

The Specifications of Orders 114–119

The problem

The programming manual does not include descriptions of the hardware instructions for controlling the magnetic tapes, beyond the statement that orders 114–119 are used. [It also contains the statement that the 98 order is so used. This is correct in the pedantic sense that this order occurs in the software, but it is the equivalent of the 99 order operating instead on the reserved store, and has nothing to do with controlling the tapes.]

I have not been able to find any other full documentation that describes these orders although there scraps of information in the various descriptions of the machine. So I have been forced to work out what the hardware instructions do by examining the software which is described in “User’s guide to the EDSAC magnetic tape system” by David Barron.

Because EDSAC2 used microcode it was possible for the hardware instructions to be quite complicated; they also refer to specific hard-wired locations in the reserved store. Moreover since only specific bit patterns are used in the software it is impossible to know if particular bits could have had other functions. These might have been used, for example in special programs used for labelling a tape, but should be irrelevant to the programmer who would be accessing the tape system through the built-in routines.

There is evidence that the software has been modified at some stage. The code from 1963 to 1995 appears to be inaccessible unless by a hardware interrupt. This latter seems unlikely since the address 1963 is not simple in binary unlike the other addresses (for example 512 and 1024) which are entry points to code in the reserved store. Location 1964 contains the only 114 order which, according to the programming manual, is used for controlling the magnetic tapes. This seems implausible; the code looks like an obsolete trace routine and the 114 order was possibly involved in an interrupt.

The following sections describe my reasoning about the hardware operations and some minor unsolved issues.

Controlling the tape motors: 116 order

The order 116 f m controls the motors moving the tape through the bits of the address m . All instances of the order have $a(21)$ added to the address and this contains the tape unit number. A list of the uses of reserved store locations by magnetic tape is given in the appendix. The only bits of m that are ever set to 1 are the 1024, 128 and 32 bits. It is clear from the contexts

that the 1024 bit starts the tape and the 128 bit determines the direction, 0=forwards, 1=backwards. If all bits except the last 3 are zero the tape is stopped. The function of the 32 bit is uncertain, the context in which it occurs is:

1911	4f21	add tape number to next address
1912	116f1056	address = 1024 + 32
1913	10f126	load slave data to accumulator
1914	115f0	?
1915	7f150	???
1916	4f14	add address of source to address of write
1917	117f9	write

The 115 order is concerned with the operation of slave tapes and the details are too uncertain for this facility to be implemented. The 7f150 instruction right shifts the accumulator by 150 places, copying the sign bit. This makes no sense and I conjecture that it is a time delay. The reading and writing of the tape is done with a single head and so the 32 bit in the 116 order could be simply switching to write mode. Alternatively it could be something to do with slave tapes. In either case the bit can be ignored.

Reading blockmarks: 119 order

The tape routines contain only one such order 119f9 at 1752 which leaves the next block label in register 8. I conjecture that the address part of the order specifies the location where the block label is placed but the meaning of the least significant bit is obscure; possibly it determines which store as in the 118 order.

Since the only ways out of the loop between 1748 and 1759 are a report at 1757 if ten successive block labels are found to be faulty or the end at 1760 it is possible to deduce that M is cleared if a block label has been found and that the most significant bit of register 8 must be a 1. Register 16 is used for the count of tries.

The pattern of the block label can be deduced from the specification of the 59f19 subroutine which loads to content of register 8 into the accumulator before exiting. The pattern is

0	x	x	x		x	x	x	x	x	x	x	x	x		0	x	x	x		x	x	x	x	x	x	x	x	x	
tape number										block number										section number														block length						

In the emulator the section number is always 0. The unpacking of the block label is done at 1871 – 1876. Digit 20 is tested at 1878; it is clearly expected to be zero, presumably indicating that a valid block length follows. A 1 here will lead to an error report at 1776 if an attempt is made to write.

Automatic positioning

After a block label has been read the sign bit of register 2 is set to 1 (at 1870). The microcode contains a flip-flop that detects the passage of a blockmark and this is tested after every order. When it is set the count of blockmarks, which is held in $a(2)$ is increased one and the flip-flop reset. If $a(2) < 2047$ the tape is allowed to continue in motion otherwise the sign bit of register 2 is set to zero and the tape is stopped.

Implementing this in the emulator is fairly straightforward, the only complication is that the reserved store subroutines assume that the next blockmark will not be read until many machine instructions have been obeyed. If the next blockmark is produced too soon the computation required before it can be processed will not have been done. Conveniently the emulator checks for updates to the graphical output every 100 instructions and the flip-flop and blockmark count can be examined at this point.

Writing tape 117 order

There is only one 117 order in the subroutines, at 1917. One can deduce most of its behaviour by looking at the store dump/undump code (59f38) in 1822–1845. This copies part of the reserved store and all of the free store (except register 2046) onto tape. There has to be some way to tell the 117 order which store to use and this appears to be done by the least significant digit of the address; even addresses write from the free store, odd addresses imply the reserved store.

For a simple write (59f18) the starting address in the free store is specified in s and the number of half registers to transfer in t which is immediately stored in $a(5)$ at 1692. The value of s on entry to any reserved store subroutine is automatically stored in $a(0)$. Before use the least significant bit is removed at 1728–1730.

For a store backup the contents of $a(0)$ are overwritten by 19, (an odd number) at 1825 but the last bit of this is not removed. The relevant part of the reserved store is written to tape and then $a(0)$ is set to zero (an even number) at 1831 and the free store copied onto tape. [Note that the accumulator is cleared at 1807 and that checked writing is selected at 2032.]

The number of half registers to copy is set to 114 in $a(5)$ (at 1839) for the reserved store and later to 2046 (at 1833) for the free store. It appears that it is not possible to read from tape directly into the reserved store and during the undump process the relevant data is read into the free store and then copied to the reserved store by the instructions in 1841 – 1844.

In the preparation for any tape operation (1740 – 1742) $a(14)$ is set to set to $a(0) - 9$. The address of the 117 order is $a(14) + 9$ and so the obeyed address is equal to $a(0)$ or $a(0) + 1$. I do not know why an offset of 9 is used, possibly to keep the offset the same as that used in the 118 (read) order and save space.

The number of half registers to transfer is held in $a(5)$ but this is used to set $a(4)$, (1743 and 1745). If the transfer exceeds 100 half registers it is automatically done in successive blocks; the calculation is done at 1771 – 1777 and the code here establishes that the 117 order requires that the number of half registers to be written be set in modifier t and it transfers $t + 2$ half registers.

The calculations required for writing multiple blocks are done at 1939 – 1957.

Checked Writing and the Checksum

There are two types of checksum; for each block that is written to tape a checksum is automatically computed and written on the tape at the end of the block. When a write operation is spread over several blocks a cumulative checksum is computed in software and this is placed in the accumulator at the exit of the tape subroutine so that the user can save it and compare it when the tape is read in the future.

The microcode for the 117 order computes a 40 bit checksum which is the sum modulo $2^{40} - 1$ of the data transferred. This is clearly left in the accumulator after the 117 order and is added to the cumulative checksum held in register 12.

It is possible check a record immediately after it is written onto the tape. The route through the magnetic tape code is determined by an integer stored in $a(6)$ and often in modifier s . A table of the meanings of these integers is included in the appendix. The point at which the check is started is at 1956; $s = 5$ means that a record has been written but needs checking. It is clear that this is reached only when the data left to transfer is less than a whole block and then a complete new tape operation is started to read the record. The cumulative checksum is erased and assuming the reading is successful the checksum returned is the one computed during the read operation. If the transfer is faulty the write is repeated ($s = 2$) and in this case the calculation of the checksum is skipped by the test at 1920. This supports the interpretation that the checksum returned is derived from what is written on the tape if possible.

The purpose of the 24f8 order at 1918 following the write order is a mystery; it loads register K from register 8 which (unless it is changed) contains the block label. But register K is never used during the rest of the write operation and will be changed when the next block is written. K is used for calculating

the checksum during reading the tape (see 118 order below) and conceivably it is used in checked writing, however the checking is only done after all the blocks have been written and so some very complicated microcode would be involved. My guess is that the hardware has been modified and the use of K here is obsolete. Because it was difficult to rewire the reserved store an obsolete but harmless instruction could remain.

Reading tape: 118 order

The route through the reserved store is fixed by $a(6)$ which is usually copied in to s . Normal read is $s = 4$ and read checked writing is $s = 5$. The only 118 order occurs at 1784, with an address depending s . For a normal read operation the obeyed address is $a(14) + 9$ which equals $a(0)$, the location in the free store to place the data. Note that this is necessarily even. The addition of $a(14)$ to the address is suppressed if one is checking a previously written record and the obeyed address of the 118 order is 9, which is odd. Obviously if one is checking what has just been written one does not want to put the result into the free store and it is clear that the least significant bit of the address determines whether this happens. The significance of the 8 bit is unclear.

The number of half registers to read is equal to $t + 2$ as in the write order.

A checksum for the data read from the tape is formed and the difference between this and the checksum written on the tape during writing is placed in the accumulator which is tested immediately following the 118 order.

If M is not zero the read operation is repeated up to ten times before an error is reported. Note that the count for this is held in the more significant bits of register 4 which also contains the count of the number of half registers to transfer. This could have been to save space or even maybe leave a challenge for people trying to understand the code!

Since the cumulative checksum for the read operation has to be returned by the subroutine this must be placed somewhere. A successful read of a block jumps to 1924 and the use of the 37 order in 1925 proves that the block checksum is put into K . It is tempting to see some connection between the 8 bit in the address of the 117 and 118 orders and the unexplained loading of K from register 8 at 1918 but I can see no consistent story.

Slave tapes: 115 order

A tape is designated a slave tape by 59f39 obeys the code at 1762 – 1767 which places the slave parameters in location 126. The contents of 126 are

changed at 1931 – 1938 to take into account other tape movements. At 1770, 1903 and 1914 the order 115*f*0 occurs, preceded by copying register 126 into the accumulator. From the lack of any other code relevant to moving slave tapes I deduce that the control of slave tape is done entirely in microcode and the 115 order sets up the operation using data in the accumulator. This is obviously quite complicated and slave motion of tapes serves no purpose in an emulator where one is not restricted by the speed of a physical tape I have decided not to implement slave motion. The 115 order simply does nothing, but to avoid problems if a user creates a slave tape the sign bit of register 126 is set to 1, indicating that the slave movement is finished..

Appendix

Annotated Reserved Store

The file `annotated_reserved_store.txt` contains a partially annotated copy of the orders in the reserved store.

‘Route’ numbers used in the code

These are initially derived from the modifier bits of the pseudo-order following the call to the reserved store routine. They are modified by the code at 1698 – 1705 and stored in *a*(6) and have the following meanings:

- 0 write
- 1 checked write
- 2 first write failed on reading: re-write
- 3 position
- 4 read
- 5 data has been written, check it but don't store the result
- 6 checksum failed after second write, mark block as faulty

Use of the reserved store by magnetic tape

0	value of s on entry to 59 order
1	return address of 59 order
2	sign bit 1=tape moving, 0=tape stopped, $a(2)$ = count of blocks to move
3	?
4	more significant bits=count of tries at reading, $a(4)$ =number of half registers to transfer
5	number of half registers to transfer originally requested
6	'route' number for path through code, see above
7	?
8 + 9	block label
10 + 11	1 digit shifted right one place every block written or read
12 + 13	cumulative checksum
14	added to address of read and write orders
15	used to hold address for jumps between sections of code
16 + 17	more significant bits used for count of tries to get blockmark
17	requested block number
21	tape number
22 + 23	pattern of block errors
126	slave data
127	used to hold address for jumps between sections of code