Floppy disc system for the scientific computer — 1

8in disc stores 400K bytes

by J. H. Adams, B.Sc. M.Sc.

Storage of data in small computer systems is often accomplished by a 300-baud cassette tape recording. With a transfer rate of only 2K-bytes per minute, this method makes locating and transferring long strings of data a rather slow process. The introduction, through the users' club[1], of a more advanced operating system for the computer[2], and the availability of memory expansion kits, has made a faster store very desirable. To solve this problem the author has developed a store based on an 8in flexible (floppy) disc, which can accommodate 400K-bytes of data and transfer 0.5K-bytes per second.

Recordings on disc are made by converting the data bytes into a serial stream of 1s and 0s at a rate of 250,000 bits per second, i.e. one bit every 4μs, truncating the 1s down to about 0.5μs pulses and then interleaving a regular stream of 0.5μs pulses from the system clock as shown in Fig. 1. These pulses are used to reverse the current in the recording head and, hence, the sign of the flux recorded onto the disc. Converting the parallel input data to a stream of pulses is most easily achieved by one of the controller i.cs which are, in essence, dedicated microprocessors combined with programmed logic arrays to feed control information between the controller, the disc-drive electronics and the computer. Recordings are made on concentric rings, or tracks, 77 on an 8in disc, and a drive unit with two motors rotates the disc and steps the combined record/read/erase head from track to track. Optical devices provide signals which indicate when the head is over the outermost track 0 and, using a small hole punched in the disc, an index pulse to indicate when the disc starts each revolution.

The electronics in the drive unit convert t.t.l. levels to switching currents in the head and vice versa, operate the stepping motor and provide erasing signals. Other functions may include door locking, motor-on indication, adjustment of recording current on inner tracks, separation of data, disabling the write operation on write-protected discs, and loading the head against the disc on read operations. This drive unit contains most of these features, although separation of data is achieved in the controller i.c.

Recording format

At 360 r.p.m. it is possible to record over 5000 bytes of data on each track of the disk. To allow the controller to identify recorded clock pulses from serial data pulses from the disc, the start of the decoding process is triggered by the index pulse, and the recording begins with a standard code which the controller can recognise and synchronize with. This code is often produced by repetitive recording of the byte 00, i.e. the clock pulses are recorded with no interleaved data pulses. The next task for the controller is recognition of the start of the first byte in the data stream. As all possible data bytes may start the stream, no single byte can be reserved for this purpose. Instead, a data byte with a few of the clock pulses missing is used and is known as a mark byte. Normal bytes can be thought of as data bytes interleaved with the clock byte FF, i.e. all eight pulses. A typical mark byte is data byte FC, interleaved by the clock byte C7. After this index mark, about 5000 bytes of data follow and the recording runs to the start of the next index pulse with a final code of bytes, usually 00s or FFs. The total number of code bytes is determined by the accuracy of the clock and drive motor.

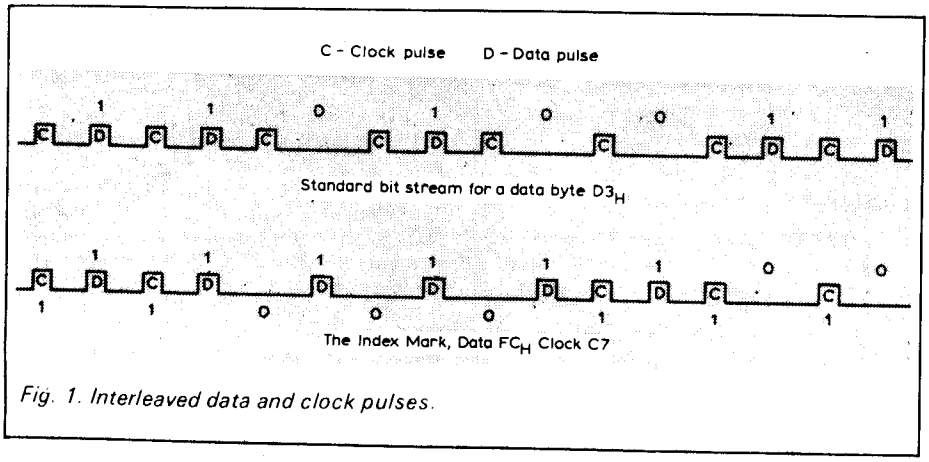


Fig. 1. Interleaved data and clock pulses.

Sectored tracks

If data transfers, which match the above, are all that is required of the disc, it is an efficient way of using the system in terms of bytes stored per disc. Usually, however, transfers are of variable length and, as it is not directly possible to access part of the way through a track, there is a limit of one recording per track, no matter how short the data block. To improve the potential disc capacity, each track is split into sectors which each require start and stop codes and identification marks. This leaves less space for data, but normally provides the most efficient mode. Such a format, now widely in use, is the IBM 3740 which fits 26 data sectors into each track, with 128 data bytes in each sector as illustrated in table 1.

In the present format, sectors consist of an identifying block followed by the data. Six 00s synchronize the clock/data separator, an address mark (data FE, clock C7) indicates the boundaries of the bytes, and, as previously explained, track and sector numbers are given. This is followed by a CRC, which is a two-byte cyclic recognition code used by the controller to check for errors when reading information. The sector then has a short code, immediately followed by six more 00s, a data mark (data FB, clock C7), the 128 bytes of data, a two byte CRC for the data, and a final code. Each track has 26 of these sectors end to end, prefixed by a large block of 00s, an index mark (data FC, clock C7), and trailed to the end of the track by a code. The copious supply of synchronizing bytes and CRC codes can, with suitable software in the computer, produce a very reliable system.

Formatting all of this information onto the disc is a complicated operation,

IBM 3740 sector

6 bytes 00	Ident mark	Track no.	00	Sector no.	00	CRC 2 bytes	17 bytes 00	Data mark	Data 128 bytes	CRC 2 bytes	27 bytes FF
Ident field							Data field				

IBM 3740 track

40 bytes FF	6 bytes 00	Index mark	26 bytes FF	26 sectors	approx. 240 bytes FF

Table 1. Formatted disc arrangement.

and most discs are supplied formatted with dummy data (usually byte E5). Such discs are marked 128 bytes per sector or per record, soft sectored. The last mentioned term means that the start of the sectors is indicated by software recorded onto the disc, as opposed to permanent hard sectoring, achieved by punching index holes in the disc for each sector (sometimes called 33 hole media for this reason). One disadvantage of soft sectoring is that, if formatting information gets magnetically corrupted, the sectors affected become useless. For this reason, even unused discs should be treated with care. Fortunately, if this should happen, the controller can re-format tracks to this and a number of other formats.

Computer-controller interface

To the computer, the controller looks like four input and four output ports. However, to address the controller, the computer only needs to supply one line each from IC_2 and IC_3, (the computer's input and output-port decoders) along with address lines A_3 and A_4 as shown in Fig. 2. Because neither of the address lines go to the decoding i.cs, they take no part in decoding the 8-bit port addresses which the Z80 sends along the bottom eight address lines during I/O instructions. Therefore, I/O commands such as IN(05), IN(0D), IN(15) and IN(1D) will activate the same line from IC_2, the bottom three bits of each number being the same, 101, but each provides a different combination on A_3 and A_4. By connecting the IN line to the controller's $\overline{RE}$ (read enable) input, and the two address lines to the A_0 and A_1 inputs, all four controller registers may be read by the computer. In a similar fashion, one line from IC_3 drives the $\overline{WE}$ (write enable) line of the controller, which allows the computer to write information into any of the registers. For details of these see Table 2.

As well as the data bus into and out of the device, and the four control lines described above, there are two lines from the controller to the computer. One indicates that, either through natural completion or through a failure, the controller has finished an operation and wants servicing, $\overline{INTRQ}$, the other, $\overline{DRQ}$, indicates that the controller desires a data transfer either to or from it. This information is present in the status register but, because of the high rates of data transfer taking place, these lines must be used to enter the Z80, through the interrupt line, in preference to the much slower polling of the status register. The Z80 can therefore keep up with the steady demand for, or supply of data between it and the controller. For this reason, part of the interface consists of a simple but effective interrupt controller.

Controller disc-drive interface

Lines from the controller to the drive comprise step and direction signals for the head-position motor, data and gating signals for the write operation,

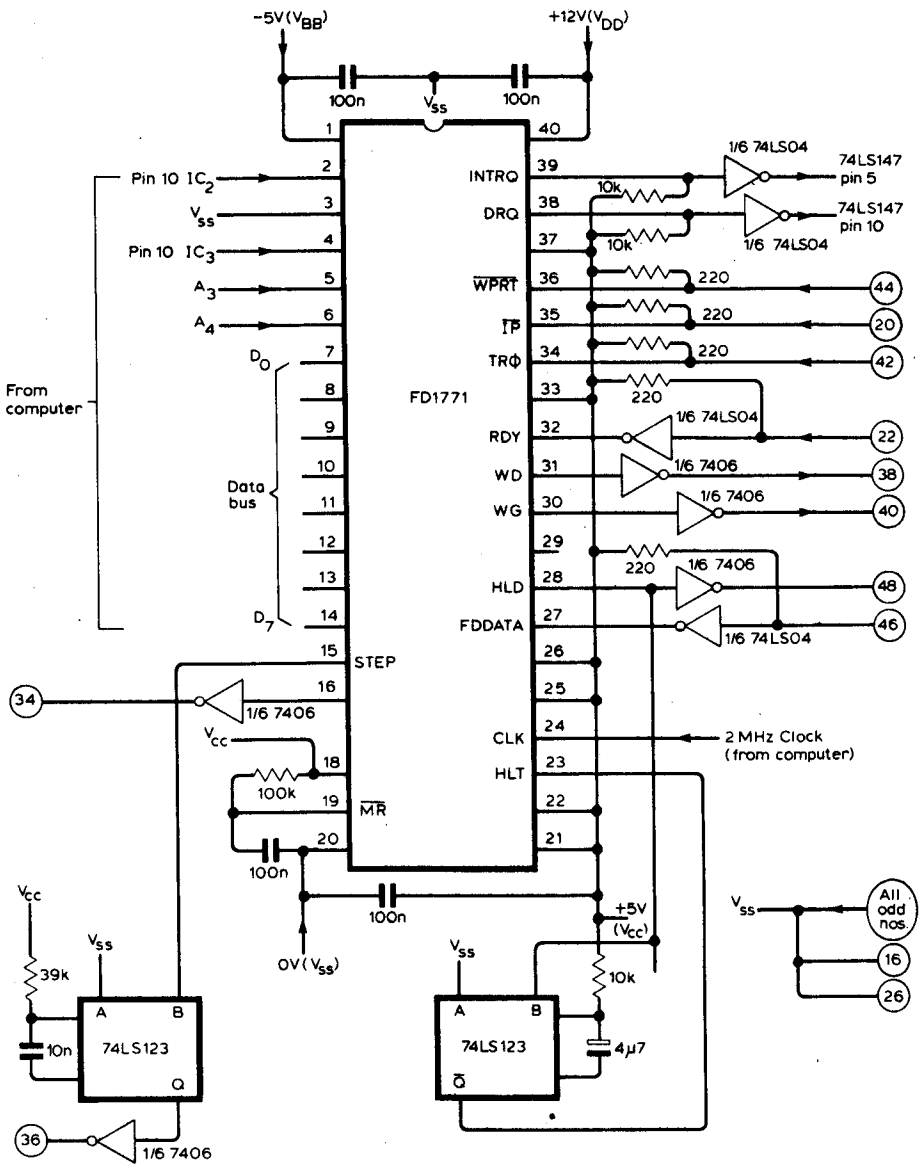


Fig. 2. Floppy-disc controller/formatter.

A3	A4	Register	Addressed as	Remarks
0	0	Status	IN A,(05)	Read on $\overline{\text{INTRQ}}$, checked for CRC error and record not found bits during Read, Write and Seek.
		Command	OUT (A0),A	Receives commands from the computer.
0	1	Track	IN A,(0D) OUT (A8),A	This register normally contains the current head position, 00 to 4C. It is reset to 00 on completion of the Restore command.
1	0	Sector	IN A,(15) OUT (B0),A	This register holds the desired sector number for use during Read and Write operations
1	1	Data	IN A,(1D) OUT (B8),A	Used during Read and Write operations as the source and destination of data bytes. During Seeks, it holds the desired track number, towards which the head and the track-register step.

Table 2. Register structure of the floppy-disc controller. Note that these details refer to the controller in this interface. More details are given in the data sheet[3].

and a head loading signal. In the opposite direction, the drive provides data in the form of interleaved clock and data pulses, the index pulse and signals to indicate that the head is over track zero, the disc drive is ready for use, and whether the disc is write-protected. Some drives provide a head-loaded signal, or can load the head onto the disc in the 10ms delay provided internally by the controller. As this drive does not, a monostable is used to provide a delay signal which is triggered by the outgoing headload line. To improve noise immunity when transferring signals to and from the electromechanical drive, the lines at both ends are pulled high via low value resistors, and high-current sinking buffers are used to drive signals to and fro.

Interrupt controller

Interrupts allow external hardware to divert the microprocessor temporarily from its stream of instructions, and accept instructions from, or more usually, to call a block of instructions which deal with the hardware's needs. In the computer, the $\overline{\text{INT}}$ line to the Z80 is driven by the MM57109 which only requires the Z80 to read (for Mk I and II systems) or transfer (for Mk III), data between the two. This is a fairly simple procedure and the interrupt mode is chosen, which causes a call to the address 0038 where, in the earlier two systems, there is a short routine to read the data. In the Mk III, there is a re-enable of the interrupt and return so that the interrupt line makes the Z80 pause until the MM57109 is ready for a data transfer. With the disc, faster and more complex responses are required because either of the two interrupting lines can become active separately or simultaneously and, depending upon the operation in hand, different responses may be required. To provide the extra flexibility without sacrificing speed, Mode 2 interrupts must be possible, and this is carried out by the interrupt controller in Fig. 3.

In Mode 2, after the Z80 has completed its existing operation, it responds to the interrupt by asking the interrupting device to supply an 8-bit byte onto the data lines which it uses, in conjunction with the previously loaded I register, to form the address of the first of two consecutive memory locations where it will find the starting address of a subroutine to be executed. With these two locations in the r/w.m., the disc operating system can make alterations to their contents and so alter the Z80's response to, in particular, a $\overline{\text{DRQ}}$ interrupt to cope with read, write and verify operations which are at the heart of a disc system.

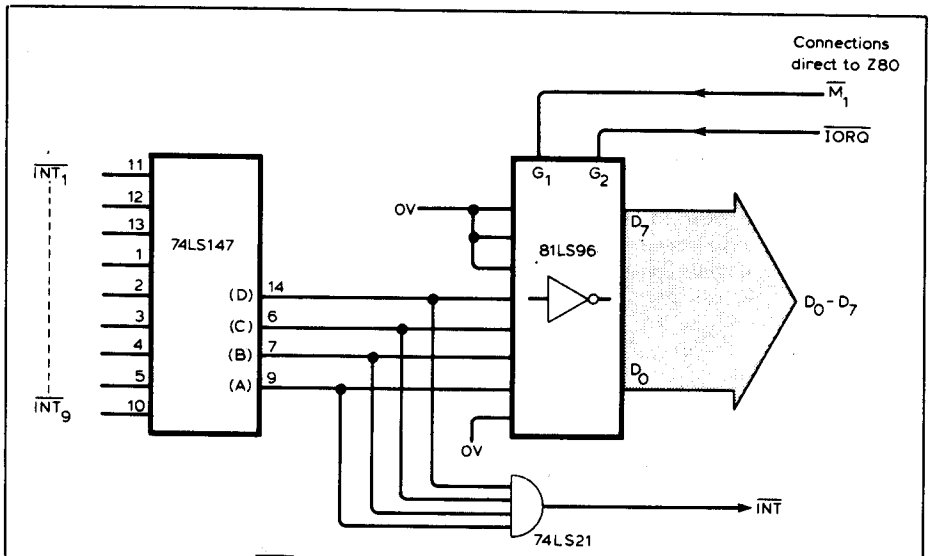
The circuit uses a 74LS147 priority encoder to generate a 4-bit code derived from the highest priority active input line. This code forms bits 1 to 4 of the byte which, when both $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ are active low, (a combination which only occurs during the Z80 interrupt acknowledge sequence) is gated onto the data lines. Direct connection to the Z80 is necessary because, during the interrupt response, both $\overline{\text{RD}}$ and $\overline{\text{WR}}$ are inactive and the bus transceiver isolates the Z80 from the main data bus.

The connection from the MM57109 to the $\overline{\text{INT}}$ pin of the Z80 must be broken and re-routed through one of the unused inputs to the interrupt controller. As the 57109 is now driving a low-power Schottky device, the pull-down resistor on this line must be changed to 10kΩ connected to the −5V supply.

Part 2 will describe the controller circuit and software.

References

1. Users' club, contact Mr P. L. Probetts, 50 Cromwell Road, Wimbledon, London SW19 8LZ.
2. Mk III monitor, contact the author at 5 The Close, Radlett, Hertfordshire WD7 8HA (Radlett 5723).
3. Data sheet, Western Digital FD 1771, Mar 77. □



$\overline{\text{INT}}$ line active	Priority	Byte fed into the Z80 for a type 2 interrupt
1	lowest	E3
2		E5
3		E7
4		E9
5		EB
6		ED
7		EF
8		F1
9	highest	F3

Fig. 3. Interrupt controller.