ATTACHED RESOURCE COMPUTING SYSTEM ARC User's Guide

Version 1

December 1, 1977

Model Code No. 50299

DATAPOINT CORPORATION

The leader in dispersed data processing ™

ATTACHED RESOURCE COMPUTING SYSTEM
ARC

User's Guide

Version 1

December 1, 1977

PREFACE

This document describes the Datapoint ARC (Attached Resource Computer) System. This product makes it possible to construct a highly flexible, extremely powerful dispersed multiprocessing system using Datapoint's Advanced Business Processor series of computers. Anywhere from two to hundreds of processors may be incorporated, interconnected via an advanced new interprocessor data distribution subsystem. Most user programs, even those written in assembler language, will need no modifications to run on the Attached Resource Computer.

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

Now that the concepts of dispersed data processing have been widely accepted, it should no longer be necessary to discuss in detail the many advantages and improvements that can be achieved by distributing business data processing tasks among a large number of smaller computers. However, in order to fully understand the concept and purpose behind the Attached Resource Computer, it is first necessary to start from the past techniques of business electronic data processsing. From there, one may proceed to what Datapoint Corporation believes to represent the ultimate direction of dispersed data processing for business.

1.1 Mainframe Batch Processing

The first computers were single-process devices. Initially no more than calculating tab card equipment, these machines accepted a batch of input data, generally on cards, performed some process upon it, and generated a batch of output. Then a different program would be loaded, and this cycle repeated. Long delays resulted between data preparation and the availability of results. Inventory reports would almost inevitably be out of date before they even finished printing. And the processing resources of the machine frequently were very poorly utilized.

In order to make better utilization of the costly central processing units, multiprogramming operating systems were designed for the mainframes. These very complex software systems allowed more than one process to use the mainframe at a time, so that peripheral device delays, such as disk accesses, could be overlapped with CPU processing requirements associated with other processes. Memory requirements rapidly escalated as the number of processes co-resident in the machine went up by factors of three, five, or more. Eventually the processor would still bog down as the total processing tasks expanded to consume every available machine cycle. And in attempting to gain additional performance, more complex operating systems consumed higher and higher percentages of machine cycles, leaving even fewer for actual user program processing.

As the load increased toward machine saturation, there were two choices. The first was to just trade in the older machine for a bigger one. The second was to buy another machine and transfer some of the processing tasks over to it. But problems of common files needed by many of the application systems frequently made it difficult to split the processing up among many independent processors, so a big new mainframe would be required. A while back, upgrading to a larger mainframe was a horrifying prospect, since it frequently meant rewriting entire application systems for the different instruction set of the larger machine.

Then came the age of the compatible mainframes. This concept, in theory, made it possible to upgrade from one mainframe to the next in the series without program changes, since the series all used the same instruction set. This concept almost worked, except for the fact that as one advanced through the series, the operating systems were not compatible. Business users having gone through this operating system conversion know how much fun (ahem!) it was.

But the basic problem still remained. Eventually, the limits associated with the size and capacity of a single mainframe are reached. Mainframes keep getting faster, but the maximum capacity one can expect to get from a single processor is within sight. And what happens when that point is reached?

1.2 Timesharing

Another development of the multiprogramming concept was timesharing. This approach is another way to attempt to get more utilization from a single processor. Even more significant is that timesharing (with its inherent interactive approach to computing) tended to "humanize" computers and make them more responsive to user needs. Users responded by finding more ways in which they could apply the benefits of computing to their own areas, and again the load grew. Timesharing services can simply buy more computers and assign the user base among a larger number of machines. But if many of those users are sharing a set of common files, then those files generally must be duplicated for each machine. And if the users each need to be able to update the common files, such as inventory, order entry, or accounts receivable, then duplicated files can sometimes become very difficult to deal with.

1.3 Transaction Processing and Interactive Inquiry

As a result of the time sharing approach, business began to observe that the time value of data is sometimes quite significant. Airlines were perhaps among the first users to set up on-line, transaction-oriented processing systems to handle reservations and other processing requirements. These systems attached many hundreds of terminals, all of which could access and modify common files, with immediate response to varied inquiries and literally up-to-the-second information. These systems typically used large mainframes for support, generally running specially written operating systems. Since the success of these early on-line systems, business has started realizing more and more that data is more valuable if it is current than if it is days or weeks old. When a customer calls and wishes to place an order, it is more desirable to know that an item is in inventory than to know that there had been one in inventory two weeks ago. When a customer wishes to know his account balance, it is much better to give him his current balance, rather than what his balance had been three weeks ago.

Some businesses printed thousands of pages of voluminous and expensive reports that were frequently obsolete before they finished printing. These reports would try to pre-answer every question an executive might want to know. The report would be delivered to his office, perhaps in triplicate. Then any time he had a question, all he had to do was try to find the answer, somewhere in a thousand pages of printout. And when he found his answer, it was probably no longer current anyway.

As a result of the difficulty created by trying to print literally tons of reports, there developed The Great Printer Race. The mainframes could (and did) generate reports faster than any existing printer could put the reports onto paper, so the mainframe companies developed printers that could print at almost unbelievable rates, hundreds of pages per minute, or more. This technological tour-de-force made it possible to print reports so fast that all the employees of the company, working together, couldn't begin to read them all. This was called progress. And gradually management came to the realization that of the tons of costly reports they were producing, only a small fraction were ever even being looked at by human eyes. But, the reports had to be printed. And distributed. And stored. ...Or did they? Or has the availability of dispersed data processing made many printed reports, previously believed essential, as obsolete as electronic calculators have made books of tables of trigonometric functions, logarithms and square roots?

So transaction processing and interactive inquiry grew. If an executive has a quick inquiry to the data base, the computer itself should do the looking. Giving an up-to-the-minute answer. One which permits better decisions to be made on more timely information. And saving costly paper and machine resources. Source data capture and data base inquiry are just two of many prominent business functions which derive substantial benefits from an on-line transaction processing approach.

But, lo! The big machines with their orientation to batch processing and their complicated multiprogramming operating systems do not lend themselves favorably to online transaction processing. They bog down badly under moderate transaction loads. They require huge, multi-million dollar mainframes to support the necessary number of terminals. And worst of all, when the big mainframe goes down, all the data processing tasks going on throughout the company -- including every terminal in the place -- goes down with it.

1.4 Networks

The ultimate solution of the problems described above is clearly not simply an infinitely fast machine. With a little thought it is obvious that the only way to achieve an arbitrarily great amount of processing throughput is with arbitrarily many processors. Also, using a number of relatively independent processors would seem to provide a solution to the downtime problem, since (if the system were designed properly) a failure in one processor would affect at most only a limited portion of the total processing going on.

So it behooves one to set about finding some way to apply large numbers of processors to solving typical business problems. The concept of a network -- using a number of smaller computers which cooperate to collectively process large amounts of information -- seems like the solution to many of the problems mentioned above. U.S. Navy Captain Grace Hopper, a legend in her own time, has a favorite analogy. If a farmer has a wagon in which he takes his produce to market, and that wagon becomes too heavy for one horse to pull, the solution is clearly not to sell his horse and buy a stronger one (even if stronger ones are available!). The solution is obviously to get another horse to help with the load. By doing this, several advantages are realized. First, it is not necessary to retrain another horse from the beginning. Current operating procedures are disrupted either minimally or not at all. Second, one can proceed to buy a third horse, and a fourth, and so on. The system is open-ended, and can grow to match the increasing load. Third, the incremental costs of increased capacity are both small, and known. Small, mass-produced processors deliver better bang-for-the-buck than large, costly, mainframes --- the incremental costs of adding additional processing power are low. Fourth, when the increased load no longer fits on one wagon, the farmer can buy a second wagon and simply split up his team of horses. The system can be easily and flexibly rearranged to fit changing needs. Fifth, if one of the horses gets sick, the farmer can probably still get his load to town by hitching up the others. It is much less likely that some single misfortune will block the continuing operation of the business.

All that is required to make computers perform together like a team of horses is the appropriate organization and linkage that allow them to work together efficiently. Most networks, however, require major changes in operating procedures and applications programming. Frequently, entirely different approaches are required. And by the time a firm's data processing has matured to the point where they recognize the advantages of a networked approach to their processing requirements, so many programs have been written making so many assumptions about the processing environment that it is almost unthinkable to throw away this investment in applications programming and start all over, no matter how attractive it seems for other reasons.

1.5 ARC

What is needed is a simple, straightforward way to distribute a modern, on-line transaction or batch processing load among a variable number of processors. One which allows additional processors to be added easily as required. One which need not depend absolutely upon the continued availability of some single critical central controlling machine. One which can be implemented without changes to existing applications systems. One that does not disrupt operations. One whose components are treated as building blocks that can be rearranged as a firm's data processing needs change.

ARC is Datapoint's solution to these problems.

CHAPTER 2. ARC CONCEPTS AND OVERVIEW

ARC is basically different from conventional processor networking systems. Traditional multiprocessor networking systems break the terminal-computer-database path between the terminal and the computer, leaving all the actual processing at the place where the data is stored. Terminals, printers, card readers, and other unit record equipment in general may be near the user, but this is essentially still a timesharing or RJE situation. It's not very good at disguising the fact that the processing is still really occurring somewhere else, regardless of whether the user has local "intelligence" or not. And the "main computer", wherever it is, is still a critical component whose failure takes down everything, everywhere.

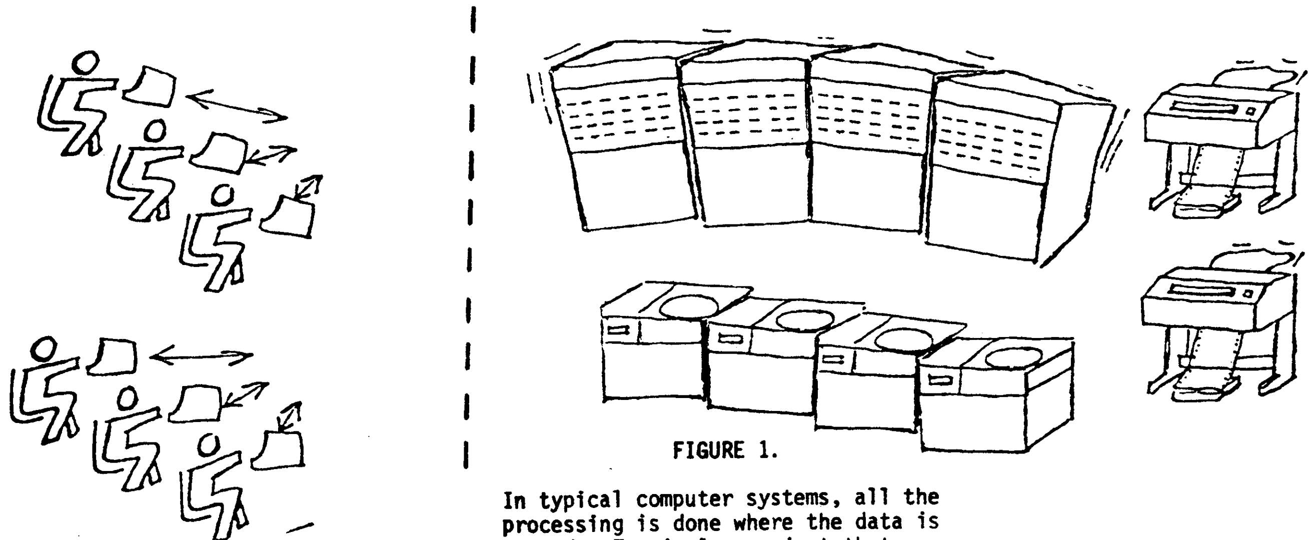


FIGURE 1.

In typical computer systems, all the processing is done where the data is stored. Terminals are just that.

ARC is different because the ARC system breaks the terminal-computer-database path between the computer and the database. All applications computing occurs in the processor from which the processing request originates. The actual data files being used may be off somewhere else, where they may be referenced by somebody else simultaneously, if desired. One of the major differences between other networks and ARC is that ARC permits more raw CPU processing power to be attached to a database than would be practical to provide (or try to utilize) in a single processor.

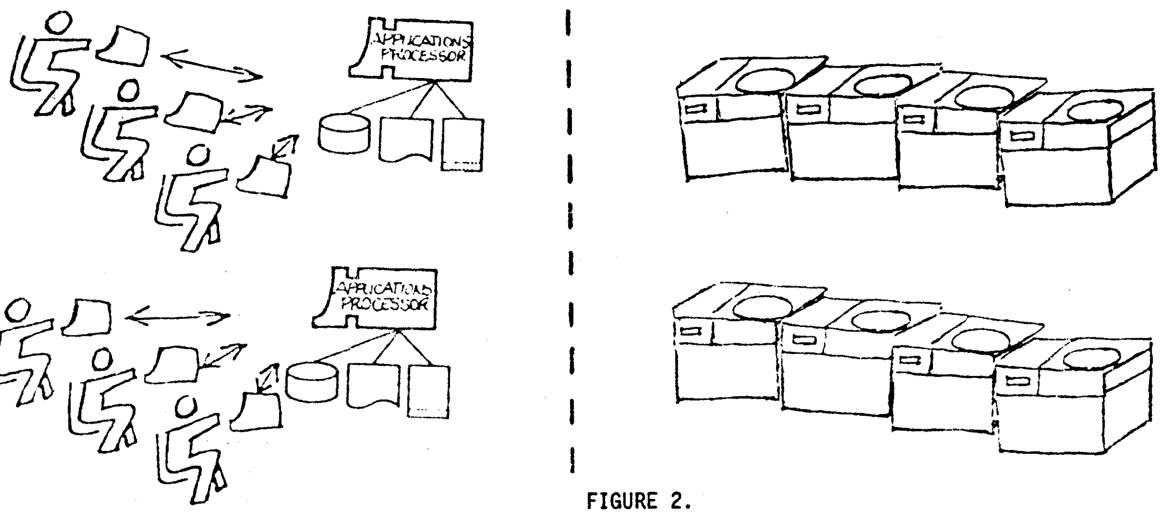


FIGURE 2.

In ARC, the data processing load is dispersed to where the users are. The common data files are stored away from the applications processors.

One main reason why this approach has not been used before is that this approach typically results in very heavy data transfer requirements among processors in the system, and data communications has traditionally been slow, error prone, heavy in overhead, generally expensive to implement and with major software changes being required.

2.1 ARC -- Hardware vs. Software

The product which comprises ARC consists of both a hardware and a software product. Both products work together to provide a series of novel solutions to traditionally nasty data processing problems.

The hardware product consists of an extremely high speed, error-controlled, independent packet-switching system for reliable, short haul, low cost multiprocessor interconnection. This subsystem consists of RIM (Resource Interface Module) adapters which attach to the standard I/O bus of Datapoint Advanced Business Processors and permit communication between processors at millions of bits per second over lengths of inexpensive, shielded single-conductor coaxial cable. This system, the outgrowth of years of Datapoint's experiments in distributed multiprocessor interconnection, contains its own dedicated, high speed processors which monitor and completely control the operation of the link, including data transmission, buffer management, error control, automatic subsystem reconfiguration, and related tasks. As a result, the computers to which the RIM adapters (hereafter simply RIMs) are attached require zero processor overhead for link control, leaving the computers free for applications programs and other useful work.

The ARC software component efficiently utilizes the RIM subsystem (sometimes referred to as the Interprocessor Bus) to provide essentially transparent access to disks (and therefore files) stored elsewhere as though the disks were directly attached to the processor's I/O bus. This permits many different processors to be running independent tasks on any mix of local (private) disks, unit record devices, and communications or terminal equipment while at the same time accessing disks which are concurrently being accessed by other processors executing other programs.

Once an ARC system is assembled, there are essentially three classes defining the status of each Advanced Business Processor at any given instant:

1. Non-participating. No change in operating procedures whatsoever is required due to a processor's simply being attached to ARC's Interprocessor Bus. Processors may be bootstrapped, powered down, powered up, serviced, or anything else, completely independently of the fact that they are attached to the Bus. Active utilization of the RIM subsystem is completely voluntary. Any program that could be run on the configuration before the RIMs are attached can be run exactly the same way afterwards.

2. Participating, Applications Processor. This status occurs when a processor logically "mounts" a remote volume and begins accessing it via the RIM subsystem. A local disk may or may not be present, depending upon the local configuration of the applications processor.

3. Participating, File Processor. This status occurs when a computer is serving as a shared file resource. When a computer is acting as a File Processor, it is making its local disks available to applications processors. File processors do not run application programs while they are serving as File Processors, since the computer is quite busy in servicing the requests made of its shared file resources by applications processors.

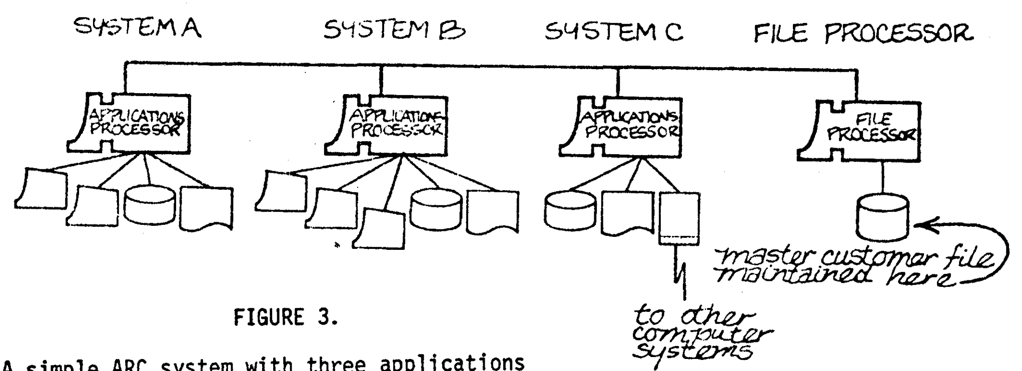


FIGURE 3.

A simple ARC system with three applications processors and only one file processor.

2.2 The File Processor Concept

One basic concept behind the File Processor approach is that, by performing actual computing related to data separately from where it is stored, the only processing at the storage site is that required for simply the actual retrieval and storage of the data. This makes it possible to increase total processing capability associated with a single file to much more than would be practical within a single processor.

The File Processor's function is to buffer data, optimize use of its shared disk resources, resolve access conflicts, coordinate data base update transactions, provide data security, and service incoming data requests. Again, no applications programs run on a processor while it is serving as a File Processor.

The only requirements for a processor to serve as a File Processor are that it be running DOS.D versions 2.4 or above, be attached to a RIM, and that it have a 9370-series or 9374-series disk system attached to its I/O bus.

2.3 The Applications Processor Concept

Applications processors are the computers which actually process the data, either under the control of user applications programming, systems utilities, or other programs. These computers can be equipped with any complement of local I/O devices desired, including disks, tapes, communications adapters, printers, card readers, local (or remote) terminals, or any other supported devices. (Note that since ARC supports DOS.D only, local diskettes and 9350-series cartridge disks can be used under ARC only for limited purposes, such as COPYFILE to transfer files from these devices into the global ARC data structure for further processing).

Under ARC, the applications processor gains substantial new flexibility. DOS.D allows up to sixteen volumes to be on line at any given time, and under ARC these can be local drives or remote drives, either 9370-series or 9374-series, in any combination. Local disks or remote disks can be transferred from one logical drive to another without physically moving the packs. Disks can be taken offline or returned to online status either physically or logically or both, depending upon processing requirements. Names of volumes online and their logical drive numbers at any given time can be displayed. Remote disks (those resident at a File Processor within the RIM subsystem) can be logically mounted for access as if they were directly attached, and logically dismounted with the same ease.

2.4 Resource Interface Modules

The RIM is the basic hardware component which interfaces an Advanced Business Processor to the coaxial cable system to allow interprocessor data interchange. The RIMs attached to the coaxial cable (RIM bus) collectively monitor and control the entire operation of the total RIM subsystem. A complete assemblage of RIMs, the interconnecting coaxial cable system, and any associated hubs (be patient) are referred to collectively as an Interprocessor Bus.

One important aspect of the Interprocessor Bus is the method of addressing employed within the system. Each RIM has two eight-bit addresses associated with it. The first is the I/O bus address. This is the address which the computer must send down its I/O bus to give instructions to the RIM. The second is the RIM bus address. This is the address determining which RIM receives a given packet transmitted through the Interprocessor Bus. Both addresses are determined by strapping options within the RIM and can be changed in the field by a customer service representative.

These addresses are completely internal to the ARC system software, and applications programs have no knowledge of them whatsoever. All addressing requirements are taken care of automatically by ARC. The only time that a user must consider the address structure of his multiprocessing system is during the time that addresses are being chosen when his configuration is being established. Once the installation is complete, operators and programmers running within the Attached Resource Computer need no knowledge of actual addressing details.

RIMs are connected by lengths of low-cost, shielded, single-conductor RG-62 coaxial cable. A minimum RIM link would be two RIMs connected by a single run of coaxial cable (coax). RIM systems including more than two RIMs require one or more "hubs". Hubs come in two basic types, and these are referred to as either "active hubs" or "passive hubs".

2.5 Active Hubs

Active hubs are the devices which permit interconnection of many RIMs to form large RIM systems. Active hubs perform signal conditioning functions for improved reliability of the Interprocessor Bus. In addition, active hubs provide the connecting points for eight (or optionally sixteen) lengths of coax. Lengths of up to 2000 feet of coax may be used between any RIM and its corresponding active hub.

More than sixteen processors can be interconnected by running a length of coax from the first active hub to a second active hub, which will then allow attaching a total of up to thirty processors (fifteen per hub, since one cable port per hub is used for connecting the two hubs together). In like manner, more active hubs may be attached, each up to 2000 feet apart, with individual processors each up to 2000 cable feet from their respective active hub. In this fashion, very large systems of interconnected computers can be established, subject to the restriction that the two most distant points within the Interprocessor Bus be within about four cable miles of each other (maximum of ten active hubs along the longest path between RIMs).

The total number of processors that can be directly attached to the same Interprocessor Bus is 255. However, multiple Interprocessor Busses can be attached to each computer and therefore disjoint Interprocessor Busses can share common applications processors and/or File Processors. Hence, the aggregate throughput of an ARC system is essentially unlimited.

2.6 Passive Hubs

In systems where four or fewer computers are to be interconnected, a special, lower-cost "passive hub" can be used. The passive hub is similar to an active hub, except that it can be used to attach no more than four RIMs together and does not perform the signal conditioning functions as performed by active hubs.

When using a passive hub, since passive hubs do not perform signal conditioning functions, a few special rules apply. First, passive hubs may not be attached to active hubs on the same Interprocessor Bus. Second, they may not be attached to other passive hubs. They may only be directly connected to RIMs. Third, the lengths of the longest two cables attached to any given passive hub must total less than 200 feet.

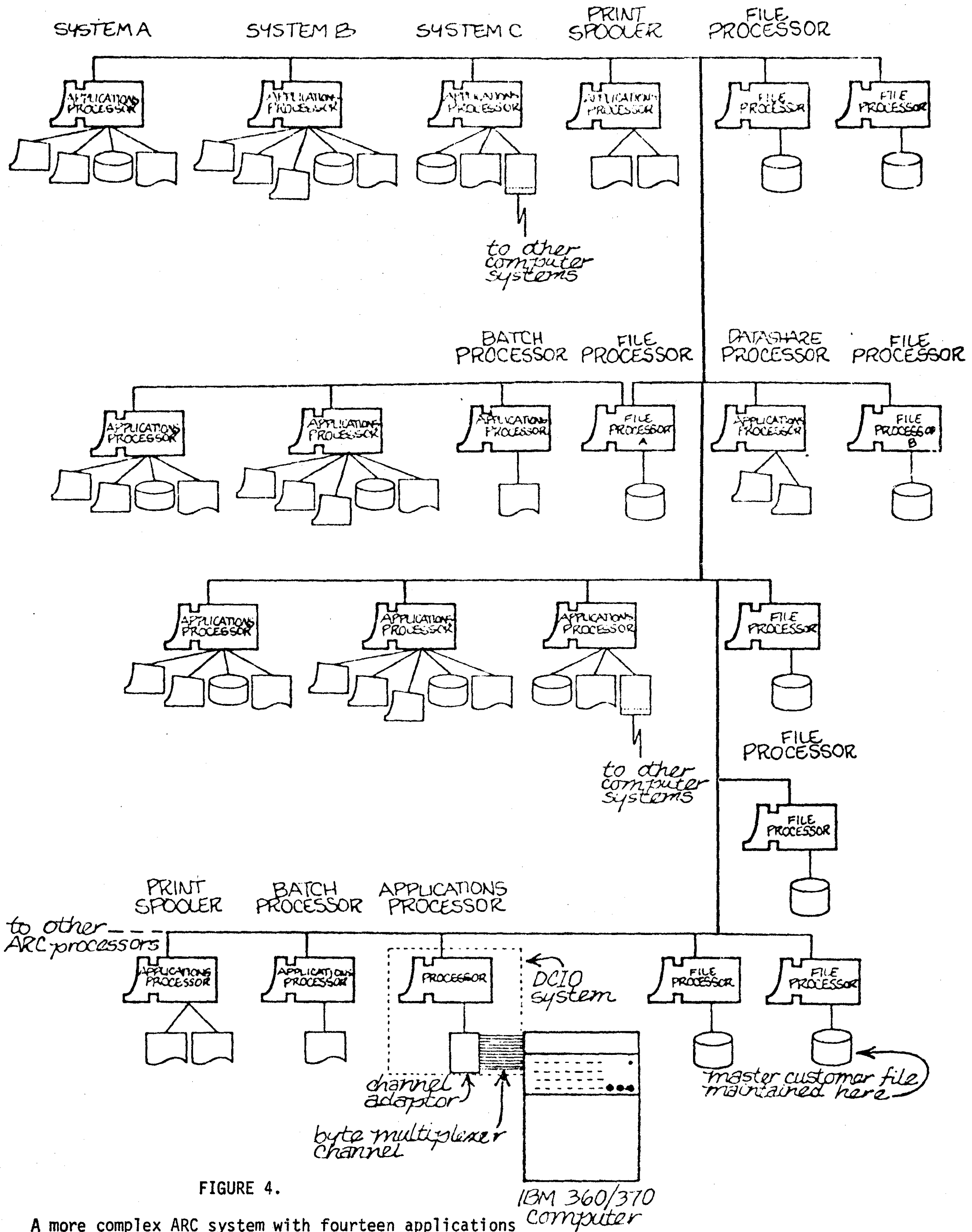


FIGURE 4.

A more complex ARC system with fourteen applications processors, eight file processors, and a mainframe. This ARC also has many local disks, CRT workstations, two printer spoolers and incorporates a local subnet.

2.7 Scope of Volume Access

The set of volumes accessible at any given time by an applications processor is determined by use of the MOUNT command, to be described in a later chapter, which is issued from the console of the applications processor.

Volumes accessed through the Interprocessor Bus are always (initially at least) referenced by volume name. Volume names can be up to eight bytes long and follow the same naming conventions as standard DOS file names. Each time a volume is mounted, it is placed into some one of logical drives 0-15, where it remains until either it is actively dismounted, replaced by another volume, or physically taken offline. Once a volume has been logically mounted, it can be accessed either by referencing the volume name, or the logical drive number in which the volume is mounted.

DOS.D assumes that local volume names are unique; that is, no two volumes directly online to an applications processor may have the same volume name. At any given File Processor, the same assumption applies. Under ARC, it is possible to have multiple volumes of the same name at each of two or more File Processors on the same RIM system under certain conditions. Which volume is referenced when a MOUNT command is issued is determined by the user name and which of those duplicate named volumes that user is authorized to use. In general, it is a good practice to keep volume names unique within the network.

2.8 Use of Subdirectories

Any given volume may have up to 31 subdirectories, the first two of which are SYSTEM and MAIN. Under ARC, access to subdirectories on local disks (those directly attached to an applications processor) is exactly as normal under standard DOS.D operation. Subdirectories on local disks may be created, renamed, removed; the current subdirectory may be changed, and files moved readily from subdirectory to subdirectory, using the standard SUR and NAME commands.

In order to prevent file naming conflicts on shared disks at File Processors, different users of a shared volume are placed into different subdirectories at the time the volume is mounted. (Under DOS.D, up to thirty files on any given volume may all have the same name, as long as they are in different subdirectories other than SYSTEM. See the DOS USER'S GUIDE for a further discussion of subdirectories).

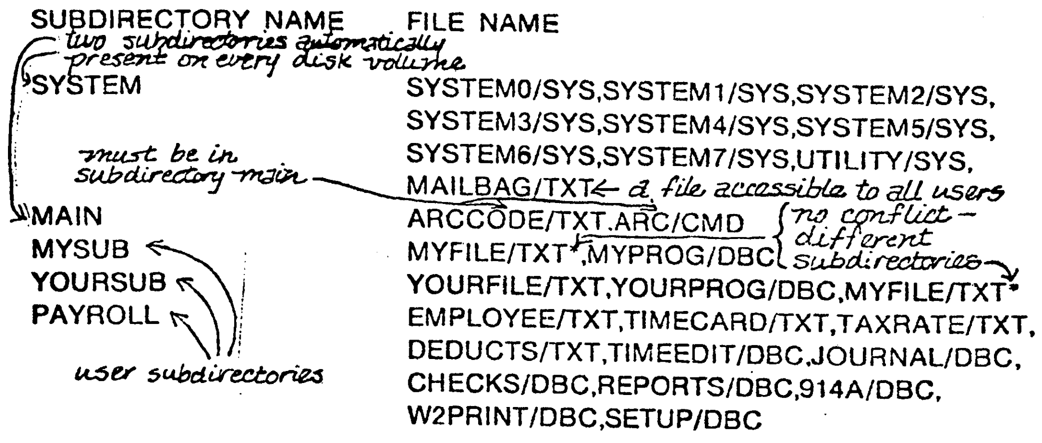
The subdirectory into which the user is placed is determined by his user name. For example, if a user's name is PAYROLL, his application processor will be automatically placed into subdirectory PAYROLL on the volumes mounted at a File Processor. Different users which intend to use files of the same name on shared volumes must therefore be in different subdirectories to guarantee against file naming conflicts.

Since each subdirectory has individual codeword protection under ARC, changing the "current subdirectory" without going through the appropriate security procedures (a proper volume MOUNT) is inhibited. This implies that current subdirectories on remotely accessed volumes, once mounted, cannot be changed via the SUR command. Likewise, subdirectories on remotely accessed volumes may not be renamed, created, or deleted, as this might adversely affect other concurrent users of the volume.

However, files can be moved from a user's current subdirectory to that of another user (even SYSTEM or MAIN) using the NAME command. As under ordinary DOS, files in SYSTEM can be accessed by all users of the volume, regardless of user name.

Since ARC normally prohibits use of user names SYSTEM or MAIN on remotely accessed volumes, SYSTEM may be used as a sort of public-access mode for selected files, while MAIN may be used for special files on the volume which are not to be accessed by any user during ARC operation (such as codeword files or files which are used only for local operations while the owning machine is not running as a File Processor). It is generally a good idea to write-protect files in SYSTEM subdirectory to help avoid their inadvertently being changed by multiple users, except for those files specifically intended for that type of processing.

The important fact to notice about the different users of a shared volume being in different subdirectories is that they may independently run system utilities, such as CHAIN, SORT, or EDIT, with all scratch files on a common volume if desired. No conflict of name usage will occur, since the work files will all be in different subdirectories. When the ARC file processor monitor program (ARC/FP) is later taken down, files associated with each user remain in that user's subdirectory to simplify disk backup and maintenance operations.


two subdirectories automatically present on every disk volume
must be in subdirectory main
a file accessible to all users
no conflict - different subdirectories
user subdirectories

SUBDIRECTORY NAME	FILE NAME
SYSTEM	SYSTEM0/SYS,SYSTEM1/SYS,SYSTEM2/SYS, SYSTEM3/SYS,SYSTEM4/SYS,SYSTEM5/SYS, SYSTEM6/SYS,SYSTEM7/SYS,UTILITY/SYS, MAILBAG/TXT
MAIN	ARCCODE/TXT,ARC/CMD
MYSUB	MYFILE/TXT*,MYPROG/DBC
YOURSUB	YOURFILE/TXT,YOURPROG/DBC,MYFILE/TXT*
PAYROLL	EMPLOYEE/TXT,TIMECARD/TXT,TAXRATE/TXT, DEDUCTS/TXT,TIMEEDIT/DBC,JOURNAL/DBC, CHECKS/DBC,REPORTS/DBC,914A/DBC, W2PRINT/DBC,SETUP/DBC

FIGURE 5.

Different files in different subdirectories. Note the two different files both named MYFILE/TXT.

In the above figure, note that the file MYFILE/TXT exists in both MAIN and YOURSUB subdirectories. These two different MYFILE files are distinct and independent, since they both exist in lower-level (anything other than SYSTEM) subdirectories.

2.9 Data Security

Data security has also been taken into consideration in the design of ARC. First, local disks (those directly attached to an applications processor) are absolutely private, with access under ARC limited to only that processor. (This applies to ARC only; other interprocessor data communication, such as Datashare networking, which can be used concurrently, is of course not affected). Access to remote disks (at those machines then serving as File Processors) requires a volume name, a user name corresponding to a subdirectory name on that volume, and the valid codeword associated with that volume and user name. Different volumes may have different combinations of valid user names if desired, and user names may use different codewords on different volumes. The degree of security desired can therefore be selected by appropriate assignments of users and files to various volumes and user name subdirectories. Security across volumes is much greater than that among different users within a single volume.

2.10 Clock and Calendar Services

Under ARC, the active File Processors cooperate to maintain a time-of-day clock and a perpetual calendar. The clock and calendar may be accessed (from any Datapoint-supported language) by reading the desired values from a file called ARCCLOCK/TXT. The file(s) may be on any File Processor volume(s), in subdirectory SYSTEM. The date is available as month, day, year and day of week. The time is given as hours, minutes, and seconds. More detailed information on the format of this data, and examples on how it may be accessed from various languages, can be found in the appendix entitled, "Accessing ARC/FP Clock and Calendar Services".

2.11 Write Protected Volumes

Volumes which contain especially critical, read-only data, or programs that are not to be modified, may be marked as "write protected". This write protection, determined by the VOLIDPRO field in the DOS VOLID sector, prevents File Processors from permitting any change to the data stored on the protected volume. A command at the File Processor console (issued while the File Processor is active) can change the volume protection status temporarily, or the VOLIDPRO field may be changed on disk by using the PROTVOL command (described in a later chapter).

CHAPTER 3. COMMANDS AND PROCEDURES

The command structure and procedures for operating ARC have purposely been kept simple for ease of use and flexibility. This chapter describes procedures for the many operations that can be performed under ARC.

3.1 Establishing a File Processor

Refer to Appendix C for a description of how to prepare a disk for use from a File Processor. After the disks have been prepared with user names and codewords, to allow the disks at a given computer to be accessed via ARC, the operator at the system console simply enters:

```
ARC
```

This command starts the ARC/FP Monitor, which accepts and services various requests from the applications processors via the Interprocessor Bus.

When ARC/FP comes up, it places the File Processor machine into subdirectory MAIN on all drives. Therefore, any files which ARC uses on disk volumes (such as ARC/CMD or ARCCODE/TXT) should generally be in subdirectory MAIN on their respective volumes. (An exception is ARCCLOCK/TXT, which should always be in subdirectory SYSTEM).

When a File Processor initially comes up, all volumes which are online at the File Processor become eligible for access by applications processors. Any disk at a File Processor may be physically removed and replaced with a different one as long as the disk is not being used. (Subsequent accesses to that disk, if any, will result in "drive offline" status being returned to the application program). Once a disk comes physically online, it should become eligible for applications processor access within about fifteen seconds.

Each disk that comes on line is treated by the File Processor as a completely new volume, (or set of logical volumes) and any volume which is taken off and brought back on line also comes on as a new volume. References to the old volume are not redirected automatically; to reference the newly online volume, it must be remounted by the applications processor.

3.2 Taking Down a File Processor

Once an operator is certain that all remote access to a File Processor's disks has completed, ARC/FP is taken down by simply rebooting the File Processor machine, which returns it to normal, standalone DOS.D operation. Subsequent applications processor access to its disks, if any, results in "drive offline" status being returned to the applications processor program requesting the access.

Note that any given processor can be either a File Processor or an applications processor at any given time, and interchangeably. The only criteria determining which of the two it is, is the software in use. Also, note that an ARC system is not limited to only a single File Processor. There may be as many File Processors as desired, up to the limit of 255 processors per Interprocessor Bus. (Remember, however, that since DOS.D supports only sixteen logical drives on line, any given applications processor may only deal with files at up to sixteen File Processors concurrently). Specific configuration details are discussed in greater detail later in this User's Guide.

3.3 Down-Line Loading

Under ARC, it is not required for an applications processor to have a disk controller or local disks at all. All accesses, including DOS bootstrap, can be accomplished through the ARC system. (Obviously, a down-line loaded applications processor cannot serve as a File Processor).

To bootstrap such a processor via ARC, place the ARC boot tape in the rear deck and reboot the machine normally. This causes the computer to search the RIM system(s) for an active File Processor from which it can be bootstrapped. As it performs this searching, a periodic clicking noise will indicate the cycling. Once the initial bootstrap has completed successfully, the user must enter his user name, codeword, and the name of the volume from which he wishes to load the DOS system.

If the information entered is correct and the volume requested is located at an active File Processor, DOS comes up as normal and from then on, operation is exactly as if a local disk were present. If no such disk is found, or the user has given invalid user names and/or codewords, then ARC will ask again for the correct names.

If upon booting the ARC boot tape, a loud howling sound occurs, then either no RIM is present on the computer or something is wrong with it. (Perhaps the RIM is unplugged).

If the bootstrap results in a slow, regular clicking sound (about once per second), and no ARC signon message occurs (and the cassette is not moving), then either no File Processor is up (or has a valid ARC/CMD file, see below) or else there is trouble of some sort in the RIM's connection to the rest of the system. The coaxial cable may be disconnected, or perhaps a failure within the hub is indicated. If only one File Processor is available, a failure in its RIM may be indicated.

The bootstrap information that is transmitted from a File Processor in response to a bootstrap request is resident in the ARC library file ARC/CMD. As long as ARC/CMD is resident on one or more of the disks at File Processor (in MAIN or SYSTEM subdirectories), the File Processor is eligible to support down-line loading.

If the bootstrap results in very infrequent clicks but still no ARC signon message, this indicates a very high error rate or excessive loading down of the RIM bus. If the condition persists, call a Customer Service Representative to diagnose the problem.

3.4 Applications Processors and the MOUNT Command

Applications processors which have local disks (and RIMs, of course) require no preliminary preparation of their local disks for ARC applications processor operation. The only requirement is that the MOUNT command be resident on a disk at the applications processor so that the command can be used.

Once the DOS is up and running, whether down-line loaded or booted from a local disk, the MOUNT command can be used to examine or change the set of volumes available for access to the applications processor. Running the MOUNT command (in any of its forms) establishes the executing processor as an ARC applications processor, if it has not been so established already.

Being established as an ARC applications processor essentially means that the ARC Multiprocessing System Interface software has been loaded into main memory and initialized. This software resides in System Memory and 4K of user memory. The 4K of user memory required by ARC is removed from the upper end of memory (0130000-0137777) on 5548 and 1170 model computers, and hence programs which expand to use all of user memory will normally still operate within the slightly reduced 44K user memory area.

There are many variants and options on the MOUNT command. First, the different functions and options will be described. Examples will follow section by section. It all sounds a great deal more complicated than it actually is, but should readily become understood after a few minutes of experimentation.

3.4.1 The Display Mode

The display mode of MOUNT performs a function analogous to the CAT command and works much the same way. In the display mode, MOUNT will show the volume name of the volume(s) mounted in either the specified or all logically online drives.

To obtain a display of the volumes in all (logically) online drives, it is only necessary to enter from the system console:

```
MOUNT
```

To obtain a similar display but only for the single volume in a specific logical drive, enter:

```
MOUNT :Dn
```

where n is the logical drive number to be displayed. Note that the same form of this command has a different function if the specified drive is logically offline when the command is issued (see below).

To simply find out the logical drive number in which a specific (already mounted) volume resides, enter:

```
MOUNT :volumename
```

Drives which are logically offline are not displayed when the all-drive display form of MOUNT is used. Logical drives which are online but directed to physically offline disk units (or disks without Volume IDs) are indicated by the number of the physical drive, and the legend "LOCAL".

3.4.2 The Move Mode

The MOUNT command can also be used to logically move a volume from one logical drive to another. The drive from which the volume is moved is taken logically offline upon completion of the move. Any volume which may have been accessed via the destination drive is logically dismounted before the move. The subdirectory of the volume being moved does not change as a result of the operation. Both source and destination drive numbers may be specified either by volume name or by logical drive number. Also note that the MOUNT command can be used to move local disks from one logical drive to another even when no RIM is present, since ARC can run with all-local disks (and no RIM) just as well as with all-remote disks (and no local disk controller or drives).

As an example of the move mode of the MOUNT command, moving the volume in logical drive three to logical drive one only requires entering the command:

```
MOUNT :D3,:D1
```

Note again that this operation removes the volume previously mounted in drive one (if any) and leaves drive three logically offline.

As another example, let's assume that a file processing run has completed, and it is now desired to mount the disk containing the new master file in the same drive that used to contain the original master file. This can be accomplished by simply entering:

```
MOUNT :<newvolume>,:<oldvolume>
```

where <newvolume> is the volume name of the volume containing the revised master file, and <oldvolume> is the volume name of the volume containing the original master file.

Of course, one can also specify either the destination or source drive by volume name and the other by drive number in any combination, as needs dictate.

Moving remote volumes from drive to drive using the MOUNT command in no way interferes with their drive number locations or otherwise disturbs any other applications processor concurrently using those volumes.

3.4.3 Taking a Volume Logically Offline

Any of the sixteen logical drives may be taken logically offline (regardless of whether or not it is physically present with or without a disk spinning in it) by simply moving no drive into the drive to be taken offline. That sounds more confusing than it is to do. For example, if the volume in drive eleven is to be taken offline, whether remote or local, the operator simply enters:

```
MOUNT ,:D11
```

Likewise, the volume to be taken offline may be specified by volume name, for example:

```
MOUNT ,:volumename
```

3.4.4 Bringing a Local Volume Online

There are two forms of "online" that determines the availability of local volumes. One is logically online, the other is physically online. Both must be true for a disk to be accessable. If a local drive is logically online, then it is only necessary to mount a disk in the drive and bring it physically online to access that disk. However, once a local drive has been taken logically offline, the MOUNT command must be used to bring that drive back logically online before access is possible to any disk that may be in it. To bring logical drive five back online, enter at the system console:

```
MOUNT :DR5
```

If that drive is already logically online, the request is simply treated as a display mode request, and the volume name (if any) of the volume is displayed. If the drive is logically online but physically offline, the local physical drive number being referenced through the specified logical drive number is indicated, with the reminder "LOCAL" also being displayed.

Since the drive being brought online was previously offline, of course the actual drive number to use must be given (the local drive to be brought online cannot be specified by volume name).

3.4.5 Bringing a Remote Volume Online

To bring a remote volume (i.e. one resident at a File Processor) online, several key things must be known by ARC. First, the volume name to be mounted must be known. Second, a user name and codeword must be available. ARC remembers the first valid user name and codeword combination it is given, and these are used as default values for subsequent MOUNT operations unless they are overridden for specific mounts. (The "ZAP" option, described later, allows establishing new default values if desired). The user name and codeword to be overridden, if applicable, are specified using options described below.

The basic form of a MOUNT request for a remote volume consists of only a volume name and a drive number where the disk is to be mounted. For example:

```
MOUNT MYDISK,:D12
```

Note especially that the volume name, MYDISK, is not preceded by a colon, and that this distinguishes the mounting of a remote volume from the moving of an already mounted volume to a different logical drive, as discussed in a previous section.

This command will attempt to locate a volume named MYDISK which the previously established user name and codeword has permission to use. A regular clicking sound indicates the progress of the search for the volume. After several unsuccessful tries through each RIM online to the applications processor, an error message is given and MOUNT will simply return to the operating system. The WAIT option, described later, makes it possible to require completion of the mount operation before continuing (which is useful especially for MOUNTs contained in chain files).

The drive in which the remote volume is to be mounted may be specified either by the logical drive number or by the volume name of the volume already in the specific logical drive. If it doesn't matter particularly which drive the volume is to be mounted in, it is not necessary to specify the destination drive. In this case, ARC will choose a drive in which to mount the remote disk. In picking the default destination drive, ARC looks first for logically offline drives, then for physically offline drives. If all sixteen logical drives are in use, ARC does not presume to know which ones are no longer needed, but will then require that a specific drive be indicated.

3.4.6 The WAIT option

One of the options to the MOUNT command is the "WAIT" option. This option is only meaningful when a remote volume is being mounted. Without the option, several attempts to locate the desired volume are made through each of the one or more attached RIM systems before giving an error message and returning to the DOS. If the WAIT option is specified, the MOUNT command will wait indefinitely for the mount to succeed, or until the operator at the applications processor aborts the operation by holding down the KEYBOARD key until the abort has been acknowledged. The abort should be acknowledged within an average of less than five seconds per RIM attached (directly) to the applications processor.

Volume mounts can fail for a variety of reasons. Obvious ones include an incorrectly entered volume name, user name, or codeword. Volume mounts may also fail on occasion due to heavy loading of the File Processor or RIM system. This occurs when the File Processor misses the mount request message (which can occur if its buffers are full when the request is made) or when heavy loading down of the File Processor having the desired volume makes it impossible for the File Processor to respond within about four seconds. If the problem persists, it may also indicate problems with the Interprocessor Bus and should be referred to either a Customer Service Representative or a Systems Engineer for diagnosis.

All options are entered by keying the full option name, separated by commas if more than one is used, after a semicolon. The semicolon follows any volume parameters. For example, to mount volume SYSTEM (using the default user name and codeword) with the WAIT option, it is only necessary to enter:

```
MOUNT SYSTEM;WAIT
```

3.4.7 The STATUS Option

The STATUS option requests that the status of the destination drive be displayed upon successful completion of the MOUNT command. If no specific destination drive is applicable, then the mounted volume status of all logically online drives is displayed. The format of the display is the same as the standard display form of the MOUNT command, described above. This option is primarily useful to verify that the desired MOUNT operation has in fact been performed, without having to enter the display operation as a separate step.

3.4.8 The NAME Option

The NAME option allows the user to override the default name, for the specific MOUNT command operation only, with a different one. (If no user name has yet been established, the name supplied becomes the default name for subsequent mounts). The name may be supplied directly on the command line, for example:

```
MOUNT volser;NAME=username
```

If supplied in this format, the user name may be supplied from a chain file, if desired. User names supplied in this fashion must adhere to DOS file name rules. User names which do not adhere to these rules may optionally be entered directly from the keyboard, bypassing any CHAIN-programmed responses, by simply specifying the NAME option by itself, with no immediate value specified. For instance:

```
MOUNT volser;NAME
```

This results in a prompt appearing on the screen to which the console operator must key in the user name to be used for the volume mount. This can be useful if it is preferred not to leave valid volume names with matching user names laying around in chain files on disk.

3.4.9 The CODE Option

The CODE option works exactly like the NAME option except it allows the specification of the codeword to be used for the MOUNT operation. For many users, the best procedure will be to use the NAME=username option and the CODE option on the command line from their chain files. This way, it is possible to give the correct subdirectory name (user name) from the chain file, but still require the operator to key in the correct codeword from the console, helping to guard against unauthorized usage.

If both NAME and CODE options are specified without immediate values, the values are requested from the operator in the order the options were specified, scanning from left to right.

As an alternative way of supplying the codeword, the codeword may also be specified as the third item on the command line, immediately after the destination drive specifier (if present) and separated from it by a comma. (In order to use either this form or the CODE=newcode form, the codeword must follow DOS file name conventions). For example,

```
     MOUNT volser,:DR2,MYCODE;NAME=SCHMUCK
```

When the codeword is entered as the third command line item in this fashion, the command line is erased immediately upon the MOUNT command being issued (to help to avoid accidental disclosure of the codeword to bystanders). This can be especially useful when MOUNT commands are enclosed in a CHAIN file, and it is desired not to leave the MOUNT command lines, showing user names and codewords, sitting on the screen to be seen by passers-by.

3.4.10 The ZAP Option

The ZAP option causes the previous default values for user name and codeword to be forgotten. The user name and codeword specified either in the same or a following invocation of the MOUNT command will supercede the old defaults.

For example, to establish a new set of defaults for user name and codeword, the following command:

```
     MOUNT   ;ZAP,NAME=newname,CODE=newcode
```

will cause the newly specified name and codeword to replace the ones previously in use. If the NAME and CODE options are not specified, the default name and codeword establishment is deferred until a user name or codeword is specified in a subsequent MOUNT. Names and codewords are separate defaulted items as far as this is concerned, so the ZAP, NAME, and CODE options may be used in any combination desired.

3.5 The ARCCODE File

Access to files at File Processors is protected by several levels of access protection. One of these levels is the codeword required for each user to gain access to a given File Processor volume.

For each user name on a given volume, there is precisely one eight-byte codeword matching that name which will permit access to the volume. These codewords are stored in a file named ARCCODE/TXT in a manner exactly similar to the way the user names (subdirectory names) are stored in the first data sector of SYSTEM7/SYS. There are twenty-nine available user subdirectories and therefore twenty-nine codewords. They are eight bytes long, each, and stored adjacent to each other starting in physical byte nineteen (user data byte sixteen) of the sector. (The first sixteen data bytes are where the codewords for SYSTEM and MAIN subdirectories go if they exist). The sector is terminated with an 015 and a 3, just like any standard non-space compressed text sector. (The ARCCODE/TXT file is terminated with a standard EOF mark). The fact that this sector follows standard DOS conventions permits users to create programs of their choice, using DATABUS, DATASHARE, COBOL, RPG, BASIC, or ASSEMBLER, to manipulate and modify the codeword sector. In this way, users can custom-tailor their codeword maintenance programs to provide any facilities they find useful in implementing their own installations' security conventions.

A sample DATABUS program, named DSARCID, that can manipulate these tables is included with the ARC release package.

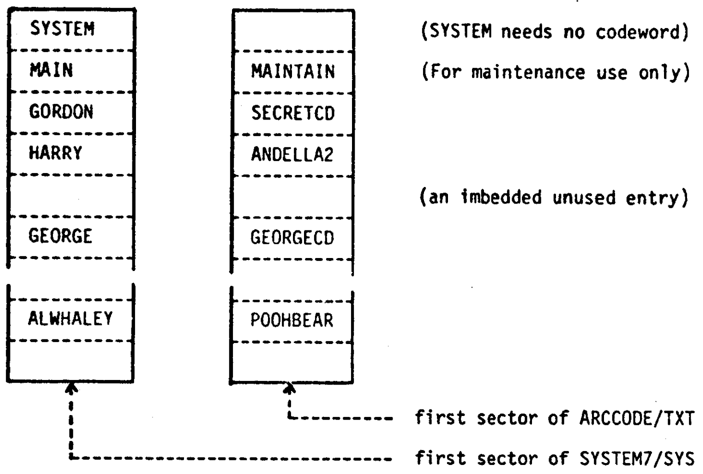

SYSTEM
MAIN
GORDON
HARRY
GEORGE
ALWHALEY
MAINTAIN
SECRETCD
ANDELLA2
GEORGECD
POOHBEAR
(SYSTEM needs no codeword)
(For maintenance use only)
(an imbedded unused entry)
first sector of ARCCODE/TXT
first sector of SYSTEM7/SYS

FIGURE 6.

Diagrams of the user name and codeword sectors. The sectors can be processed using high level languages.

In the above diagram, note the one-for-one relationship between user names and codewords. Also, note that deleted user names are indicated by fields of spaces. It would be invalid to move following user names up to close up the gap, since their location within the sector determines the corresponding subdirectory number which is used internally by DOS. Also note that the codeword for each user name begins in the same location within its sector of ARCCODE/TXT as the user name itself appears within the first sector of SYSTEM7/SYS.

The codeword file ARCCODE/TXT, one being associated with each volume at a File Processor, allows users to have different codewords for each of their volumes for added security. (If no ARCCODE/TXT file exists on a volume at File Processor, the user name is simply used again as the codeword for the volume as well).

In any case, however, codeword maintenance cannot be done while the owning File Processor is operating. Any attempt to access (for either reading or writing) the ARCCODE/TXT codeword sector will result in a "DRIVE OFFLINE" status being returned from the File Processor. (The same status is returned for any attempt being made to modify the subdirectory sector of SYSTEM7/SYS). To help protect against accidental attempts to access the ARCCODE/TXT file while File Processor is active, it is suggested that the file be placed into subdirectory MAIN. In this way it can still easily be maintained while the owning processor is not running as a File Processor.

3.6 Auto-Execute under ARC

Since individual users of a shared volume at a File Processor may wish to auto-execute different programs (or even have no program set for auto-executing), ARC permits each subdirectory (user) of File Processor volumes to have individual auto-execute indicators. (The auto-execute feature of DOS is described in the DOS USER'S GUIDE). The ARC auto-execute table record is the second record in the ARCCODE/TXT file on each volume. This table contains, for each user subdirectory, an auto-execute PFN as an octal ASCII number. This permits each user to independently employ the DOS AUTO, AUTOKEY, and MANUAL commands in whatever fashion his or her needs dictate.

A few special considerations do apply when using auto-execute under ARC, however. The first is important only when using the DOS AUTOKEY command. The AUTOKEY command stores the AUTOKEY command line into the AUTOKEY/CMD file itself. In order for different users to be able to AUTOKEY different command lines, each such user must have his own copy of the AUTOKEY command. One way to do this is to rename the standard AUTOKEY/CMD file to something else, such as "AUTOKEYC/CMD". This file should be in subdirectory SYSTEM, so all users can access it. When it is desired to AUTOKEY a command line, first the user simply copies AUTOKEYC/CMD into a new file, AUTOKEY/CMD, which will then be in his private subdirectory. This copy of the AUTOKEY command may then be run or AUTO'ed exactly as normal.

3.6.1 The AUTOCLR Command

The other special consideration has to do with volume reorganization. The BACKUP command ordinarily clears the auto-execute flag (in the CAT) for volumes which are the result of a reorganization. This is necessary since the reorganization may reassign completely different physical file numbers (PFNs) which bear no relationship to those the corresponding files had before the reorganization. Reorganizing a File Processor's disk volumes (when the File Processor is not active, of course), while correctly clearing the auto-execute flag in the CAT, will not clear the auto-execute PFN table in ARCCODE/TXT. If this condition is not altered, users logging on to the reorganized volume as their system volume will find that they may be auto'ed on the wrong programs. If this occurs, the users may simply reboot, suppress the incorrect auto-execution with the KEYBOARD key, and then issue the MANUAL command, which will clear the invalid auto-execute PFN. Alternatively, the AUTOCLR command will clear all the ARC user auto-execute PFNs on the drive specified, by entering:

```
AUTOCLR :volumename
```

The AUTOCLR command should be run from the File Processor console while the File Processor is not active. AUTOCLR can, of course, be run from a CHAIN file if desired.

3.7 Write-Protected Volumes

As mentioned previously, File Processors recognize a special field (named VOLIDPRO) in the DOS VOLID record. If this field is set to a "W" upon the volume coming online at the File Processor, the volume is marked as being write-protected. This status is indicated at File Processor by the legend "(W)" appearing after the volume name on the display. An attempt to modify any sector on a write-protected volume results in a beep at the requesting applications processor, followed by the same result as if the volume had gone momentarily offline.

Write-protected volumes are also special in that all write-protected volumes are forced by File Processor to have a fully-allocated DOS cluster allocation table. This makes non-specific-volume file PREP operations automatically skip across any write-protected volumes. New files will then default to being placed on the first (lowest numbered logical drive), non-write-protected volume.

3.7.1 The PROTVOL Command

The VOLIDPRO field in the VOLID sector is set or cleared with the PROTVOL command. This command works much like the DOS "CHANGE" command. For example, the command to write protect a volume would be:

```
PROTVOL :volumename;W
```

Likewise, the command to clear the volume write protection is entered:

```
PROTVOL :volumename;X
```

The write-protect status of the volume may be displayed by simply entering:

```
PROTVOL :volumename
```

Once a volume has come on line at a File Processor, ARC/FP's internal write protect bit for the volume may be modified (without affecting the VOLID sector copy on the disk) by a command issuable at the File Processor console. For example, to temporarily write protect the volume XYZ1234, the operator at the ARC/FP console would enter:

```
DISABLE WRITES TO XYZ1234
```

The success of the operation is confirmed by the appearance of the volume write-protect flag "(W)" next to the volume name on the ARC/FP display.

Predictably, there is a similar command to temporarily remove the write-protection of a write-protected volume. To temporarily permit writes to volume MYDISK3, the operator at the ARC/FP console might enter:

```
ENABLE WRITES TO MYDISK3
```

Just as predictably, success of the command is indicated by the disappearance of the volume write protect indicator next to the volume entry for MYDISK3.

Note that the write-protecting of a volume using the VOLIDPRO field does not affect the ability to write on the volume using standalone DOS when ARC/FP is not active.

3.8 The ARCID Command

The ARCID command enables an operator to readily name and rename disk volumes, as well as establish, inspect, modify and delete user names and codewords.

The ARCID command is intended to be run at a file processor while it is not actively participating in ARC. The ARCID command is started by entering the command:

```
ARCID :drivespec
```

The drive spec in the command line only serves to provide ARCID with an initial volume for processing. It is, of course, simple to select other volumes once ARCID is active.

One of the first things which ARCID does upon accessing a volume is to create an ARCCODE/TXT file on the volume if none is already present. The file will be automatically moved into subdirectory MAIN upon completion of ARCID via a CHAIN file.

Once the ARCCODE/TXT file has been created, ARCID displays MAIN and the twenty-nine user names on the selected volume. Next to each user name, any corresponding codeword is displayed as a field of asterisks.

Following this, it asks the user for one or more of a series of commands which tell it what operations to perform. The format of these various commands and what they do is described in the following sections.

3.8.1 Selecting Volumes

ARCID deals with one volume at a time. The user may change the currently selected volume at any time by simply entering a valid drive spec defining the desired volume. The volume may, of course, be specified either by drive number or by volume name. Remember to enter a leading colon before the drive spec, just as for a DOS command line. The colon is how ARCID knows that what you are typing is a drive spec and not a user name.

3.8.2 Assigning Volume Names

ARCID may be used to assign volume names to disk volumes, along with optional owner-supplied information. The additional information is strictly for administrative purposes. The only use DOS makes of this additional information is to display it during such commands as FILES to aid in identification of the specific disk volume.

To assign a volume name of TESTPAY to the disk in drive two, simply enter a drive spec defining the drive in which the disk to be named is located, and then the new volume name, separated by a left corner bracket:

```
:DR2<TESTPAY
```

The user-specified identification information, if it is to be supplied, should follow the new volume name, separated from it by a single comma. For example, to supply a name of ARCSYS to the disk in drive zero, and to record on the disk the date on which it was established, the operator might enter:

```
:DO<ARCSYS,FORMATTED 12/1/77
```

Certain of the commands under ARCID, including all commands to name and rename disk volumes, are referred to as "deferred commands". These commands are queued and do not actually get fully processed until ARCID finishes and returns to DOS, at which point a CHAIN file is started which processes all the deferred commands.

In addition to assigning the volume name, which is a deferred operation, this command will also select the indicated volume for subsequent ARCID operations.

3.8.3 Changing the Name of Named Volumes

Likewise, changing the name of a disk volume can be accomplished by the same technique. The main difference is that, since the disk being renamed is already accessable by volume name, it is not necessary to specify the physical drive in which the disk to be renamed is located.

For example, to rename volume NEWPAY to OLDPAY, the operator only need enter:

```
:NEWPAY<OLDPAY,user information
```

Again, recall that the commands which name and rename volumes are deferred, and therefore do not actually get processed until ARCID completes its operations and returns to DOS. The only immediate effect of the volume renaming command is to select the volume specified for processing by later commands.

3.8.4 Examining User Codewords

User codewords may be displayed, one at a time, for inspection by simply entering the user's name. The codeword will be displayed next to the corresponding user's name on the CRT. The codeword will remain displayed either until the next command is entered, or until the operator presses the ENTER key.

3.8.5 Adding User Names

A new user name for the selected volume may be established by simply entering the new name desired. Optionally, the new user's codeword may be assigned during the same operation by simply also entering the new codeword, separated from the user name by a comma:

```
NEWUSER,NeW code
```

Since subdirectory names (user names) are scanned by the operating system routine SCANFS, all user names must agree with the DOS file naming standard. That is, they may consist of from one to eight alphameric characters. All alphabetics must be upper case only. On the other hand, codewords (while they must be eight bytes long) may contain any character enterable from the keyboard, including imbedded spaces and punctuation. In the example above, notice how the codeword specified contains both upper and lower case letters, along with an embedded space.

3.8.6 Changing a User Name

Changing a user name, predictably, looks a lot like changing a volume name. Simply enter the user name to be changed, a left corner bracket, and the new name. Again, the codeword may also be changed in the same operation by placing it after the new codeword and separated from it by a comma.

Changing a user name and a code word in this fashion does not change the files which the user has available to him within his subdirectory. No files are lost or misplaced during the renaming operation. In other words, every file in the user's subdirectory before the operation will still be in his subdirectory afterwards. The only difference is that the user must enter the new name and codeword to access the volume.

3.8.7 Removing Valid User Names

Removal of user names is a deferred command like volume naming. This implies that user names which are deleted will continue to appear on the display while the ARCID program is still active. The reason why user names are not deleted until afterwards is that the user being deleted might have one or more files in his subdirectory on the volume, and if so, just removing the user name directly would leave his files on disk in a non-existent subdirectory. By deferring the command, all files belonging to the user are automatically moved into subdirectory MAIN at the time the user name is deleted from the volume.

To remove a valid user name, simply enter the name of the user to be removed, followed by only a left corner bracket. Literally, this amounts to replacing a valid user name with a "null" name. For example, to remove TOM from the list of authorized users of some volume, the command might look like:

```
TOM<
```

As an additional check when removing a user, an arrow appears on the display next to the user name to be deleted. The user must verify the operation by entering a "Y" before the command to delete the user name becomes queued.

3.8.8 Changing Codewords

Changing a user codeword is as simple as entering the user name, a comma, and the new codeword. For example:

```
ACTSRECV,over*duE
```

3.8.9 What if I Forget?

If the user forgets the exact form for the command desired, simply entering a "?" will display a list of commands available within ARCID.

3.8.10 Returning to DOS

As per DOS convention, entering a "*" causes ARCID to exit back to DOS. Any deferred commands which were requested will be executed after ARCID returns to the operating system.

CHAPTER 4. CONFIGURING AN ARC SYSTEM

Due to the extremely great flexibility that is available when configuring a system such as ARC, it would be impossible in these pages to describe all possible configurations. Rather, the principles that must be adhered to and other considerations will be detailed. Even once an ARC system is installed, it can be reconfigured in many different ways to give either more or less cost for more or less performance.

Appendix A contains a few examples of how to design ARC configurations for various applications.

4.1 Basic RIM Constraints

Each RIM has eight-bit I/O bus and RIM bus addresses, as mentioned previously. The eight-bit RIM bus address permits up to 255 RIMs to be attached per single RIM system. (No two RIMs within the same Interprocessor Bus may have the same RIM bus address). Six different I/O bus addresses have been assigned for use by RIMs under ARC. These addresses are, primary address first: 0234, 0232, 0231, 0254, 0252, and 0251. This means that each applications processor or File Processor can participate concurrently in up to six distinct RIM systems. Alternately, one or more of the six RIMs may be connected to the same Interprocessor Bus (with different RIM bus addresses). This can be useful, on File Processors, to permit higher total data transfer rates than might be possible with only a single RIM. An additional benefit of having more than one RIM on the same bus for a file processor is that doing so will reduce the number of missed MOUNT requests.

The choice of RIM bus addresses is not completely arbitrary. When multiple RIMs are attached to a single processor within the same Interprocessor Bus, these RIMs should all have consecutive, monotonically increasing RIM bus addresses. In other words, no other RIMS on the same bus on any other processor should have a RIM bus address between the smallest and largest RIM bus address in use on the subject processor. This nullifies the effect of multiple path uncertainties in message routing within ARC, which could otherwise result in indefinite postponement and other system software failures. In order to allow provision for future use of multiple RIMs per processor, it is suggested to allow several unused RIM bus addresses on either side of each allocated one.

4.2 File Residency Options

One factor that can greatly affect the throughput of the Attached Resource Computer is where files are located. Files which are very small but accessed heavily by a number of users are frequently best placed on a shared disk at a File Processor. This is because File Processors perform least-recently-used buffering of sectors read from their disks, and therefore could have the requested sector already in memory. This would eliminate having to read it from the local disk. If, on the other hand, the File Processor is extremely busy, it may be preferable to duplicate the file on a local disk for each applications processor which needs it. A third alternative, again for files which are accessed on a read-only basis, would be to place copies of the file on each of several volumes at different File Processors. This would help by distributing the file load across more access paths (the RIMs at the different File Processors), and more file processing power to deal with the heavy request load.

On example of a situation in which the technique just mentioned might be helpful is in the case of a large, ISAM-organized data file which is to be used by several dozen or more applications processors throughout the company simultaneously. Such a file might be a master part description file, which could be used by Purchasing, Order Entry, Spare Parts, Accounts Payable, Inventory Control, Engineering and Manufacturing systems, all simultaneously. Perhaps it is indexed in several different ways, for example by part number, vendor code and vendor part number, and stock room location and bin number. Since ISAM indices are always enqueued (requested on an exclusive-use basis) during the ISAM lookup operation, this would prevent access to the index by more than one applications processor at any given instant. Duplicating the file and indices at more than one file processor would permit accesses to the file at each of the file processors to proceed in parallel, resulting in a higher throughput and better overall performance.

File residency and how it affects cost, throughput, disk storage requirements, privacy and response times all vary for different specific processing loads and applications, so generalizations are difficult to make. Imaginative use of the unusual flexibility and facilities of ARC make it uniquely adaptable to fit almost any transaction-oriented business processing requirements.

4.3 How Many File Processors?

The minimum number of File Processors required to take advantage of ARC's shared volume accessing facilities is one. A single File Processor can accommodate up to almost 200 million bytes of disk storage, on sixteen logical drives. Again, however, interesting alternatives exist which allow ARC to adapt to individual processing requirements.

Distributing a set of files used heavily by a large number of users among more than one File Processor can result in considerable improvements in performance. This permits the total disk transaction load to be distributed among more than one disk controller, permitting total overlap of disk transfers. Having additional File Processors also spreads the request load among them, resulting in lighter average loads at each File Processor and therefore better overall performance.

Another, less obvious, benefit is that, for each File Processor there is a separate, large, least-recently used list reflecting the most recently referenced disk sectors. Increasing the number of disk controllers and File Processors relative to the amount of disk space present results in a higher percentage of the total disk space being resident already in buffers at a File Processor, which can improve performance substantially.

Having multiple File Processors also brings with it the advantages of additional reliability. When multiple processors may be used as File Processors, this provides additional protection against system downtime. One File Processor being inoperable, with adequate backup facilities designed into the configuration, can probably be reduced to the level of a minor annoyance rather than a catastrophic business interruption.

On the other hand, if the applications processors each have their own local disk systems and only use the File Processor intermittently to access, say, one common file, then one File Processor may be quite adequate.

4.4 How Many Applications Processors?

This question depends almost entirely on the level of throughput required. One applications processor may be quite satisfactory for handling sixteen terminals under Datashare, but in cases where more processing speed is required, these same sixteen terminals will probably experience substantially better performance if spread out among anywhere from two up to sixteen applications processors.

If more than one applications processor is desired in a single place, all referencing volumes via ARC, while still having a set of "local" common volumes accessed only among those several applications processors, there are several options. One is to set up another File Processor, locally, to control the string of shared local disks and coordinate their use among the sixteen applications processors. This new File Processor may be on the same bus as the original, remote File Processors. Alternatively, if the data in the files is extremely sensitive (for example, payroll or costing information) and it is not desired to have its owning File Processor by any means accessible from, perhaps, applications processors in another department, a File Processor could be established on a local RIM system which would attach through an additional RIM on each processor to be allowed access to the sensitive data. In this way, each of the secured department's processors has free access (if permitted by the volume name, user name, codeword security checking) to the department's private data (via the department's local RIM system) while still maintaining easy access, just as before, to data accessible via the Interprocessor Bus (one or more) coming into the department from outside.

This approach of using multiple, partially-overlapped RIM systems (such RIM systems are referred to as "conjoint" if they share one or more common processors, and "disjoint" if not) allows unparalleled versatility to match virtually any data processing requirement.

4.5 How About Dial-Up Access?

Due to the unusually high speed and structure of ARC, it is not really practical at the current time to directly extend full ARC features across the painfully slow facilities currently available within the switched telephone network. However, other data communications software already supported by Datapoint, such as Datapoll, RJE terminal emulators, UNITERM, MTE, MULTILINK, Datashare Networking, and the myriad of user-supplied data communications packages can be used from ARC applications processors, in almost infinitely many combinations, to help front-end remote Datapoint or other computers into the ARC multiprocessing system.

Since each applications processor under ARC can access up to almost two hundred million bytes of high-speed disk storage within ARC's storage hierarchy at any given time, and can execute communications programs of virtually any kind independently of other applications processors, the primary limiting factor of configuration flexibility available is the imagination of the user.

CHAPTER 5. UPDATING OF SHARED FILES

In any system which allows multiple concurrent processes to update shared data items, the problem of how to coordinate these updates is present. This problem is complicated since deciding whether or not an update should be made may depend upon many other data items, and it may be necessary to know the precise instantaneous state of all of them before a valid update will be possible. If several of these are being concurrently updated in a haphazard fashion by different people, it is easy to see that things could become very confused.

This chapter describes the use of the portion of ARC which resolves this dilemma. This portion of ARC is called the Enqueue/Dequeue Subsystem, and something performing its function exists in all multi-process operating systems supporting shared data updating. The discussion in this chapter is intended for systems-oriented programmers who are familiar with assembly language programming; others who are interested may read the chapter purely out of curiosity. In any case, DATASHARE programs which are properly written for multi-port environments using "PI" instructions will automatically invoke the mechanisms described in this chapter (thanks to the DATASHARE system) without further ado.

5.1 Exclusive Use (Enqueue) Timeout

One important aspect of the design of ARC is to ensure that the system never becomes locked up in a state known as "indefinite postponement". This frustrating situation occurs when two processes each wait indefinitely for the other to finish. Since any processor could theoretically fail at any given time, it is not possible to simply turn over a file (or set of files) to any given applications processor for its unlimited private updating. Doing so would lock out access to those files for updating by other programs, and if the first processor were to never finish its operation due to some unforseen failure, this lockout might be permanent. This cannot be allowed to occur.

Therefore, under ARC, all permissions to update one or more files on an exclusive basis expire after a short time, about one minute. This allows multiple processes to perform update transactions upon shared files, as long as these update transactions are not so lengthy as to monopolize the resource. The user is responsible for ensuring that any updates performed, therefore, do not require over one minute, total wall-clock time. After one minute has passed from the beginning of the granting of exclusive update permission, that permission is relinquished (without further warning). Normally, the user's transactions will require only a few seconds each, and the completion of his transaction (as indicated by Datashare's PI statement count expiring, to use a Datashare program as an example) will automatically release exclusive use of the resource so that it may be immediately allocated to any other user requesting permission to use it.

5.2 Multiple Level Enqueuing

Under ARC, provision exists for enqueues at several different levels. These levels correspond to the naturally nested functions occuring as a result of requests made by applications programs. These levels are categorized as follows:

Level 1. Operating system level. This level corresponds to various DOS operations, including but not limited to Disk Storage Management. Applications programs never use enqueue level one directly. One or more transactions at level one could occur during a level two transaction.

Level 2. Individual access level. This level corresponds to one ISAM file access or the like. (e.g., one Datashare statement). One or more transactions at level two could occur during a level three transaction.

Level 3. Multiaccess transaction level. This level corresponds to Datashare PI statements. This level of enqueue supports transactions which may include a number of individual accesses, some of which might result in DOS Disk Storage Management functions being performed.

5.3 Requesting Exclusive Use

Exclusive use of a resource (a file or a volume containing many files) involves two steps. The first is requesting the exclusive use, known as enqueuing (pronounced n-q-ing) and the second is known as dequeuing (pronounced d-q-ing). One enqueues the resource before beginning to look at the data items which must not change during the course of the update. Then, when the update is complete, one dequeues the resource to release it for use by others, who might also have update requests pending for the resource.

To enqueue a resource, ARC must be given an enqueue level (corresponding to the numbers above), and an identification of the resources desired. Resources are described by logical drive number and physical file number, both quantities that can be found for opened files by looking within the DOS Logical File Table. (See the DOS USER'S GUIDE for a discussion of the LFT). A System Call instruction is used to invoke the ARC Enqueue/Dequeue mechanism. When the System Call instruction is executed, the L register contains a binary three, indicating that an Enqueue/Dequeue request is being made. The C register contains the Enqueue level, in binary. Enqueue levels two and three are used for applications programs, as mentioned above. Enqueue requests must be properly nested; one cannot enqueue a resource at level three while a level two enqueue is still current. The DE register pair when the System Call instruction is executed points to a parameter list in main memory, which consists of byte pairs terminated by a byte pair of (0377,0377). The first byte of each pair is the physical file number of the file being enqueued. The second byte of each pair is the logical drive number where the file resides (0-15). Physical file number one, being always reserved for system use, has a special function under ARC. Enqueuing PFN 1 on a drive requests the enqueue of all files on the volume. Up to sixteen byte pairs, plus the terminating pair, may appear in the parameter list.

Upon return from the system call, the CARRY and ZERO flags indicate the overall status of the operation, as follows:

CARRY FALSE indicates that no failure occurred. In this case, if ZERO is also TRUE, this means that the Enqueue/Dequeue operation was a success. The resource is available for updating (or has been released if dequeuing, see below). If ZERO is FALSE, this indicates that the Enqueue operation must be retried. The Enqueue was not granted, but not due to any failure on the part of the requesting program.

CARRY TRUE indicates that the Enqueue/Dequeue request failed. If ZERO is also TRUE, this indicates that the failure was a failure of the RIM interconnection. This might be caused by a hardware or software failure within the RIM system or the ARC software interfacing to the RIM system. If ZERO is FALSE, this indicates a logical failure or inconsistent request, which will not succeed if retried. This might occur, for example, if the drive containing a requested file has been taken offline from the owning File Processor. Retrying an Enqueue/Dequeue operation which has returned with Carry True is generally fruitless.

Again, note that these enqueue requests can be nested.

However, the total time the outermost enqueue will last is a maximum of one minute. Also, no new items may be requested at level two; any items requested at level two must be logically a subset of items requested at level three. For example, if a program requests a file at level two, it must have previously requested either that file or the volume on which it resides (except in the special case of ISAM files, described below) at level three, if any level three enqueue has been previously granted to the given applications processor. Note, however, that level two enqueues can of course enqueue any desired items when no level three enqueue is outstanding.

Also, note that all enqueue requests at a given level must be given within one enqueue system call. It is an error to attempt to enqueue a resource at a given level when any resource already is enqueued at that level or below (equal or smaller enqueue level number).

5.4 Releasing an Enqueued Resource

Releasing an enqueued resource is similar to enqueuing it, except that the DE pair (instead of pointing to a parameter list in memory) simply contains binary zero. The level number for the dequeue operation is in C as before, and corresponds to the value in C when the Enqueue was requested. The return conditions are also the same as for Enqueue operations.

5.5 Exclusive Updating of ISAM Files

When enqueuing ISAM files for update, several special considerations should be taken into account.

First, one ISAM data file may be associated with many ISAM index (/ISI) files. However, to enqueue any ISAM file, one need only enqueue the related data file. This has the effect of enqueuing all related ISAM index files. The /ISI file(s) should not be enqueued.

Second, a level two enqueue should be performed on an indexed file before starting a search (or any other operation) through an index for the file. Otherwise, a concurrently executing program could modify the index file being searched, which might cause the search to yield an invalid result.

Third, the one-minute time limit on enqueues makes it especially important to keep ISAM indices well-organized. If the

ISAM index trees get so badly out of balance that updating transactions (e.g. inserts) take many minutes, this will result in applications processors' enqueues running out before their operations are completed, resulting in possible invalidation of the data file. It is most important that these time limits on enqueues be respected.

5.6 Reduction of Safe Enqueue Time

When multiple File Processors are in use, the total time available before an enqueue starts to run out may be reduced by the amount of time required to satisfy other enqueues requested at the same time. For example, if sixteen File Processors are all in use, and sixteen files are being enqueued (one at each File Processor), the first enqueue which is granted starts timing out as soon as it is granted by the first File Processor. It may take up to a second or so (worst case) for each successive enqueue to be granted by its respective File Processor, resulting in a total of perhaps 45 seconds of wall-clock time available for the transaction.

This is only one of many reasons why update operations (and enqueued time) should be kept as short as absolutely possible. Unusually heavy processing loads or the like could result in extended transaction times, so the less time spent in enqueues, the better. The other major reason for keeping enqueued time to a minimum is that, by definition, enqueuing a resource seriously impacts the additional throughput gains that would otherwise be achieved through the overlapping of multiple concurrent processing tasks. Minimizing resource enqueued time can result in considerably improved system throughput, especially when heavy contention exists for the use of a small number of heavily used resources.

APPENDIX A. SELECTING AN ARC CONFIGURATION

This appendix gives a few detailed examples of how ARC systems can be configured for various purposes.

The ARC system is an extremely versatile and highly flexible system. As with anything which is so flexible, a bit of care taken initially will pay off handsomely later as the configuration expands to fulfill increased processing loads. Since the eight-bit RIM bus addresses must be unique within one Interprocessor Bus, for example, some coordinating authority within the user organization ought to be available to ensure proper address assignments and carefully thought out cabling patterns. This coordinating authority is on the order of the job responsibility of a "data base administrator".

Good systems analysis techniques can also be helpful in determining which users share specific volumes, and the best techniques for maintaining shared files. Some files may be best kept on an applications processor's local disks. Others might profit by being kept at a File Processor, but within the user's private subdirectory. Still others may be best kept in subdirectory SYSTEM on a volume which a number of users may share. Changes in optimum file storage approaches may occur as the extent and applications of the ARC system grow and adapt to changing needs.

The following paragraphs give some guidelines for analyzing system tasks and choosing an appropriate ARC configuration. Careful analysis of system workload will result in an optimum system configuration. However, one of the strengths of ARC is that the system is extremely flexible; few, if any, irrevocable decisions must be made. Designers can shift the location of data, index, and program files, or adjust the mix of file procesors and applications processors, to obtain the best possible performance under changing conditions. forgiving; few (if any) irrevocable decisions must be made, and users should feel free to experiment with various locations for data files, index files, program files, and various such aspects of their own ARC environment.

Therefore, this section deals, of necessity, with many generalities. Hopefully, the section will help the user to develop some feel for the way ARC reacts, and the almost instinctive understanding which may result is the best aid in optimizing a user's own configuration.

A.1 From One to Two Processors

Perhaps many new ARC users will move to ARC from a heavily loaded, single 5500 processor Datashare system. Let us assume that they have sixteen workstations at which a variety of transactions against several different data files are made. Transactions may include lookups into large indexed sequential files, such as customer or inventory files. The user may desire improved response time, better throughput, concurrent RJE to another computer, spooled printing, or more terminals. What paths are open to them within ARC?

The simplest multi-processor ARC system would be to add two RIMs and a second 5500. One RIM would be attached to each processor. The existing 5500 could serve as the File Processor, and the multiport communications adapters would be moved to the new 5500, which would serve as the applications processor. This upgrade might be expected to improve lookup times on the user's ISAM files (since higher-level ISAM index sectors would tend to become resident in the LRU list at the File Processor), and reduce time spent loading frequently used pages of Datashare code (also due to the LRU list maintained at File Processor).

From this simplest ARC configuration, the next upgrade path becomes less clear. If the user's Datashare system tends to be compute-bound, doing a lot of arithmetic, logical and character handling functions, the best approach might be to upgrade the applications processor to either a 6000 series processor, or two 5500s and dividing the terminals between them. If the Datashare system is heavily I/O bound, perhaps the "working set" of disk sectors is larger than the LRU list at the File Processor. In this case, upgrading the File Processor to a 6600 (which has two to three times as large an LRU list as a 5500) might make a considerable improvement in total performance.

A.2 More than Two Processors

A still larger LRU list can be achieved by dividing the disks at a File Processor among two or more File Processors. This approach, using a separate disk controller for each File Processor, not only increases the size of the total LRU list (since each File Processor has its own list), but can also help (in multiple applications processor systems) by permitting multiple simultaneous disk transfers to disks at different File Processors.

A.3 Many More than Two Processors

In general, then, compute-bound systems can best profit by adding more applications processors and dividing the processing load among them. Individual workstations can generally be easily divided among almost any number of applications processors, as required. Additional applications processors may be employed for printing, remote job entry, tape or card reader handling, program generation, communications, or other functions.

Systems which are suffering under heavy disk I/O loads can probably benefit the most by dividing the disk drives among more File Processors, and even dividing the files across more disk volumes if necessary. Moving all /ISI files onto a volume of their own, at their own 6000 series File Processor, for example, effectively guarantees that more than 300 sectors worth of ISAM indices will be resident in the File Processor's LRU list, permitting rapid access to those sectors without any disk seeks being required for subsequent access. Programs which perform numerous lookups in large ISAM files during each transaction may net large performance improvements by using this approach.

Improved reliability and less sensitivity to hardware breakdown can also be achieved by configuring multiple File Processors. If, for example, the user's system has three File Processors and a total of twelve 9370-series disk drives (four per File Processor), adding two more drives per File Processor would give sufficient spare spindles that a failed File Processor's disk volumes could be moved to the spare drives and processing restarted, allowing processing to continue while the down system is being serviced.

APPENDIX B. RIM AND I/O BUS ADDRESS ASSIGNMENTS

As mentioned earlier, both I/O bus addresses and RIM bus addresses are determined by internal jumpers within the RIMs. These addresses can be modified by Datapoint customer service representatives. However, it is necessary for the CSR to be told which addresses the RIMs are to be strapped for.

B.1 RIM I/O Bus Addresses

The first RIM on each processor is normally assigned an I/O bus address of 0234. Subsequent RIMs on the same processor are normally assigned I/O bus addresses of 0232, 0231, 0254, 0252 and 0251, in that order. MOUNT operations search through the RIMs in that sequence, so using earlier I/O bus addresses from the set for the RIMs through which the most frequent MOUNTs occur reduces time required to mount those volumes. (Note that speed of access is the same, regardless of the RIM's I/O bus address, after the volume is initially mounted).

B.2 Several RIMS on a Single Processor

When attaching multiple RIMs to a single processor, the RIMs may be connected to either the same, or different, Interprocessor Busses. (The Interprocessor Bus, to clarify, is the coaxial cable which connects to other RIMs through hubs. The RIM I/O bus is the standard, multi-conductor processor I/O bus which connects the processor to the RIM and other I/O devices). Connecting more than one RIM between a processor and a single Interprocessor Bus is most useful for File Processors, where a very heavily loaded File Processor in a large system can achieve higher RIM throughput and fewer missed volume mounts.

B.3 RIM Bus Addressing

When connecting multiple RIMs between a processor and the same RIM bus, the RIM bus addresses should be consecutive. Therefore, when initially assigning RIM bus addresses, reserving six consecutive RIM bus addresses for each processor may be a good idea. For example, the very first processors in a newly installed ARC system (where only a single RIM per processor, and only a single RIM bus, are used) might be assigned RIM bus addresses of

1, 7, 015, 023, 031, and so on. This approach allows up to forty processors to be attached to the single RIM bus before any breaking up of six-address groups is required. Subsequent RIMs might be assigned addresses of 4, 012, 020, 026, 034, and so forth, still leaving each processor with three consecutive RIM bus addresses available.

B.4 One Processor Straddling Several Busses

The more typical case of more than one RIM per processor is when one or more of the RIMs attach to separate RIM busses. This allows connecting any given applications processor or File Processor into as many as six different RIM bus systems concurrently.

When configuring multiple RIMs on a single processor, each RIM attaching to different Interprocessor Busses, each such RIM should be given the same RIM bus address as its neighbors on that same applications processor.

APPENDIX C. SETTING UP A FILE PROCESSOR

Preparing a system for use as a File Processor is a straightforward operation. First, the ARC system software must reside on some disk volume(s) that will be mounted during ARC operation. Volumes can be mounted and dismounted freely, as long as some volume is always up that has the ARC File Processor system file (ARC/CMD) on it, in subdirectory MAIN (or SYSTEM).

C.1 Loading the ARC/CMD File

The ARC/CMD file contains both the File Processor system monitor and the ARC initializing program which is down-line loaded to applications processors not having their own local disks. The file ARC/CMD is generally kept in subdirectory MAIN. This permits File Processors (which always run in subdirectory MAIN) to find the file, so that down-line loading can be supported. In addition, it helps to ensure against accidental naming conflicts or other accidental misuse of the file by participating applications processors.

For example, to load the ARC system to the volume "PAYROL01" at File Processor, (while File Processor is not active), place the ARC release cassette into the front deck and enter at the system console:

```
SUR MAIN:PAYROL01
MIN ;AO:PAYROL01
NAME ARC/CMD:PAYROL01,,MAIN
```

C.2 Building the ARCCLOCK/TXT File

The file ARCCLOCK/TXT, if one is present, should be in subdirectory SYSTEM. This allows File Processor to see the file, as well as applications processors. The file on disk is never actually changed by the File Processor, so whatever contents the user places in the file during initialization will remain there indefinitely. Normally, the file is initialized to a time record containing seven three-byte fields, each consisting of two spaces and an ASCII '0'. This format permits the file to be read easily from any Datapoint language, including languages supporting unformatted reads (such as BASIC, which uses the intermediate spaces between numbers as delimiters).

The ARCCLOCK file can be easily constructed by using the DOS "BUILD" command as follows:

```
BUILD ARCCLOCK/TXT:volumename;*
  0  0  0  0  0  0  0
*
```

Note that, as mentioned before, each zero is preceded by two spaces, and there are seven zeros in all.

C.3 Defining Valid User Names

Each user which is to be permitted access to a given volume must have a subdirectory matching his user name on that volume. For this, the standard DOS "SUR" command may be used to create named subdirectories on the specific volumes as required.

For example, the following command would establish "TCARLSON" as a valid user name on volume "XYZ":

```
SUR TCARLSON/NEW:XYZ
```

Alternatively, the ARCID command may be used, as will be described in the next section.

Once the subdirectories for the volume have been established (thereby determining who will be allowed to MOUNT the volume), then user codewords must be assigned.

C.4 Defining the Valid User Codewords

The easiest way to initially establish user names and codewords is to use the ARCID program. Let us assume that one has four volumes at a file processor and wishes to name these ACTPAY1, ACTPAY2, ACTPAY3, and SPOOLACT. There are an assortment of users to establish on each volume. In addition, a printer unspooling processor will be used to print reports which have been generated (in standard print-file format) onto the volume named SPOOLACT.

To accomplish the above, the sequence of commands the operator would enter might look like:

```
ARCID :DRO
:DRO<ACTPAY1
SUSAN,SUS CODE
KAREN,KAREN'Sc
```

```
THOMAS,02/13/51
ROBERT,8203398
:DR1<ACTPAY2
SUSAN,SUS CODE
JAMES,123*45 R
KAREN,KAREN'Sc
ROBERT,8203398
:DR2<ACTPAY3
SUSAN,SUS CODE
JAMES,123*45 R
KAREN,KAREN'Sc
ROBERT,8203398
THOMAS,02/13/51
:DR3<SPOOLACT
SUSAN,DIFFERnT
KAREN,KAREN'Sc
THOMAS,02/13/51
ROBERT,APPLEPIE
UNSPOOL,SOMECODE
*
```

Note especially how not all users are permitted access to the same volumes, and how some users might have different codewords on different volumes.

APPENDIX D. ACCESSING ARC/FP CLOCK AND CALENDAR SERVICES

As mentioned previously, File Processors cooperate to provide clock and calendar services to the applications processors participating under ARC. This information is provided in the form of a file, called ARCCLOCK/TXT, which applications programs may read from any volume resident at an active File Processor. The file must be in subdirectory SYSTEM. This file should be write-protected, and must not be moved, created, or deleted while the File Processor is active.

The reason for the ARCCLOCK file not being created or moved is that the File Processor will try to find the ARCCLOCK/TXT file on each disk as it is brought on line. If none is found, the fact is recorded at the File Processor and the fact that one is subsequently created is not recognized by the File Processor. Likewise, if the file were to be KILLed, the File Processor would not realize that either, and any file subsequently created overlapping the same space on disk previously occupied by ARCCLOCK/TXT would be treated as if it were still part of the ARCCLOCK file.

D.1 Structure of the ARCCLOCK File

The file contains one or more logical records. The first of these is a 21-byte record containing seven three-byte fields. Each field contains at least one leading space followed by a one or two byte, right-justified ASCII decimal number. The record is not space compressed. The seven fields, in order from left to right, are as follows:

1. YEAR, last two digits, (0, 77-99).
2. MONTH, (1-12).
3. DAY OF MONTH, (1-31).
4. DAY OF WEEK, (0(=Sat)-6(=Fri)).
5. HOUR, (0(=midnite)-23).
6. MINUTES, (0-59).
7. SECONDS, (0-59).

If the year is set to zero, this implies that either the ARCCLOCK file is not on a File Processor volume (perhaps it is on a local volume), or the clock at that File Processor has not been set. This convention, that of recognizing a year zero as an unset year, is the reason why it is suggested to initialize all seven values of the file on disk to zeros when ARCCLOCK/TXT is originally built.

The ARCCLOCK/TXT files at different File Processors may differ due to both imprecise initial time settings and errors developing due to irregularities in the File Processors' internal crystal clocks. Also, delays within the File Processor or RIM system due to heavy loading or other causes can cause some jitter with regard to the time value returned. Therefore, the ARCCLOCK value should not be used for timing events lasting only very short times, since significant errors could occur. Normally, the ARCCLOCK values returned from different File Processors connected to the same RIM bus will be within a few seconds of each other.

D.2 Accessing the Clock and Calendar from Databus

It might be appropriate to give some examples showing how program segments in each of several languages might be able to access the wall time clock and calendar information in the ARCCLOCK/TXT file.

Example DATABUS/DATASHARE program segment:

```
CLOKFIL   FILE
YEAR      FORM     2
MONTH     FORM     2
DAY       FORM     2
WKDAY     FORM     1
HOUR      FORM     2
MINS      FORM     2
SECS      FORM     2
JUNK      DIM      1
SEQ       FORM     "-1"
          ..
          OPEN CLOKFIL,"ARCCLOCK"
          GOTO ERROR IF OVER
          READ CLOKFIL,SEQ;JUNK,YEAR,JUNK,MONTH:
                   JUNK,DAY,JUNK,JUNK,WKDAY:
                   JUNK,HOUR,JUNK,MINS,JUNK,SECS
          COMPARE "0" TO YEAR
          GOTO NOCLOCK IF EQUAL
```

In the above program, the JUNK data item is used to prevent wasting additional unused digit space in the other data items. Intermediate spaces in the clock record are read into the JUNK data item. This also makes it somewhat simpler to output the time or date using two-byte, right-justified and zero-filled formatting if desired.

While the Datashare system normally supplies a clock and Julian calendar, the ARCCLOCK file eliminates the need to tediously convert Julian dates to month/day format and likewise eliminates the need to compute the day of the week.

D.3 Accessing Clock/Calendar from BASIC

An especially simple example using BASIC:

```
25 OPEN #1,"ARCCLOCK"
30 READ #1,YEAR,MONTH,DAY,WKDAY,HOURS,MINS,SECS
35 IF EOF#1 THEN STOP
40 IF YEAR=0 THEN GO TO ...
```

These few examples should be adequate to demonstrate a few of the many techniques for reading the clock and calendar information from the ARCCLOCK file.

APPENDIX E. ARC/FP CONSOLE COMMANDS

This appendix defines the commands available from the console of the ARC/FP Monitor. These commands include commands which are useful both for performance monitoring and for administrative purposes. The commands can essentially be grouped into two major classes, the first being performance monitoring commands and the second being security-related commands.

E.1 Performance Monitoring Commands

This group of commands enables audible feedback to give an indication of ARC/FP activity of several different varieties. An audible "click" can be caused when one of several possible operations occurs. These operations include the receipt of a message through the Interprocessor Bus, the transmission of a message through the Interprocessor Bus, a request from an applications processor to read a sector from a volume, likewise to write a sector to a volume. Clicks can also be enabled while enqueue requests are pending, or when dequeue requests are processed. The general format for these commands are as follows:

```
ENABLE CLICK ON <occasion>     or
DISABLE CLICK ON <occasion>
```

The <occasion> can be any one of:

```
RECEIVE
TRANSMIT
READ
WRITE
ENQUEUE
DEQUEUE
```

Some examples of valid commands are:

```
ENABLE CLICK ON RECEIVE
DISABLE CLICK ON ENQUEUE
ENABLE CLICK ON WRITE
DISABLE CLICK ON READ
```

The different occasions are defined as follows:

1. RECEIVE. When this click option is enabled, a click is

sounded for each message received through the Interprocessor Bus. This includes messages which may later be discarded without reply by ARC/FP.

2. TRANSMIT. This click option, when enabled, causes a click to sound upon each packet transmitted from ARC/FP through the Interprocessor Bus.

3. READ. The READ click option causes a click for each sector read processed by ARC/FP. The click occurs even if the sector is found in the LRU list and not physically read from the disk volume to satisfy the request.

4. WRITE. This click option causes a click for each sector write request packet received by ARC/FP.

5. ENQUEUE. This click option occurs whenever an enqueue request is processed by ARC/FP. This option will cause irregular clicks when enqueues are being processed at normal rates, but an enqueue blocking action (which occurs when a processor requests a resource already in use by another processor) will result in a rapid sequence of these clicks. This can be useful to get an intuitive feel for the degree of system degradation occurring due to conflicting enqueue requests being processed.

6. DEQUEUE. This option results in a click sounding for each dequeue request processed by ARC/FP.

In addition to the individual options, there are master commands which enable or disable all click options as a group. These two commands are:

```
ENABLE ALL CLICKING     and
DISABLE ALL CLICKING
```

These commands, as their names imply, turn on and turn off all of the individual click options, respectively.

E.2 Security-related Commands

The other commands are useful for administration and maintenance of the disks at the File Processor. These commands have each been described earlier in this document, and are mentioned here primarily so that all the console commands will be available together in this appendix for easy reference.

The first such security-related commands allow one to turn on and off the volume write protection for individually specified volumes. As described earlier, these commands do not actually change the write-protect flag on the volume (although, using PROTVOL, it too could be changed while the volume is temporarily write-enabled via the console command). Instead, these commands merely temporarily enable or disable writes by setting or resetting ARC/FP's copy of the volume write protect flag in memory. The form of these commands is:

```
ENABLE WRITES TO <volumename>     and
DISABLE WRITES TO <volumename>
```

An example of a valid command for a volume named PAYDATA would look like:

```
ENABLE WRITES TO PAYDATA
```

The other security-related command allows the operator to momentarily permit applications processors to mount a single volume (at a time) in subdirectory MAIN. This allows a user at an applications processor (if he has the correct code word and knows exactly when to issue his MOUNT request) to mount a volume giving him access to subdirectory MAIN. This can be of use for administrative purposes to allow (carefully) updating of stand-alone program files which are maintained in that subdirectory. Once the volume is mounted into subdirectory MAIN by an applications processor, the operator at the ARC/FP console can safely "close the window" without interfering with the already-mounted applications processor's access to subdirectory MAIN on the given volume.

The console command to temporarily enable access to subdirectory MAIN looks like:

```
ENABLE MAIN ON <volumename>
```

Upon acceptance of this command, user logons to MAIN on the indicated volume are accepted until the operator again presses the ENTER key (in response to the reply "PRESS "ENTER" TO DISABLE AND CONTINUE"), which "closes the window".

APPENDIX F. USING THE ARCBOOT DISKETTE

While it is not possible to use an 1150 or 1170 series computer as an ARC File Processor, these may be used as non-local-disk Applications Processors. This appendix describes the characteristics peculiar to the use of a diskette-based computer when used as an ARC Applications Processor.

F.1 Contents of Release Diskette

The ARCBOOT diskette release consists of a diskette containing the bootstrap program, called ARCBOOT/ABS. This program must be loaded and executed to down-line load ARC and DOS.D from an active File Processor. To run the program, it is necessary to first bootstrap the diskette operating system in the normal manner. Then, the LGOPROG command (released separately) must be used to load and execute ARCBOOT/ABS. The use of the LGOPROG command is described in the LGOPROG User's Guide.

F.2 ARC Operation on Diskette-Based Processors

Bootstrapping ARC from diskette in this manner may be simplified somewhat by using the DOS "AUTO" and "AUTOKEY" commands to automatically invoke LGOPROG and run the ARCBOOT program. If this technique is employed, it is possible to make a diskette which automatically brings up ARC by merely pressing the RESTART and RUN keys.

Once the file ARCBOOT/ABS has been executed via the LGOPROG command, operation of ARC is just as it would be if ARC had been loaded from the ARC boot tape on a 5500 series processor.

It is possible to use the COPYFILE command to copy files from the local diskettes into files maintained under DOS.D at a File Processor. However, ARC does not support the use of COPYFILE to transfer files from a File Processor's disks to local diskettes on an applications processor. The only way that COPYFILE can directly transfer files in that direction is to attach a diskette unit to the File Processor and use COPYFILE at that machine while it is not active as a File Processor.

Since it is not possible to use the Datapoint 1150 or 1170 as a File Processor, the ARCBOOT diskette release, as noted above, includes only the bootstrap program. The ARC cassette release, containing ARC/FP and Applications Processor commands, is also required for ARC system operation.

APPENDIX G. USING 3800 AND 6000 SERIES COMPUTERS

Datapoint 3800 and 6000 series computers, including the 3810, 3820, 6010, and 6020, may also be used as applications processors under ARC. These machines automatically bootstrap from ARC and the Interprocessor Bus, since they have no cassette tape drives. Other than the source of the initial bootstrap, and the fact that cassette tape software is inoperable, these models behave just the same as has been described earlier.

APPENDIX H. ASSEMBLY LANGUAGE SYSTEMS UNDER ARC

In general, assembly language systems that use the DOS for their disk operations and generally adhere to the DOS programming conventions discussed in the DOS USER'S GUIDE should run under ARC without modifications. The differences from normal DOS operation that might interfere with a currently operating user-written assembly language system are detailed in the sections which follow.

H.1 Memory Residency

The ARC system uses just under 8K of memory in the Applications Processor. Therefore, the appropriate DOS FUNCTION should be used to determine the upper bound of user-program usable memory. (In general, the size of memory available after ARC system software is loaded is the amount of memory physically addressable below 0160000 (which 4K is referred to as System RAM), minus 4K. A further 4K is removed for 3800 series processors, which require the 4K sector of memory at 0150000 for buffers and other system information). User programs must not use any memory above their allotted space for any purpose.

H.2 Addressing the Disk

Since there may be no 9370-series Mass Storage Disk attached to the I/O bus of the computer, programs running under ARC obviously must not attempt to address the disk controller or do any form of access to the disk system themselves. This includes the use of the DOS routine DSKWAT, which references the "currently selected" disk drive. The results of calling DSKWAT while ARC is active are undefined.

H.3 Use of Foreground

ARC does not use the DOS foreground handling mechanism at all. It uses foreground time only for event timing during transfers through the Interprocessor Bus to other computers and even during these times the processor time involved is less than 50 microseconds per millisecond. Therefore, user foreground routines should generally require no changes to run under ARC. ARC uses only the firmware millisecond interrupt vector in the

0170000 sector of memory to support its foreground timing operations.

H.4 Stack Usage

Under ARC, some of the DOS routines, including most disk operations, use one stack level more than the corresponding routines use when running in a standalone mode. Applications systems programmed in assembly language which use every one of the sixteen available hardware stack locations will probably need to be modified to save one or more stack levels at one or more places internally to achieve reliable operation under ARC.

H.5 Multiprocessing Environment

Any programs, whether written in assembly language or otherwise, which rely upon the fact that they have exclusive use of any file, may encounter conflicts under ARC when those files are at a File Processor and are concurrently being modified by another Applications Processor. Either the ARC Enqueue/Dequeue facility must be employed, as described in an earlier chapter, or the file may be protected against concurrent access by locating it in a user's private subdirectory.

In general, these restrictions are minimal, and should only affect a very few users.

INDEX

Manual Name______________________________

Manual Number______________________________

READER'S COMMENTS

Did you find errors in this manual? If so, specify by page.

____

____

____

____

____

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

____

____

____

____

____

____

Name______________________________ Date____________________

Organization____

Street____

City____________________ State__________ Zip Code____________________

All comments and suggestions become the property of Datapoint.

Fold Here

Fold Here and Staple

First Class
Permit
5774
San Antonio
Texas

BUSINESS REPLY MAIL
No Postage Necessary if mailed in the United States

Postage will be paid by:

DATAPOINT CORPORATION
Product Marketing
8400 Datapoint Drive
San Antonio, Texas 78284