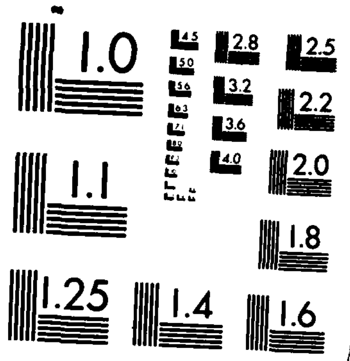
AD-A164 051 IMPLEMENTATION OF THE AFIT/ENG FACULTY AND STUDENT DATABASE MANAGEMENT SYSTEM(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING 1/3

UNCLASSIFIED D A GAITROS DEC 85 AFIT/GCS/ENG/85D-5 F/G 9/2 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

1

AD-A164 051

IMPLEMENTATION OF THE AFIT/ENG FACULTY AND STUDENT DATABASE MANAGEMENT SYSTEM

THESIS

David Alan Gaitros
Captain, USAF

AFIT/GCS/ENG/85D-5

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

86 2 12 033

1

AFIT/GCS/ENG/85D

IMPLEMENTATION OF THE AFIT/ENG FACULTY AND STUDENT DATABASE MANAGEMENT SYSTEM

THESIS

David Alan Gaitros
Captain, USAF

AFIT/GCS/ENG/85D-5

AFIT/GCS/ENG/85D-5

IMPLEMENTATION OF THE AFIT/ENG FACULTY AND STUDENT DATABASE MANAGEMENT SYSTEM

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science

David A. Gaitros

Captain, USAF

December 1985

Preface

The AFIT/ENG database has the ability to store and manipulate student and faculty data utilizing the TOTAL Database Management System and the Forms Management System. This study was undertaken to design the complex network of application programs needed to maintain the database system. Additionally, a library of standard routines were created to ease the amount of code needed to create the programs and aid in the maintenance of the system. One of the programs was implemented and tested to validate the design and to serve as an example of how the standard routines were implemented.

I would like to thank my advisor, Dr Gary B. Lamont for his time, expertise, and confidence in my abilities. Additionally, I would like to thank Robert Ewing and Captain Steve Woffinden for their assistance in guiding me through the experience of writing a thesis. Finally, I would like to express my sincere gratitude to my wife, Cindy, for her never ending faith and encouragement through this assignment and the past nine years.

Table of Contents

List of Figures

AFIT/GCS/85D-5

Abstract

This study took the works of the previous AFIT/ENG Student and Faculty Database system thesis efforts and design and implemented the application software for the project. The basic purpose of the thesis was to provide a sound design for the application programs that would interface with the TOTAL Database Management System and the Forms Management System. The entire system was to be designed with the notion that it would be modified and enhanced. A series of standard interface routines were created to act as a layer between the TOTAL DBMS. The resulting routines were abstracted and used as an extension to the Pascal programming language.

The education plan portion of the database was used as a prototype to develop the requirements of the human-computer interface. The program was then redesigned and implemented using the standard routines and the specifications developed from the prototype. A menu driven system was used to implement the design utilizing the Forms Management System as the screen interface. The education plan program is an example of the structured approach used in interpreting the design of the database system. The program contains examples of scrolled screens, database calls, linked list routines, and data abstraction. Additional programs were written to demonstrate the capabilities interfacing with the GKS graphics package, transmition of data to the registrars office, and to show the continuity of the design.

Chapter 1

I. Introduction

Background

A database management system is a computer based system whose overall purpose is to record and retrieve information. (1:3). Database management systems offer a convenient and practical means to store, manipulate, and retrieve information. There are three basic types of database systems: relational, hierarchial, and network(1:450). The relational database represents the information as a series of tables. The hierarchial database represents information as a tree structure with some data being subordinate to other data or records. The network database is by far the most complex because it involves linking related pieces of information together to avoid duplication. This structure provides the user of the database a great deal of flexibility in storing and retreiving large amounts of data. The Air Force Institute of Technololgy (AFIT) School Engineering, Department of Electrical and Computer Engineering has a need to maintain information on students, faculty, courses, and class rooms. Several proposals for an AFIT/ENG database have been submitted and a prototype is under construction in the Department of Electrical and Computer Engineering, Information Sciences Laboratory. This project is under the supervision of Dr. Gary B. Lamont and Engineer Robert Ewing and is designed to provide students with a valuable learning tool and a service to the School of Engineering. The overall purpose of this thesis project is to establish a working database manangement system for the Department of Electrical and Computer Engineering, AFIT.

Three AFIT/EN thesis written by Jeffrey S. Ricks, Robert S. Colburn, Dean S. Alfred, and Myron E. Pangman have proposed a database management system in detail using the VAX 11/780 (VMS Operating System) and a network database system called TOTAL. These theses describe in detail the file types, display forms, network schemes, dataset selections, data description language, and anticipated application program design as well as the feasibility of such a project. The design of the database was approached as a solution to the demands of the Electrical and Computer Engineering Department's request to store and manipulate more information. The design was intended to create a "user-friendly" information center using the database system TOTAL(16), a forms management utility called FMS(10), and an application programming language such as FORTRAN or PASCAL.

This proposed database management system is being used in classroom projects in EE 6.46 (Computer Database Systems), and DBTA efforts. This has given students the opportunity to work on a real database system, write applications programs, and use schemas while studying DBMS concepts. During these courses several modifications and enhancements were suggested to improve the efficiency and make the system more "user-friendly".

Statement of Problem

The purpose of this thesis investigation is to analyze previous thesis effort design and implementation, the work done by the students of previous EE 6.46 classes, and indentify modifications and enhancements. The modifications to the schemas and application software will be reviewed and incorporate in the system. A formal design phase will be conducted to reflect modifications to the new schema and enhancements. The application programs developed as class projects will be reviewed for further ideas to aid in the design of a complete software package to implement the database structure. A study will be conducted to test the feasibility of linking the AFIT/EN Database Management System with the AFIT scheduling system and integrating a backup and restore capability into the design. The system will be designed to generate the Division Facutly Schedule and Manpower Requirement and Expenditure document as prepared by the School of Engineering. This feature will take the instructors hours spent on teaching classes, conducting short courses, thesis students, and PHD students and generate statistics based on this data. Requirements for generating a graphical representation of the data for managment information needs will be conducted. The overal design will include the use of a commercial form display technique called FMS that is currently available on the ISL VAX 11/780 (VMS Operating System).

Scope

The purpose of this thesis effort is to identify existing problems with the database system and concentrate on the design of the system. The study will confine the use of the database system to the AFIT/ENG Department although recognizing the need for an all-inclusive AFIT database (2:1-3) This effort will provide a solid base for further program development and implementation by enhancing the current database schema, analyzing current user needs, and providing a detailed design analysis of the overall database system. Software design for this project will consider time and space requirements into consideration for algorithm design. In addition, a portion of the design will be implemented for use by the faculty and students.

Assumptions

No attempt will be made to justify the need for an AFIT/EN database system nor will any attempt be made to analyze the need to apply another type of database such as a relational or hierarchial. It's assumed that these topics were discussed in sufficient detail in previous thesis efforts (2).

Summary of Current Knowledge

The proposed AFIT Database (2) currently contains information and schema definitions on faculty, departments, students, thesis, section leaders, school courses, school quarters, course text books, class times, room capacities, schedules, master degree requirements, and course sequences. (See Appendix A) Several enhancements to the schema have been proposed by students in the EE 6.46 class while developing application programs for the AFIT Database System. Some of these enhancements are a direct result of the changing requirements for a database system and others stem from the introduction of new or improved software and hardware capabilities in the Information Sciences Laboratory.

Standards and Notations

The time analysis for algorithms will use the notation O(n) which transalates to big-oh of n (3:21-23). This indicates that the algorithm takes an order of time to execute which is dependent upon the number of inputs designated by "n". Space analysis of an algorithm will also be in terms of the number of inputs designated by "n".

Structure charts developed by Stevens, Myers, and Constatine (4:60) will be used to represent the modular characteristics of the applications programs. These charts will be used because of the highly modular and structured design of the proposed system. The Software Development Workbench (SDW) will be used as the automated tool for designing and documenting the development of the system. Sructure charts and diagrams will be drawn on the system flowchart portion of the graphics editor while the SADT charts will be drawn on the AUTOIDEF (8:173) of the Requirements Definitions portion of the Software Development Workbench (SDW). The use of the SDW is intended to decrease the number of errors in the software and to allow easy changes to the preliminary design of the sytem. Documentation standards will conform to the AFIT/ENG Development Documentation Guidelines Draft #2 and Standards document published September 26, 1984.

Approach

The project will consist of the following phases.

1) A preliminary evaluation of the current AFIT/EN Database will be conducted to identify deficiencies and weaknesses in the schema. Reevaluation of the current requirements will be necessary because of the time from the last research done on this

topic (1983).

2) Interviews and research will be conducted to find any additional errors that have been found by previous EE 6.46 classes in their efforts to write software to interface with the database. Enhancements to the database schema will be included with this phase to accomodate new requirements for information.

3) A new database schema will be completed and compiled to reflect the changes made as a result of the investigation.

4) Input and output standards to the new database will be completed to establish guidelines for future efforts. An updated version of the Frames Management System (FMS Version 2.1) will be used as the input/output media. The objective will be to make a self documenting system, easing the burden on the user by using a menu driven system with query capability. An alternative system could use the lower level modules to develop a command driven system for the expert users and on remote terminals with limited graphic capability.

5) An initial design of the system will follow the top-down structured design techniques with emphasis placed on efficiency of the program modules. At the same time, a bottom up development will be conducted to design the abstract routines needed to interface with the TOTAL DBMS. This will be done to provide an abstraction between the programmer and the database system and to facilitate the coding phase of the development cycle.

6) A study will be conducted to address the feasibility of using a graphics package to generate graphs of managment

information compiled by the DBMS. Examples of this would include the representaion of the Division Faculty Schedule and Manpower Requirement Expenditure Document as a bar chart of faculty and division manpower utilization.

7) A portion of the design will be implemented to further establish programming and documentation standards and to demonstrate the usefulness of the database system to the faculty. The majority of the development will be in the main module programs and the bottom layer utility programs.

Materials and Equipment

Implementation and testing will be done on the ISL VAX 11/780 computer system with the VMS operating system, and application programs. Enough disk memory space will be required to hold a test database and associated utililty software. A nine-track tape drive system will be required to backup software, data, and documentation in case of system failure. A work station for this project is required and will include a desk and a computer terminal with connection to the VAX 11/780 located in the Information Sciences Laboratory (ISL).

Other Support

The cooperation of the faculty and students in gathering current data to test the database is required as well as the software developed in past EE 6.46 efforts to aid in the design of the database system. These efforts will aid in identifying past deficiencies and contribute to the accuracy of the testing of specific algorithms. The Software Development Workbench (SDW) will be used as the automated design tool to aid in the design phase and to test its application to software design phase.

Chapter 2

II. General Approach to Requirements Definition

This chapter will cover the functional requirements for a student and faculty database system. This section of the thesis will deal directly with the targeted users of the proposed system in trying to establish the requirements for the system. Previous work done by Allred (12), Ricks and Colburn (13), and Pangman (2) were to define the AFIT/ENG database and to implement the database schema. These theses efforts focused on the initial requirements for information to be contained in a database for The AFIT School of Engineering. In the software development life cycle (4:13), the steps through the requirements definition have been reached on the database schema design. As discussed in chapter 1, the design of the application software has been largely ignored. A review of the initial concepts and requirements of the database schema will be conducted to insure that the current configuration supports these requirements. In addition, requirements for the application software will be defined in this chapter. The software requirements definition phase for this software design will be divided into two parts, the functional requirements and the user interface requirments.

The first part is the functional requirements of the system and defines what the system must be able to provide to the using organization and design how it will be done. The topics to be covered are modular grouping of the software, standard database functions (ADD, UPDATE, DELETE, REVIEW), the limitations placed on the system, response time, memory requirements, graphic capabilities, required reports, and data syntax and compatibility checks.

The second part of the requirements definition phase is defining the "human computer interface" of the system. This section will describe the design of the interfaces to the system. The success of the design effort depends a great deal on how effectively the methodology externalizes issues with each group in the human interface (4:202). In this section a targeted group of users will be defined so a profile of the "average user" can be defined. In the requirement definition phase, interaction between the analyst and the user is very important to the success of any design. One of the most common reasons systems fail is because the definition of system requirements is inadequate (14:139). Assumptions on the experience level of the user are necessary in developing the requirements for the interface portion of the software. A prototype database application program of the education plan (EDPLAN) system will be constructed to further define the requirements of the user. By experimenting with this program and gaining user feedback the interface and functional requirements can be further refined. This will also give the eventual users of the system a hand in the developement which should make implementation easier. A complete requirements list developed this and previous thesis efforts can be found in Appendix F and will be referenced thourghout this chapter.

The users seem to be more satisfied with a qualitative definition which, in many cases, specifies the system in generalities and in terms of benefits to be derived. To reach the level of detail that the designer wants, the users must actually enter what they consider to be problem solving, or design (how) mode: they must arrive at functions that will solve their problems. Designers of systems are usually thinking a step ahead of the user (14:140). Table 2-1 (extracted form the article Pinpoint Requirments (14:140)) shows the analyst or designer views as apposed to the users views.

THE OBJECTIVE OF REQUIREMENTS DEFINITION

Objective: To define what the system will do:

ANALYST/DESIGNER	USER
Functional definition	Qualitative definition
Precise	Interpretation to be
Complete	All request to be met
Frozen	Flexible definition
Definition produce within allotted time	Definition ongoing process
Resulting system implemented within schedule and budject	Favorable impact of system on departmental budgets
Good system	System will work

Figure 2-1 Objective of Requirements Definition

Database Schema Review and Modification

The current database schema was reviewed during the development work in the fall session of the computer database class EE 6.46. Several enhancements and modifications were recommended by the classes during that session. It was very important to establish the database schema early in the requirments phase in order to provide a solid base by which to design the application software. The schema is important because it defines the relationships of data and records to each other and also defines the very essence of the database system. The modifications discussed below are a direct result from an early investigation into the database schema (Requirement 1, Appendix F). The first real users of the new database were the new Engineering and Computer Science students entering the summer 1985 quarter.

Several of the network relationships within the database posed a problem with the current configuration. The first problem was found in the Master Degree Requirements File(Requirement 1.1, Appendix F). This file maintained the different graduate degree programs and the minimum requirements needed for graduation. Within this file was a link to a students social security number. The file could not be created unless a students social security number was assigned to that field. Previous thesis efforts were searched to determine what purpose the field served. After finding no valid purpose, the field was deleted from the schema.

There exists within the originial schema two files that hold text book information. The Master Text Book File holds information on required text books for a particular class, the Master Book File holds information on books associated with a class but on a more detailed manner. Since the Master Book File was also connected via links to AFIT courses, a duplication of information was indicated. The Master Text Book File was deleted from the schema and all links to other master and variable files were removed(Requirement 1.2, Appendix F).

The other major change to the database took place late in the 1985 spring quarter. The field length of the class field was originally 6 and took the form "EE45Ø ". This configuration allowed for a suffix to indicate a lab or special sessions of a class. To allow for compatibility with other database systems, the field was lengthened to 8 characters and now takes the form "EENG45Ø "(Requirement 1.3, Appendix F).

During the course of developing low level interface modules to the TOTAL DBMS, it was discovered that the faculty or student files could not be deleted without destroying student thesis information. Since the thesis information is usually kept for research this presented a problem. At the same time, duplicate information was found in the Master Thesis Catalog (NTIC). To solve the problem, the thesis information was moved from the variable file to the Thesis Master File (THES) and the Thesis catalog file should be removed from the schema (Requirement 1.4, Appendix F).

Functional Requirements

Functional File Grouping

The file structure of the AFIT/EN Database (Appendix A), as proposed has 16 master files and 21 variable files but all of these files can be group into the following categories:

1. Faculty : The files that belong in this group are the Faculty Master file, and the Master Department File and all related variable files.

2. Student : The files that belong in this group are the Student Master file, the Master Section Number file, and the associated variable files.

3. Course/Room: The files that belong in this group are the Master Course File, Master Quarter file, Master Book File,

Master Order File, Master Class Time File, Master Building/Room File, Master Room Capacity File, Master Day Scheduling File, and associated variable files.

4. Thesis: The file that belong in the Thesis grouping are the Master Thesis Catalog Number File, the Master Thesis Number File, and associated variable files.

5. Sequence/Degree Requirements: This grouping contains the Master Sequence File, the Master Degree Requirements File, and associated variable files.

This grouping is necessary to allow for update priviledges to related files within a module and to reduce the amount of code required to operate any application program. A complete description of the master files and associated variable files can be found in Appendix A of this document. Therefore, the requirement exist that the database system be in a modular and structured form keeping related functions in seperate modules. The structure of the database file system allows for this. The following requirements exist for the overall system to conform to a structured design, ease of maintenance, and to to insure database integrety and security (Requirement 2.1, Appendix F):

1. Each functional group of files must have a standard set of functions associated with each file and variable file if one is required. These function must include the ability to add records or information, delete records or information, update database information, and the ability to review the information stored in the files. All information within a database is eligible for update except the items used as keys. If a key is

found to be in error then the entire record must be deleted and then added again to maintain the integrity of the database.

2. Each functional group of files must be maintained by a seperately compiled and completely autonomous program. This will keep memory requirements to a minimum, help the database manager to maintain the integrity of the data by excluding users from access to unauthorized modules, and ease in maintenance of the system

3. The software design will follow a top down structured approach but with a standard set of utility subroutines that perform common functions among the modules such as signing off and on the database, error routines, reading, writing, and deleting records from the database, and standard messages. This will permit the database to be further abstacted while adding validity and reliability to the software.

Required Standard Database Functions

To maintain any database, certain standard functions are required (Requirement 2.2, Appendix F). In addition, many other functions are required that are unique to the TOTAL database system. This section will describe the requirements of these functions in detail.

Each master file must have the ability to add, update, delete, and review the information within a given record. Selecting these records should be done via a unique key as described in the database record in Appendix C and the routines for these records are described in Appendix G. The file functions are described below:

1. ADD: Adding a master or variable record to a file. The

information should be edited for the proper syntax and compatibility with other data items. Typing should be kept to a minimum and the user should be allowed to select from groups of items to enter rather than typing in the data. This will keep keystroke errors to a minimum and aid in the integrity of the database.

2. UPDATE: Changing the information within a given record of a file. This function will allow the user to modify fields within a record except the record key. If the record key is found to be in error than the record must be deleted and added with the proper key. As in the ADD function, typing should be kept to a minimum and the user should be allowed to select from a group of items to enter to keep the error rate down. The same syntax, range, and compatibility checks should be used in the update that were used in the ADD function.

3. DELETE: Remove a record from a master and variable files. This function will permit the authorized user to remove a record from the database system. The user should select the record from a list of keys and be permitted to review the record before deletion. The system shall prompt the user to insure that the record shown is the one to be deleted. After deletion, the record should be stored to provide for restoring the data in case of user error.

4. REVIEW: This function will display the data within a specified record or series of records. A great deal of flexibility can be installed in this function. However, care must be take to keep the function as standard for each of the

file groupings as possible to make the system easier to use. The ability to create a hardcopy of the data shall also be an option associated with this function.

<u>System Limitations/Constraints</u>

The limitations on the system are really assumptions as to the number of students, faculty, books, classes. These will be made to aid in defining the storage capacity needed to run the system and the size of the programs. These requirments must be defined when generating the database. This is an important aspect of the system that was given little attention in other theses efforts. The following limitations apply to data items or records within the database system and are based on current AFIT figures (Requirement 2.3f, Appendix F):

1. Faculty Requirements: According to the faculty and staff roster maintained in AFIT/ENA, there are over 130 instructors, staff members, adjunt professors, and lecturers. This number does not include the secreteries and support personnel. Room should be allotted for 200 people to allow for growth and additional personnel. Each faculty record requires 460 bytes of storage space, so 92,000 bytes of storage will be required to store all of the AFIT/EN faculty.

2. Student Requirements: As of the writing of this thesis, there are 532 students enrolled in the School of Engineering according to AFIT/ENA rosters which includes part time and foreign students. To allow for expansion of the near future, the number of records allocated for students should be 585, which is ten percent greater then the number currently enrolled. This number assumes that when a student leaves, his/her records are

archived and removed from the current database. The logical record size for the student file is 460 bytes of memory, which means that 269,100 bytes of storage must be reserved to hold the master student file. When allocating the space for a student, we must remember that for each student there are several variable files. They are the Variable Awards File (VHAW), Variable Course File (CRSE), Variable Education File (VEDU),and the Variable AFIT Courses and Credits File (VCQR)

3. Department Requirements: There are currently five departments within the school of Engineering: Aeronautical and Astonautical, Mathematics, Electrical and Computer Engineering, Physics,and Operational Research. Space should be made available for seven departments to include those of the staff and faculty plus any additional departments added at a later date. The space required for this master file is 40 bytes or 280 bytes total required to store all of the departments on disk.

4. Thesis Catalog Requirements: This is a difficult requirement to specify because it is unknown if all theses will be required to be cataloged or just the ones whose students are currently enrolled. Since all theses are cataloged in the library, it would be a duplication of effort to catalog them in the database, however, the ability to call up a past thesis or search the database for a particular thesis would be a powerful management tool. For the time being, space will be reserved for those thesis whose students are currently enrolled. The decision to catalog all theses will be made at a later date. A seperate system could be used to catalog past theses that would be

available in the library for student use and updated on a quarterly basis.

5. Class Section Requirements: Within each department, there can exist several types of degrees. Each type of degree will have associated with it, a class of students. Currently, there are 15 different types of degrees awarded at the AFIT School of Engineering. However, at any one time there may be two sections overlapping for a certain degree. Space must be reserved for 30 sections which equates into 1500 bytes of storage. This assumes that a sections data will be archived upon the sections graduation.

6. Course Requirments: According to the AFIT 1985-1986 catalog, there are 493 courses offered in the AFIT School of Engineering. The faculty strives to keep AFIT and its courses up to date with the state of the art technology. For this reason, this file will be higly volatile. Course are continuously being added, deleted, or changed. It is expected that the database will be regenerated on an annual basis, so the requirement to store 493 courses plus ten percent (542) should be sufficient. Since 1218 bytes are required to store a course record, 69,376 bytes are required to store all of the records for one year.

7. Class Quarter Requirements: It is expected that information on quarters should be maintained for one year previous to the current quarter and 2 years in advance of the current quarter for a total of 3 years or 12 quarters. Each Master Quarter record requires 40 bytes, so 480 bytes should be allocated for the Master Quarter File.

8. Course Book Requirements: The average number of books

required for a course is between 1 and 2. This exact average is difficult to come by because of the volatile nature of the courses and instructors. For the purposes of this thesis, we will assume that no more than two books will be required for a class. With the number of courses set at 542, storage must be set aside for 1084 records. This equates into 140,920 bytes of storage required.

9. Daily Class Time Requirements: For the purpose of this thesis, we will assume that classes start on the hour with the first class starting at 0800 hours and that last class starting at 1700 hours. This is a total of 10 class times. However, each quarter there are a number of special instances where class times do not conform with this standard. To accomodate special requests, an additional 5 records shall be allocated for class start times that do not begin on the hour and/or are outside the 0800 to 1700 range. Each class time requires 20 bytes of storage for total of 300 bytes required for the total file.

10.Building/Room Requirements: Capacity for each room will be set at 30 which is the default enrollment value. Currently, classes are scheduled for buildings 640 and 641. The number of rooms vary from quarter to quarter. Initially, the limit will be set to 100 rooms total. With each record requiring 32 bytes of storage, 3200 bytes will be reqiured to store the required building/room records.

11.Degree/Sequence Requirements: This is also a very volatile file because of the changing degree requirements and the flexibility given the students and instructors when selecting a degree and sequence. For this purpose the sequences will have a number from 0 thru 999, and the degrees will have a number from 0 thru 99. The Master Sequence file will require 60,000 bytes to store the maximum number and the Master Degree File will require 6600 bytes to store the maximum number of degrees.

Required Response Time

Because of the size of the system and the fact that other programs will be running on the system at the same time, the issue of response time will remain flexible. However, it will be important to keep response time to a minimum to prevent psycological stepdown(5). If the system does not respond to the user within five to eight seconds then the user becomes unsure of what is happening to the system. With this in mind, the system should not be allowed to go for more than eight seconds (5) without reassuring the user that the system is still working on their problem (Requirement 2.4, Appendix F).

Required Reports

For future requirements,several additions to the basic system have already been suggested by some faculty members and students (Requirement 2.5, Appendix F). The first of these is a package to generate the Division Faculty Schedule and Manpower Requirement Expenditure Document using the information contained in the database. The basic understanding of the document, the formulas and the concept were derived from Dr. Gary Lamont. Interviews from Dr. Biezad (Lt Col, USAF) and from Dr Seward (Maj, USAF) have established the basic guidelines and needs for the system. This information will be reviewed to establish the feasibility of including this concept in the design. See

Appendix E for examples of the listing required and the graphic bar chart report.

> "The School of Engineering faculty workload is measured in terms of quantifiable activities representing a measure of faculty productivity and output. These activities include classroom courses and lectures, laboratory courses, MS thesis supervision, and PhD dissertation supervision. This particular set of faculty activities represents the quantity of educational output in an academic institution. All other activities such as course and program development, professional development, faculty research, consulting, etc., are not easily quantifiable.. This second set of activities contributes to the quality of educational output. Although the principal function of this second set is in a supporting role, its importance must not be underestimated. For example, lack of adequate research time for the faculty will lead to the deterioration of academic excellence and the loss of Air Force unique courses. Thus the school dean has the responsibility of striking a proper balance between the quantifiable teaching activities (and their efficiency and effectiveness) and the non-quantifiable supporting functions to achieve the overall objectives for individual programs." (18)

The 1983 manpower standards are expressed as a set of formulas in terms of four workload factors X1, X2, X3, X4 where

X1 = Contact hours for degree education.

X2 = Contact hours for PCE.

X3 = Master's degree graduates.

X4 = Doctoral degree students (dissertation phase only)

The manpower formulas obtained from the AFIT/ENG department are shown in figures 2.2 through 2.3

$$\text{Faculty} = (8068 + 8.008*X1 + 5.530*X2 + 81.48X3 + 217.3*X4) / 1742$$

$$\text{Dept Heads} = 8710/1742 + 5$$

$$\text{Secretaries} = (7230 + 1.062X1 + 0.7333*X2 + 10.80*X3 + 28.81*X4)/1742$$

Figure 2-2 EN Manpower Formulas

$$\text{ENS Faculty} = (1366 + 8.008*X1 + 81.48*X3)/1742$$

$$\text{Other Dept Facutly} = (1676 + 8.008*X1 + 5.530*X2 + 81.48*X3 + 217.3*X4)/1742$$

$$\text{ENS Secretaries} = (1413 + 1.062*X1 + 10.80*X3)/1742$$

$$\text{Other Dept Secretaries} = (1454 + 1.062*X1 + 0.7333*X2 + 10.80*X3 + 28.81*X4)/1742$$

Figure 2-3 EN Department Manpower Formulas

The second enhancement involves the ability to generate a class schedule using the information in the database and calling the scheduling system already in use. Captain Michael Mullennex was given as the point of contact in the scheduling office. He attended the EE 646 Database class in the Winter of 1985 and helped develop a prototype of such a system. This prototype built a temporary file of the information needed by the scheduling system and relied upon the user to initiate the job. The initial requirements state that this initiation of the class scheduler be made from the database system. The primary input to the scheduling system is the education plans produced by the students.

Until recently, the education plans were written by the students, checked for sequence and graduation requirements by the students faculty advisor, and then input into the system by the department secretary. The requirement exist to allow the student to generate education plans using the databaseand produce a hardcopy for their records. This will relieve the burden on the department secretary of entering the education plans on the computer. Udates to the education plan will depend upon department policy. A sample of what the edplan report should look like can be found in Appendix F.

In addition to the education plan requirements, there is a need for the faculty and students to be able to validate the students degree requirements and check to proper sequence entries. This would be done automaticaly when the student enters the education plan. Additionally, the program should have the ability to caclulate grade point averages on selected classes for the faculty only. This ability should be available on hard copy as well as interactively (2:H-1).

A printed listing of currently enrolled students should be made available to authorized individuals. The output should contain the number and names of the students programmed to take all classes for fiscal/calendar years. (2:H-2)

A listing should be available of all or partial lists of course information (quarters course taught, instructor, text used, credit hours, title, and number). (2:H-2)

Data Syntax and Compatibility Checks

To insure that accurate data is passed to the database system, data syntax and compatibility checks must be used whenever entering data (Requirement 2.6, Appendix F). The Forms

Management System (FMS) could be used for this to some degree, but the majority of the syntax and data compatibility checks must be done by the application software. For a complete description of data syntax and compatibility rules see Appendix C.

Computer Interface Requirements

Defining what information must be presented to the user and what the user must input to the system is a straight forward process. In the case of the AFIT/EN Student and Faculty Database System, the information has already been defined by previous thesis efforts. Up till know, the requirements definition has been confined to the manipulation of this information to achieve user goals. This section of the chapter will deal with the "human computer interface" (5) requirements definition (Requirement 3, Appendix F). This section will identify the targeted group of users that will use the system most often and come up with a standard profile. Using this profile as a standard, the system can be tailored to the react to the user. However, there are some difficulties in trying to define a standard user.

Users who try to define a system with the designers' goals in mind can find themselves in a predicament, particularly when one of the following is true(14:140):

1. The system in question is just not definable by "traditional" means.

2. The system can be defined, but the user doesn't know what he/she wants.

3. The user knows what he/she wants but can't articulate it.
When confronted by an undefinable system, the user is faced with

the impossible task of defining what he/she wants without knowing if it will work or not (14:146). They must perform some very difficult activities, such as reducing problem solutions to functional terms, visualilzing system components and the interaction of these components during everday operations, and discriminating between alternative approaches. Unfortunately, the only sure way to determine if a system will be acceptable to the users is to allow the user to try system. Prototyping addresses this problem and will be attempted for this application. With fast prototyping, construction of system will begin after a bare minimum of requirements are defined.

Targeted User Group

Pangman (2:2-5) described a mixed category of possible users for the system. Among them are secretaries, professors and instructors, department administrators, and students. For the application of this thesis effort, the user population will be confined to the student and faculty of the Electrical and Computer Engineering Department. This presents a problem because there is such a varied background among the users, including educational experience, computer background, and even typing abilities. Figure 2-4 lists the characteristics of the chosen "standard user" (Requirement 3.1, Appendix F).

1. Will be Between the age of 21 and 55.
2. Has at least 2 years of college.
3. Has either run a word processor, computer, or typewriter.
4. Will be considered a casual user of the system. (no more than once a week.
5. Will have limited access to manuals.
6. Will not know many of the abbreviations used in the current database.
7. Will not have access to Social Security numbers because of the privacy act.
8. May not have an American Social Security numbers. (i.e. foreign students)

Figure 2-5 Standard User Characteristics

Database Prototype Development

Normally, the first logical step would be to conduct a full requirements definition cycle before the actual production of code. However, because of the vast amount of work done by the EENG 6.64 Database class and by engineer Robert Ewing, there exists a complex and working prototype model currently being used by new students to enter student ed plans onto the database (Requirement 3.2, Appendix F). This creates a perfect environment for feedback on the "user-friendliness" of the system, and has proven very effective in designing useful computer software software for non-computer personnel.

> "This quick and dirty system has one purpose, and that is to show the users what they are asking for and give then some working knowledge of the results that can be achieved by the system they have defined." (14:146)

Other purposes include showing the user alternative methods for doing the same task and at the same time demonstrating new features of the system. It is important to give the users alternatives in describing there system.

This technique was used by Mr. C. Gerald Morrison and the autnor of this thesis at the 552 Airborne Warning and Control Division to design a computer database system for aircraft maintenance and scheduling problems. Mr. Morrsion used interviews and Air Force manuals on aircraft maintenance and scheduling to come up with a preliminary concept of the system. Then, using a CROMEMCO II computer and the COBOL language with a special display interface, he designed a rough package that displayed and accepted specific aircraft information. After showing this to the commander, likes and dislikes were noted as well as suggestions for future enhancements. In contrast to conventional programming technology, which restrains the programmer in the interest of orderly development, fast prototype development of systems must amplify the programmer or analyst in the interests of maximizing his effectiveness. This can require a small number of programmers to make essentially arbitrary transformations to very large amounts of code (6:23). However, the systems analyst or designer must be ready to discard the prototype and start designing the system from scratch.

To gain further insight into the needs of the user, interviews will be conducted to gain the thoughts of personnel who are not familiar with the AFIT/ENG Database System. The targeted group will be the new GCS-86D and GE-86D students who arrived in May of 1985. These students are required to enter their education plan via the ISL VAX 11/780 using the TOTAL Database Management system and the education plan software developed by engineer Robert Ewing. Students will be questioned as to whether they thought the system was easy to use, self documenting, and useful.

After interviewing each student personally, they will be interviewed as a group in hopes that ideas and comments from other students will spark conversation and ideas from others. It is very important to gain this information in the early stages of the design because the EDPLAN system, as developed by engineer Robert Ewing, will be used as a prototype for further design considerations and to add standardization to the system. These interviews will help in gaining useful information in formulating the degree requirements portion of the database. The suggestions of the students will be assembled and scrutinize to ascertain their usefulness.

Interview Results

Fifteen out of the thirty students who used the system were questioned as to their likes and dislikes of the education plan database prototype. An informal interview was performed because of the nature of the questions. The main objectives behind the interview was to find out how easy the system was to use, how long it took them to learn to use it, how "user-friendly" it was, which features they liked , and what they did not like about the system.

The program itself was designed to be self documenting. Once a user logged on, the program was executed and the user was prompted for specific answers to questions. Almost all of the students were able to use the system within a couple of minutes, however, several of the students who were not familiar with computers had difficulty in changing typing errors. Once a field was typed in and the user had moved on to the next field, many had difficulty in backing up to the previous field. Most of them, re-entered the program and started the process over again. This created many extra copies of the edplan being printed off at the laser printer.

The edplan program approached the problem of entering data via the standard approach. The student was required to know which class he/she was supposed to enter, if the class was given in a specific quarter, if the class belonged in the sequence selected, or even if the class existed. This required a great deal of typing and chance for error. However, very few of those individuals interviewed selected this as a problem area. Most excepted this as a normal way of doing business, except a few students who were not familiar with these procedures and some who were exceptionally skilled in software development.

Some of the suggestions made by the students and the faculty were on the "user-friendliness" of the system. It was noticed by several students that many of them were entering education plans that were very similar. Many had selected specific degree sequences that required the same classes. It was suggested that default edplans be made part of the system so a student could declare a sequence and update an edplan instead of creating a new one. The other suggestion was to have the students select classes from a list instead of typing them in, or at least provide an edit check of the classes before the database is updated (Requirement 3.4, Appendix F).

Summary

This chapter characterized the AFIT/EN departments requirements for a complete and integrated database system and outlined the steps taken in the preliminary design and source test of the proposed system. Although the requirements of the system will continue to change as more data is needed, as the school grows, and as new data becomes available, the importance of a good requirements definition phase will be evident in the design phase. The requirements have called for a system that can be easily expanded and maintained by the faculty and student body. In addition, the system will be designed to accomodate the "casual user" of the system such as new students and instructors.

III. Database Design

In chapter 2, the requirements for the AFIT/EN student and faculty database system have been well defined through several iterations of the requirements definition phase by this and previous thesis efforts. This chapter will present the preliminary and detailed design phase of the application software for the AFIT/EN Student and Faculty Database System. The preliminary design phase is sometimes referred to as a high-level design or system model showing what the system will accomplish in its task, but not specifically how it will be done. (4:12) In this portion of the design, system flowcharts are of use to present the system in an abstract pictorial form. The preliminary design should also be in such a form as to make it independent of machine and programming language. On the other hand, the detailed design of this system will take the preliminary design and refine it to the point where it can then be implemented onto a particular machine using a particular language. There should be a logical mapping from the requirements definitions to the preliminary design and then to the detailed design of the system. It is important at this point to continue to review the requirements of the system for errors and feasibility.

The design of the system will follow the top-down structured approach to software design. Structured design is a set of proposed general program design considerations and techniques for making codeing, debugging, and modification easier, faster, and less expensive by reducing complexity. (14:328) Simplicity of design will be the major theme when dividing the system into seperate pieces. To reduce the complexity even more, some of the linked lists that are to be used as a data structure and the database system TOTAL will have operations defined on them to improve the interface between the application programmer and the database system.

In this chapter, the schema will be reviewed to define and identify the relations that were built into the database system. The requirements of the system defined in chapter two will then be mapped into the preliminary design of the overall system. Finally, the preliminary design of the database system will be refinded and further defined into the detailed design of the sytem in two seperate categories. The first category will be the functional design of the system and the second will include the design of the interface to the system. Sometimes, these steps of the software development phase are considered as well defined steps with absolute borders on the definition of each phase. In this application of the waterfall model of software development, the process becomes more of an evolution of development where one phase transitions to the next with no distinct border.

Database Relationships

It was obvious, upon review of the database schema for the AFIT/EN database system, what some of the relationships and applications were invisioned by the original creators of the database. Many of the characteristics of the database schema appear to take on the traits of a relational database system but still maintain the complexity and speed of a network database. Each database relation will be examined for some of the relationships suggested by faculty members and students and there apparent application in the design of the database software. These applications will then be fit into the overall design of the system later in the chapter.

The student and faculty master files are linked by common variable files. Figure 3-1 gives a list of the master files and variable files along with there respective file codes. Each student and faculty member share common information in the Variable Honors and Awards File, Section File, Advisor File, and Thesis File. Figure 3-2 shows the database relationship to these two files. These relationships allow the application programmer to extract certain information:

1. Select a complete or partial list of all of the students who have the same faculty advisor, thesis advisor, and committee member(Figure 3-3).

2. Select the students and faculty who have attended the same university and achieved the same degree.

3. Retrieving information on the section adivisor and section leader for any given student (Figure 3-4).

4. Calculating the work load on an instructor by extracting the number of students he/she advises on thesis and dissertations.

5. Select and print all of the students who have signed up for the same class.

6. Retrieve the courses an instructor is teaching, or has

taught list thesis students he/she advises.

	MASTER FILES
FACT	MASTER FACULTY FILE
DEPT	MASTER DEPARTMENT FILE
STDT	MASTER STUDENT FILE
THES	MASTER THESIS NUMBER FILE
SECT	MASTER SECTION NUMBER FILE
MCRS	MASTER COURSE FILE
MQTR	MASTER QUARTER FILE
MBKT	MASTER BOOK FILE
MORD	MASTER ORDER BOOK FILE
TIME	MASTER CLASS TIME FILE
BLRM	MASTER BUILDING/ROOM FILE
CPTY	MASTER ROOM CAPACITY FILE
DAYS	MASTER DAY SCHEDULING FILE
MSSF	MASTER SEQUENCE FILE
MDEG	MASTER DEGREE REQUIREMENTS FILE
	VARIABLE FILES
VEDU	VARIABLE EDUCATION FILE
FSOC	VARIABLE FACULTY SOCIETY FILE
FCMT	VARIABLE FACULTY DEPARTMENT AND COMMITTEE MEMBER FILE
VHAW	VARIABLE HONORS AND AWARDS FILE
FINT	VARIABLE FACULTY INTERESTS FILE
FCOM	VARIABLE FACULTY PUBLICATIONS AND PRESENTATIONS FILE
FTDY	VARIABLE FACULTY TDY FILE
MCRS	VARIABLE STUDENT COURSE FILE
THTL	VARIABLE THESIS TITLE FILE
SECL	VARIABLE SECTION LEADER FILE
VCQR	VARIABLE QUARTER FILE
VREQ	VARIABLE PRE-REQUISITE FILE
VCBK	VARIABLE BOOK LINK FILE
VNMO	VARIABLE NUMBER ORDERED FILE
SCHD	VARIABLE CLASS SCHEDULE FILE
CLSR	VARIABLE CLASSROOM FILE
TCMT	VARIABLE THESIS COMMITTEE MEMBER FILE
FADV	VARIABLE FACULTY ADVISOR FILE
VINS	VARIABLE INSTRUCTOR STATISTICS FILE
TADV	VARIABLE THESIS ADVISOR FILE
VPDQ	VARIABLE PROFESSIONAL DEVELOPMENT FILE
VMSS	VARIABLE SEQUENCE FILE

Figure 3-1 Master and Variable File list

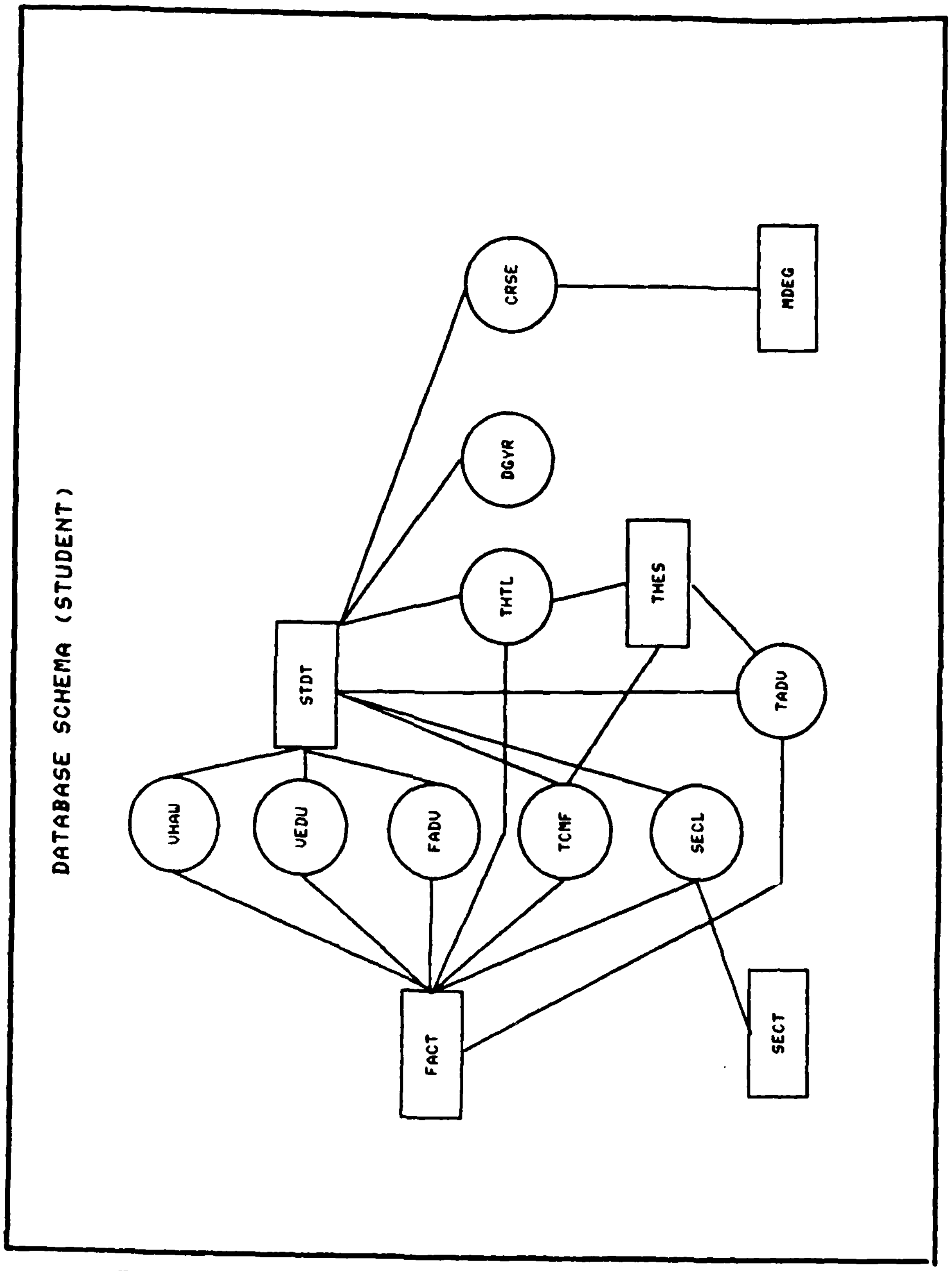


FIGURE 3-2 STUDENT AND FACULTY RELATIONSHIPS

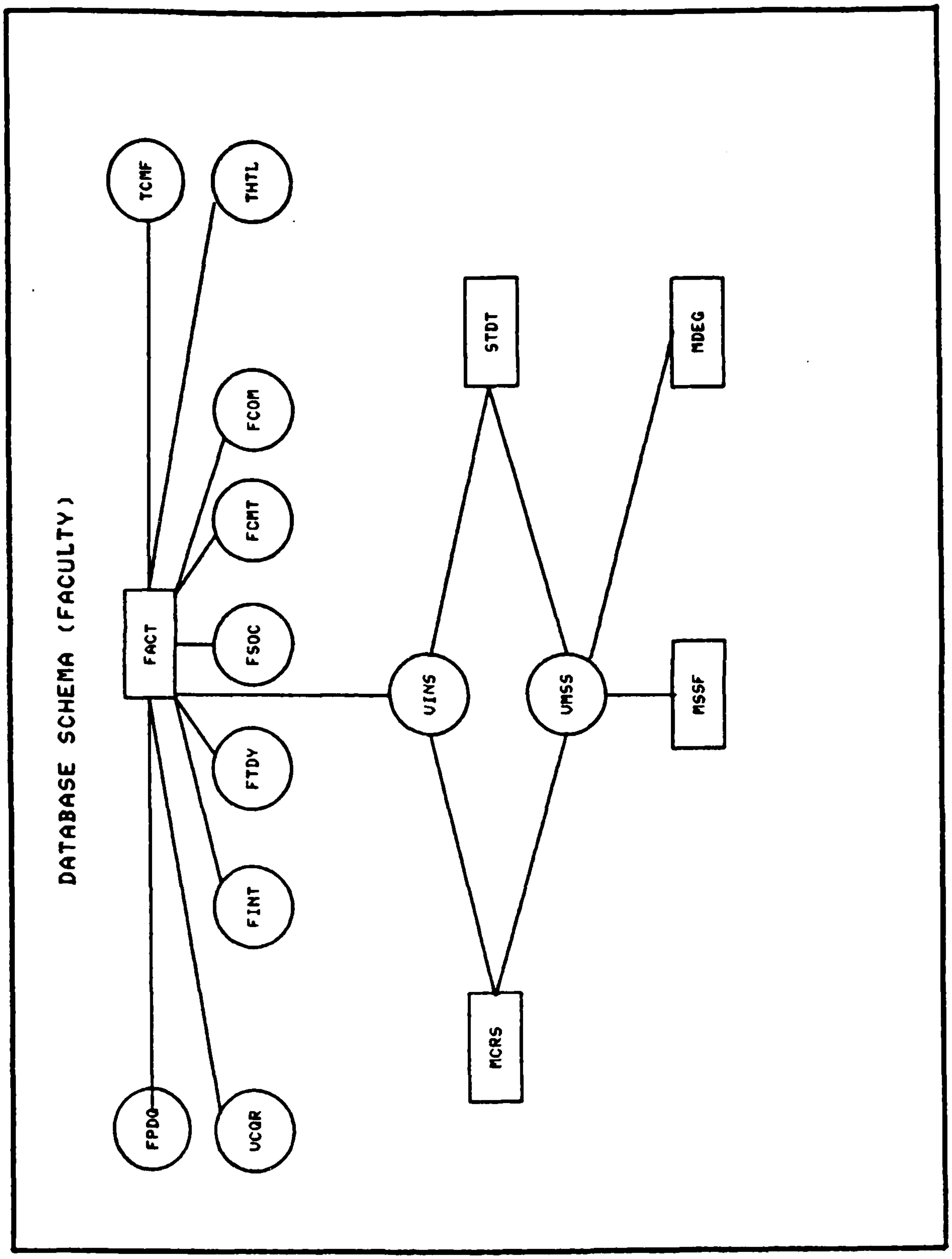


FIGURE 3-3 FACULTY RELATIONSHIPS

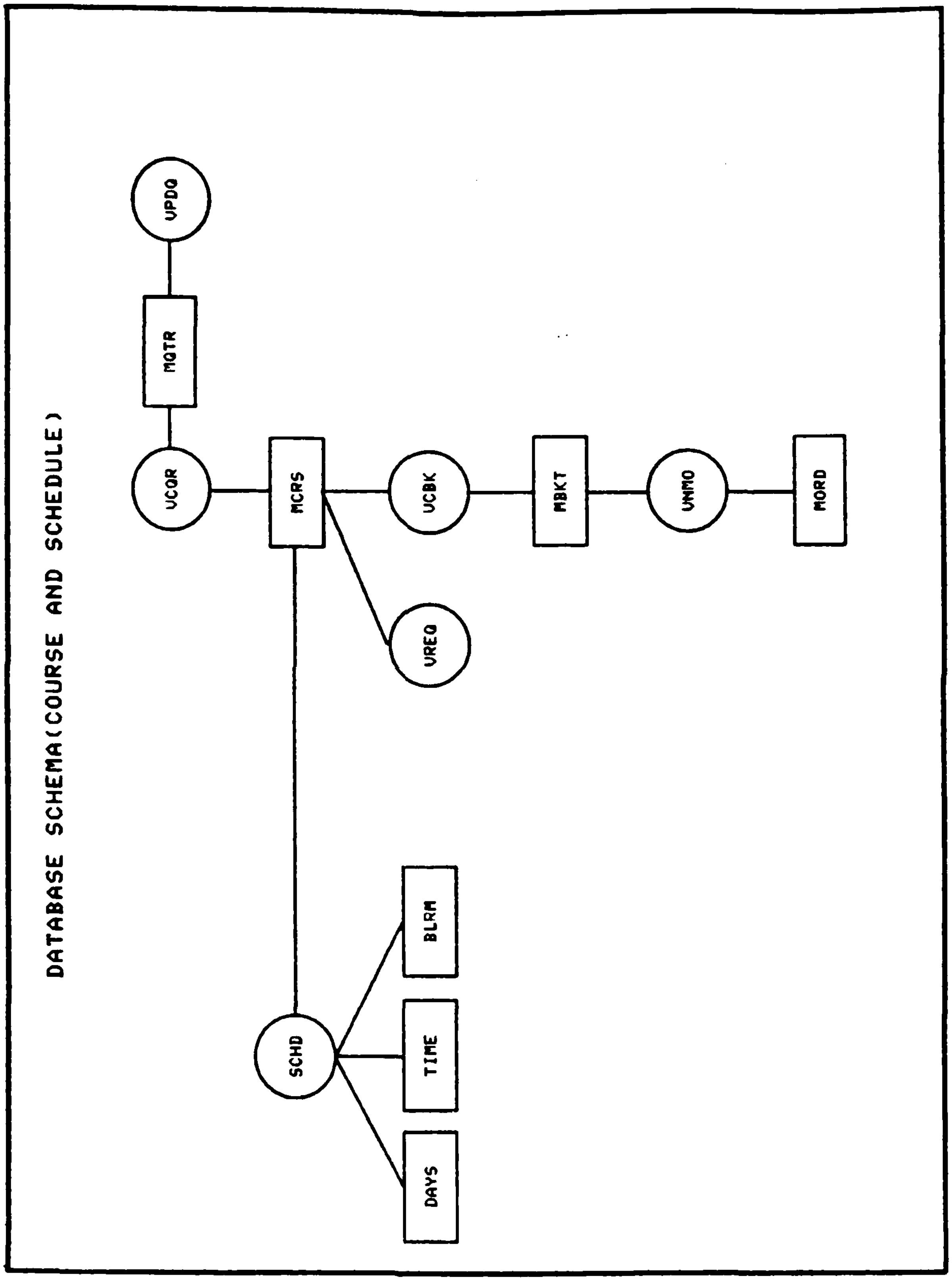


FIGURE 3-5 COURSE AND SCHEDULE RELATIONSHIPS

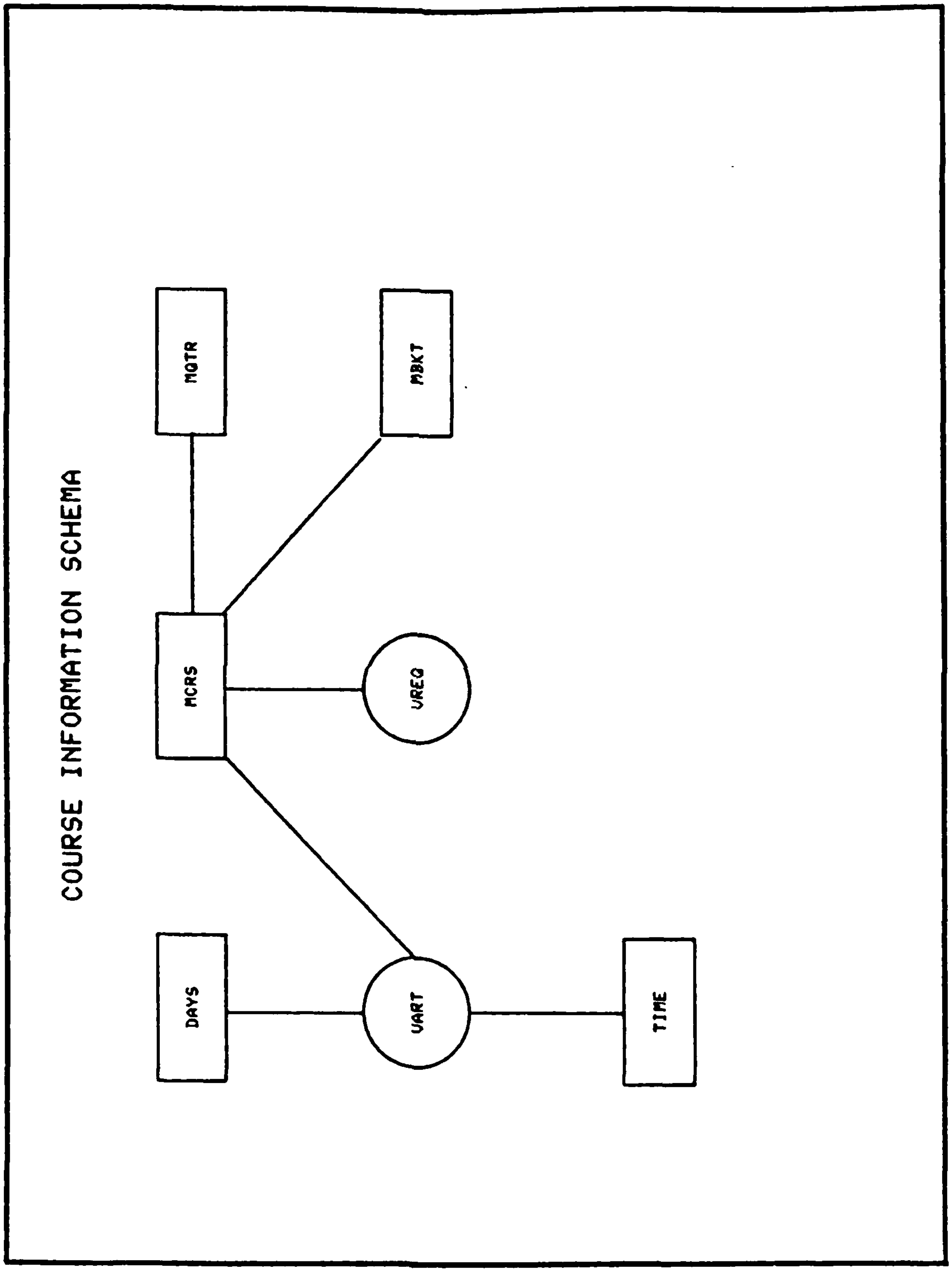


FIGURE 3-5 COURSE INFORMATION SCHEMA

The Master Course File schema has the potential to provide a great deal of information to the user of the system. Figure 3-5 contains the basic relationship of the Master Course File schema. Each course is linked to a variable quarter file which contains a series of records that designates when the course is offered and the Variable Schedule File which contains information as to when and where the course is given. Linked via the schedule file is the Master Class Time file and the Master Day file. Using these relations, some of the applications are:

1. When updating information on a course, there is only a need to change it in one location which makes the change global thoughout the entire database.

2. The database can be searched to examine the text books required for that course, the availability of those books, and the price of them.

3. The database can be searched to determine what classes are offered during a ceratain day and/or time. This would be useful for a student who is looking to replace a dropped class.

The sequence and degree requirements schema gives the faculty the ability to verify whether a student meets the requirments for a specific degree and sequence within a department. The Master Sequence File and the Master Degree Requirement File are linked through the Variable Sequence File. Using these relationships, the faculty will no longer need to check a students education plan to insure graduation requirements for a course sequence are met. The database can also be used to calculate the students eligibility for graduation.

PRELIMINARY DESIGN

The preliminary design phase of the software development life cycle bacically takes the requirements as defined in the requirements definition and maps these into a abstract picture of the system. At this stage in the development life cycle, system flowcharts and diagrams play an important part in depicting the system. Appendix D contains all of the system structure charts referenced in this chapter. Each requirement will be implemented in the design of the system (if applicable) and justified. To evaluate alternatives for dividing programs into modules, it becomes necessary to examine and evaluate types of "connections" between modules. A connection is a reference to some label or address defined elsewhere outside the module (15:329). When designing the system, it will become important to group modules together that are functionally related into a common seperately compiled program, and define operations on these groupings.

Main Program Module Design

Because of the database schema and previous specifications in chapter 2, there exists a requirement for the database system to be in a structured form and for the five file groupings to be contained in separately compiled programs. Appendix D has the system diagram for the five main functional groupings that are named STDTMOD, FACTMOD, MCRSMOD,THESMOD, SEQUMOD. These modules will perform the following functions:

1. STDTMOD: Maintain the Student Master File, Honors and Awards file, Student Course File, and the Sequence File through standard add, update, delete, and review routines. The only

exception is that students may not have access to grade changes. This must be done by a seperate module for security reasons. Students should only have access to their education plans and personal data while maintaining the ability to update their locator card on the computer.

2. FACTMOD: Maintain the Faculty Master File, Society File, Faculty Advisor File, Section File, Department and Committee Files, Honors and Awards File, Publications and Presentation File, TDY File, and the Instructor Statistics File.

3. MCRSMOD: Maintain the Master Course File, Master Quarter File, Master Book File, Master Order File, Master Class Time File, Master Building/Room File, Master Room Capacity File, Master Day Scheduling File, Variable Quarter File, Variable Requisite File, Variable Book Link File, Variable Number Ordered File,and Variable Classroom File. Modules to add master files, delete file, update files and review files will be included within this module.

4. THESMOD: Maintain the Master Thesis Number File, Variable Thesis Title File, Variable Thesis Committee Member File, and Thesis Advisor File.

5. SEQUMOD: Maintain the Master Sequence File, Master Degree Requirement File and Variable Sequence File. These Files will be maintained with add, delete, update, and review modules. Because of the changing requirements within the departments, these files will are seperated from the associated course module. Faculty and students should be allowed to vary sequences and degree requirements to obtain maximum flexibility. Therefore, the

degree requirements and sequences obtained are to be used as a guide and not a rule.

Data and File Abstraction

To make it easier on the user, the programmer, and the designer, operations are needed to act as an interface between a data structure and the TOTAL Database Maintenance System. These operations or algorithms must meet the following criteria (3:2):

1. There are zero or more quantities which are externally supplied.

2. At least one quantity is produced.

3. Each instruction must be clear and unambiguous.

4. When the instructions of an algorithm are traced, in all cases the algorithm will terminate after a finite number of steps.

5. Every instruction must be sufficiently basic that it can in principle be carried out by a person using only a pencil and paper.

By using these operations on the data types and files, we can then control the manipulation of data to such a degree that we can then validate the algorithms and add to the correctness of our program. It is important to note that the operations on these data types and files are descibed in terms of what is done but not how the operation should occur. This division of the tasks, called specification and implementation, is useful because it helps to control the complexity of the entire process (2:7). It is hoped that by doing this, the further development of application software will be made easier.

Each Master File and Variable File will have associated with it, a group of modules that act as an interface to the TOTAL Database System. Each file must have a record described in the TYPE declaration section with the ability to make it a linked list if necessary. The Figure 3-6 shows these modules where "XXXX" represents the four letter file code. For instance,to write to the Student Master File (STDT), the module WRMSTDT(STDT_RECORD) would be used. For a complete description of the common routines see Appendix H. These modules will have the following characteristics:

1. Each module will return a status code which notifies the calling routine of the success or failure of the database operation.

2. Each module will be responsible for transfering data from the record to or from the database schema. The TOTAL database system expects the data to be entered as a continuous string of data. This attribute should be hidden from the applications programmer. This is accomplished by defining the record types in the standard type declaration module that all programs wil have access to.

3. The write to variable records should handle the instance where there is more than one master record linked to the variable record. It may be difficult to define a standard set of variable record functions depending upon the linkage reference and the key of the master record associated with it.

MASTER FILE OPERATIONS

MODULE	FUNCTION
WRMXXXX	Write to a master file.
RDMXXXX	Read From a master file in a random or sequencial manner.
DLMXXXX	Delete a record from a master file.
ADMXXXX	Add a record to a master file.

VARIABLE FILE OPERATIONS

MODULE	FUNCTION
ADCXXXX	Add Variable Continue. Add a record after the record that is currently pointed to. the string.
ADAXXXX	Add Variable record after the second record in the string.
ADBXXXX	Add Variable record before the record being pointed to.
RDVXXXX	Reads the next variable record in the string.
RDRXXXX	Reads the previous variable record.
RDDXXXX	Read direct. Reads the record directly pointed to by reference pointers.
WRVXXXX	Writes a Variabel record.
DLDXXXX	Delete the current record pointed to by reference pointers.

Figure 3-6 Generic Standard Database Procedures

Throughout the database, there will be many instances where a search will be required on the Faculty, Student, and Staff files. Since these routines will be used in almost all of the programs, it will be important to define them early in the development of the system and implement the procedures in computer code. These list routines will be used to search for a student or faculty member by name or section and must be kept in alphabetical order at all times. In addition, the speed at which the lists are searched and sorted are of the utmost importance to the user. The operations on the list will be defined as:

1. ADDNAME: Adds a faculty or students data to the list of names.

2. DELNAME: Removes a faculty or students data from the list.

3. FINDNAME: Finds a faculty or student depending upon the last name and position the search started in the list.

4. CREATE: Creates a empty list with a header and trailer record.

5. BUILDLINKLIST: Builds the linked list of student or faculty names from reading the database sequentially.

A complete description of the axioms and procedure definitions can be found in Appendix H.

Standard Database Functions

Using the standard utilities that have been created to access the database, the standard functions associated with the database become easier. Each master file will be accessed through a series of menus and standard database functions. At the lowest level of access, each menu will give the user the ability to add records, delete records, update records, and review records with the option to produce a hardcopy. Figure 3-7 shows an example of a menu used in the Database Class (EENG646).

```
              COURSE SEQUENCE DATABASE MODULE

          OPTION                 DESCRIPTION

            1                    ADD A SEQUENCE

            2                    UPDATE A SEQUENCE

            3                    DELETE A SEQUENCE

            4                    REVIEW A SEQUENCE

            5                    CHECK A STUDENTS SEQUENCE

            9                    EXIT TO PREVIOUS MENU

            SELECT OPTION (1-5,9) __
```

Table 3-7

Master Sequence File Menu

The commercial frames system, from Digital Equipment Corporation (DEC) called the Form Management System (FMS), will be used to produce all of the displays seen by the user. Appendix E contains the frames that have been produced and used at the writing of this thesis. It is designed for ease of formulation and simulation in order to collect the transmited data in an orderly manner. By using this utility, standardization will be built in the interface to the database from the users point of view.

It is important to note that the number "9" is used to exit to the previous menu instead of the number "6" which would have been the next logical number in sequence. This was done for two reasons: 1) The number used to exit all of the screens in the system will be "9" and 2) The remaining digits will be reserved

for furture adaptations to the system. By standardizing the selection of items in the menus, the transition from using one portion of the system to another should be easy.
See Appendix E for the frames used in this development.

Database Generation and Initialization

The following procedures on generating and initializing the AFIT/EN database were taken from the TOTAL Database Users Manual, publication number p10-0001-00 with alterations to the procedure by Mr. Robert Ewing. This procedure is to be used whenever the sizes of the files or alterations in the schema are to be changed. See requirement number 2.3.1 through 2.3.11 for file size specifications. Performing this function allows the Database Manager to change the size of the files to accomodate an increase in students, faculty, course, sequence, and theses.

"The following steps should be followed for each Data Base Descriptor Module,(DBMOD), the user wishes to generate.

1. Code the input DBDL statements as described in chapter 4, the Data Base Generation chapter of the TOTAL Users Manual.

2. The DBGEN program will accept a sequential source file of 80 bytes image records.

3. Execute the DBGEN utility. This utility will read the DBGL statements if required and print the data-base documentation lilsting including all diagnostics and statistical messages. If an output file is required, it will be created with the output filename given with the extension .MAC. The following statements will execute DBGEN from a terminal provied the user has update priviledges:

```
$SET DEF DUA3:[TOTAL]
$RUN [TOTAL]DBG
 DBG>AFITDB,MAC=AFITDB.DBG
$MCR MAC AFITDB,AFITDB/-SP=AFITDB
$RUN DBF
FMT>DBMOD=AFITDB
```

FMT/

The output file will be the name of the .MAC file while the input file will contain the input DBDL. The default extention for the input file is .DBG. Upon completion of the database generation, if errors are detected they will be documented in the output listing at the terminal. The output listing file is not spooled. To spool the listing file the following step is necessary:

```
$PRINT LSTF11.DBG
```

4. Assemble the generated source program providing the DBMOD object file.

5. The Data-Base Descriptor Module (DBMOD) is now available for use with TOTAL and the utilities.

To start the database in operation, following the following VMS instructions:

```
1.   $SUBMIT TOTALINIT

2.   $RUN TOTALPRM

3.    TOT>AFITDB

4.    TOT>/
```

At this point, all of the files are empty and must be restored (16:3-5)"

Reports Generations and Design

Some of the required reports listed below were specified by Pangman (2 Appendix H of his thesis). Some of these requirements were specifically requested in the interviews conducted while others reflect anticipated requests by the users. Other required reports are a result of recent interviews and requests by the faculty. These reports are listed in requirement numbers 2.5.1 through 2.5.4 of Appendix F.

Required Reports

1. Division Faculty Schedule Manpower and Requirements Expenditure Document. (Requirement 2.5.1)

2. Faculty Workload Distribution /ENG
   (Requirement 2.5.2)
3. Enrolled Student Listing (Requirement 2.5.3)
4. Course Listing Information (Requirement 2.5.4)
5. Student Locator (Requirement 2.5.5)
6. Course Enrollment
7. Education Plans

Syntax and Compatibility Routines

The integrity of the data is vital to providing management with vital information. Requirement 2.6 of Appendix F states the necessity for a complete list of data syntax and compatibility tests to be performed on the data items when entered into the database system.

The TOTAL database management system handles many of the compatibility routines and the Forms Management System (FMS) will also handle many of the syntax operations. Items such as rank, dates, AFSC's, phone numbers, race, religion, sex, aero ratings, and items entered as years must be edited by the the application programs. See Appendix C for a complete list of syntax and compatibility rules.

Detail Design

Functional Design

The detailed design phase of the waterfall model of software development will be a refinement of the preliminary design to the extent that the detailed design can then be implemented in a computer language.(4:12) This portion of the detailed design will deal with developing a strategy of building software to perform the required functions assigned to the database. It involves taking the logical design of the system and making it a physical design that can be mapped into the implementation phase. The design of the proposed system imposes layers of software between the user and the machine itself. Figure 3-8 identifies these layers.

Database User
Layer 1 (Menu Select Software)
Layer 2 (Function Select)
Layer 3 (Functions Performed)
Layer 4 (Function Unique Modules)
Layer 5 (Standard Database Functions)
Layer 6 (TOTAL DBMS)
Layer 7 (System Software)
Layer 8 (Machine Level)

Figure 3-8 AFIT/EN Software Layer Description

This part of the chapter will take the preliminary design steps and refine them into the detailed design. The main program modules specified in the first part of this chapter will be defined functionally and by means of structured diagrams. The data and file operations will be implemented and tested in this chapter to facilitate the implementation of the application software. The standard database functions will also be implemented and tested. These two steps are required to construct the layer 5 (Figure 3-8) of the database in order to form a "user-friendly" interface to the TOTAL database system. Each step of the design should relate to some portion of the requirements in Appendix F.

Database Application Software Layer Description

1. Layer 1: This layer of software consists of the main module routines and offers the closest interaction to the user. These modules present choices to the user as to what portion of the database they wish to work with. (i.e. FACTMOD, STDTMOD, THESMOD, MCRSMOD, SEQUMOD). This layer may be implemented in a command file mode and should check to see if the TOTAL database system is in operation.

2. Layer 2: This layer is associated with each of the five distinct functional modules (requirement 2.1, Appendix F). These modules will present to the user a choice of what files or information they wish to modify (Add, Update, Delete, or Review) or if selecting an alternative function associated with the file such as the scheduling routine, report listings, or running management information programs.

3. Layer 3: This layer of software will perform the actual function requested in layer 2. The adding of records, deleting of records, updates and reviews will be part of the software required for these modules. These layers are still responsible for prompting the user for information and instructions.

4. Layer 4. This layer will be hidden from the interactive database user. These modules perform operations unique to the functions of layer 3. Some examples of these would be formatting a report for course listings, syntax checks on the records of information, preparing links to select variable records, and frames displays. In fact, any module that is used by only one function would be considered part of this layer.

5. Layer 5: These modules are standard routines that act as a "user-friendly" interface to the TOTAL Database System. Adding of master records, reading master records, link list manipulation, signing on and off, and status checking are some of the routines required.

6. Layer 6: This layer is the TOTAL Database System itself. This layer is presented to the application programmer in the forms of DATBAS subroutine calls.

7. Layers 7 & 8: These layers are system software that interacts with the operating system, and the machine itself.

Standard Database Module Descriptions

Layer 5 of the design of the system allows the programmer to have ease of access to the TOTAL database system. Because of the wide range of operations allowed by TOTAL, a subset of these operations will be selected for use. The justification of the operation will be included with the description of the module.

Master File Data Set Functions

The following procedures are required as the interface to the AFIT Database system through TOTAL. To operate these procedures, a standard set of type declarations is required to be included with any run of the database.

1. PROCEDURE SIGNONOROFF(OPER:BUFF5; DATASET:BUFF4): This will log the applications program onto the database. The parameter passed to the routine will identify which of the five main modules is running and the appropriate schema which idetifies what files can be accessed will be loaded. This routine should also log off the application program from the database. Every program that uses the database is required to sign on and identify itself to TOTAL.

2. PROCEDURE CHECKSTATUS(OK): This procedure checks the status of the database call and returns to the caller a boolean variable, (TRUE = Good database read, FALSE = Error in the call). If an error occurs, the checkstatus routine should return a message to the user of the outcome and offer some diagnostics. This procedure should be called at the end of each DATBAS procedure call.

3. PROCEDURE RDMXXXX(VAR XXXX : XXXX_REC; KEY: typekey): This procedure will read a master record from the database. The "XXXX" is to be replaced by the four letter file code of the file as defined in the database generation listing (figure 3-1). The program will format a record of the type XXXX_REC with the information passed in the buffer area. The

data item "KEY" will contain the master key of the record. This procedure is required for all master records in the database.

4. PROCEDURE WRMXXXX(XXXX: XXXX_REC): This procedure writes a record back to the database after an update transaction. The information contained in the record of type XXXX_REC is transferd into the buffer area in the same order the data appears in the record. This procedure is required in only those modules that have update operations.

5. PROCEDURE ADMXXXX(XXXX: XXXX_REC; KEY: typekey): The add master routine is used to add a new master record to the database. The operation is almost identical to the WRMXXXX except for the function assigned to the database call. This routine is used only by those modules which are allowed to add new records to the database.

6. PROCEDURE DLMXXXX(KEY: typekey): This routine reads the master record in the update mode to insure it is part of the database, if it is, then the user is prompted to insure he wishes to delete the record. If the record is to be deleted, all variable file links are deleted and stored in a save file in case restoration is desired, and then the master file is deleted and saved.

Variable Data Set Functions

These routines are a subset of the functions permitted by the TOTAL database system. The READR and ADDVB operations were omitted because combinations of the other operations could perform the same functions. Because of the flexibility of the variable record and the number of links, and master file keys, the files that these are to be assigned to will be limited to those files with master record lengths that have the same key length and/or have only one variable link.

1. PROCEDURE ADCXXXX(XXXX: XXXX_REC; KEY: keytype); This routine will add a variable record after the variable record currently pointed to by the database pointer.

2. PROCEDURE RDVXXXX(VAR XXXX:XXXX_REC; VAR VREFERENCE: BUFF4;
CODE:typekey); This routine reads the variable record from the file designated by the four digit code "XXXX" that occurs in the next string after the record pointed to by the VREFERENCE pointer. The information is then formatted into the record defined by the record type XXXX_REC. This routine is a standard sequential read function and will be used in almost all applications.

3. PROCEDURE RDDXXXX(VAR XXXX:XXXX_REC; VREFRENCE:BUFF4; CODE: typekey): This routine will read a record directly from the database file with the code XXXX pointed to by the code contained in the parameter: VREFERENCE. The information will then be formatted into the record of type XXXX_REC and returned to the calling program. This routine is useful in the application of updating a database record.

4. PROCEDURE WRVXXXX(XXXX:XXXX_REC; VREFERENCE:BUFF4; CODE:typekey): The Write Variable record routine writes a record directly to the database in the same position it was retrieved from. This record must be read in the update mode before it is written back to the file. This routine is useful in applications

for updating the variable records in the database.

5. PROCEDURE DLDXXXX(XXXX:XXXX_REC; VREFERENCE:BUFF4; CODE: typekey): The Delete Variable Record routine removes a record from the database in the file indicated by the four letter code: XXXX. The record must be read in the update mode before it is deleted.

Database Backup Utilities

It has been proven by years of experience that any given machine will sooner or later break down. To guard against such disasters, a system of backup utility programs will be employed to retrieve the database information and store the data in sequential files that can be read and transformed back to the database. Each of the five main modules will have the responsibility for backing up and restoring the database. The files each module is responsible for is depicted in figure 3-9.

MASTER RECORDS	VARIABLE RECORDS
FACT	VEDU
DEPT	FSOC
	FCMT
	VHAW
	FINT
	FTDY
	FCOM
	FADV
	VINS
	VPDQ

FIGURE 3-9(1) FACTMOD BACKUP FILES

MASTER RECORDS	VARIABLE RECORDS
STDT	MCRS
SECT	SECL

FIGURE 3-9(2) STDTMOD BACKUP FILES

MASTER RECORDS	VARIABLE RECORDS
THES	THTL
	TCMT
	TADV

FIGURE 3-9(3) THESMOD BACKUP FILES

MASTER RECORDS	VARIABLE RECORDS
MQTR	VCQR
MBKT	VREQ
MORD	VCBK
TIME	VNMO
BLRM	SCHD
CPTY	CLSR
DAYS	
MCRS	

FIGURE 3-9(4) MCRSMOD BACKUP FILES

MASTER RECORDS	VARIABLE RECORDS
MSSF	VMSS
MDEG	

FIGURE 3-9(5) SEQUMOD BACKUP FILES

Database Algorithms Selection

To eliminate the need for sorting and to decrease the search time involved with large list, a doubly linked list will be maintained by the programs for each master file associated with the module. This will reduce the amount of database accesses needed by maintaining a minimum amount of information in a linked list. Since the lists will be inserted in alphabetic order, no sorting will be required, so the insertion of an item will take on the average O(N/2) number of record comparrisons, where (N) is the number of records in the list. The search for an item will take the same amount of time. The linked list approach also allows us to dynamically allocate storage for the records which eliminates the need to maintain a large unused storage area. However, for less volatile files, such as the Master Section File, a simple array of records can be loaded and maintained at the beginning of the interactive session. The routines needed to maintain and access these lists are described below using psuedo code.

1. ADD_XXXX(XXXX:XXXX_PTR; HEADER:XXXX_PTR);

```
 BEGIN
   If the HEADER is nil then create a new header
       record
   else get the next record after the header record.
      while the search pointer is not nil and
        the input node is > searched node then begin
           get the next record
     end
     insert the new pointer in the list.
  end.
```

2. DEL_XXX (CTRL:typekey; VAR HEADER:XXXX_PTR);

```
  BEGIN
    While not at end of list and not found do
       get next record
```

```
          end do;
          if found then remove node
         END;

  3. FIND_XXXX( Name: nametype; CTRL:ctrltype; LIST: listtype);
        BEGIN
         while no match (name  = list.name) do
             current.ptr := current.ptr^.next
           if found then ctrl := current.ctrl
           else ctrl := spaces.
        END;
```

Interface Design Considerations

The four main features of a "user-friendly" interface for the AFIT/ENG Database System are:

1. Allowing the user to navigate through the control paths of the interface, following the structure as though traversing a tree.

2. Allowing the user to enter input data, and have the user select data to avoid user input errors. Example: Selecting a student from a list of students instead of looking up the social security number to enter as a key.

3. Informing the user of errors in as much detail as possible and always giving the user a way to recover.

4. The aid of a help function to tell the user what the machine is expecting as input.

5. Avoid putting default values in the menu to prevent the accidental selection of an option such as "9" (exit). Initial tests showed that when response was slow, users would hit the return key to elicit a response and inadvertently exit the program.

The method of integrating these features are through the design of the program in a structured form to allow for the tree structure to be implemented, the visual design of the human interfaces, the selection of phrases that would be familiar to the user, and the design of the help displays.

Visual Design Considerations

Since the "user-friendliness" of a system is determined by the user through the visual displays and interfaces, the design of the screen output media becomes very important. The main consideration behind the desing of the screen frames and display formats will be the following criteria:

1: The displays will remain consistant from one portion of the database to another. Student and faculty information will be displayed in the same manner as will the course sequences, degree requirements, and educations plans.

2. Items selected from a menu will remain consistent thoughout the database, using numbers to select a particular function. When performing adds, updates, deletes, reviews, and special functions within a frame, the number 1 will be used to select the add, number 2 to select the update, 3 to select the delete, the number 4 to select the review, and 5 through 8 for special functions. The number 9 will be used to exit the menu.

3. Each screen should allow the user to abort the function and recover to the previous menu. The information such as how to move through a screen, how to enter data, how to abort the screen, and how to call the help text should be displayed on the frame or available upon pressing the Pf2 key.

4. The title of the screen should appear at the top of the form in oversize letters in reverse video or bold colors. The use of bolding should be limited as it tends to clutter the

screen and this type of graphics is not compatable with many terminals. Columns of data such as the kind that appear on the course sequence add, update, and review screens can be displ·'ed in reverse video to highlight that they are data and not titles.

Help Level Integration and Tutorials

There are three kinds of users to any system:

1: The novice is a new user of the system, and has little or no experience with similar systems. This user requires a great deal of assistance and training.

2: The casual user is one that uses the system three or four times a month or a new user that's had experience with some similar systems. This kind of user needs help occasionally but can otherwise use the system effectively.

3: The expert user is a person who is very familiar with the system and uses it on a day to day basis. Little or no help is required for this type of user.

The use of the database system will assumed to be on a casual level.(2) The majority of work will come at the end and beginning of quarter when new students are added, old students are archived, classes are changed, and schedules are run. During the majority of the quarter, the use of the database system will be directed to generating management information and running of course and student listings. There will be very few people categorized as expert users. For this reason, a menu driven system was selected instead of a operation and operand system.

The system will be configured to accomodate the casual user most of the time. The novice user need only select the tutorials that will be implemented with each main module or select the PF2 (help) key to find out what the machine is expecting.

<u>Summary Of Design Considerations</u>

Besides the enhancements to the database schema and the structure charts in the appendices, the following represent all of the design considerations presented in this chapter.

1. The entire AFIT/ENG Database System will be decomposed into five main modules or programs. The programs are name (1) FACTMOD, (2) STDTMOD, (3) MCRSMOD, (4) THESMOD, (5) SEQUMOD. Each of these modules will have update responsibility for different parts of the database.

2. Certain routines and list structures should be abstracted into pre-defined operations. The AFITDB TOTAL database system will have record type definitions and read, write, update, and delete routines defined on them. In addition, a linked list of often used record names and keys will be maintained to decrease the search time required to access the TOTAL database system.

3. Each master file will have at the very least a set of standard functions the user will be able to perform. The user should be able to add, update, delete, and review records with the option for hardcopy.

4. To accomodate changes in the size of the AFIT organization, the database should have the ability to save records, re-generate the database, and restore the database records.

5. The requirements stated in Appendix E state the need for certain reports. These include but are not limited to: (1)

Division Faculty Workload Distribution Document (2) Enrolled Student listing (3) Couse listing, (4) Student locators (5) Education Plans and, (6) Sequence and Degree Requirements Listing.

6. To insure the integrity of the database, the data must have syntax and compatiblity checks performed when entering data.

7. To make the program easy to maintain and modify, an eight layer structure of software should be used (figure 3-8).

8. A library of standard database modules should be maintained in a seperate library file for future programmers to use in creating new modules or enhancing old modules. These routines should not rely on any global variables except for one. STATUS should be defined as "STATUS:BUFF4" through out any program as a global variable.

9. Each of the main modules (figure 3-9) should have the responsibilty to backup and restore those files the module has update priviledges for. This option should be password protected and should be done on a weekly basis.

10. When programming the interface to the user, four things should be considered: (1) Paths through the system should be well defined and should not change, (2) Have the user select input instead of typing it in to improve data integrity, (3) Display detailed error messages, (4) Insure the PF2 key works in all cases to provide the user with all the information a novice would need.

11. The visual design of the system should : (1) Limit the amount of color used on the VT240 terminals, (2) Maintain standard selection terms (i.e. 1=ADD, 2=UPDATE...9=EXIT), (3)

Each screen should allow the user to exit without any changes and, (4) Each screen should have a title at the top of the screen in double letters.

12. The system should be designed for the casual user and the novice should have access to all of the help and tutorials needed through menu selections and the PF2 key.

13. The lower level routines should be designed in such a manner to allow maintenance programmers to develop a command language database system to accomodate the expert users and allow access to the database through devices that are not compatible with FMS.

IV. IMPLEMENTATION AND TEST

This chapter discusses how the proposed design of an AFIT/ENG Faculty and Student Database System was implemented on a Digital Equipment Corporation (DEC) 11/780 computer using a VMS operating system. The implementation of this system was affected by many different factors. The availability of software that has already been written greatly affected the design considerations outlined in chapter 3. The implementation and testing of the system presented in this chapter will identify those characteristic of the system that deviate from the original design.

The topics discussed in this chapter include: an evaluation of the database prototype, the configuration of the host computer, forms Management System features and usage, the TOTAL database interface and integration problems, the development of the EDPLAN program, database generation techniques and problems, system integration, and module test plans.

Prototype Evaluation

During the final weeks of the design phase and the start of the implementation phase, 15 students were ask to participate in a mock database session. All of the people ask to participate fit the description of the standard user. Each individual was given a task to perform and taught how to logon to the computer, how to start the database, and how to use the help key(PF2). The subjects were observed to see if they could accomplish the task without the use of manuals or assistance. During the session, the subjects were encouraged to critique the system.

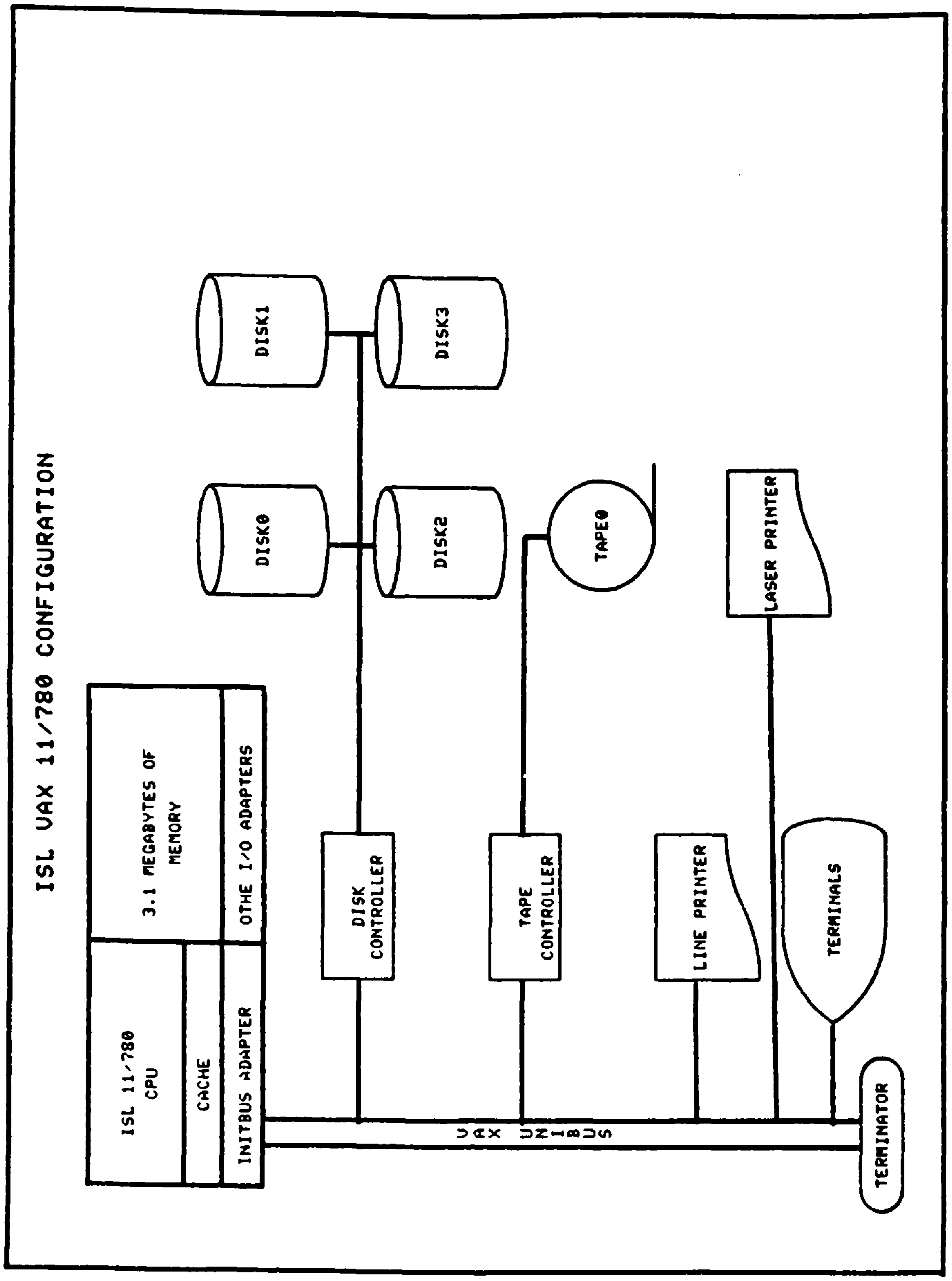


FIGURE 4-1 ISL 11/780 VAX HARDWARE CONFIGURATION

Several comments were common to most of the users and several offered improvements to the system. The list below is a sample of comments that were common to over half of those tested.

1. There was a lack of help menus and tutorials on most of the systems, but some were very good. The best was the help menu in the education plan module which used a secondary frame to display more information.

2. Abbreviations were unclear and should be avoided when possible.

3. When the users pressed the PF2 (Help key) they expected to see examples of what they needed to enter.

4. Several screens did not allow the user to exit without performing some action that caused a change. There should be a way to exit each function and screen.

5. The add, update, and review functions look very similar. At times the user forgot what mode he/she was in.

6. The default option in a menu (9 = exit) caused a problem when the system was slow to respond. The user would tap the return key and completely exit the system. The default should be taken out and left blank.

7. Menus changed from when the user originally signed on (edplans), this confused the user. They did not know if they were in the correct part of the database system. The menus should remain consistant.

8. There were two types of scrolling and selection of records. One method had the user scroll a list by, select a record and then enter a coded number that appeared beside the

record. The other method also scrolled the items by, but the user selected a record by placing an "x" beside the record to be selected. The majority of the users prefered the first method because the placing of the cursor beside the location took longer than it did to just enter a number.

9. The faculty module had the beeper turned on which bother the user. The beeper served no purpose other than to embarris the user.

10. When upd ting course sequences and edplans, the updating started at the personal data part. This usually does not change and the majority of those tested suggested that the position of the cursor be placed at the first class list and allow the user to back up to the personal data if need be.

Host Computer Configuration

The configuration of the host computer shall be a major concern in the future implementation of transfer of the AFIT/ENG Faculty and Student database system to another computer. This section of chapter 4 will deal with the computer set up and the file systems needed to support the TOTAL DBMS and AFIT database.

The computer used in this thesis development was the ISL VAX 11/780 located in room 245 of the building 640 (AFIT School of Engineering). The system contains 2.5 megabytes of main memory, four RK07 disk drives, 8 to 10 terminals, one on line printer, and one laser printer. The two types of terminals used in the implemention were the VT250 and the VT100. The VT250 had color capability and was used in some applications. The operating system used on the computer was version 4.2 of the VAX/VMS operating system upgraded from version 3.6 half way through the implementation. This produced several errors in the software because the version 2.Ø of the TOTAL DBMS was not compatible with the new operating system. The version of the TOTAL Database System was upgraded from 2.Ø to 3.5. The VMS operating system employed demand paging system using a fixed size of 512k bytes per page.

The TOTAL DBMS was maintained and run from the directory [AFITDB.TOTAL] and maintained under the disk pack identified by DUAØ:. To access the TOTAL database system commands, the user must logon under the user id of "AFITDB". Once on the system, the following VMS instruction was needed to put the user inside the directory: "SET DEF DUAØ:[AFITDB.TOTAL]". The TOTAL DBMS could then be generated, submitted and started execution. All of the datafiles and TOTAL utility programs were kept under the same directory name for ease of access and maintainability. To Back the database files, the entire data files were copied to an identical directory protected by the system identification on the disk pack DUAØ: To restore the database, the files were just copied back to the original disk pack under a user id with system priviledges.

The application software (i.e. FACTMOD, SEQUMOD,...etc) were maintained on the same disk pack as the TOTAL DBMS but under a seperate directory called "AFITDB". To compile and link a program with the TOTAL database system, the following VMS commands must be used:

```
$PAS/NOWARN  programname.PAS
```

```
$LINK programname,DUA0:[AFITDB.TOTAL]NATDATBAS,NATBUF
```

The term "programname" is the file name given to location of the source program. To then execute the program, the instruction $RUN programname is used. This assumes that the programmer has correctly signed on and accessed the database system properly (see appendix F for examples).

Forms Management System Utilization

Each module (i.e STDTMOD, FACTMOD ...etc) has associated with it the FMS library of the same name that containing all of the menus and screens the module needs to use for an interactive session. Therefore, the frames library associated with the STDTMOD.EXE module would be maintained as STDTMOD.FLB. The screens used are described in Appendix E and follow certain conventions.

The conventions or standards were developed using the prototype as a means of testing the reaction of different techniques on perspective users. The standards developed from the use of the prototype are:

1. Different transactions were selected using a number instead of a alphabetic symbol such as ADD, or DEL. The number 1 was always used to add records, 2 is always used to update and so on.

2. The number 9 is always used to exit to the previous menu or system. This also limits the number of selections on a screen by not allowing room for anymore selection on the screen.

3. Auto tabbing was not used as it is often confusing to the casual user and promoted mistakes in entering data. When

entering text information, data contained in one field would spill into another field by mistake.

4. The use of scrolled areas to select names of students and faculty, book titles, sequence titles, degree names, and section names reduced the need to memorize keys such as social security numbers, sequence numbers, and lengthy book titles.

5. The use of bold areas, reverse video, double wide letters, and boxes were kept to a minimum because of their incompatibility with other terminals that were not VT52, VT100, and VT240 used in the development of the system.

6. Each screen employed a method to exit the current operation without change. For menus it was accomplished using the number 9, and for other screens, hitting return on the first field caused the program to exit to the previous operation.

7. The method used to move around screen is by use of the 'TAB' and 'Backspace' or 'CTRL H' keys. The tab advances the cursor while the Backspace or 'CTRL H' key moved the cursor back one field. The 'Delete' key removed a character up until the beginning of the field. Furthor specifics can be found in the VAX-11 Software Reference Manual (10).

8. Often, more than one transaction of the same type would need to be accomplished such as updates to the students edplans. Usually, a secretary would bring several students records to update at one time. This would require the screen to be displayed to update students records until it was cancelled by one of the methods mentioned before.

9. The FMS program employs two basic types of GET calls

that retreive information from the screen. The first is FDV$GETAL which retrieves the entire screen into on large buffer where the program must retreive the information. This allows the use of the 'Backspace' key to move to the previous field but the program has no control of the operation until the FMS driver releases the information. The other is the FDV$GET which retreives one item at a time but does not allow the user to back up to the previous item but does allow for edit checks on the items as they are entered. The latter of the two was used because it allowed for edit checks and the data item 'TERMINATOR' was checked for the 'Backspace' key to allow the user the ability to go back to the previous field.

Education Plan Prograrm Developement

The requirements for the EDPLAN program were essentially defined by engineer Robert Ewing in his development of the EDPLAN prototype. The main requirements for the program were to 1) enable new students to enter their personal information and an initial education plan and sequence declaration, 2) allow the students to update the edplans, 3) allow students and faculty to review and print the education plans, and 4) generate a file that could be passed to the scheduling office that would allow them to schedule classes.

The first step in creating the program was to copy the files TYPE.PAS and UTIL.PAS from the DUA1:[PANGMAN] directory into a file called EDPLAN.PAS. The files TYPE.PAS and UTIL.PAS hold the standard type declarations, standard database routines, and link list routines as described in chapter 3. A forms library was created and called STDTMOD.FLB since the edplan program will eventually become a part of the STDTMOD module. At this point all syntax errors and as many logic errors as possible had been detected so the code contained in EDPLAN.PAS was in working condition.

A main routine was created and performed the following steps:

1. Initialized the FMS driver and opened the form library.

2. Signed on to the database system. If the database system was not operational then the program exited.

3. Built the linked list for the faculty, students, and an array of the class section.

4. Set up a WHILE loop to detect the type of transaction the user wanted:

   a. Add an Education Plan;

   b. Update an Education Plan;

   c. Delete an Education Plan;

   d. Review an Education Plan;

   e. List Students in a Section;

   f. Print an Education Plan for a Student;

   g. Print Education Plans for a Section;

   h. Generate the registration summary file;

   i. or Exit the Pogram.

To test the calls to the various routines, stubs were put in the place of the actual routines and the software was tested to insure it signed on to the database system, initialized the FMS driver, and made correct calls to the stubs when ever a transaction was selected. The program then tested to see if it signed off of the database.

The routines needed to add the education plan and basic student information were created first and tested. It was important to insure this worked first for several reasons. The first reason is that most of the other routines read information from the files. The update, delete, and review routines would use several of the functions and procedures developed in the add procedure. Since most of the procedures to read and write to the database had been developed and tested (Appendix I), the majority of the programming effort was concentrated on providing a "user-friendly" interface to the program and providing as much error checking as possible. Help messages and menus were programmed in to provide information to the novice user.

The next set of routines developed were the update routines. The only difference between the add and the update routines was the need to read a students master and course records. This involved selecting a student from the database system. Since few individuals would have access to the social security numbers, it was decided to develop a way to access a record using the last name. If more than one student had the same last name, a list of these names would appear and allow a selection to be made. A combination of the linked list routines, and the FMS scrolling ability aided in this design.

With the above routines completed and tested, the review and list students routines were completed using a combination of the add, and update routines. These were completed and tested in a matter of hours because all of the routines called had been developed and tested earlier. The last routines completed were the printing routines which basically followed the review routines in their format. Using this program as a basis, other pieces of software can be developed in minimal time if some basic rules layed down by this thesis as followed.

<u>Edplan Program Unique Routine Descriptions</u>

The following descriptions are of the routines developed for the EDPLAN program only. However, using these routines are guides will aid in the development of other programs.

1. PROCEDURE FINDSECTION(VAR FIND: LINK_PTR; SEARCHSECT:BUFF8);

This routine finds the first occurance of a student who belongs to a specific section passed via the SEARCHSECT parameter. The starting location is passed in the FIND parameter and the location is passed back through the same pointer. The routine was designed to be called until the entire list of student or faculty members had been exhausted.

2. PROCEDURE DISPLAY_NAME(VAR NAME:BUFF28; VAR CTRL:BUFF9; CURR:LINK_ARRAY);

This routine was designed to find a student's or faculty member's social security number given his last name or a subset of his last name. This is accomplished by searching the linked list of students or faculty for all the names that match the input name passed by the parameter NAME. when this routine is called, if more than one name is found to match, then all of the matching names are passed through the parameter CURR. Using the FMS scrolling feature, the names are presented to the user and the user can either make a selection or abandon the screen, in which case, a blank social security number is returned. The blank social security number indicates no names were found.

3. PROCEDURE GETVCQR (VAR VCQR: VCQR_ARRAY; CTRL: BUFF9);

This module reads the courses a student has entered into his education plan into an array of variable course quarter records, allowing up to 12Ø records. It formats the records into the array so they appear in the proper column when displayed by FMS.

4. PROCEDURE FILLEDPLAN (STDT:STDT_REC; VCQR:VCQR_ARRAY);

This procedure fills the FMS screen 'EDPLAN1' with the students name, rank, social security number, box number, primary afsc, education code and courses.

5. PROCEDURE GETADVISOR(VAR NAME:BUFF28; VAR CTRL:BUFF9);

The procedure GETADVISOR retrieves a faculty members social security number and master record by calling the procedure FINDNAME for all occurences of instructors with the same last name. If only one is found then the record is read and the information passed back, if more than one is found then the procedure DISPLAY_NAME is called to select one of them.

6. PROCEDURE GETSTUDENT(VAR NAME:BUFF28; VAR CTRL:BUFF9);

This procedure performs the same function as GETADVISOR except it is done for students names.

7. PROCEDURE UPTSEQ(VCQR: VCQR_ARRAY);

This routine deletes and re-creates the associated course files for the student. The array is sorted based upon the year and quarter the course will be taken, and then the copy of the courses in the CRSE_ARRAY are displayed and updated to reflect if the courses are SEQA, SEQB, MATH, THES, or WAIV type courses.

8. PROCEDURE ADDPLAN;

This procedure initially adds a students personal data and 6basic edplan into the system. The procedure UPTSEQ is called to write the course information to the database.

9. PROCEDURE UPTEDPLAN;

This procedure updates a students education plan by reading in his/her previous plan and displaying this to the user. Once the plan is read in, the courses are sorted based on the year and quarter and written back to the database. These courses are then copied into an array of records of type CRSE_ARRAY where the user defines them as SEQA, SEQB, THES, MATH, or WAIV courses. This routine allows the programmer to add edit checks while the courses are read in from the screen so that at the end of the session, all course entered are valid. The student is not deleted from the database and the course taken are archived in the Registrars office.

10. PROCEDURE DELEDPLAN;

This module deletes a students education plan from the data base but does not delete the student information from the STDT master file. The student is selected using the GETSTUDENT routine and the edplan is displayed to the user to insure that this is the plan they wish to delete. Only after the user has affirmed the decision will the plan be deleted.

11. PROCEDURE REVEDPLAN;

This routine functions much in the same way that the UPTEDPLAN routine does except it does not allow the user to udpate any information.

12. PROCEDURE LISTSTDT;

This routine lists all of the students who belong to a specific section by searching the linked lists and passing the selected names to the DISPLAY_NAME routine.

13. PROCEDURE PRINTCOURSE(COURSE:BUFF8; VAR CUM,QTR: INTEGER);

The PRINTCOURSE subroutine accepts as parameters the 8 character course code and the cumulative and quarterly hour totals. The procedure reads the master course file to obtain the title of the course and its credit hours, and then converts the number to an integer and adds them to the totals. The procedure then writes the course description to the file "RECORDS".

14. PROCEDURE PRINTSEQ(STDT: STDT_REC; SECTION:BUFF8);

This procedure produces the second page of the education plan report by printing the course sequence declaration as described by the student or faculty member.

15. PROCEDURE PRINTHEAD;

This procedure prints the header for each quarter of the education plan report or the header for each sequence.

16. PROCEDURE PRINTTAIL(VAR CUM,QTR: INTEGER);

This procedure prints the totals for each quarter and the cumulative totals.

17. PROCEDURE PRINTEDPLAN(SSAN: BUFF9; FLAG: BOOLEAN);

This procedure controls the printing of the first page of the education plan. It prints the course a student is taking by the quarter. It then reads the VCQR file, formats the output and prints the information. If a flag is sent to the routine as

TRUE, then the second page of the report is also printed which contains the sequence declarations.

18. PROCEDURE PTREDPLAN;

The procedure reads in the student's name whose edplan is requested. If the edplan is found, the user must enter whether a request for the second page is needed. The PRINTEDPLAN routine is then called.

19. PROCEDURE SECEDPLAN;

This procedure reads the section code from the user and gathers all of the students who belong in that section from the linked list. The names are then sent one by one to the procedure PRINTEDPLAN. The file 'RECORDS' is then printed and deleted.

Other Modules Under Development

In order to accomplish many of the tasks required in the EDPLAN program, many other forms of data were required in the database system. For instance, faculty records, section description, courses offered, department information, and several other items needed to be in the database in order for the variable records to be added. A section advisor record (FADV) could not be added if a faculty member with the specified social security number did not exist and the master section record did not exist. The basic maintenance modules for all of the master files were constructed at one time by the EENG 646 database class. A brief description of the modules constructed using the above programming techniques and some of those gathered from the WINTER 1985 EENG 646 class are described below.

1) FACTMOD.PAS: This program allowed the addition, update, deletion and review of all of the faculty master records.

2) SEQMOD.PAS: This module allows members to describe course sequences so students will be able to check their education plans to insure they have enough credits in the right courses to graduate. Course sequences can be added, updated, deleted, reviewed, and students can check their education plans against them.

3) BOOK.PAS: This file maintains the text book titles, order information, and re-order information. Currently, book information can be added, updated, deleted and reviewed.

<u>System Integration</u>

The integration of the system was simple and very flexible. The design of the system allowed the Data Base Administator several options. Each module (i.e STDTMOD, FACTMOD...etc) could be called individualy, allowing access to certain users by using the system priviledge codes. Several of the programs could be combined into a large program with a small main routine to drive them both . Or, a .COM (command file) could be used to control the calls to each of the modules.

The latter of the options was chosen because it allowed the mixing of modules with out re-compiling the programs and linking the data base system with the object modules. This also allowed the DBMS Manager to re-start the TOTAL DBMS system by use of a filename.COM file if the first call to the TOTAL DBMS failed to sign on. This solution relies on the version of VMS currently on the machine and could pose a problem in the future if a different operating system was used. A better solution would be to have a program which called the modules as external references much, using the programs as subroutines.

Test Plan

A formal test plan was used to test the lower level database modules in the beginnig to insure they worked properly with the database. Several conditions had to be met in order to insure their validity:

1. Each module had to work with an empty database.

2. Each module had to return a valid status code.

3. Each module was required to return or send valid data and to detect data sent in other than character form.

4. The linked list routines must work with an empty list of names, and up to the maximum number of names allowed in the database. Some special cases occured with one and two names in the system, but these were detected and handled.

The validation of the standard routines was conducted very early in the design phase. The modules were working at the time when the overall system was being coded. They were used as extensions to the PASCAL language and not as independent modules. The test plans followed by this thesis are identical to those developed by Pangman (2) and Bailor (7) in their theses on the same subject. The test plans can be found in Appendix I.

Summary

This chapter developed the details of coding and implementing a portion of the AFIT/ENG Database Design on the VAX 11/780 and the TOTAL DBMS. The education plan was chosen because it was under prototype development by the dep rtment, and has a great potential for decreasing the workload on the faculty. The actual configuration of the VAX 11/780 was described to aid in future development of the AFIT/EN Database and in the configuration of the system to facilitate transfer to another computer when the sytem becomes fully operational. The EDPLAN program was examined in detail to provide future programmers and analyst an insight into the program development methodology and design. It is hoped that any future attempts will follow the structured programming of the EDPLAN module during modifications of the code, and on any programs currently under developement.

Chapter 5

V. Conclusions and Recommendations

Introduction

This chapter presents the conclusions, problem areas, and recommendations derived from the results achieved by this study. Thoughout the course of this effort the human computer interface and user requirements were highlighted as well as the Software Development Life Cycle. From the beginning, the project proceeded upon the assumption that the software produced by this project would be used by the AFIT Shool of Engineering Department Electrical and Computer Engineering, if not the entire school, as a protoype for future development. The main effort was to produce a standard set of software products that adhered to the practices of good software engineering.

Conclusion

The intial part of this effort was to examine the requirements of the AFIT/ENG Department and compare these requirements with past theses efforts by Pangman (2), Allred(12), and Ricks(13). Some of the requirements had changed and are still changing. Information such as faculty personal data, course names, privacy act regulations, sequence requirments, degree requirements, and department changes will greatly effect the database structure and the associated computer programs.

The next step was to examine the functional requirements of the database and the user-interface requirements. The functional part of the database described what data was to be stored, how the data was stored, how the software was to interact, the overall structure, the reports to be generated, and the purpose of the system. At this point it was decided to define some standard functions that would act as auxiliary operations to the Pascal language. These functions and procedures would be available to all of the program segments and modules.

The human-interface requirements took into account how the information was presented to the user, the skill level of the average user, average age, and access priviledge. Using these critera the characteristics of a standard user were developed by which the interface systems could be taylored. A prototype of the education plan program (EDPLAN) was developed and tested on the incoming class of students where. They were required to enter their education plan on the system. The students were then interviewed to find their likes, dislikes, how long it took them to learn to use the system, and how "user-friendly" they thought it was. Using the feedback from the prototype system, a real education program was developed during the implementation stage of the Software Development Life Cycle.

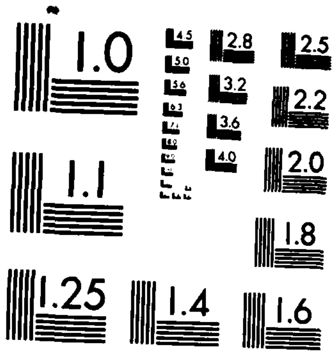
Having defined the requirments for the system, the next step was to perform a preliminary design of the AFIT/ENG Database System. The database schema, as defined by Pangman (2) and as modified by engineer Robert Ewing, was examined for relationships in the structure that were intended to be mapped into some type of query. These relationships were translated into further refinements of the requiements. The method for development chosen was a combination of top-down and bottom-up design. The overall picture was depicted in structure charts while the common routines were completely developed and tested. This provides a better foundation for development and use as tools for future modifications. The requirements for the system were well defined and mapped almost in a one-to-one correspondence into a design of the system.

The detailed design of the system was a refinement process from the preliminary design phase. There wasn't an exact point in the process where the preliminary design ended and the detailed design began. It was more of a smooth transition or evolution of software. Because of the complexity of the design, layers of software were developed following the stategy of the ISO network software. Each layer was developed to perform specific tasks at specific levels. Using dummy procedures as stubs, each layer was developed seperately from the others. This provided a front end processor to the TOTAL DBMS. The detailed design phase also involved defining the help level descriptions and tutorial screens which are important to the type of user described in the early phases of the development.

The next phase on the Software Development Life Cycle was the implementation phase or the coding phase. This phase involved taking the detailed design and putting it into a computer language (Pascal and FMS). Much of the software which was required to enter data into the database was developed from the EENG 646 DATABASE SYSTEM class and integrated by engineer Robert Ewing. These modules were inspected and changed to fit the design as described by this thesis. Additional software was developed to prove the design to be a solid one. User comments

AD-A164 051 IMPLEMENTATION OF THE AFIT/ENG FACULTY AND STUDENT DATABASE MANAGEMENT SYSTEM(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING 2/3

UNCLASSIFIED D A GAITROS DEC 85 AFIT/GCS/ENG/85D-5 F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

were used thoughout the implementation phase to fine tune the system to the needs and cognitive styles of the users. The EDPLAN program was developed to provide an example of the validity of the design and to stand as an example to guide the future software developments and modifications.

Using the standard database routines developed in this project, the availability of additional software from the EENG 646 class, and the use of the FMS (Forms Management System) caused the overall development of the system to proceeded very smoothly. The maintenance and any future developments should benefit from the extensive research provide by this and previous efforts. All software developed from this project is kept in the Information Sciences Laboratory as version 2.2 of the AFIT/ENG Faculty and Student Database Management System.

Problems Encountered

The most prevelent problem in the beginning was the learning curve required to use the TOTAL DBMS. The description of the schema, signing on to the database system, declaring external procedures, and the complex calling routines required to access the TOTAL DBMS were the main problems. This was addressed by putting the emphasis on the thesis in developing routines that all future programmers could use to perform these functions in an easy and timely manner.

The worst problem was caused when the operating system was changed from VMS version 3.6 to VMS verison 4.2. This changed the way TOTAL interfaced with the operating systems mail programs, the device definitions, and the logon commands required to run the software. The version of TOTAL currently running (2.2) is the same version that runs on the PDP-11 computer. A run time simulation program is used to run the PDP-11 code on the VAX 11/780. This software was also required to be updated when the operating system was changed.

The other problem associated with the change in operating systems was the change in the Pascal compile and the FMS definition program. The new operating system will not allow strings of different sizes to be assigned. Before, trunction was expected and blanks were padded at the end of the sending field. A long recieving field would flag a Pascal compile error. The new compiler would not flag an error if the sending field was to large but the new operating system would issue an error message and abort the program upon executing the assignment statement. The software was changed so all receiving and sending fields on assignments statements were of the same length. The new FMS driver program was compiled using the new version of the Pascal compiler using the following format:

```
$PASCAL/ENVIRONMENT FDVDEF.
```

Recommendations

The modular approach to the design of this system has evolved over several iterations of the requirements and preliminary design phases. It was the intention from the beginning that the maintenance phase of the Software Life Cycle would contribute much to the capabilities of the AFIT/ENG Database System. Future work should concentrate initially on the basic maintenance software that safeguards the data integrity and provides a "user friendly" interface to the system. The first large effort that should be put in to this database is the entry of data on to the database is the faculty information, thesis titles and authors, sequence and degree requirments, and complete scheduling information.

Currently, once education plans or grades are entered into the database, they are transfered by tape or manually to the admissions office for input into their system. A set of standards and methods should be negotiated between the two parties to accomplish this task automatically. This would also involve forming a set of rules and standards between the School of Logistics, School of Engineering and School of Civil Engineering. The main objective of this database is to reduce the workload humans have to perform and allow faculty to concentrate on academics instead of administrative duties.

Another recommendation is to phase in the ability to produce the Graduate Credit Record using the database system. A simple formula will be used to produce the credit record which calculates the grades by taking thesis courses and the highest grades from all graduate level course and calculating the grade point average. The user will be allowed to change these decisions by tagging and untagging specific course and viewing the change in credit hours and grade point average in a real time environment. Examples of this screen can be seen in Appendix E.

The department heads should draft a formal letter of responsibilities for the database. Issues such as who enters course grades, education plan changes, sequence requirements,and has access to privacy act information should be addressed before the database is totally integrated in the School of Engineering. Some these problems solved if a driver could be found for the Burroughs terminals that would allow the faculty members to have access to the database from their office. Currently, the Burroughs terminals cannot recognize the FMS graphics signals. Finding a VT1ØØ simulator for these systems would work, or changing the FDVDEF.PAS program that interacts with FMS would also be a solution.

The TOTAL DBMS system currently operates in a secondary mode to the VMS operating system. It must go through a PDP-11 simulator in order to operate a 16-bit system in a 32-bit addressing environment. A study should be conducted to the possibility of converting the current TOTAL DBMS to a version that operates under the VMS 4.2 operating system without an emulatotr. There currently exist a database called ULTRA (17) that would satisfy this need. The ULTRA Database Management System is compatible to TOTAL and provides a relational database front-end processor. This would by all indications improve the performance of the database by a considerable margin and add new capabilities to the system. This would also eliminate the need for an Ingress version of the database system.

The AFIT/ENG Database Management System is a large and complex software development effort that should proceed in a series of steps to insure proper implementation. The first milestone should be to finish and validate the student education plan program. During this phase, the software needed to maintain the database should be developed and tested. This would include faculty, course, student, thesis, and sequence information. The database works best when all of the relationships can be linked together. The next step is to completely implement the database within the Department of Electrical and Computer Engeering. The next steps would be to phase in the other departments within the School of Engineering and begin development on the management information programs. The success of the AFIT/ENG Database Management System for Faculty and Students will depend upon the software engineering and managerial abilites of those maintain the system. It is hoped that this thesis has provided a sound foundation for that success.

Bibliography

1. Date, C. J. An Introduction to Database Systems (Third Edition). Reading, Menlo Park, London, Amsterdam, Dom Mills, Sydney: Addison-Wesley Publishing Company, 1982

2. Pangman, Myron E. Complete Development and Implement AFIT/EN Database Management System, Masters Thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, 1983

3. Horowitz, Ellis and Sahni, Sartaj. Fundamentals of Data Structures In Pascal. Rockville:Computer Science Press, Inc., 1984

4. Peters, Lawrence J.Software Design: Methods and Techniques, Yourdon Press, New York,1981

5. Woffinden, Duard S. Lecture notes from EE9.93, Software Engineering, School of Engineering, Air Force Institute of Technology, Air University (AU), Wright-Patterson AFB, OH, 1985.

6. Sheil, B. A. "Power Tools for Programmers," Datamation Magazine, pages 19-30,(1983)

7. Bailor, Pail D. Development of a Data Base Management System Performance Monitor, Masters Thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, 1983

8. Lamont, Gary B. PH.D., Hadfield, Steven M.The Software Development Workbench: An integrated Software Development Environment. Unpublished Text. School of Engineering, Air Force Institute of Technology, Wright-Pattterson AFB, Ohio

9. Wiederhold, Gio. Database Design New York: McGraw Hill, Inc., 1977

10. Digital Equipment Corporation, VAX-11 FMS Software Reference Manual Order No. AA-J260A-TE. Digital Equipement Corporation, Maynard, Ma, September, 1980

11. Pressman, Roger S. Software Engineering: A Practitioners Approach. New York: McGraw Hill Book Company, 1982

12. Allred, Dean S. Consolidated AFIT Database,Masters Thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 1980 (AD 124376).

13. Ricks, Jeffrey S. and Robert S. Colburn, Ananlysis of Information Requirements and Design of the Consolidated AFIT Database and Information System (CADIS) with an AFIT/CI Implementation Design,Masters Thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson

AFB, OH, 1980 (ADA 124647)

14. Scharer, Laura.,"Pinpointing Requirements",Datamation Magazine, pages 139 - 151, April, 1981

15. Stevens, W. P., Myers, G. L. and Constantine, L.L., "Structured Design",IBM System Journal,Volume 13, Number 2,pages 328 through 352, 1974

16. Cincom Systems, Inc. TOTAL User's Guide. Digital Equipment Corporation, Canada, 1980

17. Cincom Systems, Inc.ULTRA Interactive Data Base. Company Brochure, Digital Equipment Corporation, Canada 1985

Vita

David Alan Gaitros was born 18 February 1955 in Decatur, Illinois. He graduated from high school in Cerro Gordo, Illinois in 1973 and attended Southern Illinois University, Carbondale, Illinois from which he recieved a Bachelor of Arts in Computer Science and Mathematics in May 1977. Upon graduation he was commissioned a Second Lieutenant in the United States Air Force through ROTC. He served as a systems analyst and source selection board recorder during his first assignement. His last assignment was as the Chief of the Test and Development Section, Utility Software Branch, Mission Support Directorate of the 552nd Airborne Warning and Control Division. He entered the Air Force Institute of Technology in June of 1984.

Permanent address: 302 East Carter St.
P.O Box 15
Cerro Gordo, Illinois
61818

APPENDIX A

AFIT/ENG FACULTY AND STUDENT DATABASE GENERATION

The following appendix is the source code used by the TOTAL DBMS to define the master files, variable files, data names, and relationships for the AFIT/ENG database. The data files, data names, data links, data sizes, number of logical records, record length, and records per blocks are all defined by the database generation. When the database is first generated in the form below, all files sizes are fixed and empty.

```
BEGIN-DATA-BASE-GENERATION     / THIS IS THE NEW VERSION OF THE /
DATA-BASE-NAME=AFITDB          / EXPANDED AFITDATABASE. /
SHARE-IO                       / DEVICES ARE LISTED AS RA81 INSTEAD /
IOAREA=MAS1                    / OF RK07.  THE TITLE OF THIS IS /
IOAREA=MAS2                    / AFITDB.DBG   RLE  16 AUG 1985 /
IOAREA=MAS3
IOAREA=MAS4
IOAREA=MAS5
IOAREA=MAS6
IOAREA=MAS7
IOAREA=MAS8
IOAREA=MAST
IOAREA=VAR1
IOAREA=VAR2
IOAREA=VAR3
IOAREA=VAR4
IOAREA=VAR5
IOAREA=VART
IOAREA=VARX

END-IO
```

```
--------------------- MASTER FACULTY FILE ----------------

BEGIN-MASTER-DATA-SET
DATA-SET-NAME=FACT          / FACULTY MASTER /
IOAREA=MAS5
MASTER-DATA
FACTROOT=8
FACTCTRL=9                  / FACULTY SSN /
FACTLKSE=8                  / SECTION LINK /
FACTLKSO=8                  / SOCIETY LINK /
FACTLKED=8                  / EDUCATION LINK /
FACTLKHA=8                  / HONORS & AWARDS LINK /
FACTLKIN=8                  / INTEREST LINK /
FACTLKCO=8                  / PUBLICATIONS & PRESENTAIONS LINK /
FACTLKTD=8                  / TDY LINK /
FACTLKCM=8                  / DEPT & COMMITTEE LINK /
FACTLKTH=8                  / LINK TO THESIS /
FACTLKCQ=8                  / GRADE LINK /
FACTLKPD=8                  / LINK TO PROFESSIONAL DEV QTR /
FACTLKTA=8                  / LINK TO THESIS ADVISOR /
FACTLKIS=8                  / LINK TO INSTRUCTOR STATISTICS /
FACTLKAD=8                  / LINK TO FACULTY ADVISOR /
FACTLKTC=8                  / LINK TO THESIS COMMITTEE MEMBER /
FACTNAME=28                 / FACULTY MEMBERS NAME,LAST,FIRST,MI/
FACTRANK=3                  / MIL/CIV RANK (O-OFFICER,G-CIV,NN-RANK)/
FACTSRVC=2                  / MILITARY SERVICE /
FACTDOCM=6                  / DATE OF COMMISSION /
FACTHDAT=6                  / DATE HIRED /
FACTSALR=5                  / SALARY /
FACTDOBI=6                  / DATE OF BIRTH /
FACTSEXX=1                  / SEX /
FACTAERO=10                 / AERO RATING /
FACTDTSC=6                  / DUTY AFSC /
FACTPMSC=6                  / PRIMARY AFSC /
FACTDORK=6                  / DATE OF RANK /
FACTYRSS=2                  / YEARS OF SERVICE /
FACTADDR=40                 / CURRENT ADDRESS -/
                            / NUMBER,STREET,CITY,STATE,ZIP/
FACTHPHN=7                  / HOME PHONE (EXCHANGE,EXTENSION)/
FACTEADR=40                 / EMERGENCY ADDRESS /
FACTMSTA=1                  / MARITAL STATUS /
FACTSPOS=12                 / SPOUSE FIRST NAME /
FACTSDOB=6                  / SPOUSE DATE OF BIRTH /
FACTNDEP=2                  / NUMBER OF DEPENDENTS /
FACTRACE=2                  / RACE /
FACTRELN=2                  / RELIGION /
FACTOFIC=12                 / OFFICE RM NUMBER /
FACTOPHN=7                  / OFFICE PHONE (EXCHANGE,EXTENSION)/
FACTLORG=50                 / LAST ORGANIZATION /
FACTTITL=50                 / LAST POSITION TITLE /
FACTDEPT=6                  / EXPECTED AFIT DEPARTURE DATE /
END-DATA

TOTAL-LOGICAL-RECORDS=1000
LOGICAL-RECORD-LENGTH=462
```

```
LOGICAL-RECORDS-PER-BLOCK=5     / BLOCKSIZE=256Ø /
DEVICE=RA81
DRIVE=3Ø,5ØØØ,DU3
END-MASTER-DATA-SET
```

```
---------------------- MASTER DEPARTMENT FILE ------------

BEGIN-MASTER-DATA-SET
DATA-SET-NAME=DEPT     / DEPARTMENT MASTER /
IOAREA=MAS2
MASTER-DATA
DEPTROOT=8
DEPTCTRL=4             / DEPT CODE /
DEPTLKCM=8             / DEPT & COMMITTEE LINK /
DEPTNAME=20
END-DATA

TOTAL-LOGICAL-RECORDS=200
LOGICAL-RECORD-LENGTH=40
LOGICAL-RECORDS-PER-BLOCK=12    / BLOCKSIZE=512 /
DEVICE=RA81
DRIVE=31,5000,DU3
END-MASTER-DATA-SET


------------------- MASTER STUDENT FILE -----------------

BEGIN-MASTER-DATA-SET
DATA-SET-NAME=STDT  / STUDENT MASTER /
IOAREA=MAS4
MASTER-DATA
STDTROOT=8          / REQUIRED /
STDTCTRL=9          / STUDENT SOCIAL SECURITY /
STDTLKAW=8          / LINK TO VHAW (AWARDS) /
STDTLKCR=8          / LINK TO CRSE (COURSE) /
STDTLKDG=8          / LINK TO VEDU (VARIABLE EDUCATION) FILE /
STDTLKCQ=8          / LINK TO VCQR (AFIT COURSES & CREDITS /
STDTLKTH=8          / LINK TO THESIS FILE /
STDTLKSE=8          / LINK TO SECTION FILE /
STDTLKTA=8          / LINK TO THESIS ADVISOR /
STDTLKIS=8          / LINK TO INSTRUCTOR STATISTICS /
STDTLKAD=3          / LINK TO FACULTY ADVISOR /
STDTLKCM=8          / LINK TO THESIS COMMITTEE MEMBER /
STDTSEQN=3          / MASTER SEQUENCE CONTROL NUMBER */
STDTNAME=28         / STUDENT NAME (LAST,FIRST,MI) /
STDTRANK=3          / MIL/CIV RANK(O-OFFICER,G-CIV,NN-RANK /
STDTGRAD=1          / HAS STUDENT ALREADY GRADUATED/LEFT AFIT? /
STDTSRVC=2          / MILITARY SERVICE /
STDTAERO=10         / AERO RATING /
STDTDORK=6          / DATE OF RANK /
STDTDOCM=6          / DATE OF COMMISSION /
STDTYRSS=2          / YEARS OF SERVICE /
STDTSEXX=1          / SEX /
STDTBOXN=4          / BOX NUMBER /
STDTDTSC=6          / DUTY AFSC /
STDTPMSC=6          / PRIMARY AFSC /
STDTADDR=40         / CURRENT ADDRESS /
STDTEADR=40         / EMERGENCY ADDRESS /
STDTHMPH=7          / HOME PHONE NUMBER /
STDTDTPH=7          / DUTY PHONE NUMBER /
```

```
STDTEDCD=5          / EDUCATION CODE /
STDTDOBH=6          / DATE OF BIRTH /
STDTPOBH=40         / PLACE OF BIRTH /
STDTMSTA=1          / MARITAL STATUS - M(MARRIED),D(IVORCED),ETC
STDTSPOS=12         / SPOUSE FIRST NAME /
STDTSDOB=6          / SPOUSE DATE OF BIRTH /
STDTMSPS=1          / MILITARY SPOUSE /
STDTNDEP=2          / NUMBER OF DEPENDENTS /
STDTRACE=2          / RACE /
STDTRELN=2          / RELIGION /
STDTLCMD=5          / LOSING COMMAND /
STDTLORG=50         / LAST ORGANIZATION /
STDTTITL=50         / LAST POSITION TITLE /
STDTDURN=2          / DURATION AT LAST DUTY ASSIGNMENT /
END-DATA

TOTAL-LOGICAL-RECORDS=5000
LOGICAL-RECORD-LENGTH=460
LOGICAL-RECORDS-PER-BLOCK=5 BLOCKSIZE=2300 MAS1=2560
DEVICE=RA81
DRIVE=26,5000,DU3

END-MASTER-DATA-SET


 --------------------- MASTER THESIS NUMBER FILE -----------

BEGIN-MASTER-DATA-SET
DATA-SET-NAME=THES           / THESIS NUMBER MASTER /
IOAREA=MAS6

MASTER-DATA
THESROOT=8
THESCTRL=10                  / THESIS CATALOGING NUMBER /
THESLKTH=8                   / LINK TO VARIABLE THESIS TITLE FILE /
THESLKTA=8                   / LINK TO THESIS ADVISOR /
THESLKTC=8                   / LINK TO THESIS COMM MEMBER FILE /
THESTITL=50                  / THESIS TITLE /
THESSPON=50                  / THESIS SPONSOR /
THESLOCN=50                  / THESIS LOCATION /
THESCLAS=12                  / THESIS CLASSIFICATION /
THESNAME=28                  / STUDENT NAME FOR ARCHIVE PURPOSES /
END-DATA

TOTAL-LOGICAL-RECORDS=5000
LOGICAL-RECORD-LENGTH=232
LOGICAL-RECORDS-PER-BLOCK=4
DEVICE=RA81
DRIVE=22,5000,DU3

END-MASTER-DATA-SET
```

```
 --------------------- MASTER SECTION NUMBER FILE --------

BEGIN-MASTER-DATA-SET
DATA-SET-NAME=SECT        / SECTION NUMBER MASTER FILE /
IOAREA=MAS6

MASTER-DATA
SECTROOT=8
SECTCTRL=8                / SECTION NUMBER (EX., GCS-84D) /
SECTLKSE=8                / LINK TO SECTION LEADER FILE /
SECTLKAD=8                / LINK TO FACULTY ADVISOR /
SECTLSSN=9                / SECTION LEADER SSSN /
SECTGRDT=6                / GRADUATION DATE /
SECTENDT=6                / ENTRY DATE /
SECTNRSN=3                / NUMBER OF STUDENTS IN SECTION /
END-DATA

TOTAL-LOGICAL-RECORDS=500
LOGICAL-RECORD-LENGTH=56
LOGICAL-RECORDS-PER-BLOCK=9
DEVICE=RA81
DRIVE=23,5000,DU3

END-MASTER-DATA-SET


 --------------------- MASTER COURSE FILE -----------------

BEGIN-MASTER-DATA-SET
DATA-SET-NAME=MCRS             / COURSE DATA MASTER FILE /
IOAREA=MAS3

MASTER-DATA
MCRSROOT=8                     / REQUIRED BY TOTAL /
MCRSCTRL=8                     / COURSE NUMBER /
MCRSCRHR=1                     / COURSE CREDIT HOURS/
MCRSLKCQ=8                     / LINK TO QUARTER /
MCRSLKRQ=8                     / LINK TO REQUISITE /
MCRSLKCB=8                     / LINK TO BOOK TITLE /
MCRSLKSC=8                     / LINK TO SCHD /
MCRSLKSS=8                     / LINK TO COURSE SEQUENCE /
MCRSLKIS=8                     / LINK TO INSTRUCTOR STATISTICS /
MCRSLCHR=1                     / COURSE LECTURE HOURS DATA /
MCRSLBHR=1                     / COURSE LAB HOUR DATA /
MCRSSZLM=2                     / SIZE LIMIT DATA /
MCRSTITL=50                    / TITLE DATA /
MCRSREST=1                     / RESTRICTED (FROM GRAD REQ) COURSE /
END-DATA
DEVICE=RA81
TOTAL-LOGICAL-RECORDS=2000
LOGICAL-RECORD-LENGTH=130
LOGICAL-RECORDS-PER-BLOCK=7     / BLOCKSIZE = 1024 /
DRIVE=12,5000,DU3
END-MASTER-DATA-SET
```

```
---------------------- MASTER QUARTER FILE ----------------

BEGIN-MASTER-DATA-SET
DATA-SET-NAME=MQTR              / QUARTER DATA MASTER FILE /
IOAREA=MAS7

MASTER-DATA
MQTRROOT=8                      / REQUIRED BY TOTAL /
MQTRCTRL=4                      / QUARTER NUMBER /
MQTRLKCT=8                      / LINK TO COURSE /
MQTRLKPD=8                      / LINK TO PROF DEV QTR /
MQTRSTDT=6                      / QUARTER START DATE(DAY,MO,YR) /
MQTRSPDT=6                      / QUARTER STOP DATE (DAY,MO,YR) /
END-DATA

DEVICE=RA81
TOTAL-LOGICAL-RECORDS=1ØØØ
LOGICAL-RECORD-LENGTH=4Ø
LOGICAL-RECORDS-PER-BLOCK=12
DRIVE=13,5ØØØ,DU3
END-MASTER-DATA-SET


---------------------- MASTER BOOK FILE -------------------

BEGIN-MASTER-DATA-SET
DATA-SET-NAME=MBKT              / BOOK INFORMATION MASTER FILE /
IOAREA=MAS1
MASTER-DATA
MBKTROOT=8                      / REQUIRED BY TOTAL /
MBKTCTRL=4Ø                     / BOOK TITLE NAME /
MBKTLKBK=8                      / LINK TO COURSE THRU VCBK /
MBKTLKNO=8                      / LINK TO NUMBER ORDERED /
MBKTATHR=28                     / BOOK AUTHOR NAME (LAST,FIRST,MI) /
MBKTPUBL=28                     / BOOK PUBLISHER NAME /
MBKTNAVL=6                      / NUMBER OF BOOKS AVAILABLE /
MBKTPRCE=4                      / BOOK PRICE /
END-DATA

DEVICE=RA81                       PHYSICAL ENVIRONMENT
TOTAL-LOGICAL-RECORDS=5ØØØ
LOGICAL-RECORD-LENGTH=13Ø
LOGICAL-RECORDS-PER-BLOCK=3
DRIVE=14,5ØØØ,DU3
END-MASTER-DATA-SET
```

```
 --------------------- MASTER ORDER FILE ------------------

BEGIN-MASTER-DATA-SET
DATA-SET-NAME=MORD          / BOOK ORDERING INFORMATION MASTER FILE /
IOAREA=MAS1
MASTER-DATA
MORDROOT=8                  / REQUIRED BY TOTAL /
MORDCTRL=7                  / MASTER ORDER NUMBER /
MORDLKBO=8                  / LINK TO BOOK THRU VNMO /
MORDORDT=6                  / ORDER NUMBER /
MORDDUDT=6                  / DUE DATE /
MORDCMPY=2Ø                 / COMPANY /
MORDADDR=4Ø                 / COMPANY ADDRESS /
MORDPHNE=1Ø                 / COMPANY PHONE NUMBER WITH AREA CODE /
END-DATA

DEVICE=RA81                     PHYSICAL ENVIRONMENT
TOTAL-LOGICAL-RECORDS=1ØØØ
LOGICAL-RECORD-LENGTH=1Ø6
LOGICAL-RECORDS-PER-BLOCK=4
DRIVE=15,5ØØØ,DU3
END-MASTER-DATA-SET


 --------------------- MASTER CLASS TIME FILE -----------
BEGIN-MASTER-DATA-SET
DATA-SET-NAME=TIME     / MASTER CONTAINING COURSE TIMES /
IOAREA=MAST
MASTER-DATA
TIMEROOT=8
TIMECTRL=4             / MILITARY CLOCK TIME /
TIMELKSC=8             / LINKPATH TO SCHEDULE FILE /
END-DATA

TOTAL-LOGICAL-RECORDS=36ØØ
LOGICAL-RECORD-LENGTH=2Ø
LOGICAL-RECORDS-PER-BLOCK=25
DEVICE=RA81
DRIVE=Ø1,5ØØØ,DU3
END-MASTER-DATA-SET


 -------------------- MASTER BUILDING/ROOM FILE ----------


BEGIN-MASTER-DATA-SET
DATA-SET-NAME=BLRM    / MASTER CONTAINING ROOMS AND /
                      /    BLDG #'S FOR SCHEDULING /
IOAREA=MAST
MASTER-DATA
BLRMROOT=8
BLRMCTRL=8            / BUILDING AND ROOM NUMBER /
BLRMLKSC=8            / LINKPATH TO SCHEDULE FILE /
BLRMLKCL=8            / LINKPATH TO CLASSROOM FILE /
```

```
END-DATA

TOTAL-LOGICAL-RECORDS=5000
LOGICAL-RECORD-LENGTH=32
LOGICAL-RECORDS-PER-BLOCK=16
DEVICE=RA81
DRIVE=02,5000,DU3
END-MASTER-DATA-SET

 --------------------- MASTER ROOM CAPACITY FILE ----------

BEGIN-MASTER-DATA-SET
DATA-SET-NAME=CPTY      / MASTER CONTAINING ROOM CAPACITIES /
IOAREA=MAST
MASTER-DATA
CPTYROOT=8
CPTYCTRL=4              / CAPACITY NUMBER /
CPTYLKCL=8              / LINKPATH TO CLASSROOM FILE /
END-DATA

TOTAL-LOGICAL-RECORDS=700
LOGICAL-RECORD-LENGTH=20
LOGICAL-RECORDS-PER-BLOCK=25
DEVICE=RA81
DRIVE=03,5000,DU3
END-MASTER-DATA-SET

 ---------------------- MASTER DAY SCHEDULING FILE --------

BEGIN-MASTER-DATA-SET
DATA-SET-NAME=DAYS       / MASTER WHICH CONTAINS DAYS OF WEEK /
IOAREA=MAST
MASTER-DATA
DAYSROOT=8
DAYSCTRL=4               / DAY OF THE WEEK /
DAYSLKSC=8               / LINKPATH TO SCHEDULE FILE /
END-DATA

TOTAL-LOGICAL-RECORDS=30
LOGICAL-RECORD-LENGTH=20
LOGICAL-RECORDS-PER-BLOCK=25
DEVICE=RA81
DRIVE=04,5000,DU3
END-MASTER-DATA-SET
```

```
 ----------- MASTER SEQUENCE FILE ------------

BEGIN-MASTER-DATA-SET
DATA-SET-NAME=MSSF       / COURSE SEQUENCES MASTER /
IOAREA=MAS8
MASTER-DATA
MSSFROOT=8
MSSFCTRL=3               / COURSE SEQUENCE NUMBER /
MSSFSEQN=40              / SEQUENCE NAME /
MSSFLKSS=8               / VARIABLE SEQUENCE FILE LINK /
END-DATA

TOTAL-LOGICAL-RECORDS=5000
LOGICAL-RECORD-LENGTH=60
LOGICAL-RECORDS-PER-BLOCK=8    / BLOCKSIZE=512 /
DEVICE=RA81
DRIVE=42,5000,DU3
END-MASTER-DATA-SET


 ------------------ MASTER DEGREE REQUIREMENTS FILE -----------------

BEGIN-MASTER-DATA-SET
DATA-SET-NAME=MDEG       / DEGREE REQUIREMENTS MASTER /
IOAREA=MAS8
MASTER-DATA
MDEGROOT=8
MDEGCTRL=2               / NUMBER IDENTIFYING TYPE GRAD DEGREE /
MDEGNAME=40              / NAME OF TYPE OF DEGREE /
MDEGLKCR=8               / COURSE LINK /
MDEGLKSS=8               / COURSE SEQUENCE LINK /
END-DATA

TOTAL-LOGICAL-RECORDS=5000
LOGICAL-RECORD-LENGTH=66
LOGICAL-RECORDS-PER-BLOCK=7    / BLOCKSIZE=512 /
DEVICE=RA81
DRIVE=43,5000,DU3
END-MASTER-DATA-SET
```

```
BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=VEDU       / EDUCATION VAR FILE /
IOAREA=VAR1
BASE-DATA
VEDUFSSN=9               / FACULTY SSN /
FACTLKED=8=VEDUFSSN      / LINK TO FACULTY MASTER /
VEDUSTDT=9               / STUDENT SSN /
STDTLKDG=8=VEDUSTDT      / LINK TO STUDENT MASTER /
VEDUUNIV=40              / INSTITUTION (UNIVERSITY) ATTENDED /
VEDUDEGR=40              / DEGREE EARNED /
VEDUYEAR=4               / YEAR DEGREE AWARDED /
END-DATA

DEVICE=RA81
DRIVE=32,5000,DU3
TOTAL-LOGICAL-RECORDS=5000
LOGICAL-RECORD-LENGTH=120
LOGICAL-RECORDS-PER-BLOCK=4    / BLOCKSIZE=512 /
END-VARIABLE-ENTRY-DATA-SET

 --------------------- VARIABLE FACULTY SOCIETY FILE -----

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=FSOC       / SOCIETY VAR FILE /
IOAREA=VAR1
BASE-DATA
FSOCFSSN=9               / FACULTY SSN /
FACTLKSO=8=FSOCFSSN
FSOCSOCY=40              / SOCIETIES TO WHICH INDIVIDUAL BELONGS  /
FSOCDUM1=8               / PADDING TO INCREASE REC LENGTH /
END-DATA

DEVICE=RA81
DRIVE=33,5000,DU3
TOTAL-LOGICAL-RECORDS=10000
LOGICAL-RECORD-LENGTH=66
LOGICAL-RECORDS-PER-BLOCK=7    / BLOCKSIZE=512 /
END-VARIABLE-ENTRY-DATA-SET

 -------- VARIABLE FACULTY DEPT & COMMITTEE FILE --------

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=FCMT               / DEPT AND COMMITTEE VAR FILE /
IOAREA=VAR1
BASE-DATA
FCMTCODE=2
FCMTFSSN=9                       / FACULTY SSN=9 /
FACTLKCM=8=FCMTFSSN              / LINK TO FACULTY MASTER SSN /
FCMTDCOD=4                       / DEPARTMENT DCOD /
DEPTLKCM=8=FCMTDCOD              / LINK TO DEPT MASTER DEPT CODE /
FCMTDATA=14                      / REDEFINED DATA AREA LENGTH /
```

```
RECORD-CODE=DP                   / COMMITTEE MEMBERSHIP /
RECORD-CODE=CM                   / ADDITIONAL COMMITTEE MEMBERSHIPS /
END-DATA

DEVICE=RA81
DRIVE=34,5000,DU3
TOTAL-LOGICAL-RECORDS=5000
LOGICAL-RECORD-LENGTH=46
LOGICAL-RECORDS-PER-BLOCK=11     / BLOCKSIZE=512 /
END-VARIABLE-ENTRY-DATA-SET


 --------------- VARIABLE HONORS & AWARDS FILE -----

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=VHAW      / HONORS & AWARDS VAR FILE /
IOAREA=VAR1
BASE-DATA
VHAWCODE=2
VHAWFSSN=9              / FACULTY SSN /
FACTLKHA=8=VHAWFSSN     / LINK TO FACULTY SSN /
VHAWSTDT=9              / STUDENT SOCIAL SECURITY NUMBER /
STDTLKAW=8=VHAWSTDT     / LINK TO STDT (STUDENT MASTER) /
VHAWDATA=16             / REDEFINING DATA LENGTH AREA /
RECORD-CODE=HN          / HONOR AREA /
RECORD-CODE=AW          / AWARD AREA /
END-DATA

DEVICE=RA81
DRIVE=36,5000,DU3
TOTAL-LOGICAL-RECORDS=10000
LOGICAL-RECORD-LENGTH=54
LOGICAL-RECORDS-PER-BLOCK=9    / BLOCKSIZE=512 /
END-VARIABLE-ENTRY-DATA-SET


 --------------- VARIABLE FACULTY INTERESTS FILE ----------

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=FINT       / INTEREST AREA VAR FILE /
IOAREA=VAR1
BASE-DATA
FINTFSSN=9               / FACULTY SSN /
FACTLKIN=8=FINTFSSN      / LINK TO FACULTY SSN /
FINTAREA=15              / AREA OF INTEREST /
END-DATA

DEVICE=RA81
DRIVE=37,5000,DU3
TOTAL-LOGICAL-RECORDS=10000
LOGICAL-RECORD-LENGTH=34
LOGICAL-RECORDS-PER-BLOCK=16    / BLOCKSIZE=512 /
END-VARIABLE-ENTRY-DATA-SET
```

```
 -------------- VARIABLE FACULTY PUBS AND PRESENTATIONS ---

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=FCOM       / PUBLICATIONS & PRESENTATIONS VAR FILE /
IOAREA=VAR1
BASE-DATA
FCOMCODE=2
FCOMFSSN=9               / FACULTY SSN /
FACTLKCO=8=FCOMFSSN      / LINK TO FACULTY FSSN /
FCOMDATA=61              / REDEFINED DATA AREA LENGTH /
RECORD-CODE=PB           / PUBLICATION DATA AREA /
RECORD-CODE=PR           / PRESENTATIONS DATA AREA /
END-DATA

DEVICE=RA81
DRIVE=38,5000,DU3
TOTAL-LOGICAL-RECORDS=10000
LOGICAL-RECORD-LENGTH=82
LOGICAL-RECORDS-PER-BLOCK=6
END-VARIABLE-ENTRY-DATA-SET


 --------------- VARIABLE FACULTY TDY FILE ---------------

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=FTDY        / TDY VAR FILE /
IOAREA=VAR1
BASE-DATA
FTDYFSSN=9                / FACULTY SSN/
FACTLKTD=8=FTDYFSSN       / LINK TO FACULTY SSN /
FTDYCOST=7                / COST OF TDY DATA IN THIS FILE /
FTDYDEST=20               / DESTINATION /
FTDYBDAT=6                / BEGIN DATE /
FTDYEDAT=6                / END DATA /
END-DATA

DEVICE=RA81
DRIVE=39,5000,DU3
TOTAL-LOGICAL-RECORDS=10000
LOGICAL-RECORD-LENGTH=58
LOGICAL-RECORDS-PER-BLOCK=9       / BLOCKSIZE=512 /
END-VARIABLE-ENTRY-DATA-SET

 --------------- VARIABLE STUDENT COURSE FILE -------------

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=CRSE        / COURSE VARIABLE /
IOAREA=VAR4
BASE-DATA
CRSESTDT=9                / STUDENT SOCIAL SECURITY NUMBER /
STDTLKCR=8=CRSESTDT       / LINK TO STUDENT (STUDENT MASTER) /
```

```
CRSEMDEG=2                / TYPE GRAD DEGREE (NUMBER - FROM MDEG) /
MDEGLKCR=8=CRSEMDEG       / LINK TO MASTER DEGREE REQUIREMENTS /
CRSENUMB=8                / COURSE NUMBER /
CRSENAME=20               / COURSE NAME /
CRSEGRAD=2                / COURSE GRADE /
CRSEBEGN=4                / QUARTER STUDENT TOOK OR WILL TAKE COURSE /
CRSECOLL=30               / COLLEGE ATTENDED /
CRSEWAIV=1                / COURSE WAIVED? (Y/N) /
END-DATA
DEVICE=RA81
TOTAL-LOGICAL-RECORDS=50000
LOGICAL-RECORD-LENGTH=94
LOGICAL-RECORDS-PER-BLOCK=11   / BLOCKSIZE = 1024 /
DRIVE=27,10500,DU3
END-VARIABLE-ENTRY-DATA-SET


 --------------- VARIABLE THESIS TITLE FILE ---------------

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=THTL             / VAR FILE FOR STUDENT THESIS DATA /
IOAREA=VAR4

BASE-DATA
THTLTHES=10                    / DEPARTMENT THESIS NUMBER /
THESLKTH=8=THTLTHES            / LINK TO THES /
THTLFACT=9                     / FACULTY ADVISOR FSSN /
FACTLKTH=8=THTLFACT            / LINK TO FACULTY FSSN /
THTLSTDT=9                     / STUDENT SSSN /
STDTLKTH=8=THTLSTDT            / LINK TO STUDENT SSSN /
END-DATA

TOTAL-LOGICAL-RECORDS=5000
LOGICAL-RECORD-LENGTH=52
LOGICAL-RECORDS-PER-BLOCK=10     / BLOCKSIZE 520 /
DEVICE=RA81
DRIVE=24,5000,DU3

END-VARIABLE-ENTRY-DATA-SET


 --------------- VARIABLE SECTION LEADER FILE -------------

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=SECL           / VARIABLE SECTION LEADER FILE /
IOAREA=VAR3

BASE-DATA
SECLSECT=8                   / RELATED TO SECT (SECTION NUMBER) /
SECTLKSE=8=SECLSECT          / LINK TO SECT /
SECLSTDT=9                   / STUDENT SSSN /
STDTLKSE=8=SECLSTDT          / LINK TO STUDENT SSSN /
SECLFACT=9                   / FACULTY FSSN /
FACTLKSE=8=SECLFACT          / LINK TO FACULTY FSSN /
END-DATA
```

```
TOTAL-LOGICAL-RECORDS=5000
LOGICAL-RECORD-LENGTH=52
LOGICAL-RECORDS-PER-BLOCK=10
DEVICE=RA81
DRIVE=25,5000,DU3

END-VARIABLE-ENTRY-DATA-SET


 --------------------- VARIABLE QUARTER FILE ---------------


BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=VCQR            / VARIABLE QUARTER FILE /
IOAREA=VAR5
BASE-DATA
VCQRCODE=2
VCQRNMBR=8                    / COURSE NUMBER CONTROL FIELD /
MCRSLKCQ=8=VCQRNMBR           / LINK FROM MASTER COURSE /
VCQRIDEN=4                    / QUARTER IDENT CONTROL FIELD /
MQTRLKCT=8=VCQRIDEN           / LINK FROM MASTER QUARTER /
VCQRDATA=20                   / REDEFINED VAR QUARTER DATA /
RECORD-CODE=QC                / CODE IS QC (FOR COURSE) /
RECORD-CODE=QS                / CODE IS QS (FOR STUDENT) /
STDTLKCQ=8=VCQRSSSN           / LINK TO MASTER STUDENT /
RECORD-CODE=FQ                / CODE IS FQ (FOR FACULTY) /
FACTLKCQ=8=VCQRFSSN           / LINK TO FACULTY SSN /
END-DATA

DEVICE=RA81
TOTAL-LOGICAL-RECORDS=50000
LOGICAL-RECORD-LENGTH=50
LOGICAL-RECORDS-PER-BLOCK=10
DRIVE=16,5000,DU3
END-VARIABLE-ENTRY-DATA-SET


 --------------------- VARIABLE REQUISITE FILE -------------

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=VREQ           / VARIABLE REQUISITE FILE /
IOAREA=VARX
BASE-DATA
VREQCODE=2                   /  CODED RECORD FOR REQUISITE /
VREQNMBR=8                   /  COURSE NUMBER CONTROL FIELD /
MCRSLKRQ=8=VREQNMBR          /  LINK FROM MASTER COURSE /
VREQDATA=14                  /  REDEFINED REQUISITE DATA /
RECORD-CODE=CR               /  CODE IS COREQUISITE /
RECORD-CODE=PR               /  CODE IS PREREQUISITE /
END-DATA

DEVICE=RA81
TOTAL-LOGICAL-RECORDS=5000
LOGICAL-RECORD-LENGTH=32
```

```
LOGICAL-RECORDS-PER-BLOCK=18
DRIVE=17,5000,DU3
END-VARIABLE-ENTRY-DATA-SET

------ VARIABLE BOOK LINK FILE -------------

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=VCBK             / VARIABLE BOOK LINK FILE /
IOAREA=VARX
BASE-DATA
VCBKNMBR=8                     / COURSE NUMBER CONTROL FIELD /
MCRSLKCB=8=VCBKNMBR            / LINK FROM MASTER COURSE /
VCBKTITL=40                    / BOOK TITLE CONTROL FIELD /
MBKTLKBK=8=VCBKTITL            / LINK FROM MASTER BOOK TITLE /
END-DATA

DEVICE=RA81
TOTAL-LOGICAL-RECORDS=5000
LOGICAL-RECORD-LENGTH=64
LOGICAL-RECORDS-PER-BLOCK=8
DRIVE=19,5000,DU3
END-VARIABLE-ENTRY-DATA-SET

 ---------------------- VARIABLE NUMBER ORDERED FILE --------

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=VNMO             / VARIABLE NUMBER OF TEXTS/
                               / ORDERED FILE /
IOAREA=VARX
BASE-DATA
VNMOTITL=40                    / BOOK TITLE CONTROL FIELD /
MBKTLKNO=8=VNMOTITL            / LINK FROM MASTER BOOK TITLE /
VNMONMBR=7                     / ORDER NUMBER CONTROL FIELD /
MORDLKBO=8=VNMONMBR            / LINK FROM MASTER ORDER NUMBER /
VNMONORD=3                     / NUMBER ORDERED DATA ITEM /
END-DATA

DEVICE=RA81
TOTAL-LOGICAL-RECORDS=5000
LOGICAL-RECORD-LENGTH=68
LOGICAL-RECORDS-PER-BLOCK=7
DRIVE=20,5000,DU3
END-VARIABLE-ENTRY-DATA-SET

 --------------- VARIABLE CLASS SCHEDULE FILE -------------

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=SCHD        / VAR FILE TO CONTAIN CLASS DATA /
IOAREA=VART
BASE-DATA
SCHDNSTD=3              / NUMBER OF STUDENTS IN CLASS /
SCHDNMBR=8              / COURSE NUMBER /
```

```
MCRSLKSC=8=SCHDNMBR     / LINKPATH TO COURSE NUMBER FILE /
SCHDDAYS=4              / DAY CLASS MEETS /
DAYSLKSC=8=SCHDDAYS     / LINKPATH TO DAYS FILE /
SCHDTIME=4              / TIME CLASS STARTS /
TIMELKSC=8=SCHDTIME     / LINKPATH TO TIME FILE /
SCHDBLRM=8              / BUILDING AND ROOM NUMBER /
BLRMLKSC=8=SCHDBLRM     / LINKPATH TO BUILDING AND ROOM FILE /
SCHDFNTM=4              / CLASS FINISH TIME /
END-DATA

DEVICE=RA81
TOTAL-LOGICAL-RECORDS=5ØØØ
LOGICAL-RECORD-LENGTH=64
LOGICAL-RECORDS-PER-BLOCK=8
DRIVE=Ø5,5ØØØ,DU3
END-VARIABLE-ENTRY-DATA-SET


 --------------- VARIABLE CLASSROOM FILE ------------------

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=CLSR      / FILE TO CONTAIN CLASSROOM DATA /
IOAREA=VART
BASE-DATA
CLSRBLRM=8              / BUILDING AND ROOM NUMBER /
BLRMLKCL=8=CLSRBLRM     / LINKPATH TO BUILDING AND ROOM FILE /
CLSRCPTY=4              / CAPACITY OF ROOM /
CPTYLKCL=8=CLSRCPTY     / LINKPATH TO CAPACITY FILE /
CLSREQPT=2              / TYPE(S) OF EQUIPMENT IN ROOM /
CLSRTYPE=3              / CODE FOR TYPE OF ROOM /
CLSRCFLG=1              / CODE FOR SECURITY CLASSIFICATION LEVEL OF ROOM /
END-DATA

DEVICE=RA81
TOTAL-LOGICAL-RECORDS=5ØØØ
LOGICAL-RECORD-LENGTH=34
LOGICAL-RECORDS-PER-BLOCK=14
DRIVE=Ø6,5ØØØ,DU3
END-VARIABLE-ENTRY-DATA-SET


 --------------- VARIABLE THESIS COMMITTEE MEMBER FILE -------------

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=TCMF         / VAR THESIS COMM MEMBER FILE /
IOAREA=VAR3

BASE-DATA
TCMFFACT=9                 / FACULTY SSN /
FACTLKTC=8=TCMFFACT        / LINK TO FACULTY FSSN /
TCMFSTDT=9                 / STUDENT SSN /
STDTLKCM=8=TCMFSTDT        / LINK TO STUDENT SSSN /
TCMFTHES=1Ø                / DEPARTMENT THESIS NUMBER /
THESLKTC=8=TCMFTHES        / LINK TO THESIS DEPARTMENT NUMBER /
END-DATA
```

```
TOTAL-LOGICAL-RECORDS=5000
LOGICAL-RECORD-LENGTH=54
LOGICAL-RECORDS-PER-BLOCK=9
DEVICE=RA81
DRIVE=44,5000,DU3

END-VARIABLE-ENTRY-DATA-SET


 --------------- VARIABLE FACULTY ADVISER FILE -------------

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=FADV             / VARIALE FACULTY ADVISOR FILE /
IOAREA=VAR3

BASE-DATA
FADVSECT=8                     / SECT CTRL (SECT NUMBER) /
SECTLKAD=8=FADVSECT            / LINK TO SECT /
FADVSTDT=9                     / STUDENT SSSN /
STDTLKAD=8=FADVSTDT            / LINK TO STUDENT SSSN /
FADVFACT=9                     / FACULTY FSSN /
FACTLKAD=8=FADVFACT            / LINK TO FACULTY FSSN /
END-DATA

TOTAL-LOGICAL-RECORDS=5000
LOGICAL-RECORD-LENGTH=52
LOGICAL-RECORDS-PER-BLOCK=10
DEVICE=RA81
DRIVE=45,5000,DU3

END-VARIABLE-ENTRY-DATA-SET


 --------------- VARIABLE INSTRUCTOR STATISTICS FILE -------------

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=VINS      / INST STATS FOR USE WITH/
                        / INSTRUCTOR LOAD DATA /
IOAREA=VAR1

BASE-DATA
VINSSTDT=9              / STUDENT SSSN /
STDTLKIS=8=VINSSTDT     / LINK TO STUDENT SSSN /
VINSNMBR=8              / MCRS CTRL (COURSE NUMBER) /
MCRSLKIS=8=VINSNMBR     / LINK TO MCRS (MASTER COURSE FILE) /
VINSFACT=9              / FACULTY SSSN /
FACTLKIS=8=VINSFACT     / LINK TO FACULTY FSSN /
END-DATA

TOTAL-LOGICAL-RECORDS=5000
LOGICAL-RECORD-LENGTH=52
LOGICAL-RECORDS-PER-BLOCK=10
DEVICE=RA81
DRIVE=46,5000,DU3
```

```
END-VARIABLE-ENTRY-DATA-SET

 --------------- VARIABLE THESIS ADVISOR FILE -------------

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=TADV      / VAR THESIS ADVISOR FILE /
IOAREA=VAR3

BASE-DATA
TADVTHES=10            / IDENTIFIES TADV TIED TO THES NUMBER /
THESLKTA=8=TADVTHES    / LINK TO THES /
TADVSTDT=9             / STUDENT SSSN /
STDTLKTA=8=TADVSTDT    / LINK TO STUDENT SSSN /
TADVFACT=9             / FACULTY FSSN /
FACTLKTA=8=TADVFACT    / LINK TO FACULTY FSSN /
END-DATA

TOTAL-LOGICAL-RECORDS=5000
LOGICAL-RECORD-LENGTH=54
LOGICAL-RECORDS-PER-BLOCK=9
DEVICE=RA81
DRIVE=47,5000,DU3

END-VARIABLE-ENTRY-DATA-SET


 --------------- VARIABLE PROFESSIONAL DEVELOPMENT FILE -------------

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=VPDQ             / VAR FILE TO DET INST PROF DEV QTRS /
IOAREA=VAR2

BASE-DATA
VPDQMQTR=4                     / TIED TO MQTR (QUARTER NUMBER) /
MQTRLKPD=8=VPDQMQTR            / LINK TO MQTR (MASTER QUARTER FILE) /
VPDQFACT=9                     / FACULTY SSSN /
FACTLKPD=8=VPDQFACT            / LINK TO FACULTY FSSN /
END-DATA

TOTAL-LOGICAL-RECORDS=5000
LOGICAL-RECORD-LENGTH=30
LOGICAL-RECORDS-PER-BLOCK=17
DEVICE=RA81
DRIVE=48,5000,DU3

END-VARIABLE-ENTRY-DATA-SET
```

```
--------------- VARIABLE SEQUENCE FILE -------------

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=VMSS              / VARIABLE SEQUENCE FILE /
IOAREA=VAR5

BASE-DATA
VMSSMSSF=3                      / TIED TO MASTER COURSE SEQUENCE NUMBER /
MSSFLKSS=8=VMSSMSSF             / LINK TO MASTER COURSE SEQUENCE FILE /
VMSSNMBR=8                      / TIED TO MASTER COURSE NUMBER /
MCRSLKSS=8=VMSSNMBR             / LINK TO MASTER COURSE /
VMSSMDEG=2                      / TIED TO MASTER DEG REQUIREMENT NUMBER /
MDEGLKSS=8=VMSSMDEG             / LINK TO MASTER DEGREE /
VMSSCRSS=30                     / LISTS WHICH COURSES BELONG IN SEQUENCE /
END-DATA

TOTAL-LOGICAL-RECORDS=5000
LOGICAL-RECORD-LENGTH=86
LOGICAL-RECORDS-PER-BLOCK=6
DEVICE=RA81
DRIVE=49,5000,DU3

END-VARIABLE-ENTRY-DATA-SET
END-DATA-BASE-GENERATION
```

Appendix B

FILE/DATA ITEM SYNTAX AND COMPATIBILITY RULES

The following lists of data names represents all of the items contained in the database generation source file. The first colomn is the named data item. The second column is the description of the data item. The third column is the syntax of the data item and specifies if the item is numeric, alphabetic, and the format of the data item. The last column specifies if the data item has any compatibility rules to be applied. This appendix is used in the development of error routines.

------------------------MASTER FACULTY FILE----------------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
FACTCRTL	SSAN	ALL NUMERIC 0-9	
FACTLKSE	LINK TO SECTION	N/A	SECTION MUST EXIST IN MASTER SECTION FILE.
FACTLKCM	LINK TO DEPT/COMM	N/A	DEPARTMENT CODE MUST EXIST IN MASTER DEP. FILE.
FACTNAME	FACULTY NAME	ALPHBETIC	SEE NOTE #1
FACTRANK	FACULTY RANK	E1..E9, O1..O10, G1..G13	
FACTSRVC	MILITARY SERVICE	NUMERIC	IN YEARS
FACTDOCM	DATE OF COMMISION	DDMMYY	OFFICER ONLY
FACTHDAT	DATE HIRED	DDMMYY	
FACTSALR	SALARY	NUMERIC	ROUND TO NEAREST DOLLAR
FACTDOBI	DATE OF BIRTH	DDMMYY	
FACTSEXX	FACULTY GENDER	"M" OR "F"	
FACTAERO	AERO RATING	ALPHA NUMERIC	
FACTDTSC	DUTY AFSC	EX: 4935B	MILITARY ONLY
FACTPMSC	PRIMARY AFSC	EX: 4935B	MILITARY ONLY
FACTDORK	DATE OF RANK	DDMMYY	
FACTYRSS	YEARS OF SERVICE	NUMERIC	ROUND TO NEAREST YEAR
FACTADDR	CURRENT ADDRESS	ALPHA NUMERIC	
FACTHPHN	HOME PHONE	NUMERIC	DOES NOT INCLUDE AREA CODE
FACTEADR	EMERGENCY ADDRESS	ALPHA NUMERIC	
FACTMSTA	MARITAL STATUS	"M" OR "S"	
FACTSPOS	SPOUSES' NAME	ALPHABETIC	
FACTSDOB	SPOUSE DATE OF	DDMMYY	

	BIRTH		
FACTNDEP	NUMBER OF DEPEND.	NUMERIC	
FACTRACE	RACE	ALPHABETIC	
FACTRELN	RELIGION	ALPHABETIC	
FACTOFIC	OFFICE RM NUMBER	ALPHANUMERIC	
FACTOPHN	OFFICE PHONE NO.	NUMERIC	EXCHANGE/EXTENSION
FACTLORG	LAST ORGANIZATION	ALPHANUMERIC	
FACTTITL	LAST DUTY TITLE	ALPHANUMERIC	
FACTDEPT	DEPARTURE DATE	DDMMYY	

NOTE #1: EXAMPLE OF CORRECT FORMAT: SMITH JOHN D

------------------------MASTER DEPARTMENT FILE--------------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
DEPTCTRL	DEPARTMENT CODE	ALPHABETIC	
DEPTNAME	DEPARTMENT NAME	ALPHABETIC	

------------------------MASTER STUDENT FILE----------------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
STDTCTRL	STUDENT SSAN	NUMERIC	
STDTLKSE	SECTION	N/A	DECLARED SECTION MUST EXIST IN MASTER SECTION FILE.
STDTLKAD	LINK TO FACULTY ADVISOR	N/A	FACULTY MEMBER MUST EXIST IN MASTER FACULTY FILE.
STDTSEQN	SEQUENCE CONTROL	NUMERIC	SEQUENCE MUST EXIST IN MASTER SEQUENCE FILE.
STDTNAME	STUDENT NAME	ALPHANUMERIC	
STDTRANK	STUDENT RANK	E1..E9 O1..O10 G1..G16	
STDTGRAD	GRADUATE OF AFIT?	"Y" OR "N"	
STDTSRVC	MILITARY SERVICE	NUMERIC	ROUND TO NEAREST YEAR
STDTAERO	AERO RATING	ALPHANUMERIC	
STDTDORK	DATE OF RANK	DDMMYY	
STDTDOCM	DATE OF COMMISSION	DDMMYY	MILITARY ONLY
STDTYRSS	YEARS OF SERVICE	NUMERIC	ROUNT TO NEAREST YEAR
STDTSEXX	STUDENT GENDER	"M" OR "F"	
STDTBOXN	STUDENT BOX NUMBER	NUMERIC	
STDTDTSC	DUTY AFSC	EX: 4935B	MILITARY ONLY
STDTPMSC	PRIMARY AFSC	EX: 4935B	MILITARY ONLY
STDTADDR	CURRENT ADDRESS	ALPHANUMERIC	STREET/CITY/STATE ETC..
STDTEADR	EMERGENCY ADDRESS	ALPHANUMERIC	STREET/CITY/STATE ETC..
STDTHMPH	HOME PHONE NUMBER	NUMERIC	
STDTDTPH	DUTY PHONE NUMBER	NUMERIC	
STDTEDCD	EDUCATION CODE	ALPHANUMERIC	
STDTDOBH	DATE OF BIRTH	DDMMYY	
STDTPOBH	PLACE OF BIRTH	ALPHANUMERIC/ADDRESS	
STDTMSTA	MARITAL STATUS	"M" OR "S"	
STDTSPOS	SPOUSES´ FIRST NAME	ALPHABETIC	
STDTSDOB	SPOUSES´ DATE OF BIRTH	DDMMYY	
STDTMSPS	MILITARY SPOUSE	"Y" OR "N"	
STDTNDEP	NO. OF DEPENDENTS	NUMERIC	
STDTRACE	STUDENT RACE	ALPHABETIC	
STDTRELN	STUDENTS RELIGION	ALPHABETIC	
STDTLCMD	LOSING COMMAND	ALPHABETIC	TAC,SAC,MAC, ETC..
STDTLORG	LAST ORGANIZATION	ALPHANUMERIC	
STDTTITL	LAST DUTY TITLE	ALPHABETIC	
STDTDIRN	DURATION OF LAST DUTY ASSIGNMENT	NUMERIC	ROUND TO NEAREST YEAR

-----------------------MASTER THESIS CATALOG NUMBER FILE---------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
ADNRCTRL	THESIS CODE	ALPHANUMERIC	
ADNRLKTH	LINK TO THESIS TITLE	N/A	

-----------------------MASTER THESIS NUMBER FILE----------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
THESCTRL	THESIS CATALOG NO.	NUMERIC	
THESLKTA	LINK TO THESIS ADVISOR FILE	N/A	ADVISOR MUST EXIST IN THE FACULTY MASTER FILE.
THESLKTC	LINK TO THESIS COMMITTEE MEMBER	N/A	COMMITTEE MEMBER MUST EXIST IN FACULTY FILE

----------------------MASTER SECTION NUMBER FILE----------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
SECTCTRL	SECTION NUMBER	ALPHA-NUMBERIC	
SECTLKSE	*LINK TO SECTION* LEADER	N/A	*STUDENT MUST EXIST* STUDENT MASTER FILE
SECTLKAD	LINK TO FACT ADVISOR	N/A	FACULTY ADVISOR MUST EXIST IN FACULTY MASTER FILE
SECTLSSN	SECTION LEADER SSAN	NUMERIC	SSAN MUST BE IN STUDENT STUDENT MASTER FILE.
SECTGRDT	GRADUATION DATE	DDMMYY	
SECTENDT	ENTRY DATE	DDMMYY	
SECTNRSN	NUMBER OF STUDENTS	NUMERIC	

-----------------------MASTER COURSE FILE-----------------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
MCRSCTRL	COURSE NUMBER	ALPHANUMERIC	
MCRSCRHR	CREDIT HOURS	NUMERIC	
MCRSLCHR	LECTURE HOURS	NUMERIC	
MCRSSZLM	SIZE LIMITATION	NUMERIC	DEFAULT TO 30.
MCRSTITL	COURSE TITLE	ALPHANUMBERIC	
MCRSREST	RESTRICTED	"Y" OR "N"	

-----------------------MASTER QUARTER FILE----------------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
MQTRCTRL	QUARTER NUMBER	TWO ALHA SEASON TWO DIGIT YEAR	
MQTRSTDT	START DATE	DDMMYY	
MQTRSPDT	STOP DATE	DDMMYY	

-----------------------MASTER BOOK FILE-------------------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
MBKTCTRL	BOOK TITLE (KEY)	ALPHANUMERIC	
MBKTLKBK	LINK TO COURSE	N/A	COURSE MUST EXIST IN MASTER COURSE FILE
MBKTATHR	BOOK AUTHOR	ALPHABETIC	
MBKTPUBL	BOOK PUBLISHER	ALPHANUMERIC	
MBKTNABL	NUMBER AVAIL	NUMERIC	
MBKTPRCE	PRICE	NUMERIC	LAST TWO DIGITS DECIMAL

-----------------------MASTER ORDER FILE------------------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
MORDCTRL	MASTER ORDER NUMBER	NUMERIC	
MORDORDT	ORDER NUMBER	NUMERIC	
MORDDUDT	DUE DATE	DDMMYY	
MORDCMPY	COMPANY	ALPHANUMERIC	
MORDADDR	ADDRESS	ALPHANUMERIC	
MORDPHNE	PHONE	NUMERIC	WITH AREA CODE

-----------------------MASTER CLASS TIME FILE-----------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
TIMECTRL	CLASS TIME	NUMERIC	MILITARY CLOCK 0000-2400

-----------------------MASTER BUILDING/ROOM FILE-----------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
BLRMCTRL	BUILDING AND ROOM NUMBER	NUMERIC	

-----------------------MASTER ROOM CAPACITY FILE-----------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
CPTYCTRL	CAPACITY NUMBER	NUMERIC	DEFAULT IS 30
CPTYLKCL	LINK TO CLASSROOM FILE	N/A	ROOM MUST EXIST IN MASTER BUILDING/ROOM FILE

-----------------------MASTER DAY SCHEDULING FILE-----------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
DAYSCTRL	DAY OF THE WEEK	MOND,TUES,WEDN, THUR,FRID	

-----------------------MASTER SEQUENCE FILE-----------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
MSSFCTRL	COURSE SEQUENCE NUMBER	NUMERIC	
MSSFSEQN	SEQUENCE NAME	ALPHANUMERIC	

-----------------------MASTER DEGREE REQUIREMENTS FILE-----------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
MDEGCTRL	GRADUATE DEGREE CODE	NUMERIC	
MDEGNAME	DEGREE NAME	ALPHANUMERIC	

------------------------VARIABLE EDUCATION FILE-------------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
VEDUFSSN	FACULTY SSAN	NUMERIC	MUST HAVE AN ASSOCIATED MASTER RECORD
VEDUSTDT	STUDENT SSAN	NUMERIC	MUST HAVE AN ASSOCIATED MASTER RECORD
VEDUUNIV	UNIVERSITY ATT.	ALPHABETIC	
VEDUDEGR	DEGREE EARNED	ALPHABETIC	
VEDUYEAR	YEAR EARNED	NUMERIC	19XX

------------------------VARIABLE FACULTY SOCIETY FILE-------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
FSOCFSSN	FACUTLY SSAN	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
FSOCSOCY	SOCIETIES TO WHICH PERSON BELONGS	ALPHANUMERIC	

------------------------VARIABLE FACULTY DEPT & COMMITTE FILE------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
FCMTFSSN	FACULTY SSAN	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
FCMTDCOD	DEPARTMENT CODE	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
FCMTNAMD	NAME OF COMMITTEE	ALPHABETIC	
FCMTDEPD	NAME OF DEPARTMENT	ALPHABETIC	
FCMTNAMC	NAME OF COMMITTEE	ALPHABETIC	
FCMTNAMC	NAME OF DEPARTMENT	ALPHABETIC	

------------------------VARIABLE HONORS AND AWARDS FILE------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
VHAWFSSN	FACULTY SSAN	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
VHAWSTDT	STUDENT SSAN	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
VHAWHONR	HONORS RECIEVED	ALPHANUMERIC	
VHAWDATE	DATE RECIEVED	DDMMYY	
VHAWAWED	AWARDS RECIEVED	ALPHANUMERIC	
VHAWADAT	DATE RECIEVED	DDMMYY	

------------------------VARIABLE FACULTY INTERESTS FILE------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
FINTFSSN	FACULTY SSAN	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD.
FINTAREA	AREA OF INTEREST	ALPHANUMERIC	

------------------------VARIABLE FACULTY PUBS AND PRESENTATIONS----------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
FCOMFSSN	FACULTY SSAN	NUMERIC	MUST HAVE AN ASSOCIATED MASTER RECORD
FCOMNAME	TITLE OF PUB	ALPHANUMERIC	
FCONDATE	DATE OF PUB	DDMMYY	
FCOMCOAU	NAMES OF CO-AUTHORS	ALPHABETIC	
FCOMORGN	PRESENTATION GIVEN AT THIS ORGAN.	ALPHANUMERIC	
FCOMPDAT	DATE PRESENTAION GIVEN	DDMMYY	

------------------------VARIABLE FACULTY TDY FILE------------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
FTDYFSSN	FACUTLY SSAN	NUMERIC	MUST HAVE AN ASSOCIATED MASTER RECORD
FTDYCOST	COST OF TDY	NUMERIC	LAST TWO PLACES REP. CENTS
FTDYDEST	DESTINATION	ALPHABETIC	
FTDYBDAT	BEGINNING DATE	DDMMYY	
FTDYEDAT	ENDING DATE	DDMMYY	

------------------------VARIABLE STUDENT COURSE FILE--------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
CRSESTDT	STUDENT SSAN	NUMERIC	MUST HAVE AN ASSOCIATED MASTER RECORD
CRSEMDEG	TYPE DEGREE	ALPHABETIC	MUST HAVE AN ASSOCIATED MASTER RECORD
CRSENUMB	COURSE NUMBER	---	MUST BE CONTAINED IN THE MASTER COURSE FILE
CRSENAME	COURSE NAME	ALPLHABETIC	
CRSEGRAD	COURSE GRAD	DDMMYY	
CRSEBEGN	QUARTER BEGAN	---	MUST EXIST IN THE MASTER QUARTER FILE
CRSECOLL	COLLEGE ATTENDED	ALPHABETIC	
CRSEWAIV	COURSE WAIVED	"Y" OR "N"	

-----------------------VARIABLE THESIS TITLE FILE----------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
THTLADNR	THESIS CATALOG NO	NUMERIC	MUST HAVE AN ASSOCIATED MASTER RECORD
THTLTHES	DEPARTMENT THES NO	NUMERIC	MUST HAVE AN ASSOCIATED MASTER RECORD
THTLFACT	FACYLTY ADVISORS SSAN	NUMERIC	MUST HAVE AN ASSOCIATED MASTER RECORD
THTLSTDT	STUDENTS SSAN	NUMERIC	MUST HAVE AN ASSOCIATED MASTER RECORD
THTLTITL	THESIS TITLE	ALPHANUMERIC	
THTLSPON	SPONSOR	ALPHANUMERIC	
THTLLOCN	THESIS LOCATION	ALPHANUMERIC	
THTLCLAS	THESIS CLASSIFICATION	"SECRET","TOP-SECRET", ETC...	

-----------------------VARIABLE SECTION LEADER FILE--------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
SECLSECT	SECTION NUMBER	ALPHANUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
SECLSTDT	SECTION LEADER SSAN	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
SECLFACT	FACULTY ADVISOR SSAN	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD

-----------------------VARIABLE QUARTER FILE--------------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
VCQRNMBR	COURSE NUMBER	ALPHANUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
VCQRIDEN	QUARTER NUMBER	ALPHANUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
VCQRSSSN	STUDENT SSAN	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
VCQRGRAD	GRADUATE DATA	ALPHANUMERIC	
VCQRFSSN	FACULTY SSAN	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD

-----------------------VARIABLE REQUISITE FILE------------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
VREQCODE	CODED RECORD NUMBER	NUMERIC	
VREQNMBR	COURSE NUMBER	ALPHANUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
VREQRNUM	PRE-REQUISITE COURSE NUMBER	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
VREQPNUM	PRE-REQUISITE COURSE NUMBER	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD

-----------------------VARIABLE BOOK LINK FILE------------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
VCBKNMBR	COURSE NUMBER	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
VCBKTITL	BOOK TITLE	ALPHANUMERIC	MUST HAVE ASSOCIATED MASTER RECORD

-----------------------VARIABLE NUMBER ORDERED FILE-------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
VNMOTITL	BOOK TITLE	ALPHANUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
VNMONMBR	ORDER NUMBER	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
VNMONORD	NUMBER ORDERED	NUMERIC	

------------------------VARIABLE CLASS SCHEDULE FILE--------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
SCHDNSTD	NUMBER OF STUDENTS	NUMERIC	
SCHDNMBR	COURSE NUMBER	ALPHANUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
SCHDDAYS	DAY CLASS MEETS	ALPHABETIC	MUST HAVE ASSOCIATED MASTER RECORD
SCHDTIME	TIME CLASS MEETS	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
SCHDBLRM	BUILDING/ROOM	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
SCHDFNTM	FINISH TIME	NUMERIC	24 HOUR CLOCK.

------------------------VARIABLE CLASSROOM FILE-------------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
CLSRBLRM	BUILDING AND ROOM	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
CLSRCPTY	CAPACITY	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
CLSREQPT	TYPE OF EQUIPEMENT	ALPHANUMERIC	
CLSRTYPE	CODE TYPE FOR ROOM	ALPHANUMERIC	
CLSRCFLG	SECURITY LEVEL	ALPHANUMERIC	

------------------------VARIABLE FACULTY ADVISOR FILE--------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
FADVSECT	SECTION	ALPHANUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
FADVSTDT	STUDENT SSAN	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
FADVFACT	FACULTY SSAN	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD

------------------------VARIABLE INSTRUCTOR STATISTICS FILE--------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
VINSSTDT	STUDENTS SSAN	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
VINSNMBR	COURSE NUMBER	ALPHANUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
VINSFACT	FACUTLY SSAN	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD

------------------------VARIABLE PROFESSIONAL DEVELOPMENT FILE-----------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
VPDQMQTR	QUARTER NUMBER	ALPHANUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
VPDQFACT	FACULTY SSAN	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD

------------------------VARIABLE SEQUENCE FILE--------------------------

NAME	DESCRIPTION	SYNTAX	COMPATIBILITY RULES
VMSSMSSF	COURSE SEQUENCE NUMBER	NUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
VMSSNMBR	COURSE NUMBER	ALPHANUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
VMSSMDEG	DEGREE NUMBER	ALPHANUMERIC	MUST HAVE ASSOCIATED MASTER RECORD
VMSSCRSS	COURSES IN SEQUENCE	ALPHANUMERIC	

APPENDIX C

STANDARD DATABASE RECORD TYPE DECLARATIONS

The Pascal constant and type declarations described in this appendix were used in the development of the AFIT/ENG Database application programs. The file is named TYPE.PAS and is maintained seperately from the applications programs. The file was intended to be used as a #INCLUDE file so maintenance could be performed on the file without the need to recompile the software.

Included in this file are the constants, buffer types, records types, and array types needed for the standard database routines. This file is required whenever the standard routines are used within a program.

```
CONST
        EXTRA1  = ' ';
        EXTRA2  = '  ';
        EXTRA3  = '   ';
        EXTRA4  = '    ';
        EXTRA5  = '     ';
        EXTRA10 = '          ';
        EXTRA25 = '                         ';
        EXTRA40 = '                                        ';
        FACTCONST1 = 'FACTCTRLFACTNAMEFACTRANKFACTSRVCFACTDOCM';
        FACTCONST2 = 'FACTHDATFACTSALRFACTDOBIFACTSEXXFACTAERO';
        FACTCONST3 = 'FACTDTSCFACTPMSCFACTDORKFACTYRSSFACTADDR';
        FACTCONST4 = 'FACTHPHNFACTEADRFACTMSTAFACTSPOSFACTSDOB';
        FACTCONST5 = 'FACTNDEPFACTRACEFACTRELNFACTOFICFACTOPHN';
        FACTCONST6 = 'FACTLORGFACTTITLFACTDEPTEND.            ';
        STDTCONST1 = 'STDTCTRLSTDTSEQNSTDTNAMESTDTRANKSTDTGRAD';
        STDTCONST2 = 'STDTSRVCSTDTAEROSTDTDORKSTDTDOCMSTDTYRSS';
        STDTCONST3 = 'STDTSEXXSTDTBOXNSTDTDTSCSTDTPMSCSTDTADDR';
        STDTCONST4 = 'STDTEADRSTDTHMPHSTDTDTPHSTDTEDCDSTDTDOBH';
        STDTCONST5 = 'STDTPOBHSTDTMSTASTDTSPOSSTDTSDOBSTDTMSPS';
        STDTCONST6 = 'STDTNDEPSTDTRACESTDTRELNSTDTLCMDSTDTLORG';
        STDTCONST7 = 'STDTTITLSTDTDURNEND.                    ';
        DEPTCONST1 = 'DEPTCTRLDEPTNAMEEND.                    ';
        THESCONST1 = 'THESCTRLTHESTITLTHESSPONTHESLOCNTHESCLAS';
        THESCONST2 = 'THESNAMEEND.                            ';
        SECTCONST1 = 'SECTCTRLSECTLSSNSECTGRDTSECTENDTSECTNRSN';
        SECTCONST2 = 'END.                                    ';
        MCRSCONST1 = 'MCRSCTRLMCRSCRHRMCRSLCHRMCRSLBHRMCRSSZLM';
        MCRSCONST2 = 'MCRSTITLMCRSRESTEND.                    ';
        MQTRCONST1 = 'MQTRCTRLMQTRSTDTMQTRSPDTEND.            ';
```

```
MBKTCONST1 = 'MBKTCTRLMBKTATHRMBKTPUBLMBKTNAVLMBKTPRCE';
MBKTCONST2 = 'END.                                    ';
MORDCONST1 = 'MORDCTRLMORDORDTMORDDUDTMORDCMPYMORDADDR';
MORDCONST2 = 'MORDPHNEEND.                            ';
TIMECONST1 = 'TIMECTRLEND.                            ';
BLRMCONST1 = 'BLRMCTRLEND.                            ';
CPTYCONST1 = 'CPTYCTRLEND.                            ';
DAYSCONST1 = 'DAYSCTRLEND.                            ';
MSSFCONST1 = 'MSSFCTRLMSSFSEQNEND.                    ';
MDEGCONST1 = 'MDEGCTRLMDEGNAMEEND.                    ';
VEDUCONST1 = 'VEDUFSSNVEDUSTDTVEDUUNIVVEDUDEGRVEDUYEAR';
VEDUCONST2 = 'END.                                    ';
FSOCCONST1 = 'FSOCFSSNFSOCSOCYFSOCDUM1END.            ';
FCMTCONST1 = 'FCMTCODEFCMTFSSNFCMTDCODFCMTDATAFCMTNAMD';
FCMTCONST2 = 'FCMTDEPDFCMTNAMCFCMTDEPCEND.            ';
VHAWCONST1 = 'VHAWCODEVHAWFSSNVHAWSTDTVHAWDATAVHAWHONR';
VHAWCONST2 = 'VHAWDATEVHAWAWRDVHAWADATEND.            ';
FINTCONST1 = 'FINTFSSNFINTAREAEND.                    ';
FCOMCONST1 = 'FCOMCODEFCOMFSSNFCOMDATAFCOMNAMEFCOMDATE';
FCOMCONST2 = 'FCOMCOQUFCOMORGNFCOMPDATEND.            ';
FTDYCONST1 = 'FTDYFSSNFTDYCOSTFTDYDESTFTDYBDATFTDYEDAT';
FTDYCONST2 = 'EN                                      ';
CRSECONST1 = 'CRSESTDTCRSEMDEGCRSENUMBCRSENAMECRSEGRAD';
CRSECONST2 = 'CRSEBEGNCRSECOLLCRSEWAIVEND.            ';
THTLCONST1 = 'THTLADNRTHTLTHESTHTLFACTTHTLSTDTTHTLTITL';
THTLCONTS2 = 'THTLSPONTHTLLOCNTHTLCLASEND.            ';
SECLCONST1 = 'SECLSECTSECLSTDTSECLFACTEND.            ';
VCQRCONST1 = 'VCQRCODEVCQRNMBRVCQRIDENVCQRSSSNEND.    ';
VCQRCONST2 = '*CODE=QSVCQRNMBRVCQRIDENEND.            ';
VREQCONST1 = 'VREQCODEVREQNMBRVREQDATAVREQRNUMVREQBLKA';
VREQCONST2 = 'VREQPNUMVREQBLKBEND.                    ';
VCBKCONST1 = 'VCBKNMBRVCBKTITLEND.                    ';
VNMOCONST1 = 'VNMOTITLVNMONMBRVNMONORDEND.            ';
SCHDCONST1 = 'SCHDNSTDSCHDNMBRSCHDDAYSSCHDTIMESCHDBLRM';
SCHDCONST2 = 'SCHDFNTMEND.                            ';
CLSRCONST1 = 'CLSRBLRMCLSRCPTYCLSREQPTCLSRTYPECLSRCLFG';
CLSRCONST2 = 'END.                                    ';
TCMFCONST1 = 'TCMFFACTTCMFSTDTTCMFTHESEND.            ';
FADVCONST1 = 'FADVSECTFADVSTDTFADVFACTEND.            ';
VINSCONST1 = 'VINSSTDTVINSNMBRVINSFACTEND.            ';
TADVCONST1 = 'TADVTHESTADVSTDTTADVFACTEND.            ';
VPDQCONST1 = 'VPDQMQTRVPDQFACTEND.                    ';
VMSSCONST1 = 'VMSSMSSFVMSSNMBRVMSSMDEGVMSSCRSSEND.    ';
EXTRA      = '                                        ';
```

```
TYPE
     BUFF1    = CHAR;
     BUFF2    = PACKED ARRAY[1..2] OF CHAR;
     BUFF3    = PACKED ARRAY[1..3] OF CHAR;
     BUFF4    = PACKED ARRAY [1..4] OF CHAR;
     BUFF5    = PACKED ARRAY [1..5] OF CHAR;
     BUFF6    = PACKED ARRAY [1..6] OF CHAR;
     BUFF7    = PACKED ARRAY [1..7] OF CHAR;
     BUFF8    = PACKED ARRAY [1..8] OF CHAR;
     BUFF9    = PACKED ARRAY [1..9] OF CHAR;
     BUFF10   = PACKED ARRAY [1..10] OF CHAR;
     BUFF11   = PACKED ARRAY [1..11] OF CHAR;
     BUFF12   = PACKED ARRAY [1..12] OF CHAR;
     BUFF14   = PACKED ARRAY [1..14] OF CHAR;
     BUFF15   = PACKED ARRAY [1..15] OF CHAR;
     BUFF16   = PACKED ARRAY [1..16] OF CHAR;
     BUFF17   = PACKED ARRAY [1..17] OF CHAR;
     BUFF18   = PACKED ARRAY [1..18] OF CHAR;
     BUFF19   = PACKED ARRAY [1..19] OF CHAR;
     BUFF20   = PACKED ARRAY [1..20] OF CHAR;
     BUFF21   = PACKED ARRAY [1..21] OF CHAR;
     BUFF22 =   PACKED ARRAY [1..22] OF CHAR;
     BUFF23   = PACKED ARRAY [1..23] OF CHAR;
     BUFF24   = PACKED ARRAY [1..24] OF CHAR;
     BUFF25   = PACKED ARRAY [1..25] OF CHAR;
     BUFF28   = PACKED ARRAY [1..28] OF CHAR;
     BUFF30   = PACKED ARRAY [1..30] OF CHAR;
     BUFF33   = PACKED ARRAY [1..33] OF CHAR;
     BUFF36   = PACKED ARRAY [1..36] OF CHAR;
     BUFF38   = PACKED ARRAY [1..38] OF CHAR;
     BUFF40   = PACKED ARRAY [1..40] OF CHAR;
     BUFF48   = PACKED ARRAY [1..48] OF CHAR;
     BUFF50   = PACKED ARRAY [1..50] OF CHAR;
     BUFF61   = PACKED ARRAY [1..61] OF CHAR;
     BUFF80   = PACKED ARRAY [1..80] OF CHAR;
     BUFF120  = PACKED ARRAY [1..120] OF CHAR;
     BUFF122  = PACKED ARRAY [1..122] OF CHAR;
     BUFF130  = PACKED ARRAY [1..130] OF CHAR;
     BUFF200  = PACKED ARRAY [1..200] OF CHAR;
     BUFF240  = PACKED ARRAY [1..240] OF CHAR;
     BUFF236  = PACKED ARRAY [1..236] OF CHAR;
     BUFF280  = PACKED ARRAY [1..280] OF CHAR;
     BUFF400  = PACKED ARRAY [1..400] OF CHAR;
     BUFF408  = PACKED ARRAY [1..408] OF CHAR;
     BUFF480  = PACKED ARRAY [1..480] OF CHAR;
```

```
FACT_REC = RECORD;
          CTRL : BUFF9;    {* SSAN *}
          NAME : BUFF28;
            {* FACULTY MEMBERS NAME,LAST,FIRST,MI*}
          RANK : BUFF3;
            {* MIL/CIV RANK (O-OFFICER,G-CIV,NN-RANK)*}
          SRVC : BUFF2;    {* MILITARY SERVICE *}
          DOCM : BUFF6;    {* DATE OF COMMISSION *}
          HDAT : BUFF6;    {* DATE HIRED *}
          SALR : BUFF5;    {* SALARY *}
          DOBI : BUFF6;    {* DATE OF BIRTH *}
          SEXX : BUFF1;    {* SEX *}
          AERO : BUFF10;   {* AERO RATING *}
          DTSC : BUFF6;    {* DUTY AFSC *}
          PMSC : BUFF6;    {* PRIMARY AFSC *}
          DORK : BUFF6;    {* DATE OF RANK *}
          YRSS : BUFF2;    {* YEARS OF SERVICE *}
          ADDR : BUFF40;   {* CURRENT ADDRESS - *}
          HPHN : BUFF7;    {* HOME PHONE (EXCHANGE,EXTENSION)*}
          EADR : BUFF40;   {* EMERGENCY ADDRESS *}
          MSTA : BUFF1 ;   {* MARITAL STATUS *}
          SPOS : BUFF12;   {* SPOUSE FIRST NAME *}
          SDOB : BUFF6 ;   {* SPOUSE DATE OF BIRTH *}
          NDEP : BUFF2 ;   {* NUMBER OF DEPENDENTS *}
          RACE : BUFF2 ;   {* RACE *}
          RELN : BUFF2 ;   {* RELIGION *}
          OFIC : BUFF12;   {* OFFICE RM NUMBER *}
          OPHN : BUFF7 ;   {* OFFICE PHONE (EXCHANGE,EXTENSION)*}
          LORG : BUFF50;   {* LAST ORGANIZATION *}
          TITL : BUFF50;   {* LAST POSITION TITLE *}
          DEPT : BUFF6 ;   {* EXPECTED AFIT DEPARTURE DATE *}
END;

DEPT_REC = RECORD;
          CTRL : BUFF4 ;       {* DEPT CODE *}
          NAME : BUFF20
END;
```

```
STDT_REC = RECORD;
          CTRL : BUFF9;         {* STUDENT SOCIAL SECURITY *}
          SEQN : BUFF3;         {*  MASTER SEQUENCE CONTROL NUMBER *}
          NAME : BUFF28;        {*  STUDENT NAME (LAST,FIRST,MI) *}
          RANK : BUFF3;         {*  MIL/CIV RANK(O-OFFICER,G-CIV,NN-RANK *}
          GRAD : BUFF1;         {*  HAS STUDENT ALREADY GRADUATED/LEFT AFIT?
          SRVC : BUFF2;         {*  MILITARY SERVICE *}
          AERO : BUFF10;        {*  AERO RATING *}
          DORK : BUFF6;         {*  DATE OF RANK *}
          DOCM : BUFF6;         {*  DATE OF COMMISSION *}
          YRSS : BUFF2;         {*  YEARS OF SERVICE *}
          SEXX : BUFF1;         {*  SEX *}
          BOXN : BUFF4;         {*  BOX NUMBER *}
          DTSC : BUFF6;         {*  DUTY AFSC *}
          PMSC : BUFF6;         {*  PRIMARY AFSC *}
          ADDR : BUFF40;        {*  CURRENT ADDRESS *}
          EADR : BUFF40;        {*  EMERGENCY ADDRESS *}
          HMPH : BUFF7;         {*  HOME PHONE NUMBER *}
          DTPH : BUFF7;         {*  DUTY PHONE NUMBER *}
          EDCD : BUFF5;         {*  EDUCATION CODE *}
          DOBH : BUFF6;         {*  DATE OF BIRTH *}
          POBH : BUFF40;        {*  PLACE OF BIRTH *}
          MSTA : BUFF1;         {*  MARITAL STATUS -*}
          SPOS : BUFF12;        {*  SPOUSE FIRST NAME *}
          SDOB : BUFF6;         {*  SPOUSE DATE OF BIRTH *}
          MSPS : BUFF1;         {*  MILITARY SPOUSE *}
          NDEP : BUFF2;         {*  NUMBER OF DEPENDENTS *}
          RACE : BUFF2;         {*  RACE *}
          RELN : BUFF2;         {*  RELIGION *}
          LCMD : BUFF5;         {*  LOSING COMMAND *}
          LORG : BUFF50;        {*  LAST ORGANIZATION *}
          TITL : BUFF50;        {*  LAST POSITION TITLE *}
          DURN : BUFF2;         {*  DURATION AT LAST DUTY*}
END;


THES_REC = RECORD;
          CTRL : BUFF10;  {*THESIS CATALOGING NUMBER*}
          TITL : BUFF50;  {*THESIS TITLE*}
          SPON : BUFF50;  {*THESIS SPONSOR*}
          LOCN : BUFF50;  {*THESIS LOCATION*}
          CLAS : BUFF12;  {*THESIS CLASSIFICATION*}
          NAME : BUFF28;  {* STUDENTS NAME FOR ARCHIVE *}
END;
```

```
SECT_REC = RECORD;
          CTRL : BUFF8;    {*SECTION NUMBER (EX., GCS-84D)*}
          LKSE : BUFF8;        {*LINK TO SECTION LEADER FILE*}
          LKAD : BUFF8;        {*LINK TO FACULTY ADVISOR*}
          LSSN : BUFF9;        {*SECTION LEADER SSSN*}
          GRDT : BUFF6;        {*GRADUATION DATE*}
          ENDT : BUFF6;        {*ENTRY DATE*}
          NRSN : BUFF3;        {*NUMBER OF STUDENTS IN SECTION*}
END;

MCRS_REC = RECORD;
          CTRL : BUFF8;    {*COURSE NUMBER*}
          CRHR : BUFF1;        {*COURSE CREDIT HOURS*}
          LCHR : BUFF1;    {*COURSE LECTURE HOURS DATA*}
          LBHR : BUFF1;    {*COURSE LAB HOUR DATA*}
          SZLM : BUFF2;    {*SIZE LIMIT DATA*}
          TITL : BUFF50;   {*TITLE DATA*}
          REST : BUFF1;        {*RESTRICTED (FROM GRAD REQ) COURSE*}
END;

MQTR_REC = RECORD;
          CTRL : BUFF4;    {*QUARTER NUMBER*}
          STDT : BUFF6;        {*QUARTER START DATE(DAY,MO,YR)*}
          SPDT : BUFF6;        {*QUARTER STOP DATE (DAY,MO,YR)*}
END;

MBKT_REC = RECORD;
          CTRL : BUFF40;   {*BOOK TITLE NAME*}
          ATHR : BUFF28;       {*BOOK AUTHOR NAME (LAST,FIRST,MI)*}
          PUBL : BUFF28;       {*BOOK PUBLISHER NAME*}
          NAVL : BUFF6;    {*BOOKS AVAILABLE*}
          PRCE : BUFF4;    {*BOOK PRICE*}
END;

MORD_REC = RECORD;
          CTRL : BUFF7;    {*MASTER ORDER NUMBER*}
          ORDT : BUFF6;    {*ORDER NUMBER*}
          DUDT : BUFF6;    {*DUE DATE*}
          CMPY : BUFF20;   {*COMPANY*}
          ADDR : BUFF40;   {*COMPANY ADDRESS*}
          PHNE : BUFF10;   {*COMPANY PHONE NUMBER WITH AREA CODE*}
END;

TIME_REC = RECORD;
          CTRL : BUFF4;        {*MILITARY CLOCK TIME*}
END;

BLRM_REC = RECORD;
          CTRL : BUFF8;        {*BUILDING AND ROOM NUMBER*}
END;

CPTY_REC = RECORD;
          CTRL : BUFF4;        {*CAPACITY NUMBER*}
END;
```

```
DAYS_REC = RECORD;
        CTRL : BUFF4;          {*DAY OF THE WEEK*}
END;

MSSF_REC = RECORD;
        CTRL : BUFF3;          {*COURSE SEQUENCE NUMBER*}
        SEQN : BUFF40;         {*SEQUENCE NAME*}
END;

MDEG_REC = RECORD;
        CTRL : BUFF2;          {*NUMBER IDENTIFYING TYPE GRAD DEGREE*}
        NAME : BUFF40;         {*NAME OF TYPE OF DEGREE*}
END;

VEDU_REC = RECORD;
        FSSN : BUFF9;          {*FACULTY SSN*}
        STDT : BUFF9;          {*STUDENT SSN*}
        UNIV : BUFF40;         {*INSTITUTION (UNIVERSITY) ATTENDED*}
        DEGR : BUFF40;             {*DEGREE EARNED*}
        YEAR : BUFF4;          {*YEAR DEGREE AWARDED*}
END;

FSOC_REC = RECORD;
        FSSN : BUFF9;          {*FACULTY SSN*}
        SOCY : BUFF40;         {*SOCIETIES TO WHICH INDIVIDUAL BELONGS *}
        DUM1 : BUFF8;          {*PADDING TO INCREASE REC LENGTH*}
END;

FCMT_REC = RECORD;
        CODE : BUFF2;
        FSSN : BUFF9;          {*FACULTY SSN : BUFF9*}
        DCOD : BUFF4;          {*DEPARTMENT DCOD*}
        DATA : BUFF14;         {*REDEFINED DATA AREA LENGTH*}
        NAMD : BUFF10;         {*NAME OF COMMITTEE*}
        DEPD : BUFF4;          {*NAME OF DEPARTMENT ??*}
        NAMC : BUFF10;         {*NAME OF OTHER COMMITTEE*}
        DEPC : BUFF4;          {*NAME OF OTHER DEPARTMENT ??*}
END;

VHAW_REC = RECORD;
        CODE : BUFF2;
        FSSN : BUFF9;          {*FACULTY SSN*}
        STDT : BUFF9;          {*STUDENT SOCIAL SECURITY NUMBER*}
        DATA : BUFF16;         {*REDEFINING DATA LENGTH AREA*}
        HONR : BUFF10;         {*HONORS RECEIVED*}
        DATE : BUFF6;          {*DATE HONOR RECEIVED*}
        AWRD : BUFF10;         {*AWARDS RECEIVED*}
        ADAT : BUFF6;          {*DATE AWARD RECEIVED*}
END;
```

```
FINT_REC = RECORD;
        FSSN : BUFF9;                {*FACULTY SSN*}
        AREA : BUFF15;           {*AREA OF INTEREST*}
END;

FCOM_REC = RECORD;
        CODE : BUFF2;
        FSSN : BUFF9;            {*FACULTY SSN*}
        DATA : BUFF61;           {*REDEFINED DATA AREA LENGTH*}
        NAME : BUFF25;           {*TITLE OF PUBLICATION*}
        DATE : BUFF6;            {*DATE OF PUBLICATION*}
        COAU : BUFF30;           {*NAME(S) OF CO-AUTHOR(S)*}
        ORGN : BUFF25;           {*PRESENTATION GIVEN TO THIS ORGANIZATION*}
        PDAT : BUFF6;            {*PRESENTATION DATE*}
END;

FTDY_REC = RECORD;
        FSSN : BUFF9;            {*FACULTY SSN*}
        COST : BUFF7;            {*COST OF TDY DATA IN THIS FILE*}
        DEST : BUFF20;           {*DESTINATION*}
        BDAT : BUFF6;            {* DATE*}
        EDAT : BUFF6;            {*END DATA*}
END;

CRSE_REC = RECORD;
        STDT : BUFF9;            {*STUDENT SOCIAL SECURITY NUMBER*}
        MDEG : BUFF2;            {*TYPE GRAD DEGREE (NUMBER - FROM MDEG)*}
        NUMB : BUFF8;            {*COURSE NUMBER*}
        NAME : BUFF20;           {*COURSE NAME*}
        GRAD : BUFF2;            {*COURSE GRADE*}
        BEGN : BUFF4;            {*QUARTER STUDENT TOOK OR WILL TAKE COURSE*}
        COLL : BUFF30;           {*COLLEGE ATTENDED*}
        WAIV : BUFF1;            {*COURSE WAIVED? (Y/N)*}
END;

THTL_REC = RECORD;
        THES : BUFF10;           {*DEPARTMENT THESIS NUMBER*}
        FACT : BUFF9;            {*FACULTY ADVISOR FSSN*}
        STDT : BUFF9;            {*STUDENT SSSN*}
END;

SECL_REC = RECORD;
        SECT : BUFF8;            {*RELATED TO SECT (SECTION NUMBER)*}
        STDT : BUFF9;            {*STUDENT SSSN*}
        FACT : BUFF9;            {*FACULTY FSSN*}
END;

VREC = RECORD;
        CODE : BUFF2;
        NMBR : BUFF8;        {*COURSE NUMBER CONTROL FIELD*}
        IDEN : BUFF4;        {*QUARTER IDENT CONTROL FIELD*}
        SSSN : BUFF9;            {*STUDENT SSN DATA*}
        VREF : BUFF4;
END;
```

```
VREQ_REC = RECORD;
        CODE : BUFF2;    {*CODED RECORD FOR REQUISITE*}
        NMBR : BUFF8;    {*COURSE NUMBER CONTROL FIELD*}
        DATA : BUFF12;   {*REDEFINED REQUISITE DATA*}
        RNUM : BUFF8;    {*REQUISITE COURSE NUMBER*}
        BLKA : BUFF6;    {*RECORD BTYE FILLER FOR TOTAL*}
        PNUM : BUFF8;    {*REQUISITE COURSE NUMBER*}
        BLKB : BUFF6;    {*RECORD BYTE FILLER FOR TOTAL*}
END;

VCBK_REC = RECORD;
        NMBR : BUFF8;    {*COURSE NUMBER CONTROL FIELD*}
        TITL : BUFF40;   {*BOOK TITLE CONTROL FIELD*}
END;

VNMO_REC = RECORD;
        TITL : BUFF40 ;      {*BOOK TITLE CONTROL FIELD*}
        NMBR : BUFF7;    {*ORDER NUMBER CONTROL FIELD*}
        NORD : BUFF3;    {*NUMBER ORDERED DATA ITEM*}
END;


SCHD_REC = RECORD;
        NSTD : BUFF3 ;       {*NUMBER OF STUDENTS IN CLASS*}
        NMBR : BUFF8 ;       {*COURSE NUMBER*}
        DAYS : BUFF4;        {*DAY CLASS MEETS*}
        TIME : BUFF4;        {*TIME CLASS STARTS*}
        BLRM : BUFF8;        {*BUILDING AND ROOM NUMBER*}
        FNTM : BUFF4;        {*CLASS FINISH TIME*}
END;

CLSR_REC = RECORD;
        BLRM : BUFF8;        {*BUILDING AND ROOM NUMBER*}
        CPTY : BUFF4;        {*CAPACITY OF ROOM*}
        EQPT : BUFF2;        {*TYPE(S) OF EQUIPMENT IN ROOM*}
        TYPP : BUFF3;        {*CODE FOR TYPE OF ROOM*}
        CFLG : BUFF1;        {*CODE FOR SECURITY CLASSIFICATION LEVEL OF
END;

TCMF_REC = RECORD;
        FACT : BUFF9;        {*FACULTY SSN*}
        STDT : BUFF9;        {*STUDENT SSN*}
        THES : BUFF10;       {*DEPARTMENT THESIS NUMBER*}
END;

FADV_REC = RECORD;
        SECT : BUFF8;        {*SECT CTRL (SECT NUMBER)*}
        STDT : BUFF9;        {*STUDENT SSSN*}
        FACT : BUFF9;        {*FACULTY FSSN*}
END;
```

```
VINS_REC = RECORD;
        STDT : BUFF9;          {*STUDENT SSSN*}
        NMBR : BUFF8;          {*MCRS CTRL (COURSE NUMBER)*}
        FACT : BUFF9;          {*FACULTY SSSN*}
END;


TADV_REC = RECORD;
        THES : BUFF10;         {*IDENTIFIES  TIED TO THES NUMBER*}
        STDT : BUFF9;          {*STUDENT SSSN*}
        FACT : BUFF9;          {*FACULTY FSSN*}
END;

VPDQ_REC = RECORD;
        MQTR : BUFF4;          {*TIED TO MQTR (QUARTER NUMBER)*}
        FACT : BUFF9;          {*FACULTY SSSN*}
END;

VMSS_REC = RECORD;
        MSSF : BUFF3;          {*TIED TO MASTER COURSE SEQUENCE NUMBER*}
        NMBR : BUFF8;          {*TIED TO MASTER COURSE NUMBER*}
        MDEG : BUFF2;          {*TIED TO MASTER DEG REQUIREMENT NUMBER*}
        CRSS : BUFF30;         {*LISTS WHICH COURSES BELONG IN SEQUENCE*}
        VREF : BUFF4;          {*RECORD REFERENCE                      *}
END;

MSSF_PTR = ^MSSF_RECORD;
MSSF_RECORD = RECORD;
        CTRL : BUFF3;          {*COURSE SEQUENCE NUMBER*}
        SEQN : BUFF40;         {*SEQUENCE NAME*}
        NEXT : MSSF_PTR;
        PREV : MSSF_PTR;
END;
MDEG_PTR = ^MDEG_RECORD;
MDEG_RECORD = RECORD;
        CTRL : BUFF2;          {*NUMBER IDENTIFYING TYPE GRAD DEGREE*}
        NAME : BUFF40;         {*NAME OF TYPE OF DEGREE*}
        NEXT : MDEG_PTR;
        PREV : MDEG_PTR;
END;
LINK_PTR = ^LINK_RECORD;
LINK_RECORD = RECORD
        NAME : BUFF28;
        CTRL : BUFF9;
        RANK : BUFF3;
        DEPT : BUFF4;
        SECT : BUFF8;
        NEXT : LINK_PTR;
        PREV : LINK_PTR;
END; {* LINK_RECORD *}
```

```
{*****************************************************}
{*   THESE TYPE DECLARATIONS ARE USED IN VARIOUS   *}
{*   LIST PROCESSING THAT IS UNIQUE TO THE EDPLAN *}
{*   PROGRAM.                                      *}
{*****************************************************}

   SECT_ARRAY = ARRAY [1..100] OF SECT_REC;
   LINK_ARRAY = ARRAY [1..100] OF LINK_RECORD;
   CRSE_ARRAY = ARRAY [1..30] OF CRSE_REC;
   VCQR_ARRAY = ARRAY [1..120] OF VCQR_REC;
   VMSS_ARRAY = ARRAY [1..30] OF VMSS_REC;
```

Appendix D

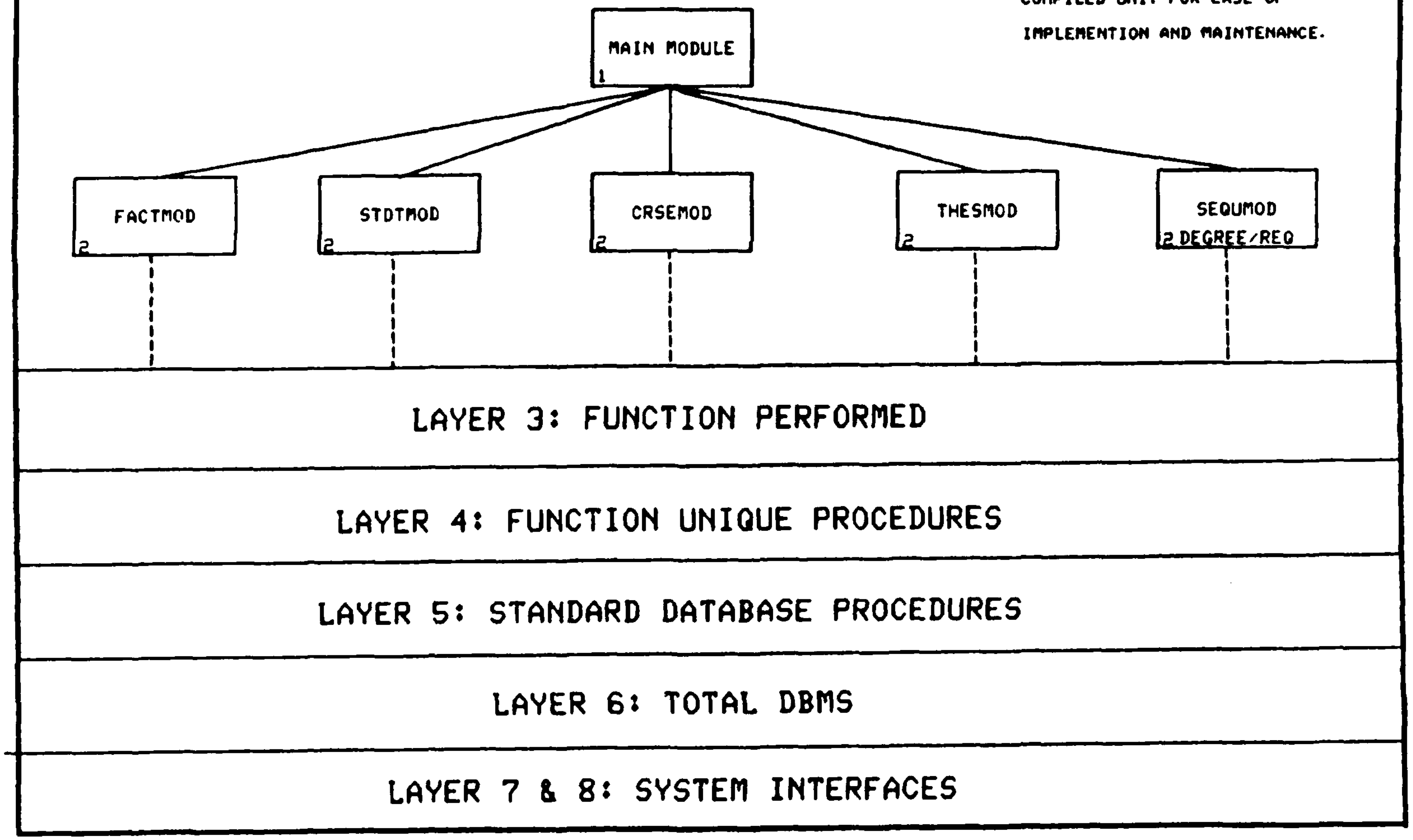
SYSTEM FLOWCHARTS

This portion contains the structure charts for the anticipated application program structure. The charts are intended to describe the system in terms of a map that the systems analyst and users can follow to a specific function. The first chart (page D-2) depicts the system in terms of the seperate applications programs. FACTMOD, STDTMOD, CRSEMOD..etc are seperately compiled programs spawned as seperate processes by a main program. The standard database routines have been omitted because they are considered abstractions of a data structure which in this case is the TOTAL Database Management system.

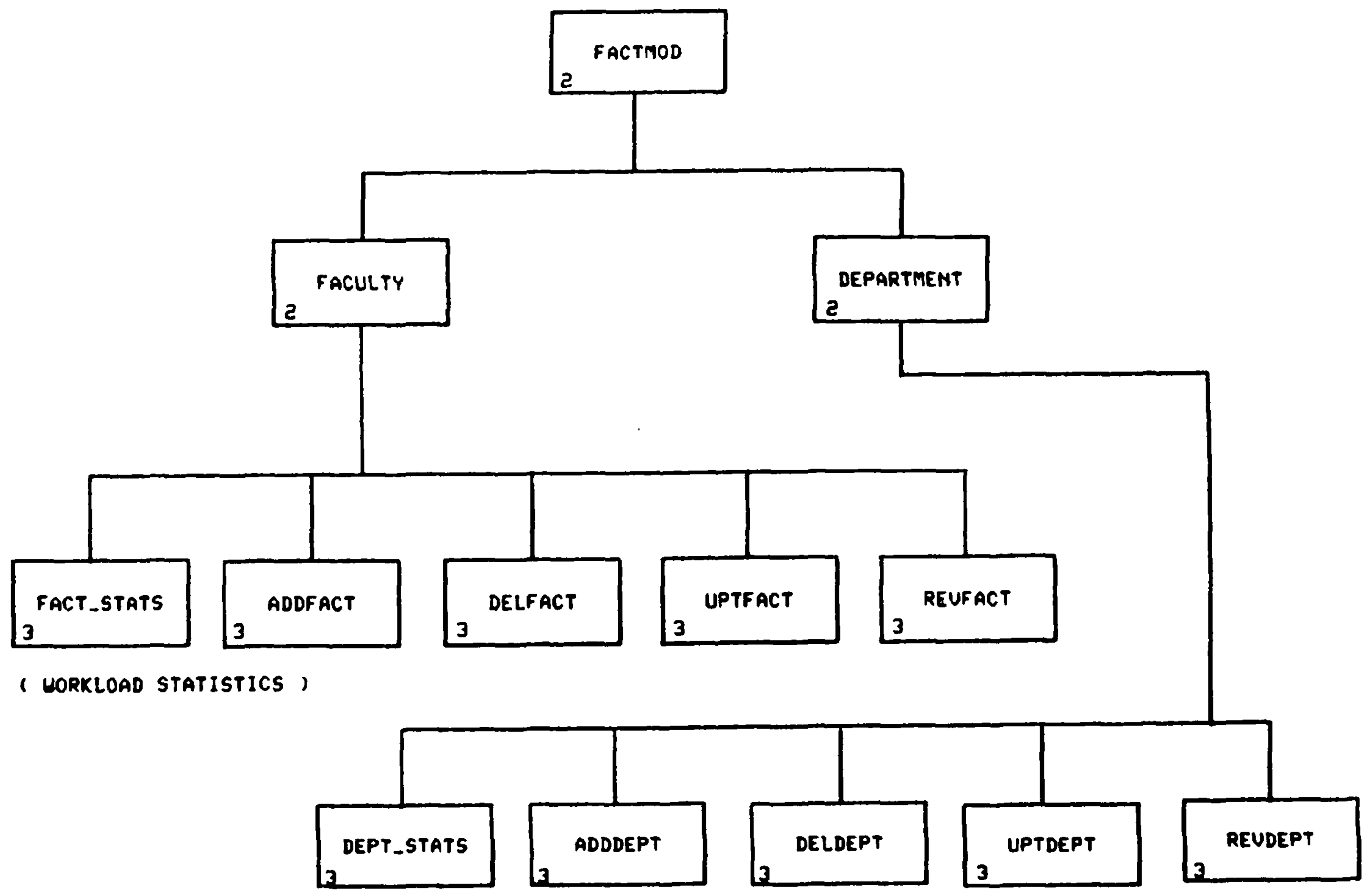
AFIT/EN DATABASE SYSTEM

COMPLETE SYSTEM OVERVIEW

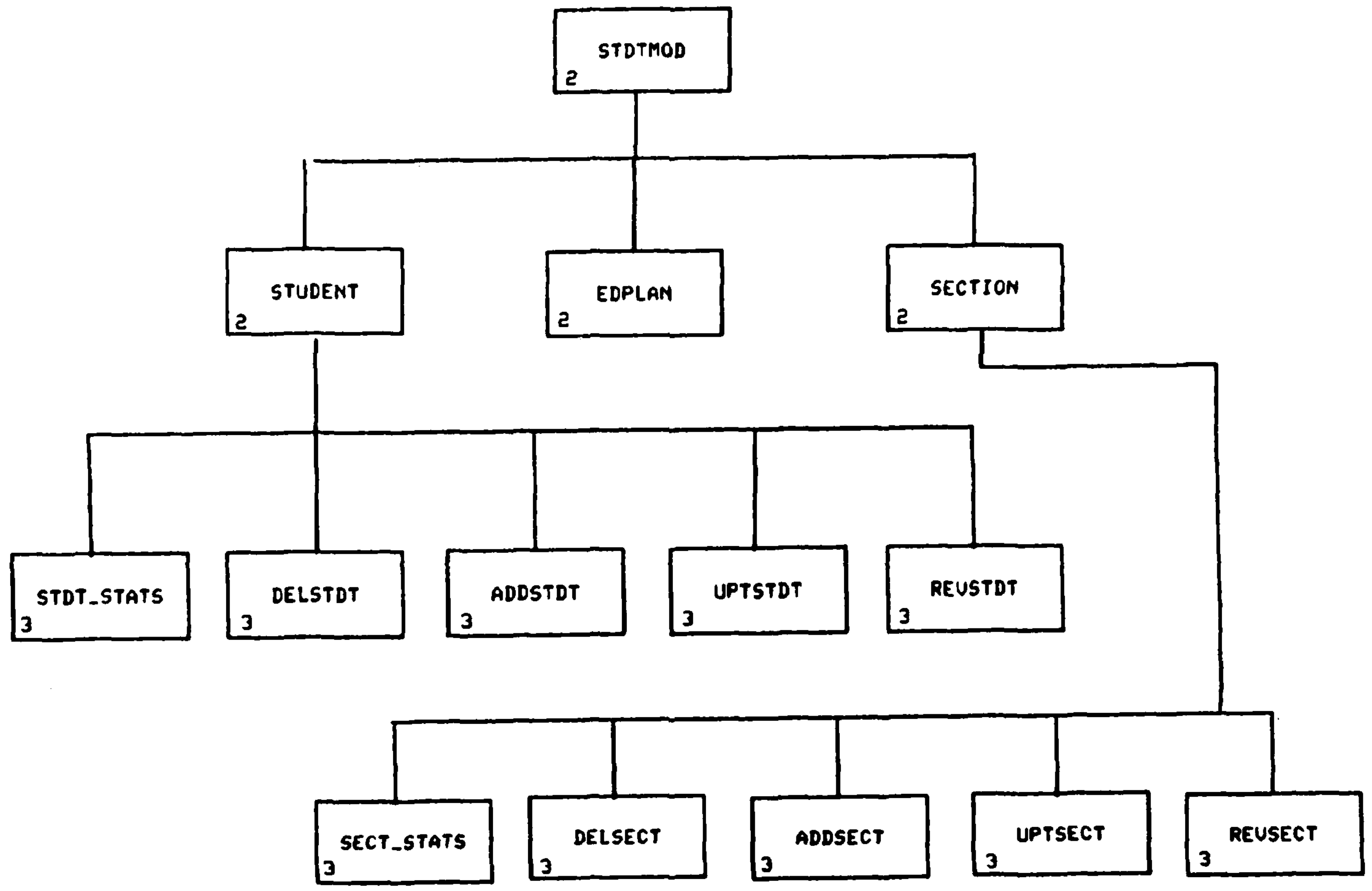
NOTE: EACH MODULE IS A SEPERATELY COMPILED UNIT FOR EASE OF IMPLEMENTION AND MAINTENANCE.


MAIN MODULE
1
FACTMOD
2
STDTMOD
2
CRSEMOD
2
THESMOD
2
SEQUMOD
2 DEGREE/REQ
LAYER 3: FUNCTION PERFORMED
LAYER 4: FUNCTION UNIQUE PROCEDURES
LAYER 5: STANDARD DATABASE PROCEDURES
LAYER 6: TOTAL DBMS
LAYER 7 & 8: SYSTEM INTERFACES

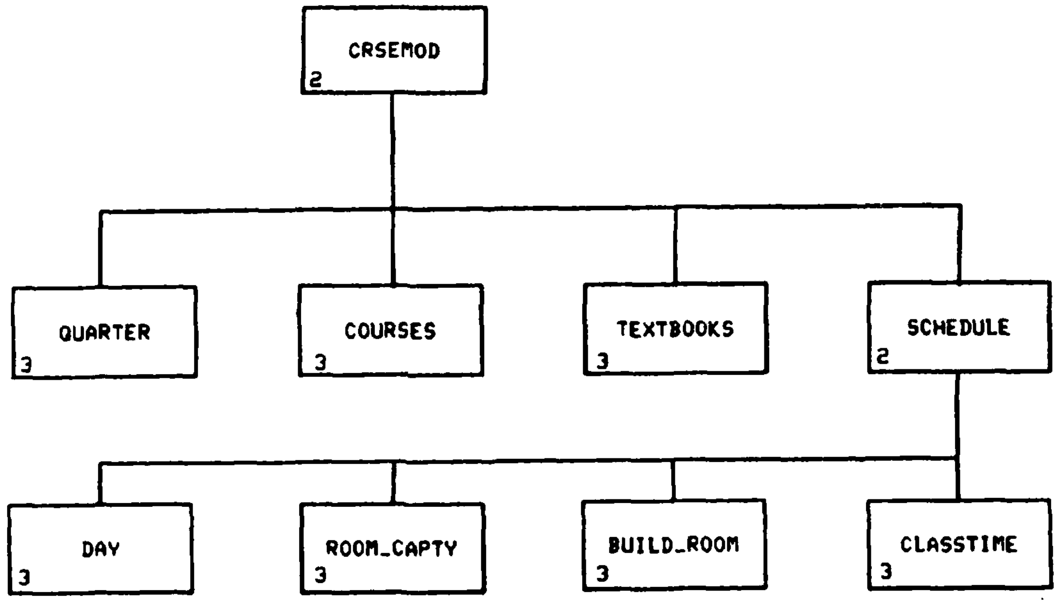
FACULTY AND DEPARTMENT MODULES

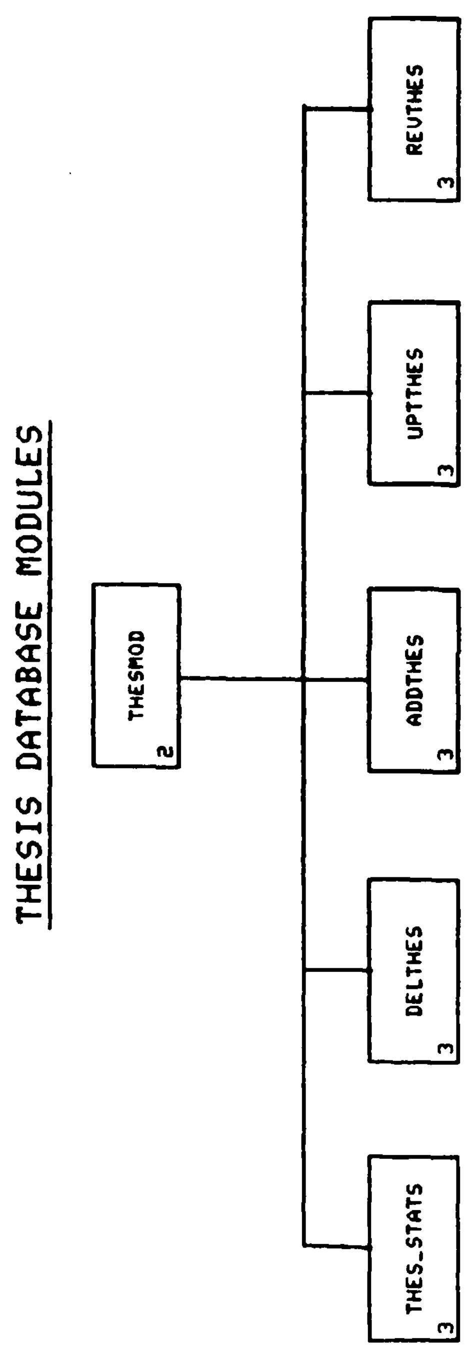


STUDENT AND SECTION MODULES

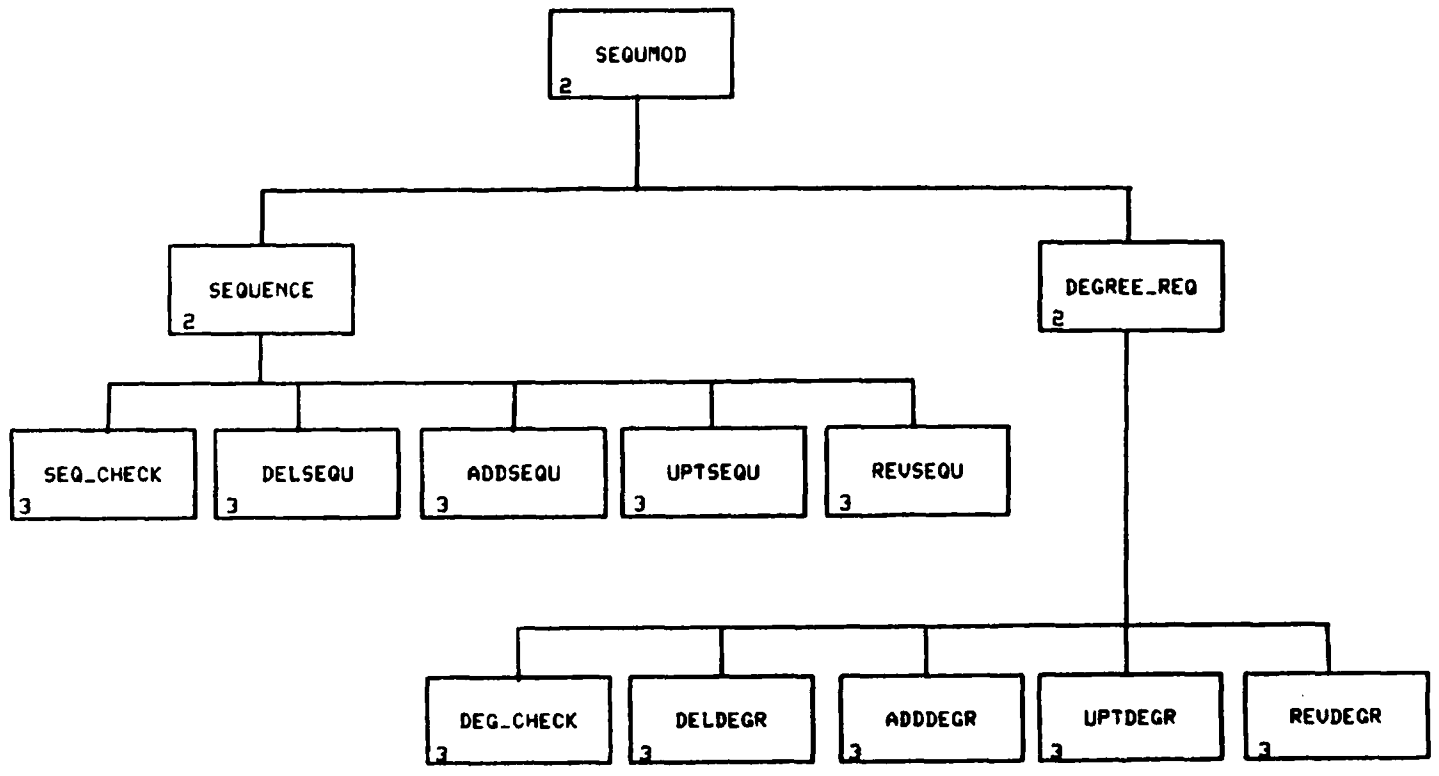


COURSE DATABASE MODULES

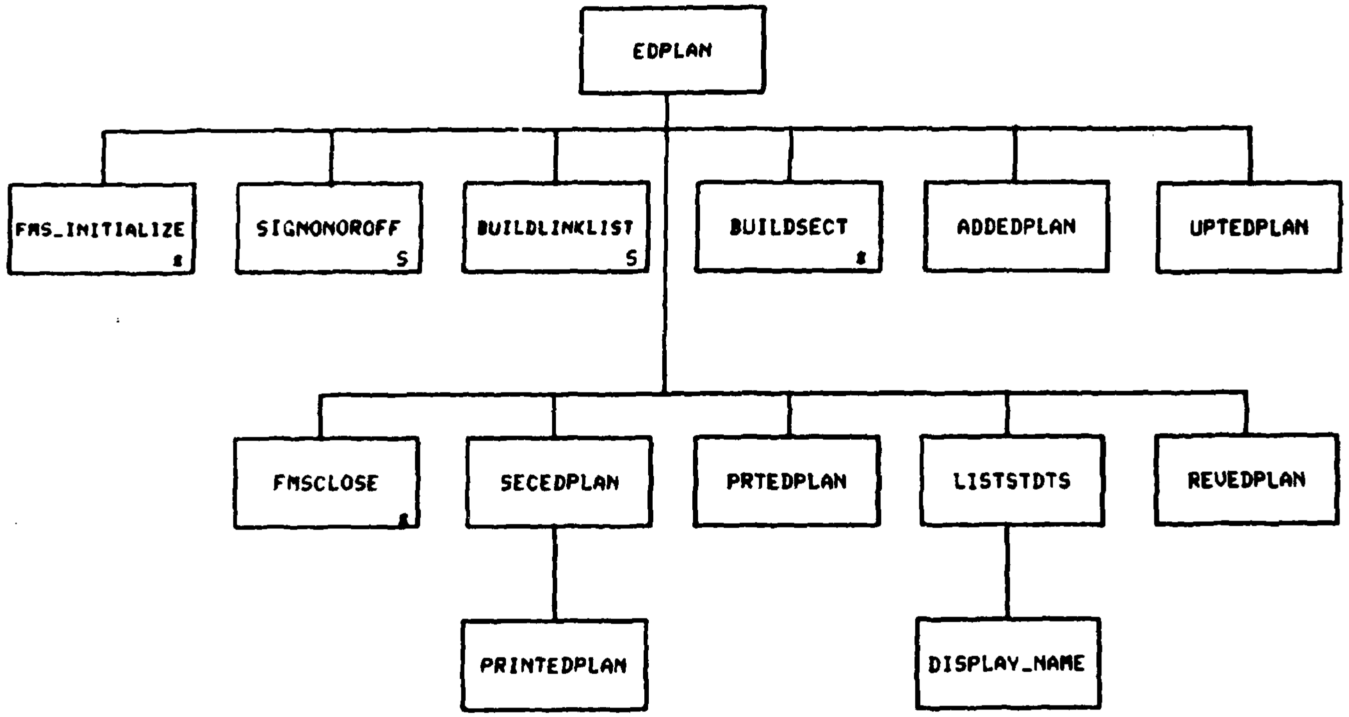



THESIS DATABASE MODULES

SEQUENCE/DEGREE REQUIREMENT MODULE

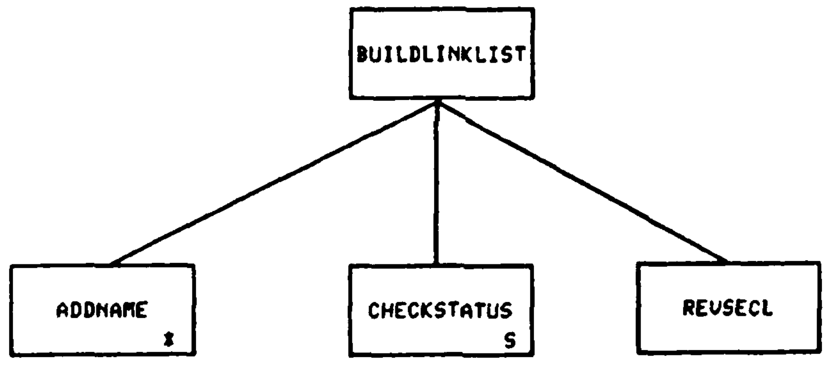


EDUCATION PLAN MODULE (1)



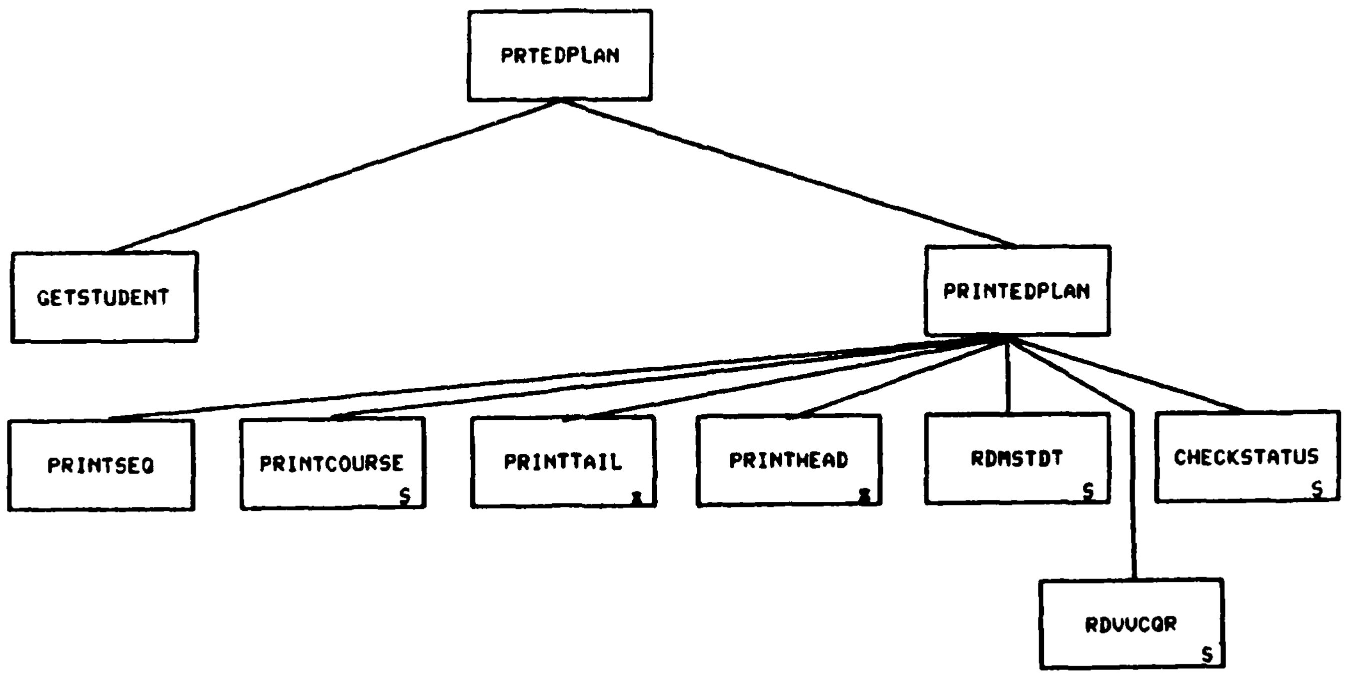
NOTE: THE '*' INDICATES THIS MODULE MAKES NO CALLS.
AN 'S' INDICATES CALLS TO STANDARD DBMS ROUTINES

EDUCATION PLAN MODULE (2)



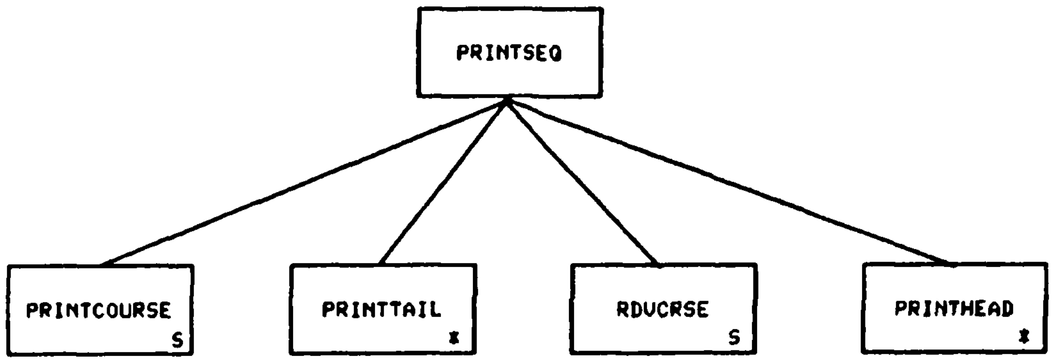
NOTE: AN '*' INDICATES NO MODULE CALLS ARE MADE
AN 'S' INDICATES ONLY STANDARD DBMS CALLS ARE MADE

EDUCATION PLAN MODULE (3)



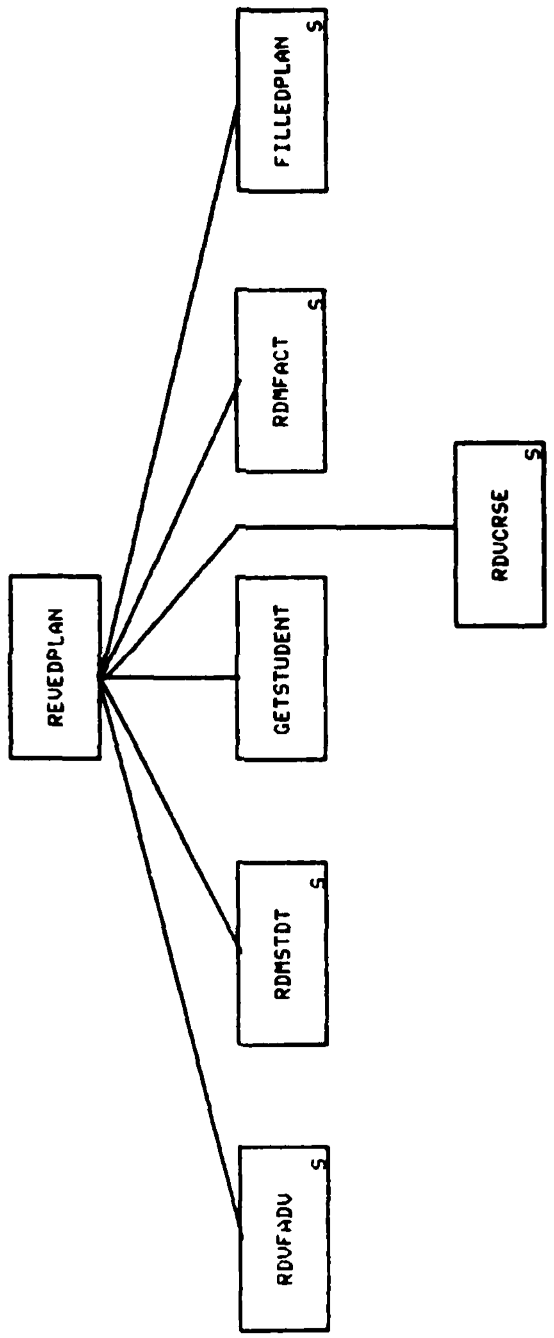
NOTE: A '*' INDICATES THIS ROUTINE MAKES NO CALLS.
AN 'S' INDICATES ONLY STANDARD DBMS CALLS ARE MADE

EDUCATION PLAN MODULE (4)

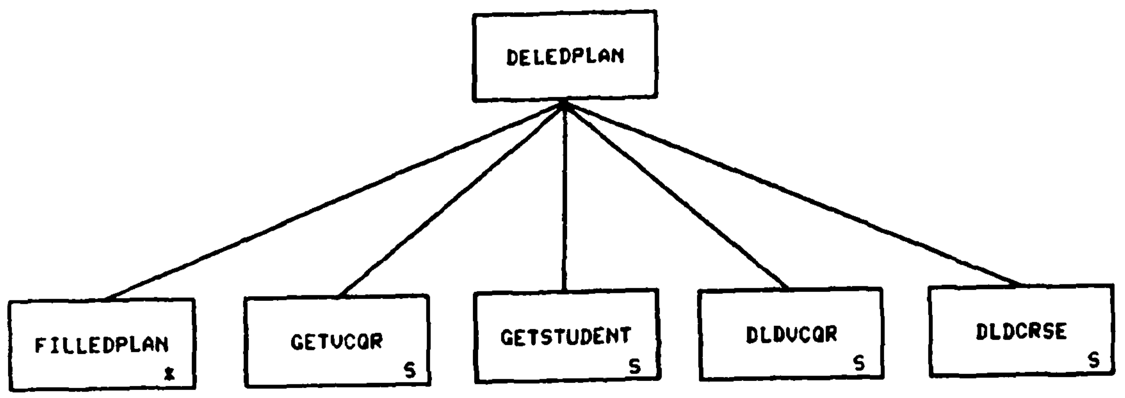


NOTE: AN '*' INDICATES THIS MODULE MAKES NO CALLS
AN 'S' INDACATES STANDARD DBMS CALLS ONLY

EDUCATION PLAN MODULE (5)



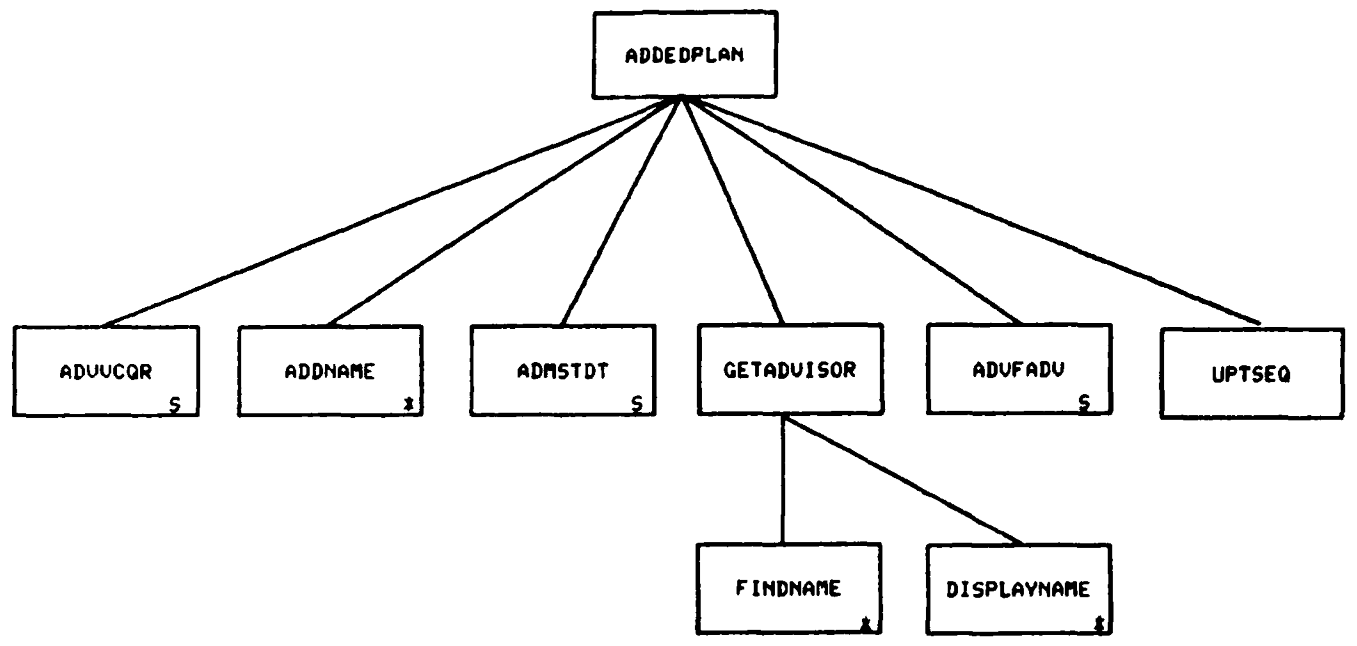
EDUCATION PLAN MODULE (6)



NOTE: AN '*' INDICATES THIS MODULE MAKES NO CALLS

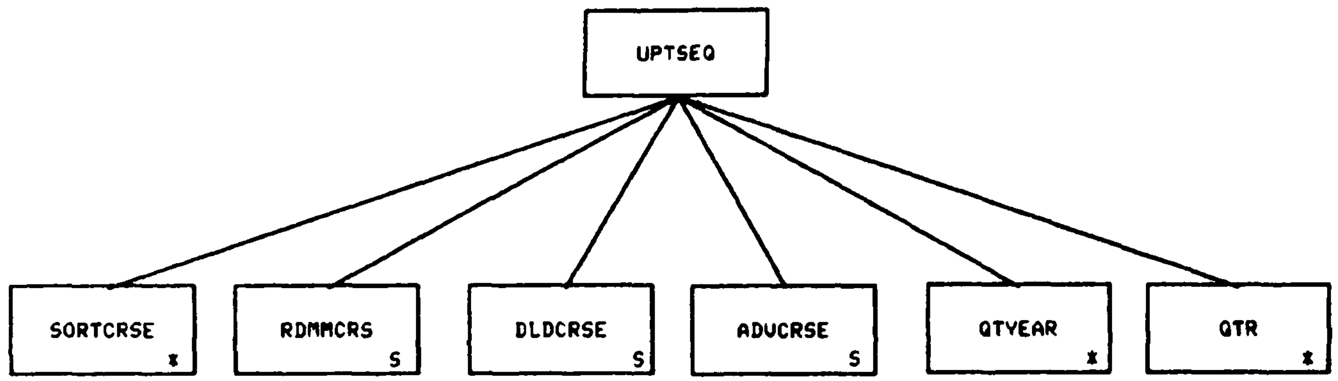
AN 'S' INDICATES CALLS TO STANDARD DBMS ROUTINES

EDUCATION PLAN MODULE (7)



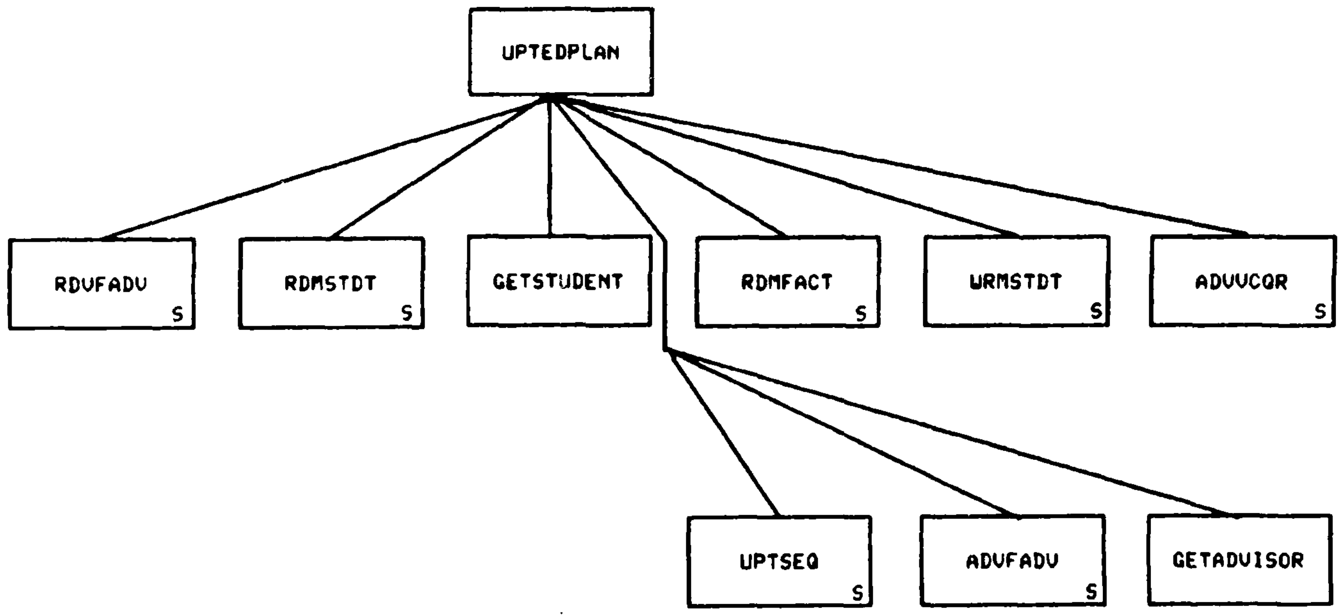
NOTE: AN '*' INDICATES NO CALS ARE MADE
AN 'S' INDICATES STANDARD DBMS CALLS ONLY

EDUCATION PLAN MODULE (8)



NOTE: AN '*' INDICATES NO MODULES CALLS ARE MADE
AN 'S' INDICATES STANDARD DBMS CALLS ONLY

EDUCATION PLAN MODULE (9)



NOTE: '*' INDICATES NO SUBROUTINE CALLS MADE.
AN 'S' MEANS CALLS TO STANDARD DBMS ROUTINES ONLY

Appendix E

AFITDB Frames Descriptions

The following is a compilation of the FMS form screens contained within the library which supports the AFIT/ENG DBMS. Each screen must be resident within a library in order to require only one open and close statement within a program. The listings were created using the instruction FMS/DESCRIPTION/IMAGE "formname". This creates a file called formname.lis which can then be printed in the format described in this appendix.

The three column format is produced using the fms form utility (FUT). The date column lists the latest date changes were made to each form, while column three shows the work space area , in bytes, reserved for each screen within the FMS driver work space area.

```
Form Application Aids V2.2
19-NOV-1985 08:58

Library DUA1:[PANGMAN.AFITDB]STDTMOD.FLB;1,
 created: 12-SEP-1985 13:13

Date and time of last modification: 19-NOV-1985 08:57

Form name      Creation date/time    Workspace size (bytes)

EDPLAN1         19-NOV-1985 08:57         5779
NAMESEL         18-SEP-1985 11:36         1029
HELPEDPL        18-SEP-1985 13:01         1109
EDPLAN          19-NOV-1985 08:32         1359
GETNAME          2-OCT-1985 12:39         1215
DELEDPLAN       29-OCT-1985 09:21         781
NAMESECT        29-OCT-1985 09:43         1003
GETSECT          2-OCT-1985 12:37         987
SELSECT          2-OCT-1985 10:59         1019
```

Form name	Creation date/time	Workspace size (bytes)
STUDSEL	29-OCT-1985 09:34	1103
PRINTOPT	18-OCT-1985 14:29	585
TAPESEL	29-OCT-1985 09:33	1029
SELALL	19-NOV-1985 08:34	773
WORKON	22-OCT-1985 08:32	855
DELNAME	24-OCT-1985 11:56	631
STDTMENU	24-OCT-1985 10:57	615
DELMENU	24-OCT-1985 11:07	659
DELSECT	24-OCT-1985 11:59	531
WORKDEL	24-OCT-1985 12:01	409
ALLORSOME	29-OCT-1985 09:20	675
WORKONSECT	25-OCT-1985 11:12	403
SEQHELP	19-NOV-1985 08:37	1021
SEQABWO2	19-NOV-1985 08:41	3705
NAMESECT2	29-OCT-1985 09:47	1043
LISTSECT	29-OCT-1985 10:25	711
SEQPAGE	30-OCT-1985 08:53	2139
LOADING	6-NOV-1985 09:48	645
GRAPHMENU	8-NOV-1985 09:47	693
NOSCREEN	8-NOV-1985 10:25	159
GRAPHSEL	7-NOV-1985 13:33	989
EDPLANHELP	19-NOV-1985 08:50	879
EDPLANHELP2	19-NOV-1985 08:55	1153

Form: EDPLAN1

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
 1|                         AFITDB/ENG:STUDENT EDPLAN                 V2.2       |1
 2|  NAME  :                                                                     |2
 3|SSAN:          |AFSC:    | ACADEMIC ADVISOR:                           | HELP  |3
 4|RANK:          | AEC:    |     SECTION CODE:                           |USE PF2|4
 5|BOXN:          |         |                                             |       |5
 6|                                                                              |6
 7|------------------------------------------------------------------------------|7
 8| QTR          -----------      COURSES         -------------------           |8
 9|       |        |        |        |        |        |        |        |       ||9
10|       |        |        |        |        |        |        |        |       ||10
11|       |        |        |        |        |        |        |        |       ||11
12|       |        |        |        |        |        |        |        |       ||12
13|       |        |        |        |        |        |        |        |       ||13
14|       |        |        |        |        |        |        |        |       ||14
15|       |        |        |        |        |        |        |        |       ||15
16|       |        |        |        |        |        |        |        |       ||16
17|       |        |        |        |        |        |        |        |       ||17
18|       |        |        |        |        |        |        |        |       ||18
19|       |        |        |        |        |        |        |        |       ||19
20|       |        |        |        |        |        |        |        |       ||20
21|       |        |        |        |        |        |        |        |       ||21
22|       |        |        |        |        |        |        |        |       ||22
23|       |        |        |        |        |        |        |        |       ||23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: NAMESEL

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
 1|                                                                              |1
 2|                                                                              |2
 3|                           SELECT FACULTY ADVISOR                             |3
 4|                                                                              |4
 5|                 A UNIQUE NAME COULD NOT BE FOUND GIVEN YOUR                  |5
 6|                 INPUT.  THE FACULTY MEMBERS BELOW MATCH YOUR                 |6
 7|                 INPUT.  PLEASE SELECT ONE OF THEM.                           |7
 8|                                                                              |8
 9|                  +-----------------------------------------+                 |9
10|                  |                                         |                 |10
11|                  |                                         |                 |11
12|                  |                                         |                 |12
13|                  |                                         |                 |13
14|                  |                                         |                 |14
15|                  |                                         |                 |15
16|                  |                                         |                 |16
17|                  |                                         |                 |17
18|                  +-----------------------------------------+                 |18
19|                         +-----------------------------------------------------+|19
20|                         |  USE ARROW KEYS TO SCROLL THE SCREEN.  ENTER AN     ||20
21|                         |  "X" TO SELECT A RECORD.  TO EXIT WITHOUT SELECT-   ||21
22|                         |  ING A RECORD, HIT RETURN.                          ||22
23|                         +-----------------------------------------------------+|23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: HELPEDPL

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
 1|                                                                                |1
 2|          LISTED BELOW ARE BRIEF DESCRIPTIONS OF ALL OF THE SELECTIONS          |2
 3|                                                                                |3
 4|          1.  THIS MODULE ADDS A NEW STUDENT TO THE DATABASE SYSTEM AND         |4
 5|              THE STUDENTS EDUCATION PLAN.  THE STUDENT SHOULD KNOW             |5
 6|              HIS/HER SOCIAL SECURITY NUMBER, SECTION, AND FACULTY ADVISOR.     |6
 7|                                                                                |7
 8|          2.  THIS MODULE WILL ALLOW THE USER TO CHANGE ALL OF THE INFORMATION  |8
 9|              ENTERED IN  THE ADD ROUTINE EXCEPT THE SOCIAL SECURITY NUMBER.    |9
10|                                                                                |10
11|          3.  TO DELETE AN EDUCATION PLAN FOR A STUDENT, USE THIS SELECTION.    |11
12|              NOTE: THIS WILL NOT DELETE THE STUDENT FROM THE DATABASE.         |12
13|                                                                                |13
14|          4.  THIS IS THE SAME AS THE UPDATE ONLY NO CHANGES WILL BE PERMITTED. |14
15|                                                                                |15
16|          5.  THIS MODULE WILL LIST THE STUDENTS IN THE DATABASE BY SECTION     |16
17|              NAME.                                                             |17
18|                                                                                |18
19|          6.  THIS MODULE WILL PRINT AN INDIVIDUALS EDUCATION PLAN.             |19
20|                                                                                |20
21|          7.  THIS MODULE PRINTS AN ENTIRE SECTIONS EDUCATION PLAN.             |21
22|                                                                                |22
23|                                                                                |23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: EDPLAN

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
 1|                                AFITDB DATABASE V2.2                            |1
 2|                              STUDENT EDUCATION PLANS                           |2
 3|               +------------------------------------------------+              |3
 4|               | 1.      ADD A NEW STUDENT AND EDUCATION PLAN   |              |4
 5|               |                                                |              |5
 6|               | 2.      UPDATE A STUDENT'S EDUCATION PLAN      |              |6
 7|               |                                                |              |7
 8|               | 3.      DELETE A STUDENT'S EDUCATION PLAN      |              |8
 9|               |                                                |              |9
10|               | 4.      REVIEW STUDENT'S EDUCATION PLAN        |              |10
11|               |                                                |              |11
12|               | 5.      LIST STUDENTS BY SECTION               |              |12
13|               |                                                |              |13
14|               | 6.      PRINT A STUDENT'S EDUCATION PLAN       |              |14
15|               |                                                |              |15
16|               | 7.      PRINT A SECTION'S EDUCATION PLAN       |              |16
17|               |                                                |              |17
18|               | 8.      CREATE REGISTRAR SUMMARY FILE.         |              |18
19|               |                                                |              |19
20|               | 9.      EXIT TO PREVIOUS MENU                  |              |20
21|               +------------------------------------------------+              |21
22|                SELECT OPTION (1-9)==>                                          |22
23|                                                                                |23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: GETNAME

```
          1         2         3         4         5         6         7         8
 12345678901234567890123456789012345678901234567890123456789012345678901234567890
 --------------------------------------------------------------------------------
  1|                                                                              |1
  2|                                                                              |2
  3|                      ENTER THE LAST NAME OF THE STUDENT                      |3
  4|                                                                              |4
  5|                                                                              |5
  6|                                                                              |6
  7|                       +--------------------------------+                     |7
  8|                       |                                |                     |8
  9|                       +--------------------------------+                     |9
 10|                                                                              |10
 11|                                                                              |11
 12|           ENTER THE LAST NAME OF THE STUDENT WHOSE RECORD YOU WISH TO        |12
 13|        WORK WITH.  IF ONLY ONE STUDENT IS FOUND WITH THAT LAST NAME, THEN    |13
 14|        HIS/HER RECORD WILL BE READ AND PRESENTED. IF MORE THAN ONE STUDENT   |14
 15|        HAS THAT LAST NAME, ANOTHER SCREEN WILL APPEAR WITH THOSE STUDENTS    |15
 16|        THAT MATCH THE ENTRY ABOVE.  YOU NEED NOT ENTER THE ENTIRE LAST NAME. |16
 17|        ENTER AS MUCH OF THE NAME AS POSSIBLE TO LIMIT THE POSIBILITIES.  TO  |17
 18|        EXIT THIS SCREEN, HIT RETURN WITHOUT ENTERING ANY CHARACTERS.         |18
 19|                                                                   +-+        |19
 20|        DO YOU WISH TO PRINT THE LAST PAGE OF THE EDPLAN REPORT(Y,N) | |        |20
 21|                                                                   +-+        |21
 22|                                                                              |22
 23|                                                                              |23
 --------------------------------------------------------------------------------
 12345678901234567890123456789012345678901234567890123456789012345678901234567890
          1         2         3         4         5         6         7         8
```

Form: DELEDPLAN

```
          1         2         3         4         5         6         7         8
 12345678901234567890123456789012345678901234567890123456789012345678901234567890
 --------------------------------------------------------------------------------
  1|                          STUDENT EDPLAN DELETION                             |1
  2|                                                                              |2
  3|                                                                              |3
  4|                                                                              |4
  5|                                    +-----------------------------+           |5
  6|               STUDENT'S LAST NAME: |                             |           |6
  7|                                    +-----------------------------+           |7
  8|                                                                              |8
  9|             ENTER THE WHOLE OR PARTIAL LAST NAME OF THE STUDENT WHOSE        |9
 10|        EDPLAN YOU WISH TO DELETE.  NOTE THAT THE EDPLAN AND ONLY THE         |10
 11|        EDPLAN WILL BE DELETED.  THE STUDENTS RECORD WILL REMAIN.  TO         |11
 12|        EXIT THIS SCREEN WITH NO CHANGE, HIT RETURN WITHOUT ENTERING ANY      |12
 13|        CHARACTERS.                                                           |13
 14|                                                                              |14
 15|                                                                              |15
 16|                                                                              |16
 17|                                                                              |17
 18|                                                                              |18
 19|                                                                              |19
 20|                                                                              |20
 21|                                                                              |21
 22|                                                                              |22
 23|                                                                              |23
 --------------------------------------------------------------------------------
 12345678901234567890123456789012345678901234567890123456789012345678901234567890
          1         2         3         4         5         6         7         8
```

Form: NAMESECT

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
 1|                                                                                |1
 2|                                                                                |2
 3|                         LIST OF STUDENT BY SECTION                             |3
 4|                                                                                |4
 5|                                DISPLAY ONLY                                    |5
 6|                                                                                |6
 7|                             SECTION:                                           |7
 8|                                                                                |8
 9|                  +-----------------------------------------+                   |9
10|                  |                                         |                   |10
11|                  |                                         |                   |11
12|                  |                                         |                   |12
13|                  |                                         |                   |13
14|                  |                                         |                   |14
15|                  |                                         |                   |15
16|                  |                                         |                   |16
17|                  |                                         |                   |17
18|                  +------+----------------------------------+------------------+|18
19|                         |  USE ARROW KEYS TO SCROLL THE SCREEN.  HIT RETURN  ||19
20|                         |  TO EXIT THE SCREEN.  NOTE: THIS IS A DISPLAY OF   ||20
21|                         |  INFORMATION ONLY. USE 'E' TO EXIT THE SCREEN OR   ||21
22|                         |  HIT THE RETURN KEY.                               ||22
23|                         +----------------------------------------------------+|23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: GETSECT

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
 1|                                                                                |1
 2|                                                                                |2
 3|                               ENTER SECTION CODE                               |3
 4|                                                                                |4
 5|                                                                                |5
 6|                                      +----------+                              |6
 7|                           SECTION =  |          |                              |7
 8|                                      +----------+                              |8
 9|                                                                                |9
10|                ENTER THE SECTION CODE OF THE CLASS YOU WISH TO WORK WITH.  IF  |10
11|         THE SECTION YOU ENTERED IS NOT A VALID SECTION THE PROGRAM WILL RETURN |11
12|         TO THE PREVIOUS MENU.                                                  |12
13|                                                                                |13
14|                                  +------------------------------+              |14
15|          WORKING ON STUDENT =    |                              |              |15
16|                                  +------------------------------+              |16
17|                                                                  +----+        |17
18|          WOULD YOU LIKE THE SECOND PAGE OF THE REPORT (Y,N)      | N  |        |18
19|                                                                  +----+        |19
20|                                                                                |20
21|                                                                                |21
22|                                                                                |22
23|                                                                                |23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: SELSECT

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
 1|                                                                                |1
 2|                                                                                |2
 3|                             SELECT CLASS SECTION                               |3
 4|                                                                                |4
 5|                                                                                |5
 6|      +---------------------------------------------------------------------+   |6
 7|      |                                                                     |   |7
 8|      |                                                                     |   |8
 9|      |                                                                     |   |9
10|      |                                                                     |   |10
11|      |                                                                     |   |11
12|      |                                                                     |   |12
13|      |                                                                     |   |13
14|      |                                                                     |   |14
15|      |                                                                     |   |15
16|      |                                                                     |   |16
17|      |                                                                     |   |17
18|      +---------------+-----------------------------------------------------+   |18
19|                      |      USE THE TAB KEY AND UP AND DOWN ARROW KEYS     |   |19
20|                      | TO MOVE THROUGH THE SCREEN.  PUT AN "X" BESIDE THE  |   |20
21|                      | SECTION YOU WISH TO WORK WITH OR PRINT.  HIT RETURN |   |21
22|                      | WITHOUT ENTERING ANYTHING TO EXIT THIS FUNCTION,    |   |22
23|                      +-----------------------------------------------------+   |23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: STUDSEL

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
 1|                                                                                |1
 2|                                                                                |2
 3|                            SELECT STUDENT RECORD                               |3
 4|                                                                                |4
 5|                  A UNIQUE NAME COULD NOT BE FOUND GIVEN YOUR                   |5
 6|                  INPUT.  THE STUDENT LIST BELOW MATCH YOUR                     |6
 7|                  INPUT.  PLEASE SELECT ONE OF THEM OR HIT                      |7
 8|                  HIT RETURN TO EXIT WITHOUT SELECTING A NAME.                  |8
 9|                   +----------------------------------------+                   |9
10|                   |                                        |                   |10
11|                   |                                        |                   |11
12|                   |                                        |                   |12
13|                   |                                        |                   |13
14|                   |                                        |                   |14
15|                   |                                        |                   |15
16|                   |                                        |                   |16
17|                   |                                        |                   |17
18|                   +----------------------------------------+                   |18
19|                          +----------------------------------------------------+|19
20|                          |  USE ARROW KEYS TO SCROLL THE SCREEN.  ENTER AN    ||20
21|                          |  "X" TO SELECT A RECORD.  TO EXIT WITHOUT SELECT-  ||21
22|                          |  ING A RECORD, HIT RETURN WITH NO "X" ENTRY.       ||22
23|                          +----------------------------------------------------+|23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: PRINTOPT

```
           1         2         3         4         5         6         7         8
  12345678901234567890123456789012345678901234567890123456789012345678901234567890
  --------------------------------------------------------------------------------
 1|                                                                                |1
 2|                                                                                |2
 3|                                                                                |3
 4|                        EDUCATION PLAN PRINT OPTION                             |4
 5|                                                                                |5
 6|                                                                                |6
 7|                                                                                |7
 8|            WOULD YOU LIKE TO PRINT THIS EDUCATION PLAN (Y,N) N                 |8
 9|                                                                                |9
10|                                                                                |10
11|            WOULD YOU LIKE TO PRINT THE SEQUENCE PAGE (Y,N) N                   |11
12|                                                                                |12
13|                                                                                |13
14|                                                                                |14
15|                                                                                |15
16|                                                                                |16
17|                                                                                |17
18|                                                                                |18
19|                                                                                |19
20|                                                                                |20
21|                                                                                |21
22|                                                                                |22
23|                                                                                |23
  --------------------------------------------------------------------------------
  12345678901234567890123456789012345678901234567890123456789012345678901234567890
           1         2         3         4         5         6         7         8
```

Form: TAPESEL

```
           1         2         3         4         5         6         7         8
  12345678901234567890123456789012345678901234567890123456789012345678901234567890
  --------------------------------------------------------------------------------
 1|                                                                                |1
 2|                                                                                |2
 3|                        REGISTRAR TAPE BUILD FUNCTION                           |3
 4|                                                                                |4
 5|                      (CREATES STUDENT SUMMARY.DAT FILE )                       |5
 6|                                                                                |6
 7|                          +-------------------------+                           |7
 8|                          |        ===>             |                           |8
 9|                          |        ===>             |                           |9
10|                          |        ===>             |                           |10
11|                          |        ===>             |                           |11
12|                          |        ===>             |                           |12
13|                          |        ===>             |                           |13
14|                          |        ===>             |                           |14
15|                          |        ===>             |                           |15
16|                          |        ===>             |                           |16
17|             +------------+-------------------------+------------+              |17
18|             | PLACE AN 'X' BESIDE THE SECTIONS YOU WISH         |              |18
19|             | TO GENERATE A SUMMARY FILE FOR.  YOU MAY          |              |19
20|             | UN-SELECT A SECTION BY PLACING A SPACE IN THE     |              |20
21|             | MARKER TO THE LEFT. TO EXIT, HIT RETURN, TO       |              |21
22|             | EXIT AND CANCEL , ENTER AND 'E'.                  |              |22
23|             +---------------------------------------------------+              |23
  --------------------------------------------------------------------------------
  12345678901234567890123456789012345678901234567890123456789012345678901234567890
           1         2         3         4         5         6         7         8
```

Form: SELALL

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
  1|                                                                                |1
  2|                                                                                |2
  3|                        SUMMARY FILE GENERATION FUNCTION                        |3
  4|                                                                                |4
  5|                                                                                |5
  6|        THIS MODULE ALLOWS THE USER TO GENERATE THE SUMMARY FILE NEEDED         |6
  7|        BY THE REGISTRARS OFFICE TO SCHEDULE CLASS. SUMMARYS CAN BE             |7
  8|        GENERATED FOR ALL OF THE SECTIONS OR JUST SELECTED SECTIONS.            |8
  9|                                                                                |9
 10|                                                                                |10
 11|                        1       GENERATE SUMMARY FILE FOR ALL                   |11
 12|                                SECTIONS.                                       |12
 13|                                                                                |13
 14|                        2       SELECT SPECIFIC SECTIONS TO BE                  |14
 15|                                GENERATED.                                      |15
 16|                                                                                |16
 17|                        9       EXIT TO PREVIOUS MENU.                          |17
 18|                                                                                |18
 19|                        SELECT OPTION (1,2,9)==>                                |19
 20|                                                                                |20
 21|                                                                                |21
 22|                                                                                |22
 23|                                                                                |23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: WORKON

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
  1|                                                                                |1
  2|                           SUMMARY FILE GENERATION                              |2
  3|                                                                                |3
  4|                                                                                |4
  5|        +--------------------------------------------------------------+        |5
  6|        |                   +--------+                                 |        |6
  7|        |       SECTION ==> |        |                                 |        |7
  8|        |                   +--------+                                 |        |8
  9|        |                            +------------------------------+  |        |9
 10|        |       PRINTING STUDENT ==> |                              |  |        |10
 11|        |                            +------------------------------+  |        |11
 12|        |                                                              |        |12
 13|        +--------------------------------------------------------------+        |13
 14|                                                                                |14
 15|                                                                                |15
 16|                                                                                |16
 17|                                                                                |17
 18|                                                                                |18
 19|                                                                                |19
 20|                                                                                |20
 21|                                                                                |21
 22|                                                                                |22
 23|                                                                                |23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: DELNAME

```
                 1         2         3         4         5         6         7         8
        12345678901234567890123456789012345678901234567890123456789012345678901234567890
        --------------------------------------------------------------------------------
      1|                                                                                |1
      2|                                                                                |2
      3|                         ENTER THE STUDENTS LAST NAME                           |3
      4|                                                                                |4
      5|                                YOU WISH TO DELETE                              |5
      6|                                                                                |6
      7|                                                                                |7
      8|                                                                                |8
      9|                                                                                |9
     10|                    +-------------------------------+                           |10
     11|            NAME==> |                               |                           |11
     12|                    +-------------------------------+                           |12
     13|                                                                   +-+          |13
     14|            ARE YOU SURE YOU WANT TO DELETE THIS RECORD (Y,N)      | |          |14
     15|                                                                   +-+          |15
     16|                                                                                |16
     17|                                                                                |17
     18|                                                                                |18
     19|                                                                                |19
     20|                                                                                |20
     21|                                                                                |21
     22|                                                                                |22
     23|                                                                                |23
        --------------------------------------------------------------------------------
        12345678901234567890123456789012345678901234567890123456789012345678901234567890
                 1         2         3         4         5         6         7         8
```

Form: STDTMENU

```
                 1         2         3         4         5         6         7         8
        12345678901234567890123456789012345678901234567890123456789012345678901234567890
        --------------------------------------------------------------------------------
      1|                                                                                |1
      2|                                                                                |2
      3|                         AFITDB/ENG STUDENT DATABASE                            |3
      4|                                                                                |4
      5|                                                                                |5
      6|                                                                                |6
      7|                 1.      ADD A STUDENT TO THE DATABASE.                         |7
      8|                                                                                |8
      9|                 2.      UPDATE A STUDENTS PERSONNEL RECORD.                    |9
     10|                                                                                |10
     11|                 3.      DEL A STUDENT(S) FROM THE DATABASE (ARCHIVE).          |11
     12|                                                                                |12
     13|                 4.      UPDATE A STUDENTS PERSONNEL RECORD.                    |13
     14|                                                                                |14
     15|                 5.      RUN THE EDUCATION PLAN PROGRAM.                        |15
     16|                                                                                |16
     17|                 9.      EXIT TO PREVIOUS MENU.                                 |17
     18|                                                                                |18
     19|                 SELECT OPTION (1-5,9)==>                                       |19
     20|                                                                                |20
     21|                                                                                |21
     22|                                                                                |22
     23|                                                                                |23
        --------------------------------------------------------------------------------
        12345678901234567890123456789012345678901234567890123456789012345678901234567890
                 1         2         3         4         5         6         7         8
```

Form: DELMENU

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
 1|                                                                                |1
 2|                                                                                |2
 3|                           DELETE STUDENT FUNCTION                              |3
 4|                                                                                |4
 5|                                                                                |5
 6|        THIS FUNCTION REMOVES THE STUDENT OR STUDENTS FROM THE DATABASE         |6
 7|        SYSTEM AND STORES ALL OF THEIR INFORMATION IN FILE CALLED ARCHIVE.DAT.  |7
 8|        YOU CAN EITHER REMOVE A SINGLE STUDENT OR ARCHIVE AN ENTIRE SECTION.    |8
 9|                                                                                |9
10|                1       DELETE A STUDENT BY NAME.                               |10
11|                                                                                |11
12|                2       DELETE AN ENTIRE SECTION.                               |12
13|                                                                                |13
14|                9       EXIT TO PREVIOUS MENU.                                  |14
15|                                                                                |15
16|                SELECT OPTION(1,2,9)==>                                         |16
17|                                                                                |17
18|                                                                                |18
19|                                                                                |19
20|                                                                                |20
21|                                                                                |21
22|                                                                                |22
23|                                                                                |23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: DELSECT

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
 1|                                                                                |1
 2|                                                                                |2
 3|                                                                                |3
 4|                      DELETE AND ARCHIVE AN ENTIRE SECTION                      |4
 5|                                                                                |5
 6|                                                                                |6
 7|                                    +----------+                                |7
 8|        ENTER THE SECTION CODE ==>  |          |                                |8
 9|                                    +----------+                                |9
10|                                                                                |10
11|                                                +-+                             |11
12|        ARE YOU SURE YOU WISH TO DO THIS, (Y,N) | |                             |12
13|                                                +-+                             |13
14|                                                                                |14
15|                                                                                |15
16|                                                                                |16
17|                                                                                |17
18|                                                                                |18
19|                                                                                |19
20|                                                                                |20
21|                                                                                |21
22|                                                                                |22
23|                                                                                |23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: WORKDEL

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
  1|                                                                                |1
  2|                                                                                |2
  3|                                                                                |3
  4|                                                                                |4
  5|                       CURRENTLY ARCHIVING AND DELETING :                       |5
  6|                                                                                |6
  7|                                                                                |7
  8|                       +-----------------------------+                          |8
  9|                       |                             |                          |9
 10|                       +-----------------------------+                          |10
 11|                                                                                |11
 12|                                                                                |12
 13|                               (PLEASE STAND BY:)                               |13
 14|                                                                                |14
 15|                                                                                |15
 16|                                                                                |16
 17|                                                                                |17
 18|                                                                                |18
 19|                                                                                |19
 20|                                                                                |20
 21|                                                                                |21
 22|                                                                                |22
 23|                                                                                |23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: ALLORSOME

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
  1|                                                                                |1
  2|                                                                                |2
  3|                                                                                |3
  4|                     SELECT AN ENTIRE SECTION TO BE PRINTED                     |4
  5|                                                                                |5
  6|                                                                                |6
  7|                      OR SELECTED STUDENT WITHIN A SECTION                      |7
  8|                                                                                |8
  9|                                                                                |9
 10|                                                                                |10
 11|               1        PRINT ALL STUDENTS EDPLANS WITHIN A SECTION.            |11
 12|                                                                                |12
 13|               2        PRINT SELECTED STUDENTS EDPLAN WITHIN AN SECTION.       |13
 14|                        ( AN ADDITIONAL SCREEN WILL APPEAR TO ALLOW )           |14
 15|                        ( YOU TO PICK THE STUDENTS TO BE PRINTED.   )           |15
 16|                                                                                |16
 17|               9        EXIT TO PREVIOUS FUNCTION.                              |17
 18|                                                                                |18
 19|               SELECT OPTION (1,2,9)==>                                         |19
 20|                                                                                |20
 21|                                                                                |21
 22|                                                                                |22
 23|                                                                                |23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: WORKONSECT

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
 1|                                                                                |1
 2|                                                                                |2
 3|                          CURRENTLY PRINTING EDPLAN                             |3
 4|                                                                                |4
 5|                                                                                |5
 6|                                                                                |6
 7|                     +------------------------------+                           |7
 8|                                                                                |8
 9|  FOR:  |                                                            |          |9
10|                                                                                |10
11|                     +------------------------------+                           |11
12|                                                                                |12
13|                                                                                |13
14|                                                                                |14
15|                                                                                |15
16|                                                                                |16
17|                                                                                |17
18|                                                                                |18
19|                                                                                |19
20|                                                                                |20
21|                                                                                |21
22|                                                                                |22
23|                                                                                |23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: SEQHELP

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
 1|                                                                                |1
 2|                                                                                |2
 3|        ENTER TO THE LEFT OF THE COURSE, THE TYPE OF SEQUENCE THE COURSE        |3
 4|        BELONGS TO.  FOR INSTANCE, EENG799 WILL ALMOST ALWAYS BE THE            |4
 5|        THESIS COURSE AND SHOULD CONTAIN AN ENTRY 'THES'.  OTHER VALID          |5
 6|        SEQUENCE CODES UNDG,SEQA,SEQB,MATH,WAIV.  AS A SHORT CUT, THE FOLLOWING |6
 7|        ONE LETTER CODES CAN BE ENTERED IN PLACE OF TYPING THE ENTIRE FOUR      |7
 8|        LETTER CODE:                                                            |8
 9|                                                                                |9
10|                                    A = SEQA                                    |10
11|                                    B = SEQB                                    |11
12|                                    M = MATH                                    |12
13|                                    W = WAIV                                    |13
14|                                    U = UNDG                                    |14
15|                        OR A SPACE WILL DELETE THE FIELD                        |15
16|                                                                                |16
17|        ENTER THE CODE AND HIT TAB OR RETURN AND THE ACTUAL CODE WILL APPEAR.   |17
18|        TAB, BACKSPACE, AND RETURN KEYS ALLOW YOU TO TRAVERSE THE LIST.  THE    |18
19|        FUNCTION WILL NOT TERMINATE UNTIL THE CURSOR PASSES THE LAST FIELD AND  |19
20|        THE RETURN KEY IS HIT.                                                  |20
21|                                                                                |21
22|                                                                                |22
23|                                                                                |23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: SEQABW02

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
  1|                             SEQUENCE DECLARATION                               |1
  2|               THE FOLLOWING COURSES REPRESENT THE SEQUENCES AS                 |2
  3|       THEY CURRENTLY EXIST IN THE COURSE FILES. SEQA REPRESENTS THE           |3
  4|       DEGREE REQUIRED COURSES, SEQB REPRESENTS THE SEQUENCE COURSES,          |4
  5|       MATH, THES, AND WAIV REPRESENT MATH, THESIS AND WAIVED COURSE.          |5
  6|       UNDG REPRESENT THE UNDERGRADUATE COURSES.                                |6
  7|                                                                                |7
  8|                                                                                |8
  9|                   |                  |                  |                |     |9
 10|                   |                  |                  |                |     |10
 11|                   |                  |                  |                |     |11
 12|                   |                  |                  |                |     |12
 13|                   |                  |                  |                |     |13
 14|                   |                  |                  |                |     |14
 15|                   |                  |                  |                |     |15
 16|                   |                  |                  |                |     |16
 17|                   |                  |                  |                |     |17
 18|                   |                  |                  |                |     |18
 19|                   |                  |                  |                |     |19
 20|                   |                  |                  |                |     |20
 21|                   |                  |                  |                |     |21
 22|                   |                  |                  |                |     |22
 23|                   |                  |                  |                |     |23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: NAMESECT2

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
  1|                                                                                |1
  2|                                                                                |2
  3|                          LIST OF STUDENT BY SECTION                            |3
  4|                                                                                |4
  5|                        SELECT STUDENTS TO BE PRINTED                           |5
  6|                                                                                |6
  7|                              SECTION:                                          |7
  8|                                                                                |8
  9|                   +----------------------------------------+                   |9
 10|                   |                                        |                   |10
 11|                   |                                        |                   |11
 12|                   |                                        |                   |12
 13|                   |                                        |                   |13
 14|                   |                                        |                   |14
 15|                   |                                        |                   |15
 16|                   |                                        |                   |16
 17|                   |                                        |                   |17
 18|                   +------+---------------------------------+------------------+|18
 19|                          |  USE ARROW KEYS TO SCROLL THE SCREEN.  HIT RETURN  ||19
 20|                          |  TO EXIT THE SCREEN.  PUT AN 'X' BESIDE THE NAMES  ||20
 21|                          |  OF THOSE STUDENTS WHOSE EDPLAN YOU WISH TO PRINT. ||21
 22|                          |  ENTER AN 'E' TO CANCEL THIS FUNCTION.             ||22
 23|                          +----------------------------------------------------+|23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: LISTSECT

```
          1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
--------------------------------------------------------------------------------
 1|                                                                              |1
 2|                                                                              |2
 3|                         LIST OF STUDENTS BY SECTION                          |3
 4|                                                                              |4
 5|                                                                              |5
 6|                                                                              |6
 7|                                    +----------+                              |7
 8|                    ENTER SECTION = |          |                              |8
 9|                                    +----------+                              |9
10|                                                                              |10
11|                                                                              |11
12|               ENTER THE SECTION CODE OF THE CLASS YOU WISH TO WORK WITH.  IF |12
13|        THE SECTION YOU ENTERED IS NOT A VALID SECTION THE PROGRAM WILL RETURN|13
14|        TO THE PREVIOUS MENU.  THE LIST OF STUDENTS ASSOCIATED WITH THE SECTION|14
15|        WILL APPEAR ON THE NEXT SCREEN IN  A SCROLLED AREA.                   |15
16|                                                                              |16
17|                                                                              |17
18|                                                                              |18
19|                                                                              |19
20|                                                                              |20
21|                                                                              |21
22|                                                                              |22
23|                                                                              |23
--------------------------------------------------------------------------------
12345678901234567890123456789012345678901234567890123456789012345678901234567890
         1         2         3         4         5         6         7         8
```

Form: SEQPAGE

```
          1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
--------------------------------------------------------------------------------
 1|                          SEQUENCE DECLARATION DISPLAY                        |1
 2| SEQUENCE A: DEGREE REQUIRED COURSES:                                         |2
 3|                                                                              |3
 4|------------------------------------------------------------------------------|4
 5| SEQUENCE B: REQUIRED SEQUENCE COURSES:                                       |5
 6|                                                                              |6
 7|------------------------------------------------------------------------------|7
 8| MATH COURSES:                                                                |8
 9|                                                                              |9
10|------------------------------------------------------------------------------|10
11| THESIS COURSE:                                                               |11
12|                                                                              |12
13|------------------------------------------------------------------------------|13
14| OTHER GRADUATE COURSES:                                                      |14
15|                                                                              |15
16|                                                                              |16
17|------------------------------------------------------------------------------|17
18| UNDERGRADUATE COURSES:                                                       |18
19|                                                                              |19
20|------------------------------------------------------------------------------|20
21| WAIVED COURSES:                                                              |21
22|                                                                              |22
23|                                                                              |23
--------------------------------------------------------------------------------
12345678901234567890123456789012345678901234567890123456789012345678901234567890
         1         2         3         4         5         6         7         8
```

Form: LOADING

```
             1         2         3         4         5         6         7         8
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
    --------------------------------------------------------------------------------
  1|                         WELCOME TO THE DATABASE SYSTEM                         |1
  2|                                                                                |2
  3|                      AFIT/ENG DATA BASE MANAGEMENT SYSTEM                      |3
  4|                                                                                |4
  5|                                                                                |5
  6|                   +-------------------------------------+                      |6
  7|                                                                                |7
  8||       LOADING THE                           FILE.PLEASE STAND BY          |   |8
  9|                                                                                |9
 10|                   +-------------------------------------+                      |10
 11|                                                                                |11
 12|                                      +-----+                                   |12
 13|                    NUMBER OF RECORDS |     |                                   |13
 14|                                      +-----+                                   |14
 15|                                                                                |15
 16|                                                                                |16
 17|                                                                                |17
 18|                                                                                |18
 19|                                                                                |19
 20|                                                                                |20
 21|                                                                                |21
 22|                                                                                |22
 23|                                                                                |23
    --------------------------------------------------------------------------------
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
             1         2         3         4         5         6         7         8
```

Form: GRAPHMENU

```
             1         2         3         4         5         6         7         8
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
    --------------------------------------------------------------------------------
  1|                                                                                |1
  2|                       GRAPHIC DEMONSTRATION PROGRAM                            |2
  3|                                                                                |3
  4|                                                                                |4
  5|                       1.   GRAPH NUMBER OF STUDENTS IN A SECTION               |5
  6|                                                                                |6
  7|                       2.   GRAPH NUMBER OF STUDENTS IN A COURSE                |7
  8|                                                                                |8
  9|                       3.   SET MODE TO PLOTTER                                 |9
 10|                                                                                |10
 11|                       4.   SET MODE TO SCREEN (DEFAULT)                        |11
 12|                                                                                |12
 13|                       5.   AFIT/ENG FACULTY WORKLOAD (USING FMS)               |13
 14|                                                                                |14
 15|                       6.   AFIT/ENG DEPARMENT WORKLOAD (USING FMS)             |15
 16|                                                                                |16
 17|                       9.   EXIT PROGRAM                                        |17
 18|                                                                                |18
 19|                       SELECT OPTION (1-4,9)==>                                 |19
 20|                                                                                |20
 21|                                                                                |21
 22|                                                                                |22
 23|                                                                                |23
    --------------------------------------------------------------------------------
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
             1         2         3         4         5         6         7         8
```

Form: NOSCREEN

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
 1|                                                                                |1
 2|                                                                                |2
 3|                                                                                |3
 4|                                                                                |4
 5|                                                                                |5
 6|                                                                                |6
 7|                                                                                |7
 8|                                                                                |8
 9|                                                                                |9
10|                                                                                |10
11|                                                                                |11
12|                                                                                |12
13|                                                                                |13
14|                                                                                |14
15|                                                                                |15
16|                                                                                |16
17|                                                                                |17
18|                                                                                |18
19|                                                                                |19
20|                                                                                |20
21|                                                                                |21
22|                                                                                |22
23|                                                                                |23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

Form: GRAPHSEL

```
            1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
   --------------------------------------------------------------------------------
 1|                                                                                |1
 2|                                                                                |2
 3|                       SELECT SECTIONS TO BE GRAPHED                            |3
 4|                                                                                |4
 5|                                                                                |5
 6|                                                                                |6
 7|                          +------------------------+                            |7
 8|                          |       ===>             |                            |8
 9|                          |       ===>             |                            |9
10|                          |       ===>             |                            |10
11|                          |       ===>             |                            |11
12|                          |       ===>             |                            |12
13|                          |       ===>             |                            |13
14|                          |       ===>             |                            |14
15|                          |       ===>             |                            |15
16|                          |       ===>             |                            |16
17|             +------------+------------------------+----------+                 |17
18|             | PLACE AN 'X' BESIDE THE SECTIONS YOU WISH      |                 |18
19|             | TO GENERATE A SUMMARY FILE FOR.  YOU MAY       |                 |19
20|             | UN-SELECT A SECTION BY PLACING A SPACE IN THE  |                 |20
21|             | MARKER TO THE LEFT. TO EXIT, HIT RETURN, TO    |                 |21
22|             | EXIT AND CANCEL , ENTER AND 'E'.               |                 |22
23|             +------------------------------------------------+                 |23
   --------------------------------------------------------------------------------
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
            1         2         3         4         5         6         7         8
```

APPENDIX F

Data-set Requirement Tabulations

This part of the appendix was obtained from Maj Pangmans Thesis on the AFIT/EN Database System (2). The left side of this appendix lists the specific requirements requested that the expanded AFIT Database accommodate, while the right side show the subsequent location where ethe requirement has been included within it. The data requirements obtained by 'interviews' also consisted of the repeat conversation with administrative personnel as will as those originally interviewed. This document was used to check the requirements requested against the atomic values needed to support those re-quirements. Those requirements items which have more than one data-set listed as the location where the requirement is found indicate that either area could contain such data depending on the application progoram or that the application program required to obtain such data can do so utilizing the listing data-set links.

REQUIREMENT DATA	TABLE DATA - WHERE THE REQUIREMENT DATA ARE LISTED IN THE DATA-SET
1. Change the thesis course credi from 4 to any number of hours.	MCRS - Course control number has six digits. (ex. ee79A - 1 credit hour)
2. Fac Advisor --> Students Fac Advisor --> Thes Students Student --> Fac Advisor Student --> Thes Advisor	FADV - Linked to FACT,STDT, SECT TADV - Linked to FACT,STDT, THES
3. Thesis Data --> Student Advisor	STDT TADV

	Box Number	STDT
	Class	SECT
	Thesis Topic (Title?)	THTL
	Committee Members	TCMF
	Thesis Title	THTL
4. Instructors with Course Listing -->		
	Instructors	FACT
	Course Linsting	MCRS
5. Course Listing -->	Title	MCRS
	Course Number	MCRS
	Number Credit Hour	MCRS
	Quarter	MQTR
	Course Text	MBKT
	Instructor	VINS
6. Projected Enrollments -->	Course	MCRS
	Number of Hours	MCRS
	Quarter	MQTR
	Number of Books to order	VNMO
7. Ed Plan -->	Student Name	STDT
	SSAN	STDT
	Course(Num. & Title)	MCRS
	Section (Size Limit)	MCRS
	Credit Hours	MCRS
	Class	SECT
	Quarter	MQRT
	Grades	CRSE
	Prerequisites	MREQ
	Waiver	CRSE
	Course Sequence	MSSF
8. Teaching Loads -->	a. Instructors Name	FACT
	b. Course(s) Taught	MCRS/VCQR
	c. Quarter	MQTR
	d. Type of Course	MCRS
	e. Number of Section	MCRS
	f. Number of Thesis/ Instructors	FACT/TADV/THES
	g. Short Term Hours Taught	MCRS/FACT
	h. Profssional Developement Quarter Listing	VPDQ
	i. Faculty Advisor Listing	FADV
	j. Thesis Committee Listing	TCMF
9. Course Data -->	Student Name	STDT
	SSAN	STDT

	Service	STDT
	Rank	STDT
	Class	STDT/SECL/SECT
	Home Telephone	STDT
	Box Number	STDT
10. Instructor Statistics -->	Instructor Name	FACT
	SSAN	FACT
	Courses Taught	MCRS
	Students In Course by:	
	Name	STDT
	SSAN	STDT
	Class	SECL
	Grades	CRSE
11. Catalo Faculty Publications-->	Name	FACT
	SSAN	FACT
	Title	FCOM
	Date	FCOM
	Topic (TITLE)	FCOM
12. Room Usage -->	Room Location	BLRM
	Room Capacity	CPTY
	Time Schedule	TIME/DAYS/SCHD
13. Projected Enrollments -->	Course	MCRS
	Quarter	MQTR
	Student SSAN	STDT
	Student Name	STDT
	Student Class	SECL
14. Proffessional Duties -->	Awards	FHAW
	Committee Membership	FCMT
	Committee Duties	FCMT
	Professional Soc.	FSOC
	Interest Area	FINT
15. Student Data -->	Name	STDT
	SSAN	STDT
	Box Number	STDT
	Home Phone Number	STDT
	Advisor	TADV
	Class	SECT
	Service	STDT
	Graduated AFIT?	STDT
16. Graduate Record Data -->	Verify Degree Requirements	MDEG
	Course Sequences	MSSF
	Course	MCRS
	Math Credits	MCRS

APPLICATIONS SOFTWARE REQUIREMENTS LIST

This portion of the requirements list specifies the requirements for the application software and computer interface requirements. The requirements are numbered for easier reference in the thesis.

1. Database Schema Enhancements and Modifications

1.1 Graduate Credit Record Modification.

1.1.1 Remove the student social security number link from the Variable Sequence File and from the Master Student File.

1.1.2 Remove references to the student social security number link from all other files.

1.2 Text Book file modification.

1.2.1 Text book file and book file contain duplicate information. Delete the text book file from the database.

1.2.2 Remove all references to the text book file from other files.

1.3 Course Field Modification.

1.3.1 Increase the field size from 6 characters to 8 characters to make the field compatible to other database systems.

1.4 Thesis Catalog File (NTIC).

1.4.1 Information contained in the variable files belongs in the Master Thesis File (THES). The variable files must contain links only to allow for deletion of students and faculty.

1.4.2 Thesis Catalog file and Thesis file contain duplicate information. The Thesis Catalog file should be removed from the schema and all links removed from the system.

1.4.3 Re-generate the database using the new file system.

2. Functional Requirements.

2.1 Functional file grouping. There are five distinct file groups. Each one is to be maintained by a seperately compiled program to aid in modularity and design.

2.1.1 Faculty: Maintains faculty master file, department master file and related variable files.

2.1.2 Student: Maintains the student master file, education plans and related variable files.

2.1.3 Courses: Maintains the master course file, master textbook file, day file, time file, room capacity file, and quarter file as well as the related variable files.

2.1.4 Thesis: Maintains the thesis master file and controls the links to students, faculty,and departments.

2.1.5 Sequence and Degree Modules: Maintains the master sequence file, master degree requirement file and associated variable files.

2.2 Standard Database Functions.

2.2.1 Add: Adds a record to the database. This involves adding a master record and associated variable records if necessary.

2.2.2 Update: Updates a master record and in the database. Must be able to detect if the record exists and update associated variable records.

2.2.3 Delete: Delete a master record. In some cases such as for the faculty and students records, the data should be archived for historical data and records. Thesis information should be kept for student database searches.

2.2.4 Review: Scan the information in a master record and link variable records in read mode only. Maintain the ability to print the information.

2.2.5 WRMXXXX: Write an updated master file record to the AFIT Database where "XXXX" is the four character file code.

2.2.6 RDMXXXX: Read a record from the master file of the AFIT Database where "XXXX" is the four character file code.

2.2.7 DLMXXXX: Delete a master record from the AFIT Database and all of the associated variable records. "XXXX" designates the four character file code.

2.2.8 ADMXXXX: Add a new record to the AFIT Database system where "XXXX" is the four character file code.

2.2.9 ADCXXX: add a variable record continue, adds a record after the current record pointed to.

2.2.10 ADAXXXX: Add a variable record after the second record pointed to.

2.2.11 ADBXXXX: Add a record before the one pointed to in the record string.

2.2.12 RDVXXXX: Read the next variable record in the string of records.

2.2.13 RDRXXXX: Read the previous variable record in the string.

2.2.14 RDDXXXX: Read direct. Reads the record directly pointed to by the reference pointer.

2.2.15 WRVXXXX: Writes a variable record back to the database system that previously existed.

2.2.16 DLDXXXX: Deletes a record pointed to by the reference pointer.

2.3 Database Limitations. These limitations apply to the database generation because TOTAL allocates the disk space for the entire file system at once. Disk space must be contiguous.

2.3.1 Faculty file limitations

Room should be allocated for 200 people in the database with 460 bytes for each individual. Blocksize of the record should be 5 records or 2560 bytes.

2.3.2 Student file limitations:

Space should be allocated for 585 students in the database with 460 bytes per student. The blocksize of the file should be the same as the faculty (2560 byte blocks).

2.3.3 Department file limitations:

There are currently five departments, however, space should be allocated for seven in case of growth. Each records should contain 40 bytes and the blocksize should be set to 512 bytes.

2.3.4 Thesis file limititations:

At this time, space should be allocated for 585 thesis which is identical to the number of students enrolled. Each record contains 232 bytes, blocksize should be set at 922 bytes.

2.3.5 Class section limitations:

With the ability to overlap the number of section a degree is offered to, space for 30 sections should be allowed. Each section requires 56 bytes and a blocksize of 144 bytes.

2.3.6 Course file limitations:

The number of course required to be store is 542. This allows for a ten percent growth in the next year. Each record requires 128 bytes with a block size of 1024.

2.3.7 Class quarter limitations:

The number of quarters is expected to remain the same. Space should be allocated for 3 years of classes, or 12 quarters. Each record requires 40 bytes with a block size of 480 bytes per block.

2.3.8 Course Book limitations: Each course will require on the average of 2 books. With a set number of classes at 542, storage must be made available for 1084 books. Each record requires 130 with a blocksize of 390 bytes.

2.3.9 Class time limitations:
With each class time starting on the hour between the hours of 0800 and 1700, there are 10 class times. To accomodate special requests, an additional 5 classes will be added. Room should be made available for 15 classes, each record requies 20 bytes with a blocksize of 500.

2.3.10 Building and Room limitations:
The only capacity for the rooms ever put on the current database is 30, which is the enrollement limitation. However, room should be allowed for 10 records for the room capacity and 100 rooms total for the Master Building and Room file. The Master Room capacity file requires 20 bytes per record and 400 byte blocksize and the Master Building and Room file needs 32 bytes per record and a 512 bytes per block.

2.3.11 Degree and Sequence limitations:
Because of the changing requirements, these two files should be allowed their total capacity of 100 and 1000 records respectfully. The Master Sequence File should allow for 1000 records with each record requiring 60 bytes and 512 byte blocksize. The Master Degree Requirement file needs 66 bytes per record and 512 byte blocksize.

2.4 Response Time Requirments. The objective is to keep response time to a minimum. Because of the relative low number of records that are in each file, (less then 400 on the average) this can be done by keeping an internal list of files in a simple links list structure. There are currently a need for the following linked list structures.

2.4.1 Student file: Keep track of the students name, social security number, class section, and box number. Define add, update, delete and find operations on this list.

2.4.2 Faculty file: Keep track of the faculty names, social security number, and advisor section. Define the add, update, delete, and find operations on this list.

2.4.3 Class Section: Keep a sorted list of class sections, advisors, and section leaders in a linked list. Define the Add and find operations on the structure.

2.5 Required Reports.

2.5.1 Division Faculty Schedule and Manpower Requirements Expenditure Document: See this Appendix for examples of these documents.

2.5.2 Education Plans: Contains the students information, class schedule by quarter, sequence A and B.

2.5.3 Listing of Enrolled Students: Give the students enrolled in courses at AFIT by department and section numbers.

2.5.4 List Courses and information on when they are offered, credit hours, and instructors.

2.5.5 Student Locator listing: Contains the information currently contained on the student locator cards in the ENA office.

2.6 Data Syntax: See Appendix C for a complete list of data syntax and compatibility tests to be performed on the data items to protect the data integrety.

3. Computer Interface Requirements.

3.1 Identify targeted user group and develop system on the critera of the average user as defined by this thesis.

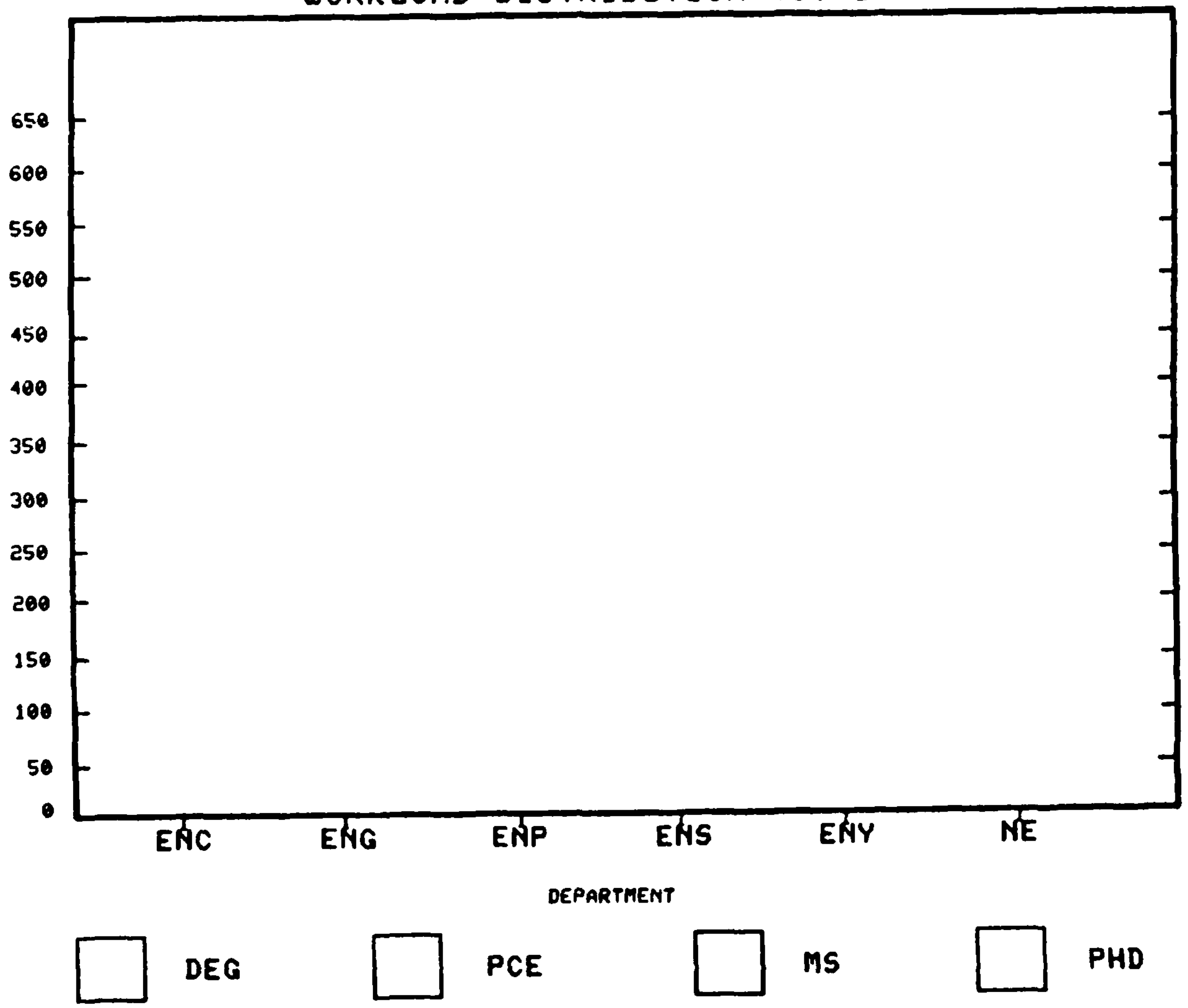
3.2 Develop a prototype of the Education Plan program to demonstrate to possible users for feedback on "user-friendliness" of the system.

3.3 Develop menus for the system that are standard in structure and self documenting. No more than seven selections should be on any one menu.

3.4 Item Selection: Whenever possible, let the user select from a group of items such as course to eliminate the need to type in data. This should enhance data integrety and make the system simpler to use. Make the selection as narrow as possible. For instance, if selecting from a group of classes, allow the user to ask for only EENG classes or MATH classes.

AFIT/EN FACULTY

WORKLOAD DISTRIBUTION (CY 84)

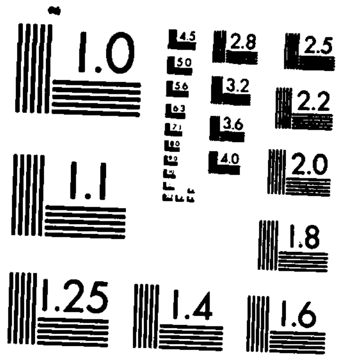


```
                    SCHOOL OF ENGINEERING STUDENT RECORD
GAITROS, DAVID A.                   CPT  4924    GE-85D

PLAN    INITIAL __________ REVISED __________ FINAL __________
PROGRAM APPROVAL: M. S.
  ADVISOR__________  ACADEMIC__________  THESIS__________

*QTR-YR*      NUMBER              COURSE TITLE                 HRS GRADE PTS
SUMMER  1984________________________________________
          MATH531           MATH METHODS OF COMPUTE          4
          MATH592           MANAGERIAL STATISTICS I          3
          COMM600           TECHNICAL WRITING                2
          MATH445           INTRO TO ALGORITHM DESI          4
                          CUM HRS: 13  CUM GPR   GTR HRS 13  QTR GPR:

FALL    1984________________________________________
          EENG450           INTRO TO LOGIC DESIGN            5
          MATH692           MANAGERIAL STATISTICS I          3
          EENG586           INFO STRUCTURES                  4
          EENG589M        OPERATING SYSTEMS                  2
                          CUM HRS: 27  CUM GPR   GTR HRS 14  QTR GPR:

WINTER  1985________________________________________
          EENG588           COMPUTER SYSTEMS ARCHIT          4
          EENG593           SOFTWARE ENGINEERING             4
          EENG646           COMPUTER DATA BASE SYS           4
          EENG698           THESIS SEMINAR                   0
          COMM698           SEMINAR IN TECH COMMUNI          2
                          CUM HRS: 41  CUM GPR   GTR HRS 14  QTR GPR:

SPRING  1985________________________________________
          EENG690           SOFTWARE SYS PROGRAMMIN          2
          MATH555           INTRO TO ADA                     4
          EENG799           INDEPENDENT STUDY                4
          OPER548           MANG ANAL & SIM I                4
                          CUM HRS: 55  CUM GPR   GTR HRS 14  QTR GPR:

SUMMER SHORT________________________________________
          EENG545           SOFTWARE SYS ACQUISITIO          2
                          CUM HRS: 57  CUM GPR   GTR HRS  2  QTR GPR:

SUMMER  1985________________________________________
          EENG799           INDEPENDENT STUDY                4
          OPER648           MANG ANAL & SIM II               4
                          CUM HRS: 65  CUM GPR   GTR HRS  8  QTR GPR:

FALL    1985________________________________________
          EENG799           INDEPENDENT STUDY                4
          MATH568           INTERACOMMIVE COMPUTER           4
          EENG793           ADVANCED SOFTWARE ENG            4
                          CUM HRS: 77  CUM GPR   GTR HRS 12  QTR GPR:


 AFITDB/ENG  FORM #1 21-NOV-1985
```

```
                                          PAGE 2
                        SCHOOL OF ENGINEERING STUDENT RECORD
                                AFITDB/ENG    21-NOV-1985

GAITROS, DAVID A.              CPT GE-85D                    COURSE HRS
SEQA_____________________________________________
          EENG589M          OPERATING SYSTEMS                    2
          EENG588             COMPUTER SYSTEMS ARCHIT            4
          EENG593             SOFTWARE ENGINEERING               4
SEQB_____________________________________________

          EENG586             INFO STRUCTURES                    4
          OPER548             MANG ANAL & SIM I                  4
          OPER648             MANG ANAL & SIM II                 4
MATH_____________________________________________

          MATH592             MANAGERIAL STATISTICS I            3
          MATH692             MANAGERIAL STATISTICS I            3
THESIS___________________________________________

          EENG799             INDEPENDENT STUDY                  4
OTHER GRAD_______________________________________

          EENG698             THESIS SEMINAR                     0
          COMM698             SEMINAR IN TECH COMMUNI            2
          EENG690             SOFTWARE SYS PROGRAMMIN            2
          MATH555             INTRO TO ADA                       4
          MATH531             MATH METHODS OF COMPUTE            4
          COMM600             TECHNICAL WRITING                  2
          EENG545             SOFTWARE SYS ACQUISITIO            2
          EENG646             COMPUTER DATA BASE SYS             4
          MATH568             INTERACOMMIVE COMPUTER             4
          EENG793             ADVANCED SOFTWARE ENG              4
UNDERGRAD________________________________________

          MATH445             INTRO TO ALGORITHM DESI            4
          EENG450             INTRO TO LOGIC DESIGN              5

 LAST ITEM
```

AD-A164 051 IMPLEMENTATION OF THE AFIT/ENG FACULTY AND STUDENT DATABASE MANAGEMENT SYSTEM(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING 3/3

UNCLASSIFIED D A GAITROS DEC 85 AFIT/GCS/ENG/85D-5 F/G 9/2 NL


1.0
4.5
5.0
5.6
6.3
7.1
8.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

APPENDIX G

ABSTRACT DATA TYPE DEFINITIONS

STUDENT AND FACULTY LINKED LIST ABSTRACT DATA TYPE.

The AFIT/ENG Database system requires that sorting and searching must be accomplished on the faculty and student data names in a highly volotile environment. To facilitate this highly dynamic arena, an internal list of the data must be kept and maintained. The data type is implemented with a singly linked list using pointers and a dynamic memory allocation scheme. Modules should be independent of global variables to make transporting and maintenance easier. The following data type and modules should be defined for all applications of the master student and faculty files:

```
type

          LINK_PTR = ^LINK_RECORD;
          LINK_RECORD = RECORD;
               NAME : BUFF28;
               CTRL : BUFF9;
               RANK : BUFF3;
               DEPT : BUFF4;
               SECT : BUFF8;
               NEXT : LINK_PTR;
               PREV : LINK_PTR;
          END; {* LINK_RECORD *}
```

MODULE NAME	PARAMETERS	FUNCTION
BUILDLINKLIST	LINK_PTR,FILE HEADER)	Reads the database master student or faculty file sequentially and inserts the name and data into the list.
ADDNAME	LINK_PTR,FILE HEADER	Adds a record to the link list in alphabetical order. The HEADER parameter will point to the faculty or student list.
DELNAME	SSAN,HEADER	Deletes a record from the list for

		a specific social security number.
FINDNAME	NAME,SSAN, LINK_PTR	Finds a record using the last name as a key or a portion of the last name. For instance, this routine would find the first occurance of the name that began with the letter "G". The search starts at the next position from LINK_PTR but not inclusive.
ISEMPTY	HEADER	Detemines if the list is empty.
CREATE	HEADER	Creates a header record for a list.

<u>Student and Faculty List Axiom definition</u>

```
     Structure LINKLIST(LINK_PTR,HEADER,FILE)
     DECLARE   CREATE(HEADER) ==> HEADER;
               BUILDLINKLIST(HEADER,FILE) ==> HEADER
               ADDNAME(LINK_PTR,HEADER) ==> HEADER
               DELNAME(LINK_PTR,HEADER) ==> HEADER
               ISEMPTY(HEADER) ==> BOOLEAN
     FOR ALL A in LINKLIST, ptr as LINK_PTR, file as FILE,
               head as HEADER, LET
               ISEMPTY(CREATE(head)) ::= true;
               ISEMPTY(ADDNAME(ptr,CREATE(head))) ::= false
               DELNAME(ptr,ADDNAME(ptr,head)) ::= head
               DELNAME(ptr,CREATE(head)) ::= error
     end
end LINKLIST
```

Appendix H

STANDARD DATABASE ROUTINES

These routines are required to enable the application programmers to interface with the Total Database Management System in an easier fashion. By doing this, we hope to decrease the amount of time involved in developing the required software and in training personnel to maintaining the programs.

MODULE NAME	PARAMETERS	FUNCTION
SIGN	"SINON,SINOF"	Log onto the database system.
SCHEMALOAD	SCHEMA	Defines the parameters, files, and file disposition of the database at the time of "SINON"
CHECKSTATUS	STATUS,OK	Checks the return status of a call to the database system. Calls an error routine in case the status is not "****".
ERRORCODES	STATUS	Displays the message associated with the error code to the user and waits for 8 seconds before clearing the screen.
WAIT	TIME	Performs a wait function which may allow the user to view a message before the next function is displayed.
OUTPUTERROR	STATUS	Maintains the error messages for the TOTAL DBMS system.
WRMXXXX	XXXX_REC	The generic write master file routine used to generate a write master routine for a specific master file.
RDMXXXX	XXXX_REC	The generic read master file routine.
ADMXXXX	XXXX_REC	The generic add master record routine. The control key must be asssigned to XXXX.CTRL before the call.

DLMXXXX	XXXX_REC	The generic delete a master record routine. All viable records must be deleted before this routine is called.

The following Pascal program procedures are the complete set of standard routines developed and used thoughout this effort. The programs reside in a single file and are read into another file when creating an application software package for the AFIT/ENG Database System. The standard data types must also be included in the file if these routines are to be used.

```
{***************************************************************************}
{*      DATE: 30/05/85                                                     *}
{*      NAME: WAIT                                                         *}
{*      DESCRIPTION: THIS MODULE ALLOWS THE PROGRAMMER TO SPECIFY AN       *}
{*              A WAITING PERIOD BEFORE GOING TO THE NEXT INSTRUCTION.     *}
{*              THIS IS USEFUL THAT WHEN DISPLAYING ERROR MESSAGES TO      *}
{*              THE USER AND ALLOWING THE USER TO SEE THE MESSAGE WITH-    *}
{*              OUT HITTING RETURN TO CONTINUE.                            *}
{*      GLOBAL VARIABLES USED: NONE                                        *}
{*      GLOBAL VARIABLES CHANGED: NONE                                     *}
{*      MODULES CALLED: NONE                                               *}
{*      CALLING MODULES: OUTPUTERROR                                       *}
{*      AUTHOR: CAPT DAVID A. GAITROS                                      *}
{*                                                                         *}
{***************************************************************************}

PROCEDURE WAIT ( WAITVAR: INTEGER);
VAR  INDEX,I : INTEGER;
BEGIN
        I := 1;
        FOR INDEX := 1 TO WAITVAR* 100 DO
             I := INDEX
END; {* WAIT *}
```

```
{*************************************************************************}
{*      DATE: 30/05/1985                                                 *}
{*      NAME: OUTPUTERROR                                                *}
{*      DESCRIPTION: THIS MODULE ACCEPTS THE ERROR CODE FROM THE         *}
{*              DATABASE AND TRANSLATES THAT INTO AN ERROR MESSAGE.      *}
{*                                                                       *}
{*      GLOBAL VARIABLES USED: NONE                                      *}
{*      GLOBAL VARIABLES CHANGED: NONE                                   *}
{*      MODULES CALLED: NONE                                             *}
{*      CALLING MODULES: CHECKSTATUS                                     *}
{*      AUTHOR: CAPT DAVID A. GAITROS                                    *}
{*                                                                       *}
{*************************************************************************}
PROCEDURE OUTPUTERROR(STATUS : BUFF4);

BEGIN
        WRITE (' *** ERROR STATUS CODE *** = ');
        WRITE ( STATUS);
        WRITELN;
        IF STATUS = 'BCCR' THEN
           WRITELN(OUTPUT,' BAD CYLINDER CONTROL RECORD ')
        ELSE IF STATUS = 'BCTL' THEN
           WRITELN(OUTPUT,' BLANK CONTROL FIELD ')
        ELSE IF STATUS = 'DBNF' THEN
           WRITELN(OUTPUT,' DATA BASE NOT FOUND ')
        ELSE IF STATUS = 'DBPR' THEN
           WRITELN(OUTPUT,' DATA BASE ACCESSED IN PRIVATE MODE ')
        ELSE IF STATUS = 'DUPM' THEN
           WRITELN(OUTPUT,' DUPLICATE MASTER RECORD ')
        ELSE IF STATUS = 'DUPO' THEN
           WRITELN(OUTPUT,' DUPLICATE OPEN OF A DATA SET ')
        ELSE IF STATUS = 'ENTF' THEN
           WRITELN(OUTPUT,' ELEMENT NOT FOUND ')
        ELSE IF STATUS = 'EXSO' THEN
           WRITELN(OUTPUT,' EXTRA SINON COMMAND ')
        ELSE IF STATUS = 'FAIL' THEN
           WRITELN(OUTPUT,' COMMUNICATION FAILURE (F) ')
        ELSE IF STATUS = 'FATL' THEN
           WRITELN(OUTPUT,' FATAL ERROR ')
        ELSE IF STATUS = 'FNOP' THEN
           WRITELN(OUTPUT,' FILE NOT OPEN ')
        ELSE IF STATUS = 'FNTF' THEN
           WRITELN(OUTPUT,' FILE NOT FOUND ')
        ELSE IF STATUS = 'FTYP' THEN
           WRITELN(OUTPUT,' INVALID FILE TYPE ')
        ELSE IF STATUS = 'FULL' THEN
           WRITELN(OUTPUT,' FILE LOADED TO CAPACITY ')
        ELSE IF STATUS = 'FUNC' THEN
           WRITELN(OUTPUT,' INVALID FUNCTION CODE ')
        ELSE IF STATUS = 'HELD' THEN
           WRITELN(OUTPUT,' RECORD HELD ')
        ELSE IF STATUS = 'ICHN' THEN
           WRITELN(OUTPUT,' INVALID LINKAGE PATH CHAIN ')
        ELSE IF STATUS = 'IMDL' THEN
           WRITELN(OUTPUT,' INVALID MASTER DELETE ')
```

```
ELSE IF STATUS = 'IOER' THEN
   WRITELN(OUTPUT,' I/O ERROR (F) ')
ELSE IF STATUS = 'IPAR' THEN
   WRITELN(OUTPUT,' INVALID NUMBER OF PARAMETERS ')
ELSE IF STATUS = 'IRLC' THEN
   WRITELN(OUTPUT,' INVALID RECORD LOCATION (F) ')
ELSE IF STATUS = 'IVBF' THEN
   WRITELN(OUTPUT,' INVALID BUFFER SIZE ')
ELSE IF STATUS = 'IVRC' THEN
   WRITELN(OUTPUT,' INVALID RECORD CODE ')
ELSE IF STATUS = 'IVRD' THEN
   WRITELN(OUTPUT,' INVALID VARIABLE ENTRY DATA SET READ ')
ELSE IF STATUS = 'IVRP' THEN
   WRITELN(OUTPUT,' INVALID REFERENCE PARAMETER ')
ELSE IF STATUS = 'IVTF' THEN
   WRITELN(OUTPUT,' INVALID TOTAL FILE ')
ELSE IF STATUS = 'IVWD' THEN
   WRITELN(OUTPUT,' INVALID WRITE DIRECT ')
ELSE IF STATUS = 'LDnn' THEN
   WRITELN(OUTPUT,' LOADER ERROR ')
ELSE IF STATUS = 'LOAD' THEN
   WRITELN(OUTPUT,' VAR ENTRY FILE LOADED BEYOND CYL. LOAD LIMIT ')
ELSE IF STATUS = 'LOCK' THEN
   WRITELN(OUTPUT,' DATA SET LOCKED (F) ')
ELSE IF STATUS = 'LGER' THEN
   WRITELN(OUTPUT,' LOGGING I/O ERROR (F) ')
ELSE IF STATUS = 'LGNA' THEN
   WRITELN(OUTPUT,' LOG FILE NOT ACTIVE ')
ELSE IF STATUS = 'LSZE' THEN
   WRITELN(OUTPUT,' LOG SIZE ERROR ')
ELSE IF STATUS = 'MFNF' THEN
   WRITELN(OUTPUT,' MASTER FILE NOT FOUND ')
ELSE IF STATUS = 'MLNF' THEN
   WRITELN(OUTPUT,' MASTER LINK NOT FOUND ')
ELSE IF STATUS = 'MRNF' THEN
   WRITELN(OUTPUT,' MASTER RECORD NOT FOUND ')
ELSE IF STATUS = 'NHLD' THEN
   WRITELN(OUTPUT,' RECORD NOT HELD ')
ELSE IF STATUS = 'NLAT' THEN
   WRITELN(OUTPUT,' NO LOGGING ATTACH ')
ELSE IF STATUS = 'NOIO' THEN
   WRITELN(OUTPUT,' NO ASSIGNED I/O AREA (F) ')
ELSE IF STATUS = 'NOTO' THEN
   WRITELN(OUTPUT,' TOTAL NOT AVAILABLE ')
ELSE IF STATUS = 'NRCV' THEN
   WRITELN(OUTPUT,' NO RECOVERY MODE ')
ELSE IF STATUS = 'NSMR' THEN
   WRITELN(OUTPUT,' NO SECONDARY MASTER RECORD FOUND ')
ELSE IF STATUS = 'PNUL' THEN
   WRITELN(OUTPUT,' POSSIBLE NULL RECORD ')
ELSE IF STATUS = 'POOL' THEN
   WRITELN(OUTPUT,' INSUFFICIENT POOL AREA ')
ELSE IF STATUS = 'QFUL' THEN
   WRITELN(OUTPUT,' RESERVATION QUEUE IS FULL ')
ELSE IF STATUS = 'RSVD' THEN
```

```
               WRITELN(OUTPUT,' RESERVED DATA SET ')
            ELSE IF STATUS = 'SEND' THEN
               WRITELN(OUTPUT,' AN ERROR OCCURRED ON SENDING THE DATA ')
            ELSE IF STATUS = 'TFUL' THEN
               WRITELN(OUTPUT,' TASK TABLE IS FULL ')
            ELSE IF STATUS = 'UACM' THEN
               WRITELN(OUTPUT,' UNDEFINED ACCESS MODE (F) ')
            ELSE IF STATUS = 'UCTL' THEN
               WRITELN(OUTPUT,' UNEQUAL CONTROL FIELD ')
            ELSE IF STATUS = 'ULGO' THEN
               WRITELN(OUTPUT,' UNDEFINED LOGGING OPTIONS (F) ')
            ELSE IF STATUS = 'UPDE' THEN
               WRITELN(OUTPUT,' UPDATE MODE ERROR ')
            ELSE IF STATUS = 'VMRE' THEN
               WRITELN(OUTPUT,' VARIABLE READ MASTER ERROR (F) ')
            ELSE WRITELN(OUTPUT,' UNDEFINED ERROR CODE ');
   WRITELN(OUTPUT);
   WAIT(500);
   END; (*OUTPUTERROR*)
```

```
{***************************************************************************}
{*      DATE: 30/05/1985                                                   *}
{*      NAME: CHECKSTATUS                                                  *}
{*      DESCRIPTION: MODULE CHECKSTATUS DETERMINES IF A CALL TO THE        *}
{*              TOTAL DATABASE WAS IN ERROR.                               *}
{*                                                                         *}
{*      GLOBAL VARIABLES USED: NONE                                        *}
{*      GLOBAL VARIABLES CHANGED: NONE                                     *}
{*      MODULES CALLED: OUTPUTERROR                                        *}
{*      CALLING MODULES:                                                   *}
{*      AUTHOR: CAPT DAVID A. GAITROS                                      *}
{*                                                                         *}
{***************************************************************************}

PROCEDURE CHECKSTATUS(VAR OK:BOOLEAN);


BEGIN
   IF STATUS = '****' THEN
        OK := TRUE
   ELSE BEGIN
        OK := FALSE;
        OUTPUTERROR (STATUS);
        WRITE (', HIT RETURN TO CONTINUE ');
        READLN
   END
END {* CHECKSTATUS *};
```

```
{**************************************************************************}
{*      DATE: 23/05/85                                                   *}
{*      NAME: SIGNONOROFF                                                *}
{*      DESCRIPTION: THIS MODULE LOADS THE DATABASE SCHEMA DEPENDING     *}
{*              UPON THE USERS REQUEST AND SIGNS ON OR OFF OF THE        *}
{*              DATABASE.                                                *}
{*                                                                       *}
{*      GLOBAL VARIABLES USED: NONE                                      *}
{*      GLOBAL VARIABLES CHANGED: NONE                                   *}
{*      MODULES CALLED: DATBAS,CHECKSTATUS                               *}
{*      CALLING MODULES: MAIN                                            *}
{*      AUTHOR: CAPT DAVID A. GAITROS                                    *}
{*                                                                       *}
{**************************************************************************}

PROCEDURE SIGNONOROFF( ONOROFF : BUFF5; DATABASE: BUFF4);
CONST
   FACT1 = 'SECTIONSAFITDBUPDATENLFACTPRIVXXXXDEPTPRIVXXXX';    {* 46 *}
   FACT2 = 'STDTSHREXXXXTHESSHREXXXXSECTSHREXXXXMCRSSHREXXXX';  {* 48 *}
   FACT3 = 'MQTRSHREXXXXMBKTSHREXXXXMORDSHREXXXXTIMESHREXXXX';  {* 48 *}
   FACT4 = 'BLRMSHREXXXXCPTYSHREXXXXDAYSSHREXXXXMSSFSHREXXXX';  {* 48 *}
   FACT5 = 'MDEGSHREXXXXVEDUPRIVXXXXFSOCPRIVXXXXFCMTPRIVXXXX';  {* 48 *}
   FACT6 = 'VHAWPRIVXXXXFINTPRIVXXXXFCOMPRIVXXXXFTDYPRIVXXXX';  {* 48 *}
   FACT7 = 'CRSEPRIVXXXXTHTLPRIVXXXXSECLPRIVXXXXVCQRPRIVXXXX';  {* 48 *}
   FACT8 = 'VREQPRIVXXXXVCBKPRIVXXXXVNMOPRIVXXXXSCHDPRIVXXXX';  {* 48 *}
   FACT9 = 'CLSRPRIVXXXXTCMFPRIVXXXXFADVPRIVXXXXVINSPRIVXXXX';  {* 48 *}
   FACT10= 'TADVPRIVXXXXVPDQPRIVXXXXVMSSPRIVXXXXEND.         ';  {* 48 *}

   STDT1 = 'SECTIONSAFITDBUPDATENLFACTSHREXXXXDEPTSHREXXXX';    {* 46 *}
   STDT2 = 'STDTPRIVXXXXTHESSHREXXXXSECTPRIVXXXXMCRSSHREXXXX';  {* 48 *}
   STDT3 = 'MQTRSHREXXXXMBKTSHREXXXXMORDSHREXXXXTIMESHREXXXX';  {* 48 *}
   STDT4 = 'BLRMSHREXXXXCPTYSHREXXXXDAYSSHREXXXXMSSFSHREXXXX';  {* 48 *}
   STDT5 = 'MDEGSHREXXXXVEDUPRIVXXXXFSOCPRIVXXXXFCMTPRIVXXXX';  {* 48 *}
   STDT6 = 'VHAWPRIVXXXXFINTPRIVXXXXFCOMPRIVXXXXFTDYPRIVXXXX';  {* 48 *}
   STDT7 = 'CRSEPRIVXXXXTHTLPRIVXXXXSECLPRIVXXXXVCQRPRIVXXXX';  {* 48 *}
   STDT8 = 'VREQPRIVXXXXVCBKPRIVXXXXVNMOPRIVXXXXSCHDPRIVXXXX';  {* 48 *}
   STDT9 = 'CLSRPRIVXXXXTCMFPRIVXXXXFADVPRIVXXXXVINSPRIVXXXX';  {* 48 *}
   STDT10= 'TADVPRIVXXXXVPDQPRIVXXXXVMSSPRIVXXXXEND.         ';  {* 48 *}


   MCRS1 = 'SECTIONSAFITDBUPDATENLFACTSHREXXXXDEPTSHREXXXX';    {* 46 *}
   MCRS2 = 'STDTSHREXXXXTHESSHREXXXXSECTSHREXXXXMCRSPRIVXXXX';  {* 48 *}
   MCRS3 = 'MQTRPRIVXXXXMBKTPRIVXXXXMORDPRIVXXXXTIMEPRIVXXXX';  {* 48 *}
   MCRS4 = 'BLRMPRIVXXXXCPTYPRIVXXXXDAYSPRIVXXXXMSSFSHREXXXX';  {* 48 *}
   MCRS5 = 'MDEGSHREXXXXVEDUPRIVXXXXFSOCPRIVXXXXFCMTPRIVXXXX';  {* 48 *}
   MCRS6 = 'VHAWPRIVXXXXFINTPRIVXXXXFCOMPRIVXXXXFTDYPRIVXXXX';  {* 48 *}
   MCRS7 = 'CRSEPRIVXXXXTHTLPRIVXXXXSECLPRIVXXXXVCQRPRIVXXXX';  {* 48 *}
   MCRS8 = 'VREQPRIVXXXXVCBKPRIVXXXXVNMOPRIVXXXXSCHDPRIVXXXX';  {* 48 *}
   MCRS9 = 'CLSRPRIVXXXXTCMFPRIVXXXXFADVPRIVXXXXVINSPRIVXXXX';  {* 48 *}
   MCRS10= 'TADVPRIVXXXXVPDQPRIVXXXXVMSSPRIVXXXXEND.         ';  {* 48 *}


   THES1 = 'SECTIONSAFITDBUPDATENLFACTSHREXXXXDEPTSHREXXXX';    {* 46 *}
   THES2 = 'STDTSHREXXXXTHESPRIVXXXXSECTSHREXXXXMCRSSHREXXXX';  {* 48 *}
   THES3 = 'MQTRSHREXXXXMBKTSHREXXXXMORDSHREXXXXTIMESHREXXXX';  {* 48 *}
```

```
     THES4 = 'BLRMSHREXXXXCPTYSHREXXXXFINTSHREXXXXMSSFSHREXXXX';   {* 48 *}
     THES5 = 'MDEGSHREXXXXVEDUPRIVXXXXFSOCPRIVXXXXFCMTPRIVXXXX';   {* 48 *}
     THES6 = 'VHAWPRIVXXXXFINTPRIVXXXXFCOMPRIVXXXXFTDYPRIVXXXX';   {* 48 *}
     THES7 = 'CRSEPRIVXXXXTHTLPRIVXXXXSECLPRIVXXXXVCQRPRIVXXXX';   {* 48 *}
     THES8 = 'VREQPRIVXXXXVCBKPRIVXXXXVNMOPRIVXXXXSCHDPRIVXXXX';   {* 48 *}
     THES9 = 'CLSRPRIVXXXXTCMFPRIVXXXXFADVPRIVXXXXVINSPRIVXXXX';   {* 48 *}
     THES10= 'TADVPRIVXXXXVPDQPRIVXXXXVMSSPRIVXXXXEND.        ';   {* 48 *}


     MSSF1 = 'SECTIONSAFITDBUPDATENLFACTSHREXXXXDEPTSHREXXXX';     {* 46 *}
     MSSF2 = 'STDTSHREXXXXTHESSHREXXXXSECTSHREXXXXMCRSSHREXXXX';   {* 48 *}
     MSSF3 = 'MQTRSHREXXXXMBKTSHREXXXXMORDSHREXXXXTIMESHREXXXX';   {* 48 *}
     MSSF4 = 'BLRMSHREXXXXCPTYSHREXXXXFINTSHREXXXXMSSFPRIVXXXX';   {* 48 *}
     MSSF5 = 'MDEGPRIVXXXXVEDUPRIVXXXXFSOCPRIVXXXXFCMTPRIVXXXX';   {* 48 *}
     MSSF6 = 'VHAWPRIVXXXXDAYSPRIVXXXXFCOMPRIVXXXXFTDYPRIVXXXX';   {* 48 *}
     MSSF7 = 'CRSEPRIVXXXXTHTLPRIVXXXXSECLPRIVXXXXVCQRPRIVXXXX';   {* 48 *}
     MSSF8 = 'VREQPRIVXXXXVCBKPRIVXXXXVNMOPRIVXXXXSCHDPRIVXXXX';   {* 48 *}
     MSSF9 = 'CLSRPRIVXXXXTCMFPRIVXXXXFADVPRIVXXXXVINSPRIVXXXX';   {* 48 *}
     MSSF10= 'TADVPRIVXXXXVPDQPRIVXXXXVMSSPRIVXXXXEND.        ';   {* 48 *}


TYPE
     BUFF46 = PACKED ARRAY [1..46] OF CHAR;
     BUFF48 = PACKED ARRAY [1..48] OF CHAR;

VAR
     FUNCTIONS : BUFF5;
     SCHEMA: BUFF480;
     ENDIT: BUFF4;
     INDEX : INTEGER;
     OK    : BOOLEAN;

PROCEDURE DATBAS (%STDESCR FUNCTIONS: BUFF5; STATUS : BUFF4;
                  SCHEMA : BUFF480; ENDIT: BUFF4); FORTRAN;
BEGIN
        ENDIT := 'END.';
        FUNCTIONS := ONOROFF;
        IF DATABASE = 'FACT' THEN
           SCHEMA := FACT1 + FACT2 + FACT3 + FACT4 + FACT5 +
                FACT6 + FACT7 + FACT8 + FACT9 +FACT10+ EXTRA2
        ELSE IF DATABASE = 'STDT' THEN
           SCHEMA := STDT1 + STDT2 + STDT3 + STDT4 + STDT5 +
                STDT6 + STDT7 + STDT8 + STDT9 + STDT10 + '  '
        ELSE IF DATABASE = 'MCRS' THEN
           SCHEMA := MCRS1 + MCRS2 + MCRS3 + MCRS4 + MCRS5 +
                MCRS6 + MCRS7 + MCRS8 + MCRS9 + MCRS10+ EXTRA2
        ELSE IF DATABASE = 'THES' THEN
           SCHEMA := THES1 + THES2 + THES3 + THES4 + THES5 +
                THES6 + THES7 + THES8 + THES9 + THES10+ EXTRA2
        ELSE SCHEMA := MSSF1 + MSSF2 + MSSF3 + MSSF4 + MSSF5 +
                MSSF6 + MSSF7 + MSSF8 + MSSF9 + MSSF10 + EXTRA2;
        DATBAS(FUNCTIONS,STATUS,SCHEMA,ENDIT);
        CHECKSTATUS(OK);
        IF OK THEN
```

```
            IF FUNCTIONS = 'SINON' THEN
               BEGIN
                 WRITELN;
                 WRITELN;
                 WRITELN('THE PROGRAM IS SIGNED ON TO THE DATABASE ');
                 WRITELN
               END
            ELSE
              BEGIN
                 WRITELN;
                 WRITELN;
                 WRITELN('THE PROGRAM IS SIGNED OFF OF THE DATABASE ');
                 WRITELN
              END
         ELSE
           BEGIN
             WRITELN;
             WRITELN;
             WRITELN('DATABASE ERROR, DATABASE IS NOT SIGNED ON ')
           END


END;   (*SIGNONOROFF *)
{* LAYER 5 *}
{**************************************************************************}
{*      DATE:  01/08/85                                                  *}
{*      NAME:   WRMSTDT                                                  *}
{*      DESCRIPTION: THIS MODULE WRITES A NEW STUDENT MASTER RECORD TO   *}
{*              THE DATABASE.  THE MODULE EXPECTS THE RECORD TO BE       *}
{*              IN THE FORMAT OF THE STDT_REC DATA TYPE.                 *}
{*                                                                       *}
{*      FILES READ: AFITDB                                               *}
{*      FILES WRITTEN: AFITDB                                            *}
{*      GLOBAL VARIABLES USED: NONE                                      *}
{*      GLOBAL VARIABLES CHANGED: NONE                                   *}
{*      MODULES CALLED: CHECKSTATUS                                      *}
{*      CALLING MODULES:                                                 *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                              *}
{*                                                                       *}
{**************************************************************************}

PROCEDURE  WRMSTDT (STDT:STDT_REC);
VAR
        FUNCTIONS: BUFF5;
        DATASET : BUFF4;
        SSAN    : BUFF9;
        ELEMENTS: BUFF280;
        BUFFER  : BUFF408;
        ENDIT   : BUFF4;
        INDEX   : INTEGER;
        OK      : BOOLEAN;

PROCEDURE DATBAS(%STDESCR FUNCTIONS: BUFF5; STATUS: BUFF4; DATASET: BUFF4;
            SSAN: BUFF9; ELEMENTS: BUFF280; BUFFER:BUFF408; ENDIT:BUFF4);
            FORTRAN;
```

```
BEGIN

        FUNCTIONS := 'WRITM';
        DATASET   := 'STDT';
        SSAN      := STDT.CTRL;
        ELEMENTS := STDTCONST1 + STDTCONST2 + STDTCONST3 +STDTCONST4 +
                    STDTCONST5 + STDTCONST6 + STDTCONST7;
        FOR INDEX := 1 TO 408 DO BUFFER [INDEX] := ' ';
        ENDIT := 'END.';
        WITH STDT DO BEGIN
           BUFFER := CTRL + SEQN + NAME + RANK + GRAD + SRVC + AERO + DORK +
                DOCM + YRSS + SEXX + BOXN + DTSC + PMSC + ADDR + EADR + HMPH +
                DTPH + EDCD + DOBH + POBH + MSTA + SPOS + SDOB + MSPS + NDEP +
                RACE + RELN + LCMD + LORG + TITL + DURN + EXTRA + EXTRA + EXTRA;
                DATBAS (FUNCTIONS,STATUS,DATASET,SSAN,ELEMENTS,BUFFER,ENDIT);
                CHECKSTATUS(OK)
        END
END;




{* LAYER 5 *}
{*************************************************************************}
{*      DATE:  01/08/85                                                  *}
{*      NAME:  RDMFACT                                                   *}
{*      DESCRIPTION: THIS ROUTINE READS A RECORD FROM THE STUDENT        *}
{*              MASTER FILE AND PUTS THE INFORMATION INTO THE RECORD     *}
{*              FACT OF TYPE FACT_REC;                                   *}
{*                                                                       *}
{*      FILES READ: AFITDB                                               *}
{*      FILES WRITTEN:  NONE                                             *}
{*      GLOBAL VARIABLES USED:  NONE                                     *}
{*      GLOBAL VARIABLES CHANGED: NONE                                   *}
{*      MODULES CALLED: CHECKSTATUS                                      *}
{*      CALLING MODULES:                                                 *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                              *}
{*                                                                       *}
{*************************************************************************}

PROCEDURE RDMFACT (VAR FACT:FACT_REC);
VAR
        FUNCTIONS: BUFF5;
        DATASET : BUFF4;
        SSAN    : BUFF9;
        ELEMENTS: BUFF280;
        BUFFER  : BUFF408;
        ENDIT   : BUFF4;
        INDEX,I : INTEGER;
        OK      : BOOLEAN;
```

```
PROCEDURE DATBAS(%STDESCR FUNCTIONS: BUFF5; STATUS: BUFF4; DATASET: BUFF4;
            SSAN: BUFF9; ELEMENTS: BUFF280; BUFFER:BUFF408; ENDIT:BUFF4);
            FORTRAN;

BEGIN

        FUNCTIONS := 'READM';
        DATASET   := 'FACT';
        SSAN      := FACT.CTRL;
        ELEMENTS := FACTCONST1 + FACTCONST2 + FACTCONST3 +FACTCONST4 +
                    FACTCONST5 + FACTCONST6;
        FOR INDEX := 1 TO 408 DO BUFFER [INDEX] := ' ';
        ENDIT := 'END.';
        DATBAS(FUNCTIONS,STATUS,DATASET,SSAN,ELEMENTS,BUFFER,ENDIT);
        CHECKSTATUS(OK);
        IF OK THEN BEGIN
           WITH FACT DO BEGIN
                FOR I := 1 TO 9 DO CTRL [I] := BUFFER [I];
                FOR I := 1 TO 28 DO NAME [I] := BUFFER [I+9];
                FOR I := 1 TO 3 DO RANK [I] := BUFFER [I+37];
           END;
        END;
END;
```

```
{* LAYER 5 *}
{**************************************************************************}
{*      DATE:  01/08/85                                                   *}
{*      NAME:  ADMFACT                                                    *}
{*      DESCRIPTION: THIS MODULE ADDS A FACULTY MASTER RECORD TO THE      *}
{*              AFIT DATABASE SYSTEM ASSUMING THAT THE MEMBER  DOES NOT   *}
{*              EXIT.  THE INFORMATION SHOULD BE  IN THE RECORD PASSED    *}
{*              TO THE MODULE OF TYPE FACT_REC.                           *}
{*                                                                        *}
{*      FILES READ:     NONE                                              *}
{*      FILES WRITTEN:  AFITDB                                            *}
{*      GLOBAL VARIABLES USED:  NONE                                      *}
{*      GLOBAL VARIABLES CHANGED: NONE                                     *}
{*      MODULES CALLED: CHECKSTATUS                                        *}
{*      CALLING MODULES:                                                   *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                                *}
{*                                                                         *}
{***************************************************************************}

PROCEDURE ADMFACT (FACT:FACT_REC);
VAR
        FUNCTIONS: BUFF5;
        DATASET : BUFF4;
        SSAN    : BUFF9;
        ELEMENTS: BUFF280;
        BUFFER  : BUFF408;
        ENDIT   : BUFF4;
        INDEX   : INTEGER;
        OK      : BOOLEAN;

PROCEDURE DATBAS(%STDESCR FUNCTIONS: BUFF5; STATUS: BUFF4; DATASET: BUFF4;
            SSAN: BUFF9; ELEMENTS: BUFF280; BUFFER:BUFF408; ENDIT:BUFF4);
            FORTRAN;

BEGIN

        FUNCTIONS := 'ADD-M';
        DATASET   := 'FACT';
        SSAN      := FACT.CTRL;
        ELEMENTS := FACTCONST1 + FACTCONST2 + FACTCONST3 +FACTCONST4 +
                    FACTCONST5 + FACTCONST6 + FACTCONST7;
        FOR INDEX := 1 TO 408 DO BUFFER [INDEX] := ' ';
        ENDIT := 'END.';
        WITH FACT DO BEGIN
           BUFFER :=CTRL+NAME+RANK+SRVC+DOCM+HDAT+SALR+DOBI+SEXX+AERO+DTSC
                +PMSC+DORK+YRSS+ADDR+HPHN+EADR+MSTA+SPOS+SDOB+NDEP+RACE+RELN+
                OFIC+OPHN+LORG+TITL+DEPT+EXTRA+EXTRA+EXTRA;
                DATBAS (FUNCTIONS,STATUS,DATASET,SSAN,ELEMENTS,BUFFER,ENDIT);
                CHECKSTATUS(OK)
        END
END;
```

```
{* LAYER 5 *}
{*************************************************************************}
{*      DATE:  01/08/85                                                  *}
{*      NAME:  WRMFACT                                                   *}
{*      DESCRIPTION: THIS MODULE UPDATES A FACULTY MASTER RECORD TO THE  *}
{*              AFIT DATABASE SYSTEM ASSUMING THAT THE MEMBER  DOES      *}
{*              EXIST.  THE INFORMATION SHOULD BE  IN THE RECORD PASSED  *}
{*              TO THE MODULE OF TYPE FACT_REC.                          *}
{*                                                                       *}
{*      FILES READ:     NONE                                             *}
{*      FILES WRITTEN:  AFITDB                                           *}
{*      GLOBAL VARIABLES USED:  NONE                                     *}
{*      GLOBAL VARIABLES CHANGED: NONE                                   *}
{*      MODULES CALLED: CHECKSTATUS                                      *}
{*      CALLING MODULES:                                                 *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                              *}
{*                                                                       *}
{*************************************************************************}

PROCEDURE ADMFACT (FACT:FACT_REC);
VAR
        FUNCTIONS: BUFF5;
        DATASET : BUFF4;
        SSAN    : BUFF9;
        ELEMENTS: BUFF280;
        BUFFER  : BUFF408;
        ENDIT   : BUFF4;
        INDEX   : INTEGER;
        OK      : BOOLEAN;

PROCEDURE DATBAS(%STDESCR FUNCTIONS: BUFF5; STATUS: BUFF4; DATASET: BUFF4;
            SSAN: BUFF9; ELEMENTS: BUFF280; BUFFER:BUFF408; ENDIT:BUFF4);
            FORTRAN;

BEGIN

        FUNCTIONS := 'WRITM';
        DATASET  := 'FACT';
        SSAN     := FACT.CTRL;
        ELEMENTS := FACTCONST1 + FACTCONST2 + FACTCONST3 +FACTCONST4 +
                    FACTCONST5 + FACTCONST6 + FACTCONST7;
        FOR INDEX := 1 TO 408 DO BUFFER [INDEX] := ' ';
        ENDIT := 'END.';
        WITH FACT DO BEGIN
           BUFFER :=CTRL+NAME+RANK+SRVC+DOCM+HDAT+SALR+DOBI+SEXX+AERO+DTSC
                +PMSC+DORK+YRSS+ADDR+HPHN+EADR+MSTA+SPOS+SDOB+NDEP+RACE+RELN+
                OFIC+OPHN+LORG+TITL+DEPT+EXTRA+EXTRA+EXTRA;
                DATBAS (FUNCTIONS,STATUS,DATASET,SSAN,ELEMENTS,BUFFER,ENDIT);
                CHECKSTATUS(OK)
        END
```

```
END;
```

```
{* LAYER 5 *}
{*************************************************************************}
{*      DATE:  01/08/85                                                  *}
{*      NAME:  RDMSTDT                                                   *}
{*      DESCRIPTION: THIS ROUTINE READS A RECORD FROM THE STUDENT        *}
{*              MASTER FILE AND PUTS THE INFORMATION INTO THE RECORD     *}
{*              STDT OF TYPE STDT_REC;                                   *}
{*                                                                       *}
{*      FILES READ: AFITDB                                               *}
{*      FILES WRITTEN:  NONE                                             *}
{*      GLOBAL VARIABLES USED:  NONE                                     *}
{*      GLOBAL VARIABLES CHANGED: NONE                                   *}
{*      MODULES CALLED: CHECKSTATUS                                      *}
{*      CALLING MODULES:                                                 *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                              *}
{*                                                                       *}
{*************************************************************************}

PROCEDURE RDMSTDT (VAR STDT:STDT_REC);
VAR
        FUNCTIONS: BUFF5;
        DATASET : BUFF4;
        SSAN    : BUFF9;
        ELEMENTS: BUFF280;
        BUFFER  : BUFF408;
        ENDIT   : BUFF4;
        INDEX   : INTEGER;
        OK      : BOOLEAN;

PROCEDURE DATBAS(%STDESCR FUNCTIONS: BUFF5; STATUS: BUFF4; DATASET: BUFF4;
            SSAN: BUFF9; ELEMENTS: BUFF280; BUFFER:BUFF408; ENDIT:BUFF4);
            FORTRAN;
```

```
BEGIN

        FUNCTIONS := 'READM';
        DATASET   := 'STDT';
        SSAN      := STDT.CTRL;
        ELEMENTS := STDTCONST1 + STDTCONST2 + STDTCONST3 +STDTCONST4 +
                    STDTCONST5 + STDTCONST6 + STDTCONST7;
        FOR INDEX := 1 TO 408 DO BUFFER [INDEX] := ' ';
        ENDIT := 'END.';
        DATBAS(FUNCTIONS,STATUS,DATASET,SSAN,ELEMENTS,BUFFER,ENDIT);
        CHECKSTATUS(OK);
        IF OK THEN BEGIN
           WITH STDT DO BEGIN
                FOR INDEX := 1 TO 6 DO BEGIN
                   DOCM [INDEX] := BUFFER [INDEX + 62];
                   DORK [INDEX] := BUFFER [INDEX + 56];
                   DTSC [INDEX] := BUFFER [INDEX + 75];
                   PMSC [INDEX] := BUFFER [INDEX + 81];
                   DOBH [INDEX] := BUFFER [INDEX + 186];
                   SDOB [INDEX] := BUFFER [INDEX + 245]
                END;
                FOR INDEX := 1 TO 40 DO BEGIN
                   ADDR [INDEX] := BUFFER [INDEX + 87];
                   EADR [INDEX] := BUFFER [INDEX + 127];
                   POBH [INDEX] := BUFFER [INDEX + 192]
                END;
                FOR INDEX := 1 TO 50 DO BEGIN
                   LORG [INDEX] := BUFFER [INDEX + 313];
                   TITL [INDEX] := BUFFER [INDEX + 363]
                END;
                FOR INDEX := 1 TO 2 DO BEGIN
                   SRVC [INDEX] := BUFFER [INDEX + 45];
                   YRSS [INDEX] := BUFFER [INDEX + 68];
                   NDEP [INDEX] := BUFFER [INDEX + 252];
                   RACE [INDEX] := BUFFER [INDEX + 254];
                   RELN [INDEX] := BUFFER [INDEX + 256];
                   DURN [INDEX] := BUFFER [INDEX + 363]
                END;
                FOR INDEX := 1 TO 7 DO BEGIN
                   HMPH [INDEX] := BUFFER [INDEX + 167];
                   DTPH [INDEX] := BUFFER [INDEX + 174]
                END;
                FOR INDEX := 1 TO 5 DO BEGIN
                   EDCD [INDEX] := BUFFER [INDEX + 181];
                   LCMD [INDEX] := BUFFER [INDEX + 258]
                END;
                FOR INDEX := 1 TO 3 DO BEGIN
                   SEQN [INDEX] := BUFFER [INDEX + 9];
                   RANK [INDEX] := BUFFER [INDEX + 40]
                END;
                FOR INDEX := 1 TO 4 DO BEGIN
                   BOXN [INDEX] := BUFFER [INDEX + 71];
                   NAME [INDEX] := BUFFER [INDEX + 12];
                   AERO [INDEX] := BUFFER [INDEX + 46];
                   SPOS [INDEX] := BUFFER [INDEX + 233];
```

```
                           CTRL [INDEX] := BUFFER [INDEX]
                        END;
                   AERO [10] := BUFFER [57];
                        FOR INDEX := 5 TO 9 DO BEGIN
                           NAME [INDEX] := BUFFER [INDEX + 12];
                           SPOS [INDEX] := BUFFER [INDEX + 233];
                           CTRL [INDEX] := BUFFER [INDEX]
                        END;
                        FOR INDEX := 10 TO 12 DO BEGIN
                           NAME [INDEX] := BUFFER [INDEX + 12];
                           SPOS [INDEX] := BUFFER [INDEX + 233];
                        END;
                        FOR INDEX := 13 TO 28 DO
                           NAME [INDEX] := BUFFER [INDEX + 12];
                           SEXX  := BUFFER [70];
                           MSTA  := BUFFER [232];
                           MSPS  := BUFFER [251]
                   END
                END
        END;
```

```
{*************************************************************************}
{*      DATE:  01/08/85                                                  *}
{*      NAME:  ADMSTDT                                                   *}
{*      DESCRIPTION: THIS MODULE ADDS A STUDENT MASTER RECORD TO THE     *}
{*              AFIT DATABASE SYSTEM ASSUMING THAT THE STUDENT DOES NOT  *}
{*              EXIT.  THE INFORMATION SHOULD BE  IN THE RECORD PASSED   *}
{*              TO THE MODULE OF TYPE STDT_REC.                          *}
{*                                                                       *}
{*      FILES READ:     NONE                                             *}
{*      FILES WRITTEN:  AFITDB                                           *}
{*      GLOBAL VARIABLES USED:  NONE                                     *}
{*      GLOBAL VARIABLES CHANGED: NONE                                   *}
{*      MODULES CALLED: CHECKSTATUS                                      *}
{*      CALLING MODULES:                                                 *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                              *}
{*                                                                       *}
{*************************************************************************}

PROCEDURE ADMSTDT (STDT:STDT_REC);
VAR
        FUNCTIONS: BUFF5;
        DATASET : BUFF4;
        SSAN    : BUFF9;
        ELEMENTS: BUFF280;
        BUFFER  : BUFF408;
        ENDIT   : BUFF4;
        INDEX   : INTEGER;
        OK      : BOOLEAN;

PROCEDURE DATBAS(%STDESCR FUNCTIONS: BUFF5; STATUS: BUFF4; DATASET: BUFF4;
            SSAN: BUFF9; ELEMENTS: BUFF280; BUFFER:BUFF408; ENDIT:BUFF4);
            FORTRAN;

BEGIN

        FUNCTIONS := 'ADD-M';
        DATASET   := 'STDT';
        SSAN      := STDT.CTRL;
        ELEMENTS := STDTCONST1 + STDTCONST2 + STDTCONST3 +STDTCONST4 +
                    STDTCONST5 + STDTCONST6 + STDTCONST7;
        FOR INDEX := 1 TO 408 DO BUFFER [INDEX] := ' ';
        ENDIT := 'END.';
        WITH STDT DO BEGIN
           BUFFER := CTRL + SEQN + NAME + RANK + GRAD + SRVC + AERO + DORK +
                DOCM + YRSS + SEXX + BOXN + DTSC + PMSC + ADDR + EADR + HMPH +
                DTPH + EDCD + DOBH + POBH + MSTA + SPOS + SDOB + MSPS + NDEP +
                RACE + RELN + LCMD + LORG + TITL + DURN + EXTRA + EXTRA + EXTRA;
                DATBAS (FUNCTIONS,STATUS,DATASET,SSAN,ELEMENTS,BUFFER,ENDIT);
                CHECKSTATUS(OK)
        END
END;
```

```
{************************************************************************}
{*      DATE:  01/08/85                                                *}
{*      NAME:  WRMSTDT                                                 *}
{*      DESCRIPTION: THIS MODULE WRITES A STUDENT MASTER RECORD TO THE  *}
{*              AFIT DATABASE SYSTEM ASSUMING THAT THE STUDENT DOES NOT *}
{*              EXIT.  THE INFORMATION SHOULD BE  IN THE RECORD PASSED  *}
{*              TO THE MODULE OF TYPE STDT_REC.  THIS MODULE ASSSUMES   *}
{*              THE RECORD ALREADY EXISTS ON THE DATABASE SYSTEM        *}
{*                                                                     *}
{*      FILES READ:     NONE                                           *}
{*      FILES WRITTEN:  AFITDB                                         *}
{*      GLOBAL VARIABLES USED:  NONE                                   *}
{*      GLOBAL VARIABLES CHANGED: NONE                                 *}
{*      MODULES CALLED: CHECKSTATUS                                    *}
{*      CALLING MODULES:                                               *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                            *}
{*                                                                     *}
{************************************************************************}

PROCEDURE WRMSTDT (STDT:STDT_REC);
VAR
        FUNCTIONS: BUFF5;
        DATASET : BUFF4;
        SSAN     : BUFF9;
        ELEMENTS: BUFF280;
        BUFFER   : BUFF408;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS(%STDESCR FUNCTIONS: BUFF5; STATUS: BUFF4; DATASET: BUFF4;
            SSAN: BUFF9; ELEMENTS: BUFF280; BUFFER:BUFF408; ENDIT:BUFF4);
            FORTRAN;

BEGIN

        FUNCTIONS := 'WRITM';
        DATASET   := 'STDT';
        SSAN      := STDT.CTRL;
        ELEMENTS  := STDTCONST1 + STDTCONST2 + STDTCONST3 +STDTCONST4 +
                     STDTCONST5 + STDTCONST6 + STDTCONST7;
        FOR INDEX := 1 TO 408 DO BUFFER [INDEX] := ' ';
        ENDIT := 'END.';
        WITH STDT DO BEGIN
           BUFFER := CTRL + SEQN + NAME + RANK + GRAD + SRVC + AERO + DORK +
                DOCM + YRSS + SEXX + BOXN + DTSC + PMSC + ADDR + EADR + HMPH +
                DTPH + EDCD + DOBH + POBH + MSTA + SPOS + SDOB + MSPS + NDEP +
                RACE + RELN + LCMD + LORG + TITL + DURN + EXTRA + EXTRA + EXTRA;
                DATBAS (FUNCTIONS,STATUS,DATASET,SSAN,ELEMENTS,BUFFER,ENDIT);
                CHECKSTATUS(OK)
        END
END;
```

```
{*************************************************************************}
{*      DATE:  02/08/85                                                 *}
{*      NAME:  WRMSECT                                                  *}
{*      DESCRIPTION: THIS MODULE WRITES AN UPDATED RECORD BACK TO THE   *}
{*              MASTER SECTION FILE OF THE AFIT DATABASE.  THE INFO IS  *}
{*              PASSED TO THE MODULE VIA THE SECT_REC RECORD.           *}
{*                                                                      *}
{*      FILES READ: NONE                                                *}
{*      FILES WRITTEN:  AFITDB                                          *}
{*      GLOBAL VARIABLES USED: NONE                                     *}
{*      GLOBAL VARIABLES CHANGED: NONE                                  *}
{*      MODULES CALLED: CHECKSTATUS                                     *}
{*      CALLING MODULES:                                                *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                             *}
{*                                                                      *}
{*************************************************************************}
PROCEDURE WRMSECT (SECT:SECT_REC);
VAR
        FUNCTIONS: BUFF5;
        DATASET : BUFF4;
        CODE    : BUFF8;
        ELEMENTS: BUFF80;
        BUFFER  : BUFF40;
        ENDIT   : BUFF4;
        INDEX   : INTEGER;
        OK      : BOOLEAN;

PROCEDURE DATBAS(%STDESCR FUNCTIONS: BUFF5; STATUS: BUFF4; DATASET: BUFF4;
            CODE: BUFF8; ELEMENTS: BUFF80; BUFFER:BUFF40; ENDIT:BUFF4);
            FORTRAN;

BEGIN

        FUNCTIONS := 'WRITM';
        DATASET   := 'SECT';
        CODE      := SECT.CTRL;
        ELEMENTS := SECTCONST1+SECTCONST2;
        FOR INDEX := 1 TO 40 DO BUFFER [INDEX] := ' ';
        ENDIT := 'END.';
        WITH SECT DO BEGIN
           BUFFER := CTRL+LSSN+GRDT+ENDT+NRSN+EXTRA;
           DATBAS(FUNCTIONS,STATUS,DATASET,CODE,ELEMENTS,BUFFER,ENDIT);
           CHECKSTATUS(OK);
END
END;
```

```
{*************************************************************************}
{*      DATE:  02/08/85                                                  *}
{*      NAME:  RDMSECT                                                   *}
{*      DESCRIPTION: THIS MODULE READS A RECORD FROM THE MASTER SECTION *}
{*              FILE OF THE AFIT DATABASE AND FORMATS THE SECT_REC       *}
{*              RECORD WITH THE INFORMATION.                             *}
{*                                                                       *}
{*      FILES READ: AFITDB                                               *}
{*      FILES WRITTEN: NONE                                              *}
{*      GLOBAL VARIABLES USED: NONE                                      *}
{*      GLOBAL VARIABLES CHANGED: NONE                                   *}
{*      MODULES CALLED: CHECKSTATUS                                      *}
{*      CALLING MODULES:                                                 *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                              *}
{*                                                                       *}
{*************************************************************************}
PROCEDURE RDMSECT(VAR SECT:SECT_REC; CODE:BUFF8);
VAR

        FUNCTIONS: BUFF5;
        DATASET : BUFF4;
        ELEMENTS: BUFF80;
        BUFFER  : BUFF40;
        ENDIT   : BUFF4;
        INDEX   : INTEGER;
        OK      : BOOLEAN;

PROCEDURE DATBAS(%STDESCR FUNCTIONS: BUFF5; STATUS: BUFF4; DATASET: BUFF4;
            CODE: BUFF8; ELEMENTS: BUFF80; BUFFER:BUFF40; ENDIT:BUFF4);
            FORTRAN;

BEGIN

        FUNCTIONS := 'READM';
        DATASET  := 'SECT';
        CODE     := SECT.CTRL;
        ELEMENTS := SECTCONST1+SECTCONST2;
        FOR INDEX := 1 TO 40 DO BUFFER [INDEX] := ' ';
        ENDIT := 'END.';
        WITH SECT DO BEGIN
           DATBAS(FUNCTIONS,STATUS,DATASET,CODE,ELEMENTS,BUFFER,ENDIT);
           CHECKSTATUS(OK);
           IF OK THEN BEGIN
                FOR INDEX := 1 TO 3 DO BEGIN
                   CTRL [INDEX] := BUFFER [INDEX];
                   LSSN [INDEX] := BUFFER [INDEX + 9];
                   GRDT [INDEX] := BUFFER [INDEX + 18];
                   ENDT [INDEX] := BUFFER [INDEX + 24];
                   NRSN [INDEX] := BUFFER [INDEX + 30]
                END;
                FOR INDEX := 4 TO 6 DO BEGIN
                   CTRL [INDEX] := BUFFER [INDEX];
                   LSSN [INDEX] := BUFFER [INDEX + 9];
                   GRDT [INDEX] := BUFFER [INDEX + 18];
                   ENDT [INDEX] := BUFFER [INDEX + 24];
                END;
                FOR INDEX := 7 TO 8 DO BEGIN
                   CTRL [INDEX] := BUFFER [INDEX];
                   LSSN [INDEX] := BUFFER [INDEX + 9];
                END;
                INDEX := 9;
                LSSN [INDEX] := BUFFER [INDEX + 9];
           END;
     END
END;
```

```
{**********************************************************************}
{*      DATE:  02/08/85                                               *}
{*      NAME:  ADMSECT                                                *}
{*      DESCRIPTION: THIS MODULE WRITES AN NEW  RECORD BACK TO THE    *}
{*              MASTER SECTION FILE OF THE AFIT DATABASE.  THE INFO IS *}
{*              PASSED TO THE MODULE VIA THE SECT_REC RECORD.         *}
{*                                                                    *}
{*      FILES READ: NONE                                              *}
{*      FILES WRITTEN:  AFITDB                                        *}
{*      GLOBAL VARIABLES USED: NONE                                   *}
{*      GLOBAL VARIABLES CHANGED: NONE                                *}
{*      MODULES CALLED: CHECKSTATUS                                   *}
{*      CALLING MODULES:                                              *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                           *}
{*                                                                    *}
{**********************************************************************}
PROCEDURE ADMSECT (SECT:SECT_REC);
VAR
        FUNCTIONS: BUFF5;
        DATASET : BUFF4;
        CODE     : BUFF8;
        ELEMENTS: BUFF80;
        BUFFER  : BUFF40;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS(%STDESCR FUNCTIONS: BUFF5; STATUS: BUFF4; DATASET: BUFF4;
            CODE: BUFF8; ELEMENTS: BUFF80; BUFFER:BUFF40; ENDIT:BUFF4);
            FORTRAN;

BEGIN

        FUNCTIONS := 'ADD-M';
        DATASET  := 'SECT';
        CODE      := SECT.CTRL;
        ELEMENTS := SECTCONST1+SECTCONST2;
        FOR INDEX := 1 TO 40 DO BUFFER [INDEX] := ' ';
        ENDIT := 'END.';
        WITH SECT DO BEGIN
           BUFFER := CTRL+LSSN+GRDT+ENDT+NRSN+EXTRA+EXTRA;
           DATBAS(FUNCTIONS,STATUS,DATASET,CODE,ELEMENTS,BUFFER,ENDIT);
           CHECKSTATUS(OK)
        END
END;
```

```
{************************************************************************}
{*      DATE:  02/08/85                                                 *}
{*      NAME:  WRMMCRS                                                  *}
{*      DESCRIPTION: THIS MODULE WRITES AN UPDATED RECORD TO THE        *}
{*              MASTER COURSE FILE USING THE INFORMATION PASSED IN      *}
{*              THE MCRS RECORD OF TYPE MCRS_REC.                       *}
{*                                                                      *}
{*      FILES READ: NONE                                                *}
{*      FILES WRITTEN:  AFITDB                                          *}
{*      GLOBAL VARIABLES USED: NONE                                     *}
{*      GLOBAL VARIABLES CHANGED: NONE                                  *}
{*      MODULES CALLED: CHECKSTATUS                                     *}
{*      CALLING MODULES:                                                *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                             *}
{*                                                                      *}
{************************************************************************}
PROCEDURE WRMMCRS (MCRS:MCRS_REC);

VAR
        FUNCTIONS: BUFF5;
        DATASET : BUFF4;
        CODE     : BUFF8;
        ELEMENTS: BUFF80;
        BUFFER   : BUFF80;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS(%STDESCR FUNCTIONS: BUFF5; STATUS: BUFF4; DATASET: BUFF4;
            CODE: BUFF8; ELEMENTS: BUFF80; BUFFER:BUFF80; ENDIT:BUFF4);
            FORTRAN;

BEGIN

        FUNCTIONS := 'WRITM';
        DATASET   := 'MCRS';
        CODE      := MCRS.CTRL;
        ELEMENTS := MCRSCONST1+MCRSCONST2;
        FOR INDEX := 1 TO 80 DO BUFFER [INDEX] := ' ';
        ENDIT := 'END.';
        WITH MCRS DO BEGIN
           BUFFER := CTRL+CRHR+LCHR+LBHR+SZLM+TITL+REST+EXTRA+EXTRA;
           DATBAS(FUNCTIONS,STATUS,DATASET,CODE,ELEMENTS,BUFFER,ENDIT);
           CHECKSTATUS(OK)
        END
END;
```

```
{************************************************************************}
{*       DATE:  02/08/85                                                *}
{*       NAME:  RDMMCRS                                                 *}
{*       DESCRIPTION: THIS ROUTINE READS A MASTER COURSE RECORD FROM    *}
{*               THE DATABASE AND FORMATS THE RECORD MCRS OF TYPE       *}
{*               MCRS_REC WITH THE DATA IF THE READ WAS ACCOMPLISHED.   *}
{*                                                                      *}
{*       FILES READ: AFITDB                                             *}
{*       FILES WRITTEN: NONE                                            *}
{*       GLOBAL VARIABLES USED: NONE                                    *}
{*       GLOBAL VARIABLES CHANGED: NONE                                 *}
{*       MODULES CALLED: CHECKSTATUS                                    *}
{*       CALLING MODULES:                                               *}
{*       AUTHOR: DAVID A GAITROS, CAPT, USAF                            *}
{*                                                                      *}
{************************************************************************}
PROCEDURE RDMMCRS (VAR MCRS:MCRS_REC);
VAR
        FUNCTIONS: BUFF5;
        DATASET : BUFF4;
        ELEMENTS: BUFF80;
        BUFFER  : BUFF80;
        ENDIT   : BUFF4;
        INDEX   : INTEGER;
        OK      : BOOLEAN;
        CODE    : BUFF8;
PROCEDURE DATBAS(%STDESCR FUNCTIONS: BUFF5; STATUS: BUFF4; DATASET: BUFF4;
            CODE: BUFF8; ELEMENTS: BUFF80; BUFFER:BUFF80; ENDIT:BUFF4);
            FORTRAN;

BEGIN

        FUNCTIONS := 'READM';
        DATASET   := 'MCRS';
        CODE := MCRS.CTRL;
        ELEMENTS := MCRSCONST1+MCRSCONST2;
        FOR INDEX := 1 TO 80 DO BUFFER [INDEX] := ' ';
        ENDIT := 'END.';
        WITH MCRS DO BEGIN
           DATBAS(FUNCTIONS,STATUS,DATASET,CODE,ELEMENTS,BUFFER,ENDIT);
           CHECKSTATUS(OK);
           IF OK THEN BEGIN
                FOR INDEX := 1 TO 8 DO  CTRL [INDEX] := BUFFER [INDEX];
                FOR INDEX := 1 TO 50 DO TITL [INDEX] := BUFFER [INDEX +13];
                FOR INDEX := 1 TO 2 DO SZLM [INDEX] := BUFFER [INDEX +11];
                CRHR := BUFFER [8];
                LCHR := BUFFER [9];
                LBHR := BUFFER [10];
                REST := BUFFER [63]
           END
        END
END;
```

```
{*************************************************************************}
{*      DATE:  02/08/85                                                  *}
{*      NAME:  ADMMCRS                                                   *}
{*      DESCRIPTION: THIS MODULE ADDS A MASTER RECORD TO THE COURSE      *}
{*              MASTER FILE USING THE INFORMATION PASSED IN THE          *}
{*              MCRS RECORD OF TYPE MCRS_REC.                            *}
{*                                                                       *}
{*      FILES READ: NONE                                                 *}
{*      FILES WRITTEN:  NONE                                             *}
{*      GLOBAL VARIABLES USED: NONE                                      *}
{*      GLOBAL VARIABLES CHANGED: NONE                                   *}
{*      MODULES CALLED: CHECKSTATUS                                      *}
{*      CALLING MODULES:                                                 *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                              *}
{*                                                                       *}
{*************************************************************************}
PROCEDURE ADMMCRS (MCRS:MCRS_REC);

VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        CODE     : BUFF8;
        ELEMENTS: BUFF80;
        BUFFER   : BUFF80;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS(%STDESCR FUNCTIONS: BUFF5; STATUS: BUFF4; DATASET: BUFF4;
            CODE: BUFF8; ELEMENTS: BUFF80; BUFFER:BUFF80; ENDIT:BUFF4);
            FORTRAN;

BEGIN

        FUNCTIONS := 'ADD-M';
        DATASET   := 'MCRS';
        CODE      := MCRS.CTRL;
        ELEMENTS := MCRSCONST1+MCRSCONST2;
        FOR INDEX := 1 TO 80 DO BUFFER [INDEX] := ' ';
        ENDIT := 'END.';
        WITH MCRS DO BEGIN
           BUFFER := CTRL+CRHR+LCHR+LBHR+SZLM+TITL+REST+EXTRA+EXTRA;
           DATBAS(FUNCTIONS,STATUS,DATASET,CODE,ELEMENTS,BUFFER,ENDIT);
           CHECKSTATUS(OK)
        END
END;
```

```
{****************************************************************************}
{*      DATE:  02/08/85                                                     *}
{*      NAME:  WRMMQTR                                                      *}
{*      DESCRIPTION:  THIS MODULE WRITES AN UPDATED RECORD BACK TO THE      *}
{*              AFIT DATABASE IN THE MASTER QUARTER FILE.  THE RECORD       *}
{*              IS PASSSED TO THE MODULE IN THE MQTR_REC FORMAT.            *}
{*                                                                          *}
{*      FILES READ: NONE                                                    *}
{*      FILES WRITTEN: NONE                                                 *}
{*      GLOBAL VARIABLES USED: NONE                                         *}
{*      GLOBAL VARIABLES CHANGED: NONE                                      *}
{*      MODULES CALLED: CHECKSTATUS                                         *}
{*      CALLING MODULES:                                                    *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                                 *}
{*                                                                          *}
{****************************************************************************}
PROCEDURE WRTMSTRMQTR (MQTR:MQTR_REC);

VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        CODE     : BUFF4;
        ELEMENTS: BUFF80;
        BUFFER   : BUFF80;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS(%STDESCR FUNCTIONS: BUFF5; STATUS: BUFF4; DATASET: BUFF4;
            CODE: BUFF4; ELEMENTS: BUFF80; BUFFER:BUFF80; ENDIT:BUFF4);
            FORTRAN;

BEGIN

        FUNCTIONS := 'WRITM';
        DATASET   := 'MQTR';
        CODE      := MQTR.CTRL;
        ELEMENTS := MQTRCONST1 + EXTRA;
        FOR INDEX := 1 TO 80 DO BUFFER [INDEX] := ' ';
        ENDIT := 'END.';
        WITH MQTR DO BEGIN
           BUFFER := CTRL+STDT+SPDT+EXTRA+EXTRA;
           DATBAS(FUNCTIONS,STATUS,DATASET,CODE,ELEMENTS,BUFFER,ENDIT);
           CHECKSTATUS(OK)
        END
END;
```

```
{**************************************************************************}
{*      DATE:  02/08/85                                                  *}
{*      NAME:  RDMMQTR                                                   *}
{*      DESCRIPTION: THIS MODULE READS A MASTER RECORD FROM THE MASTER   *}
{*              QUARTER FILE AND FORMATS THE RECORD "MQTR" OF TYPE       *}
{*              MQTR_REC IF THE READ WAS ACCOMPLISHED.                   *}
{*                                                                       *}
{*      FILES READ: AFITDB                                               *}
{*      FILES WRITTEN:  NONE                                             *}
{*      GLOBAL VARIABLES USED: NONE                                      *}
{*      GLOBAL VARIABLES CHANGED: NONE                                   *}
{*      MODULES CALLED: CHECKSTATUS                                      *}
{*      CALLING MODULES:                                                 *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                              *}
{*                                                                       *}
{**************************************************************************}


PROCEDURE RDMMQTR (VAR MQTR:MQTR_REC; CODE:BUFF4);

VAR
        FUNCTIONS: BUFF5;
        DATASET : BUFF4;
        ELEMENTS: BUFF80;
        BUFFER  : BUFF80;
        ENDIT   : BUFF4;
        INDEX   : INTEGER;
        OK      : BOOLEAN;

PROCEDURE DATBAS(%STDESCR FUNCTIONS: BUFF5; STATUS: BUFF4; DATASET: BUFF4;
            CODE: BUFF4; ELEMENTS: BUFF80; BUFFER:BUFF80; ENDIT:BUFF4);
            FORTRAN;

BEGIN

        FUNCTIONS := 'READM';
        DATASET  := 'MQTR';
        ELEMENTS := MQTRCONST1 + EXTRA;
        FOR INDEX := 1 TO 80 DO BUFFER [INDEX] := ' ';
        ENDIT := 'END.';
        WITH MQTR DO BEGIN
           DATBAS(FUNCTIONS,STATUS,DATASET,CODE,ELEMENTS,BUFFER,ENDIT);
           CHECKSTATUS(OK);
           IF OK THEN BEGIN
                FOR INDEX := 1 TO 4 DO BEGIN
                   CTRL [INDEX] := BUFFER [INDEX];
                   STDT [INDEX] := BUFFER [INDEX + 4];
                   SPDT [INDEX] := BUFFER [INDEX + 10]
                END;
                FOR INDEX := 5 TO 6 DO BEGIN
                   STDT [INDEX] := BUFFER [INDEX + 4];
                   SPDT [INDEX] := BUFFER [INDEX + 10]
                END
            END
        END END;
```

```
{**********************************************************************}
{*      DATE:  02/08/85                                               *}
{*      NAME:  ADMMQTR                                                *}
{*      DESCRIPTION: THIS MODULE ADDS A RECORD TO THE MASTER QUARTER  *}
{*              FILE IN THE AFIT DATABASE.  THE RECORD MUST NO EXIT   *}
{*              AND MUST BE PASSED TO THIS MODULE IN THE MQTR_REC     *}
{*              FORMAT.                                               *}
{*                                                                    *}
{*      FILES READ: NONE                                              *}
{*      FILES WRITTEN:  AFITDB                                        *}
{*      GLOBAL VARIABLES USED: NONE                                   *}
{*      GLOBAL VARIABLES CHANGED: NONE                                *}
{*      MODULES CALLED: CHECKSTATUS                                   *}
{*      CALLING MODULES:                                              *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                           *}
{*                                                                    *}
{**********************************************************************}
PROCEDURE ADMMQTR (MQTR:MQTR_REC);

VAR
        FUNCTIONS: BUFF5;
        DATASET : BUFF4;
        CODE    : BUFF4;
        ELEMENTS: BUFF80;
        BUFFER  : BUFF80;
        ENDIT   : BUFF4;
        INDEX   : INTEGER;
        OK      : BOOLEAN;

PROCEDURE DATBAS(%STDESCR FUNCTIONS: BUFF5; STATUS: BUFF4; DATASET: BUFF4;
            CODE: BUFF4; ELEMENTS: BUFF80; BUFFER:BUFF80; ENDIT:BUFF4);
            FORTRAN;

BEGIN

        FUNCTIONS := 'ADD-M';
        DATASET   := 'MQTR';
        CODE      := MQTR.CTRL;
        ELEMENTS  := MQTRCONST1 + EXTRA;
        FOR INDEX := 1 TO 80 DO BUFFER [INDEX] := ' ';
        ENDIT := 'END.';
        WITH MQTR DO BEGIN
           BUFFER := CTRL+STDT+SPDT+EXTRA+EXTRA;
           DATBAS(FUNCTIONS,STATUS,DATASET,CODE,ELEMENTS,BUFFER,ENDIT);
           CHECKSTATUS(OK)
        END
END;
```

```
{*************************************************************************}
{*      DATE:  08/08/85                                                  *}
{*      NAME:  ADVCRSE                                                   *}
{*      DESCRIPTION: THIS MODULE ADDS A VARIABE RECORD TO THE CRSE       *}
{*              VARIABLE FILE AFTER THE RECORD POINTED TO BY THE         *}
{*              VREFERENCE PARAMETER.  THE INFORMATION PASSED TO THIS    *}
{*              MODULE IS IN THE RECORD FORMAT OF TYPE CRSE_REC.         *}
{*                                                                       *}
{*      FILES READ: NONE                                                 *}
{*      FILES WRITTEN:  AFITDB                                           *}
{*      GLOBAL VARIABLES USED: NONE                                      *}
{*      GLOBAL VARIABLES CHANGED: NONE                                   *}
{*      MODULES CALLED: CHECKSTATUS                                      *}
{*      CALLING MODULES: LAYER 4, AND LAYER 3                            *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                              *}
{*                                                                       *}
{*************************************************************************}


PROCEDURE ADVCRSE(CRSE:CRSE_REC;VAR VREFERENCE:BUFF4;
                                    CODE      :BUFF9);

VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        VLINKPATH: BUFF8;
        ELEMENTS : BUFF80;
        BUFFER   : BUFF240;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS (%STDESCR FUNCTIONS:BUFF5; STATUS:BUFF4; DATASET:BUFF4;
                  VREFERENCE:BUFF4; VLINKPATH:BUFF8; CODE: BUFF9;
                  ELEMENTS: BUFF80; BUFFER: BUFF240; ENDIT: BUFF4); FORTRAN;


BEGIN
        FUNCTIONS := 'ADDVC';
        ENDIT := 'END.';
        DATASET := 'CRSE';
        VLINKPATH := 'STDTLKCR';
        ELEMENTS := CRSECONST1+CRSECONST2;
        FOR INDEX := 1 TO 240 DO
           BUFFER [INDEX] := ' ';
        WITH CRSE DO BEGIN
           BUFFER :=STDT+MDEG+NUMB+NAME+GRAD+BEGN+COLL+WAIV+EXTRA+EXTRA+EXTRA+
                EXTRA+EXTRA;
           DATBAS (FUNCTIONS,STATUS,DATASET,VREFERENCE,VLINKPATH,CODE,ELEMENTS,
                BUFFER,ENDIT);
           CHECKSTATUS(OK)
        END
END; {* ADVCRSE *}
```

```
{**********************************************************************}
{*      DATE:  08/08/85                                               *}
{*      NAME:  WRVCRSE                                                *}
{*      DESCRIPTION: THIS MODULE WRITES AN UPDATED VARIABLE RECORD    *}
{*              FROM THE FILE CRSE TO THE DATABASE TO ITS ORIGINAL    *}
{*              POSITION WITHIN THE STRING.  THE INFORMATION IS PASSED *}
{*              TO THE MODULE VIA THE RECORD IN THE CRSE_REC FORMAT.  *}
{*                                                                    *}
{*      FILES READ: NONE                                              *}
{*      FILES WRITTEN:  AFITDB                                        *}
{*      GLOBAL VARIABLES USED: NONE                                   *}
{*      GLOBAL VARIABLES CHANGED: NONE                                *}
{*      MODULES CALLED: CHECKSTATUS                                   *}
{*      CALLING MODULES: LAYER 4, AND LAYER 3                         *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                           *}
{*                                                                    *}
{**********************************************************************}

PROCEDURE WRVCRSE(CRSE:CRSE_REC;VAR VREFERENCE:BUFF4;
                                    VLINKPATH :BUFF8;
                                    CODE      :BUFF9);

VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        ELEMENTS : BUFF80;
        BUFFER   : BUFF240;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS (%STDESCR FUNCTIONS:BUFF5; STATUS:BUFF4; DATASET:BUFF4;
                   VREFERENCE:BUFF4; VLINKPATH:BUFF8; CODE: BUFF9;
                   ELEMENTS: BUFF80; BUFFER: BUFF240; ENDIT: BUFF4); FORTRAN;


BEGIN
        FUNCTIONS := 'WRITV';
        ENDIT := 'END.';
        DATASET := 'CRSE';
        ELEMENTS := CRSECONST1+CRSECONST2;
        FOR INDEX := 1 TO 240 DO
           BUFFER [INDEX] := ' ';
        WITH CRSE DO BEGIN
           BUFFER :=STDT+MDEG+NUMB+NAME+GRAD+BEGN+COLL+WAIV+EXTRA+EXTRA+EXTRA+
                EXTRA+EXTRA;
           DATBAS (FUNCTIONS,STATUS,DATASET,VREFERENCE,VLINKPATH,CODE,ELEMENTS,
                BUFFER,ENDIT);
           CHECKSTATUS(OK)
        END
END; {* WRVCRSE *}
```

```
{***********************************************************************}
{*      DATE:  08/08/85                                                *}
{*      NAME:  RDVCRSE                                                 *}
{*      DESCRIPTION: THIS MODULE READS A VARIBLE FILE FROM THE DATABASE *}
{*              FROM THE FILE CRSE. IT FORMATS THE INFORMATION INTO    *}
{*              A RECORD OF TYPE CRSE_REC AND RETURNS IT TO THE        *}
{*              CALLING PROCEDURE.                                     *}
{*                                                                     *}
{*      FILES READ: AFITDB                                             *}
{*      FILES WRITTEN:  NONE                                           *}
{*      GLOBAL VARIABLES USED: NONE                                    *}
{*      GLOBAL VARIABLES CHANGED: NONE                                 *}
{*      MODULES CALLED: CHECKSTATUS                                    *}
{*      CALLING MODULES: LAYER 4, AND LAYER 3                          *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                            *}
{*                                                                     *}
{***********************************************************************}

PROCEDURE RDVCRSE(VAR CRSE:CRSE_REC; VAR VREFERENCE:BUFF4;
                                         CODE     :BUFF9);


VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        ELEMENTS : BUFF80;
        VLINKPATH: BUFF8;
        BUFFER   : BUFF240;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS (%STDESCR FUNCTIONS:BUFF5; STATUS:BUFF4; DATASET:BUFF4;
                  VREFERENCE:BUFF4; VLINKPATH:BUFF8; CODE: BUFF9;
                  ELEMENTS: BUFF80; BUFFER: BUFF240; ENDIT: BUFF4); FORTRAN;


BEGIN
        FUNCTIONS := 'READV';
        ENDIT := 'END.';
        DATASET := 'CRSE';
        VLINKPATH := 'STDTLKCR';
        ELEMENTS := CRSECONST1+CRSECONST2;
        FOR INDEX := 1 TO 240 DO
           BUFFER [INDEX] := ' ';
        WITH CRSE DO BEGIN
           DATBAS (FUNCTIONS,STATUS,DATASET,VREFERENCE,VLINKPATH,CODE,
                 ELEMENTS,BUFFER,ENDIT);
                 CHECKSTATUS(OK);
           IF OK THEN BEGIN
                WAIV := BUFFER [75];
                FOR INDEX := 1 TO 9 DO STDT [INDEX] := BUFFER [INDEX];
                FOR INDEX := 1 TO 2 DO MDEG [INDEX] := BUFFER [INDEX +9];
                FOR INDEX := 1 TO 8 DO NUMB [INDEX] := BUFFER [INDEX + 11];
                FOR INDEX := 1 TO 20 DO NAME [INDEX] := BUFFER [INDEX + 19];
```

```
                              FOR INDEX := 1 TO 2 DO GRAD [INDEX] := BUFFER [INDEX + 39];
                              FOR INDEX := 1 TO 4 DO BEGN [INDEX] := BUFFER [INDEX + 41];
                              FOR INDEX := 1 TO 30 DO COLL [INDEX] := BUFFER [INDEX + 45];

                         END
                      END
           END; {* RDVCRSE *}
```

```
{*************************************************************************}
{*      DATE:  08/08/85                                                  *}
{*      NAME:  RDDCRSE                                                   *}
{*      DESCRIPTION: THIS MODULE READS A VARIABLE FILE RECORD FROM THE  *}
{*              FILE CRSE DIRECTLY FROM THE DATABASE USING THE POINTER  *}
{*              PARAMETER "VREFERENCE" AS THE KEY.  THE INFORMATION IS  *}
{*              FORMATTED AND STORED IN A RECORD OF TYPE CRSE_REC;      *}
{*                                                                       *}
{*      FILES READ: AFITDB                                               *}
{*      FILES WRITTEN:  NONE                                             *}
{*      GLOBAL VARIABLES USED: NONE                                      *}
{*      GLOBAL VARIABLES CHANGED: NONE                                   *}
{*      MODULES CALLED: CHECKSTATUS                                      *}
{*      CALLING MODULES: LAYER 4, AND LAYER 3                            *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                              *}
{*                                                                       *}
{*************************************************************************}

PROCEDURE RDDCRSE(VAR CRSE:CRSE_REC; VAR VREFERENCE:BUFF4;
                                         VLINKPATH :BUFF8;
                                         CODE      :BUFF9);


VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        ELEMENTS : BUFF80;
        BUFFER   : BUFF240;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS (%STDESCR FUNCTIONS:BUFF5; STATUS:BUFF4; DATASET:BUFF4;
                   VREFERENCE:BUFF4; VLINKPATH:BUFF8; CODE: BUFF9;
                   ELEMENTS: BUFF80; BUFFER: BUFF240; ENDIT: BUFF4); FORTRAN;


BEGIN
        FUNCTIONS := ´READD´;
        ENDIT := ´END.´;
        DATASET := ´CRSE´;
        ELEMENTS := CRSECONST1+CRSECONST2;
        FOR INDEX := 1 TO 240 DO
           BUFFER [INDEX] := ´ ´;
        WITH CRSE DO BEGIN
           DATBAS (FUNCTIONS,STATUS,DATASET,VREFERENCE,VLINKPATH,CODE,
                 ELEMENTS,BUFFER,ENDIT);
           CHECKSTATUS(OK);
           IF OK THEN BEGIN
                WAIV := BUFFER [75];
                FOR INDEX := 1 TO 9 DO STDT [INDEX] := BUFFER [INDEX];
                FOR INDEX := 1 TO 2 DO MDEG [INDEX] := BUFFER [INDEX +9];
                FOR INDEX := 1 TO 8 DO NUMB [INDEX] := BUFFER [INDEX + 11];
                FOR INDEX := 1 TO 20 DO NAME [INDEX] := BUFFER [INDEX + 19];
                FOR INDEX := 1 TO 2 DO GRAD [INDEX] := BUFFER [INDEX + 39];
```

```
                  FOR INDEX := 1 TO 4 DO BEGN [INDEX] := BUFFER [INDEX + 41];
                  FOR INDEX := 1 TO 30 DO COLL [INDEX] := BUFFER [INDEX + 45];
            END
         END
END; {* RDDCRSE *}
```

```
{*************************************************************************}
{*      DATE:  08/08/85                                                  *}
{*      NAME:  DLDCRSE                                                   *}
{*      DESCRIPTION: THIS MODULE DELETES A RECORD FROM THE DATABASE      *}
{*              FROM THE FILE CRSE.  THIS RECORD MUST HAVE BEEN READ     *}
{*              FIRST WITH INTENT TO UPDATE AND THE RECORD POINTER       *}
{*              PASSED TO THIS MODULE IN THE PARAMETER "VREFERENCE".     *}
{*                                                                       *}
{*      FILES READ: NONE                                                 *}
{*      FILES WRITTEN:   AFITDB                                          *}
{*      GLOBAL VARIABLES USED: NONE                                      *}
{*      GLOBAL VARIABLES CHANGED: NONE                                   *}
{*      MODULES CALLED: CHECKSTATUS                                      *}
{*      CALLING MODULES: LAYER 4, AND LAYER 3                            *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                              *}
{*                                                                       *}
{*************************************************************************}

PROCEDURE DLDCRSE(VAR CRSE:CRSE_REC; VAR VREFERENCE:BUFF4;
                                        CODE     :BUFF9);


VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        VLINKPATH: BUFF8;
        ELEMENTS : BUFF80;
        BUFFER   : BUFF240;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS (%STDESCR FUNCTIONS:BUFF5; STATUS:BUFF4; DATASET:BUFF4;
                   VREFERENCE:BUFF4; VLINKPATH:BUFF8; CODE: BUFF9;
                   ELEMENTS: BUFF80; BUFFER: BUFF240; ENDIT: BUFF4); FORTRAN;


BEGIN
        FUNCTIONS := 'READV';
        ENDIT := 'END.';
        VLINKPATH := 'STDTLKCR';
        DATASET := 'CRSE';
        ELEMENTS := CRSECONST1+CRSECONST2;
        FOR INDEX := 1 TO 240 DO
           BUFFER [INDEX] := ' ';
           DATBAS (FUNCTIONS,STATUS,DATASET,VREFERENCE,VLINKPATH,CODE,
                 ELEMENTS,BUFFER,ENDIT);
           CHECKSTATUS(OK);
           IF (OK) AND (VREFERENCE <> 'END.')THEN BEGIN
           FUNCTIONS := 'READV';
           DATBAS (FUNCTIONS,STATUS,DATASET,VREFERENCE,VLINKPATH,CODE,
                 ELEMENTS,BUFFER,ENDIT);
           CHECKSTATUS(OK);
           END;
END; {* DLDCRSE *}
```

```
{***********************************************************************}
{*      DATE:  08/08/85                                                *}
{*      NAME:  ADCVREQ                                                 *}
{*      DESCRIPTION: THIS MODULE ADDS A VARIABE RECORD TO THE VREQ     *}
{*              VARIABLE FILE AFTER THE RECORD POINTED TO BY THE       *}
{*              VREFERENCE PARAMETER.  THE INFORMATION PASSED TO THIS  *}
{*              MODULE IS IN THE RECORD FORMAT OF TYPE VREQ_REC.       *}
{*                                                                     *}
{*      FILES READ: NONE                                               *}
{*      FILES WRITTEN:  AFITDB                                         *}
{*      GLOBAL VARIABLES USED: NONE                                    *}
{*      GLOBAL VARIABLES CHANGED: NONE                                 *}
{*      MODULES CALLED: CHECKSTATUS                                    *}
{*      CALLING MODULES: LAYER 4, AND LAYER 3                          *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                            *}
{*                                                                     *}
{***********************************************************************}


PROCEDURE ADCVREQ(VREQ:VREQ_REC;VAR VREFERENCE:BUFF4;
                                    VLINKPATH :BUFF8;
                                    CODE      :BUFF8);

VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        ELEMENTS : BUFF80;
        BUFFER   : BUFF240;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS (%STDESCR FUNCTIONS:BUFF5; STATUS:BUFF4; DATASET:BUFF4;
                   VREFERENCE:BUFF4; VLINKPATH:BUFF8; CODE: BUFF8;
                   ELEMENTS: BUFF80; BUFFER: BUFF240; ENDIT: BUFF4); FORTRAN;


BEGIN
        FUNCTIONS := 'ADDVC';
        ENDIT := 'END.';
        DATASET := 'VREQ';
        ELEMENTS := VREQCONST1+VREQCONST2;
        FOR INDEX := 1 TO 240 DO
           BUFFER [INDEX] := ' ';
        WITH VREQ DO BEGIN
           BUFFER :=CODE+NMBR+DATA+RNUM+BLKA+PNUM+BLKB+EXTRA+EXTRA+EXTRA+EXTRA
                +EXTRA;
        END;
           DATBAS (FUNCTIONS,STATUS,DATASET,VREFERENCE,VLINKPATH,CODE,ELEMENTS,
                BUFFER,ENDIT);
           CHECKSTATUS(OK)
END; {* ADVVREQ *}
```

```
{***********************************************************************}
{*      DATE:  08/08/85                                               *}
{*      NAME:  WRVVREQ                                                *}
{*      DESCRIPTION: THIS MODULE WRITES AN UPDATED VARIABLE RECORD    *}
{*              FROM THE FILE VREQ TO THE DATABASE TO ITS ORIGINAL    *}
{*              POSITION WITHIN THE STRING.  THE INFORMATION IS PASSED *}
{*              TO THE MODULE VIA THE RECORD IN THE VREQ_REC FORMAT.  *}
{*                                                                    *}
{*      FILES READ: NONE                                              *}
{*      FILES WRITTEN:  AFITDB                                        *}
{*      GLOBAL VARIABLES USED: NONE                                   *}
{*      GLOBAL VARIABLES CHANGED: NONE                                *}
{*      MODULES CALLED: CHECKSTATUS                                   *}
{*      CALLING MODULES: LAYER 4, AND LAYER 3                         *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                           *}
{*                                                                    *}
{***********************************************************************}

PROCEDURE WRVVREQ(VREQ:VREQ_REC;VAR VREFERENCE:BUFF4;
                                    VLINKPATH :BUFF8;
                                    CODE      :BUFF8);

VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        ELEMENTS : BUFF80;
        BUFFER   : BUFF240;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS (%STDESCR FUNCTIONS:BUFF5; STATUS:BUFF4; DATASET:BUFF4;
                   VREFERENCE:BUFF4; VLINKPATH:BUFF8; CODE: BUFF8;
                   ELEMENTS: BUFF80; BUFFER: BUFF240; ENDIT: BUFF4); FORTRAN;


BEGIN
        FUNCTIONS := 'WRITV';
        ENDIT := 'END.';
        DATASET := 'VREQ';
        ELEMENTS := VREQCONST1+VREQCONST2;
        FOR INDEX := 1 TO 240 DO
           BUFFER [INDEX] := ' ';
        WITH VREQ DO BEGIN
           BUFFER :=CODE+NMBR+DATA+RNUM+BLKA+PNUM+BLKB+EXTRA+EXTRA+EXTRA+
                    EXTRA + EXTRA
        END;
           DATBAS (FUNCTIONS,STATUS,DATASET,VREFERENCE,VLINKPATH,CODE,ELEMENTS,
                BUFFER,ENDIT);
           CHECKSTATUS(OK)
END; {* WRVVREQ *}
```

```
{***********************************************************************}
{*      DATE:  08/08/85                                                *}
{*      NAME:  RDVVREQ                                                 *}
{*      DESCRIPTION: THIS MODULE READS A VARIBLE FILE FROM THE DATABASE *}
{*              FROM THE FILE VREQ. IT FORMATS THE INFORMATION INTO    *}
{*              A RECORD OF TYPE VREQ_REC AND RETURNS IT TO THE        *}
{*              CALLING PROCEDURE.                                     *}
{*                                                                     *}
{*      FILES READ: AFITDB                                             *}
{*      FILES WRITTEN:  NONE                                           *}
{*      GLOBAL VARIABLES USED: NONE                                    *}
{*      GLOBAL VARIABLES CHANGED: NONE                                 *}
{*      MODULES CALLED: CHECKSTATUS                                    *}
{*      CALLING MODULES: LAYER 4, AND LAYER 3                          *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                            *}
{*                                                                     *}
{***********************************************************************}

PROCEDURE RDVVREQ(VAR VREQ:VREQ_REC; VAR VREFERENCE:BUFF4;
                                         VLINKPATH :BUFF8;
                                         CODE      :BUFF8);


VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        ELEMENTS : BUFF80;
        BUFFER   : BUFF240;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS (%STDESCR FUNCTIONS:BUFF5; STATUS:BUFF4; DATASET:BUFF4;
                   VREFERENCE:BUFF4; VLINKPATH:BUFF8; CODE: BUFF8;
                   ELEMENTS: BUFF80; BUFFER: BUFF240; ENDIT: BUFF4); FORTRAN;


BEGIN
        FUNCTIONS := 'READV';
        ENDIT := 'END.';
        DATASET := 'VREQ';
        ELEMENTS := VREQCONST1+VREQCONST2;
        FOR INDEX := 1 TO 240 DO
           BUFFER [INDEX] := ' ';
           DATBAS (FUNCTIONS,STATUS,DATASET,VREFERENCE,VLINKPATH,CODE,
                 ELEMENTS,BUFFER,ENDIT);
           CHECKSTATUS(OK);
        WITH VREQ DO BEGIN
           IF OK THEN BEGIN
                FOR INDEX := 1 TO 2 DO CODE [INDEX] := BUFFER [INDEX];
                FOR INDEX := 1 TO 8 DO NMBR [INDEX] := BUFFER [INDEX+2];
                FOR INDEX := 1 TO 6 DO BEGIN
                   RNUM [INDEX] := BUFFER [INDEX + 22];
                   BLKA [INDEX] := BUFFER [INDEX + 28];
                   PNUM [INDEX] := BUFFER [INDEX + 34];
```

```
                          BLKB [INDEX] := BUFFER [INDEX + 40]
                       END;
                       FOR INDEX := 1 TO 12 DO DATA [INDEX] := BUFFER [INDEX+10]
                  END
               END
       END; {* RDVVREQ *}
```

```
{*************************************************************************}
{*      DATE:  08/08/85                                                  *}
{*      NAME:  RDDVREQ                                                   *}
{*      DESCRIPTION: THIS MODULE READS A VARIABLE FILE RECORD FROM THE   *}
{*              FILE VREQ DIRECTLY FROM THE DATABASE USING THE POINTER   *}
{*              PARAMETER "VREFERENCE" AS THE KEY.  THE INFORMATION IS   *}
{*              FORMATTED AND STORED IN A RECORD OF TYPE VREQ_REC;       *}
{*                                                                       *}
{*      FILES READ: AFITDB                                               *}
{*      FILES WRITTEN:  NONE                                             *}
{*      GLOBAL VARIABLES USED: NONE                                      *}
{*      GLOBAL VARIABLES CHANGED: NONE                                   *}
{*      MODULES CALLED: CHECKSTATUS                                      *}
{*      CALLING MODULES: LAYER 4, AND LAYER 3                            *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                              *}
{*                                                                       *}
{*************************************************************************}

PROCEDURE RDDVREQ(VAR VREQ:VREQ_REC; VAR VREFERENCE:BUFF4;
                                         VLINKPATH :BUFF8;
                                         CODE      :BUFF8);


VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        ELEMENTS : BUFF80;
        BUFFER   : BUFF240;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS (%STDESCR FUNCTIONS:BUFF5; STATUS:BUFF4; DATASET:BUFF4;
                   VREFERENCE:BUFF4; VLINKPATH:BUFF8; CODE: BUFF8;
                   ELEMENTS: BUFF80; BUFFER: BUFF240; ENDIT: BUFF4); FORTRAN;


BEGIN
        FUNCTIONS := 'READD';
        ENDIT := 'END.';
        DATASET := 'VREQ';
        ELEMENTS := VREQCONST1+VREQCONST2;
        FOR INDEX := 1 TO 240 DO
           BUFFER [INDEX] := ' ';
           DATBAS (FUNCTIONS,STATUS,DATASET,VREFERENCE,VLINKPATH,CODE,
                 ELEMENTS,BUFFER,ENDIT);
           CHECKSTATUS(OK);
        WITH VREQ DO BEGIN
           IF OK THEN BEGIN
                FOR INDEX := 1 TO 2 DO CODE [INDEX] := BUFFER [INDEX];
                FOR INDEX := 1 TO 6 DO BEGIN
                   RNUM [INDEX] := BUFFER [INDEX + 22];
                   BLKA [INDEX] := BUFFER [INDEX + 28];
                   PNUM [INDEX] := BUFFER [INDEX + 34]
                END;
```

```
                    FOR INDEX := 1 TO 8 DO NMBR [INDEX] := BUFFER [INDEX +2];
                    FOR INDEX := 1 TO 12 DO DATA [INDEX] := BUFFER [INDEX +10]
               END
            END
    END; {* RDDVREQ *}
```

```
{*************************************************************************}
{*      DATE:  08/08/85                                                  *}
{*      NAME:  DLDVREQ                                                   *}
{*      DESCRIPTION: THIS MODULE DELETES A RECORD FROM THE DATABASE      *}
{*              FROM THE FILE VREQ.  THIS RECORD MUST HAVE BEEN READ     *}
{*              FIRST WITH INTENT TO UPDATE AND THE RECORD POINTER       *}
{*              PASSED TO THIS MODULE IN THE PARAMETER "VREFERENCE".     *}
{*                                                                       *}
{*      FILES READ: NONE                                                 *}
{*      FILES WRITTEN:   AFITDB                                          *}
{*      GLOBAL VARIABLES USED: NONE                                      *}
{*      GLOBAL VARIABLES CHANGED: NONE                                   *}
{*      MODULES CALLED: CHECKSTATUS                                      *}
{*      CALLING MODULES: LAYER 4, AND LAYER 3                            *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                              *}
{*                                                                       *}
{*************************************************************************}

PROCEDURE DLDVREQ(VAR VREQ:VREQ_REC; VAR VREFERENCE:BUFF4;
                                         VLINKPATH :BUFF8;
                                         CODE      :BUFF8);


VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        ELEMENTS : BUFF80;
        BUFFER   : BUFF240;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS (%STDESCR FUNCTIONS:BUFF5; STATUS:BUFF4; DATASET:BUFF4;
                  VREFERENCE:BUFF4; VLINKPATH:BUFF8; CODE: BUFF8;
                  ELEMENTS: BUFF80; BUFFER: BUFF240; ENDIT: BUFF4); FORTRAN;


BEGIN
        FUNCTIONS := 'DELVD';
        ENDIT := 'END.';
        DATASET := 'VREQ';
        ELEMENTS := VREQCONST1+VREQCONST2;
        FOR INDEX := 1 TO 240 DO
           BUFFER [INDEX] := ' ';
           DATBAS (FUNCTIONS,STATUS,DATASET,VREFERENCE,VLINKPATH,CODE,
                 ELEMENTS,BUFFER,ENDIT);
           CHECKSTATUS(OK);
END; {* DLDVREQ *}
```

```
{****************************************************************************}
{*      DATE:19/07/85                                                      *}
{*      NAME:   CONVERT_TO_DISP                                            *}
{*      DESCRIPTION:    CONVERTS A NUMERIC INTEGER INTO A 3 DIGIT          *}
{*              DISPLAY CHARACTERS.                                        *}
{*                                                                         *}
{*      FILES READ:                                                        *}
{*      FILES WRITTEN:                                                     *}
{*      GLOBAL VARIABLES USED:                                             *}
{*      GLOBAL VARIABLES CHANGED:                                          *}
{*      MODULES CALLED:                                                    *}
{*      CALLING MODULES:                                                   *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                                *}
{*                                                                         *}
{****************************************************************************}
FUNCTION CONVERT_TO_DISP(INNUM:INTEGER): BUFF3;
VAR
        HOLDDISP: BUFF3;
        DIVISOR : INTEGER;
        HOLDNUMB: INTEGER;
        INDEX   : INTEGER;
        DISPNUM : PACKED ARRAY [0..9] OF CHAR;

BEGIN
        DISPNUM [0] := '0';
        DISPNUM [1] := '1';
        DISPNUM [2] := '2';
        DISPNUM [3] := '3';
        DISPNUM [4] := '4';
        DISPNUM [5] := '5';
        DISPNUM [6] := '6';
        DISPNUM [7] := '7';
        DISPNUM [8] := '8';
        DISPNUM [9] := '9';
        HOLDNUMB := INNUM;
        DIVISOR  := INNUM;
        FOR INDEX := 3 DOWNTO 1 DO BEGIN
            DIVISOR := DIVISOR DIV 10;
            DIVISOR := DIVISOR*10;  {* STRIP OFF LAST DIGIT *}
            HOLDDISP [INDEX] := DISPNUM [HOLDNUMB - DIVISOR];
            HOLDNUMB := HOLDNUMB DIV 10;
            DIVISOR := DIVISOR DIV 10
        END;
        CONVERT_TO_DISP := HOLDDISP;
END;
```

```
{**************************************************************************}
{*      DATE:                                                             *}
{*      NAME: READSECT                                                    *}
{*      DESCRIPTION:  READS THE SECTION THAT A STUDENT BELONGS TO.        *}
{*                                                                        *}
{*      FILES READ:                                                       *}
{*      FILES WRITTEN:                                                    *}
{*      GLOBAL VARIABLES USED:                                            *}
{*      GLOBAL VARIABLES CHANGED:                                         *}
{*      MODULES CALLED:                                                   *}
{*      CALLING MODULES:                                                  *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                               *}
{*                                                                        *}
{**************************************************************************}
PROCEDURE RDVSECL(VAR SECL:SECL_REC;VAR VREFERENCE:BUFF4; SSAN:BUFF9);
VAR  TEMPBUFF20 : BUFF20;
     OK : BOOLEAN;
     VLINKPATH  : BUFF8;
     FUNCTIONS : BUFF5;
     DATASET : BUFF4;
     ELEMENTS: BUFF80;
     BUFFER : BUFF80;
     RLSE : BUFF4;
     I : INTEGER;
PROCEDURE DATBAS (%STDESCR FUNCTIONS : BUFF5; STATUS : BUFF4; DATASET : BUFF4;
                  VREFERENCE : BUFF4; VLINKPATH : BUFF8; SSAN : BUFF9;
                  ELEMENTS : BUFF80; BUFFER : BUFF80; RLSE :
                  BUFF4); FORTRAN;

BEGIN
    FUNCTIONS := 'READV';
    DATASET := 'SECL';
    RLSE := 'END.';
    VLINKPATH := 'STDTLKSE';
    ELEMENTS := SECLCONST1+EXTRA;
    FOR I := 1 TO 80 DO BUFFER [I] := ' ';
    DATBAS (FUNCTIONS, STATUS, DATASET, VREFERENCE, VLINKPATH, SSAN,
           ELEMENTS, BUFFER, RLSE);
    CHECKSTATUS(OK);
    IF OK THEN BEGIN
        FOR I := 1 TO 8 DO SECL.SECT[I] := BUFFER [I];
        FOR I := 1 TO 9 DO SECL.STDT[I] := BUFFER [I+8];
        FOR I := 1 TO 9 DO SECL.FACT[I] := BUFFER [I+17];
    END;
END;
```

```
{************************************************************************}
{*      DATE: 18/07/85                                                  *}
{*      NAME:   FINDSECTION.PAS                                         *}
{*      DESCRIPTION:    SEARCHES THE STUDENT LINK LIST FOR A STUDENT    *}
{*              THAT BELONGS TO A SPECIFIC SECTION.  THIS ALGORITHM     *}
{*              ASSUMES THE LOCATION PASSED IS EITHER A HEADER RECORD   *}
{*              OR HAS ALREADY BEEN SEARCHED.                           *}
{*                                                                      *}
{*      FILES READ:                                                     *}
{*      FILES WRITTEN:                                                  *}
{*      GLOBAL VARIABLES USED:                                          *}
{*      GLOBAL VARIABLES CHANGED:                                       *}
{*      MODULES CALLED:                                                 *}
{*      CALLING MODULES:                                                *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                             *}
{*                                                                      *}
{************************************************************************}

PROCEDURE FINDSECTION (VAR FIND:LINK_PTR; SEARCHSECT:BUFF8);
   VAR FOUND: BOOLEAN;
BEGIN
        FIND := FIND^.NEXT;
        FOUND := FALSE;
        WHILE (FIND<>NIL) AND (NOT FOUND) DO
           IF FIND^.SECT = SEARCHSECT THEN FOUND := TRUE
           ELSE FIND := FIND^.NEXT
END;
```

```
{***********************************************************************}
{*       DATE:  08/08/85                                               *}
{*       NAME:ADVVCQR                                                  *}
{*       DESCRIPTION: THIS MODULE ADDS A VARIABE RECORD TO THE VCQR    *}
{*               VARIABLE FILE AFTER THE RECORD POINTED TO BY THE      *}
{*               VREFERENCE PARAMETER.  THE INFORMATION PASSED TO THIS *}
{*               MODULE IS IN THE RECORD FORMAT OF TYPE VCQR_REC.      *}
{*                                                                     *}
{*       FILES READ: NONE AT ALL                                       *}
{*       FILES WRITTEN:  AFITDB                                        *}
{*       GLOBAL VARIABLES USED: NONE                                   *}
{*       GLOBAL VARIABLES CHANGED: NONE                                *}
{*       MODULES CALLED: CHECKSTATUS                                   *}
{*       CALLING MODULES: LAYER 4, AND LAYER 3                         *}
{*       AUTHOR: DAVID A GAITROS, CAPT, USAF                           *}
{*                                                                     *}
{***********************************************************************}
PROCEDURE ADVVCQR(VCQR:VCQR_REC;VAR VREFERENCE:BUFF4;CODE:BUFF9);
VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        VLINKPATH: BUFF8;
        ELEMENTS : BUFF80;
        BUFFER   : BUFF240;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS (%STDESCR FUNCTIONS:BUFF5; STATUS:BUFF4; DATASET:BUFF4;
                   VREFERENCE:BUFF4; VLINKPATH:BUFF8; CODE: BUFF9;
                   ELEMENTS: BUFF80; BUFFER: BUFF240; ENDIT: BUFF4); FORTRAN;
BEGIN
        FUNCTIONS := 'ADDVC';
        ENDIT := 'END.';
        VLINKPATH := 'STDTLKCQ';
        DATASET := 'VCQR';
        ELEMENTS := VCQRCONST1+EXTRA;
        FOR INDEX := 1 TO 240 DO
           BUFFER [INDEX] := ' ';
        WITH VCQR DO
           BUFFER :=CODE+NMBR+IDEN+SSSN+EXTRA+EXTRA+EXTRA+EXTRA+EXTRA+EXTRA;
           DATBAS (FUNCTIONS,STATUS,DATASET,VREFERENCE,VLINKPATH,CODE,ELEMENTS,
                BUFFER,ENDIT);
           CHECKSTATUS(OK);
           IF OK THEN WRITELN('RECORD ADDED');
END; {* ADVVCQR *}
```

```
{***********************************************************************}
{*      DATE:  08/08/85                                               *}
{*      NAME:  WRVVCQR                                                *}
{*      DESCRIPTION: THIS MODULE WRITES AN UPDATED VARIABLE RECORD    *}
{*              FROM THE FILE VCQR TO THE DATABASE TO ITS ORIGINAL    *}
{*              POSITION WITHIN THE STRING.  THE INFORMATION IS PASSED *}
{*              TO THE MODULE VIA THE RECORD IN THE VCQR_REC FORMAT.  *}
{*                                                                    *}
{*      FILES READ: NONE                                              *}
{*      FILES WRITTEN:  AFITDB                                        *}
{*      GLOBAL VARIABLES USED: NONE                                   *}
{*      GLOBAL VARIABLES CHANGED: NONE                                *}
{*      MODULES CALLED: CHECKSTATUS                                   *}
{*      CALLING MODULES: LAYER 4, AND LAYER 3                         *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                           *}
{*                                                                    *}
{***********************************************************************}
PROCEDURE WRVVCQR(VCQR:VCQR_REC;VAR VREFERENCE:BUFF4;CODE:BUFF9);
VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        ELEMENTS : BUFF80;
        VLINKPATH: BUFF8;
        BUFFER   : BUFF240;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS (%STDESCR FUNCTIONS:BUFF5; STATUS:BUFF4; DATASET:BUFF4;
                   VREFERENCE:BUFF4; VLINKPATH:BUFF8; CODE: BUFF9;
                   ELEMENTS: BUFF80; BUFFER: BUFF240; ENDIT: BUFF4); FORTRAN;


BEGIN
        FUNCTIONS := 'WRITV';
        VLINKPATH := 'STDTLKCQ';
        ENDIT := 'END.';
        DATASET := 'VCQR';
        ELEMENTS := VCQRCONST1+EXTRA;
        FOR INDEX := 1 TO 240 DO
           BUFFER [INDEX] := ' ';
        WITH VCQR DO
           BUFFER :=CODE+NMBR+IDEN+SSSN+EXTRA+EXTRA+EXTRA+EXTRA+EXTRA+EXTRA;
           DATBAS (FUNCTIONS,STATUS,DATASET,VREFERENCE,VLINKPATH,CODE,ELEMENTS,
                BUFFER,ENDIT);
           CHECKSTATUS(OK)
END; {* WRVVCQR *}
```

```
{**************************************************************************}
{*      DATE:  08/08/85                                                   *}
{*      NAME:  RDVVCQR                                                    *}
{*      DESCRIPTION: THIS MODULE READS A VARIBLE FILE FROM THE DATABASE *}
{*              FROM THE FILE VCQR. IT FORMATS THE INFORMATION INTO      *}
{*              A RECORD OF TYPE VCQR_REC AND RETURNS IT TO THE          *}
{*              CALLING PROCEDURE.                                        *}
{*                                                                        *}
{*      FILES READ: AFITDB                                                *}
{*      FILES WRITTEN:  NONE                                              *}
{*      GLOBAL VARIABLES USED: NONE                                       *}
{*      GLOBAL VARIABLES CHANGED: NONE                                    *}
{*      MODULES CALLED: CHECKSTATUS                                       *}
{*      CALLING MODULES: LAYER 4, AND LAYER 3                             *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                               *}
{*                                                                        *}
{**************************************************************************}

PROCEDURE RDVVCQR(VAR VCQR:VCQR_REC; VAR VREFERENCE:BUFF4;
                                         CODE      :BUFF9);


VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        ELEMENTS : BUFF80;
        BUFFER   : BUFF240;
        VLINKPATH: BUFF8;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;
        I        : INTEGER;
PROCEDURE DATBAS (%STDESCR FUNCTIONS:BUFF5; STATUS:BUFF4; DATASET:BUFF4;
                  VREFERENCE:BUFF4; VLINKPATH:BUFF8; CODE: BUFF9;
                  ELEMENTS: BUFF80; BUFFER: BUFF240; ENDIT: BUFF4); FORTRAN;
BEGIN
        FUNCTIONS := 'READV';
        ENDIT := 'END.';
        VLINKPATH:= 'STDTLKCQ';
        DATASET := 'VCQR';
        ELEMENTS := VCQRCONST1+EXTRA;
        FOR INDEX := 1 TO 240 DO
           BUFFER [INDEX] := ' ';
           DATBAS (FUNCTIONS,STATUS,DATASET,VREFERENCE,VLINKPATH,CODE,
                 ELEMENTS,BUFFER,ENDIT);
                 CHECKSTATUS(OK);
        WITH VCQR DO BEGIN
           IF OK THEN BEGIN
                FOR I := 1 TO 2 DO CODE [I] := BUFFER [I];
                FOR I := 1 TO 8 DO NMBR [I] := BUFFER [I+2];
                FOR I := 1 TO 4 DO IDEN [I] := BUFFER [I+10];

           END
        END
END; {* RDVVCQR *}
```

```
{*************************************************************************}
{*      DATE:  08/08/85                                                  *}
{*      NAME:  RDDVCQR                                                   *}
{*      DESCRIPTION: THIS MODULE READS A VARIABLE FILE RECORD FROM THE   *}
{*              FILE VCQR DIRECTLY FROM THE DATABASE USING THE POINTER   *}
{*              PARAMETER "VREFERENCE" AS THE KEY.  THE INFORMATION IS   *}
{*              FORMATTED AND STORED IN A RECORD OF TYPE VCQR_REC;       *}
{*                                                                       *}
{*      FILES READ: AFITDB                                               *}
{*      FILES WRITTEN:  NONE                                             *}
{*      GLOBAL VARIABLES USED: NONE                                      *}
{*      GLOBAL VARIABLES CHANGED: NONE                                   *}
{*      MODULES CALLED: CHECKSTATUS                                      *}
{*      CALLING MODULES: LAYER 4, AND LAYER 3                            *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                              *}
{*                                                                       *}
{*************************************************************************}

PROCEDURE RDDVCQR(VAR VCQR:VCQR_REC; VAR VREFERENCE:BUFF4;CODE:BUFF9);


VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        ELEMENTS : BUFF80;
        VLINKPATH: BUFF8;
        BUFFER   : BUFF240;
        ENDIT    : BUFF4;
        INDEX,I  : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS (%STDESCR FUNCTIONS:BUFF5; STATUS:BUFF4; DATASET:BUFF4;
                   VREFERENCE:BUFF4; VLINKPATH:BUFF8; CODE: BUFF9;
                   ELEMENTS: BUFF80; BUFFER: BUFF240; ENDIT: BUFF4); FORTRAN;
BEGIN
        FUNCTIONS := 'READD';
        ENDIT := 'END.';
        VLINKPATH := 'STDTLKCQ';
        DATASET := 'VCQR';
        ELEMENTS := VCQRCONST1+EXTRA;
        FOR INDEX := 1 TO 240 DO
           BUFFER [INDEX] := ' ';
           DATBAS (FUNCTIONS,STATUS,DATASET,VREFERENCE,VLINKPATH,CODE,
                 ELEMENTS,BUFFER,ENDIT);
           CHECKSTATUS(OK);
        WITH VCQR DO BEGIN
           IF OK THEN BEGIN
                FOR I := 1 TO 2 DO CODE [I] := BUFFER [I];
                FOR I := 1 TO 8 DO NMBR [I] := BUFFER [I+2];
                FOR I := 1 TO 4 DO IDEN [I] := BUFFER [I+10];
           END
        END
END; {* RDDVCQR *}
```

```
{*********************************************************************}
{*      DATE:  08/08/85                                              *}
{*      NAME:  DLDVCQR                                               *}
{*      DESCRIPTION: THIS MODULE DELETES A RECORD FROM THE DATABASE  *}
{*              FROM THE FILE VCQR.  THIS RECORD MUST HAVE BEEN READ *}
{*              FIRST WITH INTENT TO UPDATE AND THE RECORD POINTER   *}
{*              PASSED TO THIS MODULE IN THE PARAMETER "VREFERENCE". *}
{*                                                                   *}
{*      FILES READ: NONE                                             *}
{*      FILES WRITTEN:   AFITDB                                      *}
{*      GLOBAL VARIABLES USED: NONE                                  *}
{*      GLOBAL VARIABLES CHANGED: NONE                               *}
{*      MODULES CALLED: CHECKSTATUS                                  *}
{*      CALLING MODULES: LAYER 4, AND LAYER 3                        *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                          *}
{*                                                                   *}
{*********************************************************************}

PROCEDURE DLDVCQR(VAR VCQR:.CQR_REC; VAR VREFERENCE:BUFF4;
                                         CODE     :BUFF9);


VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        ELEMENTS : BUFF80;
        BUFFER   : BUFF240;
        VLINKPATH:BUFF8;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS (%STDESCR FUNCTIONS:BUFF5; STATUS:BUFF4; DATASET:BUFF4;
                   VREFERENCE:BUFF4; VLINKPATH:BUFF8; CODE: BUFF9;
                   ELEMENTS: BUFF80; BUFFER: BUFF240; ENDIT: BUFF4); FORTRAN;


BEGIN
        FUNCTIONS := 'DELVD';
        ENDIT := 'END.';
        VLINKPATH := 'STDTLKCQ';
        DATASET := 'VCQR';
        ELEMENTS := VCQRCONST1+EXTRA;
        FOR INDEX := 1 TO 240 DO
           BUFFER [INDEX] := ' ';
           DATBAS (FUNCTIONS,STATUS,DATASET,VREFERENCE,VLINKPATH,CODE,
                 ELEMENTS,BUFFER,ENDIT);
           CHECKSTATUS(OK);
END; {* DLDVCQR *}
```

```
{***********************************************************************}
{*      DATE:  08/08/85                                                *}
{*      NAME:  ADVFADV                                                 *}
{*      DESCRIPTION: THIS MODULE ADDS A VARIABE RECORD TO THE FADV     *}
{*              VARIABLE FILE AFTER THE RECORD POINTED TO BY THE       *}
{*              VREFERENCE PARAMETER.  THE INFORMATION PASSED TO THIS  *}
{*              MODULE IS IN THE RECORD FORMAT OF TYPE FADV_REC.       *}
{*                                                                     *}
{*      FILES READ: NONE                                               *}
{*      FILES WRITTEN:  AFITDB                                         *}
{*      GLOBAL VARIABLES USED: NONE                                    *}
{*      GLOBAL VARIABLES CHANGED: NONE                                 *}
{*      MODULES CALLED: CHECKSTATUS                                    *}
{*      CALLING MODULES: LAYER 4, AND LAYER 3                          *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                            *}
{*                                                                     *}
{***********************************************************************}

PROCEDURE ADVFADV(FADV:FADV_REC;VAR VREFERENCE:BUFF4;CODE:BUFF9);

VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        VLINKPATH: BUFF8;
        ELEMENTS : BUFF80;
        BUFFER   : BUFF200;
        ENDIT    : BUFF4;
        INDEX    : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS (%STDESCR FUNCTIONS:BUFF5; STATUS:BUFF4; DATASET:BUFF4;
                   VREFERENCE:BUFF4; VLINKPATH:BUFF8; CODE: BUFF9;
                   ELEMENTS: BUFF80; BUFFER: BUFF200; ENDIT: BUFF4); FORTRAN;


BEGIN
        FUNCTIONS := 'ADDVC';
        ENDIT := 'END.';
        VLINKPATH := 'STDTLKCQ';
        DATASET := 'FADV';
        ELEMENTS := FADVCONST1+EXTRA;
        FOR INDEX := 1 TO 200 DO
           BUFFER [INDEX] := ' ';
        WITH FADV DO
           BUFFER :=SECT+STDT+FACT+EXTRA+EXTRA+EXTRA+EXTRA+EXTRA;
           DATBAS (FUNCTIONS,STATUS,DATASET,VREFERENCE,VLINKPATH,CODE,ELEMENTS,
                BUFFER,ENDIT);
           CHECKSTATUS(OK)
END; {* ADVFADV *}
```

```
{**********************************************************************}
{*      DATE:  08/08/85                                               *}
{*      NAME:  RDVFADV                                                *}
{*      DESCRIPTION: THIS MODULE READS A VARIABE RECORD FROM THE      *}
{*              VARIABLE FILE AFTER THE RECORD POINTED TO BY THE      *}
{*              VREFERENCE PARAMETER.  THE INFORMATION PASSED FROM THIS *}
{*              MODULE IS IN THE RECORD FORMAT OF TYPE FADV_REC.      *}
{*                                                                    *}
{*      FILES READ: NONE                                              *}
{*      FILES WRITTEN:  AFITDB                                        *}
{*      GLOBAL VARIABLES USED: NONE                                   *}
{*      GLOBAL VARIABLES CHANGED: NONE                                *}
{*      MODULES CALLED: CHECKSTATUS                                   *}
{*      CALLING MODULES: LAYER 4, AND LAYER 3                         *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                           *}
{*                                                                    *}
{**********************************************************************}
PROCEDURE RDVFADV(FADV:FADV_REC;VAR VREFERENCE:BUFF4;CODE:BUFF9);
VAR
        FUNCTIONS: BUFF5;
        DATASET  : BUFF4;
        VLINKPATH: BUFF8;
        ELEMENTS : BUFF80;
        BUFFER   : BUFF200;
        ENDIT    : BUFF4;
        INDEX,I  : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS (%STDESCR FUNCTIONS:BUFF5; STATUS:BUFF4; DATASET:BUFF4;
                  VREFERENCE:BUFF4; VLINKPATH:BUFF8; CODE: BUFF9;
                  ELEMENTS: BUFF80; BUFFER: BUFF200; ENDIT: BUFF4); FORTRAN;


BEGIN
        FUNCTIONS := 'READV';
        ENDIT := 'END.';
        VLINKPATH := 'STDTLKCQ';
        DATASET := 'FADV';
        ELEMENTS := FADVCONST1+EXTRA;
        FOR INDEX := 1 TO 200 DO
           BUFFER [INDEX] := ' ';
        WITH FADV DO BEGIN
           DATBAS (FUNCTIONS,STATUS,DATASET,VREFERENCE,VLINKPATH,CODE,ELEMENTS,
                BUFFER,ENDIT);
           CHECKSTATUS(OK);
           IF OK THEN BEGIN
                FOR I := 1 TO 8 DO SECT [I] := BUFFER [I];
                FOR I := 1 TO 9 DO STDT [I] := BUFFER [I+8];
                FOR I := 1 TO 9 DO FACT [I] := BUFFER [I+17];
           END
        END {* WITH *}
END; {* RDVFADV *}
```

```
{**************************************************************************}
{*      DATE: 9/07/85                                                     *}
{*      NAME:   ADDNAME                                                   *}
{*      DESCRIPTION: THIS MODULE ADDS A NAME TO A LINK LIST PASSED TO     *}
{*              THE MODULE AS A PARAMETER.  THIS ROUTINE WAS DESIGNED     *}
{*              TO ADD TO A FACULTY LIST OR STUDENT LIST.                 *}
{*                                                                        *}
{*      FILES READ:     NONE                                              *}
{*      FILES WRITTEN:  NONE                                              *}
{*      GLOBAL VARIABLES USED: NONE                                       *}
{*      GLOBAL VARIABLES CHANGED: NONE                                    *}
{*      MODULES CALLED: NONE                                              *}
{*      CALLING MODULES:                                                  *}
{*      AUTHOR: DAVID GAITROS                                             *}
{*                                                                        *}
{**************************************************************************}

PROCEDURE ADDNAME (LINK:LINK_PTR; VAR HEAD: LINK_PTR);
   VAR
      FOUND : BOOLEAN;
      PREV,CURR : LINK_PTR;
BEGIN
   FOUND := FALSE;
   IF HEAD = NIL THEN BEGIN
           NEW(CURR);
           CURR^.NAME := '                         ';
           CURR^.CTRL := '000000000';
           CURR^.SECT := '        ';
           CURR^.NEXT := LINK;
           HEAD := CURR         {* FIRST RECORD IN LIST *}
        END
        ELSE BEGIN
           PREV := HEAD;
           CURR := HEAD^.NEXT;
           WHILE ((CURR <> NIL) AND (LINK^.NAME > CURR^.NAME)) DO BEGIN
                   PREV := CURR;
                   CURR := CURR^.NEXT
              END;
           LINK^.NEXT := PREV^.NEXT;
           PREV^.NEXT := LINK;
        END
END;
```

```
{***********************************************************************}
{*      DATE:   09/07/85                                               *}
{*      NAME:   FINDNAME                                               *}
{*      DESCRIPTION: THIS ROUTINE WILL SEARCH THE SPECIFIED LINK LIST  *}
{*              FOR A NAME.  IT WILL ASSUME THAT THE CURRENT LOCATI5~ON *}
{*              HAS ALREADY BEEN SEARCHED AND THAT THE HEAD RECORD IS A *}
{*              IS A DUMMY RECORD.  ALSO, IN CASE OF A RECORD NOT FOUND,*}
{*              THE RETURNED NAME FIELD WILL BE BLANK.  NOTE THAT THE  *}
{*              NAME RETURNED WILL BE THE FULL NAME, AND THE INPUT NAME *}
{*              WILL BE ONLY THE LAST NAME OR ANY PORTION OF IT.       *}
{*      NOTE: WHENEVER USING THIS ROUTINE, BE SURE TO PASS IT A WORK   *}
{*              POINT BECAUSE ITS VALUE WILL BE CHANGED.               *}
{*                                                                     *}
{*      FILES READ:     NONE                                           *}
{*      FILES WRITTEN:  NONE                                           *}
{*      GLOBAL VARIABLES USED:  NONE                                   *}
{*      GLOBAL VARIABLES CHANGED:NONE                                  *}
{*      MODULES CALLED:  NONE                                          *}
{*      CALLING MODULES:                                               *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                            *}
{*                                                                     *}
{***********************************************************************}

PROCEDURE FINDNAME(VAR NAME:BUFF28;  VAR SSANOUT:BUFF9; VAR SEARCH :LINK_PTR);
VAR FOUND,MATCH : BOOLEAN;
    INDEX : INTEGER;
BEGIN
   SSANOUT := '         ';
   FOUND := FALSE;
   SEARCH := SEARCH^.NEXT;
   WHILE (NOT FOUND) AND (SEARCH <> NIL) DO BEGIN
        INDEX := 1; MATCH := TRUE;
        WHILE (MATCH) AND (NAME [INDEX] <> ' ') DO BEGIN
           IF NAME [INDEX] <> SEARCH^.NAME [INDEX] THEN MATCH := FALSE;
           INDEX := INDEX + 1
        END;
        IF MATCH THEN BEGIN
           FOUND := TRUE;
           SSANOUT := SEARCH^.CTRL;
           NAME := SEARCH^.NAME
           END
        ELSE
           SEARCH := SEARCH^.NEXT
    END
END; {* FINDNAME *}
```

```
{***************************************************************************}
{*      DATE:  08/19/85                                                    *}
{*      NAME:  FMS_INITIALIZE                                              *}
{*      DESCRIPTION:  THIS MODULE INITIALIZES THE CALLS TO FMS AND         *}
{*              SETS THE SYSTEM TO READ THE FRAMES IN FROM THE             *}
{*              FRAME LIBRARY SPECIFIED IN THE FDV$OPEN ROUTINE            *}
{*              CALL.                                                      *}
{*      CALLING MODULES: MAIN                                              *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                                *}
{*                                                                         *}
{***************************************************************************}
PROCEDURE FMS_INITIALIZE;

BEGIN
        FDV$ATERM(TCA := TCA, SIZE := 12, CHANNEL := 2);
        FDV$AWKSP (WKSP := WORKSPACE,SIZE := 2000);
        FDV$LOPEN('STDTMOD',1);
END; {* FMS_INITIALIZE *}
```

```
{************************************************************************}
{*      DATE:  08/19/85                                                 *}
{*      NAME:  FMS_CLOSE                                                *}
{*      DESCRIPTION:  THIS MODULE RELEASES THE WORKSPACE AND CLOSES THE *}
{*              FMS LIBRARY FILE INDICATED IN THE FDV$LOPEN STATEMENT   *}
{*              IN THE FMS_INITIALIZE ROUTINE.                          *}
{*                                                                      *}
{*      CALLING MODULES: MAIN                                           *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                             *}
{*                                                                      *}
{************************************************************************}
PROCEDURE FMS_CLOSE;

BEGIN
        FDV$LCLOS;
        FDV$DWKSP (WKSP := WORKSPACE );
        FDV$DTERM (TCA := TCA );
END;    {* FMS_CLOSE *}
```

```
{***********************************************************************}
{*      DATE: 09/07/85                                                 *}
{*      NAME: BUILDLINKLIST                                            *}
{*      DESCRIPTION:    THIS ROUTINE BUILDS THE LINK LIST FOR THE      *}
{*                      FACULTY AND STUDENT FILE.  NAMES ARE STORED IN *}
{*                      ALPHABETIC                                     *}
{*                      ORDER.                                         *}
{*                                                                     *}
{*      FILES READ:   AFITDB                                           *}
{*      FILES WRITTEN:  NONE                                           *}
{*      GLOBAL VARIABLES USED:  NONE                                   *}
{*      GLOBAL VARIABLES CHANGED:NONE                                  *}
{*      MODULES CALLED: ADDFACT                                        *}
{*      CALLING MODULES:                                               *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                            *}
{*                                                                     *}
{***********************************************************************}

PROCEDURE BUILDLINKLIST(VAR HEADER:LINK_PTR; FILEIN:BUFF4);
VAR  NAME: BUFF28;
     SSAN: BUFF9;
     NUMBER: INTEGER;
     NUM   : BUFF3;
     PTR,LINK: LINK_PTR;
     SECTION: BUFF8;
     SECL: SECL_REC;
     VREF : BUFF4;
     ELEMENTS : BUFF40;
     INDEX : INTEGER;
     FUNCTIONS : BUFF5;
     DATSET : BUFF4;
     QUALIFIER: BUFF4;
     ENDIT: BUFF4;
     OK : BOOLEAN;
     BUFFER: BUFF50;
PROCEDURE DATBAS(%STDESCR FUNCTIONS: BUFF5; STATUS: BUFF4;
                DATSET : BUFF4; QUALIFIER:BUFF4; ELEMENTS: BUFF40;
                BUFFER:BUFF50; ENDIT: BUFF4); FORTRAN;

BEGIN
        WRITELN('BUILDING LINK LIST OF ',FILEIN,' FILE, PLEASE STAND BY');
        STATUS := '    ';
        FUNCTIONS := 'RDNXT';
        QUALIFIER := 'BEGN';
        DATSET := FILEIN;
        ENDIT := 'END ';
        ELEMENTS := FILEIN+'CTRL'+FILEIN+'NAME'+FILEIN+'RANKEND.'+EXTRA;
        BUFFER   := '                                                  ';
        DATBAS(FUNCTIONS,STATUS,DATSET,QUALIFIER,ELEMENTS,BUFFER,ENDIT);
        CHECKSTATUS(OK);
        WHILE (OK) AND (QUALIFIER <> 'END.') DO BEGIN
           NEW(LINK);
           FOR INDEX := 1 TO 9 DO
                LINK^.CTRL[INDEX] := BUFFER [INDEX];
           FOR INDEX := 1 TO 28 DO
```

```
                LINK^.NAME [INDEX] := BUFFER [INDEX + 9];
           FOR INDEX := 1 TO 3 DO
                LINK^.RANK [INDEX] := BUFFER [INDEX+37];
           BUFFER   := '                                                  ';
                 IF FILEIN = 'STDT' THEN BEGIN
                  VREF := 'LKXX';
                  RDVSECL(SECL,VREF,LINK^.CTRL);
                  IF STATUS = '****' THEN LINK^.SECT := SECL.SECT;
                 END;
                 ADDNAME(LINK,HEADER);
        DATBAS(FUNCTIONS,STATUS,DATSET,QUALIFIER,ELEMENTS,BUFFER,ENDIT);
                CHECKSTATUS(OK);
        END;
END; {* BUILDLINKLIST *}
```

```
{**********************************************************************}
{*      DATE:  10/01/1985                                             *}
{*      NAME:  BUILDSECT                                              *}
{*      DESCRIPTION: THIS ROUTINE WILL BUILD A SIMPLE LIST OF ALL OF    *}
{*              SECTIONS CURRENTLY IN THE DATABASE.  THEY WILL BE     *}
{*              STORED IN SEQUENTIAL ORDER IN A LIST OF TYPE LIST_ARRAY.*}
{*                                                                    *}
{*      FILES READ: NONE                                              *}
{*      FILES WRITTEN: NONE                                           *}
{*      GLOBAL VARIABLES USED: SECTION                                *}
{*      GLOBAL VARIABLES CHANGED: SETION                              *}
{*      MODULES CALLED:                                               *}
{*      CALLING MODULES: MAIN                                         *}
{*      AUTHOR: DAVID A GAITROS, CAPT, USAF                           *}
{*                                                                    *}
{**********************************************************************}
PROCEDURE BUILDSECT;
VAR

        FUNCTIONS: BUFF5;
        DATASET : BUFF4;
        ELEMENTS: BUFF80;
        SECT : SECT_REC;
        QUALIFIER: BUFF4;
        BUFFER   : BUFF40;
        ENDIT    : BUFF4;
        INDEX,I  : INTEGER;
        OK       : BOOLEAN;

PROCEDURE DATBAS(%STDESCR FUNCTIONS: BUFF5; STATUS: BUFF4; DATASET: BUFF4;
            QUALIFIER:BUFF4; ELEMENTS: BUFF80; BUFFER:BUFF40; ENDIT:BUFF4);
            FORTRAN;

BEGIN

        FUNCTIONS := 'RDNXT';
        DATASET   := 'SECT';
        QUALIFIER := 'BEGN';
        ELEMENTS := SECTCONST1+SECTCONST2;
        FOR INDEX := 1 TO 40 DO BUFFER [INDEX] := ' ';
        ENDIT := 'END.';
        WITH SECT DO BEGIN
           INDEX := 0;
           DATBAS(FUNCTIONS,STATUS,DATASET,QUALIFIER,ELEMENTS,BUFFER,ENDIT);
           CHECKSTATUS(OK);
           WHILE (OK) AND (QUALIFIER <> 'END.') DO BEGIN
            IF OK THEN BEGIN
               FOR I := 1 TO 8 DO
                CTRL[I] := BUFFER [I];
               FOR I := 1 TO 9 DO
                LSSN [I] := BUFFER [I+24];
               INDEX := INDEX + 1;
               SECTION [INDEX] := SECT;
            END;
            DATBAS(FUNCTIONS,STATUS,DATASET,QUALIFIER,ELEMENTS,BUFFER,ENDIT);
```

```
                    CHECKSTATUS(OK);
                   END
                END
        END;
```

Appendix J

Standard AFIT/ENG Database Procedures

This is a guide to the beginning or casual user of the AFIT/ENG Database System. This guide contains instructions on the procedures needed to use the database system, logon to the system, start the database in operation, unlocking the database, creating tape files, printing documents, some of the more requested operations, and solutions to some common problems.

Guide Index

1. HOW TO LOGON TO THE SYSTEM.

The following procedure will allow you access to the system and will start the database system in operation for you. The items in **BOLDFACE** are the ones you are to type in. When the terminal is turned on, be sure the keybourd is in upper case (caps lock) mode for best results with the database. To logon the system, perform the following steps:

```
$USERNAME: AFITDB
$PASSWORD: ENG_AFIT
```

2. HOW TO START THE DATABASE IN OPERATION

When logging on to the system, sometimes an error message such as STATUS = FAIL, COMMUNICATION ERROR (FATAL ERROR) will appear. This means that the TOTAL Data Base Management System is not running. This procedure will allow you to start it in operation and start the program again.

a. Hit the keys "CTRL" and "C" at the same time. This will stop the program and put you into the operating system. A $ should appear on the screen.

b. Type the following commands. The items in **BOLDFACE** are the instructions you must enter.

```
$SET DEF [AFITDB.TOTAL]
$SUBMIT TOTALINIT
   Expect some messages here.
$RUN TOTPRM
  >AFITDB
  >/
$SET DEF [-]
$@AFITDB
```

3. HOW TO UNLOCK THE DATABASE SYSTEM

Sometimes when a program you or someone else is running aborts or stops in an abnormal manner, this locks up the files on the database system. An error message such as **STATUS = LOCK, DATA SET LOCK (F)** will appear when this occurs. To solve this problem, it will be necessary for you to stop the program you are in and unlock the database. To do this follow the steps below.

a. Stop the program by type the keys "CTRL" and "C" at the same time. This will stop the program and issue a "$".

b. Type the following commands. Remember, you need type only those commands in **BOLDFACE**.

```
$SET DEF [AFITDB.TOTAL]
$RUN ULK
  ULK>DBMOD=AFITDB
  ULK>FILES=ALL.
  ULK>/
$SET DEF [-]
$@AFITDB
```

NOTE: A common mistake is forgetting to put the period (.) at the end of **FILES=ALL.**

4. HOW TO CREATE A TAPE FOR RR.

First, all sections must be printed by using the print command in EDPLANS (ED) or selecting a tape generation function. This creates the file SUMMARY.DAT on the disk. This file must be transfered from disk format to a tape. A nine blank or old tape must be mounted on the tape drive (MSA0:) by following the instructions underneath the cover. Once the tape has been threaded, close the lid and press the online button, and then the load button located on the front panel. Perform the followinG steps:

a. Hit the keys "CTRL" and "C" at the same time to stop any program running.

b. Enter the following commands. You must type only those commands shown here in **BOLDFACE.**

```
$INIT MSAØ:
$MOUNT MSAØ:/FOREIGN
$RUN SYS$SYSTEM:FLX
   FLX>/RS
   FLX>MSØ:/ZE/DO
   FLX>NSØ:DUØ:[AFITDB]SUMMARY.DAT
   FLX>MSØ:/DO/DI  {* SHOW LISTING  *}
```

c. Hit the keys "CTRL" and "C" at the same time.

d. Unload the tape.

5. HOW TO SELECT ITEMS

This data base system is called a menu driven program. This means that you perform a function by selecting it from a list of functions shown on the screen. These will appear as one or two alphapbetic codes or numbers. To select a function, just type the number or letter(s) shown beside the function and hit the "RETURN" or "ENTER" key, usually just to the right of the left pinky finger. Anytime you are confused as to what the system is expecting, hit the PF2 key located on the keypad to the right of the key bouard.

6. HOW TO CHECK TO SEE IF TOTAL IS RUNNING.

To check to see if TOTAL is running, you must be in the operating system. Do this by hitting the keys "CTRL" and "C" at the same time and then type the following command:

```
$SHOW QUEUE SYS$BATCH
```

The response, if TOTAL is running, will appear similar to:

```
JOBNAME        USERNAME       ENTRY          STATUS

TOTALINIT      AFITDB         ---            status
```

If TOTAL is not running then usually nothing will appear.

7. WHAT TO DO IF THE SYSTEM IS NOT RUNNING

Perform function number 2 of this document.

8. HOW TO EXIT THE SYSTEM

Follow the directions on the menus. The last menu you see should have been the first one to appear on the screen when you logged on. When you type "EX", this will stop the program and log you off of the system.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distributi nounlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GCS/ENG/85D-5	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering	6b. OFFICE SYMBOL (If applicable) AFIT/ENG
7a. NAME OF MONITORING ORGANIZATION	
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, OH 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)
9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS.
11. TITLE (Include Security Classification) See Box 19	

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.

12. PERSONAL AUTHOR(S)
Gaitros, David A., B.A., CAPT, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
MS THESIS	FROM ______ TO ______	1985 December	271

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB. GR.
09	02	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Data Bases, Data Base Management System, DBMS TOTAL, Network Database, FMS, Forms Management System

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

Title: Implementation of the AFIT/ENG Faculty and Student Database Management System

Thesis Advisor: Dr. Gary B. Lamont, professor, EE Department

Approved for public release: IAW AFR 190-1
[illegible]
LYNN E. WOLAVER 16 JAN 86
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED [✓] SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
Dr. Gary B. Lamont, EE Department	513-255-3450	AFIT/ENG

DD FORM 1473, 83 APR   EDITION OF 1 JAN 73 IS OBSOLETE.   UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

This study took the works of the previous AFIT/ENG Student and Faculty Database System thesis efforts and design and implemented the application software for the project. The basic purpose of the thesis was to provide a sound design for the application programs that would interface with the TOTAL Database Management System and the Forms Management System. The entire system was to be designed with the notion that it would be modified and enhanced. A series of standard interface routines were created to act as a layer between the TOTAL DBMS. The resulting routines were abstracted and used as an extension to the Pascal programming language.

The education plan portion of the database was used as a prototype to develop the requirements of the human-computer interface. The program was then redesigned and implemented using the standard routines and the specifications developed from the prototype. A menu driven system was used to implement the design utilizing the Forms Management System as the screen interface. The education plan program is an example of the structured approach used in interpreting the design of the database system. The program contains examples of scrolled screens, database calls, linked list routines, and data abstraction. Additional programs were written to demonstrate the capabilitees interfacing with the GKS graphics package, transmition of data to the registrars office, and to show the continuity of the design.