DATABUS11
DB11
User's Guide

Version 2

July, 1976

Model Code No. 50234

DATAPOINT CORPORATION

The Leader in Dispersed Data Processing

DATABUS 11
DB11

USER´S GUIDE

Version 2

July, 1976

Model Code No. 50234

PREFACE

DATABUS 11 is a high level business language compiler designed for use with the Datapoint Disk Operating Systems, (DOS "dot" Series).

This manual describes the run-time characteristics of the DATABUS 11 Interpreter. For a description of the DOS DATABUS language see the DATABUS compiler manual (DBCMP).

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

DATABUS 11 is similar to the Datapoint DATASHARE 3 Multiple Terminal computer system. The primary difference is that DATASHARE 3 supports multiple remote terminals whereas DATABUS 11 supports only the processor console as an operator input/output device. DATABUS 11 also handles a high-speed line printer or servo printer and provides indexed-sequential as well as random and sequential file accessing, thus providing a powerful data entry and processing facility.

In addition, Datapoint DOS with its variety of utility and higher level language systems may be used in conjunction with DATABUS 11, enabling processing of tasks not appropriate to the DATABUS language.

Using virtual memory techniques, DATABUS 11 allows programs with a 16K byte area for executable statements. This, in combination with the ability of the compiler to accommodate over 3400 labels, enables the user to create and use large high level language programs. To provide rapid program execution, the data area of the executing program is maintained in main memory and not swapped.

Any of the Datapoint system printers may be connected to the DATABUS 11 configuration. Printer output is buffered to allow maximum program execution speed.

All program execution in DATABUS 11 occurs in the DATABUS language. Console command interpretation may be handled in a special DATABUS program, the MASTER program which enables the user to completely define his own console command and security system.

Program generation is performed under the Datapoint Disk Operating System, using the general purpose DOS editor and DOS DATABUS Compiler, DBCMP.

CHAPTER 2. SYSTEM OPERATION

This chapter discusses loading the DATABUS 11 System on Diskette under DOS.C or on Cassette under DOS.A,.B, .C, .D, or .E and the use of the DATABUS 11 Interpreter. The use of the DATABUS 11 Compiler is discussed in the Program User´s Guide for DBCMP.

2.1 System Loading

The DATABUS 11 System is available on either Cassette or Diskette media. The DOS files furnished with DATABUS 11 are the Interpreter: DB11/CMD, DB11/OV1 thru DB11/OV6 and DBBACK/CMD.

2.1.1 Loading From Cassette

The DATABUS 11 system programs are contained on one cassette. The cassette is in the DMF (DOS Multiple File) format which includes a directory of the files on the tape. To load the DATABUS 11 files to Diskette, keyin:

```
MIN ;A
```

The MIN (Multiple IN) program will be activated and will display the date of creation of the tape, the file names in the tape directory, and each file name as the file is being loaded. If the file already exists on the diskette, the MIN program will ask if it is to be overstored. The operator can decide to overstore the file or can tell it not to overstore the file in which case MIN will allow the file to be stored under a different name. Consult the DOS User´s Guide for further information on its operation.

The DATABUS 11 Interpreter system files can be re-named to any name desired as long as the command file and all the overlays have the same name. For example, if DB11/CMD was re-named DB/CMD, then DB11/OV1 thru DB11/OV6 would have to be named DB/OV1 thru DB/OV6.

2.1.2 Loading from Diskette

If the DATABUS 11 System is obtained on Diskette media, additional copies of the system should be generated for backup purposes using the DOS commands; DOSGEN, COPY, and/or BACKUP.

2.2 Program Execution

If the DATABUS 11 Interpreter is named DB11/CMD then a DATABUS 11 program (eg. "PROGA") can be executed by entering:

```
DB11 PROGA
```

The DATABUS program compiled and filed under the name PROGA/DBC will begin execution. This program will continue executing until an irrecoverable or untrapped error is detected or until a STOP instruction is executed. At this time system control will return to the DOS.

The general form for the DATABUS 11 Interpreter command is:

```
DB11 [<object>][;<parameters>]
```

If a DATABUS 11 program is not specified, i.e., entering only:

```
DB11
```

will cause the interpreter to assume a default program name of "MASTER", search for a program cataloged as MASTER/DBC and begin executing this program.

The very first program to begin execution under DB11 is considered to be the "master" program, regardless of what name it has. This program will continue execution until a STOP is executed, at which time control will return to the DOS.

The "master" program can cause another DATABUS 11 program to begin execution through the use of the CHAIN instruction. In this case, when this new program executes a STOP instruction or has an irrecoverable or untrapped error, control is transferred to the start of the master program instead of to DOS.

At initialization the interpreter verifies that a file named "ROLLFILE/SYS" exists. If this file does not already exist, the interpreter will attempt to create a 75 sector file by that name.

Due to DOS sector allocation conventions the actual number of sectors required may be somewhat larger. If the required number of sectors is not available, the DOS error message for insufficient file space will be displayed.

Two parameters are available for setting the Databus internal clock. These are the clock set parameter and the date set parameter. The clock set parameter has the form:

```
Chh:mm:ss
```

where hh:mm:ss is the time used on a 24 hour clock. Note that both :mm and :ss are optional, and if omitted are assumed zero. Hence, in order to set the time to 19:25, the parameter would be C19:25 . The date set parameter has the form:

```
Dyy.ddd
```

where yy is the last two digits of the year (75 thru 99) and ddd is the Julian date. For example, March 3, 1977 would be specified as

```
D77.Ø62
```

2.3 Special Programming Considerations

1) The "CONSOLE" statement is inoperative in DATABUS 11 and will be ignored by the interpreter.

2) When a program chained to by the master program has had an untrapped or irrecoverable error, the error message displayed on the console is also copied into the first 11 bytes of the data area. If the master program has an 11 byte string variable declared common, as the first variable in the data list, the error message will be available to it for inspection upon being restarted.

APPENDIX A. INSTRUCTION SUMMARY

SYNTACTIC DEFINITIONS

<label>	is a letter, followed by any combination of up to seven (7) letters and digits.
<string>	is any sequence of characters with the exception of the forcing character (#) which itself will not become part of the character sequence. The character following the # will become part of the character sequence (eg. another # or ").
<svar>	A name assigned to a statement defining a character string variable.
<nvar>	A name assigned to a statment defining a numeric string variable.
<ssvar>	A name assigned to a statment defining a source character string variable. This variable is unchanged as a result of the instruction.
<dsvar>	A name assigned to a statement defining a destination character string variable. This variable is generally changed as a result of the instruction.
<snvar>	A name assigned to a statement defining a source numeric string variable. This variable is unchanged as a result of the instruction.
<dnvar>	A name assigned to a statement defining a destination numeric string variable. This variable is generally changed as a result of the instruction.
<slit>	is a literal of the form "<string>".
<nlit>	is a literal of the form "<string>" where string is a valid numeric string.
<char>	is any single character of the form

	"<string>" where string is of length one (1).
<occ>	is an octal control character (ØØ1 to Ø37 inclusive).
<list>	Any combination of <slit>, <occ>, <list controls> <nvar> and <svar>.
<nlist>	A list of numeric variables each pair of which is separated by a comma (,). The list may be continued on more that one line by placing a colon (:) after the last variable on the line to be continued.
<slist>	A list of character string variables each pair of whihc 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 of to be continued.
<blist>	The name assigned to the first of a set of physically contiguous numeric string or character string variables.
<index>	A numeric variable used in connection with list accessing.
<dnum>	A decimal number between -128 and 127.
<dnum1>	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 2Ø inclusive.
<flag>	One of the following flags: OVER, LESS, ZERO, or EOS (EQUAL and ZERO are two names for the same flag) that are used to indicate the result of some DATABUS operation.

<event>	The occurance of a program trap: PARITY, RANGE, FORMAT, CFAIL, or IO.
<skey>	A numeric or character string variable used with SEARCH.
<DOS file spec>	A DOS compatible file specification (see DOS user´s guide).
<file>	A name assigned to a FILE declaration.
<ifile>	A name assigned to a IFILE declaration.
<rn>	A numeric variable which contains a positive record number (>=Ø) used to randomly READ or WRITE a file.
<seq>	A numeric variable which contains a negative number (<Ø) used to READ or WRITE a file sequentially.
<key>	A non-null string variable used as a key to indexed I/O accesses.
<null>	A null string variable used as a key to an indexed read.

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>
<label>  EQUATE    <dnum>
         INC       <DOS file spec>
         INCLUDE   <DOS file spec>
```

FILE DECLARATIONS

```
<label>  FILE
<label>  IFILE
```

DATA DEFINITIONS

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

   CONTROL

         GOTO     <label>
         GOTO     <label> IF <flag>
         GOTO     <label> IF NOT <flag>
         BRANCH   <index><prep><list>
         CALL     <label>
         CALL     <label> IF <flag>
         CALL     <label> IF NOT <flag>
         RETURN
         RETURN   IF <flag>
         RETURN   IF NOT <flag>
         STOP
         STOP     IF <flag>
         STOP     IF NOT <flag>
         CHAIN    <svar>
         CHAIN    <slit>
         TRAP     <label> IF <event>
         TRAPCLR  <event>
         ROLLOUT  <svar>
         ROLLOUT  <slit>
         PI       <dnum3>
         TABPAGE
```

CHARACTER STRING HANDLING

MATCH	<svar><prep><svar>
MATCH	<slit><prep><svar>
MOVE	<ssvar><prep><dsvar>
MOVE	<sslit><prep><dsvar>
MOVE	<ssvar><prep><dnvar>
MOVE	<sslit><prep><dnvar>
MOVE	<snvar><prep><dsvar>
MOVE	<snlit><prep><dsvar>
APPEND	<ssvar><prep><dsvar>
APPEND	<sslit><prep><dsvar>
APPEND	<snvar><prep><dsvar>
CMOVE	<ssvar><prep><dsvar>
CMOVE	<char><prep><dsvar>
CMOVE	<occ><prep><dsvar>
CMATCH	<svar><prep><dvar>
CMATCH	<char><prep><dvar>
CMATCH	<svar><prep><char>
CMATCH	<svar><prep><occ>
CMATCH	<occ><prep><dvar>
BUMP	<dsvar>
BUMP	<dsvar><prep><dnum>
RESET	<dsvar><prep><dnum>
RESET	<dsvar><prep><ssvar>
RESET	<dsvar><prep><snvar>
RESET	<dsvar>
ENDSET	<dsvar>
LENSET	<dsvar>
CLEAR	<dsvar>
EXTEND	<dsvar>
LOAD	<dsvar><prep><index><prep><slist>
STORE	<ssvar><prep><index><prep><slist>
STORE	<sslit><prep><index><prep><slist>
CLOCK	TIME<prep><dsvar>
CLOCK	DAY<prep><dsvar>
CLOCK	YEAR<prep><dsvar>
TYPE	<svar>
SEARCH	<skey><prep><blist><prep><nvar><prep><dsvar>
REPLACE	<ssvar><prep><dsvar>
REP	<ssvar><prep><dsvar>
REPLACE	<sslit><prep><dsvar>
REP	<sslit><prep><dsvar>

ARITHMETIC

```
      ADD       <snvar><prep><dnvar>
      ADD       <nlit><prep><dnvar>
      SUB       <snvar><prep><dnvar>
      SUB       <nlit><prep><dnvar>
      SUBTRACT  <snvar><prep><dnvar>
      SUBTRACT  <nlit><prep><dnvar>
      MULT      <snvar><prep><dnvar>
      MULT      <nlit><prep><dnvar>
      MULTIPLY  <snvar><prep><dnvar>
      MULTIPLY  <nlit><prep><dnvar>
      DIV       <snvar><prep><dnvar>
      DIV       <nlit><prep><dnvar>
      DIVIDE    <snvar><prep><dnvar>
      DIVIDE    <nlit><prep><dnvar>
      MOVE      <snvar><prep><dnvar>
      MOVE      <nlit><prep><dnvar>
      COMPARE   <nvar><prep><nvar>
      COMPARE   <nlit><prep><nvar>
      LOAD      <dnvar><prep><index><prep><nlist>
      STORE     <snvar><prep><index><prep><nlist>
      STORE     <nlit><prep><index><prep><nlist>
      CHECK11   <svar><prep><svar>
      CK11      <svar><prep><slit>
      CHECK1Ø   <svar><prep><svar>
      CK1Ø      <svar><prep><slit>
```

INPUT/OUTPUT

```
      KEYIN     <list>[;]
      DISPLAY   <list>[;]
      BEEP
      PRINT     <list>[;]
      RELEASE
      PREPARE   <file>,<svar|slit>
      PREP      <file>,<svar|slit>
      OPEN      <file|ifile>,<svar|slit>
      CLOSE     <file|ifile>
      WRITE     <file>,<rn|seq>;<;|<list>[;]>
      WRITE     <ifile>,<rn|seq|key>;<;|<list>[;]>
      WRITAB    <file>,<rn|seq>;<;|<list>[;]
      WEOF      <file|ifile>,<rn|seq>
      UPDATE    <ifile>;<;|<list>[;]
      READ      <file>,<rn|seq>;<;|<list>[;]>
      READ      <ifile>,<rn|seq|key|null>;<;|<list>[;]>
      READKS    <ifile>;<;|<list>[;]>
      DELETE    <ifile>,<key>
```

```
INSERT    <ifile>,<key>
```

APPENDIX B. INPUT/OUTPUT LIST CONTROLS

CONTROL	USED IN	FUNCTION
*P<m>:<n>	KD	Causes the cursor to be positioned horizontally and vertically to the column and line indicated by the numbers <m> (horizontal 1-8Ø) and <n> (vertical 1-12). These numbers may either be literals or numeric variables.
*N	KDP	Causes the cursor or printer to be positioned in Column 1 of the next line.
*EL	KD	Causes the line to be erased from the current cursor position.
*EF	KD	Causes the screen to be erased from the current cursor position to the bottom of the display.
*ES	KD	Causes the cursor to be positioned at horizontal position 1 of the top row of the display and the entire display to be erased.
*EOFF	K	Prevents character echo to the display during Keyboard input operations.
*EON	K	Causes character echo to the display during Keyboard input operations.
*+	KDP	Turn on Keyin Continuous for KEYIN or suppression of spaces after the logical length for DISPLAY and PRINT.
*+	W	Turn on space compression during WRITE.
*-	KDP	Turn off Keyin Continuous (turned off at the end of the statement) or the suppression of spaces after the logical length.
*-	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 or WRITAB operations.
*<nvar>		The logical file pointers are moved to that character position relative to the current physical record.
;	KDP	Suppress a new line function when occurring at the end of a list.
*F	P	Causes the printer to be positioned to the top of form.
*L	KDP	Causes a linefeed to be displayed or printed.
*C	KDP	Causes a carriage return to be displayed or printed.
*T	K	Time out after 2 seconds have elapsed between successively entered characters for KEYIN statement.
*W	KD	Pause for one second.
*JL	K	Left-justify numeric variable and zero-fill at right if there is no decimal point. Left justify string variable and blank fill to ETX (zero fill if *ZF option is given).
*JR	K	Right-justify string variable and blank-fill at left (zero fill if *ZF option is given).
*ZF	KDPW	zero-fill string or numeric variable.
*DE	K	Restrict string input to digits (Ø-9) only.
*IT	K	Turn-on Text Mode (invert alphabetic input).
*IN	K	Turn-off Text Mode.
*MP	W	Convert numeric variable to

"Minus-overpunch" format.

APPENDIX C. ERROR CODES

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

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

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

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

Note that the last three items shown above cannot be trapped. The B error will only show up if somehow an invalid object file is executed or if the system is failing. The U error will happen if the programmer forgets to perform a call or in some other fashion manages to execute a RETURN instruction without a corresponding CALL having been previously executed, or calls are nested more than eight levels deep. The A error will happen if a PI instruction is executed while interrupts are currently prevented.

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

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

APPENDIX D. INTERPRETER I/O TRAP CODES

A - an access sequentially by key was attempted before any indexed sequential access was made using the logical file.
B - the READ mechanism ran off the end of a sector without encountering a physical end of record character (ØØ3).
C - an operation on a closed logical file was attempted.
D - a non-READ non-DELETE indexed sequential operation was attempted where the specified key already exists in the index.
E - an EOF mark without at least four zero´s was encountered.
F - insufficient file space available.
I - the index file specified in an OPEN statement does not exist on the specified drive(s).
J - the index file found by the OPEN statement does not reside in the correct physical location on the disk (index files may never be moved, they must always be re-created).
K - a null key was supplied in an operation where the key may not be null.
M - the data file specified in the OPEN statement does not exist on the specified drive(s).
N - the data file name specified in the OPEN or PREPARE statement was null.
O - the index file name specified in the OPEN statement was null.
P - the file specified in the PREPARE statement had some type of DOS protection (either write or delete) or; an I/O statement tried to modify a file that had DOS write protection.
T - the tab value in the READ or WRITAB statement was off the end of the sector.
U - an EOF mark was encountered while a record was being deleted in the indexed sequential file.
V - one of the DATABUS overlay files could not be loaded by the DOS loader.
W - an index file pointer sector could not be read.
X - an index file header sector could not be read.
Y - the R.I.B. of the data file pointed to by the index file could not be read. (VWXY errors can be caused by parity errors, the drive being switched off line, or the disk cartridge being swapped with another while an operation is taking place.)