# WINDRUSH FLEX UTILITIES PACKAGE



First release: 1 November 1983

## T R A D E M A R K  N O T I C E S

# I N D E X



## UTILITIES



NOTE 1: The 'FASTBACK' utility has been optimized for Windrush implementations of FLEX. We will not guarantee its performance on any other hardware or software environments.

# I N D E X



NOTE 2: These files are part of our standard FLEX package and are included here only for completeness. If you are buying this package of utilities for use on a FLEX system manufactured by someone else these files should NOT be used as they are specifically configured for our hardware.

# WINDRUSH FLEX UTILITIES PACKAGE




Windrush Micro Systems, Worstead Labs, N. Walsham, Norfolk, NR28 9SA, England

## A C K N O W L E D G E M E N T

This entire package of FLEX utilities and the enhanced print spooler were designed, developed and debugged by Neil P. Jarman.

## 1.0 WINDRUSH UTILITIES PACKAGE

Several of the enclosed utilities are functional replacements for the TSC counterparts which we supply with our configured versions of FLEX. Under the terms of our FLEX license we must supply all of the standard utilities from TSC.

If you are an experienced FLEX user you will know that several of the TSC commands that are supplied with FLEX could be improved. The following commands we supply fall into this category.

| TSC COMMAND NAME | ... replaced by ... | WMS COMMAND NAME |
|---|---|---|
| COPY-TSC.CMD | | COPY-WMS.CMD |
| I .CMD | | IN .CMD |
| O .CMD | | OUT .CMD |
| P .CMD | | PR .CMD |
| RENAME .CMD | | RE-NAME .CMD |
| VERSION .CMD | | VER .CMD |
| XOUT .CMD | | XXOUT .CMD |

If you are an experienced FLEX user you probably know that TSC market a package called 'UTILITIES'. Several of the utilities in this package are very useful to the user, but again, with a little effort could be improved. The commands that we supply that fall into this category are as follows:

| TSC COMMAND NAME | ... replaced by ... | WMS COMMAND NAME |
|---|---|---|
| DIR .CMD | | DIR .CMD |
| MAP .CMD | | MAP .CMD |
| MEMEND .CMD | | MEMEND .CMD |
| N .CMD | | N .CMD |
| NAME .CMD | | NAME .CMD |
| PDEL .CMD | | PDEL .CMD |
| RUN .CMD | | LOAD .CMD |
| Y .CMD | | Y .CMD |
| ZAP .CMD | | ZAP .CMD |

## P L E A S E  N O T E

DON't get the impression that the above mentioned utilities we are supplying are simply patched versions of the TSC utilities; each utility has been designed from scratch to do a specific task, usually with more options and refinement than the original utility!

## 1.0 WINDRUSH UTILITIES (continued)

The next group of utilities are entirely new:

```
CLEAN   .CMD
DATECOPY.CMD
DATIME  .CMD
DUMP    .CMD
F       .CMD *
FASTBACK.CMD
FDATE   .CMD
FREEMAP .CMD
MAKECMD .CMD
PCOPY   .CMD
PRINTT  .CMD *
PRT     .CMD *
PROMPT  .CMD
QCHECKK .CMD *
QCHECK  .SYS *
S       .CMD *
SETCLOCK.CMD
SETFORM .CMD *
SPLTITLE.CMD *
SPOOL   .SYS *
SPOOL-XX.CMD *
SKIP    .CMD *
YEAR    .CMD
```

* ... All utilities thus marked are part of the enhanced printer spooler package. The utility entitled 'SPLTITLE.CMD' is currently scheduled for addition to this package in mid '85.

One feature common to ALL Windrush utilities (including our FORMATTER) is that if you wish to abort the command all you have to do is hit ^C (CONTROL-C). Most utilities will abort immediately. Some utilities, like DATECOPY, ZAP, XXOUT, etc., will only abort after they have completed the current operation and have closed all files correctly.

## 2.0 ENHANCED SPOOLER

This release of the print spooler for FLEX (tm of Technical Systems Consultants, Inc.) is COMPLETELY different code than the original version. The spooler itself has evolved over some two to three years continually going from good to better.

All the support utilities are also original work, though those that replace the original versions supplied with FLEX still retain the same user-interface. In view of the extensive effort and investment put in by the author the source code for the spooler and utilities will not be released.

Some of the improvements to the spooler are: automatic seeking to the top of the next page after a file has been output, intelligent handling of the N(ext) file command in QCHECK to ensure that the next printed file really is the requested one, ability to F(reeze) the spooler immediately in QCHECK, skipping to a given page number (SKIP) if the system had to be killed while spooling (i.e. looping in a user process), devolving of most of the FMS function into the spooler thereby minimising the interruptions to foreground disc activity, increase in the number of queued files (20 in release 7.0), 'hooks' for printing a title/banner page.

The following files are provided in the spooler package:

```
F        .CMD
OUT      .CMD
PRINTT   .CMD
PRT      .CMD
QCHECKK  .CMD
QCHECK   .SYS
S        .CMD
SETFORM  .CMD
SKIP     .CMD
SPLTITLE.CMD
SPOOL    .SYS
SPOOL-XX.SYS
XXOUT    .CMD
```

## 2.1 SPOOLER INSTALLATION

Suggested installation of the spooler is to GET 'SPOOL.SYS' and execute 'SPOOL-XX.CMD' as part of your startup file. Ensure that QCHECK.SYS is residing on your system drive as this is the help file for QCHECK.

DO NOT attempt to install the enhanced printer spooler until you have the standard printer spooler supplied by TSC operating. Generally all that will be required is to integrate a spooler timer handler similar to 'SPOOL-XX' into your system. The 'PRINT' and 'QCHECK' commands should then operate as defined by TSC.

The enchanced spooler needs the same routines that the original spooler used for initialising, starting and stopping of the interrupt timer source. Similary you should have an interrupt handler which determines if the spooler needs servicing or if another device caused the interrupt.

Finally the console driver table should reflect the location of both your IRQ and SWI3 vectors and the handler routine. These will be exactly the same routines as those required for the TSC print spooler.

A brief synopsis of the function of each of the files follows:

| | | |
|---|---|---|
| F | .CMD | is used to Format a standard FLEX text file into a spoolable version. Optional titles can be included. |
| OUT | .CMD | this utility produces an file for submission to the spooler. As with the new P it honours the SETFORM parameters. |
| PRINTT | .CMD | this is the utility used to submit files to the spooler for subsequent printing. If the spooler is inactive, this utility will also start it running. |
| PRT | .CMD | this is a new version of P and PR which is included here for completeness as it draws on a data table within the spooler. Using PRT will format the output to the parameters set using SETFORM (q.v.). |
| QCHECKK | .CMD | this is the spooler management utility which allows changes to both the spooler itself and/or the queue. Note this utility requires that the file called 'QCHECK.SYS' resides on the system drive. |
| QCHECK | .SYS | this is a help file for use with the QCHECK utility. |
| S | .CMD | is used to Strip LFs from a spoolable file. It's main use is for producing a file to be merged into a word processor. |
| SETFORM | .CMD | allows various parameters to be modified regarding the printer and/or paper used by the printer. |
| SKIP | .CMD | this is the utility which allows seeking to, say, page 71 of a long document which was terminated in mid-flow. |
| SPLTITLE | .CMD | this prints out a large title banner (when available). |
| SPOOL | .SYS | this is the actual memory resident spooler code. |
| SPOOL-XX | .CMD | this is the timer hardware driver and interrupt handler. |
| XXOUT | .CMD | this utility deletes all .OUT files which are not in the spooler queue. |

## 2.2 FUTURE SPOOLER ENHANCEMENTS

There is one further feature that both QCHECK and the spooler have 'hooks' for and that is printing of a banner page. The code for banners is quite large (1k Byte) and must be system resident. Additionally it must be integrated into the real-time clock which is also system dependant. A special version is planned for the Windrush 3U and 6U micro-computer development system using the TIM2 board with it's HD146818 battery-backed clock-calendar chip. The banner software provides a title page with the file name (no extension) in large text across the paper, correctly justified if it is less than 8 characters. Below the banner is the complete file name, and below that is the day, date and time all in normal size print. This is very useful when multiple copies of some text are produced in the same day as the time reveals the latest version.

This utility, called SPLTITLE, will be available in mid '85. If the package you purchase does not include this utility contact the factory for availability. We will provide it under the terms of our normal upgrade service.

## 3.0 WHICH UTILITIES SHOULD I USE

Whether you decide to use our utilities in preference to the TSC utilities is purely a personal choice. If you decide to use all of our utilities in preference to the TSC utilities you might find it ergonomic to rename the utilities we supply to the names used by TSC. If you have been using FLEX for some time you may find it difficult to stop typing P,CAT 1 instead of PR,CAT,1. Old habits are difficult to break!

If you decide to use our utilities in lieu of the TSC utilities PLEASE format a fresh disk and copy the following TSC files onto it. This way should you ever want to use the TSC utilities again you will have them all in one place.

```
COPY-TSC.CMD
I       .CMD
O       .CMD
P       .CMD
PRINT   .CMD
QCHECK  .CMD
RENAME  .CMD
VERSION .CMD
XOUT    .CMD
```

Likewise make a duplicate copy of the disk we supply with this package. Then rename the files on the duplicate disk as follows:

```
COPY-WMS.CMD    ------------------>   COPY    .CMD
IN      .CMD    ------------------>   I       .CMD
OUT     .CMD    ------------------>   O       .CMD
PR      .CMD    ------------------>   P       .CMD
PRINTT  .CMD    ------------------>   PRINT   .CMD
QCHECKK .CMD    ------------------>   QCHECK  .CMD
RE-NAME .CMD    ------------------>   RENAME  .CMD
VER     .CMD    ------------------>   VERSION .CMD
XXOUT   .CMD    ------------------>   XOUT    .CMD
```

The remainder of the files in this utilities package will not clash in name or function with any of the other standard utilities supplied with FLEX. If you have the optional TSC 'UTILITIES' package the following utilities in our package have similar functions to the files of the same name supplied by TSC.

```
DIR     .CMD
MAP     .CMD
MEMEND  .CMD
N       .CMD
NAME    .CMD
PDEL    .CMD
RUN     .CMD  (we call ours 'LOAD')
Y       .CMD
ZAP     .CMD
```

Again you must choose which utility command you wish to use. If you are ever in doubt as to the origin of a utility use our 'VER' command. It will identify the source of the utility as WMS (us), SWT (South West Technical Products) or TSC (Technical Systems Consultants).

## 4.0 ENHANCEMENTS TO FLEX ITSELF

There is one more file which we have supplied which is called 'FIXES.SYS'. This file contains three patches to the inner workings of FLEX itself. The main areas of improvement are:

1) Typing ^R (CONTROL-R) for 'REPEAT' will print the last command line issued to FLEX and leave the cursor positioned at the end of the line. You can now either type <RETURN> to re-execute the line or back-space and make any desired corrections to the line and then type <RETURN> to execute it. This latter feature is useful when a command syntax error is posted by FLEX as, more often than not, the error will be in the last part of the command line (due to MURPHY's law).

2) Accessing an illegal drive number, i.e. drive #4 on up, will not result in the usual strange behaviour of accessing the drive number specified - 4! i.e. if you type DIR,4 you will get a directory listing of drive #0 if this patch is not in place!

3) Hexadecimal number letters 'A' through 'F' may be typed in either upper or lower case. This patch also applies to the low-level hex input routines described in the FLEX PROGRAMMERS MANUAL.

To incorporate the fixes simply type:

```
+++GET,FIXES.SYS<CR>
```

We suggest that you include the above command in your 'STARTUP' sequence.

# CLEAN

The CLEAN utility is provided to assist in the correct cleaning of the heads on disc drive units. After confirming the prompt, the heads will step in to the centre track and back out again for one minute. This ensures even coverage of the cleaning media, and therefore extends it's useful life.

## DESCRIPTION

The syntax for CLEAN is:

```
CLEAN,<drive>
```

where <drive> is a valid on-line drive.

The command will abort if the <drive> is illegal/invalid. If the call is successful the <drive> will be restored (stepped to track #0) and a prompt is given to insert the cleaning disc into the drive. When this is done, hitting <RETURN> will cause the cleaning to commence.

On completion of the cleaning, the utility prints a confirmatory message and sends a bell code ($07) to the system console.

NOTE: At ANY STAGE including the initial prompt, hitting ^C (control-C) will abort the command and return to FLEX.

```
* * * * * * * * * *
*                 *
*  W A R N I N G  *
*                 *
* * * * * * * * * *
```

There are some very dubious cleaning disks available. Confirm with the disc drive supplier that using a particular cleaning disc on your drive does not invalidate the warranty.

# COPY

The COPY utility allows files to be copied from disc to disc or, if a copy to another name is used, to the same disc, this utility, unlike the TSC 'COPY' utility, does not alter the file date information record. This command is supplied named 'COPY-WMS.CMD' so that it will not conflict with the TSC command. Feel free to rename it.

## DESCRIPTION

There are three ways of invoking COPY:

```
COPY,<drive>,<drive>[,<matchlist>]
COPY,<filespec>,<drive>
COPY,<filespec>,<mfilespec>
```

<drive> is a valid on-line drive,
<matchlist> is either (or both) part(s) (or all) of a filename or extension,
<filespec> is a filename with an extension,
<mfilespec> is a minimum of a filename - with an optional extension.

The first case is the one mostly used. It allows copying of either an entire disc to another, or just matched files from one drive to another. Matching is on either part (or all) of the name (or extension) or both. The <drive>s must be valid on-line drives or the command will abort. Some examples of this type of copy follow:

```
+++COPY,0,1
+++COPY,1,0,.TXT,A_B,STY.T
```

The first example will copy ALL files from drive # 0 to drive # 1.

The second will copy those files from drive # 1 to drive # 0 which have a .TXT extension, then those which start off with the characters A_B, then those that start with STY and have an extension starting with '.T'.

NOTE: If a file exists on the destination drive, a prompt is issued regarding deletion of the file found. If Y is answered an "are you sure" prompt returns and if Y is answered to this one the file will be deleted and the source version copied across. Typing N to either prompt will not delete or copy the file and the directory search is continued. This prompting method holds true for all invocations of COPY.

The second type of call is useful for copying only one file from a drive to another. An example follows:

```
+++COPY,0.P.CMD,1
```

This copies JUST the file P.CMD from drive # 0 to drive # 1. Using:

```
+++COPY,0,1,P.CMD
```

would have copied ALL command files starting with a P - which is not always required !

*** BE AWARE OF THIS DISTINCTION ***

## COPY

(continued)

The last type of call is the one least used. It copies JUST <filename> from the source disc, to the destination disc (may be the same disc) using <mfilename> as the new name. The extension will remain the same if <mfilename> does not include an extension, otherwise the extension will be as requested in <mfilename>. Some examples follow:

```
+++COPY,1.WORK.TXT,NEW_VER
+++COPY,0.P.CMD,0.P_FILE.BIN
```

The first example copies the file WORK.TXT from drive # 1 to a file to be called NEW_VER.TXT on the assigned WORK drive. The second example shows how the extension may also be altered. Here the file P.CMD on drive # 0 is copied to a file called P_FILE.BIN also on drive # 0.

Both <filename> and <mfilename> may exclude the drive number in any calls and the drive will always default to the WORK drive, or drive # 0 if ALL is specified as the WORK drive.

During copying the success, or otherwise, of the copy is displayed on the terminal (unless re-vectored using the I command).

NOTE 1: At any stage, ^C (control-C) may be sent from the terminal. This will cause COPY to terminate in an orderly fashion and return to FLEX.

NOTE 2: TSC supply a utility command of the same name so we supply ours named as 'COPY-WMS.CMD'. The TSC 'COPY' command will always alter the file date information record to reflect the current system date when a file is copied. The WMS 'COPY' command will always preserve the file date information record when the file is copied.

NOTE 3: If you wish to use our 'COPY' command in lieu of the TSC 'COPY' command we suggest that you rename the TSC command to 'COPY-TSC.CMD' and rename our command 'COPY.CMD'.

# DATECOPY

DATECOPY is a utility which allows either updating or archiving of either specific files, or all files, between two nominated drives.

## DESCRIPTION

The syntax of DATECOPY is:

```
DATECOPY,<from>,<to>[,<type>][,<matchlist>]
```

<from> is the source drive,

<to> is the destination drive,

<type> omission means UPDATE EXISTING files only (default)

- `+` means UPDATE ALL files
- `-` means ARCHIVE EXISTING files
- `+-` means ARCHIVE ALL files
- `-+` means ARCHIVE ALL files

<matchlist> is either (or both) part (or all) of a filename or extension.

A simple summary of operating modes is that inclusion of: "+" may be thought of as "adding to" those that already exist, "-" may be thought of as "subtracting from" (i.e. deletion of source file after a successful copy) the source disc.

In UPDATE mode source files are only read, while in ARCHIVE mode source files are subsequently deleted after a successful copy. This is because a successful archive copy has been taken.

Irrespective of the operating mode, a copy is only affected if: the date on the source is more recent than that of the destination, the date is todays date (in case of multiple modifications on one day), the file does not exist on the destination drive AND the ALL rather than EXISTING is in existance.

In the absence of a <matchlist> all files will be checked for processing, while supply of a <matchlist> restricts the processing to files which match the supplied list.

The banner title always informs the user of the mode of operation and if the processing is restricted to a matchlist or not.

Hitting any key will start DATECOPY.

NOTE: Hitting ^C (control-C) at any stage will abort the copy cleanly and return to FLEX.

**********
*** WARNING ***
**********

By ARCHIVE we actually MEAN archive. If a file is write protected, DATECOPY ignores the fact when it deletes the source. The destination file will still retain the protection that existed on the original.

## DATECOPY

(continued)

The following set of examples explain the various calls:

```
+++DATECOPY,0,1
```

This will perform an update from drive #0 to drive #1 of existing files only i.e. UPDATE EXISTING files.

```
+++DATECOPY,2,3,+
```

This will perform an update from drive #2 to drive #3 of existing files plus copying over any that do not already exist on drive #3 i.e. UPDATE ALL files.

```
+++DATECOPY,2,0,-
```

This will copy from drive #2 to drive #0 newer files that already exist on drive #0 followed by deletion of the source file on drive #2 i.e. ARCHIVE EXISTING files.

```
+++DATECOPY,3,2,+,-  ... or ... +++DATECOPY,3,2,-,+
```

This will copy from drive #3 to drive #2 newer files that already exist and those that do not already exist on drive #2, again followed by deletion of the source file on drive #3 i.e. ARCHIVE ALL files.

The above four examples are for bulk copying, however often only a particular type of file may be important. For these cases a matchlist is supplied and the following four examples show how this is done:

```
+++DATECOPY,0,1,.TXT
```

This will update existing files with a .TXT extension, from drive #0 to drive #1 i.e. UPDATE EXISTING matched files.

```
+++DATECOPY,0,2,+,A,.B
```

This will update all files that start with 'A' and then those with an extension that starts with '.B' from drive #0 to drive #2 i.e. UPDATE ALL matched files.

```
+++DATECOPY,3,1,-,DEV.C
```

This will archive existing files that start with DEV and have got an extension starting with .C from drive #3 to drive #1 i.e. ARCHIVE EXISTING matched files.

```
+++DATECOPY,3,2,-,+,.BAS,HELP  ... or ... +++DATECOPY,3,2,+,-,.BAS,HELP
```

This will archive all files that have a .BAS extension and then those that start with HELP from drive #3 to drive #2 i.e. ARCHIVE ALL matched files.

The examples show the various methods in which this useful utility may be used. Before hitting the key to start, check that the banner information is correct. If the wrong action is taken, this is not detrimental as a file can NEVER VANISH from the system. An accidental ARCHIVE will merely re-position it the other drive.

# DATIME

The DATIME command is used to read the current date from a clock-calendar chip. If the data is valid it will be displayed on the terminal and the FLEX system information updated accordingly.

## DESCRIPTION

The syntax for DATIME is:

```
DATIME<CR>
```

There are two basic versions of 'DATIME' available:

DATIME-N ... which is for the National MM58167 type clock-calendar
DATIME-H ... which is for the Hitachi HD146818 clock-calendar

The former is coded in assembly language suitable for MACE and is supplied to match the address of the MM58167 chip on the Windrush 3U-TIM1 board.

The latter is coded in PL/9 and is supplied to match the address of the HD146818 chip on the Windrush 3U-TIM2 board.

You should rename the command file that suits your hardware 'DATIME.CMD'

The source and object files are supplied. The output format of the two commands is somewhat different owing to the ability of a high level language like PL/9 to produce a formatted output much easier than can be done in assembly language.

Invoking the DATIME command for the MM58167 will produce the following format if the data in the clock is correct:

THURSDAY -- 10 NOVEMBER 1983 -- 7:49 PM

the data displayed will also be loaded into the FLEX system information, viz:

SYSMON at $CC0E
SYSDAY at $CC0F
SYSYR at $CC10

It should be noted that the MM58167 does not keep track of the YEAR nor does it make any allowances for leap years. The 'YEAR' displayed is derived from the FLEX 'SYSYR' which is generally set by the 'YEAR' utility, (qv) within the STARTUP (qv) file.

If the software does not find the thousandths and hundredths of seconds counter rolling over it will assume that the TIM1 board has been removed or that the clock has died and present the following message:

- CLOCK NOT PRESENT! -

If the software determines that the data stored in any of the clock registers is not valid the following message will be displayed:

DATA INVALID --- RE-SET CLOCK

## DATIME

(continued)

Invoking the DATIME command for the HD146818 will produce the following format if the data in the clock is correct:

```
* * * * * * * * * * * * * * * * * * * * * * * * *
*                                               *
*     TUESDAY,  8 JANUARY 1985   6:36:59 PM     *
*                                               *
* * * * * * * * * * * * * * * * * * * * * * * * *
```

the data displayed will also be loaded into the FLEX system information, viz:

SYSMON at $CC0E
SYSDAY at $CC0F
SYSYR at $CC10

Since the HD146818 has the ability to maintain the current YEAR this information is read directly from it.

If any of the data read is invalid the following will be displayed:

```
* * ILLEGAL DATA IN CLOCK * *
```

The files present on the disc are as follows:

DATIME-N.ASM ... assembly language source for MM58167 version (TIM1).
DATIME-N.CMD ... FLEX TCA command for MM58167 version.

DATIME-H.PL9 ... PL/9 source for HD146818 version (TIM2).
DATIME.H.CMD ... FLEX TCA command for HD146818 version.

You should use the 'RE-NAME' utility to rename the FLEX TCA command appropriate for your hardware 'DATIME.CMD'

The clock setting up utility, SETCLOCK, is documented seperately.

# DIR

The DIR utility is used to produce a directory listing of all, or selected, files on one, or more, on-line drives. Two options exist: display random and protection flags in lieu of the sector count, display beginning and ending track/sector in lieu of the date.

## DESCRIPTION

The syntax for DIR is:

```
DIR[<option>][,<drives>][,<matchlist>]
```

| | | |
|---|---|---|
| <option> | "\" | means display begin/end track/sector |
| | "/" | means display random/protection flags, |
| <drives> | is: <drive #>[,<drives>] | |
| | <drive #> is a valid on-line drive, | |
| <matchlist> | is: <match>[,<matchlist>] | |
| | <match> is either (or both) part (or all) of a filename or extension (if preceded by a period "."). | |

Omission of <option> defaults to the most commonly required information - date and sector counts. Use of the "\" option yields the begin/end track/sector for each file in lieu of the date. This option is useful for finding out where a file starts and ends on the disc. Using the "/" option yields the random and protection flags in lieu of the sector count.

Omission of <drives> defaults to the defined work drive, which in turn defaults to each on-line drive if ALL is defined as the work drive.

Omission of <matchlist> defaults to all the directory entries. In this latter case any "holes" in the directory are printed as blanked out fields. This gives a "feel" for where subsequently created files will reside in the directory.

In order to compress the size of large directory listings, two files are printed per line. The actual information provided on a directory listing are: header information detailing the drive #, disc name, version number and creation date; format information regarding number of sides, density and number of tracks; information on the files that match the <matchlist>; summary information detailing the number of files found on the disc, the size of the largest, the number of sectors used and finally the number of sectors remaining.

The actual information printed for each file is a function of if an option is selected, and if so which one. Normal default gives: relative number in the directory, name, extension, creation date, total sectors used. The "\" option gives: relative number in the directory, name, extension, first sector used, last sector used, total sectors used. The "/" option gives: relative number in the directory, name, extension, creation date, random file flag, write/delete/catalogue protection flags.

Illegal creation dates are printed as such. Unrecognised disc formats, such as those produced by Winchesters, yield "?S/?D" as the sides/density information.

## DIR

(continued)

Some example invocations follow:

```
+++DIR
```

This will produce a directory listing for the full directory of the work drive with sector counts. If the work drive is ALL the directories of all on-line drives are printed out.

```
+++DIR\
```

This will produce a directory listing for the full directory of the work drive with sector counts and first/last sector information in lieu of the creation date. If the work drive is ALL the directories of all on-line drives are printed out.

```
+++DIR/
```

This will produce a directory listing for the full directory of the work drive with random and protection flags in lieu of the total sectors used information. If the work drive is ALL the directories of all on-line drives are printed out.

```
+++DIR,2
```

This will produce a directory listing for the full directory of drive #2.

```
+++DIR\,1
```

This will produce a directory listing for the full directory of drive #1 with sector counts and first/last sector information in lieu of the creation date.

```
+++DIR/,3
```

This will produce a directory listing for the full directory of drive #3 with random and protection flags in lieu of the total sectors used information.

```
+++DIR,2,L,.S
```

This will produce a directory listing for the all files starting with L and then all files with an extension starting with .S on drive #2.

```
+++DIR\,1,L,.S
```

This will produce a directory listing for the all files starting with L and then all files with an extension starting with .S on drive #1 with sector counts and first/last sector information in lieu of the creation date.

## DIR

(continued)

```
+++DIR/,3,L,.S
```

This will produce a directory listing for the all files starting with L and then all files with an extension starting with .S on drive #3 with random and protection flags in lieu of the total sectors used information.

```
+++DIR,1,2,L,.S
```

This will produce a directory listing for the all files starting with L and then all files with an extension starting with .S on drives #1 and #2.

```
+++DIR\,1,O,P,.AS
```

This will produce a directory listing for the all files starting with P and then all files with an extension starting with .AS on drives #1 and #O with sector counts and first/last sector information in lieu of the creation date.

```
+++DIR/,3,2,INV,.BIN
```

This will produce a directory listing for the all files starting with INV and then all files with an extension .BIN on drives #3 and #2 with random and protection flags in lieu of the total sectors used information.

```
+++DIR,1,2,LI.T
```

This will produce a directory listing for the all files starting with LI and having an extension starting with .T on drives #1 and #2.

```
+++DIR\,1,O,JUNK.AS
```

This will produce a directory listing for the all files starting with JUNK and having an extension starting with .AS on drives #1 and #O with sector counts and first/last sector information in lieu of the creation date.

```
+++DIR/,3,2,INV.BIN
```

This will produce a directory listing for the all files starting with INV and an extension .BIN on drives #3 and #2 with random and protection flags in lieu of the total sectors used information.

The complexity of the <matchlist> is only limited by the maximum size of the command line, which is 127 characters plus the <RETURN> and this value must include the DIR and separator characters. Also any number of drives may be referenced, though it only makes sense to do each drive once!

# DUMP

The DUMP command is used to display the contents of memory in HEX and ASCII format.

## DESCRIPTION

The syntax for DUMP is:

```
DUMP,<address>
```

<address> is: any memory address in the form <nnnn> with 'n' representing a hexadecimal digit.

Once the command is entered the contents of memory will be dumped in 256 byte block segments. As soon a a block is dumped the display will stop. Hitting any key other than <CR> will result in the next 256 byte block being dumped. Hitting <CR> will terminate the dump and return control to FLEX.

### NOTE

It is a good idea to turn off the TTYSET PAUSE (TTYSET,PS=N) when using this utility otherwise annoying stoppages may occur when the screen fills up.

You can use the PR, and OUT utilities to re-direct the output to a printer or a disc file respectively.

This utility is optimised for an 80 column VDU.

# F

The F utility is provided to (F)ormat a normal system based text file to be prepared for submitting to the spooler. An optional title may be incorporated, in which case the title and date occur on the top, and the page number at the bottom, of each sheet. F fully honours the parameters set up using the SETFORM utility. If the text file has very long lines, F will introduce a CR/LF sequence so that the printer width is not exceeded.

## DESCRIPTION

The syntax for F is:

F,<sfile>[,<dfile>[,<text>]]

<sfile> is a source file which defaults to a .TXT extension on the work drive. If work is defined as ALL, F will search from drive #0 onwards until the file is found,

<dfile> is the destination file which will become a spooler prepared version of <sfile>. It defaults to the same name as <sfile>, with a .OUT extension on the system drive. If system is defined as ALL drive #0 is used,

<text> is the title required on each page and can be a maximum of the width defined in SETFORM less 18. These 18 characters cover the date and one space minimum separation between the title and date. The title is left and the date right justified. This puts a minimum of 19 characters on the width to be specified using SETFORM.

The page number is centered on the page. Both the title and page number lines have one guard line associated, and so the printable depth will be 4 less than specified in SETFORM when <text> is included. This puts a minimum of 5 lines on the printable depth defined in SETFORM.

**********
* IMPORTANT *
**********

If <dfile> is to be completely defaulted AND a title is required, <dfile> MUST BE NULLED OUT WITH A COMMA.

F

(continued)

It is very important to realise that <dfile> has been given a very powerful defaulting mechanism. The following are all valid:

```
<#>
<#>.<body>
<#>.<body>.<ext>   (conventional)

<body>
<body>.<#>
<body>.<ext>
<body>.<ext>.<#>   (conventional)

.<ext>
.<ext>.<#>
```

these are in addition to a complete default. The components are defined as:

```
<#>    is a valid drive #,
<body> is a valid body definition (up to 8 characters),
<ext>  is a valid extension (up to 3 characters).
```

After being invoked, F displays the source and destination files that it will be trying to use. This is to assist the user in knowing just where the file went (the minimum requirement defaults for the destination file are VERY POWERFUL!). In the case of <sfile>, a "?" will be shown in place of the drive number if the work default is ALL. This is because the source file could be residing on any drive, or none at all.

Some examples will help to show how the defaults work:

```
+++F,BACKUP
```

This will Format the file BACKUP.TXT on the work drive into a spoolable version called BACKUP.OUT on the system drive. Skips at the top and bottom, and the printed depth per page will all honour the current SETFORM parameters.

```
+++F,HISTORY,,AN EXAMPLE OF A TITLE !!
```

This will Format HISTORY.TXT on the work drive into a spoolable version called HISTORY.OUT on the system drive. The 24 characters 'AN EXAMPLE OF A TITLE !!' will be printed left justified after the top skip with a right justified date followed by one spacing blank line. On the penultimate line there will be a blank spacing line, followed on the ultimate line by a centered "Page <n>" where <n> is the current page number. As the title and page number eat up 4 extra lines, the effective printed depth is 4 lines less than would be printed if the title option was not included.

YOU DO NOT NEED TO ALTER SETFORM PARAMETERS IN ANY WAY AS 'F' WILL AUTOMATICALLY ADJUST ITSELF TO SUIT THE ADDITIONAL TITLE.

F

(continued)

```
+++F,INFO.BAK.0,.SCR
```

This will format INFO.BAK on drive #0 into a spoolable version called INFO.SCR on the system drive.

```
+++F,DEAD.INF.3,2,RECOVERY OF A DEAD SYSTEM
```

This Formats DEAD.INF from drive #3 into DEAD.OUT on drive #2 with titles and page numbers.

```
+++F,PRICES.2,,Summer 1984 price list
```

Here a price list text file called PRICES.TXT on drive #2 is Formatted into PRICES.OUT on the system drive with a suitable title on each page. Note the use of double commas to total <dfile> defaulting.

As there are 5 ways of describing the source file, 10 ways of specifying the destination and an optional title, there are a total of 100 combinations. There are far too many to show examples for ALL possible invocations. The rule is IF IT MAKES SENSE TO YOU then 'F' should understand. This is one reason why the file names are displayed for you.

# FASTBACK

The FASTBACK command is used to make rapid a backup copy of a disc. The backed up version will be identical in every way to the original. As it does copying by track and not file there is no check of the structure of the information. For this reason normal COPYing should not be deferred in place of FASTBACKing. The verification phase of FASTBACK may optionally be suppressed by inclusion a minus sign.

## DESCRIPTION

The general syntax of the FASTBACK command is invoked:

```
FASTBACK,<from>,<to>[,-]
```

| | |
|---|---|
| <from> | is the source drive. |
| <to> | is the destination drive. |
| '-' | is the optional 'no verify' which is about 30% faster than a verified backup. The no-verify option is NOT preferred because there is no checking that the destination copy is intact. |

In the event that there is an incompatibility between discs, FASTBACK will report the format of the source disc in order to allow the destination disc to be similarly formatted.

We cannot over emphasize that this facility does not check file structure on the disc it is copying. If the source disc has a corrupted file on it the corrupted file will be transferred to the destination disc. Only the normal 'COPY' command will check for proper file structure and report errors, this is why COPY is so much slower.

FASTBACK is provided for those of you out there who do not value their time or the data on their discs enough to enforce rigid archive and backup rules. These rules require rigid self discipline as the backup copies are not reqired 99% of the time. The only time that the self discipline pays off is when you have the enevitable accident, usually caused by operator error or a hardware fault.

## FOOD FOR THOUGHT

A typical programmers work disc generally has in excess of two man-months of programming work on it. Most of this work has been tested, evaluated and debugged. Imagine that you are just logging out of the editor and someone knocks the plug out of the wall accidentally or the power fails. The universal laws of chance are overwhelmed by 'MURPHYS LAW' which dictates that any failure will occur in the area that will cause the most damage. Thus the disc system will write random garbage all over the disc directory as the processor goes into its death rattle!

Sure, some of the FLEX 'DIAGNOSTICS' utilities can help recover some of the files on the disc. Many files will not be recoverable however and the amount of aggravation associated with recovering a disc with a crashed directory is too horrific to describe! Is the chance of this occuring worth the price of a disc and 30 minutes of time to DAILY format a fresh disc and COPY all files from the current disc onto it?

## DISCS ARE CHEAP, YOUR TIME IS NOT!

# FDATE

FDATE is a utility which will either display or modify a file's creation date.

## DESCRIPTION

The syntax for FDATE is:

```
FDATE?[,]<filename>

FDATE,<filename>

FDATE,<filename>,<date>
```

<filename> is the filename and defaults to a .TXT extension on the work drive,

<date> is the required new date in the form: DD,MM,YY.

The first type of call is used to find out the date of the individual file, used in preference to either CAT of DIR which give a lot of unrequired information in addition. The second type of call will force the file to appear to have been created today, i.e. it uses the current date registers for modifying the file creation date. Finally the last type of call will do the same as the second, except that the date will be that which is provided as opposed to the current system date. An example of each type follows:

```
+++FDATE?PROJECT ... or ...  +++FDATE?,PROJECT (whichever suits you!)
```

This will display the creation date of the file PROJECT.TXT on the current work drive.

```
+++FDATE,COPYOTO1.CMD
```

This will update the creation date of the file called COPYOTO1.CMD on the work drive so that it appears to have been produced today.

```
+++FDATE,2.JUNKET.SCR,1,9,63
```

This will update the creation date of the file called JUNKET.SCR on drive #2 so that it appears to have been created on 1 September 1963 !!!

The main use for FDATE is to put a realistic date for a file which has an illegal creation date. The WMS DIR and VER utilities, to name but two, will not show a date which is invalid. Another use is if YEAR had been used to force DATECOPY (another WMS utility) not to copy over files which were created today. If, subsequently, more files have been produced, they will reflect the new year (normally something like 99!) unless the YEAR utility had been used to reset the real year. In these cases a swift FDATE will restore a correct date - after resetting the year.

# FIXES

This file is not a utility, it contains three patches to the inner workings of FLEX itself.

## DESCRIPTION

The syntax is:

```
+++GET,FIXES.SYS
```

This will load the overlay into memory over the appropriate portions of FLEX. We suggest that you include the above command in your 'STARTUP' sequence. The main areas of improvement are:

1) Typing ^R (CONTROL-R) for 'REPEAT' will print the last command line issued to FLEX and leave the cursor positioned at the end of the line. You can now either type <RETURN> to re-execute the line or back-space and make any desired corrections to the line and then type <RETURN> to execute it. This latter feature is useful when a command syntax error is posted by FLEX as, more often than not, the error will be in the last part of the command line (due to MURPHY's law).

2) Accessing an illegal drive number, i.e. drive #4 on up, will not result in the usual strange behaviour of accessing the drive number specified - 4! i.e. if you type DIR,4 you will get a directory listing of drive #0 if this patch is not in place!

3) Hexadecimal number letters 'A' through 'F' may be typed in either upper or lower case. This patch also applies to the low-level hex input routines described in the FLEX PROGRAMMERS MANUAL.

# FREEMAP

The FREEMAP utility allows the linked list of free sectors on a disc to be reformed so that subsequently created files have the best chance of being formed contiguously and with minimum head movement.

## DESCRIPTION

The syntax for FREEMAP is:

```
FREEMAP,<drive>
```

<drive> is a valid on-line drive.

An example of invoking FREEMAP follows:

```
+++FREEMAP,2
```

This would reform the free chain of drive # 2.

FREEMAP operates by doing two passes through the disc. On the first pass the complete free chain is read, and a sector map is formed for those groups of sectors which are logically adjacent. The second, and final, pass alters the free chain pointers in order to make the free chain linear (in a logical sense) starting from the outermost track (track # 0) to the innermost track.

It is important to realise that following a FREEMAP operation on a disc, the structure of deleted files may be corrupted (this will only occur if the file was fragmented when originally created). This information is imparted as it may not be obvious that a utility, such as the TSC 'UNDELETE', recovers deleted files from the free chain. For this reason if a fragmented file is deleted and then the disc has its free chain restructured by FREEMAP it will not be possible for UNDELETE to recover the file.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                           *
* THIS UTILITY OPERATES IN MAIN MEMORY AND NOT IN THE TCA   *
*                                                           *
*    DO NOT INVOKE FROM WITHIN BASIC, PL/9, MACE, ETC       *
*                                                           *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

# IN

The IN utility allows input to be fetched from a disc based text file instead of the keyboard as does the TSC 'I' command. Unlike the TSC 'I' command 'IN' will echo all 'input' from the disc to the system console.

## DESCRIPTION

The syntax for the IN command is:

```
IN,<tfile>,<command>
```

<tfile> is the filename of the file which contains the text to be used in the command,

<command> is the command that is required to be executed.

When a command is invoked with this preamble, it will take any input from the text file <tfile> just as though it had been typed from the keyboard. Unless specified commands, namely certain Windrush ones, say anything to the contrary, the format of the <tfile> is exactly as would be typed normally. <tfile> defaults to a .TXT extension on the WORK drive.

The reason that the Windrush commands may be different is in those cases where a utility has a command line invocation, e.g. FORMAT or SETUP. In these cases the format of the <tfile> can be very confusing as some inputs require a <RETURN> after typing and some get a keystroke immediately. For this reason FORMAT, SETUP and possibly future utilities will take a <tfile> input string which is identical to it's command line call counterpart.

Given the following text file:

1.UP_DATE.TXT is a file which contains just 3 characters: a <RETURN>, a <RETURN> and finally a "Y", being created thus:

```
+++BUILD,1.UP_DATE
#<RETURN>  <-------- hit 'carriage return' on your keyboard
#<RETURN>
#Y
##
```

an example of a case for the use of the IN command follows:

```
+++IN,1.UP_DATE,NAME
```

The NAME command allows changing the volume name, version number and date of a disc. Entering <RETURN> to a prompt will leave the original unchanged. This means that the example will: invoke the NAME command, leave the name intact, leave the volume number intact, change the date to today's date.

It is the ability to re-direct the I/O so that input comes from a file that can give a great deal of flexibility to operating some of the more complex commands.

In summary - any command that needs input from the keyboard, could be invoked using "IN" so that the actual data comes from a prepared text file. By by having several different files, the way that the command operates becomes a function of the actual <tfile> specified.

# LOAD

The LOAD utility allows a program or binary file to be loaded into memory at an absolute address, and informs what the new start address is if a program is loaded.

## DESCRIPTION

The syntax of the LOAD utility is:

LOAD,<address>,<fname>

<address> is the base address where the <fname> will be loaded in memory,

<fname> is the filename of the command or binary file to be loaded.

The defaults for <fname> are a .CMD file on the assigned system drive.

LOAD has two main uses: loading binary files into RAM memory so that an EPROM may be programmed, loading a command/utility which has been written in Position Independent Code (PIC) in readiness for it's execution (or debugging) at the new address. An example of each use follows:

```
+++LOAD,0000,1.NEW_MON.BIN
+++LOAD,1000,DEBUG
```

The first example will load what would appear to be a new version of a system monitor into memory starting at $0000. This could be a prerequisite of an EPROM programmer which does not itself contain a binary file offset loader. The fact that there is no transfer address will be indicated.

The second example will load the TSC DEBUG utility into memory at a base address of $1000 and the transfer address, in this case $1000, will be indicated. Execution at this address (via the JUMP command) will perform a cold start of the DEBUG package.

Two points should be borne in mind about LOAD. Firstly it DOES NOT ALTER ANY OF THE CODE THAT IS LOADED, it merely positions it in memory at a different address. For this reason, any command/utility which is expected to be relocated and execute at the new address ...

```
* * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                   *
*  MUST BE WRITTEN IN POSITION INDEPENDANT CODE!    *
*                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * *
```

## LOAD

(continued)

The second point is that LOAD has no way of knowing the structure of all the different sections of the file that it may have to load, it therefore places the first binary record loaded at the base address specified. All further records will load into memory with the same offset applied. If the file comprises a series of blocks of code at incrementing starting addresses, as will be the case for most programs, no special action need be taken when using 'LOAD'.

If, however, you are one of those people who write programs with blocks of code at higher addresses in front of blocks of code at lower addresses you must do a bit of extra work before using 'LOAD'. The following example will clarify this:

```
+++MAP ODD_BALL.BIN.1

C600 - C6E7
C100 - C107
C118 - C4C9

TRANSFER ADDRESS is at $C3DF
```

Given the above program map, consider the following invocation of LOAD:

```
+++LOAD,0,ODD_BALL.BIN.1
```

The addresses actually loaded into will be:

```
C600 - C6E7   -->  0000 - 00E7
C100 - C107   -->  FB00 - FB07
C118 - C4C9   -->  FB18 - FEC9

TRANSFER ADDRESS is at $FDDF
```

This is obviously a mistake as the code will have attempted to overlay the monitor. The only reason for this quirk is that the code is mapped in a sort of backward way. To overcome this it should be noted that the lowest address from MAP was $C100 and further that the first address encountered was $C600. The following formula will then hold true:

LOAD ADDRESS = REQUIRED CODE BASE + FIRST ADDRESS - LOWEST ADDRESS

In the example then:

LOAD ADDRESS = 0 + $C600 - $C100  i.e. $0500

So using the following invocation:

```
+++LOAD,500,ODD_BALL.BIN.1
```

The addresses actually loaded into will be:

```
C600 - C6E7   -->  0500 - 05E7
C100 - C107   -->  0000 - 0007
C118 - C4C9   -->  0018 - 03C9

TRANSFER ADDRESS is at $02DF
```

# MAKECMD

The MAKECMD utility allows a set of FLEX command calls to be strung together to make one command. This may not sound like much but once you start using it you will wonder how you ever got along without it!

## DESCRIPTION

The syntax for MAKECMD is:

```
MAKECMD,<filename>[<comnd>]
```

<filename> is the name of the command being produced/examined which defaults to a .CMD extension on the system drive,

<comnd> is ,<command>[<params>][<sep><comnd>]

<command> is a system command,

<params> is the optional parameter list for the <command>,

<sep> is the current TTYSET 'EL' separator, usually a colon ':'.

If the above syntax frightens you, don't worry - it is because MAKECMD is very powerful. The recursive definition of <comnd> means that MANY different commands may appear one after the other when separated by the present TTYEOL character (normally a ":" on default). By leaving off the <comnd> altogether, the existing <filename> will be displayed on the terminal.

MAKECMD works by doing a direct text substitution of the command list when invoked by the actual <filename>. This is mentioned in case a generated command list includes another MAKECMD generated command !!!. If a generated command includes a generated command invokation the rest of the original command list will be replaced by the new list. On the other hand, generated commands may be made to "chain" onto the end of each other which can be highly useful.

The following examples should unravel the foregoing:

```
+++MAKECMD,BACKUP
```

This is the "examine" mode of using MAKECMD. The text that will be substituted for "BACKUP" is displayed on the terminal.

```
+++MAKECMD,DIR_TXT,DIR,0,1,.TXT
```

This will produce a command called "DIR_TXT.CMD" on the system drive. Typing DIR_TXT would then cause DIR,0,1,.TXT to "appear" to have been typed, the effect being to print the file names on drive #0 and then #1 with a .TXT extension.

## MAKECMD

(continued)

```
+++MAKECMD,ZAP_TXT,DIR,0,1,.TXT:ZAP,0,1,.TXT
```

This will produce a command called "ZAP_TXT" on the system drive. Typing ZAP_TXT would then cause:

```
DIR,0,1,.TXT:ZAP,0,1,.TXT
```

to "appear" to have been typed, the effect now being to print the file names on drive #0 and then and #1 with a .TXT extension, then to ZAP those files. This allows the user to see if anything important is about to disappear, and in so doing allow abort of the ZAP by the ^C (control-C) breakout before the file is erased.

The examples have gradually increased in complexity from a simple "examination" of an existing file, through building a single command and then on to building multiple commands. The final example shows how to produce a set of commands that all chain each other - WHY BOTHER? The EXEC command is sometimes inappropriate for certain tasks, eats into MEMEND and takes longer to load than a single sector!

Let us assume that two "macros" (for want of a better word) are to be required for making system backups: one to a single density disc, one to a double density disc. Further, let us require all .SYS files to be copied first, followed by all .CMD files and finally the rest of the files on the source disc. The backup shall be made from drive #0 to drive #1. For this task three files should be made as follows:

```
+++MAKECMD,COPY_SYS,COPY,0,1,.SYS,.CMD:N,COPY,0,1
+++MAKECMD,MAKSDSYS,FORMAT,+1,D,S,80,Y,9,ARCHIVE.SYS,1:COPY_SYS
+++MAKECMD,MAKDDSYS,FORMAT,+1,D,D,80,Y,9,ARCHIVE.SYS,1:COPY_SYS
```

This has produced three macros: COPY_SYS, MAKSDSYS and MAKDDSYS.

Typing COPY_SYS at any time will do the actual copying as required.

Typing MAKSDSYS will firstly format drive #1 as single density (q.v. the Windrush FORMAT utility documentation) and then invoke the COPY_SYS by the chaining, thereby copying as required.

Typing MAKDDSYS will firstly format drive #1 as double density and then invoke the COPY_SYS by the chaining, thereby copying as required.

It is far more preferable to use the new macros than type out the complete command lines each time a backup is needed - if only because the probability of mis-typing is greatly reduced!

NOTE: MAKECMD always produces a single sector - never more. As the substituted text is placed directly into FLEX's input buffer a command line can be up to 127 characters long plus the closing <RETURN>. If the length is in doubt use MAKECMD to examine the produced command and this can be assertained.

# MAP

The MAP utility allows a file to be examined as a binary image, so producing a map of where it would fit in memory. It is consistent with the GET memory resident command, that is a text file MAY APPEAR to have binary records in it and a GET would indeed load those as though they were binary records! Not many people realize this fact of life with FLEX!

## DESCRIPTION

The syntax for invoking MAP is:

```
MAP,<fname>
```

<fname> is a filename which defaults to a .BIN type on the assigned WORK drive.

It only makes sense to MAP a file containing binary records, though using it on a text file will show where GET would load it!

MAP will print out each contiguous block of the <fname> and if no records are found will report the fact. In addition the TRANSFER address, or lack of it, will be printed. Two examples follow:

```
+++MAP,MONITOR.BIN
F800 - F816
F820 - FEEC
FFE0 - FFED
FFF2 - FFFF

File has no TRANSFER ADDRESS
```

```
+++MAP,0.EXEC.CMD
C100 - C1E8

TRANSFER ADDRESS is at $C100
```

The first example is that of a a system monitor object file. As such it is just straight binary and does not include a transfer address.

The second example is of the EXEC command. This comprises a single contiguous block of memory $E9 bytes long starting at $C100 with an execution address at $C100.

# MEMEND

The MEMEND utility is used to examine/change the value of the FLEX 'MEMEND' two byte variable at $CC2B/C.

## DESCRIPTION

There are three ways of invoking MEMEND, the syntax is:

```
MEMEND
MEMEND?
MEMEND,[$]<value>
```

<value> is the required highest memory location that any system utility will use, optionally preceeded by the $ sign. The value is always a hexadecimal one.

The first case is used to find the highest RAM address that can be used. A non-destructive memory test is carried out and the new value will be placed in MEMEND (at $CC2B) if it is lower or the same as the present value. In case some memory resident command, like PROMPT, is sitting up in high memory a "Change" prompt is issued if the old value is lower than the new - ANSWER "Y" at your own peril. This is a safety catch and only if you are sure that nothing system dependant is above the old value should you type "Y". An example of a system dependant feature is PROMPT which catches all output and by altering MEMEND some other utility may overwrite PROMPT thereby hanging the system. An example of this invocation follows:

```
+++MEMEND .... the following is an example of the output:

New MEMEND is higher than old:

NEW is $BFFF
OLD is $BC41

Change to new (Y/N) ? N
MEMEND is set at $BC41
```

Here the MEMEND value was left (wisely) untouched.

The second call just prints out the current value of MEMEND. An example follows:

```
+++MEMEND? .... would result in output similar to:

MEMEND is set at $BFOF
```

The third call is used to set MEMEND manually and again checking is used on the old and new values. The output looks very similar to the second example and so is not reproduced, however it is called thus:

```
+++MEMEND,1234
```

Here the MEMEND will be set to the value $1234. If MEMEND is already lower the prompt will be issued for updating it.

# N

The N utility is used as a source of input which always returns a "N" character as though it had been typed from the keyboard in response to a request for input.

## DESCRIPTION

The command line syntax for the N command is:

```
N,<command>
```

<command> is the actual command that is to execute followed, if necessary, by it's parameters.

Whenever <command> expects to get input, it will "see" an "N" typed at it. This may seem a trivial task, but the following example (where it is often used) demonstrates it's usefulness:

```
+++N,COPY,0,1
```

This will copy all files from drive # 0 to drive # 1. If a file exists on drive # 1, the "delete" prompt will automatically be answered by an "N", therefore the effect of this command line would be to copy all files across which do not exist.

# NAME

The NAME utility allows the name, and/or version number and/or date to be modified on a disc. Any or all the components may be altered.

## DESCRIPTION

The syntax for NAME is:

```
NAME[,<drive>]
```

<drive> is the optional drive number which defaults to the work drive.

The command will print the current banner information in a similar manner to DIR and then prompts for the name to be changed. Entering <RETURN> will leave the data untouched, ^C (control-C) will cause an immediate return to FLEX and anything else will be taken as the replacement text. After name comes the version number, again the same holds about entering or skipping data. Finally the "Change DATE (Y/N) ?" prompt appears. If a "Y" is entered the current system date will be put to the disc, otherwise "N" or ^C should be typed. ^C would not touch the disc, while "N" would apply the updates to the disc.

An example follows:

This calls NAME using the work drive default. The following banner is an example of what is displayed:

```
DRIVE: 1        DISC: FLEX    .         VERSION:     0      CREATED: 2-JAN-84

New NAME          ?   <RETURN>
New VERSION       ?   <RETURN>
Change DATE (Y/N) ?   Y
```

In this case only the date was changed, the <RETURN> meaning leave the original data intact.

The main use of NAME is to actually change the name of a disc for archival purposes, and/or update the version number.

# OUT

The OUT utility is for output re-direction of any command to a file for subsequent printing by the spooler. It's effect is identical to the PR utility except that instead of accessing the printer directly, the output goes into a text file. There is therefore no difference between the paper outputs from following two examples:

```
+++PR,DIR
+++OUT,DIR,DIR:PRINTT,DIR
```

except that the second example, using OUT, could be printed n times at users leisure AND while the system can be used for other things. Note that the file DIR.OUT will remain available until it is deleted either absolutely or by the XXOUT utility.

## DESCRIPTION

The syntax of the OUT utility is:

```
OUT,<filename>,<command>
```

<filename> is the file that is to be produced which defaults to a .OUT extension on the system drive,

<command> is the actual command that is being invoked to produce an output.

As with the PR utility, the output file will include top and bottom skips to by-pass the fold on fan-fold paper and will honour the current values set up by the SETFORM utility. If the <command> sends out continuous text with no CR code then OUT will do an automatic CR/LF sequence when the width exceeds the defined width of the printer. Some examples of O follow:

```
+++OUT,DIR,DIR
```

This will produce a file called DIR.OUT on the system drive which will contain the output from the DIR command.

```
+++OUT,LISTING,DATECOPY,1,0,-,.CMD
```

This is a very useful example of OUT. Here the file LISTING.OUT on the system drive will give a very thorough history of what happened when drive #1 had all it's command files archived onto drive #0.

```
+++OUT,LIST_ZAP.1,ZAP,.TXT
```

This example would produce a file called LIST_ZAP.OUT on drive #1 that was a history of the ZAP command.

All files produced using OUT are directly spoolable. Take care though if you are dynamically playing around with SETFORM as the spooler uses the data table from SETFORM as it is running. Normally this will only need to be set up for one given printer at a time, or if 11" paper is used instead of, say, 12".

# PCOPY

The PCOPY utility allows files to be copied from disc to disc or, if a copy to another name is used, the same disc. It differs from the COPY command in that a prompt is always posted before a copy - hence the name P(rompting)COPY.

## DESCRIPTION

There are three ways of invoking PCOPY and the syntax is:

```
PCOPY,<drive>,<drive>[,<matchlist>]
PCOPY,<filespec>,<drive>
PCOPY,<filespec>,<mfilespec>
```

<drive> is a valid on-line drive,

<matchlist> is either (or both) part (or all) of a filename or extension,

<filespec> is a filename with an extension,

<mfilespec> is a minimum of a filename - with an optional extension.

The first case is the one mostly used. It allows copying of either an entire disc to another, or just matched files from one drive to another. Matching is on either part (or all) of the name (or extension) or both. The <drive>s must be valid on-line drives or the command will abort. Some examples of this type of copy follow:

```
+++PCOPY,0,1
+++PCOPY,1,0,.SYS,C_D,P.C
```

The first example will copy ALL files where the "Copy" prompt is confirmed from drive # 0 to drive # 1. The second will copy those confirmed files from drive # 1 to drive # 0 which have a .SYS extension, then those confirmed which start off with the characters C_D, then those confirmed that start with P and have a extension which starts with a C.

NOTE: If a file exists on the destination drive, a prompt is issued regarding deletion of the file found. If Y is answered an "are you sure" prompt returns and if Y is answered to this one the file will be deleted and the source version copied across. Typing N to either prompt will not delete or copy the file and the directory search is continued. This prompting method holds true for all invocations of PCOPY.

The second type of call is useful for copying just one file from one work disc to another with only two drives on a system. The prompt allows time to change, for example, destination and system discs before confirming:

```
+++PCOPY,0.P.CMD,1
```

This copies JUST the file P.CMD from drive # 0 to drive # 1 if the prompt to copy is confirmed.

## PCOPY

(continued)

The last type of call is the one least used. It copies JUST <filename> from the source disc, to the destination disc (may be the same disc) using <mfilename> as the new name. As with the second type of call, this is most useful when copying between different work discs with only two drives on a system. The extension will remain the same if <mfilename> does not include an extension, otherwise the extension will be as requested in <mfilename>. Some examples follow:

```
 +++PCOPY,1.WORK.TXT,NEW_VER
 +++PCOPY,0.P.CMD,0.P_FILE.BIN
```

The first example copies the file WORK.TXT from drive # 1 to a file to be called NEW_VER.TXT on the assigned WORK drive if the prompt is confirmed. The second example shows how the extension may also be altered. Here the file P.CMD on drive # 0 is copied to a file called P_FILE.BIN also on drive # 0, again only if the prompt is confirmed.

Both <filename> and <mfilename> may exclude the drive number in any calls and the drive will always default to the WORK drive, or drive # 0 if ALL is specified as the WORK drive.

During copying the success, or otherwise, of the copy is displayed on the terminal (unless re-vectored using the I command).

NOTE: At any stage, ^C (control-C) may be sent from the terminal. This will cause PCOPY to terminate in an orderly fashion and return to FLEX.

# PDEL

The PDEL utility is a prompting deletion utility that would normally be used in preference to the ZAP utility. As the name implies, the deletion action is prompted giving the option to leave the file intact.

## DESCRIPTION

The syntax for the PDEL utility is:

```
PDEL[,<drives>][,<matchlist>]
```

<drives> is: <drive #>[,<drives>] where <drive #> is a valid on-line drive,

<matchlist> is: <match>[,<matchlist>] where <match> is either (or both) part(s) (or all) of a filename or extension.

Omission of <drives> defaults to the defined work drive which in turn defaults to each drive on the system if ALL is defined. Omission of a <matchlist> defaults to all directory entries.

Having typed a command line, each file that matches the <matchlist> is printed as a "Copy" option. No further prompting is made soBE SURE WHEN YOU TYPE A 'Y' otherwise the file will be deleted. Any key other than ^C (control-C) will be taken as a no and the next file will be offered for deletion.

NOTE: If ^C is typed at any time, a clean return to FLEX will be affected.

Some examples of PDEL follow:

```
+++PDEL
```

This will cause all files on the work drive to be offered for deletion. If the work drive is defined as ALL then each on-line drive will be selected in turn.

```
+++PDEL,1
```

This will offer all files on drive #1 to be offered for deletion.

```
+++PDEL,2,P
```

This will offer all files that are found on drive #2 and start with a P to be offered for deletion.

```
+++PDEL,1,.SYS
```

This will offer all files that have a .SYS extension on drive #1 to be offered for deletion.

```
+++PDEL,3,2,TE.BAS,.BAC
```

This will offer for deletion those files that start with TE and have a .BAS extension, followed by those files that have a .BAC extension on drive #3 and then drive #2

# PR

The PR utility allows re-direction of output to the printer rather than the terminal. PR fully honours the TTYSET WD, DP and EJ parameters. PR also disables the normal TTYSET PS automatic halt feature during the print operation and restores it when the print operation is completed. However output to the printer may still be manually halted by hitting ESC and restarted by ESC, or aborted by RETURN.

## DESCRIPTION

The syntax for the PR command is:

```
PR,<command>[,<args>]
```

<command> is any system command,

<args> are any arguments for <command>.

This command is most useful when used with either the CAT or DIR utilities as it will produce a hard copy of the files. If PR is used in a multiple call command line it will only be in effect for the command that follows PR, and not any of the following ones. Some examples of the use of PR follow:

```
+++PR,DIR,.ASM
```

This example will re-direct the output of the DIR command to the printer using a match list of .ASM.

```
+++PR,LIST,PROJECT.BAS:DIR,.BAS
```

This example will list the BASIC source file PROJECT.BAS to the printer, and then all .BAS files will be displayed on the terminal as a result of the DIR command.

NOTE: Some utilities do some output to the terminal direct, and other output via the FLEX PUTCHR routine. PR will honour all PUTCHR output but can not print any output that by-passes the PUTCHR routine.

# PRT

The PRT utility allows re-direction of output to the printer rather than the terminal. This new version formats the output to allow a skip at the top and bottom of a page in order to miss the folds in fan-fold paper. Though not an actual spooler utility, it uses the data table within the spooler area (set by SETFORM) and so is included here for completeness.

## DESCRIPTION

The syntax for the PRT command is:

```
PRT,<command>[,<args>]
```

<command> is any system command,

<args> are any arguments for <command>.

This command is most useful when used with either the CAT or DIR utilities as it will produce a hard copy of the files. If PRT is used in a multiple call command line it will only be in effect for the command that follows the PRT and not any of the following ones.

The actual number of lines printed on a page, along with the number skipped at the top and bottom, are configured by using the SETFORM utility. PRT honours ALL the parameters set up using SETFORM. Automatic CR/LF sequences will occur if a line is too wide for the width of the printer, also hitting ESC during printing will halt the printout, hitting ESC again will resume, while RETURN will abort the print and return to FLEX. Some examples of the use for PRT follow:

```
+++PRT,DIR,.ASM
```

This example will re-direct the output of the DIR command to the printer using a match list of .ASM.

```
+++PRT,LIST,PROJECT.BAS:DIR,.BAS
```

This example will list the BASIC source file PROJECT.BAS to the printer, and then all .BAS files will be displayed on the terminal as a result of the DIR command.

NOTE: Some utilities do some output to the terminal direct, and other output via the FLEX PUTCHR routine. PRT will honour all PUTCHR output but can not print any output that by-passes the PUTCHR routine.

# PRINTT

The PRINTT utility is used to submit files to the spooler in order to allow a previously formatted text file to be printed while the system can still be used to perform normal functions.

## DESCRIPTION

The syntax for PRINTT is:

```
PRINTT,<filename>[,+,<no>]
```

<filename> is the file to be printed which defaults to a .OUT extension on the system drive,

<no> is the optional numer of additional copies required.

When PRINTT is invoked the file and repeats, if specified, are submitted to the spooler for printing. If the spooler is inactive, print will initialise it and report the fact to the terminal. Irrespective of whether the spooler was running or not, confirmation that the file was submitted is given along with the number of repeats.

Some examples of PRINTT follow:

```
+++PRINTT,DIR
```

This will submit the file DIR.OUT on the system drive to the spooler. Only the one copy will be printed.

```
+++PRINTT,BACKUP.1
```

This will submit BACKUP.OUT.1 to the spooler, again with no repeats.

```
+++PRINTT,ARCHIVE,+,1
```

This will submit the file ARCHIVE.OUT on the system drive to the spooler with an additional copy, i.e. a total of two copies will be provided.

```
+++PRINTT,DIARY.TXT.1,+,2
```

This will submit DIARY.TXT.1 to the spooler with two additional copies, i.e. a total of three copies will be produced.

## PRINTT

(continued)

NOTE 1: As previously mentioned, the spooler operates on prepared files which actually include LFs. This allows the printable output files produced by Word Processors, MACE and PL/9 to be submitted directly to the spooler.

NOTE 2: The new OUT (output re-direction) utility produces spooler compatible files directly, q.v. O and SETFORM commands.

NOTE 3: Only standard text files are in the wrong format and these should be post-processed by the F (for Format) utility to produce a file that is in the correct format. F includes the option to have a page title and number, q.v. F and SETFORM commands.

NOTE 3: The spooler can have up to 20 jobs in it's queue. If the queue is full when a PRINTT is requested, you will be informed to try later when the queue has reduced.

**********
* WARNING *
**********

The spooler determines which line is being sent to the printer in order that it can skip the printer to the top of a clean sheet for the next job. This is done by counting the LFs sent to the printer. If you are spooling a file with printer control characters in, ensure that hex $A is never sent as a possible count #, or only print one job at a time.

If the later is your only course of action, check the paper position on the printer and adjust as necessary before submitting the next job. An alternative would be to use the spooler management utility QCHECKK to suspend the spooler after the file is completed, check the paper when supended, adjust if necessary, and then restart the spooler. If the next file is a repeat or another ODD-BALL then put the spooler back into suspension. This is the recommended way to treat Word Processor output files which ARE KNOWN to possibly contain LF characters which ARE NOT ACTUAL LFs. The systems engineer responsible for configuring the Word Processor to your installation will be able to tell you if your printer/Word Processor configuration will exhibit this facet.

THIS WARNING WILL ONLY AFFECT A VERY SMALL PERCENTAGE OF INSTALLATIONS AND IS MENTIONED HERE FOR COMPLETENESS IN ORDER TO EXPLAIN WHY A FEW PEOPLE MAY FIND WORD PROCESSED OUTPUT FILES TERMINATE WITH THE PAPER IN THE WRONG PLACE!

# PROMPT

PROMPT is a utility that provides a prompting message when FLEX is doing a pause in outputting to the terminal. Often a system can appear "DEAD" when it is actually waiting for ESC or RETURN. With PROMPT installed a great deal of confidence is provided that nothing will suddenly appear to have died.

## DESCRIPTION

PROMPT is invoked by:

```
+++PROMPT
```

It provides re-direction of output and as it has to be memory resident relocates itself right at the top of user memory.

If for some reason all the memory must be used for a large utility it is desirable to be able to remove PROMPT until it can be re-loaded. This is achieved by typing ^D (control-D) when the prompt is displayed to remove it from the system. PROMPT will then restore all vectors and MEMEND to their original values.

The recommended way to use PROMPT is to invoke it in the STARTUP file. Beware of removing PROMPT if something else has been subsequently placed below it. This is because PROMPT will restore MEMEND to the original value after BOOT and all protection of the code below PROMPT will be removed. In practice once PROMPT is installed, it is never removed - this is because it is very useful and reassuring to have the prompt displayed.

# QCHECKK

The QCHECKK utility is a management aid which enables the user to manipulate various aspects of the spooling function. This command is accompanied by a file called 'QCHECK.SYS' which serves as a 'HELP' file.

## DESCRIPTION

The syntax for QCHECKK is:

```
QCHECKK
```

The invokation is trivial because as the commands are called from within QCHECKK. To give a feel for the range of commands available, the following is what appears when A, for assistance, is typed as the command:

```
HELP
====

Q               Displays the files in the Spooler queue,
F               Freezes the Spooler, i.e. an IMMEDIATE halt,
S               Stop the Spooler AFTER the current file is completed,
G               Go and re-start the Spooler ( used after an F or S ),

R,#<posn>,<no> Repeat the file at <posn>, <no> times,
N,#<posn>       Make <posn> the next job to be printed,

T               Terminate the current output - repeats will continue,
D,#<posn>       Delete the file at <posn>,
K               Kill ALL Spooler output,

<RETURN>        Returns from QCHECK.
```

The commands are grouped into: high level spooler directives, file manipulation directives, deletion directives. A description of each group, with examples, follow.

NOTE: The file entitled 'QCHECK.SYS' must be present on the same disc as this command if this command is to be used. QCHECK.SYS is used by the 'A' (assistance) command.

## QCHECKK

(continued)

### HIGH LEVEL DIRECTIVES

These commands affect the way in which the spooler functions.

Q — Is used to display the information on files which are in the spooler queue. For each file, it's position, name, number of repeat copies and number of sectors are displayed. Following the information for each file, Q will then print the number of the current sector being output. This is in order to give a feel for how much of the present file has been output to the printer.

F — Is used to cause an immediate Freeze on the spooler, i.e. on return to the operating system the spooler will be held in a locked state until it is freed, q.v. G and S. Subsequent typing of Q will confirm that the spooler is indeed frozen.

S — Is used to Stop the spooler after the current file has been output. If any repeats were requested they will neither be lost nor printed. This command is useful if a file containing `false' LF codes is being spooled. The spooler will interpret all LF codes as a line feed and so will leave the paper incorrectly positioned for the next file. THIS WILL ONLY AFFECT A VERY SMALL PERCENTAGE OF INSTALLATIONS - MOST INTELLIGENT PRINTERS TRY AND AVOID USING $A AS A CONTROL CODE or BYTE COUNT. If F had been previously typed and now S is typed, the frozen state will be replaced by the stop state and so the printing of the present file will continue until it is completely printed. Typing Q will confirm that the spooler will stop, or is stopped.

G — Is used to restart the spooler if either the F or S command had been previously used. This command will always leave the spooler in the state where it is neither frozen nor stopped. On return to the operating system, the spooling function will resume. Typing Q will confirm that the spooler is neither frozen or stopped.

## QCHECKK

(continued)

### FILE MANIPULATION DIRECTIVES

These commands affect only the individual file referred to in the call.

R — Is used to alter the Repeat count for a given file. Two parameters are required: the position of the file in the queue (always preceded by a hash, "#", symbol), the number of repeats which must lie in the range 0..255. Some examples follow:

Command ? R,#2,0

This will reset the repeat count for the file at position #2 to zero, i.e. no more copies.

Command ? R,#3,20

This will cause the file at position #3 to have 20 repeat copies.

N — Is used to make a particular file job (by job we mean the original plus any associated copies) the next one to be printed. Only one parameter is provided which is the position of the file before the N command was issued. N will reschedule any repeats of the present job in order that the next file to be printed really is the one requested (but see the note below). Typing Q will confirm the new position of the queue. Some examples follow:

Command ? N,#11

This will make the file at position #11 (and repeats if applicable) the next job to be output to the printer. Unless the spooler queue is full, any repeats from the present job will be resheduled (but see note below).

Command ? N,#17
Command ? N,#14

The result of the above sequence of two commands is to change the priorities of two files in the queue. THIS IS NOT RECOMMENDED without a Q in between as the file positions will all change AND there may be an extra job appear as a result of rescheduling any repeats from the present job. The rule then is: modify the queue for a file, check the new queue, modify the queue for another file (if necessary).

#### NOTE

THAT IF THE QUEUE IS FULL, THE PRESENT JOB's REPEATS CAN'T BE RESCHEDULED - THIS IS THE ONLY CASE WHERE THE FILE SPECIFIED IN "N" WOULD NOT BE THE NEXT FILE TO BE PRINTED.

## QCHECKK

(continued)

### DELETION DIRECTIVES

These commands either operate on a specified file (D), the current file (T), or the spooler function itself (K).

T — Is used to Terminate the present file being printed. The spooler will clean up the position of the paper and then either print the repeats (if any were requested) or else move on to the next file in the queue. In order to completely ZAP the present output use the D command (below). An example follows:

```
Command ? T
```

This will terminate the present copy being printed. If any repeats were requested, these will now be printed - otherwise the next file will be printed.

D — Is used to remove the file at a given position from the queue. If the position is other than the current file the queue is merely modified to remove the file, otherwise the effect is as though the repeats of the present job were cleared followed by the T command being issued. Some examples follow:

```
Command ? D,#4
```

This removes the file at position #4 in the queue and re-shuffles the queue as necessary.

```
Command ? D,#1
```

This would be used in preference to the T command to completely remove the present file from the queue when repeats were requested. Using T alone would terminate the present copy and then start on a repeat.

K — Is the command that should be issued to Kill the whole of the spooler. It will position the print head to the top of the page, tidy up the queue, close down the spooler cleanly and then turn off the print spooler interrupt generation device. An example follows:

```
Command ? K
```

This Kills the spooler DEAD!

## QCHECKK

(continued)

### GENERAL

Two other commands may be used. One is the pseudo <RETURN> command which just returns control back to the operating system. The other is if A is typed (for Assistance). Typing A will list the options available (excluding the A command as you must have known what A did to invoke assistance!) and uses the file called QCHECK.SYS which should be on the system drive. In the event that a parameter is in error, or the syntax is incorrect, or an illegal command is attempted QCHECK will suggest that you type A to get the commands and syntax listed to the terminal.

In order to prevent any software hazards, the spooler is actually inhibited during the QCHECKK utility. This is mentioned in order to explain why, for instance, typing T 'appears' to have no actual effect. On issuing the <RETURN> to get back to the operating system, the desired action will take place.

# RE-NAME

The RE-NAME utility is used to alter the name of any file on the system. It is similar to the TSC utility but it is easier to use if you only wish to alter the extension of a file name.

## DESCRIPTION

The syntax for RENAME is as follows:

```
RE-NAME,<from>,<to>
```

<from> is the original file name which will default to being on the work drive, with a .TXT extension,

<to> is the new file name. Only the parameter(s) that need changing have to be provided though if you like typing, the whole name and extension may be specified.

RE-NAME can be used to change system file names to a shorter version, though this means that your system becomes non-standard: not a very desirable feature. The main use is to alter just the extension part, say from .TXT to .BAK so that the original remains around rather than having some editors do it for you. Some examples follow:

```
+++RE-NAME,WORK,.BAK
```

This will rename the file called WORK.TXT on the work drive to a name of WORK.BAK, though only the name is altered - the file is not touched.

```
+++RENAME,WORK.BAK,.TXT
```

This will rename the file called WORK.BAK to WORK.TXT.

```
+++RENAME,1.SYSTEM.DOC,INFO
```

This will rename the file called SYSTEM.DOC on drive # 1 to a file called INFO.DOC.

When RE-NAME is invoked, it will display the original and required file names. This is in order to ensure that you are fully aware of what RE-NAME is renaming to what! Success, or otherwise, of the rename is also displayed after the rename.

# S

The S utility is provided to (S)trip any LFs ($0A) from a text file. This is required if a spoolable file is to be read or merged into a word processor. This is because certain word processors treat control codes as printer embedment for example.

## DESCRIPTION

The syntax for S is:

```
S,<sfile>[,<dfile>]
```

<sfile> is a source file containing the LFs. It defaults to a .OUT extension on the system drive. If system is defined as ALL, S will search from drive #0 onwards until the file is found,

<dfile> is the destination file which will be a stripped version of <sfile>. It defaults to the same name as <sfile>, with a .TXT extension on the work drive. If work is defined as ALL drive #0 is used.

It is very important to realise that <dfile> has been given a very powerful defaulting mechanism. The following are all valid:

```
<#>
<#>.<body>
<#>.<body>.<ext>   (conventional)

<body>
<body>.<#>
<body>.<ext>
<body>.<ext>.<#>   (conventional)

.<ext>
.<ext>.<#>
```

these are in addition to a complete default. The components are defined as:

```
<#>    is a valid drive #,
<body> is a valid body definition (up to 8 characters),
<ext>  is a valid extension (up to 3 characters).
```

After being invoked, S displays the source and destination files that it will be trying to use. This is to assist the user in knowing just where the file went (the minimum requirement defaults for the destination file are VERY POWERFUL!). In the case of <sfile>, a "?" will be shown in place of the drive number if the system default is ALL. This is because the source file could be residing on any drive, or none at all.

S

(continued)

Some examples will help to show how the defaults work:

```
+++S,DIR
```

Here the source file resides on the system drive as DIR.OUT and the destination file will be on the work drive and called DIR.TXT.

```
+++S,CAT,0
```

The source is now CAT.OUT, again on the system drive, but this time the destination file will be called CAT.TXT on drive #0.

```
+++S,ARCHIVE.BAK,TRIAL
```

The source is now ARCHIVE.BAK on the system drive, while the destination is TRIAL.TXT on the work drive.

```
+++S,LIST_ZAP.2,.TMP
```

Here the source is 2.LIST_ZAP.OUT and the destination is LIST_ZAP.TMP on the work drive.

```
+++S,JUDO.BAS,.SRC.3
```

Now the source is JUDO.BAS on the sytem drive, and the destination file is 3.JUDO.SRC

As there are 5 ways of describing the source file, and 10 ways of specifying the destination, there are a total of 50 combinations. There are far too many to show examples for ALL possible invocations. The rule is IF IT MAKES SENSE TO YOU then S should understand. This is one reason why the file names are displayed for you.

In all the above examples, any occurrences of hex $A (i.e. LFs) were stripped in accordance with the requirement of producing a file which could be merged into, for example, a Word Processed document.

# SETCLOCK

This utility is used to set the various registers within the clock calendar chip and the FLEX system (SYSMON, SYSDAY and SYSYR) information.

## DESCRIPTION

The syntax of this command is:

```
   +++SETCLOCK
```

There are two basic versions of 'SETCLOCK' available:

SETCLK-N ... which is for the National MM58167 type clock-calendar.
SETCLK-H ... which is for the Hitachi HD146818 clock-calendar.

The former is coded in assembly language suitable for MACE and is supplied to match the address of the MM58167 chip on the Windrush 3U-TIM1 board.

The latter is coded in PL/9 and is supplied to match the address of the HD146818 chip on the Windrush 3U-TIM2 board.

You should rename the command file that suits your hardware 'SETCLOCK.CMD'.

The source and object files are supplied. The output format of the two commands is somewhat different owing to the ability of a high level language like PL/9 to control an input line and accept decimal data much easier than can be done in assembly language.

Invoking the SETCLOCK utility for the MM58167 will which will result in a question and answer (Q/A) session as follows:

```
* * *  WINDRUSH CLOCK-CALENDAR SETTING UTILITY  * * *

YEAR           (83-99):1985
MONTH          (01-12):01          see note 1
DAY OF MONTH   (01-31):08          see note 1
DAY OF WEEK    (01-07):01          see notes 1 and 2
TIME - HOURS   (00-23):12          see notes 1 and 3
TIME - MINUTES (00-59):00          see note 1
RE-DO AGAIN?     (Y-N):N
HIT ANY KEY TO START CLOCK <RETURN>

+++
```

NOTE 1: A leading zero '0' must be present 1 through 9, i.e. 01, 02, ... 09.

```
NOTE 2: MON  TUE  WED  THU  FRI  SAT  SUN
         01   02   03   04   05   06   07
```

## SETCLOCK

(continued)

NOTE 3: The clock is set in 24 hour time thus:

| | |
|---|---|
| 00 = 12:00 PM (midnight) | 13 = 1:00 PM |
| 01 = 1:00 AM | 14 = 2:00 PM |
| 02 = 2:00 AM | 15 = 3:00 PM |
| 03 = 3:00 AM | 16 = 4:00 PM |
| 04 = 4:00 AM | 17 = 5:00 PM |
| 05 = 5:00 AM | 18 = 6:00 PM |
| 06 = 6:00 AM | 19 = 7:00 PM |
| 07 = 7:00 AM | 20 = 8:00 PM |
| 08 = 8:00 AM | 21 = 9:00 PM |
| 09 = 9:00 AM | 22 = 10:00 PM |
| 10 = 10:00 AM | 23 = 11:00 PM |
| 11 = 11:00 AM | 24 = 12:00 PM (midnight) |
| 12 = 12:00 AM (noon) | |

The HD146818 version runs like this:

```
YEAR          (84 - 99): 1985 <CR>
DAY OF WEEK   (MON = 1):    1 <CR>  (1 to 7 valid)
DAY OF MONTH  ( 1 - 31):   1  <CR>
MONTH OF YEAR ( 1 - 12):   1  <CR>
HOUR OF DAY   ( 1 - 12):   12 <CR>
MINUTES       ( 0 - 59):   00 <CR>
SECONDS       ( 0 - 59):   00 <CR>
AM OR PM      ( A - P ):   AM <CR>  (you don't type the 'M')

HIT ANY KEY TO START CLOCK ....
```

This version will restrict you to within a pre-defined field. You may edit the data by back-spacing over an entry. Once the data is ready simply hit <CR> (carriage return). Note that leading zeros are not required in this version.

If an incorrect response is given you will be re-prompted for the information. Hitting ^C (CONTROL-C) at any time will return control to FLEX without altering the clock registers or the FLEX system information. This applies to both versions.

The disc files associated with this utility are as follows:

SETCLK-N.ASM ... assembly language source file for MM58167 version (TIM1).
SETCLK-N.CMD ... FLEX TCA command for MM58167 version.

SETCLK-H.PL9 ... PL/9 source file for HD146818 version (TIM2).
SETCLK-H.CMD ... FLEX TCA command for the HD146818 version.

You should use the RE-NAME utility to rename the appropriate command file 'SETCLOCK.CMD'.

# SETFORM

SETFORM is used to set up the data table contained within the spooler that enables text to be printed within a defined "window" on a page of paper.

## DESCRIPTION

The syntax for SETFORM is:

```
SETFORM[,<tskip>,<depth>,<bskip>,<width>]
```

<tskip> is the number of lines that should be skipped at the top of a page,

<depth> is the depth of printable lines,

<bskip> is the number of lines that should be skipped at the bottom of a page,

<width> is the number of character positions across the page.

The sum of <tskip>, <depth> and <bskip> should match the number of effective lines on the page. For example, an 11" deep page of paper has 66 effective lines if the printer has a vertical spacing of 6 lines per inch. The <width> parameter is provided mainly for centering of page numbers when F is used with the title option.

The range for each parameter are:

```
<tskip>   0..255
<depth>   5..255
<bskip>   0..255
<width>  19..255
```

If the parameters are omitted SETFORM will display the existing format, otherwise the new values will be updated and displayed. If one parameter needs changing ALL have to be specified. In the event of an error in a parameter, SETFORM informs which parameter is invalid. Some examples of SETFORM follow:

## SETFORM

(continued)

```
   +++SETFORM
```

Here SETFORM will only print out the current values that P, O, F and the spooler itself will use. The following few lines are the result of such a call:

```
Top      margin is:   2 Lines.
Printable depth is:  61 Lines.      (NOTE: sheet length is 66 lines.)
Bottom   margin is:   3 Lines.
Printable width is:  80 Characters.
```

```
   +++SETFORM,5,53,8,132
```

This example will set up the data table for a 132 character width printer, with a top margin of 5 lines, 53 printable lines and 8 bottom margin lines.

```
******************
* IMPORTANT NOTE *
******************
```

Do not use SETFORM in it's updating mode if the spooler is active. This is because the spooler keeps a count of the line on the current page as it is being output. As each page is printed this value is compared to the total length of each page in order to allow the spooler to SEEK TO THE TOP OF PAGE in preparation for the next file to be spooled. Altering the data table will cause the spooler to leave the paper in an incorrect position. It is permissible to use SETFORM to check on the current values at any time.

# SKIP

The SKIP utility allows restarting the spooler on a specific page.

## DESCRIPTION

The syntax for the SKIP utility is:

```
SKIP
```

It is a simple invokation because it is data driven as a result of the answers to prompted information. There are two ways of using SKIP: the first is used if the spooler is and has been operating normally; the second is used if the printer has been touched so that the spooler may have lost count of the line numbers. An example of each follows:

```
+++SKIP
```

The title banner is printed followed by a request for the page number. This should lie in the range 1..256. SKIP then asks if you want to abort the skip operation. Assuming you carry on, SKIP asks if the printer has been touched. This example assumes that it hasn't so you would type "N" (anything other than "Y" is taken as a no). SKIP then asks if the answers are all correct. Typing "Y" will cause the spooler to seek to the required page, otherwise you will be prompted from the page number and continue from there onwards.

```
+++SKIP
```

The title banner is printed followed by a request for the page number. When SKIP asks if you want to abort the skip operation type an "N". When SKIP asks if the printer has been touched, reply with a "Y". SKIP then asks you to position the paper in the printer and then hit a key to continue. You should now position the paper correctly in the printer. Hitting any key will cause the spooler to seek to the required page. There is no option to go back over the answers this time as the printer has been fiddled with and so a normal resumption could not be affected.

If the spooler was part way through outputting to the printer when the first example was used, it will terminate the print leaving the paper in the correct position by seeking to the top of the next page. Irrespective of the type of SKIP used, the spooler will then read through the file until it finds the required page, and then it will start printing from there onwards.

## SPOOL.SYS & SPOOL-XX

These files are part of the WINDRUSH enhanced print spooler package. The file entitled 'SPOOL.SYS' is the enhanced print spooler overlay for FLEX. The file(s) entitled 'SPOOL-XX.CMD' are various overlays which are configured for specific hardware timers which generate the IRQ interrupts required by the print spooler.

### DESCRIPTION

The general syntax of these utilities is:

```
GET,SPOOL.SYS

 ... and ...

SPOOL-XX
```

The former will load in the spooler overlay and the latter will load in the hardware timer overlay. We suggest that you include these two calls in your STARTUP file.

We can supply spooler interrupt timer drivers for a variety of devices, the Hitachi HD146818 clock-calendar, the Motorola MC6840 programable timer, the National MM58167 clock-calendar, and the Rockwell R6522 VIA. We use the '-XX' suffix to destinguish between the four types. The first letter codes H, M, N, and R apply to Hitachi, Motorola, National and Rockwell respectively. Since these devices may reside at different addresses we use the second letter code is used to differentiate between the various possible addresses. The following are a few examples of the spooler timer drivers we can supply:

```
SPOOL-HC    is the HD146818 overlay for our 6U-CPU/MEM1 board.
SPOOL-HT    is the HD146818 overlay for our 3U-TIM2 board.

SPOOL-MP    is the MC6840 overlay for our 'PHOENIX' system.
SPOOL-MT    is the MC6840 overlay for our 3U-TIM1 or TIM2 boards.

SPOOL-NP    is the MM58167 overlay for our 'PHOENIX' system.
SPOOL-NT    is the MM58167 overlay for our 3U-TIM1 board.

SPOOL-RC    is the R6522 overlay for our 3U-CPU1 board.
SPOOL-RT    is the R6522 overlay for our 3U-TIM1 or TIM2 boards.
```

The exact address of the device can be found by looking at the assembly language source of these modules. If these utilities are part of a 'CONFIGURED' FLEX disk we generally only supply the version appropriate for the hardware we are supplying it with to prevent confusion. The executable command file will have a '.CMD' suffix and the assembly language source file will have a '.ASM' suffix.

Executing any of these commands will cause the appropriate areas of FLEX to be overlayed with the spooler timer drivers. Once the overlay has loaded control will be returned to FLEX.

> If you wish to use the standard FLEX spooler DON'T load the file called 'SPOOL.SYS'; simply load the 'SPOOL-XX' file. If you choose not to use our spooler DON'T use PRINTT, QCHECKK, SETFORM or SKIP as these are configured for our enhanced spooler.

# SPLTITLE

The SPLTITLE utility is used in conjuction with the Windrush Micro Systems enhanced spooler. It allows spooled files to have a title page which depicts the: file name (no drive number or extension) in a large bold banner; the complete filename specification; day, date and time.

## DESCRIPTION

The syntax for this command is:

```
SPLTITLE
```

This loads the body of the utility below MEMEND as it must be resident for the spooler to access it.

After loading the body, submission of jobs to the print spooler with the PRINTT utility default to titles enabled. To inhibit the title page, the following example invocation of PRINT can be used:

```
+++PRINTT,DIARY,N
```

This will submit O.DIARY.OUT to the spooler WITHOUT the title page being printed.

If, subsequently, it is realised that the titles should/should not be printed, then QCHECK may be used to dynamically change the title attribute. The H command (Headings) is used as T is already used (Terminate). Use the A command (Assistance) within QCHECK to find the actual syntax required.

NOTE: QCHECKK is clever enough to recognise when the SPLTITLE utility has been invoked, and so the H command is only valid (and thus visible to the A command) after SPLTITLE has been installed.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                       *
*      REMEMBER SPLTITLE RESIDES BELOW MEMEND.          *
*                                                       *
* THIS MEANS THAT MEMEND WILL BE LOWER THAN NORMAL!     *
*                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

This information is being printed in advance of the release of the SPLTITLE utility. Therefore this utility may not be present on your disk. When released only the object code of the utility will be supplied.

We plan to produce the the following versions:

```
SPTLE-HC  ... configured for the HD146818 on 6U-CPU/MEM1.
SPTLE-HT  ... configured for the HD146818 on 3U-TIM2.

SPTLE-NT  ... configured for the MM58167 on 3U-TIM1.
```

You should rename the utility appropriate for your hardware 'SPLTITLE.CMD'.

# VER

The VER utility is used to interrogate system commands and utilities to determine which version is being used. All .CMD files should have such a version number coded into them - unless badly written!

## DESCRIPTION

The syntax for VER is:

```
VER,<filename>[,<filename>]
```

<filename> is the filename in question. The defaults for the filename are a .CMD extension on the system drive.

It is possible to optionally include more filenames in the same invocation of VER - this will help when checking version numbers of several different files. VER can detect version numbers as being in one of three main formats: TSC, SWTP and WMS. Some examples follow:

```
+++VER,VER
```

This will display the version number of the release of VER being used.

```
+++VER,MACE,PL9
```

This will display the current version numbers of MACE and PL9 on your system (if you have them!).

```
+++VER,TEST.BIN.1,DUMP.2
```

Here the file TEST.BIN on drive #1 will be interrogated, followed by the file DUMP.CMD on drive #2.

VER will print out the filename, creation date and version number for each requested file. If the system is defaulted to ALL the utility could be on any drive, and so a "?" is printed for the drive #. Various problems can occur and they will be displayed, examples are: unoptimised, not binary, invalid, can't locate the file.

# XXOUT

The XXOUT utility is used to delete files with a .OUT extension which are not in the spooler queue..

## DESCRIPTION

The syntax for XXOUT is:

```
XXOUT[,<drive>]
```

<drive> is an optional drive number. If omitted the drive defaults to the defined system drive. If in turn this is defined as ALL then all on-line drives will be processed.

XXOUT deletes all those files that are found on the drive(s) specified/defaulted which are NOT IN THE SPOOLER QUEUE. This utility is therefore a safer way than using ZAP to clean up the .OUT files as it will not leave the spooler pointing into the free chain! Some examples follow:

```
+++XXOUT
```

Here, all .OUT files on the system drive (or all on-line drives if system is defined as ALL) that are NOT IN THE SPOOLER QUEUE will be deleted.

```
+++XXOUT,3,1
```

Those .OUT files on drive #3 and then #1 will be deleted if they are NOT IN THE SPOOLER QUEUE.

# Y

The Y utility is used as a source of input which always returns a "Y" character as though it had been typed from the keyboard in response to a request for input.

## DESCRIPTION

The command line syntax for the Y command is:

```
Y,<command>
```

<command> is the actual command that is to execute followed, if necessary, by it's parameters.

Whenever <command> expects to get input, it will "see" a "Y" typed at it. This may seem a trivial task, but the following example (where it is often used) demonstrates it's usefulness:

```
+++Y,COPY,0,1
```

This will copy all files from drive # 0 to drive # 1. If a file exists on drive # 1, both the the "delete" and "are you sure" prompts will automatically be answered by a "Y", therefore the effect of this command line would be to copy all files existing on drive # 0 to drive # 1 in addition to those that did not exist.

# YEAR

The YEAR utility is to alter the system year register, or display the current system year which is stored at $CC10.

## DESCRIPTION

The syntax for YEAR is:

```
YEAR

YEAR,<year>
```

<year>  is a number in the range 0..99.

The first type of call will print out what the computer thinks the present year actually is. The second type will set the system year to become <year>. An example of each follows:

```
+++YEAR
```

This will merely inform you of the system's year.

```
+++YEAR,85
```

This will set the system's year register to '85'.

YEAR is normally used because some types of real-time clock-calendar chips do not have a year register internally. When such a system is booted up, YEAR should be invoked in the STARTUP file/sequence followed by a call on the utility that reads the clock-calander to put the day and month into the date registers of the computer. The call to YEAR will only need to be changed/edited annually, e.g. on 1st January 1986 it should be made to be "YEAR,86", and on 1st January 1987 "YEAR,87". For all times in-between, the STARTUP file will cause the currently defined year to be put into the computer's year register.

# ZAP

The ZAP utility is used to erase files without any user intervention. It is also very useful for deleting all files on a disc very quickly - hence it should be USED WITH CAUTION. It will not delete any WRITE or DELETE protected files ... but it will have tried to!

## DESCRIPTION

The syntax for the ZAP utility is:

```
ZAP[,<drives>][,<matchlist>]
```

<drives> is: <drive #>[,<drives>] where <drive #> is a valid on-line drive,

<matchlist> is: <match>[,<matchlist>] where <match> is either (or both) part(s) (or all) of a filename or extension.

Omission of <drives> defaults to the defined work drive which in turn defaults to each drive on the system if ALL is defined. Omission of a <matchlist> defaults to all directory entries.

Having typed a command line no further action is needed.

NOTE: If a BIG MISTAKE has been made, hitting ^C (control-C) at any time will result in a clean return to FLEX.

Some examples of ZAP follow:

```
+++ZAP
```

This will cause all files on the work drive (defined by ASN) to be deleted. This invocation can be CATASTROPHIC if the work drive is defined as 'ALL'.

```
+++ZAP,0
```

This will cause all files on drive #0 to be deleted.

```
+++ZAP,1,A
```

This will cause all files that are found on drive #1 and start with an A to be deleted.

```
+++ZAP,2,.TMP
```

This will cause all files that have a .TMP extension on drive #2 to be deleted.

```
+++ZAP,0,1,PR.DEV,.OUT
```

This will cause those files that start with PR and have a .DEV extension, followed by those files that have a .OUT extension on drive #0 and then drive #1 to all be deleted.