Fifteen Programmers, 400 Computers, 36,000 Sensors, and FORTH

by

Elizabeth D. Rather
FORTH, Inc.

ABSTRACT

A large, recently completed Mid-East Airport featured in its design a powerful, integrated computer system incorporating security and access control, fire and life safety monitoring, environmental monitoring and control, power distribution, fuel farm monitoring, water distribution, control of runway lights, and numerous related applications. The system is being supplied by AVCO Corp., and is programmed in Forth by a team of programmers from FORTH, Inc. and AVCO. This paper discusses some technical and management highlights from this experience.

Project Overview

The Airport

The Airport in question opened for traffic in December, 1983. In addition to serving the capital city, it is the hub of the national air transportation system. The airport is capable of serving 20 million passengers per year. Demand is expected to approach 15 million by the year 2000; the local population by that year is expected to exceed 2 million.

Although the traffic volume is still not large, in its design and architecture the airport is a magnificent showplace. It occupies an area roughly 12 by 20 kilometers in size, including not only the standard airport facilities (terminals, runways, hangars, etc.) but also a village for 3,000 airport employees and their families (including apartments, schools, recreational facilities, shops, mosques, etc.). There are about 90 buildings, including a Royal Terminal for the King and his guests and a public mosque in the center of the airport with a capacity of 5,000 worshipers.

The airport was constructed over a period of five years. The Construction Manager was Arabian Bechtel Corp., Ltd. (ABCL). Over 150 contractors participated in the work. Among these was AVCO Corp., who contracted to provide an integrated computer system to perform various site management functions throughout the airport. The primary functions included:

1. **Fire and Life Safety Monitoring.** The FLS subsystem monitors fire alarms, smoke detectors, noxious gas detectors, and similar devices throughout the airport, reporting detections to six operator stations in various locations. A critical timing requirement in the system is that a fire alarm must be presented to the responsible operator within three seconds after detection.

2. **Heating, Ventilation and Air Conditioning.** The HVAC subsystem is initially required to perform primarily a monitoring function, with control functions taking place only upon request of the operator, although some advisory capacity exists. Closed loop control is expected to be added as a future enhancement. This subsystem monitors thermocouples and other sensors, and enables one of four HVAC operators to control Air Conditioners, etc., from remote stations. In addition, reports can show trends of groups of points as well as other summary data over various periods of time.

3. **Security.** The Security subsystem includes access control for all airport facilities. It incorporates badge readers and intrusion detectors in many areas, as well as control of door and gate locks. It also includes special fire sensors so that doors in a fire area may be opened.

4. **Power Distribution.** Power utilization is monitored throughout the airport, and a large number of special reports are available showing trends of power usage. In addition, operator control of generators is handled through the system.

5. **Runway Lighting.** A special subsystem controls the runway lights for the airport.

6. **Water Treatment.** Water is of special concern in a desert country. The airport requires up to 3.1 million gallons per day for drinking, fire protection and other uses. The airport has five local water wells, each approximately 1,700 meters (about one mile) deep, and its own water treatment plant. Water usage is monitored throughout the airport, and pumps and other equipment are monitored through this subsystem.

7. **Fuel Farm.** The main jet fuel distribution network consists of 11,000 meters of 24-inch steel pipe and 5,000 meters of 16-inch steel pipe with 100 hydrants. Main jet fuel storage facilities include six

storage tanks with a total capacity equal to a seven day fuel supply. A special subsystem monitors the tanks of avaiation fuel and other fuel products stored in the fuel farm.

8. **Central Plant.** The Central Plant houses the generators and other heavy equipment. Over six kilometers of utility tunnels connect the Central Plant to the twenty-one buildings that house the airport's major activities. In addition to controlling the major equipment in the Central Plant, the computer system maintains records of scheduled preventive maintenance operations performed on it as well as other items of equipment in various locations.

Computer Hardware Configuration

Control of the major subsystems outlined above is provided through an integrated computer system whose major components include:

8 DEC PDP-11/44 computers;
378 custom 8086-based computers, each with dual cpu's, main memory and power supply;
320 custom 8085-based "Local Security Panels."

The custom computers were designed and built by AVCO Electronics Division (AED), Huntsville, Alabama. AED also wrote the software for the LSP's (in assembly language) and developed prototype software for the other computers. The prototype software was developed using FORTRAN and MACRO-11 on the PDP-11's, running under RSX-11M; for the 8086's RMX-86 and assembly language was used.

The PDP-11's are configured with 4 megabytes of main memory. Four are configured with three 80-Mb disk drives (two fixed, one removable), and the others have dual RL02 drives. All have 8-port DZ-11 multiplexors; these are attached to VT-100 terminals during development. Some ports have IDT color graphics terminals for operator stations. The PDP-11's communicate with each other using DMR-11 and DMC-11 high speed serial lines (1 M-bit/sec).

In the original design of the system there were five additional PDP-11/34 computers used as communications concentrators. When the software was

changed to polyFORTH these were replaced by special LSI-11 based communications processors called ICPs. Using the ICPs the PDP-11s communicate with the 8086-based computers using SDLC lines running at 38,400 baud.

The 8086-based computers come in several configurations, all of which are minor variations on a central theme: dual processors on a bus similar to Multibus with 128K to 1Mb memory, some portion of which can be ROM. The two processors operate in parallel, and in case of failure of one the other can take over. They have independent back-planes, power-supplies and SDLC lines, but share a set of I/O cards of various types (analog input and output, digital input and output, and serial lines to LSP's and other devices) with which they perform the monitoring functions for which they are designed. Some of these (called RCDRs) are configured with 10 Mb Winchester disks and terminals for operator stations; the others lack disks, and can be lumped together for this paper under the term RMT (Remote Multiplex Terminal).

The entire community of computers is divided into two main groups: the Digital Data System (DDS) includes primary monitoring and control for the FLSS, HVAC, PSS, RSS and Security subsystems. The Process Monitoring System (PMS) handles the Central Plant, Fuel Farm and Water Treatment subsystems. The overall organization of these is portrayed in ***Figure 1***. The Central Process Computer (CPC), which is the cornerstone of the PMS, communicates with both the Building Automation Computers (BAC1 and BAC2), so that the data base in the latter contains complete information about the state of the airport.

Within the DDS, the top of the hierarchy is occupied by the twin BAC computers, who maintain the primary data base and support most of the operator terminals. A similar pair of PDP-11's called Security Control Computers (SCC1 and SCC2) handle the Security subsystem and communicate with the RMT's through twelve multi-drop SDLC links.

The entire system is responsible for monitoring up to 36,000 sensor points (of which about 17,000 are presently defined). Digital inputs and contact inputs generate interrupts when they change state; other kinds of inputs are sampled about once per second. The sampling takes place in the RMT computers; changes of state are reported to the PDP-11 responsible for each RMT, which then routes the event to one or more operator stations designated as having interest in a changed point.

There are nineteen operator stations and four "Programmer" stations. Each is configured with two color graphics screens and a keyboard. The screens have

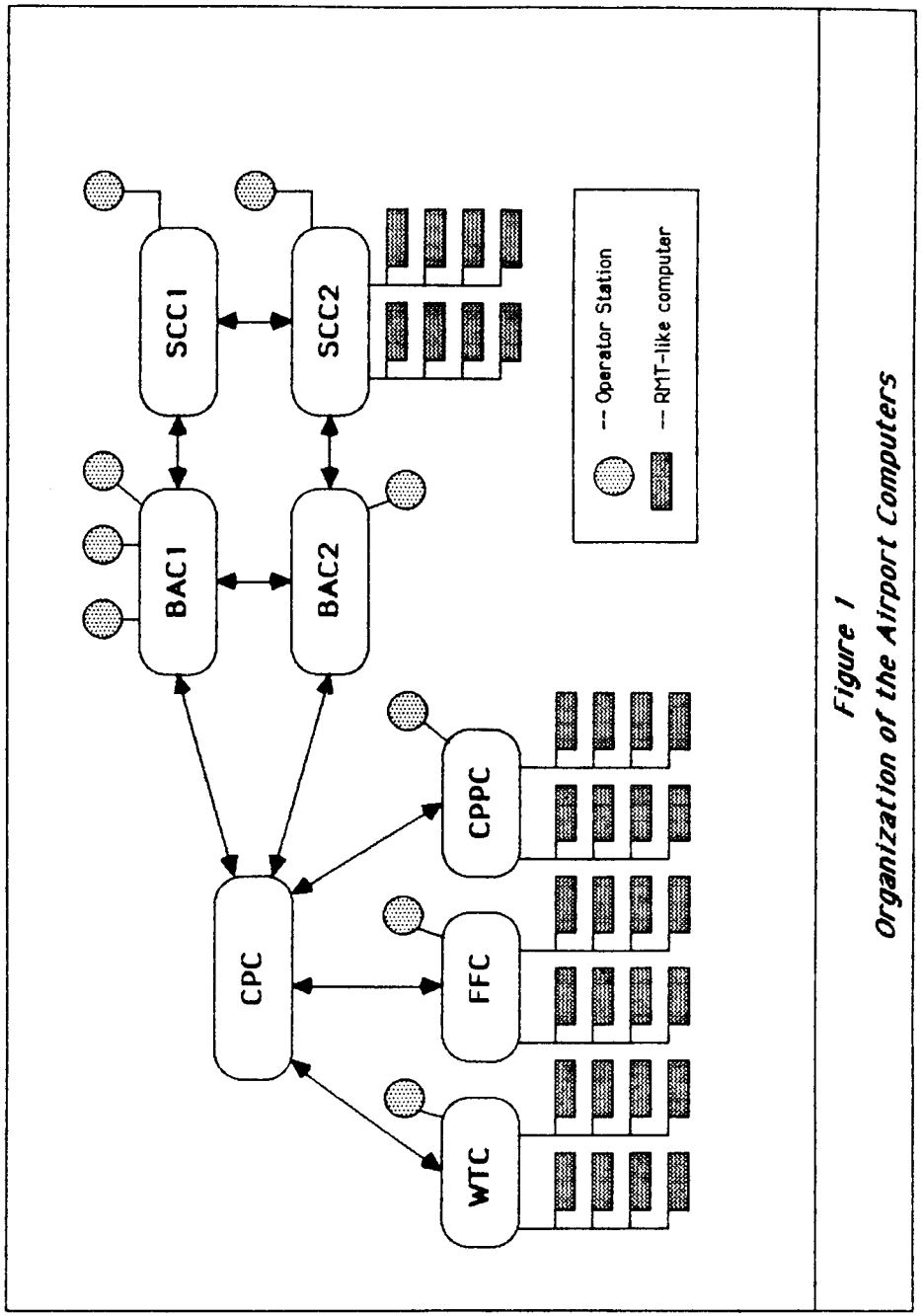


Figure 1
Organization of the Airport Computers

a resolution of 512x512 pixels and can display eight colors. One screen is used to display graphic data (process diagrams, building floor plans, etc.), while the other is used for menus, status displays, reports, and data entry. Each operator station also has two printers, one of which is used for event logging and the other for reports.

Application Overview

The list of subsystems given above (FLSS, etc.) may be thought of as representing a *vertical* list of the applications running on the computer system. As the system was designed in a highly integrated fashion, however, it is also possible to describe it in terms of a *horizontal* list of functions which serve all subsystems. These are:

System Functions include basic operating system functions: disk and terminal drivers plus the general running of a standard polyFORTH system. These functions exist on all computers.

Communication Functions include conversing with other computers over the two line disciplines (DMR/DMC and SDLC) provided. A specially optomized version of the clusterFORTH system (*Rochester Conference 1984*) provides communication services for all computers.

Data Acquisition includes monitoring points, detecting events and alarms, and reporting these up-link to a master computer. This is performed in the RMT computers.

Data Handling includes the function of performing updates in the data base in response to reported events, and routing the events to other computers and/or operating stations that have interest in the events. Security data is handled exclusively in the Security Control Computers (SCC1 and SCC2); other DDS data is acquired by RMTs connected to SCC1 and SCC2 and routed to BAC1 and BAC2 for handling. PMS data is handled by the computer responsible for each subsystem (Fuel Farm, etc.) and routed via the CPC to BAC1 and BAC2 for maintenance of the central data base.

Data Base Management includes handling the files of data in the PDP-11 computers. This is done using the standard polyFORTH Data Base Support System. The main data base resides on the disks attached to

BAC1 and BAC2; the Security data, however, can only be accessed from SCC1 and SCC2. Copies of portions of the data base are passed to various of the other computers as needed.

Man-machine Interface Functions include display of point data, reporting of alarms, file editing and reporting, and similar functions which take place at the color terminals at the operator stations. Certain operators have been designated as "Programmers"; these are the only ones able to make changes in the data base. MMI functions feature menu-driven operation, simplified data entry, and a full-feature graphics editing package for maintaining the graphic data in the system.

The software implementation of these functions will be described in subsequent sections of this paper.

System and Application Software

System software for the PDP-11's and 8086's is based on polyFORTH. In fact, even the LSI-11 boards that provide the "intelligence" for the SDLC communications processors run a polyFORTH system. polyFORTH is designed to be easily modified for use in a specific application; this flexibility was used to full advantage in this system to deliver not only faster performance than was achieved in the prototype software but also a remarkably powerful software development environment.

polyFORTH has been used in applications similar to the ones included in this project (though never all at once!), and members of the software team were often able to adapt previously developed application capabilities for use here. Among the major components that derived from prior work were the clusterFORTH communications protocol, many techniques using the polyFORTH Data Base Support System, an extensive graphics package, and memory management techniques for both the PDP-11's and 8086's.

Some new technology was also developed: significant improvements to the design of cluster FORTH, a "template" for performing generalized searching and editing functions on the data base, a graphics editor, and (perhaps most important) techniques for dealing with software development in a multi-programmer environment. This section will discuss some of these topics.

Operating System Issues

polyFORTH is available in both native (standalone) versions and versions that run as tasks under various operating systems. Considerable thought was given to using an RSX-11 version of polyFORTH early in the project, but in the end all system software was based on native polyFORTH. The primary reason for this choice was performance: the prototype software developed under RSX fell far short of meeting the performance requirements in the system, even though the communcations routines and other critical functions were written in assembler. At the 8086 level, the lack of full multi-tasking in the RMX environment led to both performance problems and excessively elaborate code to perform the required concurrent data acquisition and communications functions.

From the operating system point of view, there were only three *kinds* of computers: the PDP-11's, the LSI-11's in the ICP communications processors, and the 8086-based computers. Each had its own special considerations:

PDP-11 Operating System

There were two major operating system issues on the PDP-11's: memory management and the providing of disk services for a variety of slave computers.

polyFORTH has a standard memory management system for PDP-11's, following DEC's design. At any one time a task operates in a 16-bit address space, which may be mapped in increments of up to 8K bytes to any physical location. The mapping of physical memory adopted is shown in ***Figure 2***, while the map visible to a running task is described in ***Figure 3***. This mapping allows tasks with up to 16K bytes of local dictionary on top of a shared dictionary of 48K bytes of re-entrant code, for as many tasks as needed. These reside in 18-bit memory space (up to 256K). 22-bit space to 1Mb is available for tables and queues. The upper three megabytes are used for a RAM data base, which was implemented by modifying **BLOCK** to interpret a 24-bit block number whose high-order byte is 128 or greater as residing in that region. A special display is produced for the operator on every boot and in response to the command **ALLOCATION**, to show how much memory is available in each section. This arrangement works reasonably well, although a 68000-based computer running polyFORTH/32 might have proven even more flexible.

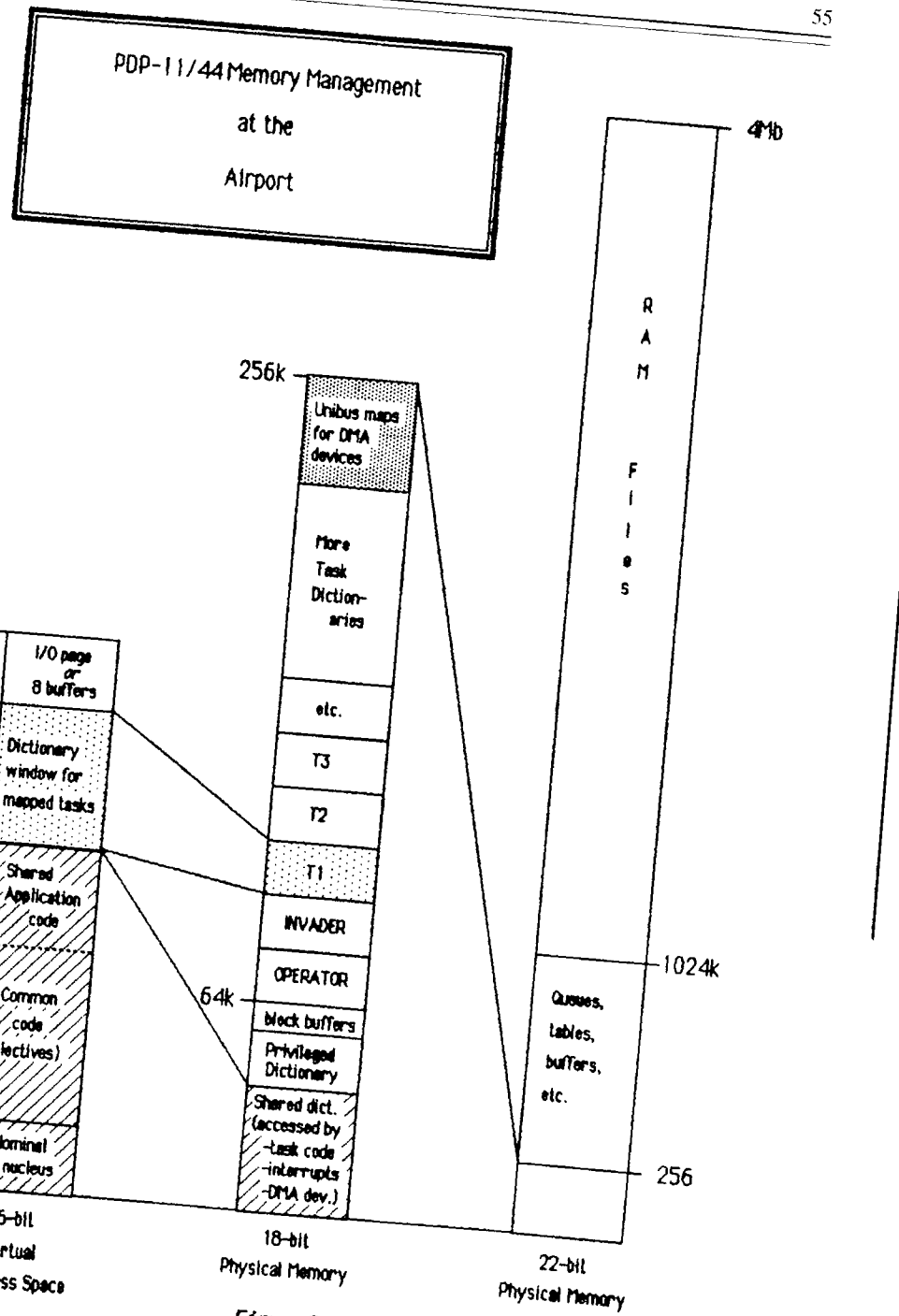


Figure 2

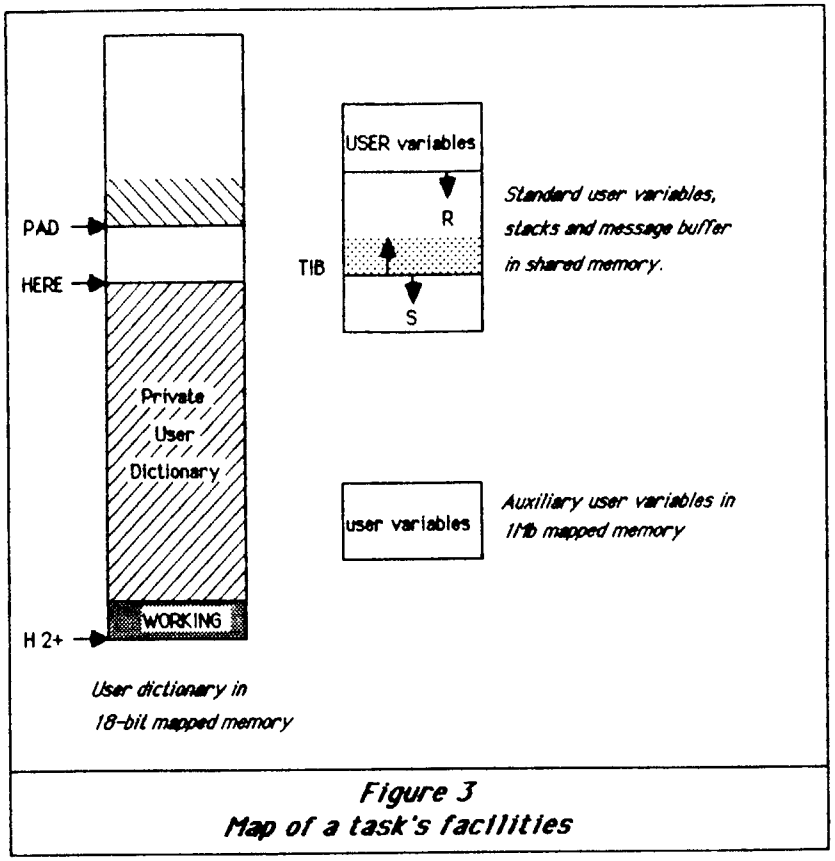


Figure 3
Map of a task's facilities

The PDP-11's provide disk service to each other and to the mostly diskless 8086's on two channels of the clusterFORTH network. The implementation of the disk server will be discussed in the next section; of interest here is the way in which disk was organized and used by various parties.

The BAC and SCC computers each have three drives of 65,840 blocks each. Drive 2 is a removable cartridge. The others have dual RL02 disks, whose full capacity of 10,240 blocks is truncated to 9,600 blocks because that is a more convenient (and entirely adequate) size. The big disks are organized in large regions whose size is 19,200 blocks (or one dual RL02). The first of these on Drive 0 contains Forth source, divided into eight 2400-block partitions. Within each partition there are 1200 blocks for source and 1200 for "shadow block" documentation. The command **n PART** selects a partition. The first four partitions contain

operational software, and the second four contain working copies of these. Each partition represents a major software component: the system and application software for the PDP, ICP and 8086 computers in the first three, and the CPU-independent MMI application in the fourth.

The data base resides in a 19,200 block region. There are two central copies which all running applications access, which are selected by the commands **EDITING** and **OPERATIONAL**. The editing copy is for editing (changing the configuration of points, etc.); whenever an editing session is complete and a revised file ready for use it is copied to the operational data base. A **LOCAL** data base on each computer can be used by programmers to test new software without endangering on-going operations elsewhere.

A convenient display serves to remind users of the organization and status of disk in response to the command **DISKS**. An example is given in ***Figure 4*** In addition, the computer and partition are identified at the top of every **LIST**, and the command **WHERE** reminds a user which is his current data base.

LSI-11 ICP Systems

A fairly standard non-memory managed polyFORTH runs in the embedded LSI-11 processor that controls the ICP communications processors; this system is unique only in that it depends upon its host PDP-11 to provide disk and terminal services via clusterFORTH. There may be up to five ICP's in each PDP-11, and each is capable of servicing seven SDLC links. There are eight tasks in the ICP, one for each link plus the **OPERATOR** task which communicates via the host PDP-11.

The manufacturer of the ICP, Simpact, supplies a real-time executive for the system, which is designed to interface to RSX in the host. At the high baud rates we are using, however (38,400), it can support only two links running concurrently. Due to the zero-overhead interrupt servicing and other operating system efficiencies in polyFORTH we have successfully demonstrated concurrent support for all seven links under heavy load for extended periods.

8086-based RMT Class Computers

The largest group of computers are those based on a pair of redundant 8086's. There are four basic configurations:

179 3 0 BAC1

Prt	From	To	Contains
0	0	2400	11/44
1	2400	4800	86/RMT
2	4800	7200	11/ICP
3	7200	9600	MMIS software common to PDPs and RCDRs
4	9600	12000	Editing copy of 0 PART
5	12000	14400	Editing copy of 1 PART
6	14400	16800	Backup of 2 PART as of 12/18/84
7	16800	19200	Backup of 3 PART as of 4/12/85
019200	042200		Data Base Edit Copy (DRIVE 0 23000 blocks)
085040	108040		Data Base Local Copy (DRIVE 1 23000 blocks)
Drive 1 last copied from Drive 0: 4/10/85			
150880	175345		Data Base (DRIVE 2 23000 blocks)

Figure 4
Display of current disk organizati on

1. The basic RMT has no disk and no local terminal. It communicates with its PDP-11 host on a multi-drop SDLC line.

2. An MCS is like an RMT except it also supports slave RMTs on up to two multi-drop SDLC lines.

3. A DSP is like an RMT except it has a local VT100 terminal and one or two local printers.

4. An RCDR has a local 10 Mb Winchester disk and one or two operator stations consisting of two IDT color displays and one keyboard, plus two printers and (in some cases) a graphics digitizing tablet.

All of these are supported by a common set of ROMs containing an 8K polyFORTH nucleus plus sufficient software to establish communications with the PDP-11 master and request additional boot information. This

consists of a custom program supporting the particular configuration, plus an extensive set of data tables describing the hardware environment in which the computer is to operate: what types of points are connected to it, how often they should be monitored, what their expected condition or value is, etc.

Software development for the RMT family uses the clusterFORTH network. Via clusterFORTH, a programmer at a terminal connected to one of the PDP-11's can "plug" into any specified RMT, at which point the PDP becomes "transparent", providing disk and terminal services through special server channels. The effect is that the programmer is communicating directly with the RMT and running its polyFORTH. Although the line speed of 38,400 baud is not great, the basic efficiency of the software plus double-buffering makes transmission delays virtually invisible.

The number of tasks in RMT-like computers depend on their responsibilities. Each has background tasks for passing events uplink, receiving and processing downlink commands, and processing digital input interrupts, plus a terminal task into which a programmer can plug. In addition, there is a task for each 32-channel analog board, each serial line to one of the 8085-based Local Security Panels (LSP's), and tasks for the IDT operator stations, printers, etc. that might be present. For example, one DSP has 33 LSP tasks, each controlling two serial ports to the LSP (for redundancy), plus the console and printer tasks. All 33 LSP tasks are running their 4800-baud lines continuously and at full speed, with no line delays.

Network Communications

The general principles of the clusterFORTH networking scheme were described at the 1984 Rochester Conference. This implementation is modified somewhat to reduce the number of transmissions to complete a transaction from four to two. This is done by slightly lengthening the packet, so that each transmission can acknowledge the past one and also transact current business. A description of the communications control fields is shown in *Figure 5a*, and a logical diagram of the protocol is shown in *Figure 5b*.

The protocol supports up to 24 slaves on each multi-drop line. The SCC computers each have twelve multi-drop lines, plus several single lines to operator stations; the others have lighter loads. Twelve logical "channels"

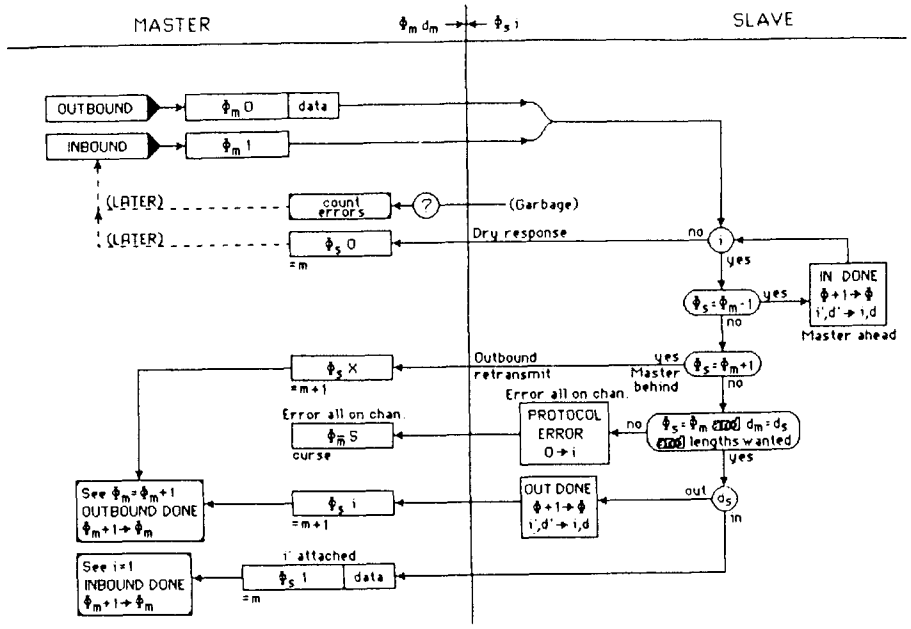


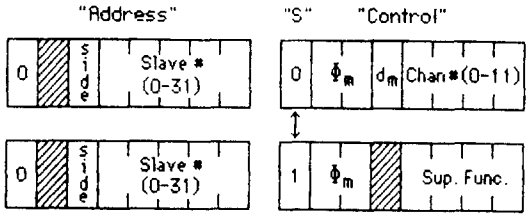
Figure 5a
Logical Diagram of clusterFORTH Protocol

are supported, each of which carries a particular kind of traffic. Channel 0 carries alarms and other events up-link from the RMT's; it has the highest priority, as channel 0 traffic can be carried with every poll. Other channels are polled for individually.

"Supervisory function" codes communicate information about message errors or traffic collisions, computer resets, phase requests, and similar privileged transactions.

There is a bi-level hierarchy in the cluster. In the center is the community of PDP-11's, using 1 M-bit serial lines. The topology of this group is shown in *Figure 6*. The paired BAC computers are the masters. Below all the PDP-11's except CPC there is also a community of RMT-like computers, served by multi-drop or single-drop SDLC links. Some RMT's are one stage removed from their PDP masters, by being served on a downlink from an MCS. The MCS routes messages that it recognizes from the "drop address" to its dependents.

Master's Poll



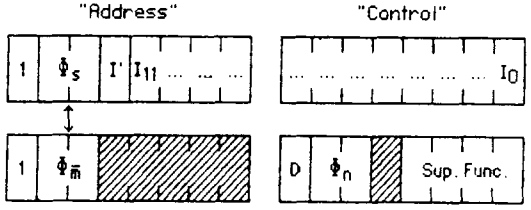
Slave# Maximum of 31 slaves on any hose.

Phi_m Phase; zero initially, set to zero upon any Slave reset. Advances by 1 for each Master state change. There is one state change involved in the moving of each message.

d_m Direction in which Master wants to move data. 0=outbound from Master, 1=inbound from Slave.

S When the Slave sees this bit set it responds only with its supervisory message format.

Slave Responses



Slaves DO NOT RESPOND to garbage.

Phi_s Slave Phase; operates like master's, but leads or lags it by 1 during message transfer. (Lead during outbound; lag during inbound.)

I' Whenever inbound data are sent to Master, I'=1 if there is further traffic queued for this channel. Not meaningful on outbound response.

I_j Interest bit for channel j. 1 indicates interest in traffic on that channel.

Phi_{-m} Complement of master's phase indicates a supervisory response.

D Slave requests a diagnostic halt of the ICP. Overrides all other fields in this byte.

Phi_n Contains Slave phase on requested channel only when supervisory function code is 3.

Figure 5b
clusterFORTH Message Control Fields

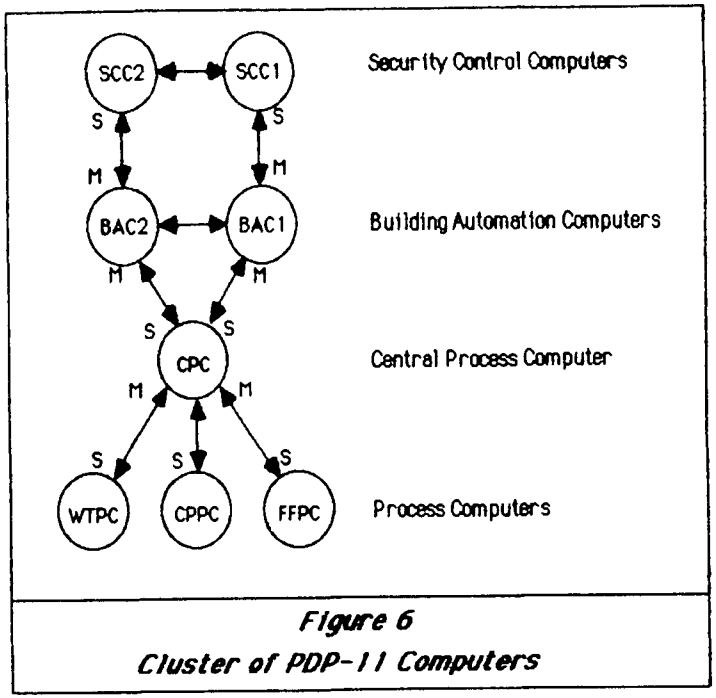


Figure 6
Cluster of PDP-11 Computers

Data Base Organization

Data base structure plays a key role in all the airport applications. The main data base resides on the three 80 Mb CDC disks attached to the BAC computers, although the security data can only be accessed by the SCC computers. The other PDP-11's have local files, as do the RCDRs that serve the operator stations. RAM data bases in the RMT-like computers contain tables describing the points that each station handles.

The standard polyFORTH Data Base Support System was used to describe the airport data base. Enhancements included support for multiple chains of records from other files, a structure somewhat analogous to CODASYL "sets". *Figure 7* shows some of the logical relationships in the data base. There are about twenty-five files, occupying about 22,000 blocks on disk. Several more files occupy the upper 3 Mbytes of memory in the PDP-11's; these contain data required by message routing and other time-critical functions of the system.

There are two main chains. The one most visible to operators is the chain

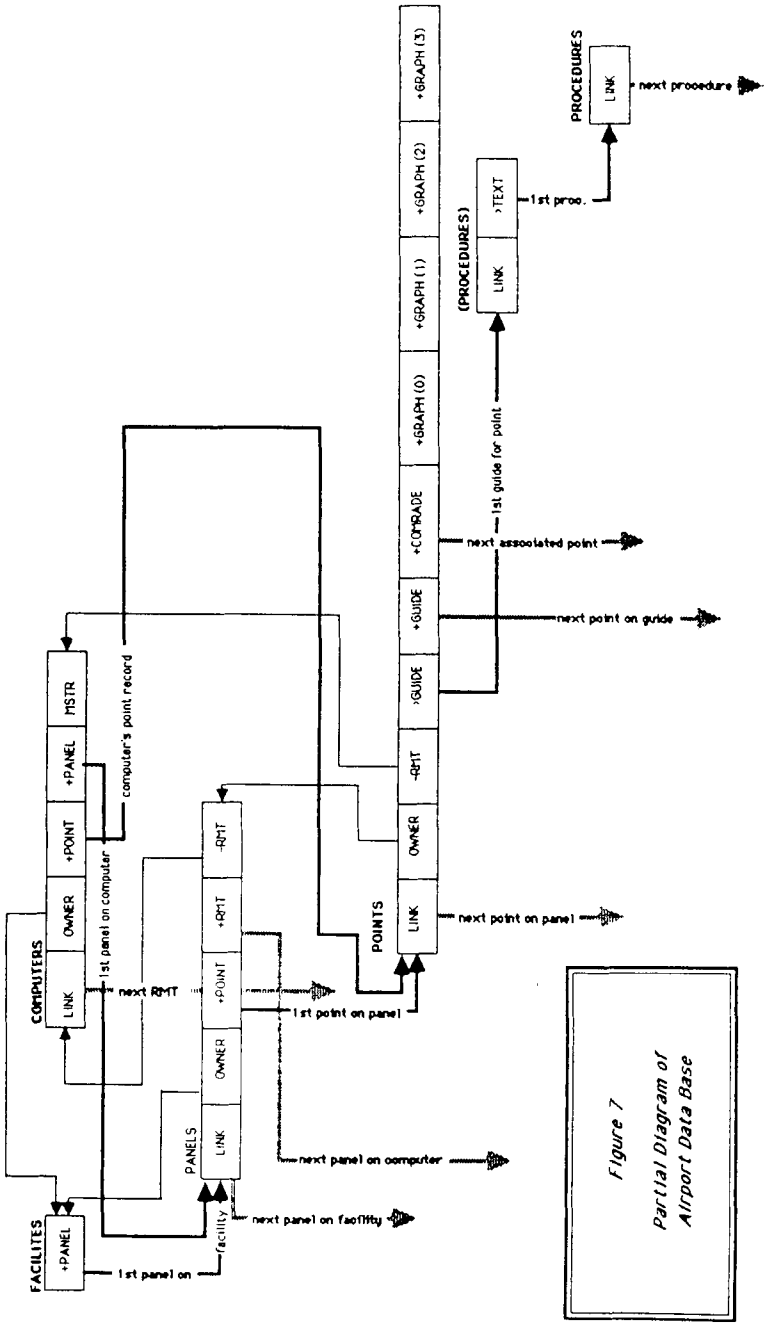


Figure 7
Partial Diagram of Airport Data Base

from a facility (usually a building) to the "panels" within that building to the points attached to that panel. A panel, in most cases, is a physical collection of lights, dials, meters, switches, etc., although there are some "pseudo-panels" defined to handle points that aren't attached to a physical panel. Points are chained from panels, and there are several subordinate files that are chained from points: graphics (to display, for example, the floor plan of a room containing a fire alarm) and operator guides (to specify a procedure to be followed in case of an alarm on this point) are the most interesting.

The second chain is of little interest to operators but is vital to the system. It represents the communications links from computers to points. The top of these chains is in the **COMPUTERS** file, and links master PDP-11 to ICP (communications processor) to hose (which port on the ICP) and thence to each RMT-like computer on that hose. The RMT record, in turn, is chained to the panels supported by that computer. At this point the two chains merge, so that one can then access all points for the panel regardless of how the panel was selected.

Each element in a chain points to the next item in the same chain (e.g. panels for this facility) via a standard field called **LINK**, and also contains a backward pointer to the **OWNER** record (in this case, the facility) for the chain. There are other pointers for direct access; for example, each **POINTS** record points to its owner RMT computer. Standard words are included for managing and following these chains.

A pointer consists of a 16-bit integer, which is the record number of the related item in its file. Thus access is direct (maximum of one disk access) to get from a record to a related one.

Approximately 16,000 points had been edited into RSX files using the prototype software. To initialize the polyFORTH files, tapes were made of the RSX files, and these were then read into temporary disk files, from which individual fields could be picked up and put into the new files. The prototype software was organized using a 32-bit key which was constructed from the network address of each point. Thus all accesses to a point required a search based on this key. There were several files, each containing a subset of the data for a point; for example, the fields peculiar to analog points were all in a separate file. All of these were accessed by the same internal key; thus to obtain *all* the data about a point several searches might be required. In the polyFORTH data base, all information about a point is kept in the point record, and the record size is large enough to accomodate the data required

by the most complex point. Each record has a **KIND** field that identifies the type of record (digital, analog, etc.); fields shared by all kinds are located near the beginning of the record, and the remainder is formatted in a way dependent upon the needs of that kind of point. All internal references to a point are based on the point's record number, which is a function of its physical location on disk. Thus, there are no searches: all internal access require a maximum of one disk access.

External access (e.g., from operators) is according to a *point ID* which has three fields identifying the facility, panel, and point. For example, point **AC1-JLP-002-CR01** is a card (badge) reader attached to panel JLP-02 in facility AC1. Common usage has two digits in a panel number used separately, three in that field of the external ID. The full external ID appeared in all point records in the RSX data base as another key on which searches could be made. The polyFORTH data base uses the hierarchy implied in the external ID to structure the data base itself. Thus, a search for the point in this example would require a binary search of the ordered index to facilities to find the record for AC1; then the panels chained to AC1 would be searched until JLP-02 was found, and finally the points chained to JLP-02 are searched until CR01 is reached. As most of these chains are fairly short, the search time is reduced and the size of the data base is also decreased by not having to keep the full external key in all point records (for display, it can be reconstructed using the backward pointers).

The panel codes (e.g., JLP-02) are encoded for internal storage. The three letters (e.g., JLP) identify a *panel type*, one of about 20. The numeric suffix is simply converted as an integer, and added to the panel type multiplied by 1000H, the result being stored as a 16-bit integer. This saves both space and time in the searches. Treating the suffix as an integer also resolves the ambiguity of the leading zeroes, as -002 and -02 are then identical. Some panel numbers were found to have a letter suffix; since none extended beyond *D*, however, this problem was neatly resolved by converting the numeric part in hex.

In practice, it appears that an operator will tend to be working with groups of points on the same panel, or at least in the same building, for a while. To avoid forcing him to deal with the entire external ID for each point (not to mention the computer having to do a full search) user variables are defined to "remember" at all times a current facility and a current panel. As long as he is dealing with the same panel, only the point number needs to be changed, and within the same facility only the panel.

A *template* facility allows a user to move around in the data base performing easily specified selection and display functions. The template is driven by a set of masks that represent the record in the file being searched. The user may specify ranges of values in any number of fields, for example, and then select an operation to be performed for records that pass the filter. Thus, for example, one may request a display of all points on panels in a certain subsystem (e.g., HVAC), or all analog points with values out of range. It may also be used for "gang updating" (e.g., setting certain initial values in all analog points in the 5-20 mA subcategory). The template has proven invaluable for data base initialization and maintenance.

In later stages of development a potentially awkward situation developed: A team of engineers was exhaustively testing the RMT interface boards and communictions facilities. Meanwhile, a team of data entry operators was verifying other parts of the data base and editing it into conformity with the intended hardware configuration. At the same time, a dozen or so Forth programmers with varying degrees of expertise were writing new routines to edit, update, and display this data base. These groups had conflicting needs: the engineers needed a stable data base that changed only under carefully controlled circumstances; the data entry people needed a data base they could edit; and the programmers needed a realistic data base that they could change (or even destroy if things went wrong) without compromising the efforts of the other groups. On top of all this lay the inherent confusion of eight main computers with local disks.

To handle this, three separate data bases were established, called **LOCAL**, **EDITING** and **OPERATIONAL**. The first was a copy of the data base on Drive 0 of all computers except SCC1, available to the programmers and considered scratch. The **EDITING** data base was on SCC1's Drive 0, and was the one that data entry operators and others making conservative changes used. The **OPERATIONAL** data base resided on SCC1's Drive 2, and was changed at most once a day, at which time it was copied from the **EDITING** data base following exhaustive checks for validity. A set of utilities was developed to facilitate the reconciliation process and ensure integrity of the data base.

Man-machine Interface

Over the years we have worked on many Forth applications designed for unsophisticated users (and reluctant typists). The users of the applications at this airport present some special problems. Not only are they presumed to be hesitant users, virtually none speak English as their primary language.

Although Arabic characters are available on the IDT display terminals, the use of Arabic was only specified for limited applications, since the users may not be Arabs but third country nationals.

The official language is English, but every effort is made to keep displays simple, and typing is kept to an absolute minimum. On an operator's terminal, the only keys used are a few specially labeled function keys and "arrow" cursor position keys. Even facilities, panels, and points are selected by moving a highlighted field up or down through the list presented. Color is used to identify points in various states, and special symbols are used to represent points of various types on floor plans and other graphical displays.

There are three types of users on the system:

Operators are the least sophisticated and least powerful. They can review the status of points, request various displays and reports, and process alarms using pre-defined procedures called *operator guides*.

Programmers are really special operators who can change the data base (add or delete points and other items, change initial values or other processing parameters, etc.). Their powers are much greater than operators, but they still work strictly from pre-defined menus and editing templates such as the graphics editor.

System Programmers are what we would call programmers. They will receive some training in Forth, and can do such things as define special transformations to be applied to data or write special search or display operations to be plugged into the data base template. Even they will not normally have access to all of the system except under special circumstances.

Operators and Programmers both have two IDT screens and one keyboard. One of the IDT screens is dedicated for graphics, and the other for menus and other text displays. The color and graphics capability of the latter is used to highlight and clarify information in the displays. For example, a point in alarm is presented in magenta . As it passes through various stages of processing it turns yellow, green, and finally white. An operator station also has two printers, one dedicated to printing events as they occur and the other available to the operator for reports.

There are a number of tasks defined to handle an operator station: one for

each IDT screen, one processing keyboard input and vectoring commands to other tasks for processing, one each for performing the logic associated with the graphics and menu screens, respectively, and one for each printer. In addition, there are background tasks dedicated to handling communications (receiving incoming events, transmitting down-link events and commands, and monitoring the points associated with the station itself).

A number of structures aid these tasks in performing their functions: alarms are kept in a circular buffer, strings to be displayed on the IDT's are in FIFO queues, etc. These structures reside in extended memory, along with other tables that govern the behavior of the station.

Programmer stations also have a "bit pad" graphics tablet used for editing floor plans, process diagrams, and other graphical displays.

Project Organization and Management

The larger the project, the more important the management of the project becomes. Moreover, the very power that Forth brings to a project can exaggerate the problems caused by either too rigid or too loose a management style: since Forth programmers can move very rapidly, it is imperative that there be a well-defined statement of the application's requirements to coordinate their efforts. At the same time, rigid adherence to a predefined detailed specification renders the team unable to take advantage of Forth's qualities as a flexible design tool.

Prior to the decision to use Forth in the airport project, the management of the project was assumed by a group from AVCO's Systems Division in Wilmington, Massachusetts. These people had used Forth in several projects, mostly relating to defense work, and were well acquainted with both its power and its dangers. This final section will discuss the type of team they assembled and the methods by which we coordinated our efforts.

Design Team

R. A. Norbedo, of AVCO, functioned as technical director and principle designer. He was assisted in early stages by E. Rather, of FORTH, Inc., (data base) and G. Bailey of Athena Programming, working through FORTH, Inc.

(communications). A team of about half a dozen AVCO programmers were assigned the vital task of studying each of the major subsystems and assembling the details of their requirements to feed into the designs. Design work was published initially in memos, and discussed in design meetings with members of the team. Formal design reviews every few months kept representatives of the customer organizations apprised of progress, and provided an opportunity for their feedback into the process. For each of these design reviews, Norbedo assembled the current state of the design into massive volumes which became the "bible" for designers and programmers alike during the ensuing months.

Although Norbedo had spent some years as a programmer, he had never programmed in Forth. He learned quite a great deal about it, however, and was able to tailor his designs to take maximum advantage of its characteristics. He was also exceptionally good at fielding a lively discussion of design tradeoffs among senior Forth programmers until either a clear consensus emerged or an informed decision could be made.

Detail designs were mostly done by the senior programmer responsible for each specific subsystem, and reviewed by Norbedo and other members of the design team. The programmer worked from requirements developed by Norbedo and the AVCO researchers, which were described both in writing and in design review meetings.

The customers were especially concerned about the Man-Machine interface. As this was especially difficult to visualize in the abstract, a simulation was developed using the IDT terminals and dummy data so that ideas for screen formatting, menu handling, etc. could be tested realistically. This simulation was even taken to the site and demonstrated to the supervisors of the people that would eventually use it.

Programming Team

The programming team averaged six or seven FORTH, Inc. programmers and eight or nine AVCO programmers during a given period. Specific individuals were phased in and out of the project, however, so that the total number of programmers involved over the life of the project was nearly twice that many.

As the FORTH, Inc. group were more experienced in Forth and in some of the applications involved, they assumed the lead role on each critical subsystem,

while the AVCO programmers worked on specific application areas. As technical director, Norbedo was the primary supervisor for the group.

Although some of the AVCO programmers had previous experience in Forth, many did not. Several "on-site" courses in Forth were given by FORTH, Inc. during the project, including a special course in the polyFORTH Data Base Support system and a course in Advanced System Techniques for the more experienced members of the group.

A number of techniques were used to keep this large group coordinated and informed, which were fairly successful:

Proximity was probably the most important ingredient. The programming took place in Everett, Mass. for the first six months, and then moved to Huntsville. This required considerable personal sacrifice for the individuals, not to mention substantial expense for AVCO. But given the complexity of the system, the consequences of dividing the project geographically would have been disastrous. In fact, virtually all the programming took place with all active programmers in one large lab, which meant that they could (and did) converse with each other on a regular basis.

Peer review took place on both a formal and informal basis. Meetings were held frequently in which various programmers described the implementation of critical functions, and were then grilled by the rest of the programming team offering suggestions or seeking further understanding. Since everyone could expect a turn in the spotlight, it was all taken in good humor, and the meetings were usually extremely helpful for the programmer as well as the rest of the team. Within the lab, programmers frequently called on each other for help and advice. These informal conversations led to more shared code, reducing the tendancy to "re-invent wheels".

Shared disk was also valuable, as it facilitated the sharing of code. Three of the PDP-11's were used for software development, but they shared disk through the cluster. Although there were occasional frayed tempers when someone made a change that inadvertantly affected others, this was more than offset by the ease with which their efforts were integrated in later stages.

"Newspapers" were established on the disk wherein changes that impacted others could be published and useful added features advertised. On booting, a "help screen" was displayed at every terminal which, among other things, showed the date at which the newspaper had been updated. In addition to the common newspaper, each programmer had a region of disk in which daily logs and "things to do" lists were kept, and in which messages could be left from others.

Each of the senior programmers was assigned a major responsibility in an area having well-defined functional boundaries. For example, Leon Wagner was responsible for communications and data acquisition on the RMT-like computers, Dean Sanderson wrote the system and communications software in the ICP computers (and took over the PDP-11 communications from Greg Bailey), and Ted Antonakis wrote the software in the PDP-11's that accepted "events" which originated in Leon's RMTs and passed up Dean's communications channels. Each step in this chain required close communications between the responsible programmers: Dean and Leon learned a great deal about each other's code, as did Dean and Ted. Ted and Leon together agreed on the exact form of an event packet, but otherwise didn't have as much overlap.

Informal groupings of programmers discussed technical issues of a far-ranging nature, however. These occurred spontaneously, perhaps starting as casually as one overhearing another's chance remark and responding to it, with a two-hour discussion ensuing. These were often extremely valuable, as they allowed the participants to learn about other aspects of the system, reap the benefits of someone else's experience or creative insights, or even just have a knowlegable sounding board for design ideas or possible problem solutions.

"How on earth are you going to test it?"

The system integration and testing phase of every project is the most lengthy and difficult. On the airport project this was compounded by the sheer amount of hardware available, much of which had been custom designed and built. Fortunately the very nature of Forth facilitates thog9
rough testing of both software and hardware.

Each programmer was responsible for testing modules as they were written, and providing "shadow block" documentation for them. Programmers were vigorously encouraged to follow the editing standards established by FORTH, Inc. to ensure stylistic consistency and readability. As a result, programmers were able to read each other's code quite easily -- which was essential, as travel schedules rarely found all programmers on site at any

one time, and people frequently had to investigate or trouble-shoot code developed by an absent member of the group.

Behind the programmers stood a virtual army of people in need of *using* the software, so that the real test came by use. For example, as soon as the clusterFORTH network was operational, programmers were using it to do their work. An extensive data base had been entered using the prototype software, so that as the Forth data base developed there was data to enter and edit, and reports to be run and analyzed. Similarly, as soon as the graphics editor was done entry of approximately 2,000 pictures commenced. Since the programmers were still on site as their work was being used, they could easily iron out any subtle bugs that were uncovered.

The most massive testing job involved the hardware. For every one of the several hundred RMT-like computers, there were modems and various types of interface boards that had to be extensively tested. Although preliminary tests had been previously performed using special test routines written in assembly language, the development of the Forth drivers allowed for more extensive tests to be run over time, gathering detailed statistics on error rates and measurement consistency. These tests were performed by engineers and technicians, who learned how to run the Forth routines and analyze the results.

Since the programmers assigned to hardware-related functions were very familiar with digital hardware, they were in a good position to help resolve errors that couldn't easily be attributed to hardware malfunctions. Problems might be detected by either an engineer or a programmer. If a programmer found a suspected hardware problem, he generally asked Norbedo to assign an engineer or technician to help, and often wrote some special diagnostic routines to assist in the debugging process.

The contract called for a formal 30-day "shop test" of the entire system prior to delivery. At the time this is written, preparations for this test are in an advanced state. All but one of the PDP-11's and about a third of the 8086-based computers have been assembled in a large test facility at AVCO Electronics Division in Huntsville, Alabama. The data base is being edited to reflect this configuration. A precise list of the functions to be available for shop test has been prepared, and specific tests have been devised to exercise all of these functions exhaustively. The tests will be performed by an independent test team, while the programmers will move to Boston to continue work on less critical subsystems. This shop test will provide a thorough dress rehearsal for final installation at the airport.

When will we be finished?"

Forth programmers become accustomed to achieving spectacular results in a very short time, and they tend to be more temperamentally suited to short, intense battles rather than extended campaigns. For most of the people on this project, this is the largest project they have worked on. As all the pieces finally fall into place, there is a sense of accomplishment that is almost electric.

In our hearts, however, we realize that there is a long way to go, even after shop test begins--the long phase of fine tuning, adjustments to things that *work* to make them work more comfortably and conveniently; the entry of the rest of the data base; configuration of security zones and access areas, and a myriad of other small details. In addition, there is already a list of features not included in this contract that the customer will likely find attractive as later extensions.

The plan calls for the AVCO programmers to gradually assume responsibility for the entire system, and a special team is being formed to assist with the final installation in the Middle East. Documentation is being written, supplemented by video tape lectures for training hardware and software maintenance people. For these critical members of the team the project is just beginning.

Conclusions

There are two essential requirements for a successful project: competent management and a skilled professional implementation team. Given these, Forth can increase productivity, shorten the overall time required, and make best use of the hardware and other resources available. Without them, the project is likely doomed regardless of the software used.

As this is written, the project is still incomplete, and it is premature to offer any absolute verdict. All of us who have worked on this project have learned a great deal, both in terms of technical skills and new ways of working together. We have been pleased with the ease with which we have adapted the software to meet our needs. Norbedo and other members of the AVCO management team are convinced that there is no way that our achievements to date could have been obtained with any other programming methods they have used, and plans are under way to use polyFORTH in future large projects at AVCO.