

AD-A133 271

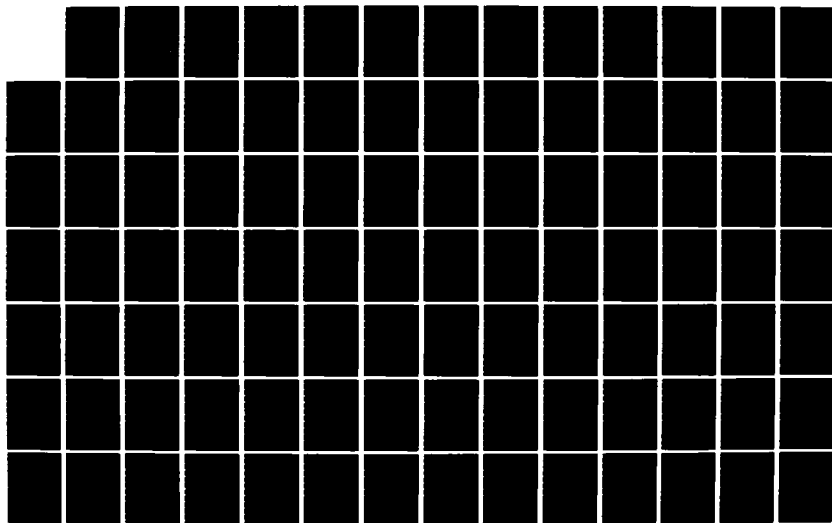
COMPUTER ASSISTED SCHEDULING FOR AIR FORCE TACTICAL
FIGHTER SQUADRONS(U) ARMY COMMAND AND GENERAL STAFF
COLL FORT LEAVENWORTH KS B C DUGLE 03 JUN 83
SBI-AD-E750 845

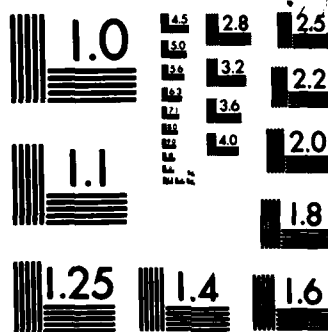
1/2

UNCLASSIFIED

F/G 12/1

NL





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

AD-A133 271

(2)

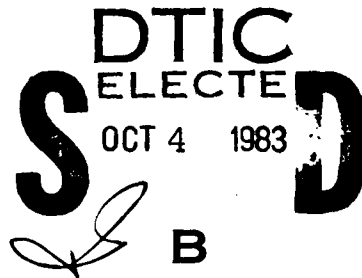
COMPUTER ASSISTED SCHEDULING FOR
AIR FORCE TACTICAL FIGHTER SQUADRONS

A thesis presented to the Faculty of the U.S. Army
Command and General Staff College in partial
fulfillment of the requirements for the
degree

MASTER OF MILITARY ART AND SCIENCE

by

BRIAN C. DUGLE, MAJ, USAF
B.M.E., General Motors Institute, 1968
M.B.A. in Aviation, Embry-Riddle
Aeronautical University, 1982



Fort Leavenworth, Kansas
1983

Approved for public release, distribution unlimited.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
	ADA133 271	
4. TITLE (and Subtitle) COMPUTER ASSISTED SCHEDULING FOR AIR FORCE TACTICAL FIGHTER SQUADRONS		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Dugle, Brian C., Major, USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Student at the U.S. Army Command and General Staff College, Fort Leavenworth, Kansas, 66027		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS HQ TRADOC, Attn: ATCS-D, Fort Monroe, VA 23651		12. REPORT DATE 3 June 1983
		13. NUMBER OF PAGES 119
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Master of Military Art and Science (MMAS) thesis prepared at CGSC in partial fulfillment of the Masters program requirements, U.S. Army Command and General Staff College, Fort Leavenworth, Kansas 66027		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) COMPUTER ASSISTED SCHEDULING, SCHEDULING, MICROCOMPUTER APPLICA- TIONS, PROGRAMMING, BASIC PROGRAMMING, BASIC PROGRAM		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This project develops an algorithm modeling part of the squadron scheduling function. The thesis includes a description of the scheduling function, brief descriptions of some work previously published on computer aids to scheduling, and describes the ap- proach taken in developing the algorithm. The bulk of the thesis is a listing of the programs written to demonstrate the algorithm. The programs are written in Microsft BASIC-80, version 5.21, which is compatible with the Cromemco microcomputers supplied		

to fighter squadrons in the Air Force. The programs allow data entry for a weekly schedule, show which pilots are qualified and available for each activity, and allow selection of an individual pilot for each.

**COMPUTER ASSISTED SCHEDULING FOR
AIR FORCE TACTICAL FIGHTER SQUADRONS**

**A thesis presented to the Faculty of the U.S. Army
Command and General Staff College in partial
fulfillment of the requirements for the
degree**

MASTER OF MILITARY ART AND SCIENCE

by

**BRIAN C. DUGLE, MAJ, USAF
B.M.E., General Motors Institute, 1968
M.B.A. in Aviation, Embry-Riddle
Aeronautical University, 1982**

**Fort Leavenworth, Kansas
1983**

Approved for public release, distribution unlimited.

MASTER OF MILITARY ART AND SCIENCE

THESIS APPROVAL PAGE

Name of Candidate Major Brian C. Dugle
Title of Thesis Computer-Assisted Scheduling for Air Force Fighter
Squadrons

Approved by:

David I. Drummond, Thesis Committee Chairman
Mr. David I. Drummond, MS

Donald F. Hayes, Member, Graduate Faculty
Major Donald F. Hayes, MMAS

Lieutenant Colonel William B. Allard, Member, Consulting Faculty
Lieutenant Colonel William B. Allard, MS

Accepted this 31st day of May 1983 by Philip J. Brooks,
Director, Graduate Degree Program.

The opinions and conclusions expressed herein are those of the student author and do not necessarily represent the views of the U.S. Army Command and General Staff College or any other governmental agency. (References to this study should include the foregoing statement.)

COMPUTER ASSISTED SCHEDULING FOR AIR FORCE TACTICAL FIGHTER SQUADRONS, by Major Brian C. Dugle, USAF, 119 pages.

This project develops an algorithm modeling part of the squadron scheduling function. The thesis includes a description of the scheduling function, brief descriptions of some work previously published on computer aids to scheduling, and describes the approach taken in developing the algorithm. The bulk of the thesis is a listing of the programs written to demonstrate the algorithm. The programs are written in Microsoft BASIC-80, version 5.21, which is compatible with the Cromemco microcomputers supplied to fighter squadrons in the Air Force. The programs allow data entry for a weekly schedule, show which pilots are qualified and available for each activity, and allow selection of an individual pilot for each.



Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

TABLE OF CONTENTS

Chapter		
One.	INTRODUCTION	1
	Background	
	Problem Statement	
	Hypothesis Statement	
	Purpose	
	Organization	
Two.	REVIEW OF LITERATURE	5
	Information Search	
	Computerizing TAC Scheduling	
	A Rand Study	
	Application of Linear Programming	
	A System Now In Use	
	Summary	
Three.	MODEL DESCRIPTION.	11
	Requirement	
	Qualification	
	Availability	
	Currency	
	The Deconfliction Model	
	Model Capabilities and Limitations	
	Summary	
Four.	APPLICATION OF THE DECONFLICTION PROGRAM	16
	Starting Out	
	System Operation	
	Daily Update	
	Summary	
Five.	CONCLUSIONS AND RECOMMENDATIONS.	21

Annex

**A. TRAINING REQUIREMENTS AND THE
SCHEDULING SYSTEM 26**

Introduction
Graduated Combat Capability
General Requirements
MAJCOM Requirements
F-15 Aircrew Training
Summary

B. DATA DEFINITION AND STORAGE. 37

Data Types
Defining What Is Stored
Data Mass Storage Media
Specific Data Types
Summary

C. PROGRAM LISTINGS 47

SELECTED BIBLIOGRAPHY. 117

CHAPTER 1

INTRODUCTION

Background

The most difficult and time consuming task in any flying squadron is scheduling. This job is performed by one or more crewmembers as an additional duty. It requires matching the training requirements of between 40 and several hundred individuals, depending on the type unit, with a schedule of available training assets. These consist of aircraft, flying routes, training areas, ground scoring sites, and numerous ground training events.

A typical fighter squadron today might support training of 40 pilots. Often less than half are available to fill the 16 to 24 sorties flown each day. In addition, some of the pilots are required to fill "duties" such as Runway Supervisory Unit (RSU) officer, Squadron Duty Officer, or Supervisor of Flying (SOF). Alert duty, meetings and appointments must also fit into the schedule.

Many of the squadron pilots have duty positions outside the squadron building or have "additional duties" that take up most of their time. A typical 1- or 2-hour flight itself takes up five to six hours when briefing and debriefing times are included. Flying is considered a

relatively hazardous job; safety considerations dictate a limit to the length of duty in a day which includes flying and a minimum amount of "crew rest" prior to filling flying duties on a subsequent day. Most of the training missions and events have currency or recency restrictions associated with them for much the same reasons. Under some conditions a pilot is restricted to a syllabus or specific order of missions with subsequent flights depending on successful completion of a previous training sortie.

These many factors and constraints make it difficult to devise a schedule that fits, much less one that is optimized. Schedulers often work extremely long hours without much job satisfaction. Operations Officers are generally responsible for the scheduler's product and spend even longer hours reviewing and revising what the scheduler has done.

Problem Statement

Scheduling in a flying unit is highly complex, subject to error, and makes less than optimum use of training resources resulting in discouraged schedulers and reduced combat readiness.

Hypothesis Statement

It is hypothesized that it is possible to aid the scheduler by modeling the scheduling function on a microcomputer and by helping to create alternative schedules. Such a program model must work on the equipment now being deliv-

ered to Air Force fighter squadrons: a Cromemco System 2 using an 8-bit Z-80A microprocessor with the CDOS operating system, two 380 KByte floppy diskette drives, a 5 MByte hard disk, a Zenith Z-19 terminal, a dot matrix or letter quality printer, a modem, and the Microsoft BASIC programming language, version 5¹. The program algorithm must consider all relevant factors or it is unlikely to be used. The system must be flexible to allow for major and minor changes to requirements, availability, and objectives.

Purpose

The purpose of this research is to develop an algorithm to model the squadron scheduling function in sufficient detail to make the product useful. The major difficulty is that the problem is complex, the sources of data diverse, and the guidance subject to many levels of interpretation and emphasis. The goal is to define the logical structure of the scheduling function and translate it into code usable by the available hardware. Initially, the program is to be specifically designed for the F-15 squadrons

¹This description was obtained from Maj Dave Smith, Wing Training Officer of the 1st Tactical Fighter Wing, Langley AFB, VA. His wing is one of the first to receive this hardware and will help to evaluate the programs resulting from this project. The terms "KByte" and "MByte" refer to 1024 and 1,048,576 bytes of mass storage capacity respectively. A byte is one character (eight bits) of data on this system; each byte can have 256 different values (2^8). These values can be interpreted differently in different context which allows flexibility to represent nearly anything.

of Tactical Air Command (TAC) in the continental United States (CONUS) and United States Air Forces in Europe (USAFE). The basic ideas may be expanded to a more general form applicable to other fighter units.

The F-15 unit provides a good starting point for several reasons. The squadron consists only of pilots (single-seat aircraft) which reduces the level of model complexity. The mission includes one major type of flying (Air Superiority) rather than several. Perhaps most important the author's recent experience is with F-15 units which makes their problems more familiar.

Organization

Chapter 2 contains a brief review of some other systems applying computers to the problem of scheduling. Chapter 3 is a description of the model developed in this project. Chapter 4 is a brief guide to its application in the typical fighter squadron. Chapter 5 concludes with a discussion of a system envisioned at the start of this project and recommends areas for further research. Annex A is review of the myriad requirements limiting the scheduler's options such as Air Force flying regulations and manuals and regulations governing scheduling. Annex B covers the storage formats available for the data required to make scheduling decisions. Annex C includes the listings of the programs developed during this project.

CHAPTER 2

REVIEW OF LITERATURE

Information Search

A review of the literature on computer assisted scheduling indicates that anyone currently working on the subject has declined to write about it. Very limited references were found in the Defense Technical Information Center database, the library card catalogs, or in the indexes to various periodicals including the papers written at Maxwell AFB¹. None of these indicated specific work on using micro-computers to aid the scheduler.

One of the few references found includes a very general paper written by Major Richard Strunk submitted as a reserch project to the Air command and Staff College in April, 1977. Some material published on an uncompleted project for the Strategic Air Command as a part of United States Air Force Project Rand represents in-depth study of the subject in a different context. A thesis written by an Air Force officer attending CGSC in 1980 covers a different aspect of the subject. Further digging has uncovered some other work done by industrious individuals which has been

¹Location of the Air War College and the Air Command and Staff College.

described in conversation with the authors but which has not been formally documented for publication.

Computerizing TAC Scheduling

As noted above, the research project prepared by Major Strunk is somewhat general². He stated his objective was to develop and evaluate a Computer Assisted Scheduling Program in order to answer the question, "Can Tactical Air Command (TAC) Operations be computer scheduled?" He described some factors that go into determining how this might be done including a very elaborate flow chart for a series of scheduling programs. The flowchart is 22 pages long and quite detailed. In his concluding chapter, Major Strunk admitted that he was not a computer programmer; his evaluation of the ability to have a computer schedule TAC operations was to state that his flowchart showed it could be done. A portion of the flowcharted program was coded in BASIC, but he observed that it was far from satisfactory in that form.

A Rand Study

Dr. Morton B. Berman of the Rand Corporation spent two years researching and writing a series of reports on a very ambitious project for the Strategic Air Command (SAC)³.

²Richard R. Strunk, Can TAC Operations be Computer Scheduled? (Maxwell AFB, AL: ACSC, 1977).

³Morton B. Berman, The DOSS Prototype. (Santa Monica, CA: Rand Corporation, #WN-9484-PR, 1976, and

A great deal of this time was spent observing flying and maintenance activities and procedures at several SAC bases to gather data on the problem of resource allocations. The last paper published (in 1976) was originally intended only as an interim progress report on development of a Decision Oriented Scheduling System (DOSS) Prototype.

According to the preliminary conclusions and experiences of those using the prototype system, it had great promise. Dr. Berman saw some significant problems ahead but the project was shelved due to a lack of funds before he could complete his work. He stated that his opinion was use of a large mainframe computer (all that was available for his project) was somewhat cumbersome for this type work. He also voiced the opinion that the problem of scheduling in Tactical Air Command type fighter units was much more difficult and involved than in Strategic Air Command, where his work was done⁴.

This difference is one of scope and scale; Dr. Berman's prototype system involved all aspects of both Operations and Maintenance scheduling. This is a more manageable problem in SAC due to the far fewer flights per aircraft per day as compared to fighter operations.

Scheduling Aircrews and Aircraft: Problems of Resource Allocation in the Strategic Air Command. (Santa Monica: Rand Corporation, #R-1610-PR, 1975).

⁴Telephone conversation with Dr. Berman, 13 October 1982.

Application of Linear Programming

Major Carlton L. Pannell submitted a thesis to CGSC entitled A LINEAR PROGRAMMING APPLICATION TO AIRCREW SCHEDULING⁵. The primary thrust of his application was to optimize the distribution of training assets based on scores achieved on the bombing range and a supervisor's subjective evaluation. A section of the thesis was devoted to the specific problem of building and deconflicting a weekly schedule, but not in the detail attempted by this project.

A System Now In Use

The Colorado Air National Guard flying A-7D aircraft (a type of fighter) out of Buckley Field has developed a system that has been working for about four years⁶. The National Guard has unique problems due to the part time nature of many of their personnel and their consequent severely constrained availability. These same factors make it more difficult for them to throw "manhours" at a job (such as scheduling) and live with it, so Major Ron Germano was given the funds to acquire the services of a time-shared mainframe computer to help with scheduling and maintaining records on the pilots of his unit.

⁵Carlton L. Pannell, Major, USAF. A LINEAR PROGRAMMING APPLICATION TO AIRCREW SCHEDULING. (Ft. Leavenworth, KS: CGSC, 1980).

⁶Telephone conversation with Maj Ron Germano, 162d TFS, Buckley Field, CO, 12 October 1982.

He used an established database files structure (supplied as part of the software available with the time-shared system) to store a large volume of information. The accessing methods available with this system allowed him to search for and link data in different data files. Major Germano's program applies arbitrary values or weights to currency and recency data and current training accomplishments data on each pilot and lists the pilot's relative priority for a particular type of training.

The currency and recency items are based on guidance from the various regulations and manuals covering the training required for each category of pilot. The relative weights come from priorities established by the Operations Officer and Commander. Using the priority lists thus developed, the program can then fill a "shell" or listing of the available training missions for a given week.

This system is currently in use, although it is constantly being updated. The product is currently used as a starting point then "hand massaged" to accomodate other constraints. He characterized the accessing language provided for use in manipulating the data stored by the time-shared system as "like Pascal"⁷. With the capabilities of

⁷Pascal is a high-level structured language developed by Dr. Niklaus Wirth of Institut fuer Informatik, ETH Zurich, Switzerland. See Kathleen Jensen and Niklaus Wirth, PASCAL: User Manual and Report: 2d ed, (New York: Springer-Verlag, 1974, corrected printing 1978), and Grogono, Peter. Programming in Pascal: Revised Edition, (Reading, MA: Addison-Wesley, 1980, 1978). Structured programming is also

this language, Major Germano has been able to develop a program which stores the relevant data to make scheduling decision, to assign some factors or values which reflect the guidance of his bosses and higher headquarters on what is acceptable, and to produce a beginning schedule from it.

Summary

The approaches of the systems introduced above vary from that of this project in many ways. With the possible exception of the system being employed by the National Guard unit, little attempt was made to faithfully model the actions of the human scheduler. Deconfliction is the single biggest problem for the human scheduler; it is very difficult to remember every detail about the availability and conflicts of 40 individual pilots. This appears to be the greatest potential contribution of this project.

covered by Brian W. Kernighan, and P. J. Plauger. The Elements of Programming Style: 2d ed (New York: McGraw-Hill, 1978, 1974). The significance of structured programming is its emphasis on "top-down" or big to little structure and the resulting understandability of the code. This concept is one that will be applied in this project.

CHAPTER 3

MODEL DESCRIPTION

Requirement

The requirement of this project is to develop an algorithm modeling the Tactical Fighter Squadron scheduler. This might be done at several different levels of complexity or fidelity; it will be developed here in the simple form including only the pure scheduling function of deconfliction.

Annex A includes specific data used daily by the scheduler and training officer in the typical fighter squadron. Of this data, qualification, availability, and currency are the factors of immediate concern to the scheduler. Since this project is modeling the scheduling function, these are the factors considered.

Qualification

Every scheduled activity includes certain qualification requirements. For an ACBT mission, for instance, the pilot must be qualified for air combat training missions, and must be a flight leader under some conditions. A pilot upgrading to flight lead status would require an IP (Instructor Pilot) or a squadron supervisor on his wing. This

illustrates that each pilot has qualification attributes and each activity on the schedule has qualification requirements.

Availability

Pilots are tasked regularly with meetings, associated with either primary or additional duties, with individual training needs, and with such things as dental appointments and annual physicals. Scheduling coordination for these activities is often made individually with the outside organization involved, either by the pilot himself or by the squadron scheduling or training personnel. Once a pilot is scheduled for such an activity, his availability for normal daily or weekly duties and training missions is restricted. Each activity on the schedule has a scheduled time with attendant time requirements for some period before and after the scheduled time. Any pilot with other commitments any time during the activity period is not available as a candidate to fill that activity. Thus, the activity has an associated required availability period and the pilot has an attribute of available or not available during it.

Currency

Currency restrictions arise primarily from the need for regular practice of flying skills. Inexperienced pilots are often given shorter currency periods than experienced pilots, as shown in Annex A. Leave, extended periods of bad

weather, and other conflicts make it common for a few pilots to be out of currency at any given time. Regaining currencies such as landing, ACBT, or low level intercepts are not too difficult, generally requiring a flight under the supervision of an instructor pilot or a squadron supervisor. One goal of scheduling is to reduce the number of recurrency flights to a minimum to preserve scarce squadron training resources. Again, each activity has associated currency requirements and each pilot has currency attributes.

The Deconfliction Model

At the most simple level, the scheduling algorithm must accomplish "deconfliction". This is the process of making certain no pilot is scheduled for incompatible activities at the same time and that each pilot is qualified and current (or has the required supervision) for the activity scheduled. Further, the process must insure that each activity has someone assigned to do it. In mathematical terms, this model may be described as follows.

Let $\{S_{ij}, 1 \leq i \leq 48, 1 \leq j \leq 7\}$ be a period of time. S_{ij} is the i^{th} half-hour on the j^{th} day of the week and the $\{S_{ij}\}$ refers to a single type of training. In particular, let $\{S_{ij}\}_k$ be the shell slice for the k^{th} activity of K possible types of activities.

Let S_{ijk} be the full shell for all $1 \leq k \leq K$ type

activities to be scheduled.

The problem is to assign the pilots to the shell $\{S_{ijk}\}$, subject to pilot constraints, pilot availability, and the requirements of activity k .

Let f be a function on the elements of the shell S_{ijk} such that $S_{ijk} = 0$ if no pilot is assigned and $S_{ijk} = -1$ if a pilot is assigned.

Thus we want to minimize

$$\sum_{k=1}^K \sum_{j=1}^7 \sum_{i=1}^{48} f(S_{ijk})$$

or, since the day of the week is irrelevant and all half-hour periods are equivalent,

$$\min \sum_{k=1}^K \sum_{i=1}^{336} f(S_{ik})$$

where S_{ik} is the k^{th} activity to be scheduled during the i^{th} half-hour period.

This description of the model shows its simplification to the most basic level of scheduling, that is deconfliction alone. Beyond this point, guidance from the Operations Officer and information from the training section may be used to optimize the training outcome of the scheduled activities. For purposes of this project, only the fit of qualified, available pilots and their currency status will be considered.

Model Capabilities and Limitations

Effective application of the program using this model should reduce the "busy work" and oversights of the scheduler tremendously. The price paid for this aid is that the data used by the system must be kept up-to-date. Qualification data changes least often and could be updated weekly. Availability and currency data must be updated daily if the information is to be of any use.

The deconfliction model is based on the weekly schedule cycle and is updated daily during the execution of the schedule. The program presents candidates for each schedule activity who have the required attributes of qualification and availability and shows if they are current or not. This should allow the scheduler to base his choice on factors outside the model to achieve further training effectiveness.

Summary

Given good data from which to make selections and the speed, responsiveness, and accuracy of operation of the microcomputer, the scheduler's job should become one of selecting the "best" candidate for an activity rather than "a" candidate who seems to fit.

CHAPTER 4

APPLICATION OF THE DECONFLICTION PROGRAM

The program listing included at Annex C is written to apply to a typical single-mission, single-seat fighter squadron. The concept could be expanded to cover broader applications but time constraints precluded that in this project.

Application of the model to the typical scheduling operation should be primarily oriented to the weekly scheduling cycle. Entry of the "shell" and production of multiple weekly data files and schedule files should allow some progress towards optimization beyond the deconfliction model of the program. This chapter will discuss use of the program.

Starting Out

A learning period must be expected before good availability data will be routinely provided to the system. This will require a policy that, after a certain date in the implementation process, scheduling decisions will be based only on data actually provided to the system. Such a "hard line" attitude will be instrumental in getting good availability data into the system at an early stage. This data

must often come from the pilot himself because most availability data generated by the system will be managed internally. Thus, the learning period involves all squadron personnel.

This availability data is one part of the information stored in individual pilot data files. These are sequential files in ASCII¹ character form so that they can be inspected with a simple text editor. A utility with prompts for the information in the proper format is also provided. These files are sequential access files for compactness; their individual nature makes access time a minor consideration, especially with speed of a hard disk.

All other information required for proper operation of the system should come from within the scheduling section. This includes other data contained in the individual pilot files such as name, service number (SSAN), and other administrative data, and qualification and currency data. The qualification names are user definable and may be expanded to much larger capacity than the fifteen slots provided. Each qualification attribute is a "yes" or "no", that is, training or upgrade status qualifications must be handled by a separate qualification name.

Currency data is included in ASCII character form also, but is in Julian date format including a year digit.

¹See Annex B, DATA DEFINITION AND STORAGE, for more explanation on this subject.

This allows easy conversion within the program to a form suitable for comparison with the schedule activity date. Since some currency periods are within the normal scheduling cycle, currency status is provided to the scheduler but is not used as a filter for selecting the candidates for a given activity.

System Operation

Once the pilot data files have been developed, the scheduler must begin entering the weekly schedule shell. This will include all activities for which the squadron must provide pilots. Some of these may be a standard set of duties (for example, SOF, RSU, Alert, and so on) that will be required on a regular or rotating basis. Most flights and ground training events will have to be entered individually each week. All shell data will be stored in a single file for the week including the activity code, the activity time as hours and minutes of the day, the start and end of the activity time period in minutes from the week beginning, and the pilot code if one has been assigned in advance.

Once the shell is complete and the pilot data is available, the scheduler may make any number of attempts at filling the schedule. Each iteration will start with a schedule data file built from the shell data and the pilot data files. Once it has been made, it may be copied and a sequence number assigned to distinguish it from others.

The weekly schedule data file includes pilot

qualification, availability, and currency data in a compact matrix form for quick manipulation by the program. A random access file format aids this speed and ease of access. Also included is the data from the shell on each activity and a matrix of which pilots are qualified and available.

Building a trial schedule requires the scheduler to select an activity to fill, check the candidates provided by the system against outside priorities, and make a tentative selection. After each selection the program must update that pilot's availability data and the pilot availability data for any activities affected by this selection. A flag is provided if the current selection results in the number of candidates for another activity dropping to zero. This condition may be alleviated by using resources outside the squadron or by "un-selecting" that pilot and making another choice. This mechanism provides for minimizing the schedule filling function described in chapter 3.

Once the schedule is completed, an alternate schedule may be developed or this one may be made firm. The firm schedule selected may be used to update the pilot availability data files so that a historic record of all scheduled activity is maintained. This may require periodic purging of old data to keep file sizes and access times acceptable.

Daily Update

The firm schedule will be selected at the time determined by the local scheduling cycle. Once firmed up,

it must continue to be updated with currency information, as well as with any changes made in activities. Since all qualification, currency and availability data for the week is included in the weekly data file, it must be specifically kept up-to-date as pilots accomplish events or sorties which change their status.

If a selection was made based on anticipated events that did not transpire, a check of currency and qualification status on a daily basis will find the problem. Since all availability data and currencies are accessed, the system can be used to find an alternate candidate for the activity or to change the supervision provided.

Summary

Use of the system developed during this project should allow the scheduler to spend his time more productively, resulting in fewer oversight errors and the opportunity to optimize other factors not included in the program. This may result in a higher quality product rather than just a schedule that satisfies.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

The initial goal of this project was to develop a program that would automatically produce a schedule. Several factors made this goal impossible to achieve, the biggest of which was time. This chapter will describe some of the thoughts developed towards this goal so that others working on similar projects may get some insight. In addition, some suggestions are made about the programming language.

The scheduling function is almost inextricably linked with training in a typical fighter squadron. The program developed in this project includes some overlap into the area of training in keeping track of currencies and qualifications. A program that successfully produces a complete schedule will need much more training type information. This data will include much of that described in Annex A, the requirements of TACM 51-50. This must be carefully integrated into the real world system of official Air Force record keeping so that duplication of effort is avoided. This will require retrieval of the official data from its storage medium, usually the base level main frame computer, and operation based on what is stored there.

An alternative is to store the training

accomplishment data on the microcomputer hard disk and supply the base level equipment from there. This could pose some data security problems and is not likely to be approved.

Updating the base level system could be done with the help of a communications package and the modem over normal telephone lines. If two-way data flow could be established with proper safe guards for quality checking on the data sent to the main frame, the microcomputer system could have access to current, accurate training data.

Given this access, further programs could be developed which would allow the computation of the number of requirements remaining for each pilot, in each category of training, and this data could be used to prioritize who would be automatically selected for a given sortie. The priority basis should take into account not only requirements remaining, but also the opportunities remaining to accomplish those requirements. A quotient of remaining divided by opportunities would produce a fractional number which would contain this relative value.

The priority established for one person to use a sortie might have relatively little to do with a different need by another for the same activity. For instance, one pilot might require an ACBT sortie for training while another required it for currency. The decision on who needed it might be based on subjective data or data not available

to the microcomputer, but this could be simulated by applying a weight factor to each persons need. This weight factor would then be the means of providing differing strategies of schedule filling for the program. One strategy weighting currency very heavily might result in a totally different product than another which weighted training needs more heavily.

Another area for further research is the language used in developing this program. Interpreted BASIC is relatively slow compared to some other languages, and its structure allows rather poor programming practices. This has been avoided as much as possible during this project but no doubt has crept in. The need for an easily understood program in this instance, as in many cases, is the need many users will find to change it, whether slightly or greatly. Even a well written program will take many hours of study to become familiar with the author's pattern or structure. A poorly written program may be totally undecipherable even to the author in six months time. Conversion of the basic ideas of this program into another language such as Pascal or Modula 2, a new language introduced by Dr. Niklaus Wirth, could prove very beneficial in the future¹.

The other possibility for increasing the speed of this program would be to compile it into machine code. The producer of this dialect of BASIC, Microsoft Corporation,

¹See Annex B for more information on programming.

has a compiler for it. However, the compiler places further limitations on the structure available; the program included here was not written within these limitations.

Thus, there are three recommendations: expand the project to include the training data needed to produce a schedule automatically; write the code in a better, faster language; or modify the code to allow compiling it for greater speed.

ANNEXES

ANNEX A

TRAINING REQUIREMENTS AND THE SCHEDULING SYSTEM

Introduction

Training is normally considered to be a separate functional area within the staff structure of a fighter squadron, however, the training requirements of each individual are what drive the formation of the schedule. This annex will describe the requirements levied in TAC MANUAL 51-50¹ and the resulting scheduling decisions that a algorithm must model faithfully.

This project has been limited to the goal of developing the scheduling algorithm, but the data needed to make effective decisions for scheduling will in many cases be identical to that needed for planning by both schedulers and the training staff. Some of this data must be stored in official Air Force records such as Air Force Technical Order (AFTO) form 781, Aerospace Vehicle Flight Data Document, in Flight Records, and either TAC Automated Flying Training Management System (TAFTRAMS) or Air Force Operations Management System (AFORMS). The latter, AFORMS, is to be a

¹DEPARTMENT OF THE AIR FORCE, Headquarters Tactical Air Command, FLYING TRAINING: TACTICAL FIGHTER/RECONNAISSANCE AIRCREW TRAINING. TAC MANUAL 51-50, Volume I, 26 October 1981.

universal training and flight data system which all units will eventually use. For these reasons, the implementation of the algorithm devised in this project must be consistent with the basic information format and needs of these systems or the goal of usefulness will not be met.

Another impact of the training manual requirements on the scheduling system is the need to forecast the specific needs of the unit as a whole. Although this is again normally a training staff function, the scheduler is often in the middle of the process because of his direct use of the results.

Graduated Combat Capability

TACM 51-50² is based on the concept that the unit commander, normally the squadron commander, has the best knowledge of the specific training needs of his pilots. The Graduated Combat Capability (GCC) system gives him the ability to assign training assets to achieve various levels of capability depending on the amount of those assets and the experience and individual ability of his people. Volume I of the manual is common to the three Tactical Air Force (TAF) Major Commands (MAJCOMs), TAC, USAFE, and PACAF. Chapter 6 of Volume I is written by each of the MAJCOMs to

²Abbreviations for the manuals in this chapter will be: TACM 51-50 for reference to the whole series of volumes, Volume I for that specific volume exclusive of the MAJCOM chapter, TAC Chapter 6 or USAFE Chapter 6 for their respective chapters, and Volume VII for the F-15 specific volume.

reflect the individual needs of the theater and mission, and applies to all types of fighters in each MAJCOMs inventory. The subsequent volumes of TACM 51-50 reflect the training requirements unique to the specific aircraft. Volume VII includes this information for the F-15.

The training of all aircrew members is broken down into three basic phases by TACM 51-50. IQT is the Initial Qualification Training phase and is normally completed at an RTU or Replacement Training Unit. There are occasions when an operational unit must "train from scratch", but they are kept to a minimum.

MQT is the Mission Qualification Training phase that leads to the first or lowest level of Mission Ready (MR) status. MQT is accomplished in part at the RTU and completed at the gaining operational unit. An aircrew completing MQT at his unit is qualified at level A of the unit Designed Operational Capability (DOC) and can effectively accomplish the units basic mission.

The final type training covered in TACM 51-50 is CT, Continuation Training. This is the day-to-day training accomplished by all the squadron pilots to maintain their mission proficiency or to advance to a higher level. The squadron scheduler is concerned with the requirements of MQT and CT training and the many upgrade programs that fall in these areas. The algorithm modeling the scheduler must allow for making decisions based on diverse requirements of

these programs.

General Requirements

The flying training requirements of Volume I are specified in Table 3-1:

- 6 penetrations (instrument flying)
- 12 precision approaches
- 12 non-precision approaches
- 2 night landings
- 3 air-to-air refuelings (AAR)
- 2 night sorties (credited if 1 hour or 60% of the flight was during darkness)
- 30 minimum total sorties

These requirements apply to all fighter aircraft training regardless of the specific type (albeit with some exceptions) but do not address the training needs of specific missions. The specified training must be accomplished during each training cycle; these are defined as six-month periods beginning 1 January and 1 July. Additional requirements of Volume I include Annual Instrument and Mission or Tactical Qualification evaluation flights and associated examinations, Aircrew Weapons and Tactics Academics, and Target Area Certification or Verification. Rules and supporting tables are provided for prorating training requirements of arriving or departing personnel (who are available only part of the training period) or for other contingencies.

Certain reports of individual and unit capability status are based on the number of sorties flown each month by the squadrons pilots. Since the scheduler is the primary planner of sorties, the sustainability of a given sortie rate is within his purview even though the report itself is normally prepared by the training staff. This reporting philosophy is specified in Volume I. Also included are various definitions of types of training sorties, only one of which can be accomplished per flight, and events of which several may be accomplished.

MAJCOM Requirements

The final chapter of Volume I is written individually by each of the TAF MAJCOMs. This project is involved with two of these: TAC Chapter 6, TAC AND ARF³ AIRCREW TRAINING⁴ and USAFE Chapter 6, TACTICAL FIGHTER/RECONNAISSANCE AIRCREW TRAINING⁵. These additions are applicable to all types of fighters but specific to the command of their assignment.

TAC's Chapter 6 specifies the type of training data

³ARF is Air Reserve Forces, both Air National Guard and US Air Force Reserve.

⁴DEPARTMENT OF THE AIR FORCE, Headquarters Tactical Air Command, TAC AND ARF TRAINING: FIGHTER AND RECONNAISSANCE. TACM 51-50 Volume I, Chapter 6, 15 February 1982.

⁵DEPARTMENT OF THE AIR FORCE, Headquarters United States Air Forces in Europe, Flying Training: TACTICAL FIGHTER/RECONNAISSANCE AIRCREW TRAINING. USAFE Chapter 6 to TACM 51-50, Volume I, 1 October 1982.

that must be tracked if the unit does not have TAFTRAMS or AFORMS available. This data is basically that which is needed to make scheduling decisions. It includes:

- Unit sorties required and accomplished
- Individual sortie standards
- Requirements and accomplishments for each assigned GCC level
- Totals for each month for the semi-annual training period
- Individual monthly flying time accomplished
- Individual required events accomplished
- Individual weapon delivery data on events required for MR qualification⁶

TAC Chapter 6 also defines the types of ground training in three categories: Category I - Mission Essential, Category II - General Flying Related, and Category III - Other Training Related to Aircrews. This training must also be scheduled and affects availability both during conduct of the training and by its effect on crew rest.

Additional training guidance included in this document covers instrument training requirements, the composition of Realistic Training Sorties, Red Flag or equivalent training, Chemical Warfare Defense (CWD) training (in the aircraft and simulator), instructor currency and minimums, and additional TAC semi-annual requirements. These requirements consist of:

⁶TAC Chapter 6, p. 6-3.

EC (electronic combat) events	12
Instrument sorties (inexperienced pilots)	2
Night AAR	1
No HUD (head up display) approaches: one-half of Table 3-1 requirements	
Formation Takeoffs	4
CW (chemical warfare) Exercise	1 annually ⁷

TAC Chapter 6 specifies the following as goals:

Red Flag participation	1 annually
Formation events:	
Day Takeoff	12
Night Takeoff	2
Day Landing	3
Departure (wing)	6
Approach (wing)	6 ⁸

Table 6-12 covers another subject basic to design of a successful scheduling algorithm, currencies. The following list is excerpted from that table leaving out some of the complicating qualifiers that do not apply to the F-15 aircraft or pilots.

⁷TAC Chapter 6, Table 6-10, p. 6-39

⁸TAC Chapter 6, Table 6-11, p. 6-39

<u>Accomplishment (Event/Sortie)</u>	<u>Inexperienced Pilot</u>	<u>Experienced Pilot</u>
Day Landing	30 days	45 days
Night Landing	15	30
AAR (Day or Night)	S i x M o n t h s	
ACBT (Air Combat Training)	90	90
<u>Formation Events</u>		
Takeoff (Day or Night)	60	90
Day Landing	60	90
Low Level Flying	60	90
IP Rear Seat Landing		30
IP Instruction Flight		60
Dart	1 8 M o n t h s	

These requirements are constraints or considerations that must be taken into account by the scheduling algorithm being developed. These may be different than those imposed by another command and may vary further depending on the specific type aircraft.

USAFE Chapter 6 has similar type information but the currency numbers vary, different categories are defined, and some guidance is much more specific. Paragraph 6-24 requires inexperienced pilots to fly a non-demanding sortie if they have not flown within 22 to 30 days and requires the same of experienced pilots who have not flown for 31 to 45 days. The experience levels are defined in Volume I and the

non-demanding category is explained in paragraph 6-23 of USAFE Chapter 6.

Other currencies specified in the USAFE chapter include regaining landing currency after varying periods, night landing, air refueling, wing formation landings, precision approaches, rear seat landing for instructors, and flights while wearing CWD gear. Each of the type events and sorties required later in the chapter are defined in paragraph 6-25. For the F-15, Table A3-1 specifies the training requirements for maintaining the various levels of GCC qualification. Some are defined as guidelines under some conditions, but they are essentially required for purposes of the scheduling algorithm.

GCC Level Sorties

	Level:	A	B	C
Total (Inex/Exp)		40/36	68/60	82/70
1 month GCC rate		7/6	12/10	14/12
3 month GCC rate		20/18	34/30	41/35

Weapons Events (Required)

Dart		Qual	
Gun Tracking	6	12	18
WSEP ⁹		1 (sortie)	

⁹WSEP is Weapons System Evaluation Program.

GCC Events

Intercepts	20/16	26/22	32/28
ECCM ¹⁰	2	4	6
Alert Scramble	2	3	4
Integrated Msn/Joint Ex	1	2	3
Comm Jam	2	4	6
ACBT Sorties	31/27	43/37	50/42
BFM/DBFM ¹¹ Sorties	2	2	2
Instrument/Proficiency Sorties	2	4	4
AAR	3	3	3
Captive AIM-9	6	8	10
CWD Sorties	1	1	1
ACMI ¹² Sorties	4	6	8

F-15 AIRCREW TRAINING

Volume VII¹³ of the series is specific to the F-15 aircraft. Paragraph 2-9 lists minimum sorties and events to be accomplished during MQT, often a level of training the scheduler must be concerned with. Chapter 3 includes the

¹⁰ECCM is Electronic Counter Countermeasures.

¹¹BFM is Basic Fighter Maneuvers, DBFM is the same mission flown with dissimilar aircraft.

¹²ACMI is the Air Combat Maneuvering Instrumentation, a realistic training enhancement.

¹³DEPARTMENT OF THE AIR FORCE, Headquarters Tactical Air Command, Flying Training: F-15 AIRCREW TRAINING. TAC MANUAL 51-50, Volume VII, 26 March 1982.

minimum number of simulator hours required for each training period, among other items.

Summary

This annex has shown some of the sources and numbers that the scheduling algorithm must be capable of handling. Of more significance than the numbers is their variation depending on the situation. A given scheduler has essentially the same type problems as any other but the specifics of requirements vary widely depending on location, experience level of the pilots, weather affecting the base, and maintenance capability currently enjoyed. The algorithm must be able to take such diverse factors into account and simulate the many small decisions the human scheduler would normally make largely on intuition and produce a product--the schedule. Its success will lie, if it is successful, in making its programmed decisions without forgetting the details that sometimes escape the human scheduler in his flurry of work.

ANNEX B

DATA DEFINITION AND STORAGE

The previous annex showed the sources and types of data required by the algorithm to make programmed "decisions". The significant factors are the variety and variation of these data from one location to another. The user will have to be able to define and redefine data storage parameters as the system is used, both to initialize it and to react to changes in guidance or regulations. This annex will describe the way different data types may be assigned by the user and the general types of data the algorithm must be able to access and manipulate.

Data Types

A microcomputer actually stores only one representation of data--the byte. A byte is defined as eight bits, each of which can have the value "on" or "off". The context in which a given byte is presented to the microprocessor determines how it will be interpreted. Several general types are available in Microsoft BASIC¹, including string or character, integer, and single or double precision real variables. Characters are stored with one byte used for

¹Trade Mark of Microsoft Corp., Bellevue, WA.

each letter, digit of a number, or special code. This code is called ASCII, for American Standard Code for Information Interchange. Seven bits of each byte are used in this code which results in 2^7 or 128 possible meanings. The eighth bit may be left blank or it may be used for a parity check on the other seven bits. In some systems the eighth bit is used to define another 128 characters used for graphics. Integers are stored in two consecutive bytes and may have the value -32768 to 32767. The number 32767 is 2^{15} less one--the two bytes are interpreted as a binary number with the most significant, or sixteenth, bit used as a "sign" bit. Note the difference in representing the number 32767 in ASCII or as an integer: ASCII requires five bytes while the binary form requires only two.

Real numbers, those that can have fractional values, are stored in either four or eight bytes as single or double precision variables. Single precision can represent numbers to six significant figures while double represents sixteen significant figures. Since these numbers are stored in binary format, the fractional portion is subject to a very small error when converting to and from decimal².

²This is not a problem in most applications but must be considered if the result of a calculation is based on the difference of two numbers, especially if the result is at or near the limit of the number's precision. The most common example of this type difficulty is in interest calculations for accounting applications; daily interest numbers can be very small but are used in long iterations which compound a very small error into a significant one. Money calculations are required to balance to the penny. This fact must be

The flexibility of these data types will allow complete and compact storage of the data required by this model. Names of pilots, for instance, will be stored as a string of characters while currency dates will be stored as an integer or binary value. Storing a date as letter and number characters might seem insignificant at first glance, but seven bytes versus two becomes quite significant when storing many different dates for each of 40 or more pilots.

Defining What Is Stored

The hardware or machine and program language dependent data storage limitations will allow the application to store any type variable data that may be needed. The application program, or implementation of the scheduling algorithm, must store, access, and manipulate the data in a meaningful way. Since this will depend on many factors, including what command guidelines and regulations affect the unit, how many pilots are assigned, how many different missions must be considered, and so on, a means of storing not only the data but the meaning of the data must be devised.

considered in the design of the algorithm, so that its effects are not significant. Number precision data is from the OSBORNE 1 User's Reference Guide by Thom Hogan and Mike Iannamico, Hayward, CA: Osborne Computer Corp., 1981, revised 2/22/82.

Data Mass Storage Media

The storage medium available for this program is called a disk; in this case either floppy or hard disk. The difference in these is mainly one of capacity and access/transfer speed, the type files that may be stored are essentially identical. A disk is a random access medium for mass storage, that is, it can be accessed directly throughout its capacity. A sequential access device, on the other hand, must read everything up to the position of the required data in order to find that data.

An example of sequential access is the cassette tape. It must be played until the desired selection is reached; it must be rewound to find specific data again. Use of the tape counter makes fast forwarding to the vicinity of a selection possible, but finding one note or word of a particular song would be difficult without listening to a complete passage.

The random access disk has the data stored on it in rings or tracks. There are many tracks so the amount of data on each is a small portion of the total. Even though the data is stored sequentially on each track, it can be found very quickly by reading the whole track or a sector of the track. Thus the disk is a good medium to have for storing the data required by this project.

Data is stored on the disk in files. Each file may include many records, each of which stores a unit of the

file. This can be visualized as each of the sheets containing responses to a questionnaire. Each record is then divided up into fields, or continuing the analogy, the responses to each question on the questionnaire. For this project, a file could contain records for each pilot showing his name, Social Security Account Number (SSAN), birth month, training status code, and qualifications. These divisions by pilot would be the fields, the complete data on a given pilot would be a record, and all the data on the pilots of the squadron would be a file.

Disk files themselves may be either random access or sequential access files. Sequential access files may be found directly by the storage medium, the disk, but must be read sequentially. Random access files may be accessed by individual record directly. The advantage of sequential files is their conservation of storage space--very little overhead is used in storing the information. Random access files require each record to be a consistent length, so a specific record position can be calculated. This means that if the longest name in the squadron has twenty five letters in it, even the shortest name will also effectively take up the same twenty five bytes of storage. Perhaps more significant is the new pilot whose name will not fit into the existing name field--not the best situation.

The point of this discussion is that data storage must be considered carefully so that changes can be accommo-

dated. Speed of access and active storage space within the computer must weigh against disk space available. Most significant is the ability to change the mode of storage as requirements change. This suggests the use of a file to store the meaning of the contents of another file. Allowing the user to define what, where and how the data he/she needs will be stored will make for maximum flexibility in application to varying locations, guidance and regulations.

Specific Data Types

The algorithm being developed deals with pilot's personal data, qualifications, currencies, requirements, accomplishments, and availability. Time periods may vary from three years, the longest currency period presently needed, to a few minutes. Dates may be needed in terms of days or months, or may refer to times years away. Effective manipulation of data in these forms will require a few standardizing decisions up front.

Individual name and personal data will mostly be string or individual character type. If internal data manipulation is accomplished with subscripted or array variables, then this data may most easily fit into a sequential file. Initialization of run time variables to the portions of personal data needed would be quick and accomplished only once. Prompts and other user interface messages could insert the name while manipulating the data in an array. Data such as currencies could be maintained in a list by pilot or

by the currency requirement, depending on the use being made of the data.

The basic concept of the scheduling model is that a relationship of priority exists between the number of opportunities available to accomplish a requirement and the number of those requirements that remain. Implementing this concept requires subtracting the individual pilot accomplishments from the total required for the given item. By eliminating periods that are known to be unavailable, a quotient representing the relative priority of pilots for a specific training asset can be established. If only those current and available are considered, then the pilot with the highest priority is the one assigned to use that item.

This concept will require storage of many data items at once. If the requirement is to fill a flight lead ACBT slot, for instance, the algorithm must check all pilots for flight lead status, ACBT currency, and availability during the time period of the flight. Then, assuming more than one pilot is available who fills these criteria, each pilot's priority for the ACBT flight must be calculated and compared. When the highest priority is determined, given all factors and weights to consider, that individual must then be made "not available" for the duration of the flight and the briefing and debriefing times associated with it. With that pilot's data updated for the potential flight completion and the flight itself filled, the next priority of

requirement must be examined in the same way.

Thus it becomes obvious that implementation of this algorithm requires storage of and access to availability, qualification, and currency data. A schedule is normally built on a weekly basis with names, but tentative plans may be made over longer periods. The availability data must be stored in a format allowing any degree of precision required by the situation. A month or more in advance, the scheduler may be looking at half-day time increments; he will be looking at parts of hours, perhaps minutes, when making a final daily schedule.

The concept of a file defining the use of a file makes it possible to store the standard data in a given application very compactly. The range of integer numbers allows currency data to be stored as the units digit of the year times 1000 plus the Julian date. For example, 30 January 1983 would be $3 \times 1000 + 30$ or 3030. Availability usually requires two times to define it, the beginning and the duration or end time. Since a training period is six months long, the day of the period times 100 plus the half-hour of the day would fit into the integer number range available³. This limits the resolution of the system to the half-hour block that includes the start or end time, but that may be sufficient for most long range factors.

³A maximum of 184 days times 100 equals 18400, plus 24 hours in a day times 2 equals a maximum value well within the limits of integer values.

A date and time block providing one minute resolution would need four digits for the time of day and four more for the day and year. If the duration were limited in some way, it could reduce the storage space required for the end data, but using the same format reduces the complexity of coding and decoding without limiting flexibility. For instance, the system could accommodate both a 30 minute duration haircut appointment and a 179 day temporary duty (TDY) assignment without modification.

One factor to consider in currency data storage is the form of the requirement, currency dates, and the method for their comparison. At machine level, the easiest comparison is a logical AND or a subtraction. Since this is done in binary form, if the data were stored in binary also, it could have a beneficial effect on speed of operation. This type of data storage and comparison technique will be used for the availability checking routine. Numbers up to 255 in decimal, or 2^8 less one, can be compared directly this way with an eight bit microprocessor.

Summary

The data storage is driven by several factors, the machine and language capabilities, mass storage characteristics, and the nature of the information to be stored. The variations from one user environment to another will require significant user input into what information is stored and how. The general types of information will be character

strings, numbers, and dates, and their form will be the smallest that can be used consistent with the range and resolution required.

ANNEX C

PROGRAM LISTINGS

The programs listed on the following pages show the ability of a microcomputer to handle the magnitude and detail of the scheduling problem at squadron level. Several statements are included which "stub" certain routines; these were not required to demonstrate the algorithm and were not completed due to time constraints.

These programs were written on an Osborne 1 with the software included in the purchase price of that system. The listing was done on an IDS Prism 132 printer in the 10 character per inch correspondence font mode.

Any reader with intent to apply these programs to an actual scheduling job is encouraged to contact the author for a copy of the latest version on disk.

```

100  '*** CURDEF.SET *****
110  'program dated 16 May 1983
120  '
130  'This program sets or changes the values stored for
140  '     each currency code
150  '
160  DEFINT A-Z
170  CLR$ = CHR$(26): DOWN$ = CHR$(10)
180  MID$ = CLR$ + STRING$(8,10)
190  DIM EVENT$(10)
200  '
210  OPEN "R", 1, "CUR.DEF", 28
220  FIELD#1, 2 AS N1$, 20 AS N12$, 2 AS N13$, 2 AS N14$,
230  ' 2 AS N15$
240  PRINT MID$.SCRN$ "Enter currency code number to change
250  ' or 0 to quit"
260  PRINT: PRINT "What number?"; INPUT " ", CODE
270  IF CODE = 0 THEN CLOSE: END ELSE IF CODE > 15 THEN PRINT
280  "Error, out of range (max is 15)": GOTO 250
290  GET#1, CODE
300  CUR.CODE = CVI(N1$)
310  CUR.NAME$ = N12$
320  PER.EX = CVI(N13$)
330  PER.INX = CVI(N14$)
340  EVNT = CVI(N15$)
350  IF CUR.CODE <> CODE THEN PRINT "File error: record number not
360  ' equal to currency code": PRINT "Press any key to continue...":
370  ' DUMMY$ = INPUT$(1)
380  PRINT MID$.SCRN$ "Current data:"
390  PRINT CUR.CODE CUR.NAME$ PER.EX PER.INX EVNT
400  PRINT
410  PRINT "Enter:"
420  PRINT "      0 if all correct, no changes"
430  PRINT "      1 to change currency name"
440  PRINT "      2 to change experienced currency period"
450  PRINT "      3 to change inexperienced period"
460  PRINT "      4 to change updating event number"
470  PRINT "Which choice?"; A = VAL(INPUT$(1)): PRINT A
480  IF A = 0 THEN LSET N1$ = MKI$(CODE): PUT#1, CODE: GOTO 240
490  ELSE IF A > 4 THEN PRINT "Error, enter a number from 0 to 4 only,
500  ' try again. ": GOTO 430
510  ON A COSUB 500, 580, 640, 700
520  GOTO 340
530  '
540  '_____subroutines_____
550  '
560  PRINT MID$.SCRN$ CUR.CODE CUR.NAME$ PER.EX PER.INX EVNT
570  PRINT: PRINT "What is the new currency name?"; INPUT CUR.NAME$
580  IF LEN(CUR.NAME$) > 20 THEN PRINT "Too long, only 20 characters
590  ' will be saved"
600  PRINT "Enter 0 if currency name is correct, 1 to change"

```

```

de it: " : A = VAL(INPUT$(1)): PRINT A
540     IF A = 1 THEN GOTO 510 ELSE IF A <> 0 THEN PRINT"Err
or, 0 or 1 only, try again...": GOTO 530
550     LSET N12$ = CUR.NAME$
560     RETURN
570     '
580     PRINT MID$.SCRN$: CUR.CODE CUR.NAME$: PER.EX PER.INX EV
NT
590     PRINT: PRINT"Enter the period of currency for experi
enced pilots"
600     INPUT"What is the currency period (days)? ", PER.EX
610     LSET N13$ = MKI$(PER.EX)
620     RETURN
630     '
640     PRINT MID$.SCRN$: CUR.CODE CUR.NAME$: PER.EX PER.INX EV
NT
650     PRINT: PRINT"Enter the period of currency for inexpe
rienced pilots"
660     INPUT"What is the currency period (days)? ", PER.INX
670     LSET N14$ = MKI$(PER.INX)
680     RETURN
690     '
700     PRINT MID$.SCRN$: CUR.CODE CUR.NAME$: PER.EX PER.INX EV
NT
710     PRINT: PRINT"Enter the event number that updates thi
s currency (? for help) : "
720     INPUT"What is the event number? ", EVNT$
730     IF EVNT$ = "?" THEN GOSUB 790
740     EVNT = VAL(EVNT$)
750     IF EVNT < 1 OR EVNT > 10 THEN PRINT"Enter a number f
rom 1 to 10 only...": GOSUB 790: GOTO 740
760     LSET N15$ = MKI$(EVNT)
770     RETURN
780     '
790     OPEN "R", 2, "CUREVNT.DEF", 26
800     FIELD#2, 2 AS N21$, 20 AS N22$
810     I = 0
820     FOR I = 1 TO 10
830         GET#2, I
840         EVENT$(I) = N22$
850     NEXT
860     FOR I = 1 TO 10
870         PRINT I "- " EVENT$(I)
880     NEXT
890     INPUT"Which event number? ", EVNT$
900     RETURN

```

```

100  '*** ACTDEF SET *****
110  'program dated 16 May 1983
120
130  'This program sets or changes the values stored for
140  '    each activity code
150
160  DEFINT A-Z
170  CLR$ = CHR$(26): DOWN$ = CHR$(10): MID$.SCRN$ = CLR$:
  FOR I = 1 TO 8: MID$.SCRN$ = MID$.SCRN$ + DOWN$: NEXT
180  DIM ST.T(3), END.T(3), GP$(3), CUR.CAT$(15), QUAL.CA
T$(15)
190  ST.T(0) = 0: ST.T(1) = 15: ST.T(2) = 135: ST.T(3) =
165
200  END.T(0) = 0: END.T(1) = 90: END.T(2) = 180: END.T(3
) = 240
210  GP$(0) = "Non-duty activities": GP$(1) = "Duty/non-f
lying activities"
220  GP$(2) = "Flying activities": GP$(3) = "Long flight
activities"
230
240  OPEN "R", 1, "ACT DEF", 46
250  FIELD#1, 2 AS N1$, 20 AS N12$, 2 AS N13$, 2 AS N14$
, 10 AS N15$, 10 AS N16$
260
270  PRINT MID$.SCRN$: "Enter activity code number to chang
e or 0 to quit"
280  PRINT: PRINT "What number?";: INPUT " ", CODE
290  IF CODE = 0 THEN CLOSE: END ELSE IF CODE > 255 THEN
PRINT "Error, out of range (max is 255)": GOTO 280
300  GET#1, CODE
310  ACT.CODE = CVI(N1$)
320  ACT.NAME$ = N12$
330  ST.T = CVI(N13$)
340  END.T = CVI(N14$)
350  CUR$ = N15$
360  QUAL$ = N16$
370  IF ACT.CODE (<) CODE THEN PRINT "File error record nu
mber not equal to activity code": PRINT "Press any key to con
tinue...":
DUMMYS = INPUT$(1)
380  PRINT MID$.SCRN$: "Current data:"
390  PRINT ACT.CODE ACT.NAME$ ST.T END.T
400  PRINT
410  PRINT "Enter:"
420  PRINT "          0 if all correct, no changes"
430  PRINT "          1 to change activity name"
440  PRINT "          2 to change start or end time offset
s"
450  PRINT "          3 to check currency requirements"
460  PRINT "          4 to check qualification requirement
s"
470  PRINT "Which choice?";: A = VAL(INPUT$(1)): PRINT A
480  IF A = 0 THEN LSET N1$ = MKI$(CODE): PUT#1, CODE: G
OTO 270 ELSE IF A > 4 THEN PRINT "Error, enter a number from
0 to 4 only,"

```

```

try again...": GOTO 470
490     ON A GOSUB 540, 620, 790, 1030
500     GOTO 380
510
520     _____subroutines_____
530
540     PRINT MID$SCRNS: ACT NAMES:
550     PRINT "What is the new activity name?": INPUT ACT NAM
Es
560     IF LEN(ACT NAMES) > 20 THEN PRINT "Too long, only 20
characters will be saved"
570     PRINT "Enter 0 if activity name is correct, 1 to chan
ge it: ": A = VAL(INPUT$(1)): PRINT A
580     IF A = 1 THEN GOTO 460 ELSE IF A <> 0 THEN PRINT "Err
or, 0 or 1 only, try again...": GOTO 490
590     LSET N12$ = ACT NAMES:
600     RETURN
610
620     PRINT MID$SCRNS "Start offset is the time before the
activity that availability is required"
630     PRINT "End offset is the time for the activity and de
brief or travel time following"
640     J = ACT CODE / 64
650     PRINT "This activity code group includes " GP$(J) ":"
660     PRINT ST.T(J) "is the standard number of minutes set
for start offset"
670     PRINT END.T(J) "is the standard end offset"
680     PRINT "Enter: "
690     PRINT " 0 if the old offsets are correct"
700     PRINT " 1 to change to the standard offsets"
710     PRINT " 2 to enter different offsets"
720     PRINT "Which choice? ": A = VAL(INPUT$(1)): PRINT A
730     IF A = 0 THEN GOTO 760 ELSE IF A > 2 THEN PRINT "Erro
r, 0, 1, or 2 only, try again...": GOTO 720
740     IF A = 1 THEN ST.T = ST.T(J): END.T = END.T(J)
750     IF A = 2 THEN INPUT "Start offset (minutes): ", ST.T:
INPUT "End offset (minutes): ", END.T
760     LSET N13$ = MKI$(ST.T): LSET N14$ = MKI$(END.T)
770     RETURN
780
790     OPEN "R", 2, "CUR.DEF", 28
800     FIELD#2, 2 AS N21$, 20 AS N22$, 2 AS N23$, 2 AS N24$
, 2 AS N25$
810     I = 0
820     FOR I = 1 TO 15
830         GET#2, I
840         CUR.CAT$(I) = N22$
850     NEXT
860     PRINT MID$SCRNS "Up to five combinations of currenci
es are allowed for each activity"
870     PRINT "For each currency category enter 1 if it appli
es, 0 if it does not..."
880     CUR(1) = 0: J = 1
890     FOR I = 1 TO 15
900         PRINT CUR.CAT$(I) "?: ": BIT = VAL(INPUT$(1))

```

```

100 PRINT BIT
110 IF BIT THEN CUR(J) = CUR(J) + 2*(I-1)
120 NEXT
130 PRINT "This set complete, enter 0 if done, 1 to enter
another set"; A = VAL(INPUT$(1)); PRINT A
140 IF A = 1 THEN J = J + 1; IF J > 5 THEN PRINT "No more
room for currency sets"; J = 5 ELSE CUR(J) = 0 GOTO 890
150 IF A <> 0 THEN PRINT "Error, enter 0 or 1 only, try a
gain..."; GOTO 930
160 CURR$ = ""
170 FOR I = 1 TO J
180 CURR$ = CURR$ + MKIS(CUR(I))
190 NEXT
2000 LSET N15$ = CURR$
2010 CLOSE#2: RETURN
2020 '
2030 OPEN "R", 2, "QUAL DEF", 22
2040 FIELD#2, 2 AS N21$, 20 AS N22$
2050 I = 0
2060 FOR I = 1 TO 15
2070 GET#2, I
2080 QUAL.CAT$(I) = N22$
2090 NEXT
2100 PRINT MID$SCRN$ "Up to five combinations of qualific
ations are allowed for each activity"
2110 PRINT "For each qualification category enter 1 if it
applies, 0 if it does not..."
2120 QUAL(1) = 0; J = 1
2130 FOR I = 1 TO 15
2140 PRINT QUAL.CAT$(I) " "; BIT = VAL(INPUT$(1))
2150 PRINT BIT
2160 IF BIT THEN QUAL(J) = QUAL(J) + 2*(I-1)
2170 NEXT
2180 PRINT "This set complete, enter 0 if done, 1 to enter
another set"; A = VAL(INPUT$(1)); PRINT A
2190 IF A = 1 THEN J = J + 1; IF J > 5 THEN PRINT "No more
room for qualification sets"; J = 5 ELSE QUAL(J) = 0; GOTO
2130
2200 IF A <> 0 THEN PRINT "Error, enter 0 or 1 only, try a
gain..."; GOTO 1170
2200 QUAL$ = ""
2210 FOR I = 1 TO J
2220 QUAL$ = QUAL$ + MKIS(QUAL(I))
2230 NEXT
2240 LSET N16$ = QUAL$
2250 CLOSE#2: RETURN
2260 '

```

```

100  '*** UPDATE *****
110  'program dated 21 May 1983
120  '
130  'This program allows entry of availability data for
140  '    pilots
150  '
160  'variables required:
170  '    none
180  '
190  'returns:
200  '    PILnn.DAT files updated and in order
210  '
220  DEFINT A-Z
230  CLR$ = CHR$(26): DOWN$ = CHR$(10): ESC$ = CHR$(27)
240  MID$ = CLR$ + STRING$(6,10)
250  HOME$ = CHR$(30): CLR.LINE$ = ESC$ + "T"
260  C$ = "Enter: 0 if correct, 1 to change it: "
270  E$ = "Error, enter 0 or 1 only, try again..."
280  '
290  MAX.PIL.NUM = 60
300  IF P$(0,0) <> CHR$(255) THEN ERASE P$: DIM P$(MAX.PI
L.NUM,4)
310  IF QUAL$(0) <> CHR$(255) THEN ERASE QUAL$: DIM QUAL$
(15)
320  '
330  'open key file...
340  GOSUB 4080
350  'read in all names...
360  GOSUB 4010
370  CLOSE
380  '
390  'open qual.def file...
400  GOSUB 4120
410  FOR I = 1 TO 15: GET#2, I: QUAL$(I) = N22$: NEXT: CL
OSE
420  '
430  'open curevnt.def file...
440  GOSUB 4160
450  FOR I = 1 TO 10: GET#2, I: TRIM$ = N22$: GOSUB 2490.
EVENT$(I) = TRIM$: NEXT: CLOSE
460  '
470  PRINT MID$.SCRNS "      Enter:"
480  PRINT"      0 to quit, all done"
490  PRINT"      1 to add a new pilot data file"
500  PRINT"      2 to change data in existing data fi
le"
510  PRINT"      3 to delete a pilot data file"
520  PRINT" Which choice? ": SEL = VAL(INPUT$(1)): PRIN
T SEL
530  IF SEL <= 0 THEN END ELSE IF SEL > 3 THEN PRINT"Erro
r, enter a number 0 to 3 only, try again...": GOTO 520
540  ON SEL GOSUB 600, 1340, 1360
550  GOTO 470
560  '
570  '____add_pilot_data_____

```

```

580      '
590      'print names to screen...
600      GOSUB 3890
610      '
620      INPUT "Which pilot number do you want to use? ", NUM
630      IF NUM = 0 GOTO 1280
640      NUM$ = STR$(NUM)
650      NUM$ = MID$(NUM$, 2)
660      '
670      'check if file already exists...
680      ON ERROR GOTO 700
690      FILENAME$ = "PIL" + NUM$ + ".DAT": OPEN "I", 1, FILE
NAME$
700      IF ERR = 53 THEN RESUME 750
710      PRINT FILENAME$ " exists on this disk, confirm you w
ant to overwrite (destroy) it"
720      PRINT: PRINT "Enter 0 to continue, 1 to NOT overwrite
this file: "; A$ = INPUT$(1): PRINT A$
730      IF A$ = "1" GOTO 420
740      IF A$ <> "0" THEN PRINT E$: GOTO 720
750      ON ERROR GOTO 0
760      CLOSE
770      '
780      'name data entered at subroutines...
790      FOR I = 1 TO 3: ON I GOSUB 4270, 4360, 4400: NEXT
800      '
810      'check admin data, update key file...
820      GOSUB 2380
830      '
840      'now put in individual file...
850      OPEN "O", 1, FILENAME$
860      WRITE#1, NUM$
870      WRITE#1, L NAME$, F NAME$, MI$
880      WRITE#1, RANK$
890      WRITE#1, SSAN$
900      '
910      'ask and save qualifications at once...
920      WRITE#1, "QUALIFICATIONS."
930      FOR I = 1 TO 15
940      PRINT MID$ SCRN$
950      PRINT "Enter a 1 digit if the qualification applies,
0 if it does not"
960      PRINT: PRINT QUAL$(I) "? 0 or 1: "; A = VAL(INPUT$(
1)): IF A < 0 OR A > 1 THEN PRINT A E$: GOTO 960
970      WRITE#1, A
980      NEXT
990      CLOSE #2
1000     '
1010     'same for currency dates...
1020     WRITE#1, "CURRENCIES:"
1030     'open curr event name file as #2...
1040     GOSUB 4160
1050     FOR I = 1 TO 10
1060     PRINT MID$ SCRN$
1070     PRINT "Enter the date " EVENT$(I) " last accomplished

```

```

or 0 for none: "
1080   GOSUB 5630
1090   WRITE#1, DATE
1100   NEXT
1110   CLOSE #2
1120   '
1130   'open activity definition file.
1140   GOSUB 4200
1150   A = 1: N = 0: MAX.N = 10
1160   GOSUB 1920
1170   'save max number activities...
1180   MAX.N = N
1190   '
1200   'entries complete, sort them...
1210   GOSUB 4890
1220   'then check for conflicts...
1230   GOSUB 5070
1240   'print to file...
1250   WRITE#1, "ACTIVITIES SCHEDULED:", MAX.N
1260   FOR I = 1 TO MAX.N: PRINT#1, ACT$(I): NEXT
1270   CLOSE
1280   RETURN
1290   '
1300   '
1310   '_____change_or_add_to_existing_pilot_data_files_____
1320   '
1330   'get and check pilot number...
1340   GOSUB 1730
1350   '
1360   'read file into memory...
1370   GOSUB 3550
1380   PRINT MID$.SCRNS: CVI(P$(NUM,1)) " - " P$(NUM,2) P$(NUM,3) " " P$(NUM,4)
1390   PRINT: PRINT"      Enter:"
1400   PRINT"      0 if no more changes or additions, a
ll done"
1410   PRINT"      1 to change admin data (name, rank,
SSAN)"
1420   PRINT"      2 to change qualification data"
1430   PRINT"      3 to update currency data"
1440   PRINT"      4 to add, change or delete availabil
ity data"
1450   PRINT" Which choice? ";; A = VAL(INPUT$(1)): PRINT
A
1460   '
1470   IF A = 0 THEN GOSUB 2900: RETURN
1480   IF A > 4 THEN PRINT"Error, enter 0 to 4 only, try ag
ain..." GOTO 1450
1490   ON A GOSUB 2580, 3130, 3230, 3330
1500   GOTO 1380
1510   '
1520   '
1530   '_____delete_complete_pilot_data_file_____
1540   '
1550   'get and check pilot number...

```

```

1560   COSUB 1730
1570   PRINT"Enter 0 to delete this file, 1 to abort delete
      action"
1580   INPUT"Which one"; D
1590   IF D <> 0 THEN PRINT"Exiting delete mode, file NOT d
      elated..."; FOR I = 1 TO 1000: NEXT: GOTO 1680
1600   OPEN "O", 3, TMP.FIL$: CLOSE#3: COSUB 3030: KILL FIL
      ENAME$
1610   'reset key file...
1620   COSUB 4080
1630   LSET N1$ = MKI$(NUM): LSET N2$ = "Not in use": LSET
      N3$ = " " : LSET N4$ = " "
1640   PUT#1, NUM
1650   CLOSE#1
1660   'reset memory variables...
1670   P$(NUM,2) = "Not in use": P$(NUM,3) = " " : P$(NUM,4
      ) = " "
1680   RETURN
1690   '
1700   '_____subroutines_____
1710   '
1720   'get and confirm pilot number...
1730   PRINT MID$SCRN$
1740   PRINT"Enter the last name or pilot number:"; INPUT"
      ", ANSWER$
1750   IF ASC(LEFT$(ANSWER$,1)) < 58 THEN NUM = VAL(ANSWER$
      ): THIS.NUM = -1 ELSE NUM = 0: L.NAME$ = ANSWER$
1760   'look for name match...
1770   WHILE NUM < MAX.PIL.NUM AND NOT THIS.NUM
1780       NUM = NUM + 1
1790       IF L.NAME$ = LEFT$(P$(NUM,2),LEN(L.NAME$)) T
      HEN THIS.NUM = -1 ELSE THIS.NUM = 0
1800   WEND
1810   IF THIS.NUM = 0 THEN COSUB 3890: PRINT: PRINT"Enter
      pilot number: "; INPUT NUM
1820   IF NUM = 0 THEN GOTO 1890
1830   PRINT MID$SCRN$ CVI(P$(NUM,1)) " - " P$(NUM,2) P$(NU
      M,3) " " P$(NUM,4)
1840   PRINT"Enter 0 if this the correct entry; 1 if not co
      rrect:"; THIS.NUM = VAL(INPUT$(1)): PRINT THIS.NUM
1850   IF THIS.NUM = 1 THEN THIS.NUM = 0: GOTO 1810
1860   IF THIS.NUM <> 0 THEN PRINT"Error, enter 0 or 1 only
      ...": GOTO 1830
1870   'have correct number, get filenames...
1880   COSUB 3810
1890   RETURN
1900   '
1910   'input a new activity...
1920   WHILE A
1930       N = N + 1
1940       PRINT"Enter activity code (? for help) ";
1950       INPUT" ", CODE$
1960       IF CODE$ = "?" THEN COSUB 4450 ELSE IF CODE$
      = "0" THEN GOTO 2400
1970       ACT.CODE = VAL(CODE$)

```

```

1980      IF ACT.CODE < 1 OR ACT.CODE > 255 THEN PRINT
"Entry is out of range...": GOSUB 4450: GOTO 1970
1990      IF (ACT.CODE AND 63) = 63 THEN OTHER = -1 EL
SE OTHER = 0
2000      IF OTHER THEN INPUT "What is the activity nam
e? ", ACT.NAME$: GOTO 2060
2010      GET#2, ACT.CODE
2020      TRIM$ = N22$
2030      GOSUB 2490
2040      ACT.NAME$ = TRIM$
2050
2060      PRINT MID.SCRN$
2070      PRINT "Enter the date " ACT.NAME$ " starts or
occurs on:"
2080      GOSUB 5630
2090      ACT.DATE = DATE: END.DATE = 0
2100      PRINT MID.SCRN$
2110      PRINT "Enter the scheduled time: ";
2120      INPUT TIME$: GOSUB 5190
2130      IF OTHER THEN GOSUB 4690: GOTO 2230 ELSE IF
ACT.CODE = 62 OR ACT.CODE = 61 THEN GOSUB 4820: GOTO 2230
2140      'not other and not 61 or 62...
2150      PRINT MID.SCRN$
2160      PRINT "Enter:"
2170      PRINT "          0 if standard time offsets a
pply"
2180      PRINT "          1 to change them"
2190      PRINT "Which choice? "; A$ = INPUT$(1): PR
INT A$.
2200      IF A$ = "1" THEN GOSUB 4690: GOTO 2230
2210      IF A$ <> "0" THEN PRINT E$: GOTO 2160
2220      START = CVI(N23$): END.T = CVI(N24$)
2230      ACT.ST.TIME = TIME - START: ACT.END.TIME = T
IME + END.T
2240      IF END.DATE = 0 THEN END.DATE = ACT.DATE
2250      ACT.LN$ = STRING$(25,32)
2260      MID$(ACT.LN$,1,5) = STR$(ACT.CODE)
2270      MID$(ACT.LN$,6,5) = STR$(ACT.DATE)
2280      MID$(ACT.LN$,11,5) = STR$(ACT.ST.TIME)
2290      MID$(ACT.LN$,16,5) = STR$(END.DATE)
2300      MID$(ACT.LN$,21,5) = STR$(ACT.END.TIME)
2310      ACT.LN$ = ACT.LN$ + ACT.NAME$
2320      IF N >= MAX.N THEN GOSUB 3740
2330      ACT$(N) = ACT.LN$
2340      PRINT MID.SCRN$ "          Check the activity d
ata."
2350      PRINT N "          " ACT$(N)
2360      PRINT C$, " A = VAL(INPUT$(1)): PRINT A
2370      IF A = 1 THEN GOTO 1940
2380      IF A <> 0 THEN PRINT E$: GOTO 2360
2390
2400      PRINT "Enter:"
2410      PRINT "  0 if entries complete"
2420      PRINT "  1 if more activities to enter"
2430      PRINT "Which one? "; A = VAL(INPUT$(1)): PRI

```

```

NT A
2440             IF A < 0 OR A > 1 THEN PRINT E$; GOTO 2430
2450     WEND: 'activity entry loop.
2460 RETURN
2470
2480
2490     L = LEN(TRIM$) + 1: L.CHRS = CHR$(0)
2500     WHILE ASC(L.CHRS) < 33
2510         L = L - 1
2520         L.CHRS = MID$(TRIM$,L,1)
2530     WEND
2540     TRIM$ = LEFT$(TRIM$,L)
2550 RETURN
2560
2570     'change admin data...
2580     PRINT MID.SCRN$
2590     PRINT "Check the pilot data:"
2600     PRINT: PRINT "Pilot number assigned: "; NUM$
2610     PRINT: PRINT L.NAME$, " ", F.NAME$, " ", MI$, " ", RANK$
        " " SSAN$
2620     PRINT: PRINT
2630     PRINT "    Change which entry:"
2640     PRINT "            0 - no more changes, all correct"
2650     PRINT "            1 - name"
2660     PRINT "            2 - rank/grade"
2670     PRINT "            3 - SSAN"
2680     PRINT "            4 - change all entries"
2690     PRINT "    Which one? "; A$ = INPUT$(1): PRINT A$
2700     IF A$ = "0" THEN GOTO 2760
2710     IF VAL(A$) > 4 THEN PRINT "Error, enter 0 to 4 only,
        try again...": GOTO 2690
2720     ON VAL(A$) GOSUB 4270, 4360, 4400, 4250
2730
2740     'entries are correct, put in key file...
2750     'open key file...
2760     GOSUB 4080
2770     'and save key data...
2780     LSET N1$ = MKI$(NUM)
2790     LSET N2$ = L.NAME$
2800     INIT$ = LEFT$(F.NAME$,1) + LEFT$(MI$,1)
2810     LSET N3$ = INIT$
2820     LSET N4$ = RANK$
2830     PUT#1, NUM
2840     CLOSE#1
2850     'put in memory array...
2860     P$(NUM,1) = MKI$(NUM): P$(NUM,2) = L.NAME$ + STRING$(
        20-LEN(L.NAME$),32): P$(NUM,3) = INIT$: P$(NUM,4) = RANK$
2870 RETURN
2880
2890     'put all data in individual file (called from many r
        outines)...
2900     OPEN "O": 1, TMP.FILE$
2910     WRITE#1, NUM$
2920     WRITE#1, L.NAME$, F.NAME$, MI$
2930     WRITE#1, RANK$

```

```

2940 WRITE#1, SSAN$
2950 WRITE#1, QUAL.ID$
2960 FOR I = 1 TO 15: WRITE#1, QV(I): NEXT
2970 WRITE#1, CUR.ID$
2980 FOR I = 1 TO 10: WRITE#1, CUR.DT(I): NEXT
2990 WRITE#1, ACT.ID$, MAX.N
3000 FOR I = 1 TO MAX.N: PRINT#1, ACT$(I): NEXT
3010 CLOSE
3020 'and rename files...
3030 ON ERROR GOTO 3090
3040 KILL BAK.FIL$
3050 ON ERROR GOTO 0
3060 NAME FILENAME$ AS BAK.FIL$
3070 NAME TMP.FIL$ AS FILENAME$
3080 GOTO 3100
3090 IF ERR = 53 THEN RESUME 3050 ELSE GOTO 3050
3100 RETURN
3110 '
3120 'change qual data...
3130 PRINT MID$.SCRN$ CVI(P$(NUM,1)) " - " P$(NUM,2) P$(NUM,3) " " P$(NUM,4)
3140 FOR I = 1 TO 15: PRINT I TAB(6) QUAL$(I) QV(I): NEXT
3150 '
3160 PRINT: PRINT"Enter 0 if all correct or qual number to change"
3170 PRINT"Which number <0-15>?"; INPUT QN
3180 IF QN = 0 THEN RETURN ELSE IF QN > 15 THEN PRINT"Error, a number from 0 to 15 only, try again...": GOTO 3170
3190 IF QV(QN) = 1 THEN QV(QN) = 0 ELSE QV(QN) = 1
3200 GOTO 3140
3210 '
3220 'update currency dates...
3230 PRINT MID$.SCRN$ CVI(P$(NUM,1)) " - " P$(NUM,2) P$(NUM,3) " " P$(NUM,4)
3240 FOR I = 1 TO 10: PRINT I TAB(6) EVENT$(I) TAB(28) CUR.DT(I): NEXT
3250 PRINT: PRINT"Enter 0 if correct or item number to change"
3260 PRINT"Which number <0-10>"; INPUT CN
3270 IF CN <= 0 THEN RETURN ELSE IF CN > 10 THEN PRINT"Error, enter a number from 0 to 10 only, try again...": GOTO 3260
3280 PRINT"Enter the new currency date (accomplished date)": GOSUB 5630
3290 CUR.DT(CN) = DATE: GOTO 3230
3300 'returns on zero entry above...
3310 '
3320 'add, change or delete availability data...
3330 PRINT MID$.SCRN$ CVI(P$(NUM,1)) " - " P$(NUM,2) P$(NUM,3) " " P$(NUM,4)
3340 PRINT" Enter:"
3350 PRINT" 0 if activity changes completed"
3360 PRINT" 1 to add new activities"
3370 PRINT" 2 to change existing activities"
3380 PRINT" 3 to delete activities"

```

```

3390 PRINT " Which choice? "; A = VAL(INPUT$(1)); PRINT
A
3400 IF A = 0 THEN RETURN ELSE IF A > 3 THEN PRINT "Error,
enter a number from 0 to 3 only, try again."; GOTO 3390
3410 ON A GOSUB 3460, 3510, 3520
3420 GOTO 3330
3430 'return selected with zero response above...
3440 '
3450 'open def file, get activity entries...
3460 GOSUB 4200: PRINT MID$.SCRN$; N = MAX.N: A = 1: GOSU
B 1920: CLOSE#2: MAX.N = N
3470 'sort and check for conflicts...
3480 GOSUB 4890: GOSUB 5070
3490 RETURN
3500 '
3510 PRINT "Change not written yet...": DUMMY$ = INPUT$(1)
: RETURN
3520 PRINT "Delete not written yet...": DUMMY$ = INPUT$(1)
: RETURN
3530 '
3540 'open data file and read into memory, close...
3550 OPEN "I", 2, FILENAME$
3560 INPUT #2, NUM$, L.NAME$, F.NAME$, MI$, RANK$, SSAN$
3570 IF EOF(2) THEN GOTO 3700 ELSE INPUT#2, QUAL.ID$
3580 IF QUAL.ID$ <> "QUALIFICATIONS:" THEN PRINT "Qual dat
a not found":
3590 IF QV(0) <> -1 THEN ERASE QV: DIM QV(15)
3600 FOR I = 1 TO 15: IF EOF(2) THEN GOTO 3700 ELSE INPUT
#2, QV(I): NEXT
3610 IF EOF(2) THEN GOTO 3700 ELSE INPUT#2, CUR.ID$
3620 IF CUR.ID$ <> "CURRENCIES:" THEN PRINT "Cur data not
found":
3630 FOR I = 1 TO 10: IF EOF(2) THEN GOTO 3700 ELSE INPUT
#2, CUR.DT(I): NEXT
3640 IF EOF(2) THEN GOTO 3700 ELSE INPUT#2, ACT.ID$, MAX.
N
3650 IF ACT.ID$ <> "ACTIVITIES SCHEDULED:" THEN PRINT "Act
ivity data not found":
3660 FOR N = 1 TO MAX.N
3670 IF EOF(2) THEN PRINT "EOF before MAX.N..." MA
X.N N;: DUMMY$ = INPUT$(1): GOTO 3700
3680 LINE INPUT#2, ACT$(N)
3690 NEXT
3700 CLOSE#2
3710 RETURN
3720 '
3730 'dynamic array size increase...
3740 IF TMP$(0) <> CHR$(255) THEN ERASE TMP$: DIM TMP$(MA
X.N)
3750 FOR M = 1 TO MAX.N: TMP$(M) = ACT$(M): NEXT
3760 ERASE ACT$: DIM ACT$(MAX.N + 10)
3770 FOR M = 1 TO MAX.N: ACT$(M) = TMP$(M): NEXT
3780 MAX.N = MAX.N + 10
3790 RETURN
3800 'make pilnn.dat filenames...

```

```

3810     NUM$ = STR$(NUM)
3820     NUM$ = MID$(NUM$,2)
3830     FILENAME$ = "PIL" + NUM$ + ".DAT"
3840     TMP.FILE$ = "PIL" + NUM$ + ".$$$"
3850     BAK.FILE$ = "PIL" + NUM$ + ".BAK"
3860 RETURN
3870 '
3880 'print all pilot names to screen...
3890 PRINT CLR$
3900 FOR I = 1 TO 20
3910     NUM = CVI(P$(I,1)): L.NAME$ = P$(I,2): INIT$ = P$(I,
3920     PRINT USING "###"; NUM;: PRINT " - " LEFT$(L.NAME$,1
3930     NUM = CVI(P$(I+20,1)): L.NAME$ = P$(I+20,2): INIT$ =
3940     PRINT TAB(27) USING "###"; NUM;: PRINT " - " LEFT$(L
3950     NUM = CVI(P$(I+40,1)): L.NAME$ = P$(I+40,2): INIT$ =
3960     PRINT TAB(55) USING "###"; NUM;: PRINT " - " LEFT$(L
3970 NEXT
3980 RETURN
3990 '
4000 'get pilot names from key file...
4010 FOR I = 1 TO MAX.PIL.NUM
4020     GET#1, I
4030     P$(I,1) = N1$: P$(I,2) = N2$: P$(I,3) = N3$:
4040     P$(I,4) = N4$
4050 NEXT
4060 RETURN
4070 'open and field def files...
4080 OPEN "R", 1, "PILNAM.DEF", 27
4090 FIELD#1, 2 AS N1$, 20 AS N2$, 2 AS N3$, 3 AS N4$
4100 RETURN
4110 '
4120 OPEN "R", 2, "QUAL.DEF", 22
4130 FIELD#2, 2 AS N21$, 20 AS N22$
4140 RETURN
4150 '
4160 OPEN "R", #2, "CUREVNT.DEF", 26
4170 FIELD#2, 2 AS N21$, 20 AS N22$, 2 AS N23$, 2 AS N24$
4180 RETURN
4190 '
4200 OPEN "R", 2, "ACT.DEF", 46
4210 FIELD#2, 2 AS N21$, 20 AS N22$, 2 AS N23$, 2 AS N24$
4220 , 10 AS N25$, 10 AS N26$
4230 RETURN
4240 'correct all name area variables...
4250 FOR I = 1 TO 3: ON I GOSUB 4270, 4360, 4400: NEXT I
4260 RETURN

```

```

4270 PRINT MID$.SCRNS$
4280 INPUT "What is the pilot's last name? ", L$.NAME$
4290 PRINT MID$.SCRNS$
4300 INPUT "What is his first name? ", F$.NAME$
4310 PRINT MID$.SCRNS$
4320 LINE INPUT "Enter his middle initial(s), 'Jr.', etc.,
or 0 (zero) for none: ", MI$
4330 IF MI$ = "0" THEN MI$ = ""
4340 RETURN
4350 '
4360 PRINT MID$.SCRNS$
4370 INPUT "What is his rank/grade? ", RANK$
4380 RETURN
4390 '
4400 PRINT MID$.SCRNS$
4410 INPUT "What is his service number (SSAN)? ", SSAN$
4420 RETURN
4430 '
4440 'read in activity codes and names, assumes def file
open as #2...
4450 PRINT "Select the desired activity category"
4460 PRINT " 1 for non-duty (leave, TDY, etc)"
4470 PRINT " 2 for non-flying duty activities"
4480 PRINT " 3 for flying activities"
4490 PRINT "Which category? "; A = VAL(INPUT$(1)); PRINT
A
4500 GP = (A - 1)*64
4510 FOR I = 1 TO 21
4520 GET#2, I + GP
4530 ACT.CODE = CVI(N21$)
4540 ACT.NAME$ = N22$
4550 PRINT USING "###"; ACT.CODE; PRINT " - " ACT.NAME
$;
4560 GET#2, I + GP + 21
4570 ACT.CODE = CVI(N21$)
4580 ACT.NAME$ = N22$
4590 PRINT TAB(27) USING "###"; ACT.CODE; PRINT " - "
ACT.NAME$;
4600 GET#2, I + GP + 42
4610 ACT.CODE = CVI(N21$)
4620 ACT.NAME$ = N22$
4630 PRINT TAB(55) USING "###"; ACT.CODE; PRINT " - "
ACT.NAME$;
4640 NEXT
4650 PRINT: PRINT "Which activity code?"; INPUT " ", CODE$
4660 RETURN
4670 '
4680 'other, input start and end time offsets...
4690 PRINT MID$.SCRNS$
4700 PRINT "Enter the amount of time (hrs:min) needed prio
r to the scheduled"
4710 PRINT "activity time (e.g. travel time to a meeting o
r briefing time)"
4720 PRINT: PRINT "How much time? "; GOSUB 5450
4730 START = DUR

```

```

4740 PRINT MID.SCRNs
4750 PRINT"Enter the amount of time for the activity, include debriefing, "
4760 PRINT"return travel, etc as applicable"
4770 PRINT: PRINT"How much time? "; GOSUB 5450
4780 END.T = DUR
4790 RETURN
4800 '
4810 'long duration activities--leave, tdy, etc...
4820 PRINT MID.SCRNs
4830 PRINT"What is the ending date of " ACT.NAMEs
4840 GOSUB 5630
4850 END.DATE = DATE
4860 RETURN
4870 '
4880 'sort activities...
4890 SWAP. = -1: LAST = MAX.N - 1
4900 WHILE SWAP.
4910     SWAP. = 0
4920     FOR I = 1 TO LAST
4930         SD1 = VAL(MID$(ACT$(I),6,5)): ST1 = VAL(MID$(ACT$(I),11,5))
4940         SD2 = VAL(MID$(ACT$(I+1),6,5)): ST2 = VAL(MID$(ACT$(I+1),11,5))
4950         IF (SD1 > SD2) OR (SD1 = SD2 AND ST1 > ST2)
THEN GOSUB 5000
4960     NEXT
4970     LAST = LAST - 1
4980 WEND
4990 RETURN
5000     TMP$ = ACT$(I+1)
5010     ACT$(I+1) = ACT$(I)
5020     ACT$(I) = TMP$
5030     SWAP. = -1
5040 RETURN
5050 '
5060 'conflict check, done after activities sorted by start...
5070     LAST = MAX.N - 1
5080     FOR I = 1 TO LAST
5090         ED1 = VAL(MID$(ACT$(I),16,5)): ET1 = VAL(MID$(ACT$(I),21,5))
5100         SD2 = VAL(MID$(ACT$(I+1),6,5)): ST2 = VAL(MID$(ACT$(I+1),11,5))
5110         '
5120         'conflict is TRUE if first activity ends after second starts...
5130         IF (ED1 < SD2) OR (ED1 = SD2 AND ET1 < ST2)
THEN CONFLICT = 0 ELSE CONFLICT = -1
5140         IF CONFLICT THEN PRINT"Conflict found with:"
: PRINT ACT$(I): PRINT ACT$(I+1)
5150     NEXT
5160 RETURN
5170 '
5180 'time of day validating routine ...

```

```

5190     NT = 0
5200     T$ = ""
5210     WHILE T$ <> ":" AND NT < LEN(TIMES$)
5220         NT = NT + 1
5230         T$ = MID$(TIMES$,NT,1)
5240     WEND
5250     IF NT = 0 GOTO 5390
5260     IF NT = LEN(TIMES$) THEN NT = LEN(TIMES$) - 1: MIN = V
AL(RIGHT$(TIMES$,2)) ELSE MIN = VAL(RIGHT$(TIMES$,LEN(TIMES$) -
NT))
5270     HR=VAL(LEFT$(TIMES$,NT-1))
5280     BAD = 0
5290     IF MIN < 0 OR MIN > 59 THEN BAD = -1
5300     IF HR < 0 OR HR > 24 THEN BAD = -1
5310     TIME = HR*60 + MIN
5320     T$ = CHR$(INT(HR/10)+48)
5330     I$ = CHR$((HR MOD 10)+48)
5340     M$ = CHR$(INT(MIN/10)+48)
5350     E$ = CHR$((MIN MOD 10)+48)
5360     TIMES$ = T$ + I$ + M$ + E$
5370     IF BAD THEN PRINT"Time " TIMES$ " not understood, ple
ase re-enter:"; INPUT " ", TIMES$: GOTO 5190
5380     NT = 0: T$ = "": I$ = "": M$ = "": E$ = "": BAD = 0:
HR = 0: MIN = 0
5390 RETURN
5400
5410
5420     --- This routine accepts an input of numbers until a
colon is keyed,
5430     then allows only two digits up to a value of
60.
5440
5450     CK$ = ""
5460     DIGIT$ = INPUT$(1)
5470     IF ASC(DIGIT$)<48 OR ASC(DIGIT$)>58 THEN PRINT "Nume
rical digits or colon (:) only, please re-enter: "; GOTO 54
60
5480     CK$ = CK$ + DIGIT$
5490     IF RIGHT$(CK$,1) <> ":" THEN 5460
5500     HR = VAL(LEFT$(CK$,LEN(CK$)-1))
5510     MINS = INPUT$(2)
5520     IF VAL(MINS) > 60 THEN PRINT "Max number of minutes
is 60, please re-enter: "; GOTO 5510
5530     MIN = VAL(MINS)
5540     CK$ = CK$ + MINS
5550     PRINT: PRINT "The interval entered is: " CK$ ", is t
his correct?"
5560     PRINT C$; A$ = INPUT$(1): PRINT A$
5570     IF A$ = "1" THEN PRINT"Re-enter interval from beginn
ing: "; GOTO 5450
5580     IF A$ <> "0" THEN PRINT E$: GOTO 5560
5590     DUR = HR*60 + MIN
5600 RETURN
5610
5620

```

```

5630 IF MONTH$(0) (> CHR$(255) THEN ERASE MONTH$
5640 DIM MONTH$(12)
5650 MONTH$(1) = "JAN"
5660 MONTH$(2) = "FEB"
5670 MONTH$(3) = "MAR"
5680 MONTH$(4) = "APR"
5690 MONTH$(5) = "MAY"
5700 MONTH$(6) = "JUN"
5710 MONTH$(7) = "JUL"
5720 MONTH$(8) = "AUG"
5730 MONTH$(9) = "SEP"
5740 MONTH$(10) = "OCT"
5750 MONTH$(11) = "NOV"
5760 MONTH$(12) = "DEC"
5770 '
5780 IF FIRST.DAY(0) = 0 THEN ERASE FIRST.DAY
5790 DIM FIRST.DAY(12)
5800 '
5810 'reset FIRST.DAY(3..12) if correcting a date...
5820 '
5830 FIRST.DAY(1) = 1
5840 FIRST.DAY(2) = 32
5850 FIRST.DAY(3) = 60
5860 FIRST.DAY(4) = 91
5870 FIRST.DAY(5) = 121
5880 FIRST.DAY(6) = 152
5890 FIRST.DAY(7) = 182
5900 FIRST.DAY(8) = 213
5910 FIRST.DAY(9) = 244
5920 FIRST.DAY(10) = 274
5930 FIRST.DAY(11) = 305
5940 FIRST.DAY(12) = 335
5950 '
5960 '--- get the date ---
5970 '
5980 INPUT "What is the date (Day Month Year)", DATE$
5990 IF DATE$ = "0" THEN DATE = 0: RETURN
6000 '
6010 'put the date chars in individual variables...
6020 '
6030 IF D$(0) (> CHR$(255) THEN ERASE D$
6040 DIM D$(LEN(DATE$))
6050 FIRST.DLMTR = 0
6060 '
6070 FOR I.V = 1 TO LEN(DATE$)
6080     D$(I.V) = MID$(DATE$, I.V, 1)
6090     IF FIRST.DLMTR (>) 0 THEN 6180
6100 'if first delimiter not set, look for it; allow almo
st
6110 '    any char except letters or numbers to delimit...
6120     D = ASC(D$(I.V))
6130     IF D < 48 THEN DLMT = -1
6140     IF (D > 57 AND D < 65) THEN DLMT = -1
6150     IF (D > 90 AND D < 96) THEN DLMT = -1
6160     IF DLMT THEN FIRST.DLMTR = I.V

```

```

6170             DLMT = 0
6180     NEXT
6190
6200     'assume the last two chars are the year...
6210
6220     YEAR = VAL(RIGHT$(DATE$,2))
6230
6240     'find the day...
6250     'if a delimiter was found then day is the value
6260     'before the delimiter, otherwise the day is either
6270     'the first character or the first two characters of
6280     'the string--assume the first two characters if the
6290     'second character is not a letter
6300
6310     IF FIRST.DLMTR THEN DAY = VAL(LEFT$(DATE$,FIRST.DLMT
R - 1)) ELSE IF ASC(D$(2)) < 58 THEN DAY = VAL(LEFT$(DATE$,2
)): FIRST.DL
MTR = 2 ELSE DAY = VAL(LEFT$(DATE$,1)): FIRST.DLMTR = 1
6320
6330     'find the month...
6340     'just look at three characters past the day or past
6350     'the first delimiter
6360     '    - month could be a number or letters
6370     '    - convert lower case letters to upper
6380
6390     MONTH$=""
6400     MON.NUM = 0
6410     FOR I.V = 1 TO 3
6420         IF ASC(D$(FIRST.DLMTR+I.V)) < 58 THEN MON.NU
M = -1
6430         IF ASC(D$(FIRST.DLMTR+I.V)) > 96 THEN D$(FIR
ST.DLMTR+I.V) = CHR$(ASC(D$(FIRST.DLMTR+I.V))-32)
6440         MONTH$ = MONTH$ + D$(FIRST.DLMTR+I.V)
6450     NEXT
6460
6470     'MONTH$ is now a string of numbers or letters,
6480     '    MON.NUM is TRUE if it is numbers
6490
6500     IF MON.NUM THEN MONTH=VAL(LEFT$(MONTH$,2)): GOTO 658
0
6510     FOR I.V = 1 TO 12
6520         FOR J.V = 1 TO 3
6530             IF MID$(MONTH$,J.V,1) = MID$(MONTH$(
I.V),J.V,1) THEN TEST = -1 ELSE TEST = 0
6540             IF NOT TEST GOTO 6570. 'one not mat
ching is enough
6550         NEXT J.V
6560         IF TEST THEN MONTH = I.V. GOTO 6580: 'found
a match
6570     NEXT I.V
6580     IF MONTH < 1 OR MONTH > 12 THEN INPUT"Month not unde
rstood--enter the month as a one or two digit number (1..12)
", MONTH: GO
TO 6580
6590

```

```

6600 'MONTH is now valid, set MONTH$ if reqd...
6610 '
6620 IF MON.NUM THEN MONTH$ = MONTH$(MONTH)
6630 '
6640 'check if this is a leap year...
6650 '
6660 IF YEAR/4 = INT(YEAR/4) THEN LEAP.YEAR = -1 ELSE LEA
P.YEAR = 0
6670 '
6680 'if so must increment first day values after
6690 '    February...
6700 '
6710 IF LEAP.YEAR THEN FOR I.V = 3 TO 12: FIRST.DAY(I.V)
= FIRST.DAY(I.V) + 1: NEXT
6720 '
6730 'make sure the number of days is valid for the
6740 '    month
6750 '    compute max days in month...
6760 IF MONTH = 12 THEN MAX.DAYS = " 31" ELSE MAX.DAYS =
STR$(FIRST.DAY(MONTH + 1) - FIRST.DAY(MONTH))
6770 MAX.DAYS = MID$(MAX.DAYS,2,2)
6780 '    then check range
6790 IF DAY < 1 OR DAY > VAL(MAX.DAYS) THEN PRINT "Day of
month not understood--input day as a number <1.." MAX.DAYS
">"; INPUT"
" , DAY
6800 '
6810 'now put it together and see if correct...
6820 '
6830 DAYS = STR$(DAY): YR$ = STR$(YEAR): DATES = DAYS + "
" + MONTH$ + YR$
6840 PRINT"The date entered is: "; DATES
6850 PRINT: PRINT C$;
6860 A$ = INPUT$(1)
6870 IF A$ = "1" THEN GOTO 5830: 'try again...
6880 IF A$ <> "0" THEN PRINT E$: GOTO 6840
6890 '
6900 'date is valid and checked correct, make the julian
6910 '    date...
6920 '    julian date form is year digit * 1000 + juli
an date
6930 '
6940 DATE = VAL(RIGHT$(STR$(YEAR),1))*1000 + FIRST.DAY(MO
NTH) + DAY -1
6950 '
6960 'reset all variables not needed
6970 '
6980 ERASE D$
6990 YEAR = 0: MONTH = 0: DAY = 0: MON.NUM = 0
7000 FIRST.DLMTR = 0: A$ = "": MAX.DAYS = ""
7010 DAYS = "": MONTH$ = "": YR$ = ""
7020 RETURN
7030 '

```

```

100  '*** SHELL SET ****
110  'program dated 17 May 1983
120
130  'This program allows entry of the schedule shell
140  '    data for a given week
150
160  'variables required:
170  '    none
180
190  'returns:
200  '    SHELLnn.DAT files updated and in order
210
220  DEFINT A-Z
230  CLR$ = CHR$(26): DOWN$ = CHR$(10): ESC$ = CHR$(27)
240  MID$ = CLR$ + STRING$(6,10)
250  UP$ = CHR$(11): MOV.LEFT$ = CHR$(8): MOV.RIGHT$ = CH
R$(12)
260  HOME$ = CHR$(30): CLR.LINE$ = ESC$ + "T"
270  C$ = "Enter: 0 if correct, 1 to change it: "
280  E$ = "Error, enter 0 or 1 only, try again..."
290
300  MAX.PIL.NUM = 60
310  DIM P$(MAX.PIL.NUM,4)
320
330  'get pilot names...
340  GOSUB 3250
350  GOSUB 3180
360  CLOSE
370
380  PRINT MID$.SCRN$ "      Enter:"
390  PRINT"          0 to quit, all done"
400  PRINT"          1 to add a new shell data file"
410  PRINT"          2 to change data in existing data fi
le"
420  PRINT"          3 to delete a shell data file"
430  PRINT" Which choice? ";: SEL = VAL(INPUT$(1)): PRIN
T SEL
440  IF SEL <= 0 THEN END ELSE IF SEL > 3 THEN PRINT"Erro
r, enter a number 0 to 3 only, try again...": GOTO 430
450  PRINT MID$.SCRN$ "Enter the week starting date (Sunda
y):"
460  GOSUB 4370
470  WK.DATE = DATE: WK.DATES = DATE$
480  WK.NUM = (WK.DATE MOD 1000)\7: WK.NUM$ = MID$(STR$(W
K.NUM),2)
490  FILENAME$ = "SHELL" + WK.NUM$ + ".DAT"
500
510  ON SEL GOSUB 570, 1090, 1840
520  GOTO 380
530
540  '____new_shell_data_____
550
560  'check if file already exists...
570  ON ERROR GOTO 590
580  OPEN "I", 1, FILENAME$

```

```

590     IF ERR = 53 THEN RESUME 640
600     PRINT FILENAME$ " exists on this disk, confirm you w
ant to overwrite (destroy) it"
610     PRINT: PRINT "Enter 0 to continue, 1 to NOT overwrite
this file: "; A$ = INPUT$(1): PRINT A$
620     IF A$ = "1" GOTO 380
630     IF A$ <> "0" THEN PRINT E$: GOTO 610
640     ON ERROR GOTO 0
650     CLOSE
660     '
670     'open shell data file as #1...
680     GOSUB 2900
690     '
700     'open activity definiton file as #2...
710     GOSUB 3290
720     '
730     A = 1: N = 0: MAX.N = 10
740     WHILE A
750         N = N + 1
760         PRINT MID$.SCRN$;
770         GOSUB 2280
780         '
790         PRINT "Enter:"
800         PRINT " 0 if entries complete"
810         PRINT " 1 if more activities to enter"
820         PRINT "Which one? "; A = VAL(INPUT$(1)): PRI
NT A
830         IF A < 0 OR A > 1 THEN PRINT E$;: GOTO 820
840         'if greater than dimension of variable, then expand
it...
850         IF N >= MAX.N THEN GOSUB 2970
860         ACT$(N) = ACT$.LN$
870     WEND: 'activity entry loop...
880     'save max number activites...
890     MAX.N = N
900     '
910     'entries complete, sort them...
920     GOSUB 3710
930     'print to file...
940     '
950     FOR I = 1 TO MAX.N
960         LSET N9$ = ACT$(I)
970         'make first word equal to record number...
980         LSET N1$ = MKI$(I)
990         PUT#1, I
1000    NEXT
1010    'last entry is all 255 chars...
1020    LSET N9$ = STRING$(30,255): LSET N2$ = MKI$(32767):
PUT#1
1030    CLOSE
1040    RETURN
1050    '
1060    '
1070    '____change_or_add_to_existing_shell_data_files____
1080    '

```

```

1090 PRINT MID$ SCRN$: WK.DATES: " - " FILENAME$:
1100 PRINT: PRINT" Enter:"
1110 PRINT" 0 if no more changes or additions, a
11 done"
1120 PRINT" 1 to add activity data"
1130 PRINT" 2 to change activity data"
1140 PRINT" 3 to delete activity data"
1150 PRINT" Which choice? "; A = VAL(INPUT$(1)): PRINT
A
1160 '
1170 IF A = 0 THEN RETURN
1180 IF A > 3 THEN PRINT E$: GOTO 1150
1190 '
1200 'open shell data file and act.code file...
1210 GOSUB 2900: GOSUB 3290
1220 A.T = 0: N = 0
1230 WHILE A.T <> 32767
1240 N = N + 1
1250 GET#1, N: SEQ.NUM = CVI(N1$): A.T = CVI(N2$): IF (N
<> SEQ.NUM) AND (N <> -1) THEN PRINT"Error in " FILENAME$: "
record" N "n
ot equal to sequence number" SEQ.NUM
1260 WEND
1270 MAX.N = N
1280 '
1290 ON A GOSUB 1330, 1670, 1840
1300 CLOSE: GOTO 1090
1310 '
1320 'add a new activity to shell data file...
1330 N = MAX.N: GOSUB 2280: MAX.N = N
1340 DATA.TIME = -1
1350 WHILE DATA.TIME <= ACT.SCHED.TIME
1360 K = K + 1
1370 GET#1, K: DATA.TIME = CVI(N2$)
1380 WEND
1390 '#k is first record > than activity to insert...
1400 MEM = FRE(0): IF MEM < (MAX.N - K + 1)*32 THEN PRINT
"Not enough memory, moving one record at a time...": GOTO 15
40
1410 IF ACT$(0) <> CHR$(255) THEN ERASE ACT$: DIM ACT$(MA
X.N + 1 - K)
1420 FOR M = K TO MAX.N
1430 GET#1, M
1440 ACT$(M - K + 1) = N9$
1450 NEXT
1460 LSET N9$ = ACT.LN$: LSET N1$ = MKI$(K)
1470 PUT#1, K
1480 FOR M = K + 1 TO MAX.N + 1
1490 LSET N9$ = ACT$(M - K)
1500 LSET N1$ = MKI$(M)
1510 PUT#1, M
1520 NEXT
1530 GOTO 1640
1540 GET#1, K: TMP1$ = N9$
1550 LSET N9$ = ACT.LN$: LSET N1$ = MKI$(K)

```

```

1560     PUT#1, K
1570     FOR M = K + 1 TO MAX.N
1580         GET#1, M: TMP2$ = N9$
1590         LSET N9$ = TMP1$: LSET N1$ = M
1600         PUT#1, M
1610         TMP1$ = TMP2$
1620     NEXT
1630     LSET N9$ = TMP1$: PUT#1
1640     MAX.N = MAX.N + 1
1650 RETURN
1660 '
1670     PRINT MID$.SCRN$ "Enter the sequence number to change
or (? for help):";
1680     INPUT " ", A$
1690     IF A$ = "?" THEN GOSUB 1970 ELSE N = VAL(A$)
1700     IF N < 1 OR N > MAX.N THEN PRINT "Out of range...": G
OSUB 1970
1710     'good sequence number entered...
1720     GET#1, N
1730     SEQ.NUM = CVI(N1$): ACT.SCHED.TIME = CVI(N2$): ACT.C
ODE = ASC(N3$): PIL.NUM = ASC(N4$): ACT.ST.TIME = CVI(N5$):
ACT.END.TIME
= CVI(N6$): ACT.NAME$ = N7$
1740     PRINT MID$.SCRN$ SEQ.NUM; (ACT.SCHED.TIME MOD 1440);
ACT.NAME$
1750     PRINT "Enter 0 if this is the correct activity, 1 to
search further: ";
1760     A = VAL(INPUT$(1)): PRINT A
1770     IF A = 1 THEN GOTO 1670 ELSE IF A <> 0 THEN PRINT E$
: GOTO 1750
1780     '
1790     'add activity entry...
1800     A = 1
1810     GOSUB 2280
1820 RETURN
1830 '
1840     PRINT "Delete not written yet...". RETURN
1850     '
1860     '____delete_complete_shell_data_file____
1870     '
1880     PRINT MID$.SCRN$ WK.DATES$ FILENAME$
1890     PRINT "Enter 0 to delete this file, 1 to abort delete
action"
1900     INPUT "Which one"; D
1910     IF D <> 0 THEN PRINT "Exiting delete mode, file NOT d
eleted...": FOR I = 1 TO 1000: NEXT: GOTO 1920
1920 RETURN
1930 '
1940     '____subroutines____
1950     '
1960     'display shell file 20 lines at a time...
1970     M = 1: A.T = 0
1980     WHILE A.T <> 32767
1990         GET#1, M: S.N = CVI(N1$): A.T = CVI(N2$): A.C = ASC(
N3$): P.N = ASC(N4$): S.T = CVI(N5$): E.T = CVI(N6$): A.N$ =

```

```

N79
2000     IF A.T (>) 32767 THEN PRINT S.N A.T A.C P.N S.T E.T A
.N9
2010     M = M + 1
2020     IF M MOD 20 = 1 THEN PRINT"Press <RETURN> to continu
e or sequence number if found:"; INPUT " ", A$: IF A$ = "" T
HEN GOTO 206
0 ELSE N = VAL(A$): GOTO 2050
2030     WEND
2040     PRINT"At end of shell data file for " WK.DATES " , "
FILENAME$: PRINT: PRINT"Press <RETURN> to start over or sequ
ence number
found:"; INPUT " ", A$: IF A$ = "" THEN GOTO 1970 ELSE N = V
AL(A$): GOTO 2050
2050     RETURN
2060     PRINT UP: CLR.LINES; GOTO 1980
2070
2080     'get and confirm pilot number...
2090     PRINT MID.SCRN$;
2100     PRINT"Enter the last name or pilot number:"; INPUT"
", ANSWER$
2110     IF ASC(LEFT$(ANSWER$,1)) < 58 THEN NUM = VAL(ANSWER$
): THIS.NUM = -1 ELSE NUM = 0: L.NAME$ = ANSWER$
2120     'look for name match...
2130     WHILE NUM < MAX.PIL.NUM AND NOT THIS.NUM
2140         NUM = NUM + 1
2150         IF L.NAME$ = LEFT$(P$(NUM,2),LEN(L.NAME$)) T
HEN THIS.NUM = -1 ELSE THIS.NUM = 0
2160     WEND
2170     IF THIS.NUM = 0 THEN GOSUB 3060. PRINT. PRINT"Enter
pilot number: "; INPUT NUM
2180     IF NUM = 0 THEN GOTO 2250
2190     PRINT MID.SCRN$ CVI(P$(NUM,1)) " - " P$(NUM,2) P$(NU
M,3) " " P$(NUM,4)
2200     PRINT"Enter 0 if this the correct entry; 1 if not co
rrect:"; THIS.NUM = VAL(INPUT$(1)) PRINT THIS.NUM
2210     IF THIS.NUM = 1 THEN THIS.NUM = 0. GOTO 2170
2220     IF THIS.NUM (>) 0 THEN PRINT"Error, enter 0 or 1 only
...": GOTO 2190
2230     'have correct number...
2240     PIL.NUM = NUM
2250     RETURN
2260
2270     'input a new activity...
2280     PRINT"Enter activity code (? for help):";
2290     INPUT " ", CODE$
2300     IF CODE$ = "?" THEN GOSUB 3340 ELSE IF CODE$ = "0" T
HEN GOTO 2780
2310     ACT.CODE = VAL(CODE$)
2320     IF ACT.CODE < 1 OR ACT.CODE > 255 THEN PRINT"Entry i
s out of range...": GOSUB 3340: GOTO 2310
2330     IF (ACT.CODE AND 63) = 63 THEN OTHER = -1 ELSE OTHER
= 0
2340     IF OTHER THEN INPUT"What is the activity name? ", AC
T.NAME$: GOTO 2400

```

```

2350 GET#2, ACT.CODE
2360 TRIM$ = N22$
2370 GOSUB 2810
2380 ACT.NAME$ = TRIM$
2390 '
2400 PRINT MID.SCRN$;
2410 PRINT" Enter the day " ACT.NAME$ " occurs on:"
2420 PRINT"          1 - Sunday"
2430 PRINT"          2 - Monday"
2440 PRINT"          3 - Tuesday"
2450 PRINT"          4 - Wednesday"
2460 PRINT"          5 - Thursday"
2470 PRINT"          6 - Friday"
2480 PRINT"          7 - Saturday"
2490 PRINT" Which day?";
2500 D = VAL(INPUT$(1))
2510 IF D < 1 OR D > 7 THEN PRINT"Error, enter a number 1
to 7 only, try again...": GOTO 2490
2520 PRINT MID.SCRN$;
2530 PRINT"Enter the scheduled time: ";
2540 INPUT TIME$: GOSUB 3920
2550 ACT.SCHED.TIME = (D-1)*1440 + TIME
2560 IF OTHER THEN GOSUB 3580: GOTO 2660
2570 'not other...
2580 PRINT MID.SCRN$;
2590 PRINT" Enter:"
2600 PRINT"          0 if standard time offsets apply"
2610 PRINT"          1 to change them"
2620 PRINT" Which choice? "; A$ = INPUT$(1): PRINT A$
2630 IF A$ = "1" THEN GOSUB 3580: GOTO 2660
2640 IF A$ <> "0" THEN PRINT E$: GOTO 2590
2650 START = CUI(N23$): END.T = CUI(N24$)
2660 ACT.ST.TIME = ACT.SCHED.TIME - START: ACT.END.TIME =
ACT.SCHED.TIME + END.T
2670 PRINT MID.SCRN$ "Is a pilot already assigned to this
activity?"
2680 PRINT"Enter 0 if no pilot assigned or the pilot name
or number to specify which pilot: ";
2690 INPUT A$
2700 IF A$ <> "0" THEN ANSWER$ = A$: GOSUB 2110 ELSE PIL.
NUM = 255
2710 ACT.LN$ = STRING$(30,0)
2720 MID$(ACT.LN$,3,2) = MKI$(ACT.SCHED.TIME)
2730 MID$(ACT.LN$,5,1) = CHR$(ACT.CODE)
2740 MID$(ACT.LN$,6,1) = CHR$(PIL.NUM)
2750 MID$(ACT.LN$,7,2) = MKI$(ACT.ST.TIME)
2760 MID$(ACT.LN$,9,2) = MKI$(ACT.END.TIME)
2770 MID$(ACT.LN$,11) = ACT.NAME$
2780 RETURN
2790 '
2800 'trim trailing spaces...
2810 L = LEN(TRIM$) + 1: L.CHRS = CHR$(0)
2820 WHILE ASC(L.CHRS) < 33
2830     L = L - 1
2840     L.CHRS = MID$(TRIM$,L,1)

```

```

2850      WEND
2860      TRIM$ = LEFT$(TRIM$,L)
2870      RETURN
2880      '
2890      'open and field shell data file...
2900      OPEN "R", 1, FILENAME$, 30
2910      '          seq.num act.sched.time act.code pil.num act
st.time act.end.time act.name
2920      FIELD#1, 2 AS N1$, 2 AS N2$, 1 AS N3$, 1 AS N4$, 2 A
S N5$, 2 AS N6$, 20 AS N7$
2930      FIELD#1, 30 AS N9$
2940      RETURN
2950      '
2960      'dynamic array size increase...
2970      MEM = FRE(0): IF MEM < 320 THEN PRINT"Not enough fre
e memory, save this to disk and continue...": MEM = -1: RETU
RN
2980      IF TMP$(0) <> CHR$(255) THEN ERASE TMP$: DIM TMP$(MA
X.N)
2990      FOR M = 1 TO MAX.N: TMP$(M) = ACT$(M): NEXT
3000      ERASE ACT$: DIM ACT$(MAX.N + 10)
3010      FOR M = 1 TO MAX.N: ACT$(M) = TMP$(M): NEXT
3020      MAX.N = MAX.N + 10
3030      RETURN
3040      '
3050      'print all pilot names to screen...
3060      PRINT CLR$
3070      FOR I = 1 TO 20
3080      NUM = CVI(P$(I,1)): L.NAMES$ = P$(I,2): INIT$ = P$(I,
3): RANK$ = P$(I,4)
3090      PRINT USING "###"; NUM;: PRINT " - " LEFT$(L.NAMES$,1
1) INIT$: " " RANK$;
3100      NUM = CVI(P$(I+20,1)): L.NAMES$ = P$(I+20,2): INIT$ =
P$(I+20,3): RANK$ = P$(I+20,4)
3110      PRINT TAB(27) USING "###"; NUM;: PRINT " - " LEFT$(L
.NAMES$,11) INIT$: " " RANK$;
3120      NUM = CVI(P$(I+40,1)): L.NAMES$ = P$(I+40,2): INIT$ =
P$(I+40,3): RANK$ = P$(I+40,4)
3130      PRINT TAB(55) USING "###"; NUM;: PRINT " - " LEFT$(L
.NAMES$,11) INIT$: " " RANK$;
3140      NEXT
3150      RETURN
3160      '
3170      'get pilot names from key file...
3180      FOR I = 1 TO MAX.PIL.NUM
3190          GET#1, I
3200          P$(I,1) = N1$: P$(I,2) = N2$: P$(I,3) = N3$:
P$(I,4) = N4$
3210      NEXT
3220      RETURN
3230      '
3240      'open and field def files...
3250      OPEN "R", 1, "PILNAM.DEF", 27
3260      FIELD#1, 2 AS N1$, 20 AS N2$, 2 AS N3$, 3 AS N4$
3270      RETURN

```

```

3280      '
3290      OPEN "R", 2, "ACT.DEF", 46
3300      FIELD#2, 2 AS N21$, 20 AS N22$, 2 AS N23$, 2 AS N24$
, 10 AS N25$, 10 AS N26$
3310      RETURN
3320      '
3330      'read in activity codes and names, assumes def file
open as #2..
3340      PRINT"Select the desired activity category"
3350      PRINT"  1 for non-duty (leave, TDY, etc)"
3360      PRINT"  2 for non-flying duty activities"
3370      PRINT"  3 for flying activities"
3380      PRINT"Which category? "; A = VAL(INPUT$(1)); PRINT
A
3390      GP = (A - 1)*64
3400      FOR I = 1 TO 21
3410          GET#2, I + GP
3420          ACT.CODE = CVI(N21$)
3430          ACT.NAME$ = N22$
3440          PRINT USING "****"; ACT.CODE;; PRINT " - " ACT.NAME
$;
3450          GET#2, I + GP + 21
3460          ACT.CODE = CVI(N21$)
3470          ACT.NAME$ = N22$
3480          PRINT TAB(27) USING "****"; ACT.CODE;; PRINT " - "
ACT.NAME$;
3490          GET#2, I + GP + 42
3500          ACT.CODE = CVI(N21$)
3510          ACT.NAME$ = N22$
3520          PRINT TAB(55) USING "****"; ACT.CODE;; PRINT " - "
ACT.NAME$;
3530      NEXT
3540      PRINT: PRINT"Which activity code?"; INPUT " ", CODE$
3550      RETURN
3560      '
3570      'other, input start and end time offsets...
3580      PRINT MID.SCRN$;
3590      PRINT"Enter the amount of time (hrs:min) needed prio
r to the scheduled"
3600      PRINT"activity time (e.g. travel time to a meeting o
r briefing time)"
3610      PRINT: PRINT"How much time? "; GOSUB 4190
3620      START = DUR
3630      PRINT MID.SCRN$;
3640      PRINT"Enter the amount of time for the activity, inc
lude debriefing, "
3650      PRINT"return travel, etc as applicable"
3660      PRINT: PRINT"How much time? "; GOSUB 4190
3670      END T = DUR
3680      RETURN
3690      '
3700      'sort activities...
3710      SWAP = -1: LAST = MAX.N - 1
3720      PRINT MID.SCRN$ "Sorting";
3730      WHILE SWAP.

```

```

3740         SWAP = 0
3750         FOR I = 1 TO LAST
3760             A.T1 = CVAL(MID$(ACT$(I),3,2))
3770             A.T2 = CVAL(MID$(ACT$(I+1),3,2))
3780             IF A.T1 > A.T2 THEN GOSUB 3840
3790         NEXT
3800         LAST = LAST - 1
3810         PRINT
3820     WEND
3830 RETURN
3840     PRINT " ";
3850     TMP$ = ACT$(I+1)
3860     ACT$(I+1) = ACT$(I)
3870     ACT$(I) = TMP$
3880     SWAP = -1
3890 RETURN
3900 '
3910 'time of day validating routine...
3920     NT = 0
3930     T$ = ""
3940     WHILE T$ <> ":" AND NT < LEN(TIMES)
3950         NT = NT + 1
3960         T$ = MID$(TIMES,NT,1)
3970     WEND
3980     IF NT = 0 GOTO 4120
3990     IF NT = LEN(TIMES) THEN NT = LEN(TIMES) - 1: MIN = V
AL(RIGHT$(TIMES,2)) ELSE MIN = VAL(RIGHT$(TIMES,LEN(TIMES) -
NT))
4000     HR=VAL(LEFT$(TIMES,NT-1))
4010     BAD = 0
4020     IF MIN < 0 OR MIN > 59 THEN BAD = -1
4030     IF HR < 0 OR HR > 24 THEN BAD = -1
4040     TIME = HR*60 + MIN
4050     T$ = CHR$(HR\10+48)
4060     I$ = CHR$((HR MOD 10)+48)
4070     M$ = CHR$(MIN\10+48)
4080     E$ = CHR$((MIN MOD 10)+48)
4090     TIMES = T$ + I$ + M$ + E$
4100     IF BAD THEN PRINT "Time " TIMES " not understood, ple
ase re-enter:"; INPUT " ", TIMES: GOTO 3920
4110     NT = 0: T$ = "": I$ = "": M$ = "": E$ = "": BAD = 0:
HR = 0: MIN = 0
4120 RETURN
4130 '
4140 '
4150 ' --- This routine accepts an input of numbers until
4160 '         a colon is keyed, then allows only two
4170 '         digits up to a value of 60 ...
4180 '
4190     CK$ = ""
4200     DIGIT$ = INPUT$(1)
4210     IF ASC(DIGIT$)<48 OR ASC(DIGIT$)>58 THEN PRINT "Nume
rical digits or colon (:) only, please re-enter: "; GOTO 42
00
4220     CK$ = CK$ + DIGIT$

```

```

4230     IF RIGHT$(CK$,1) <> "." THEN 4200
4240     HR = VAL(LEFT$(CK$,LEN(CK$)-1))
4250     MINS = INPUT$(2)
4260     IF VAL(MINS) > 60 THEN PRINT "Max number of minutes
is 60, please re-enter: "; GOTO 4250
4270     MIN = VAL(MINS)
4280     CK$ = CK$ + MIN$
4290     PRINT: PRINT "The interval entered is: " CK$ ", is t
his correct?"
4300     PRINT C$; A$ = INPUT$(1): PRINT A$
4310     IF A$ = "1" THEN PRINT "Re-enter interval from beginn
ing: "; GOTO 4190
4320     IF A$ <> "0" THEN PRINT E$: GOTO 4300
4330     DUR = HR*60 + MIN
4340     RETURN
4350
4360
4370     IF MONTH$(0) <> CHR$(255) THEN ERASE MONTH$
4380     DIM MONTH$(12)
4390     MONTH$(1) = "JAN"
4400     MONTH$(2) = "FEB"
4410     MONTH$(3) = "MAR"
4420     MONTH$(4) = "APR"
4430     MONTH$(5) = "MAY"
4440     MONTH$(6) = "JUN"
4450     MONTH$(7) = "JUL"
4460     MONTH$(8) = "AUG"
4470     MONTH$(9) = "SEP"
4480     MONTH$(10) = "OCT"
4490     MONTH$(11) = "NOV"
4500     MONTH$(12) = "DEC"
4510
4520     IF FIRST.DAY(0) = 0 THEN ERASE FIRST.DAY
4530     DIM FIRST.DAY(12)
4540
4550     'reset FIRST.DAY(3..12) if correcting a date...
4560
4570     FIRST.DAY(1) = 1
4580     FIRST.DAY(2) = 32
4590     FIRST.DAY(3) = 60
4600     FIRST.DAY(4) = 91
4610     FIRST.DAY(5) = 121
4620     FIRST.DAY(6) = 152
4630     FIRST.DAY(7) = 182
4640     FIRST.DAY(8) = 213
4650     FIRST.DAY(9) = 244
4660     FIRST.DAY(10) = 274
4670     FIRST.DAY(11) = 305
4680     FIRST.DAY(12) = 335
4690
4700     '--- get the date ---
4710
4720     INPUT "What is the date <Day Month Year> ", DATE$
4730     IF DATE$ = "0" THEN DATE = 0: RETURN
4740

```

```

4750 'put the date chars in individual variables...
4760 '
4770 IF D$(0) (<) CHR$(255) THEN ERASE D$
4780 DIM D$(LEN(DATES$))
4790 FIRST.DLMTR = 0
4800 '
4810 FOR I.V = 1 TO LEN(DATES$)
4820   D$(I.V) = MID$(DATES$,I.V,1)
4830   IF FIRST.DLMTR (<) 0 THEN 4930
4840 'if first delimiter not set, look for it; allow
4850 'almost any char except letters or numbers to
4860 'delimit...
4870   D = ASC(D$(I.V))
4880   IF D < 48 THEN DLMT = -1
4890   IF (D > 57 AND D < 65) THEN DLMT = -1
4900   IF (D > 90 AND D < 96) THEN DLMT = -1
4910   IF DLMT THEN FIRST.DLMTR = I.V
4920   DLMT = 0
4930 NEXT
4940 '
4950 'assume the last two chars are the year...
4960 '
4970 YEAR = VAL(RIGHT$(DATES$,2))
4980 '
4990 'find the day...
5000 'if a delimiter was found then day is the value
5010 'before the delimiter, otherwise the day is either
5020 'the first character or the first two characters of
5030 'the string--assume the first two characters if the
5040 'second character is not a letter
5050 '
5060 IF FIRST.DLMTR THEN DAY = VAL(LEFT$(DATES$,FIRST.DLMT
R - 1)) ELSE IF ASC(D$(2)) < 58 THEN DAY = VAL(LEFT$(DATES$,2
)): FIRST.DL
MTR = 2 ELSE DAY = VAL(LEFT$(DATES$,1)): FIRST.DLMTR = 1
5070 '
5080 'find the month...
5090 'just look at three characters past the day or past
5100 'the first delimiter
5110 '   - month could be a number or letters
5120 '   - convert lower case letters to upper
5130 '
5140 MONTH$=""
5150 MON.NUM = 0
5160 FOR I.V = 1 TO 3
5170   IF ASC(D$(FIRST.DLMTR+I.V)) < 58 THEN MON.NU
M = -1
5180   IF ASC(D$(FIRST.DLMTR+I.V)) > 96 THEN D$(FIR
ST.DLMTR+I.V) = CHR$(ASC(D$(FIRST.DLMTR+I.V))-32)
5190   MONTH$ = MONTH$ + D$(FIRST.DLMTR+I.V)
5200 NEXT
5210 '
5220 'MONTH$ is now a string of numbers or letters,
5230 '   MON.NUM is TRUE if it is numbers...
5240 '

```

```

5250     IF MON.NUM THEN MONTH=VAL(LEFT$(MONTH$,2)): GOTO 5330
5260     FOR I.V = 1 TO 12
5270         FOR J.V = 1 TO 3
5280             IF MID$(MONTH$,J.V,1) = MID$(MONTH$(
I.V),J.V,1) THEN TEST = -1 ELSE TEST = 0
5290             IF NOT TEST GOTO 5320: ' one not mat
ching is enough
5300         NEXT J.V
5310         IF TEST THEN MONTH = I.V: GOTO 5330: ' found
a match
5320     NEXT I.V
5330     IF MONTH < 1 OR MONTH > 12 THEN INPUT"Month not unde
rstood--enter the month as a one or two digit number <1..12>
", MONTH: GO
TO 5330
5340     '
5350     'MONTH is now valid, set MONTH$ if reqd...
5360     '
5370     IF MON.NUM THEN MONTH$ = MONTH$(MONTH)
5380     '
5390     'check if this is a leap year...
5400     '
5410     IF YEAR/4 = YEAR\4 THEN LEAP.YEAR = -1 ELSE LEAP.YEA
R = 0
5420     '
5430     'if so must increment first day values after
5440     'February...
5450     '
5460     IF LEAP.YEAR THEN FOR I.V = 3 TO 12: FIRST.DAY(I.V)
= FIRST.DAY(I.V) + 1: NEXT
5470     '
5480     'make sure the number of days is valid for the month
5490     'first, compute max days in month...
5500     IF MONTH = 12 THEN MAX.DAYS = " 31" ELSE MAX.DAYS =
STR$(FIRST.DAY(MONTH + 1) - FIRST.DAY(MONTH))
5510     MAX.DAYS = MID$(MAX.DAYS,2,2)
5520     ' then check range
5530     IF DAY < 1 OR DAY > VAL(MAX.DAYS) THEN PRINT "Day of
month not understood--input day as a number <1.." MAX.DAYS
">": INPUT"
", DAY
5540     '
5550     'now put it together and see if correct...
5560     '
5570     DAYS = STR$(DAY): YR$ = STR$(YEAR): DATES = DAYS + "
" + MONTH$ + YR$
5580     PRINT"The date entered is: "; DATES
5590     PRINT: PRINT C$;
5600     A$ = INPUT$(1)
5610     IF A$ = "1" THEN GOTO 4570: 'try again...
5620     IF A$ <> "0" THEN PRINT E$: GOTO 5580
5630     '
5640     'date is valid and checked correct, make the julian
5650     'date...

```

```

5660      '      julian date form is year digit * 1000 + juli
an date
5670      '
5680      DATE = VAL(RIGHT$(STR$(YEAR),1))*1000 + FIRST.DAY(MO
NTH) + DAY -1
5690      '
5700      'reset all variables not needed
5710      '
5720      ERASE D$
5730      YEAR = 0: MONTH = 0: DAY = 0: MON.NUM = 0
5740      FIRST.DLMTR = 0: A$ = "": MAX.DAYS = ""
5750      DAYS = "": MONTH$ = "": YR$ = ""
5760      RETURN
5770      '

```

```

100 '*** WKDAT.SET *****
110 'program dated 17 May 1983
120
130 'This program reads all pilnn.dat files and sets
140 'the weekly data in WKnn.DAT
150
160 'variables required:
170 '    none
180
190 'returns:
200 '    WKnn.DAT file
210
220 DEFINT A-Z
230 CLR$ = CHR$(26): DOWN$ = CHR$(10): ESC$ = CHR$(27)
240 MID$ = CLR$ + STRING$(8,10)
250 UP$ = CHR$(11): MOV.LEFT$ = CHR$(8): MOV.RIGHT$ = CH
R$(12)
260 HOME$ = CHR$(30): CLR.LINE$ = ESC$ + "T"
270 'set avail period constants...
280 PERIOD.ST.TIME = 0: PERIOD.DUR = 10080: INCR = 30
290 'char positions of date/time in acts...
300 SD = 6: ST = 11: ED = 16: ET = 21
310 'fndt.tim pulls the substring value from acts
320 DEF FNDT.TIM(NL,P) = VAL(MID$(ACT$(NL),P,5))
330
340 C$ = "Enter: 0 if correct, 1 to change it: "
350 E$ = "Error, enter 0 or 1 only, try again..."
360
370 MAX.PIL.NUM = 60
380 IF P$(0,0) <> CHR$(255) THEN ERASE P$: DIM P$(MAX.PI
L.NUM,4)
390 IF QUAL$(0) <> CHR$(255) THEN ERASE QUAL$: DIM QUAL$
(15)
400
410 'open and field def files...
420 OPEN "R", 1, "PILNAM.DEF", 27
430 '    pil.num: 1.names: 1.init$: 1.rank$:
440 FIELD#1, 2 AS N1$, 20 AS N2$, 2 AS N3$, 3 AS N4$
450
460 PRINT MID$.SCRN$: "Enter the week starting date (Sunda
y): "
470 GOSUB 1730
480 WK.DATE = DATE: WK.DATES = DATES: WK.NUM = (WK.DATE
MOD 1000)\7: WK.NUM$ = MID$(STR$(WK.NUM),2)
490 PRINT"The week number is " WK.NUM$
500
510 OPEN "R", 2, "WK" + WK.NUM$ + ".DAT", 93
520 '    pil.num: 1.avail$: 1.cur.dt$: 1.qual:
net/nlt:
530 FIELD#2, 1 AS N21$, 42 AS N22$, 20 AS N23$, 2 AS N24
$, 28 AS N25$
540 FIELD#2, 93 AS N2A$
550
560 FOR PIL.NUM = 1 TO MAX.PIL.NUM
570     FOR I = 1 TO 7: NET(I) = -1: NLT(I) = -1: NE

```

```

XT
580             GET#1, PIL.NUM
590             NUM = CVAL(N1$): TRIM$ = N2$: GOSUB 1460 L.N
AME$ = TRIM$: INIT$ = N3$: RANK$ = N4$
600             PRINT"Getting data on " RANK$ " " L.NAME$ "
" INIT$
610             IF NUM <> PIL.NUM THEN PRINT"File error: rec
ord number not equal to pilot number" PRINT"Press any key t
o continue..
";: DUMMY$ = INPUT$(1)
620             NUM$ = MID$(STR$(NUM),2)
630
640             NO.FILE = 0
650             ON ERROR GOTO 680
660             FILENAME$ = "PIL" + NUM$ + ".DAT"
670             OPEN "I", 3, FILENAME$
680             IF ERR = 53 THEN NO.FILE = -1: PRINT FILENAM
E$ " not found, going to next number...": RESUME 690
690             ON ERROR GOTO 0
700             IF NO.FILE THEN LSET N2A$ = STRING$(93,0): L
SET N21$ = CHR$(PIL.NUM): GOTO 1280
710
720             'read in pilot data file...
730             GOSUB 1550
740             'close pilot data file...
750             CLOSE#3
760
770             'make data for each week data record field...
780             LSET N21$ = CHR$(NUM)
790
800             AVAIL$ = STRING$(42,255)
810             FOR N = 1 TO MAX.N
820                 ACT.CODE = VAL(LEFT$(ACT$(N),5))
830                 ACT.ST.DATE = FNDDT.TIM(N,SD)
840                 ACT.ST.TIME = FNDDT.TIM(N,ST)
850                 ACT.END.DATE = FNDDT.TIM(N,ED)
860                 ACT.END.TIME = FNDDT.TIM(N,ET)
870
880             'compute times in minutes from week beginning...
890
900             IF ACT.ST.DATE - WK.DATE > 7 THEN ST
ART.TIME = 32767 ELSE IF ACT.ST.DATE - WK.DATE < -7 THEN STA
RT.TIME = -1
0080 ELSE START.TIME = (ACT.ST.DATE - WK.DATE)*1440 + ACT.ST
.TIME
910             IF ACT.END.DATE - WK.DATE > 7 THEN E
ND.TIME = 32767 ELSE IF ACT.END.DATE - WK.DATE < -7 THEN END
.TIME = -100
80 ELSE END.TIME = (ACT.END.DATE - WK.DATE)*1440 + ACT.END.T
IME
920             'Set defined FALSE, just check activity times
930             AVAIL=-1: SET=0
940             GOSUB 3380
950             IF NOT AVAIL THEN PRINT"Conflict in
activity" N CHR$(8) " , not set. "

```

```

960             IF (ACT.CODE AND 192) <> 0 THEN GOSU
B 5150 ELSE C.REST = -1
970             IF C.REST THEN GOTO 1030
980             PRINT "Activity" N "-" ACT$(N) "does n
ot meet crew rest constraints..."
990             PRINT "Enter: 0 to ignore crew rest,
1 to NOT set this activity: ";
1000            A$ = INPUT$(1); PRINT A$
1010            IF A$ = "1" THEN GOTO 1040 ELSE IF A
$ <> "0" THEN PRINT E$: GOTO 980
1020            'set defined TRUE, set this activity in avail...
1030            AVAIL = -1; SET = -1; GOSUB 3810
1040            NEXT: 'activity for this pilot number...
1050            LSET N22$ = AVAIL$
1060
1070            CUR.DT$ = ""
1080            FOR N = 1 TO 10
1090                CUR.DT$ = CUR.DT$ + MKI$(CUR.DT(N))
1100            NEXT: 'currency event date...
1110            LSET N23$ = CUR.DT$
1120
1130            QUAL = 0
1140            FOR N = 1 TO 15
1150                IF QV(N) THEN QUAL = QUAL + 2*(N-1)
1160            NEXT: 'qualification...
1170            LSET N24$ = MKI$(QUAL)
1180
1190            'initialize all NET and NLT times as -1 values...
1200            NET.NLT$ = STRING$(28,255)
1210            FOR N = 1 TO 7
1220                IF NET(N) <> -1 THEN MID$(NET.NLT$,N
*4 - 3,2) = MKI$(NET(N))
1230                IF NLT(N) <> -1 THEN MID$(NET.NLT$,N
*4 - 1,2) = MKI$(NLT(N))
1240            NEXT
1250            LSET N25$ = NET.NLT$
1260
1270            'save all data in buffer to this pilot number record
1280            PUT#2, PIL.NUM
1290        NEXT: 'pilot number...
1300
1310        'last record, save the date (julian number and strin
g form)
1320        '          rec.num:      wk.date:      wk.date$      not use
d:
1330        FIELD#2, 1 AS N221$, 2 AS N222$, 9 AS N223$, 81 AS N
224$
1340        LSET N221$ = CHR$(MAX.PIL.NUM + 1)
1350        LSET N222$ = MKI$(WK.DATE)
1360        LSET N223$ = WK.DATES
1370        LSET N224$ = STRING$(81,0)
1380        PUT#2, MAX.PIL.NUM + 1
1390
1400        'all pilot data for the week now in one file
1410        PRINT "WK" WK.NUM$: ".DAT file now completed."

```

```

1420     END
1430
1440     '_____subroutines_____
1450
1460     L = LEN(TRIM$) + 1: L.CHR$ = CHR$(0)
1470     WHILE ASC(L.CHR$) < 33
1480         L = L - 1
1490         L.CHR$ = MID$(TRIM$, L, 1)
1500     WEND
1510     TRIM$ = LEFT$(TRIM$, L)
1520     RETURN
1530
1540     'open data file and read into memory, close...
1550     INPUT #3, NUM$, L.NAME$, F.NAME$, MI$, RANK$, SSAN$
1560     IF EOF(3) THEN GOTO 1690 ELSE INPUT#3, QUAL.ID$
1570     IF QUAL.ID$ <> "QUALIFICATIONS:" THEN PRINT"Qual dat
a not found":
1580     IF QV(0) <> -1 THEN ERASE QV: DIM QV(15)
1590     FOR I = 1 TO 15: IF EOF(3) THEN GOTO 1680 ELSE INPUT
#3, QV(I): NEXT
1600     IF EOF(3) THEN GOTO 1680 ELSE INPUT#3, CUR.ID$
1610     IF CUR.ID$ <> "CURRENCIES:" THEN PRINT"Cur data not
found":
1620     FOR I = 1 TO 10: IF EOF(3) THEN GOTO 1680 ELSE INPUT
#3, CUR.DT(I): NEXT
1630     IF EOF(3) THEN GOTO 1680 ELSE INPUT#3, ACT.ID$, MAX.
N
1640     IF ACT.ID$ <> "ACTIVITIES SCHEDULED:" THEN PRINT"Act
ivity data not found":
1650     FOR N = 1 TO MAX.N
1660         IF EOF(3) THEN PRINT"EOF before MAX.N..." MA
X.N N;: DUMMYS = INPUT$(1): GOTO 1680
1670         LINE INPUT#3, ACT$(N)
1680     NEXT
1690     RETURN
1700
1710
1720
1730     IF MONTH$(0) <> CHR$(255) THEN ERASE MONTH$
1740     DIM MONTH$(12)
1750     MONTH$(1) = "JAN"
1760     MONTH$(2) = "FEB"
1770     MONTH$(3) = "MAR"
1780     MONTH$(4) = "APR"
1790     MONTH$(5) = "MAY"
1800     MONTH$(6) = "JUN"
1810     MONTH$(7) = "JUL"
1820     MONTH$(8) = "AUG"
1830     MONTH$(9) = "SEP"
1840     MONTH$(10) = "OCT"
1850     MONTH$(11) = "NOV"
1860     MONTH$(12) = "DEC"
1870
1880     IF FIRST.DAY(0) = 0 THEN ERASE FIRST.DAY
1890     DIM FIRST.DAY(12)

```

```

1900 '
1910 'reset FIRST.DAY(3..12) if correcting a date...
1920 '
1930 FIRST.DAY(1) = 1
1940 FIRST.DAY(2) = 32
1950 FIRST.DAY(3) = 60
1960 FIRST.DAY(4) = 91
1970 FIRST.DAY(5) = 121
1980 FIRST.DAY(6) = 152
1990 FIRST.DAY(7) = 182
2000 FIRST.DAY(8) = 213
2010 FIRST.DAY(9) = 244
2020 FIRST.DAY(10) = 274
2030 FIRST.DAY(11) = 305
2040 FIRST.DAY(12) = 335
2050 '
2060 '--- get the date ---
2070 '
2080 INPUT "What is the date (Day Month Year)", DATES$
2090 IF DATES$ = "0" THEN DATE = 0: RETURN
2100 '
2110 'put the date chars in individual variables...
2120 '
2130 IF D$(0) (>) CHR$(255) THEN ERASE D$
2140 DIM D$(LEN(DATES$))
2150 FIRST.DLMTR = 0
2160 '
2170 FOR I.V = 1 TO LEN(DATES$)
2180     D$(I.V) = MID$(DATES$,I.V,1)
2190     IF FIRST.DLMTR (>) 0 THEN 2290
2200 'if first delimiter not set, look for it; allow
2210 'almost any char except letters or numbers to
2220 'delimit...
2230     D = ASC(D$(I.V))
2240     IF D < 48 THEN DLMT = -1
2250     IF (D > 57 AND D < 65) THEN DLMT = -1
2260     IF (D > 90 AND D < 96) THEN DLMT = -1
2270     IF DLMT THEN FIRST.DLMTR = I.V
2280     DLMT = 0
2290 NEXT
2300 '
2310 'assume the last two chars are the year...
2320 '
2330 YEAR = VAL(RIGHT$(DATES$,2))
2340 '
2350 'find the day...
2360 'if a delimiter was found then day is the value
2370 'before the delimiter, otherwise the day is either
2380 'the first character or the first two characters of
2390 'the string--assume the first two characters if the
2400 'second character is not a letter
2410 '
2420 IF FIRST.DLMTR THEN DAY = VAL(LEFT$(DATES$,FIRST.DLMT
R - 1)) ELSE IF ASC(D$(2)) < 58 THEN DAY = VAL(LEFT$(DATES$,2
)): FIRST.DL

```

```

MTR = 2 ELSE DAY = VAL(LEFT$(DATE$,1)) FIRST.DLMTR = 1
2430
2440 'find the month...
2450 'just look at three characters past the day or past
2460 'the first delimiter
2470 '      - month could be a number or letters
2480 '      - convert lower case letters to upper
2490
2500 MONTH$=""
2510 MON.NUM = 0
2520 FOR I.V = 1 TO 3
2530     IF ASC(D$(FIRST.DLMTR+I.V)) < 58 THEN MON.NU
M = -1
2540     IF ASC(D$(FIRST.DLMTR+I.V)) > 96 THEN D$(FIR
ST.DLMTR+I.V) = CHR$(ASC(D$(FIRST.DLMTR+I.V))-32)
2550     MONTH$ = MONTH$ + D$(FIRST.DLMTR+I.V)
2560 NEXT
2570
2580 'MONTH$ is now a string of numbers or letters.
2590 '      MON.NUM is TRUE if it is numbers
2600
2610 IF MON.NUM THEN MONTH=VAL(LEFT$(MONTH$,2)): GOTO 269
0
2620 FOR I.V = 1 TO 12
2630     FOR J.V = 1 TO 3
2640         IF MID$(MONTH$,J.V,1) = MID$(MONTH$(
I.V),J.V,1) THEN TEST = -1 ELSE TEST = 0
2650         IF NOT TEST GOTO 2680: 'one not mat
ching is enough
2660     NEXT J.V
2670     IF TEST THEN MONTH = I.V: GOTO 2690: 'found
a match
2680 NEXT I.V
2690 IF MONTH < 1 OR MONTH > 12 THEN INPUT"Month not unde
rstood--enter the month as a one or two digit number (1..12)
", MONTH: GO
TO 2690
2700
2710 'MONTH is now valid, set MONTH$ if reqd...
2720
2730 IF MON.NUM THEN MONTH$ = MONTH$(MONTH)
2740
2750 'check if this is a leap year...
2760
2770 IF YEAR/4 = YEAR\4 THEN LEAP.YEAR = -1 ELSE LEAP.YEA
R = 0
2780
2790 'if so must increment first day values after
2800 '      February...
2810
2820 IF LEAP.YEAR THEN FOR I.V = 3 TO 12: FIRST.DAY(I.V)
= FIRST.DAY(I.V) + 1: NEXT
2830
2840 'make sure the number of days is valid for the month
2850 'first, compute max days in month...

```

```

2860     IF MONTH = 12 THEN MAX.DAYS = " 31" ELSE MAX.DAYS =
STR$(FIRST.DAY(MONTH + 1) - FIRST.DAY(MONTH))
2870     MAX.DAYS = MID$(MAX.DAYS,2,2)
2880     ' then check range
2890     IF DAY < 1 OR DAY > VAL(MAX.DAYS) THEN PRINT "Day of
month not understood--input day as a number <1.." MAX.DAYS
">"; INPUT"
" , DAY
2900     '
2910     'now put it together and see if correct...
2920     '
2930     DAYS = MID$(STR$(DAY),2); YR$ = STR$(YEAR); DATE$ =
DAYS + " " + MONTH$ + YR$
2940     PRINT"The date entered is: "; DATE$
2950     PRINT: PRINT C$;
2960     A$ = INPUT$(1)
2970     IF A$ = "1" THEN GOTO 1930: 'try again...
2980     IF A$ <> "0" THEN PRINT E$: GOTO 2940
2990     '
3000     'date is valid and checked correct, make the julian
3010     ' date. julian date form is
3020     ' year digit * 1000 + julian date
3030     '
3040     DATE = VAL(RIGHT$(STR$(YEAR),1))*1000 + FIRST.DAY(MO
NTH) + DAY -1
3050     '
3060     'reset all variables not needed
3070     '
3080     ERASE D$
3090     YEAR = 0: MONTH = 0: DAY = 0: MON.NUM = 0
3100     FIRST.DLMTR = 0: A$ = "": MAX.DAYS = ""
3110     DAYS = "": MONTH$ = "": YR$ = ""
3120     RETURN
3130     '
3140     '*** CASE ***
3150     'module dated 24 April 1983
3160     '
3170     'This module includes subroutines called by other
3180     'modules in determining the case of each activity
3190     'relative to the week
3200     '
3210     'variables required:
3220     ' PERIOD.ST.TIME as an integer in minutes or o
ther time units
3230     ' PERIOD.DUR as an integer length of period
3240     ' INCR as an integer for the value of each bit
(resolution)
3250     ' START.TIME as values for the activity
3260     ' END.TIME "
3270     ' AVAIL$ as a bit string with '1' available. '
0' not avail
3280     ' AVAIL as a control code
3290     ' SET as a control code to set the time 'not a
available'
3300     '

```

```

3310 'returns:
3320 '      AVAIL as TRUE if time is available
3330 '      AVAIL$ updated if AVAIL and SET both TRUE
3340 '
3350 'subroutines used:
3360 '      all internal
3370 '
3380 GOSUB 3600
3390 GOSUB 3810
3400 RETURN
3410
3420
3430 '--- This routine determines the case of activity
3440 '      start (CASE1) and end (CASE2) relative to
3450 '      the period start and end...
3460 '
3470 '      CASE1 and CASE2 equal 1 if times are before
3480 '      the period starts, 2 if during the period,
3490 '      or 3 if after the period. Thus if CASE1 is
3500 '      3 or CASE2 is 1, the whole activity falls
3510 '      outside the period in question. If both
3520 '      CASE1 and CASE2 are 2, then the whole
3530 '      activity is within the period
3540 '
3550 '      CASE3 has a value of 1 if the whole
3560 '      activity falls on a single byte, 2 if on
3570 '      adjacent bytes, and 3 if one or more whole
3580 '      bytes fall between the start and end.
3590 '
3600 START.BIT = START.TIME\INCR
3610 START.BYTE = START.BIT\8 + 1
3620 END.BIT = (END.TIME-1)\INCR
3630 END.BYTE = END.BIT\8 + 1
3640 IF START.TIME >= PERIOD.ST.TIME THEN COND1 = -1 ELSE
COND1 = 0
3650 IF START.TIME < (PERIOD.ST.TIME + PERIOD.DUR) THEN C
OND2 = -1 ELSE COND2 = 0
3660 IF COND1 AND COND2 THEN CASE1 = 2 ELSE IF NOT COND1
THEN CASE1 = 1 ELSE IF NOT COND2 THEN CASE1 = 3
3670 IF END.TIME > PERIOD.ST.TIME THEN COND3 = -1 ELSE CO
ND3 = 0
3680 IF END.TIME <= (PERIOD.ST.TIME+PERIOD.DUR) THEN COND
4 = -1 ELSE COND4 = 0
3690 IF COND3 AND COND4 THEN CASE2 = 2 ELSE IF NOT COND3
THEN CASE2 = 1 ELSE IF NOT COND4 THEN CASE2 = 3
3700 IF END.BYTE = START.BYTE THEN CASE3 = 1
3710 IF END.BYTE - START.BYTE = 1 THEN CASE3 = 2
3720 IF END.BYTE - START.BYTE > 1 THEN CASE3 = 3
3730 RETURN
3740
3750
3760 ' This routine selects the proper routine for
3770 ' checking or setting availability based on the case
3780 ' defined by CASE1, CASE2, and CASE3
3790

```

AD-A133 271

COMPUTER ASSISTED SCHEDULING FOR AIR FORCE TACTICAL
FIGHTER SQUADRONS(U) ARMY COMMAND AND GENERAL STAFF
COLL FORT LEAVENWORTH KS B C DUGLE 03 JUN 83

2/2

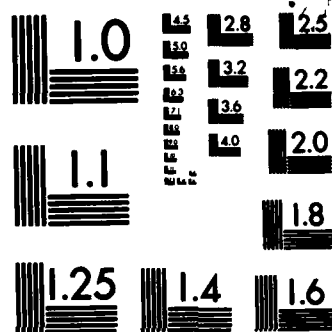
UNCLASSIFIED

SBI-AD-E750 845

F/G 12/1

NL

END



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

```

3800      if start is before period...
3810      IF CASE1=1 AND CASE2=2 THEN ON CASE3 GOSUB 3970,4070
,4070
3820      if start and end are during period...
3830      IF CASE1=2 AND CASE2=2 THEN ON CASE3 GOSUB 4200,4370
,4330
3840      if start is during period but end is after...

3850      IF CASE1=2 AND CASE2=3 THEN ON CASE3 GOSUB 4550,4680
,4680
3860      if start is before and end is after period...

3870      IF CASE1=1 AND CASE2=3 THEN FIRST.BYT=1: LAST.BYT=LE
N(AVAIL$): GOSUB 4720
3880      the final case ends before or starts after
3890      period...
3900      IF CASE1=3 OR CASE2=1 THEN PRINT"Activity is complet
ely outside the period..."
3910 RETURN

3920
3930
3940      --- This routine is used when END.BYTE$ is the
3950      first byte of AVAIL$...
3960
3970      FIRST.BIT.USED = 0: LAST.BIT.USED = (END.BIT MOD 8)
3980      BYT.TO.CK$ = LEFT$(AVAIL$,1)
3990      GOSUB 4940
4000      IF AVAIL AND SET THEN MID$(AVAIL$,1,1) = CHR$(ASC(BY
T.TO.CK$) AND (NOT MASK))
4010 RETURN

4020
4030
4040      --- This routine is used when END.BYTE points to
4050      end byte...
4060
4070      FIRST.BYT = 1: LAST.BYT = END.BYTE-1
4080      GOSUB 4820
4090      FIRST.BIT.USED = 0: LAST.BIT.USED = (END.BIT MOD 8)
4100      BYT.TO.CK$ = MID$(AVAIL$,END.BYTE,1): J = END.BYTE
4110      GOSUB 4940
4120      IF AVAIL AND SET THEN GOSUB 5060 ELSE RETURN
4130      MID$(AVAIL$,END.BYTE,1) = CHR$(ASC(BYT.TO.CK$) AND (
NOT MASK))
4140 RETURN

4150
4160
4170      --- This routine is used for the single byte case
4180      where one byte includes both start and end...
4190
4200      BYT.TO.CK$ = MID$(AVAIL$,START.BYTE,1)
4210      MASK = 0
4220      FIRST.BIT.USED = (START.BIT MOD 8)
4230      LAST.BIT.USED = (END.BIT MOD 8)
4240      GOSUB 4940
4250      IF AVAIL AND SET THEN MID$(AVAIL$,START.BYTE,1) = CH

```

```

R:(ASC(BYT TO CK:) AND (NOT MASK))
4260 RETURN
4270 '
4280 '
4290 ' --- This routine is used when one or more bytes
4300 '       separate the first and last bytes or when
4310 '       they are adjacent...
4320 '
4330 FIRST.BYT = START.BYTE+1: LAST.BYT = END.BYTE-1
4340 COSUB 4820
4350 IF NOT AVAIL THEN RETURN
4360 '       CASE3 = 2 enters here...
4370 FIRST.BIT.USED = (START.BIT MOD 8): LAST.BIT.USED =
7
4380 BYT TO CK: = MID$(AVAIL$, START.BYTE, 1)
4390 COSUB 4940
4400 IF AVAIL AND SET THEN MASK ST=MASK: BYT ST:=BYT TO C
K: ELSE IF NOT AVAIL THEN RETURN
4410 FIRST.BIT.USED = 0: LAST.BIT.USED = (END.BIT MOD 8)
4420 BYT TO CK: = MID$(AVAIL$, END.BYTE, 1): J = END.BYTE
4430 COSUB 4940
4440 IF NOT AVAIL THEN RETURN
4450 IF (CASE3=3) AND (AVAIL AND SET) THEN COSUB 5060
4460 IF NOT(AVAIL AND SET) THEN RETURN
4470 MID$(AVAIL$, START.BYTE, 1) = CHR$(ASC(BYT ST:) AND (N
OT MASK ST))
4480 MID$(AVAIL$, END.BYTE, 1) = CHR$(ASC(BYT TO CK:) AND (
NOT MASK))
4490 RETURN
4500 '
4510 '
4520 ' --- This routine is used when the last byte in the
4530 '       string is the only byte to be checked...
4540 '
4550 BYT TO CK: = MID$(AVAIL$, START.BYTE, 1)
4560 FIRST.BIT.USED = (START.BIT MOD 8): LAST.BIT.USED =
(PERIOD END.BIT MOD 8)
4570 COSUB 4940
4580 IF AVAIL AND SET THEN MID$(AVAIL$, START.BYTE, 1) = CH
R$(ASC(BYT TO CK:) AND (NOT MASK))
4590 RETURN
4600 '
4610 '
4620 ' --- This routine is used when the activity ends
4630 '       after the period and the first byte is one
4640 '       or more bytes from the end of AVAIL$. The
4650 '       last two cases of CASE3 are both checked by
4660 '       this routine...
4670 '
4680 FIRST.BIT.USED = (START.BIT MOD 8): LAST.BIT.USED =
7
4690 BYT TO CK: = MID$(AVAIL$, START.BYTE, 1)
4700 COSUB 4940
4710 FIRST.BYT = START.BYTE+1: LAST.BYT = LEN(AVAIL$)
4720 COSUB 4820

```

```

4730 IF AVAIL AND SET THEN GOSUB 5060 ELSE RETURN
4740 MID$(AVAIL$, START.BYTE, 1) = CHR$(ASC(BYT.TO.CK$) AND
(NOT MASK))
4750 RETURN
4760
4770
4780 --- This routine is used by the routines above when
4790 whole bytes are being checked for
4800 availability...
4810
4820 FOR J = FIRST.BYT TO LAST.BYT
4830 BYT.TO.CK$ = MID$(AVAIL$, J, 1)
4840 IF BYT.TO.CK$ (<) CHR$(255) THEN FIRST.BIT.US
ED=0: LAST.BIT.USED=7: GOSUB 4940
4850 IF NOT AVAIL THEN RETURN
4860 NEXT
4870 RETURN
4880
4890
4900 --- This routine is called by above routines to
4910 check availability within partial bytes of
4920 AVAIL$...
4930
4940 MASK = 0
4950 FOR K = FIRST.BIT.USED TO LAST.BIT.USED
4960 MASK = MASK + 2^K
4970 IF (ASC(BYT.TO.CK$) AND 2^K) = 0 THEN AVAIL
= 0: RETURN
4980 NEXT
4990 RETURN
5000
5010
5020 --- This routine is called when a whole byte is to
5030 be set to NOT AVAILABLE state, both AVAIL
5040 and SET are TRUE ...
5050
5060 FOR J = FIRST.BYT TO LAST.BYT
5070 MID$(AVAIL$, J, 1) = CHR$(0)
5080 NEXT
5090 RETURN
5100
5110
5120 this routine checks and sets NET and NLT times
5130 used for checking crew rest
5140
5150 DAY = START.TIME\1440 + 1
5160 IF DAY < 2 OR DAY > 6 THEN C.REST = -1: RETURN
5170
5180 IF START.TIME >= NET(DAY) OR NET(DAY) = -1 THEN ST.C
K = -1 ELSE ST.CK = 0
5190 IF END.TIME <= NLT(DAY) OR NLT(DAY) = -1 THEN END.CK
= -1 ELSE END.CK = 0
5200
5210 IF ST.CK AND END.CK THEN C.REST = -1 ELSE C.REST = 0
RETURN

```

```
5220  
5230     IF (START.TIME-720 < NLT(DAY-1)) OR (NLT(DAY-1) = -1  
    ) THEN NLT(DAY-1) = START.TIME - 720  
5240     IF (NLT(DAY) > START.TIME+720) OR (NLT(DAY) = -1) TH  
EN NLT(DAY) = START.TIME + 720  
5250     IF (NET(DAY) < END.TIME-720) OR (NET(DAY) = -1) THEN  
    NET(DAY) = END.TIME - 720  
5260     IF (NET(DAY+1) < END.TIME+720) OR (NET(DAY+1) = -1)  
THEN NET(DAY+1) = END.TIME + 720  
5270  
5280 RETURN
```

```

100  '*** SCHED.SET *****
110  'program dated 25 May 1983
120  '
130  'This program builds the bare schedule file from
140  '      shellnn.dat and wknn.dat
150  '
160  'variables required:
170  '      none
180  '
190  'returns:
200  '      SCHEDnn.xxx file
210  '
220  DEFINT A-Z
230  CLR$ = CHR$(26): DOWN$ = CHR$(10): ESC$ = CHR$(27)
240  MID_SCRN$ = CLR$ + STRING$(6,10)
250  UP$ = CHR$(11): MOV_LEFT$ = CHR$(8): MOV_RIGHT$ = CH
Rs(12)
260  HOME$ = CHR$(30): CLR_LINES = ESC$ + "T"
270  '
280  PERIOD.ST.TIME = 0: PERIOD.DUR = 10080: INCR = 30: M
AX.PIL.NUM = 60
290  DIM QUAL(MAX.PIL.NUM), AVAIL$(MAX.PIL.NUM), CUR_DT(M
AX.PIL.NUM,9)
300  DIM PILOT$(255), ACT_CUR_QUAL$(255)
310  ACT.CODE.USED$ = STRING$(32,0)
320  '
330  C$ = "Enter: 0 if correct, 1 to change it: "
340  E$ = "Error, enter 0 or 1 only, try again..."
350  '
360  PRINT MID_SCRN$: "Enter the week number:";
370  INPUT " ", WK.NUM$
380  WKDAT.FILES = "WK" + WK.NUM$ + ".DAT"
390  ON ERROR GOTO 410
400  OPEN "I", 1, WKDAT.FILES:
410  IF ERR = 53 THEN PRINT"No " WKDAT.FILES: " found, can
not continue. " ELSE CLOSE
420  ON ERROR GOTO 0
430  OPEN "R", 1, WKDAT.FILES, 93
440  FIELD#1, 1 AS N1$, 2 AS N2$, 9 AS N3$, 81 AS N4$
450  N = MAX.PIL.NUM + 1
460  GET#1, N
470  WK.DATE = CVI(N2$): WK.DATES = N3$
480  CLOSE
490  '
500  SCHED.NUM = 0
510  FIL.NAM.FOUND = 0
520  WHILE NOT FIL.NAM.FOUND
530      SCHED.NUM = SCHED.NUM + 1
540      SCHED.NUM$ = MID$(STR$(SCHED.NUM),2)
550      WHILE LEN(SCHED.NUM$) < 3
560          SCHED.NUM$ = "0" + SCHED.NUM$
570      WEND
580      FILENAME$ = "SCHED" + WK.NUM$ + "." + SCHED.NUM$
590      '
600  'check if file already exists

```

```

610      ON ERROR GOTO 630
620      OPEN "I", 1, FILENAME$
630      IF ERR = 53 THEN FIL.NAM.FOUND = -1: RESUME 650 EL
SE CLOSE#1
640      'no error indicates file was found and opened, try a
gain...
650      ON ERROR GOTO 0
660      WEND
670      PRINT"Using " FILENAME$ " for schedule data, dated "
WK.DATES
680      PRINT: PRINT C$;: A$ = INPUT$(1): PRINT A$
690      IF A$ = "1" THEN GOTO 360 ELSE IF A$ <> "0" THEN PRI
NT E$: GOTO 670
700      '
710      '_____schedule_data_____
720      '
730      OPEN "R", 1, FILENAME$, 58
740      '      seq.num: act.sched.time: act.code: pil.num:
st.time: end.time: act.name: pilots: cur.req: qual.req:
750      FIELD#1, 2 AS N11$, 2 AS N12$, 1 AS N13$, 1 AS N14$,
2 AS N15$, 2 AS N16$, 20 AS N17$, 8 AS N18$, 10 AS N19$, 10
AS N110$
760      FIELD#1, 58 AS N1A$
770      '
780      SHELL.FILE$ = "SHELL" + WK.NUM$ + ".DAT"
790      '
800      'open shell data file as #2...
810      OPEN "R", 2, SHELL.FILE$, 30
820      '      seq.num: act.sched.time: act.code: pil.num:
act.st.time: act.end.time: act.name:
830      FIELD#2, 2 AS N21$, 2 AS N22$, 1 AS N23$, 1 AS N24
$, 2 AS N25$, 2 AS N26$, 20 AS N27$
840      FIELD#2, 30 AS N2A$
850      '
860      M = 0: N = 0: END.FIL = 0: ACT.SCHED.TIME = 0: MAX.N
= 20
870      DIM ACT$(19)
880      '
890      'following string has bits 0 thru 59 'ON'
900      ALL.PILOT$ = STRING$(7,255) + CHR$(15)
910      '
920      'open act.def file as #3 for currency and qual
930      '      reqts...
940      OPEN "R", 3, "ACT.DEF", 46
950      FIELD#3, 2 AS N31$, 20 AS N32$, 2 AS N33$, 2 AS N34$
, 10 AS N35$, 10 AS N36$
960      '
970      PRINT"Getting shell data from " SHELL.FILE$ " and sa
ving in " FILENAME$
980      WHILE NOT END.FIL
990      'get shell records 20 at a time or until end
1000      '      found...
1010      WHILE (ACT.SCHED.TIME <> 32767) AND (N < MAX.N)
1020      N = N + 1

```

```

1030      GET#2, N
1040      'save temporarily in act$, make len equal to new
1050      'rec len by appending null chars...
1060      ACT$(N MOD 20) = N2A$ + STRING$(58 - LEN(N2A$), 0
1070      ACT.SCHED.TIME = CVI(N22$)
1080      WEND
1090      IF ACT.SCHED.TIME = 32767 THEN END.FIL = -1
1100      'save next 20 or all remaining act$ in schednn.xxx
1110      'file...
1120      WHILE M < N
1130          M = M + 1
1140          ACT.CODE = ASC(MID$(ACT$(M MOD 20), 5, 1))
1150          GET#3, ACT.CODE
1160          CUR.REQ$ = N35$: QUAL.REQ$ = N36$
1170          ACT.CUR.QUAL$(ACT.CODE) = CUR.REQ$ + QUAL.REQ$
1180
1190      'save the currency and qual strings with the record
1200      'and the act.code used bit for each act.code
1210      'in the schedule...
1220
1230      MID$(ACT$(M MOD 20), 39, 20) = ACT.CUR.QUAL$(ACT.C
1240      ODE)
1250      BYTE = ACT.CODE \ 8 + 1: BIT = (ACT.CODE - 1) MOD
1260      8
1270      BYT$ = MID$(ACT.CODE.USED$, BYTE, 1)
1280      IF (ASC(BYT$) AND 2*BIT) <> 2*BIT THEN MID$(ACT.
1290      CODE.USED$, BYTE, 1) = CHR$(ASC(BYT$) + 2*BIT)
1300
1310      LSET N1A$ = ACT$(M MOD 20)
1320      PUT#1, M
1330      WEND
1340      MAX.N = MAX.N + 20
1350      WEND
1360      MAX.N = N
1370      CLOSE #2, #3
1380      ERASE ACT$
1390      PRINT "Transfer completed"
1400
1410      'schednn.xxx now has a record for each activity
1420      'found in the schedule shell...
1430
1440      'open weekly data file as #3 and get avail$ and
1450      'qual data in memory...
1460      PRINT "Getting AVAIL$ and QUAL data from " WKDAT.FILE
1470      $
1480      OPEN "R", 3, WKDAT.FILE$, 93
1490      '
1500      '
1510      '
1520      '
1530      '
1540      '
1550      '
1560      '
1570      '
1580      '
1590      '
1600      '
1610      '
1620      '
1630      '
1640      '
1650      '
1660      '
1670      '
1680      '
1690      '
1700      '
1710      '
1720      '
1730      '
1740      '
1750      '
1760      '
1770      '
1780      '
1790      '
1800      '
1810      '
1820      '
1830      '
1840      '
1850      '
1860      '
1870      '
1880      '
1890      '
1900      '
1910      '
1920      '
1930      '
1940      '
1950      '
1960      '
1970      '
1980      '
1990      '
2000      '
2010      '
2020      '
2030      '
2040      '
2050      '
2060      '
2070      '
2080      '
2090      '
2100      '
2110      '
2120      '
2130      '
2140      '
2150      '
2160      '
2170      '
2180      '
2190      '
2200      '
2210      '
2220      '
2230      '
2240      '
2250      '
2260      '
2270      '
2280      '
2290      '
2300      '
2310      '
2320      '
2330      '
2340      '
2350      '
2360      '
2370      '
2380      '
2390      '
2400      '
2410      '
2420      '
2430      '
2440      '
2450      '
2460      '
2470      '
2480      '
2490      '
2500      '
2510      '
2520      '
2530      '
2540      '
2550      '
2560      '
2570      '
2580      '
2590      '
2600      '
2610      '
2620      '
2630      '
2640      '
2650      '
2660      '
2670      '
2680      '
2690      '
2700      '
2710      '
2720      '
2730      '
2740      '
2750      '
2760      '
2770      '
2780      '
2790      '
2800      '
2810      '
2820      '
2830      '
2840      '
2850      '
2860      '
2870      '
2880      '
2890      '
2900      '
2910      '
2920      '
2930      '
2940      '
2950      '
2960      '
2970      '
2980      '
2990      '
3000      '
3010      '
3020      '
3030      '
3040      '
3050      '
3060      '
3070      '
3080      '
3090      '
3100      '
3110      '
3120      '
3130      '
3140      '
3150      '
3160      '
3170      '
3180      '
3190      '
3200      '
3210      '
3220      '
3230      '
3240      '
3250      '
3260      '
3270      '
3280      '
3290      '
3300      '
3310      '
3320      '
3330      '
3340      '
3350      '
3360      '
3370      '
3380      '
3390      '
3400      '
3410      '
3420      '
3430      '
3440      '
3450      '
3460      '
3470      '
3480      '
3490      '
3500      '
3510      '
3520      '
3530      '
3540      '
3550      '
3560      '
3570      '
3580      '
3590      '
3600      '
3610      '
3620      '
3630      '
3640      '
3650      '
3660      '
3670      '
3680      '
3690      '
3700      '
3710      '
3720      '
3730      '
3740      '
3750      '
3760      '
3770      '
3780      '
3790      '
3800      '
3810      '
3820      '
3830      '
3840      '
3850      '
3860      '
3870      '
3880      '
3890      '
3900      '
3910      '
3920      '
3930      '
3940      '
3950      '
3960      '
3970      '
3980      '
3990      '
4000      '
4010      '
4020      '
4030      '
4040      '
4050      '
4060      '
4070      '
4080      '
4090      '
4100      '
4110      '
4120      '
4130      '
4140      '
4150      '
4160      '
4170      '
4180      '
4190      '
4200      '
4210      '
4220      '
4230      '
4240      '
4250      '
4260      '
4270      '
4280      '
4290      '
4300      '
4310      '
4320      '
4330      '
4340      '
4350      '
4360      '
4370      '
4380      '
4390      '
4400      '
4410      '
4420      '
4430      '
4440      '
4450      '
4460      '
4470      '
4480      '
4490      '
4500      '
4510      '
4520      '
4530      '
4540      '
4550      '
4560      '
4570      '
4580      '
4590      '
4600      '
4610      '
4620      '
4630      '
4640      '
4650      '
4660      '
4670      '
4680      '
4690      '
4700      '
4710      '
4720      '
4730      '
4740      '
4750      '
4760      '
4770      '
4780      '
4790      '
4800      '
4810      '
4820      '
4830      '
4840      '
4850      '
4860      '
4870      '
4880      '
4890      '
4900      '
4910      '
4920      '
4930      '
4940      '
4950      '
4960      '
4970      '
4980      '
4990      '
5000      '
5010      '
5020      '
5030      '
5040      '
5050      '
5060      '
5070      '
5080      '
5090      '
5100      '
5110      '
5120      '
5130      '
5140      '
5150      '
5160      '
5170      '
5180      '
5190      '
5200      '
5210      '
5220      '
5230      '
5240      '
5250      '
5260      '
5270      '
5280      '
5290      '
5300      '
5310      '
5320      '
5330      '
5340      '
5350      '
5360      '
5370      '
5380      '
5390      '
5400      '
5410      '
5420      '
5430      '
5440      '
5450      '
5460      '
5470      '
5480      '
5490      '
5500      '
5510      '
5520      '
5530      '
5540      '
5550      '
5560      '
5570      '
5580      '
5590      '
5600      '
5610      '
5620      '
5630      '
5640      '
5650      '
5660      '
5670      '
5680      '
5690      '
5700      '
5710      '
5720      '
5730      '
5740      '
5750      '
5760      '
5770      '
5780      '
5790      '
5800      '
5810      '
5820      '
5830      '
5840      '
5850      '
5860      '
5870      '
5880      '
5890      '
5900      '
5910      '
5920      '
5930      '
5940      '
5950      '
5960      '
5970      '
5980      '
5990      '
6000      '
6010      '
6020      '
6030      '
6040      '
6050      '
6060      '
6070      '
6080      '
6090      '
6100      '
6110      '
6120      '
6130      '
6140      '
6150      '
6160      '
6170      '
6180      '
6190      '
6200      '
6210      '
6220      '
6230      '
6240      '
6250      '
6260      '
6270      '
6280      '
6290      '
6300      '
6310      '
6320      '
6330      '
6340      '
6350      '
6360      '
6370      '
6380      '
6390      '
6400      '
6410      '
6420      '
6430      '
6440      '
6450      '
6460      '
6470      '
6480      '
6490      '
6500      '
6510      '
6520      '
6530      '
6540      '
6550      '
6560      '
6570      '
6580      '
6590      '
6600      '
6610      '
6620      '
6630      '
6640      '
6650      '
6660      '
6670      '
6680      '
6690      '
6700      '
6710      '
6720      '
6730      '
6740      '
6750      '
6760      '
6770      '
6780      '
6790      '
6800      '
6810      '
6820      '
6830      '
6840      '
6850      '
6860      '
6870      '
6880      '
6890      '
6900      '
6910      '
6920      '
6930      '
6940      '
6950      '
6960      '
6970      '
6980      '
6990      '
7000      '
7010      '
7020      '
7030      '
7040      '
7050      '
7060      '
7070      '
7080      '
7090      '
7100      '
7110      '
7120      '
7130      '
7140      '
7150      '
7160      '
7170      '
7180      '
7190      '
7200      '
7210      '
7220      '
7230      '
7240      '
7250      '
7260      '
7270      '
7280      '
7290      '
7300      '
7310      '
7320      '
7330      '
7340      '
7350      '
7360      '
7370      '
7380      '
7390      '
7400      '
7410      '
7420      '
7430      '
7440      '
7450      '
7460      '
7470      '
7480      '
7490      '
7500      '
7510      '
7520      '
7530      '
7540      '
7550      '
7560      '
7570      '
7580      '
7590      '
7600      '
7610      '
7620      '
7630      '
7640      '
7650      '
7660      '
7670      '
7680      '
7690      '
7700      '
7710      '
7720      '
7730      '
7740      '
7750      '
7760      '
7770      '
7780      '
7790      '
7800      '
7810      '
7820      '
7830      '
7840      '
7850      '
7860      '
7870      '
7880      '
7890      '
7900      '
7910      '
7920      '
7930      '
7940      '
7950      '
7960      '
7970      '
7980      '
7990      '
8000      '
8010      '
8020      '
8030      '
8040      '
8050      '
8060      '
8070      '
8080      '
8090      '
8100      '
8110      '
8120      '
8130      '
8140      '
8150      '
8160      '
8170      '
8180      '
8190      '
8200      '
8210      '
8220      '
8230      '
8240      '
8250      '
8260      '
8270      '
8280      '
8290      '
8300      '
8310      '
8320      '
8330      '
8340      '
8350      '
8360      '
8370      '
8380      '
8390      '
8400      '
8410      '
8420      '
8430      '
8440      '
8450      '
8460      '
8470      '
8480      '
8490      '
8500      '
8510      '
8520      '
8530      '
8540      '
8550      '
8560      '
8570      '
8580      '
8590      '
8600      '
8610      '
8620      '
8630      '
8640      '
8650      '
8660      '
8670      '
8680      '
8690      '
8700      '
8710      '
8720      '
8730      '
8740      '
8750      '
8760      '
8770      '
8780      '
8790      '
8800      '
8810      '
8820      '
8830      '
8840      '
8850      '
8860      '
8870      '
8880      '
8890      '
8900      '
8910      '
8920      '
8930      '
8940      '
8950      '
8960      '
8970      '
8980      '
8990      '
9000      '
9010      '
9020      '
9030      '
9040      '
9050      '
9060      '
9070      '
9080      '
9090      '
9100      '
9110      '
9120      '
9130      '
9140      '
9150      '
9160      '
9170      '
9180      '
9190      '
9200      '
9210      '
9220      '
9230      '
9240      '
9250      '
9260      '
9270      '
9280      '
9290      '
9300      '
9310      '
9320      '
9330      '
9340      '
9350      '
9360      '
9370      '
9380      '
9390      '
9400      '
9410      '
9420      '
9430      '
9440      '
9450      '
9460      '
9470      '
9480      '
9490      '
9500      '
9510      '
9520      '
9530      '
9540      '
9550      '
9560      '
9570      '
9580      '
9590      '
9600      '
9610      '
9620      '
9630      '
9640      '
9650      '
9660      '
9670      '
9680      '
9690      '
9700      '
9710      '
9720      '
9730      '
9740      '
9750      '
9760      '
9770      '
9780      '
9790      '
9800      '
9810      '
9820      '
9830      '
9840      '
9850      '
9860      '
9870      '
9880      '
9890      '
9900      '
9910      '
9920      '
9930      '
9940      '
9950      '
9960      '
9970      '
9980      '
9990      '

```

```

1510 FIELD#2, 93 AS N2A$
1520 '
1530 FOR P = 1 TO MAX.PIL.NUM
1540 GET#3, P
1550 IF P (>) ASC(N31$) THEN PRINT"Error in " WKDAT.FILE
$ ", record" P "<> to pilot number" ASC(N31$)
1560 AVAIL$(P) = N32$
1570 ' FOR Q = 0 TO 9
1580 ' CUR.DT(P,Q) = CVI(MID$(N33$,Q*2+1,2))
1590 ' NEXT
1600 QUAL(P) = CVI(N34$)
1610 'save in individual week's data file...
1620 TMP$ = N3A$: LSET N2A$ = TMP$
1630 PUT#2, P
1640 NEXT
1650 'save final record - date data...
1660 GET#3, P
1670 TMP$ = N3A$: LSET N2A$ = TMP$
1680 PUT#2, P
1690 CLOSE #2, #3
1700 '
1710 'check each act.code (if used) then evaluate each
1720 ' pilot for qualification and save in pilots...
1730 FOR I = 1 TO 254
1740 BYTE = I\8 + 1: BIT = (I-1) MOD 8
1750 BYT$ = MID$(ACT.CODE.USED$,BYTE,1)
1760 'set the pilots bits on only for qualified pilots...
1770 IF (ASC(BYT$) AND 2*BIT) = 2*BIT THEN GOSUB 2060:
PILOT$(I) = PILOT$
1780 NEXT
1790 '
1800 'now have pilots qualified for each activity saved
1810 ' in pilots(act.code); next, determine which
1820 ' qualified pilots are also available, then
1830 ' save with the activity record in schednn.xxx
1840 '
1850 FOR N = 1 TO MAX.N - 1
1860 PRINT"Getting AVAIL$ data for sched sequence
number" N
1870 GET#1, N
1880 ACT.CODE = ASC(N13$)
1890 IF ACT.CODE > 128 THEN PILOT$ = PILOT$(ACT.C
ODE) ELSE PILOT$ = N18$
1900 ACT.ST.TIME = CVI(N15$): ACT.END.TIME = CVI(
N16$)
1910 FOR I = 1 TO MAX.PIL.NUM
1920 AVAIL$ = AVAIL$(I)
1930 BYT$ = MID$(PILOT$,I\8+1,1): BIT = (I-1) M
OD 8
1940 IF (ASC(BYT$) AND 2*BIT) = 2*BIT THEN GOSU
B 2230
1950 NEXT
1960 LSET N18$ = PILOT$
1970 PUT#1, N
1980 NEXT

```

```

1990 PRINT "Schedule file completed": END
2000
2010 '_____subroutines_____
2020
2030 'this routine compares the qual reqd values to the
2040 '      pilot qual values and sets the pilots bit
2050 '      on if a qual match is found...
2060 PILOT$ = STRING$(8,0)
2070 PRINT "Checking pilot qualifications for activity cod
e" I
2080 FOR P = 1 TO MAX.PIL.NUM
2090     J = 0: QUAL.FOUND = 0
2100     WHILE QUAL.FOUND = 0 AND J < 5
2110         J = J + 1
2120         QUAL.REQ = CVI(MID$(ACT.CUR.QUAL$(I),J*2+9,2))
2130         IF (QUAL(P) AND QUAL.REQ) = QUAL.REQ THEN QUAL.F
OUND = -1
2140     WEND
2150     BYTE = P\8+1: BYT$ = MID$(PILOT$,BYTE,1): BIT = (P
-1) MOD 8
2160     IF QUAL.FOUND THEN MID$(PILOT$,BYTE,1) = CHR$(ASC(
BYT$) + 2*BIT): PRINT "Pilot" P "qual" ELSE PRINT "Pilot" P "n
ot qual"
2170 NEXT
2180 FOR H = 1 TO 8: PRINT ASC(MID$(PILOT$,H,1)): NEXT:
PRINT
2190 RETURN
2200
2210 'this routine checks a pilot for availability and
2220 '      turns off the pilots bit if not available...
2230 START.TIME = ACT.ST.TIME
2240 END.TIME = ACT.END.TIME
2250 AVAIL = -1: SET = 0
2260 GOSUB 2760
2270 PRINT "Pilot" I;
2280 IF NOT AVAIL THEN MID$(PILOT$,I\8+1,1) = CHR$(ASC(BY
T$) - 2*BIT): PRINT "Not avail" ELSE PRINT "Avail"
2290 RETURN
2300
2310 'trim trailing spaces...
2320 L = LEN(TRIM$) + 1: L.CHR$ = CHR$(0)
2330 WHILE ASC(L.CHR$) < 33
2340     L = L - 1
2350     L.CHR$ = MID$(TRIM$,L,1)
2360 WEND
2370 TRIM$ = LEFT$(TRIM$,L)
2380 RETURN
2390
2400 'dynamic array size increase...
2410 DIM TMP$(MAX.N)
2420 FOR M = 1 TO MAX.N: TMP$(M) = ACT$(M): NEXT
2430 ERASE ACT$: DIM ACT$(MAX.N + 10)
2440 FOR M = 1 TO MAX.N: ACT$(M) = TMP$(M): NEXT
2450 MAX.N = MAX.N + 10
2460 ERASE TMP$

```

```

2470      ON ERROR GOTO 0
2480      RETURN
2490
2500
2510
2520      *** CASE ***
2530      'module dated 24 April 1983
2540
2550      'This module includes subroutines called by other
2560      '      modules in determining the case of each
2570      '      activity relative to the week
2580
2590      'variables required:
2600      '      PERIOD.ST.TIME as an integer in minutes or o
ther time units
2610      '      PERIOD.DUR as an integer length of period
2620      '      INCR as an integer for the value of each bit
      (resolution)
2630      '      START.TIME as values for the activity
2640      '      END.TIME
2650      '      AVAIL$ as a bit string with '1' available, '
0' not avail
2660      '      AVAIL as a control code
2670      '      SET as a control code to set the time 'not a
available'
2680
2690      'returns:
2700      '      AVAIL as TRUE if time is available
2710      '      AVAIL$ updated if AVAIL and SET both TRUE
2720
2730      'subroutines used:
2740      '      all internal
2750
2760      GOSUB 2980
2770      GOSUB 3190
2780      RETURN
2790
2800
2810      --- This routine determines the case of activity
2820      '      start (CASE1) and end (CASE2) relative to
2830      '      the period start and end...
2840
2850      '      CASE1 and CASE2 equal 1 if times are before
2860      '      the period starts, 2 if during the period,
2870      '      or 3 if after the period. Thus if CASE1 is
2880      '      3 or CASE2 is 1, the whole activity falls
2890      '      outside the period in question. If both
2900      '      CASE1 and CASE2 are 2, then the whole
2910      '      activity is within the period.
2920
2930      '      CASE3 has a value of 1 if the whole
2940      '      activity falls on a single byte, 2 if on
2950      '      adjacent bytes and 3 if one or more whole
2960      '      bytes 'all between the start and end.
2970

```

```

2980     START.BIT = START.TIME\INCR
2990     START.BYTE = START.BIT\8 + 1
3000     END.BIT = (END.TIME-1)\INCR
3010     END.BYTE = END.BIT\8 + 1
3020     IF START.TIME >= PERIOD.ST.TIME THEN COND1 = -1 ELSE
COND1 = 0
3030     IF START.TIME < (PERIOD.ST.TIME + PERIOD.DUR) THEN C
OND2 = -1 ELSE COND2 = 0
3040     IF COND1 AND COND2 THEN CASE1 = 2 ELSE IF NOT COND1
THEN CASE1 = 1 ELSE IF NOT COND2 THEN CASE1 = 3
3050     IF END.TIME > PERIOD.ST.TIME THEN COND3 = -1 ELSE CO
ND3 = 0
3060     IF END.TIME <= (PERIOD.ST.TIME+PERIOD.DUR) THEN COND
4 = -1 ELSE COND4 = 0
3070     IF COND3 AND COND4 THEN CASE2 = 2 ELSE IF NOT COND3
THEN CASE2 = 1 ELSE IF NOT COND4 THEN CASE2 = 3
3080     IF END.BYTE = START.BYTE THEN CASE3 = 1
3090     IF END.BYTE - START.BYTE = 1 THEN CASE3 = 2
3100     IF END.BYTE - START.BYTE > 1 THEN CASE3 = 3
3110 RETURN

```

```

3120 '
3130 '
3140 ' This routine selects the proper routine for
3150 ' checking or setting availability based on
3160 ' the case defined by CASE1, CASE2, and CASE3
3170 '
3180 ' if start is before period...
3190 IF CASE1=1 AND CASE2=2 THEN ON CASE3 GOSUB 3340,3440
,3440
3200 ' if start and end are during period...
3210 IF CASE1=2 AND CASE2=2 THEN ON CASE3 GOSUB 3570,3740
,3700
3220 ' if start is during period but end is after...
3230 IF CASE1=2 AND CASE2=3 THEN ON CASE3 GOSUB 3920,4050
,4050
3240 ' if start is before and end is after period...
3250 IF CASE1=1 AND CASE2=3 THEN FIRST.BYT=1: LAST.BYT=LE
N(AVAIL$): GOSUB 4090
3260 ' the final case ends before or starts after p
eriod...
3270 IF CASE1=3 OR CASE2=1 THEN PRINT"Activity is complet
ely outside the period..."
3280 RETURN

```

```

3290 '
3300 '
3310 ' --- This routine is used when END.BYTE$ is the
3320 ' first byte of AVAIL$.
3330 '
3340 FIRST.BIT.USED = 0: LAST.BIT.USED = (END.BIT MOD 8)
3350 BYT.TO.CK$ = LEFT$(AVAIL$,1)
3360 GOSUB 4310
3370 IF AVAIL AND SET THEN MID$(AVAIL$,1,1) = CHR$(ASC(BY
T.TO.CK$) AND (NOT MASK))

```

```

3380 RETURN
3390
3400
3410 ' --- This routine is used when END.BYTE points to
3420 '       end byte...
3430
3440 FIRST.BYT = 1: LAST.BYT = END.BYTE-1
3450 GOSUB 4190
3460 FIRST.BIT.USED = 0: LAST.BIT.USED = (END.BIT MOD 8)
3470 BYT.TO.CK$ = MID$(AVAIL$,END.BYTE,1): J = END.BYTE
3480 GOSUB 4310
3490 IF AVAIL AND SET THEN GOSUB 4430 ELSE RETURN
3500 MID$(AVAIL$,END.BYTE,1) = CHR$(ASC(BYT.TO.CK$) AND (
NOT MASK))
3510 RETURN
3520
3530
3540 ' --- This routine is used for the single byte case
3550 '       where one byte includes both start and end...
3560
3570 BYT.TO.CK$ = MID$(AVAIL$,START.BYTE,1)
3580 MASK = 0
3590 FIRST.BIT.USED = (START.BIT MOD 8)
3600 LAST.BIT.USED = (END.BIT MOD 8)
3610 GOSUB 4310
3620 IF AVAIL AND SET THEN MID$(AVAIL$,START.BYTE,1) = CH
R$(ASC(BYT.TO.CK$) AND (NOT MASK))
3630 RETURN
3640
3650
3660 ' --- This routine is used when one or more bytes
3670 '       separate the first and last bytes or when
3680 '       they are adjacent...
3690
3700 FIRST.BYT = START.BYTE+1: LAST.BYT = END.BYTE-1
3710 GOSUB 4190
3720 IF NOT AVAIL THEN RETURN
3730 ' CASE3 = 2 enters here...
3740 FIRST.BIT.USED = (START.BIT MOD 8): LAST.BIT.USED =
7
3750 BYT.TO.CK$ = MID$(AVAIL$,START.BYTE,1)
3760 GOSUB 4310
3770 IF AVAIL AND SET THEN MASK.ST=MASK: BYT.ST$=BYT.TO.C
K$ ELSE IF NOT AVAIL THEN RETURN
3780 FIRST.BIT.USED = 0: LAST.BIT.USED = (END.BIT MOD 8)
3790 BYT.TO.CK$ = MID$(AVAIL$,END.BYTE,1): J = END.BYTE
3800 GOSUB 4310
3810 IF NOT AVAIL THEN RETURN
3820 IF (CASE3=3) AND (AVAIL AND SET) THEN GOSUB 4430
3830 IF NOT(AVAIL AND SET) THEN RETURN
3840 MID$(AVAIL$,START.BYTE,1) = CHR$(ASC(BYT.ST$) AND (N
OT MASK.ST))
3850 MID$(AVAIL$,END.BYTE,1) = CHR$(ASC(BYT.TO.CK$) AND (
NOT MASK))
3860 RETURN

```

```

3870 '
3880 '
3890 ' --- This routine is used when the last byte in the
3900 '       string is the only byte to be checked...
3910 '
3920     BYT.TO.CK$ = MID$(AVAIL$,START.BYTE,1)
3930     FIRST.BIT.USED = (START.BIT MOD 8): LAST.BIT.USED =
(PERIOD.END.BIT MOD 8)
3940     GOSUB 4310
3950     IF AVAIL AND SET THEN MID$(AVAIL$,START.BYTE,1) = CH
R$(ASC(BYT.TO.CK$) AND (NOT MASK))
3960     RETURN
3970 '
3980 '
3990 ' --- This routine is used when the activity ends
4000 '       after the period and the first byte is one
4010 '       or more bytes from the end of AVAIL$. The
4020 '       last two cases of CASE3 are both checked by
4030 '       this routine...
4040 '
4050     FIRST.BIT.USED = (START.BIT MOD 8): LAST.BIT.USED =
7
4060     BYT.TO.CK$ = MID$(AVAIL$,START.BYTE,1)
4070     GOSUB 4310
4080     FIRST.BYT = START.BYTE+1: LAST.BYT = LEN(AVAIL$)
4090     GOSUB 4190
4100     IF AVAIL AND SET THEN GOSUB 4430 ELSE RETURN
4110     MID$(AVAIL$,START.BYTE,1) = CHR$(ASC(BYT.TO.CK$) AND
(NOT MASK))
4120     RETURN
4130 '
4140 '
4150 ' --- This routine is used by the routines above when
4160 '       whole bytes are being checked for
4170 '       availability...
4180 '
4190     FOR J = FIRST.BYT TO LAST.BYT
4200         BYT.TO.CK$ = MID$(AVAIL$,J,1)
4210         IF BYT.TO.CK$ (>) CHR$(255) THEN FIRST.BIT.US
ED=0: LAST.BIT.USED=7: GOSUB 4310
4220         IF NOT AVAIL THEN RETURN
4230     NEXT
4240     RETURN
4250 '
4260 '
4270 ' --- This routine is called by above routines to
4280 '       check availability within partial bytes of
4290 '       AVAIL$...
4300 '
4310     MASK = 0
4320     FOR K = FIRST.BIT.USED TO LAST.BIT.USED
4330         MASK = MASK + 2^K
4340         IF (ASC(BYT.TO.CK$) AND 2^K) = 0 THEN AVAIL
= 0: RETURN
4350     NEXT

```

```
4360 RETURN
4370 '
4380 '
4390 ' --- This routine is called when a whole byte is to
4400 '         be set to NOT AVAILABLE state, both AVAIL
4410 '         and SET are TRUE...
4420 '
4430     FOR J = FIRST.BYT TO LAST.BYT
4440         MIDS(AVAIL$,J,1) = CHR$(0)
4450     NEXT
4460 RETURN
4470 '


---


```

```

100  '*** WKSCHED SET *****
110  'program dated 24 May 1983
120
130  'This program builds the final schedule file from
140  '      schednn.xxx and wknn.xxx
150
160  'variables required:
170  '      none
180
190  'returns:
200  '      SCHEDnn.DAT file when completed
210
220  DEFINT A-Z
230  CLR$ = CHR$(26): DOWN$ = CHR$(10): ESC$ = CHR$(27):
CR$ = CHR$(13)
240  MID.SCRN$ = CLR$ + STRING$(6,10)
250  UP$ = CHR$(11): MOV.LEFT$ = CHR$(8): MOV.RIGHT$ = CH
R$(12)
260  HOME$ = CHR$(30): CLR.LINE$ = ESC$ + "T"
270
280  PERIOD.ST.TIME = 0: PERIOD.DUR = 10080: INCR = 30: M
AX.PIL.NUM = 60
290  DIM QUAL(MAX.PIL.NUM), AVAIL$(MAX.PIL.NUM), PIL.NAM$
(MAX.PIL.NUM + 1)
300  DIM PILOT$(254), CUR.NAM$(15), EXP.DUR(15), INXP.DUR
(15), EVENT.NUM(15)
310  DIM NET(7), NLT(7)
320  ACT.CODE.USED$ = STRING$(16,0)
330  DAYS$ = "SunMonTueWedThuFriSat"
340  MONTH$ = "JANFEBMARAPRMayJUNJULAUGSEP OCTNOVDECJA
N"
350  FIRST.DAYS$ = "00103206009112115218221324427430533536
6"
360
370  C$ = "Enter: 0 if correct, 1 to change it: "
380  E$ = "Error, enter 0 or 1 only, try again..."
390
400  PRINT MID.SCRN$: "Enter the week number: "; INPUT " ",
WK.NUM$
410  FILENAME$ = "SCHED" + WK.NUM$ + ".*"
420  PRINT "Schedule files for week " WK.NUM$: PRINT
430  FILES FILENAME$: PRINT DOWN$
440  PRINT "Enter the schedule file number or 0 to start a
gain: "; INPUT " ", SCHED.NUM$
450  IF SCHED.NUM$ = "0" THEN GOTO 400
460  WHILE LEN(SCHED.NUM$) < 3
470  SCHED.NUM$ = "0" + SCHED.NUM$
480  WEND
490  FILENAME$ = "SCHED" + WK.NUM$ + "." + SCHED.NUM$
500
510  PRINT MID.SCRN$: "Using " FILENAME$ " for schedule da
ta."
520  PRINT C$: A$ = INPUT$(1): PRINT A$
530  IF A$ = "1" THEN GOTO 400 ELSE IF A$ <> "0" THEN PRI
NT E$: GOTO 520

```

```

540
550   _____schedule_data_____
560
570   OPEN "R", 1, FILENAME$, 58
580       seq.num: act.sched.time: act.code: pil.num:
    st.time:   end.time:   act.name:   pilots:   cur.req:   qu
al.req:
590       FIELD#1, 2 AS N11$, 2 AS N12$, 1 AS N13$, 1 AS N14$,
    2 AS N15$, 2 AS N16$, 20 AS N17$, 8 AS N18$, 10 AS N19$, 10
    AS N110$
600       FIELD#1, 58 AS N1A$
610
620   PRINT MID.SCRN$ "Getting date data. . ."
630   WKDAT.FILE$ = "WK" + WK.NUM$ + "." + SCHED.NUM$
640   OPEN "R", 2, WKDAT.FILE$, 93
650   FIELD#2, 1 AS N21$, 2 AS N22$, 9 AS N23$, 81 AS N24$
660   GET#2, MAX.PIL.NUM + 1
670   IF ASC(N21$) <> MAX.PIL.NUM + 1 THEN PRINT"File acce
ss error in " WKDAT.FILE$
680   WK.DATE = CVI(N22$): WK.DATES = N23$
690   IF (WK.DATE MOD 1000)\7 <> VAL(WK.NUM$) THEN PRINT"E
rror: week number " WK.NUM$ " does not agree with file " WKD
AT.FILE$ ",
dated " WK.DATES
700   CLOSE #2
710
720   PRINT MID.SCRN$ "Getting currency names from file. . ."
730   OPEN "R", 2, "CUR.DEF", 28
740       cur.num:   cur.name$:   exp.dur:   inxp.dur:
    event.num:
750       FIELD#2, 2 AS N21$, 20 AS N22$, 2 AS N23$, 2 AS N24$
    , 2 AS N25$
760       FOR I = 1 TO 15
770           GET#2, I
780           TRIM$ = N22$: GOSUB 3510: CUR.NAME$(I) = TRIM
$
790           EXP.DUR(I) = CVI(N23$)
800           INXP.DUR(I) = CVI(N24$)
810           EVENT.NUM(I) = CVI(N25$)
820       NEXT
830       CLOSE #2
840
850   PRINT MID.SCRN$ "Getting pilot names from file. . ."
860   OPEN "R", 2, "PILNAM.DEF", 27
870       pil.num:   l.name$:   init$:   rank$:
880       FIELD#2, 2 AS N21$, 20 AS N22$, 2 AS N23$, 3 AS N24$
890       FOR I = 1 TO MAX.PIL.NUM
900           GET#2, I
910           TRIM$ = N22$
920           IF TRIM$ = "Not in use" THEN GOTO
930 ELSE GOSUB 3510: PIL.NAME$(I) = TRIM$
930           PIL.NAME$(I) = N24$ + " " + PIL.NAME$(I) + "
" + N23$
940           PRINT CR$ STRING$(33,32) CR$ USING "##"; I;

```

```

PRINT " - " FIL.NAM$(I);
950 NEXT
960 FIL.NAM$(MAX.FIL.NUM+1) = "None"
970 CLOSE #2
980 '
990 N = 0: ACT.SCHED.TIME = 0
1000 PRINT MID.SCRN$ "Getting length of " FILENAME$
1010 WHILE ACT.SCHED.TIME <> 32767
1020     N = N + 1
1030     GET#1, N
1040     ACT.SCHED.TIME = CVI(N12$)
1050 WEND
1060 MAX.N = N: DIM ACT$(MAX.N)
1070 PRINT"Reading " FILENAME$ " into memory"
1080 FOR N = 1 TO MAX.N
1090     GET#1, N
1100     ACT$(N) = N1A$
1110 NEXT
1120 '
1130 PRINT MID.SCRN$ FILENAME$ " data now in memory..."
1140 '
1150 'open data file...
1160 OPEN "R", 2, WKDAT.FILE$, 93
1170 FIELD#2, 1 AS N221$, 42 AS N222$, 20 AS N223$, 2 AS
N224$, 28 AS N225$
1180 '
1190 PRINT
1200 PRINT" Enter:"
1210 PRINT"          0 to quit"
1220 PRINT"          1 to fill schedule in sequence number
r order"
1230 PRINT"          2 to fill individual sequence number
activities"
1240 PRINT"          3 to fill by activity number"
1250 PRINT" Which choice? ";
1260 SEL = VAL(INPUT$(1))
1270 IF SEL = 0 THEN GOTO 1310 ELSE IF SEL > 3 THEN PRINT
"Error, enter 0 to 3 only, try again...": GOTO 1250
1280 ON SEL GOSUB 1400, 1430, 1500
1290 PRINT MID.SCRN$: GOTO 1190
1300 '
1310 FOR N = 1 TO MAX.N
1320     LSET N1A$ = ACT$(N)
1330     PUT#1, N
1340 NEXT
1350 PRINT MID.SCRN$ "Schedule data saved in " FILENAME$
" and " WKDAT.FILE$
1360 '
1370 '_____control_subroutines_____
1380 '
1390 'step thru in sequence number order...
1400 PRINT"Not written yet...": DUMMY$ = INPUT$(1)
1410 RETURN
1420 'one seq num at a time from keyboard...
1430 PRINT MID.SCRN$ "What is the sequence number?";

```

```

1440 INPUT " ", SEQ.NUM
1450 IF SEQ.NUM <= 0 THEN GOTO 1480 ELSE IF SEQ.NUM > MAX
.N THEN PRINT "Error: sequence number too big, enter a number
from 0 (to
quit) to" MAX.N: PRINT "Try again:": INPUT " ", SEQ.NUM: GOTO
1450
1460 GOSUB 1560
1470 PRINT "Enter 0 to quit, sequence number to display, an
other schedule activity:": GOTO 1440
1480 RETURN
1490 'by activity number...
1500 PRINT "Not written yet...": DUMMY$ = INPUT$(1)
1510 RETURN
1520 '
1530 '_____subroutines_____
1540 '
1550 'display an activity and candidates on screen...
1560 IF SEQ.NUM <> CVI(MID$(ACT$(SEQ.NUM),1,2)) THEN PRIN
T "Error in file at record " SEQ.NUM
1570 ACT.SCHED.TIME = CVI(MID$(ACT$(SEQ.NUM),3,2))
1580 'get clock time, day, and date...
1590 GOSUB 3730: SCHED.TIME$ = THIS.TIME$
1600 GOSUB 3880
1610 ACT.CODE = ASC(MID$(ACT$(SEQ.NUM),5,1))
1620 PIL.NUM = ASC(MID$(ACT$(SEQ.NUM),6,1))
1630 START.TIME = CVI(MID$(ACT$(SEQ.NUM),7,2))
1640 GOSUB 3730: ST.TIME$ = THIS.TIME$
1650 END.TIME = CVI(MID$(ACT$(SEQ.NUM),9,2))
1660 GOSUB 3730: END.TIME$ = THIS.TIME$
1670 IF PIL.NUM = 255 THEN PIL.NUM = MAX.PIL.NUM + 1
1680 ACT.NAME$ = MID$(ACT$(SEQ.NUM),11,20)
1690 CAND$ = MID$(ACT$(SEQ.NUM),31,8)
1700 CAND.TOT = 0
1710 FOR I = 1 TO MAX.PIL.NUM
1720 BYTE = ASC(MID$(CAND$,I\8+1,1)): BIT = (I-1)
MOD 8
1730 IF (BYTE AND 2*BIT) = 2*BIT THEN CAND.TOT =
CAND.TOT + 1
1740 NEXT
1750 CUR.REQ = CVI(MID$(ACT$(SEQ.NUM),39,2))
1760 '
1770 PRINT LEFT$(MID$(SCRN$,5),
1780 PRINT "Sequence number:" SEQ.NUM TAB(50) "THIS DAY$ ",
" THIS DATE$ DOWN$
1790 PRINT SCHED.TIME$ " - " ACT.NAME$ TAB(35) "Assigned:
" PIL.NAME$(PIL.NUM) DOWN$
1800 NONE = -1
1810 PRINT "Currencies required:"
1820 FOR I = 0 TO 14
1830 IF (CUR.REQ AND 2*I) = 2*I THEN PRINT CUR.NAME$(I
+1) " "; IF NONE THEN NONE = 0
1840 NEXT
1850 IF NONE THEN PRINT "None"
1860 PRINT
1870 PRINT "Candidate names "

```

```

1880 IF CAND.TOT = 0 THEN PRINT"None shown as both qualif
ied and available": GOTO 2170
1890 K = 0: L = CAND.TOT\3
1900 IF CAND.TOT MOD 3 > 0 THEN L1 = L + 1 ELSE L1 = L
1910 IF CAND.TOT MOD 3 > 1 THEN L2 = L + 1 ELSE L2 = L
1920 BYTE = 0: BIT = 0: LN = 0
1930 WHILE LN < L1
1940 LN = LN + 1
1950 I = 0
1960 'find first column name to print...
1970 FOR J = 1 TO LN
1980 GOSUB 3290
1990 NEXT
2000 'print it...
2010 GOSUB 3370: IF K = CAND.TOT THEN GOTO 2140
2020 'skip 11 names
2030 FOR J = 1 TO L1
2040 GOSUB 3290
2050 NEXT
2060 'print the next one
2070 GOSUB 3370: IF K = CAND.TOT THEN GOTO 2140
2080 'skip 12 names
2090 FOR J = 1 TO L2
2100 GOSUB 3290
2110 NEXT
2120 '
2130 GOSUB 3370
2140 WEND
2150 '
2160 'screen now shows activity and available pilots...
2170 PRINT"Enter 0 to skip selection or pilot number to s
elect a pilot for this activity"
2180 INPUT"Which pilot number? ", I
2190 IF I <= 0 THEN GOTO 3060 ELSE IF I > MAX.PIL.NUM THE
N PRINT"Out of range, enter a number from 0 to" MAX.PIL.NUM
"only, try a
gain": GOTO 2180
2200 'check avail and crew rest if applicable...
2210 GET#2, I
2220 IF I <> ASC(N221$) THEN PRINT"File access error in "
WKDAT.FILES
2230 AVAIL$ = N222$
2240 NET.NLT$ = N225$
2250 FOR N = 1 TO 7
2260 NET(N) = CVI(MID$(NET.NLT$,N*4 - 3,2))
2270 NLT(N) = CVI(MID$(NET.NLT$,N*4 - 1,2))
2280 NEXT
2290 AVAIL = -1: SET = 0
2300 PRINT"Recheck availability of " PIL.NAM$(I);
2310 GOSUB 4300
2320 IF NOT AVAIL THEN PRINT" is not good, resetting stat
us...": MID$(ACT$(SEQ.NUM),I\8+31,1) = CHR$(ASC(MID$(CAND$,I
\8+1,1))) - 2
'((I-1) MOD 8)): GOTO 1560
2330 IF AVAIL AND (ACT.CODE AND 192) THEN GOSUB 3110 ELSE

```

```

C.REST = -1
2340 IF C.REST AND AVAIL THEN PRINT" is good" ELSE PRINT"
is not good"
2350 IF AVAIL AND (NOT C.REST) THEN PRINT"Crew rest rules
not met, enter 0 to ignore crew rest or 1 to NOT select thi
s pilot ";
A$ = INPUT$(1): PRINT A$ ELSE GOTO 2380
2360 IF A$ = "1" THEN GOTO 1770 ELSE IF A$ <> "0" THEN PR
INT E$: GOTO 2350
2370 'avail and crew rest check good so set this pilot in
this activity...
2380 SET = -1: GOSUB 4730
2390 BYTE = ASC(MID$(CAND$,I\8+1,1)): BIT = (I-1) MOD 8
2400 IF (BYTE AND 2*BIT) = 2*BIT THEN MID$(CAND$,I\8+1,1)
= CHR$(BYTE - 2*BIT)
2410 LSET N222$ = AVAIL$
2420 FOR N = 1 TO 7
2430 IF NET(N) <> -1 THEN MID$(NET.NLT$,N*4 - 3,2
) = MKI$(NET(N))
2440 IF NLT(N) <> -1 THEN MID$(NET.NLT$,N*4 - 1,2
) = MKI$(NLT(N))
2450 NEXT
2460 LSET N225$ = NET.NLT$
2470 PUT#2, I
2480 'update act$(seq.num) in memory...
2490 MID$(ACT$(SEQ.NUM),6,1) = CHR$(I)
2500 MID$(ACT$(SEQ.NUM),31,8) = CAND$
2510 FOR J = 1 TO MAX.N
2520 IF J = SEQ.NUM THEN GOTO 2570
2530 IF END.TIME < CVI(MID$(ACT$(J),7,2)) THEN GO
TO 2570
2540 IF START.TIME > CVI(MID$(ACT$(J),9,2)) THEN
GOTO 2570
2550 B = ASC(MID$(ACT$(J),I\8+31,1))
2560 IF B AND 2*((I-1) MOD 8) THEN MID$(ACT$(J),I
\8+31,1) = CHR$(B - 2*((I-1) MOD 8))
2570 NEXT
2580 IF PIL.NUM = MAX.PIL.NUM + 1 THEN GOTO 1560
2590 'if pil.num <> 61 then reset old pilots bit and avai
ls(pil.num)...
2600 GET#2, PIL.NUM
2610 AVAIL$ = N222$
2620 NET.NLT$ = N225$
2630 FOR N = 1 TO 7
2640 NET(N) = CVI(MID$(NET.NLT$,N*4 - 3,2))
2650 NLT(N) = CVI(MID$(NET.NLT$,N*4 - 1,2))
2660 NEXT
2670 BYTE = ASC(MID$(CAND$,PIL.NUM\8+1,1)): BIT = (PIL.NU
M-1) MOD 8
2680 IF (BYTE AND 2*BIT) = 0 THEN MID$(CAND$,PIL.NUM\8+1,
1) = CHR$(BYTE + 2*BIT)
2690 FOR I = START.BIT TO END.BIT
2700 BYTE = I\8+1: BIT = (I-1) MOD 8
2710 BYT$ = MID$(AVAIL$,BYTE,1)
2720 IF (ASC(BYT$) AND 2*BIT) = 0 THEN MID$(AVAIL

```

```

$,BYTE,1) = CHR$(ASC(BYTs) + 2*BIT)
2730 NEXT
2740 DAY = START.TIME\1440 + 1
2750 IF DAY < 2 OR DAY > 7 THEN GOTO 2960
2760 IF NLT(DAY-1) <> START.TIME - 720 THEN GOTO 2810
2770 ACT.SCHED.TIME = NLT(DAY-1): GOSUB 3730
2780 PRINT"Cancelled activity set crew rest time for endi
ng previous day: " THIS.TIME$
2790 PRINT C$;: A$ = INPUT$(1): PRINT A$
2800 IF A$ = "1" THEN GOSUB 3410: NLT(DAY-1) = NEW.T + (D
AY - 2)*1440 ELSE IF A$ <> "0" THEN PRINT E$: GOTO 2790
2810 IF NLT(DAY) <> START.TIME + 720 THEN GOTO 2860
2820 ACT.SCHED.TIME = NLT(DAY): GOSUB 3730
2830 PRINT"Cancelled activity set crew rest time for endi
ng this day: " THIS.TIME$
2840 PRINT C$;: A$ = INPUT$(1): PRINT A$
2850 IF A$ = "1" THEN GOSUB 3410: NLT(DAY) = NEW.T + (DAY
- 1)*1440 ELSE IF A$ <> "0" THEN PRINT E$: GOTO 2840
2860 IF NET(DAY) <> END.TIME - 720 THEN GOTO 2910
2870 ACT.SCHED.TIME = NET(DAY): GOSUB 3730
2880 PRINT"Cancelled activity set crew rest time for begi
nning this day: " THIS.TIME$
2890 PRINT C$;: A$ = INPUT$(1): PRINT A$
2900 IF A$ = "1" THEN GOSUB 3410: NET(DAY) = NEW.T + (DAY
- 1)*1440 ELSE IF A$ <> "0" THEN PRINT E$: GOTO 2890
2910 IF NET(DAY+1) <> END.TIME + 720 THEN GOTO 2960
2920 ACT.SCHED.TIME = NET(DAY+1): GOSUB 3730
2930 PRINT"Cancelled activity set crew rest time for begi
nning following day: " THIS.TIME$
2940 PRINT C$;: A$ = INPUT$(1): PRINT A$
2950 IF A$ = "1" THEN GOSUB 3410: NET(DAY+1) = NEW.T + DA
Y*1440 ELSE IF A$ <> "0" THEN PRINT E$: GOTO 2940
2960 PRINT FIL.NAM$(FIL.NUM) " is reset in " WKDAT.FILE$
2970 LSET N222$ = AVAIL$
2980 FOR N = 1 TO 7
2990 IF NET(N) <> -1 THEN MID$(NET.NLT$,N*4 - 3,2
) = MKI$(NET(N))
3000 IF NLT(N) <> -1 THEN MID$(NET.NLT$,N*4 - 1,2
) = MKI$(NLT(N))
3010 NEXT
3020 LSET N225$ = NET.NLT$
3030 PUT#2, FIL.NUM
3040 MID$(ACT$(SEQ.NUM),31,8) = CAND$
3050 GOTO 1560
3060 RETURN
3070
3080 '____internal_subroutines____
3090
3100 'crew rest check...
3110 'this routine checks and sets NET and NLT times used
for checking
3120 ' crew rest...
3130
3140 DAY = START.TIME\1440 + 1
3150 IF DAY < 2 OR DAY > 6 THEN C.REST = -1: RETURN

```

```

3160
3170     IF START.TIME >= NET(DAY) OR NET(DAY) = -1 THEN ST.C
K = -1 ELSE ST.CK = 0
3180     IF END.TIME <= NLT(DAY) OR NLT(DAY) = -1 THEN END.CK
= -1 ELSE END.CK = 0
3190
3200     IF ST.CK AND END.CK THEN C.REST = -1 ELSE C.REST = 0
: RETURN
3210
3220     IF (START.TIME-720 < NLT(DAY-1)) OR (NLT(DAY-1) = -1
) THEN NLT(DAY-1) = START.TIME - 720
3230     IF (NLT(DAY) > START.TIME+720) OR (NLT(DAY) = -1) TH
EN NLT(DAY) = START.TIME + 720
3240     IF (NET(DAY) < END.TIME-720) OR (NET(DAY) = -1) THEN
NET(DAY) = END.TIME - 720
3250     IF (NET(DAY+1) < END.TIME+720) OR (NET(DAY+1) = -1)
THEN NET(DAY+1) = END.TIME + 720
3260
3270 RETURN
3280
3290     IF I < MAX.PIL.NUM THEN I = I + 1 ELSE GOTO 3350
3300     BYTE = ASC(MID$(CAND$,I\8+1,1)): BIT = (I-1) MOD 8
3310     WHILE ((BYTE AND 2*BIT) <> 2*BIT) AND (I < MAX.PIL.N
UM)
3320         I = I + 1
3330         BYTE = ASC(MID$(CAND$,I\8+1,1)): BIT = (I-1)
MOD 8
3340     WEND
3350     RETURN
3360
3370     PRINT TAB((K*25+1) MOD 75) USING "##"; I; PRINT " -
" PIL.NAM$(I); K = K + 1: IF K MOD 3 = 0 THEN PRINT
3380     RETURN
3390
3400
3410     PRINT"Enter the new crew rest time:";
3420     INPUT " ", NEW.T
3430     HR = NEW.T\100: MIN = NEW.T MOD 100: BAD = 0
3440     IF HR < 0 OR HR > 24 THEN BAD = -1
3450     IF MIN < 0 OR MIN > 59 THEN BAD = -1
3460     IF BAD THEN PRINT"Time not understood, re-enter as a
4 digit number": GOTO 3410
3470     NEW.T = HR*60 + MIN
3480 RETURN
3490
3500     'trim trailing spaces...
3510     L = LEN(TRIM$) + 1: L.CHR$ = CHR$(0)
3520     WHILE ASC(L.CHR$) < 33
3530         L = L - 1
3540         L.CHR$ = MID$(TRIM$,L,1)
3550     WEND
3560     TRIM$ = LEFT$(TRIM$,L)
3570 RETURN
3580
3590     'dynamic array size increase...

```

```

3600 DIM TMP$(MAX.N)
3610 FOR M = 1 TO MAX.N: TMP$(M) = ACT$(M): NEXT
3620 ERASE ACT$: DIM ACT$(MAX.N + 10)
3630 FOR M = 1 TO MAX.N: ACT$(M) = TMP$(M): NEXT
3640 MAX.N = MAX.N + 10
3650 ERASE TMP$
3660 ON ERROR GOTO 0
3670 RETURN
3680 '
3690 '
3700 '
3710 'this routine computes the time from a time in
3720 'minutes of a week...
3730 HR = (ACT.SCHED.TIME MOD 1440)\60
3740 MIN = (ACT.SCHED.TIME MOD 1440) MOD 60
3750 T$ = MID$(STR$(HR),2): GOSUB 3790: HR$ = T$
3760 T$ = MID$(STR$(MIN),2): GOSUB 3790: MIN$ = T$
3770 THIS.TIMES = HR$ + MIN$
3780 RETURN
3790 WHILE LEN(T$) < 2
3800 T$ = "0" + T$
3810 WEND
3820 RETURN
3830 '
3840 '
3850 'this routine determines the day and date of an
3860 'activity from wk.date, wk.date$, and the
3870 'activity schedule time...
3880 DAY = ACT.SCHED.TIME\1440: THIS.DAYS = MID$(DAY$,DAY
*3 + 1,3)
3890 THIS.DATE.J = WK.DATE + DAY: DAY.J = THIS.DATE.J MOD
1000
3900 YEAR = VAL(RIGHT$(WK.DATES$,2)): IF YEAR/4 = YEAR\4 T
HEN L.YR = -1 ELSE L.YR = 0
3910 MO = 0: NEXT.MO.1ST.DAY = 1
3920 WHILE (DAY.J > NEXT.MO.1ST.DAY) AND (MO < 12)
3930 'save new 'this month', get next month...
3940 THIS.MO.1ST.DAY = NEXT.MO.1ST.DAY
3950 MO = MO + 1
3960 NEXT.MO.1ST.DAY = VAL(MID$(FIRST.DAYS$,MO*3 +
1,3))
3970 IF (MO = 12) AND L.YR THEN NEXT.MO.1ST.DAY =
NEXT.MO.1ST.DAY + 1
3980 WEND
3990 'when the day falls in the following year, loop is
4000 'terminated by mo = 12, thus...
4010 IF DAY.J > NEXT.MO.1ST.DAY THEN YEAR = YEAR + 1: TH
IS.DATE = DAY.J - NEXT.MO.1ST.DAY + 1 ELSE THIS.DATE = DAY.J
- THIS.MO.1
ST.DAY + 1
4020 THIS.DATES = MID$(STR$(THIS.DATE),2) + " " + MID$(MO
NTH$, (MO-1)*3 + 1,3) + STR$(YEAR)
4030 RETURN
4040 '
4050 '

```

```

4060      '*** CASE ***
4070      'module dated 24 April 1983
4080      '
4090      'This module includes subroutines called by other
4100      '      modules in determining the case of each
4110      '      activity relative to the week
4120      '
4130      'variables required:
4140      '      PERIOD.ST.TIME as an integer in minutes or o
ther time units
4150      '      PERIOD.DUR as an integer length of period
4160      '      INCR as an integer for the value of each bit
      (resolution)
4170      '      START.TIME as values for the activity
4180      '      END.TIME
4190      '      AVAIL$ as a bit string with '1' available, '
0' not avail
4200      '      AVAIL as a control code
4210      '      SET as a control code to set the time 'not a
available'
4220      '
4230      'returns:
4240      '      AVAIL as TRUE if time is available
4250      '      AVAIL$ updated if AVAIL and SET both TRUE
4260      '
4270      'subroutines used:
4280      '      all internal
4290      '
4300      GOSUB 4520
4310      GOSUB 4730
4320      RETURN
4330      '


---


4350      '--- This routine determines the case of activity
4360      '      start (CASE1) and end (CASE2) relative to
4370      '      the period start and end...
4380      '
4390      '      CASE1 and CASE2 equal 1 if times are before
4400      '      the period starts, 2 if during the period,
4410      '      or 3 if after the period. Thus if CASE1 is
4420      '      3 or CASE2 is 1, the whole activity falls
4430      '      outside the period in question. If both
4440      '      CASE1 and CASE2 are 2, then the whole
4450      '      activity is within the period.
4460      '
4470      '      CASE3 has a value of 1 if the whole
4480      '      activity falls on a single byte, 2 if on
4490      '      adjacent bytes, and 3 if one or more whole
4500      '      bytes fall between the start and end.
4510      '
4520      START.BIT = START.TIME\INCR
4530      START.BYTE = START.BIT\8 + 1
4540      END.BIT = (END.TIME-1)\INCR
4550      END.BYTE = END.BIT\8 + 1
4560      IF START.TIME >= PERIOD.ST.TIME THEN COND1 = -1 ELSE

```

```

COND1 = 0
4570 IF START.TIME < (PERIOD.ST.TIME + PERIOD.DUR) THEN C
OND2 = -1 ELSE COND2 = 0
4580 IF COND1 AND COND2 THEN CASE1 = 2 ELSE IF NOT COND1
THEN CASE1 = 1 ELSE IF NOT COND2 THEN CASE1 = 3
4590 IF END.TIME > PERIOD.ST.TIME THEN COND3 = -1 ELSE CO
ND3 = 0
4600 IF END.TIME <= (PERIOD.ST.TIME+PERIOD.DUR) THEN COND
4 = -1 ELSE COND4 = 0
4610 IF COND3 AND COND4 THEN CASE2 = 2 ELSE IF NOT COND3
THEN CASE2 = 1 ELSE IF NOT COND4 THEN CASE2 = 3
4620 IF END.BYTE = START.BYTE THEN CASE3 = 1
4630 IF END.BYTE - START.BYTE = 1 THEN CASE3 = 2
4640 IF END.BYTE - START.BYTE > 1 THEN CASE3 = 3
4650 RETURN
4660
4670
4680 This routine selects the proper routine for
4690 checking or setting availability based on
4700 the case defined by CASE1, CASE2, and CASE3
4710
4720 if start is before period...
4730 IF CASE1=1 AND CASE2=2 THEN ON CASE3 GOSUB 4880,4980
,4980
4740 if start and end are during period...
4750 IF CASE1=2 AND CASE2=2 THEN ON CASE3 GOSUB 5110,5280
,5240
4760 if start is during period but end is after...
4770 IF CASE1=2 AND CASE2=3 THEN ON CASE3 GOSUB 5460,5590
,5590
4780 if start is before and end is after period...
4790 IF CASE1=1 AND CASE2=3 THEN FIRST.BYT=1: LAST.BYT=LE
N(AVAIL$): GOSUB 5630
4800 the final case ends before or starts after p
eriod...
4810 IF CASE1=3 OR CASE2=1 THEN PRINT"Activity is complet
ely outside the period..."
4820 RETURN
4830
4840
4850 --- This routine is used when END.BYTE$ is the
4860 first byte of AVAIL$...
4870
4880 FIRST.BIT.USED = 0: LAST.BIT.USED = (END.BIT MOD 8)
4890 BYT.TO.CK$ = LEFT$(AVAIL$,1)
4900 GOSUB 5850
4910 IF AVAIL AND SET THEN MID$(AVAIL$,1,1) = CHR$(ASC(BY
T.TO.CK$) AND (NOT MASK))
4920 RETURN
4930
4940
4950 --- This routine is used when END.BYTE points to

```

```

4960      "      end byte
4970
4980      FIRST.BYT = 1: LAST.BYT = END.BYTE-1
4990      GOSUB 5730
5000      FIRST.BIT.USED = 0: LAST.BIT.USED = (END.BIT MOD 8)
5010      BYT.TO.CK$ = MID$(AVAIL$,END.BYTE,1): J = END.BYTE
5020      GOSUB 5850
5030      IF AVAIL AND SET THEN GOSUB 5970 ELSE RETURN
5040      MID$(AVAIL$,END.BYTE,1) = CHR$(ASC(BYT.TO.CK$) AND (
NOT MASK))
5050      RETURN
5060
5070
5080      --- This routine is used for the single byte case
5090      '      where one byte includes both start and end...
5100
5110      BYT.TO.CK$ = MID$(AVAIL$,START.BYTE,1)
5120      MASK = 0
5130      FIRST.BIT.USED = (START.BIT MOD 8)
5140      LAST.BIT.USED = (END.BIT MOD 8)
5150      GOSUB 5850
5160      IF AVAIL AND SET THEN MID$(AVAIL$,START.BYTE,1) = CH
R$(ASC(BYT.TO.CK$) AND (NOT MASK))
5170      RETURN
5180
5190
5200      --- This routine is used when one or more bytes
5210      '      separate the first and last bytes or when
5220      '      they are adjacent...
5230
5240      FIRST.BYT = START.BYTE+1: LAST.BYT = END.BYTE-1
5250      GOSUB 5730
5260      IF NOT AVAIL THEN RETURN
5270      '      CASE3 = 2 enters here...
5280      FIRST.BIT.USED = (START.BIT MOD 8): LAST.BIT.USED =
7
5290      BYT.TO.CK$ = MID$(AVAIL$,START.BYTE,1)
5300      GOSUB 5850
5310      IF AVAIL AND SET THEN MASK.ST=MASK: BYT.ST$=BYT.TO.C
K$ ELSE IF NOT AVAIL THEN RETURN
5320      FIRST.BIT.USED = 0: LAST.BIT.USED = (END.BIT MOD 8)
5330      BYT.TO.CK$ = MID$(AVAIL$,END.BYTE,1): J = END.BYTE
5340      GOSUB 5850
5350      IF NOT AVAIL THEN RETURN
5360      IF (CASE3=3) AND (AVAIL AND SET) THEN GOSUB 5970
5370      IF NOT(AVAIL AND SET) THEN RETURN
5380      MID$(AVAIL$,START.BYTE,1) = CHR$(ASC(BYT.ST$) AND (N
OT MASK.ST))
5390      MID$(AVAIL$,END.BYTE,1) = CHR$(ASC(BYT.TO.CK$) AND (
NOT MASK))
5400      RETURN
5410
5420
5430      --- This routine is used when the last byte in the
5440      '      string is the only byte to be checked...

```

```

5450
5460     BYT.TO.CK$ = MID$(AVAIL$,START.BYTE,1)
5470     FIRST.BIT.USED = (START.BIT MOD 8): LAST.BIT.USED =
(PERIOD.END.BIT MOD 8)
5480     GOSUB 5850
5490     IF AVAIL AND SET THEN MID$(AVAIL$,START.BYTE,1) = CH
R$(ASC(BYT.TO.CK$) AND (NOT MASK))
5500     RETURN
5510
5520
5530     --- This routine is used when the activity ends
5540         after the period and the first byte is one
5550         or more bytes from the end of AVAIL$. The
5560         last two cases of CASE3 are both checked by
5570         this routine...
5580
5590     FIRST.BIT.USED = (START.BIT MOD 8): LAST.BIT.USED =
7
5600     BYT.TO.CK$ = MID$(AVAIL$,START.BYTE,1)
5610     GOSUB 5850
5620     FIRST.BYT = START.BYTE+1: LAST.BYT = LEN(AVAIL$)
5630     GOSUB 5730
5640     IF AVAIL AND SET THEN GOSUB 5970 ELSE RETURN
5650     MID$(AVAIL$,START.BYTE,1) = CHR$(ASC(BYT.TO.CK$) AND
(NOT MASK))
5660     RETURN
5670
5680
5690     --- This routine is used by the routines above when
5700         whole bytes are being checked for
5710         availability...
5720
5730     FOR J = FIRST.BYT TO LAST.BYT
5740         BYT.TO.CK$ = MID$(AVAIL$,J,1)
5750         IF BYT.TO.CK$ (>) CHR$(255) THEN FIRST.BIT.US
ED=0: LAST.BIT.USED=7: GOSUB 5850
5760         IF NOT AVAIL THEN RETURN
5770     NEXT
5780     RETURN
5790
5800
5810     --- This routine is called by above routines to
5820         check availability within partial bytes of
5830         AVAIL$...
5840
5850     MASK = 0
5860     FOR K = FIRST.BIT.USED TO LAST.BIT.USED
5870         MASK = MASK + 2^K
5880         IF (ASC(BYT.TO.CK$) AND 2^K) = 0 THEN AVAIL
= 0: RETURN
5890     NEXT
5900     RETURN
5910
5920
5930     --- This routine is called when a whole byte is to

```

```
5940      '      be set to NOT AVAILABLE state, both AVAIL
5950      '      and SET are TRUE...
5960      '
5970      FOR J = FIRST.BYT TO LAST.BYT
5980          MIDS(AVAIL$,J,1) = CHR$(0)
5990      NEXT
6000  RETURN
6010      ' 

---


```

BIBLIOGRAPHY

Aho, Alfred V., John E. Hopcroft, and Jeffrey D. Ullman. The Design and Analysis of Computer Algorithms. Reading, MA: Addison-Wesley Publishing Co., 1974.

Berman, Morton B. The DOSS Prototype. #WN-9484-PR. Santa Monica, CA: Rand Corporation, 1976.

_____. Scheduling Aircrews and Aircraft: Problems of Resource Allocation in the Strategic Air Command. #R-1610-PR. Santa Monica, CA: Rand Corporation, 1975.

DEPARTMENT OF THE AIR FORCE, Headquarters Tactical Air Command. FLYING TRAINING: TACTICAL FIGHTER/RECONNAISSANCE AIRCREW TRAINING. TAC MANUAL 51-50, Volume I, 26 October 1981.

_____. TAC AND ARF TRAINING: FIGHTER AND RECONNAISSANCE. TACM 51-50 Volume I, Chapter 6, 15 February 1982.

_____. Flying Training: F-15 AIRCREW TRAINING. TAC MANUAL 51-50, Volume VII, 26 March 1982.

DEPARTMENT OF THE AIR FORCE, Headquarters United States Air Force. Flying: PLANNING AND SCHEDULING AIRCREWS AND EQUIPMENT. AF REGULATION 60-12, 22 March 1979.

DEPARTMENT OF THE AIR FORCE, Headquarters United States Air Forces in Europe. Flying: PLANNING AND SCHEDULING AIRCREWS AND EQUIPMENT. USAFE Supplement 1 to AFR 60-12 (22 March 1979), 5 August 1982.

_____. Flying Training: TACTICAL FIGHTER/RECONNAISSANCE AIRCREW TRAINING. USAFE Chapter 6 to TACM 51-50, Volume I, 1 October 1982.

Grogono, Peter. Programming in Pascal: Revised Edition. Reading, MA: Addison-Wesley, 1980, 1978.

Hogan, Thom and Mike Iannamico, OSBORNE 1 User's Reference Guide. Hayward, CA: Osborne Computer Corp., 1981, revised 2/22/82.

Horowitz, Ellis, and Sartaj Sahni. Fundamentals of Data Structures. Rockville, MD: Computer Science Press, Inc., 1982, 1976.

Jensen, Kathleen and Niklaus Wirth, PASCAL: User Manual and Report: 2d ed (corrected printing 1978). New York: Springer-Verlag, 1974.

- Kernighan, Brian W., and P. J. Plauger. The Elements of Programming Style: 2d ed. New York: McGraw-Hill, 1978, 1974.
- _____. Software Tools. Reading, MA: Addison-Wesley Publishing Co., 1976.
- Knuth, Donald E. The Art of Computer Programming: Vol. 1 Fundamental Algorithms: 2d ed. Reading, MA: Addison-Wesley Publishing Co., 1973, 1968.
- _____. The Art of Computer Programming: Vol. 2 Semi-numerical Algorithms: 2d ed. Reading, MA: Addison-Wesley Publishing Co., 1981, 1969.
- _____. The Art of Computer Programming: Vol. 3 Sorting and Searching. Reading, MA: Addison-Wesley Publishing Co., 1973.
- Leventhal, Lance A. 8080A - 8085 Assembly Language Programming. Berkeley, CA: Osborne/McGraw-Hill, 1978.
- _____. Z80 Assembly Language Programming. Berkeley, CA: Osborne/McGraw-Hill, 1979.
- Lien, David A. The BASIC Handbook: 2d ed. San Diego, CA: Compusoft Publishing, 1981.
- Osborne, Adam and David Bunnell. An Introduction to Microcomputers: Vol. 0 The Beginners Book: 3d ed. Berkeley, CA: Osborne/McGraw-Hill, 1982, 1979, 1977.
- Osborne, Adam. An Introduction to Microcomputers: Vol. 1 Basic Concepts: 2d ed. Berkeley, CA: Osborne/McGraw-Hill, 1980, 1976.
- Pannell, Carlton L. A LINEAR PROGRAMMING APPLICATION TO AIRCREW SCHEDULING. Ft. Leavenworth, KS: US Army Command and General Staff College, 1980.
- Stern, Robert A. and Nancy Stern. An Introduction to Computers and Information Processing. New York: John Wiley and Sons, 1982, 1979.
- Strunk, Richard R. Can TAC Operations be Computer Scheduled? Maxwell AFB, AL: US Air Force Air Command and Staff College, 1977.

INITIAL DISTRIBUTION LIST

Combined Arms Research Library
U.S. Army Command and General Staff College
Fort Leavenworth, Kansas 66027

Defense Technical Information Center
Cameron Station
Alexandria, Virginia 22314

Mr. David I. Drummond
U.S. Army Command and General Staff College
Department of Command
Fort Leavenworth, Kansas 66027

Lieutenant Colonel William B. Allard
3220 Homer Road
Winoma, Minnesota 55987

Major Donald Hayes
U.S. Army Command and General Staff College
Air Force Section
Fort Leavenworth, Kansas 66027

