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.

(c) Murray N. Kronick, 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, JAW81].

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

ABSTRACT ........................................ iii
ACKNOWLEDGEMENTS ................................ iv
LIST OF ILLUSTRATIONS ....................... vii

I. INTRODUCTION ................................... 1

Scope and Purpose
   Basis of a Software Environment
   The Selection of ABL

II. THE SOFTWARE DEVELOPMENT PROCESS .... 6

   Requirements Definition
   Preliminary Design
   Detailed Design
   Programming and Development
   Testing and Implementation
   Operations and Maintenance
   Performance Measurement and Evaluation

III. ABL—AN ALTERNATIVE BASED LANGUAGE .. 13

   The Abstract Program, Abstract Machine
   Backus-Naur Form of ABL
   The ABL Process
   ABL Templates
   Characteristics and Principles of ABL

IV. ABL AS A TOOL IN THE SOFTWARE DEVELOPMENT PROCESS 31

   Applicability to the Seven Development Stages
   Features of the ABL Software tool
V. THE SOFTWARE ENVIRONMENT FOR ABL

Implementation: The Host System
Description of Implemented System
ABL Interactive Interpreter

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

The Existing Environment
Problem Definition
Specification of the Automatic Dialing System
Host System for Implementation

VII. CONCLUSION

Summary and Conclusion
Future Research
Observations

APPENDIX

ABL Interactive Interpreter Listing
in Implementation Language (BASIC)

LIST OF REFERENCES
LIST OF ILLUSTRATIONS

Figure 1. Model of a Software Environment .................................................. 4
Figure 2. The Stages of Software Development ............................................... 6
Figure 3. Backus-Naur Form of ABL .............................................................. 15-18
Figure 4. The ABL Process—Calculator Example ........................................... 24
Figure 5. The ABL Process in ABL form ......................................................... 25
Figure 6. ABL Templates ................................................................................. 27
Figure 7. ABL Environment System Diagram ................................................... 38
Figure 8. ABL Compute Form Listing ............................................................... 40
Figure 9. ABL Compact Listings ...................................................................... 41
Figure 10. ABL Machine Listing ..................................................................... 42
Figure 11. ABL Program and Machine Library Listing ....................................... 43
Figure 12. ABL Interactive Interpreter Listings .............................................. 46-50
Figure 13. ABL Interactive Interpreter Screen ................................................ 52
Figure 14. Automatic Dialing System for Appointment Confirmations .............. 57-58
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" [Dijkstra74].

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 consensus 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, MORV82, JAWO81, LEBE82a, LINAB82, MORO81], was selected and further developed to serve as the kernel for the software environment.

It must be emphasized that the intention 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, OBIE80, PETE78].

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

Requirements Definition

<table>
<thead>
<tr>
<th>Preliminary Design (System Overview)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detailed Design (Specifications)</td>
</tr>
<tr>
<td>Programming and Development</td>
</tr>
<tr>
<td>Testing and Implementation</td>
</tr>
<tr>
<td>Operations and Maintenance</td>
</tr>
<tr>
<td>Performance Measurement and Evaluation</td>
</tr>
</tbody>
</table>

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 its 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 print-outs as both documentation and operations manual.

Maintenance can be thought of as accommodating 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, PAR79]. 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 modeling 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) [JEN574, KOR74]. 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

- 16 -
<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 IMPLEMENTATION CHARACTER SET.

** NOTE 2: THE IMPLEMENTATION LANGUAGE STATEMENTS ARE DEFINED 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 \( \{P_j\} \).

\( \text{Program O1 in the example} \)

4.0 For each \( P_j \), define the set of sub-tasks within the module, called steps or clusters \( \{C_j\} \).

\( \text{labelled C1, C2, C3} \)

5.0 For each \( C_j \), define the set of alternatives \( \{AL_j\} \) within the cluster.

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

6.0 For each \( AL_j \),

6.1 Define the set of data input objects \( \{DI_j\} \).

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

\( \text{the asterisks * for DI and D2 indicate input references to relevant data objects} \)

6.2 Define the set of preconditions \( \{PR_j\} \).

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 (ACj).

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 \( \{P_{D_j}\} \). 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 \( S_{11} \) and \( S_{12} \).

6.5 Define the set of data output objects \( \{D_{O_j}\} \). 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 \( N_j \).

If all relevant postconditions in the alternative are satisfied, then control is transferred to the cluster number indicated by \( N_j \).

If \( N_j = 0 \), then we terminate the current abstract program, resume the next higher level of abstraction by transferring control back to the invoking abstract program, reinstating its 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 com-
monly 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 exhaus-
tive 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: DA 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 DECIDE
   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 V DETERMINE CHARACTER ENTERED AND DECIDE
C3  V   ERROR CONDITIONS

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

PRECONDITIONS
P1  ** ** ** ** ** CHAR = 0, 9
P2  ** ** ** ** ** CHAR = '+'
P3  ** ** ** ** ** CHAR = '-'
P4  ** ** ** ** ** CHAR = '*'
P5  ** ** ** ** ** CHAR = '/'
P6  ** ** ** ** ** BUFFER = 0
P7  ** ** ** ** ** CHAR = 'm'

ACTIONS
A1  ** ** ** ** ** ** ** ** ACCUMULATOR = 0 C1
A2  ** ** ** ** ** ** ** ** CLEAR INPUT BUFFER & L.E.D. DISPLAY C2
A3  ** ** ** ** ** ** ** ** ACCEPT NEXT CHAR FROM KEYPAD C3
A4  ** ** ** ** ** ** ** ** PLACE DIGIT OR DECIMAL POINT INTO BUFFER C4
A5  ** ** ** ** ** ** ** ** ADD CONTENTS OF BUFFER INTO ACCUMULATOR C5
A6  ** ** ** ** ** ** ** ** SUBTRACT CONTENTS OF BUFFER FROM ACCUMULATOR C6
A7  ** ** ** ** ** ** ** ** MULTIPLY CONTENTS OF BUFFER TIMES ACCUMULATOR C7
A8  ** ** ** ** ** ** ** ** DIVIDE CONTENTS OF BUFFER INTO ACCUMULATOR C8
A9  ** ** ** ** ** ** ** ** PRINT CONTENTS OF ACCUMULATOR C9
A10 ** ** ** ** ** ** ** ** PRINT ERROR MESSAGE C10

POSTCONDITIONS
S11 ** ** ** ** ** ** BUFFER VALUE > 999999.99 (OVERFLOW)
S12 ** ** ** ** ** ** 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 0

Figure 4. The ABL Processor 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

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>C2</td>
<td>C3</td>
<td>C4</td>
<td>C5</td>
<td>C6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CLUSTERS / ALTERNATIVES

NAME THE PROJECT AND PERFORM A PRELIMINARY INVESTIGATION

PRECONDITIONS

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

ANY MORE DATA INPUT OBJECTS?
ANY MORE PRECONDITIONS?
ANY MORE ACTIONS?
ANY MORE POSTCONDITIONS?
ANY MORE DATA OUTPUT OBJECTS?

ACTIONS

<table>
<thead>
<tr>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
<th>A6</th>
<th>A7</th>
<th>A8</th>
<th>A9</th>
<th>A10</th>
<th>A11</th>
<th>A12</th>
<th>A13</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

NAME THE PROJECT
PERFORM A PRELIMINARY INVESTIGATION
DEFINE THE PROJECT BOUNDARIES
DIVIDE THE PROJECT INTO DISCRETE MODULES (ABSTRACT PROGRAMS)
DIVIDE THE ABSTRACT PROGRAMS INTO STEPS (CLUSTERS)
DIVIDE THE CLUSTERS INTO ALTERNATIVES
DEFINE THE DATA INPUT OBJECTS
DEFINE THE PRECONDITIONS
DEFINE THE ACTIONS
DEFINE THE POSTCONDITIONS
DEFINE THE DATA OUTPUT OBJECTS
DETERMINE THE NEXT STEP
DETERMINE THE EXCEPTION STEP

POSTCONDITIONS

<table>
<thead>
<tr>
<th>S11</th>
<th>S12</th>
<th>S13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

AFTER PRELIMINARY INVESTIGATION: CONTINUE THE PROJECT?
IS THIS STEP COMPLETE?
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" [TEIT81].

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 automation 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" [BELABO]. 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 noncorrepondance of data types [OST80]. 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 non-trivial, 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 accommodate 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. Schemas such as the Communication Port (CP) [MACBO] 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 compu-tuts 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 : COMPUTE  INITIALIZE ON POWER-UP OR CLEAR

[ ]

ACCUMLATOR = 0
CLEAR INPUT BUFFER & L.E.D. DISPLAY
ACCEPT NEXT CHAR FROM KEYPAD

[ ] 2 EXCEPTION 0

2 : COMPUTE  DETERMINE CHARACTER ENTERED AND DECODE

[ ]

CHAR = 0.9
CHAR = +
CHAR = -
CHAR = *
CHAR = /
BUFFER = 0
CHAR = =

[ ]

PLACE DIGIT OR DECIMAL POINT INTO BUFFER
ACCEPT NEXT CHAR FROM KEYPAD

[ BUFFER VALUE ] 999999.99 (OVERFLOW)

[ ] 2 EXCEPTION 3

[-,-,-,-,-,-] : ADD CONTENTS OF BUFFER INTO ACCUMULATOR

CLEAR INPUT BUFFER & L.E.D. DISPLAY
ACCEPT NEXT CHAR FROM KEYPAD

[ ACCUMULATOR VALUE ] 999999.99 (OVERFLOW)

[ ] 2 EXCEPTION 3

[-,-,-,-,-,-] : SUBTRACT CONTENTS OF BUFFER FROM ACCUMULATOR

CLEAR INPUT BUFFER & L.E.D. DISPLAY
ACCEPT NEXT CHAR FROM KEYPAD

[ ACCUMULATOR VALUE ] 999999.99 (OVERFLOW)

[ ] 2 EXCEPTION 3

[-,-,-,-,-,-] : MULTIPLY CONTENTS OF BUFFER TIMES ACCUMULATOR

CLEAR INPUT BUFFER & L.E.D. DISPLAY
ACCEPT NEXT CHAR FROM KEYPAD

[ ACCUMULATOR VALUE ] 999999.99 (OVERFLOW)

[ ] 2 EXCEPTION 3

[-,-,-,-,-,-] : DIVIDE CONTENTS OF BUFFER INTO ACCUMULATOR

CLEAR INPUT BUFFER & L.E.D. DISPLAY
ACCEPT NEXT CHAR FROM KEYPAD

[ ACCUMULATOR VALUE ] 999999.99 (OVERFLOW)

[ ] 2 EXCEPTION 3

[-,-,-,-,-,-] : PRINT CONTENTS OF ACCUMULATOR

[ ] 1 EXCEPTION 0

3 : COMPUTE  ERROR CONDITION

[ ]

PRINT ERROR MESSAGE

[ ] 1 EXCEPTION 0

Figure 8. ABL Comput Form Listing - 40 -
**COMPACT ABSTRACT PROGRAM OF ALTERNATIVES**

**ALTERNATIVE FORMAT IS:**  (HORIZONTALLY)

**CLUSTER (ALTERNATIVE) / PRECONDITIONS / ACTIONS / POSTCONDITIONS / NEXT STEP / EXCEPTION STEP**

<table>
<thead>
<tr>
<th>No.</th>
<th>Cluster</th>
<th>Precondition</th>
<th>Action</th>
<th>Postcondition</th>
<th>Next Step</th>
<th>Exception Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/0/1,2,3/0/2/0</td>
<td></td>
<td>INITIALIZE PROCEDURE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1/1/4,3/-1/2/3</td>
<td></td>
<td>NUMERIC ENTERED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2/2/5,2,3/-12/2/3</td>
<td></td>
<td><code>/</code> OPERAND ENTERED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3/3/6,2,3/-12/2/3</td>
<td></td>
<td><code>/</code> OPERAND ENTERED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4/4/7,2,3/-12/2/3</td>
<td></td>
<td><code>/</code> OPERAND ENTERED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5/5/-6/8,2,3/-12/2/3</td>
<td></td>
<td><code>/</code> OPERAND ENTERED, NON-ZERO BUFFER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6/5/6/0/0/3/3</td>
<td></td>
<td><code>/</code> OPERAND ENTERED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>7/7/9/0/1/3</td>
<td></td>
<td><code>/</code> OPERAND ENTERED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1/0/10/0/1/0</td>
<td></td>
<td>PRINT ERROR MESSAGE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 9a. ABL Compact Abstract Program of Alternatives Listing**

**COMPACT ABSTRACT PROGRAM TABLE**

**TABLE FORMAT IS:**  (HORIZONTALLY)

**CLUSTER (ALTERNATIVE) / PRECONDITIONS / O / ACTIONS / O / POSTCONDITIONS / O / NEXT STEP / EXCEPTION STEP**

<table>
<thead>
<tr>
<th>No.</th>
<th>Cluster</th>
<th>Precondition</th>
<th>Action</th>
<th>Postcondition</th>
<th>Next Step</th>
<th>Exception Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>20</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>-12</td>
</tr>
<tr>
<td>23</td>
<td>30</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>24</td>
<td>40</td>
<td>0</td>
<td>4</td>
<td>7</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>25</td>
<td>50</td>
<td>0</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>26</td>
<td>60</td>
<td>0</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>70</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>80</td>
<td>0</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>90</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 9b. ABL Compact Abstract Program Table Listing**
PORTABLE ELECTRONIC CALCULATOR MACHINE

PREDICATES
1. \(\text{CHAR} = 0\), \(\text{CHAR} = 9\)
2. \(\text{CODE} = \text{IF CHAR} < 9\), \(\text{TIME (msec)} = 10\), \(\text{DATA OBJECT INPUTS} = 3\), \(\text{OUTPUTS} = 1\)
3. \(\text{CHAR} = \ast\), \(\text{CHAR} = \ast\)
4. \(\text{CODE} = \text{IF CHAR} > 9\), \(\text{TIME (msec)} = 10\), \(\text{DATA OBJECT INPUTS} = 3\), \(\text{OUTPUTS} = 1\)
5. \(\text{CHAR} = \ast\), \(\text{CHAR} = \ast\)
6. \(\text{CODE} = \text{IF BUF} = 0\), \(\text{TIME (msec)} = 10\), \(\text{DATA OBJECT INPUTS} = 3\), \(\text{OUTPUTS} = 1\)

ACTION'S
1. \(\text{ACCU} = 0\)
2. \(\text{CLEAR INPUT BUFFER & L.E.D. DISPLAY}\)
3. \(\text{ACCEPT NEXT CHARACTER FROM KEYPAD}\)
4. \(\text{PLACE DIGIT OR DECIMAL POINT INTO BUFFER}\)
5. \(\text{ADD CONTENTS OF BUFFER INTO ACCUMULATOR}\)
6. \(\text{SUBTRACT CONTENTS OF BUFFER FROM ACCUMULATOR}\)
7. \(\text{MULTIPLY CONTENTS OF BUFFER TIMES ACCUMULATOR}\)
8. \(\text{DIVIDE CONTENTS OF BUFFER INTO ACCUMULATOR}\)
9. \(\text{PRINT CONTENTS OF ACCUMULATOR}\)
10. \(\text{PRINT ERROR MESSAGE}\)

DATA OBJECTS
1. \(\text{ACCU}\)
2. \(\text{BUF}\)
3. \(\text{CHAR}, \text{INPUT FROM KEYPAD}\)

Figure 10: ABL Machine Listing
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 SET 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 accommodate 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: 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 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 ? (PRECONDITIONS)
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 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 --- END OF INPUT TYPE ? ('.' ENTERED, ADVANCE TO NEXT TYPE)

NEXT 2 3 3 3 3 3 3 0 3
EXP 1 3 4 4 4 3 3 0 3

Figure 12b. Edit Program - 47 -
MACHINE PA ABL SET PARAMETERS MACHINE
PROGRAM: OI ABL INTERACTIVE INTERPRETER- PARAMETERS PROGRAM (LEVEL OI)

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 (9) 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 . . . . . . . . . . . . SET TRACE
C4 . . . . . . . . . . . . SET BREAKPOINT
C5 . . . . . . . . . . . . CLEAR TRACE
C6 . . . . . . . . . . . . CLEAR BREAKPOINT

PRECONDITIONS
P1 -- Y -- -- N -- -- SET 'T' RACE ON ?
P2 -- Y -- -- N -- -- SET 'B' RIVERPOINT ON ?
P3 -- Y -- -- N -- -- CLEAR TRACE ?
P4 -- Y -- -- N -- -- CLEAR BREAKPOINT ?
P5 -- Y -- -- N -- -- END 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 . . . . . . . . . . . . SET TRACE FOR SPECIFIED ALTERNATIVE
A6 . . . . . . . . . . . . PRINT "T" IN ALTERNATIVE COLUMN ON SCREEN
A7 . . . . . . . . . . . . SET BREAKPOINT FOR SPECIFIED ALTERNATIVE
A8 . . . . . . . . . . . . PRINT "B" IN ALTERNATIVE COLUMN ON SCREEN
A9 . . . . . . . . . . . . CLEAR TRACE FOR SPECIFIED ALTERNATIVE
A10 . . . . . . . . . . . . ERASE "T" FROM ALTERNATIVE COLUMN ON SCREEN
A11 . . . . . . . . . . . . CLEAR BREAKPOINT FOR SPECIFIED ALTERNATIVE
A12 . . . . . . . . . . . . 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

Figure 12c. Monitor Parameters Program
Figure 121. 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  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
A11 5 3 IF TRACE MODE, PRINT ACTION & VARIABLES & VALUES
A12 5 4 INCREMENT ACTION # (ACTION # = ACTION # + 1)
A13 1 PARSE TO DECIDE OP-CODE IN IMPLEMENTATION LANGUAGE
A14 2 PARSE OPERANDS & EXECUTE ACTION IN IMPLEMENTATION LANGUAGE
A15 1 EVALUATE POSTCONDITIONS FOR THE ALTERNATIVE
A16 1 CURRENT CLUSTER = NEXT STEP #
A17 2 2 IF BREAKPOINT, PAUSE FOR USER INTERACTION
A18 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 WANT TO RESUME EXECUTION?

NEXT: 2 3 4 3 6 2 2

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 alternative. 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 'No 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" [ELLIBO].

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       (e.g. Monday, 09:00)
   ENDING DATE/TIME          (e.g. Tuesday, 17:00)
   UP TO n PHONE CALLS       (e.g. 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 unsuccessful 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.
CLUSTER 1.0 SYSTEM STARTUP
   SYSTEM STARTUP

CLUSTER 2.0 SELECT OFFICE OR HOME PHONE NUMBER
   (2) PREFERENCE IS TO CALL PATIENT'S OFFICE NUMBER
   (3) PREFERENCE IS TO CALL PATIENT'S HOME NUMBER

CLUSTER 3.0 DETERMINE THE STATUS OF THE TELEPHONE AFTER AUTO-DIALING
   (4) LINE IS BUSY
   (5) THERE IS NO ANSWER
   (6) A CONNECTION HAS BEEN MADE (SOMEONE ANSWERS THE PHONE)

CLUSTER 4.0 RECORD THE RESULT OF THE CONVERSATION
   (7) SUCCESSFUL CONFIRMATION OF THE APPOINTMENT
   (8) PATIENT WAS NOT REACHED, CONSIDERED AS 'NO ANSWER'
   (9) OTHER CHANGE IN THE PATIENT’S FILE IS NEEDED

CLUSTER 5.0 NO MORE PATIENTS TO CALL
   (10) BOTH QUEUES ARE EMPTY. ALL COMPLETE
   (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 IS THE LINE BUSY?
P6 N Y N IS THERE A NO ANSWER (6 RINGS)?
P7 N N Y IS THERE A CONNECTION (SOMEONE AT OTHER END PICKS UP PHONE)?
P8 Y N NO ANSWER HAD A SUCCESSFUL CONFIRMATION OF THE APPOINTMENT?
P9 Y N WAS THE PATIENT HIMSELF CONTACTED?
P10 N Y ARE THERE ANY 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 GET NEXT PATIENT SUB-PROGRAM (02/0A)
A3 1 SELECT OFFICE PHONE NUMBER TO CALL
A7 2 2 DIAL TELEPHONE NUMBER THROUGH EPAX PHONE SYSTEM
A6 1 SELECT HOME PHONE NUMBER TO CALL
A8 1 LAST RETRY TYPE = BUSY
A10 2 2 INCREASE 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 ARE THERE ANY MORE PATIENTS TO CALL?

NEXT 2 3 2 2 4 2 2 2 EXP 5 3 3 3 5 5 5 0

Figure 14a. Automatic Dialing System for Appointment Confirmations
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 ELIGIBILITY 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 ELIGIBILITY 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 -- Y N 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 1 . . . 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 VAB11 Multimode Automatic Calling Unit (ACU) [RACAB1], 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 accommodate concurrent processing has been designed into the software environment, along with the possibility 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 [JAWO70].

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

MURRAY KRONICK 742S057
THESIS WORK-MASTER OF COMPUTER SCIENCE
CONCORDIA UNIVERSITY
PROPRIETARY INFORMATION
WRITTEN 26/12/81
REVISED

LET R=30
LET C=20

--- CONSTANTS ---
C9 CURRENT CLUSTER NUMBER
R,C MAXIMUM NUMBER OF ROWS, COLUMNS IN T TABLE
T COMPACTED ABSTRACT PROGRAM TABLE (SEE FILE LAYOUT ABLTXYY)
J,J LOOP CONTROL
D DISPLACEMENT IN T TABLE (USED TO TRANSFER TO & FROM H7 IN FILE)
P POSITION IN T TABLE (COLUMN NUMBER) (0..C-1)
T1 TYPE OF INPUT (1=PRECONDITION 2=ACTION)
T1 B TYPE OF INPUT CODES (P/A/S/N/E)
B10 BREAKPOINT VECTOR [Y/N]
B20 TRACE VECTOR [Y/N]

DIM T1*(5), B1*(20), B2*(20), B3*(1)
SHORT V,C9,T(1..C-1), S
LONG I,J,D,P,T1
LET T1="PASNE"

DIM M9(4), M18(4), M28(2), M38(60), M38(70), M6(4)
SHORT M1, M2(10), M3, M4(9), M5(10), M7(9,19)
LET M3="*,M3"
LET M3="*,M3"

0090 GDUS 9300
0910 SCRN U
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 QOTO 1400
*** MAIN PROGRAM ***

1000 PRINT CAT(2)
1002 ON ERROR STOP
1004 ON ERROR GOTO 1008
1009 CLOSE FILE (5)
1004 CLOSE FILE (6)
1008 ON ERROR STOP

GET PROGRAM NUMBER

1010 LET X1=9
1015 LET X2=1
1020 LET X3=2
1020 GOSUB 9340

1040 ON Z2 THEN GOTO 1030.2050.1030.1030
1045 IF Z1>2 THEN GOTO 1030
1050 LET M3$(1,4)=Z6
1059 GET MACHINE NUMBER

1110 ON ERROR GOTO 1000
1110 LET X1=21
1110 GOSUB 9340

1140 ON Z2 THEN GOTO 1130.1000.1160.1130
1150 LET M3$(1,4)=Z6
1155 LET M3=M3$(1,2)
1160 LET M1=0

1170 ON ERROR GOTO 1100
1175 SEARCH INDEX (1.2).M5=M1 DIVIDING M1.$M2
1180 ON ERROR STOP
1190 PRINT PC((21,1))M5(1,2)$M1$(1,4),1

SEE IF PROGRAM ALREADY EXISTS

1199 LET Z5="ABLP".M5$(1,4)
1210 ON ERROR GOTO 1000
1220 OPEN FILE (5.3).Z6
1230 LET Z5(1,4)="ABLP"
1240 OPEN FILE (6.512).Z6
1260 ON ERROR STOP

PROGRAM EXISTS. PRINT FIRST TWO LINES OF DESCRIPTION

1299 LET M2=""
1310 FOR M3=1 TO 2
1320 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

READ IN STORED COMPACTED TABLE INTO T

1399 FOR Z=1 TO 3
1410 READ FILE (6.2).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

1300 IF 'Y' THEN GOTO 2000
1310 LET B1$="N".B1$
1320 LET B2$="N".B2$
1330 LET S=0
1340 GOSUB 6000

EDIT [E], SET PARAMETERS [P], EXECUTE [X], SAVE [S], OR QUIT [Q]

2000 LET Z=100
2010 GOSUB 9800
2030 ON -Z THEN GOTO 3000, 7000, 2500, 2100, 2050
2050 CHAIN "TYU-810"

2099 SAVE PROGRAM

2100 FOR I=1 TO 3
2110 LET H5(I)=Z
2120 LET D=(I-1)*10
2130 FOR I=0 TO 9
2140 FOR J=0 TO C-1
2150 LET H7(I,J)=T(I*D,J)
2160 NEXT J
2170 NEXT I
2180 WRITE FILE (6, Z), H5, H3$.H6, H7
2190 NEXT I
2200 GOTO 1000

2499 CHAIN TO EXECUTE MODULE

TRACE OUTPUT TO PRINTER? [Y/N] "YN"

2500 LET Z=125
2510 GOSUB 9800
2520 LET B3$="N"
2530 ON -Z THEN GOTO 2530, 2600
2550 LET B3$="Y"
2560 OPEN FILE (15, 2), "SYSD"
2600 CHAIN "ABLA-06C", B1$, B2$, B3$
"ABLA-068" ABL INTERACTIVE INTERPRETER - LOAD & EDIT PROGRAM, SET PARAMETERS

```plaintext
2998
2999
3000 ON ERROR STOP
3010 LET II=12
3015 LET I2=4
3020 LET I3=2
3030 GOSUB 9620
3040 ON II THEN GOTO 3030, 2000, 3030, 3100
3050 LET M3.C9=I
3060 ON ERROR GOTO 3000
3070 SEARCH INDEX (5, 6), "A", M3 GIVING M3, M4
3080 ON ERROR STOP
3090 GOTO 3115

3099 LET M5(2)=M5(2)+1
3110 LET C9, M3=M5(2)
3115 PRINT PC(13, 4)
3117 PRINT USING "8x. B", M3
3119
3120 LET I1=19
3125 LET I3=60
3140 GOSUB 9540
3150 ON II THEN GOTO 3140, 3170, 3170, 3140
3160 LET M3=M5
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"

3300 LET I1=12
3310 LET I2=5
3320 LET I3=4
3330 GOSUB 9600
3340 ON II THEN GOTO 3330, 3030, 3330, 3330
3350 LET Y9, I3=1
3360 GOSUB 9720
3365 IF Y9=0 THEN GOTO 3350
3370 IF INT(I/10)>C9 THEN GOTO 3350
3375 IF INT(I/10)=I/10 THEN GOTO 3350
3380 LET M3=I
3390 FOR I=0 TO 9
3392 LET M4(I)=0
3394 NEXT I
3600 ON ERROR GOTO 3620
3610 SEARCH INDEX (5, 6), "B", M3 GIVING M5, M4
3620 ON ERROR STOP
3625 PRINT PC(13, 5)
3627 PRINT USING "8x. B", M3/10:
3699 GET "ALTERNATIVE DESCRIPTION"

3700 LET II=19
```
3710 LET I=0
3720 LET I=I+1
3730 GOSUB 9540
3740 ON I2 THEN GOTO 3730.3500.3760.3730
3750 LET M3=I2
3760 PRINT PC(I2.9);M3
3770 ON ERROR GOTO 3790
3780 INSERT INDEX (5.6);"B".M3 USING M3.14
3790 DELETE INDEX (5.6);"B".M3 USING M3.14
3870 PRINT PC(P+4.B);I2
3900 FOR P=O TO M3(3)
3910 IF T(I.P)=O THEN GOTO 4120
3920 IF T(I.P)=M3 THEN GOTO 4200
3930 IF T(I.P)=M3 THEN GOTO 4100
3940 NEXT P
3950 GOSUB 3500
3960 LET M3(3)=M3(3)+I2
3970 LET M3(I.P)=M3
3980 CHANGE M3 TO Z8 USING "B.B"
3990 PRINT PC(P+4.B)
4000 INPUT I2=O
4010 LET X1=X2=0
4020 LET X2=9
4030 LET X3=1
4040 LET X4=I2=1
4050 FOR I=1 TO R-1
4060 ON I1 THEN GOSUB 9610.9620.9610.9620.9610.9620
4070 ON I2=I THEN GOTO 4350.4320.4340.4330.4345.4350
4080 LET Z=T(I.P)
4090 IF Z=0 THEN IF T(I.C)=Z THEN GOTO 4500
4100 LET T(I.P)=C
4110 CHANGE Z TO Z8 USING "BB-B"
4120 LET Z8(I.1)=T8(I1.T1)
4130 PRINT PC(X1.X2.X3)
4140 IF T1=5 THEN GOTO 4700
4150 IF T2=2 THEN GOTO 4700
4160 LET X2=X2=1
4170 IF X2=22 THEN LET X2=9
4180 IF T1=4 THEN GOTO 4540
4190 GOTO 4680
4200 IF T1=2 THEN GOTO 4320
4210 LET T1=1
4220 LET T1=1
4230 NEXT I
4240 GOTO 4350
4250 FOR J=M3(3) TO P+1 STEP -1
4260 LET T1=1
4270 LET X1=J=4
4280 LET X2=8
4290 SHIFT T Table: ONE COLUMN TO THE RIGHT TO INSERT AN ALTERNATIVE
4999 3000 FOR J=M3(3) TO P+1 STEP -1
4000 LET T1=1
4010 LET X1=J=4
4020 LET X2=8
4030 SHIFT T Table: ONE COLUMN TO THE RIGHT TO INSERT AN ALTERNATIVE
4999
FOR I=0 TO R-1
   LET T(I,J)=T(I,J-1)
   IF I=I THEN GOTO 3190
   IF I>9 THEN GOTO 3190
   IF T(I,J)>0 THEN IF I<4 THEN GOTO 3180
   CHANGE T(I,J) TO # USING "#-"
   LET X(I,I)=X(I,I-1)
   IF I=0 THEN CHANGE T(I,J)/10 TO # USING "#-"
   PRINT PC(X/I,X/2)/26
   LET X2=X2+1
   IF I<4 THEN GOTO 3190
   LET T1=T1+1
   NEXT I
   IF I=22 THEN GOTO 3250
   FOR I=22 TO 22
   PRINT PC(X/I)/26
   NEXT I
   NEXT J
   FOR I=0 TO R-1
   LET T(I,P)=0
   NEXT I
   LET Z=P+4
   FOR I=0 TO 22
   PRINT PC(Z,I)/26
   NEXT I
   RETURN
   FOR J=0 TO M5(J)-1
   LET T1=1
   LET X1=J+4
   LET I=6
   FOR I=0 TO R-1
   IF I=22 THEN GOTO 6260
   IF I>9 THEN GOTO 6260
   IF T(I,J)>0 THEN IF T(I,J)=0 THEN GOTO 6240
   CHANGE T(I,J) TO # USING "#-"
   LET X(I,J)=X(I,J-1)
   IF I=0 THEN CHANGE T(I,J)/10 TO # USING "#-"
   PRINT PC(X/I,X/2)/26
   LET X2=X2+1
   IF I<4 THEN GOTO 6260
   LET T1=T1+1
   NEXT I
   NEXT J
   RETURN
"ABLA-06B" ABL INTERACTIVE INTERPRETER- LOAD & EDIT PROGRAM, SET PARAMETERS

--- SET PARAMETERS MODULE ---

SET/CLEAR TRACE (T/C), SET/CLEAR BREAKPOINT (B/L) OR END (E)

4999 LET Z=104
7000 LET X=104
7020 GOSUB 7900
7040 ON -Z THEN GOTO 7100, 7200, 7300, 7400, 2000
7049
7100 GOSUB 7500
7110 LET B28(P,P)="Y"
7120 PRINT PC(P=4,4,6);"T";
7130 GOTO 7000
7149
7200 GOSUB 7500
7210 LET B28(P,P)="N"
7220 PRINT PC(P=4,4,6);"T";
7230 GOTO 7000
7249
7300 GOSUB 7500
7310 LET B18(P,P)="Y"
7320 PRINT PC(P=4,2,6);"B";
7330 GOTO 7000
7349
7400 GOSUB 7500
7410 LET B18(P,P)="N"
7420 PRINT PC(P=4,2,6);"B";
7430 GOTO 7000
7449
7500 PRINT PC(12,23);"ENTER ALTERNATIVE NUMBER";
7510 LET X=30
7520 LET X=23
7530 LET X=4
7530 GOSUB 9400
7540 ON X2 THEN GOTO 7530, 7530, 7530, 7530
7550 LET Y9,23=I
7560 GOSUB 9720
7570 IF Y9=0 THEN GOTO 7530
7599
7600 LET P=0
7610 FOR I=0 TO H3(3,3)-1
7620 IF T(I,1)=Z THEN LET P=I+1
7630 NEXT I
7640 IF P=0 THEN GOTO 7530
7670 PRINT PC(0,23);CRT(3);CRT(6);
7680 RETURN

--- END ---
600 REM "ABLA-06C" ABL INTERACTIVE INTERPRETER- EXECUTE MODULE

6020 LET R=30
6030 LET C=20

6099 --- LOCAL VARIABLES ---
6101 A1, A2, A3 COLUMN # FOR ALTERNATIVES
6102 C0 CURRENT CLUSTER NUMBER
6103 C MAXIMUM NUMBER OF ROWS, COLUMNS IN T TABLE
6104 T COMPACTED ABSTRACT PROGRAM TABLE (SEE FILE LAYOUT ABLTXXY)
6107 P, P1, L1 CHARACTER POINTERS IN P6
6108 P6 INSTRUCTION TO PARSE IN HOST LANGUAGE
6109 C6 INSTRUCTION MNEMONICS IN HOST LANGUAGE
6111 P POSITION IN T TABLE (COLUMN NUMBER) (0, C-1)
6113 P0 PREDICATE VALUE 0=FALSE, 1=TRUE
6114 R2 CURRENT ROW # IN T TABLE
6115 T9, T9* VALUE OF SECOND VARIABLE IN EXPRESSION
6116 W LENGTH OF ALPHA VARIABLE
6117 M1, M2 SUBSCRIPT IN ARRAYS OF VARIABLES V8. V9
6119 B blanks
6120 B1 BREAKPOINT VECTOR (Y/N)
6122 B2 TRACER (Y/N)
6123 B3 PRINT FLAG FOR TRACER
6125 B STACK POINTER (FOR COMPOUND ACTIONS)

6140 ON ERROR GOTO 9790
6150 DIM B$ (40), C8$ (104), P6 (30), T5$ (5), T9$ (30), V8$ (3), Z1$ (128), Z2$ (5)
6160 SHORT A1, A2, A3, CV, L1, P, P1, P9, B, T (R-1), C-1, V, W, M1, M2, R2, T1, Y9 (1), Z3, Z4
6170 LONG 1, T9, X, XI, X2, Z
6180 LET C8$ (1, 80) = "ACC: CHG: CHN: VCLF: (C)\COM\DAT\DIM\LX960 FOR\RV\00\G\TIF \INX\LET\LON\NXT\TON8"
6190 LET B$ (81, 160) = "DNO\ONR\ONS\OPF\PRE\PRT\RD\RF\RET\SEIF\SF\SHT\SLE\SN\SR\WRF"
6195 LET B$ (81, 160) = "DNO\ONR\ONS\OPF\PRE\PRT\RD\RF\RET\SEIF\SF\SHT\SLE\SN\SR\WRF"
6200 LET B$ (81, 160) = "DNO\ONR\ONS\OPF\PRE\PRT\RD\RF\RET\SEIF\SF\SHT\SLE\SN\SR\WRF"

6220 LET B$ = "", B

6299 --- FILE VARIABLES ---

6300 DIM M1$ (94), M3$ (70), M6$ (4)
6320 SHORT M2$ (10), M3$ (10), M7$ (9, 19)
6330 DIM V8$ (40), V9$ (372), V12$ (40), B1$ (20), B2$ (20), B3$ (1)
6340 SHORT V1$ (19), V2$ (19), W8$ (1)
6350 LONG V3$ (39)
6400 LET V2$ = "", V2
6410 LET B$ = ", B
6420 LET V1$ = ", V1
6900 CON B1$, B2$, B3$, S
```
0998 ' *** MAIN PROGRAM ***
1000 IF B=0 THEN GOSUB 2000
1020 IF B THEN GOSUB 6500
1050 IF B THEN GOTO 1020
1500 LET V=1
1520 CHAIN "ABLA-068",V,818,828,838,5
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=1 TO M(3)-1
2130 IF INT(1.1/10)+CV 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=I
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
2283 LET R9=10
2290 GOSUB 2600
2299 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
2530 SEARCH INDEX (1.2),M(1.2),"B",T(R2,A3) GIVING M1,M2
2560 LET P9=18,617,901
2570 IF P1(A3=1,A3=1)="N" THEN GOTO 2600
2590 PRINT PC(12),M1,M(1.60),PC(12),PC(11),P9
2594 IF M9="Y" THEN PRINT FILE (13),"EXECUTING ACTION";M9
2599 IF M9(9.14)>" " THEN GOTO 6000
2630 GOTO 3000
```
EVALUATE POSTCONDITIONS

2149 LET J = 3
2700 LET R2 = R2 + 1
2720 GOSUB 7000
2740 IF P9 = 1 THEN LET C9 = T(R2 + 1, A3)
2760 IF P9 = 0 THEN LET C9 = T(R2 + 2, A3)
2790 NEXT CLUSTER IS NEXT STEP • IF POSTCONDITIONS WERE SATISFIED

2800 IF B18(A3 + 1, A3 + 1) = "N" THEN GOTO 2100
IF BREAKPOINT SET, PAUSE FOR USER INTERACTION.
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 (A, Z), 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

*** END OF MAIN PROGRAM ***
2998 LET L=(STR(C8,P$(1,3))+3)/4
2999 --REGULAR ACTION--
3000 ON L THEN GOTO 3200, 3350, 3600, 3500, 3520, 3800, 3800, 3800, 3800, 3800, 3800, 3340
3020 ON L=20 THEN GOTO 3800, 3800, 3800, 3800, 3800, 3800, 3800, 3800, 3800, 3800, 3800, 3800
3040 ON L=30 THEN GOTO 3400, 3800, 3800, 3800, 3800, 3800, 3800, 3800, 3800, 3800, 3800, 3800
3060 GOTO 3800
3199 'ACCEPT'
3200 LET P=STR(P$, ",")
3210 CHANGE P$(4..P-1) TO W
3220 ACCEPT (W);Z$
3230 LET W=P$(P-1)
3240 IF W="" THEN GOTO 3810
3260 GOSUB 8000
3270 LET V1=V1(W1), V2(W1)=Z$
3280 GOTO 3900
3300 GOSUB 8000
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$(P-1)
3370 IF W="" THEN GOTO 3810
3380 IF STR(W,P$)=O THEN GOTO 3340
3390 GOSUB 8000
3400 LET W=W
3410 GOSUB 8000
3420 LET P=STR(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 W=W
3465 LET W=P$(P-1)
3470 GOSUB 8000
3480 CHANGE V2*(V1(U2), V2(U2)) TO V5(W1)
3490 GOTO 3900
3499 'CLOSEFILE'
3500 CHANGE P$(4) TO X
3505 IF X<7 THEN GOTO 3820
"ABLA-06C" ABL INTERACTIVE INTERPRETER- EXECUTE MODULE

3813 GOTO 3880
3820 LET Z=117
3825 GOTO 3880

CHANNEL 0 MUST BE BETWEEN 7 AND 14. FILE COMMAND IGNORED.

3830 LET Z=118
3835 GOSUB 9800
3900 LET R2=R2+1
3911 ACCEPT (9), Zs
3920 GOTO 2500

INVALID OPERAND OR VARIABLE TYPE - ACTION IGNORED.

3999 LET P=STR(Ps, "=")
4000 LET P=STR(Ps, "=")
4010 IF Ps=" THEN GOTO 3830
4020 LET Ps=PS(Ps+1)
4030 IF STR(Ps(Ps), "=") THEN GOTO 4100
4040 GOSUB 9200
4045 LET W2=W1
4050 LET Ws=Ws(Ps)
4055 LET T9=0
4060 IF Ws("1")="A" THEN IF Ws("1")="Z" THEN GOTO 4080
4070 CHANGE Ps(1) TO T9
4075 GOTO 4150
4080 GOSUB 8200
4085 LET T9=W3(W1)
4090 GOTO 4150

4100 GOSUB 8000
4110 LET W2=W1
4115 LET Ps=STR(Ps(Ps), "<34>")
4120 IF Ps=" THEN GOTO 4130
4125 LET Zs=PS(Ps(Ps), "<2345>"-1)
4130 GOTO 4150
4135 LET Ws=Ws(Ps)
4140 LET Zs=Zs(Zs)
4145 GOSUB 8000
4150 LET Ws=Ws(Ps)
4160 IF STR(Ws, "=")=0 THEN GOSUB 8200
4170 IF STR(Ws, "=")=0 THEN GOSUB 8000
4180 ON STR("+", Ps(Ps), Ps(Ps))=1 THEN GOTO 3830, 4200, 4210, 4220, 4230, 4240, 4250, 4260
4200 IF STR(Ws, "=")=0 THEN GOTO 4260
4205 LET V3(W1)=V3(W2)
4210 GOTO 3900
```
5998
5999
6000 LET VB$=M5$1.4
6010 LET VB=A3
6020 LET VB(1)=R2+1
6030 OPEN FILE (0.512), "ABLSTACK"
6040
6050 LET S=S+1
6060 WRITE FILE (0.31), V8, V1, V2, V2
6070 LET S=S+1
6080 WRITE FILE (0.6), V5$, V5, V8, V9$, B1$, B2$, B3$
6090 CLOSE FILE (0)
6100 CLOSE FILE (5)
6110 CLOSE FILE (6)
6120 LET Z$="ABLPM", M1$(93.94), M1$(91.92)
6130 OPEN FILE (5.5), Z$
6140 LET Z$="BLT"
6150 OPEN FILE (4.312), Z$
6155 LET Z$="ENTERING NEXT LOWER LEVEL OF ABSTRACTION: PG/MACH, M1$(91.92), "", M1$(93.94)
6160 PRINT PC(0.23), Z$
6165 IF B3$="Y" THEN PRINT FILE (13), SKP(2)
6167 IF B3$="Y" THEN PRINT FILE (13), Z$
6180 GOSUB 2900
6190 GOSUB 5200
6195 GOTO 2020

6199
6200 FOR J=0 TO M5(3)-1
6210 LET T1=1
6220 LET X1=Y4
6230 LET X2=8
6240 FOR I=0 TO R-1
6250 IF X2=Z2 THEN GOTO 6350
6260 IF T1=O THEN GOTO 6350
6270 IF T1(J)=0 THEN IF T1C4 THEN GOTO 6340
6280 CHANGE T1(J) TO 10 USING "98"
6290 LET Z$=(T1(J)) T1=(T1(J))
6300 IF I=0 THEN CHANGE T1(J) TO Z$ USING "98"
6310 PRINT PC($X1, X21), Z$
6320 LET X2=X2+1
6330 IF T1C4 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$(I.1)="Y" THEN PRINT PC(1+4-4.6), "Y"
6430 IF B1$(I.1)="Y" THEN PRINT PC(1+4-2.6), "B"
6440 NEXT I
6480 RETURN

6499
```

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

---COMPOUND ACTION---
STORE STATE VECTOR, TRANSFER CONTROL TO SUB-PROGRAM

RUSH TWO RECORDS ONTO CONTROL STACK

PRINT EXISTING ALTERNATIVES

RESTORE STATE VECTOR, TRANSFER CONTROL BACK TO CALLING PROGRAM
6500 OPEN FILE (0,512)."ABLASTACK"
6509
6520 LET G=G-1
6530 READ FILE (0,G).V8$. V1$. V2$. V2
6540 LET S=S-1
6550 CLOSE FILE (0)
6560 LET A3=V8$
6570 LET A2=V8(1)
6580 CLOSE FILE (5)
6590 CLOSE FILE (6)
6600 CLOSE FILE (7)
6620 LET Z$="ABL\$".V8$
6630 OPEN FILE (5,5).Z$
6640 LET Z$=1.4)="ABLT"
6650 OPEN FILE (6,512).Z$
6700 LET Z$="RESUMING NEXT HIGHER LEVEL OF ABSTRACTION- P0M/MACH *,V8$. (3,4),/\$. V8$. (1,2)
6710 PRINT P0(0.23)).Z$
6720 IF B3$="V" THEN PRINT FILE (15).SKP(T2)
6730 IF B3$="V" THEN PRINT FILE (15).Z$
6740 Q0SUB 9900
6760 Q0SUB 6200
6770 QOTO 2500
EVALUATE PREDICATES (PRE + POST CONDITIONS)

DECLARE PREDICATE AND EVALUATE

SEARCH INDEX (1:2), MS(1:2), ["A", T(J,J)] GIVING MS(1:2)

LET PS=M(1,1)H(1,1)

IF E2+(J,J-1)=="N" THEN GOTO 7709

PRINT PC(1,21): MS(1,60): PC(1,22): PS(1,60)

IF E2+=="Y" THEN PRINT FILE (10), "EVALUATING PREDICATE", MS(1:2)

IF PS(1,3)=="IF " THEN GOTO 7700

IF PS(4,6)<"EOD" THEN GOTO 7199

CHANGE PS(5,9) TO X

ON EDF(1)+1 THEN GOTO 7700, 7800

IF STR(PS(4,6),"=") THEN GOTO 7400

FIND VALUE OF NUMERIC VARIABLE

LET L=(STR(VS(PS(4,6)))-1)/3

IF L<0 THEN GOTO 7710

IF PS(9,9)=="A" THEN IF PS(9,8)=="Z" THEN GOTO 7250

CHANGE PS(9,9) TO T9

GOTO 7200

LET LI=STR(VS(PS(9,11))

IF LI<0 THEN GOTO 7710

LET T9=(VS(9,11)-1)/3

IF PS(9,11)=="N" THEN GOTO 7300

PRINT PC(33,22): VS(L): PS(7,8)+9

IF B3+=="Y" THEN PRINT FILE (10), "NUMERICAL VALUES", VS(L): PS(7,8)+9

ON STR("->"<>>:"<=",PS(7,8))+1 THEN GOTO 7220, 7310, 7300, 7330, 7340, 7350, 7360

IF VS(L)<T9 THEN GOTO 7800

GOTO 7215

GOTO 7700

IF VS(L)>CT9 THEN GOTO 7800

GOTO 7220

GOTO 7200

GOTO 7300

GOTO 7230

GOTO 7320

GOTO 7310

GOTO 7300

GOTO 7300

GOTO 7320

GOTO 7310

GOTO 7300

IF VS(L)==CT9 THEN GOTO 7800

GOTO 7235

GOTO 7700

GOTO 7350

GOTO 7700

GOTO 7360

GOTO 7700

GOTO 7365

FIND VALUE OF ALPHA VARIABLE

LET L=(STR(VS(PS(4,6)))-1)/3

IF L<0 THEN GOTO 7710

IF PS(9,9)=="A" THEN IF PS(9,8)=="Z" THEN GOTO 7250

CHANGE PS(9,9) TO T9

GOTO 7200

LET LI=STR(VS(PS(9,11))

IF LI<0 THEN GOTO 7710

LET T9=(VS(9,11)-1)/3

IF PS(9,11)=="N" THEN GOTO 7300

PRINT PC(33,22): VS(L): PS(7,8)+9

IF B3+=="Y" THEN PRINT FILE (10), "ALPHA VALUES", VS(L): V2(VL(1), V2(L1))

PRINT PC(75,22): VS(VL(1), V2(L1)): PS(7,8): V2(VL(1), V2(L1))

GOTO 7350

IF VS(L)<T9 THEN GOTO 7800
ABLA-O&G ABL INTERACTIVE INTERPRETER - EXECUTE MODULE

7600 ON STR1("==", V1(L1), V2(L1)) < T9 THEN GOTO 7720, 7610, 7620, 7630, 7640, 7650
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

7700 LET Z=111
7705 GOTO 7770

7700 LET Z=111
7705 GOTO 7770

7700 LET Z=113
7700 GOSUB 9800
7700 LET P=0
7700 LET R2=1
7700 LET I=100
7800 NEXT I
7850 RETURN
2999 IF W$="" THEN RETURN
8010 LET W1=(STR(V$,W$)-1)/3
8020 IF W1>0 THEN RETURN
8050 LET X=(STR(V$,W$)-1)-1/3
8060 LET V1(X)=1
8070 IF X THEN LET V1(X)=V2(X-1)+1
8080 LET V2(X)=V1(X)+W-1
8090 LET V$(X*3+1,X*3+3)=W$  
8100 GOTO 8010
8199 FIND THE ALPHANUMERIC LOCAL VARIABLE
8200 IF W$="" THEN RETURN
8210 LET W1=(STR(V$,W$)-1)/3
8220 IF W1>0 THEN RETURN
8250 LET X=(STR(V$,W$)-1)-1/3
8260 LET V$(X*3+1,X*3+3)=W$  
8270 GOTO 8210
8220 FIND THE NUMERIC LOCAL VARIABLE
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=3460 THEN GOTO 3900
8810 IF Y9=3525 THEN IF Y9(1)=18 THEN GOTO 3950
8820 IF Y9=3590 THEN GOTO 3900
8870 IF Y9=9838 THEN GOTO 9842
8880 "PRINT PC(10,23):" UNEXPECTED ERROR ";Y9(1):" AT <7>: Y9;PC(0.20)
8900 STOP

TOTAL # LINES OF TEXT = 314
TOTAL # CHARACTERS OF TEXT = 14529
TOTAL # CHARACTERS IN 'REM' = 2359
# LINES PRINTED = 314
**FILE LAYOUT FOR: ABLHACHI  PROJECT - AB**

**DESCRIPTION:** ABL ABSTRACT MACHINE TABLE

**FORMAT CODE:**

**MEMO:**

**POINTS TO: 236**

<table>
<thead>
<tr>
<th>TYPE</th>
<th>VOLUME ID</th>
<th>PREPARED BY</th>
<th>MEMO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DYNAMIC</th>
<th>OPTIMIZE FACTOR</th>
<th>DATE PREPARED</th>
<th>REVISION NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0%</td>
<td>OCT 12.81</td>
<td>00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CATALOG</th>
<th>TOTAL NO RECORDS</th>
<th>ACTIVE NO RECORDS</th>
<th>REVISED BY</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>400</td>
<td>145</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UD SIZE</th>
<th>FILE SIZE (BLOCKS)</th>
<th>DATED REVISED</th>
</tr>
</thead>
<tbody>
<tr>
<td>116</td>
<td>104</td>
<td></td>
</tr>
</tbody>
</table>

**FIELDS**

<table>
<thead>
<tr>
<th>FIELD DESCRIPTION</th>
<th>POS SIZE</th>
<th>NAME/MASK</th>
<th>C</th>
<th>R</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACHINE NUMBER</td>
<td>1</td>
<td>2A M8 (1.2)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SECTION CODE (1)</td>
<td>3</td>
<td>1A M8 (3.3)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BUCKET</td>
<td>4</td>
<td>1A M8 (4.4)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ELEMENT NUMBER</td>
<td>5</td>
<td>19 M1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**USER DATA**

| DESCRIPTION | 7 | 60A M16(1.60) | 0 |
| CORRESPONDING EXECUTABLE CODE | 67 | 30A M16(61.90) | 0 |
| PROGRAM IDENTIFIER (COMPOUND ACTIONS)(2) | 97 | 2A M16(91.92) | 0 |
| MACHINE IDENTIFIER (COMPOUND ACTIONS)(2) | 99 | 2A M16(93.94) | 0 |
| ESTIMATED EXECUTION TIME (MILLISECONDS) | 101 | 18 M2 | 0 |
| INPUT ARGUMENTS | 103 | 95 M2 (1-3) | 0 |
| OUTPUT ARGUMENTS | 113 | 5S M2 (6-10) | 0 |

**NOTES**

<table>
<thead>
<tr>
<th>SECTION CODE</th>
<th>NUMBER</th>
<th>PURPOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(BLANK)</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>1-N</td>
<td>MACHINE DESCRIPTION</td>
</tr>
<tr>
<td>B</td>
<td>1-N</td>
<td>ACTION DESCRIPTION</td>
</tr>
<tr>
<td>C</td>
<td>1-N</td>
<td>DATA OBJECT DESCRIPTION</td>
</tr>
</tbody>
</table>

**CONTINUED**
**FILE LAYOUT FOR: ABLPXXYY PROJECT: AB**

**AS OF MAR 21, 82 AT 14:38**

**DESCRIPTION:** ABL ABSTRACT PROGRAM

**MEMO:** X=MACHINE Y=PROGRAM

**POINTS TO:** 236

<table>
<thead>
<tr>
<th>TYPE:</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHARED:</td>
<td>Y</td>
</tr>
<tr>
<td>DYNAMIC:</td>
<td>Y</td>
</tr>
<tr>
<td>OPTIMIZE_FACTOR:</td>
<td>0</td>
</tr>
<tr>
<td>DATE PREPARED:</td>
<td>OCT 12, 81</td>
</tr>
<tr>
<td>CATALOG:</td>
<td>303</td>
</tr>
<tr>
<td>TOTAL NO_RECORDS:</td>
<td>200</td>
</tr>
<tr>
<td>RECORD SIZE:</td>
<td>84</td>
</tr>
<tr>
<td>ACTIVE NO_RECORDS:</td>
<td>0</td>
</tr>
<tr>
<td>U D SIZE:</td>
<td>80</td>
</tr>
<tr>
<td>FILE SIZE (BLOCKS):</td>
<td>0</td>
</tr>
</tbody>
</table>

**REVISION NO:** 00

**FILE SIZE:**

<table>
<thead>
<tr>
<th>FIELD DESCRIPTION</th>
<th>POS</th>
<th>SIZE</th>
<th>NAME/MASK</th>
<th>C</th>
<th>R</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 SECTION CODE (1)</td>
<td>1</td>
<td>1A</td>
<td>M2$1.1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 BUCKET</td>
<td>2</td>
<td>1A</td>
<td>M2$2.2</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 ELEMENT_NUMBER</td>
<td>3</td>
<td>1S</td>
<td>M3</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER DATA</td>
<td>5</td>
<td>AOA</td>
<td>M3$1.60</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 DESCRIPTION</td>
<td>63</td>
<td>3S</td>
<td>M4 (0-4)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 INPUT ARGUMENTS</td>
<td>75</td>
<td>3S</td>
<td>M4 (5-9)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTES**

<table>
<thead>
<tr>
<th>(1) SECTION CODE</th>
<th>NUMBER</th>
<th>PURPOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLANK</td>
<td>1-N</td>
<td>PROGRAM DESCRIPTION</td>
</tr>
<tr>
<td>A</td>
<td>1-N</td>
<td>CLUSTER DESCRIPTION</td>
</tr>
<tr>
<td>B</td>
<td>1-N</td>
<td>ALTERNATIVE DESCRIPTION</td>
</tr>
</tbody>
</table>
**FILE LAYOUT FOR: ABL STACK**

**PROJECT: A**

**AS OF MAR 21, 82 AT 14:38**

**DESCRIPTI0N:** ABL CONTROL STACK FOR EXECUTION

**FORMAT CODE:** 01

**MEMO:** RECORD 01

**POINTS TO: **

<table>
<thead>
<tr>
<th>TYPE</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOLME ID</td>
<td>0</td>
</tr>
<tr>
<td>PREPARED BY</td>
<td>MK</td>
</tr>
<tr>
<td>DATE PREPARED</td>
<td>JAN 04, 82</td>
</tr>
<tr>
<td>OPTIMIZE FACTOR</td>
<td>0%</td>
</tr>
<tr>
<td>TOTAL NO RECORDS</td>
<td>20</td>
</tr>
<tr>
<td>REVISION NO</td>
<td>00</td>
</tr>
<tr>
<td>CATALOG</td>
<td>300</td>
</tr>
<tr>
<td>RECORD SIZE</td>
<td>512</td>
</tr>
<tr>
<td>FILE SIZE (BLOCKS)</td>
<td>0</td>
</tr>
<tr>
<td>DATE REVISED</td>
<td></td>
</tr>
</tbody>
</table>

**FIELDS**

<table>
<thead>
<tr>
<th>FIELD DESCRIPTION</th>
<th>POS</th>
<th>SIZE</th>
<th>NAME/MASK</th>
<th>C</th>
<th>R</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 VARIABLE NAMERS FOR ALPHA VARS. 20 * 3A</td>
<td>1</td>
<td>60A</td>
<td>V* (1, 60)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 START POSITION FOR ALPHA VARS. 20 * 1S</td>
<td>61</td>
<td>20S</td>
<td>V1 (0-19)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 ALPHA VARIABLES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 END POSITION FOR ALPHA VARS. 20 * 1S</td>
<td>473</td>
<td>20S</td>
<td>V2 (0-19)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FILE LAYOUT FOR: ABLSACK PROJECT AS
AS OF MAR 21, 82 AT 14:38

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

POINTS TO:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>SHARED</th>
<th>VOLUME ID</th>
<th>0</th>
<th>PREPARED BY: PK</th>
</tr>
</thead>
<tbody>
<tr>
<td>DYNAMIC</td>
<td>N</td>
<td>OPTIMIZE FACTOR: 0%</td>
<td>DATE PREPARED: JAN 04, 82</td>
<td></td>
</tr>
<tr>
<td>CATALOG: 300</td>
<td>TOTAL NO RECORDS: 20</td>
<td>REVISION NO: 00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RECORD SIZE: 512 | ACTIVE NO RECORDS: 0 | REVISED BY:
| V D SIZE: FILE SIZE (BLOCKS): 0 | DATE REVISED: |

# FIELD DESCRIPTION POS SIZE NAME/MASK C R P
1 VARIABLE NAMES FOR NUMERIC VARS. 40 - 5A 4 1 120A V9(1.120) 0
2 NUMERIC VARIABLES 4 121 40L VB(0-39) 0
3 CALLING MACHINE / PROGRAM 4 441 4A VB(1,4) 0
4 CALLING ALTERNATIVE (COLUMN) NUMBER 4 449 4S VB 0
5 CALLING ACTION (ROW) NUMBER 4 447 1S VB (1) 0
6 BUCKET - V9(1.23) 4 449 23A V9(1.23) 0
7 BREAKPOINT VECTOR - B1(1.20) 4 472 20A V9(24.43) 0
8 TRACE VECTOR - B2(1.20) 4 492 20A V9(44.43) 0
9 PRINT TRACE ON PRINTER? (Y/N) B3(1.11) 3 12 1A V9(64.64) 0
**FILE LAYOUT FOR: ABL XXY Y PROJECT A**

**DESCRIPTION:** ABL COMPACTED ABSTRACT PROGRAM TABLE

**FORMAT CODE:** MEMO: XX=MACHINE YY=PROGRAM

**POINTS TO:** ABCPXXYY

<table>
<thead>
<tr>
<th>TYPE:</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHARED:</td>
<td>Y</td>
</tr>
<tr>
<td>VOLUME ID:</td>
<td>0</td>
</tr>
<tr>
<td>PREPARED BY:</td>
<td>MK</td>
</tr>
<tr>
<td>DYNAMIC:</td>
<td>Y</td>
</tr>
<tr>
<td>OPTIMIZE FACTOR:</td>
<td>0%</td>
</tr>
<tr>
<td>DATE PREPARED:</td>
<td>OCT 12, 81</td>
</tr>
<tr>
<td>CATALOG:</td>
<td>310</td>
</tr>
<tr>
<td>TOTAL NO RECORDS:</td>
<td>100</td>
</tr>
<tr>
<td>REVISION NO:</td>
<td>00</td>
</tr>
<tr>
<td>RECORD SIZE:</td>
<td>32</td>
</tr>
<tr>
<td>ACTIVE NO RECORDS:</td>
<td>0</td>
</tr>
<tr>
<td>REVISED BY:</td>
<td></td>
</tr>
<tr>
<td>U D SIZE:</td>
<td></td>
</tr>
<tr>
<td>FILE SIZE (BLOCKS):</td>
<td>0</td>
</tr>
</tbody>
</table>

---

### FIELDS

<table>
<thead>
<tr>
<th>FIELD DESCRIPTION</th>
<th>POS SIZE NAME/MASK</th>
<th>C</th>
<th>R</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 RECORD MARK (+253)</td>
<td>1 1S M5</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 RECORD NUMBER [1/2/3]</td>
<td>3 1S M5 (1)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 TOTAL NUMBER OF CLUSTERS</td>
<td>5 1S M5 (2)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 TOTAL NUMBER OF ALTERNATIVES</td>
<td>7 1S M5 (3)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 TOTAL NUMBER OF PRECONDITIONS (2)</td>
<td>9 1S M5 (4)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 TOTAL NUMBER OF ACTIONS (2)</td>
<td>11 1S M5 (5)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 TOTAL NUMBER OF POSTCONDITIONS (2)</td>
<td>13 1S M5 (6)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 TOTAL NUMBER OF DATA OBJECTS (2)</td>
<td>15 1S M5 (7)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 BUCKET</td>
<td>17 3S M5 (8-10)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 MACHINE NUMBER</td>
<td>23 2A M5* (1.2)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 PROGRAM NUMBER</td>
<td>25 2A M5* (3.4)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 BUCKET</td>
<td>27 66A M5* (5.70)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13 BUCKET</td>
<td>93 5F M6 (0-4)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 COMPACTED MATRIX (1)</td>
<td>113 200S M7 (0-199)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**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)
- ACTIONS (1..A, A = ACTION NUMBER)
- POSTCONDITIONS (-P, OR +P, P = PREDICATE NUMBER)
- 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.

E0. 11 12 21 31 32 33 41 51

(2) FOR FUTURE USE.
LIST OF REFERENCES


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.


ANNEX I

ABL TABLES TO SUPPORT THE
LOGICAL DESCRIPTION IN THE
CHAPTER 3
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 FILD THE HIERARCHICALLY INTERRELATION BLOCKS OF THE WHOLE SYSTEM. THE PREDICATES ACT AS CRITERION FOR A PARTICULAR SUBFUNCTION. THEY DEFINE THE CONSTRAINTS THE ACTIONS IN THE PARTICULAR SUBFUNCTION MUST OBEY. UNLESS THESE CRITERION ARE SATISFIED, THEN 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 == DECENTRALIZED PROCESSING IS NEEDED
   1.2 == SET UP RAMBOM ACCESS DATABASE
   1.3 == MODULARIZE SYSTEM
   1.4 == SET UP INTERACTIVE #DB UPDATING NAIVE USER INTERFACE
   1.5 == SET UP INTERACTIVE #DB QUERY NAIVE USER INTERFACE
   1.6 == SET UP PSEUDOMATCH #DB QUERY POSSIBILITY
   1.7 == SET UP ACCESSORIES FOR #ATS
   1.8 == SET UP PRODUCTION CYCLE PROGRAMS
   1.9 == BODY OF THE BASIC SYSTEM IS ORGANIZED
   === $< 1, 1, 1, 1, 1, 1, 1, 2>$

2. === ORGANIZE THE REGULAR RUN OF THE SYSTEM
   2.1 == #ATS DAILY DATA USAGE PARAMETER DEFINITION
   2.2 == #ATS DAILY TRANSACTION DATA GENERATION
   2.3 == #ATS TRANSACTION DATA COLLECTION
   2.4 == #ATS TRANSACTION DATA PROCESSING
   2.5 == #ATS DAILY DATA DISTRIBUTION
   2.6 == #ATS DAILY OUTPUT USAGE
   2.7 == #ATS SYSTEM IS ORGANIZED
   === $< 2, 2, 2, 2, 2, 2, 3>$

CONTINUED ON THE NEXT PAGE
3.1 UPDATE THE COURSE INF
3.2 UPDATE THE STUDENT INF
3.3 UPDATE ATTENDANCE RELATED DATA
3.4 UPDATE ORGANIZATION IS COMPLETE

4.1 QUERY THE COURSE INF INTERACTIVELY
4.2 QUERY THE STUDENT INF INTERACTIVELY
4.3 QUERY THE CLASS INF INTERACTIVELY
4.4 QUERY THE COURSE INF PSEUDOBATCH NODE
4.5 QUERY THE STUDENT INF PSEUDOBATCH NODE
4.6 QUERY THE CLASS INF PSEUDOBATCH NODE
4.7 DATA-BASE QUERY ORGANIZATION IS COMPLETE
Modeling :0 OR THE OVERALL SYSTEM DESIGN AND MODELING OF THIS SYSTEM

THIS TABLE PROVIDES AN OVERVIEW OF 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 CRITERIA FOR A PARTICULAR SUBFUNCTION. THEY DEFINE THE CONSTRAINTS THE ACTIONS OF THE PARTICULAR SUBFUNCTION 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 OPERATIONAL SEQUENCED AND INTERFACED IN THE NEXT TABLE CALLED $SYSTALL.

1.== ORGANIZE THE BODY OF THE SYSTEM
    1.1== DECENTRALIZED PROCESSING IS NEEDED == Then-GO-TO-CLUSTER-> I
        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
            DATA TERMINAL IS AVAILABLE AND IF
            DATA COMMUNICATION IS AVAILABLE AND IF
            REMOTE TIME-SHARING COMPUTER IS AVAILABLE

    1.2== SET UP RANDOM ACCESS DATA-BASE
        1.2.1 IF P. ON-LINE SYSTEM ACCESS IS NEEDED AND IF
            DISK-STORE IS AVAILABLE AND IF
            INTERACTIVE QUERY AND UPDATE IS NEEDED AND IF
            SYSTEM USAGE IS STUDIED

    1.3== MODULARIZE SYSTEM
        1.3.1 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 AND UPDATING NAIVE USER INTERFACE == Then-GO-TO-CLUSTER-> I
        1.4.1 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 OF, IS NEEDED AND IF
            INTERACTIVE UPDATING LANGUAGE FOR THE STUDENT OFF IS NEEDED AND IF
            CLASS MASTER FILE UPDATING CAN BE DONE AUTOMATICALLY

CONTINUED ON THE NEXT PAGE
1.5 - SET UP INTERACTIVE #88 QUERY NAIVE USER INTERFACE = --THEN-GO-TO-CLUSTER--> 1

1.5.1 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 QUERY LANGUAGE FOR THE COURSE #9F, IS NEEDED AND IF
INTERACTIVE QUERY LANGUAGE FOR THE STUDENT #9F, IS NEEDED AND IF
INTERACTIVE QUERY LANGUAGE FOR THE CLASS #9F, IS NEEDED

1.6 - SET UP PSEUDOMATCH #88 QUERY POSSIBILITY = --THEN-GO-TO-CLUSTER--> 1

1.6.1 IF IMMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING AND IF

NO 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 QUERY LANGUAGE FOR THE COURSE #9F, IS NEEDED AND IF
INTERACTIVE QUERY LANGUAGE FOR THE STUDENT #9F, IS NEEDED AND IF
INTERACTIVE QUERY LANGUAGE FOR THE CLASS #9F, IS NEEDED

1.7 - SET UP ACCESSORIES FOR #ATS

1.7.1 IF HIGH VOLUME OF TRANCY TRANSACTION OCCUR (100-250 A LECTURE) AND IF
DATA COLLECTION PERIOD IS DEFINED AS 30-40 MINUTES LONG AND IF
DATA TRANSMISSION IS 30-40 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

1.8.1 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

1.9.1 IF ALL SUBFUNCTIONS IN THIS FUNCTION (CLUSTER) ARE NOT
2. --- ORGANIZE THE REGULAR RUN OF THE SYSTEM

2.1 - DAILY DATA USAGE PARAMETER DEFINITION

2.1.1 IF VARIABLE LENGTH, OUTPUT IS NEEDED

2.2 - DAILY TRANSACTION DATA GENERATION

2.2.1 IF DATA GENERATION PROCESS IS ACCEPTED BY THE TEACHERS AND IF
ERROR GENERATION POSSIBILITY IN TRANSACTION DATA IS SMALL AND IF
DATA GENERATION PERIOD IS 30-60 SECONDS

2.3 - DAILY TRANSACTION DATA COLLECTION

2.3.1 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 DATS TRANSACTION DATA PROCESSING
   2.4 IF EXTENSIVE TRANSACTION STORAGE AND ANALYSIS IS REQUIRED AND IF
       REGULAR DATS OPERATOR IS AVAILABLE FROM STAFF AND IF
       PROCEDURE IS SIMPLE AND IF
       PRINTING OF SPOOLED OUTPUT IS NOT MORE THAN 1-2 HOURS

2.5 DATS DAILY DATA DISTRIBUTION
   2.5 IF PROCEDURE IS SIMPLE

2.6 DATS DAILY OUTPUT USAGE
   2.6 IF OUTPUT VOLUME IS APPROPRIATE TO DAILY USAGE AND IF
       FALSE NEGATIVE ERROR RATE IS BELOW 10% RATE AND IF
       FALSE POSITIVE ERROR RATE IS BELOW 0.12 RATE AND IF
       PROVIDED INFORMATION CAN BE READILY USED IN DECISION MAKING AND IF
       INFORMATION VALUE IS MAXIMUM OPERATIONAL VS. OVERALL DELAY.

2.7 DATS SYSTEM IS ORGANIZED
   2.7 IF ALL SUBFUNCTIONS IN THIS FUNCTION (CLUSTER) ARE MET

3. DATA UPDATE ORGANIZATION
   3.1 UPDATE THE COURSE OFF
   3.1 IF HIGH DEGREE OF DATA-BASE ACCURACY IS NEEDED AND IF
       USER SYSTEM ACCESS ORGANIZATION IS DEFINED AND IF
       DEPARTMENT OF THE SYSTEM HAS ACCESS AND IF
      ReadOnly DATA IN THE DATA-BASE ARE AUTOMATICALLY UPDATED

3.2 UPDATE THE STUDENT OFF
   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
       ReadOnly DATA IN THE DATA-BASE ARE AUTOMATICALLY UPDATED

3.3 UPDATE ATTENDANCE RELATED DATA
   3.3 IF HIGH DEGREE OF DATA-BASE ACCURACY IS NEEDED AND IF
       USER SYSTEM ACCESS ORGANIZATION IS DEFINED AND IF
       DEPARTMENT OF THE SCHOOL HAS ACCESS AND IF
       Class RELATED DATA CHANGED OR SURE-Card-CARDS GOT LOST AND IF
       Data-BASE DATA IS UPDATED AND IF
       Data PUNCHING AND INTERPRETING IS AVAILABLE.

3.4 UPDATE ORGANIZATION IS COMPLETE
   3.4 IF ALL SUBFUNCTIONS IN THIS FUNCTION (CLUSTER) ARE MET

4. DATA-BASE QUERY ORGANIZATION
   4. DATA-BASE QUERY ORGANIZATION

4.1 QUERY THE COURSE OFF INTERACTIVELY
   4.1 IF CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED AND IF
       Object is CHANGED IF THE SYSTEM HAS ACCESS AND IF
       OBJECTIVE OF AN EFFICIENT SCHOOL MANAGEMENT AND IF
       INTERMEDIATE INFORMATION ACCESS IS NEEDED FOR DECISION MAKING AND IF
       OUTPUT VOLUME IS SMALL.

CONTINUED ON THE NEXT PAGE.
4.2 Query the student: if interactively

4.2.1 If confidentiality and security of the data-base is defined and if

- app in charge of the system has access and if
- secret app 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 Query the class: if interactively

4.3.1 If confidentiality and security of the data-base is defined and if

- app in charge of the system has access and if
- secret app 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 Query the course: if pseudomatch mode

4.4.1 If objective is an efficient school management and if

- course information is needed and if
- app 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 Query the student: if pseudomatch mode

4.5.1 If objective is an efficient school management and if

- student information is needed and if
- app 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 Query the class: if pseudomatch mode

4.6.1 If objective is an efficient school management and if

- objective is an efficient class management and if
- class information is needed and if
- app 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 query organization is complete

4.7.1 If all subfunctions in this function (cluster) are met
1. ORGANIZE THE BODY OF THE SYSTEM

1.1-DECENTRALIZED PROCESSING IS NEEDED

1.1 IF PRODUCTION CYCLE PERIOD IS DEFINED AS 1-2 DAYS LONG AND IF
   DATA COLLECTION PERIOD IS DEFINED AS 30-60 MINUTES LONG AND IF
   DEDICATED 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

1.2 IF ON-LINE SYSTEM ACCESS IS NEEDED AND IF
   DISK STORAGE IS AVAILABLE AND IF
   INTERACTIVE QUERY 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-NORMALIZE SYSTEM

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
   UNDER SYSTEM ACCESS ORGANIZATION IS DEFINED AND IF
   CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED AND IF
   SYSTEM NORMALIZATION 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 UPDATE PROGRAM
   A6> SET UP STUDENT MASTER FILE UPDATE PROGRAM
   A7> SET UP COURSE MASTER FILE QUERY PROGRAM
   A8> SET UP STUDENT MASTER FILE QUERY PROGRAM
   A9> SET UP CLASS MASTER FILE QUERY PROGRAM
   A10> SET UP DAILY INITIALIZATION INTERACTIVE PROGRAM
   A11> SET UP DATA COLLECTION PSEUDOBATCH PROGRAM
   A12> SET UP AISIS DATA PROCESSING PSEUDOBATCH PROGRAM
   A13> SET UP PROCEDURE TO PRINT SPOOLED OUTPUT

CONTINUED ON THE NEXT PAGE
1.4 SET UP INTERACTIVE AND UPDATING NAIVE USER INTERFACE — THEN-GO-TO-CLUSTER — 1
  1.4 IF HIGHEST 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
  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 QUERY LANGUAGE FOR THE COURSE #IF, IS NEEDED AND IF
  INTERACTIVE QUERY LANGUAGE FOR THE STUDENT #IF, IS NEEDED AND IF
  CLASS MASTER FILE UPDATING CAN BE DONE AUTOMATICALLY
  A11> SET UP COURSE MASTER FILE UPDATE PROGRAM
  A12> SET UP CLASS MASTER FILE UPDATE PROGRAM
  A10> UPDATE COURSE MASTER FILE INTERACTIVELY
  A11> UPDATE STUDENT MASTER FILE INTERACTIVELY
  A12> UPDATE CLASS MASTER FILE INTERACTIVELY

1.5 SET UP INTERACTIVE QUERY 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 QUERY LANGUAGE FOR THE COURSE #IF, IS NEEDED AND IF
  INTERACTIVE QUERY LANGUAGE FOR THE STUDENT #IF, IS NEEDED AND IF
  INTERACTIVE QUERY LANGUAGE FOR THE CLASS #IF, IS NEEDED
  A13> SET UP COURSE MASTER FILE QUERY PROGRAM
  A14> SET UP STUDENT MASTER FILE QUERY PROGRAM
  A15> SET UP CLASS MASTER FILE QUERY PROGRAM
  A13> QUERY COURSE MASTER FILE INTERACTIVELY
  A14> QUERY STUDENT MASTER FILE INTERACTIVELY
  A15> QUERY CLASS MASTER FILE INTERACTIVELY

1.6 SET UP PSEUDOBATCH #IF QUERY POSSIBILITY — THEN-GO-TO-CLUSTER — 1
  1.6 IF 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 QUERY LANGUAGE FOR THE COURSE #IF, IS NEEDED AND IF
  INTERACTIVE QUERY LANGUAGE FOR THE STUDENT #IF, IS NEEDED AND IF
  INTERACTIVE QUERY LANGUAGE FOR THE CLASS #IF, IS NEEDED
  A16> WRITE A FILE IN COURSE MASTER FILE QUERY LANGUAGE
  A17> WRITE A FILE IN STUDENT MASTER FILE QUERY LANGUAGE
  A18> WRITE A FILE IN CLASS MASTER FILE QUERY LANGUAGE
  A19> QUERY COURSE MASTER FILE PSEUDOBATCH NODE
  A20> QUERY STUDENT MASTER FILE PSEUDOBATCH NODE
  A21> QUERY CLASS MASTER FILE PSEUDOBATCH NODE
  A22> PRINT SPooled OUTPUT

CONTINUED ON THE NEXT PAGE
1.7 == SET UP ACCESSORIES FOR DATS == -- THEN-GO-TO-CLUSTER -> 1-
1.7 IF HIGH VOLUME OF 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
A/3 > DIRTY CLASS MASTER FILE PSEUDOBATCH NODE
A/4 > PRODUCE A AID-CASE FOR THE CLASS
A/5 > PRODUCE TEACHER DATA GENERATION KITS
A/6 > SET UP DATA COLLECTION ROUTES IN SCHOOL
A/7 > INSTALL OPTICAL MARK READER ON TERMINAL
A/8 > VERIFY DATA DISTRIBUTION ROUTES
A/9 > DETERMINE DATA USAGE PARAMETERS

1.8 == SET UP PRODUCTION CYCLE PROGRAMS == -- THEN-GO-TO-CLUSTER -> 1-
1.8 IF REGULAR DATS 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
A/10 > SET UP DAILY INITIALIZATION INTERACTIVE PROGRAM
A/11 > SET UP DATA COLLECTION PSEUDOBATCH PROGRAM
A/12 > SET UP DATS, DATA PROCESSING PSEUDOBATCH PROGRAM
A/13 > 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 NEST
A/14 > NO ACTION- NEXT CLUSTER

2.== ORGANIZE THE REGULAR RUN OF THE SYSTEM

2.1 == DATS DAILY DATA USAGE PARAMETER DEFINITION
2.1 IF VARIABLE LENGTH OUTPUT IS NEEDED
A/20 > PERFORM DAILY SYSTEM INITIALIZATION

2.2 == DATS DAILY TRANSACTION DATA GENERATION
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
A/25 > PERFORM TRANSACTION DATA GENERATION

2.3 == DATS TRANSACTION DATA COLLECTION
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 DATS OPERATOR IS AVAILABLE FROM STAFF
A/27 > PERFORM TRANSACTION DATA COLLECTION

CONTINUED ON THE NEXT PAGE
2.4 - **BATS TRANSACTION DATA PROCESSING**

2.4 IF EXTERNAL TRANSACTION STORAGE AND ANALYSIS IS REQUIRED AND IF

- MEDIA TRANSFER OPERATOR IS AVAILABLE FROM STAFF AND IF
- MEDIA TRANSFER PROCEDURE IS SIMPLE AND IF
- MEDIA TRANSFER IS SHORT AND IF
- OUTPUT CAN BE SPOOLED AND IF
- PRINTING OF SPOOLED OUTPUT IS NOT MORE THAN 1-2 HOURS

A20 - PERFORM DATA PROCESSING (REPORTING AND STORAGE)

2.5 - **BATS DAILY DATA DISTRIBUTION**

2.5 IF PROCEDURE IS SIMPLE

A20 - PERFORM DATA DISTRIBUTION

2.6 - **BATS DAILY OUTPUT USAGE**

2.6 IF OUTPUT VOLUME IS ADEQUATE TO DAILY USAGE AND IF

- FALSE NEGATIVE ERROR RATE IS 10% OR LESS AND IF
- FALSE POSITIVE ERROR RATE IS 5% OR LESS AND IF
- PROVIDED INFORMATION CAN BE READILY USED IN DECISION MAKING AND IF

A30 - VERIFY OUTPUT USAGE IN DECISION MAKING

2.7 - **BATS SYSTEM IS ORGANIZED**

2.7 IF ALL SUBFUNCTIONS IN THIS FUNCTION (CLUSTER) ARE MET

A40 - NO ACTION - NEXT CLUSTER

8. - **DATA UPDATE ORGANIZATION**

8.1 - **UPDATE THE COURSE WIF**

8.1 IF HIGH DEGREE OF DATA-准确度是需要的 AND IF

- USER SYSTEM ACCESS ORGANIZATION IS DEFINED AND IF
- WIF IN CHARGE OF THE SYSTEM HAS ACCESS AND IF

A10 - UPDATE COURSE MASTER FILE INTERACTIVELY

8.2 - **UPDATE THE STUDENT WIF**

8.2 IF HIGH DEGREE OF DATA-准确度是需要的 AND IF

- USER SYSTEM ACCESS ORGANIZATION IS DEFINED AND IF
- STUDENT OFFICE OF THE SCHOOL HAS ACCESS AND IF

A10 - UPDATE STUDENT MASTER FILE INTERACTIVELY

8.3 - **UPDATE ATTENDANCE RELATED DATA**

8.3 IF HIGH DEGREE OF DATA-准确度 is needed AND IF

- USER SYSTEM ACCESS ORGANIZATION IS DEFINED AND IF
- ATTENDANCE WIF IN CHARGE OF THE SCHOOL HAS ACCESS AND IF

A10 - 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 NOT MET
    AND NO ACTION: NEXT CLUSTER

4.== DATA-BASE QUERY ORGANIZATION

4.1 == QUERY THE COURSE MASTER FILE INTERACTIVELY

4.1 IF CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED AND IF
    MP 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> QUERY COURSE MASTER FILE INTERACTIVELY

4.2 == QUERY THE STUDENT MASTER FILE INTERACTIVELY

4.2 IF CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED AND IF
    MP IN CHARGE OF THE SYSTEM HAS ACCESS AND IF
    SECTOR MP, OF THE SCHOOL HAS ACCESS AND IF
    RECORD 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
    A33> QUERY STUDENT MASTER FILE INTERACTIVELY

4.3 == QUERY THE CLASS MASTER FILE INTERACTIVELY

4.3 IF CONFIDENTIALITY AND SECURITY OF THE DATA-BASE IS DEFINED AND IF
    MP IN CHARGE OF THE SYSTEM HAS ACCESS AND IF
    SECTOR MP, 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
    A33> QUERY CLASS MASTER FILE INTERACTIVELY

4.4 == QUERY THE COURSE MASTER FILE PEEK-O-PATCH NODE

4.4 IF OBJECTIVE IS AN EFFICIENT SCHOOL MANAGEMENT AND IF
    CONFIDENTIAL INFORMATION IS NEEDED AND IF
    MP 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
    A33> WRITE A FILE IN COURSE MASTER FILE PEER-O-PATCH NODE
    A33> QUERY COURSE MASTER FILE PEER-O-PATCH NODE
    A33> PRINT PEEK-O-PATCH OUTPUT

CONTINUED ON THE NEXT PAGE
4. AML-TRANSFORM MODELING:

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
criteria for a particular subfunction. They define the constraints
the actions in the particular subfunction must obey. Unless these
criteria 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 $INSTALL$

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 (alternatives) |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|
| C1 | I | I | I | I | I | I | I | I | I | I | I | I | I | I | I | I |
| C2 | | | | | | | | | | | | | | | | |
| C3 | | | | | | | | | | | | | | | | |
| C4 | | | | | | | | | | | | | | | | |

**Clusters**

- **C1**: Organize the body of the system
- **C2**: Organize the regular run of the system
- **C3**: Data Update Organization
- **C4**: Data Base Query 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 line-sharing computer is available
- **P6**: On-line system access is needed
- **P7**: Disk-Storage is available
- **P8**: Interactive query 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**: User system access organization is defined
- **P13**: Confidentiality and security of the data-base is defined
- **P14**: System normalization is defined
- **P15**: 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 Updation language for the course IMF is needed
- **P19**: Interactive updating language for the student IMF is needed
- **P20**: Class master file updating can be done automatically
- **P21**: Immediate information access is needed for decision making
- **P22**: Output volume is small
- **P23**: Interactive query language for the course IMF is needed
- **P24**: Interactive query language for the student IMF is needed
- **P25**: Interactive query language for the class IMF is needed

Continued on the next page.
HIGH VOLUME OF TRUANCY TRANSACTION OCCUR (100-250 A LECTURE)
DATA GENERATION PERIOD IS 30-40 SECONDS
ERROR GENERATION POSSIBILITY IN TRANSACTION DATA IS MINIMAL
INEXPENSIVE OPTICAL MARK READER IS FOUND FOR THE TERMINAL
REGULAR OPTS OPERATOR IS AVAILABLE FROM STAFF
DATA TRANSMISSION IS BAD SOMETIMES
OUTPUT PRINTING ON THE TERMINAL SOMETIMES HAS TO WAIT
VARIABLE LENGTH OUTPUT IS NEEDED
DATA GENERATION PROCESS IS ACCEPTED BY THE TEACHERS
DATA COLLECTION PROCESS IS ACCEPTED BY THE ADMINISTRATION
DATA COLLECTION ROUTE IS ORGANIZED
DATA COLLECTION ERRORS ARE MINIMAL
EXTENSIVE TRANSACTION STORAGE AND ANALYSIS IS REQUIRED
PROCEDURE IS SIMPLE
PROCEDURE IS SHORT
OUTPUT CAN BE SPOOLED
PRINTING OF SPOOLED OUTPUT IS NOT MORE THAN 1-2 HOURS
DISTRIBUTION DELAY IS DEFINED AS 2 HOURS
OUTPUT VOLUME IS ADEQUATE TO DAILY USAGE
FALSE NEGATIVE ERROR RATE IS BELOW 0.1% RATE
FALSE POSITIVE ERROR RATE IS BELOW 0.1% RATE
PROVIDED INFORMATION CAN BE READILY USED IN DECISION MAKING
INFORMATION VALUE IS MAXIMUM OPERATIONAL VS. OVERALL DELAY
THE IN CHARGE OF THE SYSTEM HAS ACCESS
RELEVANT DATA IN THE DATA-BASE ARE AUTOMATICALLY UPDATED
RECORDS OFFICE OF THE SCHOOL HAS ACCESS
SECTOR MAP OF THE SCHOOL HAS ACCESS
CLASS RELATED DATA CHANGED OR SOME ID-CHIPS GOT LOST
DATA-BASE DATA IS UPDATED
CARD PUNCHING AND INTERPRETING IS AVAILABLE
OBJECTIVE IS AN EFFICIENT SCHOOL MANAGEMENT
TEACHERS HAVE ACCESS
STUDENTS AND PARENTS HAVE INDIRECT ACCESS
OBJECTIVE IS AN EFFICIENT STUDENT AND PARENT COUNSELING
OBJECTIVE IS AN EFFICIENT CLASS MANAGEMENT
COARSE INFORMATION IS NEEDED
STUDENT INFORMATION IS NEEDED
CLASS INFORMATION IS NEEDED
ALL SUBFUNCTIONS IN THIS FUNCTION (CLUSTER) ARE NEEDED

CONTINUED ON THE NEXT PAGE
### Modeling of the Overall System Design and Modeling of This System

This table provides an overview of 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 criteria for a particular subfunction. They define the constraints the actions of the particular subfunction must obey. Unless these constraints are satisfied, a 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 'SYSTALT'.

### 17 18 19 20 21 22 23 24 25 26 27 28 (ALTERNATIVES)

<table>
<thead>
<tr>
<th>Clusters</th>
<th>Predicates</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>

### Clusters

- **Clusterr 1**: Organize the body of the system
- **Clusterr 2**: Organize the regular run of the system
- **Clusterr 3**: Data update organization
- **Clusterr 4**: Data-base query 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**: High storage is available
- **P8**: Interactive query 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**: User system access organization is defined
- **P13**: Confidentiality and security of the data-base is defined
- **P14**: System harmonization is defined
- **P15**: High volume of changes occur in the data-base
- **P16**: Data-base changes have to be implemented in 1-2 days
- **P17**: Interactive updating language for the course of, is needed
- **P18**: Interactive updating language for the student of, is needed
- **P19**: Class master file updating can be done automatically
- **P20**: Immediate information access is needed for decision making
- **P21**: Output volume is small
- **P22**: Interactive query language for the course of, is needed
- **P23**: Interactive query language for the student of, is needed
- **P24**: Interactive query language for the class of, is needed

Continued on the next page.
HIGH VOLUME OF TRANSMISSION TRANSACTION OCCUR (100-250 A LECTURE)
DATA GENERATION PERIOD IS 30-40 SECONDS
ERROR GENERATION POSSIBILITY IN TRANSMISSION DATA IS MINIMAL
INEXPENSIVE OPTICAL MARK READER IS FOUND FOR THE TERMINAL
REGULAR DATA OPERATOR IS AVAILABLE FROM STAFF
DATA TRANSMISSION IS BAD SOMETIMES
OUTPUT PRINTING ON THE TERMINAL SOMETIMES HAS TO WAIT
VARIABLE LENGTH OUTPUT IS NEEDED
DATA GENERATION PROCESS IS ACCEPTED BY THE TEACHERS
DATA COLLECTION PROCESS IS ACCEPTED BY THE ADMINISTRATION
DATA COLLECTION ROUTE IS ORGANIZED
DATA COLLECTION ERRORS ARE MINIMAL
EXTENSIVE TRANSACTION STORAGE AND ANALYSIS IS REQUIRED
PROCEDURE IS SIMPLE
PROCEDURE IS SHORT
OUTPUT, CAN BE SPOILED
PRINTING OF SPOILED OUTPUT IS NOT MORE THAN 1-2 HOURS
DISTRIBUTION DELAY IS DEFINED AS 2 HOURS
OUTPUT VOLUME IS ADEQUATE TO DAILY USAGE
FALSE NEGATIVE ERROR RATE IS BELOW 10% RATE
FALSE POSITIVE ERROR RATE IS BELOW 0.1% RATE
INFORMATION CAN BE READILY USED IN DECISION MAKING
INFORMATION VALUE IS MAXIMUM OPERATIONAL VS. OVERALL DELAY
ANM IN CHARGE OF THE SYSTEM HAS ACCESS
RELEVANT DATA IN THE DATABASE ARE AUTOMATICALLY UPDATED
COURSE OFFICE OF THE SCHOOL HAS ACCESS
SECTOR ANM OF THE SCHOOL HAS ACCESS
CLASS RELATED DATA CHANGED OR SOME ABS-CARDS GOT LOST
DATA-BASE DATA IS UPDATED
CARD PRINTING AND INTERPRETING IS AVAILABLE
OBJECTIVE IS AN EFFICIENT SCHOOL MANAGEMENT
TEACHERS HAVE ACCESS
STUDENTS AND PARENTS HAVE INDIRECT ACCESS
OBJECTIVE IS AN EFFICIENT STUDENT AND PARENT COUNSELLING
OBJECTIVE IS AN EFFICIENT CLASS MANAGEMENT
COURSE INFORMATION IS NEEDED
STUDENT INFORMATION IS NEEDED
CLASSED INFORMATION IS NEEDED
ALL SUBFUNCTIONS IN THIS FUNCTION (CLUSTER) ARE MET
CONTINUED ON THE NEXT PAGE
SYSTALL 0 (OR THE OVERALL DESCRIPTION OF THE SYSTEM)

DATA SEQUENCE OF PROCESSES 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 AND EACH OTHER. FIRST WE SET UP THE BASIC SYSTEM THAT IS THE DATA-BASE (DBS), IN IT WE STORE (DMS) AND UPDATE (UPS) UTILITY SUBSYSTEMS. SECOND WE GRAFT TO IT THE ATTENDANCE MONITORING SUBSYSTEM (AMS). FROM THEN ON WE CONCURRENTLY MAINTAIN THE EXACTNESS OF THE DATA-BASE WHILE IMPUTING, READING 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 PROCESSING HAVING THE POSSIBILITY OF BEING DECOMPOSED INTO SUB-PROCEDURES. IN ORDER TO BE ABLE TO ESTABLISH CLEARLY THE HIERARCHICAL RELATIONSHIP BETWEEN THE SUB-ACTION AND THE ACTIONS OF SYSTALL THE FOLLOWING NOTATION WILL BE USED:

SYSTALL ACTION(H) = ACTION(H) = AM> XXXXXXXXXXX

SYSTALL SUB-ACTION(H) OF ACTION(H) =

= SYSTALL ACTION(H-H) = AM-H> XXXXXXXXXXX

SYSTALL SUB-SUB-ACTION(L) OF SUBACTION(H) OF ACTION(N) =

= SYSTALL ACTION(H-H-L) = AM-H-L> XXXXXXXXXXX

FOR ALL CONTRACTIONS PLEASE REFER TO THE TABLE ON FIGURE 27.

THE PRODUCTION CYCLE OF 0 ATS IS DESCRIBED IN DETAIL IN THE FOLLOWING ACTIONS:

SYSTALL A(25) = DAILY SYSTEM INITIALIZATION
SYSTALL A(24) = DAILY TRANSACTION DATA GENERATION
SYSTALL A(27) = DAILY TRANSACTION DATA COLLECTION
SYSTALL A(26) = 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
1.1 SET UP THE BASIC SYSTEM (0 UTS. AND 0 DB.)

1.1= INSTALL TERMINAL IN SCHOOL
1.2= SET UP EMPTY DATABASE
1.3= SET UP ALL OUS, AND UPS. INTERACTIVE PROGRAMS
1.4= UPDATE ALLOCATED FIELDS IN DATABASE

2.2 INITIAL DATABASE VERIFICATION. (0 OUS. AND 0 UPS.)

2.1= VERIFY INFORMATION IN DATABASE
2.2= UPDATE INCORRECT INFORMATION IN DATABASE

2.3= NO UPDATE IS NECESSARY

3.3 SET UP THE ATTENDANCE MONITORING SUBSYSTEM (0 UTS.)

3.1= GIVE OUT ACCESSORIES FOR ATTEND.MONIT.SUBSITE
3.2= SET UP OATS.INTERACT.PROGS.FOR DAILY PRODUCTION CYCLE

4.4 NO REGULAR RUN OF THE ATTENDANCE MONITORING SUBSYSTEM (0 ATS.)

4.1= OATS, DAILY DATA USAGE PARAMETER DEFINITION
4.2= OATS, DAILY TRANSACTION DATA GENERATION
4.3= OATS, DAILY TRANSACTION DATA COLLECTION
4.4= OATS, DAILY TRANSACTION DATA PROCESSING
4.5= OATS, DAILY DATA DISTRIBUTION
4.6= OATS, DAILY OUTPUT USAGE

4.7= NO ACTION: NEXT CLUSTER

5.5 MAINTAIN THE EXACTNESS OF THE SYSTEM'S DATA (0 UPS.)

5.1= UPDATE MDM ATTENDANCE RELATED DATA
5.2= UPDATE ATTENDANCE RELATED DATA
5.3= NO ACTION: NEXT CLUSTER

6.6 INFORMATION EXTRACTING AND REPORTING (0 OUS.)

6.1= QUERY COURSE MASTER FILE INTERACTIVELY
6.2= QUERY STUDENT MASTER FILE INTERACTIVELY
6.3= QUERY CLASS MASTER FILE INTERACTIVELY
6.4= QUERY COURSE MASTER FILE PSEUDOBATCH
6.5= QUERY STUDENT MASTER FILE PSEUDOBATCH
6.6= QUERY CLASS MASTER FILE PSEUDOBATCH
SYSTALL # (OR THE OVERALL DESCRIPTION OF THE SYSTEM)

THIS SEQUENCE OF PROCEDURES PROVIDES AN OVERVIEW OF THE ENTIRE
ATTENDANCE MONITORING SYSTEM (# ATS, # DB). 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.

OTHERWISE, FIRST WE SET UP THE BASIC SYSTEM, THAT IS THE DATA-BASE (# DB)
WITH ITS QUERIES (# QDB) AND UPDATE (# UPS) UTILITY SUBSYSTEMS.
SECOND WE GRAFT IT TO THE ATTENDANCE MONITORING SUBSYSTEM (# ATS).,
FROM THEN ON WE CONCURRENTLY MAINTAIN THE EXACTNESS OF THE DATA-
BASE WHOLE 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 HIERARCHICAL RELATIONSHIP BETWEEN THE SUB-ACTION AND
THE ACTIONS OF SYSTALL THE FOLLOWING NOTATION WILL BE USED:

SYSTALL ACTION(H) = ACTION(H) = AN-XXXXXXXXXX

SYSTALL SUB-ACTION(H) OF ACTION(H) =

= SYSTALL ACTION(H-H) = AN-H-XXXXXXXXXX

SYSTALL SUB-SUBACTION(L) OF SUBACTION(H) OF ACTION(H) =

= SYSTALL ACTION(H-H-L) = AN-H-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(23) = DAILY SYSTEM INITIALIZATION
SYSTALL A(24) = DAILY TRANSACTION DATA GENERATION
SYSTALL A(25) = DAILY TRANSACTION DATA COLLECTION
SYSTALL A(26) = DAILY TRANSACTION DATA PROCESSING
SYSTALL A(27) = DAILY DATA DISTRIBUTION
SYSTALL A(28) = DAILY DATA USAGE VERIFICATION

A 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= INSTALL TERMINAL IN SCHOOL
1.1 IF NOT TERMINAL IS SET UP IN SCHOOL

CONTINUED ON THE NEXT PAGE
1.2= SET UP EMPTY DATA-BASE
  1.2 IF TERMINAL IS SET UP IN SCHOOL AND IF:
  =---------THEN-GO-TO-CLUSTER-> 1
  ' NOT ALL DATA-BASE FILES ARE SET UP ON-LINE

1.3= SET UP ALL INPUT AND UPS, INTERACTIVE PROGRAMS.
  1.3 IF ALL DATA-BASE FILES ARE SET UP ON-LINE AND IF:
  =---------THEN-GO-TO-CLUSTER-> 1
  ' NOT QUERY AND UPDATE SUBSYSTEMS ARE SET UP

1.4= UPDATE ALLOCATED FIELDS IN DATA-BASE
  1.4 IF ALL DATA-BASE FILES ARE SET UP ON-LINE AND IF:
  =---------THEN-GO-TO-CLUSTER-> 2
  ' QUERY AND UPDATE SUBSYSTEMS ARE SET UP AND IF:
  ' NOT ALLOCATED FIELDS IN THE DATA-BASE ARE FILLED

2.2 INITIAL DATA-BASE VERIFICATION. (0 QUS. AND 0 UPS.)
  =--------< 2, 3, 3 >

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

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

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

3.3 SET UP THE ATTENDANCE MONITORING SUBSYSTEM (0 UTS.)
  =--------< 3, 4 >

3.1= GIVE OUT ACCESSORIES FOR ATTENDANCE MONITORING SUBSYSTEM.
  3.1 IF NOT ATTENDANCE MONITORING SUBSYSTEM IS SET UP AND IF:
  =---------THEN-GO-TO-CLUSTER-> 3
  ' DATA-BASE CONTAINS CORRECT MONITORING INFORMATION, AND IF:
  ' NOT TRANSACTION DATA GENERATION KITS ARE READY

3.2= SET UP DATS, INTERACT. PROGS. FOR DAILY PRODUCTION CYCLE
  3.2 IF NOT ATTENDANCE MONITORING SUBSYSTEM IS SET UP AND IF:
  =---------THEN-GO-TO-CLUSTER-> 4
  ' TRANSACTION DATA GENERATION KITS ARE READY

4.4 DO REGULAR RUN OF THE ATTENDANCE MONITORING SUBSYSTEM (0 ATs.)
  =--------< 4, 4, 4, 4, 4, 4, 4, 5 >

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

4.2= DATS, DAILY TRANSACTION DATA GENERATION
  4.2 IF TASK IS ATTENDANCE MONITORING AND IF:
  =---------THEN-GO-TO-CLUSTER-> 4
  ' PRODUCTION CYCLE IS STARTED AND IF:
  ' PRODUCTION DAY IS INITIALIZED TO THE COMPUTER

4.3= DATS, DAILY TRANSACTION DATA COLLECTION
  4.3 IF TASK IS ATTENDANCE MONITORING AND IF:
  =---------THEN-GO-TO-CLUSTER-> 4
  ' PRODUCTION CYCLE IS STARTED AND IF:
  ' TRANSACTION DATA IS GENERATED

CONTINUED ON THE NEXT PAGE
4.4.6 - OATS, DAILY TRANSACTION DATA PROCESSING
   4.4.6.1 - IF TASK IS ATTENDANCE MONITORING AND IF
             PRODUCTION CYCLE IS STARTED AND IF
             TRANSACTION DATA IS COLLECTED

4.5.6 - OATS, DAILY DATA DISTRIBUTION
   4.5.6.1 - IF TASK IS ATTENDANCE MONITORING AND IF
             PRODUCTION CYCLE IS STARTED AND IF
             DATA IS PROCESSED

4.6.6 - OATS, DAILY OUTPUT USAGE
   4.6.6.1 - IF TASK IS ATTENDANCE MONITORING AND IF
             PRODUCTION CYCLE IS STARTED AND IF
             OUTPUT IS DISTRIBUTED AND IF
             ATT USAGE PARAMETERS ARE DETERMINED

4.7.6 - NO ACTION, NEXT CLUSTER
   4.7.6.1 - IF BUT TASK IS ATTENDANCE MONITORING

5.5 - MAINTAIN THE EXACTNESS OF THE SYSTEMS DATA (6 UPS. )
   5.1 - NO ACTION

5.1.1 - UPDATE NON ATTENDANCE RELATED DATA
   5.1.1.1 - IF TASK IS UPDATING THE DATA-BASE AND IF
             CHANGE INVOIUES NON OATS, RELATED DATA

5.2 - UPDATE ATTENDANCE RELATED DATA
   5.2.1 - IF TASK IS UPDATING THE DATA-BASE AND IF
             CHANGE INVOLVES OATS, RELATED DATA

5.3 - NO ACTION, NEXT CLUSTER
   5.3.1 - IF BUT TASK IS UPDATING THE DATA-BASE

6.6 - INFORMATION EXTRACTING AND REPORTING (6 UPS. )
   6.1 - QUERY COURSE MASTER FILE INTERACTIVELY
         6.1.1 - IF TASK IS QUERY THE DATA-BASE AND IF
                 COURSE INFORMATION IS NEEDED AND IF
                 SMALL AMOUNT OF OUTPUT IS NEEDED

6.2 - QUERY STUDENT MASTER FILE INTERACTIVELY
         6.2.1 - IF TASK IS QUERY THE DATA-BASE AND IF
                 STUDENT INFORMATION IS NEEDED AND IF
                 SMALL AMOUNT OF OUTPUT IS NEEDED

6.3 - QUERY CLASS MASTER FILE INTERACTIVELY
         6.3.1 - IF TASK IS QUERY THE DATA-BASE AND IF
                 CLASS INFORMATION IS NEEDED AND IF
                 SMALL AMOUNT OF OUTPUT IS NEEDED

6.4 - QUERY COURSE MASTER FILE PSEUDOMATCH
         6.4.1 - IF TASK IS QUERY THE DATA-BASE AND IF
                 COURSE INFORMATION IS NEEDED AND IF
                 LARGE AMOUNT OF OUTPUT IS NEEDED

CONTINUED ON THE NEXT PAGE
6.5* QUERY STUDENT MASTER FILE PSEUDOMATCH
   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* QUERY CLASS MASTER FILE PSEUDOMATCH
   6.6* IF TASK IS QUERRY THE DATA-BASE AND IF
        CLASS INFORMATION IS NEEDED AND IF
        LARGE AMOUNT OF OUTPUT IS NEEDED

= ---THEN-GO-TO-CLUSTER-> 4


Systall - 0 (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 other. First we set up the basic system that is the data-base (DB) with its query (QDB) and update (UDP) 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 hierarchical relationship between the sub-action and the actions of Systall the following notation will be used:

Systall Action(H) = Action(H) = AN >> 00000

Systall Sub-Action(H) of Action(H) =
   Systall Action(H-H) = AN-H 00000

Systall Sub-Subaction(H) of Subaction(H) of Action(H) =
   Systall Action(H-H-H) = AN-H-H 00000

For all contractions please refer to the table on figure 27.

The production cycle of an 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

1.1 SET UP THE BASIC SYSTEM ( Θ UTS, AND Θ DB, )

1.1 INSTALL TERMINAL IN SCHOOL
1.1 IF NOT TERMINAL IS SET UP IN SCHOOL
   A1> INSTALL THE TERMINAL IN SCHOOL

   CONTINUE ON THE NEXT PAGE
1.2= SET UP EMPTY DATA-BASE
   1.2 IF TERMINAL IS SET UP IN SCHOOL AND IF
     NOT ALL DATA-BASE FILES ARE SET UP ON-LINE
     A01 SET UP THE COURSE MASTER FILE
     A02 SET UP THE STUDENT MASTER FILE
     A03 SET UP THE CLASS MASTER FILE

1.3= SET UP ALL OOPS. AND OMS. INTERACTIVE PROGRAMS.
   1.3 IF ALL DATA-BASE FILES ARE SET UP ON-LINE AND IF
     NOT QUERY AND UPDATE SUBSYSTEMS ARE SET UP
     A01 SET UP COURSE MASTER FILE UPDATE PROGRAM
     A02 SET UP STUDENT MASTER FILE UPDATE PROGRAM
     A03 SET UP COURSE MASTER FILE QUERY PROGRAM
     A04 SET UP STUDENT MASTER FILE QUERY PROGRAM
     A05 SET UP CLASS MASTER FILE QUERY PROGRAM

1.4= UPDATE ALLOCATED FIELDS IN DATA-BASE
   1.4 IF ALL DATA-BASE FILES ARE SET UP ON-LINE AND IF
     QUERY AND UPDATE SUBSYSTEMS ARE SET UP, AND IF
     NOT ALLOCATED FIELDS IN THE DATA-BASE ARE FILLED
     A010 UPDATE COURSE MASTER FILE INTERACTIVELY
     A011 UPDATE STUDENT MASTER FILE INTERACTIVELY
     A012 UPDATE CLASS MASTER FILE INTERACTIVELY

2.2 INITIAL DATA-BASE VERIFICATION. ( # OOPS. AND # UPS. )

2.1= VERIFY INFORMATION IN DATA-BASE
   2.1 IF ALL DATA-BASE FILES ARE SET UP ON-LINE AND IF
     NOT DATA-BASE CONTENT IS VERIFIED
     A020 QUERY COURSE MASTER FILE PSEUDOMATCH NODE
     A021 QUERY COURSE MASTER FILE DATA
     A022 QUERY STUDENT MASTER FILE PSEUDOMATCH NODE
     A023 QUERY STUDENT MASTER FILE DATA
     A024 QUERY CLASS MASTER FILE PSEUDOMATCH NODE
     A025 QUERY CLASS MASTER FILE DATA

2.2= UPDATE INCORRECT INFORMATION IN DATA-BASE
   2.2 IF DATA-BASE CONTENT IS VERIFIED AND IF
     ERROR IN THE DATA-BASE
     A030 UPDATE COURSE MASTER FILE INTERACTIVELY
     A031 UPDATE STUDENT MASTER FILE INTERACTIVELY
     A032 UPDATE CLASS MASTER FILE INTERACTIVELY

2.3= NO UPDATE IS NECESSARY
   2.3 IF DATA-BASE CONTENT IS VERIFIED AND IF
     NOT ERROR IN THE DATA-BASE
     A040 NO ACTION—NEXT CLUSTER

CONTINUED ON THE NEXT PAGE
3.3 SET UP THE ATTENDANCE MONITORING SUBSYSTEM ( # UTS. ) === 3 < 3, 4 >
3.1* GIVE OUT ACCESSORIES FOR ATTENDANCE MONITORING SUBSYSTEM. == --THEN-GO-TO-CLUSTER-> 3
3.1 IF NOT ATTENDANCE MONITORING SUBSYSTEM IS SET UP AND IF
DATA-BASE CONTAINS CORRECT NON TRAVEL INFORM. AND IF
MISSING TRANSACTION DATA GENERATION KITS ARE READY
A12) SET UP CLASSES MASTER FILE PSEUDOMATCH ROUTE
A13) PRODUCE # ABS-CARDS FOR THE CLASS
A14) PRODUCE TEACHERS DATA GENERATION KITS
A15) SET UP DATA COLLECTION ROUTES IN SCHOOL
A16) INSTALL OPTICAL RACK READER ON TERMINAL
A17) VERIFY DATA DISTRIBUTION ROUTES
A20) DETERMINE DATA USAGE PARAMETERS

3.2 SET UP #ATS-DAILY PROCESSES 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 DAILY DATA COLLECTION PSEUDOMATCH PROGRAM
A23) SET UP #ATS-D, DATA PROCESSING PSEUDOMATCH PROGRAM
A24) SET UP PROCEDURE TO PRINT SPOOLED OUTPUT

4.4 DO REGULAR RUN OF THE ATTENDANCE MONITORING SUBSYSTEM ( # ATS. ) == 3 < 4, 4, 4, 4, 4, 5 >
4.1 SET #ATS-DAILY DATA USAGE PARAMETER DEFINITION
4.1 IF TASK IS ATTENDANCE MONITORING AND IF
ATTENDANCE MONITORING SUBSYSTEM IS SET UP AND IF
MISSING TRANSACTION CYCLE IS STARTED AND IF
DATA GENERATION KITS ARE DETERMINED
A25) PERFORM DAILY SYSTEM INITIALIZATION

4.2 SET #ATS-DAILY TRANSACTION DATA GENERATION
4.2 IF TASK IS ATTENDANCE MONITORING AND IF
MISSING TRANSACTION CYCLE IS STARTED AND IF
TRANSACTION DATA IS GENERATED
A26) PERFORM TRANSACTION DATA GENERATION

4.3 SET #ATS-DAILY TRANSACTION DATA COLLECTION
4.3 IF TASK IS ATTENDANCE MONITORING AND IF
MISSING TRANSACTION CYCLE IS STARTED AND IF
TRANSACTION DATA IS COLLECTED
A27) PERFORM TRANSACTION DATA COLLECTION

4.4 SET #ATS-DAILY TRANSACTION DATA PROCESSING
4.4 IF TASK IS ATTENDANCE MONITORING AND IF
MISSING TRANSACTION CYCLE IS STARTED AND IF
TRANSACTION DATA IS COLLECTED
A28) PERFORM DATA PROCESSING (REPORTING AND STORAGE)
A29) PRINT SPOOLED OUTPUT

CONTINUED ON THE NEXT PAGE
4.5 a) DAILY DATA DISTRIBUTION
   4.5 IF TASK IS ATTENDANCE MONITORING AND IF
   PRODUCTION CYCLE IS STARTED AND IF
   DATA IS PROCESSED
   → A29: PERFORM DATA DISTRIBUTION

4.6 a) DAILY OUTPUT USAGE
   4.6 IF TASK IS ATTENDANCE MONITORING AND IF
   PRODUCTION CYCLE IS STARTED AND IF
   OUTPUT IS DISTRIBUTED AND IF
   DATA USAGE PARAMETERS ARE DETERMINED
   → A30: VERIFY OUTPUT USAGE IN DECISION MAKING
   → A20: DETERMINE DATA USAGE PARAMETERS

4.7 b) NO ACTION: NEXT CLUSTER
   4.7 IF NOT TASK IS ATTENDANCE MONITORING
   → A44: NO ACTION: NEXT CLUSTER

5.1 a) UPDATE STUDENT ATTENDANCE RELATED DATA
   5.1 IF TASK IS UPDATING THE DATA-BASE AND IF
   CHANGE INVOLVES STUDENT ATTENDANCE RELATED DATA
   → A19: UPDATE STUDENT ATTENDANCE RELATED DATA
   → A12: UPDATE CLASS ATTENDANCE RELATED DATA

5.2 a) UPDATE ATTENDANCE RELATED DATA
   5.2 IF TASK IS UPDATING THE DATA-BASE AND IF
   CHANGE INVOLVES ATTENDANCE RELATED DATA
   → A19: UPDATE STUDENT ATTENDANCE RELATED DATA
   → A12: UPDATE CLASS ATTENDANCE RELATED DATA
   → A31: UPDATE NEW & ABSENT CARDS
   → A32: UPDATE TEACHERS ATTENDANCE DATA GENERATION KITS

5.3 b) NO ACTION: NEXT CLUSTER
   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 (9 OUS.)

6.1 QUERY COURSE MASTER FILE INTERACTIVELY
   6.1.1 IF TASK IS QUERY THE DATA-BASE AND IF
         COURSE INFORMATION IS NEEDED AND IF
         SMALL AMOUNT OF OUTPUT IS NEEDED
         A33> QUERY COURSE MASTER FILE INTERACTIVELY

6.2 QUERY STUDENT MASTER FILE INTERACTIVELY
   6.2.2 IF TASK IS QUERY THE DATA-BASE AND IF
         STUDENT INFORMATION IS NEEDED AND IF
         SMALL AMOUNT OF OUTPUT IS NEEDED
         A34> QUERY STUDENT MASTER FILE INTERACTIVELY

6.3 QUERY CLASS MASTER FILE INTERACTIVELY
   6.3.3 IF TASK IS QUERY THE DATA-BASE AND IF
         CLASS INFORMATION IS NEEDED AND IF
         SMALL AMOUNT OF OUTPUT IS NEEDED
         A35> QUERY CLASS MASTER FILE INTERACTIVELY

6.4 QUERY COURSE MASTER FILE PSEUDOBATCH
   6.4.4 IF TASK IS QUERY 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 QUERY LANGUAGE
         A37> QUERY COURSE MASTER FILE PSEUDOBATCH NODE
         A38> PRINT SPOOLED OUTPUT

6.5 QUERY STUDENT MASTER FILE PSEUDOBATCH
   6.5.5 IF TASK IS QUERY 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 QUERY LANGUAGE
         A40> QUERY STUDENT MASTER FILE PSEUDOBATCH NODE
         A41> PRINT SPOOLED OUTPUT

6.6 QUERY CLASS MASTER FILE PSEUDOBATCH
   6.6.6 IF TASK IS QUERY THE DATA-BASE AND IF
         CLASS INFORMATION IS NEEDED AND IF
         LARGE AMOUNT OF OUTPUT IS NEEDED
         A42> WRITE A FILE IN CLASS MASTER FILE QUERY LANGUAGE
         A43> QUERY CLASS MASTER FILE PSEUDOBATCH NODE
         A44> PRINT SPOOLED OUTPUT
3. AML-TRANSFORM SYSTALL

SYSTALL 0 (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 DEFINES THE LOGICAL INTER-RELATION AND OPERATIONAL SEQUENCE OF THE SUBSYSTEMS AMONG EACH OTHER. FIRST WE SET UP THE BASIC SYSTEM THAT IS THE DATA-BASE (DB), WITH ITS CONTENTS (CMS) AND UPDATE (UPS) UTILITY SUBSYSTEMS. THEN WE WANT TO IT THE ATTENDANCE MONITORING SUBSYSTEM (AMS), THEN WE 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 HAVING THE POSSIBILITY OF BEING DECOMPOSED INTO SUB-PROCEDURES. IN ORDER TO BE ABLE TO EXIST CLEARLY THE HIERARCHICAL RELATIONSHIP BETWEEN THE SUB-ACTION AND THE ACTIONS OF SYSTALL THE FOLLOWING NOTATION WILL BE USED:

SYSTALL ACTION(N) = ACTION(N) = AM; XXXXXXXX

SYSTALL SUB-ACTION(k) OF ACTION(N) =
   = SYSTALL ACTION(N-k) = AN-MD XXXXXXXX

SYSTALL SUB-SUB-ACTION(l) OF SUB-ACTION(M) OF ACTION(N) =
   = SYSTALL ACTION(N-M-L) = AN-M-L XXXXXXXX

FOR ALL ACTIONS PLEASE REFER TO THE TABLE ON FIGURE 27.

THE PRODUCTION CYCLE OF THE ATS IS DESCRIBED IN DETAIL IN THE FOLLOWING ACTIONS:

SYSTALL A(25) = DAILY SYSTEM INITIALIZATION
SYSTALL A(24) = DAILY TRANSACTIONS DATA GENERATION
SYSTALL A(27) = DAILY TRANSACTIONS DATA COLLECTION
SYSTALL A(28) = DAILY TRANSACTIONS 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
CLUSTERS:

1. SET UP THE BASIC SYSTEM (0 UTS, AND 0 DB.)
2. INITIAL DATA-BASE VERIFICATION (0 UTS, AND 0 UPS.)
3. ATTENDANCE MONITORING SUBSYSTEM (0 UTS. )
4. REGULAR RUN OF THE ATTENDANCE MONITORING SUBSYSTEM (0 ATS.)
5. MAINTAINED EXACTNESS OF THE SYSTEMS DATA (0 UPS.)
6. INFORMATION EXTRACTING AND REPORTING (0 UTS.)

PREDICATES:

- TERMINAL IS SET UP IN SCHOOL
- ALL DATA-BASE FILES ARE SET UP ON-LINE
- QUERY AND UPDATE QUERIES ARE SET UP
- ALLOCATED FIELDS IN THE DATA-BASE ARE FILLED
- DATA-BASE CONTENT IS VERIFIED
- ERROR IN THE DATA-BASE
- ATTENDANCE MONITORING SUBSYSTEM IS SET UP
- DATA-BASE CONTAINS CORRECT NON-TRIVIAL INFORMATION
- TRANSACTION DATA GENERATION KITS ARE READY
- TASK IS ATTENDANCE MONITORING
- PRODUCTION DAY IS INITIALIZED TO THE COMPUTER
- TRANSACTION DATA IS GENERATED
- TRANSACTION DATA IS COLLECTED
- DATA IS PROCESSED
- OUTPUT IS DISTRIBUTED
- USAGE PARAMETERS ARE DETERMINED
- TASK IS UPDATING THE DATA-BASE
- CURRICULUM INVOLVES NON-ATTS., RELATED DATA
- CURRICULUM ADOPTS DATA., RELATED DATA
- TASK IS QUERY THE DATA-BASE
- STUDENT INFORMATION IS NEEDED
- COURSE INFORMATION IS NEEDED
- CLASS INFORMATION IS NEEDED
- SMALL AMOUNT OF OUTPUT IS NEEDED
- LARGE AMOUNT OF OUTPUT IS NEEDED
- END OF THE YEAR

ACTIONS:

- INSTALL THE TERMINAL IN SCHOOL
- SET UP THE COURSE MASTER FILE
- SET UP THE STUDENT MASTER FILE
- SET UP THE CLASS MASTER FILE
- SET UP THE COURSE MASTER FILE UPDATER PROGRAM
- SET UP STUDENT MASTER FILE UPDATER PROGRAM
- SET UP COURSE MASTER FILE QUERY PROGRAM
- SET UP STUDENT MASTER FILE QUERY PROGRAM
- UPDATE COURSE MASTER FILE INTERACTIVELY
- UPDATE STUDENT MASTER FILE INTERACTIVELY
- VERIFY COURSE MASTER FILE DATA
- VERIFY STUDENT MASTER FILE DATA
- VERIFY CLASS MASTER FILE DATA

CONTINUED ON THE NEXT PAGE
A16 3
A17 5
A18 6
A19 7
A20 1
A21 2
A22 3
A23 4
A24 1
A25 1
A26 1
A27 1
A28 1
A29 1
A30 1
A31 1
A32 1
A33 1
A34 1
A35 1
A36 1
A37 1
A38 1
A39 1
A40 1
A41 1
A42 1
A43 1
A44 1

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
A21) SET UP DATA INITIALIZATION INTERACTIVE PROGRAM
A22) SET UP DATA COLLECTION PSEUDOBATCH PROGRAM
A23) SET UP I/O DS, DATA PROCESSING PSEUDOBATCH PROGRAM
A24) SET UP PROCEDURE TO PRINT SPOOLED OUTPUT
A25) PERFORM I/O SYSTEM INITIALIZATION
A26) PERFORM TRANSACTION DATA GENERATION
A27) PERFORM TRANSACTION DATA COLLECTION
A28) PERFORM DATA PROCESSING (REPORTING AND STORAGE)
A29) PERFORM DATA DISTRIBUTION
A30) VERIFY OUTPUT USAGE IN DECISION MAKING
A31) PUNCH NEW 8-ABS-CARDS
A32) UPDATE TEACHERS DATA GENERATION KITS
A33) QUERY COURSE MASTER FILE INTERACTIVELY
A34) QUERY STUDENT MASTER FILE INTERACTIVELY
A35) QUERY CLASS MASTER FILE INTERACTIVELY
A36) WRITE A FILE IN COURSE MASTER FILE QUERY LANGUAGE
A37) QUERY COURSE MASTER FILE PSEUDOBATCH NODE
A38) PRINT SPOOLED OUTPUT
A39) WRITE A FILE IN STUDENT MASTER FILE QUERY LANGUAGE
A40) QUERY STUDENT MASTER FILE PSEUDOBATCH NODE
A41) WRITE A FILE IN CLASS MASTER FILE QUERY LANGUAGE
A42) QUERY CLASS MASTER FILE PSEUDOBATCH NODE
A43) PRODUCE 0 ABS-CARDS FOR THE CLASS
A44) NO ACTION- NEXT CLUSTER
SYSTALL 0 (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 SHOES 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 QUERY (QS) 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 CONTINUALLY MONITORING 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 HIERARCHICAL 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(N) OF ACTION(N) =

= SYSTALL ACTION(N-N) = AN-HD XXXXXXXXX

SYSTALL SUB-SUB-ACTION(L) OF SUB-ACTION(N) OF ACTION(N) =

= SYSTALL ACTION(N-N-L) = AN-H-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
### Clusters -

<table>
<thead>
<tr>
<th>Clusters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1 Set up the basic system (8 UTS, and 8 DB)</td>
</tr>
<tr>
<td>C2</td>
<td>2 Initial data base verification (8 UTS, and 8 UPS)</td>
</tr>
<tr>
<td>C3</td>
<td>3 3 Set up monitoring subsystem (8 UTS)</td>
</tr>
<tr>
<td>C4</td>
<td>4 Do regular run of the attendance monitoring subsystem (8 ATs)</td>
</tr>
<tr>
<td>C5</td>
<td>5 Maintain the exactness of the systems data (8 UPS)</td>
</tr>
<tr>
<td>C6</td>
<td>6 Information extracting and reporting (8 UTS)</td>
</tr>
</tbody>
</table>

### Predicates -

<table>
<thead>
<tr>
<th>Predicates</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Terminal is set up in school</td>
</tr>
<tr>
<td>P2</td>
<td>All data base files are set up on-line</td>
</tr>
<tr>
<td>P3</td>
<td>Query and update subsystem are set up</td>
</tr>
<tr>
<td>P4</td>
<td>Allocated fields in the data base are filled</td>
</tr>
<tr>
<td>P5</td>
<td>Data base content is verified</td>
</tr>
<tr>
<td>P6</td>
<td>Error in the data base</td>
</tr>
<tr>
<td>P7</td>
<td>Attendance monitoring subsystem is set up</td>
</tr>
<tr>
<td>P8</td>
<td>Data base contains correct non transient information</td>
</tr>
<tr>
<td>P9</td>
<td>Transaction data generation kits are ready</td>
</tr>
<tr>
<td>P10</td>
<td>Task is attendance monitoring</td>
</tr>
<tr>
<td>P11</td>
<td>Production cycle is started</td>
</tr>
<tr>
<td>P12</td>
<td>Transaction data is initialized to the computer</td>
</tr>
<tr>
<td>P13</td>
<td>Transaction data is generated</td>
</tr>
<tr>
<td>P14</td>
<td>Transaction data is collected</td>
</tr>
<tr>
<td>P15</td>
<td>Data is processed</td>
</tr>
<tr>
<td>P16</td>
<td>Output is distributed</td>
</tr>
<tr>
<td>P17</td>
<td>Usage parameters are determined</td>
</tr>
<tr>
<td>P18</td>
<td>Task is updating the data base</td>
</tr>
<tr>
<td>P19</td>
<td>Change involves non data related data</td>
</tr>
<tr>
<td>P20</td>
<td>Change involves data related data</td>
</tr>
<tr>
<td>P21</td>
<td>Task is query the data base</td>
</tr>
<tr>
<td>P22</td>
<td>Student information is needed</td>
</tr>
<tr>
<td>P23</td>
<td>Course information is needed</td>
</tr>
<tr>
<td>P24</td>
<td>Class information is needed</td>
</tr>
<tr>
<td>P25</td>
<td>Small amount of output is needed</td>
</tr>
<tr>
<td>P26</td>
<td>Large amount of output is needed</td>
</tr>
</tbody>
</table>

### Actions -

<table>
<thead>
<tr>
<th>Actions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Install the terminal in school</td>
</tr>
<tr>
<td>A2</td>
<td>Set up the course master file</td>
</tr>
<tr>
<td>A3</td>
<td>Set up the student master file</td>
</tr>
<tr>
<td>A4</td>
<td>Set up the class master file</td>
</tr>
<tr>
<td>A5</td>
<td>Set up course master file update program</td>
</tr>
<tr>
<td>A6</td>
<td>Set up course master file update program</td>
</tr>
<tr>
<td>A7</td>
<td>Set up course master file query program</td>
</tr>
<tr>
<td>A8</td>
<td>Set up student master file query program</td>
</tr>
<tr>
<td>A9</td>
<td>Set up class master file query program</td>
</tr>
<tr>
<td>A10</td>
<td>Update course master file interactively</td>
</tr>
<tr>
<td>A11</td>
<td>Update student master file interactively</td>
</tr>
<tr>
<td>A12</td>
<td>Update class master file interactively</td>
</tr>
<tr>
<td>A13</td>
<td>Verify course master file data</td>
</tr>
<tr>
<td>A14</td>
<td>Verify student master file data</td>
</tr>
<tr>
<td>A15</td>
<td>Verify class master file data</td>
</tr>
</tbody>
</table>

Continued on the next page
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
A21  SET UP INITIALIZATION INTERACTIVE PROGRAM
A22  SET UP DATA COLLECTION PSEUDOBATCH PROGRAM
A23  SET UP DATA PROCESSING PSEUDOBATCH PROGRAM
A24  SET PROCEDURE TO PRINT SPOOLED OUTPUT
A25  PERFORM INITIALIZATION
A26  PERFORM TRANSACTION DATA GENERATION
A27  PERFORM TRANSACTION DATA COLLECTION
A28  PERFORM DATA PROCESSING (REPORTING AND STORAGE)
A29  PERFORM DATA DISTRIBUTION
A30  VERIFY OUTPUT USAGE IN DECISION MAKING
A31  PUNCH NEW & NEW CARDS
A32  UPDATE TEACHERS DATA GENERATION KITS
A33  QUERY COURSE MASTER FILE INTERACTIVELY
A34  QUERY STUDENT MASTER FILE INTERACTIVELY
A35  QUERY CLASS MASTER FILE INTERACTIVELY
A36  WRITE A FILE IN COURSE MASTER FILE QUERY LANGUAGE
A37  WRITE A FILE IN STUDENT MASTER FILE QUERY LANGUAGE
A38  WRITE A FILE IN CLASS MASTER FILE QUERY LANGUAGE
A39  PRODUCE & NEW CARDS FOR THE CLASS
A40  NO ACTION—NEXT CLUSTER
9. AML-Routine Initializations

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 used, 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

1.1 == Enter Procedure

2. == Evict Old Data, Enter New Parameters

2.1 == Warning, Attempt to Initialize Twice a Day

2.2 == Enter New Production Cycle Specifications

2.3 == Specification Error Recovery

2.4 == Output Selection Parameter Definition

3. == Store Parameters

3.1 == Output Selection Parameter Error Recovery

3.2 == Finalize Initialization

---3 < 2 >

---3 < 0, 2, 2, 3 >

---3 < 3, 0 >

---3 < 3 >
SYSTART A(25) = A25> = PERFORM DAILY DATA USAGE DEFINITION

AND INITIALIZATION
= INITIALIZATION WITH 18 ACTIONS
= SYSTART A26-(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
1.1=== ENTER PROCEDURE
1.1 IF START

2.=== EVICT OLD DATA, ENTER NEW PARAMETERS
2.1=== WARNING: ATTEMPT TO INITIALIZE TWICE A DAY
2.1 IF PRODUCTION CYCLE DAY IS ALREADY INITIALIZED

2.2=== ENTER NEW PRODUCTION CYCLE SPECIFICATIONS
2.2 IF NOT PRODUCTION CYCLE DAY IS ALREADY INITIALIZED AND IF
NOT ERROR RECOVERY OF ENTERED DATA

2.3=== SPECIFICATION ERROR RECOVERY
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
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
3.1=== OUTPUT SELECTION PARAMETER ERROR RECOVERY
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
3.2 IF NOT ERROR RECOVERY OF ENTERED DATA AND IF
VALID OUTPUT PARAMETERS
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
SYSTEM 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,
THOSE VALUES ARE STORED IN A SPECIAL BUFFER RECORD OF THE
STUDENT MASTER FILE UNTIL DATA PROCESSING TIME.

1.== START

1.1== ENTER PROCEDURE

1.1 IF START

A25-1> ENTER COMMAND = CALL+ABS.COM

2.== EVICT OLD DATA, ENTER NEW PARAMETERS

2.1== WARNING: ATTEMPT TO INITIALIZE TWICE A DAY

2.1 IF PRODUCTION CYCLE DAY IS ALREADY INITIALIZED

A25-2> DISPLAY LAST PRODUCTION CYCLE SPECIFICATIONS AND PARAMETERS

A25-2> WRITE ERROR MESSAGE

A25-2> INTERRUPT EXECUTION

2.2== ENTER NEW PRODUCTION CYCLE SPECIFICATIONS

2.2 IF NOT PRODUCTION CYCLE DAY IS ALREADY INITIALIZED AND IF

A25-3> DISPLAY ERROR RECOVERY OF ENTERED DATA

A25-3> DISPLAY LAST PRODUCTION CYCLE SPECIFICATIONS AND PARAMETERS

A25-3> CLEAR PREVIOUS VALID TRANSACTION FILE

A25-3> SET UP FLAG

A25-4> OPEN STUDENT MASTER FILE

A25-5> DETERMINE AAGS-RECORD FROM STUDENT MASTER FILE

A25-6> ENTER PRESENT DAYS SPECIFICATIONS

A25-7> ENTER SCHOOL DAY

A25-7> ENTER BENEFICIARY

A25-10> ENTER SCHOOL WEEK

A25-11> SET DATE FROM SYSTEM

A25-12> DISPLAY AND VERIFY ENTERED SPECIFICATIONS

CONTINUED ON THE NEXT PAGE
2.3 SPECIFICATION ERROR RECOVERY
   2.3 IF NOT PRODUCTION CYCLE DAY IS ALREADY INITIALIZED AND IF
       ERROR RECOVERY OF ENTERED DATA AND IF
       NOT VALID SPECIFICATIONS
       A25-13> DISPLAY LAST PRODUCTION CYCLE SPECIFICATIONS AND PARAMETERS
       A25-17> ENTER PRESENT DAYS SPECIFICATIONS
       A25-19> ENTER SCHOOL DAY
       A25-1c> ENTER SEMESTER
       A25-1d> ENTER SCHOOL WEEK
       A25-1e> SET DATE FROM SYSTEM
       A25-1f> DISPLAY AND VERIFY ENTERED SPECIFICATIONS

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

3.=== STORE PARAMETERS
   3.1 OUTPUT SELECTION PARAMETER ERROR RECOVERY
   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
   3.2 IF NOT ERROR RECOVERY OF ENTERED DATA AND IF
       VALID OUTPUT PARAMETERS
       A25-13> STORE ENTERED VALUES IN #MOS-REC OF #STUD-MAST-FILE
       A25-16> CLOSE STUDENT MASTER FILE
SYMPTOM A(25) = A(25) = PERFORM DAILY DATA USAGE DEFINITION

AND INITIALIZATION

= INITIALIZATION WITH 18 ACTIONS
= SYMPTOM A(25) = 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 2 2 3 : 0 == START
C2 0 2 2 3 : 0 == EVICT OLD DATA, ENTER NEW PARAMETERS
C3 : : : : 3 0 0 == STORE PARAMETERS

PREDICATES

P2 Y Y Y Y Y Y PRODUCTION CYCLE DAY IS ALREADY INITIALIZED
P3 Y Y Y Y Y Y ENTER RECOVERY OF ENTERED DATA
P4 Y Y Y Y Y Y VALID SPECIFICATIONS
P5 Y Y Y Y Y Y VALID OUTPUT PARAMETERS

ACTIONS

A25-1 ENTER COMMAND = CALL, ABSCON
A25-2 PURGE PREVIOUS VALID-TRANSACTION FILE
A25-3 SET UP FLAG
A25-4 OPEN STUDENT MASTER FILE
A25-5 ENTER OUTPUT SELECTION PARAMETERS
A25-6 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 SET DATE FROM SYSTEM
A25-12 DISPLAY AND VERIFY ENTERED SPECIFICATIONS
A25-13 ENTER OUTPUT SELECTION PARAMETERS
A25-14 DISPLAY AND VERIFY ENTERED PARAMETERS
A25-15 STORE ENTERED VALUES IN #ABS-REC OF #STUD-MAST-FILE
A25-16 CLOSE STUDENT MASTER FILE
A25-17 ENTER ERROR MESSAGE
A25-18 INTERRUPT EXECUTION
### 10. ABL-ROUTINE GENERDATA:

**Systall A(32) = 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.
2. Teachers take the roll call with the class list.
3. Each student absent has his name-card removed from the class deck and put in the pouch reserved for the absent students on the data generation kit. These cards are pre-identified and the entire rollcall takes only 30-60 seconds of the teacher's time.
4. Data generation kit is readied for collection.

---

1. **== START ==**
   1.1 **-> PICK UP NECESSARY MATERIAL**
   2. **-> DATA GENERATION**
      2.1 **-> UPDATE INDIVIDUAL DATA GENERATION KIT**
      2.2 **-> DATA GENERATION**

```plaintext
== == < 2 >
-- THEN-GO-TO-CLUSTER-> 2
== == < 2 > 0
-- THEN-GO-TO-CLUSTER-> 0
```
10. AML-Routine GENERDATA:

SYSTALL A(26) = A26> = GENERATE TRANSACTION DATA
= GENERDATA WITH 13 ACTIONS
= SYSTALL A(26) = (1..13)

THIS PROCEDURE PERFORMS THE FOLLOWING-
1. TRANSACTION DATA IS GENERATED BY THE TEACHERS AT EVERY LECTURE
2. WITH THE HELP OF THE DATA GENERATION KIT.
3. TEACHERS TAKE THE CALL WITH THE CLASS LIST
4. EACH STUDENT ACQUIRES HIS MARS-CARD REMOVED FROM THE CLASS' DECK
5. AND PUT IN THE POCKET RESERVED FOR THE ABSENT STUDENTS ON THE DATA
6. GENERATION KIT. THESE CARDS ARE PREIDENTIFIED AND THE ENTRÉ
7. CALL TAKES ONLY 30-60 SECONDS OF THE TEACHER'S TIME.
8. DATA GENERATION KIT IS READY FOR COLLECTION

1. START
   1.1 PICK UP NECESSARY MATERIAL
   1.1P NO CONDITION

2. DATA GENERATION
   2.1 UPDATE INDIVIDUAL DATA GENERATION KIT
   2.2 IF DATA GENERATION KIT IS UPDATED
   2.2 IF DATA GENERATION KIT IS UPDATED AND IF
   2.2 IF DATA GENERATION KIT IS UPDATED AND IF
   THE PROCESS IS SIMPLE AND ACCEPTED BY TEACHERS
   THE PROCESS IS SIMPLE AND ACCEPTED BY TEACHERS AND IF
   FAULTY DATA GENERATION POSSIBILTY IS MINIMAL AND ACCEPTABLE
   IT TAKES LESS THAN A MINUTE

   ==< 2 ==
   THEN-GO-TO-CLUSTER-> 2

   ==< 2, 0 ==
   THEN-GO-TO-CLUSTER-> 0
SYSTALL A(26) = A26 = GENERATE TRANSACTION DATA
= 13 ACTIONS
= SYSTALL A26-(1..13)

THIS PROCEDURE PERFORMS THE FOLLOWING:
1. TRANSACTION DATA IS GENERATED AT 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 NAME-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 30-45 SECONDS OF THE TEACHER'S TIME.
4. DATA GENERATION KIT IS READY FOR COLLECTION

1.=== START
   1.1=== PICK UP NECESSARY MATERIAL
       1.1P NO CONDITIONS
       A24-1) PICK UP DATA GENERATION KIT FROM MAILBOX
       A24-2) PICK UP NEW CLASS LISTS
       A24-3) PICK UP NEW NAME-CARDS
       A24-4) GO TO CLASS

2.=== DATA GENERATION
   2.1=== UPDATE INDIVIDUAL DATA GENERATION KIT
       2.1 IF DATA GENERATION KIT IS UPDATED
       A24-10) INSERT NEW STUDENTS NAME-CARDS IN THE DECK
       A24-11) THREW AWAY DELETED STUDENTS NAME-CARDS
       A24-12) INSERT NEW CLASS-LIST IN THE BIG POUCH

   2.2=== DATA GENERATION
       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
       A24-20) USE CLASS LIST IN BIG POUCH
       A24-21) TAKE NAME-CARDS FOR THE CLASS FROM POUCH
       A24-22) MAKE ROLL CALL
       A24-24) PUT NAME-CARDS OF PRESENT STUDENTS IN RIGHT POUCH
       A24-27) PUT NAME-CARDS OF ABSENT STUDENTS IN LEFT POUCH
       A24-28) READY THE DATA GENERATION KIT FOR COLLECTION PICK-UP
SYSTALL A(26) = A26 = GENERATE TRANSACTION DATA

= AGENERDATA WITH 13 ACTIONS
= SYSTALL A26-(1..13)

THIS PROCEDURE PERFORMS THE FOLLOWING:
1. TRANSACTION DATA IS GENERATED BY THE TEACHERS AT EVERY LECTURE
2. WITH THE HELP OF THE DATA GENERATION KIT.
3. EACH STUDENT ABSENT MAY HIS PANS-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 20-40 SECONDS OF THE TEACHER'S TIME.
4. DATA GENERATION KIT IS READY FOR COLLECTION

---

1234 (ALTERNATIVES)

- CLUSTERS -
  C1 2 : 0  === START
  C2 2 0 0  === DATA GENERATION

- PRECONDITIONS -
  P1 = Y  --- DATA GENERATION KIT IS UPDATED
  P2 = Y  --- THE PROCESS IS SIMPLE AND ACCEPTED BY TEACHERS
  P3 = Y  --- FALILTY DATA GENERATION POSSIBILITY IS MINIMAL AND ACCEPTABLE
  P4 = Y  --- IT TAKES LESS THAN A MINUTE

- ACTIONS -
  A1 1  A24-1D  PICK UP DATA GENERATION KIT FROM MAILBOX
  A2 4  A24-2D  PICK UP NEW CLASS LISTS
  A3 4  A24-3D  PICK UP NEW PANS-CARDS
  A4 4  A24-4D  GO TO CLASS
  A5 1  A24-5D  USE CLASS LIST IN BIG POUCH
  A6 4  A24-6D  TAKE PANS-CARDS FOR THE CLASS FROM POUCH
  A7 4  A24-7D  USE PANS-CARDS FOR THE ITEM
  A8 1  A24-8D  PUT PANS-CARDS OF PRESENT STUDENTS IN RIGHT POUCH
  A9 5  A24-9D  PUT PANS-CARDS OF ABSENT STUDENTS IN LEFT POUCH
  A10 1  A24-10D  REPLACE NEW STUDENTS PANS-CARDS IN THE DECK
  A11 3  A24-11D  DISCARD ABSENT STUDENTS PANS-CARDS
  A12 6  A24-12D  DISCARD NEW CLASSLIST IN THE BIG POUCH
  A13 6  A24-13D  READY THE DATA GENERATION KIT FOR COLLECTION PICK-UP
SYSTALL A27(12) = TRANSACTION DATA COLLECTION

= @DATACOLLECT WITH 14 ACTIONS
= SYSTALL A27-(1, 12)

DATA COLLECTION IS A VERY CRITICAL OPERATION OF THE DTS SYSTEM. IF IT DOES NOT RESPECT THE PREVIOUSLY DETERMINED CRITERIA, SYSTEM REJECTION WILL OCCUR.

THE LOGICAL DESCRIPTION OF @DATACOLLECT IS 3 LEVELS DEEP

AT SYSTALL A27-(12) = PERFORM FRAGMENTED DATA INPUT

= CALL FRAGMENT INPUT WITH 13 ACTIONS
= SYSTALL A27-(12)-(1, 12)

AT SYSTALL A27-(13-5) = PERFORM DATA VERIFICATION

= VERIFYDATA 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

1.1 DESIGNED 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

2.1 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
**SYSTALL A(27) = A(27) = TRANSACTION DATA COLLECTION**

= $^\text{DATACOLLECT WITH 14 ACTIONS}$
= SYSTALL A(27)-(1,14)

DATA COLLECTION IS A VERY CRITICAL OPERATION OF THE BATS SYSTEM. IF IT DOES NOT RESPECT THE PREVIOUSLY DETERMINED CRITERIA, SYSTEM REJECTION WILL OCCUR.

This routine does the following tasks:

1. DATA SPECIFICATIONS ARE COLLECTED BY THE CORRESPONDING SUPERVISOR FROM EACH TEACHER TEN MINUTES AFTER THE LECTURE STARTED.
2. ABSENT STUDENT'S CARDS ARE EXTRACTED FROM THE POUCHES.
3. EACH SUPERVISOR BRINGS 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 LECTURE'S TRANSACTION DATA.
7. OUTPUT IS PRINTED WITH A PRELIMINARY SUMMARY REPORT OF ABSENT STUDENT LISTING TO EACH ASSISTANT PRINCIPAL (REPORT A).

The logical description of DATACOLLECT is 3 levels deep:

AT SYSTALL A(27-13) = PERFORM FRAGMENTED DATA INPUT
= CALL OFRASINPUT WITH 13 ACTIONS
= SYSTALL A(27-13)-(1,13)

AT SYSTALL A(27-13-5) = PERFORM DATA VERIFICATION
= OVERTFRDATA WITH 12 ACTIONS
= SYSTALL A(27-13-5)-(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 5000 transactions per hour with an extremely low error rate.**

Continued on the next page.
1. --- ENTER TRANSACTION DATA COLLECTION PROCESS

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

1.2 --- ORGANIZATION PROBLEM
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
1.2 THEN-GO-TO-CLUSTER-1

1.3 --- INITIATE DATA COLLECTION
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
1.3 THEN-GO-TO-CLUSTER-2

2. --- DATA COLLECTION

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

2.2 --- TRANSACTION DATA IS READY
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
NONE LECTURE AFTER THIS
2.2 THEN-GO-TO-CLUSTER-2

2.3 --- LAST LECTURE TRANSACTION DATA IS READY
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 LECTURE AFTER THIS
2.3 THEN-GO-TO-CLUSTER-0
16. AIR-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 DATA SYSTEM. IF IT DOES NOT RESPECT THE PREVIOUSLY DETERMINED CRITERIA, 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 LECTURE'S 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 INFRAGINPUT WITH 13 ACTIONS
= SYSTALL A27-13>-(1..13)

AT SYSTALL A27-13< = PERFORM DATA VERIFICATION
= VERIFYDATA WITH 12 ACTIONS
= SYSTALL A27-13<-(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

1.1==DESIGN PROBLEM
1.1.1 IF DATA GENERATION PROCESS IS ACCEPTABLE TO THE TEACHERS AND IF NOT ACCEPTABLE ERROR RATE
A27> REVIEW SYSTEM DESIGN OR FACE SYSTEM REJECTION

1.2==ORGANIZATION PROBLEM
1.2.1 IF DATA GENERATION PROCESS IS ACCEPTABLE TO THE TEACHERS AND IF DATA COLLECTION ROUTE IS ORGANIZED AND IF NOT ACCEPTABLE DELAY IS 15 MINUTES FOR DATA GENERATION
A27> ORGANIZE DATA COLLECTION ROUTE TO THE CENTER

CONTINUED ON THE NEXT PAGE
1.3 **INITIATE DATA COLLECTION**

**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**

2.1 **TRANSACTION DATA IS NOT READY**

**IF** NOT TEACHERS FINISHED DATA GENERATION FOR THE CURRENT LECTURE

**A27-4** WAIT 10-15 MINUTES

2.2 **TRANSACTION DATA IS READY**

**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 WIP COLLECT @DAT-GEN-KITS FROM CLASSES

**A27-6** EXTRACT TRIANT STUDENTS' BARS-CARDS FROM THE KITS

**A27-7** KEEP @DAT-GEN-KITS IN WIP'S OFFICE

**A27-8** SEND BARS-CARDS TO THE COMPUTER CENTER OF THE SCHOOL

**A27-9** WAIT 10-15 MINUTES

**A27-10** COLLECT ALL BARS-CARDS FOR THE SCHOOL

**A27-11** READ ALL BARS-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**

**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 WIP COLLECT @DAT-GEN-KITS FROM CLASSES

**A27-6** EXTRACT TRIANT STUDENTS' BARS-CARDS FROM THE KITS

**A27-7** KEEP @DAT-GEN-KITS IN WIP'S OFFICE

**A27-8** SEND BARS-CARDS TO THE COMPUTER CENTER OF THE SCHOOL

**A27-9** WAIT 10-15 MINUTES

**A27-10** COLLECT ALL BARS-CARDS FOR THE SCHOOL

**A27-11** READ ALL BARS-CARDS FOR THE CURRENT LECTURE

**A27-12** MAKE A TEMPORARY TRANSACTION DATA FILE ON DISK

**A27-13** PERFORM FRAGMENTED DATA INPUT - CALL #FRAGINPUT
SYSTALL A(27) = A27> = TRANSACTION DATA COLLECTION
            = #DATACOLLECT WITH 14 ACTIONS
            = SYSTALL A27<(1..14).

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

THIS ROUTINE DOES THE FOLLOWING TASKS:
1 DATA GENERATION KITS ARE COLLECTED BY THE CORRIDOR SUPERVISOR
   FROM EACH TEACHER TERMINAL 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 FRAGMENT INPUT WITH 13 ACTIONS
   = SYSTALL A27-13<(1..13)

AT SYSTALL A27-13-5> = PERFORM DATA VERIFICATION
   = VERIFY DATA 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 2 6 0  = ENTER TRANSACTION DATA COLLECTION PROCESS
C2 0 2 2 0 0  = ENTER TRANSACTION DATA COLLECTION

PREDICATES
P1  N Y Y  = DATA COLLECTION ROUTE IS ORGANIZED
P2  N Y Y  = DATA GENERATION PROCESS IS ACCEPTABLE TO THE TEACHERS
P3  N Y Y  = ACCEPTABLE ERROR RATE
P4  N Y Y  = ACCEPTABLE DELAY IS 15 MINUTES FOR DATA GENERATION
P5  N Y Y  = ACCEPTABLE DELAY IS 15 MINUTES FOR DATA COLLECTION
P6  N Y Y  = TEACHERS FINISHED DATA GENERATION FOR THE CURRENT LECTURE
P7  N Y N  = MORE LECTURES AFTER THIS

CONTINUED ON THE NEXT PAGE
| A1 | 1 | A2-1) NO ACTION - NEXT CLUSTER |
| A2 | 1 | A2-2) REVIEW SYSTEM DESIGN OR FACE SYSTEM REJECTION |
| A3 | 1 | A2-3) ORGANIZE DATA COLLECTION ROUTE TO THE CENTER |
| A4 | 1 | A2-4) WAIT 10-15 MINUTES |
| A5 | 1 | A2-5) LET WP COLLECT GRADE-GRA-CH-KITS FROM CLASSES |
| A6 | 2 | A2-6) EXTRACT TRUANT STUDENTS' GRADE-CARDS FROM THE KITS |
| A7 | 3 | A2-7) KEEP GRADE-GRA-CH-KITS IN WP'S OFFICE |
| A8 | 4 | A2-8) SEND GRADE-CARDS TO THE COMPUTER CENTER OF THE SCHOOL |
| A9 | 5 | A2-9) COLLECT ALL GRADE-CARDS FOR THE SCHOOL |
| A10 | 7 | A2-10) ACTIVATE TERMINAL, CARD READER AND LOG IN |
| A11 | 8 | A2-11) READ IN ALL GRADE-CARDS FOR THE CURRENT LECTURE |
| A12 | 9 | A2-12) MAKE A TEMPORARY TRANSACTION DATA FILE ON DISK |
| A13 | 10 | A2-13) PERFORMTRANSMER DATA INPUT - CALL GRADEINPUT |
| A14 | 11 | A2-14) 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 TRANSREC.
2 CHECKS THE VALIDITY OF EACH TRANSACTION RECORD TRANS-REC BY WAY
OF REFERENCE TO THE DATA-BASE #999.
3 REJECTS ALL NON VALID TRANSACTION DATA WITH AN APPROPRIATE ERROR
MESSAGE.
4 PRODUCES AN OPTIONAL LISTING OF TRANSAT STUDENTS FOR THE CURRENT
LECTURE (REPORT "A")

THE LOGICAL DESCRIPTION OF THIS ROUTINE IS 2 LEVELS DEEP
AT SYSTALL A27-13\> IT CALLS TRANSACTION DATA VERIFICATION SUBROUTINE
= VERIFYDATA WITH 12 ACTIONS
= SYSTALL A27-13-D\>-(1..12)

AT THE END RETURN TO #$DATAcollect

1.== START WITH OUTPUT OPTION SELECTION
1.1==CALL PROCEDURE WITH REPORTING
1.2==CALL PROCEDURE WITH NO REPORTING
2.== TRANSACTION DATA VERIFICATION
2.1==ERROR -NO INITIALIZATION DONE-
2.2==ERROR -NO DATA-
2.3==COLLECT VALID TRANSACTION
3.== DATA COLLECTION SUMMARY REPORTING OPT 1 OR 2
3.1==WRITE TRIANGUL SUMMARY
3.2==WRITE ONLY MESSAGES

==3 < 2, 2 >
==3 < 0, 0, 2 >
==3 < 0, 0 >
==3 < 0, 0 >
SYSTALL A(27-13) = A27-133 = PERFORM FRAGMENTED TRANSACTION
DATA INPUT ON THE TERMINAL
= AFRAGINPUT WITH 13 ACTIONS
= SYSTALL A27-133-11.133

THIS ROUTINE IS CALLED BY THE DATA COLLECTION PROCEDURE AT SYSTALL A27-133
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 VALIATED TRANSACTION DATA FILE CALLED AFRAG.
2 CHECKS THE VALIDITY OF EACH TRANSACTION RECORD (TRANS-REC) BY WAY
   OF REFERENCE TO THE DATA-BASE IDN.
3 REJECTS ALL INVALID TRANSACTION DATA WITH AN APPROPRIATE ERROR
   MESSAGE.
4 PRODUCES AN OPTIMAL LISTING OF TRANSIT STUDENTS FOR THE CURRENT
   LECTURE (REPORT "A").

THE LOGICAL DESCRIPTION OF THIS ROUTINE IS 2 LEVELS DEEP
AT SYSTALL A27-133-5) IT CALLS TRANSACTION DATA VERIFICATION SUBROUTINE
    = VERIFYDATA WITH 12 ACTIONS
    = SYSTALL A27-133-5(1.12)

AT THE END RETURN TO DATACOLLECT

1.== START WITH OUTPUT OPTION SELECTION

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

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

2.== TRANSACTION DATA VERIFICATION

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

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

   2.3== COLLECT VALID TRANSACTION
       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 1 OR 2

3.1 WRITE TRAFFIC SUMMARY
   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
   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

3 = 0, 0
11. ABL-Routine 'FRAGINPUT'

SYSTALL 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 binary input file
to a validated transaction data file called DATADEC.
2. Checks the validity of each transaction record $\text{#TRANS-REC}$ by way
   of reference to the database $\text{##B}$.
3. Rejects all non-valid transaction data with an appropriate error
   message.
4. Produces an optional listing for triumphant students for the current
   lecture (Report 'A').

The logical description of this routine is 2 levels deep.
At SYSTALL A27-13-\> it calls transaction data verification subroutine
    = VERIFYDATA with 12 actions
    = SYSTALL A27-13-\>-(1..12)

At the end return to DATAACCOLECT

1.== START WITH OUTPUT OPTION SELECTION
     \>==\> < 2+2 >

1.1== CALL PROCEDURE WITH REPORTING
     1.1 IF start and if
      option with listing and if
      not option with no listing
      A27-13\> ENTER COMMAND = CALL+ABKPST

1.2== CALL PROCEDURE WITH NO REPORTING
     1.2 IF start and if
      not option with listing and if
      option with no listing
      A27-13-\> ENTER COMMAND = CALL+ABKPMLT

CONTINUED ON THE NEXT PAGE
2. === TRANSACTION DATA VERIFICATION ===

2.1 =ERROR =NO INITIALIZATION DONE=

2.1 IF NOT EXECUTION DAY IS INITIALIZED
   A27-13-6> WRITE ERROR MESSAGE

2.2 =ERROR =NO DATA=

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
   A27-13-1> INTERRUPT EXECUTION
   A27-13-15> COPY, TRANSACTION DATA FROM CARDS TO A DUMMY DISKFILE

2.3 =COLLECT VALID TRANSACTION=

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 "VERIFYDATA"

3. === DATA COLLECTION SUMMARY REPORTING OPT 1 OR 2 ===

3.1 =WRITE TRAVANCY SUMMARY=

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
   A27-13-1> SORT STUDENT OUTPUT RECORDS ON #HR, KEY
   A27-13-6> SORT STUDENT OUTPUT RECORDS ON STUDENT NAME
   A27-13-9> WRITE TRAVANCY 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 (#N/AWR)

3.2 =WRITE ONLY MESSAGES=

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 (#N/AWR)
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 NEEDED 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 #DA6CR.
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 OPTIMAL LISTING OF TRUANT STUDENTS FOR THE CURRENT
   LECTURE (REPORT "A").

THE LOGICAL DESCRIPTION OF THIS ROUTINE IS 2 LEVELS DEEP
AT SYSTALL A27-13> IT CALLS TRANSACTION DATA VERIFICATION SUBROUTINE
= VERIFDATA WITH 12 ACTIONS
= SYSTALL A27-13>-(1..12)

AT THE END RETURN TO #DATACOLLECT

CONTINUED ON THE NEXT PAGE
CLUSTERS -

C1  2  2  :  :  :  :  0  === START WITH OUTPUT OPTION SELECTION
C2  :  :  :  :  0  2  :  :  === TRANSACTION DATA VERIFICATION
C3  :  :  :  :  0  0  0  === DATA COLLECTION SUMMARY REPORTING OPT 1 OR 2

PREDICATES -

P1  Y  X  :  :  :  :  :  === 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
P5  Y  N  :  :  :  :  Y  === OPTION WITH LISTING
P6  N  Y  :  :  :  Y  === OPTION WITH NO LISTING

ACTIONS -

A1  1  :  :  :  :  :  :  === ENTER COMMAND = CALL+AMPKLIST 0
A2  1  :  :  :  :  :  :  === ENTER COMMAND = CALL+AMPKLIST 0
A3  1  :  :  :  :  :  :  === OPEN STUDENT MASTER FILE READ MODE
A5  3  :  :  :  :  :  :  === PERFORM INPUT TRANSACTION DATA VERIFICATION +CALL+VERIFDATA
A7  1  :  :  :  :  :  :  === SORT STUDENT OUTPUT RECORDS ON #VP. KEY
A8  2  :  :  :  :  :  :  === SORT STUDENT OUTPUT RECORDS ON STUDENT NAME
A12 5  :  :  :  :  :  :  === APPEND VALID @TRANS-REC TO PREVIOUSLY VALIDATED @TRANS-REC (@DBRDR)
12. AML-Routine VERIFDATA:

SYSFAIL A(27-13-5) = A27-13-5 PERFORM TRANSACTION DATA VERIFICATION
= VERIFDATA WITH 11 ACTIONS
= SYSFAIL A(27-13-5) < 1, 11)

THIS ROUTINE IS CALLED BY AFRAGINPUT AT SYSFAIL A27-13-5.
IT VERIFIES THE TRANSACTION DATA RECORD STRINGS FOR ERRORS
BEFORE IT APPENDS THEM TO THE DAILY VALID TRANSACTION DATA FILE (#Banner).
IT ALSO PRODUCES AN ERROR LISTING FOR THE REJECTED #TRANS-RECS.

AT THE END RETURN TO AFRAGINPUT

1. === ENTER VERIFDATA ROUTINE
   1.1 === START VERIFICATION ROUTINE
   1.1P NO CONDITIONS

2. === CHECK FOR INPUT ERROR
   2.1 === SET TRANSACTION RECORD
      2.2 IF NOT THERE IS A TRANSACTION RECORD IN BUFFER
      2.2P CHECK DATA COMMUNICATION VALIDITY
         2.2P IF THERE IS A TRANSACTION RECORD IN BUFFER
         2.3P NO DATA COMMUNICATION ERROR DETECTED
            2.3P IF THERE IS A TRANSACTION RECORD IN BUFFER AND IF
            NOT DATA TRANSMISSION ERROR IN TRANSACTION RECORD
            2.4P DATA COMMUNICATION ERROR DETECTED
               2.4P IF THERE IS A TRANSACTION RECORD IN BUFFER AND IF
               NOT DATA TRANSMISSION ERROR IN TRANSACTION RECORD

   2.5 === NO MORE TRANSACTION DATA, RETURN
      2.5P IF END OF INPUT TRANSACTION DATA BUFFER FILE

3. === TRANSACTION DATA ANALYSIS
   3.1 === CHECK TRANSACTION RECORD VALIDITY
      3.1P IF THERE IS A TRANSACTION RECORD IN BUFFER AND IF
      NOT DATA TRANSMISSION ERROR IN TRANSACTION RECORD

   3.2 === EMPTY TRANSACTION RECORD DETECTED
      3.2P IF #TRANS-REC IS EMPTY LINE

4. === DATA COLLECTION ERROR AND SUMMARY REPORTING
   4.1 === ERROR - DUPLICATE TRANSACTION RECORD DELETED
      4.1P IF #TRANS-REC IS DUPLICATE ENTRY

   4.2 === ERROR - STUDENT WHICH DROPPED COURSE REPORTED ABSENT
      4.2P IF STUDENT IS DELETED FROM CORRESPONDING COURSE

   4.3 === ERROR - SCHEDULE CONFLICT WITH OTHER COURSE
      4.3P IF #TRANS-REC COURSE HAS THE WRONG SCHEDULE

   4.4 === ERROR - DATA COMMUNICATION ERROR DETECTED
      4.4P IF DATA TRANSMISSION ERROR IN TRANSACTION RECORD

   4.5 === ERROR - DUPLICATE TRANSACTION RECORD DELETED
      4.5P IF #TRANS-REC IS DUPLICATE ENTRY

   4.6 === ERROR - STUDENT WHICH DROPPED COURSE REPORTED ABSENT
      4.6P IF STUDENT IS DELETED FROM CORRESPONDING COURSE

   4.7 === ERROR - SCHEDULE CONFLICT WITH OTHER COURSE
      4.7P IF #TRANS-REC COURSE HAS THE WRONG SCHEDULE

   4.8 === ERROR - DATA COMMUNICATION ERROR DETECTED
      4.8P IF DATA TRANSMISSION ERROR IN TRANSACTION RECORD

   4.9 === ERROR - DUPLICATE TRANSACTION RECORD DELETED
      4.9P IF #TRANS-REC IS DUPLICATE ENTRY

   4.10 === ERROR - STUDENT WHICH DROPPED COURSE REPORTED ABSENT
      4.10P IF STUDENT IS DELETED FROM CORRESPONDING COURSE

   4.11 === ERROR - SCHEDULE CONFLICT WITH OTHER COURSE
      4.11P IF #TRANS-REC COURSE HAS THE WRONG SCHEDULE

   4.12 === ERROR - DATA COMMUNICATION ERROR DETECTED
      4.12P IF DATA TRANSMISSION ERROR IN TRANSACTION RECORD

   4.13 === ERROR - DUPLICATE TRANSACTION RECORD DELETED
      4.13P IF #TRANS-REC IS DUPLICATE ENTRY

   4.14 === ERROR - STUDENT WHICH DROPPED COURSE REPORTED ABSENT
      4.14P IF STUDENT IS DELETED FROM CORRESPONDING COURSE

   4.15 === ERROR - SCHEDULE CONFLICT WITH OTHER COURSE
      4.15P IF #TRANS-REC COURSE HAS THE WRONG SCHEDULE

   4.16 === ERROR - DATA COMMUNICATION ERROR DETECTED
      4.16P IF DATA TRANSMISSION ERROR IN TRANSACTION RECORD

   4.17 === ERROR - DUPLICATE TRANSACTION RECORD DELETED
      4.17P IF #TRANS-REC IS DUPLICATE ENTRY

   4.18 === ERROR - STUDENT WHICH DROPPED COURSE REPORTED ABSENT
      4.18P IF STUDENT IS DELETED FROM CORRESPONDING COURSE

   4.19 === ERROR - SCHEDULE CONFLICT WITH OTHER COURSE
      4.19P IF #TRANS-REC COURSE HAS THE WRONG SCHEDULE

   4.20 === ERROR - DATA COMMUNICATION ERROR DETECTED
      4.20P IF DATA TRANSMISSION ERROR IN TRANSACTION RECORD
12. ADL-Routine VERIFDATA:

SYSTAIL A27-13-5 = A27-13-5: PERFORM TRANSACTION DATA VERIFICATION
      = VERIFDATA WITH 11 ACTIONS
      = SYSTAIL A27-13-5 = (11:11)

This routine is called by #TRANSPUT at SYSTAIL A27-13-5:
It verifies the transaction data record strings for errors.
Before it appears them to the daily valid transaction data file (#TRANSPUT).
It also produces an error listing for the rejected transaction records.
At the end return to #TRANSPUT.

1. == ENTER VERIFDATA ROUTINE
   1.1 == START VERIFICATION ROUTINE
       1.1P NO CONDITIONS
           A27-13-5-11: SORT ALL RECORDS IN TRANSACTION FILE.

2. == CHECK FOR INPUT ERROR
    2.1 == GET TRANSACTION RECORD
        2.1 IF NOT THERE IS A TRANSACTION RECORD IN BUFFER
            A27-13-5-10: READ NEXT #TRANSPRC INTO BUFFER FROM DUMMY INPUT FILE
    2.2 == CHECK DATA COMMUNICATION VALIDITY
        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
        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
        2.4 IF THERE IS A TRANSACTION RECORD IN BUFFER AND IF
            DATA TRANSMISSION ERROR IN TRANSACTION RECORD
            A27-13-5-5: WRITE APPROPRIATE ERROR MESSAGE TO ERROR FILE
            A27-13-5-7: COPY #TRANSPRC TO ERROR FILE
            A27-13-5-10: READ NEXT #TRANSPRC INTO BUFFER FROM DUMMY INPUT FILE
    2.5 == NO MORE TRANSACTION DATA; RETURN
        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

3.1 CHECK TRANSACTION RECORD VALIDITY

3.1 IF THERE IS A TRANSACTION RECORD IN BUFFER AND IF
NOT DATA TRANSMISSION ERROR IN TRANSACTION RECORD
A27-13-5-2) RECONSTRUCT TRANSACTION ITEMS
A27-13-5-3) RETRIEVE 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

3.2 IF #TRANS-REC IS EMPTY LINE
A27-13-5-9) WRITE APPROPRIATE ERROR MESSAGE TO ERROR FILE
A27-13-5-10) 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

4.1 ERROR - DUPLICATE TRANSACTION RECORD DETECTED-

4.1 IF #TRANS-REC IS DUPLICATE ENTRY
A27-13-5-9) WRITE APPROPRIATE ERROR MESSAGE TO ERROR FILE
A27-13-5-10) COPY #TRANS-REC TO ERROR FILE
A27-13-5-10> READ NEXT #TRANS-REC INTO BUFFER FROM DUMMY INPUT FILE

4.2 ERROR - STUDENT WHO DROPPED COURSE REPORTED ABSENT-

4.2 IF STUDENT IS DELETED FROM CORRESPONDING COURSE
A27-13-5-9) WRITE APPROPRIATE ERROR MESSAGE TO ERROR FILE
A27-13-5-10) 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-

4.3 IF #TRANS-REC COURSE HAS THE WRONG SCHEDULE
A27-13-5-9) WRITE APPROPRIATE ERROR MESSAGE TO ERROR FILE
A27-13-5-10) COPY #TRANS-REC TO ERROR FILE
A27-13-5-10> READ NEXT #TRANS-REC INTO BUFFER FROM DUMMY INPUT FILE
SYSTAT A(27-13-5) = A(27-13-5) PERFORM TRANSACTION DATA VERIFICATION
= VERIFYDATA WITH 11 ACTIONS
= SYSTAT A(27-13-5)"(1..11)

THIS ROUTINE IS CALLED BY INFRANINPUT AT SYSTAT A(27-13-5).
IT PERFORMS THE TRANSACTION DATA RECORD STRING ANA-MRNGS FOR ERRORS
BEFORE IT APPEARS THEN TO THE DAILY VALID TRANSACTION DATA FILE (08DRCR).
IT ALSO PRODUCES AN ERROR LISTING FOR THE REJECTED TRANS-RECS.

A AT THE END RETURN TO INFRANINPUT

1 2 3 4 5 6 7 8 9 10 11 12 (ALTERNATIVES)

CLUSTERS

--- ENTER VERIFYDATA ROUTINE
--- CHECK FOR INPUT ERROR
--- TRANSACTION DATA ANALYSIS
--- DATA COLLECTION ERROR AND SUMMARY-REPORTING

PREDICATES

--- THERE IS A TRANSACTION RECORD IN BUFFER
--- DATA TRANSMISSION ERROR IN TRANSACTION RECORD
--- TRANS-REC IS EMPTY LINE
--- TRANS-REC IS DUPLICATE ENTRY
--- STUDENT IS DELETED FROM CORRESPONDING COURSE
--- TRANS-REC COURSE HAS THE WRONG SCHEDULE
--- TRANS-REC IS VALID
--- END OF INPUT-TRANSACTION DATA BUFFER

ACTIONS

--- ANALYSE CHECK-SUM OF THE INPUT-TRANSACTION STRING
--- DECOMPOSE TRANS-REC INTO TRANSACTION ITEMS
--- RETRIEVE CORRESPONDING STUDENT RECORD
--- RETRIEVE CORRESPONDING COURSE RECORD
--- ANALYSE TRANSACTION ITEMS FOR VALIDITY
--- COPY TRANS-REC TO VALID TRANSACTION FILE (08DRCR)
--- COPY STUDENT DISPLAY ITEMS ON STUDENT OUTPUT RECORD
--- WRITE APPROPRIATE ERROR MESSAGE TO ERROR FILE
--- COPY TRANS-REC TO ERROR FILE
--- READ NEXT TRANS-REC INTO BUFFER FROM DUMMY INPUT FILE
--- SORT ALL RECORDS IN TRANSACTION FILE
--- NO ACTION; NEXT CLUSTER
SYSTALL A(28) = A28> = PERFORM DAILY DATS DATA PROCESSING.
= BABSPROC 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 transaction data in the data-base for further use. After this data selection phase this procedure writes a concise transaction 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 BABS1SP with 17 ACTIONS
= SYSTALL A28-11>-(1..17)

1. === START BABSPROC
1.1=== CALL PROCEDURE
2. === PREPARE ALLFILES
2.1=== PREPARE ALL FILES
3. === FILE UPDATING
3.1=== UPDATE FILES; WITH OUTPUT PARAMETERS MATCH
3.2=== UPDATE FILES; WITH OUTPUT PARAMETERS DON'T MATCH
3.3=== ERROR -DELETED DATABUBBLE-
3.4=== NO MORE UPDATING.
4. === REPORTING
4.1=== REPORTING; FINALIZATION, STOP

===8 < 2 >
---THEN-GO-TO-CLUSTER-> 2

===8 < 3 >
---THEN-GO-TO-CLUSTER-> 3

===8 < 3; 3; 3; 4 >
---THEN-GO-TO-CLUSTER-> 3

===8 < 3 >
---THEN-GO-TO-CLUSTER-> 3

==0 < 0 >
---THEN-GO-TO-CLUSTER-> 0
SYSTALL A260 = A20D = PERFORM DAILY BATCH DATA PROCESSING.
= BATCHPROC WITH 25 ACTIONS
= SYSTALL A260-(1..25)

This process analyses and stores the collected transaction data at the end of the day. It directly stores all transaction data in the data-base for further use. After this data reduction phase this procedure writes a concise transaction report (REPORT.T-3) in accordance to the format and degree of detail previously specified by a set of output selection parameters. All output is sorted and saved on disk until next morning.

The logical description of this procedure is 2 levels deep.

At SYSTALL A260-1) STORE TRANSACTION DATA IN THE DATA-BASE
= CALL @STUPB WITH 17 ACTIONS
= SYSTALL A260-11-(1..17)

1.== START @BATCHPROC ===< 2 >
1.1== CALL PROCEDURE
1.1.1 IF ALL TRANSACTION DATA IS COLLECTED AND VERIFIED IN @DBCR
2.== PREPARE ALL FILES
====< 3 >
2.1== PREPARE ALL FILES
2.1.1 IF NOT ALL TRANSACTION DATA IS SORTED IN @DBCR
3.== FILE UPDATING
====< 3, 3, 3, 4 >
3.1== UPDATE FILES; WITH OUTPUT PARAMETERS MATCH
3.1.1 IF ALL TRANSACTION DATA IS SORTED IN @DBCR AND IF NOT STUDENT CORRESPONDING TO TRNSC-REC IS DELETED FROM SCHOOL AND IF STUDENTS ABSENCE MATCH OUTPUT SELECTION PARAMETERS SET UP AND IF NOT END OF VALID TRANSACTION FILE (@DBCR)
3.2== UPDATE FILES; WITH OUTPUT PARAMETERS DON'T MATCH
3.2.1 IF ALL TRANSACTION DATA IS SORTED IN @DBCR AND IF NOT STUDENT CORRESPONDING TO TRNSC-REC IS DELETED FROM SCHOOL AND IF NOT STUDENTS ABSENCE MATCH OUTPUT SELECTION PARAMETERS SET UP AND IF NOT END OF VALID TRANSACTION FILE (@DBCR)
3.3== ERROR -DELETED STUDENT-
3.3.1 IF ALL TRANSACTION DATA IS SORTED IN @DBCR AND IF STUDENT CORRESPONDING TO TRNSC-REC IS DELETED FROM SCHOOL AND IF NOT END OF VALID TRANSACTION FILE (@DBCR)
3.4== NO MORE UPDATING
3.4.1 IF END OF VALID TRANSACTION FILE (@DBCR)
4.== REPORTING
====< 0 >
4.1== REPORTING; FINALIZATION; STOP
4.1.1 IF NO CONDITIONS
SYSTALL A(2B) = A2B = PERFORM DAILY Datas DATA PROCESSING.

= @ABSPROC WITH 25 ACTIONS
= SYSTALL A2B-(1.,25)

This process analyses and stores the collected transaction data at the end of the day. It directly stores all transaction data in the data-base for further use. After this data reduction phase, this procedure writes a concise trancy report (Report 'I') 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 A2B-11) STORE TRANSACTION DATA IN THE DATA-BASE
= CALL @ABSTUP With 17 ACTIONS
= SYSTALL A2B-11) -(1..117)

1. === START ABSPROC  
   1.1 == CALL PROCEDURE = --THEN-GO-TO-CLUSTER> 2
   1.1 IF ALL TRANSACTION DATA IS COLLECTED AND VERIFIED IN (ABACR)
   A2B-1> ENTER COMMAND = CALL @ABSPRO 0

2. === PREPARE ALLFILES  
   2.1 == PREPARE ALL FILES = --THEN-GO-TO-CLUSTER> 3
   2.1 IF NOT ALL TRANSACTION DATA IS SORTED IN (ABACR)
   A2B-2> SORT TRANS-RECS, BY STUDENT ID NO. (MAJOR KEY)
   A2B-3> SORT TRANS-RECS, BY CLASS ID. NO. (MINOR KEY)
   A2B-4> ELIMINATE DUPLICATE ENTRIES OF TRANS-RECS.
   A2B-5> REBIND TRANSACTION FILE (ABACR)
   A2B-6> OPEN STUDENT MASTER FILE
   A2B-7> OPEN COURSE MASTER FILE; CLASS MASTER FILE
   A2B-8> UPDATE PRODUCTION POINTERS IN (ABACR-RECORD)
   A2B-9> READ ALL TRANS-RECS FOR ONE STUDENT INTO BUFFER (1-6 RECORDS)

3. === FILE UPDATING  
   3.1 == UPDATE FILES; WITH OUTPUT PARAMETERS MATCH = --THEN-GO-TO-CLUSTER> 3
   3.1 IF ALL TRANSACTION DATA IS SORTED IN (ABACR) AND IF NOT STUDENT CORRESPONDING TO TRANS-REC IS DELETED FROM SCHOOL AND IF STUDENT'S ABSENCES MATCH OUTPUT SELECTION PARAMETERS SET UP AND IF NOT END OF VALID TRANSACTION FILE (ABACR)
   A2B-10> RETRIEVE STUDENT RECORD
   A2B-11> PERFORM DATA-BASE TRANCY STORAGE UPDATE -CALL @ABSTUP9-
   A2B-12> WRITE STUDENT TRANCY RELATED DATA TO STUDENT OUTPUT FILE
   A2B-13> INCORPORATE STATISTICS
   A2B-10> READ ALL TRANS-RECS INTO BUFFER FOR NEXT STUDENT (1-6 RECORDS)

CONTINUED ON THE NEXT PAGE
3.2=UPDATE FILE; WITH OUTPUT PARAMETERS DON'T MATCH  =  😏 THEN-GO-TO-CLUSTER> 3
3.2 IF ALL TRANSACTION DATA IS SORTED IN #MANCR AND IF
    NOT STUDENT CORRESPONDING TO #TRANS-REC IS DELETED FROM SCHOOL AND IF
    NOT STUDENTS ABSENCE MATCH OUTPUT SELECTION PARAMETERS SET UP AND IF
    NOT END OF VALID TRANSACTION FILE (#MANCR)
    A28-10> RETRIEVE STUDENT RECORD
    A28-11> PERFORM DATABASE TRANSACTION STORAGE UPDATE -CALL 293STUPP-
    A28-12> INCREMENT STATISTICS
    A28-13> READ ALL #TRANS-RECS INTO BUFFER FOR NEXT STUDENT (1-6 RECORDS)

3.3=ERROR -DELETED STUDENT- =  😏 THEN-GO-TO-CLUSTER> 3
3.3 IF ALL TRANSACTION DATA IS SORTED IN #MANCR AND IF
    STUDENT CORRESPONDING TO #TRANS-REC IS DELETED FROM SCHOOL AND IF
    NOT END OF VALID TRANSACTION FILE (#MANCR)
    A28-10> RETRIEVE STUDENT RECORD
    A28-11> WRITE ERROR MESSAGE
    A28-12> DISPLAY STUDENT IN ERROR FILE
    A28-13> 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 (#MANCR)
    A28-20> NO ACTION; NEXT CLUSTER

4.1= REPORTING; FINALIZATION; STOP =  😏 THEN-GO-TO-CLUSTER> 0
4.1 IF NO CONDITIONS
    A28-17> SORT ERROR FILE ON #P KEY.
    A28-18> SORT STUDENT OUTPUT FILE ON #P KEY.
    A28-19> WRITE STUDENT REPORT LA 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. AML-TRANSFORM ABSPROC

SYSTALL A(28) = A2(28) = PERFORM DAILY OATS DATA PROCESSING.

= GASPROC WITH 25 ACTIONS
= SYSTALL A2(28)-(1..25)

THIS PROCESS ANALYSES AND STORES THE COLLECTED TRANSACTION DATA AT THE END OF THE DAY. IT DIRECTLY STORES ALL TRANSACTIONS DATA IN THE DATA-BASE FOR FURTHER USE. AFTER THIS DATA REDUCTION PHASE THIS PROCEDURE WRITES A CONCISE TRANSACTION REPORT (REPORT "TR") IN ACCORDANCE TO THE FORMAT AND LEVEL OF DETAIL PREVIOUSLY SPECIFIED BY A SET OF OUTPUT SELECTION PARAMETERS. ALL OUTPUT IS SPOOLED AND SAVED ON DISK UNTIL NEXT ADJUSTING.

THE LOGICAL DESCRIPTION OF THIS PROCEDURE IS 2 LEVELS DEEP

AT SYSTALL A(28-11) STORE TRANSACTION DATA IN THE DATA-BASE
= CALL 64TRANSP WITH 17 ACTIONS
= SYSTALL A2(28-11)-(1..17)

Continued on the next page.
CLUSTERS
C1 2 0
C2 3 0
C3 3 3 4 0
C4 0

== START ABSPROC ==

== PREPARE ALLFILES ==

== FILE UPDATING ==

== REPORTING ==

PREDICATES
P1 Y
P2 N Y Y
P3 N
P4 Y
P5 N N Y

ALL TRANSACTION DATA IS COLLECTED AND VERIFIED IN (ABMCR)
ALL TRANSACTION DATA IS SORTED IN (ABMCR)
STUDENT CORRESPONDING TO (TRANS-REC) IS DELETED FROM SCHOOL
STUDENTS ABSENCE MATCH OUTPUT SELECTION PARAMETERS SET UP
END OF VALID TRANSACTION FILE (ABMCR)
START

ACTIONS
A1 1
A2 1
A3 2
A4 3
A5 4
A6 5
A7 6
A8 7
A9 8
A10 1 1 1
A11 2 2
A12 3 3
A13 4 3 2
A14 5 3
A15 5 4 4
A16 5 4
A17 5 4 4
A18 6
A19 3
A20 4
A21 5
A22 7
A23 7
A24 8
A25 1

A20-1) ENTER COMMAND = CALL, ABMCR 0
A20-2) SORT (TRANS-RECS), BY STUDENT ID. NO. (MAJOR KEY)
A20-3) SORT (TRANS-RECS), BY CLASS ID. NO. (MINOR KEY)
A20-4) ELIMINATE DUPLICATE ENTRIES OF (TRANS-RECS)
A20-5) RESEND TRANSACTION FILE (ABMCR)
A20-6) OPEN STUDENT MASTER FILE
A20-7) OPEN COURSE MASTER FILE; CLASS MASTER FILE
A20-8) UPDATE PRODUCTION POINTERS IN (TRANS-RECS)
A20-9) READ ALL (TRANS-RECS), FOR ONE STUDENT INTO BUFFER (1-6 RECORDS)
A20-10) RETRIEV STUDENT RECORD
A20-11) PERFORM DATA-BASE TRAMNACY STORAGE UPDATE -CALL A42STUUPD-
A20-12) WRITE STUDENT TRAUMACY RELATED DATA TO STUDENT OUTPUT FILE
A20-13) INCREMENT STATISTICS
A20-14) WRITE ERROR MESSAGE
A20-15) DISPLAY STUDENT ID. ERROR FILE
A20-16) READ ALL (TRANS-RECS) INTO BUFFER FOR NEXT STUDENT (1-6 RECORDS)
A20-17) SORT ERROR FILE ON (APP KEY)
A20-18) SORT STUDENT OUTPUT FILE ON (APP KEY)
A20-19) WRITE STUDENT REPORT "B" FROM STUDENT OUTPUT FILE
A20-20) WRITE ERROR REPORT FROM ERROR FILE
A20-21) COMPUTE STATISTICS FOR THE DAY
A20-22) WRITE STUDENT SUMMARY REPORT
A20-23) STORE STATISTICS
A20-24) CLOSE ALL FILES
A20-25) NO ACTION; NEXT CLUSTER
SYSTALL A(20-11) = A20-11 = STORE TRANSACTIONS IN THE DATA-BASE
   = #ABSTUPD WITH 17 ACTIONS
   = SYSTALL A(20-11)-1(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
SUBTOTAL-OF-LECTURES-MISSED-IN-COURSES, FOR THE YEAR IN THE #STUD-REC AND #CLASS-REC
SUBTOTAL-OF-LECTURES-MISSED-IN-COURSES, FOR THE SEMESTER IN THE #STUD-REC AND #CLASS-REC
DAILY CALENDAR ABSENCE PATTERN IN THE #STUD-REC
STATISTICS FOR THE DAY

1.== ENTER #ABSTUPD
   
1.1==ENTER ROUTINE

2.== UPDATE GENERAL TOTALS OF THE #STUD-REC

2.1== INCREMENT #STUD-REC TOTALS

3.== UPDATE COURSE DATA OF THE #STUD-REC AND #CLASS-REC.

   
3.1== UPDATE COURSE RELATED DATA-BASE

3.2== ERROR -REMOVED STUDENT FROM THIS COURSE

3.3== STORE DAILY MASK IN DAILY CALENDAR; RETURN
14. AML-Routine A8STUPD:

SYSTALL A20-11 = A20-11 = STORE TRANSACTIONS IN THE DATA-BASE
= A8STUPD WITH 17 ACTIONS
= SYSTALL A20-11-(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 OSTUD-REC
TOTAL-DAYS-ABSENT FOR THE SEMESTER IN THE OSTUD-REC
TOTAL-LECTURES-KISSED FOR THE YEAR IN THE OCLASS REC
TOTAL-LECTURES-KISSED FOR THE SEMESTER IN THE OCLASS-REC
LAST-DATE-ABSENT-FROM-SCHOOL IN THE OSTUD-REC
SUBTOTAL-OF-LECTURES-KISSED-IN-COURSEID, FOR THE YEAR
IN THE OSTUD-REC AND THE OCLASS-REC
SUBTOTAL-OF-LECTURES-KISSED-IN-COURSEID, FOR THE SEMESTER
IN THE OSTUD-REC AND THE OCLASS-REC
BINARY CALENDAR ABSENCE PATTERN IN THE OSTUD-REC

STATISTICS FOR THE DAY

1.== ENTER A8STUPD

1.1== ENTER ROUTINE
1.1P NO CONDITIONS

2.== UPDATE GENERAL TOTALS OF THE OSTUD-REC

2.1== INCREMENT OSTUD-REC TOTALS
2.11 IF BUFFER CONTAINS N TRANSACTIONS FOR A STUDENT (N= 1 TO 6)

3.== UPDATE COURSE DATA OF THE OSTUD-REC AND OCLASS-REC:

3.1== UPDATE COURSE RELATED DATA-BASE
3.11 IF BUFFER CONTAINS N TRANSACTIONS FOR A STUDENT (N= 1 TO 6) AND IF
NON STUDENT IS DELETED FROM COURSE CORRESPONDING TO OTRANS-REC AND IF
NOT ALL OTRANS-RECS ARE ENTERED FOR THIS STUDENT ON HIS OSTUD-REC

3.2== ERROR -DELETED STUDENT FROM THIS COURSE
3.21 IF BUFFER CONTAINS N TRANSACTIONS FOR A STUDENT (N= 1 TO 6) AND IF
STUDENT IS DELETED FROM COURSE CORRESPONDING TO OTRANS-REC AND IF
NOT ALL OTRANS-RECS ARE ENTERED FOR THIS STUDENT ON HIS OSTUD-REC

3.3== STORE BINARY MASK IN BINARY CALENDAR, RETURN
3.31 IF ALL OTRANS-RECS ARE ENTERED FOR THIS STUDENT ON HIS OSTUD-REC
SYSTALL A(20-11) = A20-11;  = STORE TRANSACTIONS IN THE DATA-BASE
                      = ABSTUPD WITH 17 ACTIONS
                      = SYSTALL A20-117-(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-DAY-ABSENT FROM SCHOOL IN THE #STUD-REC
LAST-DAY-ABSENT FROM LECTURE COURSE(S) IN THE #STUD-REC
TOTAL-OF-LECTURES-MISSED-IN-COURSE(S) FOR THE YEAR
SUBTOTAL-OF-LECTURES-MISSED-IN-COURSE(S) FOR THE SEMESTER

#IMRTE: CUMULATIVE ABSENCE PATTERN IN THE #STUD-REC
STATISTICS FOR THE DAY.

1.== ENTER ABSTUPD

1.1 ENTER ROUTINE

1.1 FE AT NO CONDITIONS

A20-11-17; NO ACTION; NEXT CLUSTER

2.== UPDATE GENERAL TOTALS OF THE #STUD-REC

2.1 INCIDENCE #STUD-REC TOTALS

2.1 IF BUFFER CONTAINS N TRANSACTIONS FOR A STUDENT (N = 1 TO 4)

A20-11-12: ENTER DATE ON #STUD-REC FOR LAST-DAY-ABSENT
A20-11-13: INCREMENT ON #STUD-REC; ALL TOTAL DAYS ABSENT BY 1
A20-11-13: INCREMENT ON #STUD-REC; ALL TOTAL NUMBERS OF LECTURES ABSENT BY N
A20-11-12: TACE NEXT #TRANS-REC IN THE BUFFER

3.== UPDATE COURSE DATA OF THE #STUD-REC AND #CLASS-REC.

3.1 UPDATE COURSE RELATED DATA-BASE

3.1 IF BUFFER CONTAINS N TRANSACTIONS FOR A STUDENT (N = 1 TO 4) 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

A20-11-10: FOR THE COURSE IN #TRANS-REC ENTER DATE FOR BLAST-DAY-ABS-IN-CR IN #STUD-REC
A20-11-11: INCREMENT SUBTOTALS (DAYS AND PERIODS ABSENT) IN #STUD-REC
A20-11-10: RETRIEVE COURSE RECORD FOR SCHMEULE FROM THE COURSE MASTER FILE
A20-11-11: RETRIEVE CLASS RECORDS FROM THE CLASS MASTER FILE
A20-11-11: INCREMENT CLASS RECORD WITH STUDENTS CORRESPONDING TO SUBTOTALS-INS-CR
A20-11-10: WRITE ON CLASS RECOR STUDENTS LAST-DAY-ABS-IN-CR
A20-11-10: REMAPTE CLASS RECORD
A20-11-11: PREPARE BINARY MASK FOR THE DAYS SCHMEULE PATTERN ABSENT BY STUDENT
A20-11-12: TACE 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 STUDENT'S NAME AND COURSE ON ERROR FILE
A28-11-15> TAKE NEXT #TRANS-REC IN THE BUFFER

3.3: "STORE BINARY MASK IN BINARY CAFEMAIN; RETURN" => THEN-GO-TO-CLUSTER -> 0

3.3 IF ALL #TRANS-RECS ARE ENTERED FOR THIS STUDENT ON HIS #STUB-REC,
A28-11-16> STORE BINARY MASK (DAILY ABSENCE PATTERN) IN THE #STUB-REC.
A28-11-17> REWRITE UPDATED #STUB-REC IN STUDENT MASTER FILE
SYSTALL A(28-11) = A28-11) = STORE TRANSACTIONS IN THE DATA-BASE
= #A80TPUD 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-COURSE IN THE #STUD-REC AND #CLASS-REC
SUBTOTAL-OF-LECTURES-MISSED-IN-COURSE IN #STUD-REC, FOR THE YEAR
IN THE #STUD-REC AND THE #CLASS-REC
SUBTOTAL-OF-LECTURES-MISSED-IN-COURSE FOR THE SEMESTER
IN THE #STUD-REC AND THE #CLASS-REC
BINARY CALENDAR ABSENCE PATTERN IN THE #STUD-REC
STASTISTICS FOR THE DAY


<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

ALTERNATIVES:

CLUSTERS:
C1 2 3 0 0 == ENTER #A80TPUD
C2 3 3 0 0 == UPDATE GENERAI 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 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-L ENTER DATE ON #STUD-REC FOR LAST-DATE-ABSENT
A2 2 A28-11-L INCREMENT ON #STUD-REC, ALL TOTAL DAYS ABSENT BY 1
A3 3 A28-11-L INCREMENT ON #STUD-REC, ALL TOTAL NUMBERS OF LECTURES MISSED BY N
A4 4 A28-11-L FOR THE COURSE IN #TRANS-REC ENTER DATE FOR LAST-DAY-ABS-IN-CR IN #STUD-REC
A5 5 A28-11-S FOR THE COURSE IN #TRANS-REC, INCREMENT TOTALS (DAYS AND PERIODS ABSENT) IN #STUD-REC
A6 3 A28-11-L RETRIEVE COURSE RECORD FOR SCHEDULE FROM THE COURSE MASTER FILE
A7 4 A28-11-L RETRIEVE CLASS RECORDS FROM THE CLASS MASTER FILE
A8 5 A28-11-L INCREMENT CLASS RECORD WITH STUDENTS CORRESPONDING TOTALS-IN-CR
A9 6 A28-11-L WRITE ON CLASS RECORD STUDENTS LAST-DAY-ABS-IN-CR
A10 7 A28-11-L WRITE CLASS RECORD
A11 8 A28-11-L WRITE BINARY MASK FOR THE DAYS SCHEDULE PATTERN (ABSENT BY STUDENT)
A12 4 9 3 A28-11-L TAKE NEXT #TRANS-REC IN THE BUFFER
A13 1 A28-11-L WRITE ERROR MESSAGE
A14 2 A28-11-L WRITE STUDENTS NAME AND COURSE ON ERROR FILE
A15 1 A28-11-L STORE BINARY MASK, DAILY ABSENCE PATTERN IN THE #STUD-REC.
A16 2 A28-11-L RETRIEVE UPDATED #STUD-REC, IN STUDENT MASTER FILE
A17 1 A28-11-L NO ACTION, NEXT CLUSTER
SYSTALL A(29) = A29 = DISTRIBUTION AND DATA DISTRIBUTION
    = DISTRIBUTION WITH 26 ACTIONS.
    = SYSTALL A29 = (0, 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 CRITERIA ESTABLISHED FOR THE INFORMATION TO REACH ITS TARGET USER AND IN ORDER TO PREVENT A QUEEN 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 ACCOMMODATE A FLOW OF VARIOUS KIND OF INFORMATION REQUESTS ABOUT THE STORED TRAUCY DATA IN THE DATA-BASE.

1. REGULAR DAILY DATA DISTRIBUTION
   1.1 DAILY MORNING DISTRIBUTION = --THEN-GO-TO-CLUSTER-> 1
   1.2 ADEPT-CARD REDISTRIBUTION AFTER EACH USE = --THEN-GO-TO-CLUSTER-> 1
   1.3 FINAL INFORMATION DISTRIBUTION = --THEN-GO-TO-CLUSTER-> 1
   1.4 OCCASIONAL TRAUCY INFORMATION REQUESTED = --THEN-GO-TO-CLUSTER-> 2

2. OCCASIONAL IRREGULAR INFORMATION DISTRIBUTION
   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 A(29) = 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 CRITERIA ESTABLISHED
FOR THE INFORMATION TO BE DELIVERED ON TIME AND TO PREVENT
A QUEUE 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 ACCOMMODATE A FLOW
OF VARIOUS KIND OF INFORMATION REQUESTS ABOUT THE STORED TURNOVER DATA
IN THE DATA-BASE.

1. == REGULAR DAILY DATA DISTRIBUTION

1.1 = DAILY MORNING DISTRIBUTION = --THEN-GO-TO-CLUSTER--> 1

1.1 IF OBJECTIVE IS AN EFFICIENT INSTRUCTIONAL 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 4 OR 5 OR 7 IS COMPLETE AND IF
  NOT DATA COLLECTION FOR THE DAY IS COMPLETE AND IF
  NOT DATA PROCESSING IS ENDED

1.2 = ABS-CARD REDISTRIBUTION AFTER EACH USE = --THEN-GO-TO-CLUSTER--> 1

1.2 IF DATA COLLECTION FOR LECTURE 2 OR 3 OR 4 OR 5 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

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

1.4 = OCCASIONAL TURNOVER 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

2.1. INDIVIDUAL REMEDIAL CONSULTATION
2.1. IF INFORMATION IS REQUIRED ON AN IRREGULAR BASIS AND IF
OBJECTIVE IS AN EFFICIENT STUDENT CONSULTATION AND IF
IMMEDIATE INFORMATION ACCESS IS POSSIBLE AND IF
ACCEPTABLE DELAY IS 2 HOURS

2.2. PLANNED FAMILY CONSULTATION
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
2.3. IF INFORMATION IS REQUIRED ON AN IRREGULAR BASIS AND IF
OBJECTIVE IS AN EFFICIENT STUDENT AND PARENT CONSULTING AND IF
IMMEDIATE INFORMATION ACCESS IS POSSIBLE

2.4. TEACHER INFORMATION
2.4. IF INFORMATION IS REQUIRED ON AN IRREGULAR BASIS AND IF
OBJECTIVE IS TO PROVIDE EFFICIENT CLASS MANAGEMENT AND IF
IMMEDIATE INFORMATION ACCESS IS POSSIBLE AND IF
ACCEPTABLE DELAY IS ONE MONTH

2.5. SCHOOL MANAGEMENT
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
2.6. IF ALL REGULAR DATA IS DISTRIBUTED AND IF
NOT INFORMATION IS REQUIRED ON AN IRREGULAR BASIS

2.7. PAUSE
2.7. IF NOT ALL REGULAR DATA IS DISTRIBUTED AND IF
NOT INFORMATION IS REQUIRED ON AN IRREGULAR BASIS
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 CRITERIA ESTABLISHED
FOR THE INFORMATION TO REACH ITS TARGET USER AND IN ORDER TO PREVENT
A QUEUE 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 ACCOMMODATE A FLOW
OF VARIOUS KINDS OF INFORMATION REQUESTS ABOUT THE STORED TRUancy DATA
IN THE DATA-BASE.

1== REGULAR DAILY DATA DISTRIBUTION
1.1==DAILY WORKING 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 MVP
A29-2) SEPARATE INPUT MARS-CARD DECK FOR EACH MVP
A29-3) SEPARATE OUTPUT FOR EACH MVP
A29-4) SEND BACK EACH MVP HIS MARS-CARDS
A29-5) SEND BACK EACH MVP HIS TRUancy REPORT "A"
A29-6) SEND BACK EACH MVP HIS TRUancy REPORT "B"
A29-7) AT THE MVP'S OFFICE PREPARE NEXT PRODUCTION CYCLE
A29-10) CHECK ERROR LISTING FOR DAMAGED MARS-CARDS
A29-11) TAKE OUT FAULTY CARDS FROM EACH DECK
A29-12) REPLACE MARS-CARDS TO THEIR RESPECTIVE MARS-SED-KITS FOR REUSE
A29-13) DISTRIBUTE MARS-SED-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 PERSONAL
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 =  
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 
A29-12 = AT THE CENTER SEPARATE ALL DATA TO CORRESPONDING %
A29-2 = SEPARATE INPUT ABS-CARD DECK FOR EACH %
A29-3 = SEPARATE OUTPUT FOR EACH %
A29-4 = SEND BACK EACH % HIS ABS-CARDS 
A29-5 = SEND BACK EACH % HIS ERROR LISTING 
A29-6 = AT THE % 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 #MAT-GEN-KIT FOR REUSE 
A29-13 = DISTRIBUTE #MAT-GEN-KITS TO TEACHERS' MAILBOXES 

1.3 = FINAL INFORMATION DISTRIBUTION =  
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 
A29-12 = AT THE CENTER SEPARATE ALL DATA TO CORRESPONDING % 
A29-2 = SEPARATE OUTPUT FOR EACH % 
A29-7 = SEND BACK EACH % HIS TRUANCY REPORT "B" 
A29-8 = SEND PRINCIPAL TRUANCY STATISTICS REPORT 
A29-14 = PERSONAL TRUANCY RELATED HUMAN COMMUNICATION 
A29-10 = MEET TRUANT STUDENT 
A29-20 = RELATE INFORMATION ON HAND 

1.4 = OCCASIONAL TRUANCY INFORMATION REQUESTED =  
1.4 IF INFORMATION IS REQUIRED ON AN IRREGULAR BASIS 
A29-25 = NO ACTION - NEXT CLUSTER 

2. = OCCASIONAL IRREGULAR INFORMATION DISTRIBUTION 

2.1 = INDIVIDUAL REMEDIAL CONSULTATION =  
2.1 IF INFORMATION IS REQUIRED ON AN IRREGULAR BASIS AND IF 
OBJECTIVE IS AN EFFICIENT STUDENT CONSULTING AND IF 
IMMEDIATE INFORMATION ACCESS IS POSSIBLE AND IF 
ACCEPTABLE DELAY IS 2 HOURS 
A29-22 = QUERY #AB, FOR RETROACTIVE STUDENT TRUANCY INFORMATION 
A29-10 = MEET TRUANT STUDENT 
A29-20 = RELATE INFORMATION ON HAND 

2.2 = PLANNED FAMILY CONSULTATION =  
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 = QUERY #AB, FOR RETROACTIVE STUDENT TRUANCY INFORMATION 
A29-10 = MEET TRUANT STUDENT 
A29-18 = 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 ACCESS IS POSSIBLE
      A29-22) QUERY HDB. FOR RETROACTIVE STUDENT TRUANCY INFORMATION
      A29-21) 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 ACCESS IS POSSIBLE AND IF
      ACCEPTABLE DELAY IS ONE MONTH
      A29-32) QUERY HDB. 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
      A29-22) QUERY HDB. 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
SYSTALL A\(29\) = A\(29\) = INFORMATION AND DATA DISTRIBUTION

= DISTRIBUTION WITH 25 ACTIONS
= SYSTALL A\(29\) = \{1...25\}

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 criteria established for the information to reach its target user and in order to prevent a queue formation of the data to be distributed at the end of the day, we must 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 accommodate a flow of various kind of information requests about the stored transacy data in the data-base.

CONTINUED ON THE NEXT PAGE
ALTERNATIVES

C1 1 1 1 2 . . . . . 0 — REGULAR DAILY DATA DISTRIBUTION
C2 . . . 1 1 1 1 1 0 1 0 — OCCASIONAL IRREGULAR INFORMATION DISTRIBUTION

CLUSTERS

P1 - - - - - - - - - Y N = ALL REGULAR DATA IS DISTRIBUTED
P2 - - - - - - - - - - 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 = ACCEPTABLE DELAY IS 2 DAYS
P9 - - - - - - - - - - Y = OBJECTIVE IS TO PROVIDE EFFICIENT CLASS MANAGEMENT
P10 - - - - - - - - - - Y = ACCEPTABLE DELAY IS 2 MONTHS
P11 - - - - - - - - - - Y = OBJECTIVE IS AN EFFICIENT SCHOOL MANAGEMENT
P12 - - - - - - - - - - Y = ACCEPTABLE DELAY IS 2-3 DAYS
P13 - - - - - - - - - - Y Y = IMMEDIATE INFORMATION ACCESS 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 WAY IS COMPLETE
P17 N N Y - - - - - - - = DATA PROCESSING IS EMERGED

ACTIONS

A1 1 1 1 A29-12 AT THE CENTER SEPARATE ALL DATA TO CORRESPONDING 6P
A2 9 3 2 A29-13 SEPARATE INPUT MARS-CARD DECK FOR EACH 6P
A3 3 3 2 A29-14 SEPARATE OUTPUT FOR EACH 6P
A4 4 4 A29-15 SEND BACK EACH 6P HIS MARS-CARDS
A5 5 5 A29-16 SEND BACK EACH 6P HIS ERROR LISTING
A6 6 6 A29-17 SEND BACK EACH 6P HIS TRUANCY REPORT "A"
A7 7 7 A29-18 SEND BACK EACH 6P HIS TRUANCY REPORT "B"
A8 8 8 A29-19 SEND PRINCIPAL TRUANCY STATISTICS REPORT
A9 9 9 A29-20 AT THE 6P'S OFFICE PREPARE NEXT PRODUCTION CYCLE
A10 10 10 A29-21 CHECK LISTING FOR DAMAGED MARS-CARDS
A11 9 8 A29-22 TAKE OUT FAULTY CARDS FROM EACH DECK
A12 10 9 A29-23 REPLACE MARS-CARDS TO THEIR RESPECTIVE #6P-GEN-KIT FOR REUSE
A13 11 10 A29-24 DISTRIBUTE #6P-GEN-KITS TO TEACHERS' MAILBOXES
A14 12 5 A29-40 PERFORM TRUANCY RELATED HUMAN COMMUNICATION
A15 13 A29-41 PARTITION TRUANCY REPORT "A" INTO SMALL SECTIONS
A16 14 3 A29-42 HAND OUT EACH SECTION TO PRE-ASSIGNED PERSONAL
A17 15 A29-43 PHONE PARENTS OF TRUANT STUDENTS
A18 16 6 A29-44 HEET TRUANT STUDENT
A19 17 6 A29-45 HEET PARENTS OF TRUANT STUDENT
A20 18 7 A29-46 RELATE INFORMATION ON HAND
A21 19 7 A29-21 HEET STAFF
A22 20 8 A29-22 QUERY #66, FOR RETROACTIVE STUDENT TRUANCY INFORMATION
A23 21 9 A29-23 QUERY #66, FOR RETROACTIVE CLASS TRUANCY INFORMATION
A24 22 10 A29-24 QUERY #66, FOR OVERALL TRUANCY STATISTICS
A25 23 11 A29-25 NO ACTION - NEXT CLUSTER
A26 24 1 A29-26 WAIT
SYSTALL A(30) = A30 \rightarrow \text{DATA USAGE DETERMINATION}

\begin{align*}
\text{DATABASE WITH 14 ACTIONS} \\
\text{SYSTALL A30} \rightarrow \{1, \ldots, 14\}
\end{align*}

DATA USAGE DETERMINATION IS THE MOST IMPORTANT FEATURE IN THE SUCCESSFUL
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 ALTOGETHER.

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 DECISION-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 USEFUL.

1. \text{OBSERVATION}

1.1 \text{USAGE VERIFICATION STUDY} \rightarrow \begin{align*}
2 \rightarrow \text{OUTPUT OPTIMIZATION} \\
2.1 \rightarrow \text{REPORT A IS TOO LONG - LOW USAGE} \\
2.2 \rightarrow \text{REPORT B IS TOO LONG - LOW USAGE} \\
2.3 \rightarrow \text{REPORT A IS MAYBE TOO SHORT} \\
2.4 \rightarrow \text{REPORT B IS MAYBE TOO SHORT} \\
2.5 \rightarrow \text{OPTIMUM OUTPUT LEVEL} \\
2.6 \rightarrow \text{SYSTEM LOGIC PROBLEM} \\
2.7 \rightarrow \text{DATA USAGE STUDY IS COMPLETE}
\end{align*}

\begin{align*}
\text{THEN-GO-TO-CLUSTER} \rightarrow 2 \\
\text{THEN-GO-TO-CLUSTER} \rightarrow 2 \\
\text{THEN-GO-TO-CLUSTER} \rightarrow 2 \\
\text{THEN-GO-TO-CLUSTER} \rightarrow 2 \\
\text{THEN-GO-TO-CLUSTER} \rightarrow 2 \\
\text{THEN-GO-TO-CLUSTER} \rightarrow 0 \\
\text{THEN-GO-TO-CLUSTER} \rightarrow 0
\end{align*}
SYSTAT A(30) = A30 = DATA USAGE DETERMINATION

= DATABASE WITH 14 ACTIONS

= SYSTAT A30 - (1..14)

DATA USAGE DETERMINATION IS THE MOST IMPORTANT FEATURE IN THE SUCCESSFUL MODELING AND IMPLEMENTATION OF THIS DATABASE 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 ALL TOGETHER.

SUBSEQUENT DATA USAGE STUDIES WILL PERMIT TO CLOSELY TUNE IN THE REPORTING PART OF THE SYSTEM TO THE REAL AND OFTEN UNEXPERIENCED NEED OF THE USERS IN HIS DECISION 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 USEFUL.

1. === OBSERVATION

1.1 = USAGE VERIFICATION STUDY

1.1 IF ALL OUTPUT IS REDISTRIBUTED IN THE PREDETERMINED DELAY AND IF

1.1.1 OUTPUT IS COMPLETE

1.1.2 OUTPUT IS NOT COMPLETE

2. === OUTPUT OPTIMIZATION

2.1 = REPORT A IS TOO LONG - LOW USAGE

2.1 IF "A" RATIO IS SMALL AND IF NOT DESIGN LOGIC PROBLEM DETECTED

2.1.1 "A" RATIO IS HIGH

2.1.2 "A" RATIO IS HIGH

2.2 = REPORT B IS TOO LONG - LOW USAGE

2.2 IF "B" RATIO IS SMALL AND IF NOT DESIGN LOGIC PROBLEM DETECTED

2.2.1 "B" RATIO IS HIGH

2.2.2 "B" RATIO IS HIGH

2.3 = REPORT A IS MAYBE TOO SHORT

2.3 IF "A" RATIO IS HIGH

2.4 = REPORT B IS MAYBE TOO SHORT

2.4 IF "B" RATIO IS HIGH

2.5 = OPTIMUM OUTPUT LEVEL

2.5 IF NOT "A" RATIO IS SMALL AND IF NOT "B" RATIO IS SMALL AND IF "A" RATIO IS HIGH AND IF "B" RATIO IS HIGH

2.6 = SYSTEM LOGIC PROBLEM

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 COMPLETE

2.7 IF DATA USAGE STUDY IS COMPLETE
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 SUCCESSFUL
SYSTEMS AND IMPLEMENTATION OF THE DATA SYSTEM INITIATIVE 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 ALTOGETHER.

SUBSEQUENT DATA USAGE STUDIES WILL PERMIT TO CLOSELY TUNE IN THE
REPORTING PART OF THE SYSTEM TO THE REAL AND OFTEN UNEXRESSED NEED
OF THE USER IN HIS DECISION 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 USEFUL.

1. Observation

1.1 = USAGE VERIFICATION STUDY

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 (@MEDPH.)
A30-2) DETERMINE NUMBER OF STUDENTS GET IN PAP. (@MEDRT.)
A30-3) DETERMINE NUMBER OF STUDENTS ON REPORT "A" (@MEDRT.)
A30-4) DETERMINE RATIO "A" = MEDPH./MEDRT.
A30-5) DETERMINE RATIO "B" = MEDRT./MEDRT.

2. Observation

2.1 = REPORT A IS TOO LONG - LOW USAGE

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

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 NOT TOO SHORT
   2.3 IF "A" RATIO IS HIGH
       A30-12 = LESENSE REPORT "A" FOR THE NEXT PRODUCTION CYCLE
       A30-12 = ADJUST OUTPUT SELECTION PARAMETERS FOR THE NEXT PRODUCTION CYCLE

2.4 = REPORT B IS NOT TOO SHORT
   2.4 IF "B" RATIO IS HIGH
       A30-10 = LESENSE REPORT "B" FOR THE NEXT PRODUCTION CYCLE
       A30-12 = ADJUST OUTPUT SELECTION PARAMETERS FOR THE NEXT PRODUCTION CYCLE

2.5 = OPTIMUM OUTPUT LEVEL
   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
   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 COMPLETE
   2.7 IF DATA USAGE STUDY IS COMPLETE
       A30-13 = NO ACTION - NEXT CLUSTER

     THEN-GO-TO-CLUSTER -> 2
SYSTALL A(30) = A(30) = DATA USAGE DETERMINATION

DATA USAGE DETERMINATION IS THE MOST IMPORTANT FEATURE IN THE SUCCESSFUL MODELING AND IMPLEMENTATION OF THIS DATA SYSTEM. INITIALLY BY STUDYING DATA USAGE, ONE CAN DETECT MAJOR FLAWS IN THE LOGICAL DESIGN OF THE SYSTEM, LONG BEFORE THE USER BECOMES FRUSTRATED BY ITS SHORTCOMINGS AND REJECTS THE SYSTEM ALTOGETHER.

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 DECISION MAKING TASK. IN THIS WAY WE CAN OPTIMIZE OUTPUT USAGE WHILE MINIMIZING OUTPUT VOLUME AND THE INHERENT DATA PROCESSING EXPENSES.

WHILE MEASURING OUTPUT USAGE OFFERS WITH EACH TYPE OF OUTPUT, BASICALLY WE HAVE TO ESTABLISH A COMPARATIVE STANDARD OF HOW MUCH ATTENTION THEY SHOULD GET TO BE CONSIDERED USEFUL.

1 2 3 4 5 6 7 8 9 (ALTERNATIVES)

CLUSTERS

C1 2 2 2 2 2 2 0 0 === OBSERVATION
C2 2 2 2 2 2 2 0 0 === OUTPUT OPTIMIZATION

PREDICATES

ALL OUTPUT IS REDISTRIBUTED IN THE PREDETERMINED DELAY
"A" RATIO IS SMALL
"B" RATIO IS SMALL
"A" RATIO IS HIGH
"B" RATIO IS HIGH
DATA USAGE STUDY IS COMPLETE
OBSERVATION IS COMPLETE
DESIGN LOGIC PROBLEM DETECTED

ACTIONS

A(30-1) DETERMINE NUMBER OF PHONE CALLS TO PARENTS (ANMP.H.)
A(30-2) DETERMINE NUMBER OF STUDENTS MET BY PMP. (ANMP.T.)
A(30-3) DETERMINE NUMBER OF STUDENTS ON REPORT "A" (ANA.H.)
A(30-4) DETERMINE NUMBER OF STUDENTS ON REPORT "B" (ANB.H.)
A(30-5) DETERMINE RATIO "A" = ANMP.H./ANMP.T.
A(30-6) DETERMINE RATIO "B" = ANMP.H./ANMP.T.
A(30-7) SHORTER REPORT "A" FOR THE NEXT PRODUCTION CYCLE
A(30-8) LEGEND REPORT "A" FOR THE NEXT PRODUCTION CYCLE
A(30-9) SHORTER REPORT "B" FOR THE NEXT PRODUCTION CYCLE
A(30-10) MAINTAIN OUTPUT LEVEL FOR THE NEXT PRODUCTION CYCLE
A(30-11) ADJUST OUTPUT SELECTION PARAMETERS FOR THE NEXT PRODUCTION CYCLE
A(30-12) NO ACTION (USE CLUSTER)
A(30-13) REVIEW SYSTEM LOGIC
A(30-14) REVIEW SYSTEM LOGIC

87
SYSTALL ACTION(11) = A11> = UPDATE THE STUDENT MASTER FILE (@ STOPDATE)


THE LOGICAL DESCRIPTION OF SYSTALL A11> IS 3 LEVELS DEEP

SYSTALL A11> = @STOPDATE WITH 27 ACTIONS
  = SYSTALL A11>- (11, 27)
  = IT CALLS IN 2 SUB-RUTINIES WHILE EXECUTING
     AT A11-14 #STRECHM
     AT A11-15 #STRECHM

SYSTALL A11-14> = STUDENT RECORD MODIFICATION
  = #STRECHM WITH 16 ACTIONS
  = SYSTALL A11-14>- (11, 16)
  = IT CALLS IN 2 SUB-RUTFINES WHILE EXECUTING
     AT A11-14-5 #STRECHMPR
     AT A11-14-6 #STRECHMCR

SYSTALL A11-14-5> = STUDENT RECORD PERSONAL ITEMS MODIFICATION ROUTINE
  = #STRECHMPR WITH 7 ACTIONS
  = SYSTALL A11-14-5>- (1..7)

SYSTALL A11-14-6> = STUDENT RECORD COURSE CHANGE ROUTINE
  = #STRECHMR WITH 14 ACTIONS
  = SYSTALL A11-14-6>- (1..14)

SYSTALL A11-15> = STUDENT RECORD DELETION ROUTINE
  = #STRECHM WITH 6 ACTIONS
  = SYSTALL A11-15>- (1..6)

CONTINUED ON THE NEXT PAGE
1. == START THE UPDATING SESSION
   1.1 == STARTING UPDATE
   2. == UPDATING FUNCTION SELECTION
   2.1 == COMMAND SELECTION
   2.2 == OPERATING MODE SELECTION
   2.3 == NON VALID CREATION
   2.4 == VALID CREATION
   2.5 == VALID MODIFICATION
   2.6 == NON VALID MODIFICATION
   2.7 == VALID DELETION
   2.8 == NON VALID DELETION
   2.9 == UNAUTHORIZED USER
   2.10 == END COMMAND WITH LISTING
   2.11 == END COMMAND WITH NO LISTING
3. == RECORD CREATION
   3.1 == RECORD CREATION
4. == RECORD MODIFICATION
   4.1 == MODIFY EXISTING RECORD
5. == RECORD DELETION
   5.1 == DELETE VALID RECORD
6. == TERMINATION OF THE UPDATING SESSION
   6.1 == TERMINATE WITH LISTING
   6.2 == TERMINATE WITHOUT LISTING
STUDENT ACTION(11) = A11> = UPDATE THE STUDENT MASTER FILE (0 STUPDATE)

STUPDATE IS ONE OF THE THREE COMPONENTS OF THE UPDATE SUBSYSTEM (0 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 MERGE 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 MARS-CARD FILE) ARE ALL UPDATED
AUTOMATICALLY.

THE LOGICAL DESCRIPTION OF SYSTALL A11> IS 3 LEVELS DEEP

SYSTALL A11> = STUPDATE WITH 27 ACTIONS
    = SYSTALL A11>-(11.27)
    = IT CALLS IN 2 SUB-Routines WHILE EXECUTING
      AT A11-14 @STRECHER
      AT A11-15 @STRECHER

SYSTALL A11-14> = STUDENT RECORD MODIFICATION
    = @STRECHER WITH 16 ACTIONS
    = SYSTALL A11-14>-(14.16)
    = IT CALLS IN 2 SUB-Routines WHILE EXECUTING
      AT A11-14-5 @STRECHER
      AT A11-14-6 @STRECHER

SYSTALL A11-14-5> = STUDENT RECORD PERSONAL ITEMS
    MODIFICATION ROUTINE
    = @STRECHER WITH 7 ACTIONS
    = SYSTALL A11-14-5>-(15.15)

SYSTALL A11-14-6> = STUDENT RECORD COURSE CHANGE ROUTINE
    = @STRECHER WITH 14 ACTIONS
    = SYSTALL A11-14-6>-(16.16)

SYSTALL A11-15> = STUDENT RECORD DELETION ROUTINE
    = @STRECHER WITH 6 ACTIONS
    = SYSTALL A11-15>-(17.17)

1. == START THE UPDATING SESSION
1.1 = STARTING UPDATE
1.1 IF COMMAND = CALL NORETUR
CONTINUED ON THE NEXT PAGE

==< 2 >==
THEN-GET-TO-CLUSTER-> 2
2. --- UPDATING FUNCTION SELECTION

2.1 --- COMMAND SELECTION

2.1 IF COMMAND = NONE, BRIEF AND IF
   NOT COMMAND = NEW, STUPID NO., AND IF
   NOT COMMAND = DEL, STUDENT NO., AND IF
   NOT COMMAND = CRE, STUDENT NO., AND IF
   NOT COMMAND = FND, AND IF
   NOT COMMAND = FND, NO. LIST, AND IF
   COMMAND = ILLEGAL

2.2 --- OPERATING MORE SELECTION

2.2 IF COMMAND = MORE, BRIEF AND IF
   MORE STUDENT RECORD TO UPDATE

2.3 --- NON VALID CREATION

2.3 IF COMMAND = CRE, STUDENT NO., AND IF
   NOT STUDENT NO. IS VALID STUDENT ON FILE AND IF
   MORE Student RECORD TO UPDATE

2.4 --- VALID CREATION

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

2.5 IF COMMAND = NEW, STUDENT NO., AND IF
   NOT STUDENT NO. IS VALID STUDENT ON FILE AND IF
   MORE STUDENT RECORD TO UPDATE

2.6 --- NON VALID MODIFICATION

2.6 IF COMMAND = NEW, STUDENT NO., AND IF
   NOT STUDENT NO. IS VALID STUDENT ON FILE AND IF
   MORE STUDENT RECORD TO UPDATE

2.7 --- VALID DELETION

2.7 IF COMMAND = DEL, STUDENT NO., PASSWORD AND IF
   NOT STUDENT NO. IS VALID STUDENT ON FILE AND IF
   PASSWORD IS VALID AND IF
   MORE STUDENT RECORD TO UPDATE

2.8 --- NON VALID DELETION

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

2.9 IF COMMAND = DEL, STUDENT NO., PASSWORD AND IF
   NOT STUDENT NO. IS VALID STUDENT ON FILE AND IF
   PASSWORD IS VALID AND IF
   MORE STUDENT RECORD TO UPDATE

CONTINUED ON THE NEXT PAGE
2.10—END COMMAND WITH LISTING
   2.10 IF COMMAND = FIN AND IF
       NOT MORE STUDENT RECORD TO UPDATE
2.11—END COMMAND WITH NO LISTING
   2.11 IF COMMAND = FIN+NO LIST AND IF
       NOT MORE STUDENT RECORD TO UPDATE
3.== RECORD CREATION
   3.1==RECORD CREATION
      3.1P NO CONDITIONS
4.== RECORD MODIFICATION
   4.1==MODIFY EXISTING RECORD
      4.1P NO CONDITIONS
5.== RECORD DELETION
   5.1==DELETE VALID RECORD
      5.1P NO CONDITIONS
6.== TERMINATION OF THE UPDATING SESSION
   6.1==TERMINATE WITH LISTING
      6.1 IF COMMAND = FIN
   6.2==TERMINATE WITHOUT LISTING
      6.2 IF COMMAND = FIN+NO LIST

== -THEN-GO-TO-CLUSTER-> 6
== -THEN-GO-TO-CLUSTER-> 6
== -THEN-GO-TO-CLUSTER-> 2
== -THEN-GO-TO-CLUSTER-> 2
== -THEN-GO-TO-CLUSTER-> 2
== -THEN-GO-TO-CLUSTER-> 2
== -THEN-GO-TO-CLUSTER-> 2
== -THEN-GO-TO-CLUSTER-> 0
== -THEN-GO-TO-CLUSTER-> 0

$<2>
$<2>
$<2>
$<0,0>
SYSTALL ACTION(11) = A11 = UPDATE THE STUDENT MASTER FILE (# STUPDATE)


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 $STRETCMEL

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 $STRECDIMP3
    AT A11-14-6 $STRECDIMCR

SYSTALL A11-14-5 = STUDENT RECORD PERSONAL ITEMS MODIFICATION ROUTINE
  = $STRECDIMP3 WITH 7 ACTIONS
  = SYSTALL A11-14-5-(1..7)

SYSTALL A11-14-6 = STUDENT RECORD COURSE CHANGE ROUTINE
  = $STRECDIMCR WITH 14 ACTIONS
  = SYSTALL A11-14-6-(1..14)

SYSTALL A11-15 = STUDENT RECORD DELETION ROUTINE
  = $STRECDL WITH 6 ACTIONS
  = SYSTALL A11-15-(1..6)

CONTINUED ON THE NEXT PAGE
1.== START THE UPDATING SESSION

1.1== STARTING UPDATE
1.1 IF COMMAND = CALL HOMEDIR
   A11-10 ATTACH COMPILED INTERACTIVE PROGRAM
   A11-15 ATTACH COURSE MASTER FILE
   A11-20 ATTACH CLASS MASTER FILE
   A11-25 ATTACH STUDENT MASTER FILE
   A11-30 ATTACH IAS-CARD FILE
   A11-35 OPEN COURSE MASTER FILE
   A11-40 OPEN STUDENT MASTER FILE
   A11-45 OPEN IAS-CARD FILE
   A11-50 CLEAR ALL BUFFERS
   A11-55 REQUEST COMMAND FROM USER
   A11-60 ENTER COMMAND

2.== UPDATING FUNCTION SELECTION

2.1== COMMAND SELECTION
2.1 IF NOT COMMAND = MORE; BRIEF AND IF
   NOT COMMAND = ADD-REMOVE NO. AND IF
   NOT COMMAND = DEL-REMOVE NO.: PASSWORD AND IF
   NOT COMMAND = CSE-REPORT NO. AND IF
   NOT COMMAND = FIN. AND IF
   COMMAND = ILL. COMMAND
   A11-10 WRITE APPROPRIATE ERROR MESSAGE
   A11-25 REQUEST COMMAND FROM USER
   A11-30 ENTER COMMAND

2.2== OPERATING NO. SELECTION
2.2 IF COMMAND = MORE; BRIEF AND IF
   MORE STUDENT RECORD TO UPDATE
   A11-15 SET UP Flag TO SUPPRESS VERIFICATION MESSAGES
   A11-20 REQUEST COMMAND FROM USER
   A11-25 ENTER COMMAND

2.3== NEW VALID CREATION
2.3 IF COMMAND = CSE-REPORT NO. AND IF
   STUDENT NO. IS VALID STUDENT ON FILE AND IF
   MORE STUDENT RECORD TO UPDATE
   A11-10 WRITE APPROPRIATE ERROR MESSAGE
   A11-25 REQUEST COMMAND FROM USER
   A11-30 ENTER COMMAND

2.4== VALID CREATION
2.4 IF COMMAND = CSE-REPORT 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
2.5 IF COMMAND = NEW 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
2.6 IF COMMAND = NEW 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
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
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
2.9 IF COMMAND = DEL STUDENT NO., PASSWORD 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
A11-22> CLOSE CLASS MASTER FILE
A11-25> CLOSE COURSE MASTER FILE
A11-23> CLOSE ADR-CARD FILE
A11-19> LOG OFF -ILLEGAL USER AT THE TERMINAL-

2.10=END COMMAND WITH LISTING
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
2.11 IF COMMAND = FIN-NO-LIST AND IF
NOT MORE STUDENT RECORD TO UPDATE
2.11A A11-27> NO ACTION; NEXT CLUSTER
CONTINUED ON THE NEXT PAGE
### 3. RECORD CREATION

#### 3.1 RECORD CREATION

3.1P: NO CONDITIONS

- A11-10: CLEAR ALL BUFFERS
- A11-12: CREATE EMPTY RECORD IN STUDENT MASTER FILE
- A11-14: PERFORM RECORD MODIFICATION; CALL #STRECCRE ROUTINE
- A11-25: REQUEST COMMAND FROM USER
- A11-26: ENTER COMMAND

### 4. RECORD MODIFICATION

#### 4.1 MODIFY EXISTING RECORD

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

#### 5.1 DELETE VALID RECORD

5.1P: NO CONDITIONS

- A11-10: CLEAR ALL BUFFERS
- A11-13: GET STUDENT RECORD INTO BUFFER
- A11-14: PERFORM RECORD DELETION; CALL #STRECDEL ROUTINE
- A11-25: REQUEST COMMAND FROM USER
- A11-26: ENTER COMMAND

### 6. TERMINATION OF THE RECORD-UPDATE SESSION

#### 6.1 TERMINATE WITH LISTING

6.1 IF COMMAND = FIN

- A11-17: END UPDATING SESSION
- A11-18: PRINT ALL MODIFIED CLASS LIST
- A11-20: CLOSE STUDENT MASTER FILE
- A11-22: CLOSE CLASS MASTER FILE
- A11-23: CLOSE COURSE MASTER FILE
- A11-24: BACK UP ALL DATABASE FILES

#### 6.2 TERMINATE WITHOUT LISTING

6.2 IF COMMAND = FIN+NDLIST

- 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 AHS-CARD FILE
- A11-24: BACK UP ALL DATABASE FILES
SYSTALL ACTION(11) = A11> = UPDATE THE STUDENT MASTER FILE (# STUPDATE)


THE LOGICAL DESCRIPTION OF SYSTALL A11> IS 3 LEVELS DEEP

SYSTALL A11> = STUPDATE WITH 27 ACTIONS
= SYSTALL ALL>-1(1, 27)
= IT CALLS IN 2 SUB-ROUTINES WHILE EXECUTING
   AT A11-14 #STRECORD
   AT A11-15 #STRECORD.

SYSTALL A11-14> = STUDENT RECORD MODIFICATION
= #STRECORD WITH 14 ACTIONS
= SYSTALL A11-14>-1(1, 16)
= IT CALLS IN 2 SUB-ROUTINES WHILE EXECUTING
   AT A11-14-5 #STRECORDER
   AT A11-14-4 #STRECORDER.

SYSTALL A11-14-5> = STUDENT RECORD PERSONAL ITEMS MODIFICATION ROUTINE
= #STRECORDER WITH 7 ACTIONS
= SYSTALL A11-14-5>-1(1, 7).

SYSTALL A11-14-6> = STUDENT RECORD COURSE CHANGE ROUTINE
= #STRECORDER WITH 14 ACTIONS
= SYSTALL A11-14-6>-1(1, 14).

SYSTALL A11-15> = STUDENT RECORD DELETION ROUTINE
= #STRECORD WITH 6 ACTIONS
= SYSTALL A11-15>-1(1, 6)

CONTINUED ON THE NEXT PAGE
CLUSTERS -

- START THE UPDATING SESSION
- UPDATING FUNCTION SELECTION
- RECORD CREATION
- RECORD MODIFICATION
- RECORD DELETION
- TERMINATION OF THE UPDATING SESSION

PREDICATES -

- COMMAND = CALL MODULE
- COMMAND = HOME; BRIEF
- COMMAND = ADD; STUDENT; NO;
- COMMAND = DEL; STUDENT; NO;
- COMMAND = CRE; STUDENT; NO;
- COMMAND = FIN;
- COMMAND = FIN; LIST
- COMMAND = ILLEGAL COMMAND

ACTIONs -

- ATTACH MAINTAINED DATA BASE
- ATTACH COURSE MASTER FILE
- ATTACH CLASS MASTER FILE
- ATTACH STUDENT MASTER FILE
- ATTACH M/M-CARD FILE
- OPEN COURSE MASTER FILE
- OPEN STUDENT MASTER FILE
- OPEN M/M-CARD FILE
- CLEAR ALL RECORDS
- SET UP FLAG TO SUPPRESS VERIFICATION MESSAGES
- CREATE EMPTY RECORD IN STUDENT MASTER FILE
- SET STUDENT RECORD INTO BUFFER
- PERFORM RECORD MODIFICATION; CALL ORM/REC ROUTINE
- PERFORM RECORD DELETION; CALL ORM/REC ROUTINE
- WRITE APPROPRIATE ERROR MESSAGE
- END UPDATING SESSION
- PRINT ALL MODIFIED CLASSLIST
- LOG OFF: ILLEGAL USER AT THE TERMINAL
- CLOSE STUDENT MASTER FILE
- CLOSE CLASS MASTER FILE
- CLOSE COURSE MASTER FILE
- BACK UP ALL DATA BASE FILES
- REQUEST COMMAND FROM USER
- ENTER COMMAND
- NO ACTION; NEXT CLUSTER
5. ABL-ROUTINE STRECMOD;

Systall action(11-14) = all-14 = student record modification routine
= STRECMOD routine with 16 actions
= Systall all-14-(1..16)

This routine is part of the student master file updatr 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 11-14-5 = calls STRECMODR routine
= student record's personal items

AT Systall 11-14-6 = calls STRECMODCR routine
= student record's course change routine

At the end return to STUPDATE

1.=== enter STRECMOD routine

1.1=== ENTER STRECMOD routine

2.=== Field content verification

2.1=== display personal items

2.2=== display students registered courses

2.3=== Next cluster; no action

3.=== Field editing

3.1=== perform personal changes

3.2=== perform course change

3.3=== perform short personal change

3.4=== perform short course change

3.5=== Next cluster; no action

4.=== Finalisation of entered changes

4.1=== Finalization

4.2=== Total error recovery

4.3=== Loop back.
SYSTALL ACTION(11-14) = A11-14O = STUDENT RECORD MODIFICATION ROUTINE
= SYSTCALL ROUTINE WITH 16 ACTIONS
= SYSTALL A11-14O-(1..16)

THIS ROUTINE IS PART OF THE STUDENT MASTER FILE UPDATE PROGRAM (USTUPDATE)
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 A11-14=5 = CALLS OSTRECMDRQ ROUTINE
= STUDENT RECORD'S PERSONAL ITEMS
MODIFICATION ROUTINE

AT SYSTALL A11-14=6 = CALLS OSTRSTCMR ROUTINE
= STUDENT RECORD'S COURSE CHANGE ROUTINE
AT THE END RETURN TO USTUPDATE

1.=== ENTER OSTRECMDRQ ROUTINE

1.1== ENTER OSTRECMDRQ ROUTINE
1.1.1 IF NO CONDITIONS

2.=== FIELD CONTENT VERIFICATION

2.1== DISPLAY PERSONAL ITEMS
2.1.1 IF DISPLAY STUDENTS PERSONAL ITEMS

2.2== DISPLAY STUDENTS REGISTERED COURSES
2.2.1 IF DISPLAY STUDENTS REGISTERED COURSES

2.3== NEXT CLUSTER NO ACTION
2.3.1 IF NOT DISPLAY STUDENTS PERSONAL ITEMS AND IF
NOT DISPLAY STUDENTS REGISTERED COURSES

3.=== FIELD EXITING

3.1== PERFORM PERSONAL CHANGES
3.1.1 IF CHANGE IS ON PERSONAL ITEMS

3.2== PERFORM COURSE CHANGE
3.2.1 IF CHANGE IS ON COURSES

3.3== PERFORM SHORT PERSONAL CHANGE
3.3.1 IF CHANGE IS ON PERSONAL ITEMS AND IF
VERIFYING MESSAGES ARE SUPPRESSED

3.4== PERFORM SHORT COURSE CHANGE
3.4.1 IF CHANGE IS ON COURSES AND IF
VERIFYING MESSAGES ARE SUPPRESSED

3.5== NEXT CLUSTER NO ACTION
3.5.1 IF NOT CHANGE IS ON PERSONAL ITEMS AND IF
NOT CHANGE IS ON COURSES

CONTINUED ON THE NEXT PAGE
4.== FINALISATION OF ENTERED CHANGES

4.1 == FINALISATION

   4.1 IF NOT MORE MODIFICATIONS OR DISPLAY ON THIS RECORD AND IF
   MAKE ENTERED MODIFICATIONS PERMANENT

4.2 == TOTAL ERROR RECOVERY

   4.2 IF NOT MORE MODIFICATIONS OR DISPLAY ON THIS RECORD AND IF
   NOT MAKE ENTERED MODIFICATIONS PERMANENT

4.3 == LOOP BACK

   4.3 IF MORE MODIFICATIONS OR DISPLAY ON THIS RECORD
SYSTALL ACTION(A1-14) = STUDENT RECORD MODIFICATION ROUTINE

THIS ROUTINE IS PART OF THE STUDENT MASTER FILE UPDATER PROGRAM (#STUPD).
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

1. AT SYSTALL A(11-14-5) = CALLS #STRECHMR ROUTINE
   = STUDENT RECORD'S PERSONAL ITEMS MODIFICATION ROUTINE

2. AT SYSTALL A(11-14-6) = CALLS #STRECHMR ROUTINE
   = STUDENT RECORD'S COURSE CHANGE ROUTINE

AT THE END RETURN TO #STUPD

1.== ENTER #STRECHMR ROUTINE

1.1== ENTER #STRECHMR ROUTINE

1.1P NO CONDITIONS

   A11-14-15 = REQUEST COMMAND FROM THE USER

2.== FIELD CONTENT VERIFICATION

2.1== DISPLAY PERSONAL ITEMS

   2.1 IF DISPLAY STUDENT PERSONAL ITEMS
   A11-14-15 = ENTER COMMAND = PERSONNEL
   A11-14-15 = DISPLAY ALL PERSONAL ITEMS
   A11-14-15 = REQUEST COMMAND FROM THE USER

2.2== DISPLAY STUDENTS REGISTERED COURSES

   2.2 IF DISPLAY STUDENTS REGISTERED COURSES
   A11-14-15 = ENTER COMMAND = COURSE
   A11-14-15 = DISPLAY ALL REGISTERED COURSES
   A11-14-15 = REQUEST COMMAND FROM THE USER

2.3== NEXT CLUSTER, NO ACTION

   2.3 IF NOT DISPLAY STUDENT PERSONAL ITEMS AND IF
   NOT DISPLAY STUDENTS REGISTERED COURSES
   A11-14-14 = NO ACTION, NEXT CLUSTER

CONTINUED ON THE NEXT PAGE
3. FIELD EDITING

3.1 PERFORM PERSONAL CHANGES
3.1.1 IF CHANGE IS ON PERSONAL ITEMS
   A11-14-2D DISPLAY ALL PERSONAL ITEMS
   A11-14-1D PERFORM PERSONAL ITEM CHANGE - CALL @STRECHOPPER-
   A11-14-15D REQUEST COMMAND FROM THE USER

3.2 PERFORM COURSE CHANGE
3.2.1 IF CHANGE IS ON COURSES
   A11-14-4D PERFORM COURSE CHANGE - CALL @STRECHCR-
   A11-14-4D DISPLAY ALL REGISTERED COURSES
   A11-14-15D REQUEST COMMAND FROM THE USER

3.3 PERFORM SHORT PERSONAL CHANGE
3.3.1 IF CHANGE IS ON PERSONAL ITEMS AND IF
   VERIFYING MESSAGES ARE SUPPRESSED
   A11-14-5D PERFORM PERSONAL ITEM CHANGE - CALL @STRECHOPPER-
   A11-14-15D REQUEST COMMAND FROM THE USER

3.4 PERFORM SHORT COURSE CHANGE
3.4.1 IF CHANGE IS ON COURSES AND IF
   VERIFYING MESSAGES ARE SUPPRESSED
   A11-14-6D PERFORM COURSE CHANGE - CALL @STRECHCR-
   A11-14-15D REQUEST COMMAND FROM THE USER

3.5 NEXT CLUSTER NO ACTION
3.5.1 IF NOT CHANGE IS ON PERSONAL ITEMS AND IF
   NOT CHANGE IS ON COURSES
   A11-14-16D NO ACTION, NEXT CLUSTER

CONTINUED ON THE NEXT PAGE
4.---FINALISATION OF ENTERED CHANGES

4.1---FINALIZATION

4.1 IF NOT MORE MODIFICATIONS OR DISPLAY ON THIS RECORD AMB IF

MAKE ENTERED MODIFICATIONS PERMANENT

AII-14-7> ENTER COMMAND = COMPLETE
AII-14-8> UPDATE CLASS MASTER FILE RECORDS
AII-14-9> UPDATE COURSE MASTER FILE RECORDS
AII-14-10> APPEND NEW LINES TO NAMS-CARD FILE
AII-14-11> REWRITE STUDENT RECORD FROM BUFFER TO STUDENT MASTER FILE

4.2---TOTAL ERROR RECOVERY

4.2 IF NOT MORE MODIFICATIONS OR DISPLAY ON THIS RECORD AMB IF

NOT MAKE ENTERED MODIFICATIONS PERMANENT

AII-14-12> ENTER COMMAND = QUIT
AII-14-13> DISREGARD ALL CHANGES TO THE RECORD
AII-14-14> CLEAR RECORD FROM BUFFER

4.3---LOOP BACK

4.3 IF MORE MODIFICATIONS OR DISPLAY ON THIS RECORD

AII-14-16> NO ACTION; NEXT CLUSTER

---0< 0, OR 2> ---0
SYSTALL ACTION(11-14) = A11-14 = STUDENT RECORD MODIFICATION ROUTINE

= STRECHRD ROUTINE WITH 16 ACTIONS
= SYSTALL A11-14-(1,1,16)

THIS ROUTINE IS PART OF THE STUDENT MASTER FILE UPDATE 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 @STRECHRD ROUTINE
= STUDENT RECORD'S PERSONAL ITEMS
MODIFICATION ROUTINE

AT SYSTALL A(11-14-6) = CALLS @STRECHRD ROUTINE
= STUDENT RECORD'S COURSE CHANGE ROUTINE

AT THE END RETURN TO @STUPDATE

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td></td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CLUSTERS -

<table>
<thead>
<tr>
<th>P1</th>
<th>Y</th>
<th>N</th>
<th></th>
<th>0</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P2</td>
<td>Y</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P5</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P6</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P7</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PREDICATES -

DISPLAY STUDENT PERSONAL ITEMS
DISPLAY STUDENTS REGISTERED COURSES
CHANGE IS ON PERSONAL ITEMS
CHANGE IS ON COURSES
MORE MODIFICATIONS OR DISPLAY ON THIS RECORD
MAKE ENTERED MODIFICATIONS PERMANENT
VERIFYING MESSAGES ARE SUPPRESSED

ACTIONS -

<table>
<thead>
<tr>
<th>A1</th>
<th>1</th>
<th>2</th>
<th></th>
<th>1</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td></td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>A11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>A12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>A13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>A14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>A15</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A16</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

A11-14-1> ENTER COMMAND = PERSONNEL
A11-14-2> DISPLAY ALL PERSONAL ITEMS
A11-14-3> ENTER COMMAND = COURSE
A11-14-4> DISPLAY ALL REGISTERED COURSES
A11-14-5> PERFORM PERSONAL ITEM CHANGE - CALL STRECHRD-
A11-14-6> PERFORM COURSE CHANGE - CALL STRECHRD-
A11-14-7> ENTER COMMAND = COMPLETE
A11-14-8> UPDATE MASTER FILE RECORDS
A11-14-9> UPDATE COURSE MASTER FILE RECORDS
A11-14-10> APPEND NEW LINES TO CARD-CARD FILE
A11-14-11> REWRITE STUDENT RECORD FROM BUFFER TO STUDENT MASTER FILE
A11-14-12> ENTER COMMAND = QUIT
A11-14-13> DISREGARD ALL CHANGES TO THE RECORD
A11-14-14> CLEAR RECORD FROM BUFFER
A11-14-15> REQUEST COMMAND FROM THE USER
A11-14-16> NO ACTION; NEXT CLUSTER
SYSTALL A(11-14-5) = PERFORM PERSONAL ITEM CHANGE
ON THE STUDENT RECORD
= STRECHMER WITH 7 ACTIONS
= SYSTALL A(11-14-5 (1..7))

THIS ROUTINE IS CALLED BY STRECHMER WHEN THE OPERATOR ENTERED COMMANDS
TO PERFORM MODIFICATIONS IN THE STUDENT RECORD'S PERSONAL AREA.
THIS IS DONE ITEM BY ITEM IN INTERACTIVE MODE.
AT THE END RETURN TO STRECHMER

1.=== ENTER STRECHMER ---$  
1.1=== ENTER ROUTINE STRECHMER  
2.=== ITEM SELECTION ---$  
2.1=== FIELDS ARE EMPTY  
2.2=== SELECT FIELDS TO BE EDITED  
3.=== DATA ENTERING ---$  
3.1=== ENTER AND VERIFY SELECTIVE DATA  
3.2=== ERROR - ITEM OUT OF RANGE -  
3.3=== ENTER SELECTIVE DATA  
3.4=== ENTER ALL DATA ITEMS

< 2 >  
< 3 >  
< 3 >  
< 0; 0; 0; 0 >  
< 0 >  
< 0 >  
< 0 >
7. AML-Routine STRECHOPPER

SYSTALL A11-14-5 = PERFORM PERSONAL ITEM CHANGE
ON THE STUDENT RECORD
= STRECHOPPER WITH 7 ACTIONS
= SYSTALL A11-14-2-(1..7)

THIS ROUTINE IS CALLED BY STRECHOPPER WHEN THE OPERATOR ENTERED COMMANDS
TO PERFORM MODIFICATIONS IN THE STUDENT RECORD'S PERSONAL AREA.
THIS IS DONE ITEM BY ITEM IN INTERACTIVE MODE.
AT THE END RETURN TO STRECHOPPER

1.== ENTER STRECHOPPER ==<
1.1==ENTER ROUTINE STRECHOPPER
1.1P NO CONDITIONS

2.== ITEM SELECTION ==<
2.1==FIELDS ARE EMPTY
2.1 IF NEWLY CREATED RECORD IN BUFFER
2.2==SELECT FIELDS TO BE EDITED
2.2 IF OLD RECORD IN BUFFER

3.== DATA ENTERING ==<
3.1==ENTER AND VERIFY SELECTIVE DATA
3.1 IF ITNO. = VALID ITNO. AND IF
NOT VERIFICATION MESSAGE IS SUPPRESSED
3.2==ERROR - ITEM OUT OF RANGE
3.2 IF NOT ITNO. = VALID ITNO.
3.3==ENTER SELECTIVE DATA
3.3 IF ITNO. = VALID ITNO. AND IF
VERIFICATION MESSAGE IS SUPPRESSED
3.4==ENTER ALL DATA ITEMS
3.4 IF NEWLY CREATED RECORD IN BUFFER

< 2 >
< 3, 3 >
< 0, 0, 0, 0 >

< 0 >
< 0, 0 >
SYSTALL A(11-14-5). = PERFORM PERSONAL ITEM CHANGE
ON THE STUDENT RECORD
= #STRECHOPPER WITH 7-ACTIONS
= SYSTALL A11-14-5-(1,7)

THIS ROUTINE IS CALLED BY #STRECHOPPER WHEN THE OPERATOR ENTERED COMMANDS
TO PERFORM MODIFICATIONS IN THE STUDENT RECORD'S PERSONAL AREA.

THIS IS DONE BY ITEM IN INTERACTIVE NODE.

AT THE END RETURN TO #STRECHOPPER

1.== ENTER #STRECHOPPER ==

1.1=ENTER ROUTINE #STRECHOPPER
1.1P NO CONDITIONS

A11-14-5-7> NO ACTION; NEXT CLUSTER

2.== ITEM SELECTION ==

2.1=FIELDS ARE EMPTY

2.1 IF NEWLY CREATED RECORD IN BUFFER

A11-14-5-7> NO ACTION; NEXT CLUSTER

2.2=SELECT FIELDS TO BE EDITED

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 ==

3.1=ENTER AND VERIFY SELECTIVE DATA

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 PERSONAL ITEMS

3.2=ERROR - ITEM OUT OF RANGE -

3.2 IF NOT ITNO. = VALID ITEM NO.

A11-14-5-6> WRITE ERROR MESSAGE

3.3=ENTER SELECTIVE DATA

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

3.4 IF NEWLY CREATED RECORD IN BUFFER

A11-14-5-4> ENTER DATA FOR ALL PERSONAL ITEMS FROM 1 TO 17
A11-14-5-5> DISPLAY ALL PERSONAL ITEMS
<table>
<thead>
<tr>
<th>Cluster</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>ENTER @STRECHOPEN</td>
</tr>
<tr>
<td>C2</td>
<td>DATA ENTERING</td>
</tr>
<tr>
<td>C3</td>
<td>ENTER @STRECHOPEN</td>
</tr>
<tr>
<td>C4</td>
<td>DATA ENTERING</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>OLD RECORD IN BUFFER</td>
</tr>
<tr>
<td>P2</td>
<td>NEWLY CREATED RECORD IN BUFFER</td>
</tr>
<tr>
<td>P3</td>
<td>TOD. = VALID ITEM NO.</td>
</tr>
<tr>
<td>P4</td>
<td>VERIFICATION MESSAGE IS SUPPRESSED</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>ENTER DATA FOR PERSONAL ITEMS FROM 1 TO 17</td>
</tr>
<tr>
<td>A2</td>
<td>ENTER COMMAND = CONNECT,ITNO.,ITNO.,...,ITNO.</td>
</tr>
<tr>
<td>A3</td>
<td>CHECK SYNTAX OF COMMAND LINE</td>
</tr>
<tr>
<td>A4</td>
<td>ENTER DATA FOR ITEMS ITNO., ITNO.,...,ITNO.</td>
</tr>
<tr>
<td>A5</td>
<td>DISPLAY ALL PERSONAL ITEMS</td>
</tr>
<tr>
<td>A6</td>
<td>WRITE ERROR MESSAGE</td>
</tr>
<tr>
<td>A7</td>
<td>NO ACTION, NEXT CLUSTER</td>
</tr>
</tbody>
</table>
SYSTALL A(11-14-6) = PERFORM COURSE CHANGES ON THE STUDENT RECORD
= OBTRECKDOR WITH 14 ACTIONS
= SYSTALL A(11-14-6)-(1..14)

THIS ROUTINE IS CALLED BY OBTRECKDOR WHEN THE OPERATOR ENTERS COMMAND TO MODIFY THE STUDENT RECORD'S COURSE REGISTRATION AREA. THIS ROUTINE PERFORMS COMPUTER-ASSISTED VERIFICATIONS AND COURSE CHANGES REFLECTING THE STUDENT'S REGISTRATION IN COURSE AND THE DATA IN THE COURSE MASTER FILE.

IT ALSO MODIFIES AUTOMATICALLY THE COURSE MASTER FILE AND THE CLASS MASTER FILE AND THE MISD-CHP FILE EACH TIME THERE IS A VALIDATED EXIT TO THE STUDENT'S COURSE REGISTRATION AREA. IN THIS WAY THE MASTER FILES ARE ALWAYS CONSISTENT.

AT THE END RETURN TO OBTRECKDOR

1. == ENTER OBTRECKDOR

1.1 == ENTER ROUTINE

2. == ACTION EVALUATION ON COMMAND VALIDITY

2.1 == NO MORE MODIFICATION RETURN

2.2 == ADD COURSE COMMAND LINE

2.3 == DROP COURSE COMMAND LINE

2.4 == ERROR -INVALID COMMAND LINE-

3. == ACTION EVALUATION ON COURSE VALIDITY

3.1 == ERROR -NON EXISTANT COURSE NO.-

3.2 == ERROR -STUDENT ALREADY REGISTERED IN COURSE-

3.3 == ERROR -NO MORE ROOM IN CLASS-

3.4 == ERROR -COURSE REQUEST HAS TIME CONFLICT WITH OTHERS

3.5 == ERROR -STUDENT IS NOT REGISTERED IN THIS COURSE-

4. == ERROR OVERRIDE

4.1 == ERROR OVERRIDE

4.2 == DONT OVERRIDE ERROR

CONTINUED ON THE NEXT PAGE
5. == AUTHORIZED COURSE MODIFICATION

  5.1 == ADD COURSES REQUESTED TO THE STUDENT RECORD

  5.2 == ADD COURSES REQUESTED TO THE #STUD.REC. AND VERIFY

  5.3 == DROP COURSES REQUESTED FROM THE #STUD.REC.

  5.4 == DROP COURSES REQUESTED FROM THE #STUD.REC. AND VERIFY

6. == FINALIZATION OR ERROR RECOVERY

  6.1 == END OF COURSE MODIFICATIONS

  6.2 == FATAL ERROR RECOVERY
8. AML-ROUTINE STRECHOCR

SYSTALL A1(11-14-6) = PERFORM COURSE CHANGES ON THE STUDENT RECORD
= STRECHOCR WITH 14 ACTIONS
= SYSTALL A11-14-6-(1..14)

THIS ROUTINE IS CALLED BY STRECHOCR WHEN THE OPERATOR ENTERS COMMANDS TO
MODIFY THE STUDENT RECORD'S COURSE REGISTRATION. 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 6505-CARD FILE EACH TIME THERE IS A VALIDATED EDIT
TO THE STUDENT'S COURSE REGISTRATION AREA. IN THIS WAY THE MASTER
FILES ARE ALWAYS CONSISTENT.

AT THE END RETURN TO STRECHOCR

1.== ENTER STRECHOCR

1.1== ENTER ROUTINE
1.1P NO CONDITIONS

2.== ACTION EVALUATION ON COMMAND VALIDITY

2.1== NO MORE MODIFICATION: RETURN
2.1 IF NOT COURSE TO BE ADDED AND IF
NOT COURSE TO BE DROPPED

2.2== ADD COURSE COMMAND LINE
2.2 IF COURSE TO BE ADDED AND IF
VALID COMMAND

2.3== DROP COURSE COMMAND LINE
2.3 IF COURSE TO BE DROPPED AND IF
VALID COMMAND

2.4== ERROR - INVALID COMMAND LINE-
2.4 IF NOT VALID COMMAND

3.== ACTION EVALUATION ON COURSE VALIDITY

3.1== ERROR - NON EXISTANT COURSE NO.-
3.1 IF NOT COURSE (COND.) IS IN COURSE MASTER FILE

3.2== ERROR - STUDENT ALREADY REGISTED IN COURSE-
3.2 IF COURSE TO BE ADDED AND IF
STUDENT IS REGISTED IN COURSE (COND.)

3.3== ERROR - NO MORE ROOM IN CLASS-
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 THE CONFLICT WITH OTHERS
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
3.5 IF COURSE TO BE DROPPED AND IF NOT STUDENT IS REGISTERED IN COURSE (CRAWD.)

4. = ERROR OVERRIDE
4.1 = ERROR OVERRIDE
4.1 IF OVERRIDE ERROR

4.2 = WANT OVERRIDE ERROR
4.2 IF NOT OVERRIDE ERROR

5. = AUTHORIZED COURSE MODIFICATION
5.1 = ADD COURSES REQUESTED TO THE STUDENT RECORD
5.1 IF COURSE TO BE ADDED AND IF VERIFICATION MESSAGES SUPPRESSED

5.2 = ADD COURSES REQUESTED TO THE STUDENT RECORD AND VERIFY
5.2 IF COURSE TO BE ADDED AND IF NOT VERIFICATION MESSAGES SUPPRESSED

5.3 = DROP COURSES REQUESTED FROM THE STUDENT RECORD
5.3 IF COURSE TO BE DROPPED AND IF VERIFICATION MESSAGES SUPPRESSED

5.4 = DROP COURSES REQUESTED FROM THE STUDENT RECORD AND VERIFY
5.4 IF COURSE TO BE DROPPED AND IF NOT VERIFICATION MESSAGES SUPPRESSED

6. = FINALIZATION OR ERROR RECOVERY
6.1 = END OF COURSE MODIFICATIONS
6.1 IF NOT FATAL ERROR INTRODUCED IN STUD.RECORD

6.2 = FATAL ERROR RECOVERY
6.2 IF FATAL ERROR INTRODUCED IN STUD.RECORD
SYSTALL A(11-14-6) = PERFORM COURSE CHANGES ON THE STUDENT RECORD
  = SYSTRECORD WITH 1A ACTIONS
  = SYSTALL A(11-14-6) (1..14)

THIS ROUTINE IS CALLED BY SYSTRECORD WHEN THE OPERATOR ENTERS COMMANDS TO MODIFY THE STUDENT RECORD'S COURSE REGISTRATION AREA. THIS ROUTINE PERFORMS COMPUTER-AIDED 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 CARD-CARD FILE EACH TIME THERE IS A VALIDATED EDIT TO THE STUDENT'S COURSE REGISTRATION AREA. IN THIS WAY THE MASTER FILES ARE ALWAYS CONSISTENT.

AT THE END RETURN TO SYSTRECORD.

1.== ENTER SYSTRECORD

  1.1== ENTER ROUTINE
  1.1P NO CONDITIONS
        A(11-14-6-14) NO ACTION; NEXT CLUSTER
  1.2== ACTION EVALUATION COMMAND VALIDITY
        2.1 NO MORE MODIFICATION; RETURN
        2.1 IF NOT COURSE TO BE ADDED AND IF NOT COURSE TO BE DROPPED
        A(11-14-6-14) NO ACTION; NEXT CLUSTER

  2.2== ADD COURSE COMMAND LINE
        2.2 IF COURSE TO BE ADDED AND IF VALID COMMAND
        A(11-14-6-11) ENTER COMMAND = ADD;CRNO;CRNO;.....CRNO
        A(11-14-6-1) CHECK SYNTAX OF COMMAND LINE

  2.3== DROP COURSE COMMAND LINE
        2.3 IF COURSE TO BE DROPPED AND IF VALID COMMAND
        A(11-14-6-2) ENTER COMMAND = DROP;CRNO;CRNO;.....CRNO
        A(11-14-6-1) CHECK SYNTAX OF COMMAND LINE

  2.4== ERROR -INVALID COMMAND LINE-
        2.4 IF NOT VALID COMMAND
        A(11-14-6-3) ENTER COMMAND = ELSE
        A(11-14-6-1) CHECK SYNTAX OF COMMAND LINE
        A(11-14-6-11) WRITE APPROPRIATE ERROR MESSAGE

CONTINUED ON THE NEXT PAGE
3. ACTION EVALUATION ON COURSE VALIDITY

3.1 =ERROR - NON EXISTANT COURSE NO. =
3.1 IF NOT COURSE (CMD) IS IN COURSE MASTER FILE
   All-14-6-11> WRITE APPROPRIATE ERROR MESSAGE
   THEN-GO-TO-CLUSTER> 2

3.2 =ERROR - STUDENT ALREADY REGISTERED IN COURSE=
3.2 IF COURSE TO BE AMENDED AND IF
   STUDENT IS REGISTERED IN COURSE (CMD)
   All-14-4-11> WRITE APPROPRIATE ERROR MESSAGE
   THEN-GO-TO-CLUSTER> 2

3.3 =ERROR - NO MORE ROOM IN CLASS=
3.3 IF COURSE TO BE AMENDED AND IF
   NOT PLACE LEFT IN CLASS FOR NEW REGISTRATION
   All-14-6-11> WRITE APPROPRIATE ERROR MESSAGE
   All-14-6-10> REQUEST OPERATOR WHETHER TO OVERRIDE ERROR
   THEN-GO-TO-CLUSTER> 4

3.4 =ERROR - COURSE REQUEST HAS TIME CONFLICT WITH OTHERS=
3.4 IF COURSE TO BE AMENDED AND IF
   COURSE SCHEDULE CONFLICT WITH OTHER ALREADY REGISTERED COURSES
   All-14-6-11> WRITE APPROPRIATE ERROR MESSAGE
   All-14-6-10> REQUEST OPERATOR WHETHER TO OVERRIDE ERROR
   THEN-GO-TO-CLUSTER> 4

3.5 =ERROR - STUDENT IS NOT REGISTERED IN THIS COURSE=
3.5 IF COURSE TO BE AMENDED AND IF
   NOT STUDENT IS REGISTERED IN COURSE (CMD)
   All-14-4-11> WRITE APPROPRIATE ERROR MESSAGE
   THEN-GO-TO-CLUSTER> 2

47 = ERROR OVERRIDE
4.1 =ERROR OVERRIDE=
4.1 IF OVERRIDE ERROR
   All-14-6-10> ENTER COMMAND = GO
   THEN-GO-TO-CLUSTER> 5

4.2 = DONT OVERRIDE ERROR=
4.2 IF NOT OVERRIDE ERROR
   All-14-6-10> ENTER COMMAND = NOGO
   THEN-GO-TO-CLUSTER> 2

CONTINUED ON THE NEXT PAGE
5. == AUTHORIZED COURSE MODIFICATION

5.1 == ADD COURSES REQUESTED TO THE STUDENT RECORD

5.1 IF COURSE TO BE ADDED AND IF
VERIFICATION MESSAGES SUPPRESSED
A11-14-6-5> ADD COURSES (CMD.) TO #STUD.REC. IN BUFFER

5.2 == ADD COURSES REQUESTED TO THE STUDENT.REC. AND VERIFY

5.2 IF COURSE TO BE ADDED AND IF
NOT VERIFICATION MESSAGES SUPPRESSED
A11-14-6-5> ADD COURSES (CMD.) TO #STUD.REC. IN BUFFER
A11-14-6-7> DISPLAY STUDENTS COURSES IN BUFFER

5.3 == DROP COURSES REQUESTED FROM THE STUDENT.REC.

5.3 IF COURSE TO BE DROPPED AND IF
VERIFICATION MESSAGES SUPPRESSED
A11-14-6-6> DROP COURSES (CMD.) FROM #STUD.REC. IN BUFFER

5.4 == DROP COURSES REQUESTED FROM THE STUDENT.REC. AND VERIFY

5.4 IF COURSE TO BE DROPPED AND IF
NOT VERIFICATION MESSAGES SUPPRESSED
A11-14-6-6> DROP COURSES (CMD.) FROM #STUD.REC. IN BUFFER
A11-14-6-7> DISPLAY STUDENTS COURSES IN BUFFER

6. == FINALIZATION OR ERROR RECOVERY

6.1 == END OF COURSE MODIFICATIONS
6.1 IF NOT FATAL ERROR INTRODUCED IN STUD.RECORD
A11-14-6-14> NO ACTION, NEXT CLUSTER

6.2 == FATAL ERROR RECOVERY
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 STRECDCR

SYSTAL A(11-14-6) = PERFORM COURSE CHANGES ON THE STUDENT RECORD

= #STRECDCR WITH 14 ACTIONS

= SYSTAL A(11-14-6)-1-(1,14)

This routine is called by #STRECDCR 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 master card file. Each time there is a validated edit to the student's course registration area, in this way the master files are always consistent.

At the end return to #STRECDR.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clusters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td></td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Predicates

- **P1** - N Y - - Y Y Y - - Y Y - -
- **P2** - N - Y - - - - - - Y Y - -
- **P3** - Y Y M - - - - Y Y - -
- **P4** - Y - Y - - N - - - -
- **P5** - Y - N - - - - M - -
- **P6** - Y - Y M - - - - - -
- **P7** - Y - Y N M - - - -
- **P8** - Y Y M - - - - - -
- **P9** - Y Y M - - - - - -
- **P10** - Y Y M - - - - - -

### Clusters

- **C1** - Enter Streedcr
- **C2** - Action Evaluation and Command Validity
- **C3** - Action Evaluation on Course Validity
- **C4** - Error Override
- **C5** - Authorized Course Modification
- **C6** - Finalization or Error Recovery

### Predicates

- **P1** - Course to Be Added
- **P2** - Course to Be Dropped
- **P3** - Valid Command
- **P4** - Course (CRNO.) Is in Course Master File
- **P5** - Student Is Registered in Course (CRNO.)
- **P6** - Place Left in Class for New Registration
- **P7** - Course Schedule Conflict with Other Already Registered Courses
- **P8** - Overrun Error
- **P9** - Verification Messages Suppressed
- **P10** - Fatal Error Introduced in Stud.Record

### Actions

- **A1** - Enter Command = Add (CRNO.,CRNO.,CRNO.)
- **A2** - Enter Command = Drop (CRNO.,CRNO.,CRNO.)
- **A3** - Enter Command = Else
- **A4** - Check Syntax of Command Line
- **A5** - Add Courses (CRNO.) to #STUD.REC. IN BUFFER
- **A6** - Display Courses (CRNO.) FROM #STUD.REC. IN BUFFER
- **A7** - Display Students Courses in Buffer
- **A8** - Enter Command = Go
- **A9** - Enter Command = No Go
- **A10** - Request Operator Whether to Override Error
- **A11** - Write Appropriate Error Message
- **A12** - Enter Command = Clear
- **A13** - Restore #STUD.REC. Course Content to Its Original Values
- **A14** - No Action, Next Cluster
6. ABL-ROUTINE STRECEL

SYSTALL ACTION(11-15) = A11-15 = STUDENT RECORD DELETION ROUTINE

= STRECEL 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-IMAGE MASTER FILES.

AT THE END RETURN TO #STUPDATE

1. === ENTER #STRECEL ROUTINE ===

1.1 === VERIFY STUDENT TO BE DELETED

2. === DELETION ===

2.1 === VERIFIED DELETION

2.2 === DELETION RECOVERY

< 2 >

< 0; 0 >

< 0 >

< 0 >
6. AML-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-UPDATE 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

1.1 IF NO CONDITIONS

2.=== DELETION ===$ < 0, 0 >

2.1===VERIFIED DELETION

2.1 IF DISPLAYED STUDENT IS REALLY TO BE DELETED

2.2===DELETION RECOVERY

2.2 IF NOT, DISPLAYED STUDENT IS REALLY TO BE DELETED
SYSTALL ACTION(11-15) = A11-15> = STUDENT RECORD DELETION ROUTINE
= #STRECEL 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 #STRECEL ROUTINE ==$< 2>
   1.1=VERIFY STUDENT TO BE DELETED
       1.1P NO CONDITIONS

       A11-15-1> DISPLAY STUDENT NAME AND ID.NO. FOR VERIFICATION

2.== DELETION ==$< 0, 0>
   2.1=VERIFIED DELETION
       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 ENROLLMENT STATISTICS ON RELATED COURSE MASTER FILE RECORDS

       2.2=DELETION RECOVERY
       2.2 IF NOT DISPLAYED STUDENT IS REALLY TO BE DELETED

       A11-15-3> ENTER COMMAND = ELSE
6. AML-TRANSFORM STRECDEL

SYSTALL ACTION(11-15) = A11-15 = STUDENT RECORD DELETION ROUTINE

= #STRECDEL WITH 6 ACTIONS
= SYSTALL A11-15>(1..6)

This routine is a 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 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 = NO
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 ENROLLMENT 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
MODELING

(M-4-P-6)

18

*****************************************************************************
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 SUBFUNCTION 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 RAN OF THE SYSTEM
3 # == DATA UPDATE ORGANIZATION
4 # == DATA-BASE QUERY ORGANIZATION

20 2B
1/11#2.3.4.5/11
1/167-9/2.3.4/1
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#18.19.30#25.31.32#23.24.25.26.37.39.40.41#42.3B.16
# == SET UP PSEUDONAM 1# OR QUERY POSSIBILITY
1/26#27#28.29#42.43.16.17.18.19.20.1
1/30-1.2.3.12.21.22.23.24.1
1/64#44/2
1/33/25/2
1/34#26#27.26.2
2/35#36.2.3.7.30.27/2
2/38.39.40.41.42.28.2
2/39#29/2
2/44.45.46.47.48.30.2
2/64/44/3
3/15.12#49.50.10.3
3/15#12.51.50.11.3
3/15.1#2.52.53.54.55.35.31.32.3
3/164#44/4
4/13.49#56.21.22.33/4
4/13.49#2.51.52.58.59.31.22.23.34/4
4/13.49#2.52.57.40.21.22.33/4
4/56.61.49.21.22.36.37.38.8/4
4/56.62.49.21.22.39.40.38/4
4/56.60.63.49.21.22.41.42.38.8
4/64#44/0

#
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(' QUERY 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-TRUNCATION 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-ATTS. RELATED DATA ')
 20: WRITE(' CHANGE INVOLVES ATTS. RELATED DATA ')
 21: WRITE(' TASK IS QUERty 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 QUERy PROGRAM ')
  8: WRITE('A8> SET UP STUDENT MASTER FILE QUERy PROGRAM ')
  9: WRITE('A9> SET UP CLASS MASTER FILE QUERy 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 ATTS. 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> QUERY COURSE MASTER FILE INTERACTIVELY ')
 34: WRITE('A34> QUERY STUDENT MASTER FILE INTERACTIVELY ')
 35: WRITE('A35> QUERY CLASS MASTER FILE INTERACTIVELY ')
 36: WRITE('A36> WRITE A FILE IN COURSE MASTER FILE QUERy LANGUAGE ')
 37: WRITE('A37> QUERY COURSE MASTER FILE PSEUDOBATCH MODE ')
 38: WRITE('A38> PRINT SPOOLED OUTPUT ')
 39: WRITE('A39> WRITE A FILE IN STUDENT MASTER FILE QUERy LANGUAGE ')
 40: WRITE('A40> QUERY STUDENT MASTER FILE PSEUDOBATCH MODE ')
 41: WRITE('A41> WRITE A FILE IN CLASS MASTER FILE QUERy LANGUAGE ')
 42: WRITE('A42> QUERY CLASS MASTER FILE PSEUDOBATCH MODE ')
 43: WRITE('A43> PRODUCE ABS-CARDS FOR THE CLASS ')
 44: WRITE('A44> NO ACTION- NEXT CLUSTER ')
END
END (* SYSTALL *)
POOR COPY
COPIE DE QUALITEE INFERIEURE

SYSTALL (H-41-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)
AND ITS SUBSYSTEM #DBU) 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 HIERARCHICAL RELATIONSHIP BETWEEN THE SUB-ACTION AND
THE ACTIONS OF SYSTALL THE FOLLOWING NOTATION WILL BE USED:

SYSTALL ACTION(N) = ACTION(N) = AN-XXXXXXXX
SYSTALL SUB-ACTION(M) OF ACTION(N) =
SYSTALL ACTION(N-M) = AN-M-XXXXXXXX
SYSTALL SUB-SUB-ACTION(L) OF SUBACTION(M) OF ACTION(N) =
SYSTALL ACTION(N-M-L) = AN-M-L-XXXXXXXX

* 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

**********************************************************************

6-29-24

1C1 SET UP THE BASIC SYSTEM (# UTS. AND # DB.)
2C2 INITIAL DATA-BASE VERIFICATION (# UGS. AND # UPS.)
3C3 SET UP THE ATTENDANCE MONITORING SUBSYSTEM (# UTS.)
5C5 MAINTAIN THE EXACTNESS OF THE SYSTEM DATA (# UPS.)
6C6 INFORMATION EXTRACTING AND REPORTING (# UGS.)

13 24
1/1-1/1/1
1/1-2/23.4/2
1/2-3/5.8-0.9/81
1/2-3/12.10-11/12/2
2/2-3/17.13.40-14.41/15/2
2/5-6/10.11/12/3
2/5-6/147/3
3/7-0.14.40/15.16/17.18/19.20/3
3/7-0.14.24.4316/17.18/19.20/3
4/10-7.11.17.12/24/4
4/10-7.11.17.25/4
4/10-11.12.26/4
4/10-11.13.27/4
4/10-11.14.28/4
4/10-11.15.29/4
4/10-11.16.17.30/20/4
4/10-11.25/4
4/10-11.26/4
5/1-10.10.11/12/4
5/1-19.10.11.12/3/31.32/4
5/1-19.10.11/12/3
5/1-19.4/4
6/21-23.33/4
6/21-22.25.34/4
6/21-22.25.35/4
6/21-24.36.37.38/4
6/21-24.41.42.38/4
6/21-24.27/4/0
> 7

INSTALL TERMINAL IN SCHOOL
SET UP EMPTY DATA-BASE
SET UP ALL #UGS. AND #UPS. INTERACTIVE PROGRAMS
UPDATE ALLOCATED FIELDS IN DATA-BASE
VERIFY INFORMATION IN DATA-BASE
UPDATE INCORRECT INFORMATION IN DATA-BASE
NO UPDATE IS NECESSARY
GIVE OUT ACCESSORIES FOR ATTEND.MONIT.SUBSYSTEM.
SET UP #ATS. INTERACT. PROGS. FOR DAILY PRODUCTION LIST
#ATS. DAILY DATA USAGE PARAMETER DEFINITION
#ATS. DAILY TRANSACTION DATA GENERATION
#ATS. DAILY TRANSACTION DATA COLLECTION
#ATS. DAILY TRANSACTION DATA PROCESSING
#ATS. DAILY DATA DISTRIBUTION
#ATS. DAILY OUTPUT USAGE
NO ACTION+NEXT CLUSTER
UPDATE NON ATTENDANCE RELATED DATA
UPDATE ATTENDANCE RELATED DATA
NO ACTION+ NEXT CLUSTER
QUERY COURSE MASTER FILE INTERACTIVELY
QUERY STUDENT MASTER FILE INTERACTIVELY
QUERY CLASS MASTER FILE INTERACTIVELY
QUERY COURSE MASTER FILE PSEUDOBATCH
QUERY STUDENT MASTER FILE PSEUDOBATCH
QUERY CLASS MASTER FILE PSEUDOBATCH
END OF THE YEAR