Orton ZX79 computer manual

Index

Chapter 1 Overview..... 2
Chapter 2 Theory of operation 3
Chapter 3 Circuit diagram 11
Chapter 4 System timing and design verification 15
Chapter 5 Photo 23
Chapter 6 Software 24

Disclaimer

Chapter 1 Overview

The computer described in this document was inspired by the ZX80 home computer, which was released for the UK home market by Sinclair in 1980. That computer used 21 ICs (not including the 5V regulator). The design I present here uses half that number of ICs. Note that the computer to be described is NOT a ZX80 or ZX81 clone and will not run either of these machines' ROM code.

The design parameters were as follows:

1. The computer must have enough ROM to support a BASIC interpreter
2. The computer must have enough RAM to run simple user programs
3. The computer must have an alphanumeric keyboard
4. The computer must be able to generate a video text display
5. The computer must have a cassette interface for program storage and retrieval
6. The computer must have expansion capability

The resulting design has a 40 key keyboard and is able to generate a 24 line by 30 character display. It uses a Z80A microprocessor running at 3.25MHz. The cassette interface uses on/off keying (OOK) to save and load programs at 300 bits per second. It has 4k bytes of ROM BASIC and 2k bytes of RAM, which is shared between programs and screen.

As with the ZX80, this computer cannot maintain a video display and run user BASIC programs simultaneously. However, a carefully timed machine code program could maintain both a video display and a running program, and this was seen to great effect in the games that were designed for the ZX80.

The descriptions which follow require frequent reference to the computer's circuit diagram, which is supplied in Chapter 3. The computer has been given the tongue-in-cheek name 'ZX79'.

Chapter 2 Theory of operation

Memory map

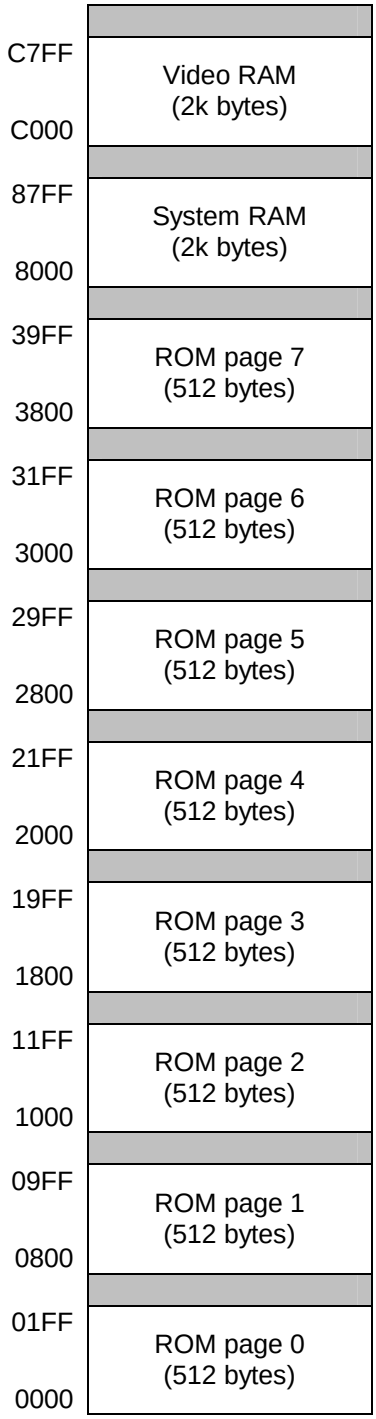
The computer has 64k bytes of addressable memory, selected through the Z80's 16 address lines. These are assigned as follows:

Z80 address line	IC5 (ROM) address line	IC1 (RAM) address line	Other effects
A0	A0*	A0	
A1	A1*	A1	
A2	A2*	A2	
A3	A3*	A3	
A4	A4*	A4	
A5	A5*	A5	
A6	A6	A6	
A7	A7	A7	
A8	A8	A8	
A9	-	A9	
A10	-	A10	LOW = assert sync (provided A14 high)
A11	A9	-	These address lines form font table row selects
A12	A10	-	
A13	A11	-	
A14	-	-	HIGH = enable IC3 for character latch ROM address * HIGH = enable video generation
A15	-	-	LOW = select ROM and enable IC6 for Z80 ROM address * HIGH = select RAM

* These ROM address lines are multiplexed between the Z80 and a character latch

This memory map has potential for contention: A15 low and A14 high will cause IC3 and IC6 to drive ROM address lines A0..A5 simultaneously. This condition must be prohibited to avoid stress to the parts involved. Simultaneous changes of A14 and A15 are also to be avoided where they might result in transient contention.

Note that the ROM's address lines are not contiguous with those of the Z80. A9 and A10 are skipped. This breaks the former's addressing into eight separate blocks of 512 bytes each. The conventional memory map seen by Z80 programs is therefore:



This is the default memory map, as seen from the point of view of the user. In fact, the RAM and ROM have multiple aliases in the memory map, some of which have special purpose for video generation (see later).

The ROM is actually an 8k byte device. Address line A12 of this part selects between executable code (addresses 0000 to 0FFF) and font tables for display generation (1000 to 1FFF). The font tables are not addressable by Z80 code and do not appear anywhere in the computer's memory map. In fact there are 8 copies of the font table in the upper half of the ROM. This is because ROM address lines A6..A8 are regarded as 'don't cares' for the purpose of font look-up.

Expansion

The above memory map shows main system RAM and video RAM having distinct addresses. In fact they are one and the same part on a basic machine. The expectation is that user programs will occupy the lower part of the available 2k bytes of RAM, and will be accessed from 8000 up. Accordingly, it is expected that the video display will occupy the upper portion of the RAM and be accessed through C400 up.

Expansion of the machine is possible by taking control over the address decoding for the on-board RAM, such that it is confined to addresses 8000 to 87FF, and C000 to FFFF. This leaves a 14k byte hole for expansion memory at 8800 to BFFF. When expanded, it is vital to ensure that the on-board RAM is still addressable at 8000 to 87FF, as essential video support code is located in here.

Address continuity

The usual practise in microprocessor system design is to decode an address, and then gate this with one of a number of strobes (e.g. /RD, /WR) When this is done the address only has to remain stable for the time that the strobe is active. However, in this computer's design the multiplexing of the lower six ROM address lines is controlled by raw address lines. This demands that the address lines are controlled, even during refresh cycles. This is accomplished using assignments to the Z80 I and R registers. Some address line discontinuities are permitted while others are not. All combinations, and their consequences, are summed up in this table:

Read, fetch or write address		Refresh address		Selection	Comment
A15	A14	A15	A14		
0	0	0	0	ROM	Permitted
0	0	0	1		Forbidden: contention
0	0	1	0		Permitted
0	0	1	1		Deprecated: transient contention possible
0	1	X	X		Forbidden: contention
1	0	0	0	System RAM	Permitted but A15 flaps
1	0	0	1		Forbidden: contention
1	0	1	0		Permitted
1	0	1	1		Permitted but A14 flaps
1	1	0	0	Video RAM	Deprecated: transient contention possible
1	1	0	1		Forbidden: contention
1	1	1	0		Permitted but A14 flaps
1	1	1	1		Permitted

There is another address continuity requirement which concerns A10. A10 is latched during read or fetch cycles for generation of the /SYNC signal. Being latched, it doesn't need the same level of continuity as A14 or A15 however, the output driver of this latch is under control of A14 and so is dependent upon this address line's continuity:

Latched A10	Read, fetch or write A14	Refresh A14	Comment
X	0	0	/SYNC high
1	X	X	/SYNC high
0	0	1	/SYNC flaps
0	1	0	/SYNC flaps
0	1	1	/SYNC low

Pixel generation is enabled when A14 and A10 are both high. Since A10 is one of the address lines that determines the byte to be read out of video RAM, the constraint A10=1 effectively confines video data to the top half of the 2k byte video RAM. The address continuity requirement here is for the purpose of preventing the ROM address changing throughout a fetch-plus-refresh event:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	m	m	m	1	n	n	n	n	n	k	k	k	k	k

Where:

mmm	Identifies a pixel row (0=topmost)
nnnnn	Identifies a character row (0=topmost)
kkkkk	Identifies a character within a row (0=leftmost)

This address, with the exception of A0..A5, must remain stable throughout the entire character fetch/font look-up process. Address lines A0..A5 are exempt because they are replaced with a character code by the ROM address multiplexer. Nonetheless, attention to the R register is still important because the R register increments on every instruction fetch, and this will eventually alter the state of the A6 refresh address line. In theory, continuity of address lines A9, A10 is not essential (the ROM is not addressed by them) but maintaining continuity of these costs nothing anyway.

Character rows must not cross a 64 byte boundary, as this would cause a change of A6 of the fetch address during the course of a character row scan. The pixel row address is contained in address lines A11..A13. These select an alias of the video memory, which allows the video RAM contents to be repeatedly scanned for pixel row generation, without it being necessary to duplicate character information.

Character set

It probably hasn't escaped the reader's notice that video generation relies on instruction fetches in order to read a series of characters from memory. In other

words, the Z80 actually executes the characters! Trouble is avoided by using character codes that correspond to innocuous 4 cycle Z80 instructions.

This scheme requires that the instructions needed to get into and out of video memory correspond to blank spaces, so as not to generate pixels as a result of these instructions. The character set, and corresponding Z80 opcodes, are as follows:

Character code (ROM address)	Z80 opcode	Z80 instruction	Character	Comment
00	40	LD B,B	A	
01	41	LD B,C	B	
02	42	LD B,D	C	
03	C3	JP nn		Used by supporting code
04	44	LD B,H	D	
05	45	LD B,L	E	
06	-	-		No 4 cycle instruction
07	47	LD B,A	F	
08	48	LD C,B	G	
09	49	LD C,C	H	
0A	4A	LD C,D	I	
0B	4B	LD C,E	J	
0C	4C	LD C,H	K	
0D	4D	LD C,L	L	
0E	-	-		No 4 cycle instruction
0F	4F	LD C,A	M	
10	50	LD D,B	N	
11	51	LD D,C	O	
12	D2	JP NC,nn		Used by supporting code
13	53	LD D,E	P	
14	54	LD D,H	Q	
15	55	LD D,L	R	
16	-	-		No 4 cycle instruction
17	57	LD D,A	S	
18	58	LD E,B	T	
19	59	LD E,C	U	
1A	5A	LD E,D	V	
1B	5B	LD E,E	W	
1C	5C	LD E,H	X	
1D	5D	LD E,L	Y	
1E	-	-		No 4 cycle instruction
1F	5F	LD E,A	Z	
20	60	LD H,B	0	
21	61	LD H,C	1	
22	62	LD H,D	2	
23	63	LD H,E	3	

Character code (ROM address)	Z80 opcode	Z80 instruction	Character	Comment
24	64	LD H,H	4	
25	65	LD H,L	5	
26	-	-		No 4 cycle instruction
27	67	LD H,A	6	
28	68	LD L,B	7	
29	E9	JP (IY)		Used by supporting code
2A	6A	LD L,D	8	
2B	6B	LD L,E	9	
2C	6C	LD L,H	"	
2D	6D	LD L,L	$	
2E	-	-		No 4 cycle instruction
2F	6F	LD L,A	:	
30	B0	OR B	(	
31	B1	OR C	)	
32	B2	OR D	-	
33	B3	OR E	+	
34	B4	OR H	*	
35	B5	OR L	/	
36	-	-		No 4 cycle instruction
37	B7	OR A	=	
38	78	LD A, B		Space character
39	79	LD A,C	<	
3A	7A	LD A, D	>	
3B	7B	LD A,E	;	
3C	7C	LD A,H	,	
3D	FD	JP (IY)		Used by supporting code
3E	-	-		No 4 cycle instruction
3F	7F	LD A,A	.	

As can be seen, there are some 56 suitable Z80 opcodes of 4 cycles duration, which are distinctive in that they do not disturb the sequence of instruction fetches required to display a line of text. Of these, 4 are lost to supporting code leaving 52 characters in total. Inevitably, there are side effects of executing characters. These amount to register manipulations, whose effects are mitigated by confining them to the Z80 alternate register set.

Keyboard scanning

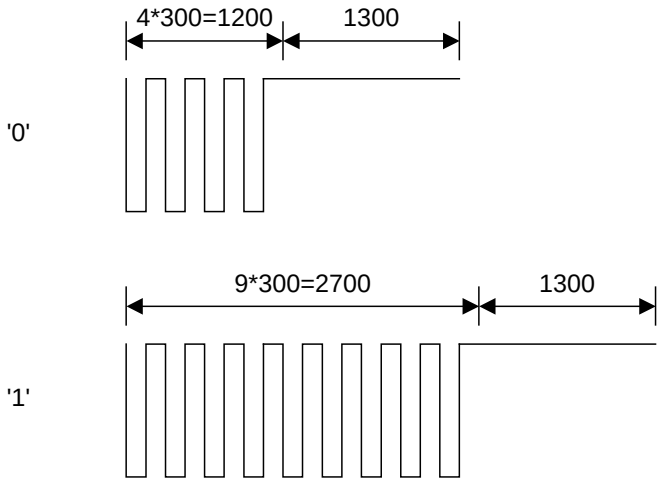
There are eight sense inputs to the computer: one cassette replay input ('CASS'); and seven keyboard column senses (KC1..KC7). Six keyboard row drives (KR1..KR6) are derived from the upper address bus (A8..A13). The keys are consequently on a 6 by 7 matrix, according to the following table (normal, unshifted key shown in each cell):

Row	Column						
	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7
2	Q	W	E	R	T	Y	U
3	A	S	D	F	G	H	J
4	shft	Z	X	C	V	B	N
5		del	ret	spc	L	M	K
6		P	0	O	9	I	8

The sole input to the computer is the /INT pin, which is driven by a selected column via a multiplexer (IC9). The state of the /INT pin is tested by momentarily enabling interrupts and then checking to see whether an interrupt was triggered as a result. Interrupts are enabled for a single NOP instruction, during which the Z80 samples the state of the /INT pin during the included refresh cycle. Consequently, the address bus is defined by the I and R registers during this time. This fact is exploited by using the refresh address to control the multiplexer, and also to provide a row select for keyboard scanning.

Cassette interface

The cassette recording output is an attenuated and filtered version of the /SYNC signal. Saving is performed as a simple stream of bits, each byte beginning with the least significant bit (LSB). A '0' bit is recorded as 4 cycles of 3.3kHz. A '1' is recorded as 9 cycles. In both cases, the burst is followed by a 1.3msec period of silence. There is nothing recorded to mark byte boundaries. There is no error detection or correction.

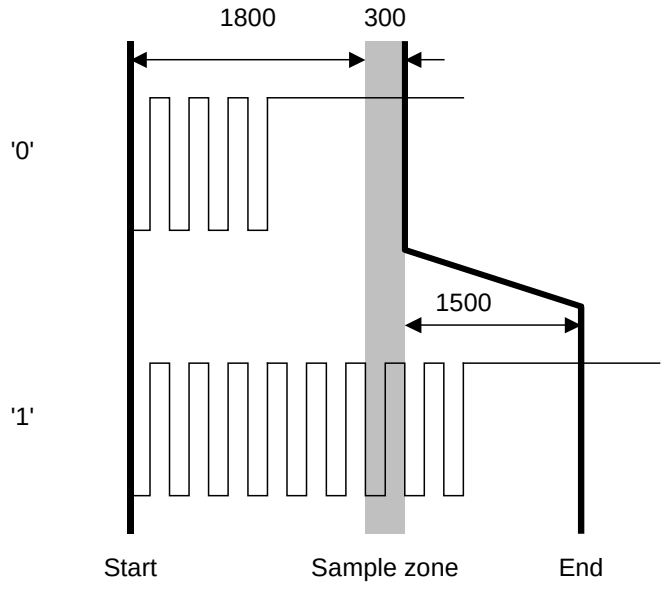


(all timings in microseconds)

Each program is stored with a header ('ZX') followed by the program as it appears in memory, including the terminating pair of 0FFH bytes.

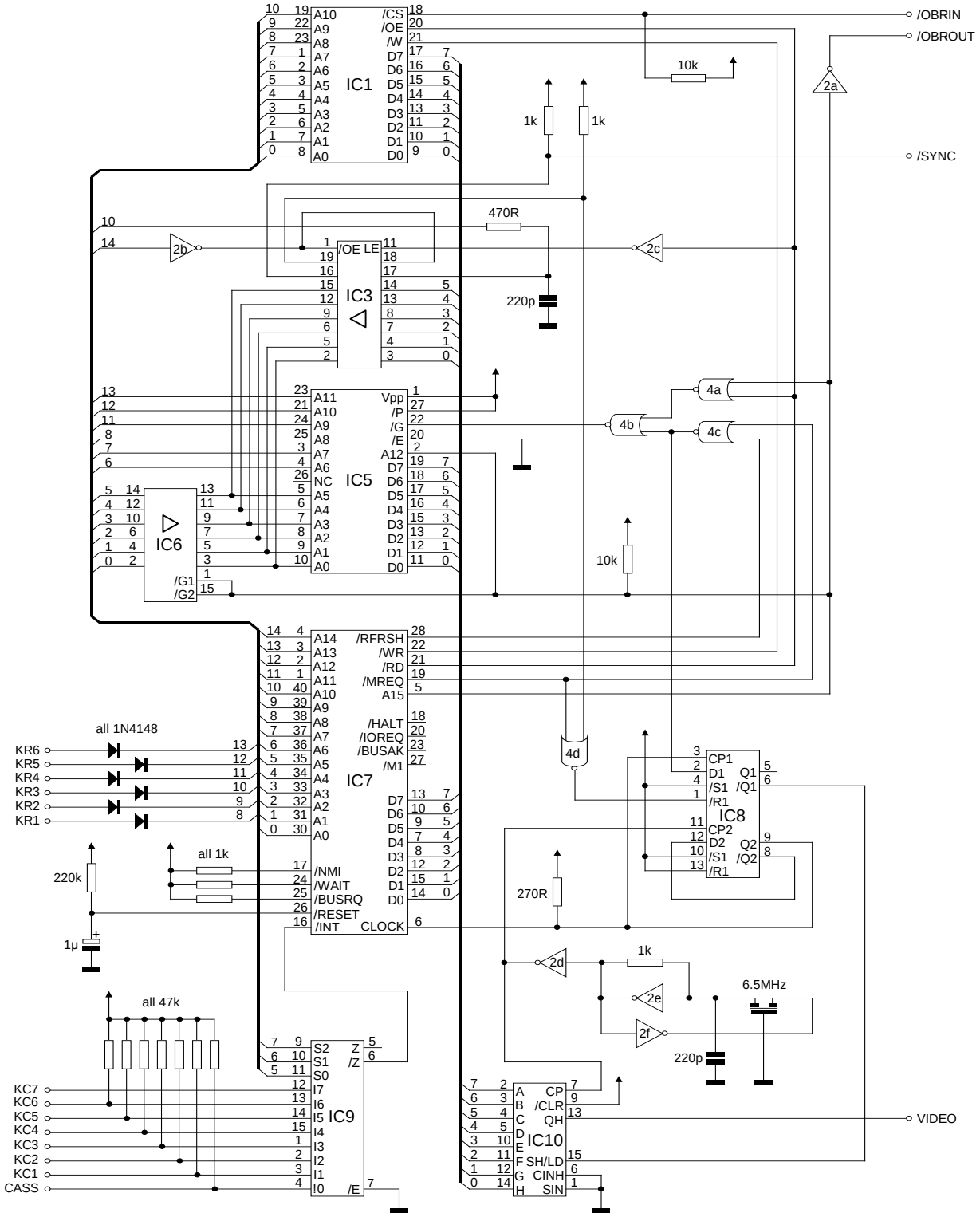
Cassette loading is achieved using the input system for keyboard scanning. One input of the column multiplexer (input 0) is dedicated as the cassette replay input. The algorithm for cassette loading begins with a program loop which waits for a 'start' pulse to arrive. In this loop, the load process can be aborted by the user, upon which any partially loaded program will be discarded. There is also a timeout in this loop: any wait longer than approximately 2 milliseconds will cause any partial byte to be discarded.

Once a start pulse is detected, it is verified a few microseconds later and then a delay of some six cycles of the burst tone occurs, so as to time past the end of a '0' burst yet stay within a '1' burst. Sampling is then performed, several times in fact, over one cycle period, in order to differentiate a '0' burst from a '1' burst. If it is determined that a '1' burst has been detected, then an additional delay is inserted to time past the end of the burst. The process then repeats by looking for further start pulses. Loading from cassette terminates when the end of program marker (two bytes of 0FFH) are read into memory.



(all timings in microseconds)

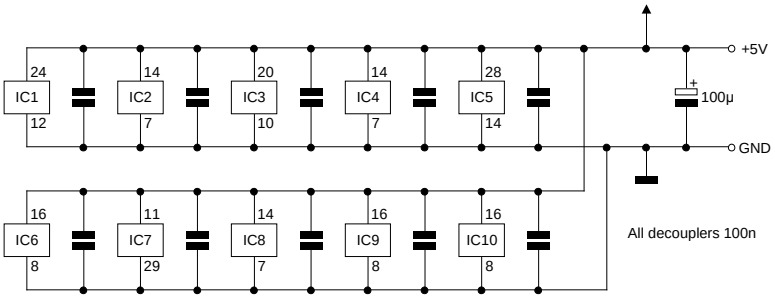
Chapter 3 Circuit diagram



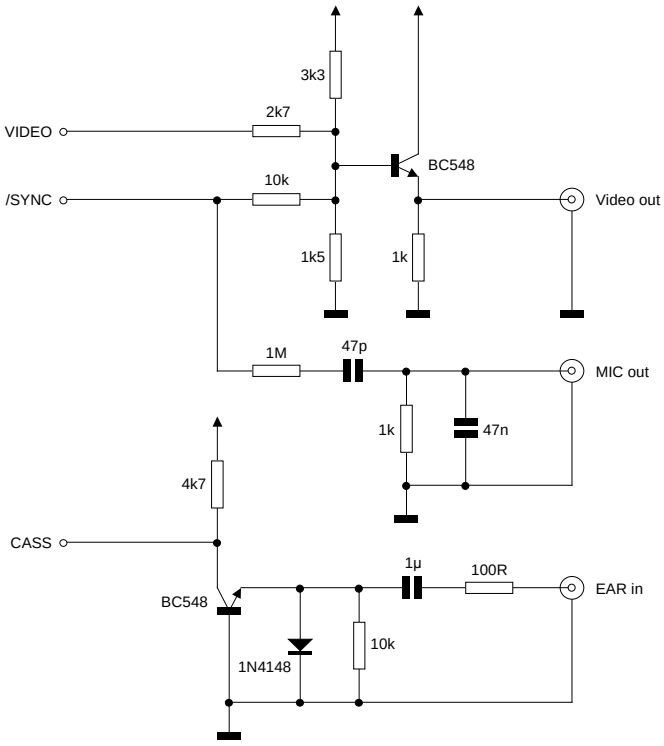
List of ICs

IC1	6116	IC6	74LS367
IC2	74LS04	IC7	Z80A
IC3	74LS373	IC8	74S74
IC4	74LS02	IC9	74LS151
IC5	2764	IC10	74LS166

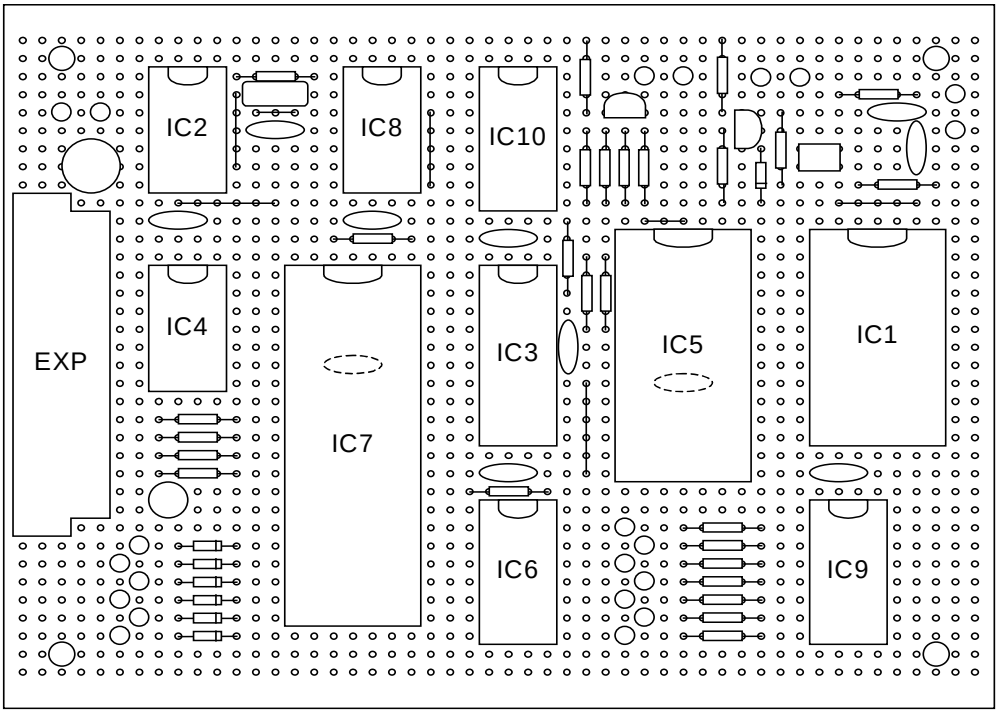
Power distribution



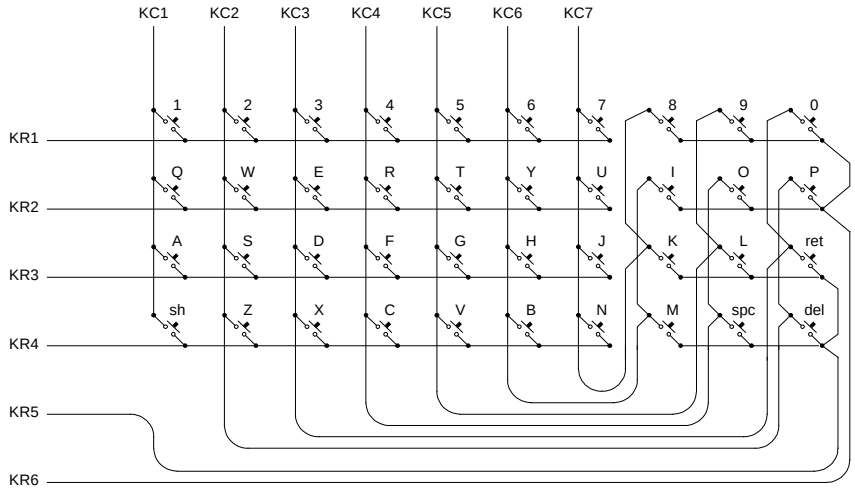
External interfaces



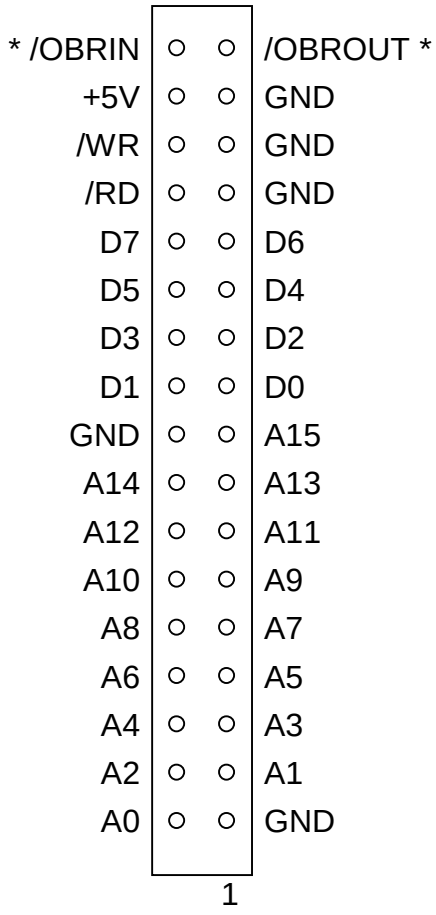
Layout



Keyboard wiring



Expansion connector



* On-board RAM selects in and out
Must be bridged on unexpanded computer

Chapter 4 System timing and design verification

General

The Z80 is clocked at 3.25MHz, resulting in a processor cycle time (Z80 t_C) of 308ns. Half cycle durations (Z80 $t_{W(\Phi L)}$, $t_{W(\Phi H)}$) will be assumed to be 154+/-30ns. In the timing diagrams which follow, all timings are in nanoseconds (ns). In each analysis, the datasheet figures most likely to cause problems are used. Where a data sheet provides only a maximum figure for some parameter, the minimum is assumed to be 0. Where only a minimum is supplied, the maximum is assumed to be ∞.

DC loading analysis

Part	Address bus load (µA)		Data bus load (µA)	
	LOW	HIGH	LOW	HIGH
6116	5	5	5	5
LS373	0	0	400	20
LS367	400	20	0	0
2764	10	10	10	10
Z80	10	10	10	10
LS151	400	20	0	0
LS166	0	0	400	20
Keyboard	100	0	0	0
TOTAL	**925**	**65**	**825**	**65**

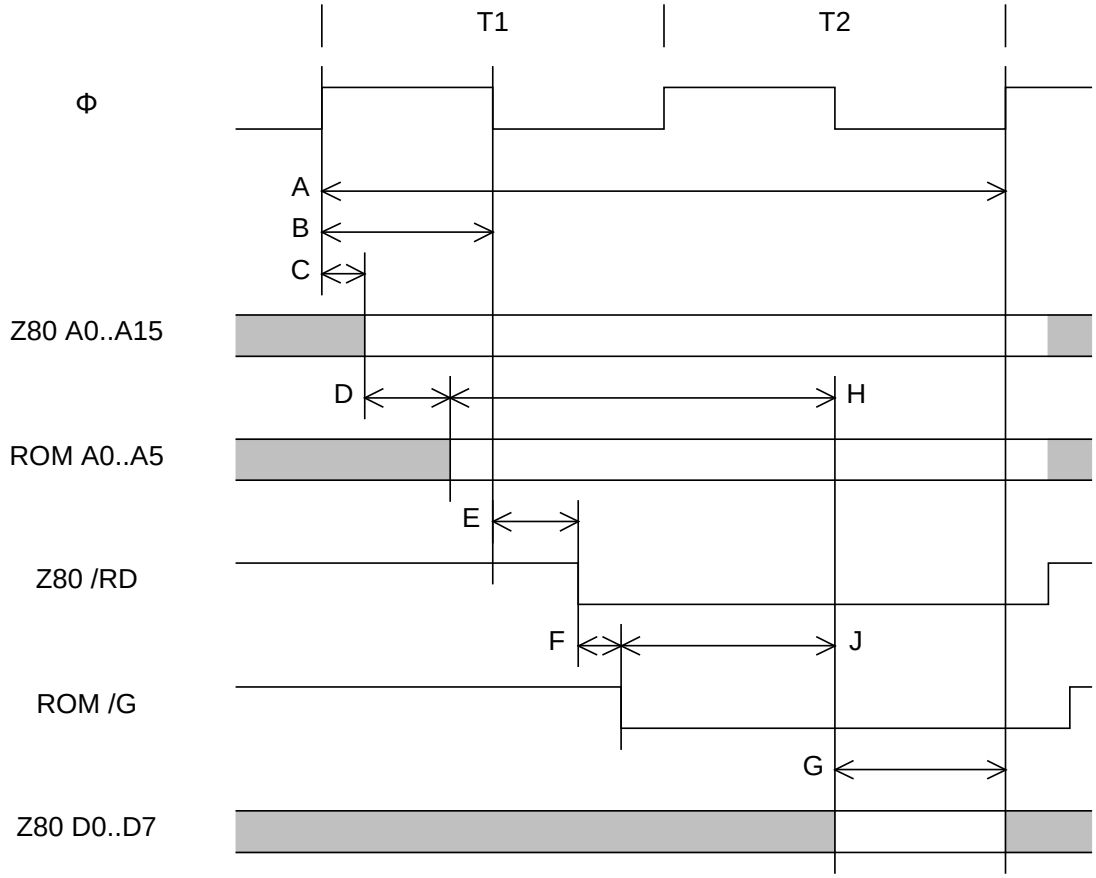
All devices are able to drive these loads. An interesting consideration is the time required for a floating data bus to become invalid. For this determination we will assume a conservative 50pF capacitive loading, a voltage change of 0.4V, and the largest data load current from the above table (825µA). This leads to a time period of:

$$t = \frac{CV}{i} = \frac{50e-12.0.4}{825e-6} = 24e-9 = 24ns$$

In other words, data will persist on the data bus for 24 nanoseconds after it is allowed to float.

ROM read timing

Instruction fetch places the greatest demands on ROM read timing (data read from ROM has similar timing structure but allows a half cycle more access time).



Where:

A = 2 * Z80 t_C = 616
B = Z80 $t_{W(\Phi H)}$ = 184
C = Z80 $t_{D(AD)}$ = 110
D = greater of LS367 t_{PLH}, $t_{PHL,}$ t_{PZH}, t_{PZL} = 40
E = Z80 $t_{DL/\Phi(RD)}$ = 95
F = LS02 t_{PLH} + LS02 t_{PHL} = 23
G = Z80 $t_{S\Phi(D)}$ = 35

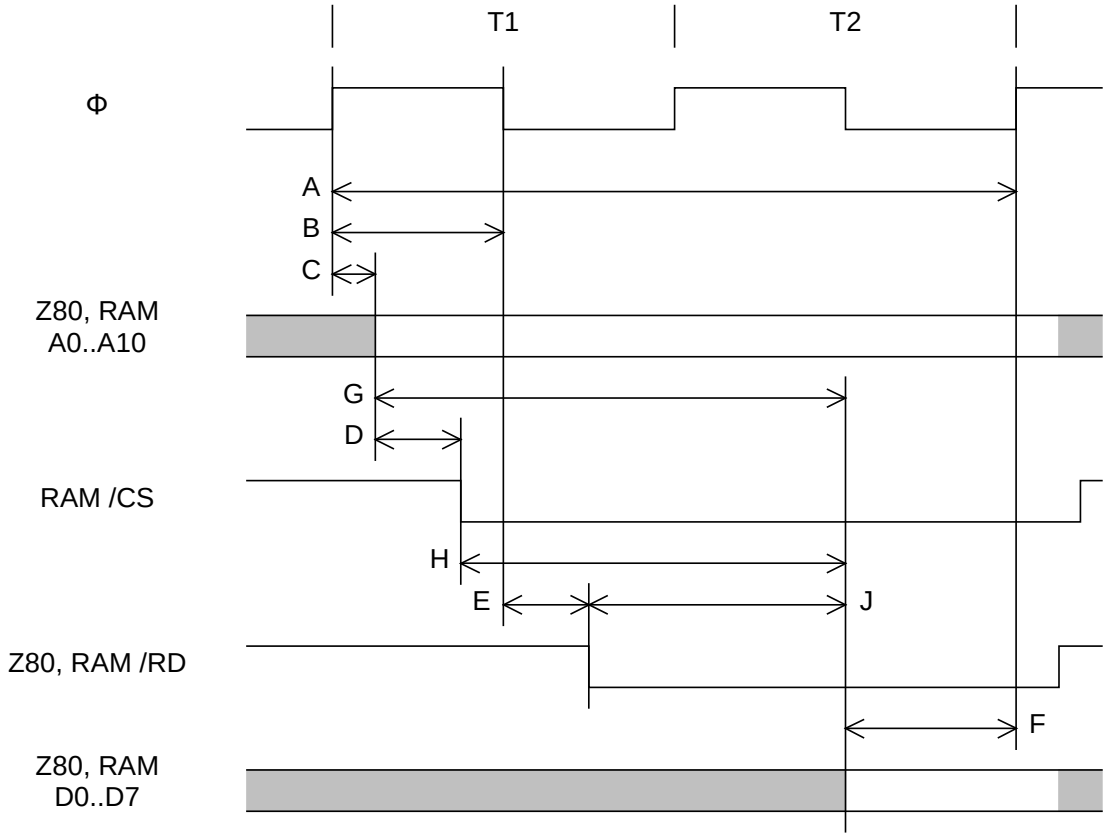
This enables us to calculate:

H (ROM address access time) = 431
J (ROM output enable access time) = 279

These figures are consistent with a 300ns 2764 part. The only other consideration is whether the ROM gets off the data bus quickly enough on termination of a read. The Z80 is quite generous in this respect, providing around one clock cycle (Z80 t_C = 308) between consecutive memory cycles. The output disable time of the 2764 (2764 t_{DF} = 100) is a fraction of this time.

RAM read timing

The timing for RAM read is very similar to that for ROM read:



Where:

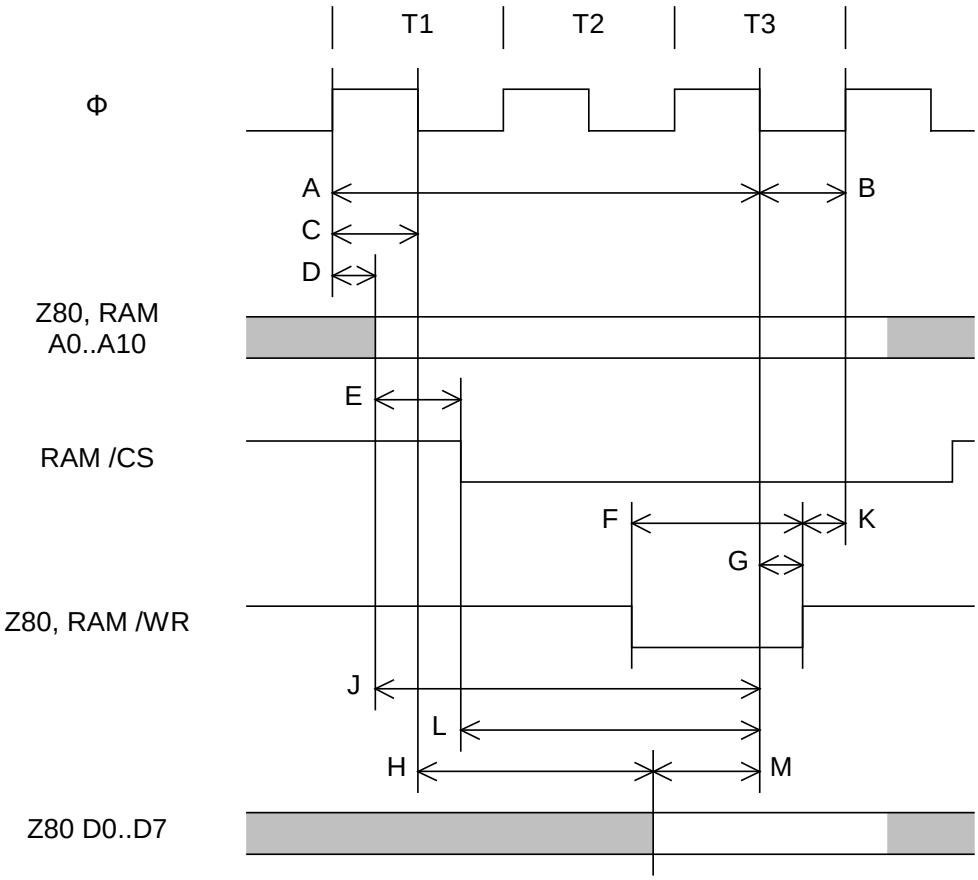
A = 2 * Z80 t_C = 616
B = Z80 $t_{W(\Phi H)}$ = 184
C = Z80 $t_{D(AD)}$ = 110
D = LS04 t_{PHL} = 15
E = Z80 $t_{DL/\Phi(RD)}$ = 95
F = Z80 $t_{S\Phi(D)}$ = 35

This enables us to calculate:

G (RAM address access time) = 471
H (RAM chip select access time) = 456
J (RAM output enable access time) = 302

These figures are consistent with a 450ns 6116 part. Again, the output disable time of the 6116 (6116 t_{DF} = 100) is a fraction of the available time.

RAM write timing



Where:

A = 2 * Z80 t_C + Z80 $t_{W(\Phi H)}$ = 740
B = Z80 $t_{W(\Phi L)}$ = 124
C = Z80 $t_{W(\Phi H)}$ = 184
D = Z80 $t_{D(AD)}$ = 110
E = LS04 t_{PHL} = 15
F = Z80 $t_{W(WRL)}$ = 278
G = Z80 $t_{DH/\Phi(WR)}$ = 80
H = Z80 $t_{D(D)}$ = 150

This enables us to calculate:

J (RAM address valid to end of write) = 630
K (RAM address, data hold time) = 44
L (RAM chip select valid to end of write) = 615
M (RAM data valid to end of write) = 406

Any 6116 part will meet these timing constraints.

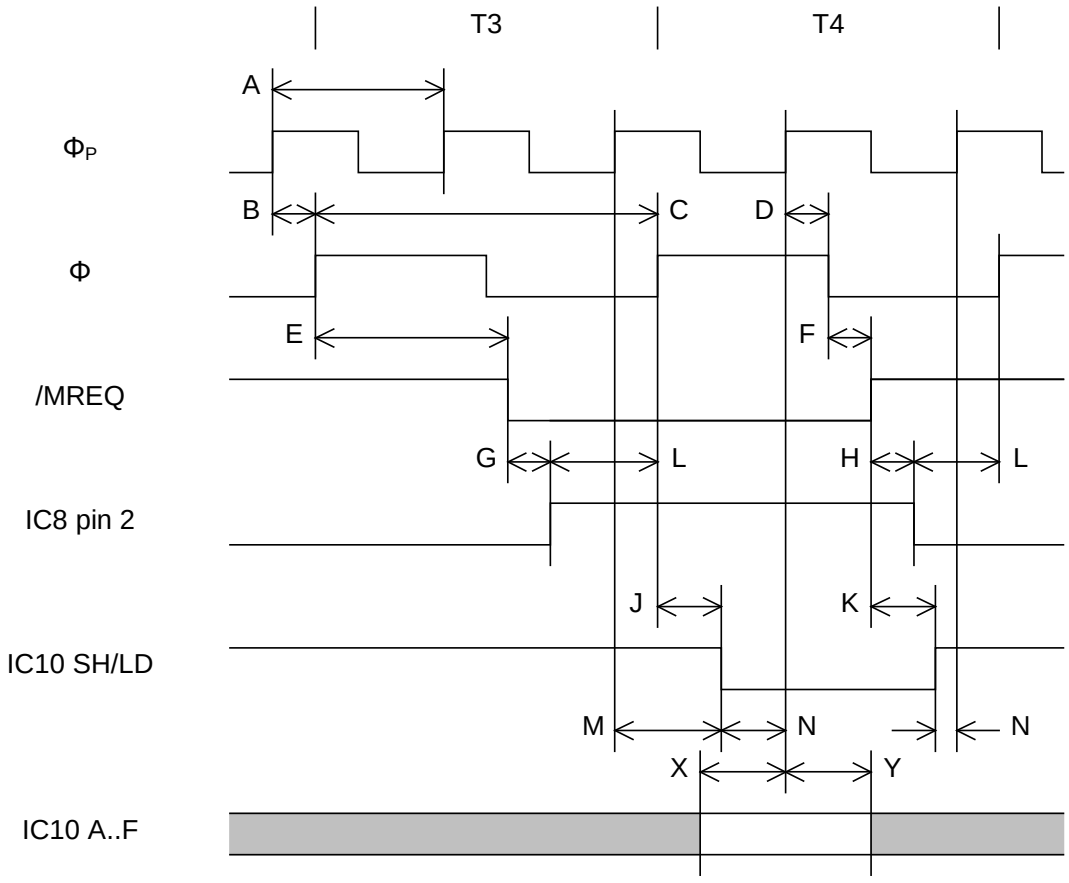
Pixel clock timing

The timing surrounding pixel generation is on two levels. At the fastest level, a pixel clock drives a shift register, and clocks a divider for generation of the processor clock. The latter clock then samples the next shift/load state for the shift register. Since the shift/load state evolves at the slower rate, there are potentially two loads performed for each displayed character. In fact, the shift/load pulse is shortened so that only a single load occurs. This shortening is commanded by /MREQ returning to the high state.

Not all refresh cycles load the pixel shift register. A load inhibit signal is provided by IC4d which is active under the following circumstances:

1. /MREQ is high (for pulse shortening)
2. A14 is low
3. The previous read (i.e. instruction fetch) was NOT from on-board RAM

Note that /RFRSH is omitted from the following diagram for clarity. While /RFRSH being low is a necessary condition for shift register load to occur, /MREQ is the determining control signal.



Where:

$A = t_{CP} = 154$

B = S74 t_{PLH} = 9
C = Z80 t_C = 308
D = S74 t_{PHL} = 9
E = Z80 $t_{DH\Phi(MR)}$ + Z80 $t_{W(MRH)}$ = 279
F = Z80 $t_{DH/\Phi(MR)}$ = 85
G = LS02 t_{PLH} = 18
H = LS02 t_{PHL} = 15
J = S74 t_{PHL} = 9
K = LS02 t_{PHL} + S74 t_{PLH} = 24
X = LS166 t_s = 20
Y = LS166 t_h = 15

This enables us to calculate:

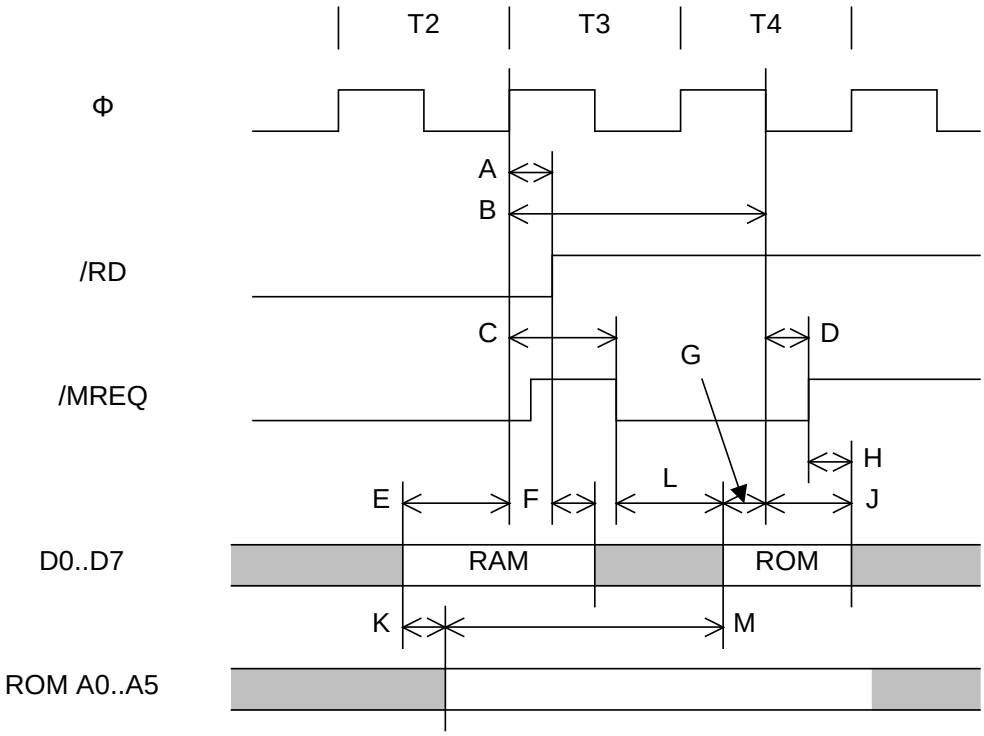
L (S74 data setup time) = 11
M (LS166 SH/LD hold time) = 15 *
N (LS166 SH/LD setup time) = 36

* Capacitance alone can guarantee this figure. The hold time requirement of the S74 (S74 t_h = 2) is similarly guaranteed.

These timings meet the LS166 and S74 specifications. Note that IC8 pin 2 is driven high exactly one cycle of the pixel clock rising edge, specifically, near the centre of T4. From the above diagram we can determine just exactly when the shift register will require stable data from the font table look-up. Relative to the processor clock falling edge in T4, we require that: pixel data is stable X + D = 29ns prior to this edge; and that pixel data holds for a minimum of Y = 15ns after this edge.

Character read and font table look-up

This analysis can draw upon the previous examination of RAM read. In that exercise we arrived at 6116 timing constraints to achieve the minimum data setup time for the Z80 (Z80 $t_{S\Phi(D)}$ = 35). As with the previous analysis, /RFRSH is low during T3 and T4, but is not included because it is gated by /MREQ. ROM address lines A6..A12 are also not included as these are guaranteed stable by software.



Where:

A = Z80 $t_{DH\Phi(RD)}$ = 85
B = Z80 t_C + Z80 $t_{W(\Phi H)}$ = 432
C = Z80 $t_{DH\Phi(MR)}$ + Z80 $t_{W(MRH)}$ = 279
D = Z80 $t_{DH/\Phi(MR)}$ = 0
E = Z80 $t_{S\Phi(D)}$ = 35
F = 6116 t_h = ?
G = 29 (from previous section)
H = LS02 t_{PHL} + LS02 t_{PLH} = 0
J = 15 (from previous section)
K = greater of LS373 t_{PLH}, t_{PHL} = 40

This enables us to calculate:

L = (ROM output enable access time) = 124
M = (ROM address access time) = 420

It is clear that, if the worst case minimum for timings D and H are taken (0), then we cannot meet the LS166 data hold time requirement of timing J. However, in reality these figures are likely to be a significant fraction of the maximum values (Z80 $t_{DH/\Phi(MR)}$ = 85, LS02 t_{PHL} + LS02 t_{PLH} = 23). In any case, the data bus will be going to the float state, and this alone will preserve the data on the data bus for a couple of dozen nanoseconds.

We must also consider the data hold time requirement of the LS373 after /RD returns high (not helped by the inverter IC2c in the LE path). This figure is LS04 t_{PHL} +

LS373 t_h = 35). Again, the 6116 data sheet provides no minimum data hold time for read cycles, instead supplying a data float figure (6116 t_{DF} = 40). But as before, the data bus is going to float, and capacitance will extend the 6116 data hold time sufficiently.

A more serious problem was encountered during the design: the A10 hold time after /RD goes high is calculated as Z80 $t_{D(AD)}$ - Z80 $t_{DH\Phi(RD)}$ = -85, which is 120ns too late to meet the latch hold requirement. This problem was resolved by including an RC delay network in the A10 path to the latch input.

Chapter 5 Photo

Chapter 6 Software

General

The ZX79 runs a dialect of BASIC inspired by National Instruments' National Industrial BASIC Language (NIBL). This is a tiny (integer) BASIC with limited string and array support.