BASIC-2 RELEASE 3.1

FUNCTIONAL SPECIFICATION

DATE: 22 October 1987
Updated on June 10, 1988

REVISION: 3.1.20
Final Update

EXPANDED MEMORY

1, 2, 4 and 8 MEG CPUs

Release 3.1 has been modified to support the new 1, 2, 4 and 8 Megabyte CPU options. The changes allow for up to 1 MB being allocated for program partitions (1 @ 61KB and 15 @ 56KB). The remaining memory is allocated to the RAMdisk which was introduced in Release 3.0.

This change also resulted in modifications to the microcoded menu file (@@), the partition generation program (@GENPART), the partition status program (@PSTAT), the field service diagnostic menu (@), the CPU instruction exerciser diagnostic (@A) and the Data Memory diagnostics (@C).

DISK OPERATIONS

FILE#

Release 3.1 of Multiuser BASIC-2 increases the number of file numbers allowed to a maximum of 256 (#0 - #255). To use file#'s . #15, the user must include a SELECT #n statement in the BASIC-2 program, where n is an integer specifying the largest file# that will be used. During Resolution phase, the system determines the largest file# specified and allocates memory from the user partition for each file# from #16 - largest file#. This effectively reduces the size of the user partition. Each file# is allocated 8 bytes of memory, when a CLEAR or LOAD RUN is executed, the memory allocated for file#'s is released back to the partition.

LIST DC T W

A new feature has been added to the LIST DC T statement. The W option when invoked results in only the names of the files on the specified disk to be displayed accross the display or printer line. No index size information or detailed file information (i.e. start and end sector) is provided. This option may be used in conjunction with the file name and file type mask options.

PROGRAM ENTRY AND DEBUGGING

LIST DT

Due to the increase in the number of available device table entries from 16 to a maximum of 256, the information displayed by LIST DT has been modified. The modification presents itself to the user by NOT displaying any unused device table entries.

LIST COM/DIM

Another method for listing defined variables has been added to the current methods of LIST COM and LIST DIM. LIST COM/DIM lists both the common variables and noncommon variables, their defined length and values in their order of definition. This is a useful aid when making use of the COM CLEAR statement to re-position the common variable pointer to clear out noncommon variables when succeeding programs are loaded.

PRINTER DRIVERS

The Generalized Printer Driver has been enhanced to better support printers which expect ASCII escape sequences instead of hexidecimal. Further optimization of the printer driver logic has improved output performance for devices not directed through a printer driver.

@LASRJV0 is a new printer driver table for the LDP8 and LCS15 printers.
@DM50/V0 is a new printer driver table for the DM5300 printer.

CORRECTED ANOMALIES

RESAVE

RESAVE facilitates updating program files by combining the SCRATCH and SAVE statements. In release 3.1, the existing file will be marked as SCRATCHed prior to attempting to saving the program. This approach exactly duplicates the results of a SCRATCH and a SAVE. For the case when the new file requires more sectors than previous, the file will be marked as SCRATCHed and an error I81 (File Full) is generated.

Release 3.0 would generate the the appropriate I81 error message but did not mark the file as SCRATCHed.

RENAME

RENAME changes the name of a file in both the catalog index as well as the program header block stored with the file. Release 3.0 had the limitation that it would not allow the renaming of SCRATCHed files. This limitation has been removed in release 3.1

MOVE

In release 3.0, the MOVE statement was enhanced to allow one to specify the size of the index and catalog area to be created on the destination platter. However, no check was made to determine if the MOVE would exceed the user specified limits. Release 3.1 has modified the MOVE statement to generate the appropriate error message in the event the user specified limits are to be exceeded.

MOVE FR

In release 3.0, the 2200T MOVE FR statement would not properly move the catalog from the source to the destination platter. In release 3.1 the statement has been corrected.

DATA LOAD BA/BM

In release 3.0, then DATA LOAD BA/BM statments had a weakness in their syntax checking logic. One case which was not properly handled was -

DATA LOAD B AT (space between the B and the A)

Release 3.1 has been modified to correctly check the syntax of the DATA LOAD BA/BM statements.

IF THEN ELSE (IMMEDIATE MODE ONLY)

The IF THEN ELSE statement was not properly handled in release 3.0 when invoked in Immediate Mode. This problem has been fixed in release 3.1.

LONG LINE ERROR

In release 3.0, entering a program line that was too long was not properly flagged with an error, and has been corrected in 3.1.

$BREAK!

$BREAK! put a partition to sleep permanantly and could not be woken up by issuing a $ALERT from another terminal. This has been changed to allow $ALERT to wake up such a partition as in earlier releases.

$GIO

In release 3.0, certain $GIO sequences would hog the disk and not release the disk at the end of the sequence. This has been changed in 3.1 to release the disk.

SELECT PRINT /000

Release 3.0 inadvertantly slowed down output to the null device, /000. This has been corrected in release 3.1.

COM CLEAR

Executing a COM CLEAR with a variable name when selected to a global partition caused the system to hang. This has been corrected in release 3.1

@MOVEFIL

The @MOVEFIL utility had the following two problems corrected:

When writing to an IBM 3741 diskette format, an invalid trailer record was created.

When an @SPAN file was created with exactly two free sectors, a program error occurred.

@FORMAT

@FORMAT displayed an inaccurate error message when attempting to format a device that was being accessed by another partition. This error message has been corrected.

@BACKUP

@BACKUP displayed an inaccurate error message if an attempt was made to backup to an unscratched platter. This has been corrected.

@INSTALL

@INSTALL displayed an inaccurate error message if an attempt was made to install the operating system to an unscratched platter. This has been corrected.

MXE REV 3.02

The @MXE0 microcode has been modified to correct two reported problems. The fixed problems are -

1) In TC mode, when sending 6-bit or 5-bit data - if a character shifted relative to the previous character was sent to the MXE, the shift code was transmitted, but character itself waited until another character was sent to the MXE. Now all characters are transmitted as sent to the MXE.

2) When using the 2436WP terminal, under certain circumstances the print stream to the local printer (204) would stop and not resume until the 2200 was re-booted. This has been fixed.

Format:

LIST [S] [title] DT

where:

title = alpha-variable or literal-string

LIST DT lists the contents of the Device Table. The current device selections for the various I/O classes and file numbers are shown. The line width for character output devices is also included.

For data files that are open, the device type in the disk-address of the associated file number indicates whether the file was opened with the T, F, or R platter specification. Device type D indicates T, 3 indicates F, and B indicates R. For example, if a data file has been opened as follows:

```
SELECT #1/320: DATALOAD DC OPEN R #1, "PLAYERS"
```

then, LIST DT displays the selection as: #1 /B20. LIST DT also displays the current sector location within the open data file. For example, for the file above the display might be:

```
#1    /B20  at 530 in 500 to 732
```

This shows that the current location is sector 530 in the data file that begins at sector 500 and ends as sector 732.

LIST DT also lists the device entries in the Master Device Table (MDT). Each entry is displayed in the following format:

```
/taa[-ppx]   where:  t = device type (3 if disk, otherwise 0)
                    aa = device address
                    pp = number of the partition using the device
                     x = X if device open exclusively for partition
                         pp, or 0 if open for partition pp
```

The Printer Device Table (PDT) is also listed by LIST DT. An entry for each available printer supported by the Generalized Printer Driver is displayed. Each entry includes the printer address, name of the printer table, and whether the printer table is enabled (ON) or disabled (OFF).

The S and title parameters of LIST DT are the same as for the other LIST statements.

Example:

```
:LIST DT

CI      /001                                    CO      /005    width 80
INPUT   /001                                    PRINT   /215    width 132
PLOT    /413                                    LIST    /005    width 80
TAPE    /000

#0      /D11                                    #2      /D10
#5      /B20    at 530 in 500 to 732            #6      /D1F
#7      /D12    at 1234 in 1234 to 1945         #10     /D31
#15     /320    at 12222 in 11946 to 12223      #128    /D5F

MDT:    /004        /215-120    /310        /320-03X

PDT:    /015 @PM010V0 ON    /016 @PM0120V0 OFF
```

Format:

```
RESAVE [DC] [<[W][S][R]>] T [$] [file#,     ] [!] file-name [start] [,[end]]
                            [disk-addr,] [P]
```

where:

```
file-name = alpha-variable or literal-string
start     = starting line #
end       = ending line #
```

RESAVE saves the program in memory over an existing file. The file name specifies the name of the file to be overwritten and becomes the name of the program file on disk. The file to be overwritten can be active or scratched, and can be either a data or a program file. If the file does not exist or is not large enough to hold the program, an error results. If the file is not large enough to hold the program, it is marked in the catalog index as a SCRATCHed file.

RESAVE operates as a combination of the two statements SCRATCH file and SAVE over an existing file. The parameters in the RESAVE statement are the same as for SAVE. See SAVE for a description of the save parameters.

Examples of valid syntax:

```
RESAVE T "CONVERT"
RESAVE T ! "MONEY" 1000,2000
RESAVE T/D22, F$
RESAVE T#1, "LAYOVER" 100
RESAVE <SR> T $ "PROG"
RESAVE <R> RP "RP" ,200
RESAVE DC <S> F $ /D25, P A$ 100,900
```

Format:

```
RENAME platter [file#,     ] old-file-name TO new-file-name
               [disk-addr,]
```

where:

file-name = alpha-variable or literal-string

RENAME changes the name of an existing file to some other name. After the file has been renamed, the file can only be accessed by using the new file name. The program header, which is saved along with the program, is also modified to reflect the name change. This was done to ensure that various platter recovery utilities would recover the file using the new name. Except for the program header, the file is not modified, occcupying the same sectors as it did before the rename.

Examples of valid syntax:

```
RENAME T "OLDFILE" TO "NEWFILE"
RENAME F/320, O$ TO N$
RENAME T/D10, "TEST001" TO A$
```

Format:

```
MOVE platter [file#,     ] TO platter [file#,     ] [LS=expr1] [,END=expr2]
             [disk-addr,]             [disk-addr,] [END=expr2]
```

where:

expr1 = an expression such that 1 <= value <= 255

expr2 = an expression whose value <= highest sector address on platter

---

The MOVE (platter) statement copies all active files from the first platter specified to the second platter specified. Scratched files and temporary files are not copied. Thus, MOVE provides a means to recover space lost to scratched files by creating a new platter containing only the active files on the old platter.

Before files are copied, MOVE (platter) scratches the destination platter. The LS parameter specifies the size of the Index to be created on the destination platter. If LS is not specified the size of the Index will be the same as the Index on the source platter. The END parameter specifies the size of the Catalog Area on the destination platter; the value of expr2 is the highest sector address in the Catalog Area. If END is not specified, the highest sector address is the same as on the source platter.

If the LS and END parameters are not specified, the type of Index created is the same as that of the source platter (ie, a SCRATCH DISK or SCRATCH DISK ' index). If LS and/or END is specified, a SCRATCH DISK ' index is created.

Prior to attempting to move a file from the source to the destination platter, the statement determines whether there is enough room in the destination catalog to accomodate the file. If not, the MOVE statement is terminated with the appropriate error message.

Following a MOVE, the VERIFY statement can be used to ensure that the files were recorded without error. Note, MOVE does not modify the source platter.

To execute a MOVE, approximately 800 bytes of memory must be available for buffering (not occupied by a BASIC-2 program or variables); otherwise an error A03 results and the MOVE is not performed. The large buffer minimizes the time required for the MOVE operation.

Examples of valid syntax:

```
MOVE F TO R
MOVE T/D65, TO T/D60,
MOVE T#1, TO T#2, LS=4, END=1279
MOVE T TO T/B20, END=10000
```

Format:

```
LIST [S] [title] DC platter [file#,     ] [file-name-mask] [file-type] [W]
                            [disk-addr,]
```

where:

```
title = alpha-variable or literal-string
mask  = alpha-variable or literal-string

                ( P)
file-type = ( D)  ...
                (SP)
                (SD)
```

W = output file names only option

LIST DC lists the contents of the specified disk platter. LIST DC first shows information regarding the size of the Catalog Index and Catalog Area followed by a listing of files on the platter. For example,

```
:LIST DC T

INDEX SECTORS = 0005'
END CAT. AREA = 1231
CURRENT END   = 0089

NAME      TYPE START  END  USED  FREE
DATA-LL     P  00006 00027 00022 00000
2231W       P  00028 00030 00003 00000
XPRINT     SP  00031 00033 00003 00000
JUNK       SD  00034 00043 00009 00001
INVTORY     D  00044 00089 00020 00026
```

INDEX SECTORS is the number of sectors allocated for the Catalog Index. The single quote (') after the number of index sectors indicates that the index was created with the SCRATCH DISK ' statement, the (') is not displayed for indexes created with SCRATCH DISK. END CAT. AREA is the sector address of the last sector reserved for storing files. CURRENT END is the last sector that has been used.

For each cataloged file LIST DC shows the file name, file type, starting and ending sector addresses of file, the number of sectors used in the file, and the number of sectors not used. The file type is either;

```
 P  for program file
 D  for data file
SP  for scratched program file
SD  for scratched data file
 ?  for an invalid entry in the index
```

For data files, the number of sectors currently used in the file is originally set to one, and is updated only when an end-of-file record is written to the file.

The W option (for wide) directs the system to use a different output display. This option results in only the file names being displayed across the screen. For example,

```
:LIST DC T W

DATA-LL   2231W     XPRINT    JUNK      INVTORY
```

Format:

```
              COM
  LIST [title] DIM
              COM/DIM
```

where:

title = alpha-variable or literal-string

LIST COM, LIST DIM, LIST COM/DIM statements list the currently defined variables and their current values. LIST COM lists the defined common variables. LIST DIM lists the defined noncommon variables. LIST COM/DIM lists the defined common variables in their order of definition, and then lists the defined noncommon variables in their order of definition. The dimensions of arrays and the length of alpha variables are shown as would appear in a DIM or COM statement.

Values of alpha variables are displayed in both ASCII and hex notation. Nonprintable characters (i.e., hex(00)-hex(0F)) are displayed as periods (.) in the ASCII field. If the value is long, only the first 16 characters are displayed. Alpha array values are displayed as a single string starting at the first element.

For numeric arrays, as many elements that fit on a single line are displayed. The elements in row 1 are displayed, then row 2, etc.

Examples:

```
:LIST DIM
A              123.45
B1             0
B2(5)          -1 2 0 0 0
B$6            "AB..CD" 41 42 0D 0A 43 44
M$(256)1       "Wang Laboratories" 57 61 6E 67 20 4C 61 62 6F 72 61 74 6F
N(3,4)         .874539284 .777430912 .314985239222 -.0002438216 .10138327

:LIST COM/DIM
F$7            "house  " 6B 6F 75 73 65 20 20
C              33
P(4)           0 1 3 0
H$(3)2         "CNLLAL" 43 4E 4C 4C 41 4C
---------------
Y$5            "     " 20 20 20 20 20
J0(10)         6 4 1 0 0 0 0 0 0 0
Q$11           "001-00-1234" 30 30 31 2D 30 30 2D 31 32 33 34
```

Examples of valid syntax:

```
LIST COM
LIST DIM
LIST COM/DIM
LIST "title" DIM
```