Floppy disc system for the scientific computer — 2

Interfacing a disc drive to the controller

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

This interface has been designed to operate with the Data Recording Equipment model 7100 8in disc drive, but should be easily adapted to suit others. The main advantage of an 8in drive over a 5¼in system is its greater storage capacity, 77 tracks of 3¼Kbytes each using the IBM format described in part one, compared with 35 tracks of 2¼Kbytes each. The disadvantage is greater cost. This concluding article describes how the drive is matched to the floppy-disc controller, and illustrates the salient points to check when considering other drives.

Whichever drive is used, the length of the cable, flat or twisted pairs, between the drive and the interface must be kept as short as is reasonably possible, and separate from the power cables. Each power cable should have its own return and there must be a good connection between the frame of the drive and the case of the computer.

When considering the signals to and from the drive, their polarity and timing must be examined. Most drives use the active-low principle for their inputs and outputs, i.e. a true state is logical zero (represented by $0 \leqslant V \leqslant 0.4V$) and a false input is logical 1 (represented by $2.5V < V < 5.25V$). Open collector drivers are generally used for outputs, and low value pull-up resistors on the inputs provide a full 5V swing and keep the line impedance down, both of which improve noise immunity. One implication of this arrangement is that, to pull a line to zero, the driving device will need to sink the current supplied by the receiving gate and by the pull-up resistor, typically 40mA. For a logical 1, no current is required from the driving device. The controller i.c. signals are mostly active-high, so inverters are used as receivers on all inputs except $\overline{IP}$ (index pulse) and $\overline{WPRT}$ (write protect), and 220Ω pull-up resistors are used with high current-sinking, inverting, open-collector drivers on the five active outputs. If a drive with some active-high inputs is used, the equivalent non-inverting buffers, 7407, or pairs of 7406 in series must be used. Note that ordinary t.t.l. is used for driving the interface cable because the L and LS series do not have the required current-sinking capacity. Table 3 gives some timing information for the 5¼in drive, the 7100 and the WD1771 controller.

When used with an 8in unit, the WD1771 must be clocked at 2MHz, whilst with a 5¼in disc 1MHz is used. This is necessary to meet the standard data rates used for the two sizes, and results in the doubling of all pulse timings for the i.c. when used with the smaller disc. There are other timing requirements connected with the application of power and selection of the drive, but these can be allowed for in the programming of the computer.

Stepping time

Most drives offer the option of keeping the head permanently loaded against the disc. This speeds operations by eliminating head-loading delay, but does increase wear on the head and disc. In this operating system, the head is only loaded when necessary which is usually after it has stepped to the track required. Settling time is irrelevant if it is less than the head loading time. The interval between stepping pulses is programmable by the bottom two bits of the Step instruction byte as described on page four of the data sheet. With the 7100 drive the fastest (6ms) rate can be used, whereas with the 5¼in unit, the drive can only just keep up with the slowest stepping rate. Stepping rate is probably the most critical timing factor because if a drive cannot step as fast as the controller's slowest rate, the two are virtually incompatible.

Stepping pulse

Virtually all drives can step on the pulse provided by the 1771. However, one exception found by the author is an obsolete version of the 7100. As this unit may still be available, the interface has been designed to operate with it and the current version.

The obsolete 7100 is most easily recognised by the absence of three d.i.l. sockets and header plugs next to the edge connector on the p.c.b., and the presence of two power resistors and three power transistors instead of one resistor and four transistors near the opposite corner to the edge connector. To allow compatibility, a monostable stretches the stepping pulse to 10μs.

Head-loading time

Ten milliseconds after the HLD (Head Load) output of the controller becomes active (20ms for the 5¼in disc), the HLT (Head loaded test) input is sampled and when it is low the controller proceeds. If the combined loading and settling time for the head is less than 10ms, this input can be wired low. If the disc-drive electronics provide a head-loaded and ready signal, this can be connected to HLT. If neither is true, as it is for both of these drives, HLD should trigger a monostable to produce the necessary delay before HLT becomes low. Because most drives will need this monostable if they are to be used with the head normally un-loaded, it is important to establish the total delay before the head is ready for use, i.e. the loading and settling time. Note that one value cannot be inferred from the other by comparing the stepping rate figures for the two drives.

Drive options

Most drives offer wiring options, and in this system one for direct control of head loading by the controller is used. To select this option on the current model, remove the link joining pins 13 and 14 on plug PP2 (the middle one of three referred to earlier)

Table 3. Timing information for disc drivers and the controller.

	5¼in drive	WD1771 @ 1MHz	DRE 7100 8in drive	WD1771 @ 2MHz
Track to track stepping + settling times	40 + 10ms	programmable to 12, 20 or 40ms	4 + 14ms	programmable to 6, 10 or 20ms
Stepping pulse width	1μs min.	8μs	600ns (10μs on older units min.)	4μs
Head load and settling time	75ms	HLT sampled after 20ms, therefore monostable is required	30ms	HLT sampled after 10ms, therefore monostable is required

and join pins 3 and 14 together. On the obsolete version, remove the short wire link joining the points marked HL and SI and connect a wire from HL to the pad at the end of the p.c.b. track coming from the edge connector tab numbered 18.

This change allows the controller to drive the head-load circuit through the previously unused pin 18 on the edge connector. The current and obsolete units should now be interchangeable.

System software

The software in table 4 is not a full disc operating system, but it illustrates the basic functions required to position the head, read and write records of any length from 128bytes to 256Kbytes with error checking, and to re-format corrupted tracks. With the drive and interface connected to the computer, move the head towards the centre of the drive by turning the stepping motor by hand. Apply mains to the motor and then switch on the computer. Put a disc into the unit and close the door. If the system is working, the head should quickly step to the outermost track 00 because the charging delay, caused by the RC network on pin 19 of the controller i.c., holds that pin (master reset) momentarily low in a similar way to the circuit used on the Z80. One of the actions which takes place during the resetting sequence is a Restore command, which moves the head out in this way. If it doesn't, check that the wiring is correct. If it steps out but does not stop, check that the track 00 line from the drive to the controller functions. With the software loaded, RUN 1D00 and then READ space. In response to the prompts DESTINATION: TRACK: SECTOR: NUMBER OF SECTORS: type 8200 40 space 0 space 8 space respectively. The head should move in to track 40 and load 8 sectors (1Kbyte) of data from the disc, starting with sector 0 to computer locations 8200 to 85FF, i.e. onto the v.d.u. With the IBM formatted disc, these should appear as percentage or proportional symbols.

At the end of the read, which should take less than one second, the head should release from the disc and READY occur. If a reading error occurs, the computer will attempt to re-read the particular sector up to twenty times. A corruption should be evident by rubbish appearing on the v.d.u., the controller recognises it by computing the CRC from a permutation of the data from the sector and comparing it with the pre-recorded CRC. Each sector takes up two lines on the v.d.u. If the corruption begins in a line and keeps changing, the data is corrupted. If the reading process seems to stop at the end of a line, the controller is having trouble recognising the Ident Field for the track or sector and, therefore, the track needs to be re-formatted (described later). With an undamaged disc most reads are successful first time, but if the operation fails for the 20 times that it is attempted, the message ERROR AT TRACK XXX SECTOR XXX appears. To force an error into the system and observe this feature, try

1D00	ED	5E	3E	1D	ED	47	DD	21	E4	1D	DB	05	LE	1D	3E	FF
1D10	D3	A0	FB	7E	7E	08	7E	FE	20	20	FB	08	CD	CE	03	FE
1D20	12	20	21	CD	E1	02	04	05	13	14	09	0E	01	14	09	0F
1D30	0E	3A	20	1D	CD	DE	03	CD	CE	03	CD	FD	1D	3E	E7	CD
1D40	57	1E	18	F9	FE	17	20	33	CD	E1	02	13	0F	15	12	03
1D50	05	3A	20	1D	CD	DE	03	CD	CE	03	CD	FD	1D	3E	71	32
1D60	F3	1D	3E	57	D3	A0	FB	7E	3E	F5	C1	C1	CD	57	1E	18
1D70	EC	1A	2F	D3	B8	13	FB	F9	76	18	ED	FE	0E	20	64	CD
1D80	CE	1E	79	32	A3	1D	21	00	C0	0E	28	3E	FF	23	10	FB
1D90	0E	0E	3E	00	23	10	FB	3E	FC	23	0E	1A	3E	FF	23	10
1DA0	FB	11	01	38	0E	1A	0E	0E	3E	00	23	10	FB	3E	FE	23
1DB0	72	23	3E	00	23	73	23	36	00	23	3E	F7	23	0E	11	3E
1DC0	00	23	10	FB	3E	FB	23	0E	80	3E	E5	23	10	FE	3E	F7
1DD0	23	0E	1B	3E	FF	23	10	FB	1C	0D	20	CA	0E	00	3E	FF
1DE0	23	10	FB	C7	FF	FF	FF	DB	1D	2F	12	13	FB	F9	7E	18
1DF0	72	F9	1D	E7	1D	DB	1D	18	F3	DB	05	2F	C9	CD	E1	02
1E00	14	12	01	03	0B	3A	20	1D	CD	CE	1E	DD	71	00	CD	CE
1E10	03	CD	61	02	13	05	03	14	0F	12	3A	20	1D	CD	CE	1E
1E20	DD	71	01	CD	CE	03	CD	E1	02	0E	15	0D	02	05	12	20
1E30	0F	0E	20	13	05	03	14	0F	12	13	3A	20	1D	CD	CE	1E
1E40	DD	71	02	CD	B9	1E	3E	EB	D3	A0	FB	7E	E6	18	20	F6
1E50	E5	21	00	00	39	D1	C9	32	F3	1D	0E	14	3E	77	D5	D3
1E60	A0	FB	7E	E6	18	28	34	D1	3E	73	10	F2	11	80	82	CD
1E70	E1	02	05	12	12	0F	12	20	01	14	20	14	12	01	03	0B
1E80	20	1D	DD	7E	00	CD	DA	1E	CD	E1	02	20	13	05	03	14
1E90	0F	12	20	1D	DD	7E	01	CD	DA	1E	C7	C1	DD	35	02	CA
1EA0	00	00	DD	34	01	DD	7E	01	FE	1E	20	0D	DD	3E	01	01
1EB0	DD	34	00	3E	A3	D3	A0	FB	7E	DD	7E	00	2F	D3	B8	DD
1EC0	7E	01	2F	D3	B0	C9	7E	E6	0F	4F	7E	FE	20	C8	E6	0F
1ED0	47	79	07	4F	07	07	81	80	18	EF	C5	CD	EA	1E	CE	30
1EE0	EB	70	23	71	23	77	23	EB	C1	C9	0E	30	48	FE	04	38
1EF0	05	04	D6	04	18	F7	FE	0A	38	05	0C	D6	0A	18	F7	C9

Table 4. System software.

Table 5. Software subroutines.

1DF9 Used in READ and WRITE to convert the typed in track number, sector number and number of sectors from decimal to binary, and then dump them into locations 1DE4 to 6 respectively using the index register. These bytes are then sent to the controller, which is then told to step into this track and, by reading an indent field, verify that the head is over the correct track. Also, by reading the CRC, verifies that the track number has been correctly read and does not match the track register's contents by chance. The data destination/source address is transferred from HL to DE and, by clearing HL and adding SP to it, the contents of the stack pointer register are loaded into HL.

1E57 Used in READ and WRITE. On entering this routine, the A register holds a byte which is dumped at 1DF3 to be used by a $\overline{\text{DRQ}}$ interrupt as the lower part of the interrupt routine address (1DE7 for READ, 1D71 for WRITE, 1DF5 for VERIFY WRITE). 20_{10} is loaded into B and DE, which holds the destination/source address, is saved on the computer stack in case a re-read or -write is necessary. The controller is then instructed to read a sector of data to that and succeeding locations. After the read, at 1E63, a check is made for the correct CRC and for the existence of the track and sector. If no faults have occurred, execution jumps to 1E9B where the saved DE is discarded from the stack into BC and, using indexed operations, the number of sectors byte is decremented. If this operation sets the byte to zero, an exit is made because the READ is complete. Otherwise, the sector and, if necessary, the track number are updated for the next sector to be read, the information is sent to the controller registers and another sector is read. If the operation to read the sector fails, the starting address of the data is popped back off the stack and B is decremented. If this does not reduce it to zero, a re-read is attempted and after 20 attempts execution passes to 1E6C et al and the error message appears.

1EC6 Loads decimal data from the keyboard and converts it to binary in register C.

1EDA Displays the contents of A, converted to decimal, on the v.d.u.

1DE7 The READ interrupt routine, called by a $\overline{\text{DRQ}}$. This routine transfers the byte in the controller's data register to the Z80, inverts it to its true form, stores it in the location pointed to by DE, and increments DE ready for the next byte. The interrupt system in the Z80 is automatically disabled when an interrupt is accepted so that the Z80 can service the interrupting device without interference from the interrupting device itself. Standard service routines usually finish with a re-enabling of the interrupt system and then a return. To ensure that the return will occur, the Z80 does not re-enable the interrupt until it has executed the instruction after the enabling instruction F3. This service routine does not have a return, but it uses this one protected instruction after the F3, F9, to load HL into the stack pointer, SP, register. SP is increased by two when the subroutine 1DF9, which loaded HL with the SP,

ended and the return address was popped off the stack. When the DRQ interrupt was accepted, the current PC (program counter) contents were pushed onto the stack and SP decreased by two, as is normal at the calling of any subroutine. Therefore, for the first DRQ, HL and SP are the same and F9 has no effect. The next byte in the subroutine is a HALT, at which the Z80 stops and waits for the second DRQ which, when it arrives, jumps the execution back to the start of the subroutine and pushes another return address onto the stack. When the data byte is read and the F9 is executed, SP, which decremented when the second interrupt was accepted, is pulled back to where it was before the interrupt occurred. Therefore this, and all future DRQs are demoted from calls to being, effectively, simple jumps to 1DE7. Whilst each return address is written on top of the last as the DRQs progress, the first call from the main program remains unaltered one position further up the stack. When all 128 DRQs have passed and the SP has been pulled back again, the INTRQ interrupt occurs and this, having a conventional return at its end, returns execution to the first popped address, i.e. where the original "read a sector" command was given. This forms a neat method of writing the main program because it makes the controller appear as part of the main processor and, more important, it saves time. There are only 32μs during which data can be transferred from the controller to the memory, and the Z80 made ready for the next interrupt. If the sequence servicing the controller takes longer than this, data will be lost and the controller will halt the reading sequence. A conventional return takes 5½μs and the jump from this returning point to the "wait for a DRQ" point requires a further 6μs. The single F9 instruction only takes 3μs, which achieves the same purpose, but just within the 32μs limit.

1D71 The WRITE interrupt routine called by DRQ. This is similar to the previous routine in that the progress of the stack pointer is arrested by repetitive loading from HL. This routine differs because the first two DRQ-pushed addresses are saved, 1D68 and 1D79 respectively. When the 128 DRQs have occurred, INTRQ causes a jump to the status reading routine after which the return occurs to 1D79 at which a jump pushes execution on the 1D68. Here the other DRQ is popped off the stack and a new vector byte, F5, is placed into the A register ready for the third type of DRQ.

1DF5 When checking a written sector, 1E57 is used as the reading subroutine. Because we are interested in the CRC and not the data on the disc, 1DF5 acts like 1DE7 when handling DRQs, except that it makes no attempt to store the unwanted data and just waits for the INTRQ. When this arrives, 1DF5 returns to the point in the main program where the CRC can be checked to see if the track just written has verified itself.

reading any sector track 77, which does not exist! Note that spaces are required after decimal information – the track, sector and number of sectors, but not after the hexadecimal destination address.

If the Read has worked type RUN 1D00, which should cause the unloaded head to return to track 00. Next type WRITE space, and in response to SOURCE type 0000, for TRACK: 40 space, for SECTOR: 0 space, and for NUMBER OF SECTORS 32 space. The head should move in and write to track 40, step to track 41 and continue writing, so that the first 26 sectors fill track 40 and the final 6 fill sectors 0 to 5 track 41. The write operation is slower because after each sector is written it is read back and checked for errors. As before, up to 20 attempts are made before the operation terminates and the ERROR message occurs. Nevertheless, it should only take a few seconds to record the entire 4K monitor.

Explanations of the software subroutines are given in Table 5. To follow the main program 1D00 to 1DFC, a disassembler such as the one given in a recent computer newsletter is useful. The interrupt mode 2 is set and the I register, which is used (as described in part one) to form the top half of the interrupt addresses, is set at 1D and the IX index register is set at 1DE4. The index register is useful as a pointer to an area of memory because any indexed Z80 instructions, i.e. instructions prefixed by DD, will use a byte in the instruction to say which byte relative to 1DE4 is to be used in the instruction. In this case byte number 00 (i.e. 1DE4 itself) stores the required track number, byte number 01 (1DE5) stores the required sector number and byte 02 (1DE6) stores the number of sectors. The status and data registers are read next (not for their contents) to reset the INTRQ and DRQ interrupt lines if they are active due to the power-on sequence. Note that this unit is designed to operate with the mark III operating system, (see the scientific computer newsletter) which contains these same four bytes, DB,05, DB and 1D. They are executed in the high level so that the MM57109 interrupts are not upset by the disc controlled conditions. 1D0E-14 illustrates the way instructions are sent to the controller. The instruction byte loads into A, in this case a Restore instruction, it is sent to the command register, the interrupt is enabled and the Z80 is halted to wait for the interrupt. When it arrives, in this case a INTRQ, the computer reads the interrupt controller byte Fl, adds it to the 1D previously stored in the I register and then reads in the byte at 1DF1 and 1DF2 as the address of the INTRQ subroutine, which is 1DF9. Execution passes to this address when the status register is read and inverted back to a true state.

Re-formatting a disc

As well as reading and writing individual sectors, the controller can read and write whole tracks using the index pulse as the start and finish of the operation. As described in part one, even before use the disc is fully recorded with ident fields and dummy data. If the ident fields become magnetically corrupted, the entire track has to be re-recorded, or re-formatted, before it can be used in the sector mode again. To do this, a block of length 5¼K bytes must be set up in r.a.m. and then recorded en bloc. Assembley of this block requires extra r.a.m. over the basic computer's memory and, given this, the operating system can synthesise the track format. I used a 32Kbyte expansion (referred to in the computer newsletter) and assembled this block at C000. To accommodate other r.a.m. locations, the byte at 1D88 must be altered. After RUN 1D00, type FORMAT space and then the track number, in decimal, to be altered. As

continued on page 57

Table 6. Floppy-disc controller commands used. The asterisked addresses are where modifications to the software are made when a track is re-formatted.

Address	Byte	Command	Function
1D0F	FF	00	RESTORE the head to track 00 and clear the track register.
1D63	57	A8	Assuming the head is to be loaded against the disc, WRITE a single record of IBM format to the track and sector specified by the respective registers, using FB as the data mark.
1E47	EB	14	SEEK the track specified by the data register by stepping the difference between it and the contents of the track register. Then, by reading an Ident Field from the track, verify that it is the correct one.
*1E5D	77	88	Assuming that the head is loaded against the disc, READ a single record of IBM format from the track and sector specified by the respective registers.
1E69	73	8C	As above, but it begins by issuing the HLD, head load, signal and waiting for the HLT signal to become active before proceeding.
1EB4	A3	5C	Load the head against the disc and then STEP IN by one track, updating the track register. Perform a verify of the track as described above.
*1D63	0B	F4	WRITE TRACK. Starting at the index pulse, data is written continuously up to the next index pulse.
*1E47	EF	10	On a badly corrupted track, it is not possible to verify the head position after a SEEK, so this version of the command omits it.

Floppy disc system

continued from page 76

explained in Table 6, two controller commands have to be altered, 1D63 to write the complete track, and 1E47 to delete the attempt to verify the head position after the seek operation, as presumably the track is being re-formatted because it is impossible to verify on that track.

Re-formatting is accomplished by proceeding as with a standard Write operation, giving the start of the block as a source and dummy data, say 1, 1 and 1 for the track, sector and number of sectors. The head should move in, load itself and write the track. After this has occured and the head has released, the computer should be manually reset using the Reset key or, preferably, Control Z. □

Table 7. Sequence for a DRQ interrupt. Note that interrupts can be accepted during LD, SP, HL, in which case the HALT will not be executed, but instructions up to and including LD, SP, HL are always executed unless $\overline{\text{NMI}}$ occurs.

(Mode 2 Interrupt)	9½ µs	
IN A,1D	5½ µs	Data read in
CPL	2 µs	Data inverted to true
LD (DE),A	3½µs	Stored in memory
INC DE	3 µs	Move to next location
EI	2 µs	Enable interrupt
LD SP,HL	3 µs	Pull back SP
HALT	2 µs	Halt for interrupt
	30½ µs	