D-X Designs Pty Ltd
P112 CPU Board
Assembly Instructions

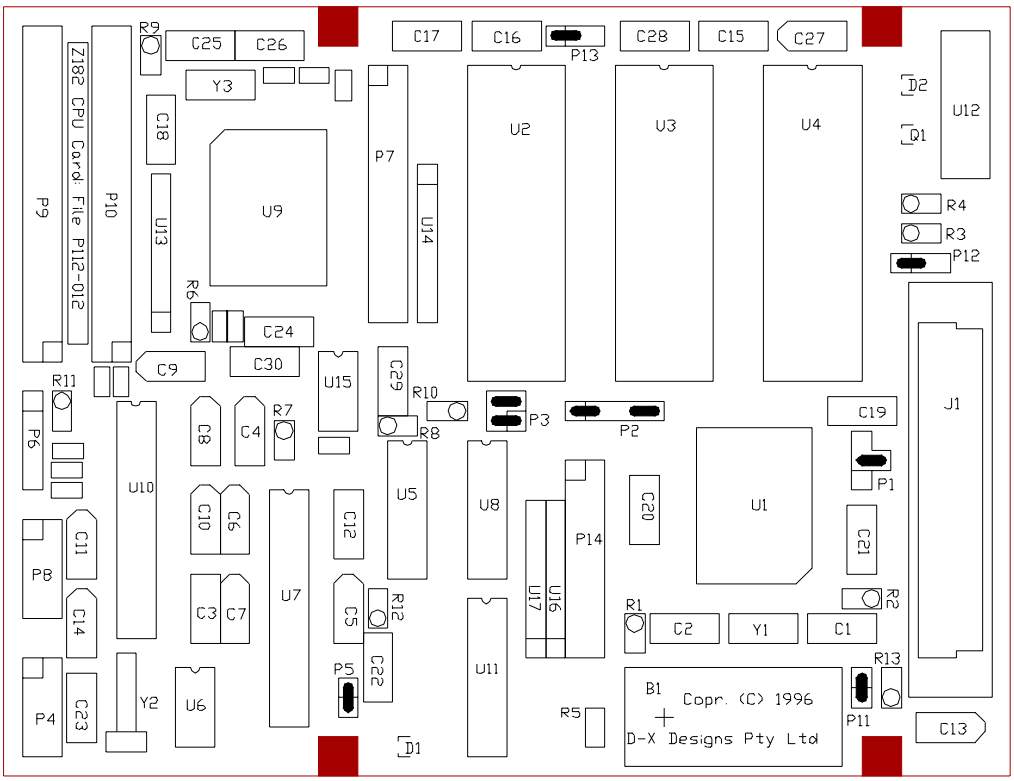
1. INTRODUCTION

This document provides general guidance on getting the board built and running. Further information is filed on the disk.

2. ASSEMBLY

Remember to observe proper anti-static precautions when assembling this board.

The board overlay is shown below. The surface-mount parts, U1, U9, D1, D2, and Q1 have been pre-installed. In the drawing, several component references have been moved from their locations on the board to inside their component outlines, for clarity. On the board, they are placed so as to be legible with the components installed.



Fit the parts in accordance with the list below. Generally, the IC's and I/O connectors have Pin 1 at the top left, with the exception of P9 and P10 (the disk-drive connectors), which have Pin 1 at bottom right (to conform with the pin arrangements on the drives). The resistor-packs U13, U16 and U17 also have Pin-1 at the bottom end.

When fitting resistors, note that they all stand up vertically, with the upper lead bent over. Fit the resistor body in the hole marked with a surrounding circle, and put the bent-over lead through the plain hole. This improves testability: the active signal is accessible at the top of the resistor.

Tantalum electrolytics are fitted with the +ve pin to the tapered end of the symbol.

The Vpp voltage converter U12 will benefit from a spot of suitable adhesive under it, as will the crystal Y2.

The connector P1 is a "special", made up from a 3-pin header strip and a single header pin. These may be held together by a shorting clip while soldering.

One of the holes for P5 is a little under-size, but the pin does go in OK.

The 11 SMD resistor sites (which carry no references) are **not** populated: they are required with some alternative versions of the SMC chip, U9.

2.1. Cables

All the basic IO connections are at the left edge of the PCB. You will need to make up the following cables to run your board:

2.1.1. Power Cable

Connects to P6. Reading from top, the pins are
GND
Vdd
Reset
Vdd
GND
The Reset pin may be left unconnected (it has a pull-up). To force a reset, pull this line low.

Nominal power consumption (PCB only) is 150mA at 5V.

2.1.2. Terminal Cable

RS-232 terminals may be connected to P4 and P8 (P4 being the primary terminal). A 10-way ribbon cable connects via a standard IDC receptacle to the board. Core 10 is clipped off, and the rest are crimped to a standard IDC DB-9 plug (Pin-1 to Pin-1). This generates the same pin-out as a PC-AT.

2.1.3. Disk Drive Cable

This is wired as for a PC-AT, ie a direct 34-way connection from P9 or P10 to the first drive, then cores 10-16 reversed in the path to the 2nd drive. In P9, the first drive will be Drive 0, and the second Drive 1. From P10, they are Drives 3 and 2 similarly (note the reversed assignment). *Note:* As of this writing, I have got Drives 0, 1 and 3 working. The Drive-2 select (Pin 98 of U9) shares with other functions, & I haven't yet managed to get it to select the drive!

2.1.4. Parallel Port Cable

A standard 26-way IDC receptacle fits the board. Clip off core 26 of the cable, and run the others to a DB-25 socket (Pin-1 to Pin-1). This generates a standard PC-type printer interface.

2.2. Parts List

22pF ceramic	4	C1	C2	C25	C26		
120pF ceramic	1	C12					
100nF bypass	14	C3	C15	C16	C17	C18	
		C19	C20	C21	C22	C23	
		C24	C28	C29	C30		
1uF, 16V tant.	9	C4	C5	C6	C7	C8	
		C9	C10	C11	C27		
22uF, 10V tant.	2	C13	C14				
7x220 respack	1	U13					
7x2k2 respack	1	U14					
7x47k respack	2	U16	U17				
4k7 resistor	3	R11	R3	R4			
10k resistor	3	R5	R7	R10			
27k resistor	1	R9					
47k resistor	6	R1	R2	R6	R8	R12	R13
2-pin Hdr	2	P5	P11				
3-pin Hdr	3	P1	P12	P13			
5-pin Hdr	2	P2	P6				
2x2-pin Hdr	1	P3					
2x5-pin Hdr	2	P4	P8				
2x10-pin Hdr	1	P14					
2x13-pin Hdr	1	P7					
2x17-pin Hdr	2	P9	P10				

```
 PCB jumper       10
 DIN48 skt.       1          J1          (3x16-way DIN skt)
 32DIP skt.       3          U2    U3    U4
 32kHz crystal    1          Y2          (RTC crystal)
 16MHz crystal    1          Y1          (CPU clock)
 24MHz crystal    1          Y3          (IO chip clock)
*BAR43C diode     1          D1          (Schottky)
*BAR43  diode     1          D2          (Schottky)
*BCW71  trans.    1          Q1
 P28F256A-150     1          U4          Intel (Flash ROM)
 74HCT00          1          U5
 74ACT02          1          U8
 74ACT139         1          U11
 HM62256BLP-7     2          U2    U3    Hitachi (Static RAM, 32kB)
 DS1202           1          U6          Dallas (RTC chip)
*FDC37C665IR      1          U9          SMC (IO combo)
 LT1133           2          U7    U10   Linear Tech. (RS232 Tx/Rx)
 NMF0512S         1          U12         Newport Instruments (5/12V conv.)
 TL7705ACP        1          U15         Texas (Reset genr.)
*Z8018216FSC      1          U1          Zilog (CPU chip)
 3V battery       1          B1          (Lithium, 1/2AA)
```

Parts prefixed "*" are pre-installed on the board.

3. SETTING UP

With the board assembled, basic tests may begin with the "obvious" checks for Vdd-Ground shorts, etc. Install the two SRAM chips in U2 and U3, and the flash ROM in U4.

3.1. Jumper Settings

The positions required for the PCB jumpers are shown in the drawing above, for the 64kB RAM supplied. See the file HARDWARE.TXT for settings for other RAM sizes. Initially, it is recommended that jumper P11 be left off, so isolating the battery. This jumper is only required when the real-time clock chip is in use. **Caution:** Lithium batteries have a low internal impedance. Do not place the populated board on a metallic surface, in case you short the solder pads on the bottom side. We recommend you put an insulating sheet under the battery area, to guard against accidents. Please also read the safety leaflet packed with the battery.

3.2. Terminal Settings

The default terminal set-up is 9600bps, 8 bits, no parity, 1 stop. The primary terminal ("console" to CP/M) connects to P4. Use a null-modem cable, or short pins 1,6 and 7 of P4 together. This is necessary to force the modem control lines active, without which the serial port will not operate.

3.3. Disk Drives

The board has run a variety of PC-AT type 3" drives, "out of the box". Specifically, the drives should be jumpered with DS1 active (as is normal in a PC-AT). There appears to be little standardisation regarding the location of the connectors on disk drives: the location of P9 will give a direct connection with Teac drives, but is awkward with Panasonic, for example.

3.4. Powering Up

The board should draw about 150mA from 5V. With a terminal connected, a message such as:

```
SMC IO chip identified (FDC37C665IR): configuring
CPU clock: 16.0MHz
RAM available: 64kB. From 40000 to 4FFFF

Z80 Series ROM-Resident Debugger V1.04: D-X Designs Pty Ltd  1996
Special P112 version: interrupts supported
Type "?" for help
```

=

should appear. You can type "?" for a list of commands, and experiment with the ROM-resident debugger (see below). At this time, only the upper 32kB of RAM is accessible: the lower 32kB is overlaid by the ROM.

With a 3" drive connected as Drive 0, and the supplied disk loaded, you can type "S" <return> at any Debug prompt to boot the system. The sign-on sequence is:

```
CCP+ Ver. 2.2
A>
```

The shareware OS DOS-Plus is now running, and offers a CP/M "look-alike" interface. Note that the shareware equivalent of DRI's "PIP" is called "PPIP". The BIOS supplied only supports two drives, "A" and "B". DOS-Plus was written by C. B. Falconer: see his copyright notice on the disk. **I have not charged you any money for the supply of DOS-Plus, or any other software on this disk or ROM.**

4. DEBUGGER

This section gives a quick guide to using the ROM debugger. Note that this code operates differently to, for example, ZSID.

4.1. Help Function

Entering "?" at a command prompt will display the help screen:

```
   C O M M A N D   H E L P

Separate all fields by one or more blanks

Commands Available:
: <Intel hex.>    Input line of Intel hex-format data
?                 Display this help
B [addr [count]]  Set/clear breakpoint
D addr [length]   Display/change memory
G [addr]          Go run program (from addr)
I port            Input & display byte from port
N [count]         Run (next N) instructions
O port data       Output data byte at port
R [register name] Display/change user registers
S [drive] [type]  Boot system from drive
                  <drive> = 0..3
                  <type>  = "3" or "5" (size)
```

The input-line editor includes support for Tab (tabstops are fixed at every 8 columns) and Backspace, which is destructive, ie it echos backspace - space - backspace to the terminal.

Do not forget to enter at least one space after the command letter: else the first command argument will be skipped. The only exception to the spaces-between-fields rule is the Intel hex-format input: this follows the Intel conventions, and must have no spaces between fields.

4.2. Set Breakpoint

A single software breakpoint may be set, at any editable location, by entering

```
B addr
```

The break is implemented by jamming a RST 38H instruction in that location, cancelling any pre-existent breakpoint. Unlike (for example) MSDOS DEBUG, breakpoints are persistent: they are not automatically cleared when the test-program hits them. Rather, they are replaced in memory when the test program is re-run, so they remain in force. To clear the breakpoint, enter "B" with no arguments.

If a count value is entered after the address, the test program will not halt until the breakpoint has been reached [count] times.

Caution: Breakpoints should only be set on the first op-code byte of a machine instruction. Setting them elsewhere will have undefined consequences.

4.3. Display/Set Memory

The command format is

```
D addr [length]
```

If a length is given, the indicated number of bytes is rounded up to the next multiple of 16, and the memory contents are displayed, 16 bytes to a lines, thus:

```
=D 500 27
0500 69 6E 20 49 6E 74 65 6C 2D 68 65 78 20 66 69 6C >in Intel-hex fil<
0510 65 3A 20 6C 61 73 74 20 62 61 64 20 6C 69 6E 65 >e: last bad line<
0520 3A 20 00 20 20 20 00 CD 36 02 DD 21 FB 10 3E E0 >: .   ..6..!..>.<
=
```

If only an address is entered, the current contents of that byte are displayed, and a new value may be entered. If a blank line is entered, the value remains unchanged, and the next byte is similarly displayed. Enter "Q" to exit from this function.

```
=D 9500
9500 00 17
9501 00
9502 00 Q
=
```

If "D" is entered alone on a line, the missing address defaults to zero. Since altering address zero is not permitted, the message

```
Cannot edit memory in this area
```

is displayed.

4.4. Go (Run Program)

The command is

```
G [addr]
```

Any set breakpoint is activated, and control passes to the test program. Registers will be loaded with the values shown by a "Display/Set Registers" command.

NB When the debugger starts up, all test-program registers, including PC and SP, default to zero. It is the user's responsibility to load meaningful values in these registers, before executing code. It is legal to start with SP=0: the stack will grow downward from FFFF.

4.5. Trace instructions

The "N" ("next") command permits program instructions to be traced through automatically. Entering "N" alone will cause one instruction to be executed, before control returns to the debugger. If a count is entered, that number of instructions will be executed before the debugger regains control. After each instruction is executed, the resulting register-values are displayed (as for the "Display/Set Registers" command below).

Caution: Tracing under the "N" command runs the target program much slower than real-time execution. There are also significant lengths of interrupt-disabled code, and the execution environment constantly switches between the user program and the debugger (which allocates its own stack space). This command should not be used to test time-critical code segments.

The TRACE function temporarily stores an EI instruction at the location *before* the instruction being traced. This location must be writeable, or the command will be refused. For example, you cannot TRACE an instruction at location 8000H, since the preceding location, 7FFFH, is in ROM, and the EI cannot be stored.

4.6. Display/Set Registers

If this command is entered with no parameters, the current register values are displayed, as follows:

```
=R
A  F  B  C  D  E  H  L  A' F' B' C' D' E' H' L'  IX   IY   PC   SP  <Instr.>
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000 0000 0000 0000 00 00 00
```

The 3 bytes at & following the PC address are displayed, regardless of the actual length of the instruction at that address.

A single parameter is allowed, being a register name as in the above example. Similarly to the display/set memory command, that register is displayed, and may be altered. The next register (in the order above) will then be displayed in its turn. Again, the letter "Q" entered alone, will quit before all registers have been displayed.

```
=R C'
C' 00 05
D' 00
E' 00 07
H' 00
L' 00
IX 0000 446
IY 0000 Q
=
```

4.7. Loading Files

The debugger can read and store files in Intel-hex format. This facility is limited to storing in RAM: the debugger cannot program the flash ROM.

Typically, the debugger is connected to a PC running a suitable terminal-emulator program, and the "text upload" facility of that program is used to transfer the file. There is no facility for altering the stored addresses: data is stored as specified in the file.

Should errors be detected in the incoming file, they are noted, and displayed at the end of the file (since otherwise they would scroll off the screen). If the file is properly terminated with a "final record" (in the proper Intel format), errors will be displayed after this line. Otherwise, they will be displayed *after* the next non-Intel command is entered (which will be understood to mean the file has ended). Error reports include the line-number of the last faulty line read. This number (alone of all debugger output) is reported in *decimal* notation.

4.8. Input from Port

The command

```
I <port>
```

will read & display a single byte from the given port address. A 16-bit address is allowed, and the high-order bits will be set correctly (for an 8-bit address, they will be zero.

4.9. Output to Port

The command

```
O <port> <data>
```

will output the byte <data> at the given port. The address is set up as for the "I" command.

4.10. System Boot

```
S <drive> <type>
```

This command will read Sector 0 (track 0) of the specified drive, to location 8000H, sumcheck it, and execute the loaded code. Normally, this will boot a suitable operating system.

The <drive> parameter may take values 0..3, defaulting to 0. <Type> is presently assigned as "3" for a 3.5", 1.44MB drive, and "5" for a 5.25", 260kB drive. Defaults are Drive 0, 3.5", so to boot from the disk supplied, entering "S" is sufficient.

5. TROUBLESHOOTING

This section offers some tips & tricks that I have learned in developing this board. The tips are in order; starting with a plain dead board, then to more bits of it working. On the production boards, **every** fault I have yet encountered came down to a missed solder joint! (Do remember that I cannot electrically test a part-built board, so don't take my soldering of the SMD parts utterly for granted. I inspect those joints visually and mechanically, but something may get through. Experience has shown the area to check first is the two "short" sides of the Z182 CPU chip; almost all the bad joints I have had, have been there!)

5.1. Stone Dead

You did of course, first use a current-limited power supply, just in case of shorts :-) At this point, there's no substitute for a 'scope. Start by verifying that the CPU clock oscillator is running (at the right frequency), and the reset pin is behaving. You can fake a Z80 (or its successor, the Z182) by plugging in an *erased* EPROM at U4: the CPU will execute FF opcodes continually. This will show a regular signature on the CRO. Be aware that the Z80182 drives the M1\ line differently to the Z-80: it is only active on interrupt acknowledge cycles.

5.2. Garbaged Sign-on Message

That message is the result of a lot of power-on self testing, so errors in it can tell you a lot. If it identifies the SMC chip, then the basic interface to that chip is OK (this has not tested interrupts or DMA). The clock-speed is measured by setting a spare serial port (on the SMC chip) to 1200bps, and counting CPU cycles while that port sends one byte. This again, proves out the clocks, and the SMC interface. The memory layout message verifies the memory mapping logic, and proves your PCB jumpers are correct!

5.3. Disk Drive won't Work

Don't forget to wind out your power-supply current limit before trying the drive! This problem is most likely to manifest when you try to boot the system. The boot process does not give much in the way of error messages: you can learn much more by simulating the boot process manually. Use the debugger to enter the following bytes in memory:

```
8000: CF          ;RST 08, ie disk services
8001: FF          ;RST 38, ie break to debugger
```

Now set up registers as follows (registers not listed are don't care):

```
A     02          ;"Read" command
B     01          ;Read 1 sector
C     00          ;Track 0, side 0
D     00          ;Drive 0
E     01          ;Sector 1
HL    8200        ;Buffer address
IX    0480        ;Hardware parameters table in ROM (see note)
PC    8000        ;Point to the code previously entered
SP    0000        ;Stack down from FFFF
```

Note: The drive parameter table (IX value) may move with new ROM code revisions. The "future-proof" way to set IX is to use the pointer at 000B, ie `ld ix,(0BH).` This will always work.

Insert the system disk, and execute "G". The system will attempt to read the boot sector. When the debugger regains control, display the registers. If the read was successful, HL will contain 8400H (ie it will advance past the buffer), and the carry bit (Bit-0 of F) will be zero. After an error, carry will be set, and A will contain the error code, thus:

```
calerr      equ   1     ;Error in a recalibration function
ioerr       equ   2     ;Error posted by Result Phase: see (HL)
badsect     equ   3     ;Sector/Length will overrun track
lostdat     equ   4     ;DMA did not transfer full block
timeout     equ   5     ;Interrupt timed out
badcmd      equ   6     ;Invalid function code
sekerr      equ   7     ;Seek error
```

Further, the DE pair will point at the result vector returned by the controller hardware. The first two bytes are most useful, the bit assignments being:

Byte 1

Bits 7, 6	00 - Normal termination
	01 - Abnormal termination (*This is a normal termination, in this system*)
	10 - Invalid command
	11 - Abnormal due to polling
Bit 5	Seek or Recalibrate complete
Bit 4	Track-0 signal not found
Bit 3	Unused
Bit 2	Selected side
Bits 1, 0	Selected drive

Byte 2

Bit 7	Did not see TC at end of sector (*This is normal, in this system*)
Bit 6	Unused
Bit 5	CRC error
Bit 4	DMA under/overrun
Bit 3	Unused
Bit 2	No data seen
Bit 1	Write Protect
Bit 0	Missing address mark

5.4. An Example

One board worked on everything except booting the system. Running the above disk test gave a "DMA underrun" error. A continuity checker showed a missed joint at the DMA request input to the CPU. Voilà!

6. WARRANTY, Etc.

If you have purchased this board fully assembled, D-X Designs Pty Ltd warrants it in respect of faulty components or workmanship, for a period of ninety days from purchase, providing it has not been subject to damage or abuse. Defective boards must be returned, freight prepaid, to our Australian address. We will (at our option) repair or replace such boards, and return them to you.

If you have purchased a kit of parts, we cannot guarantee performance, since the conditions of handling and assembly of parts are beyond our control.

To the extent permitted by applicable laws, this warranty excludes all others express or implied, and defines our complete liability. This warranty, and the associated contract of purchase, shall be governed by the laws of the State of Western Australia.

This board, and its accompanying software, are not designed or intended for use in safety-critical or life-support applications. No liability of any kind is accepted for such use. See also the file LICENSE.TXT on the accompanying diskette, with respect to software.

This board is not a complete electronic system, but a component thereof. As such, it does not include shielding measures to limit spurious electromagnetic emissions. It is the responsibility of the person constructing a complete apparatus, to take such precautions as may be necessary to comply with relevant regulations regarding such emissions.

D-X Designs Pty Ltd
7 Buchan Close
SPEARWOOD
Western Australia 6163
Tel/fax: +61 8 9434 4280
Email: daveb@iinet.net.au