Popular Radio-Electronics

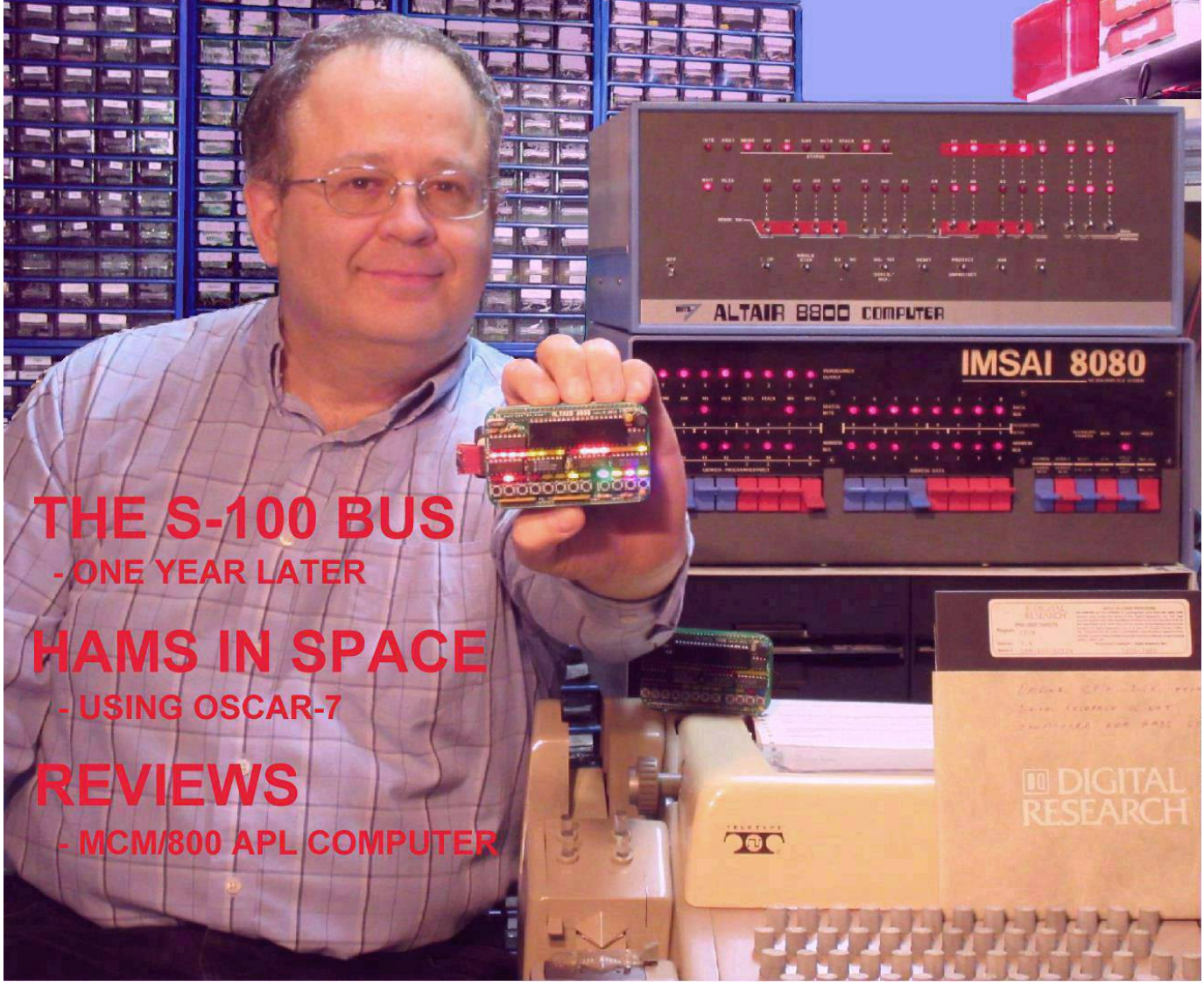
January 1976
75 cents

EXCLUSIVE!

ALTAID 8800

WORLDS FIRST MICRO-COMPUTER THAT FITS IN YOUR POCKET

BUILD IT FOR UNDER $100



THE S-100 BUS
- ONE YEAR LATER

HAMS IN SPACE
- USING OSCAR-7

REVIEWS
- MCM/800 APL COMPUTER

Popular Radio-Electronics

Volume 6, no.1 January 1976

Brother Industries Ltd
Publisher

Lee A. Hart
Editorial Director

Josh Bensadon
Technical Editor

Herb Johnson
Walter Miraglia
Dave Dunfield
Rich Cini
Contributing Editors

Robert Noyce
John Titus
Ed Roberts
Bill Gates
Gary Kildall
Forrest Mims
Don Lancaster
Ward Christensen
Martin Eberhard
Design Consultants

Heath Robinson
Edsel Murphy
Product Reviewers

Dewey Cheatham & Howe
Legal Services

TMSI c/o Lee Hart
814 8th Ave N, Sartell MN 56377 (USA)
http://www.sunrise-ev.com/8080.htm
leeahart@earthlink.net

last revised: 15 Dec 2024

Editorial

New Year's Day is a time to reflect on what we've accomplished over the past year, and what we plan to accomplish in the new year. But what a year it's been!

The AMSAT/OSCAR-7 amateur radio satellite just celebrated its first anniversary in orbit. Hams the world over are now communicating and sending messages with their own satellites.

For years, we've heard that computers will soon be a household item. But it was just a year ago that the Altair 8800 home computer was introduced to make this a reality. Its design and S-100 bus quickly set the standard for hobby computing. Today, there are over a hundred vendors selling S-100 boards and computers.

It is clear that semiconductor technology is driving this rapid pace of change. Intel CEO Gordon Noyce has observed that integrated circuit performance is doubling every two years. The EIA has reported that computers have an annual growth rate of 50% per year.

So what's next? Science fiction writers have speculated that we will soon be carrying computers in our pockets, ready to do our calculations, note taking, information retrieval, communications, and more.

That future has arrived! In this issue, we present the **ALTAID 8800**, the first pocket micro-computer with all the power of these earlier machines, at a fraction of the size and cost. It features a 2 MHz Intel 8080A CPU, 64K of memory, and up to 448K of solid-state disk storage, all in a package the size of an Altoids tin.

Let me offer some insight into our goals in presenting this momentous project. We were determined that it should not be another big box of blinking lights -- fun to build and watch, but of limited usefulness. We wanted a computer for learning vital new skills. One that is expandable, almost without limit. A project to unleash our reader's creativity and imagination, to invent new applications that will become a part of our daily lives in this new age of computers.

Welcome to the forefront of an exciting new future!

Lee A. Hart

Editor

By Gil Bates and Chip Hacker

The era of the pocket computer -- a favorite fixture in science fiction -- has arrived! The **Altaid 8800** microcomputer packs full-size performance in a pocket-sized package. Best of all, you can build it yourself for under $100! In this issue of Popular Radio-Electronics, we are proud to present the complete plans and construction details for this amazing computer kit.

The Altaid 8800 is not a toy or demonstrator. It is a serious personal computer, with all the capability and performance of the full-size MITS Altair 8800 in a package that fits in your pocket.

The small size and low price are the result of advances in semiconductor technology, clever packaging, minimizing parts count, and a highly innovative design.

Look inside most computers, and what do you find? Air, and lots of it! Parts are widely spaced because they need lots of air for cooling. The large spacings increase the need for bus driver ICs. The large case also provides room for the power supplies needed for so many parts.

The Altaid 8800 takes a different approach. Instead of TTL, it uses CMOS logic to greatly reduce power requirements. Without the heat, parts can be packed much tighter. The power supply can be much smaller (even battery operation is feasible). The tight packing also reduces the size and cost of the PC boards.

CMOS logic also has very low input current. Together with the shorter trace lengths due to the small size, this eliminates the need for numerous bus driver ICs.

The result is a complete computer on just two 3.5" x 2" PC boards. The CPU board has the micro-processor, with its clock, power supply, and support ICs. It also has the Front Panel interface; switches and LEDs to load and examine memory and run programs.

The MIO (Memory/Input/Output) board has the memory, parallel, serial, and bus interface to connect the Altaid 8800 to other devices.

That's it! Just connect power (5v at about 0.5amp), and you have a working computer, with no extra boards or accessories needed.

CPU: The heart of any computer is its CPU (Central Processor Unit). The Altaid 8800 uses the industry-standard Intel 8080A. It runs at 2 MHz, and can address up to 64K of memory and 256 I/O ports at a time. Its widespread use in many other computers means that it won't become obsolete any time soon.

The 8080A needs three supply voltages (+5v, +12v, and -5v). To simplify things, a tiny DC/DC converter is included on the CPU board to generate these voltages from a single +5v supply.

MEMORY: The MIO board has sockets for two "bytewide" memory chips. This new standard makes it possible to upgrade memory in the future just by changing chips -- no extra boards are needed. Memory beyond 64K is bank-selectable, to allow almost unlimited expansion.

The RAM socket supports everything from today's 2K byte 6116, to new chips up to 512K. That's as much storage as two 8" floppy disks! Battery backup holds RAM data without power.

The ROM socket supports EPROMs from a 2K byte 2716 to as much as 64K as they become available. The kit comes with programs in ROM for the Front Panel, a serial monitor, and a BASIC operating system.

I/O: The Altaid 8800 includes an RS-232/TTL serial port that runs at up to 38.4K baud. There is also a second audio input/output port to load and save programs and data on cassettes.

An 8-bit parallel input and output port is provided. It is normally used to control the Front Panel switches and LEDs; but is also available for other uses.

An expansion bus is included, so additional boards can be added to the stack for your own projects or expansion.

SOFTWARE: A computer is useless without software. The popularity of the 8080A means that more software is available for it than any other microcomputer. Monitors, text editors, assemblers, debuggers, and languages like BASIC are already available; with more announced every day.

Microprocessors are designed as controllers, to replace boards full of hardware logic with a few chips and software. The Altaid 8800 follows this design philosophy, and uses software in ROM to drastically simplify the Front Panel and serial port hardware.

To tie it all together, the Altaid 8800 runs the CP/M-80 operating system. CP/M works like an electronic filing cabinet for all your programs and data. Instead of laboriously typing them in, or slowly loading them from cassette tapes, they can be accessed in the blink of an eye from the on-board memory.

PARTS LIST

CPU BOARD

28	D0-11, A0-15	LED, T1 or 2x5mm, red/yel/grn
1	C1	33uF 10v electrolytic capacitor
1	C4	4.7uF 25v 0.1" ceramic capacitor
5	C2,C3,C9-11	0.1uF X7R 0.1" ceramic capacitor
1	C5	47pF NPO 0.1" ceramic capacitor
1	C6	330pF X7R 0.1" ceramic capacitor
1	C7	0.01uF X7R 0.1" ceramic capacitor
1	C8	4.7uF 16v tantalum capacitor
1	D01	1N5818 Schottky diode
1	D02	1N5241B 11v zener diode
1	D03	1N4148 signal diode
1	L1	100uH inductor
2	P1. P2	30-pin male header
1	R1	100x4 8-pin isolated SIP resistor
1	R2	10Kx7 8-pin bussed SIP resistor
1	R3	12Kx8 9-pin bussed SIP resistor
1	R4	2.2Kx3 6-pin isolated SIP resistor
1	R5	200K 5% 1/4w resistor
1	R6	0.47ohm 5% 1/4w resistor
1	R7	220 ohm 5% 1/4w resistor
1	R8	18.2K 1% 1/4w resistor
1	R9	2K 5% 1/4w resistor
12	S0-11	5mm tactile pushbutton switch
1	U1	8080A microprocessor & socket
1	U2	8224 clock generator-driver
1	U3	74HC174 hex latch
1	U4	74HC02 quad 2-input NOR
1	U5	74HC14 hex inverter, schmitt trigger
1	U6	74LS145 decimal decoder
1	U7	MC34063 switching regulator
1	Y1	18.432MHz crystal

MIO BOARD

2	C1,C2	33uF 10v electrolytic capacitor
1	C3	4.7uF 25v 0.1" ceramic capacitor
2	C4, C5	0.047uF X7R axial ceramic capacitor
4	C6-C8, C11	0.1uF X7R 0.1" ceramic capacitor
1	C9	0.68uF 50v 0.2" capacitor
1	C10	0.22uF 50v 0.2" capacitor
2	D1, D6	1N4148 signal diode
4	D2-D5	1N5818 Schottky diode
60	JP1, JP2	socket pins for 0.025" square posts
1	J1	miniature stereo phone jack
1	P1	6-pin male header
1	Q1	FJN4303 PNP with 22K resistors
2	Q2,Q3	2N3904 NPN transistor
1	R1	2.7K 5% 1/4w resistor
1	R2	1K 5% 1/4w resistor
1	R3	2.2Kx4 8-pin isolated SIP resistor
1	R4	10Kx4 8-pin isolated SIP resistor
1	R5	3.3K 5% 1/4w resistor
1	U1	2716-27512 (2K-64K) EPROM, programmed, with IC socket
1	U2	6116-628512 (2K-512K) RAM, with IC socket
1	U3	74HCT151 8-channel multiplexer
1	U4	74HC138 3-to-8 line decoder
1	U5	74HC259 addressable latch
1	U6	74HC273 octal latch
1	U7	74HC245 octal transceiver
1	U8	1489 RS-232 receiver

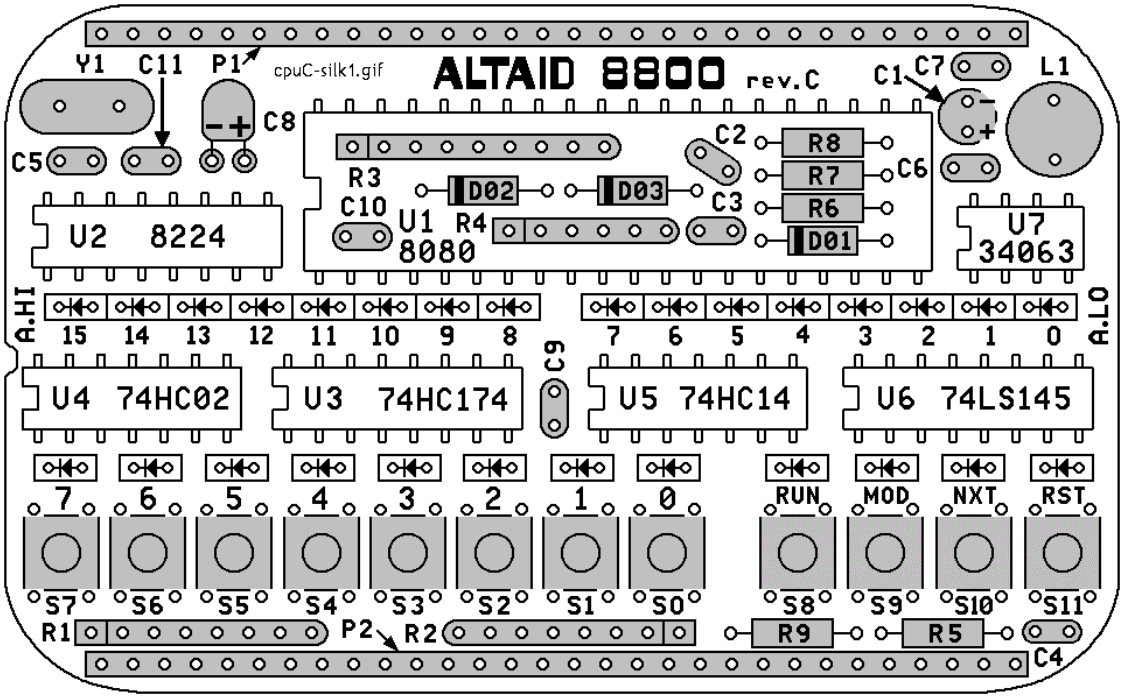
CONSTRUCTION: Due to the small size, printed circuit boards are highly recommended. PC boards, parts, complete kits, and additional information are available from Itty Bitty Micro Company at nominal cost.

Assembly is not difficult; but this is not a project for an inexperienced builder. The parts are small, and solder pads are close together.

Normally, only the CPU, RAM, and ROM chips are socketed. They are the most likely to be changed; and sockets provide a bit more height (some parts are located under these ICs to save space). You can use sockets for the other ICs, but do not use cheap ones; they can be a reliability headache!

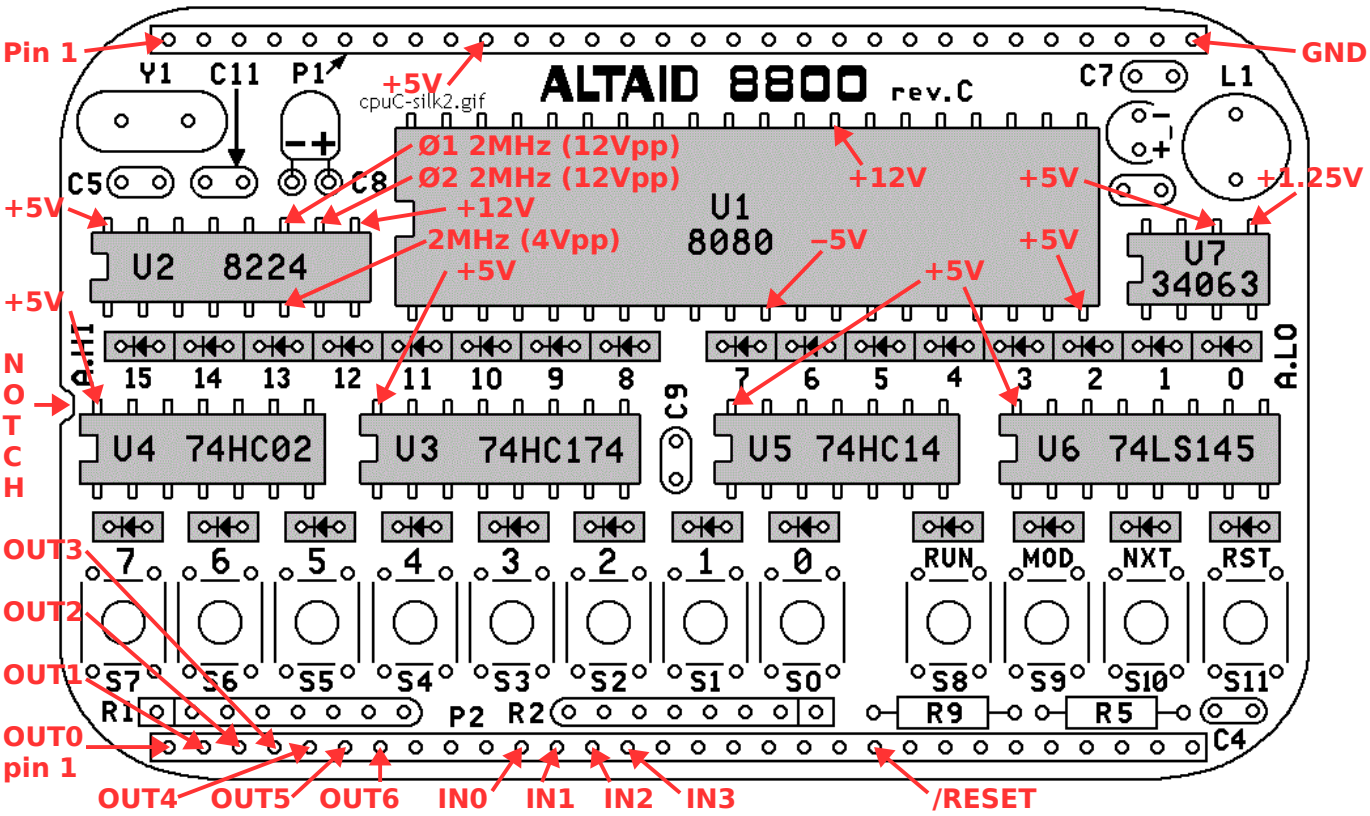
SPECIFICATIONS

Processor: Intel 8080A 8-bit NMOS
Speed: 2 uSec (min) bus cycle time
Instructions: 78 (181 with variants)
Registers: 12 bytes (PC, SP, AF, BC, DE, and HL)
ROM Memory: 2K-64K bytes, in 8K-32K banks
RAM Memory: 2K-512K bytes, in 8K-32K banks
I/O Ports: one 8-bit input, one 8-bit output, TTL or RS-232 serial port, audio cassette I/O port
Front Panel: Examine memory, change memory, or begin program execution at any address
Expansion: Full 60-pin bus for accessory boards
Power: +5vdc at 0.5a
Size: 3.5"W x 2.1"H x 0.75"H

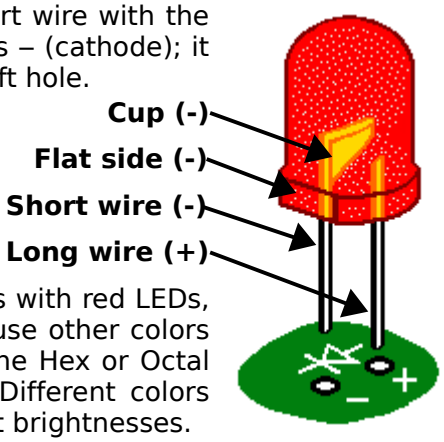


CPU rev.C ASSEMBLY: Install the small parts first (shaded in the illustration above). Mark each box ▣ as the parts are installed. If you aren't sure if you have the right resistor value or diode polarity, check it with your multimeter.

- ☐ C1: 33uF electrolytic capacitor. It is polarized: Install so its +/– markings match the board.
- ☐ C2, C3, C9-11: 0.1uF radial ceramic ("104").
- ☐ C4: 4.7uF ceramic (marked "475").
- ☐ C5: 47pF ceramic capacitor ("470").
- ☐ C6: 330pF ceramic capacitor ("331").
- ☐ C7: 0.01uF ceramic capacitor ("103").
- ☐ C8: 4.7uF 16v tantalum capacitor. Polarized! Install it with the + mark on the right (next to the 8080). Lay it down flat as shown above.
- ☐ D01: Schottky diode (black, marked "1N5818") The banded end goes on the left as shown.
- ☐ D02: 11v Zener diode (reddish glass, marked "1N5241B" or just "241B"). Band on left end.
- ☐ D03: Signal diode (reddish glass, marked "1N4148"). Band on left end.
- ☐ Y1: 18.432 MHz crystal (marked "R184LEA").
- ☐ S0-S11: Tactile switches, 5mm square.
- ☐ R1: 100x4 isolated 8-pin SIP resistor. (Kit part is red, marked "L83S101"). Printed side down.
- ☐ R2: 10Kx7 bussed 8-pin SIP resistor. (Kit part is yellow, "8R-1-103"). Printed side on top.
- ☐ R3: 12Kx8 bussed 9-pin SIP resistor. (red, "L91C123"). Pin 1 on left. Bend flat, and solder.
- ☐ R4: 2.2Kx3 isolated 6-pin SIP. (red, marked "L63S222"). Print down. Bend flat, then solder.
- ☐ R5: 200K resistor (red-black-black-org-brn).
- ☐ R6: 0.47 ohm (black-yellow-violet-silver-brown).
- ☐ R7: 220 ohm (red-red-brown-gold).
- ☐ R8: 18.2K (brown-gray-red-red-brown).
- ☐ R9: 2K (red-black-red-gold).
- ☐ L1: Inductor 100uH. (Kit part is surface-mount; use pieces of scrap wire to add leads to it.)
- ☐ P1-P2: 30-pin headers. **WAIT**! Install them later, when you assemble the MIO board.



☐ LEDs D0-11, A0-15: LEDs are polarized; the **long** wire is + (anode); put it in the **right** hole. The short wire with the "cup" inside is – (cathode); it goes in the left hole.



The kit comes with red LEDs, but you can use other colors to separate the Hex or Octal digits. Note: Different colors have different brightnesses.

ICs and SOCKETS:

☐ U1: Install two 20-pin socket strips (Jameco 78642), or a 40-pin machined-pin IC socket (Jameco 41136) with the center bars cut out to leave room for the parts underneath.

U2-U7: Install the ICs directly, or in IC sockets. Pin 1 goes in the lower left! The text on all the ICs should be right-side-up.

☐ U2: 8224 or KP580 clock generator-driver.
☐ U3: 74HC174 hex latch.
☐ U4: 74HC02 quad 2-input NOR gate.
☐ U5: 74HC14 hex Schmitt-trigger inverter.
☐ U6: 74LS145 decimal decoder/driver.
☐ U7: MC34063 or CHN063 switching regulator.

TESTING: Don't insert your priceless 8080 yet! Let's see if everything works first. The Front Panel switches and LEDs are perfect for this. You'll need some clip leads and a multimeter. An oscilloscope is better, but not necessary.

Connect a regulated 5v power supply; +5v to P1 pin 10, GND to P1 pin 30. Current should not exceed 100mA. Check voltages on these IC pins:

☐ U1 pin 20=+5v, pin 28=+12v, pin 11=−5v
☐ U2 pin 16=+5v, pin 9=+12v
☐ U3, U6 pin 16=+5v
☐ U4, U5 pin 14=+5v
☐ U7 pin 6=+5v, pin 5=+1.25v

The voltage on /RESET (P2 pin 21) should be high (+5v), and go low (GND) when you press and hold the RST (Reset) key for 2 seconds.

Connect OUT4=GND, OUT5=+5v, OUT6=+5v (binary code 110=6). This sets the "6" output of decoder U6 low. Connect /RESET to OUT0 (P2 pin 1). The "RUN" LED should be on, and go out when you press and hold the RESET key.

Test the other LEDs the same way. Connect /RESET to OUT0-3, and a binary code to OUT4-6 to select each of the 7 rows of 4 LEDs.

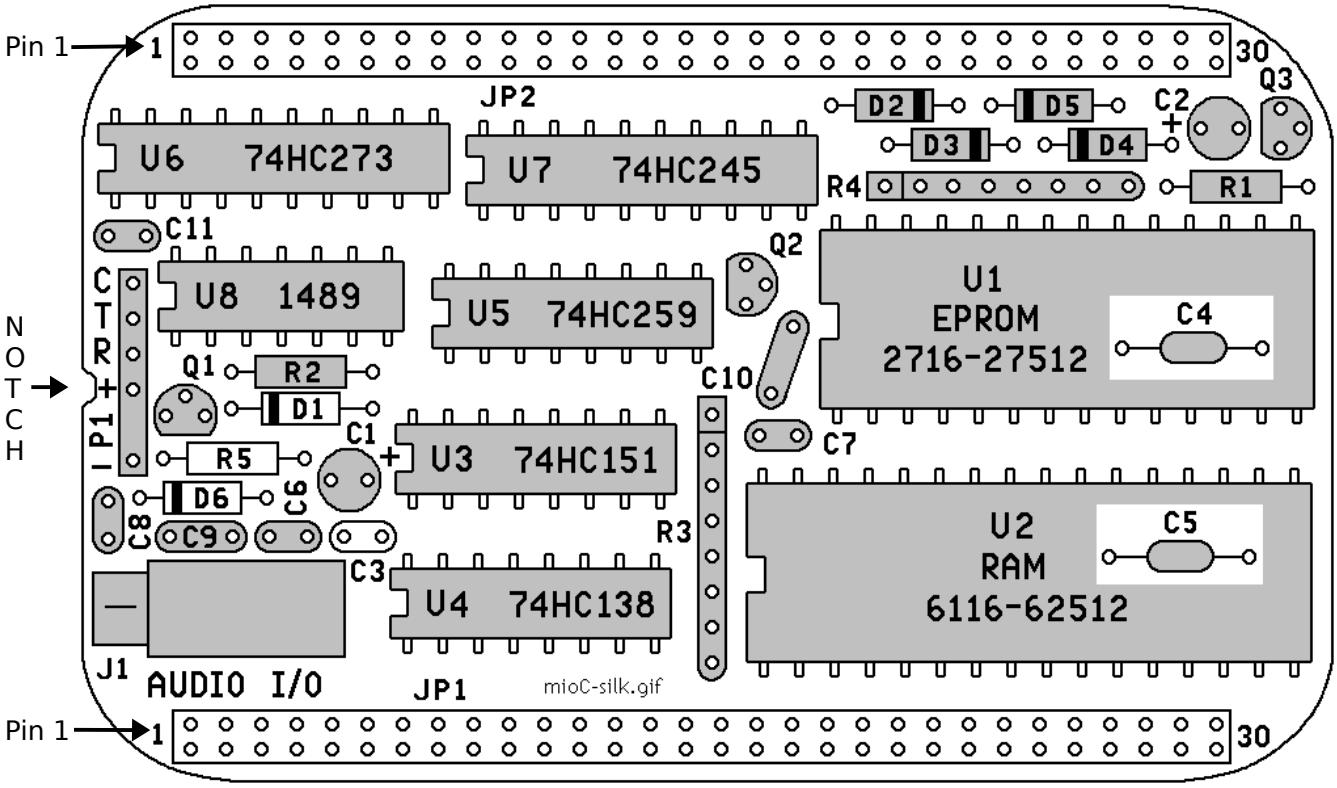
Test the other switches by selecting its row with OUT4-OUT6, and check the voltage on its IN0-IN3 line with your meter. IN0-IN3 will be high, and go low when you press the key.

Test for a 2 MHz signal at U2 pin 6 (Ø2TTL, 0-4v), pin 10 (Ø2, 0-12v), and pin 11 (Ø1, 0-12v).

If it all checks out, **NOW** insert the 8080 into its socket. Continue to assemble the MIO board.

MIO rev.C ASSEMBLY: Install the shaded parts (shown below) as done on the CPU card. Mark each box ▣ as the parts are installed.

- ☐ C1, C2: 33uF capacitor. These are polarized; install so the +/– leads match the board.
- ☐ C4, C5: 0.047uF X7R axial ceramic ("473").
- ☐ C6, C7, C8, C11: 0.1uF X7R radial ceramic (marked "104").
- ☐ C9: 0.68uF radial ceramic ("684").
- ☐ C10: 0.22uF radial ceramic ("224").
- ☐ D2-D5: 1N5818 Schottky diode. Banded end of D2 and D3 on right, D4 and D5 on left.
- ☐ R1: 2.7K 5% ¼w resistor (red-violet-red-gold)
- ☐ R2: 1K 5% ¼w (brown-black-red-gold).
- ☐ R3: 2.2Kx4 isolated 8-pin SIP resistor. (Kit part is black, marked "8B222G"). The printed side goes on the left.
- ☐ R4: 10Kx4 isolated 8-pin SIP resistor (red, marked "L83S103"). Printed side on bottom.
- ☐ J1: Miniature phone jack. Bend the 3 tabs down to fit into the holes on the PC board.
- ☐ P1: 6-pin male header. Remove pin 2 as a key.
- ☐ Q1: FJN4303 PNP transistor with 22K internal base resistors. (Kit part is marked "R4303").
- ☐ Q2, Q3: 2N3904 NPN transistor ("2N3904").
- ☐ U1: 2716-27512 (2K-64K) EPROM. (Kit part is a 32K EPROM marked "ALTAID05"). Install a 28-pin IC socket, or two 14-pin socket strips. Then install your EPROM. (If you use a 24-pin EPROM, install it at the right end of the socket, so socket pins 1-2-27-28 are empty.)
- ☐ U2: 6116-628512 (2K-512K) RAM. Install a 32-pin IC socket, or two 16-pin socket strips. Then install your RAM. A 512K RAM is supplied with the kit. (Install a 24-pin or 28-pin RAM at the right end of the socket.)
- ☐ U3: 74HCT151 8-channel multiplexer.
- ☐ U4: 74HC138 3-to-8 line decoder.
- ☐ U5: 74HC259 addressable latch.
- ☐ U6: 74HC273 octal latch.
- ☐ U7: 74HC245 octal transceiver.
- ☐ U8: 1489 or 75189 quad RS-232 receiver.



EPROM SIZE: Install wires at W1-W2-W3 (under U1 on bottom of board) for the size used.

	W1	W2	W3
☐ 64K (27512, 27C512)	2-3	1-2, 4-5	3-4
☐ 32K (27256, 27C256)	2-3	2-3, 4-5	3-4
(the kit has a 32K 27C256 marked "ALTAID05")			
☐ 16K (27128, 27C128)		2-3, 4-5	1-2, 3-4
☐ 8K (2764, 27C64)		2-3, 4-5	1-2
☐ 4K (2732, 27C32)		4-5	2-3
☐ 2K (2716, 27C16)		3-4	2-3

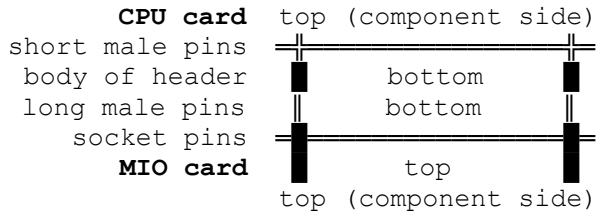
RAM SIZE: Install wires at W4-W5-W6 (under U2 on bottom of board) for the RAM size used. Wires go between the center hole and A or B. Foil jumpers are already in place for a 512K RAM; cut them and install new wires for other sizes.

	W4	W5	W6
☐ 512K (628512)	B	B	B
☐ 128K (62128)	B	B	A
☐ 32K (62256)	B	B	A
☐ 8K (6264)	B	A	A
☐ 2K (6116)	A	A	

JP1, JP2: There are two ways to build it:

☐ Hard -- but thin enough to fit in an Altoids tin.

Use female socket pins for JP1 and JP2 (Mill-Max #0415-0-15-01-16-27-10-0; 60 are supplied with the kit). They space the cards 0.1" apart, so they will fit in the Altoids tin.

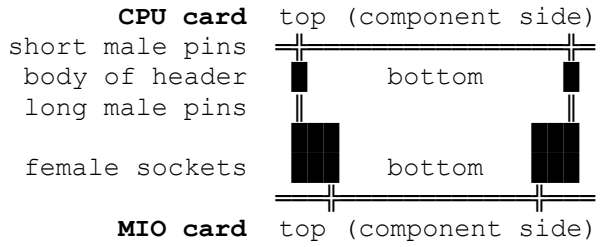


1. Push the socket pins onto the short end of the pins of the CPU card headers.
2. Place the closed end of the socket pins on a table or hard surface. Gently push or tap the long end of each header pin until it "bottoms out" in the socket pin.
3. Fit the two cards together, so the header pins enter the bottom holes of the CPU card, and the socket pins enter the bottom holes on the MIO card. **Be sure P1 connects JP1, and P2 connects JP2!** This holds the two cards in proper alignment.
4. **NOW** solder all the pins of P1 and P2 on the CPU card, and JP1 and JP2 on the MIO card. Check carefully for any unsoldered pads or solder shorts between pads.

☐ Easy -- but too thick to fit in an Altoids tin.

Use female sockets (Molex KK #22-18-2101; not supplied) for JP1 and JP2. They space the cards 0.4" apart; too thick to fit in an Altoids tin, but easy to get if you bought bare cards.

The "10" in the part# is for a 10-pin version, which is easier to get. You'll need six 10-pin parts to make the two 30-pin connectors.

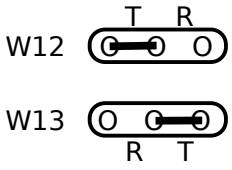


1. Plug the long pins of the P1 and P2 headers (from the CPU card) into the top holes of the KK sockets.
2. Place the KK sockets on the bottom of the MIO card, with their pins in the smaller holes (away from the edge); and their top holes closer to the edge of the board.
3. Fit the two cards together so the short pins of the headers enter the bottom holes of P1 and P2 on the CPU card. **Be sure P1 connects JP1, and P2 connects JP2!** This holds the two cards in proper alignment.
4. **NOW** solder all the pins of P1 and P2 on the CPU card, and JP1 and JP2 on the MIO card. Check carefully for any unsoldered pads or solder shorts between pads.

SERIAL I/O: There are two types of serial ports (TTL and RS-232) that use different logic levels and polarities. Do **ONE** of the following:

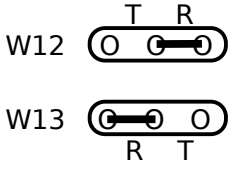
For **TTL** idle = +3.3v to +5v
Do these steps: active = 0 to +0.5v

☐ C3: **Short** location C3 with a piece of wire.
☐ R5: Do not install R5 at this location. Instead:
☐ D1: Install R5 3.3K 5% ¼w (orange-orange-red-gold) in place of D1.
☐ D6: Do not install D6.
☐ W12, W13 Jumpers (on bottom of board, under U8). Foil jumpers should already be in place for TTL. If not, add them like this:



For **RS-232** idle = –5v to –15v
Do these steps: active = +5v to +15v

☐ C3: 4.7uF 50v ceramic (marked "475").
☐ D1, D6: 1N4148 signal diode. Position the banded end on the left as shown.
☐ R5: 3.3K 5% ¼w (orange-orange-red-gold).
☐ W12, W13 Jumpers (on bottom of board, under U8): **Cut** the "T" foil jumpers, and install wire jumpers to configure it like this:

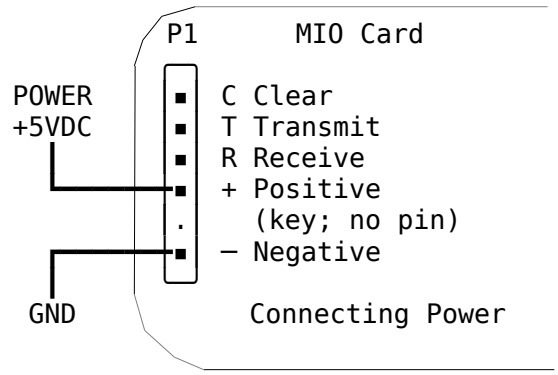


☐ Carefully plug the two cards together, solder sides facing each other. Trim any long leads so a piece of paper can freely slide between the cards. **Be sure P1 plugs into JP1, and P2 plugs into JP2!** There is a wide stripe on the edge next to P2 and JP2, and a notch on the pin 1 end of each PC card.

Congratulations! Your Altaid 8800 is now fully assembled. The next step is to power it up, and see what it can do.

POWER IT UP: Connect a regulated 5 VDC power supply that can provide about 500mA. Connect +5V to P1 pin 3, and GND to P1 pin 1 on the MIO card as shown below.

If you assembled it for a TTL serial interface, you can also power it from your PC terminal with a USB-serial adapter (Sparkfun DEV-09718, www.sparkfun.com/products/9718 or equal).



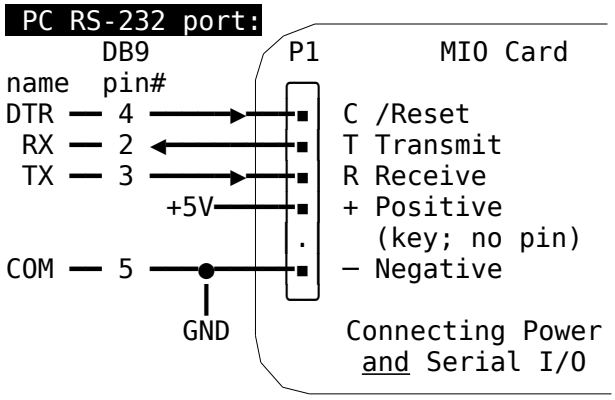
FRONT PANEL OPERATION: The following instructions assume you have the ALTAID05 EPROM and 512K of RAM installed.

1. **RESET THE 8080:** Press the RESET key (S11) for two seconds. This does a brief self-test, and the right four address LEDs (A3-A0) will count down from 1111 to 0000 as RAM is initialized. It then enters MONITOR mode. See https://youtu.be/4og5nvrUBLQ for a brief video showing the self-test operation.
2. **MONITOR MODE:** The Address LEDs show the last address set. The Data LEDs show the contents of that address. The default address is 1111 1111 0100 0111 (FF47 hex, 377 107 octal), which is the 'tick' counter; so the Data LEDs will be continuously counting up.
3. **SELECT MODE:** Press the MODE key (S9). The MODE LED will light, and the high half of the Address LEDs (A15-A8) will start flashing.
4. Now press the Data keys to set the high half of the address to any desired value. Each time a Data key is pressed, it toggles the corresponding bit in the selected group.
5. Press the **MODE** key again. The low half of the Address LEDs (A7-A0) will start flashing.
6. Press the Data keys to set the low half of the address to any desired value.
7. Press the **MODE** key again. Now the Data LEDs (D7-D0) will start flashing.
8. Press the Data keys to set the Data to any desired value. Note: addresses from 0-32K (0xxx xxxx xxxx xxxx) are in the EPROM, and so can be examined, but not changed.
9. Press **MODE** again. This returns to MONITOR mode. The MODE LED will turn off, the Address LEDs show your selected address, and the Data LEDs will show its contents.
10. Each successive **MODE** key cycles from Monitor, to Addr.High, to Addr.Low, to Data, and back to Monitor again.
11. From MONITOR mode, press the NEXT key (S10). This increments the address (adds 1), and shows its contents.
12. From MONITOR mode, press the Data 7 and NEXT keys. This decrements the address (subtracts 1), and shows its contents.
13. Holding down the NEXT key auto-repeats.
14. **RUN:** Once you have entered a program, and set a valid starting address, press the RUN key. The RUN LED will light, and the 8080 will jump to that address to execute the program at that location.

    While your program is running, the Address and Data LEDs will continue to show the contents of the last selected address. This is one way for your program to display results on the Front Panel.
15. Press **RESET** for 2 seconds to stop a running program, and return to the MONITOR mode.

SERIAL MONITOR CONNECTION: The Altaid 8800 can also be controlled with a PC or terminal serial port; TTL or RS-232. Connect the T (Transmit), R (Receive), C (Clear), 5VDC (+), and Ground (–) pins of P1 on the MIO card as shown below.

Note that TX and RX cross! Connect your TX to R, and RX to T. DTR is optional; if present, a falling edge on DTR will reset the 8080.



TTL serial port: If assembled for TTL serial, connect it the same as above. The Sparkfun 5v USB-serial cable makes this easy; it plugs right in (black=GND, red=+5V, orange=TX, yellow=RX, and green=DTR).

SERIAL MONITOR OPERATION:

First, configure your serial terminal, or a PC with a Terminal program (Hyperterm, Realterm, Teraterm, etc.) If your PC doesn't have a Terminal program, download and install one. These instructions assume Teraterm, as it is one of the easier ones to configure and use.

If your PC has no RS-232 ports, you will also need a USB-serial adapter. Most of these require a device driver, which you must also find, download, and install. If you use the Sparkfun adapter, Windows 7 and later versions should already have the needed driver installed.

Next, test your serial port by itself by shorting its RX to TX.

Use the Windows Device Manager to look for COM ports with the USB-Serial cable connected. An RS-232 port will probably be COM1 or COM2. A USB-serial adapter will be called "Serial USB" or similar, and have a name like COM3. Note the name of the COM port you will be using.

Start your Terminal program, and configure it as follows. In TeraTerm for example, click "Setup... Serial port..." and then set it as shown:

```
Port = COM3 (or the port your PC will use)
Speed = 9600 (19200 and 38400 also work)
Data = 8 bit
Parity = none
Stop bits = 1 bit
Flow control = none
Transmit delay = 10 msec/char, 50 msec/line
```

Close the Serial port window. Click "Setup... Terminal...", set as shown, and close when done.

```
New-line: Receive = CR, Transmit = CR
```

Now, whatever you type on your keyboard should be echoed to your screen. When that works, remove the TX-RX shorting jumper, and connect your serial port to the Altaid 8800.

The serial monitor defaults to 9600 baud. For other rates, press RESET and hold down the D4 key for 9600, D5 for 19200, D6 for 38400, or D7 for an alternate 38400 (faster, but may miss characters). Then release the RESET key.

Your Terminal should now be communicating with the Altaid! The Monitor will print its sign-on message. Ignore the "RAM DISK Chksum BAD" message; you haven't formatted the RAM disk yet, and don't have a backup battery to maintain its contents without power.

You will now see the MENU> prompt. Type '?' for a menu of monitor commands, or a command letter, then any optional numbers (in hexadecimal), then the ENTER key.

```
COLD-BOOT
Altaid 8800 v0.5 Itty Bitty Micro Co
RAM DISK Chksum=AD8B = BAD
MENU>?

HELP
B -BAUD 1,2,3,4
C -CP/M
D -Dump
E -Edit
F -Format RAM DISK
K -RAM DISK ChkSum
M -RAM/ROM Select
G -Go (Exec)
O -Output to port
I -Input to port
R -Read Sector
W -Write Sector
X -XMODEM R,S
T -ISR_TIMER
L -LED_PANEL
V -SET FP Addr
>
MENU>
```

Command examples: User input is underlined.

H or ?	Help command to print Menu
B n	set BAUD rate. 1=9600, 2=19.2K, 3=38.4K, 4=alternate 38.4K.
C	Cold boot (start) CP/M.
D xxxx yyyy	Dump (display) memory from xxxx to yyyy in hex and ASCII. Pauses after each screen, or ESC to exit.
E xxxx	Edit memory starting at xxxx.
ROM	Monitor shows memory bank(ROM)
aaaa : dd	Then shows address & its data.
aaaa : dd NN	Type new data NN. Monitor writes
aaaa : dd NN nn	it, then reads it to verify,
aa+1 : dd	then steps to next address. Type another new value NN, or ENTER or ESC to exit.
G xxxx	Go (execute) program at xxxx. Use RESET to exit a hung program.
F	Format RAM disk for CP/M.
FORMAT? Y	Type Y for Yes, anything else for no.
K	Checksum RAM disk contents.
M n	Memory bank select. 0 = ROM, 1-F = RAM.
O nn pp	Output nn to port pp.
I pp nn	Input from port pp. nn=value read.
R xxxx	Read sector xxxx from RAM disk. 0-0DFF is RAM, 0E00-0E4F is ROM.
W xxxx	Write sector xxxx to RAM disk.
X R	Xmodem Receive (PC to RAM disk).
X S	Xmodem Send (RAM disk to PC).
T	Toggle Timer on / off.
L	Toggle Front Panel on / off.
V aaaa	Set address to View on Front Panel.

CP/M OPERATING SYSTEM: CP/M (Control Program for Microcomputers) was created in 1974 by Gary Kildall of Digital Research Inc. It was the first commercial disk operating system for microcomputers, and became the industry standard for the 8080-Z80 family of CPUs. CP/M allowed the same programs to run on many different computers, and played a vital role in creating the personal computer industry.

The Altaid 8800 ROM contains the three essential programs needed to run CP/M:

1. The BIOS (Basic I/O System) has the hardware-specific code to talk to the screen, keyboard, serial ports, disk drives, and other hardware devices.
2. The BDOS (Basic Disk Operating System) is the file system. It provides a standard way to read/write data to disks and other devices.
3. The CCP (Console Command Processor) is used to give commands to CP/M itself. It runs when you aren't running any other programs (such as a game, word processor, BASIC etc.)

"Booting" CP/M loads these programs from ROM into RAM memory, and runs them.

To Run CP/M:

☐ Connect and configure a serial terminal.

☐ Press RESET to start the Altaid 8800 Monitor. If this is the first power-up, you will get a "RAM DISK Chksum = xxxx BAD" message.

☐ FORMAT the RAM disk with the "F" command, then "Y" for Yes. This creates a blank A: disk and a valid checksum for it. The RAM disk is 456K (with a 512K RAM at U2). Its contents are not affected by RESET, and it will retain data without power if you connect a 3v lithium cell or three 1.5v cells in series for battery backup: **+** to VIN (J2 pin 20), **–** to GND (J2 pin 10).

☐ Turn the timer off with the "T" command. This reduces flickering of the LEDs, and the RAM disk may have problems if the timer is on.

☐ Start CP/M with the "C" command. CP/M will display its sign-on message and A> prompt. "61K" is the TPA (Transient Program Area), where programs load and run. That may seem small, but it's enough for the vast majority of CP/M programs!

☐ Type DIR<enter> for a DIRectory of files. PIP, XM, and MONITOR are pre-loaded and ready to use. PIP (Peripheral Interchange Program) copies files within CP/M), XM (XMODEM) transfers files to/from your PC, and MONITOR exits to the Monitor. You also have all the built-in commands on page 1 of the CP/M Quick Reference; DIR, ERA, REN, SAVE, TYPE, USER.

Loading CP/M Programs from the Web:

☐ For example, let's load ZORK, the famous Infocom Adventure game! First, create a CPM working directory on your PC. Download this file of CP/M disk images into it, and unZIP it: http://sunrise-ev.com/photos/z80/SD-CARD.zip

☐ Download the CP/M Disk Explorer program at http://sunrise-ev.com/photos/z80/CPMDEv13.zip Unzip & save it in your CPM working directory. Re-name the .XEX file to .EXE, and run the program. See the CPM-DI~1.PDF file for help.

☐ Start the CP/M Disk Explorer program. Double-click DISK-B.BIN in the lower left window. The files in it will appear in the right window. Drag ZORK1.COM and ZORK1.DAT back into the lower left window. Now they are available in your CPM working directory.

☐ Perform the following steps to copy them to your Altair 8800 A: disk. Type the underlined parts, followed by the <Enter> key:

```
MENU>c
Altaid 8800 – 61K CP/M 2.2

WBOOT

a>dir
A: PIP   COM : XM    COM : MONITOR COM
a>xm zork1.com /R /C
       (after the <Enter> key, click
       File..Transfer..XMODEM..Send..
       and select the ZORK1.COM file)
File created
Receive
OK
Received 68 blocks
WBOOT

a>xm zork1.dat /R /C
       (after the <Enter> key, click
       File..Transfer..XMODEM..Send..
       and select the ZORK1.DAT file)
File created
Receive
OK
Received 664 blocks
WBOOT

a>zork1  (now run the game!)
ZORK I: The Great Underground Empire

West of House
You are standing in an open field
west of a white house, with a boarded
front door. There is a Mailbox here.
       (You're on your own from here.
        Don't get eaten by a grue!)
```

Finding CP/M Software

You can download CP/M programs from the web, or from other vintage software sources. Be sure it is written for the 8080, and not the Z80 or some other CPU. There are CP/M versions for many different computers and CPUs!

There is a VAST amount of CP/M software; assemblers, languages, games, editors, word processors, spreadsheets, utilities, and more. But it's a bit of an archeological "dig" to find them on the web.

One source is the Walnut Creek CP/M CDROM archive, which contains over 19,000 CP/M programs. It can be downloaded from a number of sources. Here is one example:

https://archive.org/details/cdrom-1994-11-walnutcreek-cpm

More “gold” can be found in the various CP/M Users Group archives. They distributed software on floppy disks, some of which have survived on the web (in various places, in various formats). Some of them can be found at the unofficial CP/M web site:

http://www.cpm.z80.de/

And of course, you can get software from various vintage floppy disks. It can be a challenge to find a working system that can read the disks. But if you do, you can use XMODEM to transfer the files to the Altaid 8800 with a serial interface.

Console Configuration

CP/M calls its keyboard and screen the "console". It was originally a teletype, but people quickly switched to one of many different "terminals" (Lear Siegler ADM-3, Heath H19, DEC VT-52 or VT-100, etc.) Today, your terminal is likely to be a PC running a "Terminal" program that emulates one or more of these original terminals.

Each Terminal has its own idiosyncratic set of control codes and ESC sequences for its keyboard and screen. CP/M programs are often configured to work with the command set of a particular Terminal. If you have a different Terminal, the program may run; but keyboard functions, cursor positioning, graphic characters etc. will be wrong.

Most Terminal program have settings to emulate different terminal command sets. But some of these may not be fully implemented or buggy. Experiment to see what works the best.

If your Terminal program can't emulate the right command set, most CP/M programs can be re-configured to use something your Terminal program **does** support. VT-100 is a good choice as it is supported by just about all Terminal programs. Look for a version of the CP/M program already configured for it. Or, find instructions or a CONFIG utility to re-configure the program you've got for a VT-100.

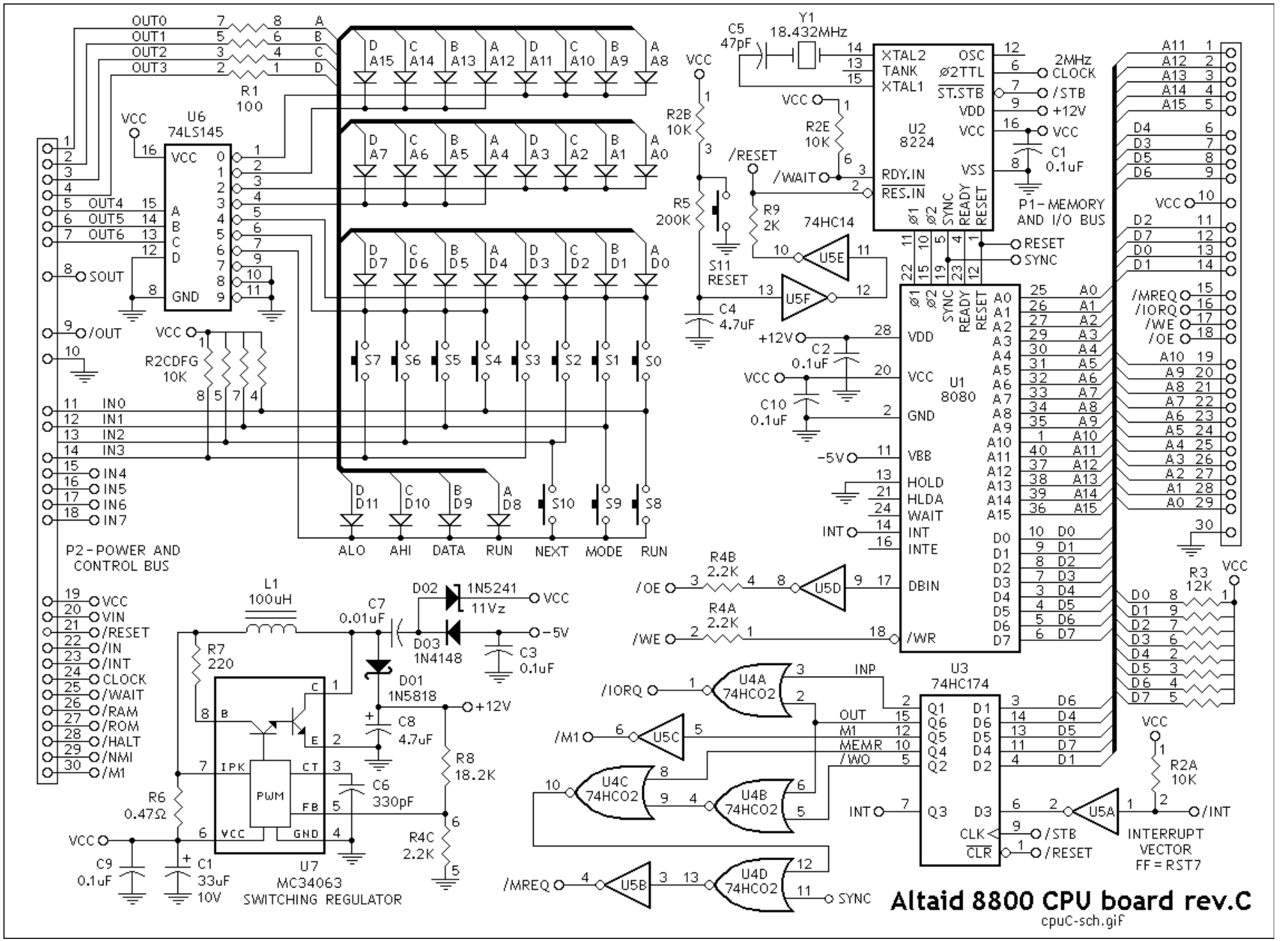
In Case of Difficulty

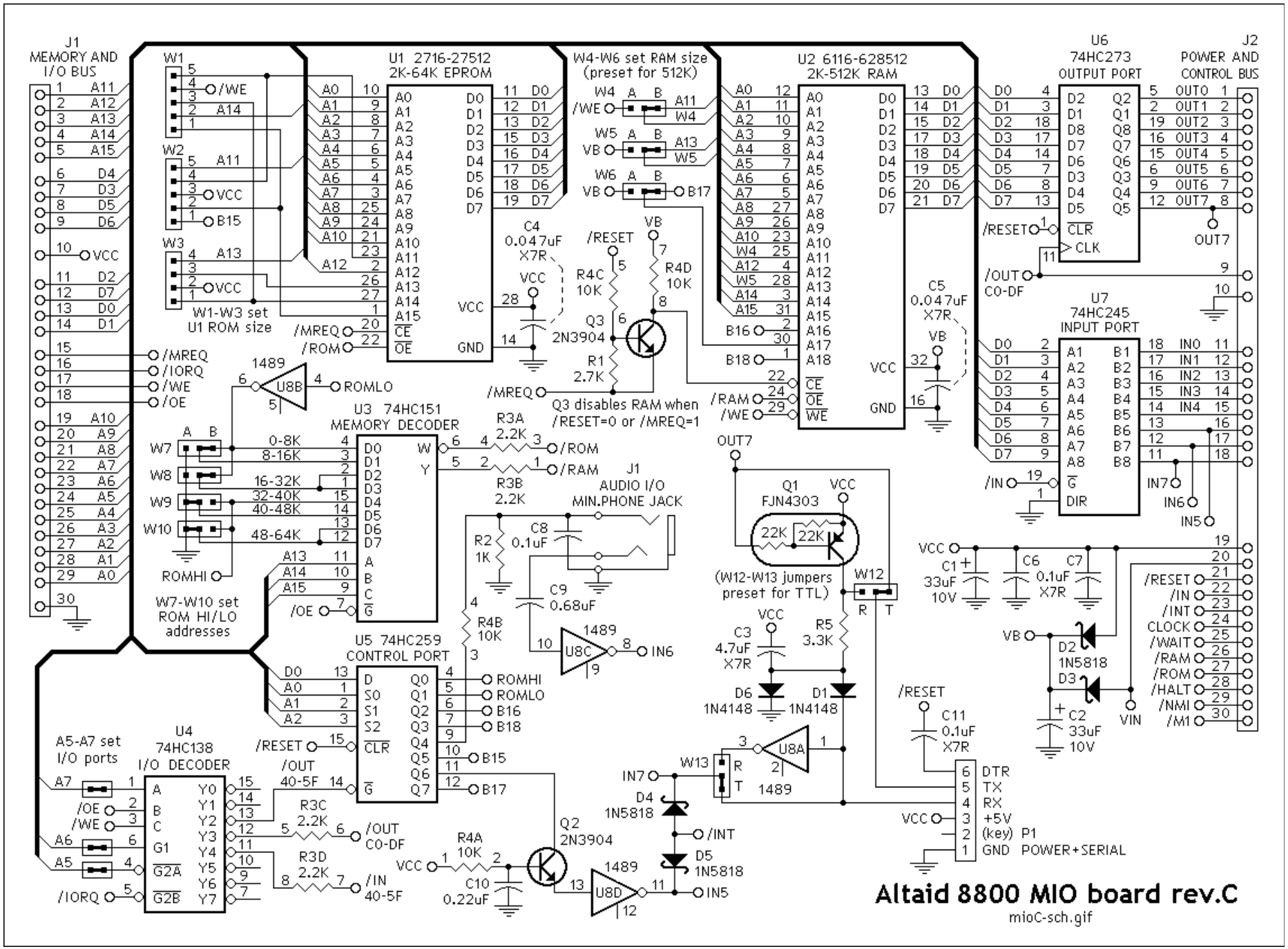
The most common sources of trouble are:

- ☐ Bad soldering. Check for solder bridges, unsoldered connections, or cold solder joints.
- ☐ Parts in the wrong places, or backwards.
- ☐ Plugging the CPU and MIO cards together backwards. (Don't panic; it won't kill them.)
- ☐ If you unplug the two memory chips on the MIO card, R3 on the CPU card will pull the data bus high (to FFh). This will continuously execute RST 7 instructions, which a) push the program counter onto the stack, b) decrement the stack pointer by two, and c) jump to address 0038h. It thus exercises all the data, address, and control lines. Use an oscilloscope to look for any lines that are stuck high, stuck low, shorted, or floating.

Forward into the past...

Hopefully, this is enough to open the door into the rich world of vintage computing. For more details, see the "CP/M quick reference" pages at the end of this manual.


OUT0
OUT1
OUT2
OUT3
R1
100
U6
74LS145
SOUT
/OUT
R2CDFG
10K
IN0
IN1
IN2
IN3
IN4
IN5
IN6
IN7
P2-POWER AND
CONTROL BUS
VCC
VIN
/RESET
/IN
/INT
CLOCK
/WAIT
/RAM
/ROM
/HALT
/NMI
/M1
ALO
AHI
DATA
RUN
NEXT
MODE
RUN
L1
100uH
C7
0.01uF
D02
1N5241
11Vz
D03
1N4148
D01
1N5818
C3
0.1uF
-5V
+12V
R7
220
C8
4.7uF
R8
18.2K
R6
0.47Ω
C6
330pF
R4C
2.2K
C9
0.1uF
C1
33uF
10V
U7
MC34063
SWITCHING REGULATOR
C5
47pF
Y1
18.432MHz
R2B
10K
R2E
10K
R5
200K
S11
RESET
/RESET
/WAIT
R9
2K
74HC14
U5E
U5F
C4
4.7uF
U2
8224
XTAL2
TANK
XTAL1
OSC
ø2TTL
ST.STB
VDD
VCC
RDY.IN
RES.IN
VSS
SYNC
READY
RESET
2MHz
CLOCK
/STB
+12V
C1
0.1uF
P1-MEMORY
AND I/O BUS
RESET
SYNC
U1
8080
+12V
C2
0.1uF
C10
0.1uF
-5V
HOLD
HLDA
WAIT
INT
INTE
DBIN
/WR
R4B
2.2K
R4A
2.2K
/OE
/WE
U5D
U3
74HC174
U4A
74HCO2
U4B
74HCO2
U4C
74HCO2
U4D
74HCO2
U5A
U5B
U5C
/IORQ
/M1
/MREQ
INP
OUT
M1
MEMR
/WO
INT
CLK
CLR
/STB
/RESET
/INT
INTERRUPT
VECTOR
FF = RST7
R3
12K
R2A
10K
SYNC
Altaid 8800 CPU board rev.C
cpuC-sch.gif


J1
MEMORY AND
I/O BUS
W1
W2
W3
W1-W3 set
U1 ROM size
U1 2716-27512
2K-64K EPROM
W4-W6 set RAM size
(preset for 512K)
U2 6116-628512
2K-512K RAM
U6
74HC273
OUTPUT PORT
J2
POWER AND
CONTROL BUS
U7
74HC245
INPUT PORT
C4
0.047uF
X7R
C5
0.047uF
X7R
Q3
2N3904
R1
2.7K
R4C
10K
R4D
10K
Q3 disables RAM when
/RESET=0 or /MREQ=1
U3 74HC151
MEMORY DECODER
W7-W10 set
ROM HI/LO
addresses
R3A
2.2K
R3B
2.2K
J1
AUDIO I/O
MIN.PHONE JACK
Q1
FJN4303
(W12-W13 jumpers
preset for TTL)
U5 74HC259
CONTROL PORT
R2
1K
C8
0.1uF
C9
0.68uF
R4B
10K
U4
74HC138
I/O DECODER
A5-A7 set
I/O ports
R3C
2.2K
R3D
2.2K
R4A
10K
C10
0.22uF
Q2
2N3904
C3
4.7uF
X7R
R5
3.3K
D6
1N4148
D1
1N4148
D4
1N5818
D5
1N5818
D2
1N5818
D3
C11
0.1uF
X7R
C1
33uF
10V
C6
0.1uF
X7R
C7
C2
33uF
10V
DTR
TX
RX
+5V
(key) P1
GND POWER+SERIAL
Altaid 8800 MIO board rev.C
mioC-sch.gif

Digital Research CP/M 2.2 Quick Reference – page 1

CP/M is in control when you see the A> prompt. "A" is the current disk drive. Type a Command, any optional filename(s) to use, and the ENTER or RETURN key. Use upper or lower case (CP/M will convert lowercase to uppercase).

CP/M first checks for a built-in command (DIR, ERA, REN, SAVE, TYPE, or USER), then for a .COM file on the current disk (PIP, STAT etc.) If found, it runs it (you don't need to type the .COM part). If not found, it repeats the command with "?" after it as an error message. When the command finishes, it returns to the CP/M A> prompt, ready for the next command.

Each file has a unique FILENAME, with three parts:

d:	disk drive (A-P), followed by a colon (:)
name	then 1-8 alphanumeric characters, including ! @ # $ % + - (or) but not < > . , : ; = ? * [or]
.extent	a period (.) and 0-3 more alphanumeric characters for the file type. Examples:
.ASM	assembly language source file
.BAS	BASIC program file
.COM	executable program file
.DAT	data file
.DOC	formatted document file
.HEX	Intel HEX format file
.SUB	submit (batch) file
.TXT	simple text file

Filenames can be abbreviated, or use wild cards. "?" matches any single character. "*" matches any string of characters.

D:	for any files on the D: disk drive
FILE	for FILE on the current disk drive
D:FILE	for FILE on the D: drive
FILE.TXT	for FILE.TXT on the current drive
D:FILE.TXT	for FILE.TXT on the D: drive
FILE?	any 5-character name starting with FILE
FILE.*	any file named FILE. with any extent
F*.TXT	any .TXT file starting with F
.	matches everything

CONVENTIONS

{...} Text in curly brackets are optional filenames. (Do not type the curly brackets).

[...] Text in square brackets are "switches" to enable or disable various options. (You must type the square brackets). For example, PIP uses [v] for verify.

CONTROL CODES perform various "control" actions. "^" means hold down the CTRL key, then type the letter. Some control codes may have their own key.

^C	Warm boot; cancel command and re-start CP/M
^G	Ring the console's bell
^H	(Backspace). Backup and delete last character
^I	(Tab). Move right to next tab stop
^J	(Line Feed). Move down a line
^M	(Return or Enter). Process current command, and wait for the next command on a new line
^P	Toggle printer on/off (to duplicate console output).
^Q	Resume a long listing that was paused with ^S
^S	Stop (pause) a long listing
^X	Cancel; backup and delete all characters on line
^Z	Marks end of a file

BUILT-IN commands are part of CP/M (called the CCP), and don't take up any file space on disks. Filenames are optional, and can include wildcards.

B:	change current disk to B
DIR	directory of all files on current disk
DIR B:	all files on B: disk
DIR B:*.COM	all .COM files on a disk
ERA file(s)	erase file (or files) on a disk
ERA *.*	erase <u>all</u> files on a disk (caution!)
REN new=old	rename old file to the new name
SAVE n file	save "n" 256-byte blocks of memory starting at 0100h in file
TYPE file	type (display on console) file
USER n	set user number (n=0 to 15)

TRANSIENT Commands are .COM program files on a disk. They can be anything (games, programming languages, word processors, spreadsheets, etc.) CP/M comes with a simple but powerful set, able to write programs, or even rebuild CP/M itself to run on other computers.

ASM - Assembles an 8080 source .ASM file into .HEX and/or .LST files. Use DDT or LOAD to convert the .HEX file into an executable .COM file.

ASM FILE	where FILE is named FILE.ASM and all files are on the same disk as ASM.
ASM FILE.SHL	same, but use disks other than A:
S- -	is source disk for FILE.ASM
- H -	is destination disk for FILE.HEX (or Z for none)
- - L	is destination for listing FILE.LST (or Z for none)

Digital Research CP/M 2.2 Quick Reference – page 2

DDT - Dynamic Debugging Tool. 8080 monitor program to examine and change memory, disk, and CPU registers; and run, trace, or single-step programs. All values are in HEX. Type ^C to exit.

DDT	(no filename) wait for commands
DDT filename	load file, and wait for commands
Astart	Assemble at 'start' address
Dstart{,end}	Dump memory in hex and ASCII
Fstart,end,bb	Fill memory start to end with bb
G{start}{,end}	Go run program at PC, or at optional start address. Optional breakpoint at end.
Hn1,n2	Hex math; show n1+n2, n1–n2
Ifilename	Insert filename in FCB (for R command)
L{start}{,end}	List (disassemble) memory into assembler from start to end
Mstart,end,to	Move start-end block to 'to' addr
R{offset}	Read file (in FCB). No offset over-writes CP/M at 0! Use 0100 etc.
Sstart	Set (change) memory from "start", 1 byte at a time. Type any non-hex character to end.
T{n}	Trace n instructions (show all steps)
U{n}	Untrace n instructions (only show last step)
X	eXamine (show) all CPU registers.
X{register}	eXamine/change selected register
1-bit flags	Carry, Zero, Minus, Even, Interdigit
1-byte regs	A (accumulator)
2-byte regs	B, D, H, P, S (2 bytes)

DUMP - Displays the contents of a file in HEX format. Each line shows the address and next 16 bytes of the file, and continues until the end of the file. DUMP is also supplied in .ASM form as an example program to learn how to write, assemble, and load CP/M programs.

DUMP D:FILE.TXT	displays FILE.TXT on D: ^S to pause, ^C to exit

ED - Text editor. More like a typewriter than a screen editor, ED commands move an invisible character pointer "p" to perform actions at its location. "n" is the number of characters/lines (default is 1), and can be + – or "#" (for all). Separate multiple commands on a line with ^Z.

ED is crude; but powerful. It edits files larger than memory by loading, editing, and saving it a block at a time. It can edit anything, and accepts .SUB commands (with XSUB) for automatic editing.

ED D:FILE.TXT opens FILE.TXT on D: for editing. If FILE.TXT is not found, create it.

ED commands:

load and save files (read disk –> buffer –> write disk)

nA	Append (read) n lines into buffer (#A for all)
nW	Write n lines of buffer to disk (#W for all)
H	save edited file, clear buffer for more edits
nX	Write n lines to X.SUB file (copy)
R	Read X.SUB file (paste)
E	Exit; save edited file, rename old file .BAK
O	abandon edits, and restart with Original file
Q	Quit without saving

moving the "p" pointer

B	move to Beginning, or –B move to end
n:	move to line "n"
nL	move "n" lines
nC	move "n" characters
nF text	Find nth occurrence of "text" in buffer
nN text	Find nth occurrence of "text" in entire file

displaying and editing text

nT	Type "n" lines (T or 1T is current "p" line)
I text^Z	Insert "text" at "p", and move "p" to end
nD	Delete "n" characters at "p" (–before +after)
nK	Kill (delete) n lines at "p" (–before +after)
nS old^Z new^Z	Substitute (replace) old with new "n" times (–before "p", +after "p", # all)

other commands

nM command(s) ^Z	Macro; repeat command(s) "n" times. 0 or 1 repeats until end of file.
+U	convert to Uppercase, –U don't convert
V	line numbers (+V display, –V don't display)
0V	shows free/total space in buffer

LOAD - Reads an Intel-format .HEX file and writes an executable .COM file. The .HEX file is usually produced by ASM, and set up to run at 0100h under CP/M. The new .COM file can be run by typing its name at the CP/M prompt, like PIP or any other program file.

LOAD B:FILE	load FILE.HEX from B: disk, and write FILE.COM to the same disk

MOVCPM - Move CP/M to use a different memory size from 20-64K. (Not used with Z80MC or Altaid 8800.)

MOVCPM n	create "n" kilobyte CP/M system
MOVCPM *	(* or blank) create max size CP/M
MOVCPM * *	create max size CP/M, and leave it in memory (ready to save with SYSGEN)

Digital Research CP/M 2.2 Quick Reference – page 3

PIP - Peripheral Interchange Program (transfers files between disks and peripherals). Format is PIP destination=source1 {,source2...}[switches]

(Note: Destination is FIRST; the reverse of the DOS COPY command! Think of PIP as "LET A:=source").

PIP	(no filenames) wait for commands
PIP B:=FILE	copy FILE from current disk to B:
PIP B:=A*.TXT	copy all .TXT files from A: to B:
PIP CON:=FILE	copy (display) FILE on console
PIP NEW=OLD	copy OLD & name the copy NEW
PIP B:=*.COM[V]	copy and Verify all .COM files from current drive to B:
PIP LETTER=HEAD,BODY[T8P]	Copy HEAD & BODY to LETTER, expand Tabs to 8 spaces, and start a new page every 60 lines

The PIP [...] switches are

Dn	Delete characters after column n
E	Echo transfers to console
F	Form Feed (page ejects) removed
Gn	Get file from from User area n
H	Hex data transfer
I	Ignore :00 (null record in HEX file)
L	convert to Lower case
N{2}	Number lines. N2 for "line#:<Tab>"
O	Object (binary) file; ignore ^Z
Pn	new Page every n lines(default60)
Qabc^Z	Quit copying at string "abc"
R	Read hidden .SYS (system) files
Sabc^Z	Start copying at string "abc"
Tn	expand Tabs to next nth character
U	convert to Upper case
V	Verify that copy is written correctly
W	Write over read-only destination file
Z	Zero the high bit (bit 7 or parity)

STAT - Status of disk drives, files, and I/O devices.

STAT filename(s) {$options}
where options are: $R/O makes file read-only
$R/W makes file read-write
$DIR shows file in DIRectory
$SYS hides from DIRectory

STAT	(no filenames) lists free space and R/W status of all disks
STAT B:	free space and status of B: disk
STAT FILE.TXT	size of the file FILE.TXT
STAT *.TXT	sorted list & size of all .TXT files
STAT B:*.*	sorted list, size, SYS (hidden), and R/O (read-only) status of all files on B:
STAT *.COM $R/O	set all .COM files to read-only
STAT DSK:	statistics of all disk drives
STAT USR:	list all user numbers with active files
STAT VAL:	summary of disk and I/O names:
	A: to P: disk drive names
	CON: console
	RDR: reader (serial input)
	LST: list (printer)
	PUN: punch (serial output)

SUBMIT - Executes a list of commands automatically (like a DOS .BAT file). Commands are listed in a .SUB file, which must be on A:. Prepare this file with a text editor. Use $1 $2 ... for as placeholders for filenames, which get filled in by the SUBMIT command line. For example, suppose the MAKE.SUB file contains

```
ASM $1
LOAD $1
```

Type SUBMIT MAKE DUMP to run this .SUB file. SUBMIT will run ASM and LOAD, replacing each $1 with DUMP from the SUBMIT command line.

SYSGEN - System Generation. Saves a new copy of CP/M in a disk's "boot" tracks. (Not used with Z80MC or Altaid 8800; their RAM disks have no "boot" tracks.)

SYSGEN	starts the program
Source Drive (RETURN to skip)	type drive letter to read boot tracks from, or RETURN if they are already in memory from MOVCPM
Destination on x: type RETURN	finish and exit

XSUB - extend SUBMIT. When XSUB is the first line in a .SUB file, SUBMIT will read program inputs (as well as file names) from a .SUB file and feed them to the programs executed. Example: if DDT.SUB contains:

```
XSUB
DDT $1
D0100,01FF
G0
```

Type SUBMIT DDT DUMP.COM to run this .SUB file. SUBMIT will run DDT, replacing $1 with DUMP.COM, then tell DDT to Display the first FF bytes of it (i.e. from 0100-01FF), then exit.

Note: Each line in a .SUB file ends with <CR>, so XSUB can only feed command lines that end in <CR>. Thus, XSUB cannot feed single-character commands with no <CR> to other programs.

THE S-100 MICROCOMPUTER