<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
</tr>
<tr>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
<td>50</td>
</tr>
<tr>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>56</td>
<td>57</td>
<td>58</td>
<td>59</td>
<td>60</td>
</tr>
<tr>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>69</td>
<td>70</td>
</tr>
<tr>
<td>71</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>78</td>
<td>79</td>
<td>80</td>
</tr>
<tr>
<td>81</td>
<td>82</td>
<td>83</td>
<td>84</td>
<td>85</td>
<td>86</td>
<td>87</td>
<td>88</td>
<td>89</td>
<td>90</td>
</tr>
</tbody>
</table>
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A
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.
**Computer Assisted Scheduling for Air Force Tactical Fighter Squadrons**

Dugle, Brian C., Major, USAF

Student at the U.S. Army Command and General Staff College, Fort Leavenworth, Kansas, 66027

HQ TRADOC, Attn: ATCS-D, Fort Monroe, VA 23651

Approved for public release; distribution unlimited.

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

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.
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:

Mr. David I. Drummond, MS

Major Donald F. Hayes, MAS

Lieutenant Colonel William B. Allard, MS

Accepted this 31st day of May 1983 by Philip L. Gooden

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.)
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.
# TABLE OF CONTENTS

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

---

This description was obtained from Maj Dave Smith, Wing Training Officer of the 1st Tactical Fighter Wing, Langely 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. 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 AFB\(^1\). None of these indicated specific work on using microcomputers to aid the scheduler.

One of the few references found includes a very general paper written by Major Richard Strunk submitted as a research 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

\(^1\)Location 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 general. 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).  

---


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 that in Strategic Air Command, where his work was done.

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.


Telephone 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 SCHEDULING*. 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 years. 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.

---

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

6 Telephone 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". With the capabilities of

---

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 curren-
cies such as landing, ACBT, or low level intercepts are not too difficult, generally requiring a flight under the super-
vision 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 ASCII\(^1\) 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.

\(^1\)See 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 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 satisfices.
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 person's needs. 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 future.

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,

\[1\text{See Annex B for more information on programming.}\]
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 51-50\(^1\) and the resulting scheduling decisions that an 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

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 ARF\(^3\) AIRCREW TRAINING\(^4\) and USAFE Chapter 6, TACTICAL FIGHTER/RECONNAISSANCE AIRCREW TRAINING\(^5\). 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

\(^3\)ARF is Air Reserve Forces, both Air National Guard and US Air Force Reserve.


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 qualification

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:

---

6 TAC 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 follow-
ing list is excerpted from that table leaving out some of
the complicating qualifiers that do not apply to the F-15
aircraft or pilots.

\(^7\)TAC Chapter 6, Table 6-10, p. 6-39

\(^8\)TAC Chapter 6, Table 6-11, p. 6-39
<table>
<thead>
<tr>
<th>Accomplishment (Event/Sortie)</th>
<th>Inexperienced Pilot</th>
<th>Experienced Pilot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day Landing</td>
<td>30 days</td>
<td>45 days</td>
</tr>
<tr>
<td>Night Landing</td>
<td>15</td>
<td>30</td>
</tr>
<tr>
<td>AAR (Day or Night)</td>
<td>Six Months</td>
<td></td>
</tr>
<tr>
<td>ACBT (Air Combat Training)</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td><strong>Formation Events</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Takeoff (Day or Night)</td>
<td>60</td>
<td>90</td>
</tr>
<tr>
<td>Day Landing</td>
<td>60</td>
<td>90</td>
</tr>
<tr>
<td>Low Level Flying</td>
<td>60</td>
<td>90</td>
</tr>
<tr>
<td>IP Rear Seat Landing</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>IP Instruction Flight</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>Dart</td>
<td>18 Months</td>
<td></td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Level</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total (Inex/Exp)</td>
<td>40/36</td>
<td>68/60</td>
<td>82/70</td>
</tr>
<tr>
<td>1 month GCC rate</td>
<td>7/6</td>
<td>12/10</td>
<td>14/12</td>
</tr>
<tr>
<td>3 month GCC rate</td>
<td>20/18</td>
<td>34/30</td>
<td>41/35</td>
</tr>
</tbody>
</table>

**Weapons Events (Required)**

<table>
<thead>
<tr>
<th>Event</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dart</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gun Tracking</td>
<td>6</td>
<td>12</td>
<td>18</td>
</tr>
<tr>
<td>WSEP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WSEP^9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

^9WSEP is Weapons System Evaluation Program.
GCC Events

<table>
<thead>
<tr>
<th>Event</th>
<th>20/16</th>
<th>26/22</th>
<th>32/28</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercepts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECCM(^1)</td>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Alert Scramble</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Integrated Msn/Joint Ex</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Comm Jam</td>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>ACBT Sorties</td>
<td>31/27</td>
<td>43/37</td>
<td>50/42</td>
</tr>
<tr>
<td>BFM/DBFM(^1)(^1) Sorties</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Instrument/Proficiency Sorties</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>AAR</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Captive AIM-9</td>
<td>6</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>CWD Sorties</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ACMI(^1)(^2) Sorties</td>
<td>4</td>
<td>6</td>
<td>8</td>
</tr>
</tbody>
</table>

F-15 AIRCREW TRAINING

Volume VII\(^1\)\(^3\) 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

\(^1\)ECCM is Electronic Counter Countermeasures.

\(^1\)\(^1\)BFM is Basic Fighter Maneuvers, DBFM is the same mission flown with dissimilar aircraft.

\(^1\)\(^2\)ACMI is the Air Combat Maneuvering Instrumentation, a realistic training enhancement.

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.
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 BASIC\(^1\), including string or character, integer, and single or double precision real variables. Characters are stored with one byte used for

\(^1\text{Trade Mark of Microsoft Corp., Bellevue, WA.}\)
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 decimal$^2$.

$^2$This 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 available. 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.

47
```
100 '*** CURDEF .SET ****************************
110 'program dated 16 May 1983
120 'This program sets or changes the values stored for
130 'each currency code
140 'DEFINT A-Z
150 'CLRS = CHR$(26): DOWN$ = CHR$(10)
160 MID.SCRNS = CLRS + STRING$(8,10)
170 DIM EVENTS$(10)
180 OPEN "R", 1, "CUR.DEF", 28
190 FIELD$I, 2 AS N1$, 20 AS N12$, 2 AS N13$, 2 AS N14$.
200 PRINT MID.SCRNS "Enter currency code number to change
210 or 0 to quit";
220 IF CODE a 0 THEN CLOSE: END ELSE IF CODE > 15 THEN PRINT"Error, out of range (max is 15)"; GOTO 250
230 GET$I, CODE
240 CURCODE = CVI(N1$)
250 CUR$ = N12$ + N13$ + N14$ + N15$
260 IF CUR.CODE () CODE THEN PRINT"File error: record number not equal to currency code";
270 PRINT"Press any key to continue ..."; DUMMY$ = INPUT$(1)
280 PRINT MID.SCRNS "Current data:";
290 PRINT CUR.CODE CUR$ PER.EX PER.INX EVNT
300 PRINT MID.SCRNS "Enter:
310 PRINT" 0 if all correct, no changes"
320 PRINT" 1 to change currency name"
330 PRINT" 2 to change experienced currency period"
340 PRINT" 3 to change inexperienced period"
350 PRINT" 4 to change updating event number"
360 PRINT" Which choice?"; A = VAL(INPUT$(1))
370 IF A = 0 THEN LSET N1$ = MKI$(CODE);
380 PUT$I, CODE: GOTO 240 ELSE IF A > 4 THEN PRINT"Error, enter a number from 0 to 4 only, try again ...": GOTO 430
390 ON A GOSUB 500, 580, 640, 700
400 GOTO 340
410 'subroutines
420 PRINT MID.SCRNS CUR.CODE CUR$ PER.EX PER.INX EVNT
430 PRINT "What is the new currency name";
440 IF LEN(CUR$) > 20 THEN PRINT"Too long, only 20 characters will be saved"
450 PRINT"Enter 0 if currency name is correct, 1 to chan
```
A = VAL(INPUT$(1)); PRINT A
540 IF A = 1 THEN GOTO 510 ELSE IF A <> 0 THEN PRINT"Error. 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 EVNT
590 PRINT: PRINT"Enter the period of currency for experienced 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 EVNT
650 PRINT: PRINT"Enter the period of currency for inexperienced 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 EVNT
710 PRINT: PRINT"Enter the event number that updates this currency (? for help): "
720 INPUT"What is the event number? " , EVNT$
730 IF EVNT$ = "?" THEN GOSUB 790
740 EVNT = VAL(EVNT$)
750 IF EVNT <= 1 OR EVNT > 10 THEN PRINT"Enter a number from 1 to 10 only..."; GOSUB 790: GOTO 740
760 LSET N15$ = 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
50

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$:
180 FOR I = 1 TO 8: MID.SCRN$ = MID.SCRN$ + DOWN$: NEXT
190 DIM ST.T(3), END.T(3), GP$(3), CUR CAT$(15), QUAL.CAT$(15)
200 ST.T(0) = 0: ST.T(1) = 15: ST.T(2) = 135: ST.T(3) = 165
210 END.T(0) = 0: END.T(1) = 90: END.T(2) = 180: END.T(3) = 240
220 GP$(0) = "Non-duty activities" GP$(1) = "Duty/non-flying activities"
230 GP$(2) = "Flying activities" GP$(3) = "Long flight activities"
240 OPEN "R", 1, "ACT DEF", 46
250 FIELD#1, 2 AS NI11$, 20 AS NI23$, 2 AS NI33$, 2 AS NI43$, 10 AS NI54$, 10 AS NI64$
260 
270 PRINT MID.SCRN$ "Enter activity code number to change 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(NI11$)
320 ACT.NAMES = NI23$
330 ST.T = CVI(NI33$)
340 END.T = CVI(NI43$
350 CUR$ = NI54$
360 QUAL$ = NI64$
370 IF ACT.CODE () CODE THEN PRINT "File error record number not equal to activity code": PRINT "Press any key to continue...":
380 DUMMYS = INPUTS(1)
390 PRINT MID.SCRN$ "Current data:
390 PRINT ACT.CODE ACT.NAMES 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"
450 PRINT " 3 to check currency requirements"
460 PRINT " 4 to check qualification requirements"
470 PRINT "Which choice?": A = VAL(INPUTS(1)) PRINT A
480 IF A = 0 THEN LSET NI11$ = MKI$(CODE): PUTOI, CODE: GOTO 270 ELSE IF A > 4 THEN PRINT "Error, enter a number from 0 to 4 only."
try again..." : GOTO 470

ON A GOSUB 540, 620, 790, 1030

GOTO 380

' _subroutines

PRINT MID.SCRNS ACT NAMES

PRINT "What is the new activity name" ; : INPUT ACT NAMES

IF LEN(ACT NAMES) > 20 THEN PRINT "Too long, only 20 characters will be saved"

PRINT "Enter 0 if activity name is correct, 1 to change it" ; : A = VAL(INPUTS()) : PRINT A

IF A = 1 THEN GOTO 460 ELSE IF A <> 0 THEN PRINT "Error, 0 or 1 only, try again..." : GOTO 490

LSET N12$ = ACT NAMES

RETURN

PRINT MID.SCRNS "Start offset is the time before the activity that availability is required"

PRINT "End offset is the time for the activity and de brief or travel time following"

J = ACT CODE / 64

PRINT "This activity code group includes " GP$(J) " ; "

PRINT ST.T(J) "is the standard number of minutes set for start offset"

PRINT END.T(J) "is the standard end offset"

PRINT "Enter :"

PRINT "0 if the old offsets are correct"

PRINT "1 to change to the standard offsets"

PRINT "2 to enter different offsets"

PRINT "Which choice? " ; : A = VAL(INPUTS() (1)) : PRINT A

IF A = 0 THEN GOTO 760 ELSE IF A > 2 THEN PRINT "Error, 0, 1, or 2 only, try again..." : GOTO 720

IF A = 1 THEN ST.T = ST.T(J) : END.T = END.T(J)

IF A = 2 THEN INPUT "End offset (minutes):", END.T

LSET N13$ = MKI$(ST.T) : LSET N14$ = MKI$(END.T)

RETURN

OPEN "R", 2, "CUR.DEF", 28

FIELD#2, 2 AS N21$, 20 AS N22$, 2 AS N23$, 2 AS N24$

I = 0

FOR I = 1 TO 15

GET#2, I

CUR.CAT$(I) = N22$

NEXT

PRINT MID.SCRNS "Up to five combinations of currencies are allowed for each activity"

PRINT "For each currency category enter 1 if it applies, 0 if it does not"

CUR$(I) = 0 : J = 1

FOR I = 1 TO 15

PRINT CUR.CAT$(I) "? " : BIT = VAL (INPUTS(1))
PRINT BIT

910    IF BIT THEN CUR(J) = CUR(J) + 2^(I-1)
920    NEXT
930    PRINT "This set complete, enter 0 if done, 1 to enter another set"; A = VAL(INPUTS(1)) PRINT A
940    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
950    IF A <> 0 THEN PRINT "Error, enter 0 or 1 only, try again..."; GOTO 930
960    CURR$ = ""
970    FOR I = 1 TO J
980    CURR$ = CURR$ + MKS(CURR(I))
990    NEXT
1000   LSET N15$ = CURR$
1010   CLOSE#2: RETURN
1020   OPEN "R", 2, "QUAL DEF", 22
1030   FIELD#2, 2 AS N21$, 20 AS N22$
1040   I = 0
1050   FOR I = 1 TO 15
1060     GET#2, I
1070     QUAL.CAT$ (I) = N22$
1080   NEXT
1090   PRINT MID_SCRNS "Up to five combinations of qualifications are allowed for each activity"
1100   PRINT "For each qualification category enter 1 if it applies, 0 if it does not..."
1110   PRINT BIT "If BIT THEN QUAL(J) = QUAL(J) + 2^(I-1)
1120   NEXT
1130   PRINT "This set complete, enter 0 if done, 1 to enter another set"; A = VAL(INPUTS(1)); PRINT A
1140   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 1130
1150   IF A <> 0 THEN PRINT "Error, enter 0 or 1 only, try again..."; GOTO 1170
1160   QUAL$ = ""
1170   FOR I = 1 TO J
1180     QUAL$ = QUAL$ + MKS(QUAL(I))
1190   NEXT
1200   LSET N16$ = QUAL$
1210   CLOSE#2: RETURN
1220   " 
This program allows entry of availability data for pilots.

Variables required: none

returns: PILnn.DAT files updated and in order

DEFINT A-Z
CLR$ = CHR$(26): DOWN$ = CHR$(10): ESC$ = CHR$(27)
MID.COMS$ = CLR$ + STRING$(6,10)
HOME$ = CHR$(30): CLR.LINE$ = ESC$ + "T"
C$ = "Enter: 0 if correct, 1 to change it: "
E$ = "Error, enter 0 or 1 only, try again..."

MAX.PIL.NUM = 60
IF P$(0,0) <> CHR$(255) THEN ERASE P$: DIM P$(MAX.PIL.NUM, 4)
IF QUAL$(0) <> CHR$(255) THEN ERASE QUAL$: DIM QUAL$(15)

open key file...
GOSUB 4080
read in all names...
GOSUB 4010
CLOSE

open qual def file...
GOSUB 4120
FOR I = 1 TO 15: GET#2, I: QUAL$(I) = N2S. NEXT: CLOSE

open curevnt def file...
GOSUB 4160
FOR I = 1 TO 10: GET#2, I: TRIMS = N2S. GOSUB 2490.
EVENT$(I) = TRIMS: NEXT: CLOSE

PRINT MID.COMS$ " Enter:"
PRINT" 0 to quit, all done" PRINT" 1 to add a new pilot data file"
PRINT" 2 to change data in existing data file"
PRINT" 3 to delete a pilot data file"
PRINT" Which choice? ": SEL = VAL(INPUT$(1)). PRINT T SEL
IF SEL <= 0 THEN END ELSE IF SEL > 3 THEN PRINT"Error. enter a number 0 to 3 only, try again..." GOTO 520
ON SEL GOSUB 400, 1340, 1360
GOTO 470
__add_pilot_data__
54

580 'print names to screen...
590 GOSUB 3890
600
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 'check if file already exists...
670 ON ERROR GOTO 700
680 FILENAMES$ = "PIL" + NUM$ + " .DAT": OPEN "1", 1, FILE NAME$,
700 IF ERR = 53 THEN RESUME 750
710 PRINT FILENAMES$ " exists on this disk, confirm you want 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 'name data entered at subroutines...
780 FOR I = 1 TO 3: ON 1 GOSUB 4270, 4360, 4400: NEXT
800
810 'check admin data, update key file...
820 GOSUB 2580
830
840 'now put in individual file...
850 OPEN "O", 1, FILENAME$
860 WRITE#1, NUM$
870 WRITE#1, L NAMEx, F NAMEx, MIt
880 WRITE#1, RANKex
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. SCRNS$
950 PRINT "Enter a 1 digit if the qualification applies, 0 if it does not"
960 PRINT: PRINT QUALs(1) "? 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. SCRNS$
1070 PRINT "Enter the date " EVENTs(I) " last accomplished
or 0 for none:
1080 GOSUB 5630
1090 WRITE#1, DATE
1100 NEXT
1110 CLOSE #2
1120 GOSUB 4200
1130 A = 1: N = 0: MAX.N = 10
1140 GOSUB 1920
1150 'save max number activities...
1160 MAX.N = N
1170 'entries complete, sort them...
1180 GOSUB 4890
1190 'then check for conflicts...
1200 GOSUB 5070
1210 'print to file...
1220 WRITE#1, "ACTIVITIES SCHEDUL", MAX.N
1230 FOR I = 1 TO MAX.N: PRINT#1, ACTS(I): NEXT
1240 CLOSE
1250 RETURN
1260 GOSUB 4200
1270 GOSUB 4890
1280 GOSUB 5070
1290 WRITE#1, "ACTIVITIES SCHEDULED.", MAX.N
1300 FOR I = 1 TO MAX.N: PRINT#1, ACTS(I): NEXT
1310 CLOSE
1320 RETURN
1330 GOSUB 4200
1340 GOSUB 4890
1350 GOSUB 5070
1360 PRINT MID_SCRN$ CVI(P$(NUM, 1)) " - " P$(NUM, 2) P$(NUM, 3) " - " P$(NUM, 4)
1370 PRINT: PRINT" Enter:" 0 if no more changes or additions, a
1380 PRINT" 1 to change admin data (name, rank, SSAN)"
1390 PRINT" 2 to change qualification data"
1400 PRINT" 3 to update currency data"
1410 PRINT" 4 to add, change or delete availabil
1420 PRINT" Which choice?": A = VAL(INPUT$(1)): PRINT
1430 IF A = 0 THEN GOSUB 2900: RETURN
1440 IF A > 4 THEN PRINT"Error, enter 0 to 4 only, try ag
1450 GOTO 1450
1460 ON A GOSUB 2580, 3130, 3230, 3330
1470 QUIT
1480 GOTO 1380
1500 GOTO 1380
1510 GOTO 1380
1520 GOTO 1380
1530 GOTO 1380
1540 GOTO 1380
1550 GOTO 1380
1560 'get and check pilot number.
1560 GOSUB 1730
1570 PRINT "Enter 0 to delete this file, 1 to abort delete action"
1580 INPUT "Which one"; D
1590 IF D () 0 THEN PRINT "Exiting delete mode. File NOT deleted."
1600 FOR I = 1 TO 1000: NEXT: GOTO 1680
1600 OPEN "O", 3, TMP.FILS: CLOSE # 3: GOSUB 3030: KILL FIL ENAME$ = "Not in use": LSET N2$ = "" LSET N3$ = "" LSET N4$ = ""
1640 PUT # 1, NUM
1650 CLOSE # 1
1660 PRINT "Reset memory variables...
1670 P$(NUM, 2) = "Not in use": P$(NUM, 3) = "": P$(NUM, 4) = ""
1680 RETURN
1690 'Subroutines
1700 'Get and confirm pilot number...
1710 PRINT MID SCRN$ = "": INPUT" ", ANSWER$
1720 IF ASC(LEFT$(ANSWER$, 1)) < 58 THEN NUM = VAL(ANSWER$)
1730 ELSE NUM = 0: L.NAMES = ANSWER$
1740 'Look for name match...
1750 WHILE NUM < MAX PIL NUM AND NOT THIS NUM
1760 NUM = NUM + 1
1770 IF L.NAMES = LEFT$(P$(NUM, 2), LEN(L.NAMES)) THEN
1780 THIS NUM = -1 ELSE THIS NUM = 0
1790 WEND
1800 IF THIS NUM = 0 THEN GOSUB 3890: PRINT: PRINT "Enter pilot number: ": INPUT NUM
1820 IF NUM = 0 THEN GOTO 1890
1830 PRINT MID SCRN$: CVI(P$(NUM, 1)) = "": P$(NUM, 2) = P$(NUM, 3) = "": P$(NUM, 4) = ""
1840 PRINT "Enter 0 if this the correct entry; 1 if not correct: "; THIS NUM = VAL(INPUT$(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"
1870 'Have correct number, get filenames...
1880 GOSUB 3810
1890 RETURN
1900 'Input a new activity...
1910 WHILE A
1920 N = N + 1
1930 PRINT "Enter activity code (for help) "; CODE$
1940 INPUT " ", CODE$
1950 IF CODE$ = "?" THEN GOSUB 4450 ELSE IF CODE$ = "0" THEN GOTO 2400
1960 ACT.CODE = VAL(CODE$)
1980 IF ACT.CODE < 1 OR ACT.CODE > 255 THEN PRINT "Entry is out of range..."; GOSUB 4450: GOTO 1970
1990 IF (ACT.CODE AND 63) = 63 THEN OTHER = -1 ELSE OTHER = 0
2000 IF OTHER THEN INPUT "What is the activity name?", ACT.NAMES: GOTO 2060
2010 GET#2, ACT.CODE
2020 TRIM$ = N22S
2030 GOSUB 2490
2040 ACT.NAMES = TRIM$
2050 PRINT MID.SCRN$
2060 PRINT "Enter the date " ACT.NAMES " starts or occurs on:"
2070 GOSUB 5630
2080 ACT.DATE = DATE: END.DATE = 0
2090 PRINT MID.SCRN$
2100 PRINT "Enter the scheduled time: "
2110 INPUT TIMES: COSUB 5190
2120 IF OTHER THEN GOSUB 4690: GOTO 2230 ELSE IF ACT.CODE = 62 OR ACT.CODE = 61 THEN GOSUB 4820: GOTO 2230
2130 IF OTHER THEN GOSUB 4690: GOTO 2230
2140 'not other and not 61 or 62...
2150 PRINT MID.SCRN$
2160 PRINT "Enter:"
2170 PRINT "0 if standard time offsets apply"
2180 PRINT "1 to change them"
2190 PRINT "Which choice? " A$ = INPUT$(1): PRINT A$
2200 IF A$ = "1" THEN GOSUB 4690: GOTO 2230
2210 IF A$ = "0" THEN PRINT E$: GOTO 2160
2220 START = CVI(N23S): END.T = CVI(N24S)
2230 ACT.ST.TIME = TIME - START: ACT.END.TIME = TIME + END.T
2240 IF END.DATE = 0 THEN END.DATE = ACT.DATE
2250 ACT.LNS = STRING$(25,32)
2260 MID$(ACT.LNS,1,5) = STR$(ACT.CODE)
2270 MID$(ACT.LNS,6,5) = STR$(ACT.DATE)
2280 MID$(ACT.LNS,11,5) = STR$(ACT.ST.TIME)
2290 MID$(ACT.LNS,16,5) = STR$(END.DATE)
2300 MID$(ACT.LNS,21,5) = STR$(ACT.END.TIME)
2310 ACT.LNS$ = ACT.LNS$ + ACT.NAMES
2320 IF N > MAX.N THEN GOSUB 3740
2330 ACT$(N) = ACT.LNS$ 2340 PRINT MID.SCRN$ " Check the activity data:"
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$(1)): PRINT A$
NT A
2440 IF A < 0 OR A > 1 THEN PRINT ES: GOTO 2430
2450 WEND: 'activity entry loop
2460 RETURN
2470
2480
2490 L = LEN(TRIMS) + 1: L.CHRS = CHR$(0)
2500 WHILE ASC(L.CHRS) < 33
2510 L = L - 1
2520 L.CHRS = MID$(TRIMS, L, 1)
2530 WEND
2540 TRIMS = LEFT$(TRIMS, L)
2550 RETURN
2560
2570 'change admin data...
2580 PRINT MIDL.SCRN$
2590 PRINT "Check the pilot data:"
2600 PRINT: PRINT "Pilot number assigned: ": NUM$
2610 PRINT: PRINT L.NAMES ", " F.NAMES ", " MIS ", " RANK$
2620 PRINT: PRINT "Which one? ";: AIS = INPUT$(1): PRINT AIS
2630 IF AIS = "0" THEN GOTO 2760
2640 IF VAL(AIS) > 4 THEN PRINT "Error, enter 0 to 4 only, try again...": GOTO 2690
2650 ON VAL(AIS) GOSUB 4270, 4360, 4400, 4550
2660
2670 'entries are correct, put in key file...
2680 'open key file...
2690 GOSUB 4080
2700 'and save key data...
2710 LSETP N1$ = MKI$(NUM)
2720 LSETP N2$ = L.NAMES
2730 INIT$ = LEFT$(F.NAMES, 1) + LEFT$(MIS, 1)
2740 LSETP N3$ = INIT$
2750 LSETP N4$ = RANK$
2760 PUT#1, NUM
2770 CLOSE#1
2780 'put all data in memory array...
2790 PS$(NUM, 1) = MKI$(NUM): PS$(NUM, 2) = L NAMES$: STRING$(20 - LEN(L.NAMES$), 32): PS$(NUM, 3) = INIT$: PS$(NUM, 4) = RANK$
2800 RETURN
2810
2820 'put all data in individual file (called from many routines)...
59

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$, MAX.N
3000 FOR I = 1 TO MAX.N: PRINT#1, ACTS(I): NEXT
3010 CLOSE
3020 'and rename files...
3030 ON ERROR GOTO 3090
3040 KILL BAK.FILS
3050 ON ERROR GOTO 0
3060 NAME FILENAMES AS BAK.FILS
3070 NAME TIMP.FILS AS FILENAMES
3080 GOTO 3100
3090 IF ERR = 53 THEN RESUME 3050 ELSE GOTO 3050
3100 RETURN
3110 ' change qual data...
3120 PRINT MID.'SCRNS CVI(P$(NUM,1)) " - " P$(NUM,2) P$(NUM,3) ", " P$(NUM,4)
3130 FOR I = 1 TO 15: PRINT I TAB(6) QUAL$(I) QV(I): NEXT
3140 PRINT: PRINT "Enter 0 if all correct or qual number to change"
3150 PRINT "Which number (0-15)?": INPUT ON
3160 IF ON = 0 THEN RETURN ELSE IF ON > 15 THEN PRINT "Error, a number from 0 to 15 only, try again...": GOTO 3170
3170 IF QV(ON) = 1 THEN QV(ON) = 0 ELSE QV(ON) = 1
3180 GOTO 3140
3190 ' update currency dates...
3200 PRINT MID.'SCRNS CVI(P$(NUM,1)) " - " P$(NUM,2) P$(NUM,3) " P$(NUM,4)
3210 FOR I = 1 TO 10: PRINT I TAB(6) EVENTS(I) TAB(28) CUR.DT(I): NEXT
3220 PRINT: PRINT "Enter 0 if correct or item number to change"
3230 PRINT "Which number (0-10)?": INPUT CN
3240 IF CN = 0 THEN RETURN ELSE IF CN > 10 THEN PRINT "Error, enter a number from 0 to 10 only, try again...": GOTO 3260
3250 GOTO 3100
3260 ' returns on zero entry above...
3270 PRINT "Enter the new currency date (accomplished date)
3280 PRINT "Enter the new currency date (accomplished date)", GOSUB 5630
3290 CUR.DT(CN) = DATE: GOTO 3230
3300 ' add, change or delete availability data...
3310 PRINT MID.'SCRNS CVI(P$(NUM,1)) " - " P$(NUM,2) P$(NUM,3) " P$(NUM,4)
3320 PRINT "Enter:"
3330 PRINT " 0 if activity changes completed"
3340 PRINT " 1 to add new activities"
3350 PRINT " 2 to change existing activities"
3360 PRINT " 3 to delete activities"
PRINT"Which choice?": A = VAL(INPUT$(1)). PRINT
A
3400 IF A = 0 THEN RETURN ELSE IF A > 3 THEN PRINT"Error, enter a number from 0 to 3 only, try again.": GOTO 3390
3410 ON A GOSUB 3460, 3510, 3520
3420 GOTO 3330
3430 'return selected with zero response above...
3440 'open def file, get activity entries...
3450 GOSUB 4200: PRINT MID.SCRN$: N = MAX.N: A = 1: GOSUB 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$(1): RETURN
3530 'open data file and read into memory, close...
3540 OPEN "I", 2, FILENAMES
3550 INPUT #2, NUMS, L.NAMES, F.NAMES, MIS, RANKS, SSANS
3570 IF EOF(2) THEN GOTO 3700 ELSE INPUT#2, QUAL.IDS
3580 IF QUAL.ID$ <> "QUALIFICATIONS:" THEN PRINT"Qual data not found":
3590 IF GV(0) <> -1 THEN ERASE GV: DIM GV(15)
3600 FOR I = 1 TO 15: IF EOF(2) THEN GOTO 3700 ELSE INPUT #2, GV(I): NEXT
3610 IF EOF(2) THEN GOTO 3700 ELSE INPUT#2, CUR.IDS
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.IDS, MAX. N
3650 IF ACT.ID$ <> "ACTIVITIES SCHEDULED:" THEN PRINT"Activity data not found":
3660 FOR N = 1 TO MAX.N
3670 IF EOF(2) THEN PRINT"EOF before MAX.N..." MAX.X.N N: DUMMY$ = INPUT$(1): GOTO 3700
3680 LINE INPUT#2, ACT$(N)
3690 NEXT
3700 CLOSE#2
3710 RETURN
3720 'dynamic array size increase...
3730 IF TMP$(0) <> CHRS(255) THEN ERASE TMP$: DIM TMP$(MAX.X.N)
3740 FOR M = 1 TO MAX.N: TMP$(M) = ACT$(M): NEXT
3750 FOR M = 1 TO MAX.N: ACT$(M) = TMP$(M): NEXT
3760 MAX.N = MAX.N + 10
3770 RETURN
3800 'make piinn.dat filenames...
'print all pilot names to screen...
PRINT CLR$;
FOR I = 1 TO 20
NUM = CVI(P$(I,1)): L.NAME$ = P$(I,2): INITS = P$(I,3): RANK$ = P$(I,4)
PRINT USING "###": NUM: PRINT " " LEFT$(L.NAME$,1)
PRINT USING "###": INITS " " RANK$;
NUM = CVI(P$(I+20,1)): L.NAME$ = P$(I+20,2): INITS = P$(I+20,3): RANK$ = P$(I+20,4)
PRINT TAB(27) USING "###": NUM: PRINT " " LEFT$(L.NAME$,11) INIT$ " " RANK$;
NUM = CVI(P$(I+40,1)): L.NAME$ = P$(I+40,2): INITS = P$(I+40,3): RANK$ = P$(I+40,4)
PRINT TAB(55) USING "###": NUM: PRINT " " LEFT$(L.NAME$,11) INIT$ " " RANK$;
NEXT
RETURN
'
get pilot names from key file...
FOR I = 1 TO MAX.PIL.NUM
GET$1, I
NEXT
RETURN
'open and field def files...
OPEN "R", 1, "PILNAM.DEF", 27
FIELD$1, 2 AS N1$: 20 AS N2$: 2 AS N3$: 3 AS N4$
RETURN
OPEN "R", 2, "QUAL.DEF", 22
FIELD$2, 2 AS N21$: 20 AS N22$
RETURN
OPEN "R", 2, "CUREVNT.DEF", 26
RETURN
OPEN "R", 2, "ACT.DEF", 46
RETURN
'correct all name area variables.
FOR I = 1 TO 3: ON I GOSUB 4270, 4360, 4400 NEXT. RETURN

4270 PRINT MID.SCRN$;
4280 INPUT "What is the pilot's last name? ", L.NAMES$;
4290 PRINT MID.SCRN$;
4300 INPUT "What is his first name? ", F.NAMES$;
4310 PRINT MID.SCRN$;
4320 LINE INPUT "Enter his middle initial(s), 'Jr.', etc., or 0 (zero) for none: ", MIS$;
4330 IF MIS$ = "0" THEN MIS$ = ""
4340 RETURN
4350 '
4360 PRINT MID.SCRN$;
4370 INPUT "What is his rank/rank? ", RANK$;
4380 RETURN
4390 '
4400 PRINT MID.SCRN$;
4410 INPUT "What is his service number (SSAN)? ", SSAN$;
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$;
4550 PRINT USING "####"; ACT.CODE, ACT.NAME$;
4560 GET#2, I + GP + 21
4570 ACT.CODE = CVI(N21$);
4580 ACT.NAME$ = N22$;
4590 PRINT TAB(27) USING "####"; ACT.CODE, ACT.NAME$;
4600 GET#2, I + GP + 42
4610 ACT.CODE = CVI(N21$);
4620 ACT.NAME$ = N22$;
4630 PRINT TAB(55) USING "####"; ACT.CODE, ACT.NAME$;
4640 NEXT
4650 PRINT: PRINT "Which activity code? ": INPUT " ", CODE$;
4660 RETURN
4670 '
4680 'other, input start and end time offsets...
4690 PRINT MID.SCRN$;
4700 PRINT "Enter the amount of time (hrs:min) needed prior to the scheduled"
4710 PRINT "activity time (e.g. travel time to a meeting or 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, inc lude debriefing."
4760 PRINT "return travel, etc as applicable"
4770 PRINT: PRINT "How much time?": GOSUB 5450
4780 END.T = DUR
4790 RETURN
4800 'long duration activities--leave, tdy, etc...
4810 PRINT MID. SCRNS
4820 PRINT "What is the ending date of " ACT. NAMES
4830 GOSUB 5630
4840 END.DATE = DATE
4850 RETURN
4860 'sort activities...
4870 SWAP. = -1: LAST = MAX.N - 1
4880 WHILE SWAP.
4890 SWAP. = 0
4900 FOR I = 1 TO LAST
4910 SD1 = VAL(MID$(ACTS(I),6,5)): ST1 = VAL(MID$(ACTS(I),11,5))
4920 SD2 = VAL(MID$(ACTS(I+1),6,5)): ST2 = VAL(MID$(ACTS(I+1),11,5))
4930 IF (SD1 > SD2) OR (SD1 = SD2 AND ST1 > ST2)
4940 THEN GOSUB 5000
4950 NEXT
4960 LAST = LAST - 1
4970 WEND
4980 RETURN
4990 'conflict check, done after activities sorted by sta rt...
5000 TEMP$ = ACT$(I+1)
5010 ACT$(I+1) = ACT$(I)
5020 ACT$(I) = TEMP$
5030 SWAP. = -1
5040 RETURN
5050 'time of day validating routine
WHILE T$ <> "" AND NT < LEN(TIMES)
    NT = NT + 1
    T$ = MIDS(TIMES, NT, 1)
  WEND

IF NT = 0 GOTO 5390
IF NT = LEN(TIMES) THEN NT = LEN(TIMES) - 1: MIN = VAL(RIGHTS(TIMES, 2)) ELSE MIN = VAL(RIGHTS(TIMES, LEN(TIMES) - NT))

HR = VAL(LEFTS(TIMES, NT - 1))
BAD = 0
IF MIN < 0 OR MIN > 59 THEN BAD = -1
IF HR < 0 OR HR > 24 THEN BAD = -1
TIME = HR * 60 + MIN
T$ = CHR$(INT(HR / 10) + 48)
I$ = CHR$((HR MOD 10) + 48)
M$ = CHR$(INT(MIN / 10) + 48)
E$ = CHR$((MIN MOD 10) + 48)
TIMES$ = T$ + I$ + M$ + E$

IF BAD THEN PRINT "Time " TIMES " not understood, please re-enter: ";: INPUT "": TIMES$: GOTO 5190
NT = 0: T$ = "": I$ = "": M$ = "": E$ = "": BAD = 0
HR = 0: MIN = 0
5390 RETURN

--- This routine accepts an input of numbers until a colon is keyed, then allows only two digits up to a value of 60...

CK$ = ""
DIGITS = INPUT$(1)
IF ASC(DIGITS) < 48 OR ASC(DIGITS) > 58 THEN PRINT "Numerical digits or colon (:) only, please re-enter: ";: GOTO 5460

CK$ = CK$ + DIGITS
IF RIGHTS(CK$, 1) <> " " THEN 5460
HR = VAL(LEFTS(CK$, LEN(CK$) - 1))
MIN$ = INPUT$(2)
IF VAL(MINS) > 60 THEN PRINT "Max number of minutes is 60, please re-enter: "; GOTO 5510
MIN = VAL(MINS)
CK$ = CK$ + MIN$
PRINT: PRINT "The interval entered is: " CK$: is this correct?"
PRINT C$: A$ = INPUT$(1): PRINT A$
IF A$ <> "" THEN PRINT "Re-enter interval from beginning: "; GOTO 5450

DUR = HR * 60 + MIN

RETURN

---

This routine accepts an input of numbers until a colon is keyed, then allows only two digits up to a value of 60...

CK$ = ""
DIGITS = INPUT$(1)
IF ASC(DIGITS) < 48 OR ASC(DIGITS) > 58 THEN PRINT "Numerical digits or colon (:) only, please re-enter: ";: GOTO 5460

CK$ = CK$ + DIGITS
IF RIGHTS(CK$, 1) <> " " THEN 5460
HR = VAL(LEFTS(CK$, LEN(CK$) - 1))
MIN$ = INPUT$(2)
IF VAL(MINS) > 60 THEN PRINT "Max number of minutes is 60, please re-enter: "; GOTO 5510
MIN = VAL(MINS)
CK$ = CK$ + MIN$
PRINT: PRINT "The interval entered is: " CK$: is this correct?"
PRINT C$: A$ = INPUT$(1): PRINT A$
IF A$ <> "" THEN PRINT "Re-enter interval from beginning: "; GOTO 5450

DUR = HR * 60 + MIN

RETURN

---
IF MONTH$(0) () CHRS(255) THEN ERASE MONTH$
DIM MONTHS(12)
MONTH$(1) = "JAN"
MONTH$(2) = "FEB"
MONTH$(3) = "MAR"
MONTH$(4) = "APR"
MONTH$(5) = "MAY"
MONTH$(6) = "JUN"
MONTH$(7) = "JUL"
MONTH$(8) = "AUG"
MONTH$(9) = "SEP"
MONTH$(10) = "OCT"
MONTH$(11) = "NOV"
MONTH$(12) = "DEC"

IF FIRST.DAY(0) = 0 THEN ERASE FIRST.DAY
DIM FIRST.DAY(12)

'RESET FIRST.DAY(3..12) IF CORRECTING A DATE...
FIRST.DAY(1) = 1
FIRST.DAY(2) = 32
FIRST.DAY(3) = 60
FIRST.DAY(4) = 91
FIRST.DAY(5) = 121
FIRST.DAY(6) = 152
FIRST.DAY(7) = 182
FIRST.DAY(8) = 213
FIRST.DAY(9) = 244
FIRST.DAY(10) = 274
FIRST.DAY(11) = 305
FIRST.DAY(12) = 335

'--- GET THE DATE ---
INPUT "WHAT IS THE DATE (DAY MONTH YEAR)"'', DATE$
IF DATE$ = "0" THEN DATE = 0: RETURN

'PUT THE DATE CHARS IN INDIVIDUAL VARIABLES...
DIM DS(LEN(DATE$))
FIRST.DLMTR = 0
FOR I.V = 1 TO LEN(DATE$)
DS(I.V) = MIDS(DATE$, I.V, 1)
IF FIRST.DLMTR <> 0 THEN 6180
IF D$(I.V) () CHRS(255) THEN ERASE DS$
DIM D$(LEN(DATE$))
FIRST.DLMTR = 0

FOR I.V = 1 TO LEN(DATE$)
D$(I.V) = MIDS(DATE$, I.V, 1)
IF FIRST.DLMTR <> 0 THEN 6180
IF any char except letters or numbers to delimit...
DLMT = 0

NEXT

'assume the last two chars are the year...

YEAR = VAL(RIGHT$(DATE$, 2))

'find the day...

'if a delimiter was found then day is the value

'before the delimiter, otherwise the day is either

'the first character or the first two characters of

'the string--assume the first two characters if the

'second character is not a letter

IF FIRST.DLMT THEN DAY = VAL(LEFT$(DATE$, FIRST.DLMT - 1)) ELSE IF ASC($2) < 58 THEN DAY = VAL(LEFT$(DATE$, 2)) : FIRST.DLMT = 2 ELSE DAY = VAL(LEFT$(DATE$, 1)) : FIRST.DLMT = 1

'find the month...

'just look at three characters past the day or past

'the first delimiter

'month could be a number or letters

'convert lower case letters to upper

MONTH$ = ""

FOR I.V = 1 TO 3

IF ASC($1(FIRST.DLMT+I.V)) < 58 THEN MON.NUM = -1

IF ASC($1(FIRST.DLMT+I.V)) > 96 THEN $1(FIRST.DLMT+I.V) = CHR$(ASC($1(FIRST.DLMT+I.V))-32)

MONTHS = MONTH$ + $1(FIRST.DLMT+I.V)

NEXT

FOR I.V = 1 TO 3

'MONTH$ is now a string of numbers or letters,

'MON.NUM is TRUE if it is numbers

'found a match

NEXT I.V

FOR J.V = 1 TO 12

IF MID$(MONTH$, J.V, 1) = MID$(MONTH$(I.V), J.V, 1) THEN TEST = -1 ELSE TEST = 0

IF NOT TEST GOTO 6570

one not matching is enough

NEXT J.V

IF TEST THEN MONTH = I.V GOTO 6580

'found a match

NEXT I.V

IF MONTH < 1 OR MONTH > 12 THEN INPUT"Month not understood--enter the month as a one or two digit number (1..12) ", MONTH: GOTO 6580

TO 6580
MONTH is now valid, set MONTHS if reqd...

IF MON.NUM THEN MONTHS = MONTHS(MONTH)

check if this is a leap year...

IF YEAR/4 = INT(YEAR/4) THEN LEAP.YEAR = -1 ELSE LEAP.YEAR = 0

' if so must increment first day values after February...

IF LEAP.YEAR THEN FOR I.V = 3 TO 12: FIRST.DAY(I.V) = FIRST.DAY(I.V) + 1: NEXT

make sure the number of days is valid for the month
compute max days in month...

IF MONTH = 12 THEN MAX.DAYS = "31" ELSE MAX.DAYS = STRS(FIRST.DAY(MONTH + 1) - FIRST.DAY(MONTH))

MAX.DAYS = MIDS(MAX.DAYS,2,2)

then check range

IF DAY (1 OR DAY) VAL(MAX.DAYS) THEN PRINT "Day of month not understood--input day as a number (1..MAX.DAYS)")"; INPUT"

now put it together and see if correct...

DAYS = STRS(DAY): YRS = STRS(YEAR): DATES = DAYS + " + MONTH$ + YR$

PRINT "The date entered is: "; DATES
PRINT: PRINT CS;
A$ = INPUT$(1)

IF A$ = "1" THEN GOTO 5830: 'try again...

IF A$ <> "0" THEN PRINT E$: GOTO 6840

date is valid and checked correct, make the julian date...

julian date form is year digit * 1000 + julian date

DATE = VAL(RIGHT$(STRS(YEAR),1))*1000 + FIRST.DAY(MONTH) + DAY - 1

reset all variables not needed

YEAR = 0: MONTH = 0: DAY = 0: MON.NUM = 0
FIRST.DLMTR = 0: A$ = "": MAX.DAYS = "": YRS = ""
*** SHELL.SET ****************************
program dated 17 May 1983

This program allows entry of the schedule shell data for a given week

variables required:
none

returns:
SHELLnn.DAT files updated and in order

DEFINT A-Z
CLR$ = CHR$(26): DOWN$ = CHR$(10): ESC$ = CHR$(27)
MID.SCRNS = CLR$ + STRING$(6,10)
UP$ = CHR$(11): MOV.LEFT$ = CHR$(8): MOV.RIGHT$ = CHR$(12)
HOME$ = CHR$(30): CLR.LINE$ = ESC$ + "T"
C$ = "Enter: 0 if correct, 1 to change it:"
E$ = "Error, enter 0 or 1 only, try again..."
MAX.PIL.NUM = 60
DIM P$(MAX.PIL.NUM, 4)

get pilot names...
GOSUB 3250
GOSUB 3180
CLOSE

PRINT MID.SCRN$ "Enter:"
PRINT" 0 to quit, all done"
PRINT" 1 to add a new shell data file"
PRINT" 2 to change data in existing data file"
PRINT" 3 to delete a shell data file"
PRINT" Which choice?": SEL = VAL(INPUT$(1)): PRINT T
SEL
IF SEL <= 0 THEN END ELSE IF SEL > 3 THEN PRINT"Error, enter a number 0 to 3 only, try again...": GOTO 430
PRINT MID.SCRN$ "Enter the week starting date (Sunday):"
GOSUB 4370
WK.DATE = DATE: WK.DATES = DATE$
WK.NUM = (WK.DATE MOD 1000)\7: WK.NUM$ = MIDS(STR$(W
K.NUM), 2)
FILENAME$ = "SHELL" + WK.NUM$ + ".DAT"
ON SEL GOSUB 570, 1090, 1840
GOTO 380

new_shell_data

check if file already exists...
ON ERROR GOTO 590
OPEN "I", 1, FILENAME$
590 IF ERR = 53 THEN RESUME 640
600 PRINT FILENAME$ " exists on this disk, confirm you want 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 ES$: GOTO 610
640 ON ERROR GOTO 0
650 CLOSE
660 ' open shell data file as #1...
670 GOSUB 2900
680 ' open activity definition file as #2...
690 GOSUB 3290
700 A$ = 1: N = 0: MAX.N = 10
710 WHILE A$
720 N = N + 1
730 PRINT MID.SCRN$;
740 GOSUB 2280
750 PRINT "Enter:"
760 PRINT " 0 if entries complete"
770 PRINT " 1 if more activities to enter"
780 PRINT "Which one? "; A$ = VAL(INPUT$(1)): PRINT A$
790 IF A$ < 0 OR A$ > 1 THEN PRINT ES$: GOTO 820
800 ' if greater than dimension of variable, then expand it...
810 IF N >= MAX.N THEN GOSUB 2970
820 ACT$(N) = ACT$.LN$
830 WEND: ' activity entry loop...
840 ' save max number activities...
850 MAX.N = N
860 ' entries complete, sort them...
870 GOSUB 3710
880 ' print to file...
890 FOR I = 1 TO MAX.N
900 LSET N9$ = ACT$(I)
910 ' make first word equal to record number...
920 LSET N1$ = MKI$(I)
930 PUT#1, I
940 NEXT
950 ' last entry is all 255 chars...
960 LSET N9$ = STRINGS$(30, 255): LSET N2$ = MKI$(32767): PUT#1
970 CLOSE
980 RETURN
990 ' change_or_add_to_existing_shell_data_files
1090 PRINT MID_SCRNS UK.DATE$ " - " 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$(1)): PRINT  
1160 'add a new activity to shell data file...  
1170 IF A = 0 THEN RETURN  
1180 IF A > 3 THEN PRINT ES: GOTO 1150  
1190 'open shell data file and act.code file...  
1200 GOSUB 2900: GOSUB 3290  
1220 A.T = 0: N = 0  
1230 WHILE A.T <> 32767  
1240 N = N + 1  
1250 GET1, N: SEQ.NUM = CVI(N1$): A.T = CVI(N2$): IF (N <> SEQ.NUM) AND (N <> 1) THEN PRINT"Error in " FILENAME$ " record" N "n  
1260 WEND  
1270 MAX.N = N  
1280 'add a new activity to shell data file...  
1290 ON A COSUB 1330, 1670, 1840  
1300 CLOSE: GOTO 1090  
1310 'ok is first record > than activity to insert...  
1320 IF ACT$(0) <> CHR$(255) THEN ERASE ACT$: DIM ACT$(MAX.N + 1 - K)  
1330 FOR M = K TO MAX.N  
1340 GET1, M  
1350 ACT$(M - K + 1) = N9$  
1360 NEXT  
1370 LSET N9$ = ACT.LNS$ LSET N1$ = MKIS(K)  
1380 PUT1, K  
1390 FOR M = K + 1 TO MAX.N + 1  
1390 LSET N9$ = ACT$(M - K)  
1390 LSET N1$ = MKIS(M)  
1400 PUT1, M  
1410 NEXT  
1420 GOTO 1640  
1430 GET1, K: TMP1$ = N9$  
1440 LSET N9$ = ACT.LNS$ LSET N1$ = MKIS(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$)
1700 IF N < 1 OR N > MAX.N THEN PRINT "Out of range...": GOSUB 1970
1710 ' good sequence number entered...
1720 GET#1, N
1740 PRINT MID.SCRN$: 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$: GOTO 1750
1780 ' add activity entry...
1790 'add activity entry...
1800 A = 1
1810 GOSUB 2280
1820 RETURN
1830 
1840 PRINT"Delete not written yet...". RETURN
1850 
1860 __delete_complete_shell_data_file__________________________
1870 __delete_complete_shell_data_file__________________________
1880 PRINT MID.SCRN$: WK.DATES: FILENAMES:
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 deleted...": 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
N7$  IF A.T () 32767 THEN PRINT S.N A T A C P.N S T E.T A
.N$
2010  M = M + 1
2020  IF M MOD 20 = 1 THEN PRINT"Press (RETURN) to continue
 or sequence number if found"; : INPUT" " , A$: IF A$ = "" THEN GOTO 206
 0 ELSE N = VAL(A$): GOTO 2050
2030  WEND
2040  PRINT"At end of shell data file for " WK.DATES " , "
 FILENAMES: PRINT"Press (RETURN) to start over or sequence
 found:"; : INPUT" " , A$: IF A$ = "" THEN GOTO 1970 ELSE N = V
 AL(A$): GOTO 2050
2050  RETURN
2060  PRINT UPS CLR.LINES;: GOTO 1980
2070  '
2080  'get and confirm pilot number...
2090  PRINT MID.SCRNS;
2100  PRINT"Enter the last name or pilot number:"; : INPUT" ", AN
WERS
2110  IF ASC(LEFT$(ANSWERS, 1)) < 58 THEN NUM = VAL(ANSWERS
 $): THIS.NUM = -1 ELSE NUM = 0: L.NAME$ = ANSWERS
2120  'look for name match...
2130  WHILE NUM < MAX.PIL.NUM AND NOT THIS NUM
2140      NUM = NUM + 1
2150      IF L.NAME$ = LEFT$(PS(NUM, 2), LEN(L.NAME$)) THEN
2160          THIS.NUM = -1 ELSE THIS.NUM = 0
2170  WEND
2180  IF THIS.NUM = 0 THEN GOSUB 3060: PRINT PRINT"Enter
 pilot number:"; : INPUT NUM
2190  IF NUM = 0 THEN GOTO 2250
2190  PRINT MID.SCRNS CVI(PS(NUM, 1)) - " PS(NUM, 2) PS(NU
 M, 3) " PS(NUM, 4)
2200  PRINT"Enter 0 if this is the correct entry; 1 if not c
orrect:"; : THIS.NUM = VAL(INPUT$(1)) PRINT THIS NUM
2210  IF THIS.NUM = 1 THEN THIS.NUM = 0 GOTO 2170
2220  IF THIS.NUM <> 0 THEN PRINT"Error, enter 0 or 1 only
 ..."; GOTO 2190
2220  'have correct number...
2230  PIL.NUM = NUM
2250  RETURN
2240  '
2270  'input a new activity...
2280  PRINT"Enter activity code (? for help): ",
2290  INPUT" ", CODE$:
2300  IF CODE$ = "?" THEN GOSUB 3340 ELSE IF CODE$ = "0" T
HEN GOTO 2780
2310  ACT.CODE = VAL(CODE$)
2320  IF ACT.CODE < 1 OR ACT.CODE > 255 THEN PRINT"Entry i
 out of range..."; GOSUB 3340: GOTO 2310
2330  IF (ACT.CODE AND 63) = 63 THEN OTHER = -1 ELSE OTHER
 = 0
2340  IF OTHER THEN INPUT"What is the activity name? ", AC
 T.NAME$: GOTO 2400
GET#2, ACT.CODE
TRIMS = N22$
GOSUB 2810
ACT NAME$ = TRIMS
PRINT MID.SCRNS;
PRINT" Enter the day " ACT.NAME$ " occurs on:"
PRINT" 1 - Sunday"
PRINT" 2 - Monday"
PRINT" 3 - Tuesday"
PRINT" 4 - Wednesday"
PRINT" 5 - Thursday"
PRINT" 6 - Friday"
PRINT" 7 - Saturday"
PRINT" Which day?";
D = VAL(INPUT$(1))
IF D ( < 1 OR D > 7 THEN PRINT" Error, enter a number 1 to 7 only, try again...": GOTO 2490
PRINT MID.SCRNS:
PRINT" Enter the scheduled time: ";
INPUT TIMES: GOSUB 3920
ACT.SCHED.TIME = (D-1)*1440 + TIME
IF OTHER THEN GOSUB 3580: GOTO 2660
PRINT MID.SCRNS:
PRINT" Enter:"
PRINT" 0 if standard time offsets apply"
PRINT" 1 to change them"
PRINT" Which choice? ";: AS = INPUT$(1): PRINT AS
IF AS = "1" THEN GOSUB 3580: GOTO 2660
IF AS <> "0" THEN PRINT E$; GOTO 2590
START = CVI(N23$): END.T = CVI(N24$)
ACT.ST.TIME = ACT.SCHED.TIME - START: ACT.END.TIME = ACT.SCHED.TIME + END.T
PRINT MID.SCRNS" Is a pilot already assigned to this activity?"
PRINT" Enter 0 if no pilot assigned or the pilot name or number to specify which pilot. ";
INPUT AS
IF AS <> "0" THEN ANSWER$ = AS: GOSUB 2110 ELSE PIL.NUM = 255
ACT.LNS = STRING$(30,0)
MID$(ACT.LNS,3,2) = MKI$(ACT.SCHED.TIME)
MID$(ACT.LNS,5,1) = CHR$(ACT.CODE)
MID$(ACT.LNS,6,1) = CHR$(PIL.NUM)
MID$(ACT.LNS,7,2) = MKI$(ACT.ST.TIME)
MID$(ACT.LNS,9,2) = MKI$(ACT.END.TIME)
MID$(ACT.LNS,11) = ACT.NAME$
RETURN
' trim trailing spaces...
L = LEN(TRIMS) + 1: L CHR$ = CHR$(0)
WHILE ASC(L.CHR$) < 33
L = L - 1
L.CHR$ = MID$(TRIMS,L,1)
2850  WEND
2860  TRIMS = LEFT$(TRIMS, L)
2870  RETURN
2880
2890  'open and field shell data file...
2900  OPEN ">", 1, FILENAMES, 30
2910  "seq.num act.sched.time act.code pil.num act.
2920  st.time act.end.time act.name"
2930  FIELD#1, 2 AS N1$, 2 AS N2$, 1 AS N3$, 1 AS N4$, 2 A
2940  S N5$, 2 AS N6$, 20 AS N7$
2950  FIELD#1, 30 AS N9$
2960  RETURN
2970
2980  'dynamic array size increase...
2990  IF MEM < 320 THEN PRINT "Not enough fre
3000  meory, save this to disk and continue..."; MEM = -1: RETU
3010  RN
3020  IF TMP$(0) <> CHR$(255) THEN ERASE TMP$: DIM TMP$(MAX
3030  X N)
3040  FOR M = 1 TO MAX.N: TMP$(M) = ACT$(M): NEXT
3050  ERASE ACT$: DIM ACT$(MAX.N + 10)
3060  FOR M = 1 TO MAX.N: ACT$(M) = TMP$(M): NEXT
3070  MAX.N = MAX.N + 10
3080  RETURN
3090
3100  'print all pilot names to screen...
3110  PRINT CLR$
3120  FOR I = 1 TO 20
3130  NUM = CVI$(PS(I, 1)): L.NAMES = PS(I, 2): INIT$ = PS(I,
3140  3): RANK$ = P$(I, 4)
3150  PRINT USING "##": NUM;: PRINT "" LEFT$(L.NAMES, 1
3160  1) INIT$ " RANK$;
3170  NUM = CVI$(PS(I+20, 1)): L.NAMES = PS(I+20, 2): INIT$ =
3180  P$(I+20, 3): RANK$ = P$(I+20, 4)
3190  PRINT TAB(27) USING "##": NUM;: PRINT "" LEFT$(L
3200  NAMES, 11) INIT$ " RANK$;
3210  NUM = CVI$.P$(I+40, 1)): L.NAMES = PS(I+40, 2): INIT$ =
3220  P$(I+40, 3): RANK$ = P$(I+40, 4)
3230  PRINT TAB(55) USING "##": NUM;: PRINT "" LEFT$(L
3240  NAMES, 11) INIT$ " RANK$;
3250  NEXT
3260  RETURN
3270
3280  'get pilot names from key file...
3290  FOR I = 1 TO MAX.PIL.NUM
3300  GET#1, I
3310  PS(I, 1) = N1$: PS(I, 2) = N2$: PS(I, 3) = N3$: PS(I,
3320  4) = N4$
3330  NEXT
3340  RETURN
3350
3360  'open and field def files...
3370  OPEN ">", 1, "PILNAM.DEF", 27
3380  FIELD#1, 2 AS N1$, 20 AS N2$, 2 AS N3$, 3 AS N4$
3390  RETURN
OPEN "R", 2, "ACT. DEF", 46
FIELD #2, 2 AS N21%, 20 AS N22%, 2 AS N23%, 2 AS N24%
, 10 AS N25%, 10 AS N26%
RETURN

'read in activity codes and names, assumes def file
open as #2...
PRINT "Select the desired activity category"
PRINT " 1 for non-duty (leave, TDY, etc)"
PRINT " 2 for non-flying duty activities"
PRINT " 3 for flying activities"
PRINT "Which category? "; A = VAL(INPUT$(1)): PRINT A
GP = (A - 1) * 64
FOR I = 1 TO 21
GET#2, I + GP
ACT CODE = CVI(N2I%)
ACT NAMES = N22%
PRINT USING "###"; ACT.CODE; PRINT " - "; ACT.NAME$
GET#2, I + GP + 21
ACT CODE = CVI(N2I%)
ACT NAMES = N22%
PRINT TAB(27) USING "###"; ACT.CODE; PRINT " - "
ACT NAMES$
GET#2, I + GP + 42
ACT CODE = CVI(N2I%)
ACT NAMES = N22%
PRINT TAB(55) USING "###"; ACT.CODE; PRINT " - "
ACT NAMES$
NEXT
PRINT: PRINT "Which activity code? "; INPUT " ", CODE$
RETURN

'other, input start and end time offsets...'
PRINT MID. SCRNS$
PRINT "Enter the amount of time (hrs min) needed prior
to the scheduled"
PRINT "activity time (e.g. travel time to a meeting or
briefing time)"
PRINT "How much time? "; GOSUB 4190
START = DUR
PRINT MID. SCRNS$
PRINT "Enter the amount of time for the activity, inc
lude debriefing, "
PRINT "return travel, etc as applicable"
PRINT "How much time? "; GOSUB 4190
END T = DUR
RETURN

'sort activities...
SWAP = -1; LAST = MAX.N - 1
PRINT MID. SCRNS; "Sorting"
WHILE SWAP
76

3740  SWAP. = 0
3750  FOR I = 1 TO LAST
3760     A.T1 = CVI(MIDS(ACTS(I),3,2))
3770     A.T2 = CVI(MIDS(ACTS(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$ = ACTS(I+1)
3860  ACTS(I+1) = ACTS(I)
3870  ACTS(I) = TMP$
3880  SWAP. = -1
3890 RETURN
3900 'time of day validating routine...
3910  NT = 0
3920  T$ = ""
3930  WHILE T$ <> "":" AND NT < LEN(TIMES)
3940     NT = NT + 1
3950     T$ = MID$(TIMES,NT,1)
3970  WEND
3980  IF NT = 0 GOTO 4120
3990  IF NT = LEN(TIMES) THEN NT = LEN(TIMES) - 1: MIN = VAL(RIGHT$(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, please re-enter ";: INPUT" ", TIMES$: GOTO 3920
4110  NT = 0: T$ = "": I$ = "": M$ = "": E$ = "": BAD = 0: HR = 0: MIN = 0
4120 RETURN
4130
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  DIGITS$ = INPUT$(1)
4210  IF ASC(DIGITS$)<48 OR ASC(DIGITS$)>58 THEN PRINT "Numerical digits or colon (:) only, please re-enter ";: GOTO 4200
4220  CK$ = CK$ + DIGITS$
IF RIGHTS(CK$, 1) <> "." THEN 4200
HR = VAL(LEFT$(CK$, LEN(CK$) - 1))
MIN$ = INPUT$(2)
IF VAL(MIN$) > 60 THEN PRINT "Max number of minutes is 60, please re-enter: "; GOTO 4250
MIN = VAL(MIN$)
CK$ = CK$ + MIN$
PRINT: PRINT "The interval entered is: ", CK$,", is this correct?"
PRINT CS$; A$ = INPUT$(1): PRINT A$
IF A$ = "1" THEN PRINT "Re-enter interval from beginning: "; GOTO 4190
IF A$ <> "0" THEN GOTO 4300
DUR = HR*60 + MIN
RETURN

'--- get the date ---
INPUT "What is the date <Day Month Year>", DATES
IF DATES = "0" THEN DATE = 0: RETURN

IF MONTH$(0) <> CHR$(255) THEN ERASE MONTH$(12)
DIM MONTH$(12)
MONTH$(1) = "JAN"
MONTH$(2) = "FEB"
MONTH$(3) = "MAR"
MONTH$(4) = "APR"
MONTH$(5) = "MAY"
MONTH$(6) = "JUN"
MONTH$(7) = "JUL"
MONTH$(8) = "AUG"
MONTH$(9) = "SEP"
MONTH$(10) = "OCT"
MONTH$(11) = "NOV"
MONTH$(12) = "DEC"

IF FIRST.DAY(0) = 0 THEN ERASE FIRST.DAY
DIM FIRST.DAY(12)

' reset FIRST.DAY(3..12) if correcting a date...
FIRST.DAY(1) = 1
FIRST.DAY(2) = 32
FIRST.DAY(3) = 60
FIRST.DAY(4) = 91
FIRST.DAY(5) = 121
FIRST.DAY(6) = 152
FIRST.DAY(7) = 182
FIRST.DAY(8) = 213
FIRST.DAY(9) = 244
FIRST.DAY(10) = 274
FIRST.DAY(11) = 305
FIRST.DAY(12) = 335

'--- get the date ---
INPUT "What is the date <Day Month Year>", DATES
IF DATES = "0" THEN DATE = 0: RETURN
'put the date chars in individual variables...

IF DS(0) <> CHRS(255) THEN ERASE DS
DIM DS(LEN(DATES))
FIRST.DLMTR = 0

FOR I.V = 1 TO LEN(DATES)
    DS(I.V) = MIDS(DATES, I.V, 1)
    IF FIRST.DLMTR <> 0 THEN 4930
    IF first delimiter not set, look for it; allow
    almost any char except letters or numbers to
    'delimit...
    IF (ASC(D$(I.V)) > 48 THEN DLMT = -1
        IF (D) > 57 AND D < 65 THEN DLMT = -1
        IF (D) > 90 AND D < 96 THEN DLMT = -1
        IF DLMT THEN FIRST.DLMTR = I.V
        DLMT = 0
NEXT

'assume the last two chars are the year...

YEAR = VAL(RIGHTS(DATES, 2))

'find the day...

IF a delimiter was found then day is the value
before the delimiter, otherwise the day is either
the first character or the first two characters of
the string--assume the first two characters if the
'second character is not a letter

IF FIRST.DLMTR THEN
    DAY = VAL(LEFTS(DATES, FIRST.DLMTR - 1))
ELSE IF ASC(D$(2)) < 58 THEN
    DAY = VAL(LEFTS(DATES, 2))

MONTHS-""
MONNUM = 0

FOR I.V = 1 TO 3
    IF ASC(D$(FIRST.DLMTR + I.V)) < 58 THEN MON.NUM = -1
    IF ASC(D$(FIRST.DLMTR + I.V)) > 96 THEN DS(FIRST.DLMTR + I.V) = CHRS(ASC(D$(FIRST.DLMTR + I.V)) - 32)
    MONTHS = MONTHS + DS(FIRST.DLMTR + I.V)
NEXT

'MONTH$ is now a string of numbers or letters,
MON.NUM is TRUE if it is numbers...
79

5250 IF MON.NUM THEN MONTH=VAL(LEFT$(MONTH$,2)): GOTO 533
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
5300 NEXT J.V
5310 IF TEST THEN MONTH = I.V: GOTO 5330: ' found
5320 NEXT I.V
5330 IF MONTH < 1 OR MONTH > 12 THEN INPUT"Month not under
5340 "stood--enter the month as a one or two digit number (1..12)
5345 "": MONTH: GO
5350 TO 5330
5360 IF "MONTH is now valid, set MONTHS if reqd...
5370 IF MON.NUM THEN MONTHS = MONTH$(MONTH)
5380 IF "check if this is a leap year...
5390 TEST = 0
5400 IF YEAR/4 = YEAR\4 THEN LEAP.YEAR = -1 ELSE LEAP.YEAR
5410 R = 0
5420 IF so must increment first day values after
5430 "February...
5440 "February...
5450 IF LEAP.YEAR THEN FOR I.V = 3 TO 12: FIRST.DAY(I.V)
5460 = FIRST.DAY(I.V) + 1: NEXT
5470 'make sure the number of days is valid for the month
5480 'first, compute max days in month...
5490 IF MONTH = 12 THEN MAX.DAY$ = " 31" ELSE MAX.DAY$ =
STR$(FIRST.DAY(MONTH + 1) - FIRST.DAY(MONTH))
5500 MAX.DAY$ = MID$(MAX.DAY$,2,2)
5510 IF DAY (1 OR DAY) = VAL(MAX.DAY$) THEN PRINT "Day of
5520 "month not understood--input day as a number (1.."
5530 MAX.DAY$ "")": INPUT"...
5540 "., DAY
5550 'now put it together and see if correct...
5560 'DAY$ = STR$(DAY): YR$ = STR$(YEAR): DATE$ = DAY$ + "
5570 " + MONTH$ + YR$
5580 PRINT"The date entered is: "; DATE$
5590 PRINT: PRINT C$;
5600 A$ = INPUT$(1)
5610 IF A$ = "1" THEN GOTO 4570: 'try again...
5620 IF A$ <> "0" THEN PRINT E$: GOTO 5580
5630 'date is valid and checked correct, make the julian
5640 date...
' julian date form is year digit * 1000 + julian date
5670
5680  DATE = VAL(RIGHT$(STR$(YEAR),1))*1000 + FIRST.DAY(MONTH) + DAY - 1
5690
5700  'reset all variables not needed
5710
5720  ERASE D$
5730  YEAR = 0: MONTH = 0: DAY = 0: MON.NUM = 0
5740  FIRST.DLMTR = 0: AS = "": MAX.DAYS = ""
5750  DAYS = "": MONTHS = "": YRS = ""
5760  RETURN
5770
' *** WKDAT.SET ***********************************************
' program dated 17 May 1983
' This program reads all pilnn.dat files and sets
' the weekly data in WKnn.DAT
' variables required:
' none
' returns:
' WKnn.DAT file
DEFINT A-Z
CLR$ = CHR$(26): DOWN$ = CHR$(10): ESC$ = CHR$(27)
MID.SCRN$ = CLR$ + STRING$(8,10)
UPS$ = CHR$(11): MOV.LEFT$ = CHR$(9): MOV.RIGHT$ = CHR$(12)
HOMES$ = CHR$(30): CLR.LINES = ESC$ + "T"
' set avail period constants
PERIOD.ST.TIME = 0: PERIOD.DUR = 10080: INCR = 30
'SD = 6: ST = 11: ED = 16: ET = 21
'fndt.tim pulls the substring value from acts
DEF FNDT.TIM(NL,P) = VAL(MIDS(ACTS(NL),P,5))
C$ = "Enter: 0 if correct, 1 to change it: "
E$ = "Error, enter 0 or 1 only, try again...
MAX.PIL.NUM = 60
IF P$(0,0) <> CHR$(255) THEN ERASE P$: DIM P$(MAX.PIL.NUM,4)
IF QUAL$(0) <> CHR$(255) THEN ERASE QUAL$: DIM QUAL$(15)
' open and field def files...
OPEN "R", 1, "PILNAM.DEF", 27
pi1.num: i.name$: init$: rank$: nlt$: FILD#1, 2 AS N1$, 20 AS N2$, 2 AS N3$, 3 AS N4$, 450 ,
FILD#2, 93 AS N2$, , 28 AS N2$, 42 AS N2$, 20 AS N2$, 2 AS N2$, 28 AS N2$, 3 AS N2$, 530 ,
OPEN "R", 2, "WK" + WK.NUM$ + " DAT", 93
pl1.num: avail$: cur.dt$: qual: net/nlt:,
FILD#2, 1 AS N21$, 42 AS N22$, 20 AS N23$, 2 AS N2$, 540 FILD#2, 93 AS N2$, 550 ,
FOR PIL.NUM = 1 TO MAX.PIL.NUM
FOR I = 1 TO 7: NET(I) = -1: NLT(I) = -1: NE
IT
380 GET#1, PIL.NUM
390 NUM = CVI(N1$) : TRIM$ = N2$ : GOSUB 1460 L.N
AME$ = TRIM$ : INIT$ = N3$ : RANK$ = N4$
600 PRINT "Getting data on " RANK$ " ' L. NAMES $", " INIT$;
610 IF NUM () PIL.NUM THEN PRINT "File error: record number not equal to pilot number": PRINT "Press any key to continue... "; DUMMY$ = INPUT$(1)
620 NUM$ = MID$(STR$(NUM),2)
630"
640 NO.FILE = 0
650 ON ERROR GOTO 680
660 FILENAMES = "PIL" + NUM$ + ".DAT"
670 OPEN "I", 3, FILENAMES
680 IF ERR = 53 THEN NO.FILE = -1: PRINT FILENAMES " not found, going to next number... ": RESUME 690
690 ON ERROR GOTO 0
700 IF NO.FILE THEN LSET N21$ = STRING$(93,0) : LSET N21$ = 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$(N),5))
830 ACT.ST.DATE = FNDT.TIM(N,SD)
840 ACT.ST.TIME = FNDT.TIM(N,ST)
850 ACT.END.DATE = FNDT.TIM(N,ED)
860 ACT.END.TIME = FNDT.TIM(N,ET)
870"
880 ' compute times in minutes from week beginning...
890 IF ACT.ST.DATE - WK.DATE > 7 THEN STA
900 ART.TIME = 32767 ELSE IF ACT.ST.DATE - WK.DATE < -7 THEN STA
910 RT TIME = -1
920 ELSE START.TIME = (ACT.ST.DATE - WK.DATE)*1440 + ACT.ST
930 TIME
940 IF ACT.END.DATE - WK.DATE > 7 THEN E
950 ND.TIME = 32767 ELSE IF ACT.END.DATE - WK.DATE < -7 THEN END
960 TIME = -100
970 ELSE END.TIME = (ACT.END.DATE - WK.DATE)*1440 + ACT.END.T
980 IME
990 ' set defined FALSE, just check activity times...
1000 AVAIL=-1: SET=0
1010 GOSUB 3380
1020 IF NOT AVAIL THEN PRINT "Conflict in activity" N CHR$(8) ", not set."
IF (ACT.CODE AND 192) <> 0 THEN GOSU
83
5:50 ELSE C.REST = -1
960 IF C.REST THEN GOTO 1030
980 PRINT "Activity" N "ACTS(N) does not meet crew rest constraints..."
990 PRINT Enter: 0 to ignore crew rest, 1 to NOT set this activity: ";
1000 IF AS = "I" THEN GOTO 1040 ELSE IF A $ <> "0" THEN PRINT E$: GOTO 980
1020 SET defined TRUE, set this activity in available...
1030 AVAIL = -1: SET = -1: GOSUB 3810
1040 NEXT: 'activity for this pilot number...
1050 LSET N22$ = AVAIL$
1060 CUR.DTS$ = ""
1070 FOR N = 1 TO 10
1080 CUR.DTS$ = CUR.DTS$ + MKIS(CUR.DT(N))
1090 NEXT: 'currency event date...
1100 LSET N23$ = CUR.DTS$
1110 QUAL = 0
1120 FOR N = 1 TO 15
1130 IF QV(N) THEN QUAL = QUAL + 2 * (N-1)
1140 NEXT: 'qualification...
1150 LSET N24$ = MKIS(QUAL)
1160 'initialize all NET and NLT times as -1 values...
1170 NET.NLTS$ = STRING$(28,255)
1180 FOR N = 1 TO 7
1190 IF NET(N) <> -1 THEN MIDS(NET.NLTS$,N *4 - 3,2) = MKIS(NET(N))
1200 NEXT
1210 IF NLT(N) <> -1 THEN MIDS(NET.NLTS$,N *4 - 1,2) = MKIS(NLT(N))
1220 NEXT
1230 LSET N25$ = NET.NLTS$
1240 'save all data in buffer to this pilot number record
1250 PUT#2, FIL.NUM
1260 NEXT: 'pilot number...
1270 'last record, save the date (julian number and string form)
1280 rec.num: wk.date: wk.date$: not used
1290 FIELD#2, 1 AS N221$, 2 AS N222$, 9 AS N223$, 81 AS N
1300 FIELD#2, 1 AS N221$, 2 AS N222$, 9 AS N223$, 81 AS N
1310 LSET N221$ = CHR$(MAX.PIL.NUM + 1)
1320 LSET N222$ = MKIS(WK.DATE)
1330 LSET N223$ = WK DATE$
1340 LSET N224$ = STRING$(81,0)
1350 PUT#2, MAX.PIL.NUM + 1
1360 'all pilot data for the week now in one file
1370 PRINT "WK" WK.NUM$: "DAT file now completed. "
1380
1420 END
1430 'Subroutines
1440 L = LEN(TRIMS) + 1: CHRS = CHR$(0)
1460 WHILE ASC(L.CHRS) < 33
1480 L = L - 1
1490 L.CHRS = MID$(TRIMS, L, 1)
1500 WEND
1510 TRIMS = LEFT$(TRIMS, L)
1520 RETURN
1530 'Open data file and read into memory, close...
1550 INPUT #3, NUM$, L.NAMES, F.NAMES, MIS, RANKS, SSANS
1560 IF EOF(3) THEN GOTO 1690 ELSE INPUT#3, QUAL.IDs
1570 IF QUAL.IDs <> "QUALIFICATIONS:" THEN PRINT "Qual data 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.IDs
1610 IF CUR.IDs <> "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.IDs, MAX.N
1640 IF ACT.IDs <> "ACTIVITIES SCHEDULED:" THEN PRINT "Activity data not found":
1650 FOR N = 1 TO MAX.N
1660 IF EOF(3) THEN PRINT "EOF before MAX.N..." MAX.N N: DUMMYS = INPUT$(1): GOTO 1680
1670 LINE INPUT#3, ACT$(N)
1680 NEXT
1690 RETURN
1700 'Open data file and read into memory, close...
1710 'Open data file and read into memory, close...
1720 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 IF FIRST.DAY(0) = 0 THEN ERASE FIRST.DAY
1890 DIM FIRST.DAY(12)
1900  'reset FIRST.DAY(3..12) if correcting a date...
1910
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) = MIDS(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$(DATES, FIRST.DLMTR R - 1)) ELSE IF ASC(D$(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$(FIRST_DLMTR+I.V)) < 58 THEN MON.NUM
2540 M = -1
2550 NEXT
2560 
2570 'MONTH$ is now a string of numbers or letters.
2580 ' MON.NUM is TRUE if it is numbers
2590 
2600 IF MON.NUM THEN MONTH$ = VAL(LEFT$(MONTH$, 2)) : GOTO 2690
2610 FOR I.V = 1 TO 12
2620 FOR J.V = 1 TO 3
2630 IF MID$(MONTH$, J.V, 1) = MID$(MONTH$(I.V), J.V, 1) THEN TEST = -1 ELSE TEST = 0
2640 IF NOT TEST GOTO 2680: ' one not mat
2650 ching is enough
2660 NEXT J.V
2670 IF TEST THEN MONTH$ = I.V: GOTO 2690: ' found
2680 a match
2690 NEXT I.V
2700 
2710 IF MONTH < 1 OR MONTH > 12 THEN INPUT"Month not undere
2720 stood--enter the month as a one or two digit number (1..12)
2730 ", MONTH: GO TO 2690
2740 
2750 'MONTH is now valid, set MONTH$ if reqd...
2760 
2770 IF MON.NUM THEN MONTH$ = MONTH$(MONTH$)
2780 
2790 'check if this is a leap year...
2800 
2810 IF YEAR/4 = YEAR/4 THEN LEAP.YEAR = -1 ELSE LEAP.YEA
2820 R = 0
2830 
2840 'if so must increment first day values after
2850 February...
2860 
2870 IF LEAP.YEAR THEN FOR I.V = 3 TO 12, FIRST.DAY(I.V)
2880 = FIRST.DAY(I.V) + 1: NEXT
2890 
2900 'make sure the number of days is valid for the month
2910 'first, compute max days in month...
2860 IF MONTH = 12 THEN MAX.DAYS = "31" ELSE MAX.DAYS = STR$(FIRST.DAY(MONTH - 1) - FIRST.DAY(MONTH))
2870 IF MAX.DAYS = MID$(MAX.DAYS, 2, 2) 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 "
2900 , DAY
2920 'now put it together and see if correct...
2930 DAYS = MID$(STR$(DAY), 2): YRS = STR$(YEAR): DATES = DAY$ + "" + MONTH$ + YR$
2940 PRINT "The date entered is":", DATES
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$(YEAR), 1)) * 1000 + FIRST.DAY(MONTH) + DAY - 1
3050 ,
3060 'reset all variables not needed
3070 ,
3080 ERASE D$
3090 YEAR = 0: MONTH = 0: DAY = 0: MON.NUM = 0
3100 FIRST.DLMTR = 0: A$ = "": MAX.DAYS = "":
3110 DAY$ = "": 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 other 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 .
returns:
AVAIL as TRUE if time is available
AVAIL updated if AVAIL and SET both TRUE

'subroutines used:
all internal

GOSUB 3600
GOSUB 3810
RETURN

--- This routine determines the case of activity
start (CASE1) and end (CASE2) relative to
the period start and end...

CASE1 and CASE2 equal 1 if times are before
the period starts, 2 if during the period,
or 3 if after the period. Thus if CASE1 is
3 or CASE2 is 1, the whole activity falls
outside the period in question. If both
CASE1 and CASE2 are 2, then the whole
activity is within the period

CASE3 has a value of 1 if the whole
activity falls on a single byte, 2 if on
adjacent bytes, and 3 if one or more whole
bytes fall between the start and end.

```
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 COND2 = 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 COND3 = 0
3680 IF END.TIME <= (PERIOD.ST.TIME+PERIOD.DUR) THEN COND4 = -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
```

This routine selects the proper routine for
checking or setting availability based on the case
defined by CASE1, CASE2, and CASE3

88
if start is before period...
IF CASE1=1 AND CASE2=2 THEN ON CASE3 GOSUB 3970,4070

if start and end are during period...
IF CASE1=2 AND CASE2=2 THEN ON CASE3 GOSUB 4200,4370

if start is during period but end is after...
IF CASE1=2 AND CASE2=3 THEN ON CASE3 GOSUB 4550,4680

if start is during period but end is after period...
IF CASE1=1 AND CASE2=3 THEN FIRST.BYT=1: LAST.BYT=LE
N(AVAILS): GOSUB 4720

the final case ends before or starts after
period...
IF CASE1=3 OR CASE2=I THEN PRINT"Activity is complet
ely outside the period..."
RETURN

--- This routine is used when END_BYTE$ is the
first byte of AVAIL$...

FIRST.BIT.USED = 0: LAST.BIT.USED = (END.BIT MOD 8)
BYT.TOCK$ = LEFT$(AVAIL$,1)
GOSUB 4940
IF AVAIL AND SET THEN MID$(AVAIL$,1,1) = CHR$(ASC(BY
T.TOCK$) AND (NOT MASK))

--- This routine is used when END BYTE points to
each byte...
FIRST.BYT = 1: LAST.BYT = END.BYTE-1
GOSUB 4820
FIRST.BIT.USED = 0: LAST.BIT.USED = (END.BIT MOD 8)
BYT.TOCK$ = MID$(AVAIL$,END.BYTE,1): J = END.BYTE
GOSUB 4940
IF AVAIL AND SET THEN GOSUB 5040 ELSE RETURN
MID$(AVAIL$,END.BYTE,1) = CHR$(ASC(BYT.TOCK$) AND (NOT
MASK))

--- This routine is used for the single byte case
where one byte includes both start and end...
BYT.TOCK$ = MID$(AVAIL$,START BYTE,1)
MASK = 0
FIRST.BIT.USED = (START.BIT MOD 8)
LAST.BIT.USED = (END.BIT MOD 8)
GOSUB 4940
IF AVAIL AND SET THEN MID$(AVAIL$,START BYTE,1) = CH
This routine is used when one or more bytes separate the first and last bytes or when they are adjacent.

FIRST.BYT = START.BYTE+1: LAST.BYT = END.BYTE-1
GOSUB 4820
IF NOT AVAIL THEN RETURN
CASE3 = 2 enters here...
FIRST.BIT.USED = (START.BIT MOD 8): LAST.BIT.USED = 7
BYT.TO.CKS = MIDS(AVAILS,START.BYTE,1)
GOSUB 4940
IF AVAIL AND SET THEN MASK ST=MASK, BYT.ST$=BYT.TO.CKS ELSE IF NOT AVAIL THEN RETURN
FIRST.BIT.USED = 0: LAST.BIT.USED = (END.BIT MOD 8)
BYT.TO.CKS = MIDS(AVAILS,END.BYTE,1): J = END.BYTE
GOSUB 4940
IF NOT(AVAIL AND SET) THEN RETURN
MIDS(AVAILS,START.BYTE,1) = CHR$(ASC(BYT.ST$) AND (NOT MASK))
RETURN

--- This routine is used when the last byte in the string is the only byte to be checked...

BYT.TO.CKS = MIDS(AVAILS,START.BYTE,1)
FIRST.BIT.USED = (START.BIT MOD 8): LAST.BIT.USED = (PERIOD.END.BIT MOD 8)
GOSUB 4940
IF AVAIL AND SET THEN MIDS(AVAILS,START.BYTE,1) = CHR$(ASC(BYT.TO.CKS) AND (NOT MASK))
RETURN

--- This routine is used when the activity ends after the period and the first byte is one or more bytes from the end of AVAILS. The last two cases of CASE3 are both checked by this routine...

FIRST.BIT.USED = (START.BIT MOD 8) LAST.BIT.USED = 7
BYT.TO.CKS = MIDS(AVAILS,START.BYTE,1)
GOSUB 4940
FIRST.BYT = START.BYTE+1: LAST.BYT = LEN(AVAILS)
GOSUB 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 availability...
4800
4810
4820  FOR J = FIRST.BYT TO LAST.BYT
4830  BYT.TO.CK$ = MID$(AVAIL$,J,1)
4840  IF BYT.TO.CK$ () CHR$(255) THEN FIRST.BIT.US ED=0: LAST.BIT.US ED=7: GOSUB 4940
4850  IF NOT AVAIL THEN RETURN
4860  NEXT
4870  RETURN
4880
4890
4900  --- This routine is called by above routines to
4910  ' check availability within partial bytes of
4920  AVAILS...
4930
4940  MASK = 0
4950  FOR K = FIRST.BIT.US ED TO LAST.BIT.US ED
4960  MASK = MASK + 2*K
4970  IF (ASC(BYT.TO.CK$) 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
5190  K = -1 ELSE ST.CK = 0
5200
5210  IF END.TIME <= NLT(DAY) OR NLT(DAY) = -1 THEN END.CK
5220  = -1 ELSE END.CK = 0
5230
5240  IF ST.CK AND END.CK THEN C.REST = -1 ELSE C.REST = 0
5250  RETURN
5220 IF (START.TIME - 720 < NLT(DAY-1)) OR (NLT(DAY-1) = -1)
5230 THEN NLT(DAY-1) = START.TIME - 720
5240 IF (NLT(DAY) > START.TIME+720) OR (NLT(DAY) = -1) THEN
5250 NLT(DAY) = START.TIME + 720
5260 IF (NET(DAY) < END.TIME-720) OR (NET(DAY) = -1) THEN
5270 NET(DAY) = END.TIME - 720
5280 IF (NET(DAY+1) < END.TIME+720) OR (NET(DAY+1) = -1) THEN NET(DAY+1) = END.TIME + 720
5290 RETURN
program dated 25 May 1983

This program builds the bare schedule file from shellnn.dat and wknn.dat

variables required:
none

returns:
SCHEDnn.xxx file

DEFINT A-Z

CLRS = CHR$(26): DOWN$ = CHR$(10): ESC$ = CHR$(27)

MID.$ = CLRS + STRING$(6,10)

UP$ = CHR$(11): MOV.LEFT$ = CHR$(8): MOV.RIGHT$ = CHRS(12)

HOME$ = CHR$(30): CLR LINES = ESC$ + "T"

PERIOD.ST.TIME = 0: PERIOD.DUR = 10080: INCR = 30: M

DIM QUAL(MAX.PIL.NUM), AVAIL$(MAX.PIL.NUM), CUR.DT(M

DIM PILOTS(253), ACT.CUR.QUALS(235)

ACT.CODE.USED$ = STRINGS(32,0)

CS = "Enter: 0 if correct, 1 to change it:"

E$ = "Error, enter 0 or 1 only, try again..."

PRINT MID.SCRNS "Enter the week number; ":
INPUT "", WK.NUMS

WKDAT.FILES = "WK" + WK.NUMS + ".DAT"

ON ERROR GOTO 410

IF ERR = 53 THEN PRINT:"No " WKDAT.FILES " found, can not continue..." ELSE CLOSE

ON ERROR GOTO 0

OPEN "R", 1, WKDAT.FILES, 93

FIELD#1, 1 AS N1$, 2 AS N2$, 9 AS N3$, 81 AS N4$

N = MAX.PIL.NUM + 1

GET#1, N

WK.DATE = CVI(N2$): WK.DATES = N3$

CLOSE

SCHED NUM = 0

FIL.NAM.FOUND = 0

WHILE NOT FIL.NAM.FOUND

SCHED NUM = SCHED NUM + 1

SCHED NUM$ = MID$(STR$(SCHED NUM), 2)

WHILE LEN(SCHED NUM$) < 3

SCHED NUM$ = "0" + SCHED NUM$

WEND

FILENAME$ = "SCHED" + WK NUM$ + " " + SCHED NUM$

check if file already exists
94

610  ON ERROR GOTO 630
620  OPEN "I", 1, FILENAME$ 
630  IF ERR = 53 THEN FILE.NAM.FOUND = -1. RESUME 650 ELSE CLOSE1

640  'no error indicates file was found and opened, try again
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 PRINT NT E$: GOTO 670
700  
710  
720  
730  OPEN "R", 1, FILENAME$, 58
740  ' seq.num: act.sched.time: act.code: pil.num: 
750  st.time: end.time: act.name: pilots: cur.req: qual.req:
760  FIELDS1, 2 AS N11$, 2 AS N12$, 1 AS N13$, 1 AS N14$, 
770  2 AS N15$, 2 AS N16$, 20 AS N17$, 8 AS N18$, 10 AS N19$, 10 AS N10$
780  FIELDS0, 58 AS N1A$
790  SHELL.FILE$ = "SHELL" + WK.NUM$ + ".DAT"
800  'open shell data file as #2...
810  OPEN "R", 2, SHELL.FILE$, 30
820  ' seq.num: act.sched.time: act.code: pil.num: 
830  act st.time: act.end.time: act.name: 
840  FIELDS2, 2 AS N21$, 2 AS N22$, 1 AS N23$, 1 AS N24 
850  2 AS N25$, 2 AS N26$, 20 AS N27$
860  FIELDS2, 2 AS N22$
870  M = 0: N = 0: END.FIL = 0: ACT.SCHED.TIME = 0: MAX.N = 20
880  DIM ACTS(19)
890  'following string has bits 0 thru 59 'ON'
900  ALL.PILOTS = STRINGS$(7, 255) + CHR$(15)
910  'open act.def file as #3 for currency and qual reqs...
920  OPEN "R", 3, "ACT.DEF", 46
930  FIELDS3, 2 AS N31$, 20 AS N32$, 2 AS N33$, 2 AS N34$
940  , 10 AS N35$, 10 AS N36$
950  PRINT "Getting shell data from " SHELL.FILE$ and saving in " FILENAME$ 
960  WHILE NOT END.FIL
970  'get shell records 20 at a time or until end
980  found...
990  WHILE (ACT.SCHED.TIME <> 32767) AND (N < MAX N)
1000  
1010  N = N + 1
GETO2, N

'save temporarily in act$, make len equal to new
rec len by appending null chars...

ACT$(N MOD 20) = N2A$ + STRING$(58 - LEN(N2A$), 0)

ACT SCHED. TIME = CVI(N22$)

WEND

IF ACT SCHED. TIME = 32767 THEN END

' save next 20 or all remaining act$ in schednn.xxx

WHILE M < N

M = M + 1

ACT CODE = ASC(MIDS(ACTS(M MOD 20), 5, 1))

GET#3, ACT CODE

CUR REGS = N35$: QUAL REGS = N36$

ACT CUR. QUALS$(ACT CODE) = CUR REGS + QUAL REGS

MIDS$(ACT$(M MOD 20), 39, 20) = ACT CUR. QUALS$(ACT CODE)

BYTE = ACT CODE\8 + 1: BIT = (ACT CODE - 1) MOD 8

BYTE$ = MIDS$(ACT CODE. USEDS$, BYTE, 1)

IF (ASC(BYTE$) AND 2 BI$) <> 2 BI$ THEN MIDS$(ACT CODE. USEDS$, BYTE, 1) = CHRS(ASC(BYTE$) + 2 BI$)

LSET NIAS$ = ACT$(M MOD 20)

PUT#1, M

WEND

MAX N = MAX N + 20

WEND

MAX N = N

CLOSE #2, #3

ERASE ACT$.

PRINT"Transfer completed"

MAX N = MAX N + 20

OPEN "R", 3, WKDAT. FILES, 93

FIELD#3, I AS N31$, 42 AS N32$, 20 AS N33$, 2 AS N34$

OPEN "R", 2, "WK" + WK nums + "." + SCHED NUM$, 93
1510  FIELD#2, 93 AS N2A$
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(MIDS(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 pilot for qualification and save in pilotS...
1720  '  pilotS(act.code); next, determine which qualified pilots are also available, then save with the activity record in schednn.xxx
1730  FOR I = 1 TO 254
1740    BYTE = I\8 + 1: BIT = (I-1) MOD 8
1750    BYT$ = MIDS(ACT.CODE.USED$,BYTE,1)
1760    'set the pilot$ bits on only for qualified pilots...
1770    IF (ASC(BYTS) AND 2*BIT) = 2*BIT THEN GOSUB 2060: PILOT$(I) = PILOT$(I)
1780  NEXT
1790  
1800  'now have pilots qualified for each activity saved
1810  '  in pilot$(act.code); next, determine which qualified pilots are also available, then save with the activity record in schednn.xxx
1820  
1830  FOR N = 1 TO MAX.N - 1
1840    PRINT"Getting AVAIL$ data for sched sequence number" N
1850    GET#1, N
1860    ACT.CODE = ASC(N13$)
1870    IF ACT.CODE > 128 THEN PILOT$(ACT.CODE) ELSE PILOT$(ACT.CODE) = N18$
1880    ACT.ST TIME = CVI(N15$): ACT.END.TIME = CVI(N16$)
1890    FOR I = 1 TO MAX.PIL.NUM
1900      AVAIL$ = AVAIL$(I)
1910      BYT$ = MIDS(PILOT$(I)\8+1,1): BIT = (I-1) MOD 8
1920      IF (ASC(BYTS) AND 2*BIT) = 2*BIT THEN GOSUB 2230
1930      NEXT
1940    LSET N18$ = PILOT$(I)
1950    PUT#1, N
1960  NEXT
PRINT "Schedule file completed": END

'__subroutines________________________

'this routine compares the qual reqd values to the
pilot qual values and sets the pilot's bit
on if a qual match is found...
PILOTS = STRING$(8,0)
PRINT "Checking pilot qualifications for activity code" I
FOR P = 1 TO MAX.FIL.NUM
    J = 0: QUAL.FOUND = 0
    WHILE QUAL.FOUND = 0 AND J < 5
        J = J + 1
        QUAL.REQ = CVI(MID$(ACT.CUR.QUAL$(I),J*2+9,2))
        IF (QUAL(P) AND QUAL.REQ) THEN QUAL.FOUND = 1
    WEND
    BYTE = P\8+1: BYTS = MID$(PILOTS,BYTE,1): BIT = (P -1) MOD 8
    IF QUAL.FOUND THEN MID$(PILOTS,BYTE,1) = CHR$(ASC(BYTS) + 2*BIT): PRINT "Pilot" P "qual" ELSE PRINT "Pilot" P "not qual"
NEXT
FOR H = 1 TO 8: PRINT ASC(MID$(PILOTS,H,1));: NEXT:
PRINT
RETURN
'this routine checks a pilot for availability and
turns off the pilot's bit if not available...
START.TIME = ACT.ST.TIME
END.TIME = ACT.END.TIME
AVAIL = -1: SET = 0
GOSUB 2760
PRINT "Pilot" I;
IF NOT AVAIL THEN MID$(PILOTS,I\8+1,1) = CHR$(ASC(BYTS) + 2*BIT): PRINT "Not avail" ELSE PRINT "Avail"
RETURN
'trim trailing spaces...
L = LEN(TRIMS) + 1: L CHR$ = CHR$(0)
WHILE ASC(L CHR$) < 33
    L = L - 1
    L CHR$ = MID$(TRIMS,L,1)
WEND
TRIMS = LEFT$(TRIMS,L)
RETURN
'dynamic array size increase...
DIM TMP$(MAX.N)
FOR M = 1 TO MAX.N: TMP$(M) = ACT$(M) NEXT
ERASE ACT$: DIM ACT$(MAX.N + 10)
FOR M = 1 TO MAX N: ACT$(M) = TMP$(M) NEXT
MAX.N = MAX.N + 10
ERASE TMP$
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 other
2610 PERIOD.DUR as an integer length of period
2620 INCR as an integer for the value of each bit
2630 (resolution)
2640 START.TIME as values for the activity
2650 END.TIME
2660 AVAIL as a bit string with '1' available, '0' not avail
2670 AVAIL as a control code
2680 SET as a control code to set the time 'not a
2690 available'
2700 'returns:
2710 AVAIL as TRUE if time is available
2720 AVAIL updated if AVAIL and SET both TRUE
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
This routine selects the proper routine for checking or setting availability based on the case defined by CASE1, CASE2, and CASE3.

if start is before period...

if start and end are during period...

if start is during period but end is after...

if start is before and end is after period...

the final case ends before or starts after period...

if CASE1=3 OR CASE2=1 THEN PRINT "Activity is completely outside the period..."

--- This routine is used when END_BYTES is the first byte of AVAIL$. 

FIRST.BIT.Used = 0; LAST.BIT.Used = (END.BIT MOD 8)

BYT.TO.CK$ = LEFT$(AVAIL$,1)

GOSUB 4310

IF AVAIL AND SET THEN MID$(AVAIL$,1,1) = CHR$(ASC(BYT.TO.CK$) AND (NOT MASK))
--- This routine is used when END BYTE points to end byte...

FIRST.BYT = 1: LAST.BYT = END BYTE - 1

GOSUB 4190

FIRST.BIT.USED = 0: LAST.BIT.USED = (END.BIT MOD 8)

BYT.TO.CKS$ = MIDS(AVAIL$, END.BYTE, 1): J = END.BYTE

GOSUB 4210

IF AVAIL AND SET THEN GOSUB 4430 ELSE RETURN

MIDS(AVAIL$, END.BYTE, 1) = CHR$(ASC(BYT.TO.CKS$) AND (NOT MASK))

RETURN

--- This routine is used for the single byte case where one byte includes both start and end...

BYT.TO.CKS$ = MIDS(AVAIL$, START BYTE, 1)

MASK = 0

FIRST.BIT.USED = (START.BIT MOD 8)

LAST.BIT.USED = (END.BIT MOD 8)

GOSUB 4310

IF AVAIL AND SET THEN MIDS(AVAIL$, START BYTE, 1) = CHR$(ASC(BYT.TO.CKS$) AND (NOT MASK))

RETURN

--- This routine is used when one or more bytes separate the first and last bytes or when they are adjacent...

FIRST.BYT = START BYTE + 1: LAST.BYT = END BYTE - 1

GOSUB 4190

IF NOT AVAIL THEN RETURN

CASE3 = 2 enters here...

FIRST.BIT.USED = (START.BIT MOD 8): LAST.BIT.USED = 7

BYT.TO.CKS$ = MIDS(AVAIL$, START BYTE, 1)

GOSUB 4310

IF AVAIL AND SET THEN MASK.ST=MASK: BYT.ST$=BYT.TO.CKS$ ELSE IF NOT AVAIL THEN RETURN

FIRST.BIT.USED = 0: LAST.BIT.USED = (END.BIT MOD 8)

BYT.TO.CKS$ = MIDS(AVAIL$, END.BYTE, 1): J = END.BYTE

GOSUB 4310

IF NOT AVAIL THEN RETURN

IF (CASE3=3) AND (AVAIL AND SET) THEN GOSUB 4430

IF NOT (AVAIL AND SET) THEN RETURN

MIDS(AVAIL$, START BYTE, 1) = CHR$(ASC(BYT.ST$) AND (NOT MASK))

MIDS(AVAIL$, END.BYTE, 1) = CHR$(ASC(BYT.TO.CKS$) AND (NOT MASK))

RETURN
--- This routine is used when the last byte in the string is the only byte to be checked.

BYT.TO.CKS = MID$(AVAIL$, START.BYTE, 1)
FIRST.BIT.USED = (START.BIT MOD 8): LAST.BIT.USED = (PERIOD.END.BIT MOD 8)
GOSUB 4310
IF AVAIL AND SET THEN MID$(AVAIL$, START.BYTE, 1) = CHR$(ASC(BYT.TO.CKS)) AND (NOT MASK))
RETURN

--- This routine is used when the activity ends after the period and the first byte is one or more bytes from the end of AVAIL$. The last two cases of CASE3 are both checked by this routine.

FIRST.BIT.USED = (START.BIT MOD 8): LAST.BIT.USED = 7
BYT.TO.CKS = MID$(AVAIL$, START.BYTE, 1)
GOSUB 4310
FIRST.BYT = START.BYTE+1: LAST.BYT = LEN(AVAIL$)
GOSUB 4190
IF AVAIL AND SET THEN GOSUB 4430 ELSE RETURN
MID$(AVAIL$, START.BYTE, 1) = CHR$(ASC(BYT.TO.CKS)) AND (NOT MASK))
RETURN

--- This routine is used by the routines above when whole bytes are being checked for availability.

FOR J = FIRST.BYT TO LAST.BYT
BYT.TO.CKS = MID$(AVAIL$, J, 1)
IF BYT.TO.CKS <> CHR$(255) THEN FIRST.BIT.Used = 0: LAST.BIT.Used = 7: GOSUB 4310
IF NOT AVAIL THEN RETURN
NEXT
RETURN

--- This routine is called by above routines to check availability within partial bytes of AVAIL$.

MASK = 0
FOR K = FIRST.BIT.Used TO LAST.BIT.Used
MASK = MASK + 2^K
IF (ASC(BYT.TO.CKS)) AND 2^K = 0 THEN AVAIL = 0
NEXT
--- This routine is called when a whole byte is to be set to NOT AVAILABLE state, both AVAIL and SET are TRUE.

FOR J = FIRST.BYT TO LAST.BYT
MID$(AVAIL$,J,1) = CHR$(0)
NEXT
RETURN
103

'*** WKSCHED.SET ****************************
110 'program dated 24 May 1983
120 'This program builds the final schedule file from
130 ' schednn.xxx and wknn.xxx
140 'variables required:
150 '     none
160 'returns:
200 '     SCHEDnn.DAT file when completed
210 'DEFINT A-Z
220 CLR$ = CHR$(26): DOWNS = CHR$(10): ESC$ = CHR$(27):
230 CHR$ = CHR$(13):
240 MID.SCRN$ = CLR$ + STRINGS$(6,10):
250 UPS = CHR$(11): MOV.LEFT$ = CHR$(8): MOV.RIGHT$ = CHR$(12):
260 HOME$ = CHR$(30): CLR.LINE$ = ESC$ + "T"
270 PERIOD.ST.TIME = 0: PERIOD.DUR = 10080: INCR = 30:
280 MAX.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$ = STRINGS$(16,0)
330 DAYS$ = "SunMonTueWedThuFriSat"
340 MONTHS$ = "JANFEBMARAPRMAYJUNJULAUGSEPTENOVDECJAN"
350 FIRST DAYS$ = "0010320609112115218221324427430533536
370 CS = "Enter: 0 if correct, 1 to change it:"
380 ES = "Error, enter 0 or 1 only, try again..."
390 PRINT MID.SCRN$ "Enter the week number:";: INPUT" ",
400 WKNUM$ = "SCHED" + WKNUM$ + ",."
410 PRINT"Schedule files for week " WKNUM$": PRINT FILES FILENAMES$: PRINT DOWNS
420 PRINT"Enter the schedule file number or 0 to start a
430 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 FILENAMES$ = "SCHED" + WKNUM$ + ",." + SCHED.NUM$
500 ' 'PRINT MID.SCRN$ "Using " FILENAMES$ " for schedule da-
510 "" 'TA."'PRINT CS$; AS = INPUT$(1): PRINT AS
530 IF AS = "1" THEN GOTO 400 ELSE IF AS <> "0" THEN PRI-
540 NT ES$; GOTO 520
schedule_data

OPEN "R", 1, FILENAME$, 58

seq. num. act sched. time: act.code pil num:

st. time: end. time: act.name: pilots: cur. req. qu
al. req:

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$, 10
AS N110$

FIELD#1, 58 AS N1A$

PRINT MID.SCRNS "Getting date data...

WK.DAT. FILES = "WK" + WK.NUM$ + ".") + SCHED.NUM$

OPEN "R", 2, WK.DAT. FILES$, 93

FIELD#2, 1 AS N21$, 2 AS N22$, 9 AS N23$, 81 AS N24$

GET#2, MAX.PIL.NUM + 1

IF ASC(N21$) () MAX.PIL.NUM + 1 THEN PRINT "File access error in " WK.DAT. FILES$

WK.DATE = CVI(N22$): WK.DATE$ = N23$

IF (WK.DATE MOD 1000) \7 (') VAL(WK.NUM$) THEN PRINT "Error: week number " WK.NUM$ " does not agree with file " WK.DAT. FILES$",

dated " WK.DATE$ 

CLOSE #2

PRINT MID.SCRNS "Getting currency names from file...

OPEN "R", 2, "CUR.DEF", 28

cur.num: cur.nam$: exp.dur: inxp.dur:

event.num:

FIELD#2, 2 AS N21$, 20 AS N22$, 2 AS N23$, 2 AS N24$

, 2 AS N25$

FOR I = 1 TO 15

GET#2, I

TRIM$ = N22$: GOSUB 3510: CUR.NAM$(I) = TRIM$

EXP.DUR(I) = CVI(N23$)

INXP.DUR(I) = CVI(N24$)

EVENT.NUM(I) = CVI(N25$)

NEXT

CLOSE #2

PRINT MID.SCRNS "Getting pilot names from file...

OPEN "R", 2, "PILNAM.DEF", 27

pil.num: 1.name$: init$: rank$

FIELD#2, 2 AS N21$, 20 AS N22$, 2 AS N23$, 3 AS N24$

FOR I = 1 TO MAX.PIL.NUM

GET#2, I

TRIM$ = N22$

IF TRIM$ = "Not in use" THEN GOTO

ELSE GOSUB 3510: PIL.NAM$(I) = TRIM$

PIL.NAM$(I) = N24$ + " + PIL.NAM$(I) + ",

PRINT CRS STRINGS(33,32) CRS USING ":", I:
PRINT " - " PIL.NAMS(I);
950 NEXT
960 PIL.NAMS(MAX.PIL.NUM+1) = "None"
970 CLOSE #2
980 .
990 N = 0: ACT.SCHED.TIME = 0
1000 PRINT MID.SCRNS "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) = NIA$
1110 NEXT
1120 .
1130 PRINT MID.SCRNS FILENAME$ " data now in memory..."
1140 .
1150 'open data file...
1160 OPEN "R", 2, WKDAT.FILES, 93
1170 FIELDS2, 1 AS N221$, 42 AS N222$, 20 AS N234$, 28 AS N225$
1180 .
1190 PRINT"Enter:"
1200 PRINT"0 to quit"
1210 PRINT"1 to fill schedule in sequence number order"
1220 PRINT"2 to fill individual sequence number activities"
1230 PRINT"3 to fill by activity number"
1240 PRINT"Which choice?";
1250 SEL = VAL(INPUTS(1))
1260 IF SEL = 0 THEN GOTO 1310 ELSE IF SEL > 3 THEN PRINT
1270 "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$
1360 "and " WKDAT.FILES
1370 '___control_subroutines______________
1380 .
1390 'step thru in sequence number order...
1400 PRINT"Not written yet..."; DUMMY$ = INPUTS(1)
1410 RETURN
1420 'one seq num at a time from keyboard...
1430 PRINT MID.SCRN$: "What is the sequence number?";
106

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 another schedule activity:" : GOTO 1440
1480  RETURN
1490  'by activity number...
1500  PRINT"Not written yet...": DUMMY$ = INPUT$(1)
1510  RETURN
1520  '___subroutines_________________
1530  'display an activity and candidates on screen...
1540  IF SEQ.NUM () CVI(MIDS(ACT$(SEQ.NUM),1,2)) THEN PRIN
T"Error in file at record " SEQ.NUM
1550  ACT.SCHED.TIME = CVI(MIDS(ACT$(SEQ.NUM),3,2))
1560  'get clock time, day, and date...
1570  GOSUB 3730: SCHED.TIMES = THIS.TIMES
1580  GOSUB 3880
1590  ACT.CODE = ASC(MIDS(ACT$(SEQ.NUM),5,1))
1600  PIL.NUM = ASC(MIDS(ACT$(SEQ.NUM),6,1))
1610  START.TIME = CVI(MIDS(ACT$(SEQ.NUM),7,2))
1620  GOSUB 3730: ST.TIIME$ = THIS.TIMES
1630  END.TIME = CVI(MIDS(ACT$(SEQ.NUM),9,2))
1640  GOSUB 3730: ENO.TIMES = THIS.TIMES
1650  IF PIL.NUM = 255 THEN PIL.NUM = MAX.PIL.NUM + 1
1660  ACT.NAME$ = MIDS(ACT$(SEQ.NUM),11,20)
1670  CAND$ = MIDS(ACT$(SEQ.NUM),31,8)
1680  CAND.TOT = 0
1690  FOR I on 1 TO MAX.PIL.NUM
1700      BYTE = ASC(MIDS(CAND$,I+1,1)): BIT = (I-1) MOD 8
1710      IF (BYTE AND 2'BIT) = 2'BIT THEN CAND.TOT =
1720      CAND.TOT + 1
1730  NEXT
1740  'CUR.REQ = CVI(MIDS(ACT$(SEQ.NUM),39,2))
1750
1760  'PRINT LEFT$(MID. SCRNS$, 5);
1770  PRINT"Sequence number:" SEQ.NUM TAB(50) THIS DAYS ",
" THIS.DATE$ DOWN$;
1780  PRINT SCHED.TIMES " " ACT.NAME$ TAB(35) "Assigned: 
" PIL.NAMS(PIL.NUM) DOWN$;
1800  NONE = -1
1810  PRINT"Currencies required:";
1820  FOR I = 0 TO 14
1830      IF (CUR.REQ AND 2'1) = 2'I THEN PRINT CUR.NAMS(I +1) " "; IF NONE THEN NONE = 0
1840  NEXT
1850  IF NONE THEN PRINT"None"
1860  PRINT
1870  PRINT"Candidate names "
1880 IF CAND.TOT = 0 THEN PRINT "None shown as both qualified and available"; GOTO 2170
1890 K = 0: L = CAND.TOT \ 3
1900 IF CAND.TOT MOD 3 > 0 THEN L1 = L + 1 ELSE L1 = L
1910 IF CAND.TOT MOD 3 = 1 THEN L2 = L + 1 ELSE L2 = L
1920 BYTE = 0: BIT = 0: LN = 0
1930 WHILE LN < L1
1940 LN = LN + 1
1950 I = 0
1960 'find first column name to print...
1970 FOR J = 1 TO LN
1980 'GOSUB 3290
1990 NEXT
2000 'print it...
2010 GOSUB 3370: IF K = CAND.TOT THEN GOTO 2140
2020 'skip 11 names
2030 FOR J = 1 TO L1
2040 'GOSUB 3290
2050 NEXT
2060 'print the next one
2070 GOSUB 3370: IF K = CAND.TOT THEN GOTO 2140
2080 'skip 12 names
2090 FOR J = 1 TO L2
2100 'GOSUB 3290
2110 NEXT
2120 GO SUB 3370
2130 WEND
2140 'screen now shows activity and available pilots...
2150 PRINT "Enter 0 to skip selection or pilot number to select a pilot for this activity"
2160 IF I <= 0 THEN GOTO 3060 ELSE IF I > MAX.PIL.NUM THEN PRINT "Out of range, enter a number from 0 to " MAX.PIL.NUM "only, try again"; GOTO 2180
2170 'check avail and crew rest if applicable...
2180 GET#2, I
2190 IF I <> ASC(N221$) THEN PRINT "File access error in " WKDAT.FILES
2200 AVAIL$ = N222$
2210 NET.NLT$ = N223$
2220 FOR N = 1 TO 7
2230 NET(N) = CVI(MID$(NET.NLT$, N*4 - 3, 2))
2240 NLT(N) = CVI(MID$(NET.NLT$, N*4 - 1, 2))
2250 NEXT
2260 AVAIL = -1: SET = 0
2270 IF NOT AVAIL THEN PRINT "is not good, resetting status...": MID$(ACT$(SEQ.NUM), I \ 8 + 31, 1) = CHR$(ASC(MID$(CAND$, I \ 8 + 1, 1)) - 2 + ((I - 1) MOD 8)): GOTO 1560
2280 IF AVAIL AND (ACT.CODE AND 192) THEN GOSUB 3110 ELSE
C.REST = -1
2340 IF C.REST AND AVAIL THEN PRINT" is good" ELSE PRINT" is not good"
2350 IF AVAIL AND (NOT C.REST) THEN PRINT"Crew rest rules not met, enter 0 to ignore crew rest or 1 to NOT select this pilot ;"
A$ = INPUT$(1): PRINT A$ ELSE GOTO 2380
2360 IF A$ = "1" THEN GOTO 1770 ELSE IF A$ <> "0" THEN PRINT "is not good"
2370 IF AVAIL AND (NOT C.REST) THEN PRINT "Crew rest rules not met, enter 0 to ignore crew rest or I to NOT select 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$(BYTE - 2'BIT)
2410 LSET N2254 = AVAIL$
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 N2254 = 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 GOTO 2570
2540 IF START.TIME > CVI(MID$(ACT$(J),9,2)) THEN GOTO 2570
2550 NET(N) = CVI(MIDS(NET.NLT$,N*4 - 3,2))
2560 NLT(N) = CVI(MIDS(NET.NLT$,N*4 - 1,2))
2570 NEXT
2580 IF PIL.NUM = MAX.PIL.NUM + 1 THEN GOTO 1560
2590 'if pil.num () 61 then reset old pilot's bit and avail
18(pil.num)... 
2600 GET*2, PIL.NUM
2610 AVAIL$ = N2254
2620 NET.NLT$ = N2254
2630 FOR N = 1 TO 7
2640 NET(N) = CVI(MIDS(NET.NLT$,N*4 - 3,2))
2650 NLT(N) = CVI(MIDS(NET.NLT$,N*4 - 1,2))
2660 NEXT
2670 BYTE = ASC(MID$(CAND$,PIL.NUM\8+1,1)): BIT = (PIL NUM M-1) MOD 8
2680 IF (BYTE AND 2'BIT) = 0 THEN MID$(CAND$,PIL NUM\8+1,1) = CHR$(BYTE + 2'BIT)
2690 FOR I = START.BIT TO END.BIT
2700 BYTE = I\8+1: BIT = (I-1) MOD 8
2710 BYTES = MID$(AVAIL$ BYTE,1)
2720 IF (ASC(BYTES) AND 2'BIT) = 0 THEN MID$(AVAIL
$\text{BYTE} \cdot 1 = \text{CHR} \cdot (\text{ASC} \cdot \text{BYTE} + 2 \cdot \text{BIT})$

2730 NEXT

2740 DAY = START.TIME\1440 \cdot 1

2750 IF DAY \cdot 2 \ OR \ DAY \cdot 6 THEN GOTO 2960

2760 IF NLT(DAY-1) \cdot \text{START.TIME} - 720 THEN GOTO 2860

2770 ACT.SCHED.TIME = NLT(DAY-1) \cdot \text{GOSUB} 3730

2780 PRINT "Cancelled activity set crew rest time for ending previous day: " \ THIS TIMES$

2790 PRINT C$\cdot A$ = INPUT$(1)$; PRINT A$

2800 IF A$ = "1" THEN GOSUB 3410 \cdot \text{NLT(DAY-1) = NEW.T + (DAY - 2) \cdot 1440 ELSE IF A$ <> "0" THEN PRINT E$; GOTO 2790

2810 IF NLT(DAY) \cdot \text{START.TIME} + 720 THEN GOTO 2860

2820 ACT.SCHED.TIME = NLT(DAY) \cdot \text{GOSUB} 3730

2830 PRINT "Cancelled activity set crew rest time for ending this day: " \ THIS TIMES$

2840 PRINT C$\cdot A$ = INPUT$(1)$; PRINT A$

2850 IF A$ = "1" THEN GOSUB 3410 \cdot \text{NLT(DAY) = NEW.T + (DAY - 1) \cdot 1440 ELSE IF A$ <> "0" THEN PRINT E$; GOTO 2840

2860 IF NET(DAY) \cdot \text{END.TIME} - 720 THEN GOTO 2910

2870 ACT.SCHED.TIME = NET(DAY) \cdot \text{GOSUB} 3730

2880 PRINT "Cancelled activity set crew rest time for beginning this day: " \ THIS TIMES$

2890 PRINT C$\cdot A$ = INPUT$(1)$; PRINT A$

2900 IF A$ = "1" THEN GOSUB 3410 \cdot \text{NET(DAY) = NEW.T + (DAY - 1) \cdot 1440 ELSE IF A$ <> "0" THEN PRINT E$; GOTO 2900

2910 IF NET(DAY+1) \cdot \text{END.TIME} + 720 THEN GOTO 2960

2920 ACT.SCHED.TIME = NET(DAY+1) \cdot \text{GOSUB} 3730

2930 PRINT "Cancelled activity set crew rest time for beginning following day: " \ THIS TIMES$

2940 PRINT C$\cdot A$ = INPUT$(1)$; PRINT A$

2950 IF A$ = "1" THEN GOSUB 3410 \cdot \text{NET(DAY+1) = NEW.T + DAY \cdot 1440 ELSE IF A$ <> "0" THEN PRINT E$; GOTO 2940

2960 PRINT PIL.NAMS(PIL.NUM) " is reset in " \ WKDAT FILES

2970 LSET N2228 = AVAIL$

2980 FOR N = 1 TO 7

2990 IF NET(N) \cdot (N-1 THEN MID$(\text{NET.NLTS},N*4-3,2) = MKI$(\text{NET}(N))

3000 IF NLT(N) \cdot (N-1 THEN MID$(\text{NET.NLTS},N*4-1,2) = MKI$(\text{NET}(N))

3010 NEXT

3020 LSET N2258 = NET.NLTS$

3030 PUT$2, PIL.NUM

3040 MID$(\text{ACTS(SEQ.NUM)},31,8) = CAND$

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 \cdot 1

3150 IF DAY \cdot 2 OR DAY \cdot 6 THEN C REST = -1 RETURN
110

3160  IF START.TIME >= NET(DAY) OR NET(DAY) = -1 THEN ST.CK = -1 ELSE ST.CK = 0
3170  IF END.TIME <= NLT(DAY) OR NLT(DAY) = -1 THEN END.CK = -1 ELSE END.CK = 0
3180  IF ST.CK AND END.CK THEN C.REST = -1 ELSE C.REST = 0 : RETURN
3190  IF (START.TIME - 720 < NLT(DAY - 1)) OR (NLT(DAY - 1) = -1) THEN NLT(DAY - 1) = START.TIME - 720
3200  IF (NLT(DAY) > START.TIME + 720) OR (NLT(DAY) = -1) THEN NET(DAY) = START.TIME + 720
3210  IF (NET(DAY) < END.TIME - 720) OR (NET(DAY) = -1) THEN NET(DAY) = END.TIME - 720
3220  IF (NET(DAY + 1) > END.TIME + 720) OR (NET(DAY + 1) = -1) THEN NET(DAY + 1) = END.TIME + 720
3230  RETURN
3240  IF I < MAX.PIL.NUM THEN I = I + 1 ELSE GOTO 3350
3250  BYTE = ASC(MID$(CAND$, I*8 + 1, 1)) : BIT = (I-1) MOD 8
3260  WHILE ((BYTE AND 2^BIT) = 0) AND (I < MAX.PIL.NUM)
3270     I = I + 1
3280  BYTE = ASC(MID$(CAND$, I*8 + 1, 1)) : BIT = (I-1) MOD 8
3290  WEND
3290  RETURN
3300  PRINT TAB((K*25 + 1) MOD 75) USING "##": I-1: PRINT "PIL.NAM$(I): K = K + 1: IF K MOD 3 = 0 THEN PRINT
3310  RETURN
3320  PRINT "Enter the new crew rest time: ":
3330  INPUT "NEW.T = HR*60 + MIN
3340  RETURN
3350  TRIM = TRIM$(TRIM$, L)
3360  TRIM = LEFT$(TRIM$, L)
3370  RETURN
3380  ' dynamic array size increase...
111

3600  DIM TMP$(MAX.N)
3610  FOR M = 1 TO MAX.N: TMPS(M) = ACTS(M): NEXT
3620  ERASE ACTS: DIM ACTS(MAX.N + 10)
3630  FOR M = 1 TO MAX.N: ACTS(M) = TMPS(M): NEXT
3640  MAX.N = MAX.N + 10
3650  ERASE TMP$
3660  ON ERROR GOTO 0
3670  RETURN
3680  'this routine computes the time from a time in
3690  ' minutes of a week...
3700  HR = (ACT.SCHED.TIME MOD 1440) \ 60
3710  MIN = (ACT.SCHED.TIME MOD 1440) MOD 60
3720  TS = MIDS(STRS(HR),2): GOSUB 3790: HR$ = TS
3730  TS = MIDS(STRS(MIN),2): GOSUB 3790: MIN$ = TS
3740  THIS.TIMES$ = HR$ + MIN$
3750  RETURN
3760  WHILE LEN(TS$) < 2
3770    TS$ = "0" + TS$
3780  WEND
3790  RETURN
3800  'this routine determines the day and date of an
3810  ' activity from wk.date, wk.date$, and the
3820  ' activity schedule time...
3830  DAY = ACT.SCHED.TIME \ 1440: THIS.DAYS$ = MIDS(DAYS,DAY * 3 + 1,3)
3840  THIS.DATE.J = WK.DATE + DAY: DAY.J = THIS.DATE.J MOD 1000
3850  YEAR = VAL(RIGHTS(WK.DATE$,2)): IF YEAR/4 = YEAR\4 THEN
3860    L.YR = 1 ELSE L.YR = 0
3870  MO = 0: NEXT.MO.1ST.DAY = 1
3880  WHILE (DAY.J > NEXT.MO.1ST.DAY) AND (MO < 12)
3890    'save new 'this month', get next month...
3900    THIS.MO.1ST.DAY = NEXT.MO.1ST.DAY
3910    MO = MO + 1
3920    NEXT.MO.1ST.DAY = VAL(MIDS(FIRST.DAYS,M0*3 + 1,3))
3930  IF (MO = 2) AND L.YR THEN NEXT.MO.1ST.DAY = NEXT.MO.1ST.DAY + 1
3940  WEND
3950  'when the day falls in the following year, loop is
3960  ' terminated by mo = 12, thus...
3970  IF DAY.J > NEXT.MO.1ST.DAY THEN YEAR = YEAR + 1. THIS.DATE = DAY.J - NEXT.MO.1ST.DAY + 1 ELSE THIS.DATE = DAY.J - THIS.MO.1
3980  ST.DAY + 1
3990  THIS.DATE$ = MIDS(STRS(TTHIS.DATE),2) + "" + MIDS(MO NTHS,(MO-1)*3 + 1,3) + STRS(YEAR)
4000  RETURN
4010  'this routine determines...
*** CASE ***

module dated 24 April 1983

' This module includes subroutines called by other modules in determining the case of each activity relative to the week

' variables required:

PERIOD.ST.TIME as an integer in minutes or other time units

PERIOD.DUR as an integer length of period

INCR as an integer for the value of each bit (resolution)

START.TIME as values for the activity

END.TIME

AVAILs as a bit string with '1' available, '0' not available

AVAIL as a control code

SET as a control code to set the time 'not available'

'returns:

AVAIL as TRUE if time is available

AVAILs updated if AVAIL and SET both TRUE

'subroutines used:

all internal

GOSUB 4520

GOSUB 4730

RETURN

--- This routine determines the case of activity start (CASE1) and end (CASE2) relative to the period start and end...

CASE1 and CASE2 equal 1 if times are before the period starts, 2 if during the period, or 3 if after the period. Thus if CASE1 is 3 or CASE2 is 1, the whole activity falls outside the period in question. If both CASE1 and CASE2 are 2, then the whole activity is within the period.

CASE3 has a value of 1 if the whole activity falls on a single byte, 2 if on adjacent bytes, and 3 if one or more whole bytes fall between the start and end.

START.BIT = START.TIME\INCR

START.BYTE = START.BIT\8 + 1

END.BIT = (END.TIME-1)\INCR

END.BYTE = .END.BIT\8 + 1

IF START.TIME >= PERIOD.ST.TIME THEN COND1 = -1 ELSE
CONDI = 0
4570 IF START.TIME < (PERIOD.ST.TIME + PERIOD.DUR) THEN COND2 = -1 ELSE COND2 = 0
4580 IF CONDI AND COND2 THEN CASE1 = 2 ELSE IF NOT CONDI THEN CASE1 = 1 ELSE IF NOT COND2 THEN CASE1 = 3
4590 IF END.TIME > PERIOD.ST.TIME THEN COND3 = -1 ELSE COND3 = 0
4600 IF END.TIME <= (PERIOD.ST.TIME + PERIOD.DUR) THEN COND4 = -1 ELSE COND4 = 0
4610 IF COND3 AND COND4 THEN CASE2 = 2 ELSE IF NOT COND3 AND COND4 THEN CASE2 = 1 ELSE IF NOT COND3 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 THEN CASE3 = 3
4650 RETURN
4660
4670
4680 This routine selects the proper routine for checking or setting availability based on the case defined by CASE1, CASE2, and CASE3
4690
4700 if start is before period...
4710
4720 IF CASE1 = 1 AND CASE2 = 2 THEN ON CASE3 GOSUB 4880, 4980
4730 IF CASE1 = 2 AND CASE2 = 2 THEN ON CASE3 GOSUB 5110, 5280
4740 if start and end are during period...
4750 IF CASE1 = 2 AND CASE2 = 3 THEN ON CASE3 GOSUB 5460, 5590
4760 if start is during period but end is after...
4770 IF CASE1 = 2 AND CASE2 = 3 THEN ON CASE3 GOSUB 5460, 5590
4780 if start is before and end is after period...
4790 IF CASE1 = 1 AND CASE2 = 3 THEN FIRST.BYTE = 1: LAST.BYTE = LENGTH(AVAIL$) + 1: GOSUB 3630
4800 the final case ends before or starts after period...
4810 IF CASE1 = 3 OR CASE2 = 1 THEN PRINT "Activity is completely outside the period..."
4820 RETURN
4830
4840
4850 --- This routine is used when END.BYTE is the first byte of AVAIL$...
4860
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...
114

1140  " end byte ...

11410 4960  FIRST.BY = 1: LAST.BY = END.BYTE-1
11420 4970  GOSUB 5730
11430 4980  FIRST.BIT.Used = 0: LAST.BIT.Used = (END.BIT MOD 8)
11440 4990  BYT.TO.CKS = MIDS(AVAILS,END.BYTE,1): J = END.BYTE
11500 5000  GOSUB 5850
11510 5010  IF AVAIL AND SET THEN GOSUB 5970 ELSE RETURN
11520 5020  MIDS(AVAILS,END.BYTE,1) = CHR$(ASC(BYT.TO.CKS) AND NOT MASK)
11530 5030  RETURN

11540 5040  "--- This routine is used for the single byte case
11550 5050  " where one byte includes both start and end...
11560 5060 5070  5080 5090 5100
11570 5110  BYT.TO.CKS = MIDS(AVAILS,START.BYTE,1)
11580 5120  MASK = 0
11590 5130  FIRST.BIT.Used = (START.BIT MOD 8)
11600 5140  LAST.BIT.Used = (END.BIT MOD 8)
11610 5150  GOSUB 5850
11620 5160  IF AVAIL AND SET THEN MIDS(AVAILS,START.BYTE,1) = CHRS(ASC(BYT.TO.CKS) AND NOT MASK)
11630 5170  RETURN

11640 5180  "--- This routine is used when one or more bytes
11650 5190  " separate the first and last bytes or when
11660 5200  " they are adjacent...
11670 5210 5220 5230
11680 5240  FIRST.BY = START.BYTE+1: LAST.BY = END.BYTE-1
11690 5250  GOSUB 5730
11700 5260  IF NOT AVAIL THEN RETURN
11710  " CASE3 = 2 enters here...
11720 5270 5280 5290
11730 5240  FIRST.BIT.Used = (START.BIT MOD 8): LAST.BIT.Used = 7
11740 5300  BYT.TO.CKS = MIDS(AVAILS,START.BYTE,1)
11750 5310  GOSUB 5850
11760 5320  IF AVAIL AND SET THEN MASK.ST=MASK: BYT.STS=BYT.TO.CKS ELSE IF NOT AVAIL THEN RETURN
11770 5330 5340 5350
11780 5320  FIRST.BIT.Used = 0: LAST.BIT.Used = (END.BIT MOD 8)
11790 5330  BYT.TO.CKS = MIDS(AVAILS,END.BYTE,1): J = END.BYTE
11800 5340  GOSUB 5850
11810 5350  IF NOT AVAIL THEN RETURN
11820 5360 5370 5380
11830 5360  IF (CASE3=3) AND (AVAIL AND SET) THEN GOSUB 5970
11840 5370  IF NOT(AVAIL AND SET) THEN RETURN
11850 5380  MIDS(AVAILS,START.BYTE,1) = CHR$(ASC(BYT.STS) AND NOT MASK)
11860 5390  MIDS(AVAILS,END.BYTE,1) = CHR$(ASC(BYT.TO.CKS) AND NOT MASK)
11870 5400  RETURN

11880 5410 5420
11890 5430 5440
11900  "--- This routine is used when the last byte in the
11910  " string is the only byte to be checked...
This routine is used when the activity ends after the period and the first byte is one or more bytes from the end of AVAILS. The last two cases of CASE3 are both checked by this routine...

This routine is used by the routines above when whole bytes are being checked for availability...

This routine is called by above routines to check availability within partial bytes of AVAILS...

MASK = 0

FOR K = FIRST.BIT.USED TO LAST.BIT.USED
  MASK = MASK + 2^K
  IF (ASC(BYT.TO.CK$) AND 2^K) = 0 THEN AVAIL = 0
NEXT
RETURN

This routine is called when a whole byte is to
be set to NOT AVAILABLE state, both AVAIL and SET are TRUE.

FOR J = FIRST.BYT TO LAST.BYT
MID$(AVAIL$, J, 1) = CHR$(0)$
NEXT
RETURN
BIBLIOGRAPHY


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


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