLE
A13	.	.	.	.	.	.	.	1	.	A30-13> NO ACTION [illegible] CLUSTER
A14	.	.	.	.	.	.	3	.	.	A30-14> REVIEW SYSTEM [illegible]GIC

```
SYSTALL ACTION(11) = A11> = UPDATE THE STUDENT MASTER FILE (# STUPDATE)

#STUPDATE IS ONE OF THE THREE COMPONENTS OF THE UPDATE SUBSYSTEM (# UPS ).
IT IS THE STUDENT MASTER FILE UPDATER PROGRAM OPERATED BY THE USER
IN INTERACTIVE MODE ON THE TERMINAL OF THE COMPUTER CENTER IN THE SCHOOL.
IN ORDER TO UPDATE THE STUDENT MASTER FILE, A SUITABLE EASY TO LEARN
LANGUAGE WAS DESIGNED THAT HELPS THE NAIVE USER TO MODIFY THE STUDENT
RECORDS. THE MAIN FUNCTION OF THIS PROGRAM IS TO PERMIT RECORD CREATION
AND DELETION AS WELL AS MODIFICATION OF AUTHORIZED FIELDS WITHIN EACH
STUDENT RECORD.  IT ALSO PERFORMS SOME DATA VERIFICATIONS BEFORE EDITED
DATA IS PERMITTED ENTRY INTO THE DATA-BASE. THIS #UPS SUBSYSTEM COMPONENT
FUNCTIONS IN A MANNER THAT WHEN CERTAIN DATA ARE ENTERED THROUGH THIS
PROGRAM THE OTHER DATA-BASE FILES (I.E.  THE COURSE MASTER FILE,
THE CLASS MASTER FILE AND THE #ABS-CARD FILE ) ARE ALL UPDATED
AUTOMATICALLY.

        THE LOGICAL DESCRIPTION OF SYSTALL A11>  IS 3 LEVELS DEEP

   SYSTALL A11> = #STUPDATE WITH 27 ACTIONS
                = SYSTALL A11>-(1..27)
                = IT CALLS IN 2 SUB-ROUTINES WHILE EXECUTING
                      AT A11-14  #STRECMOD
                      AT A11-15  #STRECDEL

      SYSTALL A11-14> = STUDENT RECORD MODIFICATION
                      = #STRECMOD WITH 16 ACTIONS
                      = SYSTALL A11-14>-(1..16)
                      = IT CALLS IN 2 SUB-ROUTINES WHILE EXECUTING
                            AT A11-14-5  #STRECMODPER
                            AT A11-14-6  #STRECMODCR

         SYSTALL A11-14-5> = STUDENT RECORD PERSONNAL ITEMS
                             MODIFICATION ROUTINE
                           = #STRECMODPER WITH 7 ACTIONS
                           = SYSTALL A11-14-5>-(1..7)

         SYSTALL A11-14-6> = STUDENT RECORD COURSE CHANGE ROUTINE
                           = #STRECMODCR WITH 14 ACTIONS
                           = SYSTALL A11-14-6>-(1..14)

      SYSTALL A11-15> = STUDENT RECORD DELETION ROUTINE
                      = #STRECDEL WITH 6 ACTIONS
                      = SYSTALL A11-15>-(1..6)
```

CONTINUED ON THE NEXT PAGE

```
1.=== START THE UPDATING SESSION                    ===$          < 2 >
    1.1==STARTING UPDATE                        ==  --THEN-GO-TO-CLUSTER-> 2
2.=== UPDATING FUNCTION SELECTION                   ===$          < 2, 2, 2, 3, 4, 2, 5, 2, 0, 6, 6 >
    2.1==COMMAND SELECTION                      ==  --THEN-GO-TO-CLUSTER-> 2
    2.2==OPERATING MODE SELECTION               ==  --THEN-GO-TO-CLUSTER-> 2
    2.3==NON VALID CREATION                     ==  --THEN-GO-TO-CLUSTER-> 2
    2.4==VALID CREATION                         ==  --THEN-GO-TO-CLUSTER-> 3
    2.5==VALID MODIFICATION                     ==  --THEN-GO-TO-CLUSTER-> 4
    2.6==NON VALID MODIFICATION                 ==  --THEN-GO-TO-CLUSTER-> 2
    2.7==VALID DELETION                         ==  --THEN-GO-TO-CLUSTER-> 5
    2.8==NON VALID DELETION                     ==  --THEN-GO-TO-CLUSTER-> 2
    2.9==UNAUTHORIZED USER                      ==  --THEN-GO-TO-CLUSTER-> 0
    2.10==END COMMAND WITH LISTING              ==  --THEN-GO-TO-CLUSTER-> 6
    2.11==END COMMAND WITH NO LISTING           ==  --THEN-GO-TO-CLUSTER-> 6
3.=== RECORD CREATION                               ===$          < 2 >
    3.1==RECORD CREATION                        ==  --THEN-GO-TO-CLUSTER-> 2
4.=== RECORD MODIFICATION                           ===$          < 2 >
    4.1==MODIFY EXISTING RECORD                 ==  --THEN-GO-TO-CLUSTER-> 2
5.=== RECORD DELETION                               ===$          < 2 >
    5.1==DELETE VALID RECORD                    ==  --THEN-GO-TO-CLUSTER-> 2
6.=== TERMINATION OF THE UPDATING SESSION           ===$          < 0, 0 >
    6.1==TERMINATE  WITH LISTING                ==  --THEN-GO-TO-CLUSTER-> 0
    6.2==TERMINATE WITHOUT LISTING              ==  --THEN-GO-TO-CLUSTER-> 0
```

4. ABL-ROUTINE STUPDATE;


```
SYSTALL ACTION(11).= A11> = UPDATE THE STUDENT MASTER FILE (# STUPDATE)

#STUPDATE IS ONE OF THE THREE COMPONENTS OF THE UPDATE SUBSYSTEM (# UPS ).
IT IS THE STUDENT MASTER FILE UPDATER PROGRAM OPERATED BY THE USER
IN INTERACTIVE MODE ON THE TERMINAL OF THE COMPUTER CENTER IN THE SCHOOL.
IN ORDER TO UPDATE THE STUDENT MASTER FILE, A SUITABLE EASY TO LEARN
LANGUAGE WAS DESIGNED THAT HELPS THE NAIVE USER TO MODIFY THE STUDENT
RECORDS. THE MAIN FUNCTION OF THIS PROGRAM IS TO PERMIT RECORD CREATION
AND DELETION AS WELL AS MODIFICATION OF AUTHORIZED FIELDS WITHIN EACH
STUDENT RECORD.  IT ALSO PERFORMS SOME DATA VERIFICATIONS BEFORE EDITED
DATA IS PERMITTED ENTRY INTO THE DATA-BASE. THIS #UPS SUBSYSTEM COMPONENT
FUNCTIONS IN A MANNER THAT WHEN CERTAIN DATA ARE ENTERED THROUGH THIS
PROGRAM THE OTHER DATA-BASE FILES (I.E.  THE COURSE MASTER FILE,
THE CLASS MASTER FILE AND THE #ABS-CARD FILE ) ARE ALL UPDATED
AUTOMATICALLY.

        THE LOGICAL DESCRIPTION OF SYSTALL A11>  IS 3 LEVELS DEEP

  SYSTALL A11> = #STUPDATE WITH 27 ACTIONS
               = SYSTALL A11>-(1..27)
               = IT CALLS IN 2 SUB-ROUTINES WHILE EXECUTING
                     AT A11-14  #STRECMOD
                     AT A11-15  #STRECDEL

      SYSTALL A11-14> = STUDENT RECORD MODIFICATION
                      = #STRECMOD WITH 16 ACTIONS
                      = SYSTALL A11-14>-(1..16)
                      = IT CALLS IN 2 SUB-ROUTINES WHILE EXECUTING
                            AT A11-14-5  #STRECMODPER
                            AT A11-14-6  #STRECMODCR

          SYSTALL A11-14-5> = STUDENT RECORD PERSONNAL ITEMS
                              MODIFICATION ROUTINE
                            = #STRECMODPER WITH 7 ACTIONS
                            = SYSTALL A11-14-5>-(1..7)

          SYSTALL A11-14-6> = STUDENT RECORD COURSE CHANGE ROUTINE
                            = #STRECMODCR WITH 14 ACTIONS
                            = SYSTALL A11-14-6>-(1..14)

      SYSTALL A11-15> = STUDENT RECORD DELETION ROUTINE
                      = #STRECDEL WITH 6 ACTIONS
                      = SYSTALL A11-15>-(1..6)
```

1.=== START THE UPDATING SESSION ===$ < 2 >

1.1==STARTING UPDATE == --THEN-GO-TO-CLUSTER-> 2

1.1 IF COMMAND = CALL MODETUB

CONTINUED ON THE NEXT PAGE

```
2.=== UPDATING FUNCTION SELECTION                                    ===*          < 2, 2, 2, 3, 4, 2, 5, 2, 0, 6, 6 >

     2.1==COMMAND SELECTION                                          ==    --THEN-GO-TO-CLUSTER-> 2
          2.1 IF  NOT  COMMAND = MODE,BRIEF  AND IF
                  NOT  COMMAND = MOD,STUDENT.NO.  AND IF
                  NOT  COMMAND = DEL,STUDENT.NO.,PASSWORD  AND IF
                  NOT  COMMAND = CRE,STUDENT.NO.  AND IF
                  NOT  COMMAND = FIN  AND IF
                  NOT  COMMAND = FIN,NOLIST  AND IF
                  COMMAND = ILLEGAL COMMAND

     2.2==OPERATING MODE SELECTION                                   ==    --THEN-GO-TO-CLUSTER-> 2
          2.2 IF  COMMAND = MODE,BRIEF  AND IF
                  MORE STUDENT RECORD TO UPDATE

     2.3==NON VALID CREATION                                         ==    --THEN-GO-TO-CLUSTER-> 2
          2.3 IF  COMMAND = CRE,STUDENT.NO.  AND IF
                  STUDENT.NO. IS VALID STUDENT ON FILE  AND IF
                  MORE STUDENT RECORD TO UPDATE

     2.4==VALID CREATION                                             ==    --THEN-GO-TO-CLUSTER-> 3
          2.4 IF  COMMAND = CRE,STUDENT.NO.  AND IF
                  NOT  STUDENT.NO. IS VALID STUDENT ON FILE  AND IF
                  MORE STUDENT RECORD TO UPDATE

     2.5==VALID MODIFICATION                                         ==    --THEN-GO-TO-CLUSTER-> 4
          2.5 IF  COMMAND = MOD,STUDENT.NO.  AND IF
              STUDENT.NO. IS VALID STUDENT ON FILE  AND IF
              MORE STUDENT RECORD TO UPDATE

     2.6==NON VALID MODIFICATION                                     ==    --THEN-GO-TO-CLUSTER-> 2
          2.6 IF  COMMAND = MOD,STUDENT.NO.  AND IF
                  NOT  STUDENT.NO. IS VALID STUDENT ON FILE  AND IF
                  MORE STUDENT RECORD TO UPDATE

     2.7==VALID DELETION                                             ==    --THEN-GO-TO-CLUSTER-> 5
          2.7 IF  COMMAND = DEL,STUDENT.NO.,PASSWORD  AND IF
              STUDENT.NO. IS VALID STUDENT ON FILE  AND IF
              PASSWORD IS VALID  AND IF
              MORE STUDENT RECORD TO UPDATE

     2.8==NON VALID DELETION                                         ==    --THEN-GO-TO-CLUSTER-> 2
          2.8 IF  COMMAND = DEL,STUDENT.NO.,PASSWORD  AND IF
              NOT  STUDENT.NO. IS VALID STUDENT ON FILE  AND IF
              MORE STUDENT RECORD TO UPDATE

     2.9==UNAUTHORIZED USER                                          ==    --THEN-GO-TO-CLUSTER-> 0
          2.9 IF  COMMAND = DEL,STUDENT.NO.,PASSWORD  AND IF
              STUDENT.NO. IS VALID STUDENT ON FILE  AND IF
              NOT  PASSWORD IS VALID  AND IF
              MORE STUDENT RECORD TO UPDATE

CONTINUED ON THE NEXT PAGE
```

```
2.10==END COMMAND WITH LISTING                          ==   --THEN-GO-TO-CLUSTER-> 6
     2.10 IF  COMMAND = FIN  AND IF
          NOT  MORE STUDENT RECORD TO UPDATE

2.11==END COMMAND WITH NO LISTING                       ==   --THEN-GO-TO-CLUSTER-> 6
     2.11 IF  COMMAND = FIN,NOLIST  AND IF
          NOT  MORE STUDENT RECORD TO UPDATE
3.=== RECORD CREATION                                        ===$              < 2 >

  3.1==RECORD CREATION                                  ==   --THEN-GO-TO-CLUSTER-> 2
       3.1P  NO CONDITIONS

4.=== RECORD MODIFICATION                                    ===$              < 2 >

  4.1==MODIFY EXISTING RECORD                           ==   --THEN-GO-TO-CLUSTER-> 2
       4.1P  NO CONDITIONS

5.=== RECORD DELETION                                        ===$              < 2 >

  5.1==DELETE VALID RECORD                              ==   --THEN-GO-TO-CLUSTER-> 2
       5.1P  NO CONDITIONS

6.=== TERMINATION OF THE UPDATING SESSION                    ===$              < 0, 0 >

  6.1==TERMINATE  WITH LISTING                          ==   --THEN-GO-TO-CLUSTER-> 0
       6.1 IF  COMMAND = FIN

  6.2==TERMINATE WITHOUT LISTING                        ==   --THEN-GO-TO-CLUSTER-> 0
       6.2 IF  COMMAND = FIN,NOLIST
```

4. ABL-ROUTINE STUPDATE;

```
*****************************************************************************
*                                                                           *
*     SYSTALL ACTION(11) = A11> = UPDATE THE STUDENT MASTER FILE (# STUPDATE)*
*                                                                           *
*                                                                           *
*  #STUPDATE IS ONE OF THE THREE COMPONENTS OF THE UPDATE SUBSYSTEM (# UPS ).*
*  IT IS THE STUDENT MASTER FILE UPDATER PROGRAM OPERATED BY THE USER       *
*  IN INTERACTIVE MODE ON THE TERMINAL OF THE COMPUTER CENTER IN THE SCHOOL. *
*  IN ORDER TO UPDATE THE STUDENT MASTER FILE, A SUITABLE EASY TO LEARN     *
*  LANGUAGE WAS DESIGNED THAT HELPS THE NAIVE USER TO MODIFY THE STUDENT    *
*  RECORDS. THE MAIN FUNCTION OF THIS PROGRAM IS TO PERMIT RECORD CREATION  *
*  AND DELETION AS WELL AS MODIFICATION OF AUTHORIZED FIELDS WITHIN EACH    *
*  STUDENT RECORD.  IT ALSO PERFORMS SOME DATA VERIFICATIONS BEFORE EDITED  *
*  DATA IS PERMITTED ENTRY INTO THE DATA-BASE. THIS #UPS SUBSYSTEM COMPONENT*
*  FUNCTIONS IN A MANNER THAT WHEN CERTAIN DATA ARE ENTERED THROUGH THIS    *
*  PROGRAM THE OTHER DATA-BASE FILES (I.E.  THE COURSE MASTER FILE,         *
*  THE CLASS MASTER FILE AND THE #ABS-CARD FILE ) ARE ALL UPDATED           *
*  AUTOMATICALLY.                                                           *
*                                                                           *
*          THE LOGICAL DESCRIPTION OF SYSTALL A11>  IS 3 LEVELS DEEP        *
*                                                                           *
*     SYSTALL A11> = #STUPDATE WITH 27 ACTIONS                              *
*                  = SYSTALL A11>-(1..27)                                   *
*                  = IT CALLS IN 2 SUB-ROUTINES WHILE EXECUTING             *
*                        AT A11-14  #STRECMOD                               *
*                        AT A11-15  #STRECDEL                               *
*                                                                           *
*          SYSTALL A11-14> = STUDENT RECORD MODIFICATION                    *
*                          = #STRECMOD WITH 16 ACTIONS                      *
*                          = SYSTALL A11-14>-(1..16)                        *
*                          = IT CALLS IN 2 SUB-ROUTINES WHILE EXECUTING     *
*                                AT A11-14-5  #STRECMODPER                  *
*                                AT A11-14-6  #STRECMODCR                   *
*                                                                           *
*              SYSTALL A11-14-5> = STUDENT RECORD PERSONAAL ITEMS           *
*                                  MODIFICATION ROUTINE                     *
*                                = #STRECMODPER WITH 7 ACTIONS              *
*                                = SYSTALL A11-14-5>-(1..7)                 *
*                                                                           *
*              SYSTALL A11-14-6> = STUDENT RECORD COURSE CHANGE ROUTINE     *
*                                = #STRECMODCR WITH 14 ACTIONS              *
*                                = SYSTALL A11-14-6>-(1..14)                *
*                                                                           *
*          SYSTALL A11-15> = STUDENT RECORD DELETION ROUTINE                *
*                          = #STRECDEL WITH 6 ACTIONS                       *
*                          = SYSTALL A11-15>-(1..6)                         *
*                                                                           *
*****************************************************************************
```

CONTINUED ON THE NEXT PAGE

```
1.=== START THE UPDATING SESSION                                    ===$          < 2 >

    1.1==STARTING UPDATE                                      =     --THEN-GO-TO-CLUSTER-> 2
        1.1 IF  COMMAND = CALL MODETUB
                A11-1>  ATTACH COMPILED INTERACTIVE PROGRAM
                A11-2>  ATTACH COURSE MASTER FILE
                A11-3>  ATTACH CLASS MASTER FILE
                A11-4>  ATTACH STUDENT MASTER FILE
                A11-5>  ATTACH $ABS-CARD FILE
                A11-6>  OPEN CLASS MASTER FILE
                A11-7>  OPEN COURSE MASTER FILE
                A11-8>  OPEN STUDENT MASTER FILE
                A11-9>  OPEN $ABS-CARD FILE
                A11-10>  CLEAR ALL BUFFERS
                A11-25>  REQUEST COMMAND FROM USER
                A11-26>  ENTER COMMAND

2.=== UPDATING FUNCTION SELECTION                                   ===$          < 2, 2, 2, 3, 4, 2, 5, 2, 0, 6, 6 >

    2.1==COMMAND SELECTION                                    =     --THEN-GO-TO-CLUSTER-> 2
        2.1 IF  NOT  COMMAND = MODE,BRIEF  AND IF
             NOT  COMMAND = MOD,STUDENT.NO.  AND IF
             NOT  COMMAND = DEL,STUDENT.NO.,PASSWORD  AND IF
             NOT  COMMAND = CRE,STUDENT.NO.  AND IF
             NOT  COMMAND = FIN  AND IF
             NOT  COMMAND = FIN,NOLIST  AND IF
             COMMAND = ILLEGAL COMMAND
                A11-16>  WRITE APPROPRIATE ERROR MESSAGE
                A11-25>  REQUEST COMMAND FROM USER
                A11-26>  ENTER COMMAND


    2.2==OPERATING MODE SELECTION                             =     --THEN-GO-TO-CLUSTER-> 2
        2.2 IF  COMMAND = MODE,BRIEF  AND IF
            MORE STUDENT RECORD TO UPDATE
               A11-11>  SET UP FLAG TO SUPPRESS VERIFICATION MESSAGES
               A11-25>  REQUEST COMMAND FROM USER
               A11-26>  ENTER COMMAND


    2.3==NON VALID CREATION                                   =     --THEN-GO-TO-CLUSTER-> 2
        2.3 IF  COMMAND = CRE,STUDENT.NO.  AND IF
             STUDENT.NO. IS VALID STUDENT ON FILE  AND IF
             MORE STUDENT RECORD TO UPDATE
                A11-16>  WRITE APPROPRIATE ERROR MESSAGE
                A11-25>  REQUEST COMMAND FROM USER
                A11-26>  ENTER COMMAND


    2.4==VALID CREATION                                       =     --THEN-GO-TO-CLUSTER-> 3
        2.4 IF  COMMAND = CRE,STUDENT.NO.  AND IF
             NOT  STUDENT.NO. IS VALID STUDENT ON FILE  AND IF
             MORE STUDENT RECORD TO UPDATE
                A11-27>  NO ACTION, NEXT CLUSTER
```

.CONTINUED ON THE NEXT PAGE

```
2.5==VALID MODIFICATION                                    ==   --THEN-GO-TO-CLUSTER-> 4
     2.5 IF  COMMAND = MOD,STUDENT.NO,  AND IF
          STUDENT.NO, IS VALID STUDENT ON FILE  AND IF
          MORE STUDENT RECORD TO UPDATE
            A11-27>  NO ACTION, NEXT CLUSTER


2.6==NON VALID MODIFICATION                                ==   --THEN-GO-TO-CLUSTER-> 2
     2.6 IF  COMMAND = MOD,STUDENT.NO,  AND IF
          NOT  STUDENT.NO, IS VALID STUDENT ON FILE  AND IF
          MORE STUDENT RECORD TO UPDATE
            A11-16>  WRITE APPROPRIATE ERROR MESSAGE
            A11-25>  REQUEST COMMAND FROM USER
            A11-26>  ENTER COMMAND


2.7==VALID DELETION                                        ==   --THEN-GO-TO-CLUSTER-> 5
     2.7 IF  COMMAND = DEL,STUDENT.NO.,PASSWORD  AND IF
          STUDENT.NO. IS VALID STUDENT ON FILE  AND IF
          PASSWORD IS VALID  AND IF
          MORE STUDENT RECORD TO UPDATE
            A11-27>  NO ACTION, NEXT CLUSTER


2.8==NON VALID DELETION                                    ==   --THEN-GO-TO-CLUSTER-> 2
     2.8 IF  COMMAND = DEL,STUDENT.NO.,PASSWORD  AND IF
          NOT  STUDENT.NO. IS VALID STUDENT ON FILE  AND IF
          MORE STUDENT RECORD TO UPDATE
            A11-16>  WRITE APPROPRIATE ERROR MESSAGE
            A11-25>  REQUEST COMMAND FROM USER
            A11-26>  ENTER COMMAND


2.9==UNAUTHORIZED USER                                     ==   --THEN-GO-TO-CLUSTER-> 0
     2.9 IF  COMMAND = DEL,STUDENT.NO.,PASSWORD  AND IF
          STUDENT.NO. IS VALID STUDENT ON FILE  AND IF
          NOT  PASSWORD IS VALID  AND IF
          MORE STUDENT RECORD TO UPDATE
            A11-16>  WRITE APPROPRIATE ERROR MESSAGE
            A11-17>  END UPDATING SESSION
            A11-20>  CLOSE STUDENT MASTER FILE
            A11-21>  CLOSE CLASS MASTER FILE
            A11-22>  CLOSE COURSE MASTER FILE
            A11-23>  CLOSE ABS-CARD FILE
            A11-19>  LOG OFF -ILLEGAL USER AT THE TERMINAL-


2.10==END COMMAND WITH LISTING                             ==   --THEN-GO-TO-CLUSTER-> 6
     2.10 IF  COMMAND = FIN  AND IF
          NOT  MORE STUDENT RECORD TO UPDATE
        2.10A A11-27>  NO ACTION, NEXT CLUSTER


2.11==END COMMAND WITH NO LISTING                          ==   --THEN-GO-TO-CLUSTER-> 6
     2.11 IF  COMMAND = FIN,NOLIST  AND IF
          NOT  MORE STUDENT RECORD TO UPDATE
        2.11A A11-27>  NO ACTION, NEXT CLUSTER
```

CONTINUED ON THE NEXT PAGE

```
3.=== RECORD CREATION                                          ===$          < 2 >

    3.1==RECORD CREATION                                       ==    --THEN-GO-TO-CLUSTER-> 2
        3.1P  NO CONDITIONS

              A11-10>  CLEAR ALL BUFFERS
              A11-12>  CREATE EMPTY RECORD IN STUDENT MASTER FILE
              A11-14>  PERFORM RECORD MODIFICATION, CALL $STRECMOD ROUTINE
              A11-25>  REQUEST COMMAND FROM USER
              A11-26>  ENTER COMMAND

4.=== RECORD MODIFICATION                                      ===$          < 2 >

    4.1==MODIFY EXISTING RECORD                                ==    --THEN-GO-TO-CLUSTER-> 2
        4.1P  NO CONDITIONS

              A11-10>  CLEAR ALL BUFFERS
              A11-13>  GET STUDENT RECORD INTO BUFFER
              A11-14>  PERFORM RECORD MODIFICATION, CALL $STRECMOD ROUTINE
              A11-25>  REQUEST COMMAND FROM USER
              A11-26>  ENTER COMMAND

5.=== RECORD DELETION                                          ===$          < 2 >

    5.1==DELETE VALID RECORD                                   ==    --THEN-GO-TO-CLUSTER-> 2
        5.1P  NO CONDITIONS

              A11-10>  CLEAR ALL BUFFERS
              A11-13>  GET STUDENT RECORD INTO BUFFER
              A11-15>  PERFORM RECORD DELETION, CALL $STRECDEL ROUTINE
              A11-25>  REQUEST COMMAND FROM USER
              A11-26>  ENTER COMMAND

6.=== TERMINATION OF THE UPDATING SESSION                      ===$          < 0, 0 >

    6.1==TERMINATE  WITH LISTING                               ==    --THEN-GO-TO-CLUSTER-> 0
        6.1 IF  COMMAND = FIN
              A11-17>  END UPDATING SESSION
              A11-18>  PRINT ALL MODIFIED CLASS LIST
              A11-20>  CLOSE STUDENT MASTER FILE
              A11-21>  CLOSE CLASS MASTER FILE
              A11-22>  CLOSE COURSE MASTER FILE
              A11-23>  CLOSE ABS-CARD FILE
              A11-24>  BACK UP ALL DATA-BASE FILES


    6.2==TERMINATE WITHOUT LISTING                             ==    --THEN-GO-TO-CLUSTER-> 0
        6.2 IF  COMMAND = FIN,NOLIST
              A11-17>  END UPDATING SESSION
              A11-20>  CLOSE STUDENT MASTER FILE
              A11-21>  CLOSE CLASS MASTER FILE
              A11-22>  CLOSE COURSE MASTER FILE
              A11-23>  CLOSE ABS-CARD FILE
              A11-24>  BACK UP ALL DATA-BASE FILES
```

4. ABL-TRANSFORM STUPDATE;


```
*****************************************************************
*                                                               *
*    SYSTALL ACTION(11) = A11> = UPDATE THE STUDENT MASTER FILE (# STUPDATE)
*                                                               *
*                                                               *
* #STUPDATE IS ONE OF THE THREE COMPONENTS OF THE UPDATE SUBSYSTEM (# UPS ).
* IT IS THE STUDENT MASTER FILE UPDATER PROGRAM OPERATED BY THE USER
* IN INTERACTIVE MODE ON THE TERMINAL OF THE COMPUTER CENTER IN THE SCHOOL.
* IN ORDER TO UPDATE THE STUDENT MASTER FILE, A SUITABLE EASY TO LEARN
* LANGUAGE WAS DESIGNED THAT HELPS THE NAIVE USER TO MODIFY THE STUDENT
* RECORDS. THE MAIN FUNCTION OF THIS PROGRAM IS TO PERMIT RECORD CREATION
* AND DELETION AS WELL AS MODIFICATION OF AUTHORIZED FIELDS WITHIN EACH
* STUDENT RECORD.  IT ALSO PERFORMS SOME DATA VERIFICATIONS BEFORE EDITED
* DATA IS PERMITTED ENTRY INTO THE DATA-BASE. THIS #UPS SUBSYSTEM COMPONENT
* FUNCTIONS IN A MANNER THAT WHEN CERTAIN DATA ARE ENTERED THROUGH THIS
* PROGRAM THE OTHER DATA-BASE FILES (I.E.  THE COURSE MASTER FILE,
* THE CLASS MASTER FILE AND THE #ABS-CARD FILE ) ARE ALL UPDATED
* AUTOMATICALLY.
*
*         THE LOGICAL DESCRIPTION OF SYSTALL A11>  IS 3 LEVELS DEEP
*
*    SYSTALL A11> = #STUPDATE WITH 27 ACTIONS
*                 = SYSTALL A11>-(1,,27)
*                 = IT CALLS IN 2 SUB-ROUTINES WHILE EXECUTING
*                       AT A11-14  #STRECMOD
*                       AT A11-15  #STRECDEL
*
*         SYSTALL A11-14> = STUDENT/RECORD MODIFICATION
*                         = #STRECMOD WITH 16 ACTIONS
*                         = SYSTALL A11-14>-(1..16)
*                         = IT CALLS IN 2 SUB-ROUTINES WHILE EXECUTING
*                               AT A11-14-5  #STRECMODPER
*                               AT A11-14-6  #STRECMODCR
*
*             SYSTALL A11-14-5> = STUDENT RECORD PERSONNAL ITEMS
*                                 MODIFICATION ROUTINE
*                               = #STRECMODPER WITH 7 ACTIONS
*                               = SYSTALL A11-14-5>-(1..7)
*
*             SYSTALL A11-14-6> = STUDENT RECORD COURSE CHANGE ROUTINE
*                               = #STRECMODCR WITH 14 ACTIONS
*                               = SYSTALL A11-14-6>-(1..14)
*
*         SYSTALL A11-15> = STUDENT RECORD DELETION ROUTINE
*                         = #STRECDEL WITH 6 ACTIONS
*                         = SYSTALL A11-15>-(1..6)
*
*****************************************************************
```

CONTINUED ON THE NEXT PAGE

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	(ALTERNATIVES)
																			CLUSTERS -
C1	2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	0	=== START THE UPDATING SESSION ===
C2	.	2	2	2	3	4	2	5	2	0	6	6	6	.	.	.	.	0	=== UPDATING FUNCTION SELECTION ===
C3	.	.	.	.	.	.	.	.	.	.	.	.	2	.	.	.	.	0	=== RECORD CREATION ===
C4	.	.	.	.	.	.	.	.	.	.	.	.	.	2	.	.	.	0	=== RECORD MODIFICATION ===
C5	.	.	.	.	.	.	.	.	.	.	.	.	.	.	2	.	.	0	=== RECORD DELETION ===
C6	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	0	0	0	=== TERMINATION OF THE UPDATING SESSION ===
																			PREDICATES -
P1	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	COMMAND = CALL MODETUB
P2	-	N	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	COMMAND = MODE,BRIEF
P3	-	N	-	-	-	Y	Y	-	-	-	-	-	-	-	-	-	-	-	COMMAND = MOD,STUDENT.NO.
P4	-	N	-	-	-	-	-	Y	Y	Y	-	-	-	-	-	-	-	-	COMMAND = DEL,STUDENT.NO.,,PASSWORD
P5	-	N	-	Y	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	COMMAND = CRE,STUDENT.NO.
P6	-	N	-	-	-	-	-	-	-	-	Y	-	-	-	-	Y	-	-	COMMAND = FIN
P7	-	N	-	-	-	-	-	-	-	-	-	Y	-	-	-	-	Y	-	COMMAND = FIN,NOLIST
P8	-	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	COMMAND = ILLEGAL COMMAND
P9	-	-	-	Y	N	Y	N	Y	N	Y	-	-	-	-	-	-	-	-	STUDENT.NO. IS VALID STUDENT ON FILE
P10	-	-	-	-	-	-	-	Y	-	N	-	-	-	-	-	-	-	-	PASSWORD IS VALID
P11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	OUTPUT SUPPRESSING FLAG IS UP
P12	-	-	Y	Y	Y	Y	Y	Y	Y	Y	N	N	-	-	-	-	-	-	MORE STUDENT RECORD TO UPDATE
																			ACTIONS -
A1	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A11-1> ATTACH COMPILED INTERACTIVE PROGRAM
A2	2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A11-2> ATTACH COURSE MASTER FILE
A3	3	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A11-3> ATTACH CLASS MASTER FILE
A4	4	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A11-4> ATTACH STUDENT MASTER FILE
A5	5	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A11-5> ATTACH ABS-CARD FILE
A6	6	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A11-6> OPEN CLASS MASTER FILE
A7	7	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A11-7> OPEN COURSE MASTER FILE
A8	8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A11-8> OPEN STUDENT MASTER FILE
A9	9	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A11-9> OPEN ABS-CARD FILE
A10	10	.	.	.	.	.	.	.	.	.	.	.	1	1	1	.	.	.	A11-10> CLEAR ALL BUFFERS
A11	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A11-11> SET UP FLAG TO SUPPRESS VERIFICATION MESSAGES
A12	.	.	.	.	.	.	.	.	.	.	.	.	2	.	.	.	.	.	A11-12> CREATE EMPTY RECORD IN STUDENT MASTER FILE
A13	.	.	.	.	.	.	.	.	.	.	.	.	.	2	2	.	.	.	A11-13> GET STUDENT RECORD INTO BUFFER
A14	.	.	.	.	.	.	.	.	.	.	.	.	3	3	.	.	.	.	A11-14> PERFORM RECORD MODIFICATION, CALL $STRECMOD ROUTINE
A15	.	.	.	.	.	.	.	.	.	.	.	.	.	.	3	.	.	.	A11-15> PERFORM RECORD DELETION, CALL $STRECDEL ROUTINE
A16	.	1	.	1	.	.	1	.	1	1	.	.	.	.	.	.	.	.	A11-16> WRITE APPROPRIATE ERROR MESSAGE
A17	.	.	.	.	.	.	.	.	.	2	.	.	.	.	.	1	1	.	A11-17> END UPDATING SESSION
A18	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	2	.	.	A11-18> PRINT ALL MODIFIED CLASS LIST
A19	.	.	.	.	.	.	.	.	.	7	.	.	.	.	.	.	.	.	A11-19> LOG OFF -ILLEGAL USER AT THE TERMINAL-
A20	.	.	.	.	.	.	.	.	.	3	.	.	.	.	.	3	2	.	A11-20> CLOSE STUDENT MASTER FILE
A21	.	.	.	.	.	.	.	.	.	4	.	.	.	.	.	4	3	.	A11-21> CLOSE CLASS MASTER FILE
A22	.	.	.	.	.	.	.	.	.	5	.	.	.	.	.	5	4	.	A11-22> CLOSE COURSE MASTER FILE
A23	.	.	.	.	.	.	.	.	.	6	.	.	.	.	.	6	5	.	A11-23> CLOSE ABS-CARD FILE
A24	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	7	6	.	A11-24> BACK UP ALL DATA-BASE FILES
A25	11	2	2	2	.	.	2	.	2	.	.	.	4	4	4	.	.	.	A11-25> REQUEST COMMAND FROM USER
A26	12	3	3	3	.	.	3	.	3	.	.	.	5	5	5	.	.	.	A11-26> ENTER COMMAND
A27	.	.	.	.	1	1	.	1	.	.	1	1	.	.	.	.	.	.	A11-27> NO ACTION, NEXT CLUSTER

5. ABL-ROUTINE STRECMOD;

```
*************************************************************************
*                                                                       *
*    SYSTALL ACTION(11-14) = A11-14> = STUDENT RECORD MODIFICATION ROUTINE*
*                          = #STRECMOD ROUTINE WITH 16 ACTIONS          *
*                          = SYSTALL A11-14>-(1..16)                    *
*                                                                       *
* THIS ROUTINE IS PART OF THE STUDENT MASTER FILE UPDATER PROGRAM (#STUPDATE)
* IT IS CALLED BY THE PROGRAM WHEN THE OPERATOR GIVES A COMMAND TO PERFORM
* STUDENT RECORD MODIFICATIONS.                                         *
*                                                                       *
*    THIS ROUTINE'S LOGICAL DESCRIPTION IS 2 LEVELS DEEP                *
*        AT SYSTALL A(11-14-5) = CALLS #STRECMODPER ROUTINE             *
*                              = STUDENT RECORD'S PERSONNAL ITEMS       *
*                                MODIFICATION ROUTINE                   *
*                                                                       *
*        AT SYSTALL A(11-14-6) = CALLS #STRECMODCR ROUTINE              *
*                              = STUDENT RECORD'S COURSE CHANGE ROUTINE  *
*    AT THE END RETURN TO #STUPDATE                                     *
*                                                                       *
*************************************************************************
```

```
1.=== ENTER #STRECMOD ROUTINE                          ===*          < 2 >

    1.1==ENTER #STRECMOD ROUTINE                    ==   --THEN-GO-TO-CLUSTER-> 2
2.=== FIELD CONTENT VERIFICATION                       ===*          < 2, 2, 3 >

    2.1==DISPLAY PERSONNAL ITEMS                    ==   --THEN-GO-TO-CLUSTER-> 2

    2.2==DISPLAY STUDENTS REGISTERED COURSES        ==   --THEN-GO-TO-CLUSTER-> 2

    2.3==NEXT CLUSTER, NO ACTION                    ==   --THEN-GO-TO-CLUSTER-> 3
3.=== FIELD EDITING                                    ===*          < 3, 3, 3, 3, 4 >

    3.1==PERFORM PERSONNAL CHANGES                  ==   --THEN-GO-TO-CLUSTER-> 3

    3.2==PERFORM COURSE CHANGE                      ==   --THEN-GO-TO-CLUSTER-> 3

    3.3==PERFORM SHORT PERSONNAL CHANGE             ==   --THEN-GO-TO-CLUSTER-> 3

    3.4==PERFORM SHORT COURSE CHANGE                ==   --THEN-GO-TO-CLUSTER-> 3

    3.5==NEXT CLUSTER NO ACTION                     ==   --THEN-GO-TO-CLUSTER-> 4
4.=== FINALISATION OF ENTERED CHANGES                  ===*          < 0, 0, 2 >

    4.1==FINALIZATION                               ==   --THEN-GO-TO-CLUSTER-> 0

    4.2==TOTAL ERROR RECOVERY                       ==   --THEN-GO-TO-CLUSTER-> 0

    4.3==LOOP BACK                                  ==   --THEN-GO-TO-CLUSTER-> 2
```

3. ABL-ROUTINE STRECMOD;

```
/*****************************************************************************
*                                                                            *
*    SYSTALL ACTION(11-14) = A11-14> = STUDENT RECORD MODIFICATION ROUTINE    *
*                          = #STRECMOD ROUTINE WITH 16 ACTIONS               *
*                          = SYSTALL A11-14>-(1,,16)                         *
*                                                                            *
* THIS ROUTINE IS PART OF THE STUDENT MASTER FILE UPDATER PROGRAM (#STUPDATE) *
* IT IS CALLED BY THE PROGRAM WHEN THE OPERATOR GIVES A COMMAND TO PERFORM   *
* STUDENT RECORD MODIFICATIONS.                                              *
*                                                                            *
*    THIS ROUTINE'S LOGICAL DESCRIPTION IS 2 LEVELS DEEP                     *
*        AT SYSTALL A(11-14-5) = CALLS #STRECMODPER ROUTINE                  *
*                              = STUDENT RECORD'S PERSONAL ITEMS             *
*                                MODIFICATION ROUTINE                        *
*                                                                            *
*        AT SYSTALL A(11-14-6) = CALLS #STRECMODCR ROUTINE                   *
*                              = STUDENT RECORD'S COURSE CHANGE ROUTINE      *
*    AT THE END RETURN TO #STUPDATE                                          *
*                                                                            *
*****************************************************************************/
```

1.== ENTER #STRECMOD ROUTINE ==: < 2 >

1.1==ENTER #STRECMOD ROUTINE = —THEN-GO-TO-CLUSTER-> 2
1.1P NO CONDITIONS

2.== FIELD CONTENT VERIFICATION ==: < 2, 2, 3 >

2.1==DISPLAY PERSONAL ITEMS = —THEN-GO-TO-CLUSTER-> 2
2.1 IF DISPLAY STUDENT PERSONAL ITEMS

2.2==DISPLAY STUDENTS REGISTERED COURSES = —THEN-GO-TO-CLUSTER-> 2
2.2 IF DISPLAY STUDENTS REGISTERED COURSES

2.3==NEXT CLUSTER, NO ACTION = —THEN-GO-TO-CLUSTER-> 3
2.3 IF NOT DISPLAY STUDENT PERSONAL ITEMS AND IF
NOT DISPLAY STUDENTS REGISTERED COURSES

3.== FIELD EDITING ==: < 3, 3, 3, 3, 4 >

3.1==PERFORM PERSONAL CHANGES = —THEN-GO-TO-CLUSTER-> 3
3.1 IF CHANGE IS ON PERSONAL ITEMS

3.2==PERFORM COURSE CHANGE = —THEN-GO-TO-CLUSTER-> 3
3.2 IF CHANGE IS ON COURSES

3.3==PERFORM SHORT PERSONAL CHANGE = —THEN-GO-TO-CLUSTER-> 3
3.3 IF CHANGE IS ON PERSONAL ITEMS AND IF
VERIFYING MESSAGES ARE SUPPRESSED

3.4==PERFORM SHORT COURSE CHANGE = —THEN-GO-TO-CLUSTER-> 3
3.4 IF CHANGE IS ON COURSES AND IF
VERIFYING MESSAGES ARE SUPPRESSED

3.5==NEXT CLUSTER NO ACTION = —THEN-GO-TO-CLUSTER-> 4
3.5 IF NOT CHANGE IS ON PERSONAL ITEMS AND IF
NOT CHANGE IS ON COURSES

CONTINUED ON THE NEXT PAGE

```
4.=== FINALISATION OF ENTERED CHANGES                          ===$          < 0, 0, 2 >

    4.1==FINALIZATION                                          ==    --THEN-GO-TO-CLUSTER-> 0
        4.1 IF  NOT  MORE MODIFICATIONS OR DISPLAY ON THIS RECORD  AND IF
            MAKE ENTERED MODIFICATIONS PERMENANT

    4.2==TOTAL ERROR RECOVERY                                  ==    --THEN-GO-TO-CLUSTER-> 0
        4.2 IF  NOT  MORE MODIFICATIONS OR DISPLAY ON THIS RECORD  AND IF
            NOT  MAKE ENTERED MODIFICATIONS PERMENANT

    4.3==LOOP BACK                                             ==    --THEN-GO-TO-CLUSTER-> 2
        4.3 IF  MORE MODIFICATIONS OR DISPLAY ON THIS RECORD
```

5. ABL-ROUTINE STRECMOD;

```
*************************************************************************
*                                                                       *
*   SYSTALL ACTION(11-14) = A11-14> = STUDENT RECORD MODIFICATION ROUTINE*
*                         = $STRECMOD ROUTINE WITH 16 ACTIONS           *
*                         = SYSTALL A11-14>-(1..16)                     *
*                                                                       *
*  THIS ROUTINE IS PART OF THE STUDENT MASTER FILE UPDATER PROGRAM ($STUPDATE)*
*  IT IS CALLED BY THE PROGRAM WHEN THE OPERATOR GIVES A COMMAND TO PERFORM*
*  STUDENT RECORD MODIFICATIONS.                                        *
*                                                                       *
*     THIS ROUTINE'S LOGICAL DESCRIPTION IS 2 LEVELS DEEP               *
*         AT SYSTALL A(11-14-5) = CALLS $STRECMODPER ROUTINE            *
*                               = STUDENT RECORD'S PERSONNAL ITEMS      *
*                                 MODIFICATION ROUTINE                  *
*                                                                       *
*         AT SYSTALL A(11-14-6) = CALLS $STRECMODCR ROUTINE             *
*                               = STUDENT RECORD'S COURSE CHANGE ROUTINE*
*     AT THE END RETURN TO $STUPDATE                                    *
*                                                                       *
*************************************************************************
```

1.== ENTER $STRECMOD ROUTINE ==$ < 2 >

1.1==ENTER $STRECMOD ROUTINE == --THEN-GO-TO-CLUSTER-> 2
1.1P NO CONDITIONS

A11-14-15> REQUEST COMMAND FROM THE USER

2.== FIELD CONTENT VERIFICATION ==$ < 2, 2, 3 >

2.1==DISPLAY PERSONNAL ITEMS == --THEN-GO-TO-CLUSTER-> 2
2.1 IF DISPLAY STUDENT PERSONAL ITEMS
A11-14-1> ENTER COMMAND = PERSONNEL
A11-14-2> DISPLAY ALL PERSONNAL ITEMS
A11-14-15> REQUEST COMMAND FROM THE USER

2.2==DISPLAY STUDENTS REGISTERED COURSES == --THEN-GO-TO-CLUSTER-> 2
2.2 IF DISPLAY STUDENTS REGISTERED COURSES
A11-14-3> ENTER COMMAND = COURSE
A11-14-4> DISPLAY ALL REGISTERED COURSES
A11-14-15> REQUEST COMMAND FROM THE USER

2.3==NEXT CLUSTER, NO ACTION == --THEN-GO-TO-CLUSTER-> 3
2.3 IF NOT DISPLAY STUDENT PERSONAL ITEMS AND IF
NOT DISPLAY STUDENTS REGISTERED COURSES
A11-14-16> NO ACTION, NEXT CLUSTER

CONTINUED ON THE NEXT PAGE

```
3.=== FIELD EDITING                                                    ===$          < 3, 3, 3, 3, 4 >

    3.1==PERFORM PERSONNAL CHANGES                                  ==    --THEN-GO-TO-CLUSTER-> 3
        3.1 IF  CHANGE IS ON PERSONNAL ITEMS
                A11-14-2>  DISPLAY ALL PERSONNAL ITEMS
                A11-14-5>  PERFORM PERSONNAL ITEM CHANGE - CALL $STRECHOOPER-
                A11-14-15>  REQUEST COMMAND FROM THE USER


    3.2==PERFORM COURSE CHANGE                                      ==    --THEN-GO-TO-CLUSTER-> 3
        3.2 IF  CHANGE IS ON COURSES
                A11-14-6>  PERFORM COURSE CHANGE -CALL $STRECHOOCR-
                A11-14-4>  DISPLAY ALL REGISTERED COURSES
                A11-14-15>  REQUEST COMMAND FROM THE USER


    3.3==PERFORM SHORT PERSONNAL CHANGE                             ==    --THEN-GO-TO-CLUSTER-> 3
        3.3 IF  CHANGE IS ON PERSONNAL ITEMS  AND IF
              VERIFYING MESSAGES ARE SUPPRESSED
                A11-14-5>  PERFORM PERSONNAL ITEM CHANGE - CALL $STRECHOOPER-
                A11-14-15>  REQUEST COMMAND FROM THE USER


    3.4==PERFORM SHORT COURSE CHANGE                                ==    --THEN-GO-TO-CLUSTER-> 3
        3.4 IF  CHANGE IS ON COURSES  AND IF
              VERIFYING MESSAGES ARE SUPPRESSED
                A11-14-6>  PERFORM COURSE CHANGE -CALL $STRECHOOCR-
                A11-14-15>  REQUEST COMMAND FROM THE USER


    3.5==NEXT CLUSTER NO ACTION                                     ==    --THEN-GO-TO-CLUSTER-> 4
        3.5 IF  NOT  CHANGE IS ON PERSONNAL ITEMS  AND IF
              NOT  CHANGE IS ON COURSES
                A11-14-16>  NO ACTION, NEXT CLUSTER

CONTINUED ON THE NEXT PAGE
```

```
4.=== FINALISATION OF ENTERED CHANGES                                    ===$          < 0, 0, 2 >

   4.1==FINALIZATION                                                     ==    --THEN-GO-TO-CLUSTER-> 0
      4.1 IF  NOT  MORE MODIFICATIONS OR DISPLAY ON THIS RECORD  AND IF
            MAKE ENTERED MODIFICATIONS PERMENANT
               A11-14-7>  ENTER COMMAND = COMPLETE
               A11-14-8>  UPDATE CLASS MASTER FILE RECORDS
               A11-14-9>  UPDATE COURSE MASTER FILE RECORDS
               A11-14-10>  APPEND NEW LINES TO @ABS-CARD FILE
               A11-14-11>  REWRITE STUDENT RECORD FROM BUFFER TO STUDENT MASTER FILE


   4.2==TOTAL ERROR RECOVERY                                             ==    --THEN-GO-TO-CLUSTER-> 0
      4.2 IF  NOT  MORE MODIFICATIONS OR DISPLAY ON THIS RECORD  AND IF
            NOT  MAKE ENTERED MODIFICATIONS PERMENANT
               A11-14-12>  ENTER COMMAND = QUIT
               A11-14-13>  DISREGARD ALL CHANGES TO THE RECORD
               A11-14-14>  CLEAR RECORD FROM BUFFER


   4.3==LOOP BACK                                                        ==    --THEN-GO-TO-CLUSTER-> 2
      4.3 IF  MORE MODIFICATIONS OR DISPLAY ON THIS RECORD
               A11-14-16>  NO ACTION, NEXT CLUSTER
```

```
5. ABL-TRANSFORM  STRECMOD;
```

```
*****************************************************************************
*                                                                           *
*    SYSTALL ACTION(11-14) = A11-14> = STUDENT RECORD MODIFICATION ROUTINE   *
*                          = #STRECMOD ROUTINE WITH 16 ACTIONS              *
*                          = SYSTALL A11-14>-(1.,16)                        *
*                                                                           *
*  THIS ROUTINE IS PART OF THE STUDENT MASTER FILE UPDATER PROGRAM (#STUPDATE)*
*  IT IS CALLED BY THE PROGRAM WHEN THE OPERATOR GIVES A COMMAND TO PERFORM  *
*  STUDENT RECORD MODIFICATIONS.                                            *
*                                                                           *
*     THIS ROUTINE'S LOGICAL DESCRIPTION IS 2 LEVELS DEEP                   *
*         AT SYSTALL A(11-14-5) = CALLS #STRECMODPER ROUTINE                *
*                              = STUDENT RECORD'S PERSONNAL ITEMS           *
*                                MODIFICATION ROUTINE                       *
*                                                                           *
*         AT SYSTALL A(11-14-6) = CALLS #STRECMODCR ROUTINE                 *
*                              = STUDENT RECORD'S COURSE CHANGE ROUTINE      *
*     AT THE END RETURN TO #STUPDATE                                        *
*                                                                           *
*****************************************************************************
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	(ALTERNATIVES)	
														CLUSTERS -	
C1	2	.	.	.	.	.	.	.	.	.	.	.	0	=== ENTER #STRECMOD ROUTINE	===
C2	.	2	2	3	.	.	.	.	.	.	.	.	0	=== FIELD CONTENT VERIFICATION	===
C3	.	.	.	.	3	3	3	3	4	.	.	.	0	=== FIELD EDITING	===
C4	.	.	.	.	.	.	.	.	.	0	0	2	0	=== FINALISATION OF ENTERED CHANGES	===
														PREDICATES -	
P1	-	Y	-	N	-	-	-	-	-	-	-	-	-	DISPLAY STUDENT PERSONAL ITEMS	
P2	-	-	Y	N	-	-	-	-	-	-	-	-	-	DISPLAY STUDENTS REGISTERED COURSES	
P3	-	-	-	-	Y	-	Y	-	N	-	-	-	-	CHANGE IS ON PERSONNAL ITEMS	
P4	-	-	-	-	-	Y	-	Y	N	-	-	-	-	CHANGE IS ON COURSES	
P5	-	-	-	-	-	-	-	-	-	N	N	Y	-	MORE MODIFICATIONS OR DISPLAY ON THIS RECORD	
P6	-	-	-	-	-	-	-	-	-	Y	N	-	-	MAKE ENTERED MODIFICATIONS PERMENANT	
P7	-	-	-	-	-	-	Y	Y	-	-	-	-	-	VERIFYING MESSAGES ARE SUPPRESSED	
														ACTIONS -	
A1	.	1	.	.	.	.	.	.	.	.	.	.	.	A11-14-1> ENTER COMMAND = PERSONNEL	
A2	.	2	.	.	1	.	.	.	.	.	.	.	.	A11-14-2> DISPLAY ALL PERSONNAL ITEMS	
A3	.	.	1	.	.	.	.	.	.	.	.	.	.	A11-14-3> ENTER COMMAND = COURSE	
A4	.	.	2	.	.	2	.	.	.	.	.	.	.	A11-14-4> DISPLAY ALL REGISTERED COURSES	
A5	.	.	.	.	2	.	1	.	.	.	.	.	.	A11-14-5> PERFORM PERSONNAL ITEM CHANGE - CALL #STRECMODPER-	
A6	.	.	.	.	.	1	.	1	.	.	.	.	.	A11-14-6> PERFORM COURSE CHANGE -CALL #STRECMODCR-	
A7	.	.	.	.	.	.	.	.	1	.	.	.	.	A11-14-7> ENTER COMMAND = COMPLETE	
A8	.	n	.	.	.	.	.	.	2	.	.	.	.	A11-14-8> UPDATE CLASS MASTER FILE RECORDS	
A9	.	.	.	.	.	.	.	.	3	.	.	.	.	A11-14-9> UPDATE COURSE MASTER FILE RECORDS	
A10	.	.	.	.	.	.	.	.	4	.	.	.	.	A11-14-10> APPEND NEW LINES TO #ABS-CARD FILE	
A11	.	.	.	.	.	.	.	.	5	.	.	.	.	A11-14-11> REWRITE STUDENT RECORD FROM BUFFER TO STUDENT MASTER FILE	
A12	.	.	.	.	.	.	.	.	.	.	1	.	.	A11-14-12> ENTER COMMAND = QUIT	
A13	.	.	.	.	.	.	.	.	.	.	2	.	.	A11-14-13> DISREGARD ALL CHANGES TO THE RECORD	
A14	.	.	.	.	.	.	.	.	.	.	3	.	.	A11-14-14> CLEAR RECORD FROM BUFFER	
A15	1	3	3	.	3	3	2	2	.	.	.	.	.	A11-14-15> REQUEST COMMAND FROM THE USER	
A16	.	.	.	1	.	.	.	.	1	.	.	1	.	A11-14-16> NO ACTION, NEXT CLUSTER	

7. ABL-ROUTINE STRECHOOPER;


```
*****************************************************************************
*                                                                           *
*        SYSTALL A(11-14-5) = PERFORM PERSONNAL ITEM CHANGE                 *
*                             ON THE STUDENT RECORD                         *
*                           = $STRECHOOPER WITH 7 ACTIONS                   *
*                           = SYSTALL A11-14-5)-(1..7)                      *
*                                                                           *
*  THIS ROUTINE IS CALLED BY $STRECHOO WHEN THE OPERATOR ENTERED COMMANDS   *
*  TO PERFORM MODIFICATIONS IN THE STUDENT RECORD'S PERSONNAL AREA.          *
*  THIS IS DONE ITEM BY ITEM IN INTERACTIVE MODE.                           *
*                                                                           *
*          AT THE END RETURN TO $STRECHOO                                   *
*                                                                           *
*****************************************************************************
```

```
1.== ENTER $STRECHOOPER ==>                                          < 2 >
    1.1==ENTER ROUTINE $STRECHOOPER                  ==  --THEN-GO-TO-CLUSTER-> 2
2.== ITEM SELECTION ==>                                              < 3, 3 >
    2.1==FIELDS ARE EMPTY                            ==  --THEN-GO-TO-CLUSTER-> 3
    2.2==SELECT FIELDS TO BE EDITED                  ==  --THEN-GO-TO-CLUSTER-> 3
3.== DATA ENTERING ==>                                               < 0, 0, 0, 0 >
    3.1==ENTER AND VERIFY SELECTIVE DATA             ==  --THEN-GO-TO-CLUSTER-> 0
    3.2==ERROR - ITEM OUT OF RANGE -                 ==  --THEN-GO-TO-CLUSTER-> 0
    3.3==ENTER SELECTIVE DATA                        ==  --THEN-GO-TO-CLUSTER-> 0
    3.4==ENTER ALL DATA ITEMS                        ==  --THEN-GO-TO-CLUSTER-> 0
```

7. ADL-ROUTINE STRECHKOPER:

```
SYSTALL A(11-14-5) = PERFORM PERSONNAL ITEM CHANGE
                     ON THE STUDENT RECORD
                   = $STRECHKOPER WITH 7 ACTIONS
                   = SYSTALL A11-14-5)-(1,,7)

THIS ROUTINE IS CALLED BY $STRECHKO WHEN THE OPERATOR ENTERED COMMANDS
TO PERFORM MODIFICATIONS IN THE STUDENT RECORD'S PERSONNAL AREA.
THIS IS DONE ITEM BY ITEM IN INTERACTIVE MODE.

        AT THE END RETURN TO $STRECHKO
```

1.=== ENTER $STRECHKOPER ===$ ⟨ 2 ⟩

1.1==ENTER ROUTINE $STRECHKOPER == --THEN-GO-TO-CLUSTER-> 2
1.1P NO CONDITIONS

2.=== ITEM SELECTION ===$ ⟨ 3, 3 ⟩

2.1==FIELDS ARE EMPTY == --THEN-GO-TO-CLUSTER-> 3
2.1 IF NEWLY CREATED RECORD IN BUFFER

2.2==SELECT FIELDS TO BE EDITED == --THEN-GO-TO-CLUSTER-> 3
2.2 IF OLD RECORD IN BUFFER

3.=== DATA ENTERING ===$ ⟨ 0, 0, 0, 0 ⟩

3.1==ENTER AND VERIFY SELECTIVE DATA == --THEN-GO-TO-CLUSTER-> 0
3.1 IF ITNO. = VALID ITEM NO. AND IF
NOT VERIFICATION MESSAGE IS SUPPRESSED

3.2==ERROR - ITEM OUT OF RANGE - == --THEN-GO-TO-CLUSTER-> 0
3.2 IF NOT ITNO. = VALID ITEM NO.

3.3==ENTER SELECTIVE DATA == --THEN-GO-TO-CLUSTER-> 0
3.3 IF ITNO. = VALID ITEM NO. AND IF
VERIFICATION MESSAGE IS SUPPRESSED

3.4==ENTER ALL DATA ITEMS == --THEN-GO-TO-CLUSTER-> 0
3.4 IF NEWLY CREATED RECORD IN BUFFER

```
7. ABL-ROUTINE STRECMODPER;

*************************************************************************
*                                                                       *
*       SYSTALL A(11-14-5). = PERFORM PERSONNAL ITEM CHANGE             *
*                             ON THE STUDENT RECORD                     *
*                           = #STRECMODPER WITH 7-ACTIONS               *
*                           = SYSTALL A11-14-5)-(1,,7)                  *
*                                                                       *
* THIS ROUTINE IS CALLED BY #STRECMOD WHEN THE OPERATOR ENTERED COMMANDS *
* TO PERFORM MODIFICATIONS IN THE STUDENT RECORD'S PERSONNAL AREA.       *
* THIS IS DONE ITEM BY ITEM IN INTERACTIVE MODE.                         *
*                                                                       *
*       AT THE END RETURN TO #STRECMOD                                  *
*                                                                       *
*************************************************************************

  1.=== ENTER #STRECMODPER ===*                                          < 2 >
      1.1==ENTER ROUTINE #STRECMODPER                          ==   --THEN-GO-TO-CLUSTER-> 2
          1.1P  NO CONDITIONS

                A11-14-5-7>  NO ACTION, NEXT CLUSTER

  2.=== ITEM SELECTION ===*                                              < 3, 3 >
      2.1==FIELDS ARE EMPTY                                    ==   --THEN-GO-TO-CLUSTER-> 3
          2.1 IF  NEWLY CREATED RECORD IN BUFFER
                A11-14-5-7>  NO ACTION, NEXT CLUSTER

      2.2==SELECT FIELDS TO BE EDITED                          ==   --THEN-GO-TO-CLUSTER-> 3
          2.2 IF  OLD RECORD IN BUFFER
                A11-14-5-2>  ENTER COMMAND = CORRECT,ITNO.,ITNO.,....ITNO.
                A11-14-5-3>  CHECK SYNTAX OF COMMAND LINE

  3.=== DATA ENTERING ===*                                               < 0, 0, 0, 0 >
      3.1==ENTER AND VERIFY SELECTIVE DATA                     ==   --THEN-GO-TO-CLUSTER-> 0
          3.1 IF  ITNO. = VALID ITEM NO.  AND IF
                NOT  VERIFICATION MESSAGE IS SUPPRESSED
                A11-14-5-4>  ENTER DATA FOR ITEMS ITNO., ITNO.,  ....ITNO.
                A11-14-5-5>  DISPLAY ALL PERSONNAL ITEMS

      3.2==ERROR - ITEM OUT OF RANGE -                         ==   --THEN-GO-TO-CLUSTER-> 0
          3.2 IF  NOT  ITNO. = VALID ITEM NO.
                A11-14-5-6>  WRITE ERROR MESSAGE

      3.3==ENTER SELECTIVE DATA                                ==   --THEN-GO-TO-CLUSTER-> 0
          3.3 IF  ITNO. = VALID ITEM NO.  AND IF
                VERIFICATION MESSAGE IS SUPPRESSED
                A11-14-5-4>  ENTER DATA FOR ITEMS ITNO., ITNO.,  ....ITNO.

      3.4==ENTER ALL DATA ITEMS                                ==   --THEN-GO-TO-CLUSTER-> 0
          3.4 IF  NEWLY CREATED RECORD IN BUFFER
                A11-14-5-1>  ENTER DATA FOR ALL PERSONNAL ITEMS FROM 1 TO 17
                A11-14-5-5>  DISPLAY ALL PERSONNAL ITEMS
```

7. ABL-TRANSFORM STRECMODPER;

```
*************************************************************************
*                                                                       *
*        SYSTALL A(11-14-5) = PERFORM PERSONNAL ITEM CHANGE             *
*                             ON THE STUDENT RECORD                     *
*                           = #STRECMODPER WITH 7 ACTIONS               *
*                           = SYSTALL A11-14-5)-(1,.7)                  *
*                                                                       *
*  THIS ROUTINE IS CALLED BY #STRECMOD WHEN THE OPERATOR ENTERED COMMANDS *
*  TO PERFORM MODIFICATIONS IN THE STUDENT RECORD'S PERSONNAL AREA.      *
*  THIS IS DONE ITEM BY ITEM IN INTERACTIVE MODE.                       *
*                                                                       *
*        AT THE END RETURN TO #STRECMOD                                 *
*                                                                       *
*************************************************************************
```

	1	2	3	4	5	6	7	8	(ALTERNATIVES)
									CLUSTERS -
C1	2	.	.	.	.	.	.	0	=== ENTER #STRECMODPER ===
C2	.	3	3	.	.	.	.	0	=== ITEM SELECTION ===
C3	.	.	.	0	0	0	0	0	=== DATA ENTERING ===
									PREDICATES -
P1	-	-	Y	-	-	-	-	-	OLD RECORD IN BUFFER
P2	-	Y	-	-	-	-	Y	-	NEWLY CREATED RECORD IN BUFFER
P3	-	-	-	Y	N	Y	-	-	ITMD. = VALID ITEM NO.
P4	-	-	-	N	-	Y	-	-	VERIFICATION MESSAGE IS SUPPRESSED
									ACTIONS -
A1	.	.	.	.	.	.	1	.	A11-14-5-1> ENTER DATA FOR ALL PERSONNAL ITEMS FROM 1 TO 17
A2	.	.	1	.	.	.	.	.	A11-14-5-2> ENTER COMMAND = CORRECT,ITMD.,ITMD.,....ITMD.
A3	.	.	2	.	.	.	.	.	A11-14-5-3> CHECK SYNTAX OF COMMAND LINE
A4	.	.	.	1	.	1	.	.	A11-14-5-4> ENTER DATA FOR ITEMS ITMD., ITMD., ...ITMD.
A5	.	.	.	2	.	.	2	.	A11-14-5-5> DISPLAY ALL PERSONNAL ITEMS
A6	.	.	.	.	1	.	.	.	A11-14-5-6> WRITE ERROR MESSAGE
A7	1	1	.	.	.	.	.	.	A11-14-5-7> NO ACTION, NEXT CLUSTER

8. ABL-ROUTINE STRECHNBCR;


```
*************************************************************************
*                                                                       *
*        SYSTALL A(11-14-6) = PERFORM COURSE CHANGES ON THE STUDENT RECORD *
*                           = $STRECHNBCR WITH 14 ACTIONS               *
*                           = SYSTALL A11-14-6)-(1..14)                 *
*                                                                       *
*  THIS ROUTINE IS CALLED BY $STRECHNB WHEN THE OPERATOR ENTERS COMMANDS TO *
*  MODIFY THE STUDENT RECORD'S COURSE REGISTRATION AREA. THIS ROUTINE PERFORMS *
*  COMPUTER ASSISTED VERIFICATIONS AND COURSE CHANGES REFLECTING THE STUDENT'S *
*  REGISTRATION IN COURSES AND THE DATA IN THE COURSE MASTER FILE.      *
*  IT ALSO MODIFIES AUTOMATICALLY THE COURSE MASTER FILE AND THE CLASS  *
*  MASTER FILE AND THE SABS-CARD FILE EACH TIME THERE IS A VALIDATED EDIT *
*  TO THE STUDENT'S COURSE REGISTRATION AREA.  IN THIS WAY THE MASTER    *
*  FILES ARE ALLWAYS CONSISTENT.                                        *
*                                                                       *
*          AT THE END RETURN TO $STRECHNB                               *
*                                                                       *
*************************************************************************
```

```
1.=== ENTER STRECHNBCR                                  ===$          . < 2 >
    1.1==ENTER ROUTINE                                  ==   --THEN-GO-TO-CLUSTER-> 2
2.=== ACTION EVALUATIONNN COMMAND VALIDITY              ===$              < 0, 3, 3, 2 >
    2.1==NO MORE MODIFICATION, RETURN                   ==   --THEN-GO-TO-CLUSTER-> 0
    2.2==ADD COURSE COMMAND LINE                        ==   --THEN-GO-TO-CLUSTER-> 3
    2.3==DROP COURSE COMMAND LINE                       ==   --THEN-GO-TO-CLUSTER-> 3
    2.4==ERROR -INVALID COMMAND LINE-                   ==   --THEN-GO-TO-CLUSTER-> 2
3.=== ACTION EVALUATION ON COURSE VALIDITY              ===$              < 2, 2, 4, 4, 2 >
    3.1==ERROR -NON EXISTANT COURSE NO.-                ==   --THEN-GO-TO-CLUSTER-> 2
    3.2==ERROR -STUDENT ALREADY REGISTERED IN COURSE-   ==   --THEN-GO-TO-CLUSTER-> 2
    3.3==ERROR -NO MORE ROOM IN CLASS-                  ==   --THEN-GO-TO-CLUSTER-> 4
    3.4==ERROR -COURSE REQUEST HAS TIME CONFLICT WITH OTHERS  ==   --THEN-GO-TO-CLUSTER-> 4
    3.5==ERROR -STUDENT IS NOT REGISTERED IN THIS COURSE-  ==   --THEN-GO-TO-CLUSTER-> 2
4.=== ERROR OVERRIDE                                    ===$              < 5, 2 >
    4.1==ERROR OVERRIDE                                 ==   --THEN-GO-TO-CLUSTER-> 5
    4.2==DONT OVERRIDE ERROR                            ==   --THEN-GO-TO-CLUSTER-> 2
```

CONTINUED ON THE NEXT PAGE

```
5.=== AUTHORIZED COURSE MODIFICATION                          ===$          < 6, 6, 6, 6 >
    5.1==ADD COURSES REQUESTED TO THE STUDENT RECORD          ==   --THEN-GO-TO-CLUSTER-> 6
    5.2==ADD COURSES REQUESTED TO THE #STUB.REC. AND VERIFY   ==   --THEN-GO-TO-CLUSTER-> 6
    5.3==DROP COURSES REQUESTED FROM THE #STUB.REC.           ==   --THEN-GO-TO-CLUSTER-> 6
    5.4==DROP COURSES REQUESTED FROM THE #STUB.REC. AND VERIFY ==  --THEN-GO-TO-CLUSTER-> 6
6.=== FINALIZATION OR ERROR RECOVERY                          ===$          < 2, 2 >
    6.1==END OF COURSE MODIFICATIONS                          ==   --THEN-GO-TO-CLUSTER-> 2
    6.2==FATAL ERROR RECOVERY                                 ==   --THEN-GO-TO-CLUSTER-> 2
```

8. ABL-ROUTINE STRECMODCR;


```
*****************************************************************************
*                                                                           *
*        SYSTALL A(11-14-6) = PERFORM COURSE CHANGES ON THE STUDENT RECORD  *
*                           = @STRECMODCR WITH 14 ACTIONS                   *
*                           = SYSTALL A11-14-6)-(1,,14)                     *
*                                                                           *
* THIS ROUTINE IS CALLED BY @STRECMOD WHEN THE OPERATOR ENTERS COMMANDS TO  *
* MODIFY THE STUDENT RECORD'S COURSE REGISTRATION AREA. THIS ROUTINE PERFORMS*
* COMPUTER ASSISTED VERIFICATIONS AND COURSE CHANGES REFLECTING THE STUDENT'S*
* REGISTRATION IN COURSES AND THE DATA IN THE COURSE MASTER FILE.           *
* IT ALSO MODIFIES AUTOMATICALLY THE COURSE MASTER FILE AND THE CLASS       *
* MASTER FILE AND THE $ABS-CARD FILE EACH TIME THERE IS A VALIDATED EDIT    *
* TO THE STUDENT'S COURSE REGISTRATION AREA.  IN THIS WAY THE MASTER        *
* FILES ARE ALLWAYS CONSISTENT.                                             *
*                                                                           *
*        AT THE END RETURN TO @STRECMOD                                     *
*                                                                           *
*****************************************************************************
```

1.== ENTER STRECMODCR ==$ < 2 >

1.1==ENTER ROUTINE == --THEN-GO-TO-CLUSTER-> 2
1.1P NO CONDITIONS

2.== ACTION EVALUATIONON COMMAND VALIDITY ==$ < 0, 3, 3, 2 >

2.1==NO MORE MODIFICATION, RETURN == --THEN-GO-TO-CLUSTER-> 0
2.1 IF NOT COURSE TO BE ADDED AND IF
NOT COURSE TO BE DROPPED

2.2==ADD COURSE COMMAND LINE == --THEN-GO-TO-CLUSTER-> 3
2.2 IF COURSE TO BE ADDED AND IF
VALID COMMAND

2.3==DROP COURSE COMMAND LINE == --THEN-GO-TO-CLUSTER-> 3
2.3 IF COURSE TO BE DROPPED AND IF
VALID COMMAND

2.4==ERROR -INVALID COMMAND LINE- == --THEN-GO-TO-CLUSTER-> 2
2.4 IF NOT VALID COMMAND

3.== ACTION EVALUATION ON COURSE VALIDITY ==$ < 2, 2, 4, 4, 2 >

3.1==ERROR -NON EXISTANT COURSE NO.- == --THEN-GO-TO-CLUSTER-> 2
3.1 IF NOT COURSE (COND.) IS IN COURSE MASTER FILE

3.2==ERROR -STUDENT ALREADY REGISTERED IN COURSE- == --THEN-GO-TO-CLUSTER-> 2
3.2 IF COURSE TO BE ADDED AND IF
STUDENT IS REGISTERED IN COURSE (COND.)

3.3==ERROR -NO MORE ROOM IN CLASS- == --THEN-GO-TO-CLUSTER-> 4
3.3 IF COURSE TO BE ADDED AND IF
NOT PLACE LEFT IN CLASS FOR NEW REGISTRATION

CONTINUED ON THE NEXT PAGE

```
    3.4==ERROR -COURSE REQUEST HAS TIME CONFLICT WITH OTHERS      ==    --THEN-GO-TO-CLUSTER-> 4
        3.4 IF  COURSE TO BE ADDED  AND IF
              COURSE SCHEDULE CONFLICT WITH OTHER ALREADY REGISTERED COURSES

    3.5==ERROR -STUDENT IS NOT REGISTERED IN THIS COURSE-         ==    --THEN-GO-TO-CLUSTER-> 2
        3.5 IF  COURSE TO BE DROPPED  AND IF
              NOT  STUDENT IS REGISTERED IN  COURSE (CRNO.)
4.=== ERROR OVERRIDE                                              ===$          < 5, 2 >

    4.1==ERROR OVERRIDE                                           ==    --THEN-GO-TO-CLUSTER-> 5
        4.1 IF  OVERRIDE ERROR

    4.2==DONT OVERRIDE ERROR                                      ==    --THEN-GO-TO-CLUSTER-> 2
        4.2 IF  NOT  OVERRIDE ERROR
5.=== AUTHORIZED COURSE MODIFICATION                              ===$          < 6, 6, 6, 6 >

    5.1==ADD COURSES REQUESTED TO THE STUDENT RECORD              ==    --THEN-GO-TO-CLUSTER-> 6
        5.1 IF  COURSE TO BE ADDED  AND IF
              VERIFICATION MESSAGES SUPPRESSED

    5.2==ADD COURSES REQUESTED TO THE $STUD.REC. AND VERIFY       ==    --THEN-GO-TO-CLUSTER-> 6
        5.2 IF  COURSE TO BE ADDED  AND IF
              NOT  VERIFICATION MESSAGES SUPPRESSED

    5.3==DROP COURSES REQUESTED FROM THE $STUD.REC.               ==    --THEN-GO-TO-CLUSTER-> 6
        5.3 IF  COURSE TO BE DROPPED  AND IF
              VERIFICATION MESSAGES SUPPRESSED

    5.4==DROP COURSES REQUESTED FROM THE $STUD.REC. AND VERIFY    ==    --THEN-GO-TO-CLUSTER-> 6
        5.4 IF  COURSE TO BE DROPPED  AND IF
              NOT  VERIFICATION MESSAGES SUPPRESSED
6.=== FINALIZATION OR ERROR RECOVERY                              ===$          < 2, 2 >

    6.1==END OF COURSE MODIFICATIONS                              ==    --THEN-GO-TO-CLUSTER-> 2
        6.1 IF  NOT  FATAL ERROR INTRODUCED IN STUD.RECORD

    6.2==FATAL ERROR RECOVERY                                     ==    --THEN-GO-TO-CLUSTER-> 2
        6.2 IF  FATAL ERROR INTRODUCED IN STUD.RECORD
```

8. ABL-ROUTINE STRECHNDCR;

```
*****************************************************************************
*                                                                           *
*        SYSTALL A(11-14-6) = PERFORM COURSE CHANGES ON THE STUDENT RECORD  *
*                           = #STRECHNDCR WITH 1A ACTIONS                   *
*                           = SYSTALL A11-14-6>(1..14)                      *
*                                                                           *
* THIS ROUTINE IS CALLED BY #STRECHNB WHEN THE OPERATOR ENTERS COMMANDS TO   *
* MODIFY THE STUDENT RECORD'S COURSE REGISTRATION AREA. THIS ROUTINE PERFORMS*
* COMPUTER ASSISTED VERIFICATIONS AND COURSE CHANGES REFLECTING THE STUDENT'S*
* REGISTRATION IN COURSES AND THE DATA IN THE COURSE MASTER FILE.           *
* IT ALSO MODIFIES AUTOMATICALLY THE COURSE MASTER FILE AND THE CLASS       *
* MASTER FILE AND THE #ABS-CARD FILE EACH TIME THERE IS A VALIDATED EDIT     *
* TO THE STUDENT'S COURSE REGISTRATION AREA.  IN THIS WAY THE MASTER         *
* FILES ARE ALLWAYS CONSISTENT.                                             *
*                                                                           *
*        AT THE END RETURN TO #STRECHNDB                                    *
*                                                                           *
*****************************************************************************
```

1.== ENTER STRECHNDCR ===$ < 2 >

1.1==ENTER ROUTINE == --THEN-GO-TO-CLUSTER-> 2
1.1P NO CONDITIONS

A11-14-6-14> NO ACTION, NEXT CLUSTER

2.== ACTION EVALUATIONON COMMAND VALIDITY ===$ < 0, 3, 3, 2 >

2.1==NO MORE MODIFICATION, RETURN == --THEN-GO-TO-CLUSTER-> 0
2.1 IF NOT COURSE TO BE ADDED AND IF
NOT COURSE TO BE DROPPED
A11-14-6-14> NO ACTION, NEXT CLUSTER

2.2==ADD COURSE COMMAND LINE == --THEN-GO-TO-CLUSTER-> 3
2.2 IF COURSE TO BE ADDED AND IF
VALID COMMAND
A11-14-6-1> ENTER COMMAND = ADD,CRNO.,CRNO.,.....,CRNO
A11-14-6-4> CHECK SYNTAX OF COMMAND LINE

2.3==DROP COURSE COMMAND LINE == --THEN-GO-TO-CLUSTER-> 3
2.3 IF COURSE TO BE DROPPED AND IF
VALID COMMAND
A11-14-6-2> ENTER COMMAND = DROP,CRNO.,CRNO.,.....CRNO
A11-14-6-4> CHECK SYNTAX OF COMMAND LINE

2.4==ERROR -INVALID COMMAND LINE- == --THEN-GO-TO-CLUSTER-> 2
2.4 IF NOT VALID COMMAND
A11-14-6-3> ENTER COMMAND = ELSE
A11-14-6-4> CHECK SYNTAX OF COMMAND LINE
A11-14-6-11> WRITE APPROPRIATE ERROR MESSAGE

CONTINUED ON THE NEXT PAGE

3.=== ACTION EVALUATION ON COURSE VALIDITY ===$ < 2, 2, 4, 4, 2 >

3.1==ERROR -NON EXISTANT COURSE NO.- == --THEN-GO-TO-CLUSTER-> 2
3.1 IF NOT COURSE (CRNO.) IS IN COURSE MASTER FILE
A11-14-6-11> WRITE APPROPRIATE ERROR MESSAGE

3.2==ERROR -STUDENT ALREADY REGISTERED IN COURSE- == --THEN-GO-TO-CLUSTER-> 2
3.2 IF COURSE TO BE ADDED AND IF
STUDENT IS REGISTERED IN COURSE (CRNO.)
A11-14-6-11> WRITE APPROPRIATE ERROR MESSAGE

3.3==ERROR -NO MORE ROOM IN CLASS- == --THEN-GO-TO-CLUSTER-> 4
3.3 IF COURSE TO BE ADDED AND IF
NOT PLACE LEFT IN CLASS FOR NEW REGISTRATION
A11-14-6-11> WRITE APPROPRIATE ERROR MESSAGE
A11-14-6-10> REQUEST OPERATOR WHETHER TO OVERRIDE ERROR

3.4==ERROR -COURSE REQUEST HAS TIME CONFLICT WITH OTHERS == --THEN-GO-TO-CLUSTER-> 4
3.4 IF COURSE TO BE ADDED AND IF
COURSE SCHEDULE CONFLICT WITH OTHER ALREADY REGISTERED COURSES
A11-14-6-11> WRITE APPROPRIATE ERROR MESSAGE
A11-14-6-10> REQUEST OPERATOR WHETHER TO OVERRIDE ERROR

3.5==ERROR -STUDENT IS NOT REGISTERED IN THIS COURSE- == --THEN-GO-TO-CLUSTER-> 2
3.5 IF COURSE TO BE DROPPED AND IF
NOT STUDENT IS REGISTERED IN COURSE (CRNO.)
A11-14-6-11> WRITE APPROPRIATE ERROR MESSAGE

47== ERROR OVERRIDE ===$ < 5, 2 >

4.1==ERROR OVERRIDE == --THEN-GO-TO-CLUSTER-> 5
4.1 IF OVERRIDE ERROR
A11-14-6-8> ENTER COMMAND = GO

4.2==DONT OVERRIDE ERROR == --THEN-GO-TO-CLUSTER-> 2
4.2 IF NOT OVERRIDE ERROR
A11-14-6-9> ENTER COMMAND = NOGO

CONTINUED ON THE NEXT PAGE

```
5.=== AUTHORIZED COURSE MODIFICATION                                    ===$          < 6, 6, 6, 6 >

    5.1==ADD COURSES REQUESTED TO THE STUDENT RECORD                    ==    --THEN-GO-TO-CLUSTER-> 6
        5.1 IF  COURSE TO BE ADDED  AND IF
            VERIFICATION MESSAGES SUPPRESSED
                A11-14-6-5>  ADD COURSES (CRNO.)  TO $STUD.REC. IN BUFFER


    5.2==ADD COURSES REQUESTED TO THE $STUD.REC. AND VERIFY             ==    --THEN-GO-TO-CLUSTER-> 6
        5.2 IF  COURSE TO BE ADDED  AND IF
            NOT  VERIFICATION MESSAGES SUPPRESSED
                A11-14-6-5>  ADD COURSES (CRNO.)  TO $STUD.REC. IN BUFFER
                A11-14-6-7>  DISPLAY STUDENTS COURSES IN BUFFER


    5.3==DROP COURSES REQUESTED FROM THE $STUD.REC.                     ==    --THEN-GO-TO-CLUSTER-> 6
        5.3 IF  COURSE TO BE DROPPED  AND IF
            VERIFICATION MESSAGES SUPPRESSED
                A11-14-6-6>  DROP COURSES (CRNO.) FROM $STUD.REC. IN BUFFER


    5.4==DROP COURSES REQUESTED FROM THE $STUD.REC. AND VERIFY          ==    --THEN-GO-TO-CLUSTER-> 6
        5.4 IF  COURSE TO BE DROPPED  AND IF
            NOT  VERIFICATION MESSAGES SUPPRESSED
                A11-14-6-6>  DROP COURSES (CRNO.) FROM $STUD.REC. IN BUFFER
                A11-14-6-7>  DISPLAY STUDENTS COURSES IN BUFFER

6.=== FINALIZATION OR ERROR RECOVERY                                    ===$          < 2, 2 >

    6.1==END OF COURSE MODIFICATIONS                                    ==    --THEN-GO-TO-CLUSTER-> 2
        6.1 IF  NOT  FATAL ERROR INTRODUCED IN STUD.RECORD
                A11-14-6-14>  NO ACTION, NEXT CLUSTER


    6.2==FATAL ERROR RECOVERY                                           ==    --THEN-GO-TO-CLUSTER-> 2
        6.2 IF  FATAL ERROR INTRODUCED IN STUD.RECORD
                A11-14-6-12>  ENTER COMMAND = CLEAR
                A11-14-6-13>  RESTORE $STUD.REC.COURSE CONTENT TO ITS ORIGINAL VALUES
```

8. ABL-TRANSFORM STRECHODCR;

```
SYSTALL A(11-14-6) = PERFORM COURSE CHANGES ON THE STUDENT RECORD
                   = #STRECHODCR WITH 14 ACTIONS
                   = SYSTALL A11-14-6>-(1..14)

THIS ROUTINE IS CALLED BY #STRECHOD WHEN THE OPERATOR ENTERS COMMANDS TO
MODIFY THE STUDENT RECORD'S COURSE REGISTRATION AREA. THIS ROUTINE PERFORMS
COMPUTER ASSISTED VERIFICATIONS AND COURSE CHANGES REFLECTING THE STUDENT'S
REGISTRATION IN COURSES AND THE DATA IN THE COURSE MASTER FILE.
IT ALSO MODIFIES AUTOMATICALLY THE COURSE MASTER FILE AND THE CLASS
MASTER FILE AND THE #ABS-CARD FILE EACH TIME THERE IS A VALIDATED EDIT
TO THE STUDENT'S COURSE REGISTRATION AREA.  IN THIS WAY THE MASTER
FILES ARE ALLWAYS CONSISTENT.

        AT THE END RETURN TO #STRECHOD
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	(ALTERNATIVES)
																				CLUSTERS -
C1	2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	0	=== ENTER STRECHODCR ===
C2	.	0	3	3	2	.	.	.	.	.	.	.	.	.	.	.	.	.	0	=== ACTION EVALUATIONON COMMAND VALIDITY ===
C3	.	.	.	.	.	2	2	4	4	2	.	.	.	.	.	.	.	.	0	=== ACTION EVALUATION ON COURSE VALIDITY ===
C4	.	.	.	.	.	.	.	.	.	.	5	2	.	.	.	.	.	.	0	=== ERROR OVERRIDE ===
C5	.	.	.	.	.	.	.	.	.	.	.	.	6	6	6	6	.	.	0	=== AUTHORIZED COURSE MODIFICATION ===
C6	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	2	2	0	=== FINALIZATION OR ERROR RECOVERY ===
																				PREDICATES -
P1	-	N	Y	-	-	-	Y	Y	Y	-	-	-	Y	Y	-	-	-	-	-	COURSE TO BE ADDED
P2	-	N	-	Y	-	-	-	-	-	Y	-	-	-	-	Y	Y	-	-	-	COURSE TO BE DROPPED
P3	-	-	Y	Y	N	-	-	-	-	-	-	-	-	-	-	-	-	-	-	VALID COMMAND
P4	-	-	-	-	-	N	-	-	-	-	-	-	-	-	-	-	-	-	-	COURSE (CRNO.) IS IN COURSE MASTER FILE
P5	-	-	-	-	-	-	Y	-	-	N	-	-	-	-	-	-	-	-	-	STUDENT IS REGISTERED IN COURSE (CRNO.)
P6	-	-	-	-	-	-	-	N	-	-	-	-	-	-	-	-	-	-	-	PLACE LEFT IN CLASS FOR NEW REGISTRATION
P7	-	-	-	-	-	-	-	-	Y	-	-	-	-	-	-	-	-	-	-	COURSE SCHEDULE CONFLICT WITH OTHER ALREADY REGISTERED COURSES
P8	-	-	-	-	-	-	-	-	-	-	Y	N	-	-	-	-	-	-	-	OVERRIDE ERROR
P9	-	-	-	-	-	-	-	-	-	-	-	-	Y	N	Y	N	-	-	-	VERIFICATION MESSAGES SUPPRESSED
P10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	N	Y	-	FATAL ERROR INTRODUCED IN STUD.RECORD
																				ACTIONS -
A1	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A11-14-6-1> ENTER COMMAND = ADD,CRNO.,CRNO., ...CRNO
A2	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A11-14-6-2> ENTER COMMAND = DROP,CRNO.,CRNO., ...CRNO
A3	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A11-14-6-3> ENTER COMMAND = ELSE
A4	.	.	2	2	2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A11-14-6-4> CHECK SYNTAX OF COMMAND LINE
A5	.	.	.	.	.	.	.	.	.	.	.	.	1	1	.	.	.	.	.	A11-14-6-5> ADD COURSES (CRNO.) TO #STUD.REC. IN BUFFER
A6	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	1	.	.	.	A11-14-6-6> DROP COURSES (CRNO.) FROM #STUD.REC. IN BUFFER
A7	.	.	.	.	.	.	.	.	.	.	.	.	.	2	.	2	.	.	.	A11-14-6-7> DISPLAY STUDENTS COURSES IN BUFFER
A8	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	A11-14-6-8> ENTER COMMAND = GO
A9	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	A11-14-6-9> ENTER COMMAND = NOGO
A10	.	.	.	.	.	.	.	2	2	.	.	.	.	.	.	.	.	.	.	A11-14-6-10> REQUEST OPERATOR WHETHER TO OVERRIDE ERROR
A11	.	.	.	.	3	1	1	1	1	1	.	.	.	.	.	.	.	.	.	A11-14-6-11> WRITE APPROPRIATE ERROR MESSAGE
A12	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	A11-14-6-12> ENTER COMMAND = CLEAR
A13	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	2	A11-14-6-13> RESTORE #STUD.REC.COURSE CONTENT TO ITS ORIGINAL VALUES
A14	1	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	A11-14-6-14> NO ACTION, NEXT CLUSTER

6. ABL-ROUTINE STRECDEL;

```
*************************************************************************
*                                                                       *
*       SYSTALL ACTION(11-15) = A11-15> = STUDENT RECORD DELETION ROUTINE*
*                                       = #STRECDEL WITH 6 ACTIONS      *
*                                       = SYSTALL A11-15>-(1,,6)        *
*                                                                       *
* THIS ROUTINE IS PART OF THE STUDENT MASTER FILE UPDATER PROGRAM (#STUPDATE)*
* IT IS CALLED BY THE PROGRAM WHEN THE OPERATOR GIVES A COMMAND TO PERFORM*
* A STUDENT RECORD DELETION FROM THE DATA-BASE MASTER FILES.            *
*                                                                       *
*       AT THE END RETURN TO #STUPDATE                                  *
*                                                                       *
*************************************************************************
```

```
1.=== ENTER #STRECDEL ROUTINE ===$                                  < 2 >

    1.1==VERIFY STUDENT TO BE DELETED          ==   --THEN-GO-TO-CLUSTER-> 2
2.=== DELETION ===$                                                < 0, 0 >

    2.1==VERIFIED DELETION                     ==   --THEN-GO-TO-CLUSTER-> 0

    2.2==DELETION RECOVERY                     ==   --THEN-GO-TO-CLUSTER-> 0
```

6. ABL-ROUTINE STRECDEL;

```
*************************************************************************
*                                                                       *
*        SYSTALL ACTION(11-15) = A11-15> = STUDENT RECORD DELETION ROUTINE *
*                                = #STRECDEL WITH 6 ACTIONS             *
*                                = SYSTALL A11-15>-(1,,6)               *
*                                                                       *
* THIS ROUTINE IS PART OF THE STUDENT MASTER FILE UPDATER PROGRAM (#STUPDATE) *
* IT IS CALLED BY THE PROGRAM WHEN THE OPERATOR GIVES A COMMAND TO PERFORM *
* A STUDENT RECORD DELETION FROM THE DATA-BASE MASTER FILES.            *
*                                                                       *
*        AT THE END RETURN TO #STUPDATE                                 *
*                                                                       *
*************************************************************************
```

1.=== ENTER #STRECDEL ROUTINE ===# < 2 >

1.1==VERIFY STUDENT TO BE DELETED == --THEN-GO-TO-CLUSTER-> 2
1.1P NO CONDITIONS

2.=== DELETION ===# < 0, 0 >

2.1==VERIFIED DELETION == --THEN-GO-TO-CLUSTER-> 0
2.1 IF DISPLAYED STUDENT IS REALLY TO BE DELETED

2.2==DELETION RECOVERY == --THEN-GO-TO-CLUSTER-> 0
2.2 IF NOT DISPLAYED STUDENT IS REALLY TO BE DELETED

6. ABL-ROUTINE STRECDEL;

```
*************************************************************************
*                                                                       *
*        SYSTALL ACTION(11-15) = A11-15> = STUDENT RECORD DELETION ROUTINE *
*                                        = #STRECDEL WITH 6 ACTIONS     *
*                                        = SYSTALL A11-15>-(1..6)       *
*                                                                       *
* THIS ROUTINE IS PART OF THE STUDENT MASTER FILE UPDATER PROGRAM (#STUPDATE) *
* IT IS CALLED BY THE PROGRAM WHEN THE OPERATOR GIVES A COMMAND TO PERFORM *
* A STUDENT RECORD DELETION FROM THE DATA-BASE MASTER FILES.            *
*                                                                       *
*        AT THE END RETURN TO #STUPDATE                                 *
*                                                                       *
*************************************************************************
```

1.== ENTER #STRECDEL ROUTINE ==# < 2 >

1.1==VERIFY STUDENT TO BE DELETED == —THEN-GO-TO-CLUSTER-> 2
1.1P NO CONDITIONS

A11-15-1> DISPLAY STUDENT NAME AND ID.NO. FOR VERIFICATION

2.== DELETION ==# < 0, 0 >

2.1==VERIFIED DELETION == —THEN-GO-TO-CLUSTER-> 0
2.1 IF DISPLAYED STUDENT IS REALLY TO BE DELETED
A11-15-2> ENTER COMMAND = YES
A11-15-4> PUT DEACTIVATION FLAG IN STUDENT RECORD
A11-15-5> DELETE STUDENT FROM EACH CLASS MASTER FILE RECORD HE IS REGISTERED
A11-15-6> UPDATE ENROLLEMENT STATISTICS ON RELATED COURSE MASTER FILE RECORDS

2.2==DELETION RECOVERY == —THEN-GO-TO-CLUSTER-> 0
2.2 IF NOT DISPLAYED STUDENT IS REALLY TO BE DELETED
A11-15-3> ENTER COMMAND = ELSE

6. ABL-TRANSFORM STRECDEL;

```
*************************************************************************
*                                                                       *
*       SYSTALL ACTION(11-15) = A11-15> = STUDENT RECORD DELETION ROUTINE *
*                                       = $STRECDEL WITH 6 ACTIONS      *
*                                       = SYSTALL A11-15>-(1..6)        *
*                                                                       *
* THIS ROUTINE IS PART OF THE STUDENT MASTER FILE UPDATER PROGRAM ($STUPDATE)
* IT IS CALLED BY THE PROGRAM WHEN THE OPERATOR GIVES A COMMAND TO PERFORM
* A STUDENT RECORD DELETION FROM THE DATA-BASE MASTER FILES.            *
*                                                                       *
*       AT THE END RETURN TO $STUPDATE                                  *
*                                                                       *
*************************************************************************
```

```
      1 2 3 4  (ALTERNATIVES)

                 CLUSTERS -
C1    2 . . 0    === ENTER $STRECDEL ROUTINE ===
C2    . 0 0 0    === DELETION ===

                 PREDICATES -
P1    - Y N -    DISPLAYED STUDENT IS REALLY TO BE DELETED

                 ACTIONS -
A1    1 . . .    A11-15-1>  DISPLAY STUDENT NAME AND ID.NO. FOR VERIFICATION
A2    . 1 . .    A11-15-2>  ENTER COMMAND = YES
A3    . . 1 .    A11-15-3>  ENTER COMMAND = ELSE
A4    . 2 . .    A11-15-4>  PUT DEACTIVATION FLAG IN STUDENT RECORD
A5    . 3 . .    A11-15-5>  DELETE STUDENT FROM EACH CLASS MASTER FILE RECORD HE IS REGISTERED
A6    . 4 . .    A11-15-6>  UPDATE ENROLLEMENT STATISTICS ON RELATED COURSE MASTER FILE RECORDS
```

ANNEX II

INPUT TABLES USED IN THE
PASCAL PROGRAM "CLOCKIN"
TO GENERATE THE ABL TABLES
"MODELING" AND "SYSTALL"
IN ANNEX I

EE INFERIEURE

```
PROCEDURE MODELING ( ACTNB:INTEGER);

BEGIN
    CASE ACTNB OF
        -1 : WRITE(' PRODUCTION CYCLE PERIOD IS DEFINED AS 1-2 DAYS LONG ');
        -2 : WRITE(' DATA COLLECTION PERIOD IS DEFINED AS 30-40 MINUTES LONG ');
        -3 : WRITE(' INEXPENSIVE TERMINAL IS AVAILABLE ');
        -4 : WRITE(' INEXPENSIVE DATA COMMUNICATION IS AVAILABLE ');
        -5 : WRITE(' REMOTE TIME-SHARING COMPUTER IS AVAILABLE ');
        -6 : WRITE(' ON-LINE SYSTEM ACCESS IS NEEDED ');
        -7 : WRITE(' DISK-STORAGE IS AVAILABLE ');
        -8 : WRITE(' INTERACTIVE QUERRY AND UPDATE IS NEEDED ');
        -9 : WRITE(' SYSTEM USAGE IS STUDIED ');
       -10 : WRITE(' SYSTEM OPERATING SPECIALIST IS AVAILABLE IN SCHOOL ');
       -11 : WRITE(' SYSTEM OPERATING TASK HAS TO BE SHARED BY REGULAR STAFF ');
       -12 : WRITE(' USER SYSTEM ACCESS ORGANIZATION IS DEFINED ');
       -13 : WRITE(' CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED ');
       -14 : WRITE(' SYSTEM MODULARIZATION IS DEFINED ');
       -15 : WRITE(' HIGH DEGREE OF DATA-BASE ACCURACY IS NEEDED ');
       -16 : WRITE(' HIGH VOLUME OF CHANGES OCCUR IN THE DATA-BASE ');
       -17 : WRITE(' DATA-BASE CHANGES HAVE TO BE IMPLEMENTED IN 1-2 DAYS ');
       -18 : WRITE(' INTERACTIVE UPDATING LANGUAGE FOR THE COURSE #MF. IS NEEDED ');
       -19 : WRITE(' INTERACTIVE UPDATING LANGUAGE FOR THE STUDENT #MF. IS NEEDED');
       -20 : WRITE(' CLASS MASTER FILE UPDATING CAN BE DONE AUTOMATICALLY ');
       -21 : WRITE(' IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING ');
       -22 : WRITE(' OUTPUT VOLUME IS SMALL ');
       -23 : WRITE(' INTERACTIVE QUERRY LANGUAGE FOR THE COURSE #MF. IS NEEDED ');
       -24 : WRITE(' INTERACTIVE QUERRY LANGUAGE FOR THE STUDENT #MF. IS NEEDED ');
       -25 : WRITE(' INTERACTIVE QUERRY LANGUAGE FOR THE CLASS #MF. IS NEEDED ');
       -26 : WRITE(' HIGH VOLUME OF TRUANCY TRANSACTION OCCUR (100-250 A LECTURE)');
       -27 : WRITE(' DATA GENERATION PERIOD IS 30-60 SECONDS ');
       -28 : WRITE(' ERROR GENERATION POSSIBILITY IN TRANSACTION DATA IS MINIMAL ');
       -29 : WRITE(' INEXPENSIVE OPTICAL MARK READER IS FOUND FOR THE TERMINAL ');
       -30 : WRITE(' REGULAR #AIS OPERATOR IS AVAILABLE FROM STAFF ');
       -31 : WRITE(' DATA TRANSMISSION IS BAD SOMETIMES ');
       -32 : WRITE(' OUTPUT PRINTING ON THE TERMINAL SOMETIMES HAS TO WAIT ');
       -33 : WRITE(' VARIABLE LENGTH OUTPUT IS NEEDED ');
       -34 : WRITE(' DATA GENERATION PROCESS IS ACCEPTED BY THE TEACHERS ');
       -35 : WRITE(' DATA COLLECTION PROCESS IS ACCEPTED BY THE ADMINISTRATION ');
       -36 : WRITE(' DATA COLLECTION ROUTE IS ORGANIZED ');
       -37 : WRITE(' DATA COLLECTION ERRORS ARE MINIMAL ');
       -38 : WRITE(' EXTENSIVE TRANSACTION STORAGE AND ANALYSIS IS REQUIRED ');
       -39 : WRITE(' PROCEDURE IS SIMPLE ');
       -40 : WRITE(' PROCEDURE IS SHORT ');
       -41 : WRITE(' OUTPUT CAN BE SPOOLED ');
       -42 : WRITE(' PRINTING OF SPOOLED OUTPUT IS NOT MORE THAN 1-2 HOURS ');
        43 : WRITE(' DISTRIBUTION DELAY IS DEFINED AS 2 HOURS ');
       -44 : WRITE(' OUTPUT VOLUME IS ADEQUATE TO DAILY USAGE ');
       -45 : WRITE(' FALSE NEGATIVE ERROR RATE IS BELOW 10% RATE ');
       -46 : WRITE(' FALSE POSITIVE ERROR RATE IS BELOW 0.1% RATE ');
       -47 : WRITE(' PROVIDED INFORMATION CAN BE READILY USED IN DECISION MAKING');
       -48 : WRITE(' INFORMATION VALUE IS MAXIMUM OPERATIONAL VS. OVERALL DELAY ');
       -49 : WRITE(' #VP IN CHARGE OF THE SYSTEM HAS ACCESS ');
       -50 : WRITE(' REDUNDANT DATA IN THE DATA-BASE ARE AUTOMATICALLY UPDATED ');
        51 : WRITE(' RECORDS OFFICE OF THE SCHOOL HAS ACCESS ');
        52 : WRITE(' SECTOR #VP. OF THE SCHOOL HAS ACCESS ');
        53 : WRITE(' CLASS RELATED DATA CHANGED OR SOME #ABS-CARDS GOT LOST ');
       -54 : WRITE(' DATA-BASE DATA IS UPDATED ');
       -55 : WRITE(' CARD PUNCHING AND INTERPRETING IS AVAILABLE ');
       -56 : WRITE(' OBJECTIVE IS AN EFFICIENT SCHOOL MANAGEMENT ');
       -57 : WRITE(' TEACHERS HAVE ACCESS ');
       -58 : WRITE(' STUDENTS AND PARENTS HAVE INDIRECT ACCESS ');
       -59 : WRITE(' OBJECTIVE IS AN EFFICIENT STUDENT AND PARENT COUNSELLING ');
       -60 : WRITE(' OBJECTIVE IS AN EFFICIENT CLASS MANAGEMENT ');
       -61 : WRITE(' COURSE INFORMATION IS NEEDED ');
       -62 : WRITE(' STUDENT INFORMATION IS NEEDED ');
       -63 : WRITE(' CLASS INFORMATION IS NEEDED ');
       -64 : WRITE(' ALL SUBFUNCTIONS IN THIS FUNCTION (CLUSTER) ARE MET ');
         1 : WRITE('A1>  INSTALL THE TERMINAL IN SCHOOL ');
         2 : WRITE('A2>  SET UP THE COURSE MASTER FILE ');
         3 : WRITE('A3>  SET UP THE STUDENT MASTER FILE ');
         4 : WRITE('A4>  SET UP THE CLASS MASTER FILE ');
         5 : WRITE('A5>  SET UP COURSE MASTER FILE UPDATER PROGRAM ');
         6 : WRITE('A6>  SET UP STUDENT MASTER FILE UPDATER PROGRAM ');
         7 : WRITE('A7>  SET UP COURSE MASTER FILE QUERRY PROGRAM ');
         8 : WRITE('A8>  SET UP STUDENT MASTER FILE QUERRY PROGRAM ');
         9 : WRITE('A9>  SET UP CLASS MASTER FILE QUERRY PROGRAM ');
        10 : WRITE('A10>  UPDATE COURSE MASTER FILE INTERACTIVELY ');
        11 : WRITE('A11>  UPDATE STUDENT MASTER FILE INTERACTIVELY ');
        12 : WRITE('A12>  UPDATE CLASS MASTER FILE INTERACTIVELY ');
        13 : WRITE('A13>  VERIFY COURSE MASTER FILE DATA ');
        14 : WRITE('A14>  VERIFY STUDENT MASTER FILE DATA ');
        15 : WRITE('A15>  VERIFY CLASS MASTER FILE DATA ');
        16 : WRITE('A16>  PRODUCE TEACHERS DATA GENERATION KITS ');
        17 : WRITE('A17>  SET UP DATA COLLECTION ROUTES IN SCHOOL ');
        18 : WRITE('A18>  INSTALL OPTICAL MARK READER ON TERMINAL ');
        19 : WRITE('A19>  VERIFY DATA DISTRIBUTION ROUTES ');
        20 : WRITE('A20>  DETERMINE DATA USAGE PARAMETERS ');
        21 : WRITE('A21>  SET UP DAILY INITIALIZATION INTERACTIVE PROGRAM ');
        22 : WRITE('A22>  SET UP DATA COLLECTION PSEUDOBATCH PROGRAM ');
        23 : WRITE('A23>  SET UP  #AIS. DATA PROCESSING PSEUDOBATCH PROGRAM ');
        24 : WRITE('A24>  SET UP PROCEDURE TO PRINT SPOOLED OUTPUT ');
        25 : WRITE('A25>  PERFORM DAILY SYSTEM INITIALIZATION ');
        26 : WRITE('A26>  PERFORM TRANSACTION DATA GENERATION ');
        27 : WRITE('A27>  PERFORM TRANSACTION DATA COLLECTION ');
        28 : WRITE('A28>  PERFORM DATA PROCESSING (REPORTING AND STORAGE) ');
        29 : WRITE('A29>  PERFORM DATA DISTRIBUTION ');
        30 : WRITE('A30>  VERIFY OUTPUT USAGE IN DECISION MAKING ');
        31 : WRITE('A31>  PUNCH NEW # ABS-CARDS ');
        32 : WRITE('A32>  UPDATE TEACHERS DATA GENERATION KITS ');
        33 : WRITE('A33>  QUERRY COURSE MASTER FILE INTERACTIVELY ');
        34 : WRITE('A34>  QUERRY STUDENT MASTER FILE INTERACTIVELY ');
        35 : WRITE('A35>  QUERRY CLASS MASTER FILE INTERACTIVELY ');
        36 : WRITE('A36>  WRITE A FILE IN COURSE MASTER FILE QUERRY LANGUAGE ');
        37 : WRITE('A37>  QUERRY COURSE MASTER FILE PSEUDOBATCH MODE ');
        38 : WRITE('A38>  PRINT SPOOLED OUTPUT ');
        39 : WRITE('A39>  WRITE A FILE IN STUDENT MASTER FILE QUERRY LANGUAGE ');
        40 : WRITE('A40>  QUERRY STUDENT MASTER FILE PSEUDOBATCH MODE ');
        41 : WRITE('A41>  WRITE A FILE IN CLASS MASTER FILE QUERRY LANGUAGE ');
        42 : WRITE('A42>  QUERRY CLASS MASTER FILE PSEUDOBATCH MODE ');
        43 : WRITE('A43>  PRODUCE # ABS-CARDS FOR THE CLASS ');
        44 : WRITE('A44>  NO ACTION- NEXT CLUSTER ');
         0 : WRITELN;
    END (*CASE *);
```

```
MODELING          (M-4,P-6)
1B
*****************************************************************************
*                                                                           *
*     MODELING  # OR THE OVERALL SYSTEM DESIGN AND MODELING OF THIS SYSTEM   *
*     ------------------------------------------------------------------    *
*                                                                           *
*  THIS TABLE PROVIDES AN OVERVIEW IN THE MODELING STAGE OF THE SYSTEM.      *
*  #MODELING  HAS 44 ACTION PROCEDURES AND 64 PREDICATES.  THESE ACTIONS     *
*  ARE GROUPED INTO FUNCTIONS AND SUBFUNCTIONS AND FORM THE HIERARCHICALLY   *
*  INTERRELATED BUILDING BLOCKS OF THE WHOLE SYSTEM. THE PREDICATES ACT AS   *
*  CRITERIONS FOR A PARTICULAR SUBFUNCTION. THEY DEFINE THE CONSTRAINTS      *
*  THE ACTIONS IN THE PARTICULAR SUBFUNTION MUST OBEY.  UNLESS THESE         *
*  CRITERIONS ARE SATISFIED, THAT PARTICULAR SUBFUNCTION IS NOT ADEQUATELY   *
*  DESIGNED TO MEET ITS PURPOSE.                                            *
*                                                                           *
*  ONCE ALL THE ACTIONS ARE DESIGNED, THESE SAME ACTIONS WILL BE OPERATIONALLY *
*  SEQUENCED AND INTERFACED IN THE NEXT TABLE CALLED  #SYSTALL              *
*                                                                           *
*****************************************************************************
4 64.44
1 * === ORGANIZE THE BODY OF THE SYSTEM                                  ===*
2 * === ORGANIZE THE REGULAR RUN OF THE SYSTEM                           ===*
3 * === DATA UPDATE ORGANIZATION                                        ===*
4 * === DATA-BASE QUERRY ORGANIZATION                                   ===*
20 28
1/1,2,3,4,5/1/1                                  1 ' ==DECENTRELIZED PROCESSING IS NEEDED                    =='
1/6,7,8,9/2,3,4/1                                 2 ' ==SET UP RANDOM ACCESS DATA-BASE                        =='
1/-10,11,12,13,14/2,3,4,5,6,7,8,9,21,22,23,24/1      3 ' ==MODULARIZE SYSTEM                                  =='
1/15,16,17,-10,11,12,13,18,19,20/5,6,10,11,12/1 4 ' ==SET UP INTERACTIVE #DB UPDATING NAIVE USER INTERFACE==='
1/21,22,-10,11,12,13,23,24,25/7,8,9,33,34,35/1 5 ' ==SET UP INTERACTIVE #DB QUERRY NAIVE USER INTERFACE ==='
1/-21,-22,-10,11,12,13,23,24,25/36,37,39,40,41,42,38/1 6 ' ==SET UP PSEUDOBATCH #DB QUERRY POSSIBILITY   ==='
1/26,2,27,28,29/42,43,16,17,18,19,20/1            7 ' ==SET UP ACCESSORIES FOR #ATS                           =='
1/30,1,2,31,32/21,22,23,24/1                      8 ' ==SET UP PRODUCTION CYCLE PROGRAMS                      =='
1/64/44/2                                         9 ' ==BODY OF THE BASIC SYSTEM IS ORGANIZED                 =='
2/33/25/2                                        10 ' ==#ATS DAILY DATA USAGE PARAMETER DEFINITION            =='
2/34,28,27/26/2                                  11 ' ==#ATS DAILY TRANSACTION DATA GENERATION                =='
2/35,36,2,37,30/27/2                             12 ' ==#ATS TRANSACTION DATA COLLECTION                      =='
2/38,30,39,40,41,42/28/2                         13 ' ==#ATS TRANSACTION DATA PROCESSING                      =='
2/39/29/2                                        14 ' ==#ATS DAILY DATA DISTRIBUTION                          =='
2/44,45,46,47,48/30/2                            15 ' ==#ATS DAILY OUTPUT USAGE                               =='
2/64/44/3                                        16 ' ==#ATS SYSTEM IS ORGANIZED                              =='
3/15,12,49,50/10/3                               17 ' ==UPDATE THE COURSE #MF                                 =='
3/15,12,51,50/11/3                               18 ' ==UPDATE THE STUDENT #MF                                =='
3/15,12,52,53,54,55/35,31,32/3                   19 ' ==UPDATE ATTENDANCE RELATED DATA                        =='
3/64/44/4                                        20 ' ==UPDATE ORGANIZATION IS COMPLETE                       =='
4/13,49,56,21,22/33/4                            21 ' ==QUERRY THE COURSE #MF INTERACTIVELY                   =='
4/13,49,52,51,57,58,59,21,22/34/4                22 ' ==QUERRY THE STUDENT #MF INTERACTIVELY                  =='
4/13,49,52,57,60,21,22/35/4                      23 ' ==QUERRY THE CLASS #MF INTERACTIVELY                    =='
4/56,61,49,-21,-22/36,37,38/4                    24 ' ==QUERRY THE COURSE #MF PSEUDOBATCH MODE                =='
4/56,62,49,-21,-22/39,40,38/4                    25 ' ==QUERRY THE STUDENT #MF PSEUDOBATCH MODE               =='
4/56,60,63,49,-21,-22/41,42,38/4                 26 ' ==QUERRY THE CLASS #MF PSEUDOBATCH MODE                 =='
4/64/44/0                                        27 ' ==DATA-BASE QUERRY ORGANIZATION IS COMPLETE             =='
? ?
```

```
PROCEDURE SYSTALL ( ACTNB:INTEGER);

BEGIN
        CASE ACTNB OF
            -1 : WRITE(' TERMINAL IS SET UP IN SCHOOL ');
            -2 : WRITE(' ALL DATA-BASE FILES ARE SET UP ON-LINE ');
            -3 : WRITE(' QUERRY AND UPDATE SUBSYSTEMS ARE SET UP ');
            -4 : WRITE(' ALLOCATED FIELDS IN THE DATA-BASE ARE FILLED ');
            -5 : WRITE(' DATA-BASE CONTENT IS  VERIFIED ');
            -6 : WRITE(' ERROR IN THE DATA-BASE ');
            -7 : WRITE(' ATTENDANCE MONITORING SUBSYSTEM IS SET UP ');
            -8 : WRITE(' DATA-BASE CONTAINS CORRECT NON TRUANCY INFORM. ');
            -9 : WRITE(' TRANSACTION DATA GENERATION KITS ARE READY ');
            -10 : WRITE(' TASK IS ATTENDANCE MONITORING ');
            -11 : WRITE(' PRODUCTION CYCLE IS STARTED ');
            -12 : WRITE(' PRODUCTION DAY IS INITIALIZED TO THE COMPUTER ');
            -13 : WRITE(' TRANSACTION DATA IS GENERATED ');
            -14 : WRITE(' TRANSACTION DATA IS COLLECTED ');
            -15 : WRITE(' DATA IS PROCESSED ');
            -16 : WRITE(' OUTPUT IS DISTRIBUTED ');
            -17 : WRITE(' USAGE PARAMETERS ARE DETERMINED ');
            -18 : WRITE(' TASK IS UPDATING THE DATA-BASE ');
            -19 : WRITE(' CHANGE INVOLVES NON #ATS. RELATED DATA ');
            -20 : WRITE(' CHANGE INVOLVES #ATS. RELATED DATA ');
            -21 : WRITE(' TASK IS QUERRY THE DATA-BASE ');
            -22 : WRITE(' STUDENT INFORMATION IS NEEDED ');
            -23 : WRITE(' COURSE INFORMATION IS NEEDED ');
            -24 : WRITE(' CLASS INFORMATION IS NEEDED ');
            -25 : WRITE(' SMALL AMOUNT OF OUTPUT IS NEEDED ');
            -26 : WRITE(' LARGE AMOUNT OF OUTPUT IS NEEDED ');
            -27 : WRITE(' END OF THE YEAR ');
             1 : WRITE('A1>  INSTALL THE TERMINAL IN SCHOOL ');
             2 : WRITE('A2>  SET UP THE COURSE MASTER FILE ');
             3 : WRITE('A3>  SET UP THE STUDENT MASTER FILE ');
             4 : WRITE('A4>  SET UP THE CLASS MASTER FILE ');
             5 : WRITE('A5>  SET UP COURSE MASTER FILE UPDATER PROGRAM ');
             6 : WRITE('A6>  SET UP STUDENT MASTER FILE UPDATER PROGRAM ');
             7 : WRITE('A7>  SET UP COURSE MASTER FILE QUERRY PROGRAM ');
             8 : WRITE('A8>  SET UP STUDENT MASTER FILE QUERRY PROGRAM ');
             9 : WRITE('A9>  SET UP CLASS MASTER FILE QUERRY PROGRAM ');
            10 : WRITE('A10>  UPDATE COURSE MASTER FILE INTERACTIVELY ');
            11 : WRITE('A11>  UPDATE STUDENT MASTER FILE INTERACTIVELY ');
            12 : WRITE('A12>  UPDATE CLASS MASTER FILE INTERACTIVELY ');
            13 : WRITE('A13>  VERIFY COURSE MASTER FILE DATA ');
            14 : WRITE('A14>  VERIFY STUDENT MASTER FILE DATA ');
            15 : WRITE('A15>  VERIFY CLASS MASTER FILE DATA ');
            16 : WRITE('A16>  PRODUCE TEACHERS DATA GENERATION KITS ');
            17 : WRITE('A17>  SET UP DATA COLLECTION ROUTES IN SCHOOL ');
            18 : WRITE('A18>  INSTALL OPTICAL MARK READER ON TERMINAL ');
            19 : WRITE('A19>  VERIFY DATA DISTRIBUTION ROUTES ');
            20 : WRITE('A20>  DETERMINE DATA USAGE PARAMETERS ');
            21 : WRITE('A21>  SET UP DAILY INITIALIZATION INTERACTIVE PROGRAM ');
            22 : WRITE('A22>  SET UP DATA COLLECTION PSEUDOBATCH PROGRAM ');
            23 : WRITE('A23>  SET UP  #ATS. DATA PROCESSING PSEUDOBATCH PROGRAM ');
            24 : WRITE('A24>  SET UP PROCEDURE TO PRINT SPOOLED OUTPUT ');
            25 : WRITE('A25>  PERFORM DAILY SYSTEM INITIALIZATION ');
            26 : WRITE('A26>  PERFORM TRANSACTION DATA GENERATION ');
            27 : WRITE('A27>  PERFORM TRANSACTION DATA COLLECTION ');
            28 : WRITE('A28>  PERFORM DATA PROCESSING (REPORTING AND STORAGE) ');
            29 : WRITE('A29>  PERFORM DATA DISTRIBUTION ');
            30 : WRITE('A30>  VERIFY OUTPUT USAGE IN DECISION MAKING ');
            31 : WRITE('A31>  PUNCH NEW # ABS-CARDS ');
            32 : WRITE('A32>  UPDATE TEACHERS DATA GENERATION KITS ');
            33 : WRITE('A33>  QUERRY COURSE MASTER FILE INTERACTIVELY ');
            34 : WRITE('A34>  QUERRY STUDENT MASTER FILE INTERACTIVELY ');
            35 : WRITE('A35>  QUERRY CLASS MASTER FILE INTERACTIVELY ');
            36 : WRITE('A36>  WRITE A FILE IN COURSE MASTER FILE QUERRY LANGUAGE ');
            37 : WRITE('A37>  QUERRY COURSE MASTER FILE PSEUDOBATCH MODE ');
            38 : WRITE('A38>  PRINT SPOOLED OUTPUT ');
            39 : WRITE('A39>  WRITE A FILE IN STUDENT MASTER FILE QUERRY LANGUAGE ');
            40 : WRITE('A40>  QUERRY STUDENT MASTER FILE PSEUDOBATCH MODE ');
            41 : WRITE('A41>  WRITE A FILE IN CLASS MASTER FILE QUERRY LANGUAGE ');
            42 : WRITE('A42>  QUERRY CLASS MASTER FILE PSEUDOBATCH MODE ');
            43 : WRITE('A43>  PRODUCE # ABS-CARDS FOR THE CLASS ');
            44 : WRITE('A44>  NO ACTION- NEXT CLUSTER ');
             0 : WRITELN;
        END (*CASE *)
END;  (* SYSTALL*)
> ?
```

```
SYSTALL       (M-3,P-5)
51
*****************************************************************************
*                                                                           *
*          SYSTALL  # (OR THE OVERALL DESCRIPTION OF THE SYSTEM)            *
*          ------------------------------------------------                 *
*                                                                           *
*  THIS SEQUENCE OF PROCEDURES PROVIDES AN OVERVIEW OF THE ENTIRE           *
*  ATTENDANCE MONITORING SYSTEM  (#  ATS.  ).  IT SPECIFIES HOW TO          *
*  SET UP, RUN AND USE THIS SYSTEM.  IT SHOWS THE LOGICAL INTER-            *
*  RELATION AND OPERATIONAL SEQUENCE OF THE SUBSYSTEMS AMONG EACH           *
*  OTHERS.  FIRST WE SET UP THE BASIC SYSTEM, THAT IS THE DATA-BASE (#DB)   *
*  WITH ITS QUERRY (#QUS.) AND UPDATE (#UPS.) UTILITY SUBSYSTEMS.           *
*  SECOND WE GRAFT TO IT THE ATTENDANCE MONITORING SUBSYSTEM (#ATS.).       *
*  FROM THEN ON WE CONCURRENTLY MAINTAIN THE EXACTNESS OF THE DATA-BASE     *
*  WHILE ROUTINELY RUNNING THE ATTENDANCE MONITORING SUBSYSTEM              *
*  IN ORDER TO EXTRACT THE RELATED INFORMATION FROM THE DATA-BASE.          *
*                                                                           *
*                     SYSTALL ACTION(1..44)                                 *
*  SYSTALL HAS 44 ACTIONS.  EACH OF THESE IS A COMPLEX ACTION               *
*  PROCEDURE HAVING THE POSSIBILITY OF BEING DECOMPOSED INTO                *
*  SUB-PROCEDURES.  IN ORDER TO BE ABLE TO ESTABLISH CLEARLY                *
*  THE HIERARCHIAL RELATIONSHIP BETWEEN THE SUB-ACTION AND                  *
*  THE ACTIONS OF SYSTALL THE FOLLOWING NOTATION WILL BE USED:              *
*                                                                           *
*           SYSTALL ACTION(N) = ACTION(N) = AN> XXXXXXXXXX                  *
*                                                                           *
*           SYSTALL SUB-ACTION(M) OF ACTION(N) =                            *
*                  = SYSTALL ACTION(N-M) = AN-M> XXXXXXXXXX                 *
*                                                                           *
*           SYSTALL SUB-SUBACTION(L) OF SUBACTION(M) OF ACTION(N) =         *
*                  = SYSTALL ACTION(N-M-L) = AN-M-L> XXXXXXXXXX             *
*                                                                           *
*     #  FOR ALL CONTRACTIONS PLEASE REFER TO THE TABLE ON FIGURE 22.       *
*                                                                           *
*           THE PRODUCTION CYCLE OF # ATS.  IS DESCRIBED IN DETAIL          *
*           IN THE FOLLOWING ACTIONS:                                       *
*                                                                           *
*          SYSTALL A(25) = DAILY SYSTEM INITIALIZATION  #                   *
*          SYSTALL A(26) = DAILY TRANSACTION DATA GENERATION                *
*          SYSTALL A(27) = DAILY TRANSACTION DATA COLLECTION                *
*          SYSTALL A(28) = DAILY TRANSACTION DATA PROCESSING                *
*          SYSTALL A(29) = DAILY DATA DITRIBUTION                           *
*          SYSTALL A(30) = DAILY DATA USAGE VERIFICATION                    *
*                                                                           *
*                                                                           *
*              AN EXAMPLE OF THE COMPLEXITY OF SOME ACTIONS                 *
*              IS ILLUSTRATED IN SYSTALL A(11), WHOSE LOGICAL               *
*              DESCRIPTION IS 3 LEVELS DEEP.                                *
*                                                                           *
*          SYSTALL A(11) = UPDATE THE STUDENT MASTER FILE                   *
*                                                                           *
*****************************************************************************
6 27 44
1*C1 SET UP THE BASIC SYSTEM ( # UTS. AND # DB. )                      ===*
2*C2 INITIAL DATA-BASE VERIFICATION. ( # QUS. AND # UPS. )             ===*
3*C3 SET UP THE ATTENDANCE MONITORING SUBSYSTEM ( # UTS. )             ===*
4*C4 DO REGULAR RUN OF THE ATTENDANCE MONITORING SUBSYSTEM (# ATS.)    ===*
5*C5 MAINTAIN THE EXACTNESS OF THE SYSTEMS DATA  ( # UPS. )           ===*
6*C6 INFORMATION EXTRACTING AND REPORTING ( #QUS. )                   ===*
13  26
1/-1/1/1                          1 '== INSTALL TERMINAL IN SCHOOL
1/1,-2/2,3,4/1                    2 '== SET UP EMPTY DATA-BASE
1/2,-3/5,6,7,8,9/1                3 '== SET UP ALL #QUS. AND #UPS. INTERACTIVE PROGRAMS.
1/2,3,-4/10,11,12/2               4 '== UPDATE ALLOCATED FIELDS IN DATA-BASE
2/2,-5/37,13,40,14,42,15/2        5 '== VERIFY INFORMATION IN DATA-BASE
2/5,6/10,11,12/3                  6 '== UPDATE INCORRECT INFORMATION IN DATA-BASE
2/5,-6/44/3                       7 '== NO UPDATE IS NECESSARY
3/-7,8,-9/42,43,16,17,18,19,20/3  8 '== GIVE OUT ACCESSORIES FOR ATTEND.MONIT.SUBSYSTEM.
3/-7,9/21,22,23,24/4              9 '== SET UP #ATS.INTERACT.PROGS.FOR DAILY PRODUCTION CYCLE
4/10,7,-11,17/25/4               10 '== #ATS. DAILY DATA USAGE PARAMETER DEFINITION
4/10,11,12/26/4                  11 '== #ATS. DAILY TRANSACTION DATA GENERATION
4/10,11,13/27/4                  12 '== #ATS. DAILY TRANSACTION DATA COLLECTION
4/10,11,14/28,38/4               13 '== #ATS. DAILY TRANSACTION DATA PROCESSING
4/10,11,15/29/4                  14 '== #ATS. DAILY DATA DISTRIBUTION
4/10,11,16,-17/30,20/4           15 '== #ATS. DAILY OUTPUT USAGE
4/-10/44/5                       16 '== NO ACTION,NEXT CLUSTER
5/18,19/10,11,12/4               17 '== UPDATE NON ATTENDANCE RELATED DATA
5/18,20/10,11,12,33,31,32/4      18 '== UPDATE ATTENDANCE RELATED DATA
5/-18/44/6                       19 '== NO ACTION, NEXT CLUSTER
6/21,23,25/33/4                  20 '== QUERRY COURSE MASTER FILE INTERACTIVELY
6/21,22,25/34/4                  21 '== QUERRY STUDENT MASTER FILE INTERACTIVELY
6/21,24,25/35/4                  22 '== QUERRY CLASS MASTER FILE INTERACTIVELY
6/21,23,26/36,37,38/4            23 '== QUERRY COURSE MASTER FILE PSEUDOBATCH
6/21,22,26/39,40,38/4            24 '== QUERRY STUDENT MASTER FILE PSEUDOBATCH
6/21,24,26/41,42,38/4            25 '== QUERRY CLASS MASTER FILE PSEUDOBATCH
6/-21,27/44/0                    26 '== END OF THE YEAR
> ?
```