KGB 20/8/86 FLEX

FLEX EXPLAINED

Paul Izod and Alan Stirling look at the industry standard FLEX OS with particular reference to the *E&CM* hi-res computer system.

It has often been pointed out in this series of articles that FLEX only comes to life when disc drives are added to a system. Last month we described the disc controller card for the *E&CM* computer system and this month we look at the inplementation and operation of the FLEX OS on the system.

Before going into the details of FLEX, it is necessary however to discuss the general features of single-user operating systems. Since these all require the use of disk drives, a brief explanation of the means of data storage on a disk is appropriate.

The Data Format

The data areas on all disks are split up into circular tracks on the disk, with each of these tracks sub-divided into sectors – normally between 10 and 32 per track. Each sector holds between 128 and 1024 bytes of information. The smaller the number of bytes per sector, the more sectors per track. A 5¼" disk holds about 2,560 data bytes per track in single density. A 40 track single-sided/single-density 5¼" disk has a total capacity of 102,400 bytes. The start of each track is marked by an index hole in the disk near it's hub.

There are two systems used to correctly identify the sectors around each track. The most popular, called 'Soft Sectored', uses a small header area recorded onto the disk, in front of each sector, which holds the track and sector number of the sector that follows (see **Fig 1**). This information is recorded at the time that the disk is formatted, since without it the disk drive head assembly is not able to check its position on the disk. The other method, called 'Hard Sectored', used extra holes around the hub of the disk to mark the start of each sector. Extra circuitry counts pulses generated by these holes and generates the sector numbers. Soft sectored disks operate more reliably, since each sector is uniquely coded, but this is at the expense of total data capacity, as the header information takes up valuable data information space on the disk.

The interface between the computer system, and the disk drive is undertaken by a disk controller card. The key component on this board is the disk controller chip, which controls the operation of the disk drives completely. Since this chip is itself as complex as a small microprocessor, it only requires a few support chips to fully control most types of disk drives. It is the task of the disk controller card to communicate with the disk drive, reading or writing sectors of data as required. Only complete sectors of information are transferred.

Due to the intelligence of the disk controller chip, it is only necessary to write short machine code routines (called "Drivers") to gain access to any individual sector. With the addition of some form of disk drive selection hardware, it becomes a simple matter to read sectors from one disk drive and write them to another, thus copying data between two disks. Without a detailed record of where data is stored on each disk however, this basic system is unable to ensure that the sectors that it is reading contain the required data and the sectors that it is writing to do not already contain valuable information.

What is needed is a system of indexing the data stored on a particular disk. Although this index or directory can be held in memory, the obvious place to keep it is on the disk itself, alongside the data. In all disk operating systems, the start of this directory is defined, so that the machine knows where to start searching for the name allocated to the data for which it is searching. Alongside the name entry in the directory is further information about the data. This includes the track and sector where the data starts and finishes, the number of sectors that the data occupies and the type of data itself (i.e. machine code programs, text files, basic programs etc.). Data in this format is normally known as a file.

It is for the control and manipulation of these files and their

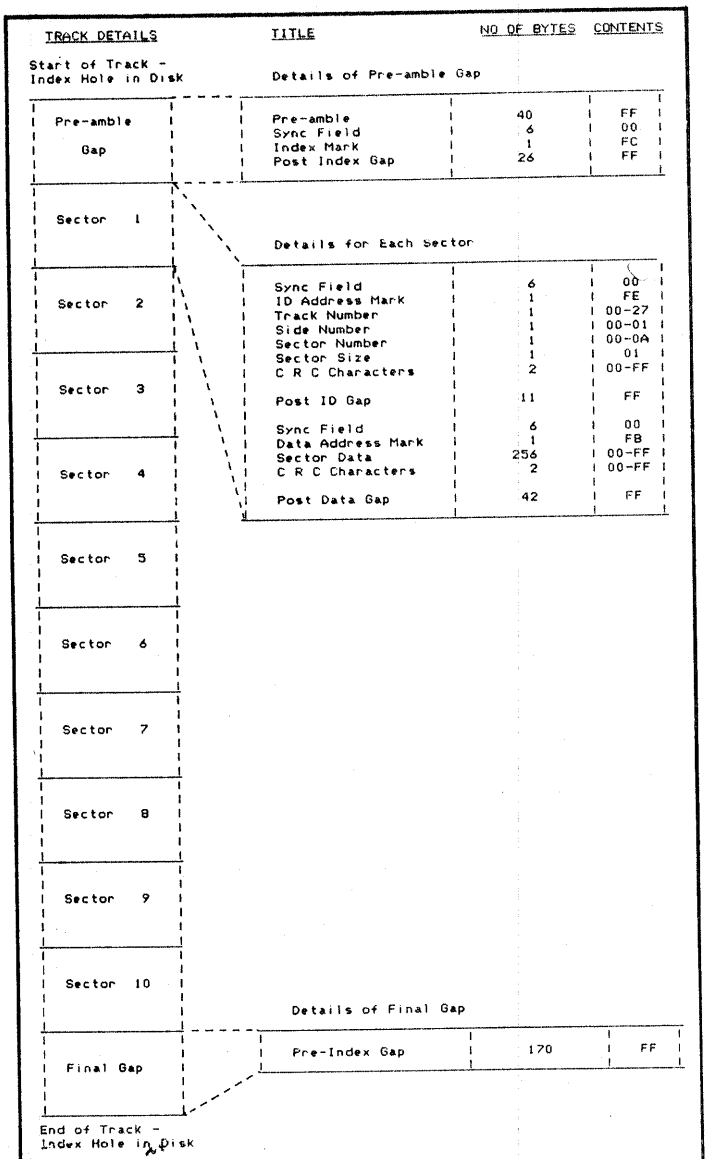
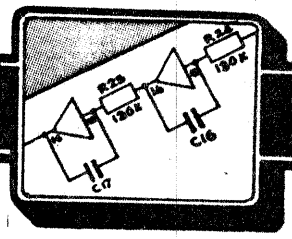


Figure 1. Standard IBM34 Format for 256 byte per sector for 5¼" floppy discs.

relevant directory entries that disk operating systems (DOS) have been developed. However since disks using the same DOS will normally be compatible, meaning that disks recorded on one systemn can be read by another – even if the two systems are produced by different manufacturers, most operating systems go further, providing a completely compatible software environment within each machine, so that programs written to run on one type of machine can be transferred on compatible disks to another machine, giving identical results when run.

Software Compatibility

This is obtained using standard routines within the DOS. The address of these routines and their functions is provided to the programmer within the documentation of the DOS. The addresses and functions of



these routines do not change between different types of machines, providing the DOS is the same. Any differences between the machines hardware configurations, such as the address of the I/O port or memory mapped screen are taken into account when the DOS is implemented on that machine. During the implementation, machine code routines are used to link the DOS to the I/O routines of the machine. It is important that these routines are written such that they allow the DOS to perform in identical ways on differing machines. Programs written to link with the standard addresses defined in the DOS therefore should have the same results on any system. It is difficult however, to write sophisticated output routines without knowing the exact screen size and format. Thus the standardisation is limited to programs with simple I/O requirements– programs with complicated graphics requirements will only run on similar hardware, since this will be accessed directly by the program.

An Overview Of FLEX

FLEX was originally written in 1977 for systems using the Motorola MC6800 processor, by Technical Systems Consultants Inc, of Forest Hill, North Carolina, USA. In 1977 it was converted to run on the MC6809 and has become a standard for these machines. It is used on many makes of machine, including SWTPC, Positron, Gimix, Smoke Signal and Mororola, with implementations on Apple II, Tandy Colour Computer and Hitachi machines. With many users world-wide there is a wide range of system software with a number of good applications packages available. With the recent adaptation for the Dragon 32, this operating system will become even more popular.

FLEX Disk Formats

Although many other operating systems use sectors of 128 bytes; FLEX uses 256 bytes per sector. This gives 10 sectors per track on 5¼" single sided/single density (SS/SD) disks, or 15 sectors on 8" SS/SD. In FLEX the sectors are numbered from 01 (Hex) upwards and the first and outermost track is numbered 00 (Hex). **Table 1** gives details of the track, sector & capacities for the more popular disk formats.

The figures in this table reflect the total amount of data that can be stored in practice, since the theoretical maximum can never be reached since track zero on each disk is reserved for the directory and other information, sectors themselves only hold 252 bytes of data, and track zero is always recorded in single density, even on double density disks in order to maintain compatibility with single density systems.

FLEX File Format

Files can be of any length, and are made up from a collection of linked sectors. These are linked by the first two bytes of each sector, which hold the track and sector number of the next sector of the file. In this way a file can occupy any unused sectors on the disk, since successive sectors do not have to be physically consecutive. The last sector of a file has these two link bytes set to zero. On most files the following two bytes hold the consecutive sector number within the file, starting with sector 1 as the first sector. Since each sector has 256 bytes and four are used by the system, there remain 252 bytes available to the user. This slight loss in storage capacity is outweighed by the flexibility of file layout on the disk. Even the directory information on track 0 uses the same format.

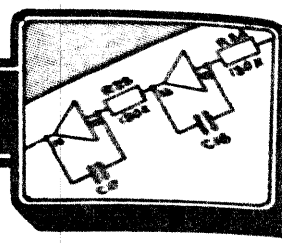
All empty or unusued sectors on the disk are linked as one large file, called the 'Free Chain'. This file also uses the track and sector links, but does not maintain a consecutive sector number.

When a file is to be written to the disk, FLEX removes the first sector from the free chain, and allocates it as the first sector of the new file. As the new file requires more sectors, FLEX re-allocates them from the free chain to the new file. They are already linked toghether, since they were linked in the free chain. At the end of the file, FLEX un-links the last sector of the file, (track and sector equal to zero), updates the directory entry for the file, and and changes the pointer to the start of the free chain. If a file is deleted, it is linked to the end of the free chain and it's entry is deleted from the directory. Although it has been deleted, with it's directory entry removed, the data will still be intact, until overwritten by another file. The disk is full when the free chain has no further free sectors.

TABLE 1 – Track, Sector and Byte Capacities of Different Sizes of FLEX Disks.

Disk Size	No. of Sides	Data Density	No. of Tracks	Largest Track Track	Largest Sector (Dec/Hex)	Total Disk Secotrs	Total Data Sectors	Total Data Bytes
5"	1	Single	40	39/27	10/0A	400	390	98280
5"	1	Double	40	39/27	18/12	712	702	176904
5"	2	Single	40	39/27	20/14	800	780	196560
5"	2	Double	40	39/27	36/24	1424	1404	353808
5"	1	Single	80	79/4F	10/0A	800	790	199080
5"	1	Double	80	79/4F	18/12	1432	1422	358344
5"	2	Single	80	79/4F	20/14	1600	1580	398160
5"	2	Double	80	79/4F	36/24	2864	2844	716688
8"	1	Single	77	76/4C	15/0F	1155	1140	287280
8"	1	Double	77	76/4C	26/1A	1991	1976	497952
8"	2	Single	77	76/4C	30/1E	2310	2280	574560
8"	2	Double	77	76/4C	52/34	3982	3952	995904

(Dec: Decimal, Hex: Hexadecimal)



Track Zero Information

This track has a special significance since it holds the disk directory and certain other important information.

The first sector holds the boot loader program. This program is machine specific, and is called into the machine by the monitor program, when FLEX is booted. Once loaded into the machine, it loads FLEX into memory, from wherever it is located on the disk. Finally, it passes control to FLEX, which begins operation. If the boot program is too large to fit into one sector, the second sector may also be used.

The third sector is called the 'System Information Record' (SIR). This holds information regarding the disk itself, and the chain of free sectors on the disk. See **Table 2.** This information is read by FLEX prior to accessing a disk file, so that FLEX can adapt to the individual disk's format.

TABLE 2 – List of Information held on System Information Record (SIR).

Volume Name of Disk	(8 Characters)
Volume Number of Disk	(1 – 65535)
Creation Date	(MM/DD/YY)
Start of Free Chain	(Track/Sector)
End of Free Chain	(Track/Sector)
Length of Free Chain	(No of Sectors)
Highest Sector Address on Disk	(Track/Sector)

Sector 4 is left blank, and is reserved for future expansion.

The directory file starts at sector 5, and continues through all the other sectors on track zero. Each directory entry takes 21 bytes, giving 12 entries per sector. If more directory entries are required, over and above those available on track zero, FLEX automatically allocates extra sectors as required from the free chain.

FLEX File Specifications

As with all operating systems, there is a structured syntax to be complied with, when entering the details of a disk file to be used. FLEX numbers the disk drives from '0' to '3', this number being used at the beginning of the file name. The file name itself must not be more than 8 characters. FLEX also allows files to use extension names of up to three letters, in order to describe the file type. The main extensions used are:

BIN	Binary Program Files
TXT	Text files
CMD	FLEX Command Files
BAS	Basic Source Code Files
SYS	FLEX System Files
BAK	Back-up Copy of Text File
BAC	Basic Compiled (Tokenised) Files
OUT	Printer-Spooling Output Files

The drive number normally precedes the file name, whereas the extension follows it. They are separated by a full stop, thus '1.LETTER.TXT', is a text file on drive '1', called 'LETTER'. Many commands presume default options in terms of drive number and extension, thus in most situations it is only necessary to enter the file name alone.

FLEX Command Structure

There are only two memory resident commands within FLEX although others can be added if required. 'GET' is used to load binary file into memory, where it will be loaded into the same memor locations from which it was saved. 'MON' is used to exit fro FLEX, back to the system monitor. All other commands are loade from disk. To initiate a command, simply enter the name of command file. FLEX will look for a file with the same name, on th system disk, (normally Drive '0'), and with the extension of '.CMD' signifying a command file. Thus entering 'CAT' will loade th catalogue command from drive '0'. Its full file specification i '0.CAT.CMD'. Any of the default options can be changed merely b entering the option desired. For example to run a binary program o drive '1', called 'TEST', enter '1.TEST.BIN'.

The system also controls the default options on two drives, known as the 'System Drive' and the 'Working Drive'. The system drive i where FLEX looks for command files, and the working drive for tex and other files. Thus if the system drive is set to '0' and the working drive to '1', the command 'LIST LETTER', will load the command file '0.LIST.CMD', which will in turn list on the screen the contents of the text file '1.LETTER.TXT'. These default drive numbers can be controlled using the 'ASN' (Assign) command. Using this same philosophy, many commands use a string of parameters entered after the command name in order to control the detailed operation of the command.

Within this simple framework, it is possible to control the operation of a sophisticated 6809 based computer, particularly if the range of commands is augmented, which is done by just adding the command file to the system disk.

E&CM

Next Month: More on Flex and the *E&CM* Hi-Res Computer.