Orton Z80 computer manual

Index

Disclaimer

The information in this document is provided purely as a record of the construction of one of my computer projects. I provide no assurance or guarantee whatsoever as to the accuracy, safety or originality of the contents of this document. I provide no warranty whatsoever as to the suitability for any purpose of the contents of this document. I accept no responsibility whatsoever for any deaths, injuries or losses resulting from the use of the information contained in this document. I have no claims to most of the technology used in my projects or described in this document, indeed, it must be assumed that I have used much copyrighted and/or patented material. But since I do what I do for purely recreational, educational or personal instructional purposes, I do not believe that I am breaking the law. It is up to the individual using the information in this document to determine whether, and to ensure that, their use of the information I provide is both legal and safe. This is supposed to be fun.

Karen Orton 2018

Chapter 1 Hardware

The Orton Z80 computer is minimalist in design, comprising of just three ICs: a 74HC04 hex inverter, a 4MHz Z80, and a 32kByte static RAM (SRAM) IC. The design follows earliest practise of requiring initial program entry through switches on the front panel of the machine. The program so entered may be a simple training program or a bootstrap loader for installation of a larger program.

The SRAM appears twice in the Z80 memory map, one image residing between 0000H and 7FFFH, while a second image resides between 8000H and 0FFFFH. There is nothing mapped into the Z80 I/O space. The computer has two modes depending on the state of the 'Run' switch (see circuit diagram). The front panel controls are as follows (see photos):

Control	**Purpose**
Address LEDs	Indicate the state of the lower half of the Z80 Address Bus
Data LEDs	Indicate the state of the Z80 Data Bus
Power switch	Turns the computer on/off
Power LED	Indicates power state
'Halt' LED	Indicates that the Z80 has executed a HALT instruction and has stopped
'Run' switch	In the up position, the Z80 is held in a wait state while the memory is edited ('program mode') In down position, the Z80 executes instructions in memory at full speed ('run mode')
'Cycle' button	This button generates a 'cycle' - a cleanly generated pulse for performing one of three actions depending on the states of the 'Reset' and 'Write' switches
'Reset' switch	When this switch is in the down position, pressing the 'cycle' button will issue a reset to the Z80 This causes the Address LEDs to go to the 'all zero' state thereby allowing program entry from location 0000H (note that, following power up, it may be necessary to issue two reset cycles to extinguish all Address LEDs)
'Write' switch	(Provided the 'Reset' switch is in the up position) In the up position, pressing the 'cycle' button will cause the Address LEDs to increment In the down position, pressing the 'cycle' button will cause the data entered on the Data switches to be written to the memory location at the address indicated on the Address LEDs
Data switches	Allows entry of data for writing to memory

Note that the issuing of a cycle while the Z80 is running will have unpredictable results. The single exception is changing the state of the 'Run' switch, which is best done while a performing a reset cycle. The procedure for changing the state of the 'Run' switch is as follows:

1. The 'Reset' switch is moved to the down position
2. The 'Cycle' button is held down
3. The 'Run' switch is changed
4. The 'Cycle' button is released

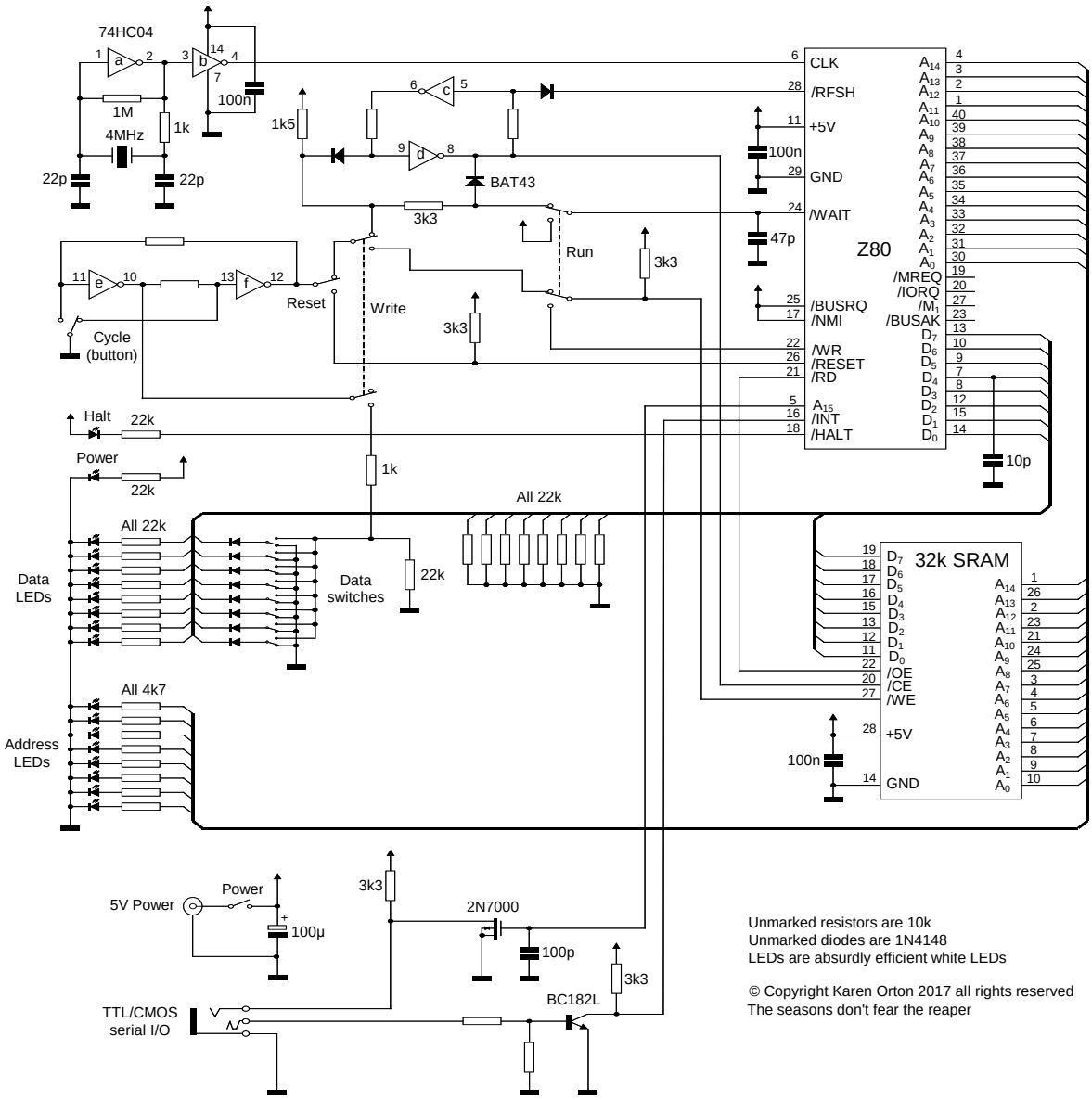
When moving from program mode to run mode, the reset cycle ensures that the Z80 begins execution from location 0000H. Programs must therefore be entered from this address. When moving from run mode to program mode, the reset cycle ensures that program entry begins from location 0000H. Note that moving from run mode to program mode should only be done while the Z80 is in a halted state. For this reason, training programs should execute a HALT instruction on completion. Switching from run mode to program mode while the Z80 is executing instructions may lead to memory corruption.

The serial port is implemented by very obscure means. The serial receive line is connected via a transistor isolator to the Z80 /INT pin. A software UART samples the state of the serial receive line by momentarily enabling interrupts and seeing whether an interrupt occurs. The serial transmit line is connected via a transistor isolator to Z80 address line A15. A software UART controls the serial transmit line by shifting Z80 execution between the two images of the SRAM. The inactive state of the serial transmit line (the high state) corresponds to execution in the lower image (0000H to 7FFFH).

Only the lower eight Z80 address lines are indicated on LEDs, as it is unlikely that anyone will have the patience to enter programs larger than 256 bytes. Larger programs can still be entered but the user must keep track of the high order address lines manually. The LEDs used are high efficiency white types that were salvaged from portable illuminators of the sort one hangs in cupboards. The LED drive current is very low (around 100uA) and puts a minimal load on the Z80/SRAM address and data lines. No buffer ICs are therefore necessary for driving the LEDs. Even at this miniscule drive current, the LEDs in question are still dazzlingly bright when viewed straight on.

Address generation during program entry relies on the Z80, which is deceived into thinking that it is executing NOPs. While the Z80 is completing an instruction fetch, the data bus is pulled to the low state via-pull down resistors thereby presenting a NOP opcode. The Z80 (well, my Z80 at least) tolerates this high impedance on the data bus provided a small capacitor is added to data line D4 to prevent crosstalk from the adjacent CLK pin. Address increment during program entry is accomplished using a handshake circuit which interacts with the Z80 /WAIT and /RFSH signals. The Z80 remains in an instruction fetch wait state until an increment cycle is issued, upon which the Z80 is permitted to fetch one (NOP) instruction and move on to the next.

Serial connection to a PC is achieved using FTDI lead type TTL-232R-5V-AJ. This lead terminates in a 3.5mm stereo jack bearing 5V TTL serial signals. A terminal emulation program should be run on the PC using the settings 9600 baud, 8 data bits, no parity, one stop bit and no handshaking.

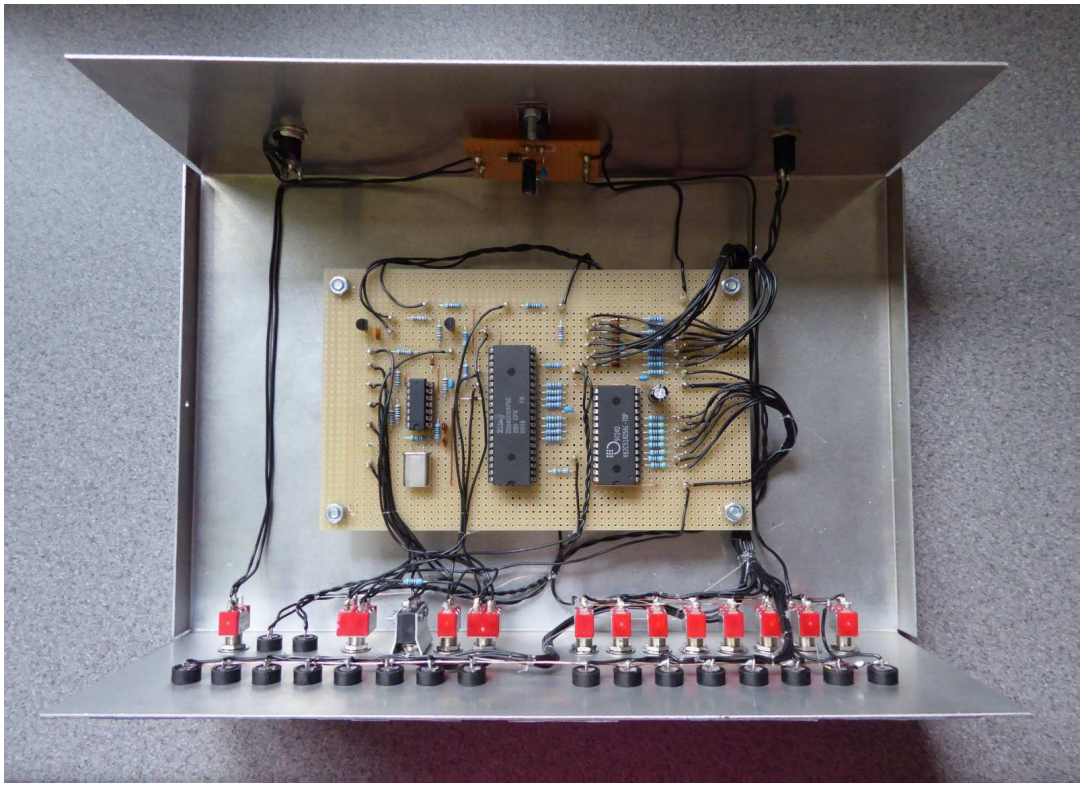


Unmarked resistors are 10k
Unmarked diodes are 1N4148
LEDs are absurdly efficient white LEDs

The seasons don't fear the reaper

Orton Z80 computer circuit diagram

Photos of Orton Z80 computer



Chapter 2 Software

So far, the only software written for the Orton Z80 computer is a simple 'echo' program to verify correct operation of the serial port. The program - echo.asm - receives characters at 9600 baud, no parity, one stop bit, and then retransmits them using the same format and baud rate. It is intended that this will form the basis of a future driver/loader for MS Basic. echo.asm was assembled using the 'TASM' assembler.