6809 FLEXTOPER Operating System



The FLEX™ Disk Operating System

COPYRIGHT INFORMATION

This entire manual is provided for the personal use and enjoyment of the purchaser. Its contents are copyrighted by Technical Systems Consultants, Inc., and reproduction, in whole or in part, by any means is prohibited. Use of this program, or any part thereof, for any purpose other than single end use by the purchaser is prohibited.

DISCLAIMER

The supplied software is intended for use only as described in this manual. Use of undocumented features or parameters may cause unpredictable results for which Technical Systems Consultants, Inc. cannot assume responsibility. Although every effort has been made to make the supplied software and its documentation as accurate and functional as possible, Technical Systems Consultants, Inc. will not assume responsibility for any damages incurred or generated by such material. Technical Systems Consultants, Inc. reserves the right to make changes in such material at any time without notice.

FLEX User's Manual

FLEX User's Manual

COPYRIGHT © 1979 by
Technical Systems Consultants, Inc.
P.O. Box 2570
West Lafayette, Indiana 47906
All Rights Reserved

FLEX is a trademark of Technical Systems Consultants, Inc.

COPYRIGHT INFORMATION

This entire manual is provided for the personal use and enjoyment of the purchaser. Its contents are copyrighted by Technical Systems Consultants, Inc., and reproduction, in whole or in part, by any means is prohibited. Use of this program, or any part thereof, for any purpose other than single end use by the purchaser is prohibited.

DISCLAIMER

The supplied software is intended for use only as described in this manual. Use of undocumented features or parameters may cause unpredictable results for which Technical Systems Consultants, Inc. cannot assume responsibility. Although every effort has been made to make the supplied software and its documentation as accurate and functional as possible, Technical Systems Consultants, Inc. will not assume responsibility for any damages incurred or generated by such material. Technical Systems Consultants, Inc. reserves the right to make changes in such material at any time without notice.

PREFACE

The purpose of this User's Guide is to provide the user of the FLEX Operating System with the information required to make effective use of the available system commands and utilities. This manual applies to FLEX 9.0 for full size and mini floppy disks. The user should keep this manual close at hand while becoming familiar with the system. It is organized to make it convenient as a quick reference guide, as well as a thorough reference manual.

GIMIX FLEX USER NOTES 1

The following programs in the TSC 6809 DISK DIAGNOSTICS package will not work with GIMIX FLEX 4.0.

PROGRAM NAME	NOTES
TEST.CMD	Will give errors for track 00 on a double density 5" disk.
VALIDATE.CMD	Will abort with the message 'INVALID SYSTEM INFO SECTOR'. This only applies to 5" disks formatted for more than 40 tracks or formatted double density, or 8" disks formatted for more than 77 tracks.
COPYR.CMD	Same as 'TEST.CMD' above.
EXAMINE.CMD	Same as 'TEST.CMD' above.
FLAW.CMD	Same as 'TEST.CMD' above.
REBUILD.CMD	Same as 'TEST.CMD' above.

NOTE: This is because these programs do not support double density or more than 40 tracks on 5" disks, or more than 77 tracks on 8" disks.

FLEX UTILITIES

All FLEX utilities from TSC run correctly with GIMIX FLEX except DIR. This program can cause the system to hang or crash if it is used while the print spooler is active. This is due to the way the spooler uses the stack. DIR will function perfectly when the print spooler is not active.

FLEX is a trademark of Technical Systems Consultants, Chapel Hill, North Carolina.

GIMIX is a registered trademark of GIMIX, Inc., Chicago, Illinois.

2000년 - 1000년 - 1000년 - 1000년 1200년 - 100년 120년 - 100년 - 100년 120년 - 100년 120년 120년 120년 120년 120년 120년 120년	
요즘 이 이 그 전문 그렇다. 그리고 있는 내용하게 되었다면 하셨다면 있다면 없다면 하는 사람들이 되었다면 하는 사람들이 되었다면 하는 것이다.	
[절차] 그림, 이 , 이 , 이 하는데, 그 나는 귀리 이 중요한 이 있는데, 그리는 이 중요한 사람들은 이 중요한 사람들은 이 경우를 받는데 하는데 그렇게 되었다.	
[[[[[[[[] - [[] - [[] - [] - [] - [] -	
경소, 사람 집단에 있는 여러는 그는 때에 그렇게 있었다. 그 사람들이 있는 이 가는 그를 하는 것 같습니다.	
존재용 그들 그는 이번에 그렇는 어린답인 네트리스 등에 발표한 바람이라면 그리고 내가 되었다면서 되었다면 되었다.	
가 있다. 그는 사람들은 인과의 가장들의 가장 하는 것 같을 하는 것이 가는 사람이 하다. 전문이 하다는 것이 되었다.	
근처들이 이 시간부터의 아고 사이의 시간의 시작 시원의 작품들이 가고 하는 것을 들어 먹었다. 그것	
그렇게 그는 물과, 그렇게 하겠어요? 하나 하나 하나 나는 그 나는 것이 없어 그렇게 되었다.	
아니아 아픈데 아랫 등은 회에는 아니라 아들에 가는 생각이 되는 것은 모양을 하는 그는 사는 것이 아니라 되었습니다. 그 없다는	
이 생물하다 하는 사람들은 사람들이 하는 사람들이 나는 사람들이 나를 하는 것이 되었다. 이 아름은 사람들이 되었다.	
대화 생기의 가장 마시 나는 그는 그 그 그 그 그 그 그 그 그 그는 그 그는 그는 그 그는 그를 하고 그는 그 가지 않는 것 같다.	
보이다. 이 친정 보이 되어 가게 하는 것이라고 하는 것이다. 그는 그는 그 그리고 사용하다는 것 수 있다고 없는데 없었다.	
(1985년 1985년) 1982년 - 1982년 - 1982년 - 1982년	
그 하는 이 사용 등장이 전쟁을 들어 가는 사람들이 아시 회사가 있었다. 하네가 나는 이 가는 이 것은 아니라는 이 모양	
이 발표하는 것이 있는 것이 되었다. 이 가는 사람들은 사람들이 되는 것이 되었다. 그는 사람들은 사람들이 되었다. 유럽하다 이 사람들은 아니라 하는 사람들이 되었다. 이 사람들이 사람들이 사용하는 것이 되었다. 사람들은 사람들은	

Page 1.2

Paragraph II (SYSTEM REQUIREMENTS)

GIMIX FLEX requires memory from location \$0000 through location \$3FFF as well as memory from location \$C000 through location \$DFFF.

Paragraph III (GETTING THE SYSTEM STARTED)

When using GMXBUG-09 with the Disk Boot Prom installed type 'U' followed by a carriage return in response to the GMXBUG-09 prompt. If you have a video system and have a Disk Boot Prom then the 'U' command will bootstrap the system.

Page 1.3

Paragraph IV (DISK FILES AND THEIR NAMES)

The actual number of sectors available to the user may vary according the density, number of tracks, and size of the disk, and whether one or both sides were formatted. See the 'FORMAT' command for more information.

Page 2.1

Paragraph 7 (GENERAL SYSTEM FEATURES)

In addition to the features already mentioned GIMIX FLEX also gives the user the following additional features: selection of stepping speed, write protect, and "double stepping" from the console, and automatic density selection. See the SYSGEN and SETUP commands for more information.

Paragraph I (DISK CAPACITY)

See addendum for Paragraph IV, Page 1.3

Paragraph V (ACCESSING DRIVES NOT CONTAINING A DISKETTE)

If a 5" drive which doesn't have a disk in it is accessed, the system will hang up until the user puts a disk in the drive. This applies to all older 5" drive models. Some new 5" drives have a READY/NOT READY output. If one of these drives is accessed when it is empty, FLEX will return 'DRIVES NOT READY'.

Page 3.4

Paragraph VIII (FLEX OPERATING SYSTEM INPUT/OUTPUT SUBROUTINES)

GIMIX FLEX uses the GMXBUG-09 (SBUG-E compatible)

	물이 집에 살아 그렇다면 그래면 그런도 말라고요.
저는 그렇게 하는 이번 이어를 내용한 일반 있는 아이들 방식으로 하는 그리는 하는 그리는 하는 것이 되었다. 그는 사람이 모든 사람이 없는 것이다.	
그리다 마일하는 어제 들어갔다. 그리는 왕으라들이 그리는 그들은 사람들에게는 이상에 하는 다른 것은 다른 모든 사람들이 되었다.	
	길이 가는 없어요. 뭐 없이 되다 그 이렇다면 그 그
나 하다. 그리고 있는 그는 그는 그들이 하는 그들이 하는 그리고 나는 그들은 것이다. 그는 생활을 받는	
그들은 이 회사들은 그리는 그는 것 같은 그는 일반을 만든 것들은 중에 그는 물이 되는 생님은	
요. 사람들 할 때 없는 그 사람들이 되는 것이 되는 것이 없는 사람들이 되었다. 그 사람들이 되었다.	
경기와 가끔 보고 있다. 이 아이들 그리고 그리고 있는데 그리고 있다. 그리고 있는데 하는데 그리고 있다. 그리고 있는데	
가득, 그림과 얼마나 보다 그 나가 하는 것 같아. 그런 그리고 하다 하게 되는 것 같아 되었다. 그 그리고 하는 것	
그리고 뭐까? 하는 사람들은 다른 모든 사람들은 물로 하다는 것은 사람들은 사람들이 되었다. 이 사람들이 없다는 사람들이 되었다.	
가는데 그 이번 이렇게 그래면 모든 이번 내가 되는 이 나는데 살아가는 사람이 가지를 만들어 이 먹는 어떻게	

input/output routines for total system compatibility.

Page 3.6

Paragraph IX (BOOTING THE FLEX DISK OPERATING SYSTEM)

If neither the VIDEO prom or the Disk Boot Prom is installed in the system the user must hand enter the bootstrap. The GIMIX FLEX bootstrap is listed on the sheet entitled 'GIMIX 6809 RELOCATABLE DISK BOOT'. For instructions on how to enter this bootstrap see the GMXBUG-09 manual.

Page 3.7

Paragraph X (REQUIREMENTS FOR THE 'PRINT.SYS' DRIVER)

Object code and source code for two printer drivers is supplied on the GIMIX FLEX system disk, one each for parallel and serial interfaces. List the file 'READ-ME' on the system disk for more information.

NOTE: GIMIX FLEX does not have a 'NEWDISK' command. This has been replaced with the 'FORMAT' command.



The disk driver routines in GIMIX FLEX 4.0 have been completely rewritten. The USEMPT, USEDC4, USEDMF, and UNUSE commands are not usable with this FLEX. ANY ATTEMPT TO USE THEM WITH GIMIX FLEX 4.0 WILL CRASH THE SYSTEM!!! Users who still need to access DC4 and DMF format disks must use GIMIX FLEX 3.6. Also, the SETUP, REPORT, and BACKUP commands have been rewritten for GIMIX FLEX 4.0. The 3.X versions of these commands will not work, and may crash the system!

	하는 다음을 되는 것이 되었다. 하는 사람이 보는 것이라고 살았다.
	도 경우는 기억에 지르면 호텔 이 보다 하는 사이를 모르는 것
가는 생활하다는 그 보다는 사람들은 사람들이 되었다. 이 교통에는 보고 있는 사람들은 사람들이 되었는 것이 되었다. 그런 사람들은 사람들이 되었다. 생활화 발생하는 것이 되었다. 그는 사람들은 사람들이 되었다. 사람들은 사람들은 사람들이 사람들이 되었다. 사람들은 사람들은 사람들이 되었다.	
그 회사는 이 나는 이 문화를 하는 것이 하고 있다. 그들은 그렇게 되는 것이 되었다. 닭	
그리는 사용물을 하는 하는 이는 이번 사람들이 그는 이 사용이라고 있다. 하는 이 하는 이 사람들은	
[발생물] 사람이 나는 사람이 되었다. 그 그래요? 그 아들이 나는 그리고 그리고 있다. 그 아들이 되었다. 그리고 없는 사람이 되었다. 그리고 있다면 살아보니 그리고 있다면 살아보니 그리고 있다.	
마르크 발생들이 현실 기업이 되었다. 그는 사람들은 그는 기업을 받는 것이 되었습니다. 그런 그는 그는 것이 되었습니다. 그는 것이 되었습니다. 1985년 1일 1985년 1일	
- 1985년 - 1일 시간에 대한 발생이 되었다. 1985년 1일	
그는 회에 하는 그 사람들이 하는 것이 되는 것은 사람들이 되었다면 하는 것이 되었다. 그는 사람들이 되는 것이 없는 것이 없는 것이다.	
그리고 있는 그 집에 살아 있는데 아이들은 그로 가려면 하는데 하는데 하는데 하는데 하는데 이렇게 되었다. 그는데 하는데 그는데 하는데 하는데 하는데 하는데 하는데 하는데 하는데 하는데 하는데 하	
는 마음을 보고 있다. 그는 사람들은 마음을 보고 있다. 그런 그는 사람들이 가는 사람들이 되었다. 그는 사람들이 되었다. 그는 사람들이 되었다. 10 - 15일	
- 이 등에 보고 있다. 그는 사람들이 되고 있는 것이 되었다. 그는 사람들이 되었다. - 이 보다는 사람들이 보고 있는 사람들이 되었다. 그는 사람들이 보고 있는 사람들이 되었다. 그는 사람들이 되었다면 되었다. 그는 사람들이 되었다면 되었다. 그는 사람들이 되었다면 되었다면 되었다면 되었다면 되었다면 되었다면 되었다면 되었다면	

TABLE OF CONTENTS

		Page
CHAPTER 1 I. II. III. IV. V. VI.	Introduction System Requirements Getting the System Started Disk Files and Their Names Entering Commands Command Descriptions	1.1 a.i , 1.2 a.i , 1.2 a.i , 1.3 1.5
CHAPTER 2		
I.	Utility Command Set	2.1
	APPEND ASN BUILD †*BACKUP CAT *COPY (-GMX,-TSC) *CLEAN *CHECKSUM *CMPBIN DATE DELETE *DCOPY EXEC *EXTEND *FORMAT *FIXDAY *FREEMAP †*HARD †*HARD 1 JUMP LINK LIST *NAME *N O P PRINT PROT QCHECK RENAME *REPORT SAVE STARTUP *SETTIME *SETUP †*SYSGEN TTYSET *TIME *UNDNARL VERIFY VERSION	A.1 A.2 B.1 C.2 C.3 C.4 C.5 D.1 E.2 F.3 H.1 L.1 J.1 L.2 N.1 N.2 O.1 P.3 Q.1 R.2 S.3 S.4 T.1 V.2 V.1 V.1 V.1 V.1 V.1 V.1 V.1 V.1 V.1 V.1
	XOUT *YEAR	X.1 Y.1
	그리아 있습니다. 그는 그 보고 있는 사람들은 가장 사람들이 되었다. 그는 그 사람들은 그리는 그 그 사람들이 가장 사람들이 그 사람이	v o

는 사람들이 있는 것이 되었다. 그런 사람들은 사람들이 되었다. 그런 사람들이 되었다. 그런 사람들이 되었다. 그런 사람들이 되었다. 그런 사람들이 되었다. 하다는 사람들이 그렇게 되었다. 그런 사람들이 보고 있다면 보다는 것이 되었다. 그런 사람들이 되었다.	
- 선생님 등학생 이 이 등 보고 있는데 기능적으로 하는데	(- 호등 기업 기업 등 개업 기업 - 기업 기업 기업 기업 기업 기업 - 기업 기업 기업 기업 기업 기업 기업
보이트리에서 보고 있습니다. 그는 사람들은 이 대통에 들어 있는 사람들은 보고 있었다. 그런 하는 것이 되었다. 보고 있는 경기를 보고 있는데 기록 하는데, 그리고 있을 때 있는데 이 가는 사람들이 되었다. 하는데 보고 있는데 그를 보고 있다.	
1월 일하일이라고 한다. 이 이 시간 보이고 있었는데 이 그 보고 있다. 그 이 그는 이 전에 가는 것이라고 있다. 2012년 - 1일 1일 - 1일	
보고 있는 사용하는 것이 하는 것이 되었다. 그는 사람이 되었다. 그 사람이 되었다는 것이 되었다. 	
경영화경하다. 1987년 - 1987년 - 1987년 1987년 - 1987년	
네 함께 발생하다 하고 보다면서는 보다면 하다면 하는데 하다면 나는데 되었다.	
됐는 그는 마른 일반 한 15 km 보고 있습니다. 그는 15 km 1 k	
고입 교통을 통해 기계를 가고 있다. 사람들은 경우 하는 이 경우를 하는 것이다.	
사람들 다음 보고 있는 것이 되는 것이 되는 것이다. 그는 사람들은 사람들이 되는 것이 되었다. 그는 것이 되었다. 	
들어 살았다. 그런 경험에 대한 경험을 보냈다면 하는 것이 되었다면 되었다면 하는 것을 받았다. 그런 경험을 받았다. 경험 있는 사람들은 사람들은 것으로 하는 것이 없었다면 하는 것을 받았다면 하는 것을 하는 것을 받았다면 하는 것을 받았다.	
사 하다 보고 있는 것을 보고 있는 것이다. 이번 사람들은 사람들이 보고 있는 것이다. 	

CHAPTER 3			
I.	Disk Capacity		3.1
II.	Write Protect		3.1
III.	The 'RESET' Button		3.1
IV.	Notes on the P Command		3.1
V .	Accessing Drives Not Containing a Disk		3.1
VI.	System Error Numbers		3.2
VII.	System Memory Map		3.3
VIII.	Flex Input/Output Subroutines	a.i ,	3.4
IX.	Booting the Flex Disk Operating System	a.ii ,	3.6
X •	Requirements for 'PRINT.SYS' driver	a.ii ,	3.7
XI.	Hardware Configuration	a.i ,	3.10
XII.	Patching FLEX to use the Time-of-Day clock	(see S	SYSGEN
CHAPTER 4			
I.	Command Summary		4.1

[†] These programs are not interchangeable with earlier versions of similar programs or other versions of GIMIX/FLEX

^{*} Programs supplied by GIMIX

마하는 보다는 사람들에 대로 5시간 경기에 되고 있었다. 하는 4호 마이트 아이들이 되었다는 사람들이 되었다는 사람들이 되었다는 것이 되었다. 경기 전혀 모르는 사람들이 되었다. 그 사람들은 사람들이 가장 기를 받는 것이 가장 보았다. 기를 하고 있었다는 사람들이 되었다는 것이 되었다는 것이 되었다. 기를 하고 있는 것이 되었다.	
으로 보고 하는 것이 하는 것이 되는 생각 시간에 대한 것이라고 있다면 보고 하는 것이라고 있다. 그는 것이라고 하는 것이 되었다. 그는 사이를 받았다면 하는 것이다. 당하는 것이 하는 것이 말하는 것이 없는 것이 되었다면 되었다면 말하는 것이 되었다. 그는 것이라고 있는 것이 되었다면 되었다면 없는 것이라고 있다.	가는 이 문제를 하는데 되었다. 보이 보는 영국가 한 전투()는 이번 전통하
다른 마이트 등 보고 있는데 보고 있는데 보고 있는데 보고 있다. 	
요한 종일 등 사람이 이번 이번 사람들이 아니는 보고 있다. 그는 사람들이 가는 사람들이 가는 것이 되었다.	
존해가 되면 하는 이번 이번 이번 사람이 하는 사람들은 사람들이 되었다. 이번 이번 살아나다	
경우를 다고 있는데 이번 사람들이 되었다. 그 사람들은 사람들이 되었다. 그 사람들이 되었다. 그 사람들이 되었다. 그는데 그는데 그 사람들이 되었다. 	
경제 발표를 받는 것이 되는 것이 되는 것이 되었다. 그는 것이 되는 것이 되는 것이 되었다. 그런 것이 되는 것이 되었다. 소화하다 교육하는 것이 되는 것이 되는 것이 되었다. 그는 것이 되는 것이 되는 것이 되었다. 그런 것이 되었다.	
발판 그렇다 하다 하다 하는 것이 나고를 보는 그 말이 하는 것은 모든 것은 하는 것인 모든다.	
사람들이 되고 있다. 이 사람들은 사람들이 되었다면 보고 있는데 보고 있는데 되었다. 그는 사람들은 사람들은 사람들이 되었다면 보고 있다면 보다 되었다면 보고 있다면 보고 있다면 보고 있다면 보고 사람들이 되었다면 보고 있다면 보다면 보고 있다면 보고	
그 사고를 가장 한 것이 되었다. 그리고 하는 것이 하는 것이 되었다. 그 그 그 사람이 되었다.	
다. 그는 사람들은 사용하는 것이 가장 보고 있는 것이 되었다. 그는 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은	

FLEX USER'S MANUAL

I. INTRODUCTION

The FLEX™ Operating System is a very versatile and flexible operating system. It provides the user with a powerful set of system commands to control all disk operations directly from the user's terminal. The systems programmer will be delighted with the wide variety of disk access and file management routines available for personal use. Overall, FLEX is one of the most powerful operating systems available today.

The FLEX Operating System is comprised of three parts, the File Management System (FMS), the Disk Operating System (DOS), and the Utility Command Set (UCS). Part of the power of the overall system lies in the fact that the system can be greatly expanded by simply adding additional utility commands. The user should expect to see many more utilities available for FLEX in the future. Some of the other important features include: fully dynamic file space allocation, the automatic "removal" of defective sectors from the disk, automatic space compression and expansion on all text files, complete user environment control using the TTYSET utility command, and uniform disk wear due to the high performance dynamic space allocator.

The UCS currently contains many very useful commands. These programs reside on the system disk and are only loaded into memory when needed. This means that the set of commands can be easily extended at any time, without the necessity of replacing the entire operating system. The utilities provided with FLEX perform such tasks as the saving, loading, copying, renaming, deleting, appending, and listing of disk files. There is an extensive CATalog command for examining the disk's file directory. Several environment control commands are also provided. Overall, FLEX provides all of the necessary tools for the user's interaction with the disk.

^{*} FLEX is a registered trademark of Technical Systems Consultants, Inc.

II. SYSTEM REQUIREMENTS

FLEX requires random access memory from location 0000 through location 2FFF hex (12K). Memory is also required from C000 (48K) through DFFF hex (56K), where the actual operating system resides. The system also assumes at least 2 disk drives are connected to the controller and that they are configured as drives #0 and #1. You should consult the disk drive instructions for this information. FLEX interfaces with the disk controller through a section of driver routines and with the operator console or terminal through a section of terminal I/O routines.

III. GETTING THE SYSTEM STARTED

Each FLEX system diskette contains a binary loader for loading the operating system into RAM. There needs to be some way of getting the loader off of the disk so it can do its work. This can be done by either hand entering the bootstrap loader provided with the disk system, or by using the boot provided in ROM if appropriate to FLEX.

As a specific example, suppose the system we are using has SWTPc's S-BUG installed and we wish to run FLEX. The first step is to power on all equipment and make sure the S-BUG prompt is present (>). Next insert the system diskette into drive 0 (the boot must be performed with the disk in drive 0) and close the door on the drive. Type "D" on the terminal if using a full size floppy system or "U" if a minifloppy system. The disk motors should start, and after about 2 seconds, the following should be displayed on the terminal:

FLEX X.X DATE (MM,DD,YY)?

+++

The name FLEX identifies the operating system and the X.X will be the version number of the operating system. At this time the current date should be entered, such as 7,3,79. The FLEX prompt is the three plus signs (+++), and will always be present when the system is ready to accept an operator command. The '+++' should become a familiar sight and signifies that FLEX is ready to work for you!

IV. DISK FILES AND THEIR NAMES

All disk files are stored in the form of 'sectors' on the disk and in this version, each sector contains 256 'bytes' of information. Each byte can contain one character of text or one byte of binary machine information. A maximum of 340 user-accessible sectors will fit on a single-sided mini disk or 1140 sectors on a single-sided full size floppy. Double-sided disks would hold exactly twice that number of sectors. Double-density systems will hold more still. The user, however, need not keep count, for the system does this automatically. A file will always be at least one sector long and can have as many as the maximum number of sectors on the disk. The user should not be concerned with the actual placement of the files on the disk since this is done by the operating system. File deletion is also supported and all previously used sectors become immediately available again after a file has been deleted.

All files on the disk have a name. Names such as the following are typical:

PAYROLL INVNTORY TEST1234 APRIL-78 WKLY-PAY

Anytime a file is created, referenced, or deleted, its name must be used. Names can be most anything but must begin with a letter (not numbers or symbols) and be followed by at most 7 additional characters, called 'name characters'. These 'name characters' can be any combination of the letters 'A' through 'Z' or 'a' through 'z', any digit 'O' through '9', or one of the two special characters, the hyphen (-) or the underscore '_', (a left arrow on some terminals).

File names must also contain an 'extension'. The file extension further defines the file and usually indicates the type of information contained therein. Examples of extensions are: TXT for text type files, BIN for machine readable binary encoded files, CMD for utility command files, and BAS for BASIC source programs. Extensions may contain up to 3 'name characters' with the first character being a letter. Most of the FLEX commands assume a default extension on the file name and the user need not be concerned with the actual extension on the file. The user may at anytime assign new extensions, overiding the default value, and treat the extension as just part of the file name. Some examples of file names with their extensions follow:

APPEND.CMD LEDGER.BAS TEST.BIN

Note that the extension is always separated from the name by a period '.'. The period is the name 'field separator'. It tells FLEX to treat the following characters as a new field in the name specification.

A file name can be further refined. The name and extension uniquely define a file on a particular drive, but the same name may exist on several drives simultaneously. To designate a particular drive a 'drive number' is added to the file specification. It consists of a single digit (0-3) and is separated from the name by the field separator '.'. The drive number may appear either before the name or after it (after the extension if it is given). If the drive is not specified, the system will default to either the 'system' drive or the 'working' drive. These terms will be described a little later.

Some examples of file specifications with drive numbers follow:

O.BASIC MONDAY.2 1.TEST.BIN LIST.CMD.1

In summary, a file specification may contain up to three fields separated by the field separator. These fields are; 'drive', 'name', and 'extension'. The rules for the file specification can be stated quite concisely using the following notation:

[<drive>.]<name>[.<extension>]
 or
<name>[.<extension>][.<drive>]

The '<>' enclose a field and do not actually appear in the specification, and the '[]' surround optional items of the specification. The following are all syntactically correct:

O.NAME.EXT NAME.EXT.O NAME.EXT O.NAME NAME.O NAME

Note that the only required field is the actual 'name' itself and the other values will usually default to predetermined values. Studying the above examples will clarify the notation used. The same notation will occur regularly throughout the manual.

V. ENTERING COMMANDS

When FLEX is displaying '+++', the system is ready to accept a command line. A command line is usually a name followed by certain parameters depending on the command being executed. There is no 'RUN' command in FLEX. The first file name on a command line is always loaded into memory and execution is attempted. If no extension is given with the file name, 'CMD' is the default. If an extension is specified, the one entered is the one used. Some examples of commands and how they would look on the terminal follow:

+++TTYSET +++TTYSET.CMD +++LOOKUP.BIN

The first two lines are identical to FLEX since the first would default to an extension of CMD. The third line would load the binary file 'LOOKUP.BIN' into memory and, assuming the file contained a transfer address, the program would be executed. A transfer address tells the program loader where to start the program executing after it has been loaded. If you try to load and execute a program in the above manner and no transfer address is present, the message, 'NO LINK' will be output to the terminal, where 'link' refers to the transfer address. Some other error messages which can occur are 'WHAT?' if an illegal file specification has been typed as the first part of a command line, and 'NOT THERE' if the file typed does not exist on the disk.

During the typing of a command line, the system simply accepts all characters until a 'RETURN' key is typed. Any time before typing the RETURN key, the user may use one of two special characters to correct any mistyped characters. One of these characters is the 'back space' and allows deletion of the previously typed character. Typing two back spaces will delete the previous two characters. The back space is initially defined to be a 'control H' but may be redefined by the user using the TTYSET utility command. The second special character is the line 'delete' character. Typing this character will effectively delete all of the characters which have been typed on the current line. A new prompt will be output to the terminal, but instead of the usual '+++' prompt, to show the action of the delete character, the prompt will be '???'. Any time the delete character is used, the new prompt will be '???', and signifies that the last line typed did not get entered into the computer. The delete character is initially a 'control X' but may also be redefined using TTYSET.

As mentioned earlier, the first name on a command line is always interpreted as a command. Following the command is an optional list of names and parameters, depending on the particular command being entered. The fields of a command line must be separated by either a space or a comma. The general format of a command line is:

<command>[,<list of names and parameters>]

A comma is shown, but a space may be used. FLEX also allows several commands to be entered on one command line by use of the 'end of line' character. This character is initially a colon (':'), but may be user defined with the TTYSET utility. By ending a command with the end of line character, it is possible to follow it immediately with another command. FLEX will execute all commands on the line before returning with the '+++' prompt. An error in any of the command entries will cause the system to terminate operation of that command line and return with the prompt. Some examples of valid command lines follow:

+++CAT 1 +++CAT 1:ASN S=1 +++LIST LIBRARY:CAT 1:CAT 0

As many commands may be typed in one command line as desired, but the total number of characters typed must not exceed 128. Any excess characters will be ignored by FLEX.

One last system feature to be described is the idea of 'system' and 'working' drives. As stated earlier, if a file specification does not specifically designate a drive number, it will assume a default value. This default value will either be the current 'system' drive assignment or the current 'working' drive assignment. The system drive is the default for all command names, or in other words, all file names which are typed first on a command line. Any other file name on the command line will default to the working drive. This version of FLEX also supports automatic drive searching. When in the auto search mode if no drive numbers are specified, the operating system will first search drive 0 for the file. If the file is not found, drive 1 will be searched and so on. When the system is first initialized the auto drive searching mode will be selected. At this time, all drive defaults will be to drive O. It is sometimes convenient to assign drive 1 as the working drive in which case all file references, except commands, will automatically look on drive 1. It is then convenient to have a diskette in drive 0 with all the system utility commands on it (the 'system drive'), and a disk with the files being worked on in drive 1 (the 'working drive'). If the system drive is 0 and the working drive is 1, and the command line was:

+++LIST TEXTFILE

FLEX would go to drive 0 for the command LIST and to drive 1 for the file TEXTFILE. The actual assignment of drives is performed by the ASN utility. See its description for details.

VI. COMMAND DESCRIPTIONS

There are two types of commands in FLEX, memory resident (those which actually are part of the operating system) and disk utility commands (those commands which reside on the disk and are part of the UCS). There are only two resident commands, GET and MON. They will be described here while the UCS is described in the following sections.

GET

The GET command is used to load a binary file into memory. It is a special purpose command and is not often used. It has the following syntax:

GET[,<file name list>]

where <file name list> is: <file spec>[,<file spec>] etc.

Again the '[]' surround optional items. 'File spec' denotes a file name as described earlier. The action of the GET command is to load the file or files specified in the list into memory for later use. If no extension is provided in the file spec, BIN is assumed, in other words, BIN is the default extension. Examples:

GET, TEST GET, 1. TEST, TEST2.0

where the first example will load the file named 'TEST.BIN' from the assigned working drive, and the second example will load TEST.BIN from drive 1 and TEST.BIN from drive 0.

MON

MON is used to exit FLEX and return to the hardware monitor system such as S-BUG. The syntax for this command is simply MON followed by the 'RETURN' key.

NOTE: to re-enter FLEX after using the MON command, you should enter the program at location CD03 hex.

UTILITY COMMAND SET

The following pages describe all of the utility commands currently included in the UCS. You should note that the page numbers denote the first letter of the command name, as well as the number of the page for a particular command. For example, 'B.1.2' is the 2nd page of the description for the 1st utility name starting with the letter 'B'.

COMMON ERROR MESSAGES

Several error messages are common to many of the FLEX utility commands. These error messages and their meanings include the following:

NO SUCH FILE. This message indicates that a file referenced in a particular command was not found on the disk specified. Usually the wrong drive was specified (or defaulted), or a misspelling of the name was made.

ILLEGAL FILE NAME. This can happen if the name or extension did not start with a letter, or the name or extension field was too long (limited to 8 and 3 respectively). This message may also mean that the command being executed expected a file name to follow and one was not provided.

FILE EXISTS. This message will be output if you try to create a file with a name the same as one which currently exists on the same disk. Two different files with the same name are not allowed to exist on the same disk.

SYNTAX ERROR. This means that the command line just typed does not follow the rules stated for the particular command used. Refer to the individual command descriptions for syntax rules.

GENERAL SYSTEM FEATURES

Any time one of the utility commands is sending output to the terminal, it may be temporarily halted by typing the 'escape' character (see TTYSET for the definition of this character). Once the output is stopped, the user has two choices: typing the 'escape' character again or typing 'RETURN'. If the 'escape' character is typed again, the output will resume. If the 'RETURN' is typed, control will return to FLEX and the command will be terminated. All other characters are ignored while output is stopped.

\dot{I}
$-$ 1 \sim 2
$rac{1}{1}$

APPEND

The APPEND command is used to append or concatenate two or more files, creating a new file as the result. Any type of file may be appended but it only makes sense to append files of the same type in most cases. If appending binary files which have transfer addresses associated with them, the transfer address of the last file of the list will be the effective transfer address of the resultant file. All of the original files will be left intact.

DESCRIPTION

The general syntax for the APPEND command is as follows:

APPEND, <file spec>[, <file list>], <file spec>

where <file list> can be an optional list of the specifications. The last name specified should not exist on the disk since this will be the name of the resultant file. If the last file name given does exist on the disk, the question "MAY THE EXISTING FILE BE DELETED?" will be displayed. A Y response will delete the current file and cause the APPEND operation to be completed. A N response will terminate the APPEND operation. All other files specified must exist since they are the ones to be appended together. If only 2 file names are given, the first file will be copied to the second file. The extension default is TXT unless a different extension is used on the FIRST FILE SPECIFIED, in which case that extension becomes the default for the rest of the command line. Some examples will show its use:

APPEND, CHAPTER1, CHAPTER2, CHAPTER3, BOOK APPEND, FILE1, 1. FILE2. BAK, GOODFILE

The first line would create a file on the working drive called 'BOOK.TXT' which would contain the files 'CHAPTER1.TXT', CHAPTER2.TXT', and 'CHAPTER3.TXT' in that order. The second example would append 'FILE2.BAK' from drive 1 to FILE1.TXT from the working drive and put the result in a file called 'GOODFILE.TXT' on the working drive. The file GOODFILE defaults to the extension of TXT since it is the default extension. Again, after the use of the APPEND command, all of the original files will be intact, exactly as they were before the APPEND operation.

The ASN command is used for assigning the 'system' drive and the 'working' drive or to select automatic drive searching. The system drive is used by FLEX as the default for command names or, in general, the first name on a command line. The working drive is used by FLEX as the default on all other file specifications within a command line. Upon initialization, FLEX assigns drive #0 as both the system and working drive. An example will show how the system defaults to these values:

APPEND, FILE1, FILE2, FILE3

If the system drive is assigned to be #0 and the working drive is assigned to drive #1, the above example will perform the following operation: get the APPEND command from drive #0 (the system drive), then append FILE2 from drive #1 (the working drive) to FILE1 from drive #1 and put the result in FILE3 on drive #1. As can be seen, the system drive was the default for APPEND where the working drive was the default for all other file specs listed.

Automatic drive searching causes FLEX to automatically scan the ready drives for the file specified. Hardware limitations prevent the mini floppy versions from searching for "ready" drives. For this reason, FLEX has been setup to ALWAYS assume drive 0 and 1 are ready. Thus if a mini floppy version of FLEX attempts to search a drive which does not have a disk loaded, it will hang up until a disk is inserted and the door closed. Alternatively, the system reset could be hit and a warm start executed (a jump to address \$CDO3). The full size floppy version CAN detect a ready condition and will not check drives which are out of the ready state during automatic drive searching.

Automatic drive searching causes FLEX to first check drive #0 for the file specified. If not there (or if not ready in the full size version), FLEX skips to drive #1. If the file is not found on drive #1 in the mini floppy version, FLEX gives up and a file not found error results. In the full size version FLEX continues to search on drives #2 and #3 before reporting an error.

DESCRIPTION

The general syntax for the ASN command is as follows:

ASN[,W=<drive>][,S=<drive>]

where <drive> is a single digit drive number or the letter A. If just ASN is typed followed by a 'RETURN', no values will be changed, but the system will output a message which tells the current assignments of the system and working drives, for example:

+++ASN
THE SYSTEM DRIVE IS #0
THE WORKING DRIVE IS #0

FLEX User's Manual

Some examples of using the ASN command are:

ASN, W=1 ASN, S=1, W=0

where the first line would set the working drive to 1 and leave the system drive assigned to its previous value. The second example sets the system drive to 1 and the working drive to 0. Careful use of drive assignments can allow the operator to avoid the use of drive numbers on file specifications most of the time!

If auto drive searching is desired, then the letter A for automatic, should be used in place of the drive number.

Example: ASN W=A ASN S=A, W=1 ASN S=A, W=A

V.	
1	
1	
`	1
	1
	1
	1
	1
	1
	1
	i
	1
	į.
	1
	1

The BUILD command is provided for those desiring to create small text files quickly (such as STARTUP files, see STARTUP) or not wishing to use the optionally available FLEX Text Editing System. The main purpose for BUILD is to generate short text files for use by either the EXEC command or the STARTUP facility provided in FLEX.

DESCRIPTION

The general syntax of the BUILD command is:

BUILD, <file spec>

where <file spec> is the name of the file you wish to be created. The default extension for the spec is TXT and the drive defaults to the working drive. If the output file already exists the question "MAY THE EXISTING FILE BE DELETED?" will be displayed. A Y response will delete the existing file and build a new file while a N response will terminate the BUILD command.

After you are in the 'BUILD' mode, the terminal will respond with an equals sign ('=') as the prompt character. This is similar to the Text Editing System's prompt for text input. To enter your text, simply type on the terminal the desired characters, keeping in mind that once the 'RETURN' is typed, the line is in the file and can not be changed. Any time before the 'RETURN' is typed, the backspace character may be used as well as the line delete character. If the delete character is used, the prompt will be '???' instead of the equals sign to show that the last line was deleted and not entered into the file. It should be noted that only printable characters (not control characters) may be entered into text files using the BUILD command.

To exit the BUILD mode, it is necessary to type a pound sign ('#') immediately following the prompt, then type 'RETURN'. The file will be finished and control returned back to FLEX where the three plus signs should again be output to the terminal. This exiting is similar to that of the Text Editing System.

en de la companya de La companya de la co

The BACKUP command is used to copy an entire diskette quickly. It copies all the information on a diskette to another diskette. The two diskettes must be the same size and format: each sector on the source diskette is copied to the corresponding sector on the destination diskette. The previous contents of the destination disk are lost. The copying process is exact: files that were segmented on the source diskette will be segmented the same way on the destination diskette. Each sector on the destination disk is read back for verification. BACKUP works properly only with diskettes formatted with the Gimix FORMAT program.

DESCRIPTION

The general syntax of the BACKUP command is:

BACKUP, (source drive #>, (destination drive #>[, CPU speed]

where (source drive #> is the drive holding the diskette to be backed up, (destination drive #> is the drive holding the diskette to be backed up to, and CPU speed is a "1" or "2" indicating the CPU clock rate assumed when the diskettes were formatted. (This is necessary because the Gimix FORMAT program has different interleave patterns for 1 Mhz and 2 Mhz CPUs. BACKUP uses the interleave pattern to read and write physically sequential sectors on the diskettes, so that an entire track can be read or written in one revolution.) This parameter defaults to 2. For example,

+++BACKUP, 0, 1

would copy the diskette in drive 0 to the diskette in drive 1, assuming a 2 Mhz type interleave. If the diskettes have different interleave patterns or the wrong pattern is indicated in the command line, BACKUP will take up to 10 times as long to run. This may also happen if either diskette has a non-Gimix format. NOTE: the actual clock speed of the CPU is irrelevant; BACKUP is only concerned with the order of the sectors as set by FORMAT.

Before the copying begins, BACKUP prints the name of the destination disk in the following prompt:

OKAY TO SCRATCH diskname.ext?

This is the last chance to abort BACKUP. The user must respond by typing "Y" or "N". If "N" is typed, BACKUP is aborted and control returns to FLEX. If "Y" is typed, BACKUP will proceed. When BACKUP is done it sends three BELL characters to the console and prints "BACKUP COMPLETE!".

BACKUP will work with 5" or 8" drives, single or double density, and single or double sided formats. The number of tracks is obtained from the System Information Record of the source diskette; any number is permitted provided both diskettes are the same. For more information on different disk formats see the FORMAT command.

는 경기에 가는 사람들이 하는 것이 되었다. 그 사람들은 사람들은 사람들은 사람들은 사람들이 되었다. 2013년 - 1915년 - 1915년 - 1917년	
	사람들은 경험 전문 경험을 보면 하는 것이 되었다. 그런 그 사람들은 사람들은 경험을 받는 것이 되었다. 사람들은 경기 대한 경험 기자를 하고 있다. 경험 등을 보고 있다는 것이 되었다면 하는 것이 되었다.
하셨잖아면 얼마 보이는 그는 그 그리고 되는 사람이 되었다는 그 모두 모든 일이다.	
(현용) (1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1	
회사 가는 사람 이번에 가게 들어 가게면 되는 것이다고 하다.	이 사용을 들어들어 하는데, 그래요요. 등을하는데 그
경화의 그는 그리는 살아들이 들어가지 않는 것이다. 학생들은 살아들은 살아들다.	
	회사회 교회에 있는 아이를 가고 있는 것 같다.
	가는 보는 사람이들 하나는 그를 만든 그는 것은 그릇을 받는다.
사용하다 보다는 사람들이 되었다. 그 사람이 있다면 보다는 사람들이 되었다. 	
중화 일요 있는 하는 그 사람들이 그 작품이 그렇지 않는데 하는 것을	
	이 하는 모양이 살아지고 이번째 보이는 그 모든
경기 (1995년 - 1일 1997년 - 1일 1997년 2월 1일	
등이 가능하다는 경기에 있는 것이 되었다. 그 사람들은 사람들이 되었다. 그 사람들은 사람들이 되었다. 1980년 - 1980년	
요즘 하기 되는 것이 되었다. 그렇게 되고 한 것이 보는 생각이 되는 것이 되었다. 그들 전에 가장 모든 것이 되었다. 생물님이 되는 사람들이 있다면 보고 한 것이 되는 것이 되었다. 사람들이 되는 것이 없는 것이 되었다.	
에 가는 이렇게 되었다. 이렇게 되었다. 이렇게 되었다. 그런 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은	

BACKUP is much faster than COPY when a large number of files are copied. The approximate time required for BACKUP is given in the following table. These times assume a 56K system. Note that data density does not affect the time required.

	8" 77-track	5" 80-track	5" 40-track
Single-sided	47 sec	82 sec	45 sec
Double-sided	106 sec	130 sec	66 sec

BACKUP checks a number of conditions before executing, and will abort if necessary. When this happens "BACKUP ABORTED" is printed, followed by one of the following error messages.

PRINT SPOOLER ACTIVE - The BACKUP command cannot be used while the print spooler is active. The user must de-activate the spooler with the 'QCHECK' command or wait until all printing has finished.

INVALID DRIVE NUMBER - The user entered an illegal drive number for the source or destination drive.

HARD DISK NOT ALLOWED - The user attempted to back up to or from a hard disk device.

DEST DISK IS PROTECTED - The destination disk is write protected, either by hardware or software. This message will be generated if the destination disk is a 96-tpi drive emulating a 48-tpi drive, since Gimix FLEX 4.0 automatically write protects such a drive.

DISKS ARE DIFFERENT SIZES - The user has attempted to back up a 5" diskette to an 8" diskette, or vice versa.

DISKS HAVE DIFFERENT FORMATS - The number of sectors per track or the number of tracks is different on the source and destination diskettes. Usually this means one diskette is double density or double sided and the other is not.

ILLEGAL DISK FORMAT - The number of sectors per track on either the source or destination diskette does not match any Gimix FLEX format. Only Gimix-formatted disks work properly with BACKUP.

ILLEGAL CLOCK RATE - The user specified a clock rate in the command line other than 1 or 2 (Mhz).

NOT ENOUGH MEMORY - The system does not have enough user memory to store one full track (13K for DS/DD 8" diskettes, less for other formats).

클러 (1984년) 전 (1985년 왕 1985년) 소프라스 왕이는 (1985년) 원리 (1985년) 전 (1985년) 원리 (1985년) 원리 (1985년) 원리 (1985년) 원리	
게 발표하는 하는 것도 하는 것으로 가장하는 것으로 보는 것으로 가장하는 것으로 가장 되었다. 그런 것으로 가장 되었다. 	
그림으로 하는 이 경우를 하는 것이 되었다. 그는 그림은 살림은 얼마나 그는 그를 가고 있다.	
고 발표하는 것이다. 설립부터 전기 기업 전기 전기 기업	
(1985) 1985 - 198	
물통을 모고 하는 지하는 그 에 보이지 않는 데 이 이번에 되지만 그는 사람들이 모르게 되었다.	
경영화 등 보는 사람들이 되는 것이 되는 것이 되면 되었다. 그는 사람들이 되는 것이 되는 것이 되는 것이 되었다. 그는 것이 되었다. 그는 것이 없는 것이 없는 것이다. 생용하는 것이 되는 것이 되었다. 그는 것	
[생물] 발생 보고 있는 것이 되었다. 그는 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은	
불림을 잃었다. 그는 이 보면에게 이 보면 하지만 하는 것이 되고 있는 그를 하는 것이 되었다. 이 모양	
용하하다는 사람은 어디다고 한 사람들은 이번 보는 사람들이 보고 있다. 그러나 보다 보다 보다 있다고	
그 경쟁 보고 그는 가는 사람이 되면 가장 맛있었다. 그 가는 이 가는 데이 얼마나 다른 사람이 없다.	
발표를 통하는데 그의 문에게 하나 되었다. 그는 사람들은 그리는 그들은 그를 다 하는데 되었다.	
성도 하고 있다. 그는 그는 그는 그 그는 그는 그는 그는 그는 그는 그는 그는 그를 모르는 것이 되었다. 그는 그는 그는 그를 모르는 그를	
나는 사람들은 마음을 하는 것이 되었다. 그는 사람들은 모든 사람들은 사람들이 되었다는 것이 되었다. 그들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람	
보다 보고 있는 것이 되었다. 그는 사람들은 사람들이 되는 것이 되었습니다. 그는 사람들이 되었다. 그는 사람들이 되었습니다. 그는 사람들이 되었습니다. 1900년 - 1일 전문에 1500년 1일	
하는 사용이 보고 있는 것이 되었다. 전혀 하면 모든 시간에 되고 있는 것으로 하는데 있는 것이다. 그는 것이라면 하는데 모든데 되었다. 그렇게 되었다. 그렇게 되었다. 그렇게 되었다. 그렇게 되었 [1942년 - 1945년 1945년 1947년	

The CATalog command is used to display the FLEX disk file names in the directory on each disk. The user may display selected files on one or multiple drives if desired.

DESCRIPTION

The general syntax of the CAT command is:

```
CAT[, <drive list>][, <match list>]
```

where <drive list> can be one or more drive numbers seperated by commas, and <match list> is a set of name and extension characters to be matched against names in the directory. For example, if only file names which started with the characters 'VE' were to be cataloged, then VE would be in the match list. If only files whose extensions were 'TXT' were to be cataloged, then .TXT should appear in the match list. A few specific examples will help clarify the syntax:

```
+++CAT
+++CAT,1,A.T,DR
+++CAT,PR
+++CAT,0,1
+++CAT,0,1..CMD..SYS
```

The first example will catalog all file names on the working drive or on all drives if auto drive searching is selected. The second example will catalog only those files on drive 1 whose names begin with 'A' and whose extensions begin with 'T', and also all files on drive 1 whose names start with 'DR'. The next example will catalog all files on the working drive (or on all drive if auto drive searching is selected) whose names start with 'PR'. The next line causes all files on both drive 0 and drive 1 to be cataloged. Finally, the last example will catalog the files on drive 0 and 1 whose extensions are CMD or SYS.

During the catalog operation, before each drive's files are displayed, a header message stating the drive number is output to the terminal. The name of the diskette as entered during the NEWDISK operation will also be displayed. The actual directory entries are listed in the following form:

NAME. EXTENSION SIZE PROTECTION CODE

where size is the number of sectors that file occupies on the disk. If more than one set of matching characters was specified on the command line, each set of names will be grouped according to the characters they match. For example, if all .TXT and .CMD files were cataloged, the TXT types would be listed together, followed by the CMD types.

In summary, if the CAT command is not parameterized, then all files on the assigned working drive will be displayed. If a working drive is not assigned (auto drive searching mode) the CAT command will display files

water materials of the could be

on all on line drives. If it is parameterized by only a drive number, then all files on that drive will be displayed. If the CAT command is parameterized by only an extension, then only files with that extension will be displayed. If only the name is used, then only files which start with that name will be displayed. If the CAT command is parameterized by only name and extension, then only files of that root name and root extension (on the working drive) will be displayed. Learn to use the CAT command and all of its features and your work with the disk will become a little easier.

The current protection code options that can be displayed are as follows:

D File is delete protected (delete or rename prohibited)
W File is write protected (delete, rename and write prohibited)
(blank) No special protection

The COPY command is used for making copies of files on a disk. Individual files may be copied, groups of name-similar files may be copied, or entire disks may be copied. The copy command is a very versatile utility. The COPY command also re-groups the sectors of a file in case they were spread all over the old disk. This regrouping can make file access times much faster. It should be noted that before copying files to a new disk, the disk must be formatted first. Refer to NEWDISK for instructions on this procedure.

DESCRIPTION

The general syntax of the COPY command has three forms:

- a. COPY, <file spec>, <file spec>
- b. COPY,<file spec>,<drive>
- c. COPY, \(drive \), \((drive \) [, \((match list \)]

where <match list> is the same as that described in the CAT command and all rules apply to matching names and extensions. When copying files, if the destination disk already contains a file with the same name as the one being copied, the file name and the message, "FILE EXISTS DELETE ORIGINAL?" will be output to the terminal. Typing Y will cause the file on the destination disk to be deleted and the file from the source disk will be copied to the destination disk. Typing N will direct FLEX not to copy the file in question.

The first type of COPY allows copying a single file into another. The output file may be on a different drive but if on the same drive the file names must be different. It is always necessary to specify the extension of the input file but the output file's extension will default to that of the input's if none is specified. An example of this form of COPY is:

+++COPY, O. TEST. TXT, 1. TEST25

This command line would cause the file TEST.TXT on drive 0 to be copied into a file called TEST25.TXT on drive 1. Note how the second file's extension defaulted to TXT, the extension of the input file.

The second type of COPY allows copying a file from one drive to another drive with the file keeping its original name. An example of this is:

+++COPY, O.LIST. CMD, 1

Here the file named LIST.CMD on drive 0 would be copied to drive 1. It is again necessary to specify the file's extension in the file specification. This form of the command is more convenient than the previous form if the file is to retain its original name after the copying process.

The final form of COPY is the most versatile and the most powerful. It is possible to copy all files from one drive to another, or to copy only those files which match the match list characters given. Some examples will clarify its use:

```
+++COPY,0,1
+++COPY,1,0,.CMD,.SYS
+++COPY,0,1,A,B,CA.T
```

The first example will copy all files from drive 0 to drive 1 keeping the same names in the process. The second example will copy only those files on drive 1 whose extensions are CMD and SYS to drive 0. No other files will be copied. The last example will copy the files from drive 0 whose names start with 'A' or 'B' regardless of extension, and those files whose names start with the letters 'CA' and whose extensions start with 'T'., to the output drive which is drive 1. The last form of copy is the most versatile because it will allow putting just the command (CMD) files on a new disk, or just the SYS files, etc., with a single command entry. During the COPY process, the name of the file which is currently being copied will be output to the terminal, as well as the drive to which it is being copied.

The COPY command is used for making copies of files on a disk. Individual files may be copied, groups of name-similar files may be copied, or entire disks may be copied. The COPY command is a very versatile utility. When files are copied onto a newly formatted disk, they are stored as contiguous groups of sectors, resulting in minimum access times. This can be a substantial improvement over an old disk on which the files are highly fragmented due to frequent rewriting.

DESCRIPTION

The general syntax of the COPY command has three forms:

- a. COPY, <file spec>, <file spec>
- b. COPY, <file spec>, <drive>
- c. COPY, \(\drive\), \(\drive\)[, \(\match,\list\)]

where (match list) is the same as that described in the CAT command and all rules apply to matching names and extensions. When files are copied, if the destination disk has a file with the same name as the file being copied, the file name and the message "FILE EXISTS - DELETE ORIGINAL?" will be displayed on the console. Typing "Y" will cause the file on the destination disk to be deleted and the file on the source disk will be copied to the destination disk. Typing "N" will direct FLEX not to copy the file in question.

The first type of COPY allows copying of a single file into another. The output file may be on a different drive, but if it is on the same drive then the file names must be different. It is always necessary to specify the input file's extension, but the output file's extension will default to that of the input file if none is specified. Example:

+++COPY, O. TEST. TXT, 1. TEST25

This command line would cause the file TEST.TXT on drive 0 to be copied to a file named TEST25.TXT on drive 1. Note that the destination file's extension defaulted to TXT, the same as the input file.

The second type of COPY allows copying a file from one drive to another with the file name unchanged. Example:

+++COPY, O.LIST.CMD, 1

Here the file named LIST.CMD on drive 0 would be copied to drive 1. It is again necessary to specify the file's extension in the file specification. This form of the command is more convenient than the first if the copied file is to have the same name.

	# 100 Head Head Total Policy (1984) - 1984 - 1985 - 1984 - 1985 - 1984 - 1985 - 1984 - 1985 - 1985 - 1985 - 19
한다면 하는 이 나는 마음이를 하다면 하다. 이 나왔는 아름다면 하는 사람들은 나는 나를 하면 하다면 살다면 하는 것이다.	
이 나도는 말이 되었는데 이 아무리 맛이 얼마나 하고 하다 나를 하셨다면 살았다.	
홍취보다는 전에서, 고기에 보이 이 회장이 하면 모양 하는데 되었다.	
그 유통하다 그 보다 그 그 그 그 그는 그 가는 것이 되는 것이 없는 것이 없다.	
중취가 하시고 하는 일으로 보네요? 이 사고를 모르는 일은 사는 것이 모르다.	
가는 사용하는 사용하는 것으로 가장 보고 있다. 그는 사람들은 사용하는 사용하는 것으로 가장 되었다. - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1	
	강하는 항 나고 말이 하는데 하는 사람들은 사람들이 다
결혼 발표됐다고 하다. 이 이 의 그들은 이 이 그는 이 사람이 되는 것 같다.	
<u> </u>	
불충분통령이 하다 그 그 그리다 하나 나는 어때 얼마나 다니다.	
마음 사용 등 경기 등 보고 있는데 그 등 이 가는 것이 되었다. 그는 그 이 가는 것이 되었다. 하는 사용 하는 것이 되었다. 그는 것이 되었다는데 그 것이 되었다. 그 것이 되었다. 당하는 사용 하는 것이 되었다. 그는 것이 되었다. 그 것이 되었다. 그 것이 되었다. 그 것이 되었다.	

The final form of the COPY command is the most versatile and the most powerful. With this form, it is possible to copy all the files on one drive to another drive, or only those files which match one of the patterns in the match list. Examples:

+++COPY,0,1 +++COPY,1,0,.CMD,.SYS +++COPY,0,1,A,B,CA.T

The first example would copy all the files on drive 0 to drive 1. The second example would copy all CMD and SYS files on drive 1 to drive 0. The third example would copy from drive 0 to drive 1 all files beginning with the letter A or the letter B, or beginning with the letters CA and with an extension beginning with the letter T. This form of the COPY command is the most versatile because it allows a set of files to be extracted from a disk. The file name is always preserved with this form. During execution, the name of each file copied is displayed on the console along with the drive to which it is copied.

The match list is processed as follows: for each partial file specification in the list, all the entries in the catalog of the source disk are tested and those that match are copied. Then the whole catalog is scanned again for matches to the next specification in the list. Thus all the files which match a given specification will be grouped together in the catalog of the output disk. If a file matches more than one specification in the list, then COPY will try to copy it as many times as it matches. Example:

+++COPY, 1, 2, ABC, .TXT

would copy the file ABC.TXT twice. The second time would generate the "FILE EXISTS - DELETE ORIGINAL?" prompt.

† Two versions of COPY are supplied with GIMIX FLEX 4.x. Except for the manner in which the file creation date is handled they are functionally identical. Use the RENAME utility to change the name of the preferred version to COPY.CMD.

COPY-TSC creates its output file through the normal FMS file creation function. Therefore the creation date of the output file is the current system date. This is the standard version of COPY normally supplied with FLEX.

COPY-GMX has been modified by GIMIX so that the creation date of the output file will be the same as that of the input file. For all files except random-access files, this date is the last date on which the file's contents were altered, and is often very useful to know. This version of COPY allows all copies of a file with the same contents to have the same date.

그 있는데 그는 이 문에서 그렇게 하지 않아, 물로 이 없는 물이 되어 되어 되었다. 물로 취약 사람들은 이 제대를 가게 되어 되었다. 그리는 이 경험이 되었다. 물로 들어올 수 있다면 하다.	
가는 되는 것이 하는 것도 없는 것이 되었다. 생각이 하는 것이 되었다. 그런 것이 되었다. 그런 사람들은 것이 되었다. 그런 사람들은 그는 것이 되었다. 그는 것이 없는 것이 없는 것이 없는 것이 그렇게 되는 것이 되는 것이 있다. 그런 것이 있다면 하는 것이 되었다. 그런 것이 되었다. 그런 것이 없는 것이 없는 것이 되었다. 그런 것이 없는 것이 없는 것이 없는 것이 없는 것이 없는 것이	
으로 있는 것이 되었다. 그런 사람들은 사람들은 사람들은 사람들은 사람들은 사람들이 되었다. 그런 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은	
마음하다. 그리고 있는 사람이 되는 것이 있다면 모든 사람들이 되었다. 그리고 있는 것이 되었다는 것이 되었다. 그리고 있는 것이 되었다면 보다 이 그리고 있다. 	
. 보고 있는 이 전문에 되고 보고 이 보장 문항들이 되는 것이 되었다. 그 사람들은 이 보고 하는데 하는데 되었다. 그 사람들이 되었다. 그 생물 전에 보고 있는데 말로 보고 있는데 보고 있는데 이 보고 있는데 말로 보고 있다. 그런데 말로 보고 있는데 보고 있는데 보고 있다.	
일본하다 그런 이 그림에도 하면 그렇게 되었다. 그는 그들은 그를 다 그리고 있다면 하는 것이다.	
경기를 받는 것이 되었다. 그는 사람들은 사람들은 사람들은 사람들은 사람들은 사람들이 되었다. 그는 사람들은 사람들은 사람들은 사람들이 되었다. 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은	
이 있었다. 그런 이번 나는 아들이 살아 있다면 하는데 그들은 이 얼마를 하는데	
생활들은 하시면 하는 사람들에게 보면서 보고 있는데 하시면 되고 보면 하는 사람들이 되었다. 그는 사람들이 되었다. 그 사람들은 다른 이 보고 있다. - 생기 하나는 사람들은 사람들은 사람들이 되었다. 그 사람들이 되었다. 그 사람들은 사람들은 사람들은 사람들이 되었다. 그 사람들이 되었다.	
하면 회사들이 있었습니다. 그 모든 전에 가장하는 그는 이 가장 것을 하는데 하는데 하는데 하다.	
그들 그 아이들 그들은 아이들이 가장 그들이 그들이 보는데 어떤 생생들이 되는데 이번 이렇게 하는 것	
현실이 있는 것이 되고 있다. 이 전에 가장 되었다. 그는 사이에 가장 되는 것이 다음이 있다는 것이 되었다. 그는 것이 없는 것이 없는 것이 없는 것이다. 생물을 하는 것이 되었다. 그는 것이 없는 것이다. 생물을 하는 것이 없는 것	

The CLEAN command has been provided to enable the user to use head cleaning diskettes. All it does is step the head in and out to insure uniform cleaning.

DESCRIPTION

The general syntax of the CLEAN command is:

CLEAN

CLEAN takes no command line parameters. It will prompt the user for the information that is needs.

To use the clean command merely type the following:

+++CLEAN

Clean will then prompt:

NUMBER OF TRACKS TO STEP?

Enter the maximum number of tracks for the drive to be cleaned as found in the manufacturers literature. Though less then the maximum number of tracks may be specified, it is recommended that only the maximum number be used. This is to insure uniform head cleaning and uniform wear on the head cleaning diskette. Entering an illegal number or zero will cause a return to FLEX. The next prompt is:

NUMBER OF DRIVE TO BE CLEANED?

Enter the drive number for the drive to be cleaned. Entering an illegal number or an escape will cause a return to FLEX. The last prompt is:

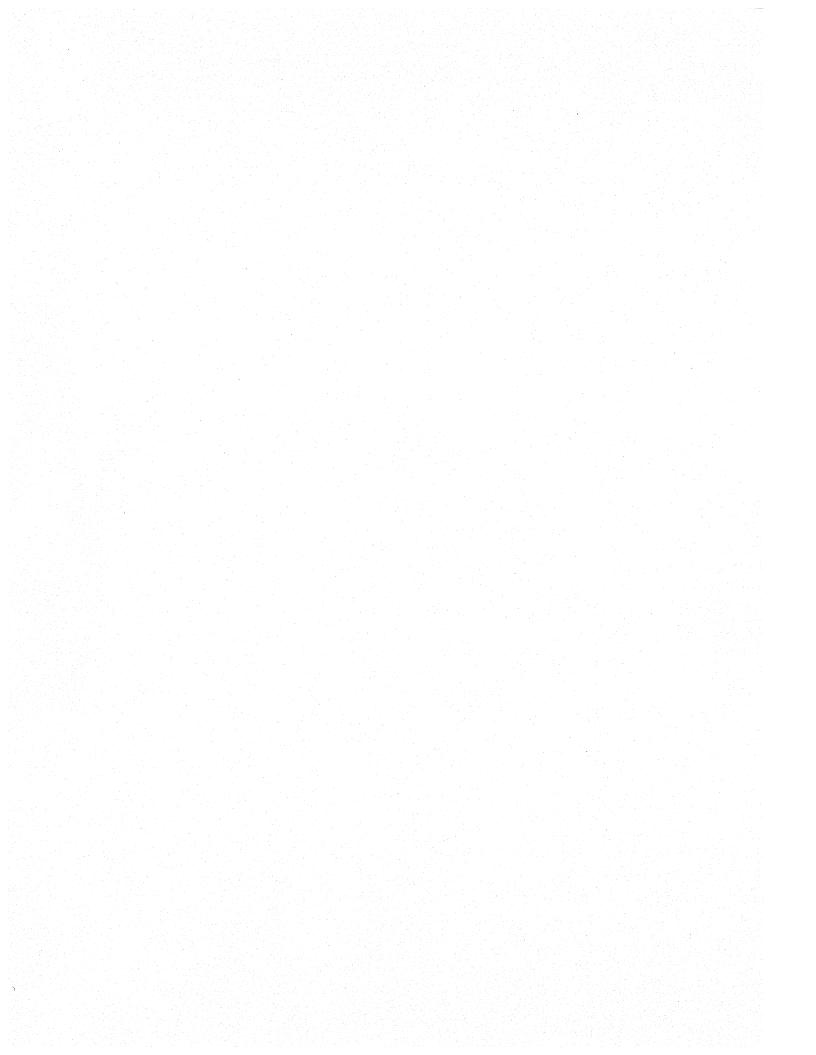
PUT CLEANING DISK IN DRIVE AN HIT 'CR' TO CLEAN THE HEAD(S)?

At this point, following the instructions that acompany the cleaning diskette. Insert the cleaning diskette in the specified drive and close the door. Then type a carriage return on the keyboard to start the cleaning process. Typing an escape will cause a return to FLEX. Typing any other character will cause the prompt to be re-printed. When finished CLEAN will print:

DONE.

And ring the terminal's bell.

NOTE: Failure to follow the manufacturers instructions can cause damage to the disk drive and/or the cleaning diskette.



CHECKSUM

The CHECKSUM command performs a 32 bit checksum on an entire disk. The program reads every sector on the disk and totals them together. This can be used to verify disk copies, check disk validity, etc.

DESCRIPTION

The general synatax of the CHECKSUM command is:

CHECKSUM[,dn]

Where 'dn' is an optional drive number. If no drive is specified CHECKSUM will use the work drive. If the work drive is set to 'ALL' an error message is printed. Some examples follow:

+++CHECKSUM +++CHECKSUM, 2

The first example will generate a CHECKSUM of the disk in the current work drive, assuming the work drive in not set to 'ALL'. The second example will generate a CHECKSUM of the disk in drive 2. The output of CHECKSUM will look like:

CHECKSUM: 0002AB02

CHECKSUM can generate the following error messages:

ILLEGAL DRIVE NUMBER

Legal drive numbers are 0, 1, 2, or 3. A drive number must be specified if the work drive is set to ALL.

INVALID DISK FORMAT

The disk uses a non-standard format or the SYSTEM INFORMATION RECORD sector may be damaged.

소리와 이번 경험을 가는 이번 수 있는 이번 보고 있는데, 이번 이번 이번 사람들은 경험을 받는데 되는데 되었다. 그런데 보고 있는데 보다 되었다. 보고 있는데 이번 기를 보고 있는데 이번 이번 기를 보고 있는데 이번 이번 기를 보고 있는데 이번 기를 보고 있다.	
위에 가장 하고 만큼 되다는 사람들은 사람들이 살아 가장 하지만 하는 것이 가장 하는 것이 되었다. 사람들이 가장 바람들이 되었다.	
그들의 사람들이 남자들이다. 그리다 그 마음이 얼마나 그 사람들이 하는 사람이 되었다. 그 사람들이 다른 사람들이 되었다.	
보고 생활하는 그는 그 이 사람이 그림이 들어 보고 있다. 그리고 그렇게 하는 얼굴을 하는 것이 되었다.	
學說 생생님, 그 이 하는 이 이 가는 사람들은 사람들은 사람들이 되었다. 그는 사람들은 사람들은 사람들은 사람들이 되었다.	
建氯磺胺 경험에 가는 사람이 있는 사람들은 사람들은 사람들은 사람들이 가는 사람들이 되었다. 그 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은	
· 보통하는 보다 가장 하는 사람들이 보다는 보다가 하는 사람들이 된다면 하는 보다는 사람들이 되었다.	
化基础하다 그는 이 시간에 가는 그리고 하고 있다. 그는 이 이 이 그 사람들은 이 그를 하고 있는데 하는 사람이 되었다. 그는	
그는 한 생님 방향에 가장 한 한 한 일반이 되었다. 그 경험 이번 함께 나는 사람들이 가장 하나 있다는 것은 사람들이 다고	
눈선들 어느 눈은 하는 것 같아 들어 가게 되는 것도 하는 것이다. 그는 말이 되는 것이 되어 되는 것은 것이다. 그는 그는 그를 다는 것이다.	
2000년 1일	
[5] [12] [14] [14] [15] [15] [15] [15] [15] [15] [15] [15	
[15] 전환, [16] 전 보다는 이 보고 있는데 보다 보는 사람이 되었다. 이 그는 사람들은 보고 있는데 하는데 되었다. 그 보다 하는데 보다 다른데 보다 되었다.	
경영선 유명하면서 이 그들은 마음이 아니는 이름이 되는 것이 되었다. 그는 그는 그는 그는 그는 그를 가지고 있다.	

CMPBIN

The CMPBIN command is used to compare the contents of two binary files and list the differences. CMPBIN is a useful tool for sorting out mislabeled or long-forgotten binary files, for tracking changes in programs, and for identifying current versions.

DESCRIPTION

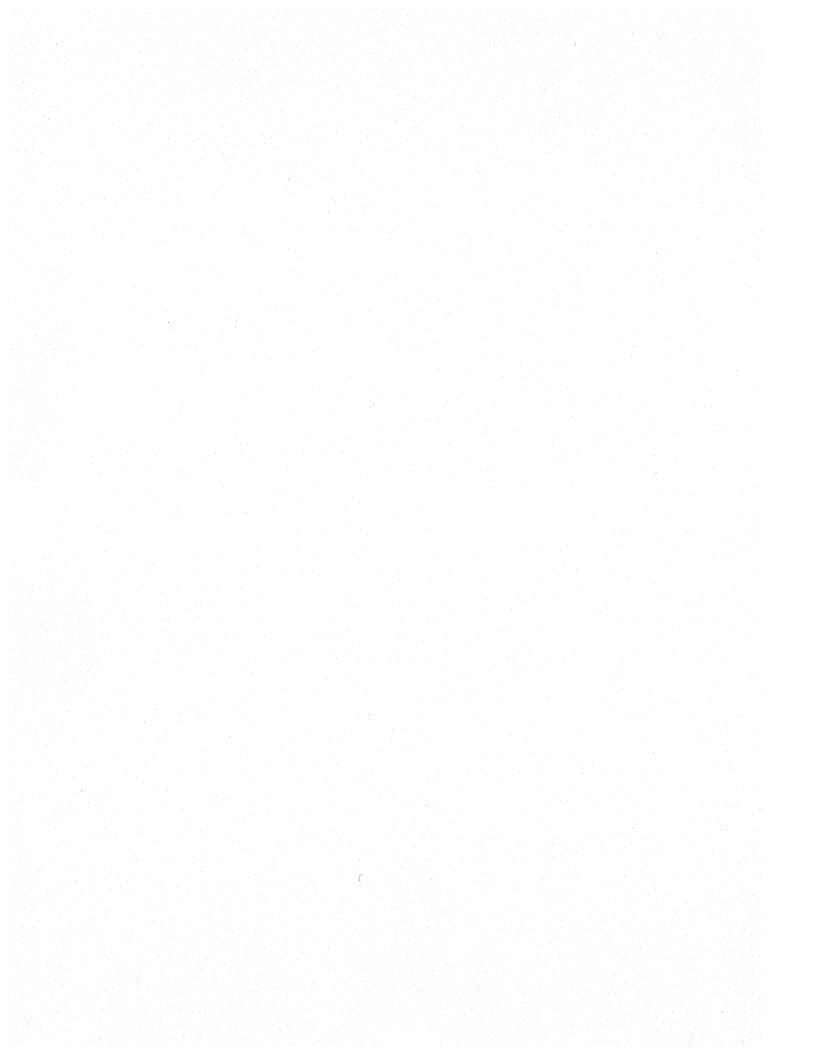
The general syntax of the CMPBIN command is:

CMPBIN, <file spec>, <file spec>

This will cause the two files to be read as FLEX binary files and compared. The default extension is BIN. The files are read as binary records, in the format described on page 45 of the FLEX Advanced Programmer's Guide. A binary record consists of a load address, a byte count, and bytes of data to be stored in memory. The data bytes from the first file are compared to the data bytes from the second file. The current load address for each file is also compared. If either is different, the address and data byte from each file is printed. Example:

+++CMPBI	N.A.BI	IN. AOLD. I	BIN
FILE			LE B
ADDRESS	BYTE	BYTE	ADDRESS
0209	A5	A7	0209
020A	56	29	020A
03E4	C6	4D	03E4

CMPBIN is a very simple-minded program, and works best only when the two files load starting at the same address. If the files differ by one file having code inserted or removed, then mismatches will be found from the point where bytes were added or removed to the end of the file. If one file is longer than the other, the extra bytes will all be mismatches, with the shorter file's contents listed as "0106 00". If the data bytes are the same, but the files were assembled to load at different addresses, then every byte will be a mismatch, but only on the addresses, which is easily seen.



DATE

The DATE command is used to display or change an internal FLEX date register. This date register may be used by future programs and FLEX utilities.

DESCRIPTION

The general syntax of the DATE command is:

DATE[, <month, day, year>]

where 'month' is the numerical month, 'day' is the numerical day and 'year' is the last two digits of the year.

+++DATE 5,2,79 Sets the date register to May 2, 1979

Typing DATE followed by a carriage return will return the last entered date.

Example:

+++DATE May 2, 1979

DELETE

The DELETE command is used to delete a file from the disk. Its name will be removed from the directory and its sector space will be returned to the free space on the disk.

DESCRIPTION

The general syntax of the DELETE command is:

DELETE,<file spec>[,<file list>]

where <file list> can be an optional list of file specifications. It is necessary to include the extension on each file specified. As the DELETE command is executing it will prompt you with:

DELETE "FILE NAME"?

The entire file specification will be displayed, including the drive number. If you decide the file should be deleted, type 'Y'; otherwise, any other response will cause that file to remain on the disk. If a 'Y' was typed, the message 'ARE YOU SURE?' will be displayed on the terminal. If you are absolutely sure you want the file deleted from the disk, type another 'Y' and it will be gone. Any other character will leave the file intact. ONCE A FILE HAS BEEN DELETED, THERE IS NO WAY TO GET IT BACK! Be absolutely sure you have the right file before answering the prompt questions with Y's. Once the file is deleted, the space it had occupied on the disk is returned back to the list of free space for future use by other files. Few examples follow:

+++DELETE, MATHPACK.BIN +++DELETE, 1.TEST.TXT, 0.AUGUST.TXT

The first example will DELETE the file named MATHPACK.BIN from the working drive. If auto drive searching is selected, the file will be deleted from the first drive it is found on. The second line will DELETE the file TEST.TXT from drive 1, and AUGUST.TXT from drive 0.

There are several restrictions on the DELETE command. First, a file that is delete or write protected may not be deleted without first removing the protection. Also a file which is currently in the print queue (see the PRINT command) can not be deleted using the DELETE command.

DCOPY

The DCOPY command is used to copy from one disk to another all files which were created on or after a given date. This permits convenient backup of only those files which are new.

DESCRIPTION

The general syntax of the DCOPY command is:

DCOPY, \(\drive\), \(\drive\), \(\lambda\), \(\lambda\), \(\lambda\), \(\lambda\), \(\lambda\)

where the first drive number is the drive to be copied from, the second drive number is the drive to be copied to. R is an option to replace existing files on the destination disk with the files being copied. $\langle month \rangle$, $\langle day \rangle$, and $\langle year \rangle$ indicate a date; only files created on or after this date will be copied. If any part of the the date is left out, DCOPY defaults to the value in the corresponding FLEX date register. For example

DCOPY, 2, 1, 6, 1, R

would copy from drive 2 to drive 1 all files created after June 1 of the current year, replacing existing copies on drive 1.

DCOPY, 0, 3, ,, 81

would copy from drive 0 to drive 3 all files created after today's date in 1981 which were not already on the disk in drive 3.

DCOPY, 1, 0, R

would copy from drive 1 to drive 0 all files which were created today, replacing existing copies of these files on drive 0.

DCOPY logs each file copied on the console with the message

n.filnam.ext TO DRIVE #n

These messages may be redirected to the printer or to a file with the P and O commands to provide a record of operations.

If the R option is selected, files on the destination disk with the same names as files to be copied will be deleted, and replaced by the copied files, unless the date of the destination disk file is more recent than that of the source disk file. In that case, DCOPY will print

DEST FILE IS NEWER - NOT COPIED

and go on to the next file. This prevents the user from "backing up" an out-of-date file onto the current file.

NOTE: the date of a random-access file is the day it was created. FLEX does not change this date when the file is accessed. The GIMIX-supplied UPDATE utility command can be used to update the date of a random-access file.

		and the Billion of the delate can be a series of a filled a place and observe assuming a phonography and a manufacture graphy and a series of the series of
		- 하는 기계를 통해 이 등을 받았다. - 그리지 기계를 통해 기계를 하는 것들을 하는 것이다.
마루 됐는 사용에 되었다고?		

The EXECute command is used to process a text file as a list of commands, just as if they had been typed from the keyboard. This is a very powerful feature of FLEX for it allows very complex procedures to be built up as a command file. When it is desirable to run this procedure, it is only necessary to type EXEC followed by the name of the command file. Essentially all EXEC does is to replace the FLEX keyboard entry routine with a routine which reads a line from the command file each time the keyboard routine would have been called. The FLEX utilities have no idea that the line of input is coming from a file instead of the terminal.

DESCRIPTION

The general syntax of the EX command is:

```
EXEC, <file spec>
```

where <file spec> is the name of the command file. The default extension is TXT. An example will give some ideas on how EXEC can be used. One set of commands which might be performed quite often is the set to make a new system diskette on drive 1 (see NEWDISK). Normally it is necessary to use NEWDISK and then copy all .CMD and all .SYS files to the new disk. Finally the LINK must be performed. Rather than having to type this set of commands each time it was desired to produce a new system diskette, we could create a command file called MAKEDISK.TXT which contained the necessary commands. The BUILD utility should be used to create this file. The creation of this file might go as follows:

```
+++BUILD,MAKEDISK
=NEWDISK,1
=COPY,0,1,.CMD,.OV,.LOW,.SYS
=LINK,1.FLEX
=#
+++
```

The first line of the example tells FLEX we wish to BUILD a file called MAKEDISK (with the default extension of .TXT). Next, the three necessary command lines are typed in just as they would be typed into FLEX. The COPY command will copy all files with CMD, OV, LOW, and SYS extensions from drive O to drive 1. Finally the LINK will be performed. Now when we want to create a system disk we only need to type the following:

```
+++EXEC, MAKEDISK
```

We are assuming here that MAKEDISK resides on the same disk which contains the system commands. EXEC can also be used to execute the STARTUP file (see STARTUP).

FLEX User's Manual

There are many applications for the EXEC command. The one shown is certainly useful but experience and imagination will lead you to other useful applications.

IMPORTANT NOTE: The EXEC utility is loaded into the very upper end of user memory. This is done by first loading EXEC into the utility file space, then calculating the proper starting address so that it will reside right up against the end of the user memory space. Next EXEC is moved to that location and a new end of memory is set to just below EXEC. When the EXEC file is finished, if the user has not further changed the memory end location, EXEC will reset it to the original value.

EXTEND

EXTEND enables the user to increase the amount of space allocated to the directory on a newly formatted disk. This prevents the directory from becoming fragmented when the number of directory entries exceeds the space allocated by the disk format program. Fragmenting the directory increases the amount of time require to access files on the disk.

DESCRIPTION

The general synatax of the EXTEND command is:

EXTEND[,dn,sn]

Where 'dn' is an optional drive number and 'sn' is the number of additional sectors to be allocated to the directory. If no drive number is specified, EXTEND defaults to the work drive and adds 10 sectors. If the work drive is set to 'ALL', EXTEND prints an error message and reurns to FLEX. The maximum number of additional sectors that can be allocated to the directory is 10. Each sector adds space for 10 entries to the directory allocation. Some examples follow:

+++EXTEND +++EXTEND, 2, 5

The first example will EXTEND the directory of the disk in the work drive by 10 sectors (100 entries). The second example will EXTEND the directory of the disk in drive #2 by 5 sectors (50 entries).

The following table lists the number of sectors/directory entries normally allocated by FORMAT:

	5"			8''
	SECTORS	ENTRIES	SECTORS	ENTRIES
SINGLE SIDED	6	60	11	110
DOUBLE SIDED	16	160	26	260

NOTE: EXTEND can only be used on a freshly formatted disk.

EXTEND can generate the following error message:

DISK CANNOT BE EXTENDED

Either there are already files on the disk or the first sector on track one was found bad when the disk was formatted.

	그리고 얼마나 아무렇게 하면 하는 사람들이 되었다. 그렇게 되었다는 사람이 나를 다 되었다.	<u></u>
	[18] 유럽 (18일) 12 (19) 보고 있는 것은 19 (19) 보고 있는 19 (19) 보고 있는 19 (19) 12 (19) 보고 있는 19 (19) 12 (19) 12 (19) 12 (19 (19) 12 (19) 13 (19) 13 (19) 13 (19) 13 (19) 13 (19) 13 (19) 13 (19) 13 (19) 13 (19) 13 (19) 13 (19) 13 (19	
	도로 있다. 마이크 가는 마이크로 모든 모든 가는 마이크 하는데 그런 그리고 있는데 보고 있다. 그리고 있는데 그리고 있는데 그리고 있다. 2008년 - 1000년 - 1000년 - 1000년 - 100년 - 1	
	등 경기 등 시간 마음이 되었다. 그는 그는 그는 그는 그는 그를 받는 것이 되었다. 그는 그를 받는 것이 되었다. 기계를 하고 있다. 그는 그는 그는 그는 그는 그는 그는 그는 그를 보는 것이 되었다. 그는 그를 받는 것이 되었다.	
	그 이 생물을 하고 있다. 그는 사람들이 되는 사람들이 되는 것 같아. 그런 사람들이 말씀 하는 것이다. 생물을 하는 것이 나는 사람들이 가능하는 것이 되었다. 그는 사람들이 되는 사람들이 되었다.	
	근데 이 어느 그 때문에 가는 사람들이 가는 사람이 되는 사람들이 걸었다.	
	어느리는 그들은 그리는 나는 말이 가는 그를 가는 것이 없는 것이 없는데 없다.	
	그렇게 하는 어느 하는 사람들이 되는 것 같아 살아 살고 있는데 그리고 있다.	7 ·
	아이 마음이 아이들에 가로 하는 것이 말이 하고 하게 되지 않는데	
	경기 가입하는 경기를 받는 것을 보고 있다. 그런 사람들이 되는 것이 되었다. 그런 것이 되었다. 그런 것이 되었다. 근 사람들이 되었다. 그런 사람들이 있는 것이 되었다. 그런 그런 그런 그런 그런 것이 되었다.	

FORMAT is used to format a new diskette. Diskettes as purchased will not work with FLEX until certain formatting information has been put on them. The FORMAT utility writes this information on the blank diskette and then verifys that the information can be read back. If FORMAT finds sectors that it cannot read it removes them from the chain of free sectors and prints their location.

DESCRIPTION

The general syntax of the FORMAT command is:

FORMAT[, <drive>]

Where (drive) is the number of the drive in which the disk to be formatted has been placed. If no drive number is specified the 'WORK' drive is used. If the 'WORK' drive is set to 'ALL' then the user is prompted for the drive number.

After FORMAT has determined the drive number it will ask:

SCRATCH DISK IN DRIVE #X ('Y' OR 'N')?

Where X is the drive number specified by the user or the 'WORK' drive. If the user types an 'N' the program will abort and return to FLEX. If a 'Y' is typed FORMAT continues with the following prompt:

DISK SIZE ('5' OR '8')?

The user then types in the size o the disk to be formatted. After this FORMAT prompts:

FORMAT SINGLE OR DOUBLE SIDED ('S' OR 'D')?

If the drive being used to format is a double sided drive and the user wants to format both sides of the disk type 'D', otherwise type 'S'. The next prompt is:

SINGLE OR DOUBLE DENSITY ('S' OR 'D')?

The user then types in whether he wants the disk to be formatted single or double density. Formatting a single density disk for double density use is not recommended. FORMAT then prompts:

NUMBER OF TRACKS TO FORMAT?

FORMAT is asking literally how many tracks does the user wishes to format. Standard sizes for 5.25" disks are: 35, 40, 70, 77, 80. Please consult the disk drive manufacturers data sheet for the particular drive being used to find the maximum number of tracks that the drive is capable of accessing. Even though it is possible to attempt to format a disk for more tracks then it is capable of, it is not recommended as it might cause damage to the disk drive. The next prompt is:

FORMAT SINGLE OR DOUBLE STEPPING ('S' OR 'D')?

		경기선 내내 남동 학교 교회 경인 회	
			가게 하는 것이 있는 것이 되었다. 그런
			요 그 이 그는 그는 맛을 가게 하고 하는 것
[발표][발표] [[[[[[[[[[[[[[[[[[[
젊은 말을 들어놓아 그 나라는 이번.	일 그는 항상이 가는 사람들이 없었다.		
		가 있다. 이 그 마음이 마음이 되었다. 200 년 지 않는 이 이 있습니다. 하나 있습니다.	일보다 (14. 보급 : # # # # # # # # # # # # # # # # # #

Some disk drives have double the normal number of tracks for that size drive. This is called a 'Double Tracking' disk drive. The Double Tracking drives have twice as many tracks per inch as regular disk drives. This makes them incompatible with regular drives. This option enables the user to create a disk that will be usable on a regular disk drive, but was formatted on a Double Tracking disk drive. This can be useful for program exchange, etc. When formatting a disk double stepped it is recommended that only fresh, i.e. disks never used, are used. This will alleviate possible interchange problems resulting from the differences between single and double tracking drives. The next prompt is:

1MHz OR 2 MHz CPU SPEED ('1' OR '2')?

This information is used by FORMAT to determine which sector interleave pattern to use when writing the disk. The sector interleave is optimized for fastest access time at each CPU speed. If the system is running at 1.5MHz, enter 2MHz when formatting 5" disks and 1MHz when formatting 8". The next prompt is:

DISK NAME?

The user enters the name of the disk that is to appear in catalog listings. If the user just enters a carriage return a disk name of 'GIMIX .CHI' is put on the disk. The next prompt is:

VOLUME NUMBER?

The user enters the volume number that is to appear in catalog listings. If carriage return is entered then the disk number will be zero (0). If the user entered a carriage return for the name prompt, this prompt will be skipped and a volume number of '60609' will be put on the disk.

After entering the volume number FORMAT then prints all the data just entered and prompts:

IS THE ABOVE CORRECT ('Y' OR 'N')?

If the data typed in is correct then type 'Y' and FORMAT will go on. If an 'N' is typed then the prompts start over again. The final prompt is:

ABORT FORMAT ('Y' OR 'N')?

This is the users LAST chance to stop the formatting and save the disk in the specified drive. Typing an 'N' will start the format WITHOUT any further user interaction. Typing a 'Y' will abort the format and return to FLEX. FORMAT will now print:

FORMATTING TRACK: XX

Where XX is the track currently being formatted. The track number will be updated as each track is formatted. After all tracks have been formatted FORMAT will print:

: : : : : : : : : : : : : : : : : : :	
하는 것이 되는 것이 되었습니다. 이렇게 들어나고 있다면 하는 것이 되었습니다. 그는 것이 되었습니다. 그는 것이 되었습니다. 그는 것이 되었습니다. 20 마다는 것이 되었습니다. 그는 것이 있는 것이 되었습니다. 그는 것이 되었습니다. 그는 것이 되었습니다. 그는 것이 되었습니다. 그는 것이 되었습니다.	
사용 등 보는 사이에 발생하는 것 같아. 이번 사용하는 사용하는 사용하는 사용하는 사용하는 사용하는 것 같아. 	
	기존 하다리면 그들다고 하를 잃었다니다면요?
발표가 있는 그는 이번 나이가 이렇게 하는데 하는데 되는 그렇게 가다고싶을데 하는데 하는데 없는데	
	시아 아이 아래는 것이 되었다니까?
고 있는 사람이 있는 것이 되었다. 그 사람들은 사람들은 사람들은 사람들은 사람들이 되었다. 교육 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은	
요리하는 이 바람들이 사람이 제가를 들어 하는데, 나가 사이를 다 되었다.	
보통하다 하는 사람이 하는 사람들이 가장 사람들이 가장 보고 있는 것이다. 그런 사람들이 다른 사람들이 되었다. 사용하다 하는 사람들이 가장 사람들이 되었다.	
#####################################	
'맞은 마다 마다 보다되는 '', '아들이 마음 보다 하다는 사람이 되었다. 그렇게 되었다.	
진행 보일이 어느로 그 만큼 나는 이 나는 것이 모르다면 된 남자들였다. 하고 이어나	
그리다는 사람이 하는 것 같아. 나는 사이를 보는 사이는 것 같아 보다를 했다. 이 보다 하다	
않는다. 그 마다가 그는 일이 하는 그런 이 같은 하고 있는데 하고 말하고 하는 것 같다.	
젊어가 들어 하는 사람은 사람들은 사람들은 사람들이 되어 있다. 그리고 사용하다 기업을 하는	
유통사람 아들 아내는 사람들이 있는데 나는 사람이 모르게 살아 들어 들었다. 그 아들은 아들이 되었다.	
보면 보면 보는 사람이 이 그는 보다 이 보고 있다. 이 이 보고 경험을 통해 있는 그리고 있다. 그런 이어, 경우 하는 남편이 되어 되고 있다. 그는 것을 하고 있다. 그 이 사람들이 되었다. 물리 기술 하는 보는 사람들이 하는 것을 보고 있는 것을 하고 있는 것이 되었다. 그리고 있는 것을 하는 보고 있는 것을 보고 있는 것을 보고 있다. 그는 것을 하는 것이 되었다.	
전 경기를 통해 보고 있는 것이 되었다. 그는 사람들에 가장 함께 보고 있는 것이 되었다. 그런 사람들이 되었다. 그런 사람들이 되었다. 그런 사람들이 되었다. 그런 사람들이 되었다. 그런 사람들이 	
마이 마이 마니아 마시아 마시아 마이 마이 얼마를 받는데 하는데 이 마이 바라를 하는데 바로 함께 되었다. 마이 마이 마시아 마시아 마이 마이 마이 마이 마이 마이 마시아 마이 마이 마시아 마이 마이 마이 마이 마이 마이 마시아 마이	
이 하는 사람이 많은 마음이 있다면 사람들은 사람들이 되었다. 그런 그들은 사람들이 되었다면 하는 것이 되었다. 1982년 - 1일 1982년 1일	

VERIFYING TRACK: XX

Where XX is the track currently being verified. FORMAT reads every sector on the disk after formatting. If it finds a sector that is can not read it removes the bad sector from the cahin of available sectors. Once a FORMAT has removed a sector it is unavailable to FLEX unless the disk is reformatted and does not error again. If a disk continually gives alof of errors or gives errors in different areas each time it is formatted the disk might be defective.

Upon successful completion FORMAT will print the following message, ring the terminal bell and then return to FLEX:

FORMATTING COMPLETED TOTAL SECTORS: XXXX

Where XXXX is the number of sectors available to the user. This number will vary depending on the number of tracks formatted, the size of the disk, whether the disk was formatted single or double sided and whether the disk was formatted single or double stepped.

The following is an explination of the possible error messages that can be generated by the FORMAT command:

NOT EOUGH MEMORY INSTALLED IN SYSTEM

This means that according to the FLEX 'MEMEND' pointer there is not enough memory installed in the system to format a disk. The user must have at least sixteen (16) 'k' of memory starting at \$0000 in addition to the RAM occupied by FLEX.

FORMATTING ABORTED

This error message is printed to inform the user that FORMAT returned to FLEX prematurely and that formatting was unsuccessful.

TOO MANY TRACKS FOR DOUBLE STEPPING

This error message means that the user tried to format more tracks than any drive is capable of handling when double stepped. When formatting double stepped the number of tracks on the drive is HALVED. The user is then prompted for the number of tracks to format, again.

ERROR WRITING BOOT SECTOR

This is a fatal error which causes the formatting to be aborted. This means that FORMAT could not put the necessary loading information on track 0, sector 1.

			g samuel by the sequence also all the		
				요즘 이 이렇듯이 뭐 하는데 뭐하는데 되었다.	
	임아, 항상 하는 것이 없는 것이 없다.				
	공연할 이 얼마가 들어났다. 그리다 그 아이다.				
				[24] : [10] - [1	
궦딦씂궦딦딦궦궦딦궦궦궦궦궦궦궦궦궦궦궦궦궦궦궦궦궦궦궦궦궦궦궦궦궦궦궦궦궦					
는 사람들은 사람들이 되었다. 그는 사람들은 사람들은 사람들은 사람들이 되었다. 그는 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은					
IN BROND TO THE PROBLEM OF THE PROB					
2000년 전한 마이트로 등이 기급된 동안 사람들은 보는 그 아프리아, 그리아 무슨 아래의 장상이 가겠었다. 보다 하는 아니라이 아이는 아들은 하는 하는 것 같아 나를 다시다니다.	선생님 경기를 하면 없이 들었습니다.	병생들 모든 사람이 하게 함께 살려면 하다			

SECTOR WAS NOT WRITTEN TO ZEROS

This is a secondary error and is only printed after a bad sector message has been printed. It tells the user that the sector did not clear when initially written to disk.

ERROR IN SECTOR LINKAGES

This is a secondary error and is only printed after a bad sector message has been printed. It tells the user that the pointers to the next sector were not written correctly.

ERROR VERIFYING SECTOR

This is a secondary error and is only printed after a bad sector message has been printed. It tells the user that the specified sector has a surface flaw and cannot be read.

FATAL ERROR

This tells the user that FORMAT found an error in a vital area of the disk and that the disk is unusable.

BAD SECTOR AT: TT-SS

This is the header message for the three secondary error messages. TT is the track number of the error and SS is the sector number of the error. FORMAT then removes the bad sector from the disk. A disk with a few bad sectors can still be used. Once a FORMAT has removed a sector it is unavailable to FLEX unless the disk is reformatted and does not error again. If a disk continually gives alot of errors or gives errors in different areas each time it is formatted the disk might be defective.

NO GOOD SECTORS ON DISK

This is a fatal error and tells the user that FORMAT could not find a single usable sector on the disk. This usually means that the disk is defective. Try formatting the disk again before rejecting it.

DRIVE NOT READY

This tells the user that the drive to be formatted in either does not have a disk in it or that the drive door is open.

DISK IS WRITE PROTECTED

This is a fatal error that tells the user that the disk in the specified drive is write protected and cannot be formatted until it is un-write protected. FORMAT will, however, format a disk that has been write protected using the SETUP command (see the 'SETUP' command decsription for more information on this option). The drive will be restored to whatever state is was in prior to formatting, when FORMAT has finished.

교육 교육 : 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1	
교통 그렇게 보일 이 집도에 전투다고요요. 반응하는 바라를 받아지는 것이 되는 것이 모든	
맞았다. 얼마 하나 나는 얼마나 하다리는 그 그리는 사람들이 모르다고	
	제상 : 이 그렇게 하는데 눈짝 그리스를 때면 그런 이 이 시험하여 있다.
상대원 동생하는데 그는 이번 보이고 있다. 아이들의 그리고 있는데 그는 그는 것	
목이 종립 교육도가 된다. 이 문학 이 등을 가는 하는 지수는 모양을 하고	
병생 경향과 내용하다 그 모든 사람이 되었다. 그리는 가는 사람이 되었다.	
이 글로 바다 그는 사람이 아이는 아이를 보고 있다. 그는 그는 그는 그는 그는 그를 모르는 것이다.	요즘 하다 하나 그는 얼마를 하는 사람들이 가는 경우 살아 되었다.
교육 레이트 그리지 않아 그리 말하는 그리 것들이는 그리는 목숨이 되었다면요?	
이 맞는데 한경 보는 이 보다는 사람들이 나는 이번 등을 되는데 되었다. 그 아이	
물론 그는 경우의 교통하는 그는 사이에 가장하다 모르 하고요?	
생활한 보고 사이트로 가지 않아 보다 하는 사람들은 사람들이 되는 사람들이 되었다.	시간 연락 전에 나는 반상으로 다른 연락 이번 모르는 전에 없었다.
동물이다. 나는 이 아이들이는 이 모든 보면 됐다고 하는 사람 있는데, 이 이 때문?	
물통하다. 1000년 1일 전 1000년 1일	
	병하고 있다는 아니라 아이들은 그는 그들은 그는 사람들은 사람들이 다른 사람이 되었다.
좋았다. 하는 경우 한 경우 하는 사람들이 가장 사람들이 되었다면 하는 것이 없었다.	
	이 경영병의 사이가 살아왔다. 그는 아이는 사람들은 사이를 가지 않아 나를 다 했다.
	보다는 것은 사람이 가장 되는 것으로 한 것으로 하는 것이 되었다. 그런 사람들은 사람들은 사람들이 되었다.
	:

WRITE FAULT IN WRITING TRACK

This indicates a hardware failure in the disk drive itself. If this message is gotten re-try the FORMAT and if it appears again the chances are that the disk drive is not functioning properly.

LOST DATA IN WRITING TRACK

This error should not normally occur. Since FORMAT inhibits the 'IRQ' and 'FIRQ' interrupts the only way to get this error message is if the system is getting 'NMI' interrupts. Eliminate the source of the interrupts and try again. If this error persists or there are definately no interrupts being generated in your system then there might be a hardware failure. If using the GIMIX 6809 PLUS CPU BOARD with the 58167 Time-of-Day clock option installed, make sure that it is not enabled for 'NMI' interrupts. See the Hardware Manual for information on how to do this.

ERROR IN ACCESSING SYSTEM INFORMATION RECORD

This means that the format and verify went properly but after verifying the disk when FORMAT went to write the disk information on track 0, sector 3 it encountered an error. This is a fatal error.

하셨습니다 마루하다 이 남인 전하다 이미를 내려가 된다면 가지만 하다면 취임 보고 시험으로 내려가 되었다.	
하게 되는 이 경우 이번 등에 보고 있다. 이 사람에 가는 하고 있는 것이 하면 이 작은 사람이 하는 것이 되었다. 사람이 되어 가장 없어?	
늦으로 들었습니다. 그는 그는 가는 그는 사람들이 되는 사람들이 되는 사람들이 되는 것이 되었다. 그는 것이다.	
선생님들이 마시막은 어디에 다른 이번에 되었다. 그 사람들은 그 등에 하나 나는 아이가 되었다. 그리고 아니는 아니는	
어려워하다 아니라 그는 사람들이 가장 하는 것이 되었다. 그 그는 그는 그는 그는 그는 그를 가는 것이다.	
보는 그 이 수 하게 하는 소문에 들어올라면서 그는 그는 그는 것이 되었다. 그 그 그 그를 모르는 이 그를 모르는 것이 되었다.	
통통과 문문 (1985년 1일 1일 등) 보고 있는 사람들은 사람들은 보고 있다. 보고 있는 다른 사람들은 보고 있는 것이 되었다.	
해석하면서 생산들이 들어 들어가 이 얼마 보이는 그림 나는 이 모양이는 이 보는데 그리고 하는데 된다는 것이다.	
그리 하는 그는 이렇게 되었다. 그는 이렇게 되었다면 보는 이렇게 하는 것이 되었다면 하는 그렇게 하는데	
, 이 목 등이 이 대한 것이 하는데 하는데 하면 하는데 하는데 이렇게 되었다. 이 본 내가 주었다. 필요 관계에 다 그	
## 사용하게 하다 중요 그 없는 이 보다 하게 하는데 하는데 하는데 하는데 하는데 하는데 하는데 하는데 하는데 되었다.	
대한 18 시간 : " 이 시하다는 이 시간에는 모든 시간 이 가는 사람이 되었다. 그는 이 사람이 나는 이 사람이 되는 것이 없다.	
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;	
할 수 있는 사람들은 사람들은 사람들은 사람들이 가는 사람들이 가는 사람들이 가는 사람들이 되었다.	

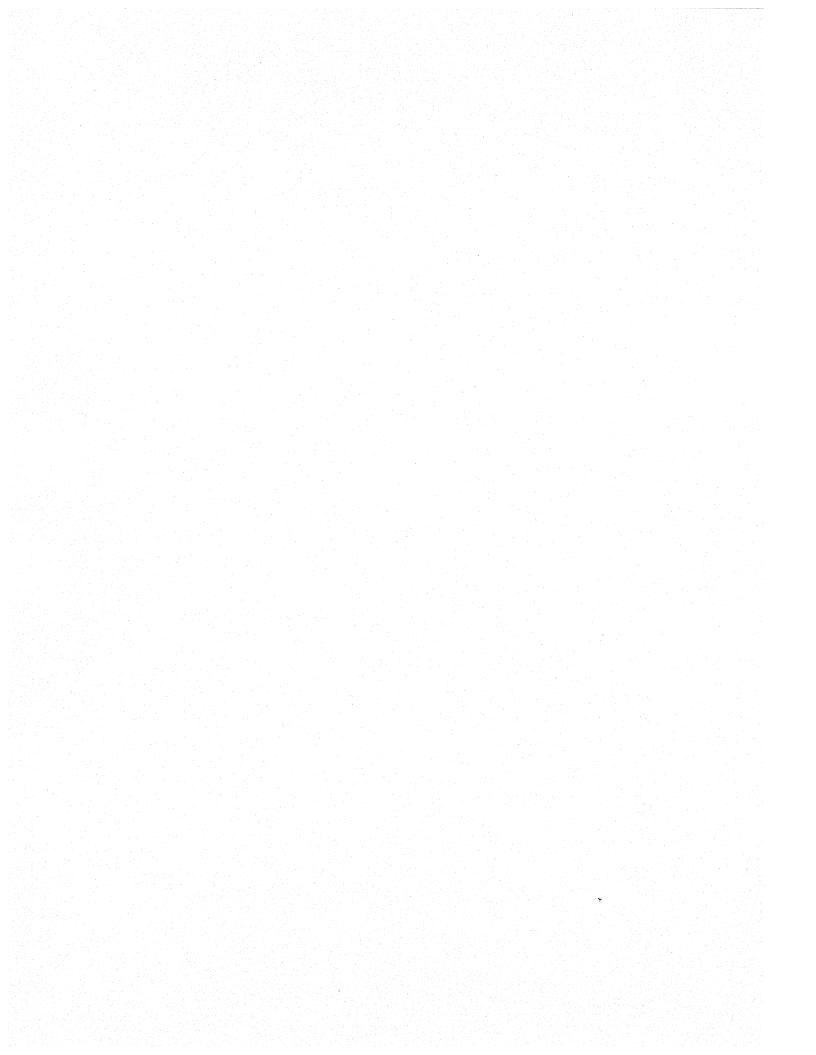
A system disk is the one from which the operating system can be loaded. Normally the system disk will also contain the Utility Command Set (UCS). The following procedure should be used when preparing system disks.

1. Initialize the diskette using FORMAT as described on the preceding pages.

NOTE: GIMIX/FLEX is distributed on single-sided, single-density disks. When formatting a disk to be used as a working system disk, it may be desirable to format the disk for greater capacity (double-density and/or double-sided). The BACKUP command can then be use to generate additional copies with the increased capacity.

- 2. Use the SYSGEN command to create a version of FLEX, with any required modifications, on the new disk.
- 3. Copy all .CMD files desired to the new disk.
- 4. Copy all .SYS files to the new disk. It should be noted that steps 3 and 4 can be done with one command, 'COPY,0,1,.CMD,.LOW,.SYS', assuming you are copying from drive 0 to drive 1. (the .LOW copies the SAVE.LOW command)
- 5. Last it is necessary to LINK the file FLEX4G.SYS to the system using the LINK commmand.

It is not necessary to make every disk a system diskette. It is also possible to create 'working' diskettes, disks which do not have the operating system on them, for use with text files or BASIC files. Remember that a diskette can not be used for booting the system unless the operating system in contained on it and it has been linked. To create a working disk, simply run FORMAT on a diskette. It will now have all of the required information to enable FLEX to make use of it. This disk, however, does not contain the disk operating system and is not capable of booting the system.



FIXDAY

The FIXDAY command is used to update the FLEX system date to the current date in the hardware clock on the GIMIX CPU Board without rebooting the system.

DESCRIPTION

The general syntax of the FIXDAY command is:

FIXDAY

This causes the day, month, and year in the hardware clock on the GIMIX CPU Board to be copied to the FLEX date registers. No output is generated. The date is normally set when FLEX is booted. But in some applications the user may want to leave FLEX running continuously several days at a time, or even permanently. This utility allows the FLEX date to be brought up to date without restarting the system.

	문화를 보면 생생하고 있는 것도 있는데 이 모든 사람들에서 가고 하는데 하는데 하는데 하는데 하는데 하는데 하는데 하는데 보다 되었다. 그는 것 같은데 이 이 이 이 이 사람들이 그 것이다. 그는 것은 말은 것 같은데 하는데 가는데 하는데 하는데 하는데 하는데 하는데 하는데 하는데 하는데 하는데 하	
다양 경기 하는 사람들은 경실 모시다면서 해 하는다. 문 보기의 물론으로 가는 사람이 없어 말을 모양한 음식을	분들하다 성으로 하시 하시아 호텔 등 분들하다 보고 하시아 하시아 하시아 하시아 보고 보고 있다. 그는 보고 있는 것은 사람들이 되었다. 그는 것이 모든 것이 되는 하다. 하시아 등 하고 있다. 그리고 있는데 그 중에 하시아 중에는 그들은 그들은 사람들이 되었다. 그는 것은 사람들이 하시아 하시아 중에 들어 있다. 그는 것이 하시아 하나 되었다.	
	는 물로 가능하게 되었다. 그는 그러워 없는 것 같아. 이 이는 말로 보이 이 이 중이다. 그는 말로 보는 말로 그렇는 그를 보고 있다. 그 사람이 모두 모든 것이다. 이 1일 이 사람들은 하는 이 그리고 말로 들어 보고 있다. 그리고 말로 그리고 말로 보이 아이들은 것이다. 그리고 말로 보는 것 같아.	
	이 문에 살아보면 요한다고 살아가 살아 이 사람들이 살아 들어 모든 모든	
	그는 경우 그는 이 눈을 하는 것들은 사람들은 바람이 나는 사람들이 살아 가지 않다.	

FREEMAP

The FREEMAP command is used to check the list of available sectors (free chain) on a FLEX formatted disk (floppy or hard) to determine the amount of fragmentation that exists.

DESCRIPTION

The general syntax of the FREEMAP command is:

FREEMAP, <drive>

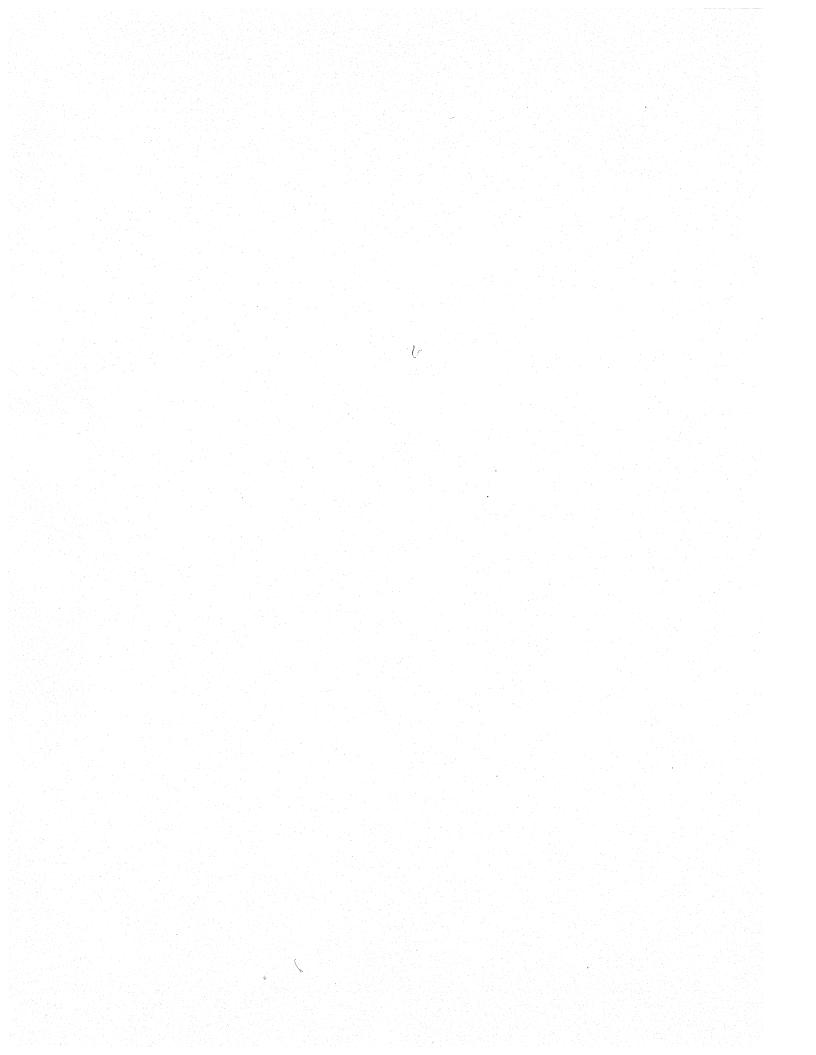
FREEMAP then scans all the sectors in the free chain of the disk in the designated drive, and lists on the console all the groups of continuous sectors found. The total number of such groups (called segments) is displayed at the end. By examining this list, the user can determine the degree of fragmentation of the disk, and decide whether to run UNSNARL on it, or copy the files on it to a new disk. Example:

FREEMAP, 1 READING FREE CHAIN 0908-1012 0706-0708 1120-1306 2208-2209 220C-220C

SEGMENTS MAPPED: 5

The segment count is printed in decimal. The output from this program can be routed to the printer or to a file with the P or O commands.

See the UNSNARL command for more information.



The HARD commands are used to attach a hard disk subsystem to $Gimix\ FLEX\ 4.0$ at any time. (SYSGEN is used to make the FLEX boot up with hard disk.)

DESCRIPTION

The general syntax of the HARD commands is:

HARD(drive #>

where (drive #> is the last character of the command name. The Gimix hard disk subsystem will be initialized and linked to FLEX. The hard disk subsystem can have one or two drives. HARD1 assigns hard disk 0 as FLEX drive 1, HARD2 as drive 2, and HARD3 as drive 3. HARD23 assumes two drives, and assigns hard disk 0 as drive 2 and hard disk 1 as drive 3.

HARDn requires 380 bytes of memory at \$E500. If the system does not have a Gimix CPU board, then the user must provide this memory some other way.

HARDn runs two diagnostics on the controller when it is loaded. If the controller fails, HARDn will print

FAULT IN CONTROLLER BUFFER

or

FAULT IN CONTROLLER INTERNALS

If this happens, please check all connections in the system and try again. If the problem persists, contact Gimix immediately.

If an error occurs during normal disk operations, HARDn prints

CONTROLLER ERROR - STATUS BYTES: nn nn nn nn

where nn nn nn nn are the four "SENSE STATUS" bytes described in the controller hardware manual. If the error persists, note the value of these four bytes and contact Gimix.

사용하는 그 그 아내 그들은 경험에 가장하는 사람들이 가장 하는 것이 되었다.	
	뭐요요. 그리는 얼마를 하는 것이 없는 그는 그는 그는 그는 그를 다니다.
이동 중요 않는데 하는데 이 사람은 생각하는데 하는데 나를 하는데 나를 하는데 했다.	
- [
생활물병 경기는 눈도 그렇게 이름을 내가 하나가 하나 있다면 하다.	
반속 지는 하래에 하게 하는 아니라 모든다. 이 아시스 하는 아름이 되어	
일본 반대에 가장이 하나는 나는 전 회원 교회 전에 가는 사람.	
그리다 하는 사람들으로 가게 되었다. 그 그 그들은 그 그는 이번 하나는	
해가일 찾는 호텔 전화 가는 그는 그는 사이지가 되는 것 같아.	
그 이 사용과 이 사람들은 그리고 있는데 하고 있다면 하는데 없다.	
(1988년 - 1982년 - 1982년 - 1984년 - 1982년 - 1982년 1982년 - 1982년	
통통 회원 및 전환 경기 전 기업을 보고 있다면 있다. 그리고 전 경기 등로 보고 있다는 것은	
함, 일이 보면한 내는 이번 아이가 보다 이번 아이가 하셨다면 하는 것 같아요.	
경우는 뭐 보이지만 하는 말로 보고 있는데 이 회사들로 다시 그들은 다른다.	
요하는 사람들은 사람들이 하는 사람들이 하는 사람들이 되었다.	
보다 하시 하는 말이 하시면 이렇게 되었다는데 나를 받아 되었다면 말했다.	·
공장 그 그리 젊은 지수를 하지만 당시를 가장 사람들이 들어가게 느껴지면 했다면 살아 있다.	

HARDFORM

HARDFORM is used to format a Gimix hard disk drive for use with Gimix FLEX 4.0. It establishes the interleave, free chain, catalog, and System Information Record (SIR).

DESCRIPTION

The general syntax of the HARDFORM command is:

HARDFORM

HARDFORM responds with the prompt

WHICH DRIVE (LUN) TO FORMAT (0/1)?

The Gimix hard disk subsystem can support one or two hard disk drives, which are referred as LUNO and LUN1. If only one hard disk drive is attached, it defaults to LUNO.

Once the drive is selected, HARDFORM begins to operate. The formatting process consists of several steps. In the first step, HARDFORM initializes the controller and drive, and performs two diagnostic tests on the controller. If either of these tests fails, HARDFORM aborts with the message

ERROR IN INITIALIZING CONTROLLER

If this message comes up, check all connections in your system and try again. If it comes up again, contact Gimix immediately! Next, HARDFORM has the controller generate the drive's internal format. This consists of the address, gap, data, and checksum fields. HARDFORM prints

GENERATING INTERNAL FORMAT

at this time. Once this internal format has been written to the disk, HARDFORM checks this format by reading each track. Defective tracks are flagged for exclusion from the free chain. At the start of this step HARDFORM prints

CHECKING INTERNAL FORMAT NOW CHECKING AT nnnn

where nnnn is the disk address of the last track checked. NOTE: HARDFORM reprints this message on the same line after every track and may cause problems on a hard copy console.

When HARDFORM finds a defective track, it prints

ERROR IN INTERNAL FORMAT AT nnnn

where nnnn is the address of the defective track. HARDFORM keeps

역사 등 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1	
사람이는 그 나는 그리는 그리다는 사람들은 가장에 가장 보는 것이 되었다. 그리는 사람들은 사람들은 사람들은 사람들은 사람들은 다른 사람들이 다른 사람들은 다른 사람들은 다른 사람들은 다른 사람들은 다른 사람들은 다른 사람들이 다른 사람들이 다른 사람들이 다른 사람들은	
그리고 하는 사람들은 그 사람들이 살아왔다면 가장 사람들이 되었다. 그는 사람들이 살아 들어 되었다. 그는 사람들이 가장 하는 것이 되었다.	
도 생기를 받았다. 이 이 그는 경기를 하고 하는 것이 되었다. 그런 이 이 이 이 이 이 이 이 이 이 이 되는 것이 이 이 이 이 이 이 이 이 이 이 이 이 이 이 이 이 이 이	
에 가장 이름 보고 있다. 이는 것 같은 사람들이 가장 하는 것이 되었다. 그 사람들이 되었다. 그 사람들이 되었다. 사용화 기를 보고 있다.	
있었다. 그 그러워 그 사람이 하다면 가는 사람들이 그를 살아하는 것만 하는 것은 다양이다.	
병원은 마음과 얼마 하고 보는 것이라고 있는 그리고 나는 그 사람들이 모든 것이다.	
가 보통하다 하는 것이 되었다. 그런 그는 사람들은 사람들은 사람들은 사람들은 사람들이 되었다. 그는 사람들은 사람들은 사람들이 되었다. 그는 사람들은 사람들은 사람들은 사람들이 되었다. 그는 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은	
: 사람들은 사람들은 사람들이 되었다. 그는 사람들은 사람들은 사람들은 사람들은 사람들이 되었다. 그는 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은	

count of the defective tracks. If the number of defective tracks exceeds 100, HARDFORM will abort with the message

100 BAD TRACKS!! - FORMATTING ABORTED

If this happens, contact Gimix immediately. When HARDFORM is through checking the internal format, it prints

INTERNAL FORMAT COMPLETE ESTABLISHING FLEX FORMAT

and begins writing out FLEX pointer data to all the sectors in the free chain. While HARDFORM does this, it prints

NOW FORMATTING AT nnnn

once every 256 sectors, where nnnn is the last sector written. This message is reprinted on the same line each time, and may cause problems on a hard copy console. After all the free chain pointers are done, HARDFORM sets up the catalog. 187 sectors are reserved for the catalog; if you need more, use the EXTEND command before putting any files on the disk.

HARDFORM will abort with the message

FATAL ERROR - FORMATTING ABORTED

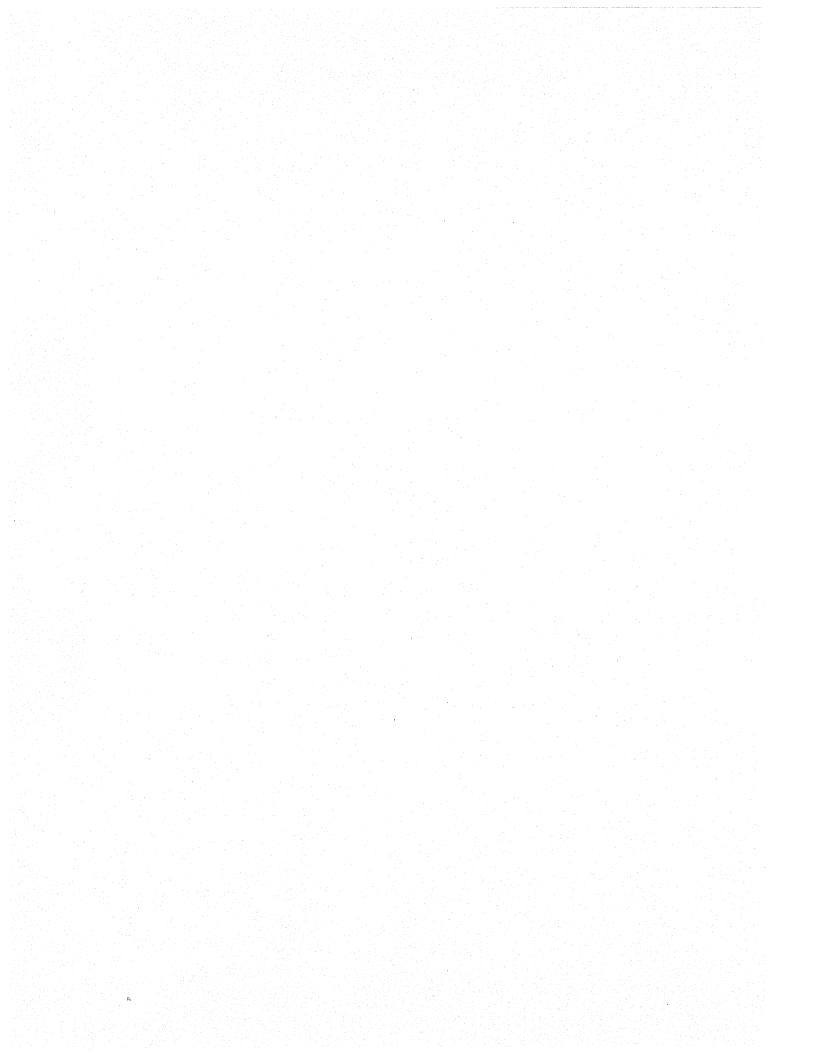
for a number of reasons, some of which have already been mentioned. HARDFORM will also abort with these messages.

ERROR ON TRACK OO

There was a formatting or write error in the catalog or SIR areas of the disk.

ERROR IN WRITING FREE LIST

There was a write error in a sector of the free chain.



The I command allows a utility to obtain input characters from a disk file rather than the terminal.

DESCRIPTION

The general syntax of the I command is:

```
I,<file spec>,<command>
```

where <file spec> is the name of the file containing the characters to be used as input and <command> is the FLEX utility command that will be executed and that will receive that input from <file spec>. The default extension on <file spec> is .TXT.

For example, say that on a startup you always wanted the file DATA.DAT deleted from the disk without having to answer the "ARE YOU SURE?" questions. This could be done in the following manner:

```
+++BUILD, YES
=YY
=#
```

The first Y will answer the "DELETE O.DATA.DAT?" question while the second Y will answer the "ARE YOU SURE?" question.

```
+++BUILD,STARTUP
=I,YES,DELETE,DATA.DAT
=#
```

Upon booting the disk, FLEX will execute the STARTUP file and perform the following operation: delete the file DATA.DAT receiving all answers to any questions from the input file YES.TXT rather than from the terminal.

See the description of the STARTUP command for more information on STARTUP.

The JUMP command is provided for convenience. It is used to start execution of a program already stored in computer RAM memory.

DESCRIPTION

The general syntax of the JUMP command is:

JUMP, <hex address>

where <nex address> is a 1 to 4 digit hex number representing the address where program execution should begin. The primary reason for using JUMP is if there is a long program in memory already and you do not wish to load it off of the disk again. Some time can be saved but you must be sure the program really exists before JUMPing to it!

As an example, suppose we had a BASIC interpreter in memory and it had a 'warm start' address of 103 hex. To start its execution from FLEX we type the following:

+++JUMP,103

The BASIC interpreter would then be executed. Again, remember that you must be absolutely sure the program you are JUMPing to is actually present in memory.

LINK

The LINK command is used to tell the bootstrap loader where the FLEX operating system file resides on the disk. This is necessary each time a system disk is created using NEWDISK. The NEWDISK utility should be consulted for complete details on the use of LINK.

DESCRIPTION

The general syntax of the LINK command is:

LINK, <file spec>

where $\langle \text{file spec} \rangle$ is usually FLEX. The default extension is SYS. Some examples of the use of LINK follow:

+++LINK,FLEX +++LINK,1.FLEX

The first line will LINK FLEX.SYS on the working drive, while the second example will LINK FLEX.SYS on drive 1. For more advanced details of the LINK utility, consult the "Advanced Programmers Guide".

The LIST command is used to LIST the contents of text or BASIC files on the terminal. It is often desirable to examine a files without having to use an editor or other such program. The LIST utility allows examining entire files, or selected lines of the file. Line numbers may also be optionally printed with each line.

DESCRIPTION

The general syntax of the LIST command is:

LIST, <file spec>[, <line range>][, +(options)]

where the <file spec> designates the file to be LISTed (with a default extension of TXT), and <line range> is the first and last line number of the file which you wish to be displayed. All lines are output if no range specification is given. The LIST command supports two additional options. If a +N option is given, line numbers will be displayed with the listed file. If a +P option is given, the output will be formatted in pages and LIST will prompt for "TITLE" at which time a title for the output may be entered. The TITLE may be up to 40 characters long. This feature is useful for obtaining output on a printer for documentation purposes (see P command). Each page will consist of the title, date, page number, 54 lines of output and a hex OC formfeed character. Entering a +NP will select both options. A few examples will clarify the syntax used:

```
+++LIST,RECEIPTS
+++LIST,CHAPTER1,30-200,+NP
+++LIST,LETTER,100
```

The first example will list the file named 'RECEIPTS.TXT' without line numbers. All lines will be output unless the 'escape character' is used as described in the Utility Command Set introduction. The second example will LIST the 30th line through the 200th line of the file named 'CHAPTER1.TXT' on the terminal. The hyphen ('-') is required as the range number separator. Line numbering and page formatting will be output because of the '+NP' option. The last example shows a special feature of the range specification. If only one number is stated, it will be interpretted as the first line to be displayed. All lines following that line will also be LISTed. The last example will LIST the lines from line 100 to the end of the file. No line numbers will be output since the 'N' was omitted.

The NAME utility enables the user to change the name, extension, volume number and date in the system information sector of a disk.

DESCRIPTION

The general synatax of the NAME command is:

NAME[, dn]

Where 'dn' is an optional drive number. If no drive is specified NAME will use the work drive. If the work drive is set to 'ALL' an error message is printed. Some examples follow:

+++NAME +++NAME, 2

The first example will change the information on the disk in the work drive, assuming that the work drive is not set to all. The second example will change the information on the disk in drive #2.

NAME prints the current disk name, extension, volume number and date and then prompts for the new name. The new name and extension should be entered, followed by a carriage return. Entering only a carriage return will retain the old name. NAME then prompts for the new volume number. The new volume number should be entered, followed by a carriage return. Entering only a carriage return will retain the original volume number. After the new name and volume number have been entered, NAME prompts:

CHANGE DATE ('Y' OR 'N')?

Entering 'Y' changes the date on the disk to the "current date", Entering 'N' retains the old date.

NAME can generate the following error message:

ILLEGAL DRIVE NUMBER

Legal drive numbers are 0, 1, 2, and 3. A drive number must be specified if the work drive is set to 'ALL'.

NOTE: If NAME is used in a command line with multipe commands, it must be the last command on the line.

The N utility enables the user to automatically answer "N" (no) to "Y or N" prompts from other utilities. The N utility is especially useful when writing EXEC files.

DESCRIPTION

The general synatax of the N command is:

N, (command string)

Where (command string) is a valid command line to be executed. If N is used in a line with multiple commands, using the end of line character, it only affects the command immediately following it. For example:

+++N,COPY,0,1

Will copy, from drive #0 to drive #1, only those files that do not already exist on drive #1, automatically answering "N" (no) to any "DELETE ORIGINAL?" and "ARE YOU SURE?" prompts that occur because of duplicate files on the two disks.

존하게 하는 그는 사람들은 아이들 하는 일본 사람들은 살아가는 요즘 하다는 사람들은 보다 들었다. 그는 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은	
일일 보는 하는데 그 물이 하고 있는데 일일 없어야 하는데 하면 하면 하면 하면 하는데 하는데 하는데 되었다. 그 하는데 되었다. 그 하는데 얼마나 없었다.	
그래요 하라고 들고 그 그라고 이 맛이 맛있다. 하장에 들어난 목 사람들이 가입하게 되었다. 한 사회에 가지하게 되었다. 현지 아니다 다 나를 하는데 하는데 하다.	
요즘 얼굴 맛이라 된다. 그는 그리는 그렇게 화가 되었다. 그는 그리는 사람들은 그리는 그리는 그리는 그리는 그리는 그리는 것이다. 그리는 것이다.	
현물과 가게 하는 이 마음이 하는 이 살이 되는 것이다. 그리고 아니는 이 것이 하는 이렇게 먹어 되는 이 어떻게 되는 이를 모르는	
사용성이 마이들은 그림을 하는 말통하는 이외 위에 이상으로 하는데 되었다. 는 이익의 그만들 점심하는 모양이 그는 회사 되는	
그렇게 가는 하는 방에는 어때는 없어요? 나는 이번도 그 원인을 모르는 내가 얼마나 그렇게 되는 것 같습니다.	
가 그렇게 되는 그림을 잃는 작업을 하는데 하시고 있다. 우리 하는 그리는 등 가는 등 가운 그렇게 하는데 하는 건강하다고 해가 많을 모시네요. 나를	
크리 회사님, 시작에 가장일의 문화를 받고 하는 회사에 본 분들은 하는 사람들이 되었다. 그 사람들이 되었다.	
그가 돌아보고 하다고 있어 살아왔다. 그는 그림 아는 그는 사람들이 가는 사람이 가는 사람들이 모였다. 그래 모든 그는 그를 받는데 그를 다 하는데 그렇게 되었다.	
분야한다. 생각 그는 이 이 이 아이는 것 같아 하면 화면을 보고 하는 회에 나와 하는 사람들이 하는 것이다.	
이 맛있다면 하다 하다 그는 사람들이 가장 하는 것이 되는 것이 되었다. 그는 사람들이 되었다면 하는 것이 없는 것이다.	
생활물 취임을 잃어 있다. 그는 그는 그는 이렇게 되어 있는 이렇게 되는 이렇게 되는 것이 되는 것이 되는 것이 되는 것이 없는데 되었다.	
"바람은 하루 바다 보다 그 아이들 아이들 때문에 되었다"는 그 사람들은 그리고 하는 것이 되었다. 그리고 있다고 있다.	
하는데	

The O (not zero) command can be used to route all displayed output from a utility to an output file instead of the terminal. The function of O is similar to P (the printer command) except that output is stored in a file rather than being printed on the terminal or printer. Other TSC software may support this utility. Check the supplied software instructions for more details.

DESCRIPTION

The general syntax of the 0 command is:

0, <file spec>, <command>

where <command> can be any standard utility command line and <file spec> is the name of the desired output file. The default extension on <file spec> is .OUT. If O is used with multiple commands per line (using the 'end of line' character ':') it will only have affect on the command it immediately precedes. Some examples will clarify its use.

+++0,CAT,CAT

writes a listing of the current disk directory into a file called CAT.OUT

+++0,BAS,ASMB,BASIC.TXT

writes the assembled source listing of the text source file 'BASIC.TXT' into a file called 'BAS.OUT' when using the assembler

	*
et a fille de la companya de la com La companya de la co	

The P command is very special and unlike any others currently in the UCS. P is the system print routine and will allow the output of any command to be routed to the printer. This is very useful for getting printed copies of the CATalog or used with the LIST command will allow the printing of FLEX text files.

DESCRIPTION

The general syntax of the P command is:

P, <command>

where <command> can be any standard utility command line. If P is used with multiple commands per line (using the 'end of line' character), it will only have affect on the command it immediately preceeds. Some examples will clarify its use:

+++P,CAT +++P,LIST,MONDAY:CAT,1

The first example would print a CATalog of the directory of the working drive on the printer. The second example will print a LISTing of the text file MONDAY.TXT and then display on the terminal a CATalog of drive 1 (this assumes the 'end of line' character is a ':'). Note how the P did not cause the 'CAT,1' to go to the printer. Consult the 'Advanced Programmer's Guide' for details concerning adaption of the P command to various printers.

The P command tries to load a file named PRINT.SYS from the same disk which P itself was retrieved. The PRINT.SYS file which is supplied with the system diskette contains the necessary routines to operate a SWTPC PR 40 printer connected through a parallel interface on PORT 7 of the computer. If you wish to use a different printer configuration, consult the 'Advanced Programmer's Guide' for details on writing your own printer driver routines to replace the PRINT.SYS file. The PR 40 drivers, however, are compatible with many other parallel interfaced printers presently on the market.

FLEX has the ability to output file stored data to a printer at the same time that it is performing other tasks. This feature is especially useful when it is necessary to print a long listing without tying up the computer. This method of printing is called PRINTER SPOOLING. In order for the printer spooling function to work, a SWTPC MP-T interrupt timer board must be installed in I/O position #4 on the computer's mother board.

DESCRIPTION

The general syntax of the PRINT command is as follows:

PRINT.<file spec>[.+<repeat #>]

where <file spec> is the name of the file to be printed. The default extension on <file spec> is .OUT. <Repeat #> is the number of additional copies of the file you wish to be printed.

For example, say that your disk had a very large number of files on it and a printer catalog listing was desired. A file containing the output information should first be created by using the O command such as:

+++0,CAT.OUT,CAT.CMD or +++0,CAT,CAT (see the description of the 0 command)

when printer output is desired the command

+++PRINT,CAT.OUT or +++PRINT,CAT

should be entered.

At this time the file CAT.OUT is stored in a buffer called a print queue (waiting list). If another PRINT command is issued before the first is finished, the second file will be in the next available location in the print queue.

After the file name to be printed has been stored in the print queue, control will return to the FLEX operating system. At this time you may perform any disk operation you want, such as deleting files, copying disks, etc. While you are using FLEX, PRINT will be outputting the desired file to the printer. PRINT will automatically wait for the printer to become ready (power up) even after the file has been entered into the print queue.

After printing the first file, the second file in the queue will be printed (if there is one), etc. The print queue may be examined or modified at any time by using the QCHECK utilty.

FLEX User's Manual

NOTE: There are several things that the user should be aware of when using the printer spooling:

- 1) Any file that is in the print queue may not be deleted, renamed, or changed in any way until it has been printed or removed by the QCHECK print queue manager utility.
- 2) Disks which contain the files in the print queue should not be removed while the files are still in the queue.
- 3) The P command should not be used while files are waiting in the print queue.
- 4) Any paper or cassette tape load or any other operation which requires that the computer accept data at precise time intervals should not be executed during a printer spooling operation.

PROT

The PROT command is used to change a protection code associated with each file. When a file is first saved, it has no protection associated with it thereby allowing the user to write to, rename, or delete the file. Delete or write protection can be added to a file by using the PROT command.

DESCRIPTION

The general syntax of the PROT command is:

PROT, <file spec>[,(option list)]

where the <file spec> designates the file to be protected and (option list) is any combination of the following options.

- D A 'D' will delete protect a file. A delete protected file cannot be affected by using the DELETE or RENAME Commands, or by the delete functions of SAVE, APPEND, etc.
- W A 'W' will write protect a file. A write protected file cannot be deleted, renamed or have any additional information written to it. Therefore a write protected file is automatically delete protected as well.
- C A 'C' will Catalog protect a file. Any files with a C protection code will function as before but will not be displayed when a CAT command is issued.
- X An 'X' will remove all protection options on a specific file.

Examples:

+++PROT CAT.CMD,XW
Remove any previous protection on the CAT.CMD
Utility and write protect it.
+++PROT CAT.CMD,X
Remove all protection from the CAT.CMD utility.
Prohibit INFO.SYS from being displayed in a catalog listing.

QCHECK

The QCHECK utility can be used to examine the contents of the print queue and to modify it contents. QCHECK has no additional arguments with it. Simply type QCHECK. QCHECK will stop any printing that is taking place and then display the current contents of the print queue as follows:

+++QCHECK			
POS	NAME	TYPE	RPT
1	TEST.	•OUT	2
2	CHPTR.	•OUT	0
3	CHPTR2.	•TXT	0
COMMAND?			

This output says that TEST.OUT is the next file to be printed (or that it is in the process of being printed) and that 3 copies (1 plus a repeat of 2) of this file will be printed. After these three copies have been printed, CHPTR.OUT will be printed and then CHPTR2.TXT. The COMMAND? prompt means QCHECK is waiting for one of the following commands:

COMMAND FUNCTION

(carriage return) Re-start printing, return to the FLEX command mode.

- Q A Q command will print the queue contents again.
- R,#N,X An R command repeats the file at position #N X times. If X is omitted the repeat count will be cleared. Example: R,#3,5
- D,#N A D command removes the file at queue position #N.

 If N=1, the current print job will be terminated.

 Example: D,#3
 - A T command will terminate the current print job. This will cause the job currently printing to quit and printing of the next job to start. If the current files RPT count was not zero, it will print again until the repeat count is 0. To completely terminate the current job use use the D,#1 command.
- N,#N A N command will make the file at position #N the next one to be printed after the current print job is finished. Typing Q after this operation will show the new queue order. Example: N,#3
 - An S command will cause printing to stop. After the current job is finished, printing will halt until a G command is issued.

FLEX User's Manual

- G A G command will re-start printing after an S command has been used to stop it.
- K A K command will kill the current print process. All printing and queued jobs will be removed from the queue. The files are not deleted from disk.

RENAME

The RENAME command is used to give an existing file a new name in the directory. It is useful for changing the actual name as well as changing the extension type.

DESCRIPTION

The general syntax of the RENAME command is:

RENAME, <file spec 1>, <file spec 2>

where <file spec 1> is the name of the file you wish to RENAME and <file spec 2> is the new name you are assigning to it. The default extension for file spec 1 is TXT and the default drive is the working drive. If no extension is given on <file spec 2>, it defaults to that of <file spec 1>. No drive is requird on the second file name, and if one is given it is ignored. Some examples follow:

+++RENAME, TEST1.BIN, TEST2 +++RENAME, 1.LETTER, REPLY +++RENAME, 0.FIND.BIN, FIND.CMD

The first example will RENAME TEST1.BIN to TEST2.BIN. The next example RENAMEs the file LETTER.TXT on drive 1 to REPLY.TXT. The last line would cause the file FIND.BIN on drive 0 to be renamed FIND.CMD. This is useful for making binary files created by an assembler into command files (changing the extension from BIN to CMD). If you try to give a file a name which already exists in the directory, the message:

FILE EXISTS

will be displayed on the terminal. Keep in mind that RENAME only changes the file's name and in no way changes the actual file's contents.

One last note of interest. Since utility commands are just like any other file, it is possible to rename them also. If you would prefer some of the command names to be shorter, or different all together, simply use RENAME and assign them the names you desire.

	1
	tana di Kabupatèn Ka
	<u>.</u>

REPORT

The REPORT command is used to list out the current system configuration: the type, size, data density, step rate, write enable, and step size for each disk drive, and the setting of the print spooler timing.

DESCRIPTION

The general syntax of the REPORT command is:

REPORT

REPORT takes no parameters and prints the system status as defined by the defaults, SYSGEN, and the SETUP command.

To use the REPORT command, type the following:

+++REPORT

The output is self-explanatory. For more information see the SETUP command.

This command uses the FLEX output routines, and its output may be routed to the printer or a file using the "P" or "O" commands.

경기 경기 시간 기계			
	물건물을 하는 사람들이 없다.		
	어디에 가장되면 아이를 하게 ?		
진행화학생님, 일반인 사람이 나 얼마?			
[연명증명회생활] : 10 - 2 - 2 - 2 - 2 - 2			

SAVE

The SAVE command is used for saving a section of memory on the disk. Its primary use is for saving programs which have been loaded into memory from tape or by hand.

DESCRIPTION

The general syntax of the SAVE command is:

SAVE,<file spec>,<begin adr>,<end adr>[,<transfer adr>]

where <file spec> is the name to be assigned to the file. The default extension is BIN and the default drive is the working drive. The address fields define the beginning and ending addresses of the section of memory to be written on the disk. The addresses should be expressed as hex numbers. The optional <transfer address> would be included if the program is to be loaded and executed by FLEX. This address tells FLEX where execution should begin. Some examples will clarify the use of SAVE:

```
+++SAVE,DATA,100,1FF
+++SAVE,1.GAME,0,1680,100
```

The first line would SAVE the memory locations 100 to 1FF hex on the disk in a file called DATA.BIN. The file would be put on the working drive and no transfer address would be assigned. The second example would cause the contents of memory locations 0 through 1680 to be SAVEd on the disk in file GAME.BIN on drive 1. Since a transfer address of 100 was specified as a parameter, typing 'GAME.BIN' in response to the FLEX prompt after saving would cause the file to be loaded back into memory and execution started at location 100.

If an attempt is made to save a program under a file name that already exists, the prompt "MAY THE EXISTING FILE BE DELETED?" will be displayed. A Y response will replace the file with the new data to be saved while a N response will terminate the save operation.

Sometimes it is desirable to save noncontiguous segments of memory. To do this it would be necessary to first SAVE each segment as a separate file and then use the APPEND command to combine them into one file. If the final file is to have a transfer address, you should assign it to one of the segments as it is being saved. After the APPEND operation, the final file will retain that transfer address.

SAVE.LOW

There is another form of the SAVE command resident in the UCS. It is called SAVE.LOW and loads in a lower section of memory than the standard SAVE command. Its use is for saving programs in the Utility Command Space where SAVE.CMD is loaded. Those interested in creating their own utility commands should consult the 'Advanced Programmer's Guide' for further details.

STARTUP

STARTUP is not a utility command but is a feature of FLEX. It is often desirable to have the operating system do some special action or actions upon initialization of the system (during the bootstrap loading process). As an example, the user may always want to use BASIC immediately following the boot process. STARTUP will allow for this without the necessity of calling the BASIC interpreter each time.

DESCRIPTION

FLEX always checks the disk's directory immediately following the system initialization for a file called STARTUP.TXT. If none is found, the three plus sign prompt is output and the system is ready to accept user's commands. If a STARTUP file is present, it is read and interpreted as a single command line and the appropriate actions are performed. As an example, suppose we wanted FLEX to execute BASIC each time the system was booted. First it is necessary to create the STARTUP file:

```
+++BUILD,STARTUP
=BASIC
=#
+++
```

The above procedure using the BUILD command will create the desired file. Note that the file consisted of one line (which is all FLEX reads from the STARTUP file anyway). This line will tell FLEX to load and execute BASIC. Now each time this disk is used to boot the operating system, BASIC will also be loaded and run. Note that this example assumes two things. First, the disk must contain FLEX.SYS and must have been LINKed in order for the boot to work properly. Second, it is assumed that a file called BASIC.CMD actually exists on the disk.

Another example of the use of STARTUP is to set system environment paramters such as TTYSET parameters or the assigning of a system and working drive. If the STARTUP command consisted of the following line:

```
TTYSET, DP=16, WD=60: ASN, W=1: ASN: CAT, O
```

each time the system was booted the following actions would occur. First, TTYSET would set the 'depth' to 16 and the 'width' to 60. Next, assuming the 'end of line' character is the ':', the ASN command would assign the working drive to drive 1. Next ASN would display the assigned system and working drives on the terminal. Finally, a CATalog of the files on drive 0 would be displayed. For details of the actions of the individual commands, refer to their descriptions elsewhere in this manual.

As it stands, it looks as if the STARTUP feature is limited to the execution of a single command line. This is true but there is a way around the restriction, the EXEC command. If a longer list of operations is desired than will fit on one line, simply create a command

file containing all of the commands desired. Then create the STARTUP file placing the single line:

EXEC,<file name>

where <file name> would be replaced by the name assigned to the command file created. A little imagination and experience will show many uses for the STARTUP feature.

By directing STARTUP to a file that does not have a return to DOS command it is possible to lockout access to DOS. You can correct the problem by hitting the RESET button and beginning execution at address \$CDO3. The STARTUP file may then be deleted and if desired, modified. Directing execution to CDO3, the DOS warm start address, bypasses the DOS STARTUP function.

The SETTIME command is provided so that the user may set the time on the Time-of-Day clock on the GIMIX 6809 PLUS CPU BOARD with the Time-of-Day clock option installed.

DESCRIPTION

The general format of the SETTIME command is:

SETTIME

SETTIME takes no parameters and prompts the user for all pertinent information needed to set the clock.

To use the SETTIME command merely type the following:

+++SETTIME

The computer will then respond like this:

MINUTES (1 - 59)?

The user then types in the minutes to be set to. The program then proceeds to prompt the user for hours, day of the week, day of the month and month.

If the computer responds to any of the prompts with this message:

INVALID INPUT, PLEASE RE-TRY.

It means that you did not enter a valid input for that prompt.

This program uses the FLEX line buffer to enable the user to delete or backspace his entry before carriage return is typed. To correct an error after carriage return has been typed the user must re-execute the SETTIME command.

After the time has been entered and the following line is showing:

TYPE ANY CHARACTER TO START THE CLOCK?

The time on the clock will stay where it has been set to until a character is typed on the keyboard.

Since this command uses the FLEX line buffer it cannot be use in multiple statement lines unless it is the last statement on the line.

발생님들이 그리고 말했다는 아이를 살고 있다. 그는 이번 모르고 그들을 하는 것 같아. 그는 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은	
도 하는 하는 것이 있는 생각이 하는 이 사람들이 되는 것을 되었습니다. 그는 사람들이 사람들이 되었습니다. 그는 사람들이 가장 사람들이 되었습니다. 	
에는 사람이 되는 것이 되어 있다. 그런 사람들은 사람들은 사람들이 되었다. 그런 사람들이 되었다. 그런 사람들이 되었다. 그런 사람들이 사람들이 되었다. 	
보다는 사용하다 하는 것 같다. 그는 사용이 되었는데 그리고 하는데 보다 하는데 그리고 하는데 되었는데 그리고 하는데 되었다. 그리고 하는데 그리고 하는데 그리고 하는데 그리고 하는데 그리고 하는데 그렇게 하는데 그리고 하는데 그런데 그리고 있는데 그리고 하는데 그리고 하는데 그리고 하는데 그리고 하는데 그리고 하는데 그리고 있는데 그리고 있는데 그리고 있는데 그리고 있는데 그리고 있는데 그리	
하셨다면도 하는 이번에 보는 이번 그들이 하는 것이 하는 하는데 하는데 하는데 하는데 하는데 하는데 하는데 하는데 하는데	
오랫 하지 않아 보고 있는 사람들이 가득하게 하고 있다. 그 사람들은 그는 그들이 말하는 것이 없다.	
그렇게 그는 내용 한 경우를 가는 것 같아 하는 것이 없는 것이 없는 것이 없는 것이 없다.	
현기는 것이 불어보면 하다고 하다. 사이들이 나라면 가는 바이를 다시하는 것이 되었다. 1985년 - 1일 - 1987년 1일 대한민국 사이들이 가는 1987년 1일 대한민국 1987년 1	
상황생활을 가는 것이 되는 것이 되는 사람들이 그리고 살아 있는 것이 되었다.	
없. 나는 물 사람들이 하면 그림도 하는 것이 하는 그들은 하고 있다면 하는 것이 되었다.	
마양 중에 보면 있다는 것이 되는 것이 되었다. 그 사이를 보고 있는 것이 되는 것이 되었다는 것이 되었다. 그는 것이 되었다. 그 사이를 보고 있다. 그는 것이 되었다. 생물을 보고 있었다.	
첫 1945년 1968년 1일 전 1968년 1일	
원래의 등 보면 이렇게 되면 하는데 그 되는 것을 보면 하는데 하는데 보다 하는데 보다 하는데 되었다.	
그리 불병도 그는 그리다는 그들이 그들이 얼마를 하는 것이 되는 것이 되는 것이 되는 것이 되었다.	
물과 그렇는 그는, 요즘 가는 이 나를 가면 하는 것 같아. 그는 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은	
하는 것이 되었다. 그런 그들은 이 마음이라는 것이 모든 것이 되었다. 그는 사람이 되었다. 그런 것이 되었다. 그런 경우에 보고 말했다. 그는 것이 되었다. 그런 것이 없는 것이 없는 것이다. 그러는 일본 전통적으로 되었다. 그런 것은 그런	

The SETUP command is used to set and display the control information kept by the operating system for each floppy disk. Using SETUP, the user can set the step rate, drive size, write protect, and double step options for each drive, and set the constant used for interval timing by the print spooler. SETUP also prints a table giving this information. SETUP operates either by interpreting a command line or in an interactive mode.

DESCRIPTION

The general syntax of the SETUP command is:

SETUP[, <drive > [options] [, <drive [options]]]</pre>

where <drive> is one of the numbers Ø, 1, 2, or 3 or the letter C, and [options] is a string of characters each of which sets one of options for that drive. (C indicates the spooler constant.) Gimix FLEX has a set of four Drive Control Blocks (DCBs), one for each each drive. The various options are controlled by flags in the DCBs. SETUP allows the user to modify these flags either with an option string following the drive number or by responding to prompts interactively. SETUP also prints a report of the current contents of the DCBs; if no drive numbers are specified then the DCBs are not changed but the report is still printed.

Each drive and options group on the command line is processed separately; an option string could be included for the first drive in the line, and omitted for the second which would be handled interactively. The report is printed after all the specified drives are processed.

If the user omits the option string after the drive number, then SETUP will print a prompt for each option and wait for a character to be typed indicating the user's choice. When the character is received SETUP makes the appropriate change to the DCB for the specified drive. The user must type one of the two characters shown, or a space to leave the option in its previous state and go on to the next prompt. The prompts and the choices for each are shown below.

INSTALL OR REMOVE DRIVE (I/R)?

If "R" is typed, the flag in the Drive Control Block indicating that the drive is installed is cleared and SETUP gets the next drive number from the command line. If "I" is typed then this flag is set and SETUP continues with the current drive. This flag is used by the FLEX disk driver routines when a drive is selected. "Removing" non-existent drives will keep the system from hanging up when a non-existent drive is accessed by FLEX.

DRIVE SIZE (5/8)?

If "5" is typed, the drive size flags in the DCB are set to treat the drive as 5". If "8" is typed, these flags are set for an 8" drive; the double stepping and fast stepping flags are also

and the control of th	
크레크리 집에 있는데, 이 발생 레이션 화대를 받는데 하는데 하다 되는데 하는데 없는데 하는데 하는데 그를 하는데	
그렇는 말이다. 아이라는 아이랑 아니라는 아이들은 아이들은 이 그리고 하는데 하는데 그리는데 이름을 다니다.	
사람들은 소리를 보고 있다. 이번 이번 전혀 전한 경기를 하는데 보다는 이번 이번 보다는 것이라고 하는데 이번 모든 것을 모든 주의가 되었다. 전에 가지 되는 것이다. 그들이 많은 그 전에 보는 그 것이 되는 것이라는 것이라면 하는데 보고 있는데 그들은 그들은 그를 보고 있는데 보다 되었다. 그런 것이 되었는데 되었다. 그 것이 되었습니다.	
[마루토리 B - 리 이 프라마트 다리 마르크 마르크 타스트 프로그	
가 이용한 교리 얼마가 되는 가게 했지 않는 아이는 모든 나는 얼마 가능한다면 하는 것이라는 모든가	
보면 보면했다. 그리고 아내는 사람이 아니는 데이 아이들은 사람들은 그리고 있다. 그렇게 되었다.	
가게 하면 있는 것이 되었다. 그는 사람들은 사람들이 되었다. 그는 사람들이 되었다. 사람들은 가게 하는 것이 되었다. 그는 사람들이 되었다.	
현실 등 개최적으로 하는 이번 경기로 있는 이번 이번 보고 있는 것이 되는 것이 되는 것이 되는 것이 되는 것이 되는 것이 되는 것이 되었다. - 2012년 - 1914년 전 1일 전 10	
보호 그런 경험, 그런 하는 일이 들어 하는 이번 하는 사람들이 되었다. 그는 이 그는 사람들이 살아 되었다.	
으로 하는 것이다. 그림, 그림, 그림, 그 사람들은 사용하는 그를 가는 다른 사람들이 되었다. 그는 그를 보는 것이 되었다. 그를 보는 것이다. 그를 보는 것이다. 그렇게 하는 것이 하는 것을 하면 하는 것이다. 그는 것이 되었다. 그는 그를 보는 것이 되었다. 그는 것이 되었다. 그를 보는 것이다. 그를 가지 않는 것이다. 그를 보는 것이다. 그를 보는 것이	
성장 전 시민 말로 그 경기 하고 있다. 그들만 있다 나는 말로 가는 바다 먹는 그는 것이다. 그는 사람이	
뭐야 할 때 그는 그는 그는 그는 그는 그를 가는 이렇게 그렇지 않는 그 그들이 그를 가는 것이 하는 것이다.	
[생동][[[[생생] [[[[[[[[[[[[[[[[[[[[[[[[[[[[
물로 보고 하는 항상 사람이 되는 것 같은 사람들이 보고 있는 수 없는데 되는 것 같아요?	
를 받는 하는 것이 되는 것으로 하는 것이 되었다. 그런 것이 되는 것이 되는 것이 되었다. 그런 것이 되었다. [18일 : 그렇게 기업 : 그런 그런 그런 그렇게 되었다. 그런 것이 없는 것이 되었다. 그런 것이 없는 것이 없는 것이 되었다. 그런 것이 되었다. 그런 것이 되었다. 그런 것이 되었다. 그런	
그를 되었는데 그 나는 그렇게 하는데 나를 잃었다면 나는 하는데 이 나이에는 본 생각이 되고 하다. 하는 사이에 나를 살아서 먹었다.	
그림들이 들었다. 그는 얼마를 하는데 그를 모양을 들는 그렇다는데 모양하는데 하셨다. 하는데 그는 그는데 그렇게 되었다.	
물건 시간에 한 그 나는 사람들이 가면 하는 얼마를 보면 가장 나를 가는 것이 되었다. 그는 사람들은 사람들은 사람들은 사람들이 되었다.	
몆씂, Berling : Herring Franch (Berling) Herring Herring (Berling) Herring Herring Herring Herring Herring Herri Herring Herring Herrin	
가 가입하는데 그는 마음 마음 경기에 가입을 되었다. 그리고 있다는 마음에 5번째 등로 하고 있다고 있다. 그런데 그는 사람은 하고 있다면 여러 모르는데 그를 다 먹어야. 보다 있다는데 그는 그리고 있는데 그를 되었다. 그를 보다 하는데 그리고 있다. 그런데 그는 그들은 사람들이 되었다. 그는데 그는데 그는데 그는데 그를 보고 있다. 그를 보고 있다. 그를 보고 있다.	

cleared, and the prompts for these options are skipped, as they are not allowed with an 8" drive.

SINGLE OR DOUBLE SIZE STEPS (S/D)?

If "S" is typed, the double stepping flag in the DCB is cleared. If "D" is typed, this flag is set, causing the seek routine in the drivers to double all head movements. This allows a 96-tpi drive to read 48-tpi diskettes. Because a 96-tpi drive can write to a 48-tpi diskette, but the diskette will not be readable by 48-tpi drives, the write protect flag is automatically set and the write protect prompt is skipped when this option is selected. CAUTION: double size stepping should never be selected for a 48-tpi drive, as it is not only useless but potentially damaging to the drive mechanism.

WRITE ENABLE OR PROTECT (E/P)?

If "E" is typed, the write enable flag will be set in the DCB; if "P" is typed, then this flag will be cleared. This prompt is skipped if double size stepping is selected.

NORMAL OR FAST STEPPING (N/F)?

If "F" is typed, a flag is set in the DCB which causes a 5" drive to step at the faster 8" rate when a seek is performed. If "N" is typed, the flag is cleared. See the next option for use of this option. This option is only applicable to 5" drives, and this prompt is skipped for 8" drives.

STEP RATE (1-4)?

This prompt requires a number from 1 to 4 as its response. The response selects one of the four stepping rates available, as shown in the table:

rate	8" drive	5" drive fast step	5" drive normal
	3 msec/track	3 msec/track	6 msec/track
2	6 msec/track	6 msec/track	12 msec/track
3 1	Ø msec/track	10 msec/track	20 msec/track
4 1	5 msec/track	15 msec/track	30 msec/track

The user should check the manufacturer's data sheet supplied with the disk drives to determine what stepping rate should be set for each drive.

"Drive number" C is the constant used by the system for timing during print spooler operation. The prompt for this option is

SET SPOOLER CLOCK FOR 1 MHZ OR 2 MHZ CPU (1/2)?

If a "1" is typed, this value is set for a 1 Mhz CPU; a "2" sets it for a 2 Mhz CPU. If FLEX has been modified to use the SWTPc MP-T timer for spooling then this option and will produce the message "FLEX IS MODIFIED FOR MP-T TIMER - SPOOL CLOCK IS FIXED".

를 받는 것이 되는 것이 되는 것이 없는 것이다. 그런 것이 되는 것이 되었다. 같은 것이 되는 것이 되는 것이 되는 것이 되고 있는데 되는 것이 되었다. 그렇게 되는 것이 되었다는데 되었다는데 되었다. 그렇게 되었다는데 되었다.	
도 이렇게 되었다. 그리고 마다 마다 이 생생이는 사람들이 되었다. 그 사람들은 사람들이 살아가고 있다고 있다. 그리고 있다. 그리고 있는 것이 되었다. 그 사람들은 사람들은 사람들은 사람들은 사람들이 되었다. 그 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은	
경찰 본 사는 사람이 보면 가는 사이를 보고 되었다. 사람들이 그리고 살아서 되는 것이 되는 것이 되었다.	
선택적으로 열심하다 하다 하는데 내고 하셨다. 이름 유명은 경험 등을 다가 들었다. 그 모모 되었다는데 있	
기를 들었다. 그는 살이 들려가 되고 말한 것이 모르는 것이 하는 것은 그 집에 없는 것이 되었다. 이 그들은	
프로그리트 왕들이 돌면 보고 하고 있었다는 그렇고 하다고 있는데 이번 전기 없다.	
기 등에 가지 하는 하는데 그는 사람이 있었습니다. 이렇게 살아 가는 사람들이 되었다.	
트립스트 (1987년 - 1987년 - 1987년 - 1987년 - 1987	
경기 등 경기 등 보는 것이 되었다. 그는 것이 되었다. 그는 것이 되었다. 그는 것이 되었다. 그는 것이 되었다. 기계 기계 등 경기 등 기계 등 기계 등 기계 등 기계 등 기계 등 기계	
보이 하는 사람들은 사람들은 경기에 가장 하는 것이 되었다. 그렇게 되었다. 그는 그는 사람들이 모든 사람들이 되었다. 그는 것이 되었다. 그는 것이 되었다. 그렇게 되었다는 것이 되었다. 그 그렇게 되었다. 그는 것이 되었다.	
있는 경기를 잃었다. 그는 그들이 그는 이번 이번 가는 그는 이번 이번도 살이 그 되는 경기를 받는다.	
그래는 공연을 통생한 시작으로 하는 경험을 하하셨다. 하는 사람은 이 그들은 이 물을 하고 있다.	
불류통하는 말이 이렇는 그리다 이는 물이는 물을 하는 것 같아 모든 하는 사람들이 되었다.	
분류하고 있다면 되었다. 중요한 1985년 1987년 - 1987년 1일	
화가, 가는 가는 하는 이 사람이 보다는 그 사람이 되는 사람이 되는 것이 하는 사람이 되었다.	
그는 방문하게 하시는 데 내가가 다고 보고를 가고 있다. 그들의 생각은 살은 사람이 있다.	
한 선택들은 그렇게 하는 것이 있습니다면 하는 이번 이번 사람이 가는 그 모든 것이다.	
불통하다는 이번 과인 이번, 모스 및 시민 대학생들은 이 모든 인터트 프린터 보고 하는 이 사람이 되는	
[사용] 그는 마음 이름이 살아 보고 살아 하면 하는데 하는데 하는데 하는데 하는데 하는데 되었다.	
나는 사람들이 되었다. 그는 사람들은 사람들이 가는 그는 사람들은 사람들이 가는 사람들이 되었다. 그는 사람들이 되었다.	나를 하는 것이 되었다고 하다죠.
마트 발표하다 한 경험 등 경험 등 기업에 되는 이 기업을 하는 것이 되었다. 이 기업을 하는 것이 되었다. 이 기업을 하는 것이 되었다. 그런 기업을 하는 것이 되었다. 그런 기업을 하는 것이 되었다. 	

Experienced users may wish to skip the slow interactive process and give all the option settings to the program at once. Or the user may want to put the SETUP command in an EXEC or STARTUP file, so that the options are set automatically. This can be done by putting the option codes in strings immediately following each drive number.

The codes used in a string are the same as the responses to the prompts. They can be in any order, provided no spaces or other separators are among them. Examples:

SETUP, ØIDF1

would "install" drive \emptyset and set it for double size steps and 3 msec/track stepping.

SETUP, 3S5N3, C1

would set drive 3 for single stepping, 5", and 30 msec/track, and set the spooler timing for a 1 Mhz CPU.

SETUP will output the following error messages when the listed conditions occur:

"x" IS NOT A VALID OPTION

A meaningless character was found in an option string.

"x" IS NOT PERMITTED NOW"

An option which is locked out by another option (such as write enable for a drive set for double size steps) was found. When this and the previous error occur SETUP goes on to the next option.

INVALID DRIVE NUMBER

A drive number other than \emptyset , 1, 2, 3, or C was found.

"n" IS A HARD DISK

Drive "n" has been switched to a hard disk device; SETUP is not usable with hard disks.

SYNTAX ERROR IN COMMAND

The command line was not acceptable to SETUP. CAUTION: the command line is executed as it is scanned. If there are acceptable option codes before the syntax error, these will have been processed before the error was encountered.

그는 일반들은 아이들이 그 토론에 반으로 불통하는데 그리겠습니다. 그들은 이 사람들이 가는 그는데 그 나는 그렇게 되었다.		
그는 회사회에 되는 그림을 하게 맞춰 이름이 가나왔다는 것 하게 하고 하는 것 같아. 그리고 있는 것이 없어 되었다면 하는데		
그는 물건들은 다른 아이들의 나를 받는 사람들은 과장을 하는 하는 하는 것이 하는데 하는데 하는데 하는데 하는데 가장이다. 나를 먹는데 하는데 하는데 하는데 하는데 하는데 하는데 하는데 하는데 하는데 하		
그는 그의 회사에는 그리는 교회의 하는 방법을 화결한 경에 가려지 않아 하는 그는 때문에 가는 것을 하는 것이라고 있다고 하다고 있다.		
그 중대통하실 이 전 이 병에 대하고 되고 말하고 있다. 하지만 하는 하는 하는 것 같아요. 그는 그는 사람이 무료하는 것은 것은		
	하실하게 하실하는 것이다.	
그 밖에 들는 그리고 하고 그는 사람들이 그들고 그 그릇들이 되어 하는 것을 하는 것이 없다고 있다.		
그가 나타 하이 시민들은 하는 그리고 있는데 나는 사람들이 가지를 받는데 그렇게 되었다. 그래요?		
그리다. 그는 그 그 이 그렇게 하다고 그렇게 하다는 그리고 그리고 있는데 그리고 있다면 하다 그리고 있다면 그리고 있다면 그리고 있다면 하다고 있다면 하다고 있다면 하다고 있다면 하다고 있다면 다른데 그리고 있다면 하다고 있다면 하다면 하다고 있다면 하는데 하다고 있다면 하다고 있		
그는 회장에서 있다. 이 전에 다른 회에 가는 이 사람들이 되는 것이 되면 그리고 있다.		
수가는 맛있다. 이 시간 아는 그는 사이는 이번 나는 어느는 이 불의 가는 사이를 하는 것이 하는데 하는데 하는데 하다. 그 이		
그는 그들이 보는 것이 되었다. 그는 이번에 얼마나 아는 아니까 되는 데 이 그들은 아니는 그런 그를 가지 않는데 하다.		
화가 가는 것이 많아 들어왔다. 그는 이 이 아이들이 되는 것이 되어 되는 것이 되었다. 그 사람들이 되었다.		
클릭을 보고 있으면 그 나는 아들 가면 가게 되는 것 같아 그는 그 말이 되는 것이 되었다.		
		1
		- 1
		-
		-
		1
		i
		i
		1
		l
		1
		ı
이 들어 보다 보이 되어야 되었다. 그는 그의 그렇게 되었다면 내용 하는 물 때문 있어? 그리고 하지 않아 그렇게 받는 모든 없는 것은	Γ	
생물에 가장이 그렇게 그리다의 하나 이름은 눈이 있다. 이를 이 그렇는 그리는 그리는 이 이번 그렇게 하고 그 그래? 그는		
보면 되는 한 남이 말라는 보면 하는 사람들이들의 그림을 보고 보고 있다면 보고 있다면 하는 모두 보 ^다	현실 경험 기계 등 보통하다 그 나는	
	2011년 - 1일 전 조실 등록 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
하는 것이 되는 것이 되는 것이 되었다. 그는 사람들은 보고 있는 것은 것이 되었다. 그는 것은 것이 되었다. 그는 것이 되었다는 것이 되었다. 사용하는 것은 것이 되었다. 그는 것이 없는 것이 되었다. 그는 것이 되었다. 그는 것이 되었다. 그는 것이 되었다. 그는 것이 되었다.		
. 이 없다. 이 생각을 하셨다는 것이 생각 하셨습니다. 그런 그림 그렇게 살아 살아 나는 아니는 아니는 아니다.		
	The state of the s	

SYSGEN

The SYSGEN command is used to build a customized version of GIMIX FLEX that is configured to match the user's system, enamed FLEX4G.SYS. With SYSGEN, the user can set the type, size, and stepping rate for each disk drive, include patch code to use the GIMIX hardware clock or substitute an MP-T timer for the GIMIX timer, set the default year, set the default system and working drive assignments, set the spooler interrupt timing to match the CPU clock rate. SETUP creates a bootable FLEX in which all these options are correctly set when FLEX is booted.

DESCRIPTION

The general syntax of the SYSGEN command is:

SYSGEN

SYSGEN then prints a header message, followed by a series of prompts. The user's responses to these prompts will set up the configuration of FLEX. The prompts and the allowed responses are explained below, in order of their occurrence in the program. Several prompts end with "(Y/N)". The allowed reponses to these prompts are "Y" for yes and "N" for no. NOTE: if SETUP repeats a prompt, it is because the user gave an illegal or meaningless answer. If this happens, read these directions carefully to determine the correct answer.

ENTER NUMBER OF DRIVE WITH BLANK DISK:

Before running SYSGEN, you should initialize a diskette with the FORMAT command, which will be your new boot diskette. This prompt asks which drive the blank disk is in; this is the disk the new FLEX will be created on. The answer must be a digit in the range 0 to 3.

ENTER NUMBER OF DRIVE WITH FLEX4.SYS AND .BIN FILES:

You must have your original diskette or a copy of it in the system for SYSGEN to get the components of the new FLEX from; this prompt asks where it is. The answer must be a digit in the range 0 to 3, and must be different from the drive with the blank disk.

INCLUDE PATCH FOR HARDWARE CLOCK-CALENDAR (Y/N) ?

Answering "Y" causes the hardware clock patch (DATEPAT.BIN) to be appended to the new FLEX, and invokes the next prompt. The patch modifies FLEX to use the contents of the Time-Of-Day clock on the GIMIX CPU board to set the system date at boot time, instead of prompting the user to enter it. It must not be included if the system does not have a GIMIX CPU board.

등급하는 항상을 하지만 모든 보다가 가장을 하려면 되면 하는 것은 하는 사람들은 가는 경기를 가지 않는 것 같아 가장 중심하다. 하다
그래, 이렇게 살아보다 뭐 뭐 하면 하는 것이 있다. 이번 사람들이 되어 하면 하는 사람들이 되었다. 그 모양하는데 한 것이다.
물에 많은 그 이 이 아니는 사람들이 되는 것이 없는 사람들이 살아 없다. 그 사람들이 되는 사람들이 아름다고 있다면 살을 다 했다.
그 보고 있는 그리고 가는 그리고 문제가 되어 가는 모든 이 들고 맞을 하게 되었다. 그리고 가지를 하는 이 모든 것이다.
그리고 그 바람이 그 모든 그림을 들어 있다. 그 얼마는 사람들이 되었다면 하는 사람들이 가장 하는 것이다.
이 가입하다 그는 이 얼굴로 보는 사람들은 사람들이 가는 가게 하는 것은 사람들이 되었다. 그는 사람들은 사람들이 되었다.
보는 이번에 가는 경우 나를 가지 않는데 이렇게 되었다. 그 사람들은 사람들은 사람들은 사람들은 사람들이 되었다.
선생님은 아이들 그들은 이번 학생님이라는 그들이 아무리면 남은 사람들이 이 가는 그를 가는 어느 얼마들까지 않는다.

ENTER DEFAULT YEAR FOR SYSTEM DATE:

The user's response to this prompt should be a decimal number from 0 to 99, followed by a carriage return. The value of this number will be the year in the FLEX date register when the new FLEX is booted. If the response is not a valid number or is out of range, the prompt will be repeated. This prompt is skipped if the answer to the previous prompt was "N".

MODIFY FOR MP-T TIMER (Y/N) ?

Answering "Y" to this prompt causes the MP-T patch (MPTPAT.BIN) to be appended to the new FLEX. The FLEX print spooler requires a source of periodic interrupts. GIMIX FLEX is set up to use the 6840 timer/counter on the GIMIX CPU board for this. Users who have a different CPU board must have a different interrupt source, such as the SWTPc MP-T timer board. This patch modifies the print spooler to use an MP-T board at \$E010.

INCLUDE HARD DISK DRIVERS (Y/N) ?

Answering "Y" to this prompt causes the hard disk drivers (HARDDISK.BIN) to be appended to the new FLEX. These drivers allow FLEX to use the GIMIX hard disk subsystem. They occupy 470 bytes at \$E500 in the scratchpad memory on the GIMIX CPU board. Users with different CPU boards must provide memory at this address in order to use the GIMIX hard disk.

HOW MANY HARD DISKS IN SYSTEM (1/2) ?

The GIMIX hard disk subsystem may have one or two hard disks; SYSGEN must know how many are in the system to properly configure FLEX. This prompt and the next two will not come up unless SYSGEN was told to include the hard disk drivers.

ENTER FLEX DRIVE ASSIGNMENT FOR HARD DISK 0 (0-3):

Each hard disk must be assigned to one of the four FLEX drive positions. Since this is done with tables, either hard disk may be assigned to any FLEX position. Assigning a hard disk to the same drive position as a floppy masks the floppy; FLEX will not be able to access it, but there are no other effects. Assigning a hard disk to position 0 has no effect on boot operations: FLEX will still be loaded from floppy disk, if a floppy is installed as drive 0. However all functions involving disk access, such as STARTUP, will access the hard disk.

ENTER FLEX DRIVE ASSIGNMENT FOR HARD DISK 1 (0-3):

This is the same as the previous prompt, except that the second hard disk is assigned. It does not appear if only one hard disk was specified. The assignment of the second hard disk must be different from the first.

	전 '루크' (Burkaran) 12 등 12 (14 24 11) 같은 물 보기 물이 되지 않는 말이 10 10 12 (15 25 25)	
시작되는 발문의 연락됐		
		[편집] 사람들 사람이 나는 바람이 들었다면요.

After the hard disk options have been handled, SYSGEN has the user set up the Drive Control Blocks (DCBs) for each of the four floppy disk drives. The prompts for this are repeated for each of the four drives, unless the system has hard disk. In that case, when SYSGEN comes to a drive which is defined as a hard disk, it prints this message

DRIVE n IS A HARD DISK

and skips on to the next drive. The prompts for floppy disks are as follows:

INSTALL OR REMOVE DRIVE n (I/R):

If "R" is typed, the "Drive Installed" flag in the DCB for drive n is cleared and SYSGEN skips on to the next drive. If "I" is typed then this flag is set and SYSGEN continues with the current drive. This flag is used by the FLEX disk driver routines when a drive is selected. "Removing" non-existent drives will keep the system from hanging up when a non-existent drive is accessed by FLEX.

ENTER DRIVE SIZE (5/8):

If "5" is typed, the drive size flags in the DCB are set to treat the drive as 5". If "8" is typed, these flags are set for an 8" drive; the fast stepping flag is also cleared, and the prompt for this option is skipped, as it is not allowed with an 8" drive.

NORMAL OR FAST STEPPING (N/F):

If "F" is typed, a flag is set in the DCB which causes a 5" drive to step at the faster 8" rate when a seek is performed. If "N" is typed, the flag is cleared. See the next option for more information. This option is only applicable to 5" drives, and this prompt is skipped for 8" drives.

ENTER STEP RATE (1-4):

This prompt requires a number from 1 to 4 as its response. The response selects one of the four stepping rates available, as shown in the table:

rate		8" drive		5" drive fast step		5" drive normal
1	3	msec/track	3	msec/track	6	msec/track
2	6	msec/track	6	msec/track	12	msec/track
3	10	msec/track	10	msec/track	20	msec/track
4	15	msec/track	15	msec/track	30	msec/track

The user should check the manufacturer's data sheet supplied with the disk drives to determine what stepping rate should be set for

사용 : 10 10 10 11 11 11 11 11 11 11 11 11 11	
그런다는 그리는 하면 가장 모든 그는 사람이 되고 있다. 그렇게 되는 그리는 그리는 그는 그는 그는 그리는 그는 그리는 그리는 그리는 그리	
아마이에 되는 그리고 있는 사람들은 마음이라는 사람이 되었다. 그는 그리고 있는 사람들은 사람들이 되었다. 그는 사람들은 사람이 사람들이 모르는 사람들은 사람들이 되었다. 교통하다 하고 있는 것이 나는 사람들이 있는 사람들이 가득하는 사람들이 되었다. 그런 사람들이 사용하는 사람들이 되었다. 그런 사람들이 사람들이 되었다. 그런 사람들이 되었다.	
그들은 이 사람이 가는 사람들이 가는 사람이 가는 사람들이 가는 사람들이 가는 사람들이 가는 사람들이 되는데 그들은 사람들이 되었다.	
마이 위에 가는 사람들은 보고 있다면 하면 하면 되었다. 그는 것이 하면 되었다. 그런 사람들이 보고 하면 가장 하면 보고 있는데 하고 있다. 그런 사람들이 되었다. 	
그렇다는 이 아이들은 아이들은 다른 아이들이 아니는 말이는 나는 살이는 그는 아이들은 다른 하다. 그렇게 하는 그	
#####################################	
가는 하기를 받았다. 그리는 말이다는 학생들은 가입하고 있는 사람들이 모르는 그리고 있다. 나는 사람	
교육은 보고 하고 그러워 많은 말이 되는 것이다. 하는데, 사람이 되는 것이 되는데 하나 없다.	
는 사례 경영 시간 생생님 그는 그 보고 되었다. 그리고 말하는 하는 사람들은 사람들에 가장 되었다. 그는 사람들은 사람들은 사람들이 되었다. 그런 사람들은 사람들이 되었다. 목사람들은 사람들은 사람들은 사람들이 가는 사람들은 사람들은 사람들이 가장되었다. 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은	
경우 그는 사람들이 그리는 그 맛있는 그 작가를 하는 것이 하는 사람들이 되는 사람들이 가장 하는 것이 되었다.	

each drive.

After all four DCBs are set up, SYSGEN issues this prompt:

SET SPOOLER CLOCK FOR 1 MHZ OR 2 MHZ CPU (1/2):

If a "1" is typed here, the spooler interrupt routines are set up for a 1 Mhz CPU; a "2" sets them up for a 2 Mhz CPU. If FLEX is modified for an MP-T timer, then this option is not allowed, and this prompt is skipped.

The last options set with SYSGEN are the default system drive number and work drive number. These are prompted for as shown here.

ENTER DEFAULT SYSTEM DRIVE NUMBER (0-3 OR A):

The user must enter a digit from 0 to 3 or the letter "A" for "ALL". Entering "A" has the same effect as in the ASN command: the file will be searched for on all the drives starting with drive 0. If the user enters the number of a drive which was set "NOT INSTALLED" during the current run of SYSGEN, SYSGEN will print

THAT DRIVE IS NOT INSTALLED

and repeat the prompt. The next prompt is

ENTER DEFAULT WORK DRIVE NUMBER (0-3 OR A):

which works the same way.

Once all the options are selected, SYSGEN prints a table showing all the settings and the prompt

IS THIS SYSTEM SET-UP CORRECT (Y/N) ?

If the user types "Y", SYSGEN confirms this answer with the prompt

ARE YOU SURE (Y/N) ?

If the user types "Y" again, then binary data to set the options at load time is appended to the new FLEX, completing it, and SYSGEN prints

SYSTEM GENERATION COMPLETE

and rings the console bell. The file FLEX4G.SYS is now on the output disk. It must be linked with the LINK command, but is otherwise ready to boot. If the user types "N" in response to either of the last two prompts, SYSGEN deletes its output file and starts over.

		an an ar de les actues en trapación de paracel en consequença Madeiro Paradesina de America a al America de Co
	등 이 보는 10년 등 경기 회에 경기 이 보면 되고 보고 보면 되는 모든 것이 이번 등 보이 그런 이번 이 제공을 하면 하는 것이다. 이 전 경기 제공을 위한 보고 이 이는 그를 하는데 화면 중에 전한 경기를 하는 것이다. 그렇게 하는데 하는 것이 되었다.	
	이 아이들이 되는 사람이 하는 사람들이 되었다. 그 사람이 나는 이렇지 않는 것이다.	
	이 불통 사람이 되는 사람들은 일반하다는 것 같아요. 그리는 사람 이동에 모든데 함께 다.	
	그림은 이렇게 되지 않는 아이들이 얼마를 하는 것 같은 사람이 없다.	
	요즘이는 그로 보이 시작을 하여 이는 그리라 하는 것 같아 그는 하고 말했다.	인공도 기존하는 시시 회교도 .
	로칠 계속은 그렇지만 하나는 어때에 그렇게 되는 사람이 없는 아이 마리에 불렀다면	
그런 얼마를 하는 것은 이외를		
	지역도 존대 시민도 이 글이 되고 있는 바람들은 그리고 살아 다	
	된 전기되어 되면 되었다. 한 화학도 하고 있었다. 그렇다 하면 이 회에 제어	
	그 연구 그 물건도 하는 말면 시작되는 방법이 방법이 되었다. 그렇게 되었다.	
	는 보는 그렇게 그렇게 되었다. 사람들의 사람들은 바라를 하는데 살아 다 다	
	보다 이 물론 하다면 들었다고 되고 이 하면 있었다. 그리아 모든 그의 회사를 되어 하는 것	
	성공이 존경 시간을 가려면 보다 나를 통해 보고 하루하고 있다. 않는데 보다	
	사람들이 하고하는 것은 경험이 가득하는 사람들이 가지만 하지만 하셨다면 하는데 보고 하다면 하는데 나는데 가는데 하는데 가지 않는데 하는데 하는데 하는데 하는데 하는데 하는데 하는데 하는데 하는데 하	
	- 경기 발생 보고 있다. 그리고 있다고 하는 하를 하는 것으로 하면 하는 것으로 되었다. 그리고 있는 것으로 되었다. 	
	- 경기 발생으로 한다. 전기 시간 교육 전기 회사를 하는 것이 되었다. 그는 것이 되었다. 	

TTYSET

The TTYSET utility command is provided so the user may control the characteristics of the terminal. With this command, the action of the terminal on input and the display format on output may be controlled.

DESCRIPTION

The general syntax of the TTYSET command is:

TTYSET[,<parameter list>]

where <parameter list> is a list of 2 letter parameter names, each followed by an equals sign ('='), and then by the value being assigned. Each parameter should be separated by a comma or a space. If no parameters are given, the values of all of the TTYSET parameters will be displayed on the terminal.

The default number base for numerical values is the base most appropriate to the parameter. In the descriptions that follow, 'hh' is used for parameters whose default base is hex; 'dd' is used for those whose default base is decimal. Values which should be expressed in hex are displayed in the TTYSET parameter listing preceded by a '\$'. Some examples follow:

```
+++TTYSET, DP=16, WD=63
+++TTYSET, BS=8, ES=3
```

The first example simply lists the current values of all TTYSET parameters on the terminal. The next line sets the depth 'DP' to 16 lines and the terminal width, 'WD' to 63 columns. The last example sets the backspace character to the value of hex 8, and the escape character to hex 3.

The following fully describes all of the TTYSET parameters available to the user. Their initial values are defined, as well as any special characteristics they may possess.

BS=hh BackSpace character

This sets the 'backspace' charcter to the character having the ASCII hex value of hh. This character is initially a 'control H' (hex 08), but may be defined to any ASCII character. The action of the backspace character is to delete the last character typed from the terminal. If two backspace characters are typed, the last two characters will be deleted, etc. Setting BS=0 will disable the backspace feature.

BE=hh Backspace Echo character

This defines the character to be sent to the terminal after a 'backspace' character is received. The character printed will have the ASCII hex value of hh. This character is initially set to a null but can be set to any ASCII character.

The BE command also has a very special use that will be of interest to some terminal owners, such as SWTPC CT-64.

If a hex 08 is specified as the echo character, FLEX will output a space (20) then another 08. This feature is very uesful for terminals which decode a hex 08 as a cursor left but which do not erase characters as the cursor is moved.

Example: Say that you mis-typed the word cat as shown below: +++CAY

typing in one CTRL-H (hex 08) would position the cursor on top of the Y and delete the Y from the DOS input buffer. FLEX would then send out a space (\$20) to erase the Y and another 08 (cursor left) to re-position the cursor.

DL=hh DeLete character

This sets the 'delete current line' character to the hex value hh. This character is initially a 'control X' (hex 18). The action of the delete character is to 'erase' the current input line before it is accepted into the computer for execution. Setting DL=O will disable the line delete feature.

EL=hh End of Line character

This character is the one used by FLEX to separate multiple commands on one input line. It is initially set to a colon (':'), a hex value of 3A. Setting this character to 0 will disable the multiple command per line capability of FLEX. The parameter 'EL=hh' will set the end of line character to the character having the ASCII hex value of hh. This character must be set to a printable character (control characters not allowed).

DP=dd DePth count

This parameter specifies that a page consists of dd (decimal) physical lines of output. A page may be considered to be the number of lines between the fold if using fan folded paper on a hard copy terminal, or a page may be defined to be the number of lines which can be displayed at any one time on a CRT type terminal. Setting DP=0 will disable the paging (this is the initial value). See EJ and PS below for more details of depth.

WD=dd WiDth

The WD parameter specifies the (decimal) number of characters to be displayed on a physical line at the terminal (the number of columns). Lines of text longer than the value of width will be 'folded' at every multiple of WD characters. For example, if WD is 50 and a line of 125 characters is to be displayed, the first 50 characters are displayed on a physical line at the terminal, the next 50 characters are displayed on the next physical line, and the last 25 characters are displayed on the third physical line. If WD is set to 0, the width feature will be disabled, and any number of characters will be permitted on a physical line.

NL=dd NuLl count

This parameter sets the (decimal) number of non-printing (Null) 'pad' characters to be sent to the terminal at the end of each line. These pad characters are used so the terminal carriage has enough time to return to the left margin before the next printable characters are sent. The initial value is 4. Users using CRT type terminals may want to set NL=0 since no pad characters are usually required on this type of terminal.

TB=hh TaB character

The tab character is not used by FLEX but some of the utilities may require one (such as the Text Editing System). This parameter will set the tab character to the character having the ASCII hex value hh. This character should be a printable character.

EJ=dd EJect count

This parameter is used to specify the (decimal) number of 'eject lines' to be sent to the terminal at the bottom of each page. If Pause is 'on', the 'eject sequence' is sent to the terminal after the pause is terminated. If the value dd is zero (which it is by default), no 'eject lines' are issued. An eject line is simply a blank line (line feed) sent to the terminal. This feature is especially useful for terminals with fan fold paper to skip over the fold (see Depth). It may also be useful for certain CRT terminals to be able to erase the previous screen contents at the end of each page.

PS=Y or PS=N PauSe control

This parameter enables (PS=Y) or disables (PS=N) the end-of-page pause feature. If Pause is on and depth is set to some nonzero value, the output display is automatically suspended at the end of each page. The output may be restarted by typing the 'escape' character (see ES description). If pause is disabled, there will be no end-of-page pausing. This feature is useful for those using high-speed CRT terminals

to suspend output long enough to read the page of text.

ES=hh EScape character

The character whose ASCII hex value is hh is defined to be the 'escape character'. Its initial value is \$1B, the ASCII ESC character. The escape character is used to stop output from being displayed, and once it is stopped, restart it again. It is also used to restart output after Pause has stopped it. As an example, suppose you are LISTing a long text file on the terminal and you wish to temporarily halt the output. Typing the 'escape character' will do this (this feature is not supported on computers using a Control Port for terminal communications). At this time (output halted), typing another 'escape character' will resume output, while typing a RETURN key will cause control to return to FLEX and the three plus sign prompt will be output to the terminal. It should be noted that line output stopping always happens at the end of a line.

The TIME command is provided so that the user may read the time on the Time-of-Day clock on the GIMIX 6809 PLUS CPU CARD with the Time-of-Day clock option installed.

DESCRIPTION

The general syntax of the TIME command is:

TIME

TIME takes no parameters and prints the time as read from the clock.

To use the TIME command merely type the following:

+++TIME

The computer output will have the following format:

FRIDAY SEPTEMBER 05, 09:44:41 AM

If the computer responds:

ERROR READING TIME, CLOCK NOT SET

It means that the program detected an invalid value from the clock and the clock needs to be set. To set the time use the SETTIME command.

This command uses the FLEX output routines and therefore the output can be re-directed with any of the FLEX output re-direction command (i.e. 'P', 'O', etc.).

If there is no Time-of-Day clock installed in your system this program may cause the CPU to loop infinitely. If this happens the only way to exit the loop is to press the 'RESET' button on the front panel.

하는데 하는데 이번 사람들이 되었다. 그는데 이번 이번 사람들은 사람들이 되었다. 그는데 이번 보면 수 있는데 이번 모든데 이번 가는데 되었다. 그런데 하는데 그렇다. - 이용된 아이트 및 사람들이 이용되는데 이번 이번 사람들이 되었다. 그는데 이번 사람들이 되었다. 그는데 이번 사람들이 되었다. 그는데 이번 사람들이 되었다.	
이 사용하는 사람들이 하는 사람들이 가장 하는 사람들이 사용하는 하는 이번 하는 사람들이 되었다. 그는 사람들이 되었다.	
그들은 보다 보다가 하시면 하시다면 보면 가장 되었다. 사람이 사람이 사람이 되었다면 하시다면 하시다면 하는데 되었다.	
요즘 보이 하는 그는 사람들은 이 모든 이 사람들이 얼마를 보는 것이 되었다. 그는 사람이 되어 살아 먹었다.	
그리아 되는 원인의 이 마리 이름 가능한다. 그는 아니라 이 사람들은 사람이 되지 않는 것을 받는다.	
일어 많이 다 회사 하는 것으로 가는 이 사람들은 그는 사람들이 하는 그가 그리고 하는데 하는데 되었다.	
화가 가는 것이 되어 되었다. 그런 그리고 그는 사이를 들었다면서 그렇게 하는 것이 되었다. 그는 사이를 받아 되었다.	
보면 하는 것이 되었다. 그런 사람들에 보고 하는 것이 되었다. 그는 사람들에 가는 사람들은 사람들이 되었다. 그런 그는 것이 되었다. 그런 사람들이 되었다. 그렇게 하는 것이 살았다. 그런 사람들이 가득하는 것이 되었다. 그는 사람들이 되었다. 그런 사람들이 가득하는 것이 되었다. 그런 것이 되었다. 그런 것이 되었다. 그런 것이 되었다. 그런 것이 살	
그리다면 하시다니 그는 것이 이번 생산되고 하면 없이 그들은 사람이 되었다. 그런 바다 그리네요.	
성용 보다는 이렇게 하는 다음 아무지 않는 것이 되어 하는데 하는데 하는데 하는데 하다 하다 하는데	
医性性性病 医动物 化反应性 医动物性结束 医内部分别 医眼球 经自己的 医二氏病 电电流通讯 医电流性神经 医皮肤 医皮肤 医皮肤 医神经病 化二甲基基 化环境 医鼻道	
용하는 하는 사용, 이름 하는 경험에 하는 경험을 보면 하는 것이 되었다. 그런 사용을 보고 있다고 보면 하는 이 이 회원을 하게 하는 사용이 되었다. 그런 함께 보고 있다. 1908년 1월 1일	

UPDATE

The UPDATE utility enables the user to change the date in a file's directory entry to the "current date".

DESCRIPTION

The general synatax of the UPDATE command is:

UPDATE, <filespec>

Where <filespec> is the name of the file for which the date is to be changed. If the file extension is not specified, UPDATE defaults to an extension of .TXT. The file's directory entry is changed to reflect the current date. UPDATE does not alter the contents of the file itself.

나는 그들은 사람들이 나는 사람들이 들어 있는 것이 모든 사람들이 되었다. 그는 사람들이 살아 없는 것이 없는 것이 없는 것이다.	
교육(1) 그런 경우에 되는 사람들은 보고 있는 것이 되는 것이 되는 것이 되었다. 그는 것이 되는 것이 되었다. 그런 하는 경우를 하는 것이 하는 것이 되는 것이 되는 것이 되는 것이 되는 것이 되는 것이 되는 것이 되었다.	
사용 발표하는 경기에 가는 경기를 보고 있는 것이 되는 것이 되었다. 그 사람들은 사람들이 되었다. 그는 것이 되었다. 대통령 이번 경기를 보고 있는 것이 되었다. 그는 것이 되었다. 그는 것이 되었다. 그는 것이 되었다.	
시 얼마에 비용하는 아들은 그들이 그렇게 되었다. 왕도 하는 것 때	
그리 얼마 보고 있는 어머니의 선명을 가게 된다는 선생님, 선생님이 아니다 있다.	
되었다고 말하다 모든 경험에 가면 없다는 그래요 중에 다면서 다가 하는 그리고 다	
마마 등 등에 하고 있다. 그는 마마와 문자는 등 마음을 하면 하다는 그런 그는 마만 그는 바람들은 경기를 받는 것은 것이 되었다. 그 그는 이 사용을 하고, 이 사람들은 기를 받는 것을 보는 것은 나는 사용을 보는 것이 되었다.	

UNSNARL

The UNSNARL command is used to reorganize the free chain on a FLEX disk (the list of all unused sectors). This helps reduce fragmenting of files on the disk, and improves access times.

DESCRIPTION

The general syntax of the UNSNARL command is:

UNSNARL, (work drive), (backup drive)

where <work drive > is the number of the drive with the disk to be reorganized, and <backup drive > is the number of the drive where the FREEMAP.TMP file can be stored. This must be different from the work drive and defaults to drive 0. UNSNARL will then read all the sectors in the free chain of the indicated disk, and make a list of all the segments in the chain.

A segment is a sector or group of sectors linked in logical order. On a newly formatted disk there is only one segment, which has all the sectors on the disk. As files are created and deleted, this segment is broken into smaller and smaller segments, and the links among them become more and more random. This results in files being stored in fragments scattered over the disk, and increases access times, especially for random-access files.

UNSNARL scans the free chain and creates a list of all the segments in the free chain. For insurance this list is saved on the backup disk as the FREEMAP.TMP file. Then UNSNARL sorts the list of segments in ascending order of disk address. Next the sector link in the last sector of each segment is pointed to the first sector of the next segment. This frequently causes several small segments to be merged into one large segment. Finally the pointers to the start and end of the free chain in the System Information Record are corrected. At each step, UNSNARL displays a descriptive message on the console. Example:

+++UNSNARL, 1, 0
READING FREE CHAIN
FREE CHAIN READ, NOW SAVING MAP
MAP SAVED, NOW SORTING EXTENT LIST
LIST SORTED, NOW RELINKING FREE CHAIN
RELINK DONE, DELETING MAP FILE

+++

The FREEMAP.TMP file is created as insurance against power glitches or other interruptions. If UNSNARL is interrupted while it is relinking the free chain, the free chain will be left in a confused state, and creating or deleting files on that disk would be impossible. FREEMAP.TMP makes it possible for UNSNARL to pick up where it left off. The UNSN1 command is the same as UNSNARL,

시계들이 그리지 그렇게 하는 나의 아래의 불제를 통해가 많아 사용하고 하면 하고 있다. 그 중에 다시 개발되다는 이 전하고 있는다.	
그렇다. 그 그 그는 그는 그 이번 맛이 많아 된 전쟁에서 하는 그 맛이 없었다면 하다가 하는 것이다. 그는	
그리다는 이 이번 이 시로시간 중요요. 그리는 여러 이글 시간을 하시면 하고 있다면 하는데, 그는 그리고 한국 시간을 하는데 하나 모다.	
그렇다 보면 들어 하고 이번 하는 사람이 하다 나는 사람들을 하는 것은 생생들이 얼마나 하는데 나는 이번 모르게 하셨다는데 하는데 되었다.	
[1886년: 1984년: 1984년: 1987년 1일 : 1984년: 1984년 1984	
[12] 14년 1월 19 등 대통령 전문 등 시민국 기업 1년 대통령 19 대학교 교육 19 대학교 1 대	
단통하면 보고 회의 경우 이 사람들이 하고 하는 것도 하는데 되는데 하는데 그 사람들이 되었다. 그 없는데 되는데 그렇게 되었다.	
어느 것 하면 없는 아니다. 시작하는 사이 가지가 아버지는 아니는 사이를 가는 사이를 가고 있다. 아니는 사이다.	

except that instead of reading the free chain to create the list of segments, it reads the list from the FREEMAP.TMP file. Once UNSNARL or UNSN1 successfully completes the relinking, the FREEMAP.TMP file is deleted.

UNSNARL has no effect whatever on files, so a fragmented file will remain fragmented until it is rewritten onto an unfragmented segment of free chain. Thus to keep fragmentation to a minimum UNSNARL should be used periodically, in order to clean up the fragmentation left by formerly fragmented files now using segments from the reorganized free chain.

UNSNARL only reduces file fragmentation, it does not eliminate it. The only way to eliminate fragmentation entirely is to format a new disk and perform a sequential copy of all files to the blank disk. This method is practical for floppies, but not for high capacity hard disks such as the GIMIX Winchester Disk Subsystems. Therefore GIMIX has developed the UNSNARL command as a way for users of our hard disk systems to avoid the degradation of system performance caused by severe file fragmentation.

NOTE: the degree of fragmentation in the free chain, and therefore the need for running UNSNARL, can be determined with the FREEMAP command.

그는 하고 있는 그는 그는 그를 살아가 있다면서 하고 말라면서 하는 것 때가 하고 말을 받을 때가는 말라면 생각하고 말라는 사람들이 없다고 하는	
물활동하면 용계 된다. 이 모든 이 자연 건강이 그는 발표에 이 아픈 전환, 제가 전환계 교육하는 일본, 전 보는 소속 등이 모든	
그런 그렇게 보는 이 전에 그려고 가를 가 먹는 것이 되었다. 그는 일본에 하는 모든 그들은 그리는 그리고 하는 모든 모든 그리고 그는 그리고 그는 그리고 그는 그리고 그는 그리고 그는 그리고 그는 그리고 그리고 그는 그리고	
어머물에는 나라도 된 그 하다. 그들이 다음에 가득 보고를 가는 수 있다. 그는 그는 이 등에 들어 내려가 나를 다 했다.	
왕선의 일반에 하는 데 가는 가는 사고 있다. 항상에 보고 하는 네트를 다 하는 사람들이 되었다. 그는 다른 그는 다른 사람들이 되었다.	
경험도 되었다.	
한 사람들은 마음이 나는 사람들은 아이들 아니는 사람들이 가지 않는 것이 하는 것이 되었다. 그는 사람들은 사람들이 다른 사람들이 되었다.	
그는 바람이 하고 아이들은 이번 하다가 하고 하는 것이 하는 사람이 되었다. 그는 사람들이 하다 하는 사람이 되었다.	
보는 하는 모든 모든 사람들이 모르는 아내를 하는 것이 되는 것이 되는 것이 되었다. 그 이렇게 되어 되었다고 있다고	
그는 그는 사람들이 되었다면 하면 되었다. 그는 사람이 있는 그리를 되어 되는 것들은 사람들이 되었다. 이 사람들은	
선생님이 아이들에게 하는데 하는 사람들이 나는데 아이들은 사람들이 나는 사람들이 다른데 하는데 되었다.	
지방 하는 어느 어느 이 이 이 중요 이 이 일이 있는 이 것이 있는 생생이다고 있는 사람이 되는 것이 되어 있다. 그는 이 생각이다.	
아이들이 있다. 그는 그 이 이 얼마 아이들이 나왔다면 하는 일반에 가지 않았다면 나는 것은 것 같아. 하는 일반이 되었다.	
지수는 가장 그는 문제들이 가는 하셨습니까 그 그 가게 하는 것 같은 것 같아. 그 그는 얼, 나는 그 모양하는	

VERIFY

The VERIFY command is used to set the File Management System's write verify mode. If VERIFY is on, every sector which is written to the disk is read back from the disk for verification (to make sure there are no errors in any sectors). With VERIFY off, no verification is performed.

DESCRIPTION

The general syntax of the VERIFY command is:

VERIFY[,ON]
 or
VERIFY[,OFF]

where ON or OFF sets the VERIFY mode accordingly. If VERIFY is typed without any parameters, the current status of VERIFY will be displayed on the terminal. Example:

+++VERIFY,ON +++VERIFY

The first example sets the VERIFY mode to ON. The second line would display the current status (ON or OFF) of the VERIFY mode. VERIFY causes slower write times, but it is recommended that it be left on for your protection.

VERSION

The VERSION utility is used to display the version number of a utility command. If problems or updates ever occur in any of the utilities, they may be replaced with updated versions. The VERSION command will allow you to determine which version of a particular utility you have.

DESCRIPTION

The general syntax of the VERSION command is:

VERSION, <file spec>

where <file spec> is the name of the utility you wish to check. The default extension is CMD and the drive defaults to the working drive. As an example:

+++VERSION, O. CAT

would display the version number of the CAT command (from $\mbox{drive 0}$) on the terminal.

TUOX

 ${\sf XOUT}$ is a special form of the delete command which deletes all files having the extension .0UT.

DESCRIPTION The general syntax of XOUT is:

XOUT[,<drive spec>]

where <drive spec> is the desired drive number. If no drive is specified all, .OUT files on the working drive will be deleted and if auto drive searching is enabled, all .OUT files on drives 1 and 2 will be deleted. XOUT will not delete any files which are delete protected or which are currently in the print queue.

Example: +++XOUT +++XOUT 1

en en de la transporte de la proposition de la companya de la companya de la companya de la companya de la com La companya de la co

The YEAR command is used to display or change the year in the internal FLEX date register. This command is used when FLEX is patched to load the current day and month from the Time-of-Day clock on the GIMIX 6809 CPU board (see the section on patching FLEX to use the Time-of-Day clock). The YEAR command should be included in the STARTUP file to set the year when the system is booted.

DESCRIPTION

The general syntax of the YEAR command is:

YEAR, [YY]

Where YY is the last two digits of the current year. If no year is entered the current year in the FLEX system date area will be printed.

To use the YEAR command type the following:

+++YEAR

or

+++YEAR,81

The first example prints the year in the FLEX date register. The second example sets the year to 1981.

The error message:

INVALID YEAR IN INPUT LINE

Indicates that an illegal value was entered for the year [YY].

그 성능 주시는 경우 이 보면도 그렇게 하면 이 얼마는 것 같아 나무지 않는 것이 되었다.	
그림, 종료 회에 있었다. 그리고 그는 그를 모르고 하루에 하는데 만든 말을 했다. 그런 나이를	
	원하네요 그 그렇게 나는 얼마다
중화물을 맞는데 하는 사람들이 하는 것이 되었다. 그 아이들이 되는 것은 사람들이 되었다.	
사일 가능했다고 있다. 그 그 사이는 사람들은 사람들은 그리고 있다. 그는 어디다.	
병에 가게 되는 사이 되는 사람들은 그는 사람들이 이번 가게 되었는데 살이 먹는 것이다.	
요즘이 되었다면요	
가득하다면 하고 있는 이 사람이 하는 사람들이 보고 있는 사람들이 되었다. 그리고 있는 사람들이 되었다. 그리고 있다면 보다 되었다. 그리고 있다면 보다 되었다면 보다 되었다면 보다 되었다면 보다 보다 되었다면	
보인함한다. 그 이 이 보는 어디에 다른 이 보고 보고 있는 다른 사람들이 보고 있는 것 같다.	
불위하는 이 여전으로 보는 물리는 강 있는 그 그 나는 그리고 하면 되어 하고 있는 그렇다는	
기존 마음 아이는 사람들이 바다 하나 나를 받는 사람들이 가지 않는데 가는 아이들이 되었다. 그	
불빛하는 조선은 학생이 그 물이 많다면 내가 있는 소리에 얼마나 가입니다. 그는 그 모든 것	
오르는 이렇게, 이번 생생들 수에 먹어나 이 이 이 이번 보면 없으면서 살이 하였다.	
민만, 그렇게 그림을 만하는 얼굴에 맞을 보고 이 전체 (19 등 4의 하고 하는 글로봇) :	
그는 하는 그는 이 맛이 되는데 얼마 있는데 나를 이 하는데 뭐라면 하는데 말하는데 하는데 나는	

The Y utility enables the user to automatically answer "Y" (yes) to

"Y or N" prompts from other utilities. The Y utility is especially useful when writing EXEC files.

DESCRIPTION

The general synatax of the Y command is:

Y, (command string)

Where (command string) is a valid command line to be executed. If Y is used in a line with multiple commands, using the end of line character, it only affects the command immediately following it. Some examples follow:

+++Y, COPY, 0, 1 +++Y, DELETE, TESTFILE. CMD

The first example will copy all files from drive #0 to drive #1, automatically answering "Y" (yes) to any "DELETE ORIGINAL?" and "ARE YOU SURE?" prompts that occur because of duplicate files on the two disks. The second example will delete the specified file, automatically answering "Y" (yes) to the "DELETE <file spec.>?" and "ARE YOU SURE?" prompts. Use caution when using the Y utility, especially when files are being deleted, since it bypasses the normal protection against unintentionally deleting the wrong file.

사이를 하시는 것으로 보고 있다. 그런
그리다 살 전에 가는 얼마나라는 그리는 이 선생님 하늘에 가
리스타일로 시민을 전 이번 시민 사람들은 기를 보고 있는 것이다.
그리는 일반 사람은 얼마는 이 아이를 보고 있다. 그 나는 사람이 없는 것이 없다.
[문문일시] 김보리는 경험 기업을 하는 경험 등을 보고 있다.
함께는 생기를 하는데 하는데 보고 살아왔다. 하는데 나를 다니다.
기본 등은 전경으로 하는 그리고 얼굴을 들었다. 그리고 그는 사람들은 그 그 없는 그 그를 다 살아 보다 되었다.
경우 등 경우 전 경우 보고 있는 것 같은 경우 등 경우 등 보고 있다.
으로 보려는 것이라는 것으로 모르는 이렇게 되는 것이 되고 있어요. 그런 그를 받는 것이 없는 것이다. 보다면서 많은 사람이 있는 것이라고 있는 것이다. 그 것이 나를 하는 것이 하고 있다면 하는 것이다.
생각하는 경우 사람들은 사람들은 사람들은 사람들은 사람들이 되었다.

GENERAL SYSTEM INFORMATION

I. DISK CAPACITY

Each sector of a FLEX disk contains 252 characters or bytes of user data (4 bytes of each 256 byte sector are used by the system). Thus a single-sided mini disk has 340 sectors or 85,680 characters or bytes of user information. A single-sided full size disk has 1140 sectors or 287,280 bytes of user data. Double-sided disks would contain exactly twice these amounts.

II. WRITE PROTECT

Floppy disks can usually be physically write protected to prevent FLEX from performing a write operation. Any attempt to write to such a disk will cause an error message to be issued. It is good practice to write protect disks which have important files on them.

A mini disk can be write protected by placing a piece of opaque tape over the small rectangular cutout on the edge of the disk. Full size floppys are just the opposite. In order to write protect a full size disk, you must remove the tape from the cutout. In other words, the notch must be exposed to write protect the disk. Some full size disks do not have this cutout and therefore cannot be write protected.

III. THE 'RESET' BUTTON

The RESET button on the front panel of your computer should NEVER BE PRESSED DURING A DISK OPERATION. There should never be a need to 'reset' the machine while in FLEX. If the machine is 'reset' and the system is writing data on the disk, it is possible that the entire disk will become damaged. Again, never press 'reset' while the disk is operating! Refer to the 'escape' character in TTYSET for ways of stopping FLEX.

IV. NOTES ON THE P COMMAND

The P command tries to load a printer driver file named PRINT.SYS from the same disk which P itself was retrieved. For the requirements of this file and on writing your own custom PRINT.SYS file, see the section on such later in this manual or consult the 'Advanced Programmer's Guide'.

V. ACCESSING DRIVES NOT CONTAINING A DISKETTE

If an attempt is made to access a minifloppy not containing a diskette, the system will hang up attempting to read until a disk is inserted and the door closed. Alternatively, you could reset the machine and begin execution at the warm start location \$CDO3.

VI. SYSTEM ERROR NUMBERS

Any time that FLEX detects an error during an operation, an appropriate error message will be displayed on the terminal. FLEX internally translates a derived error number into a plain language statement using a look-up table called ERROR.SYS. If you have forgotten to copy this .SYS file onto a disk that you are using, FLEX will report a corresponding number as shown below:

DISK ERROR #xx

where 'xx' is a decimal error number. The table below is a list of these numbers and what error they represent.

ERROR	#	MEANING
1 2		ILLEGAL FMA FUNCTION CODE ENCOUNTERED THE REQUESTED FILE IS IN USE
3		THE FILE SPECIFIED ALREADY EXISTS
4.		THE SPECIFIED FILE COULD NOT BE FOUND
5		SYSTEM DIRECTORY ERROR-REBOOT SYSTEM
6		THE SYSTEM DIRECTORY IS FULL
7		ALL AVAILABLE DISK SPACE HAS BEEN USED
8		READ PAST END OF FILE
9		DISK FILE READ ERROR
10		DISK FILE WRITE ERROR
11		THE FILE OR DISK IS WRITE PROTECTED
12		THE FILE IS PROTECTED-FILE NOT DELETED
13		ILLEGAL FILE CONTROL BLOCK SPECIFIED
14		ILLEGAL DISK ADDRESS ENCOUNTERED
15		AN ILLEGAL DRIVE NUMBER WAS SPECIFIED
16		DRIVE NOT READY
17		THE FILE IS PROTECTED-ACCESS DENIED
18		SYSTEM FILE STATUS ERROR
19		FMS DATA INDEX RANGE ERROR
20		FMS INACTIVE-REBOOT SYSTEM
21		ILLEGAL FILE SPECIFICATION
22		SYSTEM FILE CLOSE ERROR
23		SECTOR MAP OVERFLOW-DISK TOO SEGMENTED
24		NON-EXISTENT RECORD NUMBER SPECIFIED
25		RECORD NUMBER MATCH ERROR-FILE DAMAGED
26 27		COMMAND SYNTAX ERROR-RE-TYPE COMMAND
28		THAT COMMAND IS NOT ALLOWED WHILE PRINTING WRONG HARDWARE CONFIGURATION
20		WKUNG HAKDWAKE CUNFIGUKATIUN

For more details concerning the meanings of these error messages, consult the 'Advanced Programmer's Guide'.

VII. SYSTEM MEMORY MAP

The following is a brief list of the RAM space required by the FLEX Operating System. All address are in hex.

0000 - BFFF User RAM

*Note: Some of this space is used by

NEWDISK, COPY and other utilities.

C000 - DFFF Disk Operating System

CO7F System stack

C100 - C6FF Utility command space

CD00 FLEX cold start entry address CD03 FLEX warm start entry address

For a more detailed memory map, consult the 'Advanced Programmer's Guide'.

FLEX User's Manual

VIII. FLEX OPERATING SYSTEM INPUT/OUTPUT SUBROUTINES

In order for the FLEX I/O functions to operate properly, all user program character input/output subroutines should be vectored thru the FLEX operating system rather than the computer's monitor. Below is a list of FLEX's I/O subroutines and a brief description of each. All given addresses are in hexadecimal.

GETCHR at \$CD15

This subroutine is functionally equivalent to S-BUG's character input routine. This routine will look for one character from the control terminal (I/O port #1) and store it in the A accumulator. Once called, the input routine will loop within itself until a character has been input. Anytime input is desired, the call JSR GETCHR or JSR \$CD15 should be used.

GETCHR automatically sets the 8th bit to 0 and does not check for parity. A call to this subroutine affects the processor's registers as follows:

ACC. A loaded with the character input from the terminal B,X,Y,U not affected

PUTCHR at \$CD18

This subroutine is used to output one character from the computer to the control port (I/O port #1). It is functionally equivalent to the output character routine in S-BUG.

To use PUTCHR, the character to be output should be placed in the A accumulator in its ASCII form. For example, to output the letter 'A' on the control terminal, the following program should be used:

LDA #\$41 JSR \$CD18

The processor's registers are affected as follows:

ACC. A changed internally B,X,Y,U not affected

PSTRNG at \$CD1E

PSTRNG is a subroutine used to output a string of text on the control terminal. When address \$CD1E is called, a carriage return and line feed will automatically be generated and data output will begin at the location pointed to by the index register. Output will continue until a hex 04 is seen. The same rules for using the ESCAPE and RETURN keys for stopping output apply as described earlier.

The accumulator and register status after using PSTRNG are as follows:

ACC. A Changed during the operation

ACC. B Unchanged

X Contains the memory location of the last character read from the string (usually the O4 unless stopped by the ESC key)

Y,U Unchanged

NOTE: The ability of using backspace and line delete characters is a function of your user program and not of the FLEX I/O routines described above.

For additional information consult the 'Advanced Programmer's Manual'.

STAT at \$CD4E

This routine is used to determine the "status" of the input device. That is, to see if a character has been typed on the input terminal keyboard. Its function is to check for characters such as the ESCAPE key in FLEX which allows breaking of the output. This routine returns an EQual condition if no character was hit and a Not-Equal condition if a character was hit. No registers, except for the condition codes, may be altered.

IX. BOOTING THE FLEX DISK OPERATING SYSTEM

In order to read FLEX from the system disk upon powering up your system, you must have a short program in RAM or ROM memory. This program is called a 'bootstrap' loader.

If you are using a Southwest Technical Products disk system and the S-BUG monitor, there are bootstraps stored in this ROM which you can use. They are executed by simply typing a 'D' for the full size floppy or a 'U' for the mini floppy.

Those users of other hardware or monitor ROM should use the boot supplied with the hardware if compatible with FLEX. A sample boot (for the SWTPc mini system) is given here for reference.

If the system does not boot properly, re-position the system disk in the drive and re-execute the bootstrap loader.

0100 B6 0103 86 0105 B7	E018 00 E014	START	LDA LDA STA	COMREG #0 DR VREG	TURN MOTOR ON
0108 8E 010B 3D	0000	OVR	LDX MUL	#0000	DELAY FOR SPEED UP
0100 30	1F		LEAX	-1,X	
010E 26	FB		BNE	OVR	
0110 C6	OF .		LDB	#\$0F	RESTORE
0112 F7	E018		STB	COMREG	
0115 8D	2B		BSR	RETURN	
0117 F6	E018	L00P1	LDB	COMREG	
011A C5	01		BITB	#1	
011C 26	F9		BNE	LOOP1	
011E 86	01		LDA	#1	
0120 B7	E01A		STA	SECREG	
0123 8D	1D		BSR	RETURN	
0125 C6	38		LDB	#\$8C	READ WITH LOAD
0127 F7	E018		STB	COMREG	
012A 8D	16		BSR	RETURN	
012C 8E	C000		LDX	#\$C000	
012F C5	02	LOOP2	BITB	#2	DRQ?
0131 27	05		BEQ	LOOP3	
0133 B6	E01B		LDA	DATREG	
0136 A7	80		STA	0,X+	
0138 F6	E018	L00P3	LDB	COMREG	DUCYO
013B C5	01		BITB	#1	BUSY?
013D 26	F0		BNE	LOOP2	
013F 7E	C000		JMP	\$C000	
0142 8D	00	RETURN	BSR	RTN	
0144 39		RTN	RTS		

X. REQUIREMENTS FOR THE 'PRINT.SYS' PRINTER DRIVER

FLEX, as supplied, includes a printer driver that will work with most parallel type printers, such as the SWTPC PR-40. If desired, the printer driver may be changed to accomodate other types of printers. Included is the source listing for the supplied driver. Additional information on the requirements for the PRINT.SYS driver can be found in the Advanced Programmer's Guide.

- 1) The driver must be in a file called PRINT.SYS
- 2) Three separate routines must be supplied, a printer initialization routine (PINIT at \$CCCO), a check ready routine (PCHK at \$CCD8), and an output character routine (POUT at \$CCE4).
- 3) When the POUT routine is called by FLEX, the character to be output will be in the A accumulator. The output routine must not destroy the B, X, Y, or U registers. PINIT may destroy any registers. PCHK may NOT alter any registers.
- 4) The routines MUST start at the addresses specified, but may be continued anywhere in memory if there is not room where specified. If placed elsewhere in memory, be certain they do not conflict with any utilities or programs which will use them.
- 5) All three routines must end with a return from subroutine instruction (RTS).

		* * PRINT *	.SYS PI	A DRIVERS	FOR GENERAL CASE PRINTER
	E01C	PIA	EQU	\$E01C	PIA ADDRESS FOR PORT #7
		*			
		* PRINT	ER INIT	IALIZATION	(MUST BE AT \$CCCO)
CCCO CCCO 86 CCC2 B7 CCC5 86 CCC7 B7 CCCA 86 CCCC B7 CCCF 39	3A E01D FF E01C 3E E01D	PINIT	ORG LDA STA LDA STA LDA STA RTS	\$CCCO #\$3A PIA+1 #\$FF PIA #\$3E PIA+1	MUST RESIDE AT \$CCCO SELECT DATA DIRECTION REG. BY WRITING O IN DDR CONTROL SELECT ALL OUTPUT LINES PUT IN DATA DIRECTION REG. SET UP FOR TRANSITION CHECKS AND ENABLE OUTPUT REGISTER
CCD0 7D CCD3 73 CCD6 39	E01C CCE3	* PRINT PREADY	ER READ TST COM RTS	Y ROUTINE PIA PFLAG	RESET PIA READY INDICATION SET THE PRINTER READY FLAG

		*			
		* CHECK	FOR PR	INTER READ	Y (MUST BE AT \$CCD8)
CCD8			ORG	\$CCD8	PRINT TEST AT \$CCD8
CCD8 7D	CCE3	PCHK	TST	PFLAG	TEST FOR PRINTER READY
CCDB 2B	05		BMI	PCHK X	IF NEGATIVE, PRINTER READY
CCDD 7D	EO1D		TST	PIA+1	CHECK FOR TRANSITION
CCEO 2B	EE		BMI	PREADY	IF MINUS, PRINTER NOW READY
CCE2 39		PCHKX	RTS		
		* PRINT			
CCE3 FF		PFLAG	FCB	\$FF	PRINTER READY FLAG
		*			
			CO OUTO	UT OUADAGE	ED DOUTING (MUCT DE AT COCA)
		* PRINI	ER OUTP	UI CHARACI	ER ROUTINE (MUST BE AT \$CCE4)
CCE4			ORG	\$CCE4	MUST RESIDE AT \$CCE4
CCE4 8D	F2	POUT	BSR ·	PCHK	TEST FOR PRINTER READY
CCE6 2A	FC	P001	BPL	POUT	LOOP UNTIL PRINTER READY
CCE8 7F	CCE3		CLR	PFLAG	SET PRINTER FLAG NOT READY
CCEB B7	EO1C		STA	PIA	SET DATA IN OUTPUT REGISTER
CCEE 86	36		LDA	#\$36	SET DATA IN COTFOT RECTSTER SET DATA READY, HIGH TO LOW
CCFO 8D	02		BSR	POUTB	
CCF2 86	3E		LDA	#\$3E	THEN SEARCH FOR TRANSITION
CCF4 B7	EO1D	POUTB	STA	PIA+1	OF LOW LEVEL TO HIGH LEVEL
CCF7 39			RTS		

Sample Drivers for Serial Printer

The following listing is a sample set of drivers for a serial type printer using an ACIA as its interface. This set of drivers is not supplied on disk. In order to use these drivers, you must type in the source and assemble it. If you have a serial printer, you will probably want to replace the parallel PRINT.SYS file on the disk with one containing these drivers.

1 2 3 4					IVERS FOR EQUATE IF	GENERAL SERIAL PRINTER NECESSARY
5 6 7		E01C	ACIA	EQU	\$E01C	ACIA ADDRESS FOR PORT #7
8 9			* PRINT	ER INIT	IALIZATION	(MUST BE AT \$CCCO)
10 11 12	CCC0 86	13	PINIT	ORG LDA		MUST RESIDE AT \$CCCO RESET ACIA
13 14 15	CCC2 B7 CCC5 86 CCC7 B7	E01C 11 E01C		STA LDA STA		SET 8 BITS & 2 STOP
16 17	CCCA 39	EUIC		RTS	ACIA	RETURN
18 19			* * CHECK *	FOR PR	INTER READ	Y (MUST BE AT \$CCD8)
20 21 22 23 24 25	CCD8 CCD8 34 CCDA F6 CCDD 56 CCDE 56	04 E01C	PCHK	ORG PSHS LDB RORB RORB	\$CCD8 B ACIA	PRINT TEST AT \$CCD8 SAVE B ACC. GET STATUS GET TDR BIT INTO SIGN POSITION
26 27 28	CCDF 56 CCEO 35 CCE2 39	04		RORB PULS RTS	В	RESTORE B ACC. RETURN
29 30			*		UT 0UADA0T	
31 32			* PRINII	ER OUTP	UI CHARACII	ER ROUTINE (MUST BE AT \$CCE4)
33 34 35 36 37 38 39	CCE4 CCE4 34 CCE6 F6 CCE9 57 CCEA 57 CCEB 24 CCED 35	04 E01C F9 04	POUT POUT2	ORG PS HS LDB ASRB ASRB BCC PULS	\$CCE4 B ACIA POUT2 B	MUST RESIDE AT \$CCE4 SAVE B ACC. GET STATUS GET TDR BIT INTO CARRY LOOP IF NOT READY RESTORE B ACC.
40 41 42 43	CCEF B7 CCF2 39	E01D		STA RTS END	ACIA+1	WRITE OUT THE CHAR. RETURN

The system should be configured for GMXBUG-09 or SBUG-E as required.

GIMIX 6809 CPU BOARD

- 1. Select the desired CPU speed. See the CPU board manual for more information. When running at 1.5 or 2MHz the slow I/O must be enabled as on the GIMIX 6809 MOTHER BOARD. Please note that only the 68B09 will run at 2MHz.
- 2. If print spooling is to be used configure the 6840 programmable timer by jumpering clock gate #1 to ground and connecting the timer to the 'IRQ' line.

GIMIX DOUBLE DENSITY DISK CONTROLLER

1. See the disk controller hardware manual for information on on how to configure the GIMIX DOUBLE DENSITY DISK CONTROLLER.

MEMORY ADDRESSING

FOR GMXBUG-09

1. Address 8k of memory for \$C000 to \$DFFF and the remainder starting at \$0000 and up to \$BFFF. GIMIX FLEX 2.0 requires 16k of memory starting at \$0000 and ending at \$3FFF in addition to the 8k starting at \$C000 and ending at \$DFFF.

FOR SBUG-E

1. Address all of your memory starting at \$0000 and continuing through \$DFFF. Please note that GIMIX FLEX 2.0 will not support the SWTPC 128K memory arrays unless they are initialized before bootstrapping the system. You must have at least 24K of memory available to the system to run GIMIX FLEX 2.0.

그들이 하는 것 같아 이렇게 하면 살아 있다면 하는데 되었다. 그들은 사람들이 하는데	
등 (Particle) 이 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은	
그림인하다 그 그는 다음을 하셨다. 이 모든 아름은 아름은 얼마를 보는 사람들이 살아 있다는 것이 얼마나 하는데 그렇게 되었다. 그는 나는 나는	
이 문장들은 보는 이번에 보는 보면 살고 있는 생생님의 전 회원들은 보다는 이 없는 것이다. 이 사람들은 모든 사람들은 모든 사람들은 살아 없다.	
그 다른 아들은 그 그는 그는 그들은 아이들은 살이 되는 것이 되었다. 그는 그는 이번 살아 먹는 것이 되었다.	
그 나는 말이 되는 아이는 아이를 가지 않는 것이 되어 가는 그가 하는 것 같아 한 점점을 가지 않는데 되었다.	
그리는 그는 사람들은 어느 사람들에게 되었다. 그렇게 하는 것은 사람들은 사람들이 가는 사람들이 되었다. 그는 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은	

COMMAND SUMMARY

APPEND, <file spec>[, <file list>], <file spec> Default extension: .TXT Description page: A.1 ASN[, W=\drive\][, S=\drive\] Description page: A.2 BUILD, <file spec> Default extension: .TXT Description pge: B.1 BACKUP, (source drive), (destination drive)[, CPU speed] Description page: B.2 CAT[, \(drive list \)][, \(match list \)] Description page: C.1 COPY, <file spec>, <file spec> COPY, <file spec>, <drive> COPY, \(\drive\), \(\drive\)[, \(\drive\)] Descripyion page: C.2 CLEAN Description page: C.3 CHECKSUM[, <drive spec>] Description page: C.4 CMPBIN, <file spec>, <file spec> Description page: C.5 DATE[, <mm, dd, yy>] Description page: D.1 DELETE, <file spec>[, <file list>] Description page: D.2 DCOPY, \(\drive\), \(\drive\), \(\lambda\), \(\lambda\), \(\lambda\), \(\lambda\) Description page: D.3 EXEC, <file spec> Default extension: .TXT Description page: E.1 EXTEND[, \(\drive \) spec \(\), \(\text{parameter} \)] Description page: E.2 FORMAT[, <drive spec>] Description page: F.1 FIXDAY Description page: F.2 FREEMAP, <drive> Description page: F.3

	이 없이는 얼마리는 번뜩했다.
그 본 방계 문에 그리고 그 말을 다는 말을 내려 있었다. 그런 말 것은 그를 그 살이 있는 말이 모르는 그리는 그리고 그리고 있다.	
그는 주민이 그림으로 만든 이 바람들이 보다 되는 사이를 하는 것을 모든 사이를 받는 것이 없다고 있다.	
그렇게 들어 없는 사람들이 하루 바라 가게 되어 되는 이 경면 얼굴을 다 어떻게 되었다. 그는 목	
그리즘의 아이들의 얼마를 하는 것이 되는 것이 되는 아이들은 그리는 사람들이 되었다.	
이 선물 등에도 되어 들어 가장 이렇게 되는데 이번 그래요 하는데 이번 그리고 살아갔다.	
그들은 이 전에 많은 보이 말하는 그렇지만 있다. 이번 하루 보이 보는 집은 집은 그리고 있는 것은 것	
	Signature of the second
그래마의 양병 사람들은 사람들이 가득하는 사람들이 되었다. 그는 사람들이 살아 살아 있다는 사람들이 되었다.	

GET, <file spec>[, <file list>]
Default extension: .BIN
Description page: G.1

HARD<DRIVE #>

Description page: H.1

HARDFORM

Description page: H.2

I, (file spec), (command)
 Default extension: .TXT
 Description page: I.1

JUMP, (hex address)
Description page: J.1

LINK, <file spec>

Default extension: .SYS Description page: L.1

LIST, \(\file \text{spec}\)[,\(\lambda\)] range\)[,\(\N\)]

Default extension: .TXT

Description page: L.2

MON

Description page: 1.7

NAME[, <drive spec>]
Description page: N.1

N, (command string)
Description page: N.2

O, <file spec>, <command>
Default extension: .OUT
Description page: 0.1

P, <command>
Description page: P.1

PRINT, <file spec>
Default extension: .OUT
Description page: P.2

PROT, <file spec>[,(options)]
Description page: P.3

QCHECK
Description page: Q.1

RENAME, <file spec>, <file spec 2>
Default extension: .TXT
Description page: R.1

그는 전문 경기 그는 경기 가장 말라고 있다. 그는 사람들은 가장 하는 것이 되었다. 그는	
그는 진행하다는 얼굴되었다면 한 모두 중에 하는 것은 하는 것은 말을 만들어 하는 것이다.	
그는 그렇게 살아보니 아이들이 되었다면 하는데 그는 그는 그는 그들은 그는 그를 가는 것이 없다는데 없다.	불통을 하는 것 하면 하는 사람들은
보는 병원들은 사람들은 사람들은 사람들이 되었다. 그는 사람들은 사람들은 사람들이 되었다.	
불통하는 사람들이 아니는 그 말에 가는 이 나는 사람이 하는 이 같은 그리고 모양하는	
마니, 마시, 마시, 보니는 마음 사람들은 그리다 하는데 되었다면 하는데 되었다.	
요즘이 하는 이 용식을 봤는 하는 중에 하는 사람들은 그는 그는 것은 사람이 되었다.	
보기로 보는 이렇게 한국 보다로 하는 사람들이 되었다. 그렇게 되었다면 하는 것이 없는데 그렇게 되었다.	
한 동안 보다는 모양한 바로 있는데 기반에 하는 그는 그 전에 만든 맛들을 모르겠다면 하는데 있다.	
그런 사람이 생물을 보고 하는데 되고 있는데 보고 있다. 그런 이 사람들이 가장하는 사람들이 가장하는데 하는데 보고 있다. 이 사람들은 생물을 가장하는데 있어 되는 사람들은 사람들이 되었다. 그 사람들이 사람들이 되었다면 하는데 사람들이 되었다.	
그 보고 물론 경우 있는 아이들의 그리는 사람들이 되는 것이 되고 있는 것이 없는 것이다.	
현실 보는 사람들 바다 가는 물리에 많은 그는 말라면 하면 하는데, 그렇게 보고 하는 그들은 그들은 사람들이 되었다.	보다면 한 사람들이 되고 있다.
	edit film in terresis accidentation and a film of the contract

REPORT Description page: R.2

SAVE, <file spec>, <begin adr>, <end adr>[, <transfer adr>]
 Default extension: .BIN
 Description page: S.1

STARTUP

Description page: S.2

SETTIME

Description page: S.3

SETUP[, <drive>[options][, <drive[options]]]
 Description page: S.4</pre>

SYSGEN

Description page: S.5

TTYSET[, cparameter list>]
 Description page: T.1

TIME

Description page: T.2

UPDATE, <file spec>

Default extension: .TXT Description page: U.1

UNSNARL, (work drive), (backup drive)
Description page: U.2

VERIFY[, (ON or OFF)]
Description page: V.1

VERSION, <file spec>

Default extension: .CMD Description page: V.2

XOUT[, (drive spec)]
Description page: X.1

YEAR[, YY]

Description page: Y.1

Y, (command string)

Description page: Y.2

지방으로 그 이 이 이 아들에 불편하셨다. 하루 시간 사람이 되었는 그는 그들이 얼마나 있다면 하시다.	
그 저렇다 이 눈이 가득 아들이 있다면 하는데, 그렇는데 맛있다면 하다는데 그렇지만 살다면 하다 하다.	
요즘 젖은 내용 회사에 다른 이 사람이 되는 아니라 되는 것이 하를 하다 하는 것이 되는 것이다. 하는	

FLEX Programmer's Manual

FLEX Programmer's Manual

COPYRIGHT © 1978 by
Technical Systems Consultants, Inc.
P.O. Box 2570
West Lafayette, Indiana 47906
All Rights Reserved

COPYRIGHT INFORMATION

This entire manual is provided for the personal use and enjoyment of the purchaser. Its contents are copyrighted by Technical Systems Consultants, Inc., and reproduction, in whole or in part, by any means is prohibited. Use of this program, or any part thereof, for any purpose other than single end use by the purchaser is prohibited.

DISCLAIMER

The supplied software is intended for use only as described in this manual. Use of undocumented features or parameters may cause unpredictable results for which Technical Systems Consultants, Inc. cannot assume responsibility. Although every effort has been made to make the supplied software and its documentation as accurate and functional as possible, Technical Systems Consultants, Inc. will not assume responsibility for any damages incurred or generated by such material. Technical Systems Consultants, Inc. reserves the right to make changes in such material at any time without notice.

CONTENTS

Ι.	Introduction	1
II.	Disk Operating System DOS Memory Map User Callable Routines User Written Commands Disk Resident Commands Comments About Commands Examples of DOS Calls	3 8 16 17 18 19
III.	File Management System File Control Blocks FMS Entry Points FMS Global Variables FMS Function Codes Random Files Error Numbers	21 22 26 27 28 37 38
IV.	Disk Drivers	41
٧.	Disk Structures Diskette Initialization Directory Sectors Data Sectors Binary Files Text Files	43 43 44 44 45 46
VI.	Writing Utility Commands Example Program	47 49
VII.	The DOS LINK Utility	51
VIII.	Printer Routines The P Utility	53 54
IX.	General Information Interrupts in FLEX System Memory Map	57 57 57

			•		
	2.3				
	•				
	•				
		•			
					i
					1
					i
					1
	To the second				
					1
					1
					1

Preface

The purpose of the Advanced Programmer's Manual is to provide the assembler language programmer with the information required to make effective use of the available system routines and functions. This manual applies to the 6809 version of FLEX. The programmer should keep this manual close at hand while learning the system. It is organized to make it convenient as a quick reference guide as well as a thorough reference manual. The manual is not written for the novice programmer and assumes the user to have a thorough understanding of assembler language programming techniques.

Introduction

The FLEX Operating System consists of three main parts: the Disk Operating System (DOS) which processes commands, the File Management System (FMS) which manages files on a diskette, and the Utility Command Set, which are the user-callable commands. The Utility Command Set is described in the FLEX User's Guide. Details of the Disk Operating System and File Management System portions of FLEX are described in this manual, which is intended for the programmer who wishes to write his own commands or process disk files from his own program.

When debugging programs which use disk files and the File Management System, the user should take the following precautions:

- 1. Write-protect the system diskette by exposing or covering the write-protect cutout on the diskette. See the FLEX User's Guide for further details on this operation. This will prevent destruction of the system disk in case the program starts running wild.
- 2. Use an empty scratch diskette as the working diskette to which your program will write any data files. If something goes wrong and the diskette is destroyed, no valuable data will have been lost.
- 3. Test your program repeatedly, especially with "special cases" of data input which may not be what the program is expecting. Well-written programs abort gracefully when detecting errors, not dramatically.

A careful programmer, using the information in this manual, should be able to make the fullest use of his floppy disk system.

DISCLAIMER

This product is intended for use only as described in this document and the FLEX User's Guide. Technical Systems Consultants will not be responsible for the proper functioning of features or parameters. The user is urged to abide by the warnings and cautions issued in this document lest valuable data or diskettes be destroyed.

PATCHING "FLEX"

It is not possible to patch FLEX. Technical Systems Consultants cannot be responsible for any destructive side-effects which may result from attempts to patch FLEX.

THE DISK OPERATING SYSTEM

The Disk Operating System (DOS) forms the communication link between the user (via a computer terminal) and the File Management System. All commands are accepted through DOS. Functions such as file specification parsing, command argument parsing, terminal I/O, and error reporting are all handled by DOS. The following sections describe the DOS global variable storage locations (Memory Map), the DOS user callable subroutines, and give examples of some possible uses.

DOS MEMORY MAP

The following is a description of those memory locations within the DOS portion of FLEX which contain information of interest to the programmer. The user is cautioned against utilizing for his own purposes any locations documented as being either "reserved" or "system scratch", as this action may cause destruction of data.

\$CO80-\$COFF - Line Buffer

The line buffer is a 128 byte area into which characters typed at the keyboard are placed by the routine INBUF. All characters entered from the keyboard are placed in this buffer with the exception of control characters. Characters which have been deleted by entering the backspace character do not appear in the buffer, nor does the backspace character itself appear. The carriage return signaling the end of the keyboard input is, however, put in the buffer. This buffer is also used to hold the STARTUP file during a coldstart (boot) operation.

\$CCOO - TTYSET Backspace Character

This is the character which the routine INBUF will interpret as the Backspace character. It is user definable through the TTYSET DOS Utility. Default = \$08, a Control-H (ASCII BS).

- \$CCO1 TTYSET Delete Character
 - This is the character which the routine INBUF will interpret as the line cancel or Delete character. It is user definable through the TTYSET DOS Utility. Default = \$18, a control-X (ASCII CAN).
- \$CC02 TTYSET End of Line Character
 This is the character DOS recognizes as the multiple command per
 line separator. It is user definable through the TTYSET Utility.
 Default = \$3A, a colon (:).
- \$CCO3 TTYSET Depth Count
 This byte determines how many lines DOS will print on a page before
 Pausing or issuing Ejects. It may be set by the user with the
 TTYSET command. Default = 0.
- \$CCO4 TTYSET Width Count

 This byte tells DOS how many characters to output on each line. If zero, there is no limit to the number output. This count may be set by the user using TTYSET. Default = 0.

\$CCO5 - TTYSET Null Count

This byte informs DOS of the number of null or pad characters to be output after each carriage return, line feed pair. This count may be set using TTYSET. Default = 4.

\$CCO6 - TTYSET Tab Character

This byte defines a tab character which may be used by other programs, such as the Editor. DOS itself does not make use of the Tab character. Default = 0, no tab character defined.

\$CCO7 - TTYSET Backspace Echo Character

This is the character the routine INBUF will echo upon the receipt of a backspace character. If the backspace echo character is set to a \$08, and the backspace character is also a \$08, FLEX will output a space (\$20) prior to the outputting of the backspace echo character. Default = 0.

\$CCO8 - TTYSET Eject Count

The Eject Count instructs DOS as to the number of blank lines to be output after each page. (A page is a set of lines equal in number to the Depth Count). If this byte is zero, no Eject lines are output. Default = 0.

\$CC09 - TTYSET Pause Control

The Pause byte instructs DOS what action to take after each page is output. A zero value indicates that the pause feature is enabled; a non-zero value, pause is disabled. Default = \$FF, pause disabled.

\$CCOA - TTYSET Escape Character

The Escape character causes DOS to pause after an output line. Default = \$1B, ASCII ESC.

\$CCOB - System Drive Number

This is the number of the disk drive from which commands are loaded. If this byte is FF, both drives 0 and 1 will be searched. Default = drive #0.

\$CCOC - Working Drive Number

This is the number of the default disk drive referenced for non-command files. If this byte is \$FF, both drives 0 and 1 will be searched. Default = drive #0.

- \$CCOD System Scratch
- \$CCOE-\$CC10 System Date Registers

These three bytes are used to store the system date. It is stored in binary form with the month in the first byte, followed by the day, then the year. The year byte contains only the tens and ones digits.

\$CC11 - Last Terminator

This location contains the most recent non-alphanumeric character encountered in processing the line buffer. See commentary on the routines NXTCH and CLASS in the section "User-Callable System Routines".

\$CC12-\$CC13 - User Command Table Address

The programmer may store into these locations the address of a command table of his own construction. See the section called "User-Written Commands" for details. Default = 0000, no user command table is defined.

\$CC14-\$CC15 - Line Buffer Pointer

These locations contain the address of the next character in the Line Buffer to be processed. See documentation of the routines INBUFF, NXTCH, GETFIL, GETCHR, and DOCMND in the section "User-Callable System Routines" for instances of its use.

\$CC16-\$CC17 - Escape Return Register

These locations contain the address to which to jump if a RETURN is typed while output has been stopped by an Escape Character. See the FLEX User's Guide, TTYSET, for information on Escape processing. See also the documentation for the routine PCRLF in the section called "User-Callable System Routines".

\$CC18 - Current Character

This location contains the most recent character taken from the Line Buffer by the NXTCH routine. See documentation of the NXTCH routine for additional details.

\$CC19 - Previous Character

This location contains the previous character taken from the Line Buffer by the NXTCH routine. See documentation of the NXTCH routine for additional details.

\$CC1A - Current Line Number

This location contains a count of the number of lines currently on the page. This value is compared to the Line Count value to determine if a full page has been printed.

\$CC1B-\$CC1C - Loader Address Offset

These locations contain the 16-bit bias to be added to the load address of a routine being loaded from the disk. See documentation of the System Routine LOAD for details. These locations are also used as scratch by some system routines.

\$CC1D - Transfer Flag

After a program has been loaded from the disk (see LOAD documentation), this location is non-zero if a transfer address was found during the loading process. This location is also used as scratch by some system routines.

- \$CC1E-\$CC1F Transfer Address
 - If the Transfer Flag was set non-zero by a load from the disk (see LOAD documentation), these locations contain the last transfer address encountered. If the Transfer Flag was set zero by the disk load, the content of these locations is indeterminate.
- \$CC20 Error Type
 This location contains the e

This location contains the error number returned by several of the File Management System functions. See the "Error Numbers" section of this document for an interpretation of the error numbers.

\$CC21 - Special I/O Flag

If this byte is non-zero, the PUTCHR routine will ignore the TTYSET Width feature and also ignore the Escape Character. The routine RSTRIO clears this byte. Default = 0.

\$CC22 - Output Switch

If zero, output performed by the PUTCHR routine is through the routine OUTCH. If non-zero, the routine OUTCH2 is used. See documentation of these routines for details.

\$CC23 - Input Switch

If zero, input performed by GETCHR is through the routine INCH. If it is non-zero, the routine INCH2 is used. See documentation of these routines for details.

\$CC24-\$CC25 - File Output Address

These bytes contain the address of the File Control Block being used for file output. If the bytes are zero, no file output is performed. See PUTCHR description for details. These locations are set to zero by RSTRIO.

\$CC26-\$CC27 - File Input Address

These bytes contain the address of the File Control Block being used for file input. If the bytes are zero, no file input is performed. The routine RSTRIO clears these bytes. See GETCHR for details.

\$CC28 - Command Flag

This location is non-zero if DOS was called from a user program via the DOCMND entry point. See documentation of DOCMND for details.

\$CC29 - Current Output Column

This location contains a count of the number of characters currently in the line being output to the terminal. This is compared to the TTYSET Width Count to determine when to start a new line. The output of a control character resets this count to zero.

\$CC2A - System Scratch

\$CC2B-\$CC2C - Memory End

These two bytes contain the end of user memory. This location is set during system boot and may be read by programs requiring this information.

\$CC2D-\$CC2E - Error Name Vector

If these bytes are zero, the routine RPTERR will use the file ERRORS.SYS as the error file. If they are non-zero, they are assumed to be the address of an ASCII string of characters (in directory format) of the name of the file to be used as the error file. See the description of RPTERR for more details.

\$CC2F - File Input Echo Flag

If this byte is non-zero (default) and input is being done through a file, the character input will be echoed to the output channel. If this byte is zero, the character retrieved will not be echoed.

\$CC30-\$CC4D - System Scratch

\$CC4E-\$CCBF - System Constants

\$CCCO-\$CCD7 - Printer Initialize

This area is reserved for the overlay of the system printer initializization subroutine.

\$CCD8-\$CCE3 - Printer Ready Check

This area is reserved for the overlay of the system "check for printer ready" subroutine.

\$CCE4-\$CCF7 - Printer Output

This area is reserved for the overlay of the system printer output character routine. See Printer Routine descriptions for details.

\$CCF8-\$CCFF - System Scratch

USER-CALLABLE SYSTEM ROUTINES

Unless specifically documented otherwise, the content of all registers should be presumed destroyed by calls to these routines. All routines, unless otherwise indicated, should be called with a JSR instruction. In the 6809 version of FLEX the Y and U registers are preserved across all the following routines. The A,B and X registers should be considered changed except where noted otherwise. Often a value or status is returned in one of these registers.

\$CD00 (COLDS) Coldstart Entry Point

The BOOT program loaded from the disk jumps to this address to initialize the FLEX system. Both the Disk Operating System (DOS) portion and the File Management System portion (FMS) of FLEX are initialized. After initialization, the FLEX title line is printed and the STARTUP file, if one exists, is loaded and executed. This entry point is only for use by the BOOT program, not by user programs. Indiscriminate use of the Coldstart Entry Point by user programs could result in the destruction of the diskette. Documentation of this routine is included here only for completeness.

\$CD03 (WARMS) Warmstart Entry Point

This is the main re-entry point into DOS from user programs. A JMP instruction should be used to enter the Warmstart Entry Point. At this point, the main loop of DOS is entered. The main loop of DOS checks the Last Terminator location for a TTYSET end-of-line character. If one is found, it is assumed that there is another command on the line, and DOS attempts to process it. If no end-of-line is in the Last Terminator location DOS assumes that the current command line is finished, and looks for a new line to be input from the keyboard. If, however, DOS was called from a user program through the DOCMND entry point, control will be returned to the user program when the end of a command line is reached.

\$CD06 (RENTER) DOS Main Loop Re-entry Point

This is a direct entry point into the DOS main loop. None of the Warmstart initialization is performed. This entry point must be entered by a JMP instruction. Normally, this entry point is used internally by DOS and user-written programs should not have need to use it. For an example of use, see "Printer Driver" section for details.

\$CD09 (INCH) Input Character \$CD0C (INCH2) Input Character

Each of these routines inputs one character from the keyboard, returning it to the calling program in the A-register. The address portion of these entries points to a routine in the Custom I/O package. They may be altered by changing that package. The GETCHR routine normally uses INCH but may be instructed to use INCH2 by setting the "Input Switch" non-zero (see Memory Map). The user's program may change the jump vector at the INCH address to refer to some other input routine such as a routine to get a character from paper tape. The INCH2 address should never be altered. The Warmstart Entry Point resets the INCH jump vector to the same routine as INCH2 and sets the Input Switch to zero. RSTRIO also resets these bytes. User programs should use the GETCHR routine, documented below, rather than calling INCH, because INCH does not check the TTYSET parameters.

\$CDOF (OUTCH) Output Character
\$CD12 (OUTCH2) Output Character

On entry to each of these routines, the A-register should contain the character being output. Both of these routines output the character in the A-register to an output device. The OUTCH routine usually does the same as OUTCH2; however, OUTCH may be changed by programs to refer to some other output routine. For example, OUTCH may be changed to drive a line printer. OUTCH2 is never changed, and always points to the output routine in the Custom I/O package. This address may not be patched to refer to some other output routine. The routine PUTCHR, documented below, calls one of these two routines, depending on the content of the location "Output Switch" (see Memory Map). The Warmstart Entry Point resets the OUTCH jump vector to the same routine as OUTCH2, and sets the Output Switch to zero. RSTRIO also resets these locations. User routines should use PUTCHR rather than calling OUTCH or OUTCH2 directly since these latter two do not check the TTYSET parameters.

\$CD15 (GETCHR) Get Character

This routine gets a single character from the keyboard. The character is returned to the calling program in the A-register. The Current Line Number location is cleared by a call to GETCHR. Because this routine honors the TTYSET parameters, its use is preferred to that of INCH. If the location "Input Switch" is non-zero, the routine INCH2 will be used for input. If zero, the byte at "File Input Address" is checked. If it is non-zero, the address at this location is used as a File Control Block of a previously opened input file and a character is retrieved from the file. If zero, a character is retrieved via the INCH routine. The X and B registers are preserved.

\$CD18 (PUTCHR) Put Character

This routine outputs a character to a device, honoring all of the TTYSET parameters. On entry, the character should be in the A-register. If the "Special I/O Flag" (see Memory Map) is zero, the column count is checked, and a new line is started if the current If an ACIA is being used to control the monitor line is full. terminal, it is checked for a TTYSET Escape Character having been typed. If so, output will pause at the end of the current line. If the location "Output Switch" is non-zero, the routine OUTCH2 is used to send the character. If zero, the location File Output Address is checked. If it is non-zero the contents of this location is used as a address of a File Control Block of a previously opened for write file, and the character is written to the file. If zero, the routine OUTCH is called to process the character. Normally, OUTCH sends the character to the terminal. The user program may, however, change the address portion of the OUTCH entry point to go to another character output routine. The X and B registers are preserved.

\$CD1B (INBUFF) Input into Line Buffer

This routine inputs a line from the keyboard into the Line Buffer. The TTYSET Backspace and Delete characters are checked processed if encountered. All other control characters except RETURN and LINE FEED, are ignored. The RETURN is placed in the buffer at the end of the line. A LINE FEED is entered into the buffer as a space character but is echoed back to the terminal as a Carriage Return and Line Feed pair for continuation of the text on a new line. At most, 128 characters may be entered on the line, including the final RETURN. If more are entered, only the first 127 are kept, the RETURN being the 128th. On exit, the Line Buffer Pointer is pointing to the first character in the Line Buffer. Caution: The command line entered from the keyboard is kept in the Line Buffer. Calling INBUF from a user program will destroy the command line, including all unprocessed commands on the same line. Using INBUF and the Line Buffer for other than DOS commands may result in unpredictable side-effects.

\$CD1E (PSTRNG) Print String

This routine is similar to the PDATA routine in SWTBUG and DISKBUG. On entry, the X-register should contain the address of the first character of the string to be printed. The string must end with an ASCII EOT character (\$04). This routine honors all of the TTYSET conventions when printing the string. A carriage return and line feed are output before the string. The B register is preserved.

\$CD21 (CLASS) Classify Character

This routine is used for testing if a character is alphanumeric (i.e. a letter or a number). On entry, the character should be in the A-register. If the character is alphanumeric, the routine returns with the carry flag cleared. If the character is not alphanumeric, the carry flag is set and the character is stored in the Last Terminator location. All registers are preserved by this routine.

\$CD24 (PCRLF) Print Carriage Return and Line Feed

In addition to printing a carriage return and line feed, this routine checks and honors several TTYSET conditions. this routine checks for a TTYSET Escape Character having been entered while the previous line was being printed. If so, the routine waits for another TTYSET Escape Character or a RETURN to be typed. If a RETURN was entered, the routine clears the Last Terminator location so as to ignore any commands remaining in the command line, and then jumps to the address contained in the Escape Return Register locations. Unless changed by the user's program, this address is that of the Warmstart Entry Point. If, instead of RETURN, another TTYSET Escape Character was typed, or it wasn't necessary to wait for one, the Current Line Number is checked. the last line of the page has been printed and the TTYSET Pause feature is enabled, the routine waits for a RETURN or a TTYSET Escape Character, as above. Note that all pausing is done before the carriage return and line feed are printed. The carriage return and line feed are now printed, followed by the number of nulls specified by the TTYSET Null Count. If the end of the page was encountered on entry to this routine, an "eject" is performed by issuing additional carriage return, line feeds, and nulls until the total number of blank lines is that specified in the TTYSET Eject Count. The X register is preserved.

\$CD27 (NXTCH) Get Next Buffer Character

The character in location Current Character is placed in location Previous Character. The character to which the Line Buffer Pointer points is taken from the Line Buffer and saved in the Current Character location. Multiple spaces are skipped so that a string of spaces looks no different than a single space. The Line Buffer Pointer is advanced to point to the next character unless the character just fetched was a RETURN or TTYSET End-of-Line Thus, once an end-of-line character or RETURN is encountered, additional calls to NXTCH will continue to return the same end-of-line character or RETURN. NXTCH cannot be used to cross into the next command in the buffer. NXTCH exits through the routine CLASS, automatically classifying the character. On exit. the character is in the A-register, the carry is clear if the character is alphanumeric, and the B-register and X- register are preserved.

\$CD2A (RSTRIO) Restore I/O Vectors

This routine forces the OUTCH jump vector to point to the same routine as does the OUTCH2 vector. The Output Switch location and the Input Switch location are set to zero. The INCH jump vector is reset to point to the same address as the INCH2 vector. Both the File Input Address and the File Output Address are set to zero. The A-register and B-register are preserved by this routine.

\$CD2D (GETFIL) Get File Specification

On entry to this routine, the X-register must contain the address of a File Control Block (FCB), and the Line Buffer Pointer must be pointing to the first character of a file specification in the Line Buffer. This routine will parse the file specification, storing the various components in the FCB to which the X-register points. If a drive number was not specified in the file specification, the working drive number will be used. On exit, the carry bit will be clear if no error was detected in processing the specification. The carry bit will be set if there was a format error in the file specification. If no extension was specified in the file specification, none is stored. The calling program should set the default extension desired after GETFIL has been called by using the SETEXT routine. The Line Buffer Pointer is left pointing to the character immediately beyond the separator, unless the separator is a carriage return or End of Line character. If an error was detected, Error number 21 is stored in the error status byte of the FCB. The X register is preserved with a call to this routine.

\$CD30 (LOAD) File Loader

On entry, the system File Control Block (at \$C840) must contain the name of a file which has been opened for binary reading. This routine is used to load binary files only, not text files. file is read from the disk and stored in memory, normally at the load addresses specified in the binary file itself. It is possible to load a binary file into a different memory area by using the Loader Address Offset locations. The 16-bit value in the Loader Address Offset locations is added to the addresses read from the binary file. Any carry generated out of the most significant bit the address is lost. The transfer address, if any is encountered, is not modified by the Loader Address Offset. that the setting of a value in the Loader Address Offset does not modify any part of the content of the binary file. act as a program relocator in that it does not change any addresses in the program itself, merely the location of the program If the the file is to be loaded without an offset, be certain to clear the Loader Address Offset locations before calling this routine. On exit, the Transfer Address Flag is zero if no transfer address was found. This flag is non-zero if a transfer address record was encountered in the binary file, and the Transfer Address locations contain the last transfer address encountered. The disk file is closed on exit. If a disk error is encountered,

an error message is issued and control is returned to DOS at the Warmstart Entry Point.

\$CD33 (SETEXT) Set Extension

On entry, the X-register should contain the address of the FCB into which the default extension is to be stored if there is not an extension already in the FCB. The A-register, on entry, should contain a numeric code indicating what the default extension is to The numeric codes are described below. If there is already an extension in the FCB (possibly stored there by a call to GETFIL), this routine returns to the calling program immediately. If there is no extension in the FCB, the extension indicated by the numeric code in the A-register is placed in the FCB File Extension area. The legal codes are:

- 0 BIN
- 1 TXT
- 2 CMD
- 3 BAS
- 4 SYS
- 5 BAK
- 6 SCR
- 7 DAT
- 8 BAC
- 9 DIR
- 10- PRT
- 11- OUT

Any values other than those above are ignored, the routine returning without storing any extension. The X register is preserved in this routine.

\$CD36 (ADDBX) Add B-register to X-register

The content of the B-register is added to the content of the X-register. This routine is here for compatibility with 6800 FLEX.

\$CD39 (OUTDEC) Output Decimal Number

On entry, the X-register contains the address of the significant byte of a 16-bit (2 byte), unsigned, binary number. The B-register, on entry, should contain a space suppression flag. The number will be printed as a decimal number with leading zeroes suppressed. If the B-register was non-zero on entry, spaces will be substituted for the leading zeroes. If the B-register is zero on entry, printing of the number will start with the first non-zero digit.

\$CD3C (OUTHEX) Output Hexadecimal Number

On entry, the X-register contains the address of a single binary byte. The byte to which the X-register points is printed as 2 hexadecimal digits. The B and X registers are preserved.

\$CD3F (RPTERR) Report Error

On entry to this routine, the X-register contains the address of a File Control Block in which the Error Status Byte is non-zero. The error code in the FCB is stored by this routine in the Error Type location. A call to the routine RSTRIO is made and location Error Vector is checked. If this location is zero, the file ERRORS.SYS is opened for random read. If this location is non-zero, it is assumed to be an address pointing to an ASCII string (containing any necessary null pad characters) of a legal File name plus extension (string should be 11 characters long). provided file is then opened for random read. The error number is used in a calculation to determine the record number and offset of Each error the appropriate error string message in the file. message string is 63 characters in length, thus allowing 4 messages per sector. If the string is found, it is printed on the terminal. If the string is not found (due to too large of error number being encountered) or if the error file itself was not located on the disk, the error numberis reported to the monitor terminal as part of the message:

DISK ERROR #nnn

Where "nnn" is the error number being reported. A description of the error numbers is given elsewhere in this document.

\$CD42 (GETHEX) Get Hexadecimal Number

This routine gets a hexadecimal number from the Line Buffer. On entry, the Line Buffer Pointer must point to the first character of the number in the Line Buffer. On exit, the carry bit is cleared if a valid number was found, the B-register is set non-zero, and the X-register contains the value of the number. The Line Buffer Pointer is left pointing to the character immediately following the separator character, unless that character is a carriage return or End of Line. If the first character examined in the Line Buffer is a separator character (such as a comma), the carry bit is still cleared, but the B-register is set to zero indicating that no actual number was found. In this case, the value returned in the X-register is zero. If a non-hexadecimal character is found while processing the number, characters in the Line Buffer are skipped until a separator character is found, then the routine returns to the caller with the carry bit set. The number in the Line Buffer may be of any length, but the value is truncated to between 0 and \$FFFF, inclusive.

\$CD45 (OUTADR) Output Hexaecimal Address

On entry, the X register contains the address of the most significant byte of a 2 byte hex value. The bytes to which the X register points are printed as 4 hexadecimal digits.

\$CD48 (INDEC) Input Decimal Number

This routine gets an unsigned decimal number from the Line Buffer. On entry, the Line Buffer Pointer must point to the first character of the number in the Line Buffer. On exit, the carry bit is cleared if a valid number was found, the B-register is set non-zero, and the X-register contains the binary value of the number. The Line Buffer Pointer is left pointing as described in the routine GETHEX. If the first character examined in the buffer is a separator character (such as a comma), the carry bit is still cleared, but the B-register is set to zero indicating that no actual number was found. In this case, the number returned in X is zero. The number in the Line Buffer may be of any length but the result is truncated to 16 bit precision.

\$CD4B (DOCMND) Call DOS as a Subroutine

This entry point allows a user-written program to pass a command string to DOS for processing, and have DOS return control to the user program on completion of the commands. The command string must be placed in the Line Buffer by the user program, and the Line Buffer Pointer must be pointing to the first character of the command string. Note that this will destroy any as yet unprocessed parameters and commands in the Line Buffer. The command string must terminate with a RETURN character (\$D hex). commands have been processed. DOS will return control to the user's program with the B-register containing any error code received from the File Management System. The B-register will be zero if no errors were detected. Caution: do not use this feature to load programs which may destroy the user program in memory. An example of a use of this feature of DOS is that of a program wanting to save a portion of memory as a binary file on the disk. The program could build a SAVE command in the Line Buffer with the desired file name and parameters, and call the DOCMND entry point. On return, the memory will have been saved on the disk.

\$CD4E (STAT) Check Terminal Input Status

This routine may be called to check the status of the terminal input device (to see if a character has been typed on the keyboard). If a character has been hit, the Z condition code will be cleared on return (a not-equal condition). If no character has been hit, the Z condition code will be set (an equal condition). No registers, other than the CC-register, are altered.

USER-WRITTEN COMMANDS

The programmer may write his own commands for DOS. These commands may be either disk-resident as disk files with a CMD extension, or they may be memory-resident in either RAM or ROM.

MEMORY-RESIDENT COMMANDS:

A memory-resident command is a program, already in memory, to which DOS will transfer when the proper command is entered from the keyboard. The command which invokes the program, and the entry-point of the program, are stored in a User Command Table created by the programmer in memory. Each entry in the User Command Table has the following format:

```
FCC 'command' (Name that will invoke the program)
FCB 0
FDB entry address (This is the entry address of the program)
```

The entire table is ended by a zero byte. For example, the following table contains the commands DEBUG (entry at \$3000) and PUNT (entry at \$3200):

FCC	'DEBUG'	Command Name
FCB	0	
FDB	\$3000	Entry address for DEBUG
FCC	'PUNT.'	Command name
FCB	0	
FDB	\$3200	Entry address for PUNT
FCB	0	End of command table

The address of the User Command Table is made known to DOS by storing it in the User Command Table Address locations (See Memory Map).

The User Command Table is searched before the disk directory, but after DOS's own command table is searched. The DOS command table contains only the GET and MON commands. Therefore, the user may not define his own GET and MON commands.

Since the User Command Table is searched before the disk directory, the programmer may have commands with the same name as those on the disk. However, in this case, the commands on the disk will never be executed while the User Command Table is known to DOS. The User Command Table may be deactivated by clearing the User Command Table Address locations.

DISK-RESIDENT COMMANDS

A disk-resident command is an assembled program, with a transfer address, which has been saved on the disk with a CMD extension. The ASMB section of the FLEX User's Guide describes the way to assign a transfer address to a program being assembled.

Disk commands, when loaded into memory, may reside anywhere in the User RAM Area; the address is determined at assembly time by using an ORG statement. Most commands may be assembled to run in the Utility Command Space (see Memory Map). Most of the commands supplied with FLEX run in the Utility Command Space. For this reason, the SAVE command cannot be used to save information which is in the Utility Command Space or System FCB space as this information would be destroyed when the SAVE command is loaded. The SAVE.LOW command is to be used in this case. The SAVE.LOW command loads into memory at location \$100 and allows the saving of programs in the \$C100 region.

The System FCB area is used to load all commands from the disk. Commands written to run in the Utility Command Space must not overflow into the System FCB area. Once loaded, the command itself may use the System FCB area for scratch or as an FCB for its own disk I/0. See the example in the FMS section.

GENERAL COMMENTS ABOUT COMMANDS

User-written commands are entered by a JMP instruction. On completion, they should return control to DOS by jumping (JMP instruction) to the Warmstart Entry Point (see Memory Map).

Processing Arguments.

User-written commands are required to process any arguments entered from the keyboard. The command name and the arguments typed are in the Line Buffer area (see Memory Map). The Line Buffer Pointer, on entry to the command, is pointing to the first character of the first argument, if one exists. If there are no arguments, the Line Buffer Pointer is pointing to either an end-of-line character or a carriage return. The DOS routines NXTCH, GETFIL, and GETHEX should be used by the command for processing the arguments.

Processing Errors.

If the command, while executing, receives an error status from either DOS or FMS of such a nature that the command must be aborted, the program should jump to the Warmstart Entry Point of DOS after issuing an appropriate error message. Similarly, if the command should detect an error on its own, it should issue a message and return to DOS through the Warmstart Entry Point.

EXAMPLES OF USING DOS ROUTINES

1. Setting up a file spec in the FCB can be done in the following manner. This example assumes the Line Buffer Pointer is pointing to the first character of a file specification, and the desired resulting file spec should default to a TXT extension.

LDX #FCB Point to FCB
JSR GETFIL Get file spec into FCB
BCS ERROR Report error if one
LDA #1 Set extension code (TXT)
JSR SETEXT Set the default extension

The user may now open the file for the desired action, since the file spec is correctly set up in the FCB. Refer to the FMS examples for opening files.

2. The following examples demonstrate some simple uses of the basic I/O functions provided by DOS.

LDA #'A Setup an ASCII A JSR PUTCHR Call DOS out character

LDX #STRING Point to string
JSR PSTRNG Print CR & LF + string

The above simple examples are to show the basic mechanism for calling and using DOS I/O routines.

THE FILE MANAGEMENT SYSTEM

The File Management System (FMS), forms the communication link between the DOS and the actual Disk Hardware. The FMS performs all file allocation and removal on the disk. All file space is allocated dynamically, and the space used by files is immediately reusable upon that file's deletion. The user of the FMS need not be concerned with the actual location of a file on the disk, or how many sectors it requires.

Communication with the FMS is done through File Control Blocks. These blocks contain the information about a file, such as its name and what drive it exists on. All disk I/O performed through FMS is "one character at a time" I/O. This means that programs need only send or request a single character at a time while doing file data transfers. In effect, the disk looks no different than a computer terminal. Files may be opened for either reading or writing. Any number of files may be opened at any one time, as long as each one is assigned its own File Control Block.

The FMS is a command language whose commands are represented by various numbers called Function Codes. Each Function Code tells FMS to perform a specific function such as open a file for read, or delete a file. In general, making use of the various functions which the FMS offers, is quite simple. The index register is made to point to the File Control Block which is to be used, the Function Code is stored in the first byte of the File Control Block, and FMS is called as a subroutine (JSR). At no time does the user ever have to be concerned with where the file is being located on the disk, how long it is, or where its directory entry is located. The FMS does all of this automatically.

Since the file structure of FLEX is a linked structure, and the disk space is allocated dynamically, it is possible for a file to exist on the disk in a set of non-contiguous sectors. Normally, if a disk has just been formatted, a file will use consecutive sectors on the disk. As files are created and deleted, however, the disk may become "fragmented". Fragmentation results in the sectors on the disk becoming out of order physically, even though logically they are still all sequential. This is a characteristic of "linked list" structures and dynamic file allocation methods. The user need not be concerned with this fragmentation, but should be aware of the fact that files may exist whose sectors seem to be spattered all over the disk. The only result of fragmentation is the slowing down of file read times, because of the increased number of head seeks necessary while reading the file.

THE FILE CONTROL BLOCK (FCB)

The FCB is the heart of the FLEX File Management System (FMS). An FCB is a 320 byte long block of RAM, in the user's program area, which is used by programs to communicate with FMS. A separate FCB is needed for each open file. After a file has been closed, the FCB may be re-used to open another file or to perform some other disk function such as Delete or Rename. An FCB may be placed anywhere in the user's program area (except page zero) that the programmer wishes. The memory reserved for use as an FCB need not be preset or initialized in any way. Only the parameters necessary to perform the function need be stored in the FCB; the File Management System will initialize those areas of the FCB needed for its use.

In the following description of an FCB, the byte numbers are relative to the beginning of the FCB; i.e. byte 0 is the first byte of the FCB.

DESCRIPTION OF AN FCB

Byte 0 Function Code

The desired function code must be stored in this byte by the user before calling FMS to process the FCB. See the section describing FMS Function Codes.

Byte 1 Error Status Byte

If an error was detected during the processing of a function, FMS stores the error number in this byte and returns to the user with the CPU Z-Condition Code bit clear, i.e. a non-zero condition exists. This may be tested by the BEQ or BNE instruction.

Byte 2 Activity Status

This byte is set by FMS to a "1" if the file is open for read, or "2" if the file is open for writing. This byte is checked by several FMS function processors to determine if the requested operation is legal. A Status Error is returned for illegal operations.

The next 12 bytes (3-14) comprise the "File Specification" of the file being referenced by the FCB. A "File Specification" consists of a drive number, file name, and file extension. Some of the FMS functions do not require the file name or extension. See the documentation of the individual function codes for details.

Byte 3 Drive Number

This is the hardware drive number whose diskette contains the file being referenced. It should be binary 0 to 3. The next 24 bytes (4-27) comprise the "Directory Information" portion of the FCB. This is the exact same information which is contained in the diskette directory entry for the file being referenced.

Bytes 4-11 File Name

This is the name of the file being referenced. The name must start with a letter and contain only letters, digits, hyphens, and/or underscores. If the name is less than 8 characters long, the remaining bytes must be zero. The name should be left adjusted in its field.

Bytes 12-14 Extension

This is the extension of the file name for the file being referenced. It must start with a letter and contain only letters, digits, hyphens, and/or underscores. If the extension is less than 3 characters long, the remaining bytes must be zero. The extension should be left adjusted. Files with null extensions should not be created.

Byte 15 File Attributes

At present, only the most significant 4 bits are defined in this byte. These bits are used for the protection status bits and are assigned as follows:

BIT 7 = Write Protect

BIT 6 = Delete Protect

BIT 5 = Read Protect

BIT 4 = Catalog Protect

Setting these bits to 1 will activate the appropriate protection status. All undefined bits of this byte should remain 0!

Byte 16 Reserved for future system use

Bytes 17-18 Starting disk address of the file

These two bytes contain the hardware track and sector numbers, respectively, of the first sector of the file.

Bytes 19-20 Ending disk address of the file

These two byes contain the hardware track and sector numbers, respectively, of the last sector of the file.

Bytes 21-22 File Size

This is a 16-bit number indicating the number of sectors in the file.

FLEX Advanced Programmer's Guide

Byte 23 File Sector Map Indicator

If this byte is non-zero (usually \$02), the file has been created as a random access file and contains a File Sector Map. See the description of Random Files for details.

Byte 24 Reserved for future system use

Bytes 25-27 File Creation Date

These three bytes contain the binary date of the files creation. The first byte is the month, the second is the day, and the third is the year (only the tens and ones digits).

Bytes 28-29 FCB List Pointer

All FCBs which are open for reading or writing are chained together. These two bytes contain the memory address of the FCB List Pointer bytes of the next FCB in the chain. These bytes are zero if this FCB is the last FCB in the chain. The first FCB in the chain is pointed to by the FCB Base Pointer. (See Global Variables).

Bytes 30-31 Current Position

These bytes contain the hardware track and sector numbers, respectively, of the sector currently in the sector buffer portion of the FCB. If the file is being written, the sector to which these bytes point has not yet been written to the diskette; it is still in the buffer.

Bytes 32-33 Current Record Number

These bytes contain the current logical Record Number of the sector in the FCB buffer.

Byte 34 Data Index

This byte contains the address of the next data byte to be fetched from (if reading) or stored into (if writing) the sector buffer. This address is relative to the beginning of the sector, and is advanced automatically by the Read/Write Next Byte function. The user program has no need to manipulate this byte.

Byte 35 Random Index

This byte is used in conjuction with the Get Random Byte From Sector function to read a specific byte from the sector buffer without having to sequentially skip over any intervening bytes. The address of the desired byte, relative to the beginning of the sector, is stored in Random Index by the user, and the Get Random Byte From Sector function is issued to FMS. The specified data byte will be returned in the A-register. A value less than 4 will

access one of the linkage bytes in the sector. User data starts at an index value of 4.

Bytes 36-46 Name Work Buffer

These bytes are used internally by FMS as temporary storage for a file name. These locations are not for use by a user program.

Bytes 47-49 Current Directory Address

If the FCB is being used to process directory information with the Get/Put Information Record functions, these three bytes contain the track number, sector number, and starting data index of the directory entry whose content is in the Directory Information portion of the FCB. The values in these three bytes are updated automatically by the Get Information Record function.

Bytes 50-52 First Deleted Directory Pointer

These bytes are used internally by FMS when looking for a free entry in the directory to which to assign the name of a new file.

Bytes 53-63 Scratch Bytes

These are the bytes into which the user stores the new name and extension of a file being renamed. The new name is formatted the same as described above under File Name and File Extension.

Byte 59 Space Compression Flag

If a file is open for read or write, this byte indicates if space compression is being performed. A value of zero indicates that space compression is to be done when reading or writing the data. This is the value that is stored by the Open for Read and Open for A value of \$FF indicates that Write functions. compression is to be done. This value is what the user must store in this byte, after opening the file, if space compression is not desired. (Such as for binary files). A positive non-zero value in byte indicates that space compression is currently progress; the value being a count of the number of spaces processed thus far. (Note that although this byte overlaps the Scratch Bytes described above, there is no conflict since the Space Compression Flag is used only when a file is open, and the Scratch Bytes are used only by Rename, which requires that the file be closed). In general, this byte should be 0 while working with text type files. and \$FF for binary files.

Bytes 64-319 Sector Buffer

These bytes contain the data contained in the sector being read or written. The first four bytes of the sector are used by the system. The remaining 252 are used for data storage.

FILE MANAGEMENT SYSTEM - Entry Points

\$D400 - FMS Initialization

This entry point is used by the DOS portion of FLEX to initialize the File Management System after a coldstart. There should be no need for a user-written program to use this entry point. Executing an FMS Initialization at the wrong time may result in the destruction of data files, necessitating a re-initialization of the diskette.

\$D403 - FMS Close

This entry point is used by the DOS portion of FLEX at the end of each command line to close any files left open by the command processor. User-written programs may also use this entry point to close all open files; however, if an error is detected in trying to close a file, any remaining files will not be closed. Thus the programmer is cautioned against using this routine as a substitute for the good programming practice of closing files individually. There are no arguments to this routine. It is entered by a JSR instruction as though it were a subroutine. On exit, the CPU Z-Condition code is set if no error was detected (i.e. a "zero" condition exists). If an error was detected, the CPU Z-Condition code bit is clear and the X-register contains the address of the FCB causing the error.

\$D406 - FMS Call

This entry point is used for all other calls to the File Management System. A function code is stored in the Function Code byte of the FCB, the address of the FCB is put in the X-register, and this entry point is called by a JSR instruction. The function codes are documented elsewhere in this document. On exit from this entry point, the CPU Z-Condition code bit is set if no error was detected in processing the function. This bit may be tested with a BEQ or BNE instruction. If an error was detected, the CPU Z-Condition code bit is cleared and the Error Status byte in the FCB contains the error number. Under all circumstances, the address of the FCB is still in the X-register on exit from this entry point. Some of the functions require additional parameters in the A and/or B-registers. See the documentation of the Function codes for details. The B,X,Y and U registers are always preserved with a call to FMS.

GLOBAL VARIABLES

This section describes those variables within the File Management System which may be of interest to the programmer. Any other locations in the FMS area should not be used for data storage by user programs.

\$D409 - \$D40A FCB Base Pointer

These locations contain the address of the FCB List Pointer bytes of the first FCB in the chain of open files. The address in these locations is managed by FMS and the programmer should not store any values in these locations. A user program may, however, want to chain through the FCBs of the open files for some reason, and the address stored in these locations is the proper starting point. Remember that the address is that of the FCB List Pointer locations in the FCB, not the first word of the FCB. A value of zero in these locations indicates that there are no open files.

\$D40B - \$D40C Current FCB Address

These locations contain the address of the last FCB processed by the File Management System. The address is that of the first word of the FCB.

\$D435 Verify Flag

A non-zero value in this location indicates that FMS will check each sector written for errors immediately after writing it. A zero value indicates that no error checking on writes is to be performed. The default value is "non-zero".

FMS FUNCTION CODES

The FLEX File Management System is utilized by the user through function codes. The proper function code number is placed, by the user, in the Function Code byte of the File Control Block (FCB) before calling FMS (Byte 0). FMS should be called by a JSR to the "FMS Call" entry. On entry to FMS, the X-register should contain the address of the FCB. On exit from FMS, the CPU Z-Condition code bit will be clear if an error was detected while processing the function. This bit may be tested by the BNE and BEQ instructions. Note: In the following examples, the line "JSR FMS" is referencing the FMS Call entry at \$D406.

Function O - Read/Write Next Byte/Character

If the file is open for reading, the next byte is fetched from the file and returned to the calling program in the A-register. If the file is open for writing, the content of the A-register on entry is placed in the buffer as the next byte to be written to the file. The Compression Mode Flag must contain the proper value for automatic space compression to take place, if desired (see Description of the FCB, Compression Mode Flag for details). On exit, this function code remains unchanged in the Function Code byte of the FCB; thus, consecutive read/writes may be performed without having to repeatedly store the function code. When reading, an End-of-File error is returned when all data in the file has been read. When the current sector being read is empty, the next sector in the file is prepared for processing automatically, without any action being required of the user. Similarly, when writing, full sectors are automatically written to the disk without user intervention.

Example:

If reading LDX #FCB Point to the FCB
JSR FMS Call FMS
BNE ERROR Check for error
The character read is now in A.

If writing LDA CHAR Get the character
LDX #FCB Point to the FCB
JSR FMS Call FMS
BNE ERROR Check for errors
The character in A has been written

Function 1 - Open for Read

The file specified in the FCB is opened for read-only access. If the file cannot be found, an error is returned. The only parts of the FCB which must be preset by the programmer before issuing this function are the file specification parts (drive number, file name, and file extension) and the function code. The remaining parts of the FCB will be initialized by the Open process. The Open process sets the File Compression Mode Flag to zero, indicating a text file. If the file is binary, the programmer should set the File Compression Mode Flag to \$FF, after opening the file, to disable the space compression feature. On exit from FMS, after opening a file, the function code in the FCB is automatically set to zero (Read/Write Next Byte Function) in anticipation of I/O on the file.

Example:

#FCB Point to the FCB LDX [Set up file spec in FCB] LDA #1 Set open function code STA 0,X Store in FCB Call FMS JSR **FMS** BNE ERROR Check for errors The file is now open for text reading

To set for binary - continue with the following LDA #\$FF Set FF for sup. flag STA 59,X Store in suppression flag

Function 2 - Open for Write

This is the same as Function 1, Open for Read, except that the file must not already exist in the diskette directory, and it is opened for write-only access. A file opened for write may not be read unless it is first closed and then re-opened for read-only. The space compression flag should be treated the same as described in "Open for Read". A file is normally opened as a sequential file but may be created as a random file by setting the FCB location File Sector Map byte non-zero immediately following an open for write operation. Refer to the section on Random Files for more details. The file will be created on the drive specified unless the drive spec is \$FF in which case the file will be created on the first drive found to be ready.

Example:

LDX #FCB Point to FCB [Setup file spec in FCB] #2 Setup open for write code LDA Store in FCB STA 0,X JSR. FMS Call FMS BNE ERROR Check for errors File is now open for text write. For binary write, follow example in Read open.

Function 3 - Open for Update

This function opens the file for both read and write. The file must not be open and must exist on the specified drive. If the drive spec is \$FF, all drives will be searched. Once the file has been opened for update, four operations may be performed on it; 1. sequential read, 2. random read, 3. random write, and 4. close file. Note that it is not possible to do sequential writes to a file open for update. This implies that it is not possible to increase the size of a file which is open for update.

Function 4 - Close File

If the file was opened for reading, a close merely removes the FCB from the chain of open files. If the file was opened for writing, any data remaining in the buffer is first written to the disk, padding with zeroes if necessary, to fill out the sector. If a file was opened for writing but never written upon, the name of the file is removed from the diskette directory since the file contains no data.

Example:

LDX	#FCB	Point to FCB
LDA	#4	Setup close code
STA	0,X	Store in FCB
JSR	FMS	Call FMS
BNE	ERROR	Check for errors
File	has now	been closed.

Function 5 - Rewind File

Only files which have been opened for read may be rewound. On exit from FMS, the function code in the FCB is set to zero, anticipating a read operation on the file. If the programmer wishes to rewind a file which is open for writing so that it may now be read, the file must first be closed, then re-opened for reading.

Example:

Assun	ning the	file is open for read:
LDX	#FCB	Point to FCB
LDA	#5	Setup rewind code
STA	0,X	Store in FCB
JSR	FMS	Call FMS
BNE	ERROR	Check for errors
File	is now	rewound & ready for read

Function 6 - Open Directory

This function opens the directory on the diskette for access by a program. The FCB used for this function must not already be open for use by a file. On entry, the only information which must be preset in the FCB is the drive number; no file name is required. The directory entries are read by using the Get Information Record function. The Put Information Record function is used to write a directory entry. The normal Read/Write Next Byte function will not function correctly on an FCB which is opened for directory access. It is not necessary to close an FCB which has been opened for directory access after the directory manipulation is finished. The user should normally not need to access the directory.

Function 7 - Get Information Record

This function should only be issued on an FCB which has been opened with the Open Directory funcion. Each time the Get Information Record function is issued, the next directory entry will be loaded into the Directory Information area of the FCB (see Description of the FCB for details of the format of a directory entry). All directory entries, including deleted and unused entries are read when using this function. After an entry has been read, the FCB is said to "point" to the directory entry just read; the Current Directory Address bytes in the FCB refer to the entry just read. An End-of-File error is returned when the end of the directory is reached.

Example:

```
To get the 3rd directory entry -
      LDX
                     Point to FCB
            #FCB
            DRIVE
      LDA
                     Get the drive number
      STA
            3,X
                     Store in the FCB
      LDA
            #6
                     Setup open dir code
                     Store in FCB
      STA
            0,X
      JSR
            FMS
                     Call FMS
      BNE
            ERROR
                     Check for errors
      LDB
                     Set counter to 3
            #3
            #7
LOOP
      LDA
                     Setup get rec code
            0.X
                     Store in FCB
      STA
            FMS
                     Call FMS
      JSR
      BNE
                     Check for errors
            ERROR
                     Decrement the counter
      DECB
            LOOP.
                     Repeat til finished
      BNE
      The 3rd entry is now in the FCB
```

Function 8 - Put Information Record

This function should only be issued on an FCB which has been opened with the Open Directory function. The directory information is copied from the Directory Information portion of the FCB into the directory entry to which the FCB currently points. The directory sector just updated is then re-written automatically on the diskette to ensure that the directory is up-to-date. A user program should normally never have to write into a directory. Careless use of this function can lead to the destruction of data files, necessitating a re-initialization of the diskette.

Function 9 - Read Single Sector

This function is a low-level interface directly to the disk driver which permits the reading of a single sector, to which the Current Position bytes of the FCB point, into the Sector Bufffer area of the FCB. This function is normally used internally within FLEX and a user program should never need to use it. The Read/Write Next Byte function should be used instead, whenever possible. Extreme care should be taken when using this function since it does not conform to the usual conventions to which most of the other FLEX functions adhere.

Example:

LDX	#FCB	Point to FCB
LDA	TRACK	Get track number
STA		Set current track
LDA	SECTOR	Get sector number
STA	31,X	Set current sector
LDA	#9	Setup function code
STA	0 , X	Store in FCB
JSR	FMS	Call FMS
BNE	ERROR	Check for errors
The	sector is	now in the FCB

Function 10 (\$0A hex) - Write Single Sector

This function, like the Read Single Sector function, is a low-level interface directly to the disk driver which permits the writing of a single sector. As such, it requires extreme care in its use. This function is normally used internally by FLEX, and a user program should never need to use it. The Read/Write Next Byte function should be used whenever possible. Careless use of the Write Single Sector Function may result in the destruction of data, necessitating the re-initialization of the diskette. The disk address being written is taken from the Current Position bytes of the FCB; the data is taken from the FCB Sector Buffer. This function honors the Verify Flag (see Global Variables section for a description of the Verify Flag), and will check the sector after writing it if directed to do so by the Verify Flag.

Function 11 (\$0B hex) - Reserved for future system use

Function 12 (\$0C hex) - Delete File

This function deletes the file whose specification is in the FCB (drive numbers, file name, and extension). The sectors used by the file are released to the system for re-use. The file should not be open when this function is issued. The file specification in the FCB is altered during the delete process.

Example:

#FCB Point to FCB LDX [setup file spec in FCB] #12 Setup function code LDA 0,X Store in FCB STA JSR **FMS** Call FMS BNE ERROR Check errors File has now been deleted

Function 13 (\$0D hex) - Rename File

On entry, the file must not be open, the old name must be in the File Specification area of the FCB, and the new name and extension must be in the Scratch Bytes area of the FCB. The file whose specification is in the FCB is renamed to the name and extension stored in the FCB Scratch Bytes area. Both the new name and the new extension must be specified; neither the name nor the extension can be defaulted.

Example:

Point to FCB #FCB [setup both file specs in FCB] Setup function code LDA #13 0.X Store in FCB STA **FMS** Call FMS JSR Check for errors BNE ERROR File has been renamed

Function 14 (\$0E hex) - Reserved for future system use.

Function 15 (\$0F hex) - Next Sequential Sector

On entry the file should be open for either reading or writing (not update). If the file is open for reading, this function code will cause all of the remaining (yet unread) data bytes in the current sector to be skipped, and the data pointer will be positioned at the first data byte of the next sequential sector of the file. If the file is open for write, this operation will cause the remainder of the current sector to be zero filled and written out to the disk. The next character written to that file will be placed in the first available data location in the next sequential sector. It should be noted that all calls to this function code will be ignored unless at least one byte of data has either been written or read from the current sector.

Function 16 (\$10 hex) - Open System Information Record

On entry, only the drive number need be specified in the FCB; there is no file name associated with this function. The FCB must not be open for use by a file. This function accesses the System Information Record for the diskette whose drive number is in the FCB. There are no separate functions for reading or changing this sector. All references to the data contained in the System Information Record must be made by manipulating the Sector Buffer directly. This function is used internally within FLEX; there should be no need for a user-written program to change the System Information Record. Doing so may result in the destruction of data, necessitating the re-initialization of the diskette. There is no need to close the FCB when finished.

Function 17 (\$11 hex) - Get Random Byte From Sector

On entry, the file should be open for reading or update. Also, the desired byte's number should be stored in the Random Index byte of the FCB. This byte number is relative to the beginning of the sector buffer. On exit, the byte whose number is stored in the Random Index is returned to the calling program in the A-register. The Random Index should not be less than 4 since there is no user data in the first four bytes of the sector.

Example:

To read the 54th data byte of the current sector -LDX #FCB Point to the FCB LDA #54+4 Set to item + 4 STA 35,X Put it in random index LDA #17 Setup function code STA 0,X Store in FCB JSR FMS Call FMS BNE ERROR Check for errors Character is now in acc. A

Function 18 (\$12 hex) - Put Random Byte in Sector

The file must be open for update. This function is similar to Get Random Byte except the character in the A accumulator is written into the sector at the data location specified by Random Index of the FCB. The Random Index should not be less than 4 since only system data resides in the first 4 bytes of the sector.

Example:

To write into the 54th data byte of the current sector-LDX #FCB Point to the FCB LDA #54+4 Set to item + 4 35,X STA Put it in Random Index LDA Setup Function Code #18 STA 0.XStore in FCB LDA CHAR Get character to be written JSR Call FMS FMS BNE Check for errors ERROR Character has been written

Function 19 (\$13 hex) - Reserved for future system use

Function 20 (\$14 hex) - Find Next Drive

This function is used to find the next online drive which is in the "ready" state. Due to hardware limitations, the minifloppy version of FLEX performs this command differently than the full size floppy version. The functioning of the full size floppy version is as follows. If the drive number in the FCB is hex FF, the search for drives will start with drive 0. If the drive number is 0, 1, or 2, the search will start with drive 1, 2, or 3 respectively. If a ready drive is found, its drive number will be returned in the drive number byte of the FCB and the carry bit will be cleared. If no ready drive is found, the carry bit will be set and error #16 (Drives Not Ready) will be set.

The minifloppy version functions as follows. If called with a Drive Number in the FCB of hex FF, the function will return with 0 as the drive number in the FCB. If called with a 0, it will return with the drive number set to 1. In both cases the carry is cleared on return. If called with a drive number of 1 or higher, the drive number is left unchanged, the carry bit is set on return and error #16 (Drives Not Ready) is set.

Function 21 (\$15 hex) - Position to Record N

This is one of the 2 function codes provided for random file accessing by sector. The desired record number to be accessed should be stored in the FCB location Current Record Number (a 16 bit binary value). The file must be open for read or update before using this function code. The first data record of a file is record number one. Positioning to record 0 will read in the first sector of the File Sector Map. After a successful Position operation, the first character read with a sequential read will be the first data byte of the specified record. An attempt to position to a nonexistent record will cause an error. For more information on random files, see the section titled 'Random Files'.

Example:

To position to record #6 -LDX #FCB Point to the FCB #6 Set position LDA 33,X STA Put in FCB 32,X CLR -Set M.S.B to O Setup Function Code #21 LDA STA 0,X Store in FCB Call FMS JSR **FMS** BNE ERROR Check for errors Record ready to be read

Function 22 (\$16 hex) - Backup One Record

This is also used for random file accessing. This function takes the Current Record Number in the FCB and decrements it by one. A Position to the new record is performed. This has the effect of back spacing one full record. For example, if the Current Record Number is 16 and the Backup One Record function is performed, the file would be positioned to read the first byte of record #15. The file must be open for read or update before this function may be used. See 'Random Files' section for more details.

RANDOM FILES

FLEX version 9.0 supports random files. The random access technique allows access by record number of a file and can reach any specified sector in a file, no matter how large it is, in a maximum of two disk reads. With a small calculation using the number of data bytes in a sector (252), the user may also easily reach the Nth character of a file using the same mechanism.

Not all files may be accessed in a random manner. It is necessary to create the file as a random file. The default creation mode is sequential and is what all of the standard FLEX Utilities work with. The only random file in a standard FLEX system is the ERRORS.SYS file. FLEX uses a random access technique when reporting error messages. A file which has been created as a random access file may read either randomly or sequentially. A sequential file may only be read sequentially.

To create a random file, the normal procedure for opening a file for write should be used. Immediately following a successful open, set the File Sector Map location of the FCB to any non-zero value and proceed with the file's creation. It only makes sense to create text type files in the random mode. As the file is built, the system creates a File Sector Map. This File Sector Map (FSM) is a map or directory which tells the system where each record (sector) of the file is located on the disk. The FSM is always two sectors in length and is assigned record number 0 in the file. This implies that a data file requiring 5 sectors for the data will actually be 7 sectors in length. The user has no need for the FSM sectors and they are automatically skipped when opening a file for read. The FMS uses them for the Position and Backup function code operations.

The directory information of a file states whether or not a file is a random file. If the File Sector Map byte is non-zero, the file is random, otherwise it is sequential only. It should be noted that random files can be Copied from one disk to another without losing its random properties, but it can not be appended to another file.

FLEX ERROR NUMBERS

- 1 ILLEGAL FMS FUNCTION CODE ENCOUNTERED FMS was called with a function code in the Function Code byte of the FCB that was too large or illegal.
- 2 THE REQUESTED FILE IS IN USE
 An Open for Read, Update, or Write function was issued on an FCB that is already open.
- 3 THE FILE SPECIFIED ALREADY EXISTS
 - a. An Open for Write was issued on an FCB containing the specification for a file already existing in the diskette directory.
 - b. A Rename function was issued specifying a new name that was the same as the name of a file already existing in the diskette directory.
- 4 THE SPECIFIED FILE COULD NOT BE FOUND
 An Open for Read or Update, a Rename, or a Delete function was requested on an FCB containing the file specification for a file which does not exist in the diskette directory.
- 5 SYSTEM DIRECTORY ERROR REBOOT SYSTEM Reserved for future system use.
- 6 THE SYSTEM DIRECTORY SPACE IS FULL
 This error should never occur since the directory space is self expanding, and can never be filled. Only disk space can be filled (error #7).
- 7 ALL AVAILABLE DISK SPACE HAS BEEN USED
 All of the available space on the diskette has been used up by
 files. If this error is returned by FMS, the last character sent to
 be written to a file did not actually get written.
- 8 READ PAST END OF FILE

 A read operation on a file encountered an end-of-file. All of the data in the file has been processed. This error will also be returned when reading a directory with the Get Information Record function when the end of the directory is reached.
- 9 DISK FILE READ ERROR
 A checksum error was encountered by the hardware in attempting to read a sector. DOS has already attempted to re-read the failing sector several times, without success, before reporting the error. This error may also result from illegal track and sector addresses being put in the FCB.

10 - DISK FILE WRITE ERROR

A checksum error was detected by the hardware in attempting to write a sector. DOS has already tried several times, without success, to re-write the failing sector before reporting the error. This error may also result from illegal track and sector numbers being put in the FCB. A write-error status may also be returned if a read error was detected by DOS in attempting to update the diskette directory.

11 - THE FILE OR DISK IS WRITE PROTECTED

An attempt was made to write on a diskette which has been write-protected by use of the write-enable cutout in the diskette or to a file which has the write protect bit set.

- 12 THE FILE IS PROTECTED FILE NOT DELETED

 The file attempted to be deleted has its delete protect bit set and can not be deleted.
- 13 ILLEGAL FILE CONTROL BLOCK SPECIFIED

 An attempt was made to access an FCB from the open FCB chain, but it was not in the chain.
- 14 ILLEGAL DISK ADDRESS ENCOUNTERED Reserved for future system use.
- 15 AN ILLEGAL DRIVE NUMBER WAS SPECIFIED Reserved for future system use.

16 - DRIVES NOT READY

The drive does not have a diskette in it or the door is open. This message cannot be issued for mini floppys since there is no means of detecting such a state.

- 17 THE FILE IS PROTECTED ACCESS DENIED Reserved for future system use.
- 18 SYSTEM FILE STATUS ERROR
 - a. A read or Rewind was attempted on a file which was closed, or open for write access.b. A write was attempted on a file which was closed, or open for read access.
- 19 FMS DATA INDEX RANGE ERROR

The Get Random Byte from Sector function was issued with a Random Byte number greater than 255.

- 20 FMS INACTIVE REBOOT SYSTEM Reserved for future system use.
- 21 ILLEGAL FILE SPECIFICATION

A format error was detected in a file name specification. The name must begin with a letter and contain only letters, digits, hyphens, and/or underscores. Similarly with file extensions. File names are limited to 8 characters, extensions to 3.

- 22 SYSTEM FILE CLOSE ERROR
 Reserved for future system use.
- 23 SECTOR MAP OVERFLOW DISK TOO SEGMENTED
 An attempt was made to create a very large random access file on a
 disk which is very segmented. All record information could not fit
 in the 2 sectors of the File Sector Map. Recreating the file on a
 new diskette will solve the problem.
- 24 NON-EXISTENT RECORD NUMBER SPECIFIED
 A record number larger than the last record number of the file was specified in a random position access.
- 25 RECORD NUMBER MATCH ERROR FILE DAMAGED
 The record located by the FMS random search is not the correct record. The file is probably damaged.
- 26 COMMAND SYNTAX ERROR RETYPE COMMAND
 The command line just typed has a syntax error.
- 27 THAT COMMAND IS NOT ALLOWED WHILE PRINTING

 The command just entered is not allowed to operate while the system printer spooler is activated.
- 28 WRONG HARDWARE CONFIGURATION
 This error usually implies insufficient memory installed in the computer for a particular function or trying to use the printer spooler without the hardware timer board installed.

DISK DRIVERS

The following information is for those users who wish to write their own disk drivers to interface with some other disk configuration than is supplied by the vendor. Technical Systems Consultants is not in a position to write disk drivers for other configurations, nor do they guarantee the proper functioning of FLEX with user-written drivers.

The disk drivers are the interface routines between FLEX and the hardware driving the floppy disks themselves. The drivers released with the FLEX System are designed to interface with the Western Digital 1771 or 1791 Floppy Disk Formatter/Controller chip.

The disk drivers are located in RAM at addresses \$DE00 - \$DFA0. All disk functions are vectored jumps at the beginning of this area. The disk drivers need not handle retries in case of errors; FLEX will call them as needed. If an error is detected, the routines should exit with the disk hardware status in the B-register and the CPU Z-Condition code bit clear (issue a TST B before returning to accomplish this). FLEX expects status responses as produced by the Western Digital 1771 Controller. These statuses must be simulated if some other controller is used. All drivers should return with the X,Y and U registers unchanged. All routines are enterd with a JSR instruction.

\$DE00 - Read

Entry - (X) = FCB Sector Buffer Address

(A) = Track Number

(B) = Sector Number

The sector referenced by the track and sector numbers is to be read into the Sector Buffer area of the indicated FCB.

\$DE03 - Write

Entry - (X) = FCB Sector Buffer Address

(A) = Track Number

(B) = Sector Number

The content of the Sector Buffer area of the indicated FCB is to be written to the sector referenced by the track and sector numbers.

\$DE06 - Verify

Entry - (No parameters)

The sector just written is to be verified to determine if there are CRC errors.

\$DE09 - Restore

Entry - (X) = FCB Address

Exit - CC, NE, & B=\$B if write protected

CS, NE, & B=\$F if no drive

A Restore Operation (also known as a Seek to Track 00) is to be performed on the drive whose number is in the FCB.

FLEX Advanced Programmer's Guide

\$DEOC - Drive Select
 Entry - (X) = FCB Address
 The drive whose number is in the FCB is to be selected.

\$DEOF - Check Drive Ready
 Entry - (X) = FCB Address
 Exit - NE & CS if drive not ready
 EQ & CC if drive ready

This routine is setup for FLEX systems where it is possible to check the drive whose number is in the FCB for a ready status after selecting that drive and delaying long enough for the drive motor to come up to speed (approx. 2 seconds). This is not possible in the minifloppy version due to hardware limitations. In this case, this routine should not delay and should simply return a drive ready status if the drive number in the FCB is 0 or 1 or a drive not ready status for any other drive number.

\$DE12 - Quick Check Drive Ready
This routine is the same as Drive Check Ready except the 2 second delay is not done. This assumes the drive motor is already up to speed. For minifloppy versions, there is no difference in the two and this routine can simply be a jump to the Check Drive Ready routine.

Diskette Initialization

The NEWDISK command is used to "initialize" a diskette for use by the FLEX Operating System. The initialization process writes the necessary track and sector addresses in the sectors of a "soft- sectored" diskette such as is used by FLEX. In addition, the initialization process links together all of the sectors on the diskette into a chain of available sectors.

The first track on the diskette, track 0, is special. None of the sectors on track 0 are available for data files, they are reserved for use by the FLEX system. The first two sectors contain a "boot" program which is loaded by the "D" command of the SBUG monitor or by whatever comparable ROM based bootstrap is in use. The boot program, once loaded, then loads FLEX from the diskette. Another sector on track 0 is the System Information Record. This sector contains the track and sector addresses of the beginning and ending sectors of the chain of free sectors, those available for data files. The rest of track 0 is used for the directory of file names.

After initialization, the free tracks on the diskette have a common format. The first two bytes of each sector contain the track and sector number of the next sector in the chain. The next two bytes are used to store the logical record number of the sector in the file. The remaining 252 bytes are zero. Initially, all record number bytes are zero. When data is stored in a file, the two linkage bytes at the beginning of each sector are modified to point to the next sector in the file, not the next sector in the free chain. The sectors in the diskette directory on track 0 also have linkage bytes similar to those in the free chain and data files.

A FLEX diskette is not initialized in the strict IBM standard format. In the standard format, the sectors on the diskette should be physically in the same order as they are logically, i.e. sector 2 should follow sector 1, 3 follow 2, etc. On a FLEX diskette, the sectors are interleaved so that there is time, after having read one sector, to process the data and request the next sector before it has passed under the head. If the sectors are physically adjacent, the processing time must be very short. The interleaving of the sectors allows more time for processing the data. The phenomena of missing a sector because of long processing times is called "missing revolutions", and results in very slow running time for programs. The FLEX format reduces the number of missed revolutions, thus speeding up programs.

DESCRIPTION OF A DIRECTORY SECTOR

Each sector in the directory portion of a FLEX diskette contains 10 directory entries. Each entry refers to one file on the diskette. In each sector, the first four bytes contain the sector linkage information and the next 12 bytes are not used. When reading information from the directory using the FMS Get Information Record function, these 16 bytes are skipped automatically as each sector is read; the user need not be concerned with them.

Each entry in the directory contains the exact same information that is stored in the FCB bytes 4-27. See the description of the File Control Block (FCB) for more details.

A directory entry which has never been used has a zero in the first byte of the file name. A directory entry which has been deleted has the leftmost bit of the name set (i.e. the first byte of the name is negative).

DESCRIPTION OF A DATA SECTOR

Every sector on a FLEX diskette (except the two BOOT sectors) has the following format:

Bytes 0-1 Link to the next sector Bytes 2-3 File Logical Record Number Bytes 4-255 Data

If a file occupies more than one sector, the "link to the next sector" portion contains the track and sector numbers, respectively, of the next sector in the file. These bytes are zero in the last sector of a file, indicating that no more data follows (an "end-of-file" condition). The user should never manually change the linkage bytes of a sector. These bytes are automatically managed by FMS. In fact, the user need not be concerned at all with sector linkage information.

DESCRIPTION OF A BINARY FILE

A FLEX binary file may contain anything as data; all ASCII characters are allowed. Each binary file is composed of one or more binary records. There may be more than one binary record in a single sector.

A binary record looks as follows: (byte numbers are relative to the start of the record, not the beginning of a sector)

Byte 0 Start of record indicator (\$02, the ASCII STX)

Byte 1 Most significant byte of the load address Byte 2 Least significant byte of the load address

Byte 3 Number of data bytes in the record

Byte 4-n The binary data in the record

The load address portion of a binary record contains the address where the data resided when it was written to the file with the FLEX SAVE command. When the file is loaded for execution or use, it will be put in the same memory areas from which it was SAVED.

A binary file may also contain an optional transfer address record. This record gives the address in memory of te entry point of a binary program. The format of a transfer address record is as follows:

Byte 0 Transfer Address Indicator (\$16, ASCII ACK)
Byte 1 Most significant byte of the transfer address

Byte 2 Least significant byte of the transfer address

If a file contains more than one transfer address record (caused by appending binary files which contain transfer addresses), the last one encountered by the load process is the one that is used, the others are ignored.

When reading or writing a binary file through the File Management System from a user program, the calling program must process the record indicator bytes and load addresses itself; FLEX does not supply or process this information for the user.

DESCRIPTION OF A TEXT FILE

A text file (also called an "ASCII file" or "coded file") contains only printable ASCII characters plus a few special-purpose control characters. There is no "load address" associated with a FLEX text file as there is with FLEX binary files. It is the responsibility of the program which is reading the text file to put the data where it belongs.

The only control character which FLEX recognizes and processes in a FLEX text file are:

- \$0D (ASCII CR or RETURN)
 This character is used to mark the end of a line or record in the
 file.
- \$00 (ASCII NULL)
 Ignored by FLEX; if encountered in the file, it is not returned to the calling program.
- \$18 (ASCII CANCEL)
 Ignored by FLEX; if encountered in the file, it is not returned to the calling program.
- \$09 (ASCII HT or HORIZONTAL TAB)

 This is a flag character which indicates that a string of spaces has been removed from the file as a space-saving measure. The next byte following the flag character is a count of the number of space removed (2-127). The calling program sees neither the flag character nor the count character. The proper number of spaces are returned to the user program as successive characters are requested by the Read Next Byte function. When writing a file, the spaces are automatically deleted as the user program sends them to the File Management System using the Write Next Byte function. The data compression is, therefore, transparent to the calling program. (The above discussion is only valid if the file is open for Text operations. If open for Binary, the compression flag and count get passed exactly as they appear in the file.)

WRITING UTILITY COMMANDS

Utility commands are best prepared by the use of an assembler. FLEX reserves a block of memory in which medium size utilities may be placed. This memory starts at hex location \$C100 and extends through location \$C6FF. The system FCB at location \$C840 may also be used in user written utilities for either FCB space or temporary storage. No actual code should reside in this FCB space since it would interfere with the loading of the utility (FLEX is using that FCB while loading utilities).

An example will be given to demonstrate some of the conventions and techniques which should be used when writing utilities. The example, which can be found on the following pages, is a simple text file listing utility. Its syntax is:

LIST,[<FILE SPEC>]

The default extension on the file spec is TXT. The utility will simply display the contents of a text file on the terminal, line for line.

The following is a section by section description of the LIST utility. The first section of the source listing is a set of EQUATES which tell the assembler where the various DOS routines reside in memory. These equates represent the addresses given in this manual for "User Callable DOS System Routines".

The next two sections are also equates, the first to the FMS entry points, and the second references the system FCB. The actual program finally starts with the ORG statement. In this program, we will make use of the Utility Command space located at \$C100, therefore, the ORG is to \$C100.

One of the conventions which should be observed when writing DOS utilities is to always start the program with a BRA instruction. Following this instruction should be a 'VN FCB 1' which defines the version number of the utility. The 1 should of course be set to whatever the actual version number is. In this example, the version number is 1. This convention allows the FLEX VERSION Utility to correctly identify the version number of a command.

Moving down the program to the label called 'LIST2', the program needs to retrieve the file specification and get it into the FCB. Pointing X to the FCB, we can make use of the DOS resident subroutine called 'GETFIL' to automatically parse the file spec, check for errors, and set the name in the FCB correctly. If all goes well in GETFIL, the carry should be clear, otherwise there were errors in the file spec and this fact needs reported. If the carry is set, control is passed to the line with the label 'LIST9'. At this point, the error message is reported and control is returned to FLEX.

If the file spec was correct, and the carry was clear after the return from GETFIL, we want to set a default file name extension of TXT. The DOS subroutine named SETEXT will do exactly that. First it is necessary

to put the code for TXT in the A accumulator (the code is 1). X needs to be pointing to the FCB which it still is. The '1' is also put in the FCB for the future open operation. The call is made to SETEXT and the file name is now correctly set up in the FCB. Note that no errors can be generated by a call to SETEXT.

Now that we have the file spec, it is necessary to open the requested file for read. X is still pointing to the FCB so it is not necessary to The FMS Function Code for 'open a file for read' is 1 which was previously put in the FCB location O. A call to FMS is now made in an attempt to open the file. Upon return, if the Z-condition code is set, there were no errors. If there was an error, the 'BNE LIST9' will take us to the code to report the error. This section of code is the desired way to handle most FMS caused disk errors. The first thing to do is call the DOS routine RPTERR which will print the disk error message on the monitor terminal. Next, all open disk files should be closed. This can be easily accomplished by a call to the FMS close entry (FMSCLS). Finally, return control back to DOS by jumping to the WARM START entry. If the file opened successfully, control will be transferred to the line with the label 'LIST4'. At this time it is desirable to fetch characters one at a time from the file, printing them on the monitor terminal as they are received. Since line feeds are not stored in text files (carriage returns mark the end of lines, but the next line will follow immediately), each carriage return received from the file is not output as is, but instead a call to the DOS routine 'PCRLF' is made to print a carriage return and a line feed. As each character is received from the file (by a call to FMS at label LIST4), the error status is checked. If an error does occur, control is transferred to 'LIST6'. Since FLEX does not store an End of File character with a file, the only mechanism for determining the end of a file is by the End of File error generated by FMS. At 'LIST6', the error status is checked to see if it is 8 (end of file status). If it is not an 8, control is transferred to the error handling routine described above. If it is an End of File, we are finished listing the file so it must now be closed. Function Code for closing a file is 4. This is loaded into A and stored in the FCB. Calling FMS will attempt to close the file. Upon return, errors are checked, and if none found, control is transfered back to DOS by the jump to 'WARMS'.

This example illustrates many of the methods used when writing utilities. Many of the DOS and FMS routines were used. The basic idea of file opening and closing were demonstrated, as well as file I/O. The methods of dealing with various types of errors were also presented. Studying this example until it is thoroughly understood will make writing your own disk commands and disk oriented programs an easy task.

*

* SIMPLE TEXT FILE LIST UTILITY

*

* COPYRIGHT (C) 1979 BY

*

* TECHNICAL SYSTEMS CONSULTANTS, INC.

* DOS EQUATES

CD03	WARMS	EQU	\$CD03	DOS WARMS START ENTRY
CD2D	GETFIL	EQU	\$CD2D	GET FILE SPECIFICATION
CD18	PUTCHR	EQU	\$CD18	PUT CHARACTER ROUTINE
CD24	PCRLF	EQU	\$CD24	PRINT CR & LF
CD33	SETEXT	EQU	\$CD33	SET DEFAULT NAME EXT
CD3F	RPTERR	EQU	\$CD3F	REPORT DISK ERROR

* FMS EQUATES

D406 FMS EQU \$D406 D403 FMSCLS EQU \$D403

* SYSTEM EQUATES

C840 FCB EQU \$C840 SYSTEM FCB

* LIST UTILITY STARTS HERE

C100 20	C100			ORG	\$C100	
C103 8E C840 LIST2 LDX #FCB POINT TO FCB C106 BD CD2D JSR GETFIL GET FILE SPEC C109 25 34 BCS LIST9 ANY ERRORS? C10B 86 01 LDA #1 SET UP CODE C10D A7 84 STA O,X SAVE FOR READ OPEN C10F BD CD33 JSR SETEXT SET TXT EXTENSION C112 BD D406 JSR FMS CALL FMS - DO OPEN C115 26 28 BNE LIST9 CHECK FOR ERROR C117 8E C840 LIST4 LDX #FCB POINT TO FCB C11A BD D406 JSR FMS CALL FMS - GET CHAR C11D 26 OE BNE LIST6 ERRORS? C11F 81 OD CMPA #\$D IS CHAR A CR? C121 26 05 BNE LIST5	C100 20	01	LIST	BRA	LIST2	GET AROUND TEMPS
C106 BD	C102 01		VN	FCB	1	VERSION NUMBER
C126 20 EF BRA LIST4 REPEAT C128 BD CD18 LIST5 JSR PUTCHR OUTPUT THE CHARACTER C12B 20 EA BRA LIST4 REPEAT SEQUENCE	C106 BD C109 25 C10B 86 C10D A7 C10F BD C112 BD C115 26 C117 8E C11A BD C11D 26 C11F 81 C121 26 C123 BD C126 20 C128 BD	CD2D 34 01 84 CD33 D406 28 C840 D406 0E 0D 05 CD24 EF CD18	LIST4	JSR BCS LDA STA JSR JSR BNE LDX JSR BNE CMPA BNE JSR BRA JSR	GETFIL LIST9 #1 0,X SETEXT FMS LIST9 #FCB FMS LIST6 #\$D LIST5 PCRLF LIST4 PUTCHR	GET FILE SPEC ANY ERRORS? SET UP CODE SAVE FOR READ OPEN SET TXT EXTENSION CALL FMS - DO OPEN CHECK FOR ERROR POINT TO FCB CALL FMS - GET CHAR ERRORS? IS CHAR A CR? OUTPUT CR & LF REPEAT OUTPUT THE CHARACTER

FLEX Advanced Programmer's Guide

C12D A6 C12F 81 C131 26	01 08 0C	LIST6	LDA CMPA BNE	1,X #8 LIST9	GET ERROR STATUS IS IT EOF ERROR?
C133 86	04		LDA	#4	CLOSE FILE CODE
C135 A7	84		STA	0 , X	STORE IN FCB
C137 BD	D406		JSR	FMS	CALL FMS - CLOSE FILE
C13A 26	03		BNE	LIST9	ERRORS?
C13C 7E	CD03		JMP	WARMS	RETURN TO FLEX
C13F BD	CD3F	LIST9	JSR	RPTERR	REPORT ERROR
C142 BD	D403		JSR	FMSCLS	CLOSE ALL FILES
C145 7E	CD03		JMP	WARMS	RETURN TO FLEX
			END	LIST	

THE DOS LINK UTILITY

The LINK Utility provided with FLEX is a special purpose command. Its only function is to inform the "disk boot", which is on track 0, where the program resides which is to be loaded during the boot operation. Normally, LINK is used to set the pointer to the DOS program. Since DOS may reside anywhere on the disk, LINK takes the starting disk address of the file and stores it in a pointer in the boot sector. When the boot program is later executed, it simply takes this disk address, and loads the binary file which resides at that location. The load process is terminated upon the receipt of a transfer address record. At this time, control is transferred to the program just loaded by jumping to the address specified in the transfer address record. If the 'linked' program is ever moved on the disk, then it must be re-linked so the boot knows the new disk address.

LINK may be used in some specialized applications. One is the development of custom operating systems. The user may write his own operating system, link it to the boot, and use it exactly as FLEX is used now. It may also be desirable for special disks to boot in specialized programs rather than the operating system. If this is done, remember that unless the DOS is loaded during the boot process, there will not be any disk drivers or File Management System resident in memory.

PRINTER ROUTINES

There are two printer related programs provided with FLEX. One is the P Utility, the other is the PRINT.SYS file which is the actual set of printer drivers (initialize printer and output character). The P command source listing is provided on the following pages and should be self explanatory. Below you will find the requirements of the PRINT.SYS file. No source listing is provided here since one is given in the "FLEX User's Manual".

'PRINT.SYS' FILE REQUIREMENTS

The PRINT.SYS file needs to provide the system with three basic printer routines, one for printer port initialization, one for printer status, and one for output character to printer routine. The P routine and the system printer spooler use these routines to communicate with the printer. A source listing of the provided routines are included in the "FLEX User's Manual" and will not be duplicated here. The three routines and their requirements are listed here.

PINIT (\$CCCO-CCD7) This routine should initialize the printer port. No registers need be preserved.

PCHK (\$CCD8-CCE3) This routine should check to see if the printer can accept another character. Return Negative CC status if can accept, Plus if can not. Preserve A, B, X, Y, and U.

POUT (\$CCE4-CCF7) This routine should output the character in A after calling PCHK to verify the printer can accept the character. Preserve B, X, Y, and U.

THE SYSTEM PRINTER SPOOLER

FLEX contains a printer spooler module. It requires the installation of an interval timer board for operation. Essentially, the spooler is a multi-tasking system, with the output to printer function being a low priority task. Any requested disk service will cause the printer task to temporarily halt until the disk has been used. It should be noted that the SWI3 CPU vector is adjusted in this task scheduler. The PRINT command is used to activate the spooler which in turn prints the files (if any) in the print queue. Exact details of the spooling operation are not available at this time.

```
"P" UTILITY COMMAND
                    * THE P COMMAND INITIALIZES A PORT AND
                    * CHANGES THE OUTCH JUMP VECTOR IN FLEX
                    * COPYRIGHT (C) 1979 BY
                    * TECHNICAL SYSTEMS CONSULTANTS, INC.
                    * EQUATES
              C840
                    FCB
                            EQU
                                    $C840
                                    $CD30
              CD30
                    LOAD
                            EQU
             D406 FMS
                                    $D406
                            EQU
              D403
                   FMSCLS
                            EQU
                                    $D403
                                    $CD06
              CD06
                    RENTER
                            EQU
                            EQU
                                    $4
              0004
                    NFER
                            EQU
                                    $CCO9
              CC09
                    PAUSE
             CD1E
                    PSTRNG
                            EQU
                                    $CD1E
              CD3F
                    RPTERR
                            EQU
                                    $CD3F
             CD03 WARMS
                            EQU
                                    $CD03
                                    $CC11
             CC11 LSTTRM
                            EOU
                            EQU
                                    $CC02
             CCO2 EOL
             CCCO
                    PINIT
                            EQU
                                    $CCC0
             CCE4
                    POUT
                            EQU
                                    $CCE4
                            EQU
                                    $CDOF
             CDOF
                    OUTCH
                                    $CCFC
             CCFC
                    PR1
                            EQU
                                    $C100
C100
                            ORG
                    Р
                            BRA
                                    P1
                                              BRANCH AROUND TEMPS
C100 20
          01
                                              VERSION NUMBER
                                    1
C102 01
                    VN
                            FCB
                                              CHECK SYSTEM PROCESS REG
C103 B6
          CCFC
                    P1
                            LDA
                                    PR1
                            BEQ
                                    P12
                                              IS IT BUSY?
C106 27
          09
                                    #FCB
                                              POINT TO FCB
          C840
                            LDX
C108 8E
                            LDB
                                    #27
                                              SET BUSY ERROR
C10B C6
          1B
                                              STUFF IN FCB
                                    1,X
C10D E7
          01
                            STB
                                              GO REPORT ERROR
C10F 20
          45
                            BRA
                                    Р3
C111 B6
          CC11
                    P12
                            LDA
                                    LSTTRM
                                              GET LAST TERMINATOR
                            CMPA
                                    #$D
                                              IS IT A CR?
C114 81
          OD
                                    P8
C116 27
          47
                            BEQ
                                              IS IT EOL CHARACTER?
                            CMPA
                                    E0L
C118 B1
          CC02
C11B 27
           42
                            BEQ
                                    Р8
```

PAUSE

CLR

C11D 7F

CC09

DISABLE THE PAUSE FEATURE

⁻ continued -

C120 B6 C123 81 C125 26 C127 8E C12A 86 C12C A7 C12E BD C131 26 C133 86 C135 A7 C138 BD C13B BD C13E 8E C141 BF C144 7E	CCE4 39 14 C840 01 84 D406 14 FF 88 3B CD30 CCC0 CCE4 CD10 CD06	P15	LDA CMPA BNE LDX LDA STA JSR BNE LDA STA JSR LDX STX JMP	POUT #\$39 P15 #FCB #1 O,X FMS P2 #\$FF 59,X LOAD PINIT #POUT OUTCH+1 RENTER	GO INITIALIZE PORT GET OUTPUT ADDRESS
C147 A6 C149 81 C14B 26 C14D 30 C151 BD C154 20	01 04 09 8D 0014 CD1E 03	P2 P25	LDA CMPA BNE LEAX JSR BRA	1,X #NFER P3 NOPST,PCR PSTRNG P4	GET ERROR CODE IS IT "NO SUCH FILE"? POINT TO MESSAGE GO PRINT IT
C156 BD C159 BD C15C 7E	CD3F D403 CD03	P3 P4	JSR JSR JMP	RPTERR FMSCLS WARMS	REPORT ERROR CLOSE ALL FILES RETURN TO FLEX
C15F 30 C163 20	8D 0018 EC	P8	LEAX BRA	ERSTR,PCR P25	POINT TO STRING GO PRINT IT
C17A 04	50 52 49 50 22 20	NOPST ERSTR	FCC FCB FCC FCB	4	YS" NOT FOUND' BE FOLLOWED BY A COMMAND'

^{*} THE FOLLOWING CODE IS LOADED INTO * THE SYSTEM FCB WHEN THE P COMMAND IS * LOADED INTO MEMORY.

^{*} IT PRESETS THE FILE NAME IN THE FCB.

C843	ORG	\$C843
C843 FF C844 50 52 49 4E C849 00 00 00 C84C 53 59 53	FCB FCC FCB FCC	\$FF 'PRINT' 0,0,0 'SYS'
	END	Р

INTERRUPTS IN FLEX

FLEX makes extensive use of interrupts during printer spooling. Anytime there are files in the PRINT Queue (as a result of using the PRINT command) the timer board (MP-T in I/O slot #4) is activated. This board is initialized to output interrupts every 10 milliseconds. These are IRQ type interrupts and FLEX sets the IRQ vector to point to its IRQ routine. When the PRINT Queue is empty, the timer is shut off and no interrupts are generated. The SWI3 instruction is also used quite extensively in FLEX. The SWI3 vector in RAM is set by FLEX to point to its SWI3 routine. Because of the SWI3 and IRQ use, the MON command will not permit leaving FLEX while there is a file in the PRINT Queue.

All FLEX utilities, the Editor, the Assembler, the Text Processor, and BASIC are interruptable programs. When writing your own programs, if they are to be used while printing with the PRINT command (files in the print queue), they should be written to be interruptable as well. At no time should the IRQ or SWI3 vectors be changed in a utility which is to be run while printing. In general, good programming practice will yield interruptable programs.

SYSTEM MEMORY MAP

The following memory map shows the location of user RAM and several major sections of the FLEX operating system. All addresses are in hexadecimal.

ADDRESS	DESCRIPTION
0000 - BFFF	User RAM (Some of the lower end of this area is used
	by certain utilities such as NEWDISK.)
C000 - C07F	Stack Area (SP is initialized to CO7F)
C080 - C0FF	Input Buffer
C100 - C6FF	Utility Command Area
C700 - C83F	Scheduler & Printer Spooler
C840 - C97F	System FCB
C980 - CBFF	System Files Area
CCOO - D3FF	DOS
D400 - DDFF	FMS
DEOO - DFFF	Disk Drivers

and the control of t