5500 DATASHARE 3
DS35500
User's Guide

Version 1

July 1975

Model Code No. 50158

DATAPOINT CORPORATION

The Leader in
Dispersed Data Processing

Datapoint
Printing Services 76

5500 DATASHARE 3
DS35500

User's Guide

Version 1

July, 1975

Model Code No. 50158

PREFACE

This manual describes the run-time characteristics of the Datapoint 5500 DATASHARE interpreter. It is meant as reference material to the features of the interpreter and not as a tutorial. For complete information on the DATABUS language, refer to the DATABUS language reference.

TABLE OF CONTENTS

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

DATASHARE permits the simultaneous execution of up to sixteen DATABUS programs, each dealing with its own remote Datapoint CRT terminal (a system option allows one of the programs to execute using the system console instead of a remote terminal which allows DATASHARE to be run without a multi-port adaptor). The 5500 DATASHARE interpreter runs under DOS 1.2, DOS.A, or DOS.B (the same object code will run under all systems), automatically configuring itself for the proper DOS (note that the interpreter will not run under DOS.C). By running under DOS, the interpreter takes advantage of all of the DOS's file handling characteristics in addition to providing indexed-sequential, random, and sequential file accessing, thus providing a powerful data entry and processing facility. In addition to file handling, full control is provided over either a line or servo printer. This configuration allows a flexible mix of remote, batch, and interactive processing all under the control of a high level language program, enabling the user to configure the system to best suit his data processing needs.

In addition, the DOS with its variety of utility and higher level language systems may be used alternately to DATASHARE, enabling processing of tasks not appropriate to the multiple terminal environment.

Using virtual memory techniques, 5500 DATASHARE provides each program with a 32,000 byte (minus data) area for executable statements. This, in combination with the ability of the compiler to accommodate over 3400 labels, enables the user to create and use programs of over one hundred pages (a very large high level language program). To provide rapid program execution, the data area for each program is maintained in main memory and not swapped. The system can be configured to run either a 3502 or 3601 on any one of up to sixteen ports with the data area being variably partitioned among them (the data area of any one port can be configured to be from 256 to 4096 bytes).

Any of the Datapoint printer systems may be connected to the DATASHARE configuration with printing being controlled from any of the ports. If the printer is busy with one port, another port trying to access the printer will wait until the first port releases the printer.

All program execution in DATASHARE occurs in the DATABUS language. Terminal command interpretation is handled in special ANSWER and MASTER programs (unique for each port) which also handle system security. These programs are provided with the system but may be compiled like any other Databus program, enabling the user to completely define his own terminal command and security system.

Program generation is performed under the DOS using the general purpose DOS editor and DOS DATASHARE compiler.

CHAPTER 2. SYSTEM GENERATION

2.1 Loading From Cassette

The 5500 DATASHARE interpreter system programs are contained on one cassette. The cassette is in the DMF (DOS Multiple File) format which includes a directory of the files on the tape. All that is necessary to load the system files to disk is to have the MIN program catalogued on the system and to give the DOS command:

```
MIN ;A
```

The MIN (Multiple IN) program will be activated and will display the date of creation of the tape, the file names in the tape directory, and each file name as the file is being loaded. If the file already exists on the disk, the MIN program will ask if it is to be overstored. The operator can decide to overstore the file or can tell it not to overstore the file in which case MIN will allow the file to be stored under a different name. Consult the MINMOUT USER'S GUIDE for further information on this procedure.

The files on the tape are DS35500/CMD, DS35500/OV1, DS355CON/CMD, DS355BAK/CMD, DS355BTD/CMD, and ROLL5500/SYS. The first two files are the interpreter proper, the third is the configuration command, the fourth is the rollout return, the fifth is the rollout return with time and data initialization, and the last is the rollout overlay (note that this is different from the rollout used in 2200 systems).

The interpreter system files can be re-named to any name desired as long as the command file and all the overlays have the same name. For example, if DS35500/CMD was re-named DS/CMD, then DS35500/OV1 would have to be re-named DS/OV1.

Note that whenever the configuration program is executed, it writes the configuration information into the 16th LRN of a file called DS355CON/CMD. Therefore, if the command is re-named, the DS355CON/CMD file will still appear when the system is configured. ROLL5500/SYS must not be re-named if rollouts are to function.

2.2 Port Configuration

The 5500 DATASHARE system may be configured to run with from one to sixteen ports. The system is configured by running the DS355CON program. This program will first display the current configuration (if one has been made) and then ask if the configuration is to be changed. If a negative response is given, control is returned to the DOS. Otherwise, the DS355CON program will run through a sequence of questions concering the number of ports, whether the console is to be the terminal for port one, whether the servo printer is to be used for the system printer, if port one is on the console whether the multi-port is to be bypassed altogether, whether the available space is to be divided evenly among the ports, and whether ROLLOUT is to be configured. If the space is to be evenly divided, the DS355CON program will display how much space is allocated to each port. If the space is not to be evenly divided, the DSCON program will request the amount of space to be allocated to each port. The amount of space will be rounded up to the nearest multiple of 256 bytes and will be limited to that which will leave at least 256 bytes for each of the remaining ports to be configured.

Note that the total amount of space available is the machine's memory space minus 16K (for the interpreter) minus 256 times the number of ports configured (system overhead associated with each port). No more than 4096 bytes will ever be assigned to any one port regardless of the amount of space available.

2.3 Necessary Programs

Before the 5500 DATASHARE system can be used, two more sets of programs must exist. These are called the ANSWER and MASTER programs and perform the tasks of dealing with the user when he initially signs onto the system and dealing with him when he is not running another DATASHARE program. Note that all execution in the DATASHARE system occurs in the high level language and since the user writes his own ANSWER and MASTER programs, he can determine how the system command language appears. The ANSWER and MASTER programming concepts are dealt with in Chapter 4.

If an ANSWER and MASTER program do not exist for a port, it will never become active even if it is configured into the system. The ANSWER and MASTER program must have the object names ANSWERn/DBC and MASTERn/DBC where n is the number of the port for which these are the ANSWER and MASTER programs (n = 1 through 16).

If DATASHARE initialization cannot find an answer program with a name of ANSWERn/DBC, it will then search for a program with the name ANSWER/DBC. Similarly, the "second choice" master program is named MASTER/DBC. In this way, all ports may share a single answer or master program, if desired.

If a multi-drive system is being used, it is generally a good idea to keep all necessary system utilities, the DATASHARE system files, and the DATASHARE object code files, as well as the ANSWER and MASTER program object files on drive 0 and to not remove the disk in drive 0 during normal system operation.

Other programs which should be on the system include the INDEX, REFORMAT, and SORT utilities (for the generation of index files), and the MINMOUT utilities.

CHAPTER 3. SYSTEM OPERATION

3.1 Bringing Up the System

If the 5500 DATASHARE interpreter system files are named DS/CMD and DS/OV1, then the DATASHARE system is brought up by entering the DOS command:

```
DS
```

This begins a series of operations the first of which is the display of the message:

```
DS335500 1.r - SYSTEM BEING INITIALIZED
```

where r is the revision number of the particular release. If the DATASHARE system has not been configured, the message:

```
* DS35500 HAS NOT BEEN CONFIGURED *
```

is displayed. If the configuration file cannot be found on the same drive the DS/CMD file is located on, the message:

```
* DS355CON/CMD MISSING ON DRIVE d *
```

is displayed where d is the logical drive number. If the overlay cannot be found on the same drive the DS/CMD file is located on, the message:

```
* DS/OV1 MISSING ON DRIVE d *
```

is displayed. Of course, if the DATASHARE 3 interpreter system files have been re-named, the names in the above messages would be changed accordingly. If any of the above messages is displayed, the machine will return to the DOS. If the initialization is completed successfully, the system displays the message:

```
OPERATOR, PLEASE DEPRESS THE KEYBOARD OR DISPLAY KEY.
```

This action will verify that an operator is present. A design objective was that the time and date be initialized by the operator when the system was brought up but that the system also be capable of bringing itself up in the case of power failure and unattended operation. If the keyboard or display key is not depressed within 30 seconds after the message is displayed, the machine will make a series of one second beeps in an effort to attract the attention of any operational personnel within the vicinity. If the keyboard or display key is not depressed after 30 seconds of beeping, the system assumes that it is being operated in an unattended mode and should start operation without the time and date being initialized. In this case, the time and date entries at the upper right of the console screen will be blank.

If the time and date are to be initialized, the operator must depress either the keyboard or display key. Upon doing this, the screen will be initialized with a message indicating the release number of the DATASHARE system being used, the number of ports configured for that system, and the digits one through eight running down the left side of the screen and nine through sixteen down the middle of the screen. These digits denote a line which is allocated for each physical port. The CHAIN statement displays on this line the name of the program being invoked. The program running for that port may also display on this line using the CONSOLE statement. These lines are useful for informing any operational personnel of the status of the system.

To initialize the time and date, the system will display the message TIME: in the upper right part of the screen. The operator should respond to this with a four digit number indicating the current clock value in hours and minutes (HHMM). Note that no colons should be entered and that a valid 24-hour clock value must be entered. If the value is not valid, the TIME: message will be repeated. Otherwise, the system will display the message DATE: to the right of the time value just entered. The operator should respond to this with a three digit number followed by a slash followed by a two digit number. The first number should be the current julian date (a number between 1 and 365 or, on leap years, 366) and the second number should be the last two digits of the current year. Note that the format mentioned must always be followed, with leading zeros used if necessary. If the julian date is not valid, the DATE: message will be repeated. Otherwise, the system will begin execution as denoted by the wall clock display running in the upper right part of the screen. The system will then look up all of the ANSWER and MASTER program names in the DOS directory and store their physical file and logical drive numbers away in a table. Ports requesting connection during this time will be connected but no response will be made until all of the programs have been initialized. Note that an asterisk just to the right of the port number on the console screen will be displayed if the Carrier Detect signal for that port is present.

If the system is configured to run port one on the console, an alternate form of bring up the system may be used. Instead of entering the simple DOS command DS to start the system, the operator can enter:

```
DS <program>
```

where <program> is the name of a DATASHARE object code file. If this action is taken, the file <program>/DBC will be used for the answer program for port one instead of ANSWER1/DBC. The master program for port one will still be MASTER1/DBC. Also, the operator will not be requested to depress the KEYBOARD or DISPLAY keys and to enter the time or date. The time and date will be initialized to 00:00 and 000/00 respectively and execution will start immediately. Note that the console screen is blanked just before execution is begun if port one is being run on the system console. If one wishes to simply bypass the time and date entry, he can enter the command:

```
DS ANSWER1
```

which will allow the normal answer program to be executed for port one but will eliminate the request for the time and date. This feature makes it possible to run DATASHARE from the CHAIN program. To return from DATASHARE to the chaining process, a ROLLOUT must be performed with the DOS command:

```
CHAIN/OV1
```

given which will cause the CHAIN command to pick up after the last command issued. Normally, if a single DATASHARE program is to be run in a batch mode (only one port) the program should be executed using the DATABUS system, since throughput is higher.

The DATASHARE interpreter system can be configured to automatically start execution when the DOS is brought up by the use of the AUTOKEY/CMD program. DATASHARE 3 looks in the DOS command line to determine the name of the command file so the overlay name can be determined (this is what allows the 5500 DATASHARE interpreter system files to be re-named). Because of this, the standard DOS AUTO program can not be used to directly cause automatic execution of the 5500 DATASHARE interpreter system.

3.2 Taking Down the System

The DATASHARE system maintains its files totally under the control of the DOS. The DOS normally may be halted at any time without detriment to the file structure. However, halting the system after a new file has been created or after a new segment has been allocated will leave that file with the maximum amount of space allocated to it. Proper closing of the file collapses the space allocated to only that used. Thus, to be sure all files are properly closed, the system should be halted when all ports are in their MASTER programs (the operator can tell from the console screen when a port is in its MASTER program).

3.3 Fatal Error Conditions

There are error conditions within the DOS which cannot be trapped. These errors invoke a DOS overlay called the ABORT overlay which reloads the DOS to insure the presence of the DSPLY$ routine, displays an error message in the standard DOS format, and then returns control to the DOS command interpreter. The DATASHARE foreground routines reside in an area which is overlayed by the DOS and, therefore, the normal abort message routine would cause havoc when it tried to load the DOS. For this reason, the DATASHARE system overlays the DOS in a critical place that allows it to trap the action of untrappable DOS errors and store a return instruction in location zero. This effectively disables any interrupt driven execution and allows the DOS to be loaded for the abort message display. Since the 5500 DATASHARE does not overlay any part of the DOS that is loaded with the bootstrap, return to the DOS is made after the abort message display.

CHAPTER 4. ANSWER AND MASTER CONCEPTS

There are two DATABUS programs which must exist for each port for that port to be active. The first is called the ANSWER program which deals with the user when he initially connects to the system (calls on the telephone or turns on his CRT). The second program is called the MASTER program which deals with the user whenever he is not executing the ANSWER program or an application program and is generally used to allow the user to select the next application program he wishes to execute. Note that both of these programs are written in DATABUS, enabling the user to tailor the command aspects of the DATASHARE system to his particular needs. Simple and complex examples of ANSWER and MASTER programs are shown in the appendices.

4.1 System Security

The ANSWER program allows the programmer to force the user to give some type of identification before he is allowed to use the system. Note that the INTERRUPT key on the terminal is ignored while execution is taking place between the time when the system first acknowledges the presence of a user at a given port and the first chain executed by the program for that port. This means that while the user is executing in the ANSWER program for a given port when he first signs onto the system, he may not escape around the identification request and get directly into the MASTER program by simply striking the INTERRUPT key. The INTERRUPT key is also inhibited whenever a CHAIN to the user's ANSWER program is detected. The ANSWER program may also be structured to enforce file access limitations depending upon the identification of the user.

4.2 System Convenience

The ANSWER program chains to the MASTER program which usually requests from the terminal operator the name of the program he wishes to execute. This name can be generated from information supplied by the terminal operator so, for example, the operator may enter the number of a form and the MASTER program will decide which program to execute for that form number. The DOS directory cannot be accessed by the MASTER program, implying that a file must be generated which contains the names of programs and files that are to be accessed if directory service or file access limitation is to be implemented. It is very much up to the author of the ANSWER and MASTER programs to provide any convenience facilities to the terminal user.

4.3 Sample Answer and Master Programs

Appendix A contains examples of both simple and complex ANSWER and MASTER programs. Each program is edited for entry of the appropriate port number in the variable PORTN and then compiled for the given port. This procedure (editing in the port number and then compiling into an object file with the port number in its name) must be followed for each port that is to be used in the system. If a DATASHARE object file for either the ANSWER or MASTER program does not exist for a given port, the port will simply not be activated when the system is brought up.

The simple ANSWER program displays on the terminal the number of the port (the ANSWER's program name will automatically be displayed on the console). The simple ANSWER program then requests an identification and checks it for validity against a very simple rule (the identification given must be exactly the word DATAPOINT). If the word matches (note the use of both the NOT EQUAL and LESS conditions for checking for an exact match), a STOP statement is executed which causes a chain to the MASTER program. Otherwise, an indication is given that the proper identification was not entered and another request for identification is made.

The simple MASTER program merely requests the name of a program to be executed. A CHAIN is executed to the name given and if a chain failure occurs an indication is given that the name does not exist in the DOS directory and another request for a program name is made. Note that both the ANSWER and MASTER programs are written without the use of cursor positioning in the KEYIN and DISPLAY statements to aid in Teletype terminal compatibility.

The MASTER program should not assume any common data areas since it can be entered due to a program trap or the INT key being struck. For this reason, if a common data area value is to be determined (such as the port number) this should be done in the MASTER program and not in the ANSWER program.

The complex ANSWER and MASTER programs perform tasks similar to those performed by the simple programs except that a number of convenience features are added to give the system the appearance of a more conventional time sharing system. Two files are associated with the more complex programs, the SYSFILE and the

DAYFILE (system and day files). The system file contains identification code information and a table associating a given identification code (user) with a given set of programs (user's directory). The system file also contains a record for each physical port (records zero through seven) which allows any executing program to determine which user identification is associated with the given physical port at any given time. A user identification number (an index into the rest of the file from which the actual symbolic user identification can be obtained), the time at sign on, and the date at sign on are recorded in this record. The remainder of the file contains four records for each user identified in the system. Each record is broken into ten ten-character fields. The first field of the first record is the identification code. The rest of the fields in the first record and the following three records contain program names associated with the given user identification. The list of program names is terminated by a space appearing in the first column of the name. The list of user identifications is terminated by a space appearing in the first column of a user identification.

The second file associated with the complex ANSWER and MASTER programs is called the day file. This file simply contains a set of records to be displayed at sign on time. This information is used to inform users of changes in the system or any other facts pertinent to the use of the system. Note that both of these files must exist before the complex ANSWER and MASTER programs can be used. The files can be created with DATASHARE if simple ANSWER and MASTER programs exist.

The complex ANSWER program determines the month and day of the month from the julian date. It detects if the date has not been initialized by noting that the julian date is zero (an invalid initialization value). After the date is displayed, a request is made for an identification code. The identification code list in the system file is then scanned for a match with the one supplied. If a match cannot be found, an indication is given to the user and the request for identification is repeated. Note that only three tries at identification are allowed in an effort to prevent unauthorized access to the system via the technique of trying identification codes until one is struck. After the third try, the response to the user does not change but he is not allowed access to the system even if he does then enter a valid identification and an alert message is displayed on the console to alert the operator that someone who apparently does not know an identification code is trying to access the system. If a valid identification is entered within three tries, the identification index into the system file, the date of sign on, and the time of sign on are written in the record in the system file corresponding to the physical port being used and execution is passed to the MASTER program via the STOP statement.

The complex MASTER program allows a number of commands as explained in the KEYIN statement under the label HELPI. This particular program does not limit program or file access to a given user to his programs only, but such a scheme could be implemented without much difficulty.

Note that when the ANSWER program is chained to it will execute until the first KEYIN, DISPLAY, or CONSOLE statement is executed. The ANSWER program is actually executed when the terminal disconnects from the system, not when it connects to it. If the time of connection and disconnection and total connection time are being kept in a file, the ANSWER program can note when a user disconnects from the system and log the total amount of time the user was connected as the first operations in the ANSWER program. Then the KEYIN statement requesting a new user identification can be issued which will cause execution to cease for that port. The log out function will be executed when the terminal disconnects from the system. When the terminal re-connects to the system the KEYIN statement will be satisfied when the operator at the terminal enters an identification code at which time the new user can then be logged on with the time being noted in the log file. Note that when the system is initialized, all ports will appear to be logging off (since all ANSWER programs are executed) but no corresponding log on time will be set. The program must handle this special case by allowing for log offs without corresponding log on times.

CHAPTER 5. PHYSICAL SYSTEM CHARACTERISTICS

5.1 Virtual Memory

To achieve a reasonable amount of program space for many simultaneous programs, DATASHARE employs a virtual memory technique. DATASHARE code is very compact, with very few bytes of instructions being capable of invoking a large amount of processor activity. Therefore, the rate at which DATASHARE program bytes are fetched is very low. Because of this low rate, the actual program code bytes can be kept in the randomly accessible disk buffers with very little effect on program execution speed. Another characteristic of DATASHARE code is that it is never modified. Because of this, program code need only be read in and never written back out to the disk.

A different story exists in the case of the program data, however. This data is accessed at a very high rate and must be in main memory to be effectively accessible by the DATASHARE interpreter. For this reason the program data for all programs is kept resident in main memory. This fact will be shown later to have further advantages in the case of port I/O.

To implement an effective virtual memory accessing algorithm, the program code is kept on the disk as 250 byte pages. Because the code is paged in blocks, the DATASHARE programmer can make his program run much more effeciently, in many cases, by forcing his code to cross as few page boundaries as possible. Each time a page boundary is crossed, a new page must be read in if it is not already in the buffer. The paging scheme used is purely demand with the least recently used page being destroyed to make space for the new page. Actually, in a lightly loaded system, a single program could get a number of pages all resident in the disk buffer memory at once and crossing a given page boundary would not cause a disk read, but any significant loading will cause this condition to cease. Therefore, the DATASHARE programmer can assume that each time he crosses a page boundary, a new read will occur. This read will cause delay in the execution of the program. This time is time that cannot be used by any other program since the disk is busy. By causing an excessive number of page boundary crossings, the programmer can easily cause his program to execute very slowly.

However, an instruction called TABPAGE exists in DATASHARE to aid the programmer in making his execution speed as high as possible. This instruction causes the location counter in the compiler to be incremented until it is at the start of the next page (nothing will be generated if the location counter is already at the start of a page). When this instruction is executed, it causes a GOTO to the start of the next page. By using this instruction, the programmer can cause logical parts of his program to contain as few page boundaries as possible. Another way to increase execution speed is to use in-line coding as much as possible, especially for short operations, instead of the subroutine calling feature if the subroutine is located in a page different from the calling location. This is economically feasible because of the large space available for each program (32K bytes).

5.2 Scheduling

To provide optimum response time, 5500 DATASHARE handles all port I/O using interrupt driven foreground routines, which means that data transfer between the terminal and the system can occur regardless of the computational task being handled by the background program at any given time. The foreground routines actually interpret the KEYIN, DISPLAY, and CONSOLE instructions, with the background interpretive code merely passing these instructions to the foreground through a circular buffer allocated for each port. Conventional systems use such a buffer to hold the actual characters transferred between the system and the terminal. However, DATASHARE uses this buffer to hold the interpretive code bytes, thus enabling many more bytes to be transferred than can actually be held in the buffer. For example, a DISPLAY statement may contain some quoted information and then a variable name. The variable name is represented by two bytes but the contents of the variable could be fifty bytes long, enabling two bytes of buffer space to invoke the transfer of fifty bytes to the terminal. This is made possible by the fact that all program data is resident in main memory which enables the foreground routine to be executing an I/O statement for a given port even though the background program for that port may not be swapped in at the time.

The foreground and background program for a given port always execute exclusively of each other to prevent conflicts over data values. When the background program executes a DISPLAY statement, the statement is stored in the buffer for the given port and then the background program is deactivated and the foreground program activated. When the foreground program has completely executed the I/O statement, it causes a high priority interrupt to the background, which deactivates the current program and activates the one which was executing the DISPLAY statement which caused the interrupt. One important consideration which must be taken into account by the DATASHARE programmer concerning port I/O is the fact that every time an I/O instruction is completed in the foreground, the background program is swapped in. If the programmer is not careful, he can cause the system to thrash (spend most of its time swapping background programs instead of doing useful work) by causing a high rate of I/O completion interrupts. An example would be using many separate DISPLAY statements instead of one long continued statement.

The above discussion concerns only port and console I/O. All disk I/O is performed under the DOS which is a background-only operation. This means that all DOS functions are non-interruptable and long directory searches (which can take up to several seconds with a multiple drive system) will cause the response to I/O completion interrupts to be delayed. Long DOS functions, however, occur infrequently and therefore can be ignored from an average response time calculation standpoint.

Printing under 5500 DATASHARE is performed in background and foreground. The background execution sets up a line image in a 132 position buffer and when vertical paper motion is necessary, this buffer is transferred to a 850 character circular buffer which is emptied by a simple foreground process. The background execution is suspended only if the 850 character buffer becomes full. Therefore, one can complete a number of print statements without being swapped out.

When the background program resumes execution due to the completion of a foreground I/O task, it is guaranteed a minimum amount of execution time. This prevents the system from spending all of its time swapping background tasks when the foreground I/O completion rate is high.

DATASHARE is capable of driving any serial terminal device which uses an ASCII character set. Use of devices without cursor positioning features, however, will restrict the programmer from using the cursor positioning facility in the KEYIN and DISPLAY statements. If the programmer does not use the cursor positioning feature, he will be able to write a program which is Teletype machine compatible. The *ES and *EL list controls send control characters that are ignored by a 35 ASR Teletype. However, the Cursor On character which is sent before each KEYIN variable entry request and the Cursor Off which is sent after the ENTER key is struck, are Tape On and Tape Off respectively on a 35 ASR

Teletype.

DATASHARE is also capable of dealing with 103 type datasets as well as hard wired connections and full duplex four wire 202 dataset connections. It handles all of the 103 handshaking involved and needs only the proper cable to work correctly. In fact, the 3500 or 3601 hard wire cable is connected in such a way as to make the 3500 or 3601 appear as a 103 data set, with power on causing ring detect and carrier detect to be sent to the DATASHARE system. The fact that a hard wire or dataset connection is employed at a given terminal cannot be differentiated by the DATASHARE programmer.

CHAPTER 6. PHYSICAL INSTALLATION

6.1 Main peripherals

The DATASHARE system requires a disk peripheral and maintains its entire file structure under the DOS. Note that drive zero must be kept on line at all times during system operation but any other drives may be put on or off line as the maintenance of the data base requires.

Besides the disk, the other required peripheral for the operation of the DATASHARE system is the 9460 Multiple Port Communications Interface. In the following discussions the mention of a 9460 will imply that a 9462 will work as well. As far as DATASHARE is concerned, the 9460 and 9462 are equivalent. These devices are capable of driving up to eight fully independent full duplex asynchronous lines at speeds ranging from 110 to 9600 baud. The DATASHARE system is not capable of output above 125 characters per second per port and normally uses 1200 baud for direct connection and four wire 202-type modem connections and uses 300 or 110 baud for 103-type modem connections. However, any speed may be strapped in the 9460 to achieve compatibility with specific terminals as the occasion may require. Note that all ports are operated by the DATASHARE system in full duplex mode only. Since up to 16 ports can be used with 5500 DATASHARE, up to two 9460's can be attached. If this is the case, no other power drawing device can be connected to the 5500 due to power supply limitations. Ports 1 through 8 are connected to the first (standard I/O address of 0151 octal) 9460. Ports 9 through 16 are connected to the second 9460 whose I/O address must be strapped for 055 octal.

As in any 5500 installation, a 9420 parallel interface with an address of 0303 may be connected to drive a special output device but that device mus be capable of handling the output that would normally be given to an ASCII printer.

6.2 Terminal connections

In general, a terminal may be connected to the DATASHARE system in one of three ways: direct hardwire, 103-type modem, and 202-type modem. The following table shows the pin assignments on the 25-pin connector for the 9460 individual port, the 3502 CRT terminal, and a 103 or 202 type modem:

PIN	9460	3502	103/202
1	-	PROT GROUND	PROT GROUND
2	DATA OUT	DATA OUT	DATA IN
3	DATA IN	DATA IN	DATA OUT
4	REQ TO SEND	-	REQ TO SEND (202)
5	CLR TO SEND	-	CLR TO SEND
6	-	-	DATA SET READY
7	SIG GROUND	SIG GROUND	SIG GROUND
8	CARRIER DET	-	CARRIER DET
-			
20	DATA TERM RDY	DATA TERM RDY	DATA TERM RDY
-			
22	RING DETECT	-	RING DETECT

The DATASHARE system goes through the following handshaking procedure when a connection is established:

1. Clear Data Terminal Ready and Request To Send
2. Wait for Ring Detection
3. Set Data Terminal Ready and Request To Send
4. Wait up to 10 seconds for Carrier Detect
5. Go to step 1 if time out in step 4
6. Wait one second and then start the ANSWER program

This procedure will work with any of the three types of connections if the proper cable is used.

<u>DIRECT</u>

Basically, the direct connection cable swaps the data wires (pins 2 and 3) and connects Carrier and Ring Detect on one end to Data Terminal Ready on the other as shown in the following table:

9460 TO 3502 CABLE CONNECTIONS

9460	3502	3601
2	3	4
3	2	2
7	7	3
8 and 22	20	12

Note that this arrangement requires only five wires in the cable (four if the optional wire is not used). If the cable is to be made more than several hundred feet long, each of the two signal wires (the ones connecting to pins 2 and 3) should be twisted separately with a ground wire (no other shielding is necessary). Direct connections up to one thousand feet may be made if the above precautions are followed.

The 3502/3601 sets Data Terminal Ready whenever it is running. With the above cable connected, this will cause ringing and carrier to be presented to the 9460. This has the effect of causing the ANSWER program to be executed whenever power is applied to the 3502/3601.

103-TYPE MODEM

The 9460 can be connected to a 103-type modem with a one to one cable (e.g., a pin at one end is connected to a pin of the same number at the other end). Only pins 2, 3, 7, 8, 20, and 22 need to be connected but having all pins connected will also work (this being the simplest to describe to someone at a distance!). Note that 103 and 113B modems have similar pin connections.

9460 TO 103-TYPE MODEM CONNECTIONS

9460	103-TYPE MODEM
2	2
3	3
7	7
8	8
20	20
22	22

If one is calling a 103-type modem over a dial-up network, he will hear the telephone answered very shortly after it starts ringing (should take one or two rings at most). If the telephone is not answered within that amount of time, the caller either has the wrong number or the DATASHARE system is not functional. In any case, the caller may as well hang up (letting the phone ring for a long time can be very irritating at the other end). If the telephone is answered, the caller will hear the carrier from the modem connected to the 9460 which is his signal to either depress the DATA key on his modem or put the telephone handset in the data coupler (if he is using one). The DATASHARE system gives the caller ten (10) seconds to perform the necessary action to cause a carrier to be returned from his modem. If all is satisfactorily completed, one more second will pass and then the ANSWER program will begin execution. If all is not satisfactorily completed, the DATASHARE system will hang up the telephone at its end and go back to waiting for ringing to occur. Note that since the DATASHARE system does wait up to ten seconds for a satisfactory connection, if one dials the system and hangs up as soon as the telephone is answered, he will have to wait ten seconds before he can dial the same telephone again. Also note that the DATASHARE system will disconnect as soon as it loses the Carrier Detect signal from the modem. This means that disconnection will occur even if the carrier is broken only for a very short time.

202-TYPE MODEM

The DATASHARE system requires a full duplex connection to its terminals. A 202-type modem can be used in this fashion only if it is connected via a four-wire circuit. This means that one signal path must exit for data flow in one direction and a separate data path must exit for data flow in the other direction. This implies that a point-to-point connection is made between the modems (the switched telephone network cannot support four-wire connections). In this application, the 202 modem must be strapped for use in four-wire mode.

The connecting cable between the 9460 and 202 modem is similar to the one for connection to a 103-type modem except that, since 202's used in point-to-point four-wire service do not use ringing, the carrier detection signal from the 202 must be connected to both the carrier detection and ring detection inputs on the 9460.

9460 TO 202 MODEM CONNECTIONS

9460	202 MODEM
2	2
3	3
4	4
7	7
8 and 22	8
20	20

When Data Terminal Ready is supplied by the terminal device to the remote 202 modem, that modem will turn on its carrier. This carrier will cause the modem connected to the 9460 to turn on its carrier detect signal which will present ring detection and carrier detection to the DATASHARE system. The system will proceed to set its Data Terminal Ready signal which will cause the 202 modem to turn on its carrier and complete the connection. One second later the ANSWER program will begin execution. Thus, operation over a 202 modem connection will appear similar to direct connection operation.

Remote modems are connected to Datapoint 3000 series terminals via a standard modem cable supplied with the terminal. This cable provides the required Data Terminal Ready signal to cause the operational characteristics described above.

6.3 Port speed selection

The 9460 Multiple Port Communications Adaptor is software programmable to transmit and receive from five to eight information bits with either one or two stop bits. However, the DATASHARE system always uses eight information bits and sends two stop bits (it will receive signals with only one stop bit). The speed of each port may be set independently to a variety of speeds, depending on field programmable hardwire straps.

There are three clock buses within the 9460, limiting the total number of different speeds used at any one time to three. Each of these buses can be connected to one of two crystal controlled time bases. Each time base is connected to a binary dividing chain, giving speeds selectable in powers of two. The standard crystals supplied provide multiples of 110 and 300 baud. The baud rate of a bus is set by strapping from a baud rate source pin to a baud rate bus input pin. Each bus has eight baud rate output points. The baud rate of a channel is set by strapping from a baud rate bus output point to the channel baud rate input pin. The following table gives the respective pin numbers as found on the silk screening on the printed circuit card in the 9460:

BAUD RATE SOURCE

Baud rate	Pin
300	E29
600	E28
1200	E27
2400	E23
4800	E22
9600	E21
110	E33
220	E32
440	E31
880	E30
1760	E26
3520	E25
7040	E24

BAUD RATE BUS

Bus	Input	Output
1	E34	E37
2	E35	E38
3	E36	E39

CHANNEL BAUD RATE INPUT

Channel	Input
1	E13
2	E14
3	E15
4	E16
5	E17
6	E18
7	E19
8	E20

A typical installation may use baud rates of 110 for teletype machines (remote or local), 300 for remote 3502/3601 terminals using 103-type modems, and 1200 for remote 3502 terminals using 202-type modems. For this installation, one may connect bus 1 for 110 baud, bus 2 for 300 baud, and bus 3 for 1200 baud as shown in the following table.

E34 to E33	make bus 1 110 baud
E35 to E29	make bus 2 300 baud
E36 to E27	make bus 3 1200 baud

Now, if channels 1 through 3 are to be 300 baud, channels 4 through 7 1200 baud, and channel 8 110 baud, the following connections would be made:

E38 to E13, E14, E15	make ch 1-3 300 baud
E39 to E16, E17, E18, E19	make ch 4-7 1200 baud
E37 to E20	make ch 8 110 baud

Port speeds other than multiples of 110 or 300 baud can be accommodated by changing the crystal frequencies. Selection of the proper crystal should be aided by the Datapoint engineering staff.

6.4 Non-3502/3601 terminal devices

Terminals other than the Datapoint 3502/3601 can be connected effectively to the DATASHARE system. The major advantage of the 3502 is that its cursor can be positioned directly by the issuance of a three character sequence. This allows the usage of the cursor positioning list controls in the DISPLAY and KEYIN statements and greatly enhances the speed of form displays.

Terminals such as the Teletype 33 and 35 KSR or ASR may be connected either hardwire or over modem connections. In addition, conventional CRT terminals such as the Datapoint 3300 (for 300 or 1200 baud) or Datapoint 3000 (for 300 baud only) may be connected. All Datapoint 3000 series terminals use identical cable configurations for a given type of installation. The key to making a cable for a given device is to insure that both Carrier and Ring Detect on the 9460 are connected to a wire that is set when the connection is to be established and is cleared when the connection is to be broken.

APPENDIX A. PROGRAM EXAMPLES

Simple ANSWER Program

```
.
. SIMPLE ANSWER PROGRAM
.
PORTN   FORM     "4"
IDCODE  DIM      9
ID      INIT     "DATAPOINT"
.
        DISPLAY  *ES,"D A T A S H A R E   PORT ",PORTN," ON LINE"
LOOP    KEYIN    "ID: ",IDCODE
        MATCH    ID TO IDCODE
        GOTO     BADID IF NOT EQUAL
        GOTO     BADID IF LESS
        MATCH    IDCODE TO ID
        GOTO     BADID IF LESS
        STOP
BADID   DISPLAY  "*** INVALID ID ***"
        GOTO     LOOP
```

Simple MASTER Program

```
.
. SIMPLE MASTER PROGRAM
.
PORTN   FORM     "4"
FILNAM  DIM      8
.
        RELEASE
LOOP    KEYIN    *N,*EL,"PROGRAM NAME: ",FILNAM
        TRAP     NONAME IF CFAIL
        CHAIN    FILNAM
NONAME  DISPLAY  "*** NO SUCH PROGRAM ***"
        GOTO     LOOP
```

Complex ANSWER Program

```
.
. DATASHARE ANSWER PROGRAM
.
SYSFILE FILE                          FILE DECLARATION
DAYFILE FILE                          FILE DECLARATION
PORTN   FORM    "3"                   THE NUMBER OF THIS PORT
DATE    DIM     18                    TODAY'S DATE IN MONTH, DAY, YEAR
IDCODE  DIM     10
IDCTR   FORM    "3"
TIMEON  DIM     8
NFEB    FORM    "29"
RN      FORM    "000"
TIME    INIT    "00:00:00"
DAY     INIT    "000"
YEAR    INIT    "00"
NDAY1   FORM    3
NDAY2   FORM    3
NYEAR1  FORM    2
NYEAR2  FORM    2
LINE    DIM     100
.
        DISPLAY *ES,*N,"D A T A S H A R E    PORT ",PORTN;
        OPEN    SYSFILE,"SYSFILE"
START0  CLOCK   DAY TO DAY
        MOVE    DAY TO NDAY1
        CLOCK   TIME TO TIME
        CLOCK   YEAR TO YEAR
        MOVE    NDAY1 TO NDAY1
        GOTO    NODATE IF ZERO
        MOVE    YEAR TO NYEAR1
        MOVE    NYEAR1 TO NYEAR2
        DIV     "4" INTO NYEAR1
        MULT    "4" BY NYEAR1
        COMPARE NYEAR1 TO NYEAR2
        GOTO    LEAP IF EQUAL
        MOVE    "28" TO NFEB
LEAP    SUB     "31" FROM NDAY1
        GOTO    JAN IF LESS
        GOTO    JAN IF EQUAL
        SUB     NFEB FROM NDAY1
        GOTO    FEB IF LESS
        GOTO    FEB IF EQUAL
        SUB     "31" FROM NDAY1
        GOTO    MAR IF LESS
        GOTO    MAR IF EQUAL
```

```
         SUB      "30" FROM NDAY1
         GOTO     APR IF LESS
         GOTO     APR IF EQUAL
         SUB      "31" FROM NDAY1
         GOTO     MAY IF LESS
         GOTO     MAY IF EQUAL
         SUB      "30" FROM NDAY1
         GOTO     JUN IF LESS
         GOTO     JUN IF EQUAL
         SUB      "31" FROM NDAY1
         GOTO     JUL IF LESS
         GOTO     JUL IF EQUAL
         SUB      "31" FROM NDAY1
         GOTO     AUG IF LESS
         GOTO     AUG IF EQUAL
         SUB      "30" FROM NDAY1
         GOTO     SEP IF LESS
         GOTO     SEP IF EQUAL
         SUB      "31" FROM NDAY1
         GOTO     OCT IF LESS
         GOTO     OCT IF EQUAL
         SUB      "30" FROM NDAY1
         GOTO     NOV IF LESS
         GOTO     NOV IF EQUAL
         MOVE     "DECEMBER" TO DATE
         GOTO     START1
.
NOV      ADD      "30" TO NDAY1
         MOVE     "NOVEMBER" TO DATE
         GOTO     START1
.
OCT      ADD      "31" TO NDAY1
         MOVE     "OCTOBER" TO DATE
         GOTO     START1
.
SEP      ADD      "30" TO NDAY1
         MOVE     "SEPTEMBER" TO DATE
         GOTO     START1
.
AUG      ADD      "31" TO NDAY1
         MOVE     "AUGUST" TO DATE
         GOTO     START1
.
JUL      ADD      "31" TO NDAY1
         MOVE     "JULY" TO DATE
         GOTO     START1
.
```

```
JUN      ADD      "30" TO NDAY1
         MOVE     "JUNE" TO DATE
         GOTO     START1
.
MAY      ADD      "31" TO NDAY1
         MOVE     "MAY" TO DATE
         GOTO     START1
.
APR      ADD      "30" TO NDAY1
         MOVE     "APRIL" TO DATE
         GOTO     START1
.
MAR      ADD      "31" TO NDAY1
         MOVE     "MARCH" TO DATE
         GOTO     START1
.
FEB      ADD      NFEB TO NDAY1
         MOVE     "FEBRUARY" TO DATE
         GOTO     START1
.
JAN      ADD      "31" TO NDAY1
         MOVE     "JANUARY" TO DATE
.
START1   ENDSET   DATE
         MOVE     NDAY1 TO DAY
         COMPARE  "10" TO NDAY1
         GOTO     START2 IF NOT LESS
         BUMP     DAY
START2   APPEND   DAY TO DATE
         APPEND   ", 19" TO DATE
         APPEND   YEAR TO DATE
         RESET    DATE
         DISPLAY  *+," ON LINE AT ",TIME," ON ",DATE
         GOTO     DATEOK
.
NODATE   DISPLAY  " ON LINE   ";
         BEEP
         DISPLAY  "*** DATE NOT INITIALIZED ***"
DATEOK   DISPLAY  " "
         TRAP     LOOP1 IF IO
         OPEN     DAYFILE,"DAYFILE"
         MOVE     "0" TO RN
LOOP0    READ     DAYFILE,RN;LINE
         CMATCH   "9" TO LINE
         GOTO     LOOP1 IF EQUAL
         RESET    LINE TO 72
LOOP0A   BUMP     LINE BY -1
```

```
         GOTO      LOOP0B IF EOS
         CMATCH    " " TO LINE
         GOTO      LOOP0A IF EQUAL
LOOP0B   LENSET    LINE
         RESET     LINE
         DISPLAY   *+,LINE
         ADD       "1" TO RN
         GOTO      LOOP0
.
LOOP1    KEYIN     *EL,"PLEASE LOG IN: ",*N,IDCODE:
                   *C,"**********",*C,"OOOOOOOOOO"
         CLOCK     TIME TO TIMEON
         CONSOLE   *P15:1,*EL,"ID: ",IDCODE,"  TIME ON: ",TIMEON
         MOVE      IDCTR TO IDCTR
         GOTO      KABOOM IF ZERO
         MOVE      EIGHT TO RN
LOOP2    READ      SYSFILE,RN;LINE
         CMATCH    " " TO LINE
         GOTO      IDFAIL IF EQUAL
LOOP3    CMATCH    IDCODE TO LINE
         GOTO      NEXTID IF NOT EQUAL
         BUMP      LINE
         BUMP      IDCODE
         GOTO      LOOP3 IF NOT EOS
         CMATCH    " " TO LINE
         GOTO      NEXTID IF NOT EQUAL
         SUB       "1" FROM PORTN
         WRITE     SYSFILE,PORTN;RN,DATE,TIME
         CLOSE     SYSFILE
         STOP
.
NEXTID   ADD       "4" TO RN
         GOTO      LOOP2
.
IDFAIL   BEEP
         DISPLAY   "*** INVALID ID ***"
         SUB       "1" FROM IDCTR
         GOTO      LOOP1
.
KABOOM   CONSOLE   *P60:1,*EL,"ID OVERRUN"
         BEEP
         DISPLAY   "*** INVALID ID ***"
         GOTO      LOOP1
```

Complex MASTER Program

```
.
. DATASHARE MASTER PROGRAM
.
SYSFILE FILE                          FILE DECLARATION
PORTN   FORM     "3"                   THE NUMBER OF THIS PORT
ANSWER  INIT     "ANSWERX "
LINE    DIM      100
LINITM  DIM      10
RN      FORM     "000"
RNX     FORM     "000"
ONE     FORM     "1"
FOUR    FORM     "4"
EIGHT   FORM     "8"
NINE    FORM     "9"
TEN     FORM     "10"
COUNT   FORM     "00"
CMDLIN  DIM      20
HELP    INIT     "HELP"
HELLO   INIT     "HELLO"
CAT     INIT     "CAT"
RUN     INIT     "RUN"
TIME    INIT     "TIME"
DATE    INIT     "DATE"
ONLINE  INIT     "ONLINE"
PORT    INIT     "PORT"
BYE     INIT     "BYE"
.
        RELEASE
        DISPLAY  *ES
        OPEN     SYSFILE,"SYSFILE"
        SUB      ONE FROM PORTN
        READ     SYSFILE,PORTN;RN
CMDREQ  KEYIN    *ES,*N,"READY",*N,CMDLIN
TRYAGN  MATCH    HELP TO CMDLIN
        GOTO     HELPI IF EQUAL
        MATCH    HELLO TO CMDLIN
        GOTO     HELLOI IF EQUAL
        MATCH    CAT TO CMDLIN
        GOTO     CATI IF EQUAL
        MATCH    PORT TO CMDLIN
        GOTO     PORTI IF EQUAL
        MATCH    TIME TO CMDLIN
        GOTO     TIMEI IF EQUAL
        MATCH    DATE TO CMDLIN
        GOTO     DATEI IF EQUAL
        MATCH    ONLINE TO CMDLIN
        GOTO     ONLI IF EQUAL
        MATCH    BYE TO CMDLIN
        GOTO     BYEI IF EQUAL
        MATCH    RUN TO CMDLIN
        GOTO     TRYNAM IF NOT EQUAL
        CALL     GETNAM
TRYNAM  TRAP     CFAIL IF CFAIL
        CLOSE    SYSFILE
        CHAIN    CMDLIN
.
CFAIL   OPEN     SYSFILE,"SYSFILE"
        KEYIN    *N,"WHAT?",*N,CMDLIN
        GOTO     TRYAGN
.
GETNAM  BUMP     CMDLIN
        RETURN   IF EOS
        CMATCH   "0" TO CMDLIN
        GOTO     GETEXX IF LESS
        CMATCH   ":" TO CMDLIN
        GOTO     GETNAM IF LESS
        CMATCH   "A" TO CMDLIN
        GOTO     GETEXX IF LESS
        CMATCH   "[" TO CMDLIN
        GOTO     GETNAM IF LESS
GETEXX  BUMP     CMDLIN
        RETURN
.
HELPI   KEYIN    *ES,*N:
                 "ENTER: HELLO-<ID> TO SIGN ON AS ANOTHER USER",*N:
                 "       HELP       TO GET THIS INFORMATION",*N:
                 "       CAT        TO GET A LIST OF PROGRAMS",*N:
                 "       TIME       TO GET THE CURRENT TIME",*N:
                 "       DATE       TO GET THE DATE AT LOGON",*N:
                 "       ONLINE     TO GET THE TIME AT LOGON",*N:
                 "       PORT       TO GET THE PORT BEING USED",*N:
                 "       RUN-<NAME> TO RUN A PROGRAM",*N:
                 "    OR  <NAME>      TO RUN A PROGRAM",*N,*N:
                 "READY",*N,CMDLIN,*ES
        GOTO     TRYAGN
.
HELLOI  CALL     GETNAM
        MOVE     CMDLIN TO LINITM
        MOVE     EIGHT TO RNX
HELLO2  READ     SYSFILE,RNX;LINE
        CMATCH   " " TO LINE
        GOTO     IDFAIL IF EQUAL
```

```
HELLO3  CMATCH   LINITM TO LINE
        GOTO     NEXTID IF NOT EQUAL
        BUMP     LINE
        BUMP     LINITM
        GOTO     HELLO3 IF NOT EOS
        CMATCH   " " TO LINE
        GOTO     NEXTID IF NOT EQUAL
        READ     SYSFILE,PORTN;RN,LINE
        WRITE    SYSFILE,PORTN;RNX,LINE
        MOVE     RNX TO RN
        GOTO     CMDREQ
.
NEXTID  ADD      "4" TO RNX
        GOTO     HELLO2
.
IDFAIL  BEEP
        KEYIN    "*** INVALID ID ***",*N,"READY",*N,CMDLIN,*ES
        GOTO     TRYAGN
.
CATI    DISPLAY *ES,*N,"CATALOG: ",*N
        MOVE     RN TO RNX
        READ     SYSFILE,RNX;LINE
        RESET    LINE TO 11
        MOVE     "9" TO COUNT
        GOTO     CATR1
CATR    READ     SYSFILE,RNX;LINE
        MOVE     TEN TO COUNT
CATR1   RESET    LINITM TO 99
        LENSET   LINITM
        RESET    LINITM
        CMATCH   " " TO LINE
        GOTO     CATR4 IF EQUAL
CATR3   CMOVE    LINE TO LINITM
        BUMP     LINE
        BUMP     LINITM
        GOTO     CATR3 IF NOT EOS
        GOTO     CATRB
CATRA   BUMP     LINITM BY -1
CATRB   CMATCH   LINITM TO " "
        GOTO     CATRA IF EQUAL
        LENSET   LINITM
        RESET    LINITM
        DISPLAY  *+,LINITM
        SUB      ONE FROM COUNT
        GOTO     CATR1 IF NOT ZERO
        ADD      ONE TO RNX
        GOTO     CATR
```

```
.
PORTI    ADD       ONE TO PORTN
         DISPLAY   "YOU ARE ON PORT ",PORTN;
         SUB       ONE FROM PORTN
CATR4    KEYIN     *N,"READY",*N,CMDLIN,*ES
         GOTO      TRYAGN
.
TIMEI    CLOCK     TIME TO LINE
         DISPLAY   *+,"THE TIME IS ",LINE;
         GOTO      CATR4
.
DATEI    READ      SYSFILE,PORTN;LINE
         RESET     LINE TO 21
         LENSET    LINE
         RESET     LINE TO 4
         MOVE      LINE TO CMDLIN
         CMATCH    CMDLIN TO " "
         GOTO      DATEIN IF EQUAL
         DISPLAY   *+,"THE DATE AT LOG IN WAS ",CMDLIN;
         GOTO      CATR4
DATEIN   DISPLAY   "*** DATE NOT INITIALZIED ***";
         GOTO      CATR4
.
ONLI     READ      SYSFILE,PORTN;LINE
         RESET     LINE TO 29
         LENSET    LINE
         RESET     LINE TO 22
         MOVE      LINE TO CMDLIN
         DISPLAY   *+,"THE TIME AT LOG IN WAS ",CMDLIN;
         GOTO      CATR4
.
BYEI     CLOCK     TIME TO LINE
         DISPLAY   *+,"LOGGED OFF AT ",LINE
BYEE     KEYIN     CMDLIN
         RESET     ANSWER TO 6
         ADD       ONE TO PORTN
         MOVE      PORTN TO CMDLIN
         SUB       ONE FROM PORTN
         APPEND    CMDLIN TO ANSWER
         RESET     ANSWER
         TRAP      AFAIL IF CFAIL
         CHAIN     ANSWER
AFAIL    GOTO      BYEE
```

APPENDIX B. ERROR CODES

If an event occurs and the trap corresponding to that event has not been set, the message:

```
* ERROR * LLLLL X *        or
* ERROR * LLLLL X * Q
```

appears on the console display. The first form appears for all traps except I/O traps. In the event of an I/O trap, a qualification letter is given where a "Q" is shown in the example (explained below under the "I/O" trap). The LLLLL is the current value of the program counter and the X is an error letter. In most cases, LLLLL points to the instruction following the one that caused the problem. However, in certain I/O errors, LLLLL will point after the list item where the problem occurred. The following error letters can appear:

A - interruptions already prevented
B - illegal operation code
C - chain failure
F - record format error
I - I/O error
O - object code read failure
P - parity failure
R - record number out of range
U - call stack underflow or overflow

Note that A, B, O, and U errors cannot be trapped. The B error will only show up if somehow an invalid object file is executed or if the system is failing. The U error will happen if the programmer forgets to perform a call or in some other fashion manages to execute a RETURN instruction without a corresponding CALL having been previously executed, or calls are nested more than eight levels deep. The A error will happen if a PI instruction is executed while interrupts are currently prevented. The O error will happen if the object file is positioned out of range, has a parity fault, or if its drive goes off line.

The events that may be trapped are shown below. The capitalized name is the one used in the TRAP statement.

PARITY	- disk CRC error during READ or disk CRC error during write verification (the DOS retries an operation up to 5 times to get a good CRC before giving up and causing this event).
RANGE	- record number out of range (an access was made that was off the physical end of the file, a record was read which was never written, or a WRITAB was used on record which was never written)
FORMAT	- data being read into a numeric variable was not all digits and decimal point and minus sign, or decimal point in input does not agree with the decimal point in FORM, or data from disk has a negative multi-punch but no room for a minus sign in FORM, or write specified multi-punch and the last item of the field is a decimal point. The operation stops with the item in error and the statement is aborted.
CFAIL	- the specified program was not in the DOS directory or a ROLLOUT was attempted with one of the necessary system files missing, or a program containing compile-time errors was loaded.
IO	- Error during I/O statement. Either a programming error or disk failure can cause this TRAP.

APPENDIX C. INTERPRETER I/O TRAP CODES

A - an access sequentially by key was attempted before any indexed sequential access was made using the logical file.
B - the READ mechanism ran off the end of a sector without encountering a physical end of record character (003).
C - an operation on a closed logical file was attempted.
D - a non-READ non-DELETE indexed sequential operation was attempted where the specified key already exists in the index.
E - an EOF mark without at least four zero's was encountered.
I - the index file specified in an OPEN statement does not exist on the specified drive(s).
J - the index file found by the OPEN statement does not reside in the correct physical location on the disk (index files may never be moved, they must always be re-created).
K - a null key was supplied in an operation where the key may not be null.
M - the data file specified in the OPEN statement does not exist on the specified drive(s).
N - the data file name specified in the OPEN or PREPARE statement was null.
O - the index file name specified in the OPEN statement was null.
P - the file specified in the PREPARE statement had some type of DOS protection (either write, delete, or both).
T - the tab value in the READ or WRITAB statement was off the end of the sector.
U - an EOF mark was encountered while a record was being deleted in the indexed sequential file.
W - an index file pointer sector could not be read.
X - an index file header sector could not be read.
Y - the R.I.B. of the data file pointed to by the index file could not be read. (VWXY errors can be caused by parity errors, the drive being switched off line, or the disk cartridge being swapped with another while an operation is taking place.)