DATASHARE VI INTERPRETER DS6

User's Guide

Version 1

October, 1980

Document No. 50536

DATAPOINT

DATASHARE VI INTERPRETER
DS6

User's Guide

Version 1

October 1980

Document No. 50536

NOTICE

Datapoint strongly recommends that its customers use Datapoint Customer supplies. These disks, diskettes, cassettes, ribbons and other products are certified by Datapoint to meet all Datapoint Hardware specifications for consistent optimum performance.

Copyright © 1980 Datapoint Corporation. All Rights Reserved.

PREFACE

DATASHARE VI is a high-level, multi-user, interactive program which interprets Datapoint's DATABUS language. It operates on any of Datapoint's 6600, 6010, 6020, 6040, 5500, 5000, 1170, 1810, 1820, 3810, or 3820 processors with a minimum of 32K of user memory; these processors may utilize either their local disks or diskettes, or may be operated under ARC (the 3810/3820 must, of course, be operated under ARC). For simplicity, this manual will refer to file I/O operations as "disk" operations, while, in fact, the file I/O may take place on diskettes or on MOUNTed ARC packs. DATASHARE VI is capable of printer I/O and associative-indexed, indexed-sequential, random-access, and sequential file I/O.

The maximum number of 3360/3600/3670 terminals supported by DATASHARE VI is 24 terminals on a 6600/6020/6040 processor, 16 on a 6010/5500/5000 processor, and four on an 1170/1810/1820/3810/3820 processor (except for an 1810/1820 processor utilizing a 9320 disk internal multiport adaptor, where the limit is nine terminals with port-1-on-console). The configurator may limit the number of terminals configurable to fewer than the maximum because of memory size or aggregate baud rate considerations, but in no case will greater than the maximum for any given processor be allowed. Any 3360/3600/3670 terminals

to be used must be connected to a 9462 or a 9320 Multiport Communications Adaptor; any Remote Slave ports to be used must communicate through a 9402 or 9481 Communications Adaptor. If more than two 9462 Adaptors are required, or if the 9462 or 9402 Adaptors are connected to an 1800 or 3800 type processor, a 9022 Auxiliary Power Supply must be used to power them; the 9481 or 9320 adaptor, if used, is self-powered and needs no Auxiliary Power Supply. In addition, the processor's console and keyboard may be used in place of port 1.

DATASHARE VI allows the use of the function keys F1 through F5 and screen highlighting on the 1800/3800 consoles.

This manual describes the run-time characteristics of the Datapoint DATASHARE VI Interpreter. It is meant to be used as reference material for features of the Interpreter and not as a tutorial. For complete information on the DATABUS language, refer to the DBCMPLUS Compiler User's Guide (Model Code # 50321). For specific changes and requirements of a particular DATASHARE VI release, please refer to the appropriate release form.

For the sake of simplicity this manual refers to 3502 terminals as 3360's, and 3601/8200 terminals as 3600's. DATASHARE VI handles 8200 terminals as the functional equivalent of 3600's.

Ports using 8200's should be configured as 3600's.

TABLE OF CONTENTS

CHAPTER 1. DIFFERENCES BETWEEN DS6 VERSION 1 AND DS5 VERSION 3

1.1 Interpreter Replacement

The DATASHARE VI version 1 Interpreter replaces the DATASHARE V version 3 Interpreter in every aspect of DATASHARE operations.

1.2 Supported Features

DATASHARE VI version 1 supports the following options on the following processors:

Processor	No. Ports	Max. BAUD	Multilink	Remotes
6600	24	19200	YES	YES
6010	16	19200	YES	YES
6020	24	19200	YES	YES
6040	24	19200	YES	YES
5500	16	19200	YES	YES
5000	16	19200	YES	YES
1170	4	19200	YES	YES
1810	4 (9*)	4800	YES	YES
1820	4 (9*)	4800	YES	YES
3810	4	4800	YES	YES
3820	4	4800	YES	YES

* Up to nine ports are supported using two 9320 Disk Internal Multiport Adaptors and port-1-on-console.

Note: The number of ports and/or features configurable may be limited to less than the number of ports and/or features given above if, after feature configuration, insufficient memory remains for every port's minimum User Data Area size. The total number of ports available in a given configuration may also be reduced if insufficient aggregate baud rate is available to handle the desired configuration. (See table above for aggregate baud rate limitations.) In addition, the DATASHARE configurator takes no responsibility for the speed of operation of any Interpreter configurable; the general rule is if a configuration fits into the memory available, it will be allowed.

1.3 Changes to Existing Features

The following features of DATASHARE have been enhanced in this version:

UDA Size. Maximum UDA size available to each port has been increased from 7.5K to 15.5K. This maximum may be lower for SLAVE and POLL ports due to system overhead.

ACALL Size. Previous versions of DATASHARE allowed a maximum, in most configurations, of 4K for both ACALL and MULTILINK together. ACALL can now be allocated up to 4K without impacting MULTILINK allocation. This maximum may be reduced on certain configurations if insufficient memory is available.

MULTILINK Size. The MULTILINK area allocation has been increased to a maximum of 8K. This area can be allocated without impacting the ACALL allocation. This maximum may be reduced on certain configurations if insufficient memory is available.

OPEN verb. When opening an IFILE or AFILE, DATASHARE VI will search for the text file first on the same drive as the index file. If the text file can not be found on the same drive, an all drive search will then be performed.

CLOSE verb. CLOSE can now be used on shared files if a FILEPI instruction has been performed on the file while the file was open. If the file was FILEPI'd, the Interpreter will not deallocate any newly allocated space. See chapter 8 for more details.

WRITE -2. The conditions under which a write pending buffer are written to disk have changed. These buffers are only written to disk upon execution of a ROLLOUT or SHUTDOWN instruction, an explicit CLOSE for the file, or if the virtual storage buffer containing the write pending sector becomes the least-recently-used buffer and is needed for some other use. Write pending buffers are no longer written to disk upon execution of a CHAIN instruction, or a TRAP for the IO or PARITY event.

WRITE. Previous DATASHAREs performed an unnecessary pre-read on RANDOM WRITEs. This problem has been corrected. DATASHARE VI does NOT perform this unnecessary pre-read of the sector for RANDOM WRITEs. The WRITAB instruction MUST be used if a pre-read is required for the purposes of updating.

WEOF. DATASHARE now positions the file to the start of the EOF mark. The file position was left in an incorrect state in previous DATASHAREs. This change will cause DATASHARE to function according to the DBCMPLUS User's Guide.

POLLINK. Pollink now requires the user written code to be separated into three distinct PROGS, one each for foreground, background, and initialization routines. Pollink now allows the baud rate to be configured either as fixed service rate or as free-wheeling service rate. This allows non-time critical pollink applications to take advantage of the free-wheeling service available to 3600 ports. See chapter 8 for more information.

UPS. DATASHARE VI will utilize all available memory when running under UPS on a 256K processor. Use of DATASHARE VI under UPS requires UPS 2.1.

1.4 New Features

The following features have been added since DATASHARE V version 3:

AIM, the Associative Index Method. This access method allows powerful and flexible access to a data base using generic keys. This access method is available on processors with greater than 64-K of user memory. (See DBCMPLUS User's Guide, Model Code # 50321.)

KEYIN List Controls

a) *CL - Clear the key-ahead buffer.
b) *RD - Roll down the screen -- only supported on 3600s.
c) *PON - Send a Printer-on character to a terminal. This list control is supported only on 3600s.
d) *POFF - Send a Printer-off character to a terminal. This list control is supported only on 3600s.

DISPLAY List Controls

a) *RD - Roll down the screen -- only supported on 3600s.
b) *PON - Send a printer on character to a terminal. This list control is supported only on 3600s.
c) *POFF - Send a printer off character to a terminal. This list control is supported only on 3600s.

SCAN, the SCAN verb. This verb allows pattern searches within a string variable. (See DBCMPLUS 3.1 User's Guide, Model Code # 50321.)

Memory Manager for large memory environments. A memory manager has been added for large memory machines (more than 64-K) to allow the interpreter code size to exceed the 32-K limit imposed by previous versions of DATASHARE. This effectively removes the limitation on the total size of configurable options for large memory environments. The 32-K limitation is still in effect for small memory environments.

1.5 Enhanced System Performance

The performance of DATASHARE VI has been enhanced by the replacement of the background scheduler, virtual storage manager (and extended buffer manager), and disk I/O facilities. These enhancements reduce system overhead and provide an increase from a maximum of 63 virtual storage buffers to a maximum of 768 virtual storage buffers.

The level of enhanced performance is dependent on system configuration and load and will vary from one application to the next.

CHAPTER 2. INTRODUCTION

DATASHARE VI is a multi-terminal business processing system which supports many configurations. Some of the features of these configurations are:

1) Up to twenty-four terminals on a 6600/6020/6040 processor, up to sixteen terminals on a 6010/5500/5000 processor, up to four terminals on an 1170/1810/1820/3810/3820 processor using a 9462 Multiport Adaptor, or up to nine terminals on an 1810/1820 processor using two 9320 Internal Multiport Adaptors and port-1-on-console, all with internal (port-to-port) communications facilities.

2) The DATASHARE Slave Station Facility (DSSLAVE).

3) The DATASHARE External Communications Facility (MULTILINK) with variable MULTILINK program size.

4) A variable-size ACALL Facility with the ACALL program optionally specifiable at execution time.

5) A special high-speed DATABUS-type foreground/background scheduler which will be configured whenever port-1-on-console alone is configured.

The above capabilities are selectable via configuration options and will be discussed in chapter 3 of this User's Guide.

DATASHARE VI allows the concurrent execution of up to twenty-four DATABUS programs each dealing with its own terminal. The terminals may be either remote or local 3360s and 3600s, local 3670s, or Remote Slave Stations. A Remote Slave Station is a Datapoint system with diskette (or disk) which is connected to the Central Station via a dial-up communications link. Additionally, an external communications (MULTILINK) facility exists that will allow DATASHARE to appear to a mainframe as a polled, selectable terminal via a compatible communications handler. The communications handlers are released and documented as separate programs.

In addition to the above terminal configurations, DATABUS programs may also be run by DATASHARE in any configured port without any attached terminal. These "phantom" ports are prohibited from performing KEYIN, BEEP, or DISPLAY operations but

otherwise are identical in nature to an actual terminal.

The DATASHARE VI Interpreter requires a 6600/6040/6010/6020/5500/5000/1170 processor executing DOS.C, DOS.D or DOS.E in a minimum of 32K of user memory, an 1810/1820/3810/3820 processor executing DOS.G or DOS.D under ARC in a minimum of 56K of user memory, or an 1810/1820 processor executing DOS.D on a 9320 disk controller in a minimum of 56K of user memory; (the same relocatable code is LINKed at configuration-time to run on the proper processor under the proper DOS).

DOS.	Minimum Memory	ARC	UPS	Processors
C	48-K	NO	NO	1170
D	48-K	YES	YES*	6600/5500/1800/3800/5000**
E	32-K	NO	NO	5000/6600/5500
G	56-K	NO	NO	1800

*UPS support requires a 120-K processor minimum.

**The 5000 44-K and 5000 32-K processors are not supported under DOS.D, only DOS.E.

Since it executes under DOS, the Interpreter can take advantage of all of the DOS's file-handling capabilities. The Interpreter may also be used to access files randomly, sequentially, index-sequentially, or by associative index, and thus provides a powerful data entry and processing facility. In addition to file handling, full control is provided over either a local 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.

Using virtual memory techniques, DATASHARE VI provides each program with a 65,024-byte area for executable statements (less any data area). This enables a user to create and use very large DATABUS programs. To provide rapid program execution, the data area for each program is maintained in main memory and is not swapped to disk. This data area is referred to as User Data Area (UDA). The system can be configured to run up to twenty-four

ports (depending upon the executing processor's type) with the total data area size partitioned among the ports. The data area of any port can be configured from 256 to 15,872 bytes.

Datapoint system printers (local or servo) are supported at the Central site and Remote Slave sites. If the Central site 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; these programs may also handle system security. The MASTER and ANSWER programs are not provided with the system; they are to be created and compiled like any other DATABUS program. This enables a 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 DBCMPLUS Compiler. All DATABUS object code must physically exist on the local disks of the DATASHARE VI Interpreter if local disks are being utilized, or must exist on currently-MOUNTed drives if running under ARC.

CHAPTER 3. SYSTEM GENERATION

3.1 Preparing the DATASHARE VI Environment

The DATASHARE VI relocatable Interpreter files are contained on a disk, diskette, or cassette tape (depending upon the input media available to the DATASHARE processor) which should be duplicated as soon after its delivery as is possible to avoid the loss of the relocatable Interpreter files in case of a disk CRC, parity error, or cassette tape jam.

The DATASHARE VI files for 6000/5000/3800 processors are released on cassette tapes for MINing onto an ARC system. Since these processors lack cassette facilities, the cassette must be MINed either by another Applications Processor with cassette facilities, or the ARC network must be taken down and the cassette tape MINed onto the appropriate disk by the File Processor.

If the DATASHARE VI Interpreter files arrive on a cassette, they may be loaded onto disk via the command:

```
MIN;A
```

This will activate the MIN (Multiple IN) program 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 asks if it is to be overstored. Any pre-existing DATASHARE VI files should be overwritten unless different versions of DATASHARE VI are being utilized on the same system. If this is the case, each set of files should be catalogued into separate subdirectories. When a particular DATASHARE VI is to be run, the appropriate subdirectory must be invoked before initialization. This procedure should also be followed for cataloging MULTILINK and ACALL programs. Use of subdirectories will prevent DATASHARE VI from erroneously using files from another version. (Consult the DOS User's Guide for further information on this procedure.)

The DATASHARE VI Interpreter files are DS6A/TXT and DS6B/TXT (the Configurator CHAINPLS files,) and DS6A/REL and DS6B/REL (the relocatable Interpreter Library files which contain the modules necessary to build any allowable configuration of the Interpreter).

The Remote Station interpreter DSSLAVE (User's Guide Model Code # 50537), is released separately.

3.2 Configuring a DATASHARE VI Interpreter

The Configurator allows the user to select which options of DATASHARE are to be used in any particular Interpreter, and to select the port type and the amount of User's Data Area (UDA) available to each of the ports.

DATASHARE VI makes efficient use of all the memory available in all machines supported. In particular the upper 64K of 1820 and 3820 machines, and the upper 128K on 6040 machines (256K 6000s) was used by DATASHARE V Version 3 only for extended buffer storage and only if the extended buffer manager was configured. Earlier versions of DATASHARE did not use this memory at all. DATASHARE VI allows this memory to be used for any purpose: Interpreter code, UDA, or virtual storage buffers.

There are some fundamental differences in the way the Configurator organizes DATASHARE's use of memory in different machines, and some features can only be configured on some machines. In the following discussion the term "large memory environment" refers to any machine in which the configured Interpreter will have more than 64K bytes of memory for its use, and "small memory environment" refers to any machine in which the configured Interpreter will have 64K or less bytes of memory for its use.

Earlier versions of DATASHARE were restricted to an upper limit on the amount of memory that could be dedicated to Interpreter code (as opposed to UDA and buffer storage), and thus the number of features that could be configured was limited. In a large memory environment, DATASHARE VI employs a memory manager that allows parts of the Interpreter's code to be placed in upper memory which can be accessed by using the processor's sector table. This alleviates, to a great degree, the size restrictions placed on the Interpreter.

Before configuring any DATASHARE VI Interpreter, the two parts (DS6A/TXT, DS6B/TXT) of the Configurator program, DS6/TXT, must be put together. This can be done by executing the following command:

```
SAPP DS6A/TXT,DS6B/TXT,DS6/TXT
```

The program CHAINPLS is an integral part of the DATASHARE VI

configuration facility; the DATASHARE user should be aware of its main features, but a detailed knowledge of its capabilities is not necessary for proper DATASHARE configuration (consult the CHAINPLS User's Guide for details of CHAINPLS operation). The CHAINPLS/CMD file and its overlays must be available to DOS when DATASHARE VI is configured and must be run on a processor with at least 48K of user memory. In addition, the LIBSYS, LINK, KILL, and CHANGE programs must be available either as stand-alone files, or in UTILITY/SYS (see the DS6 RFM sheet supplied with the DATASHARE VI release). The file DOSEPT/REL must also be available. The command line given to DOS is simply:

```
CHAINPLS DS6
```

which will cause CHAINPLS to display a signon message and begin the configuration process. If a listing of the LINK-generated information is desired, entering

```
CHAINPLS DS6;LISTLINK
```

will cause LINK to list its output on a local printer, or

```
CHAINPLS DS6;LISTLINK=:<volid spec>
```

will cause LINK to produce a print file named <Interpreter-name>/PRT on drive <volid spec>.

If the Configurator cannot find the file DS6/REL (as is usually the case when the Configurator is first run after the DS6A/REL and DS6B/REL files have been loaded,) the message

```
. DS6/REL MUST BE CREATED FROM DS6A/REL AND DS6B/REL BEFORE
FURTHER PROCESSING.
```

will appear followed by a LIBSYS run to create DS6/REL. After this process has finished, the Configurator must be re-started to produce an Interpreter (LISTLINK or any other options used on the original command line must be re-entered). If the Configurator cannot find DS6/REL, or DS6A/REL and DS6B/REL, the message

```
. ***** FILE 'DS6A/REL' AND/OR 'DS6B/REL' NOT PRESENT!
. ***** CONFIGURATION ABORTED!
```

will appear and the Configuration will be aborted.

CAUTION: If a DS6/REL file already exists on the disk media being used to construct DATASHARE VI Version 1, the Configurator will pick up this old DS6/REL file as its supposedly new Version 1

file and will not construct a new DS6/REL file from the DS6A/REL and DS6B/REL files MINed in earlier. Old DS6/REL files should be KILLed or NAMEd differently.

If the utilities LINK/CMD and/or LIBSYS/CMD are unavailable to the Configurator, the message

***** LINK/CMD and/or LIBSYS/CMD DO NOT EXIST; CONFIGURATION ABORTED!

will appear and the Configuration will be aborted.

If the utilities KILL/CMD and/or CHANGE/CMD are unavailable to the Configurator, the message

***** KILL/CMD and/or CHANGE/CMD DO NOT EXIST; CONFIGURATION ABORTED!

will appear and the Configuration will be aborted.

If the file DOSEPT/REL does not exist, the message

***** DOSEPT/REL DOES NOT EXIST; CONFIGURATION ABORTED!

will appear and the Configuration will be aborted.

Throughout a configuration, the Configurator will assume certain default values for invalid or null answers to its configuration questions. The default value used in any given keyin will be indicated inside two matching square brackets. If, for example a yes-or-no question is asked and a default value of NO will be assumed for a null or invalid keyin, then the question would be displayed as follows:

QUESTION? [N]

If the response to the question is anything except "Y" or "YES", then the default value of "N" would be assumed. If, however, an invalid or null response is not allowed to a question, then no default answer will be shown inside the square brackets, e.g.

QUESTION? []

and a null or invalid keyin will evoke an error message and cause the question to be re-asked except for the SIGN-ON message question. A null response to this question results in a null sign-on message.

3.2.1 Specifying the DATASHARE VI Interpreter Name

The option NAME=<Interpreter-name> may be used on the DOS command line if desired:

```
CHAINPLS DS6;NAME=DS6
```

or, if the option is not specified, the Configurator asks:

```
WHAT DO YOU WISH TO NAME THIS DATASHARE VI INTERPRETER? []
```

Either response should be an up to seven character DOS-legal file name; this name will be applied to the /PRT listing file (if LISTLINK=:<volid spec> is used), and to the /CMD, /CFG, and /LEX files which are generated and used by the DATASHARE VI Configurator.

If more than seven characters are used in the file name, the Configurator will respond:

```
***** INTERPRETER NAME LONGER THAN SEVEN CHARACTERS; REENTER NAME
```

If no name is given, the Configurator will beep, display

```
***** INVALID INTERPRETER NAME!
```

and re-ask the NAME question.

If a /CFG file of that name does not exist, the Configurator will create one and give the following message:

```
--- CREATING NEW CONFIGURATION FILE '<Interpreter-name>/CFG' ---
```

If a /CFG file exists with the same name as the one entered by the user, the current configuration of the /CFG file will be displayed on the console and a negative ("N") response to the question:

```
CHANGE THE CONFIGURATION? [Y]
```

will begin the LINK of an Interpreter with the features given in the /CFG file.

If the response is "Y" or the /CFG file did not exist in the first place, then another series of questions will be asked.

3.2.2 Configuring the Processor Environment

The first question asks for a sign-on message to be entered:

ENTER A SIGN-ON MESSAGE OF UP TO 50 CHARACTERS []

This sign-on message will be displayed during the Interpreter's initialization process and is merely for user documentation purposes. If a null keyin is received for this question, no signon message will be displayed. If more than fifty characters are entered, the message will be truncated to exactly fifty characters. If the characters ', ", or # are entered into the sign-on message they will be ignored by the Configurator as they cause either CHAINPLS or LINK errors to occur.

The next question to be answered deals with the processor and its environment:

ARE YOU CONFIGURING FOR THE CURRENT ENVIRONMENT (I.E. FOR THIS <memory size>-K <processor> PROCESSOR RUNNING DOS.<dos> 2.<dos revision> {WITH ARC}|{WITH MIDS}? [Y]

where <processor> is of the category 6600, 6010, 6020, 6040, 5500, 5000, 1170, 1810, 1820, 3810, or 3820, <memory size> is the number of bytes of memory on-line and available for use by DATASHARE VI in the <processor>, <dos> is the letter of the DOS currently running, and <dos revision> is the revision number of the DOS currently running. If the configuring processor is to be the host of the DATASHARE VI Interpreter to be generated, this question should be answered with "Y". If, however, the configuring processor is not to run the Interpreter, but instead is performing the configuration for another processor, the question should be answered with "N".

If the answer to the question is "N", however, then the following questions will be asked:

WHICH PROCESSOR GROUP IS THIS CONFIGURATION FOR?
6600, 6010, 6020, 6040, 5500, 5000, 1170, 1810, 1820, 3810, OR 3820? []

should be answered with the number of the processor being configured for.

If the processor type configured is a 3810 or a 3820, the Configurator automatically selects ARC as the environment. If any other processor is configured, then the question

WILL ARC BE ACTIVE ON THIS PROCESSOR? [N]

should be answered with "Y" if ARC will be running on the processor when the DATASHARE VI Interpreter is brought up, or with "N" if ARC will not be running.

WHAT DOS. WILL BE RUNNING? (IF DOS.C, ENTER 'C', ETC.) []

should be answered with the letter of the DOS that will be active when the processor runs the DATASHARE VI Interpreter. (If ARC is to be used, the DOS letter will always be assumed to be "D".)

If DOS.D is selected, the environment is DOS (not ARC), and the processor type is not an 1810 or 1820, the question

WILL DOS.D BE RUNNING ON A MIDS (9390) DISK SYSTEM? [N]

should be answered with "Y" only if a MIDS disk is being used as the system disk.

WHAT DOS REVISION WILL BE RUNNING? (IF 2.6, ENTER '6', ETC.) []

should be answered with the revision number of the DOS that will be active when the processor runs the DATASHARE VI Interpreter.

*NOTE: DATASHARE VI interpreters must be re-linked after a DOS upgrade which results in a change in the revision level of the DOS.

HOW MANY K-BYTES OF MEMORY WILL THE PROCESSOR HAVE? []

should be answered with the number of K-bytes of memory on-line and available to DATASHARE VI in the processor which will run the Interpreter; i.e. if 56K of memory is available, then the response should be "56". The following table shows the allowable memory configurations of all processors in their respective environments.

PROCESSOR	UNDER DOS	UNDER ARC
6600	120, 88	116, 84
6010	56	52
6020	120, 88	116, 84
6040	248, 216, 184, 152	244, 212, 180, 148
5500	56, 48	52, 44
5000	56, 44, 32	52
1170	56, 48	52, 44
3810	<N/A>	48
3820	<N/A>	112, 80
1810	52, 48*	48
1820	116, 112*, 84, 80*	112, 80

* Note: The smaller memory sizes must only be used if the Interpreter will be running on an 1800 processor utilizing one or more 9320 local disk controllers for DOS.D.

If a processor type of 6600, 6020, or 6040 was given, the DOS was specified as .D or .E, and ARC was not selected, the Configurator asks

WILL UPS BE ACTIVE ON THIS PROCESSOR? [N]

The response should be "Y" if the Utility Partition Supervisor (UPS) will be in control when the Interpreter is run, and "N" otherwise.

When configuring for a UPS environment, the question above about memory size should be answered with the total amount of memory available for the machine being configured (from the above table). The Configurator will automatically adjust the memory size value to allow for the memory that will be allocated to the other partition.

Under UPS, a 6600/6020 processor is handled by DATASHARE as a small machine environment; a 6040 is handled as a large machine environment. The restrictions which apply to small machine environments apply to 6600/6020 processors under UPS. The 6040 is subject to the restrictions of the large machine environment.

3.2.3 Activating the Interpreter Options

In general, answering any of the following questions negatively, (i.e. not requesting the features they allow,) will save many bytes of processor memory which will be returned to the user in the form of available User's Data Area and extra virtual storage buffers. If the feature is not required by the DATABUS programs to be run on the interpreter, the features should not be configured.

The question:

DO YOU WANT THE SPOOL VERB ACTIVATED? [N]

should be answered with "Y" if print spooling is desired in addition to the regular PRINT capabilities, and with "N" if it is not. (Note: The print spooling module requires an extra page of User's Data Area and will reduce the maximum amount of UDA available to each port by 256 bytes; in addition, the "new" TRAPs will be activated, as below.)

If SPOOLing is selected, the Configurator automatically activates the system printer. If SPOOLing is not selected, then the question

DO YOU WANT A SYSTEM PRINTER ACTIVE? [N]

should be answered with "Y" if a system printer is to be used by the Interpreter, and with "N" if no print capabilities are needed.

If a printer is configured, the question

DO YOU WANT TO USE A SERVO PRINTER FOR THE SYSTEM PRINTER? [N]

should be answered with "Y" if a servo printer will be used as the Interpreter's system printer when the Interpreter is initialized, and with "N" if a local printer will be available to DATASHARE VI for printing.

If a printer is configured, the Configurator asks the question:

ADDRESS OF PRINTER? [nnnn]

The number nnnn is the default address of the printer. It will be 0303 if a local printer is configured, and 0132 if a servo printer is selected. A null response to this question results in the default address being used. If the printer address is different

from the default, the new address must be entered.

If a printer has been configured, the Configurator asks

DO YOU WANT TO INHIBIT PAGE EJECT ON RELEASE? [N]

This question should be answered with "Y" if a page eject is not desired upon a ROLLOUT, RELEASE, or CHAIN statement (if a PRINT statement has been executed by the program), and with "N" if a page eject is desired under these conditions.

If the Interpreter is not to be run under UPS then the question

DO YOU WANT THE CONSOLE TO BE USED FOR PORT 1? [N]

will be asked. This question should be answered with "Y" if the processor console is to be used for port 1 and with "N" if it is not.

If the Interpreter is not to run under UPS and the console is not to be used for port 1, the Configurator asks:

DO YOU WANT THE CONSOLE TO BE USED AS A SYSTEM CONSOLE? [N]

This question should be answered with "Y" if the system console is desired, and with "N" if it is not desired.

The system console is used to display the name of the DATABUS program being activated through a CHAIN instruction, as the recipient of data in the CONSOLE instruction, and displays the day and year of initialization and a running wall-time clock. If there is no need for anyone to observe this information on the console screen, the user can save many bytes of code and some system overhead by answering no to this question. (Note that the clock is still kept updated internally by the interpreter and is still available to DATABUS programs through the CLOCK instructon.)

If the Configurator has been given the parameters of 1810/1820 processor and DOS.D, the Configurator asks:

WILL A 9320 MULTIPORT ADAPTOR BE USED ON THIS PROCESSOR? [N]

This question should be answered with "Y" if the processor will utilize the 9320 internal Multiport Adaptors, and with "N" if this is not the case.

DO YOU WANT POLLINK ACTIVATED? [N]

may be answered with "N" to disable the polling feature of the Interpreter, or may be answered with "Y" to allow pollable ports to be used. If "Y" is indicated, the user-written module POLLINK/REL must be available to the Configurator for LINKing, and must contain three PROGs named POLLBG, POLLFG, and POLLINIT. The contents of each of these PROGs is discussed in the section on POLLABLE TERMINAL SUPPORT in chapter 8 of this User's Guide.

Two types of Remote Slave communications protocol are available; they are (1) asynchronous, and (2) synchronous. If either is desired, the question:

DO YOU WANT REMOTE SLAVE PROCESSING ACTIVATED? [N]

must be answered "Y"; if not, "N" is the correct response. If "Y" is entered,

DO YOU WANT (1) ASYNCHRONOUS OR (2) SYNCHRONOUS COMMUNICATIONS? []

should be answered with "1" or "2" depending upon the communications method desired.

If External Communications (MULTILINK) are desired, the question:

DO YOU WANT EXTERNAL COMMUNICATIONS ACTIVATED? [N]

should be answered with "Y". If Internal Communications only or if neither External nor Internal Communications are desired, the answer should be "N". If External Communications are requested, a later request for the amount of memory required by the Multilink Line Driver program (the /COM program) will be made, and the file <Interpreter-name>/COM will be expected to be available to the Interpreter when it is initialized.

If External Communications were not requested, the question:

DO YOU WANT INTERNAL COMMUNICATIONS ACTIVATED? [N]

will be asked. If Internal Communications are desired, the question should be answered with "Y"; if no Communications are necessary, a response of "N" will prevent any Communications from being activated.

If, at this point, Remote Slaves have not been configured, the POLLINK feature has not been configured, and External Communications have not been configured, the Configurator asks the

question:

DO YOU WANT THE 3670 PORTS OPTION ACTIVATED? [N]

A response of "Y" will activate the special foreground routines necessary to communicate with a 3670 terminal in the Interpreter and will enable the selection of port type "3670" later in the Configurator.

If the 3670 ports option has been configured, or the 9320 Multiport Adaptor has been selected, the Configurator automatically activates the Wait for CTS option, otherwise, the question:

DO YOU WANT TO WAIT FOR CTS BEFORE SENDING TO TERMINALS? [N]

must be answered with "Y" if it is desired that DATASHARE wait for a terminal's Clear-To-Send signal to go "high" before sending output characters to the terminal, or with "N" if this is unnecessary. Selection of this feature, along with proper interface cabling, will allow a serial printer to notify DATASHARE that it is not available for character reception and will avoid loss of print data in case of a paper-out or other off-line condition. If this feature is selected, ALL terminals on the system must be wired for CTS wait. (See section 7.2.1, Direct Terminal Connections for proper interface cabling requirements.)

If SPOOLing was not configured, the question

DO YOU WANT THE NEW TRAPS ACTIVATED? [N]

is presented. An answer of "Y" activates the SPOOL, INT key, F1 through F5, and <character> key traps and additionally deducts 256 bytes from the maximum UDA available to each port in the same manner as the print spooling activation. If the "new" traps are activated by the configuration of print spooling, as above, only 256 bytes of available UDA per port will be taken as the traps and print spooling modules will share the 256-byte area allocated to each port.

DO YOU WANT MEMORY RESIDENT TRAPS? [N]

should be answered with "Y" if it is desired to have the DATASHARE trap handler permanently resident in main memory, and with "N" if the trap handler is to be an overlay that is loaded by the Interpreter when a trap is taken. Unless timely responses to TRAPped events such as function keys, the interrupt key, or a character trap are required, the TRAP handler should be left as an

overlay.

If the Interpreter is not configured to run under UPS, the Configurator asks

DO YOU WANT ROLLOUT ACTIVATED? [N]

This question must be answered with "Y" if the ROLLOUT instruction is to be enabled, and "N" if the ROLLOUT instruction is to be disabled. Specifying "Y" automatically activates the SHUTDOWN verb. Specifying "N" causes the Interpreter to set the OVER flag if a ROLLOUT is requested, instead of returning to DOS.

If the ROLLOUT question is answered with "N", the Configurator then asks:

DO YOU WANT SHUTDOWN ACTIVATED? [N]

This question must be answered with "Y" if the SHUTDOWN instruction is to be enabled, and "N" if the SHUTDOWN instruction is to be disabled. Specifying "N" causes the Interpreter to set the OVER flag if a SHUTDOWN is requested, instead of returning to DOS.

If "Y" is entered in response to either the ROLLOUT or SHUTDOWN question, the message

===== CHECKING FOR ROLLOUT/SHUTDOWN MEMBERS =====

appears and the Configurator will check for the presence of the proper ROLLOUT and SHUTDOWN modules in the DSG/REL file.

DO YOU WANT THE ACALL FACILITY ACTIVATED? [N]

must be answered with "N" if no ACALL is desired, and with "Y" if either dynamic or one-program ACALL is desired. If the answer is "Y" and dynamic ACALL is desired (i.e. the ACALL module name is to be specified as the first parameter of the ACALL instruction when the instruction is executed), the question:

DO YOU WANT DYNAMIC ACALLs? [N]

must be answered "Y". If, however, only one-program ACALL is needed, "N" should be the response. The Configurator will later request the amount of memory used by the ACALL overlay area.

*NOTE: If one-program ACALL is configured, the interpreter will require the availability of a program named

<Interpreter-name>/ASM during initialization.

If the index-sequential disk I/O (ISAM) capabilities are required, then the question:

DO YOU WANT THE ISAM DISK I/O PACKAGE ACTIVATED? [N]

should be answered with "Y"; the response "N" should be used if they will never be used under the Interpreter being configured.

If the configured Interpreter is to run in a large memory environment then the Configurator asks:

DO YOU WANT THE AIM DISK I/O PACKAGE ACTIVATED? [N]

This question should be answered with "Y" if the Associate Index Access method capabilities are to be used, and with "N" if they will never be used under the Interpreter being configured.

The questions:

DO YOU WANT THE SEARCH VERB ACTIVATED? [N]
DO YOU WANT THE REPLACE VERB ACTIVATED? [N]
DO YOU WANT THE CHECK10/CHECK11 VERBS ACTIVATED? [N]

should be answered "Y" if these DATASHARE verbs are desired, and "N" if the verbs will never be used in any DATASHARE program to be run under the Interpreter being configured.

DO YOU WANT THE CLOCK VERB ACTIVATED? [N]

must be answered with "Y" if any of the functions of the CLOCK instruction are to be used, and with "N" if not. If the configuration is for a large memory environment, the CLOCK verb is made memory resident. If the configuration is for a small memory environment, the question

DO YOU WANT THE CLOCK VERB TO BE MEMORY RESIDENT? [N]

should be answered with "Y" if it is desired that the code for the CLOCK verb be permanently resident in main memory, and with "N" if the CLOCK verb is to be an overlay that is loaded by the Interpreter when a CLOCK instruction is executed.

The next configuration question:

DO YOU WANT THE EDIT VERB ACTIVATED? [N]

must be answered with "Y" if the EDIT verb is to be used in the Interpreter, and with "N" if it is not.

DO YOU WANT THE SCAN VERB ACTIVATED? [N]

must be answered with "Y" if the SCAN verb is to be used in the Interpreter, and with "N" if it is not.

The DATASHARE VI system may be configured to run with up to twenty-four ports on a 6600/6020/6040 processor, up to sixteen ports on a 6010/5500/5000 processor, up to four ports on an 1170/1810/1820/3810/3820 processor, and up to nine ports on an 1810/1820 processor utilizing port-1-on-console and two 9320 internal MPAs. When the question:

WHAT IS THE MAXIMUM PORT NUMBER TO BE CONFIGURED? []

is asked, the number of ports desired (including ports to be disabled) should be entered. If a multiple-user, time-sharing DATASHARE system is desired, the highest-numbered active port should be entered. If, however, a single port is desired and a high-throughput, port-1-on-console only, single-user Interpreter is desired, a "1" should be entered.

Next, the Configurator calculates the amount of memory available to the Interpreter; if ACALL was activated, (see above,) the following question is presented:

HOW MANY BYTES OF MEMORY MUST BE RESERVED FOR ACALL? [] (xxxx BYTES LEFT)

where xxxx is the memory size available. The maximum size available for ACALL is 4096 bytes but may be less for some configurations if insufficient memory is available. Any answer from "1" to xxxx will be rounded up to the nearest 256-byte page and will then be reserved as the number of bytes (in decimal) needed by the largest ACALL module to be used with the Interpreter being configured. Any such module(s) must be LINKed with the /LEX file which will be produced upon completion of the Configuration, and must be on-line when DATASHARE VI is initialized. If one program ACALL is to be configured, the ACALL program MUST have the name "<Interpreter-name>/ASM".

If External Communications were activated (see above), the Configurator asks for the size of the area to be reserved for the Multilink Line Driver. If the configuration is for a large memory environment, the question is preceeded by one of the following two statements:

yyyy BYTES LEFT FOR MULTILINK FOR MAXIMUM PERFORMANCE

or

MAX PERFORMANCE LIMITS EXCEEDED, FOREGROUND WILL BE IN UPPER MEMORY

For a large memory environment, if all the foreground code in the Interpreter (which includes the Multilink Line Driver) is small enough to fit in a certain region of the first 64K of memory, it will be placed there. The first message is displayed to inform the user how much memory is left for foreground until this "threshold" size is reached. If the size reserved for the MULTILINK module is less than this amount, then the foreground code will be placed in low memory and be permanently addressable. If the size reserved is larger than this amount (but below the maximum amount allowed which is displayed in the following question), the foreground code will be placed in upper memory and accessed via the sector table when needed. Upper memory foreground code will result in slightly degraded performance.

If the size of the foreground code configured so far is already such that the "threshold" size is exceeded (that is, even with a 0 size area reserved for the MULTILINK module, the foreground code would not fit in the region in low memory), the second message will be displayed.

The following question is then presented:

HOW MANY BYTES MUST BE RESERVED FOR THE ML DRIVER? [] (xxxx) BYTES LEFT

where xxxx is the memory size available. The maximum size allowed for the MULTILINK Line Driver is 8192 bytes but may be less on some configurations if insufficient memory is available. This question should be answered with the number of bytes (in decimal) that is required by the largest MULTILINK line driver to be used with the Interpreter being configured. Any such line driver must be LINKed with the /LEX file which will be produced upon completion of the Configuration, and must be on-line and have the name "<Interpreter-name>/COM" when DATASHARE VI is initialized.

3.2.4 Configuring the Ports

At this point the Configurator calculates the maximum aggregate BAUD rate that may be divided among the ports and displays the total remaining of this amount immediately prior to each port's configuration. The maximum aggregrate BAUD rate is 4800 for all 1800/3800 processors, 9600 BAUD for all non-1800/3800 processors if External Communications is configured, and 19200 BAUD for all non-1800/3800 processors if External Communications is not configured. When the display:

```
bbbbb BAUD LEFT *** PORT <n>:  ENTER TYPE: []
```

is produced, the user must enter the port type of port <n> (either "X", "3600", "3360", "3670", "SLAVE", "PHANTOM", "POLL0" through "POLL9", or "^" is acceptable). The "X" option is used to disable a port (no foreground or background service will be given the port; it will be considered "dead"). "^" may be used to exactly duplicate the type and BAUD rate of the previous port configured. (The "^" can not be used to duplicate the configuration of disabled ports, type "X"). "SLAVE" can be selected only if Remote Slave Processing has been activated. The type "POLLn" can be selected only if the POLLINK option has been activated. The type "3670" can only be selected if the 3670 ports option has been activated.

If the user responded "Y" to the earlier question about using the processor console for port 1, the Configurator automatically selects the processor's console to be used in place of port 1 when the Interpreter is run.

If 9320 internal MPAs are in use and the number of ports entered is less than "9" but greater than "1" and the type of port 1 is configured as "CONSOLE", the Configurator will configure ports 2 through 5 into the four physical connections 0 through 3 of the first 9320 internal MPA, and ports 6 through 9 into the four physical connections 0 through 3 of the second 9320 internal MPA. If 9320 internal MPAs are in use and the number of ports entered is "9", the Configurator will automatically select "CONSOLE" as port 1's type and will configure ports 2 through 9 as above.

The question:

ADDRESS of 9402? []

or

ADDRESS of 9481? []

appears if the port type selected was "SLAVE". (See Section 7.5 for valid 9402 and 9481 addresses.) The address of the 9402 or 9481 should be entered in octal with or without a preceeding 0.

If "3600" or "3360" has been entered for the port type, it may be followed by a slash (/) character and a decimal number which assigns a specific fixed-service BAUD rate to the port (i.e. 3600/1200 specifies a 3600 port with a fixed service rate of 1200 BAUD). If no fixed-rate service is required, the port type of 3600 or 3360 alone should be used to indicate this preference. (The port type "3670", although similar to the 3600 port type, may not be configured with a fixed-service BAUD rate.)

If type "SLAVE" is entered, and the synchronous communication method has been selected, the BAUD rate of communications may be specified by following the SLAVE type with a slash (/) character and a decimal number indicating the BAUD rate desired (either 1200, 2400, 4800, or 9600). Note: The 9481 Communications Adaptor must be strapped for INTERNAL CLOCK to be able to use the configurable BAUD rate.

If type "POLLn" is entered, the port will be available to the POLLINK feature of the Interpreter and will be given the logical-discipline number n, where n is a number from zero to nine representing a common poll discipline.

The port type "POLLn" may be followed by a slash (/) character and a decimal number which assigns a specific fixed-service BAUD rate to the port (i.e. POLL1/1200 specifies a POLLed port with poll discipline 1 and a fixed service rate of 1200 BAUD).

*NOTE: Any POLLINK applications which are time critical must use a specific fixed-service BAUD rate.

"3600" and "3360" ports will be set at 300 BAUD for input with output dynamically sharing system capacity unless otherwise specified. 3670 ports will be set at 300 BAUD for input in 3600-type mode; output and 3270-type input mode will dynamically

share system capacity. Dynamic output will vary depending upon the number of ports simultaneously performing output. Ports with a "SLAVE" type will be assigned a default BAUD rate of 1200 for asynchronous communications, and a default value of 4800 BAUD for synchronous communications. Port types of "CONSOLE" will be assigned an input rate of 300 baud and a dynamic output rate. "PHANTOM"s are assigned a baud rate of 0.

The 3600/3360 terminals should be hardware strapped to their maximum speed (9600 BAUD for 3600s and 4800 BAUD for 3360s). Actual display speed will be determined by software configuration.

Ports should not be configured at fixed BAUD rates greater than 300 BAUD if input is obtained from a keyboard. As a general rule, the BAUD-rate specification should be omitted for 3600s and 3360s, and electro-mechanical TTY-type devices should be configured at 150 BAUD for maximum system efficiency. "3600" should be selected for non-standard upper/lower-case devices, while "3360" should be used for upper-case only devices (like TTY and OCRs). Blocks of input characters from high-speed devices like OCR readers should be limited to 64 characters to prevent possible keyin buffer overruns during heavy system activity. (Note that characters with values below octal 040 will be discarded by DATASHARE VI.)

At this point the Configurator will calculate the amount of User Data Area (UDA) available. The amount depends upon the size of the Interpreter (controlled by the number and internal size of the options configured,) and upon the amount of on-line memory available to the Interpreter's processor. If the processor is a 6600, 6020, 6040, 1820, or 3820 it will contain memory which can be accessed by the use of the sector table. Note that previous versions of DATASHARE did not allow the upper memory on 1820/3820 processors to be used for User Data Area, and only DATASHARE V 3.1 and above used this memory at all (via the extended buffer manager). DATASHARE VI allows complete use of all memory on all machines. This extended memory can be used for UDA (as well as interpreter code and virtual storage buffers) and will be included in the number of bytes of UDA available for port configuration in addition to the memory normally available without the use of the sector table.

Each port should be configured with the minimum UDA necessary to perform its required tasks; this is done by entering an up to five byte decimal number in response to the display:

bbbbbb BYTES LEFT --- PORT <n> --- TYPE <type> --- DATA AREA SIZE? []

where bbbbbb is the maximum number of bytes of UDA remaining to the Interpreter and <type> is the the type of the port configured.

WARNING: The DATASHARE VI Interpreter will utilize unallocated free memory for virtual storage buffers. The arbitrary assignment of more data area to the ports than is necessary will usually have a negative effect on the performance of DATASHARE VI.

If at any time the Configurator detects that a user has configured too much UDA (at least 256 bytes must remain per port left to be configured), it will display:

***** INSUFFICIENT UDA AVAILABLE FOR REMAINING PORTS!

and the user will be asked to enter the UDA size for that port again.

The Configurator will round the UDA size given up to the nearest 256-byte page and will then display the message

DATA AREA RESERVED: bbbbb BYTES - FRAGMENTATION GIVEN TO VS: fffff

The Configurator must decide where in physical memory to allocate each port's User Data Area. Under certain circumstances a situation may arise where a certain amount of the memory cannot be used for UDA. This will only happen when configuring for a large memory environment and when the amount of data area for a port (including actual data area, port overhead, and any Slave or Poll overhead if applicable) is between 12.5K and 15.75K bytes. Even if the size is in this range, fragmentation will not always occur.

*NOTE: If fragmentation occurs during configuration, there is no effect on interpreter configuration or capabilities, only on the amount of memory available for User Data Area. The fragmented memory is removed from the memory available for User Data Area and allocated to the virtual storage manager for use as buffers.

When the last port has successfully had its UDA configured, the Configurator will display:

===== WRITING TO FILE <Interpreter-name>/CFG

and a series of clicks will be heard as it writes the configuration information to disk. When this has been accomplished, a re-display of the current configuration will be shown and the question:

CHANGE THE CONFIGURATION? [Y]

will be re-asked. If the response is "Y", the configuration process will begin again from the start. Otherwise, the message:

. NOW LINKING FILE <Interpreter-name>/CMD

will appear and the actual chaining to LINK to produce the Interpreter will begin.

3.2.5 The SPECIAL option

A provision has been placed in the Configurator to enable a group of special questions. To enable these questions the user must specify the SPECIAL option on the CHAINPLS command line. For example:

CHAINPLS DS6;NAME=DS6,LISTLINK,SPECIAL

The Configurator asks:

DO YOU WANT TO CHANGE THE SHIFT INVERSION LIMITS? [N]

The normal shift inversion limits are 0101 ("A") and 0132 ("Z"). If different limits are needed this question should be answered with "Y" which will cause the Configurator to ask the following two questions:

LOWER SHIFT INVERSION LIMIT? (0101) []

UPPER SHIFT INVERSION LIMIT? (0132) []

The new limits must be entered in octal. The range on the shift inversion limits is 0100 to 0136, and the lower limit must be less than or equal to the upper limit. Shift inversion is accomplished in DATASHARE VI by inverting the 040 bit of the character (i.e. 0103 ==> 0143; 0167 ==> 0127).

The question:

DISABLE FREEDOM PRINTER REAR TRACTOR SUPPORT? [N]

should be answered "Y" if the freedom printer rear tractor support is to be disabled and with "N" if it is to be enabled.

3.2.6 Two Pass Link

Due to the structure of the DATASHARE VI overlays and modules, it is necessary to LINK the Interpreter using two LINK commands. The primary purpose of the first pass of the LINK is to generate /LEX files for each module. During this first pass no external references will be resolved and LINK will display a message: UNDEFINED SYMBOLS ARE: and follow this with a list of the undefined symbols. These can be safely ignored.

During the second pass, the /LEX files produced by the first pass are used to resolve all the external references. If there really is an error that went unnoticed in the first pass, it will be detected during this second pass and LINK will abort. If the LISTLINK option is specified, the second pass will be the only one listed.

3.3 Necessary Programs

Before the DATASHARE VI Interpreter can be used, two other programs must be resident on the system's disk. These are called the ANSWER and MASTER programs; they perform the task of dealing with the user as he initially signs on to the system, and interfacing with him when he is not running another DATASHARE program. Since program execution in the DATASHARE system occurs in the high-level DATABUS language, the user writes his own ANSWER and MASTER programs, and therefore determines the structure of his system command language. The ANSWER and MASTER programming concepts are dealt with in Chapter 5.

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

DATASHARE initialization will look first for the free standing files ANSWERn/DBC and MASTERn/DBC. If either of these files can not be found, the DATABUS system library (See section 4.1.2) will be searched for the members ANSWERn and MASTERn.

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

ANSWERn/MASTERn above applies to the search for ANSWER/MASTER also.).

If a multi-drive system is being utilized, it is generally a good idea to keep all necessary system utilities, the DATASHARE VI Interpreter, and the ANSWER and MASTER program object files on the booted drive and not to remove the disk from this drive during normal system operation.

Other programs which should be available include the INDEX, REFORMAT, and SORT (or FASTSORT) utilities for the generation of ISAM index files, and AIMDEX for the generation of AIM index files.

CHAPTER 4. SYSTEM OPERATION

4.1 Bringing Up the System

In the following discussion it will be assumed that the DATASHARE VI Interpreter was named "DS6" during its configuration, although any DOS-legal file name, up to seven characters, might have been used.

*NOTE: Due to the overlay structure of the interpreter and the fact that the interpreter name is linked into the code, the /CMD file must NOT be renamed.

After the DATASHARE VI Interpreter has been configured and LINKed, it may be run by entering its name on the DOS command line:

```
DS6
```

This will begin a series of operations, the first of which displays the user-given signon message preceeded by:

```
DATASHARE VI 1.n <Date>
```

where n is the revision number of the particular release and Date is the assembly date of the /REL file used to link the interpreter. If the version of the DS6/REL and DS6/TXT files did not match at the time the interpreter was linked, the following message will be displayed:

```
VERSION OF DS6/REL AND DS6/TXT DID NOT MATCH AT LINK TIME!

DS6/REL WAS v1.r1.p1 -- DS6/TXT WAS v2.r2.p2
```

where v1.r1.p1 specifys the version of the DS6/REL file and v2.r2.p2 specifies the version of the DS6/TXT file. This is generally due to an incomplete upgrade of the DATASHARE VI release files (i.e. upgrading DS6/TXT, but not DS6/REL).

The initialization sequence checks the system environment (ARC/DOS/UPS), the DOS letter and revision, the memory size, and the processor type against the configured environment. If the DATASHARE VI configuration does not match the current environment,

then the following message will be displayed and the initialization aborted:

ILLEGAL ENVIRONMENT FOR THIS VERSION OF DATASHARE VI!

The initialization sequence then scans MCR$ for any command line options. If an illegal option is entered, the message:

BAD OPTION PARAMETER!

will be displayed and the initialization aborted. (See section 4.2.)

If the DATABUS library file specified on the command line (see Section 4.2.5) or the default library name DATASHAR/DBL cannot be found, the message:

SPECIFIED OR DEFAULT DATABUS LIBRARY NOT FOUND

will appear on the console display. The initialization sequence will not be aborted. (For more information about the DATABUS library facility of DATASHARE VI, see chapter 10.)

If a special port 1 answer program is specified on the command line (See Section 4.2.4) and it can not be found, the following message:

PORT 1 ANSWER PROGRAM MISSING

will be displayed and the initialization aborted.

The initialization sequence then verifys that all required overlay files are present in the command file library. If any internal overlay files are missing or are unreadable, the message:

<module name> -- MODULE MISSING OR UNLOADABLE

will be displayed and the initialization aborted.

If DATASHARE VI has been configured to run under UPS, the initialization sequence will verify the proper UPS environment. If either an incorrect version of UPS is active or insufficient memory is available from UPS, then the message:

INVALID UPS ENVIRONMENT FOR THIS CONFIGURATION

will be displayed and the initialization aborted.

If no fatal errors have been detected above, the initialization of the DATASHARE VI system tables will begin. The virtual storage manager initializes and displays the number of buffers allocated with the following message:

VIRTUAL STORAGE MANAGER INITIALIZED... nnn BUFFERS ALLOCATED.

where nnn is the number of buffers initialized.

The system will then look up all of the ANSWER and MASTER program names in the DOS directory and store their physical file numbers and logical drive numbers in a table. During this process, a click will occur as each port's programs are recorded.

If all has initialized properly so far and the system console has been configured, the console screen will clear and a message indicating the version and revision number of the DATASHARE system being used will be displayed, a wall-clock time display will begin running in the upper right part of the screen and one or more columns of digits will be displayed on the console screen.

These digits denote one message line which is allocated for each available port. The CHAIN instruction displays the name of any program it invokes on this line. A program may also use the line assigned to the port it is running on for operator messages by using the CONSOLE instruction. These lines are useful for informing any operations personnel of the status of the system. An asterisk just to the right of the port number on the console screen will be displayed (if the system console is configured) whenever the Carrier Detect signal for that port is present.

The final step in the initialization sequence is to load the main overlay (which lives on top of the initialization code), the foreground overlay (if foreground is in upper memory), and any MULTILINK or ACALL programs.

If the ACALL facility has been configured to be non-dynamic, a check for the presence of the file DS%/ASM (the user-written ACALL object code module) is made; if it does not exist, the message:

/ASM OR /COM MODULE MISSING OR UNLOADABLE!

will be displayed and the initialization aborted.

If external communications have been configured, a check for the presence of the file DS%/COM (the MULTILINK line driver

program) is made; if it does not exist or is invalid, the message:

/ASM OR /COM MODULE MISSING OR UNLOADABLE!

will be displayed and the initialization aborted.

The initialization sequence then loads the main overlay (and if foreground is in upper memory, the foreground overlay). If either overlay can not be loaded, the message:

MAIN/FG OVERLAY LOAD FAILURE!

will be displayed and the initialization aborted.

For details concerning the initialization of Slave Stations, consult the DSSLAVE User's Guide (Model Code # 50537).

4.2 Command Line Options

4.2.1 Time and Date Initialization

If DATASHARE VI is configured to run under ARC, it will initialize its internal time and date values to whatever time and date can be obtained from the ARC network. If, however, DATASHARE VI is not configured to run under ARC, or if a valid ARCCLOCK/TXT is not available on any MOUNTed ARC volume, then the time and date may be initialized on the command line by the use of the following options:

DS6 ;T=hhmm,D=ddd/yy

In the above options, the time (T=) is assumed to be a four-digit, 24-hour value and the date (D=) is a three-digit, zero-filled Julian date followed by a slash (/) followed by the last two digits of the current year. Any number of spaces and/or commas may be used to separate the T=hhmm and D=ddd/yy options from each other and any other options the user desires to enter. For example, to set the time to 1700 hours on the first day of February, 1979, the following line might be entered:

DS6 ;T=1700,D=032/79

Specifying the time and date on the command line will override initialization of the time and date from ARC.

4.2.2 Restarting After a ROLLOUT

Whenever a ROLLOUT instruction is performed by DATASHARE VI, a file is produced with the default name ROLLFILE/SYS (and in a large memory environment, another file default named ROLLFILE/PWS). These files contain all the information necessary to restart the Interpreter from a point immediately after the ROLLOUT instruction. In order to restart DATASHARE VI from these files, the option ";R" must be entered on the command line after the Interpreter's name, i.e.

```
DS6 ;R
```

This command will cause the message

```
DATASHARE VI 1.n <date> ROLLOUT RETURN
```

to be displayed on the console and the ROLLFILE/SYS file (and the ROLLFILE/PWS file) will be read into memory to perform the restart. (The time and/or date may be initialized by the T and D options during rollback as mentioned above in Section 4.2.1.) After the ROLLFILE(s) have been loaded, execution of DATASHARE VI will continue exactly as if the ROLLOUT instruction had caused no action to be taken.

Note: The file ROLLFILE/SYS (and ROLLFILE/PWS) are allowed to be renamed by a parameter on the command line (See Section 4.2.6).

4.2.3 Verifying Configuration

The configuratin of the interpreter can be verified by entering the command line:

```
DS6 ;V
```

This will cause the interpreter to display, during initialization, the configuration of the interpreter. Holding down the display key will cause the configuration display to pause.

4.2.4 Special Port 1 Answer Program Feature

An alternate ANSWER program name can be specified for port 1 by entering the command line:

DS6 <program>

where <program> is the name of a DATABUS object code file. If this action is taken, the file <program>/DBC will be used for the ANSWER program for port 1 instead of ANSWER1/DBC; the MASTER program for port 1 will still be MASTER1/DBC (see Chapter 5, ANSWER AND MASTER CONCEPTS, for a more thorough explanation.)

4.2.5 Using an Alternate DATASHARE Program Library

If the operator desires to use a DATASHARE program library named other than DATASHAR/DBL, he may enter its name as the second parameter on the command line:

DS6 [<program>],<DS program library>

This will cause the Interpreter to use the named library in place of DATASHAR/DBL during all CHAINing operations. The default extension for the program library is /DBL.

4.2.6 Using an Alternate ROLLFILE

If a ROLLFILE named other than ROLLFILE/SYS is desirable (for instance if two DATASHARE systems utilize a common File Processor under ARC,) a third command line parameter may be used:

DS6 [<program>][,<DS program library>][,<ROLLFILE name>]

Use of this parameter will cause the Interpreter to create a file <ROLLFILE name>/SYS (and in a large memory environment, the file <ROLLFILE name>/PWS) for ROLLOUT instead of the file(s) ROLLFILE/SYS (and ROLLFILE/PWS). In addition, if this parameter is selected along with the ;R (rollback) option, the Interpreter will use the <ROLLFILE name>/SYS (and <ROLLFILE name>/PWS) for rolling back (and will, of course, re-use the <ROLLFILE name> file(s) if a subsequent ROLLOUT is performed).

4.3 Taking Down the System

The DATASHARE VI system maintains its files under total control of the DOS. The DOS normally may be restarted at any time without detriment to the file structure. However, restarting 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. Properly closing a file allows DOS to deallocate any unused allocated file space. Thus, to insure that all files are properly closed, the system should only be restarted when all ports are idling in their MASTER programs. (A port is idling when it displays the MASTER program name on the system console.) Neither the Central nor the Remote Slave Stations should be restarted while Remote Slave Stations are on-line.

For details concerning the session termination of Slave Stations, consult the DSSLAVE User's Guide (Model Code # 50537).

4.4 Special Considerations

If a DATABUS program is to be run in a single-port environment, the program should be executed using the DATASHARE VI Interpreter configured for port-1-on-console as the only port, since throughput is much higher.

Write-buffering may be used to increase the speed of sequential writes by performing logical record WRITEs using a value of "-2" for the record specifier (see the DBCMPLUS Compiler User's Guide, Model Code # 50321, for usage of the WRITE statement).

If write-buffering is used, the DATABUS disk I/O instruction will not immediately write the sector to disk; typically the sector is merely flagged as "pending" a write to disk. The actual write to disk occurs under one of the following conditions:

a) If the user program performs a "SHUTDOWN", "ROLLOUT", or an explicit "CLOSE" of the file.

b) The virtual storage buffer containing the write-pending sector becomes the least-recently-used buffer and is required for some other purpose.

The consequences of this are as follows:

a) The DATASHARE VI Interpreter may not discover that a disk

has gone off-line until one of the above conditions occurs.

b) The DATASHARE VI Interpreter may not discover that the disk sector destined to receive the buffer has irrecoverable bad parity until one of the above conditions occurs.

c) Data contained in write-pending buffers will be lost if the processor is restarted before the physical write is performed.

Thus, a trap set for a parity (P) or disk (I*M) off-line condition during a "WRITE" instruction, for example, may not actually be taken until many instructions later. The error message generated may not have a program counter address pointing to the "WRITE" instruction responsible, but to an unrelated instruction much further along in the program.

If it becomes necessary to restart and it is critical that write-pending buffers be written, the DOS may be restarted by executing the ROLLOUT or SHUTDOWN instruction. For example:

```
ROLLOUT "FREE"
```

or

```
SHUTDOWN "FREE"
```

4.5 Fatal Error Conditions

Some error conditions within the DOS 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, then displays an error message in the standard DOS format and returns control to the DOS command interpreter. Since DATASHARE VI does not overlay any part of the DOS that is loaded by IPL, return to the DOS will be safely accomplished after the display of the abort message. See DOS User's Guide for the error messages and their causes.

CHAPTER 5. 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. The ANSWER program deals with the user when he initially connects to the system; i.e., dials in on the telephone or turns on his direct-wired terminal. The second program is called the MASTER program. The MASTER program interfaces with the user whenever he is not executing the ANSWER program or an application program. The MASTER is generally written to allow the user to select the next application program he wishes to execute. Note that both ANSWER and MASTER are written in DATABUS, enabling a user to tailor the command aspects of the DATASHARE VI system to suit his particular needs. (Simple examples of ANSWER and MASTER programs are shown in Appendix H.)

5.1 System Security

The ANSWER concept allows the DATASHARE system to force users to give some type of identification before they are allowed to use the system. 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 his port and the first CHAIN instruction executed by the program for that port. In other words, while the user is executing the ANSWER program for his port (as he first signs onto the system or later CHAINs to the ANSWER program,) he will not be allowed to escape around any identification request and go directly into the MASTER program by simply striking the INTERRUPT key.

The ANSWER program should set traps for all of the trapable errors (IO, PARITY, RANGE, etc.) to prevent an untrapped error from allowing the user to escape around the identification request. If an untrapped error should occur during the execution of the ANSWER program, the MASTER program will begin execution.

The ANSWER program may also be structured to enforce file access limitations depending upon the identification of the user.

5.2 System Convenience

The ANSWER program chains (via a STOP instruction) to the MASTER program, which usually requests the name of the program the terminal operator wishes to execute. This name is generated from information supplied by the terminal operator. For example, the operator might enter the number of a data-entry form; the MASTER program would then decide which program to execute for that form number. The DOS directory cannot be directly accessed by the MASTER program, implying that if a directory service or file access limitation is to be implemented, a file must be generated which contains the names of programs and files that are to be accessed. It is the responsibility of the author of the ANSWER and MASTER programs to provide any such convenience facilities to the terminal user.

5.3 ANSWER and MASTER Programs

The ANSWER and MASTER programs usually perform tasks which offer convenience features to give the system the appearance of a conventional time-sharing system.

If the DATASHARE object file for either the ANSWERn or ANSWER, or MASTERn or MASTER programs do not exist, port n will not be activated when the system is brought up. If a command line specified ANSWER program for port 1 is given but does not exist, then DATASHARE VI will abort during initialization and give the following message:

PORT 1 ANSWER PROGRAM MISSING

COMMON data areas used in the MASTER program should be used with caution; the MASTER program may be entered due to a program trap or the INT key being struck. If programs which have been CHAINed do not preserve this data area, the MASTER program will become confused as to the contents of the COMMON data.

When the ANSWER program is activated it executes until a KEYIN or DISPLAY statement is encountered (the ANSWER program will be activated whenever a given terminal disconnects from the system, not when it connects to it). If, say, the time of connection, disconnection and total connection time are being kept in a file, the ANSWER program's first operation might be to note when the user disconnects from the system and to log the total amount of time he was connected. Then, perhaps, a KEYIN statement requesting a new user identification would be issued and cause execution to cease for that port.

*NOTE: Since the ANSWER program will execute until the first KEYIN/DISPLAY statement, any ANSWER programs which use the DIAL verb MUST have the DIAL statement prior to any KEYIN/DISPLAY statements.

For example: A log-out function is executed when a terminal disconnects from a system. When the terminal again re-connects to the system and the user at the terminal enters an identification code, the previously-mentioned KEYIN statement would be satisfied and the new user would then be logged on with his sign-on time being recorded in a log file.

When a system is initialized, all ports will appear to be logging off (since all ANSWER programs are executed,) but no corresponding log-on times will be set. The ANSWER program must therefore handle this special case by allowing for log-offs without corresponding log-on times.

Appendix H contains examples of both ANSWER and MASTER programs.

CHAPTER 6. PHYSICAL SYSTEM CHARACTERISTICS

6.1 Virtual Memory

In order to achieve a reasonable amount of program space for many simultaneously executing programs, DATASHARE VI employs a virtual memory manager. DATABUS code is very compact, with a few bytes of object code being capable of evoking a large amount of processor activity; the rate at which DATABUS program bytes are fetched is therefore very low. This low rate allows the program code bytes to be kept in randomly-accessible memory buffers; sequentially executing program instructions thus have very little effect on program execution speed. Another characteristic of DATABUS object code is re-entrancy; since this code is never modified, it never has to be written to disk. In addition, ports which are executing the same program can share the program code pages.

Program (user's) data, however, is accessed at a very high rate and must be in main memory to be effectively available to the DATASHARE VI Interpreter. For this reason all program data is kept resident in main memory (which is divided among the ports configured); this has further advantages in the case of port/printer I/O, as will be mentioned later.

In order to facilitate an effective virtual memory accessing algorithm, the program code is kept on the disk in 250-byte pages. Whenever a byte of DATABUS object code is fetched by the Interpreter, a check is made to determine if the page of code containing the byte is immediately available or if it must be read from the disk. Any time a page boundary is crossed, a new page must be read in if it is not already in the buffer. The time required to read a page of code from disk, in order to fetch a byte, is much greater than the fetch time for a byte in a page already in a buffer. The DATABUS programmer can, in general, increase his code's execution speed by forcing his program to cross as few page boundaries as possible. Actually, in a lightly loaded system, a single program could have a number of pages all resident in memory buffers at once; crossing a page boundary would probably not cause a disk read, but any significant increase in loading would nullify this ideal condition. The DATABUS programmer should therefore assume that any time his program crosses a page boundary a disk read will occur and cause a delay

in the execution of the program; the time lost cannot be used by another program since the disk is busy. By causing an excessive number of page boundary crossings, a programmer can easily cause his program (and others) to execute very slowly.

The instruction TABPAGE exists in the DATABUS language to aid a programmer in making his program's execution speed as high as possible. During compilation, 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 was 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, a programmer can cause logical parts of his program to cross as few page boundaries as possible.

6.2 Scheduling

In order to provide optimum response time, DATASHARE VI handles all port and printer I/O using interrupt driven foreground routines. Data transfer between the devices and the system can therefore occur regardless of the computational task being handled by the background program(s) at any given time. The foreground routines interpret the KEYIN, DISPLAY, and PRINT instructions, while the background interpretive code merely passes these instructions to the foreground via a circular buffer allocated for each user. Conventional systems use such a buffer to hold the actual characters transferred between the system and the device. DATASHARE VI uses this buffer to hold interpretive code bytes which enables many more bytes of information to be transferred than can be held in the buffer.

For example, a DISPLAY statement may contain some text information followed by a variable name. The variable name would be represented by two bytes, but the contents of the variable might be fifty bytes long, enabling two bytes of buffer space to cause the transfer of fifty bytes to the device. This is made possible by the fact that all program data is resident in main memory and thus is commonly accessible to both foreground and background.

The foreground and background routines for a given user always execute exclusively of each other to prevent conflicts over data values. When the background routine executes a DISPLAY statement, the statement is stored in the buffer for the user; the background routine then relinquishes control to the foreground routine. When foreground has completely executed the I/O statement, it causes a high priority interrupt in background,

resulting in a task-switch to the program executing the DISPLAY statement.

The above discussion concerns only port and printer I/O. All disk I/O is performed under the DOS, which is a background only operation; e.g. all DOS activities are non-interruptible (by the DATASHARE background scheduler, not the machine interrupt driven foreground routine.) Long directory searches (which may take up to several seconds with multi-drive systems) will cause poor response to I/O completion interrupts. However, lengthy DOS activities occur infrequently and may be ignored from an average response time standpoint.

DATASHARE VI printing is performed in both background and foreground. The background sets up a line image in a 136 position buffer; this buffer is transferred to a larger circular buffer whenever vertical paper motion is necessary. The circular buffer is in turn emptied by a simple foreground process. Background execution is suspended only if the buffer becomes full; a program may therefore complete a number of print statements without experiencing a delay.

When the background process resumes execution due to the completion of a foreground I/O task, it is guaranteed a minimal 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.

6.3 Terminal devices

DATASHARE VI is capable of driving many non-Datapoint serial terminal devices which use the 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 compatible. The *ES and *EL list controls send control characters which are ignored by a model 35 ASR Teletype. 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 model 35 ASR Teletype.

DATASHARE VI is also capable of handling 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 3360, 3600, or 3670 hard-wire cable is connected in such a way as to give the 3360, 3600, or 3670 the appearance of a 103 data set; power-on causes Ring-Detect and Carrier-Detect to be sent to the DATASHARE system. The use of a hard-wire or dataset connection employed at a given terminal cannot be differentiated by the DATABUS programmer.

CHAPTER 7. PHYSICAL INSTALLATION

7.1 Main peripherals

The DATASHARE system requires a disk peripheral or one or more logical drives MOUNTed under ARC; since the system maintains its entire file structure under the DOS, many logical drives may be available.

*NOTE: Attempting to change mounted volumes while DATASHARE VI is running or rolled-out, may produce indeterminate results. It is not advisable to attempt changing the mounted volumes.

In addition to the disk, a 9462 or 9320 Multiple Port Communications Adaptor (MPA) is required if 3360/3600/3670 ports are to be used. 9402 or 9481 Adaptors are required for every Slave port. The 9462 MPA is capable of driving up to eight fully independent full-duplex asynchronous lines at speeds ranging from 110 to 9600 BAUD (all ports are operated by the DATASHARE system in full-duplex mode only); up to three 9462 MPAs may be used in a system configuration. The 9320 MPA is capable of driving up to four fully independent full-duplex asynchronous lines at speeds up to 9600 BAUD each; up to two 9320 MPAs may be used in a system configuration.

The 9320 multi-port adaptor is software strapped the DATASHARE VI system. The default transmit and receive baud rates are 9600 baud. Alternate baud rates can be configured for each port on a 9320 multi-port (See section 7.3.2).

The 9462/9320 MPAs should be strapped to the maximum baud for the terminal connected (4800 baud for 3360s and 9600 baud for 3600s).

*NOTE: The 9320 MPA is also refered to in DATAPOINT documentation as a Integral Four Port Communications Adaptor (IFPCA).

DATASHARE VI is normally configured at 300 BAUD input and dynamic output for direct connection of 3360 and 3600 terminals. DATASHARE uses 1200 BAUD for four wire 202-type modem connections and uses up to 9600 BAUD for 103-type modem connections although any speed may be strapped in the 9462 to achieve compatibility

with specific modems. The actual effective BAUD rates for each port are dependent on system configurtaion and load. The total aggregate BAUD rate of all ports cannot exceed the maximum allowed in the configuration (see the table given in section 1.2)

DATASHARE VI supports up to four ports on a 9462 MPA connected to an 1170/1810/1820/3810/3820 processor. The ports must be physically connected into the first four slots of a 9462 MPA which has a standard MPA I/O address of 0151 and the 9462 must be connected to a 9022 Auxiliary Power Supply on the 1810/1820/3810/3820 processors.

Up to eight terminals (9 using port-1-on-console) are supported by DATASHARE VI on two 9320 MPAs on an 1810/1820 processor. The terminals must be physically connected into the four slots of each of the 9320 MPAs. The 9320 MPAs are powered from the 9320 disk controller and needs no external power supply. The first 9320 MPA must have a microbus address of 4 and the second an address of 5.

Up to sixteen ports are supported by DATASHARE VI on a 6010/5500/5000 processor. Ports 1 through 8 must be connected to a 9462 MPA which has a standard MPA I/O address of 0151 while ports 9 through 16 must be connected to a second 9462 MPA strapped for an I/O address of 055.

DATASHARE VI supports a maximum of twenty-four ports on a 6600/6020/6040 processor. Ports 1 through 8 must be connected to a 9462 MPA which has a standard MPA I/O address of 0151, ports 9 through 16 must be connected to a second 9462 MPA strapped for an I/O address of 055, and ports 17 through 24 must be attached to a third 9462 strapped for 053; attachment of the third 9462 requires that a 9022 Auxiliary Power Supply be configured into the I/O bus prior to the third 9462.

As in any Datapoint installation, a 9420 parallel interface strapped for the address of the configured printer address may be connected to drive a special output device, but the device must be capable of handling any output that would normally be given to an ASCII printer.

7.2 Terminal connections

In general, non-Slave terminals may be connected to the DATASHARE system in one of three ways: by direct hard-wire, 103-type modem, or 202-type modem. The following table shows the pin assignments of the connectors for a 9462/9320 Multiport Adaptor, the 3360 and 3600 CRT terminals, and a 103 or 202 type modem:

PIN	9462/9320	3360	3600	9408/9409 9478/9479 103/202
1	-	PROT GROUND	PROT GROUND	PROT GROUND
2	DATA OUT	DATA OUT	DATA OUT	DATA IN
3	DATA IN	DATA IN	SIG GROUND	DATA OUT
4	REQ TO SEND	-	DATA IN	REQ TO SEND
5	CLR TO SEND	-	-	CLR TO SEND
6	-	-	8x CLOCK	DATA SET READY
7	SIG GROUND	SIG GROUND	-	SIG GROUND
8	CARRIER DET	-	RECV ONLY DATA	CARRIER DET
* * *				
12	-	-	DATA TERM RDY	-
* * *				
20	DATA TERM RDY	DATA TERM RDY	-	DATA TERM RDY
* * *				
22	RING DETECT	-	-	RING DETECT

The DATASHARE system executes the following handshaking procedure when connection to a 3360, 3600, or 3670 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 ten 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.

7.2.1 Direct

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:

9462/9320 TO 3360/3600/3670 CABLE CONNECTIONS

9462/9320	3360	3600/3670
2	3	4
3	2	2
7	7	3
8, 22 (and 5)*	20	12

* Note: This optional connection causes the port's power-on indicator to be wired to the CTS input of the Multiport Adaptor and will hold CTS true as long as power is applied to the terminal. This wire should be added to all standard 3360/3600/3670 terminal connections if an Interpreter configured with the CTS-wait option is to be used. This connection must be made for proper operation of terminals on a 9320 Multiport Adaptor and for 3670 terminals.

This arrangement requires only four wires in the cable. All cables should be shielded, regardless of length. If the cable is to be more than several hundred feet long, each of the two signal wires (the ones connected to pins 2 and 3) should be twisted separately with a ground wire. Direct connections up to one thousand feet may be made if the above precautions are followed.

DATASHARE also supports 9292 Serial Printers connected to 3600 terminals. 3600 terminals with 9292 printers are connected as follows:

9462/9320 TO 3600 TO 9292 CABLE CONNECTIONS

9462/9320	3600	9292
2	4	-
3	2	-
5 *		17, 36 *
7	1	-
-	5	41
-	6	46
-	7	40
-	8	45
8 and 22	12	-

* Note: This optional connection wires the off-line indicator of the serial printer to the CTS input of the Multiport Adaptor for the specialized use of CTS-wait to detect the off-line or paper-out printer condition.

The 3360/3600 sets Data Terminal Ready whenever it is powered-on. With the above cable connected, this will cause Ringing and Carrier to be presented to the 9462/9320. This has the effect of causing the ANSWER program to be executed whenever power turned off on the 3360/3600. The 3670 will also work properly with the above cable, but the EM3270 User's Guide (Model Code # 50486) should be consulted for a complete description of the power-up sequence performed.

7.2.2 103-Type Modem

The 9462/9320 can be connected to a 103-type modem with a one-to-one cable (i.e., 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.

7.2.3 9462/9320 to 103-Type Modem Connections

9462/9320	103-TYPE MODEM
2	2
3	3
7	7
8 (and 5*)	8
20	20
22	22

*Note: This optional connection is required when "wait for CTS" is configured. This is done only to indicate to DATASHARE VI that the port is write-ready. This signal does NOT get sent through the modem, and can NOT be used to control the terminal.

If one is calling a 103-type modem over a dial-up network, he should hear the telephone answered very shortly after it begins ringing (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 operational. In any

case the caller should hang up (letting the phone ring for a long time may be very irritating at the other end). If the telephone is answered, the caller should hear the carrier from the modem connected to the 9462/9320; this 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 before the ANSWER program begins execution. If all is not satisfactorily completed, the DATASHARE system will hang up the telephone at its end and resume waiting for ringing to occur. Since the DATASHARE system waits up to ten seconds for a satisfactory connection, if one dials the system and hangs up as soon as the telephone is answered, he must wait ten seconds before he can dial the same telephone number again. In addition, 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.

7.2.4 202-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 leased circuit. This means that one signal path must exist for data flow in one direction and a separate data path must exist for data flow in the other direction. This implies that point-to-point connections must be made between the modems (a 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 9462/9320 and 202 modem is similar to the one used for connection to a 103-type modem, except 202's used in point-to-point four-wire service do not use ringing; the Carrier Detect signal from the 202 must be connected to both the Carrier detect and Ring detect inputs on the 9462/9320.

9462/9320 TO 202 MODEM CONNECTIONS

9462/9320	202 MODEM
2	2
3	3
4	4
7	7
8, 22 (and 5*)	8
20	20

*Note: This optional connection is required when "wait for CTS" is configured. This is done only to indicate to DATASHARE VI that the port is write-ready. This signal does NOT get sent through the modem and can NOT be used to control the terminal.

When Data Terminal Ready is supplied by the terminal device to the remote 202 modem, that modem will turn on its carrier; this causes the modem connected to the 9462/9320 to turn on its Carrier detect signal which will present Ring detect and Carrier detect to the DATASHARE system. The system will set its Data Terminal Ready signal, causing the 202 modem to turn on its carrier and complete the connection; one second later the ANSWER program will begin execution. Operation over a 202 modem connection will thus appear to be similar to direct connection operation.

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

*NOTE: Use of a 9408/9409 modem requires a DAA. The following DAAs can be used with these modems: 1001A, 1001F, 9445, or 9446. Note that the 1001A and 9445 have NO auto-answer capabilities, hence they are not recommended for use at the central site. The 9478/9479 modems do not require a DAA as they have internal DAAs.

7.3 Multi-Port Adaptor Configuration

The 9462/9320 Multiple Port Communications Adaptor is software programmable to transmit and receive from five to eight information bits with either one or two stop bits. The DATASHARE system always uses one start bit, eight information bits, and sends two stop bits; however, it will receive signals with only one stop bit.

7.3.1 Port speed strapping on 9462

The speed of each port connected to a 9462 MPA may be set independently to a variety of speeds, depending on field-programmable hardware straps. There are three clock buses within the 9462, 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 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 printed circuit card in the 9462:

BAUD RATE SOURCE

Baud rate	Pin
150	E43
300	E44
600	E47
1200	E45
2400	E48
4800	E49
9600	E50
110	E51
220	E52
440	E46
880	E53
1760	E54
3520	E55
7040	E56

BAUD RATE BUS

Bus	Input	Output
1	E16	E19-E26
2	E17	E27-E34
3	E18	E35-E42

CHANNEL BAUD RATE INPUT

Channel	Input/Output
1	RRC1/TRC1
2	RRC2/TRC2
3	RRC3/TRC3
4	RRC4/TRC4
5	RRC5/TRC5
6	RRC6/TRC6
7	RRC7/TRC7
8	RRC8/TRC8

A typical installation may use BAUD rates of 110 for Teletype machines (remote or local), 300 for remote 3360/3600 terminals using 103-type modems, and 1200 for remote 3360/3600 terminals using 202-type modems. For such an installation, one might connect bus 1 for 110 BAUD, bus 2 for 300 BAUD, and bus 3 for 1200 BAUD as shown in the following table.

E16 to E51	Make bus 1 110 BAUD
E17 to E44	Make bus 2 300 BAUD
E18 to E45	Make bus 3 1200 BAUD

At this point, 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:

RRC1/TRC1 to E19	Make ch 1 300 BAUD
RRC2/TRC2 to E20	Make ch 2 300 BAUD
RRC3/TRC3 to E21	Make ch 3 300 BAUD
RRC4/TRC4 to E27	Make ch 4 1200 BAUD
RRC5/TRC5 to E28	Make ch 5 1200 BAUD
RRC6/TRC6 to E29	Make ch 6 1200 BAUD
RRC7/TRC7 to E30	Make ch 7 1200 BAUD
RRC8/TRC8 to E35	Make ch 8 110 BAUD

7.3.2 Port Speed Strapping on 9320

The 9320 internal multi-port adaptor is software strapped to default transmit and receive baud rates of 9600 baud. Alternate baud rates can be configured for each individual port on the 9320 multi-port by modifying the chain file produced by DS6/TXT.

In order to configure the alternate baud rates, the DS6 configurator should be run with the following command line:

```
CHAINPLS DS6;OP=C
```

This will cause CHAINPLS to produce a DS6/CHN file containing the commands for the LINK program to link the interpreter. If the interpreter has been configured to use the 9320 multi-port adaptor, a series of MEMORY statements will be found in the root module link which define the default baud rates.

These MEMORY statements reference eight external labels in the root module. These labels have the format:

```
MPmp9320  (m=multi-port #, m=0,1; p=port #, p=0-3)
```

Selection of an alternate transmit or receive baud rate for a particular port is accomplished by changing the corresponding MEMORY statement to the desired value indicated by the following chart.

NOTE: The transmit and receive baud rates are independent. The transmit baud for a given port can be configured for 9600 baud while the receive baud for the same port is configured for 150 baud. The value given in the memory statement is a composite of BOTH the transmit and receive baud rates.

		Transmit Baud Rate						
	XXXX	9600	4800	2400	1200	300	150	110
Receive Baud Rate	9600	0210	0230	0170	0270	0330	0350	0370
	4800	0211	0231	0171	0271	0331	0351	0371
	2400	0207	0227	0167	0267	0327	0347	0367
	1200	0213	0233	0173	0273	0333	0353	0373
	300	0215	0235	0175	0275	0335	0355	0375
	150	0216	0236	0176	0276	0336	0356	0376
	110	0217	0237	0177	0277	0337	0357	0377

7.4 Non-standard terminal devices

Terminals other than the Datapoint 3360/3600/3670s and Remote Slaves may be successfully connected to the DATASHARE system. Terminals such as the Teletype 33 and 35 KSR or ASR may be connected via hard-wire or 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. Non-terminal devices such as badge readers and OCR wands can be connected if they meet the interface requirements (RS232) of the 9462/9320. Note that all devices attached must comply with normal DATASHARE operating practices. They must not, for example, allow continuous transmission or transmission at higher rates than DATASHARE can be configured to accept.

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 9462/9320 are connected to a wire that is logically "high" when the connection is to be established and is logically "low" when the connection is to be broken. In addition, the CTS connection must also be logically "high" during connection if a 9320 MPA is used.

7.5 Telephone lines to Slaves

Every Slave port telephone line at the Central site uses a 9402 and 1001D DAA or a 9481 which is attached directly to the I/O bus. A 9402 and 1001D DAA or a 9481 is also required at each Remote Slave site.

Except for 1810/1820/3810/3820 processors, a maximum of two Communications Adaptors can be powered from any Datapoint processor (usually two 9462's or one 9462 and one 9402). Additional 9402's must be powered by either the 9450 or 9455 adaptor trays or a 9022 Auxiliary Power Supply. The 1810/1820/3810/3820 processors have no internal power supply for a 9462 or a 9402; a 9022 Auxiliary Power Supply must be installed to meet their power requirements. The 9481 Communications Adaptor is self-powered and needs no 9022 Auxiliary Power Supply.

No two 9402's may have the same address at the Central site. Central site 9402 addresses for Asynchronous communication may be any one of the following:

322 (octal)	146
350	145
344	143
342	066
330	065
324	063
154	056
152	

Asynchronous Remote Slave site 9402 address should be 0322 (octal).

No two 9481s may have the same address at the Central site. Central site 9481 addresses for Synchronous communication may be any one of the following:

321 (octal)	324
314	154
312	152
311	146
330	145

Synchronous Remote Slave site 9481 address should be 0321 (octal).

CHAPTER 8. PROGRAMMING CONSIDERATIONS

8.1 AUTO-START

The DATASHARE VI interpreter can be forced to automatically execute when the DOS is brought up by the use of the AUTOKEY/CMD program. Due to the use of an overlay library scheme, the <interpreter-name>/CMD file contains all overlays necessary to run DATASHARE VI.

If AUTOKEY is set to <interpreter-name>/CMD and the AUTOKEY/CMD program is AUTOed, the DATASHARE VI interpreter will automatically start up whenever the DOS is booted. The AUTO program should not be set to execute <interpreter-name>/CMD directly as the interpreter will scan the command line (MCR$) to determine if any options were entered by the user. MCR$ is untouched by the AUTO program and might contain invalid information leftover from another operation.

8.2 ROLLOUT and SHUTDOWN

ROLLOUT and SHUTDOWN are Central Station facilities only. A DATABUS ROLLOUT or SHUTDOWN statement will not perform the rollout function if any Remote Slave ports are on-line to the Central Station (i.e. they are dialed in). A ROLLOUT or SHUTDOWN will also not be performed if External communications are active and the MULTILINK driver program in use fails to suspend communications voluntarily within ten seconds after the initial request. If the ROLLOUT or SHUTDOWN function cannot be performed, control is returned to the DATABUS program with the OVER flag set (the user may then decide to take alternative action until all Slave ports have disconnected from the system). A ROLLOUT or SHUTDOWN request is not queued; a DATABUS program denied a ROLLOUT or SHUTDOWN request must execute another ROLLOUT or SHUTDOWN statement at a later time if DOS activation is to occur.

If a ROLLOUT or SHUTDOWN instruction is executed, and these features are not configured in the Interpreter, the OVER flag will be set.

*NOTE: When running on an 1810/1820 processor utilizing a 9320 internal multi-port, the power to the multi-port MUST remain

on while the interpreter is rolled-out. If the power to the multi-port is shut off while the interpreter is rolled-out, the rollback will NOT perform correctly. This applies only to the 9320 internal multi-port, not the 9462 multi-port.

8.3 ROLLOUT and CHAIN (or CHAINPLS)

It is possible to run DATASHARE VI from the CHAIN (or CHAINPLS) utility. To return from DATASHARE VI to the CHAINing process, a ROLLOUT must be performed with the following format:

```
ROLLOUT "CHAIN NULL"

    or

ROLLOUT "CHAINPLS NULL"
```

where the file NULL/TXT is assumed to be an empty text file. This will cause the CHAIN (or CHAINPLS) program to be loaded into memory and the empty text file to be CHAINed, performing no action but a return to the previous level of CHAINing where execution will resume after the last command issued.

*NOTE: Nested CHAINing (recursive calls to CHAIN (or CHAINPLS)) will not work properly after a ROLLOUT if the command:

```
ROLLOUT "CHAIN/OV1"

    or

ROLLOUT "CHAINPLS/OV1"
```

is issued.

8.4 OPEN

Execution of the OPEN instruction for an IFILE or AFILE will result in the following search pattern for the index and text file. The index file is opened first using the drive direction given with the file name. If no drive direction is given, an all drive search will be used. Once the index file has been opened, DATASHARE VI will search for the text file. The search will begin on the drive containing the index file. If the text file can not be found on the same drive as the index file, an all drive search will be used.

8.5 CLOSE

Execution of the CLOSE instruction will cause the logical file (FILE/IFILE/AFILE/RFILE/RIFILE) to be closed. This causes all pages remaining in the virtual storage buffers which belong to the closed file, to be written to disk.

If the file specified in the CLOSE instruction was specified in a FILEPI instruction while the file was open, DATASHARE VI assumes that the file is a shared file and will not invoke the DOS system CLOSE function for the file. Thus any space allocated to the file that is not used will not be deallocated. In this manner shared files may be closed resulting in the purging of all buffers from the DATASHARE VI virtual storage manager without the possibility of losing some of the information written to the file. Unless the DATABUS programmer is using the FILEPI instruction to tell DATASHARE that the file is shared, he should not attempt to re-use a logical file entry in a shared file environment since re-using the logical file entry is equivalent to executing a CLOSE instruction for the first file prior to opening the second file. The following pair of examples will result in identical action being taken by DATASHARE:

```
FILE1       FILE
FNAME1      INIT  "FILE1/TXT"
FNAME2      INIT  "FILE2/TXT"
.
            OPEN   FILE1,FNAME1
            ...
            OPEN   FILE1,FNAME2
```

```
FILE1       FILE
FNAME1      INIT "FILE1/TXT"
FNAME2      INIT "FILE2/TXT"
.
            OPEN FILE1,FNAME1
            ...
            CLOSE FILE1
            OPEN FILE1,FNAME2
```

8.6 Code Page Flushing Under ARC/UPS

When running under ARC or UPS, the DATASHARE VI Interpreter will flush out all buffers, prior to execution of the program, belonging to a DATABUS program from its virtual buffer pool only if the DATABUS object code file is free-standing and not write protected. If the program object file is in a library, or is write protected, the buffers will not be flushed. No buffer flushing is performed when running under DOS since the DATABUS program can not be changed while the interpreter is running.

8.7 PI and FILEPI

Although both the PI and FILEPI instructions are supported by DATASHARE VI, it is recommended that the FILEPI instruction be used. Use of the FILEPI instruction allows the DATABUS programmer to CLOSE shared files when they are no longer needed (See section 3.5). The PI instruction, since it does not specify the files involved, does not support CLOSEing shared files.

The use of the PI instruction under ARC will result in decreased system performance since a general 32 drive enqueue must be requested for each PI issued. The FILEPI instruction avoids this decreased performance by requesting an enqueue only for the files specified in the file list.

If an OPEN/PREP sequence is to be enqueued, the PI instruction MUST be used since the FILEPI instruction requires all files in the file list to be OPEN.

```
FILE    FILE
FNAME   INIT            "FILE/TXT"
.
        TRAP            NOFILE IF IO
        PI              2
        OPEN            FILE,FNAME
        PI              0
        ...
.
NOFILE  PREP            FILE,FNAME
        RETURN
```

The effect of a PI or FILEPI instruction will be cancelled by the execution of any of the following instructions: PI 0, KEYIN, DISPLAY, BEEP, PRINT, CONSOLE, PAUSE, POLL, COMWAIT, CHAIN, STOP, DSCNCT, SHUTDOWN, ROLLOUT, or remote SLAVE operation.

8.8 Null ISAM files

If it is desired to build a null ISAM file for the purpose of creating the data file through DATASHARE, the index file must be created using a new INDEX utility. This new INDEX is available with the following (or later) versions of DOS: DOS.C 2.4.3, DOS.D 2.5.1, DOS.E 2.4.3, and DOS.G 2.5. If an older version of INDEX is used to create a null ISAM file, DATASHARE VI will flag the error by posting an I * W error when the file is initially opened. If an I * W error is posted, the file must be re-INDEXed using the new INDEX utility prior to use in DATASHARE VI.

8.9 RELEASE

Execution of the RELEASE instruction will set the OVER condition flag if the system printer is unavailable for use by the port executing the RELEASE, or if the printer is not configured.

Execution of the RELEASE will not affect an open SPOOL file's data; the RELEASE instruction may only be used to control the system printer. Specifically, if the INHIBIT EJECT option is inactive, no top-of-form character will be written into the SPOOL file as though the system printer's forms were being ejected; if this feature is desired, it must be accomplished in the DATABUS program(s) performing the SPOOLing.

8.10 DEBUG

The DEBUG instruction is treated as a NOP. That is, execution is continued with no "DEBUG" activity taken.

8.11 SLAVE STATIONS

Use of Slave stations in DATASHARE VI requires the program DSSLAVE. DATASHARE VI will not operate with DS5SLAVE.

DATASHARE Slave Stations will operate with DATABUS programs exactly as written for 3360 and 3600 terminals with the following exceptions:

1) Screen positioning in KEYIN and DISPLAY instructions is limited to 12 display lines unless the Slave is run on an 1810/1820/3810/3820 processor.

2) The new key TRAPs (INT, F1, F2, F3, F4, F5, and <character>)

do not work on Slave stations.

3) ROLLOUT and SHUTDOWN (Central Site facilities) cannot be initiated directly by a Slave station; however, inter-port communications may be used to submit a ROLLOUT or SHUTDOWN request to a local port for execution after Slave Stations have disconnected (see the DBCMPLUS Compiler User's Guide, Model Code # 50321, section on COMMUNICATIONS INPUT/OUTPUT for information on internal (inter-port) communications).

4) The new list controls introduced since DBCMPLUS 1.1. are not supported at the slave station.

Note: A DATASHARE port configured for Slave operations will execute the ANSWER program without waiting for the Slave to dial in. This is to accomodate the DIAL verb, which could not be executed otherwise. The ANSWER program will execute until it encounters a KEYIN, DISPLAY, RPRINT, BEEP, or other DATABUS instruction that requires the presence of a terminal device.

Four additional DATABUS instructions are used to operate the extended facilities of DATASHARE Slave Stations: The RFILE, RIFILE and RPRINT instructions are used to access the disk files and/or local printer at a Remote site. The DSCNCT instruction is used to disconnect the telephone line when processing is complete (see the DBCMPLUS Compiler User's Guide, Model Code # 50321).

When performing Remote operations (file accesses, RPRINTs, etc.), the DATABUS programmer should generally attempt (where possible) to group as much data together into as few strings as possible. Throughput will be greater if a DATABUS programmer groups twenty variables of five characters each into a single string of length 100 before performing a Remote operation.

8.12 PHANTOM Ports

Execution of a KEYIN, DISPLAY, or BEEP instruction from a PHANTOM port, will result in an untrapable error. If the port was in ANSWER or MASTER, the port is permanently disabled and, if the system console is configured, a message is displayed on the console screen indicating that the port has been disabled. If the port was not in ANSWER or MASTER, the port will abort and chain back to the MASTER program.

8.13 ACALL

DATASHARE VI supports the ACALL (Assembler language CALL) facility. This facility allows an assembly language program to be invoked from a DATABUS program. The implementation structure is that of a user-written overlay which is loaded by the DATASHARE VI Interpreter's initialization sequence if the ACALL facility is configured for "one-program" ACALL, (i.e. non-dynamic,) or loaded upon the demand of the DATASHARE program (i.e. "dynamic" ACALL). The user-written ACALL overlay processes the DATABUS ACALL instruction. If the ACALL facility is not enabled at all, ACALL instructions will be trapped to a bad-opcode ("B") error.

If the one-program method is used, the overlay must have the name <Interpreter-name>/ASM and must be available to the Interpreter during initialization. If the dynamic method is used, the overlay need not be present at initialization, the overlay may have any legal DOS name, and it may reside either as a stand-alone absolute object file, in a library created by LIBSYS, or in the <Interpreter-name>/CMD library (having been ADDed to the /CMD library by LIBSYS.

The naming convention observed during the calling of dynamic overlays is the same as is used during a CHAIN or OPEN, i.e. "NAME/EXT:VOLID.MEMBER" (this string is specified as the first parameter of the ACALL instruction). The default extension of an ACALL overlay is /ASM. If the special-case name of ".MEMBER" is given, the Interpreter will use the /CMD library as the default "NAME/EXT" to look up the ACALL object overlay. This special case name procedure should be used when the ACALL module's execution speed is critical; the Interpreter does not have to OPEN its own /CMD module as this was done at initialization, so considerable waiting for the DOS to perform an OPEN may be avoided.

In addition, if a dynamic ACALL module called by a user has already been loaded into the ACALL area by a previous ACALL and has not been overstored by an ACALL to a different module, the Interpreter will avoid re-loading the ACALL module.

*Note: This implies that a dynamic ACALL module must be reentrant and not overstore locations within the module area expecting them to be intact upon the next ACALL entry.

ACALL programs should follow the guidelines below:

a) The ACALL program must not modify the processor base register or the processor sector table.

b) The ACALL program should reduce the amount of time it spends with interrupts disabled to the smallest possible interval. Disabling interrupts for longer than 100 microseconds may produce indeterminate results.

c) Upon exit, the ACALL program should leave the processor stack at exactly the same level as it was upon entry. If more than 6 levels of stack are used in the ACALL program, the stack must be saved by the ACALL program.

d) Performing disk operations (READ$, WRITE$, POSIT$, etc.) in an ACALL program is not recommended. If, however, it is necessary to do so, the ACALL program MUST restrict the disk operations to using buffer page 0 and LF0. Use of buffer pages other than 0 and LFT entries other than LF0 is not supported and will produce indeterminate results.

The following sections describe the interface characteristics between the ACALL program and the DATASHARE VI Interpreter.

8.13.1 ACSTRT - Starting Location For ACALL Overlay

The address ACSTRT represents the first location of the memory reserved for the overlay which processes the ACALL instruction, the size of which is determined during the Interpreter's configuration.

8.13.2 GCMOP - Get The Next Operand Location

This routine is called to fetch the next operand from the list of operands specified in an ACALL DATABUS instruction. The routine is entered via an assembler "CALL" instruction and returns FALSE CARRY if the end of the operand list has been encountered. If an operand is found, the routine returns TRUE CARRY with the address of the first byte of the operand in the HL register pair. All other registers are indeterminate. The ACALL instruction Interpreter overlay MUST call this routine until it returns FALSE CARRY before executing a RETURN instruction itself.

8.13.3 Condition Code Routines

The following routines exist in the DATASHARE VI Interpreter to manipulate the DATABUS condition flags. They are used to return condition information to the DATABUS program. They are invoked by using the assembler "CALL" instruction. Upon exit from any of the following routines, the A, H, and L registers and the condition codes are indeterminate.

CLREQL - This routine clears the DATABUS EQUAL flag. Must be called with A-register = 0.

CLRLSS - This routine clears the DATABUS LESS flag. Must be called with A-register = 0.

CLROVR - This routine clears the DATABUS OVER flag. Must be called with A-register = 0.

CLREOS - This routine clears the DATABUS EOS flag. Must be called with A-register = 0.

SETEQL - This routine sets the DATABUS EQUAL flag. May be called with any A-register value.

SETLSS - This routine sets the DATABUS LESS flag. May be called with any A-register value.

SETOVR - This routine sets the DATABUS OVER flag. May be called with any A-register value.

SETEOS - This routine sets the DATABUS EOS flag. May be called with any A-register value.

8.14 EXTERNAL COMMUNICATIONS Facility

For users desiring to develop their own compatible communications line handler, the following sections describe the interface characteristics between the MULTILINK facility and the compatible communication line handler; (all the interface routines are invoked by using the assembler "CALL" instruction). It is important to note that unless otherwise stated, no hardware registers are preserved by these interface routines.

8.14.1 MULTILINK Programming Considerations

The MULTILINK line driver program may be used to drive any hardware device not normally supported by the DATASHARE Interpreter; these devices include, but are not limited to, communications devices, extra printers, magnetic tapes and card readers. Since communications devices are usually serviced by a MULTILINK routine, the term "line driver" will apply to all user-written MULTILINK programs. The line driver is a program that is serviced by the DATASHARE Interpreter in its foreground (time-critical) state and is entered at an entry point specified by the user-written program at intervals specified by the program. Routines to perform these functions are documented in the following sections. In general, the Interpreter links to the user-written program via an assembler CALL instruction; control is returned to the Interpreter either via the RETURN instruction or by one of the "change state" routines described below.

The basic data structures to allow the DATABUS language program to communicate with the line driver are two queues, the SEND queue and the RECV queue. Execution of a SEND instruction with the route variable indicating External communications places an entry (containing the information from the associated COMLST) at the end of the SEND queue; execution of a RECV instruction places the entry at the end of the RECV queue. Execution of a COMCLR instruction removes an entry from a queue. An entry's status in the queue may have one of the following values:

Pending - the entry has not yet been processed by the line driver; at this point the COMTST instruction will return the LESS condition.

In Process - the entry is being processed by the line driver; COMTST returns LESS.

Complete - the line driver has completed processing and has released the COMLST with a "normal complete" status; COMTST returns EQUAL.

Channel Unavailable - the line driver has completed processing and has released the COMLST with an "abnormal complete" status; COMTST returns OVER.

The line driver has two separate states: primary and secondary. The primary state is used for initial setup and "handshaking". While the line driver is in this state, all attempts to place an entry in the SEND or RECV queues will be rejected and the COMLST will have the status "channel

unavailable". While in the secondary state, the line driver can manipulate the data given to it in the SEND and RECV queues and can do useful work. The current state of the line driver is determined by the "availability" flag, set or cleared by routines described below. When this flag is cleared, the line driver is in the primary state; when set, it is in the secondary state. Upon initialization, the line driver is entered at the primary entry point (as specified at LINK time or by the END statement of the assembly) in the primary state.

8.14.2 CRCCF - Change Calling Frequency

This routine is called with a value in the C-register specifying the number of milliseconds between calls to the external communications process. This number will be retained as the initial value of a down-counter used by the scheduler. The counter will be decremented each millisecond; when the count reaches zero, it will then be reset to its initial value and the external communications process will be called. The default value is 1, causing the process to be called once each millisecond. This routine may be called in either the primary or secondary state.

*NOTE: The calling frequency has a direct effect on system overhead and, therefore, on system throughput. The calling frequency should be maintained at the highest value compatible with proper operation of the MULTILINK line driver in order to minimize system overhead. A higher system throughput will be achieved by raising the value of the calling frequency when the line driver is idling and lowering the value when the line driver becomes active.

8.14.3 CRGUSR - Get the Current User's Port Number

After calling from this routine, the C-register will contain the port number for the user "owning" the current COMLST. If no COMLST is currently active, then the port number for the last used COMLST is returned. The port number is returned with a base of zero; that is, the values range from 0 to 23, corresponding to DATASHARE ports 1 through 24. This routine is most commonly used after a general poll (described below) to determine the owner of the COMLST that was found. This routine should be called only from the secondary state.

8.14.4 CRRPGO - Request Permission to GO

This routine should be called at a logical stopping point in the main communications loop in secondary state. It returns TRUE ZERO if permission to continue is granted, or FALSE ZERO if permission is denied. Denial of permission indicates that the DATASHARE background is waiting to do a ROLLOUT. If permission to go is denied, the external communications process need not terminate immediately, but it must eventually acknowledge that a suspension was requested. If it never acknowledges the suspension, the DATASHARE system will be hung in a background loop until the ROLLOUT instruction times-out. This routine should be called only from the secondary state; in the primary state, the Interpreter may ROLLOUT at any time without notifying the line driver.

8.14.5 CRSSAF - Set the Suspension Acknowledged Flag

This routine is called to set the suspension-acknowledged flag in response to a denial of permission to enter a new control sequence. This routine should not be called until all "wrap-up" operations have been performed (which may take several interrupts to accomplish). The DATASHARE system will typically perform a ROLLOUT shortly after this routine is called. If it is desirable to reinitialize the line driver after the ROLLOUT, the primary state should be entered before calling this routine.

8.14.6 CRCS$ - Change State

Calling this routine effects a return to the scheduler and sets the entry point to the communications process code to the instruction following the call. The concept is identical to that used by the interrupt handler in the DOS. The entry point set by this routine is called the "secondary state entry point". This routine should be called only from the secondary state.

8.14.7 CRCPS$ - Change Primary State

This routine works in the same manner as CRCS$. However, it sets a different entry point called the "primary state entry point". The primary state entry point is called by the scheduler when the process-available flag is clear; the secondary state entry point is called when the process-available flag is set. This routine should be called only from the primary state.

8.14.8 CRSSOK,CRSSCU - Set Status

CRSSOK - Set Status to OK
CRSSCU - Set Status Channel Unavailable

When called, these routines set a final status into the COMLST, (the <list>,) and dequeue and release it from use by the communications process. They may be called using an inactive COMLST without harm. These routines should be called only from the secondary state.

8.14.9 CRRSB - Reset to Start of Block

This routine resets various pointers to the values obtained by the last call to CRNSB (discussed below). It is used in a blocked-message discipline when it is necessary to repeat the last message block. Calling this routine will not reset either the formpointer or length pointer of any variables which may have been affected by the message block in error. Hence, a repeated message block which is significantly shorter than the original message block may cause some extraneous characters to remain in some of the variables. This routine should be called only from the secondary state.

8.14.10 CRNSB - Note Start of Block

This routine is called to capture the critical COMLST pointers at their current values. These values may be restored by the CRRSB routine. This routine should be called only from the secondary state.

8.14.11 CRCAF,CRSAF - Clear/Set the Availability Flag

CRCAF - Clear the Availability Flag
CRSAF - Set the Availability Flag

The availability flag is used to differentiate between "handshaking mode" and "active communication mode". When the availability flag is clear, the scheduler will enter the communication process at its primary entry point, and will also dequeue all posted COMLSTs with a status of "channel unavailable". This dequeuing takes place after the primary state is entered and may take several interrupts to complete. Applications programs are thus notified when they attempt to post a COMLST when a condition of active communication is still problematical (i.e.,

handshaking may fail). Similarly, as long as the availability flag is clear the communication process would be free to do any required setup, (waiting for ringing, handshake sequencing, etc.,) without the burden of disposing of a COMLST which it would be unable to handle. When the active processing of messages is possible, the process should set the availability flag and be prepared to enter itself at the secondary state entry point upon the next interrupt. The CRCS$ routine, usually used only in secondary state, is used to define this entry point. CRCAF should be called from the secondary state, CRSAF from the primary state.

8.14.12 CRSUSR - Select User

This routine is called with a DATASHARE port number in the C-register. The port number should be based to 0 (i.e., it should have a range of 0 to 23). The routine will return TRUE CARRY if the indicated port has not been configured into the DATASHARE system. Otherwise, it will return FALSE CARRY with the hardware base register set to select the data area for the specified port. This routine may be used as a quick way to determine whether a "selectively addressed device" actually exists. This routine should be called only from the secondary state.

8.14.13 CRGENP - General Poll

This routine performs the operation of "general poll" on all ports configured into the system. It scans each port in sequence for a posted sending COMLST. It returns TRUE ZERO if no sending COMLST exists anywhere. If a sending COMLST is found, it returns FALSE ZERO after internally calling CRGSCL (i.e., the communications process need not perform any additional setup on the COMLST; it is ready to deliver characters). This routine must be called at the stack level at which the process was entered by the scheduler, because it makes an internal call CRCS$. By testing only one port per each interrupt, the consumption of excessive foreground time is minimized. This routine should be called only from secondary state.

8.14.14 CRZTIM,CRGTIM - Zero, Get the Timer

CRZTIM - Zero the Timer
CRGTIM - Get the Time

These routines are used for timing intervals used for turn-around delays, time-out tests, etc. The time is counted in 1/4ths of a second, thus allowing a one-byte counter to represent from 250 milliseconds up to 60 seconds. CRZTIM has no arguments or results; CRGTIM returns the time in the C-register. These routines can be called from either primary or secondary state.

8.14.15 CRGET - Get a Character from the COMLST

This is the basic routine used to get characters from a sending COMLST. If a character is returned, it is delivered in the B-register. There are four possible return conditions for this routine, as follows:

FZ FS FC - Character in B-reg (=0203 if past length pointer).
FZ FS TC - end of physical variable encountered;
COMLST toggled to next variable;
no character returned.
TZ FS TC - end of physical variable and end of COMLST;
no character returned.
FZ TS TC - abnormal end of COMLST (i.e., COMLST is gone);
no character returned.

Characters are retrieved from a variable beginning with the first physical character and ending with the last physical character. For string variables, all characters past the length pointer are returned in the B-register as the character 0203 (note that this is only an indicator, not the 0203 at the physical end of the string. At the option of the communication process, this character may be ignored, converted to a blank, or used to trigger the generation of an end-of-field mark. This routine should be called only from the secondary state.

8.14.16 CRGDCL,CRGSCL - Get a Destination/Source COMLST

CRGDCL - Get a Destination COMLST
CRGSCL - Get a Source COMLST

These routines are called with the desired port number (0 - 23) in the C-register. If no COMLST is found, the routines return TRUE ZERO. If the desired type of COMLST is found for the

specified port, the routines return FALSE ZERO with the COMLST's status set to "in process" and the COMLST still on the queue. If a COMLST is found and it has a "gone away" status, the routines return TRUE CARRY. All pointers are set up for the first get/put, and the start of the block is noted. Since COMLST's are not automatically dequeued when exhausted or "gone away", CRSSOK or CRSSCU must be called to dequeue one COMLST before queueing another of the same type for the same port. This routine should be called only from the secondary state.

8.14.17 CRTOGL - Toggle to the Next Variable

This routine may be called by the communication process to force a source COMLST to move directly to the next variable for the next call to CRGET. This routine should be used only when handling record formats that permit variable length fields. This routine should be called only from the secondary state. There are three possible return codes:

FZ FS TC - COMLST successfully toggled to next variable.
TZ FS TC - normal end of COMLST encountered.
FZ TS TC - abnormal end of COMLST encountered.

8.14.18 CRPUT - Put a Character into a COMLST

This routine is called with a character given in the B-register. If the character is not an 0203, an attempt is made to place the character into the current variable in the COMLST. If the character is an 0203, the length pointer for the current variable will be set and the COMLST will be toggled to the next variable. Note that 0203 is only an indicator, not the physical end-of-string indicator. There are four possible return conditions, as follows:

FZ FS FC - character was successfully put.
FZ FS TC - character not put; end of variable encountered; COMLST toggled to next variable.
TZ FS TC - character not put; end of variable encountered; end of COMLST encountered.
FZ TS TC - character not put; abnormal end of COMLST.

The communication process should always insert an 0203 following the last message character (unless the end of the COMLST was already encountered) so that the length pointer for the last variable will be properly set. It should be noted that all variables in a receiving COMLST are cleared when the RECV verb is

processed. In addition, each variable is cleared when it is initially toggled to. Thus, if a message block is received in error, some variables may be left with non-zero length pointers which will not be reset properly unless the correct message block has a text length no less than the error block minus one. This routine should be called only from secondary state.

8.14.19 CRGRVA - Get the Route Variable Address

This routine is provided to allow the communication process to access the route variable for such applications as might require additional input parameters, resulting status indicators, etc. It is vitally important that once the route variable address is retrieved, all interaction with the actual data in the variable take place during the same interrupt so that the route variable does not have opportunity to disappear (i.e. as a consequence of a CHAIN operation by the background). The routine returns TRUE SIGN if the COMLST has "gone away", or FALSE SIGN with the route variable address in the BC-register pair, the formpointer in the D-register, and the length pointer in the E-register. This routine should be called only from secondary state.

8.14.20 CRSTRT,CREND,MLCTAB - External Definitions

These external definitions define addresses used by the communication process. CRSTRT should be used as the starting address of the code (i.e. ORIGIN CRSTRT). The highest address reached by the code should not reach or exceed CREND. The size of the MULTILINK code area in DATASHARE may be configured to any size up to 8K bytes depending upon options configured. MLCTAB is used by all processes and must always be the value in the X-register when any interface routine is called; i.e. a "LX MLCTAB>8" must be performed if the X-register is modified. The X-register is initialized with the value of MLCTAB>8 by the scheduler before any call is made to the external communication process. This value of the X-register is preserved by interface routines.

8.14.21 WHAT HAPPENS WHEN "THE COMLST GOES AWAY"

The expressions "abnormal end of COMLST" and "the COMLST went away" refer to a condition where the data area (hence the variables, and hence the COMLST) for a given port is no longer valid. This condition arises as a result of a CHAIN operation, whether planned (as with the CHAIN and STOP verbs) or unplanned (as with a disconnecting port or the unintentional use of the INT

key). When this happens to the port that "owns" the current COMLST in use by the communications process, it becomes necessary to "make the COMLST go away" with respect to the logical operations of the MULTILINK interface routines. Each COMLST has its own status byte that contains a "gone away" indicator. All interface routines that interact with a COMLST or its related variables check the "COMLST gone away flag" before proceeding, and, if the test is positive, produce harmless results. The communications process thus does not need to worry about performing an operation upon non-existent data. However, it should not ignore the "abnormal end of COMLST" return conditions as they will cause infinite looping. Once a "COMLST gone away" condition has been discovered, a call should be made to CRSSCU in order to dequeue the COMLST.

8.14.22 LOGICAL RELATIONSHIPS

The logical flow relationship among several of the MULTILINK interface routines may be clarified by the following skeleton communications process:

```
START   CALL    CRCPS$
        LC      3
        CALL    CRCCF
        shake hands, etc.
        CALL    CRSAF
GO      CALL    CRCS$
        CALL    CRRPGO
        JTZ     OK
        CALL    CRSSAF
        JMP     GO
OK      CALL    CRGSCL
LOOP    CALL    CRCS$
        CALL    CRGET
        JTS     DONE
        JTZ     DONE
        JTC     LOOP
        CALL    SENDIT
        RET
DONE    CALL    CRSSOK
        JMP     GO
SENDIT  .
        .
        POP
        JMP     NOSEND
        .
        .
        RET
NOSEND  CALL    CRCAF
        JMP     START
        END     START
```

The resulting relocatable module created by the assembler (SNAP3) would then be linked using the DOS linkage editor (LINK). The segment created by LINK would be ORIGINed at CRSTRT and would use the LEX file created by the Configurator to resolve all external references.

8.15 ACALL and External Communications

Since ACALL is entirely a background process and Multilink Line Drivers execute exclusively in foreground, if ACALL and External Communications are configured together they MUST NOT communicate with each other, pass parameters to each other, or use common data areas and routines.

8.16 UPS CONSIDERATIONS

DATASHARE VI may be configured to run under the 6600 Utility Partition Supervisor processor. DATASHARE VI requires UPS version 2.1 or above. All features of the DATABUS language are available except for ROLLOUT, SHUTDOWN, and CONSOLE statements, and port-1-on-console. In exchange for the features allowed in the DATASHARE partition, the DOS partition is restricted to running a limited subset of the Datapoint program library. Cassette and Servo printer activity is prohibited, and execution of other language products (i.e. BASICPLS) is not supported. Programs useful to the DATASHARE user, (such as EDITOR, FASTSORT, and DBCMPLUS,) combined with DATASHARE partition operation provide a powerful development and production facility.

8.16.1 Configuring DATASHARE VI under UPS

If a 6600, 5020, or 5040 processor is specified as the Interpreter's processor and DOS is given as the Interpreter's environment, the question

WILL UPS BE ACTIVE ON THIS PROCESSOR? [N]

should be answered "Y" if the Utility Partition Supervisor is to run the Interpreter.

8.16.2 Runtime Considerations

Since there is no method for determining what activity is taking place in the DATASHARE partition from the system console, it is risky to terminate operations without positive assurance from all DATASHARE users that their user programs are quiescent. As there is no console file (such as in PSDS4), no remote partition console (such as in PS66), and no interpartition communications facility, the user can most readily bypass the problem by modifing his ANSWER programs to first WRITE a disk record indicating his port number, execute a KEYIN, then over-WRITE this record with a character string which will indicate that his port is actively executing. (NOTE: There must be no KEYIN or DISPLAY statements between the first ANSWER program statement executed and the port-number-WRITE statement as the WRITE must be performed before any instructions requiring an on-line port are encountered. The following program demonstrates this technique:

```
ERROR     DIM       *11
.
FILE      FILE
.
POINT     FORM      5
.
PASSWORD  DIM       8
PORT      FORM      2
.
. PROCEDURE
.
BEGIN     CLOCK     PORT,PASSWORD
          SETLPTR   PASSWORD,2
          MOVE      PASSWORD,PORT
          TRAP      CONSOLOG IF IO
          OPEN      FILE,"CONSOLOG"
          TRAPCLR   IO
          WRITE     FILE,PORT;"  PORT CLOSED"
          DISPLAY   *R,*P1:24,"PORT ",PORT," ANSWER. ";
          KEYIN     "ENTER PASSWORD: ":
                    *EOFF,PASSWORD
          WRITE     FILE,PORT;"* ANSWER",PORT
. . . (ANSWER program continues)
.
.
.
CONSOLOG  PREPARE   FILE,"CONSOLOG"
          MOVE      "0" TO POINT
          WRITE     FILE,POINT;"DS6 1.x ANSWER ":
                    "PROGRAM STATUS FILE"
PREPPER   ADD       "1" TO POINT
          WRITE     FILE,POINT;"  PORT CLOSED"
          COMPARE   "24" TO POINT
          GOTO      PREPPER IF NOT EQUAL
          ADD       "1" TO POINT
          WEOF      FILE,POINT
          CLOSE     FILE
          OPEN      FILE,"CONSOLOG"
          RETURN
.
.
.
```

The Utility Partition Supervisor separates printer usage: If the DOS partition is printing, an attempt to print from a DATASHARE port will generate the same results as if another port

was using the printer. Likewise, DOS partition utilities, such as LIST and FILES, check to see if the printer is free, and they wait if it is not available. If a DATASHARE port is printing and a DOS program is repetitively requesting the printer until it becomes available, the completion of printing and subsequent RELEASE by the DATASHARE port will be completed before the DOS program is allowed to use the printer. If another DATASHARE program is also waiting on the printer in addition to the DOS program, either contender may capture the printer; the winner will be whichever program first completes the printer request sequence.

There is no provision for separating file accesses between DATASHARE and the DOS partition (PI and FILEPI have no effect outside of the Interpreter); it is entirely the user's responsibility to prevent the possibility of file damage by concurrent WRITEs or UPDATEs from both partitions.

One purpose of UPS is to simplify the simultaneous development and operation of DATASHARE applications systems. Typically, a programmer will edit and compile programs in the DOS partition, then test them in the DATASHARE environment.

8.17 POLLABLE TERMINAL SUPPORT

For users requiring pollable terminal support, the following sections describe the interface characteristics between the DATASHARE VI with the POLL facility and the user supplied protocol handlers. Each user supplied routine is called by DATASHARE at appropriate times to perform a needed function or acquire necessary information.

The user should observe several restrictions when generating the support routines. They are:

1. Do not enable or disable interrupts.
2. Do not modify the base register or the processor sector table.
3. Reduce the execution time of foreground routines to the minimum possible, at most 100 microseconds.
4. Do not use more than 2 stack levels without first saving the stack. Upon exit the stack must be restored to the state upon entry to the user program.

The POLLINK/REL file must have three PROGs within it. POLLINIT, POLLBG, and POLLFG. Each of these PROGs must contain a single PAB with the same name as the PROG.

The POLLINIT PROG contains the USRINIT routine. This routine is LINKed with the Interpreter's initialization code and executed when the system initializes. The memory occupied by this routine is overstored with other code after initialization and thus this routine will no longer be available.

The POLLBG PROG contains the background routine USRPOL.

All the other routines are foreground routines and must be in the POLLFG PROG. They are: USRRX, USRTX, USRETX, USRIRS, and USRPOLRX.

Because of the separation of background and foreground in DS6, the POLLBG and POLLFG PROGs are not LINKed together, and consequently cannot communicate with each other or use common data areas or common routines.

Each routine will be entered with the D register containing the port number (zero relative) and the E register containing the polled port ID (i.e. port type POLLn ==> E register contains n). Other registers will be defined for each entry point. If the contents of a register is not specified, then its contents are undefined. All registers, with the exception of the X-register and the base register, are available for user use and need not be preserved.

8.17.1 USRINIT - User Port Initialization

This routine is called at initialization time once for each pollable port. The user is allowed to initialize the user supplied POLLINK module execution and indicate certain POLLINK run-time options.

ENTRY None

EXIT B contains the options

Options

The following bit definitions are used in the B register for option indication:

B=0 - Pass character as received from the multi-port to the appropriate POLLINK routine.

PF$3600 - Process each character returned from USRRX as if it came from a 3600. Suppress any DATASHARE POLLINK attempt at data transparency. Toggle to next variable on 015 from multi-port. Shift inversion is in effect.

8.17.2 USRRX - User Calculations During Reception

This routine is called for each character that is received during accepted poll responses or keyin responses. The active character is contained in the A-register. The user is expected to calculate any necessary block check characters (BCC), check parity if necessary, check for an end-of-message condition, format and point to any necessary response, and set certain condition codes to inform DATASHARE of the status. This routine runs in foreground.

ENTRY A contains the received character.

EXIT HL points to insertion/response string, if necessary
D contains the length of the response string, if necessary
C contains the length of the insertion string, if necessary
B contains the condition flags
A contains character to be placed in the DATABUS variable

The structure of the HL string is as follows:

HL => insertion string for length C followed by response string for length of D.

Condition flags

The following bit definitions are used in the B register for condition flags:

B=0 - Pass character in A as received from the multi-port to the appropriate POLLINK routine.

P$USEHL - Use the string pointed to by HL for a length of C instead of the character in A. If C=0 then no characters will be used, the character will be ignored. If the P$COMPL bit is set, then following the insertion string should be a response string for a length of D. If D=0 then no response will be sent.

P$SKIP - Skip to next variable before inserting the next character.

P$COMPL - Message completed successfully, release I/O list to background. HL points to an insertion string followed by a response string. C contains the insertion string length. D contains the response string length.

P$ERROR - Message error. Release I/O list to background and indicate an error. HL points to the response string and C contains the length of the response string.

Note: Any response string return to DATASHARE will not be passed through the USRTX routine. It is therefore necessary for the USRRX routine to format the completed message, including parity and BCC.

Note: It is the user's responsibility to check incoming parity. DATASHARE does not check incoming parity before stripping off the high-order bit (bit 7) and processing the character.

8.17.3 USRTX - User Calculations During Transmission

This routine is called for each character that is taken from a DATABUS variable to be transmited. The active character is contained in the A-register. The user is expected to calculate any necessary block check characters (BCC) and set certain condition codes to inform DATASHARE of the status. This routine runs in foreground.

ENTRY — A contains the character to be transmitted.

EXIT — HL points to the insertion string, if necessary
C contains the length of the insertion string, if necessary
B contains the condition flags
A contains the character to be transmitted

Condition flags

The following bit definitions are used in the B register for condition flags:

B=0 - Continue standard processing, use character in A.

P$USEHL - Use the string pointed to by HL for a length of C instead of the character in A. If C=0 then no characters will be used and the character in A will be ignored.

P$SKIP - Skip to next variable

P$ERROR - Message error. Release I/O list to background and indicate an error. HL points to the response string and C contains the length of the response string.

Note: The user requested parity will already have been generated before entering USRTX. It is therefore not necessary for the user to generate it.

8.17.4 USRETX - User End of Transmission

This routine will be called when the I/O list of a transmit has been exhausted. The user is expected to format the block termination sequence. This routine runs in foreground.

EXIT	HL points to termination sequence including ETX (or the equivalent), BCC if required, and any other user required characters. C contains the length of the termination sequence.

8.17.5 USRIRS - User Inter Record Separators

This routine will be called between each variable of a transmit I/O list. The user is expected to format any inter-record separator sequence that may be required.

EXIT	HL points to IRS sequence C contains length of IRS sequence

8.17.6 USRPOL - User POLL

This routine will be called each time a new polling sequence is required.

ENTRY	HL points to the terminal address list variable in port working storage (see definition of the POLL verb).
EXIT	HL points to the new sequence. Caution must be taken if the routine modifies port working storage. C contains the sequence length.

Upon return the poll sequence will be processed just as any standard DATABUS variable. That is, parity will be generated by DATASHARE and the characters will be sent through USRTX.

8.17.7 USRPOLRX - User POLL RESPONSE

This routine will be called for each character that is received in response to a poll. The received character will be contained in the A register. The user is expected to inform DATASHARE how to interpret the response. The routine runs in foreground.

ENTRY — A contains the received character.

EXIT — B condition flags
A contains character to be placed in the DATABUS variable, if necessary

Condition flags

The following bit-definitions are used in the B register for condition flags:

B=0 - Continue receiving response, poll response is incomplete.

P$COMPL - Response completed. Continue polling or receiving as indicated by the P$ERROR flag.

P$ERROR - Negative response or response error. Continue polling as indicated by POLL verb. This bit will only be effective if the P$COMPL bit is set. No data character is returned.

If the P$ERROR bit is not set and P$COMPL is set, then USRPOLRX will not be called again until the next POLL verb is executed. Also USRRX will not be called until USRPOLRX has indicated a successful poll. The character returned from USRPOLRX in the A register will be passed to USRRX as the first received character.

*NOTE: It is the user's responsibility to check incoming parity. DATASHARE does not check incoming parity before stripping off the high-order bit (bit 7) and processing the character.

8.17.8 Port Dependent User Storage Area

For each port configured as pollable the POLLINK module is provided with ten bytes of port dependent storage space. That is, there is a unique ten-byte area for each port that is configured as pollable. A port's area is only addressable when that port is selected by DATASHARE with the E register at entry to a POLLINK module.

The label POLUSER is provided for user POLLINK storage addressing and is the base address of the 10 byte area. Although the X-register may not be modified by the user, this area may be addressed using "paged" instruction. DATASHARE will have set the X-register for this purpose.

For example:

```
PL          A,POLUSER+1     Get LRC accumulation byte
XRC                         XOR in present byte
PS          A,POLUSER+1     Store LRC btye
```

Note: If the 10 byte area is exceeded, either above or below, the results will be indeterminate.

8.17.9 Building POLLINK into DATASHARE

Unlike MULTILINK modules the user supplied POLLINK modules must be available at configuration time in order to build a polling DATASHARE. These modules will be included into the DATASHARE link and will become part of the DATASHARE interpreter. They need not be available thereafter until the next configuration.

CHAPTER 9. AIM DEMONSTRATION PROGRAMS - AIMCHAIN/AIMPROG

The AIMCHAIN and AIMPROG programs have been provided to familiarize DATASHARE VI users with the new Associative Index Method (AIM). These programs can be used to create and utilize an AIM file corresponding to a given text file. These programs make ample use of help screens as well as flexible and friendly error detection and recovery facilities.

AIMCHAIN asks for a mnemonic name to be assigned to each key field. Using AIMPROG, the user specifies the mnemonic name of the field for each key thus alleviating the user from the burden of having to remember the field numbers.

**NOTE: AIMCHAIN and AIMPROG are provided only as a demonstration of possible uses of AIM. They are NOT supported software.

The AIMCHAIN program takes a text file and, by invoking the AIMDEX utility, creates the AIM (/AID) file necessary for use by AIM. A /CFF file of mnemonic key names is also created by AIMCHAIN. The /CFF file is utilized by AIMPROG to generate a display screen containing the mnemonic key names.

9.1 AIMCHAIN

The AIMCHAIN program is designed to allow the user to interactively enter all information required by AIMDEX, while at the same time reviewing the options and requirements of AIMDEX.

9.1.1 Invoking AIMCHAIN

The command line for the AIMCHAIN program is as follows:

```
CHAINPLS AIMCHAIN;[NAME=<infile>][,EXT=<ext>][,DRIVE=<Dn|DRn|Volid>]
```

If the name of the text file to be processed by AIMDEX is not specified on the command line, AIMCHAIN will prompt the user for the text file name, extension, and drive. The default extension of the text file is /TXT and will be used if the extension entered is null. If no drive is specified, an all drive search will be

performed for the text file.

9.1.2 Output Files Generated

AIMCHAIN generates two output files, an AIM file (/AID, created by AIMDEX) for the text file entered as input; and a key configuration file (/CFF) containing the mnemonic key names.

AIMCHAIN will prompt the user for the name, extension and drive for the AIM file. If the user responds with a null entry for any of the above, the default values will be used in place of the null responses. The default name for the AIM file is <infile>/AID. The default drive for the AIM file is the same drive as the text file.

The configuration file name will be the same as the AIM file name with an extension of /CFF (<aimfile>/CFF). The configuration file will always be placed on the lowest available mounted drive.

9.1.3 Key Field Selection

AIMCHAIN will display a list of rules and restrictions to follow in specifying key fields, and then begin prompting for a mnemonic name, whether the field is an excluded field, and column ranges for each key field. The mnemonic name will be written to the /CFF file, and the range will be used on the AIMDEX command line. Entering a null key mnemonic will terminate key field selection. If no key fields are entered, an error message will be displayed and execution will be terminated.

Each key field is checked for validity after it is entered. If an error is detected (such as key too long or illegal column ranges) a message will be displayed and the user will be asked to enter the key again.

9.1.4 Field Sort and Error Checking

The AIMCHAIN program contains an internal sort that is automatically invoked after key entry is completed to sort the key ranges entered by the user into ascending order, as required by AIMDEX. After completion of the field sort, the key fields are checked for overlap. If any key fields are found to overlap, an error message will be displayed and execution terminated.

The sort and error check will result in a slight delay after

key entry is completed. The length of the delay is dependent upon the number of key fields entered and the relative disorder of the key fields.

9.1.5 AIMDEX Option Selection

AIMCHAIN will display a brief summary of the AIMDEX options before prompting the user to enter any desired options. Entering a null response for option selection will terminate the option selection process. Some of the options available for AIMDEX have values associated with the option. If one of these option(s) has been selected, the user will be prompted for the values to be associated with the option. A null response at this point will result in selection of the default value for the option.

Entry of an invalid option or option value will result in an error message. The user will then be asked to select another option (or option value).

9.1.6 AIMDEX Invocation

After termination of the option selection phase, AIMCHAIN will invoke the AIMDEX utility to process the text file. The command line for AIMDEX will be generated based on the user responses during key field and option selection. When AIMDEX finishes execution, the text file will be ready for access using the DATABUS program AIMPROG.

9.2 AIMPROG

AIMPROG is a DATABUS program which has been provided to demonstrate the data base inquiry capabilities of the AIM access method. The text file to be accessed must be processed by AIMDEX via the AIMCHAIN program prior to running AIMPROG.

9.2.1 Preparing a Text File for Use by AIMPROG

The following files are required prior to use of AIMPROG:

(1) A DATASHARE VI Interpreter configured for AIM

(2) The text file to be accessed via AIMPROG

(3) A /CFF file corresponding to the text file to be accessed, containing the mnemonic key names. This file will be automatically created by running AIMCHAIN.

(4) An AIM file created by AIMDEX corresponding to the text file to be accessed. This file will be automatically created by running AIMCHAIN.

9.2.2 Run Time Options

The * and the interrupt key can be used to interactively re-direct program execution as follows:

(1) * entered for filename will return you to master.

(2) ? entered for key mnemonic name will display the search help screen.

(3) * entered for key mnemonic name will abort all keys entered and return you to the file name request.

(4) Interrupt key entered during key entry will abort all keys entered so far and restart key entry.

(5) Interrupt key entered during record search and display will abort searching and give record statistics.

9.2.3 File Request

AIMPROG will prompt the user to key in the name, extension, and drive specification for the AIM file to be used. A null file name will result in the user being prompted again. The file containing the mnemonic key names is expected to have the same name as the AIM file with an extension of /CFF. The AIM file extension defaults to /AID if not specified. If the /CFF file of the AIM file can not be found, an error message will be displayed and the user will be prompted to re-enter the file specifications.

9.2.4 Key Entry

The AIM file created by AIMDEX will be searched for records meeting the requirements specified at key entry time. For a key to be used as part of the generic key by AIM, the mnemonic key name, key search type, and key data must be keyed in. Entry of a key mnemonic not in the /CFF file is not allowed and will cause AIMPROG to beep and ask for another key name. There may be anywhere from 1 to 64 keys supplied by the user. At any time during key entry, a null name entered for the mnemonic key name specifies that the user is finished entering keys. If no keys are entered, an error message will be displayed and the user will be prompted to re-start key entry.

9.2.5 Key Search Types

The allowable key search types are listed below:

X: EXACT--key data must match key field exactly

L: LEFT--key data must match left portion of key field

R: RIGHT--key data must match right protion of key field

F: FREE-FLOAT--key data must occur somewhere in key field

Entry of anything other than X, L, R, or F for a key search type is not allowed and will cause AIMPROG to beep and ask for another key search type.

9.2.6 Display Modes

AIMPROG will then ask the user which mode of displaying the records is desired. The answer to this question must be either C, W, or N.

9.2.6.1 Continuous Mode (C)

All records found by AIM are continuously scrolled onto the screen until they are all displayed. The number of records found and the elapsed time are then displayed for access method time comparisons.

9.2.6.2 Display Wait Mode (W)

All records found by AIM are singularly scrolled onto the screen, pausing between record displays for operator acknowledgement. The number of records found is then displayed.

9.2.6.3 No Display Mode (N)

The records found by AIM are not displayed on the screen but the number of records found and the elapsed time are displayed after all records have been found.

Entry of anything other than W, C, or N will cause an error message to be displayed and the user to be prompted again for the display mode. If an I/O error occurs during a search, the search will be aborted and the error message displayed on the screen.

CHAPTER 10. DATABUS LIBRARY FACILITY OF DATASHARE VI

Using the Datapoint library utility program LIBSYS, one may create library files of DATABUS object code (/DBC programs). Such libraries are accessed and utilized by DATASHARE in much the same manner as DOS uses the UTILITY/SYS file. Proper use of DATABUS library programs will result in greater system integrity, more file names available on system disks, and easier backup.

10.1 The System DATABUS Library

The syntax of DATASHARE's command line is as follows:

<Interpreter-name> [<port 1 answer pgm>], [<system DATABUS library>], [<alternate ROLLFILE name>]

If no name is specified for the <system DATABUS library>, a default name of DATASHAR/DBL will be used instead. If the specified or default <system DATABUS library> file is not found during system initialization a warning message will appear on the system console.

The <system DATABUS library> file must be a LIBSYS-created grouping of DATABUS object code programs. It is searched by DATASHARE during CHAIN operations for DATABUS object programs. During CHAINing operations, DATASHARE will search any on-line drive to find a free-standing DATABUS program name which matches the program specification operand given in the CHAIN instruction. If this search is unsuccessful, DATASHARE will then search the <system DATABUS library>. Failure to then locate the program in the library will result in a CFAIL condition.

10.2 The CHAIN program specification

DATASHARE has the ability to accept both drive numbers and volume names as components of a file specification; the CHAIN facility has now been extended to allow specification of member names within a library. The new program specification syntax of CHAIN is:

<program name>/<extension>:<drive # or VOLID>.<library member name>

Note: No intervening blanks are allowed in the character string or literal used as a program name, and the member name must consist of eight characters or less.

If a <library member name> is used in a program specification, the <program name> is presumed to be a DATABUS program library file. Failure to locate either the library or the proper member within the library will result in a CFAIL; if the <program name> is not a DATABUS library file a CFAIL will also result. If a <member name> alone is specified, a search of the <system DATABUS library> will be performed; no free-standing program search will occur. Note that the extension of the DATABUS program library file must be specified.

10.3 Examples of CHAIN specifications

```
MYPROG
```

This specification would cause DATASHARE to attempt to find the file MYPROG/DBC on any drives on-line. Failure to locate the file would cause a search of the <system DATABUS library> file for a member with the name MYPROG. Note that the extension could have been specified if the file had an extension of other than "/DBC".

```
.MYPROG
```

This specification would cause DATASHARE to attempt to locate the member MYPROG in the <system DATABUS library>. No attempt to find any free-standing file would be made; absence of a <system DATABUS library> would cause a CHAIN failure.

```
SYSLIB/DBL.MYPROG
```

This specification would cause DATASHARE to locate the file SYSLIB/DBL and interrogate the file for the member MYPROG.

```
SYSLIB/DBL:DAILY.JOBA
```

This specification would cause DATASHARE to locate the file SYSLIB/DBL on any on-line drive with a volume name of DAILY. The member JOBA would then be found and executed if present.

10.4 Possible Uses of DATABUS Libraries

The programs most frequently used by a DATASHARE system are the ANSWER and MASTER programs for the various ports in use. All ANSWER and MASTER programs may now be kept resident in a <system DATABUS library> instead of existing free-standing. This will allow more DOS directory names to remain available to the user as only the <system DATABUS library> name will be entered in the directory.

In typical business environments, most application programs belong to a certain class of processing, such as payroll or accounts receivable. Using DATABUS libraries, the organization, testing, and everyday use of specific-class programs may be greatly simplified. For example, a typical office might create the following libraries:

```
PAYROLL/DBL   containing all payroll programs
ACCTSRCV/DBL  containing all accounts receivable programs
ACCTSPAY/DBL  containing all accounts payable programs
TEST/DBL      containing new programs in the testing phase
```

APPENDIX A. INSTRUCTION SUMMARY

SYNTACTIC DEFINITIONS

<aclist>	Any combination of numeric or character string variables, FILEs, IFILEs, AFILEs, or COMLSTs separated by commas. The list may be continued on more than one line by placing a colon (:) after the last operand on the line to be continued.
<afile>	A name assigned to an AFILE declaration.
<blist>	The name assigned to the first of a set of physically contiguous numeric string or character string variables.
<brlist>	A list of execution labels separated by commas. The list may be continued on more than one line by placing a colon (:) after the last label on the line to be continued.
<char>	Any single character of the form "<string>" where string is of length one (1).
<cmlist>	A name assigned to a statement defining a COMLST data declaration.
<dlist>	Any combination of <slit> and <occ> separated by commas. The list may be continued on more than one line by placing a colon (:) after the last variable on the line to be continued.
<dnum>	A decimal number between 0 and 255.
<dnuml>	A decimal number indicating the number of digits that should precede the decimal point.

<dnum2>	A decimal number indicating the number of digits that should follow the decimal point.
<dnum3>	A decimal number between 1 and 20 inclusive.
<dnum4>	A decimal number between 1 and 64 inclusive.
<dnum5>	A decimal number between 0 and 20 inclusive.
<dnum6>	A decimal number between -128 and 127 inclusive.
<dnum7>	A decimal number between 1 and 255 inclusive.
<dnvar>	A name assigned to a statement defining a destination numeric string variable. This variable is generally changed as a result of the instruction.
<DOS file spec>	A DOS compatible file specification (see DOS user's guide).
<dsvar>	A name assigned to a statement defining a destination character string variable. This variable is generally changed as a result of the instruction.
<equ>	A name assigned to an EQUATE statement.
<event>	The occurrence of a program trap: PARITY, RANGE, FORMAT, CFAIL, IO, SPOOL, INTERRUPT, INT, F1, F2, F3, F4, F5, <svar>, or <char>.
<event1>	The occurrence of one of the following program traps: PARITY, RANGE, FORMAT, CFAIL, IO, or SPOOL.
<file>	A name assigned to a FILE declaration.

<file list>	A list of one or more FILE, RFILE, IFILE, RIFILE, and AFILE names separated by commas.
<flag>	One of the following flags: OVER, LESS, ZERO, or EOS (EQUAL and ZERO are two names for the same flag). These flags are used to indicate the result of certain DATABUS operations.
<fflag>	One of the following flags: F1, F2, F3, F4, or F5. These flags are used to indicate the status of the console's function keys, (if the function key feature is available on the processor), and are used with the GOTO instruction.
<ifile>	A name assigned to an IFILE declaration.
<index>	A numeric variable used in connection with list accessing.
<key>	A non-null string variable used as a key to indexed I/O accesses.
<label>	A letter, followed by any combination of up to seven (7) additional letters and digits.
<list>	Any combination of <slit>, <occ>, <list controls>, <nvar> and <svar> separated by commas. The list may be continued on more than one line by placing a colon (:) after the last variable on the line to be continued.
<nlist>	A list of numeric variables each pair of which is separated by a comma (,). The list may be continued on more than one line by placing a colon (:) after the last variable on the line to be continued.
<nlit>	A literal of the form "<string>" where string is a valid numeric string.

<nslist>	Any combination of numeric and character string variables separated by commas. The list may be continued on more than one line by placing a colon (:) after the last variable on the line to be continued.
<null>	A null string variable used as a key to an indexed read.
<nvar>	A name assigned to a statement defining a numeric string variable.
<occ>	An octal control character (000 to 0377 inclusive).
<occl>	An octal control character between 0 and 0177 inclusive.
<pdnum>	A positive decimal number between 0 and 127 inclusive.
<pdnuml>	A positive decimal number between 1 and 127 inclusive.
<plist>	List controls used in a POLL statement. The list controls are separated by commas. The list may be continued on more than one line by placing a colon (:) after the last control on the line to be continued.
<prep>	A comma (,) or a valid preposition BY, FROM, IN, INTO, OF, TO, USING, and WITH. (Note: A preposition is allowed for source code readability only, but any preposition may be used even if it does not make sense in English in the context of the particular verb.)
<rfile>	A name assigned to an RFILE declaration.
<rifile>	A name assigned to an RIFILE declaration.
<rn>	A numeric variable which contains a positive record number (greater than or equal to zero) used to randomly READ or WRITE a file.

<route>	A character string variable used for routing.
<seq>	A numeric variable which contains a negative number (less than zero) used to READ or WRITE a file sequentially.
<skey>	A numeric or character string variable used with SEARCH.
<slist>	A list of character string variables, each pair of which is separated by a comma (,). The list may be continued on more than one line by placing a colon (:) after the last variable on the line to be continued.
<slit>	A literal of the form "<string>".
<snvar>	A name assigned to a statement defining a source numeric string variable. This variable is unchanged as a result of the instruction.
<ssvar>	A name assigned to a statement defining a source character string variable. This variable is unchanged as a result of the instruction.
<string>	Any sequence of characters with the exception of the forcing character (#) which itself will not become part of the string literal's character sequence. The character following the # will become part of the character sequence (e.g. another # or ").
<svar>	A name assigned to a statement defining a character string variable.

FOR THE FOLLOWING SUMMARY:

Items enclosed in brackets [ ] are optional.

Items separated by the | symbol are mutually exclusive (one or the other but not both must be used).

COMPILER DIRECTIVES

```
<label>  EQU        <dnum|occ>
<label>  EQUATE     <dnum|occ>
<label>  IFC        <equ|dnum|occ>
<label>  IFEQ       <equ|dnum|occ>,<equ|dnum|occ>
<label>  IFGE       <equ|dnum|occ>,<equ|dnum|occ>
<label>  IFGT       <equ|dnum|occ>,<equ|dnum|occ>
<label>  IFLE       <equ|dnum|occ>,<equ|dnum|occ>
<label>  IFLT       <equ|dnum|occ>,<equ|dnum|occ>
<label>  IFNE       <equ|dnum|occ>,<equ|dnum|occ>
<label>  IFNG       <equ|dnum|occ>,<equ|dnum|occ>
<label>  IFNL       <equ|dnum|occ>,<equ|dnum|occ>
<label>  IFNZ       <equ|dnum|occ>
<label>  IFS        <equ|dnum|occ>
<label>  IFZ        <equ|dnum|occ>
<label>  INC        <DOS file spec>
<label>  INCLUDE    <DOS file spec>
<label>  LISTOFF
<label>  LISTON
```

FILE DECLARATIONS

```
<label>  FILE
<label>  IFILE
<label>  RFILE
<label>  RIFILE
<label>  AFILE      <dnum7>
<label>  AFILE      <dnum7>,<dnum4>
<label>  AFILE      <dnum7>,,<dnum>
<label>  AFILE      <dnum7>,<dnum4>,<dnum>
```

DATA DEFINITIONS

```
<label>  FORM       <dnum1>.<dnum2>
<label>  FORM       <dnum1>.
<label>  FORM       .<dnum2>
<label>  FORM       <dnum1>
<label>  FORM       <nlit>
<label>  DIM        <pdnum1>
<label>  INIT       <slit>
```

```
<label>   INIT      <dlist>
<label>   FORM      *<dnum1>.<dnum2>
<label>   FORM      *<dnum1>.
<label>   FORM      *.<dnum2>
<label>   FORM      *<dnum1>
<label>   FORM      *<nlit>
<label>   DIM       *<pdnum1>
<label>   INIT      *<slit>
<label>   INIT      *<dlist>
<label>   COMLST    <dnum4>

CONTROL

          ACALL     <svar>
          ACALL     <svar><prep><aclist>
          BRANCH    <index><prep><brlist>
          CALL      <label>
          CALL      <label> IF <flag>
          CALL      <label> IF NOT <flag>
          CHAIN     <svar|slit>
          DSCNCT
          FILEPI    <dnum3>;<file list>
          GOTO      <label>
          GOTO      <label> IF <flag>
          GOTO      <label> IF NOT <flag>
          GOTO      <label> IF <fflag>
          GOTO      <label> IF NOT <fflag>
          NORETURN
          PAUSE     <nvar|nlit>
          PI        <dnum5>
          RETURN
          RETURN    IF <flag>
          RETURN    IF NOT <flag>
          ROLLOUT   <svar|slit>
          SHUTDOWN  <svar|slit>
          STOP
          STOP      IF <flag>
          STOP      IF NOT <flag>
          TABPAGE
          TRAP      <label> IF <event>
          TRAP      <label> GIVING <svar> IF <event1>
          TRAP      <label> NORESET IF <event>
          TRAP      <label> GIVING <svar> NORESET IF <event1>
          TRAPCLR   <event>
```

ARITHMETIC

ADD	<snvar|nlit><prep><dnvar>
CHECK10	<svar><prep><svar|slit>
CHECK11	<svar><prep><svar|slit>
CK10	<svar><prep><svar|slit>
CK11	<svar><prep><svar|slit>
COMPARE	<nvar|nlit><prep><nvar>
DIV	<snvar|nlit><prep><dnvar>
DIVIDE	<snvar|nlit><prep><dnvar>
LOAD	<dnvar><prep><index><prep><nlist>
MOVE	<snvar|nlit><prep><dnvar>
MULT	<snvar|nlit><prep><dnvar>
MULTIPLY	<snvar|nlit><prep><dnvar>
STORE	<snvar|nlit><prep><index><prep><nlist>
SUB	<snvar|nlit><prep><dnvar>
SUBTRACT	<snvar|nlit><prep><dnvar>

LOGICAL

AND	<ssvar|char|occl><prep><dsvar>
OR	<ssvar|char|occl><prep><dsvar>
NOT	<ssvar|char|occl><prep><dsvar>
XOR	<ssvar|char|occl><prep><dsvar>

CHARACTER STRING HANDLING

APPEND	<ssvar|slit|snvar><prep><dsvar>
BUMP	<dsvar>
BUMP	<dsvar><prep><dnum6|snvar>
CLEAR	<dsvar>
CLOCK	TIME<prep><dsvar>
CLOCK	DAY<prep><dsvar>
CLOCK	YEAR<prep><dsvar>
CLOCK	VERSION<prep><dsvar>
CLOCK	PORT<prep><dsvar>
CMATCH	<ssvar|char|occl><prep><dsvar>
CMATCH	<ssvar><prep><char|occl>
CMOVE	<ssvar|char|occl><prep><dsvar>
EDIT	<ssvar|snvar><prep><dsvar>
ENDSET	<dsvar>
EXTEND	<dsvar>
LENSET	<dsvar>
LOAD	<dsvar><prep><index><prep><slist>
MATCH	<svar|slit><prep><svar>
MOVE	<ssvar|snvar|slit|nlit><prep><dsvar>
MOVE	<ssvar|snvar|nlit><prep><dnvar>
MOVEFPTR	<ssvar><prep><dnvar>
MOVELPTR	<ssvar><prep><dnvar>
REP	<ssvar|slit><prep><dsvar>
REPLACE	<ssvar|slit><prep><dsvar>
RESET	<dsvar>
RESET	<dsvar><prep><char|pdnum|snvar|ssvar>
SCAN	<ssvar|slit|occl><prep><dsvar>
SEARCH	<skey><prep><blist><prep><nvar><prep><dnvar>
SETLPTR	<dsvar>
SETLPTR	<dsvar><prep><char|pdnuml|snvar|ssvar>
STORE	<ssvar|slit><prep><index><prep><slist>
TYPE	<svar>

INPUT/OUTPUT

```
BEEP
CLOSE      <file|rfile|ifile|rifile|afile>
COMCLR     <cmlist>
COMTST     <cmlist>
COMWAIT
CONSOLE    <list>[;]
DEBUG
DELETE     <ifile|rifile|afile>,<key>
DELETEK    <ifile|rifile>,<key>
DIAL       <svar|slit>
DISPLAY    <list>[;]
FPOSIT     <file|rfile|ifile|rifile|afile>,<dnvar>,<dnvar>
INSERT     <ifile|rifile|afile>,<key>
KEYIN      <list>[;]
OPEN       <file|rfile|ifile|rifile|afile>,<svar|slit>
OPEN       <afile>,<svar|slit>,[<svar|char>]
POLL       <plist>,<ssvar>,<ssvar>;<plist>,<nslist>
PREP       <file|rfile>,<svar|slit>
PREPARE    <file|rfile>,<svar|slit>
PRINT      <list>[;]
READ       <file|rfile|afile>,<rn|seq>;<;|<list>[;]>
READ       <ifile|rifile>,<rn|seq|key|null>;<;|<list>[;]>
READ       <afile>,<slist>;<;|<list>[;]>
READKG     <afile>;<;|<list>[;]>
READKS     <ifile|rifile>;<;|<list>[;]>
RECV       <cmlist>,<route>;<slist>
RELEASE
RPRINT     <list>[;]
SEND       <cmlist>,<route>;<nslist>
SPLCLOSE
SPLOPEN    <svar|slit>[,<svar>|<slit>]
UPDATE     <ifile|rifile|afile>;<;|<list>[;]>
WEOF       <file|rfile|ifile|rifile|afile>,<rn|seq>
WRITAB     <file|rfile>,<rn|seq>;<;|<list>[;]>
WRITE      <file|rfile>,<rn|seq>;<;|<list>[;]>
WRITE      <ifile|rifile>,<rn|seq|key>;<;|<list>[;]>
WRITE      <afile>;<;|<list>[;]>
```

APPENDIX B. INPUT/OUTPUT LIST CONTROLS

In the table below, the following abbreviations are used in the USED IN column to indicate which DATABUS instructions the list controls can be used in: C=CONSOLE, D=DISPLAY, K=KEYIN, P=PRINT, P1=POLL, R=READ, W=WRITE.

CONTROL	USED IN	FUNCTION
;	KDP	Suppress a new line function when occurring at the end of a list.
;	R	Suppress scanning for logical end of record.
;	W	Suppress writing logical end of record.
*+	KDP	Turn on Keyin Continuous for KEYIN or suppression of space insertion after the logical length of a variable for DISPLAY, PRINT, and RPRINT.
*+	W	Turn on space compression during WRITE.
*+	P1	Turn on poll-continuous option in POLL.
*-	KDP	Turn off Keyin Continuous or allow insertion of spaces into a variable after its logical length for DISPLAY, PRINT, and RPRINT.
*-	W	Turn off space compression during WRITE.
*<n>	P	Causes a horizontal tab on the printer to the column indicated by the number <n>.
*<n>	RW	Tab specification for READ, WRITAB, or UPDATE operations.
*<nvar>	P	Causes a horizontal tab on the printer to the column indicated by the value of <nvar>.

*<nvar>	RW	The logical file pointers are moved to that character position relative to the current record.
*3270	KD	Enable 3270 mode in KEYIN and DISPLAY.
*B	KD	Emit an audible BEEP at the terminal.
*C	KDP	Causes a carriage return to be generated.
*CL	K	Clear the port's key-ahead buffer.
*DE	K	Restrict string input to digits (0-9) only.
*DV	K	Display a variable's value during KEYIN without performing a KEYIN operation on it.
*EF	KDC	Causes the screen to be erased from the current cursor position to the bottom of the display.
*EL	KDC	Causes the line to be erased from the current cursor position.
*EOFF	K	Prevents character echo to the display during keyboard input operations.
*EON	K	Causes character echo to the display during keyboard input operations.
*EP	KDPl	Generate even parity on outgoing bytes during KEYIN, DISPLAY, and POLL.
*ES	KDC	Causes the cursor to be positioned at horizontal position 1 of the top row of the display and the entire display to be erased.
*F	P	Causes the printer to be positioned to the top of form.
*HOFF	KD	Turn off highlighting mode (display characters normally).

*HON	KD	Turn on highlighting mode (display inverted image of all characters displayed).
*IN	KD	Clear Text-inversion mode.
*IT	KD	Set Text-inversion mode.
*JL	K	Left justify numeric variable and zero fill at right if there is no decimal point. Left justify string variable and blank-fill (or zero-fill if *ZF option is given) to end of string.
*JR	K	Right justify string variable and blank (or zero if *ZF option is given) fill at left.
*L	KDP	Causes a line feed to be generated.
*MP	W	Convert data in a numeric variable to minus overpunch format on disk.
*N	KDP	Causes the cursor or printer to be positioned in column 1 of the next line.
*NP	KDPl	Generate no parity on outgoing bytes during KEYIN, DISPLAY, or POLL.
*OP	KDPl	Generate odd parity on outgoing bytes during KEYIN, DISPLAY, or POLL.
*P<h>:<v>	KD	Causes the cursor to be positioned horizontally and vertically to the column and line indicated by the numbers <h> (horizontal 1-80) and <v> (vertical 1-24). These numbers may either be literals or numeric variables.
*P<h>:<v>	C	Causes the cursor to be positioned horizontally to the column indicated by <h> inside the area on the console reserved for terminal to operator communications (the <v> vertical position of the list control is ignored).

*PON	KD	Send a "printer on" character to a terminal.
*POFF	KD	Send a "printer off" character to a terminal.
*R	KD	Roll up the screen one line.
*RD	KD	Roll down the screen one line.
*RV	K	Retain the variable value if a keyin of ENTER only is received. Also enable the LESS flag to be set if the KEYIN is terminated by a (*T) timeout, the OVER flag if it is terminated by the NEW LINE key or function keys, and the EOS flag if it is terminated by a null entry.
*T	K	Time out after 2 seconds have elapsed between successively entered characters for KEYIN statement.
*T<n>	K	Time out after <n> seconds.
*T<n>:<m>	KPl	Specifies the time out value (n) and NAK count (m) during KEYIN and POLL.
*W	KD	Pause for one second.
*W<n>	KD	Pause for <n> seconds.
*ZF	K	Zero fill instead of blank fill string variable.
*ZF	PW	Left zero fill numeric variable.

APPENDIX C. FREEDOM PRINTER CONSIDERATIONS

The secondary tractor on a Datapoint 9232 FREEDOM printer may be selected by the following method:

1. Initialize a string variable to an octal 05 followed by two ASCII characters representing the left margin column minus 1 in hexadecimal ("00" to "83").

2. Use the initialized string variable as the 1st variable in each print statement.

For example:

```
P2          INIT   05,"3E"
.
            PRINT  P2,*F,DATA1,*20,DATA2
```

selects the secondary tractor and sets the left margin at column 63 ("3E" is column 63 minus 1 in hexadecimal) before performing top of form and printing. Tabbing to column 20 is relative to the left margin.

APPENDIX D. ERROR CODES

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

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

will appear on the screen display. The first two forms appear for all untrapped errors except I/O, AIM, SPOOL, and CHAIN traps. In the event of one of these traps, a qualification letter is given where a "Q" is shown in the example (explained below). The first 14 bytes of UDA (LP, FP, 11 data bytes, ETX...i.e. a DIM 11 field) for the program is overstored with the above error message. If the MASTER program defines an appropriate common data variable, the error data will then be available for examination after the trap. If an untrapped error occurs in MASTER, the port is shut down. The LLLLL or LLLLLL is the current value of the program counter and the X is an error letter. In most cases LLLLL or LLLLLL points to the instruction following the one that caused the problem. However, in certain I/O and SPOOLing errors, the program counter will point after the list item where the problem occurred. The following error letters can appear:

A - Interrupts already prevented
B - Illegal operation code
C - Chain failure (see Appendix E for explanation)
D - AIM error (See Appendix G for explanation)
F - Record format error
I - I/O error (see Appendix F for explanation)
L - Invalid command from Slave Station
P - Parity failure
R - Record number out of range
S - Spooling I/O error (see Appendix F for explanation)
U - Call stack underflow or overflow

The "A", "B", "L", and "U" errors cannot be trapped. An "A" error will be displayed if a PI or FILEPI instruction is executed and interrupts are already prevented or if a file specified in a FILEPI instruction is not open. A "B" error will appear only if an invalid object file is executed or if the system is failing. An "L" error will generally indicate that the telephone circuit is bad. The "U" error will occur if a programmer executes a RETURN

instruction without a corresponding CALL having been previously executed, or if CALLs are nested more than eight levels deep.

The events that may be trapped are shown below (the capitalized name shown is the name used in the TRAP statement):

CFAIL - The specified program was not in the DOS directory or in the specified DATABUS library, the library specified was not a DATABUS library, loading of a DATABUS program with an oversize user data area was attempted, or a program containing compile-time errors or one that uses unconfigured features was CHAINed. See appendix E for an explanation of the qualifying letter code.

FORMAT - Data being read into a numeric variable was not all digits and/or decimal point and minus sign, or a decimal point in the input did not agree with a decimal point in a FORM, or data input from disk had a negative multi-punch but no room was available for a minus sign in the FORM, or a WRITE-specified "multi-punch" and the last item of the field was a decimal point. The operation terminates with the item in error in an indeterminate state and the statement is aborted.

IO - Error during I/O statement. (Either a programming error or disk failure may cause this TRAP. See Appendix F for normal I/O error codes and Appendix G for AIM error codes). Note that AIM errors, which appear as D * Q format, are trapped through the IO event.

PARITY - Disk CRC (hardware) error during READ or disk CRC error during write-verification. (DOS retries the operation up to five times before indicating this failure.)

RANGE - Record number out of range. (An access was attempted beyond the physical end of the file, a record was read which was never written, or a WRITAB was used on a record which was never written.)

SPOOL - I/O error during write to spool file. (A write to the open spool file resulted in a

parity or CRC failure, a disk-space-full trap, or some other abnormal abort of a disk sector write command.) The secondary trap letter (S * <letter>) is explained within the same appendix (Appendix F) as the I * <letter> errors.

The following TRAPs are activated by keyed-in characters; these characters may be entered and used without actually executing a KEYIN instruction in the user's DATABUS program. None of these TRAP conditions will be taken unless the corresponding TRAP is set; e.g. an "untrapped F1 key" trap will never be activated if TRAP <label> IF F1 is not set and the F1 key is struck. Use of these traps requires that the Interpreter have the NEW TRAPS feature configured.

INT	- The INT (interrupt) key sequence was keyed in while the TRAP <label> IF INT trap was set.
F1	- The F1 key was keyed in while the TRAP <label> IF F1 trap was set.
F2	- The F2 key was keyed in while the TRAP <label> IF F2 trap was set.
F3	- The F3 key was keyed in while the TRAP <label> IF F3 trap was set.
F4	- The F4 key was keyed in while the TRAP <label> IF F4 trap was set.
F5	- The F5 key was keyed in while the TRAP <label> IF F5 trap was set.
<char>	- The <character> given in the TRAP <label> IF <character> trap was keyed in.

Note: None of the above key TRAPs are supported in the Slave Interpreter.

APPENDIX E. INTERPRETER CHAIN TRAP CODES

The format of the CHAIN failure trap is:

nnnnnn C * x

where nnnnnn is the address of the point of failure in the user's DATABUS program, and x is one of the following letters:

B - Bad object code file

C - Program contains instructions not configured in the Interpreter

D - A DOS trap occured during a CHAIN

L - Library not found

M - Member of library not found

N - Name of program was bad or program could not be found

R - ROLLOUT failure

U - User data area too large

APPENDIX F. INTERPRETER I/O AND SPOOLING TRAP CODES

All of the following codes apply to I/O trap letters; however, only certain of the codes are used in SPOOLing traps, the others being inapplicable to SPOOL. The errors that can occur during SPOOLing are: B,C,F,M,N, and P.

A - An access with a null key or 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 existed in the index.
F - Insufficient file space available.
G - COMLST status not clear when attempting to execute a SEND.
H - Invalid routing variable in COMLST or number of variables specified by COMLST exceeded during SEND.
I - The index file specified in an OPEN statement did not exist on the specified drive(s).
J - The index file found by the OPEN statement did not reside in the correct physical location on the disk (index files may never be moved, they must always be recreated by re-indexing).
K - A null key was supplied in an operation where the key may not be null.
M - The data file specified did not exist on the specified drive(s), or the specified drive was off-line.
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 or delete) or an I/O statement tried to modify a file that had DOS write protection.
Q - A dynamic ACALL overlay either could not be read due to disk errors or would not load into the configured ACALL area.
R - Unexpected EOF mark was encountered (before end of current logical record).
T - The tab value in the READ or WRITAB statement was off the end of the sector.
V - One of the DATASHARE overlay files could not be loaded by the DOS loader.
W - The root sector of the ISI tree was not an 012 sector. This is generally caused by using an old version of the INDEX

utility to create a null ISI file. Re-INDEX the file. See chapter 8.

X - The ISI file contains a circular link (is partially destroyed). Re-INDEX the file.

Y - An ISI tree sector could not be read. Usually indicates a partially destroyed ISI tree. Re-INDEX the file.

Z - An illegal INSERT was attempted. Either there was no valid READ or WRITE done to set up the pointers to the text file that INSERT uses, or these pointers indicate a different text file than the one associated with the file specified in the INSERT instruction.

NOTE: VWXY errors may be caused by the operator either removing a disk from its drive or swapping it with another disk while an operation is taking place.

APPENDIX G. AIM ACCESS ERROR CODES FOR DATASHARE VI

The following is a list of the error codes which can occur during an AIM access in DATASHARE. These errors are treated as I/O errors and can be trapped by the DATABUS programmer via the TRAP <label> IF IO instruction.

D * A	/AID File not found
D * C	Keys Conflict on READ
D * F	Free-Float Key Specification Error
D * H	/AID Header Sector Invalid
D * I	Insufficient Key Information on READ
D * K	Key Specification Format Error
D * L	AFILE too small for /AID File
D * M	AIM Map Damaged, Re-AIM File
D * R	No Valid READ Preceeding READKG
D * S	No More Room in /AID File, Re-AIM File
D * U	Illegal Update/Delete/Re-Read
D * V	AIMDEX and DS6AIM Versions Do Not Match

APPENDIX H. PROGRAM EXAMPLES

The following is an example of a "universal" ANSWER program; in other words, an ANSWER program that is independent of the logical port number on which it is run. (This technique of using CLOCK PORT to obtain the port's number should be adopted throughout the DATASHARE environment as opposed to the older method of compiling-in the port's number into an ANSWERxx program.)

The ANSWER program below displays the port number on the terminal (the ANSWER's program name will automatically be displayed on the console). The program then requests an identification and checks it for validity using 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 be the MASTER program. Otherwise, an indication is given that the proper identification was not entered and another request for identification is made.

The MASTER program merely requests the name of a program to be initiated and a CHAIN is executed to the name given. 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 neither the ANSWER or MASTER programs are written using cursor-positioning in the KEYIN or DISPLAY statements to aid in a Teletype terminal compatibility. The entry of a "*" for the program name causes the system to hang up the phone which provides a normal termination using the DATABUS DSCNCT instruction.

Simple ANSWER Program

```
.
. SIMPLE ANSWER PROGRAM
.
PORTN   FORM    2
IDCODE  DIM     9
ID      INIT    "DATAPOINT"
.
        CLOCK   PORT TO PORTN
        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
.
LOOP    KEYIN   *N,*EL,"PROGRAM NAME: ",FILNAM
        CMATCH  "*" TO FILNAM
        GOTO    DISCON IF EQUAL
        TRAP    NONAME IF CFAIL
        CHAIN   FILNAM
NONAME  DISPLAY "*** NO SUCH PROGRAM ***"
        NORETURN
        GOTO    LOOP
DISCON  DSCNCT
```

For a more complex example of ANSWER and MASTER programs see the DATABUS Compiler User's Guide.

APPENDIX I. DATASHARE VI UNDER ARC

I.1 Minimum Considerations

Only one modification to pre-DS6 DATABUS programs should be necessary to run them under DATASHARE in an ARC network: The DATABUS "PI" verb must be changed into the new "FILEPI" verb. This modification will allow programs running under DATASHARE VI to execute with or without ARC. The change into the FILEPI verb requires that a programmer inform DATASHARE which files must be locked out during a particular sequence of instructions. The only additional consideration is that the specified files must have been OPENed prior to the FILEPI instruction.

I.2 Use of Volume Identification

The standard DATABUS file specification used by OPEN and CHAIN allows the use of volume names (VOLID's). This is useful under ARC systems because ARC is a volume-oriented network. The logical drive numbers (assigned at execution time) may vary from user to user, especially if the same DATABUS program is used by more than one applications processor, but the volume names typically do not change. The use of volume names instead of hard-coded drive numbers therefore permits greater DATABUS program flexibility under ARC.

I.3 Write-Protected Data Files

Access time to data files, which are not modified, can be decreased by write-protecting the file. This enables DATASHARE VI to re-use buffer pages from the file which are found in memory.

I.4 ROLLOUT Under ARC

All remote volumes are controlled through sub-directories (specified at MOUNT time) in an ARC system. The use of sub-directories allows many users to share access to a single logical volume, but certain precautions must be taken in order to prevent contention problems.

When DATASHARE is initialized, certain files are created to allow the system to roll out. Under ARC these files will be created in the sub-directory currently active on the disk containing the DATASHARE object library. If, however, the ROLLOUT file already exists in the SYSTEM sub-directory, it may accidentally be overwritten by another DATASHARE system performing a ROLLOUT. Therefore, if two applications processors roll out and the ROLLFILE/SYS file is in the SYSTEM sub-directory, then the second ROLLOUT will overwrite the first and cause the first system to roll in improperly. This will also occur if two applications processors "sign on" into the same sub-directory. In order to avoid such conflicts, a unique file name should be specified on the command line by each DATASHARE user during his DATASHARE startup. The files thus created would guarantee proper operation. The file ROLLFILE/SYS (and file ROLLFILE/PWS in a large memory environment) may also be renamed in order to avoid conflicts. In general, if one applications processor should ROLLOUT into the same file as another, erroneous results should be expected.

Great care must be taken to insure that the location and number of all logical volumes associated with an applications processor prior to a ROLLOUT be identical with their configuration at roll-back. Failure to guarantee this will probably result in severly damaged files on disk. Under ARC it is not possible for DATASHARE to know whether a volume is local to an applications processor or is remotely MOUNTed.

Conversely, if two applications processors are identical in type and memory size and have the same logical volumes MOUNTed in the same positions and are "signed on" into the same sub-directories on all disks, DATASHARE may be rolled out on one processor and rolled back into another. This capability may possibly be useful but should be used with extreme caution.

When running under ARC, remote volumes that have been MOUNTed remain mounted until they are removed via the MOUNT command or by restarting the system. Terminating DATASHARE in the normal manner (depressing the RESTART and RUN/INT keys) will cause dismounting of all remote volumes. This is not always a desirable procedure and may be avoided by the use of the ROLLOUT feature whereby DATASHARE is rolled out to DOS but never rolled back, or by the use of the SHUTDOWN verb which cannot be rolled back.

APPENDIX J. THE DATASHARE RELOCATABLE LIBRARY

J.1 The DATASHARE VI Relocatable Files under DOS

The DS6/REL file contained on the Interpreter release media is a file made up of many different relocatable modules. This file (in Datapoint standard overlay library format) was created using the LIBSYS library utility program (version 2.2 or later). Since the CHAINPLS file DS6/TXT (used to configure a DATASHARE VI Interpreter) assumes that DS6/REL will be present, the DS6/REL file must not be renamed. However, the Interpreter created by CHAINPLS is a fully DOS-compatable overlay library and since the overlay loader module of the command file has the same name as the command file name, the entire library may be ADDed into the DOS program library file UTILITY/SYS by the use of the LIBSYS utility. Additionally, any DATASHARE VI-level product may be added to UTILITY/SYS without internal conflict should more than one DATASHARE VI-level command file be desired on a single disk.

J.2 The Relocatable Modules

The Relocatable Modules contained in DS6/REL are:

DS6ACALL	ACALL interface module
DS6AIM	Associative Index Acess Method (AIM) module
DS6ASLV	Asynchronous Slave module
DS6BACK	Rollback module
DS6BCSL	Background system console routines
DS6BPOLL	Background POLL routines
DS6BPTR	Background printer driver
DS6CHEK	CHECK10/CHECK11 module
DS6CLOCK	CLOCK verb module
DS6DBS	One-port DATABUS-type scheduler module
DS6DISK	Random/Sequential Disk I/O interface module
DS6DISKA	Disk interface for cartridge disk
DS6DISKB	Disk interface for mass storage disk
DS6DISKC	Disk interface for floppy disk
DS6DISKD	Disk interface for DOS.G floppy
DS6DSS	Multiple-port DATASHARE-type scheduler module
DS6DYNAC	Dynamic ACALL module
DS6EDIT	EDIT verb module
DS6FCSL	Foreground system console routines

DS6FILPI	FILEPI and PI for ARC Interpreters
DS6FMAIN	Main foreground code
DS6FPOLL	Foreground POLL routines
DS6FPTR	Foreground printer driver
DS6ICM	Internal communications module
DS6INIT	Interpreter initialization module
DS6ISAM	ISAM access module
DS6LPWS	Load-port-working-storage module
DS6M9320	9320 multiport adaptor interface module
DS6M9462	9462 multiport adaptor interface module
DS6MLWS	Working storage for MULTILINK and REPLACE verbs
DS6MMCR	Memory manager for large machines
DS6OSKNL	Root module
DS6P18	1800-type processor module
DS6P1C18	1800-type port 1 on console module
DS6P1C66	6600/5500 type port 1 on console module
DS6P55	5500-type processor module
DS6P66	6600-type processor module
DS6PBUFF	Printer output buffer
DS6PI	FILEPI and PI for non-ARC Interpreters
DS6RCHEK	ROLLOUT check overlay module
DS6REP	REPLACE verb module
DS6RMT	Slave stations interface module
DS6ROLAY	ROLLOUT overlay module
DS6SCAN	Scan verb module
DS6SPOOL	SPOOL verb module
DS6SPOVL	SPOOL verb overlay module
DS6SRCH	SEARCH verb module
DS6SSLV	Synchronous Slave module
DS6T3670	3670 interface module
DS6TRAPS	TRAP-handling mechanism module
DS6UDMGR	Memory manager for small machines
DS6UPROG	Environment-checking module
DS6UPS	UPS interface routines
DS6VERBS	Basic verb package
DS6VSEXT	Virtual storage manager for large machines
DS6VSSML	Virtual storage manager for small machines
DS6WPWS	Write-port-working-storage module
DS6XCM	External communications module

APPENDIX K. 6600 ERROR RECOVERY

DATASHARE VI running on a 6600/6020/6040 processor has the unique ability to continue operation when memory parity errors occur in port working storage by shutting down ports which access faulty areas of memory. If memory parity errors occur outside of port working storage, or any other type of system error occurs, the entire DATASHARE system will be aborted.

In the event of a system failure, the type of failure and the physical location of the failure will be displayed at the bottom of the console screen. If port 1 is not assigned to the console, and a port is shut down due to faulty memory parity, the failure address will also be displayed in the port's CONSOLE message area. Note that error messages will be lost during ROLLOUT.

The following messages are displayed to indicate system failures:

```
* PORT xx HAS BEEN DISABLED -

        MEMORY PARITY AT PHYS. LOC. xxxxxxxx *

* DATASHARE VI SYSTEM ABORTED -
```

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
DIRECTOR OF SOFTWARE SUPPORT
8550 Datapoint Drive, Mail Station# N60
San Antonio, Texas 78284