COPYCAT

USER GUIDE AND REFERENCE MANUAL

by

NIGEL W. BENNEE

COPYRIGHT © 1981 LUCIDATA LTD

All rights reserved - supplied for single end user use
unless specified in an agreement to the contrary

Published by: LUCIDATA LTD., P.O. Box 128, CAMBRIDGE, CB2 5EZ, ENGLAND

TABLE OF CONTENTS

1. INTRODUCTION

The successful use of floppy disks, as a serious mass memory device for microcomputers, has been the major hardware factor in raising the status of a low cost technical toy, the hobby computer, to that of a useful general purpose computing system.

The emergence of a multitude of micro-computer software , effectively utilizing this hardware has been the other factor. Much of this software has been inspired by the experience gained in many years of exposure to large mainframes and mini-computers. Some of it has even benefitted from hindsight and sought to avoid some of the failings of its conceptual grandparents. There is now available a large choice of floppy disk based systems, offering many of the features of large mainfromes, at a fraction of the cost, and in many cases providing a significantly better user environment.

All is not roses however, as these small systems have inherited some of the old problems, in particular mutual incompatibility. If we restrict the discussion to the problem of exchanging text files between systems running different DOSs we find few compatible media. Paper tape or the micro-equivalent of "Industry standard 9 track magnetic tape" , the Kansas City cassette, are about all there is. When it comes to the exchange of large files both methods pale in comparison to the compact reliability of the floppy disk.

COPYCAT seeks to recover the benefits of the floppy disk as a transport medium between different systems.

It should be emphasized that COPYCAT cannot perform magic. Situations will be encountered when physical incompatibility will foil attempts to read a disk. A complete section has been included in this document on the subject of physical compatibility.

COPYCAT provides utilities to list the Directories of, and to copy files from, disks created on different systems. It also provides utilities to convert text files where there are differences in code conventions, and a utility to assist in investigating physical compatibility problems.

The utilities are provided in source form to allow user adaption to variant situations.

Release 1.0 is tailored to run under Technical Systems Consultants FLEX 1.0, FLEX 2.0, or FLEX 9.0 and will execute on any (6800/6809) hardware running these operating systems.

Release 1.0 allows reading disks created on the following systems: Technical Systems Consultants mini-FLEX; Smoke Signal Broadcasting DOS68; Digital Research CP/M.

2.DESCRIPTION OF UTILITIES

The following pages contain the descriptions of the COPYCAT utilities organised according to the DOS format they support.

The release material has been assembled on the basis that the foreign disk will be 5", single sided and single density. No logical problems are forseen in these cases but in the case of double sided, double density it may be neccessary to replace the call to the FLEX entry SELECT with your own code. This situation would be apparent by the order of the sectors being mixed up.

The source code has been written in a highly modular manner to simplify adaption by the user to tackle specific problems.

In the following pages these conventions apply:

Tokens between < and > indicate the default values that are used if no explicit value appears.

The symbol 'W' indicates the current work drive.

2.1 mini-FLEX

These utilities will probably only work on systems that use WD1771 controllers.

2.1.1 Directory DIRMFX

Called by

DIRMFX <,W>

Produces a directory listing of the mini-FLEX disk mounted on the specified drive. The listing includes the name of each file, its position in the directory and its size in sectors. It also lists the track and sector addresses of the first and last sector of each file. The total number of undeleted files and the number of disk sectors occupied are reported in a summary.

2.1.2 Copy COPYMFX

Called by

COPYMFX,N.SOURCE<.TXT>,<W.>DESTINATION<.BIN>

Performs a transparent (binary) copy of the mini-FLEX file specified by SOURCE to a FLEX file specified by DESTINATION. Drive N must be explicitly stated. The number of sectors transfered are reported.

2.2 DOS68

2.2.1 Directory DIRSSB

Called by

DIRSSB <,W>

Produces a directory listing of the DOS68 disk mounted on the specified drive. The listing includes the name of each file, its position in the directory and its size in sectors. It also lists the track and sector addresses of the first and last sector of each file. The total number of undeleted files and the number of disk sectors occupied are reported in a summary.

2.2.2 Copy COPYSSB

Called by

COPYSSB,N.SOURCE<.TXT>,<W.>DESTINATION<.BIN>

Performs a transparent (binary) copy of the DOS68 file specified by SOURCE to a FLEX file specified by DESTINATION. Drive N must be explicitly stated. The number of sectors transfered are reported.

2.2.3 Conversion SC2FX

Called by

SC2FX <,W.>SOURCE<.BIN>,<,W.>DESTINATION<.TXT>

If the file specified by SOURCE is the result of copying a DOS68 text file using COPYSSB, then it may contain space compression codes which follow the DOS68 conventions. SC2FX converts such codes to those supported by FLEX. The newly created file specified by DESTINATION is a normal FLEX text file.

2.3 CP/M

2.3.1 Directory DIRCPM

Called by

DIRCPM <,W>

Produces a directory listing of the CP/M disk mounted on the specified drive. The listing includes the name of each file, its position in the directory and its size in sectors. It also lists the extent number of each entry. The total number of undeleted files and the number of disk sectors occupied are reported in a summary.

2.3.2 Copy COPYCPM

Called by

COPYCPM,N.SOURCE<.TXT>,<W.>DESTINATION<.BIN>

Performs a transparent (binary) copy of the CP/M file specified by SOURCE to a FLEX file specified by DESTINATION. Drive N must be explicitly stated. The number of sectors transfered are reported.

2.3.3 Sector Map CPMMAP

This is not a utility but belongs fairly with the CP/M descriptions.

The CP/M BIOS (Basic Input Output System) provides a simple interface to the CP/M operating system which helps to make the system so transportable. In essence the file manager of CP/M works with virtual disk sectors which are the same for (all) CP/M systems and the mapping to physical disk sectors is performed by the BIOS. In the case of the COPYCAT utilities the mapping is performed by the routine GETLS (GET Logical Sector) in conjunction with the text file CPMMAP.TXT. The mapping is usually a fairly simple sector interlace. To illustrate. The sequence of logical sectors for the directory of a 5" disk would go 0,1,2,3 etc. A software interlace of three would mean that the physical sectors used would actually have addresses 3,6,9,12 etc. to allow some time between logically sequential sectors for processing. The idea is to avoid missing the passing of the next required sector and hence invoke the penalty of a whole revolution of latency. If text files copied with COPYCPM appear to be mixed up but exhibit short range order, then the map provided is probably not appropriate for the disk you are trying to read. ANALYSIS will enable you to see the layout of the text on the disk surface BUT BEWARE, the actual sector addresses are not neccessarily contiguous and you will have to read the sector number from the record identification field in order to construct a new map.

The record identification field is flagged by an FE byte and is followed by the track number byte, side byte, sector number byte and sector size byte.

3.PHYSICAL COMPATIBILITY

A complete discussion of magnetic recording techniques and the factors affecting their usefulness is beyond the scope of this document. Readers are recommended to seek further information, if required, from manufacturers data sheets and their own systems documentation.

3.1 General Considerations

There are basically two physically distinct methods of structuring the floppy disk medium into discrete storage areas. The first, called hard sectoring uses a permanent set of holes, punched in the disk at manufacture, to once and for all determine how many sectors there will be, and where on the disk they will start. The holes are sensed by hardware and the data starts a short distance further on after relatively short clock synch and write switch gaps. The hardware also keeps a physical account of which track the head should be positioned at. Additional control information may be written in the data sector for validation purposes but this method essentially relies on the presence of mechanical marks.

The second method, called soft sectoring relies totally on information recorded on the disk to control the management of the medium. Before a new disk can be used on a soft sectored system it must be formatted. Formatting is the process of writing an empty data structure over the entire surface of the disk. Formatting is usually performed on a whole track at a time by writing a predefined track image assembled in the memory of the computer.

This image contains special bit patterns which are recognised by the disk controller hardware/firmware/software dependent on the design of the system. The bit patterns signal the presence of fields such as "record identification field follows". Such a field will contain information about the Data Field which follows it such as its physical track and sector address and the sector size. Another special bit pattern announces the start of the actual data. Fields are usually bracketed by a few zero bytes to allow for timing tolerances in the hardware which might otherwise permit new data to be written over control information thus making the disk unusable.

The logical allocation and use of the disk tracks and sectors by different systems is something that can be easily emulated by software, as the COPYCAT utilities demonstrate. Unfortunately, whilst there are a very convenient set of format standards published by IBM, there are many "not quite standard" interpretations of the soft sectored format. There are also differences in the expectations and abilities of the very clever LSI controller and formatter chips usually found controlling the disk drives. In short it is possible that even though on paper the logical structure of a disk is understood and the software exists to unravel it, there may yet be insurmountable physical incompatibility.

The philosophy behind COPYCAT is that as far as possible the user should be able to treat a foreign disk as if it were generated on his own system. It is not expected that the user should need to be in possession of specialist technical knowledge to make it work. It is also considered essential that use of the utilities should not interfere with the integrity of the host DOS, hence no hardware is accessed directly. All disk operations are performed through FLEX disk driver entries. FLEX expects 256 byte sectors but will of course read less (128). As an error would be reported if normal File Management System (FMS) calls were made, COPYCAT uses the Read Single Sector entry (READSS) of the disk drivers. The lost data error status which is returned is ignored and account only taken of the "Record not Found" status as this can be indicative of a physical incompatibility.

The utility ANALYSIS is provided to help determine the likely cause of the failure of the other DIRectory and COPY utilities.

3.2 Structural Analysis

The function of the ANALYSIS utility is to provide a diagnostic tool to investigate disk structures that have defeated the other utilities. It is intended to form the basic probe framework for the user to hang on additional specific modules as neccessary.

At the logical level, which it is expected will be the normal level of use, ANALYSIS does not access hardware directly and in consequence will function as supplied. The logical level only supports the investigation of sector contents NOT the environment of the sector.

In the event that the sector environment needs investigation, ie. gap values, header structure etc. the address of the systems controller chip (DCCHIP) must be assembled into the program so that the read track routine (READTK) can function correctly. It is assumed that the controller data path is directly accessible and is WD17XX compatible.

The read track routine provided may also function as a read sector routine if the function code is changed. Due to maximum wait time constraints on certain controller chips this code will only handle 8" double density disks properly if the processor runs at 2MHz.

3.3 Use of ANALYSIS

Called by

ANALYSIS

The program will prompt for the drive number and a RESTORE operation is then performed on that drive. The menu of options is displayed and selection may be effected by entering the first character of the menu option.

The display Single sector option requires specification of a sector in the form TTSS where the TTrack and SSector addresses are specified in hexadecimal. The display pauses after each sector has been displayed so that the user may Increment to the next, Decrement to the last, or specify a completely New sector by entering the significant character.

The display whole Track option prompts for the TTrack number and displays every byte found on the track including all gaps and special patterns. This option will always produce some output on any disk that is even remotely IBM similar. It should always be possible to discover why you cannot use the other utilities by using this feature.

The Analyse track option performs the same read track function but then scans the data several times and extracts some simple statistics, which can give guidance to the nature of the disk structure and its origins.

Finally it should be emphasised that use of ANALYSIS usually means you cannot do what you hoped you could, try paper tape.