www.jupiter-ace.co.uk

96k RAM pack

From Edwin Blink

UPDATE 4 - September 2006 - We found a small problem with the RAM paging hardware. When paging the voltage on the bank signal was below 3.5v. This did not give the logic 1 that was needed for the second page to work. This has been fixed now, by Edwin. A 1n (1000pf) capacitor needs adding between the BANK and 0v line.

At power up the 96K Ram pack is a standard 48K extension and can be used in the normal way.

Paging Memeory bank AceForth words. First clear the RAMTOP to below 16384: **16383 15384 !**
Then type in the following three words of Forth.

DEFINER CODE DOES> CALL ;

CODE PAGE0 175 C, 237 C, 79 C, 253 C, 233 C,

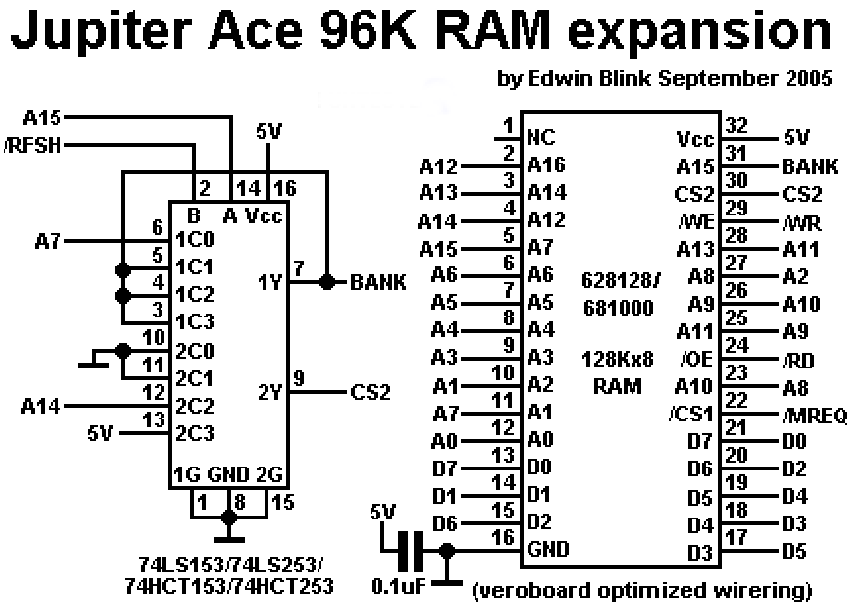
CODE PAGE1 62 C, 128 C, 237 C, 79 C, 253 C, 233 C,

Now you will have two new AceForth words PAGE0 and PAGE1. Which will swap the two 48K pages, the LED will be bright to show which page is currently in use.

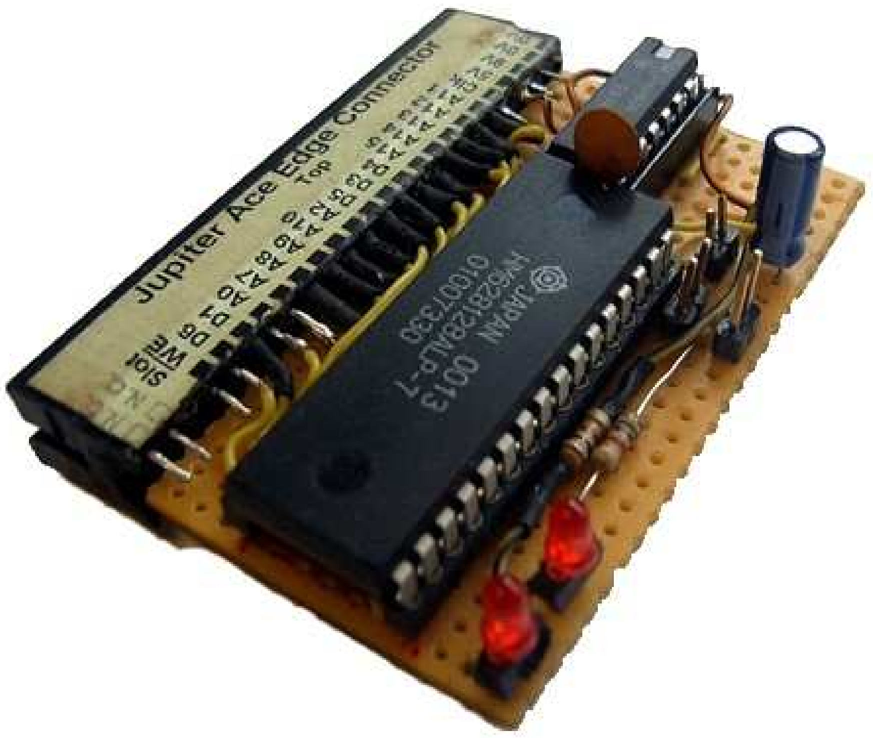
Just a note that when you swap pages you also swap the return stack that's why we lowered the RAMTOP to below 16384. You could make a Forth word to move the return stack to the PAD swap pages then copy the return stack to the new memory page. One other thing! if you have an AceForth word definition which crosses over the 16384 page boundary you will have a crash then swapping pages. A possible solution is the reserve a few bytes of RAM which crosses the page boundaries. We will be looking at these words soon.

UPDATE 3 - August 2006 - Two LED'S have been added to indicate which one of the 48K pages is paged in.
See the new images below.

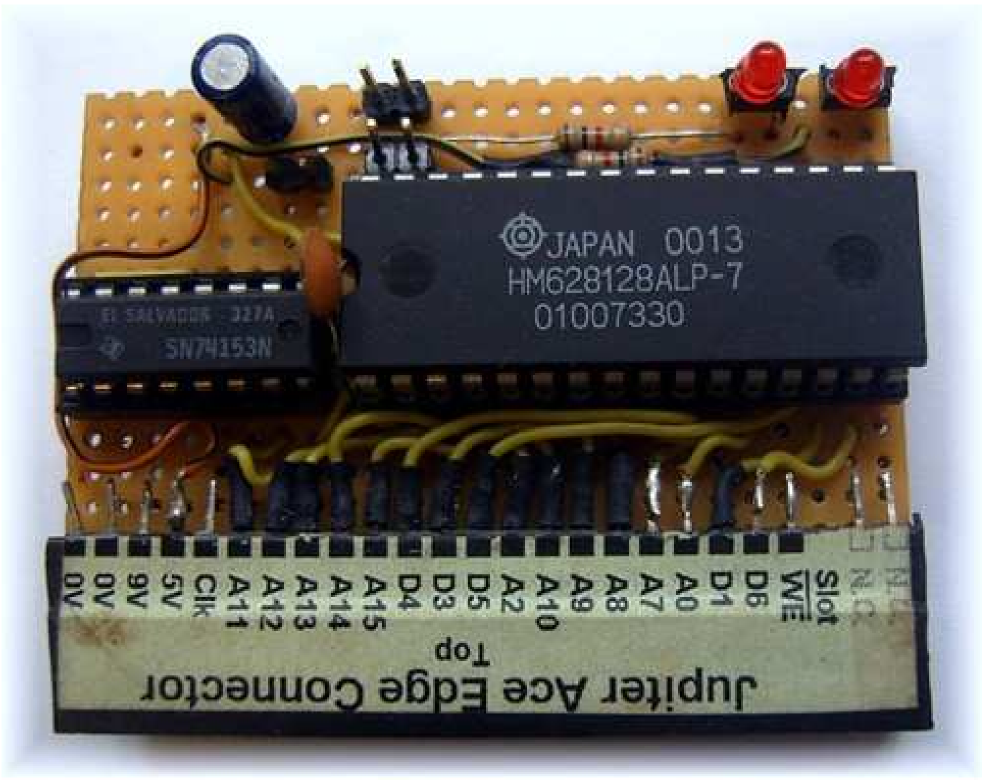
UPDATE 2 - July 2006 - this ram pack design has now been built and tested with the Ace and works fine. In the images an extra 10uf electrolytic capacitor has been added to help stabilize the power.
New! 96K RAMchecker software see below.

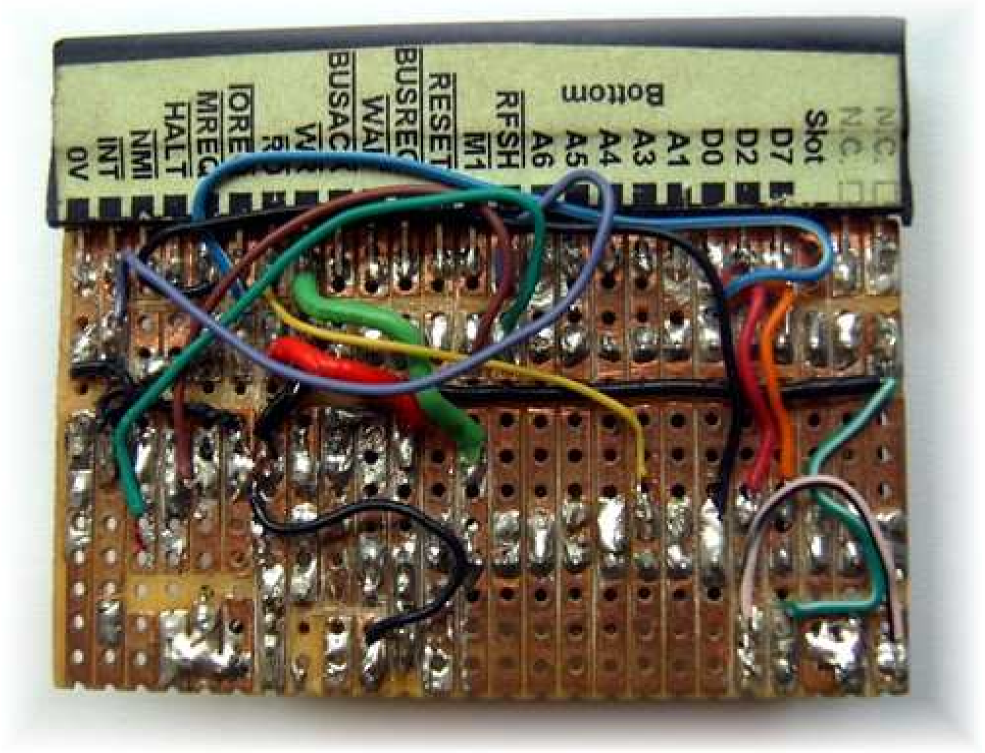

Jupiter Ace 96K RAM expansion
by Edwin Blink September 2005
A15
/RFSH
5V
A7
A14
BANK
CS2
74LS153/74LS253/
74HCT153/74HCT253
628128/
681000
128Kx8
RAM
0.1uF
(veroboard optimized wirering)

schematic



Note: That two LEDs have been added to indicate which RAM page (0 or 1) is paged in. At power up its page 0, so only one LED is bright, and occupies address from 16384 to 65535. If both LEDs are bright then page 1 occupies address 16384 to 65535. You can also see some jumpers that hardwire in RAM pages, used to track down a faulty 153 chip these jumpers will not be in future.





How it works.

The LS153 is a dual selector chip. We use one half as an OR gate to do the address decoder the other half as feedback output to create a latch. This is loaded during a Z80 refresh, A7 is bit 7 of the Refresh register. This can be programmed by software using LD A,128 LD R,A which selects the other page of 48K of RAM.

The 128K ram is accessed from 16384 to 65535, your bankswitch allows two pages at that range and internal ram is not paged out. To use the extra 48k page the stack needs to be relocated.

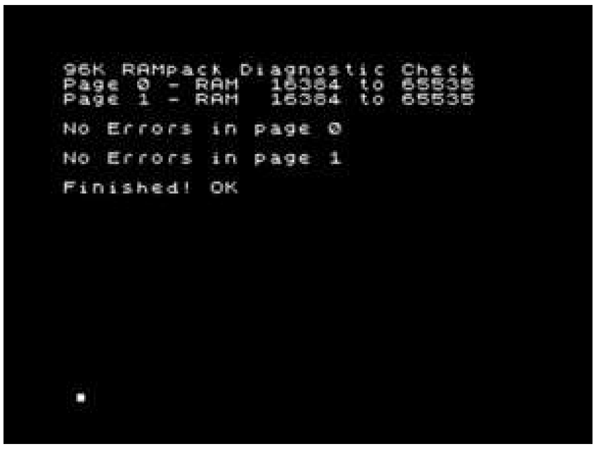
Reset: problems?..
The page that is paged out is protected against a call 0. The reset routines clears only one byte every 256 in the page that is paged in though.

Just two ICs and a small decoupling capacitor are used in the build.

Below some code to check the RAMpack is working correctly.

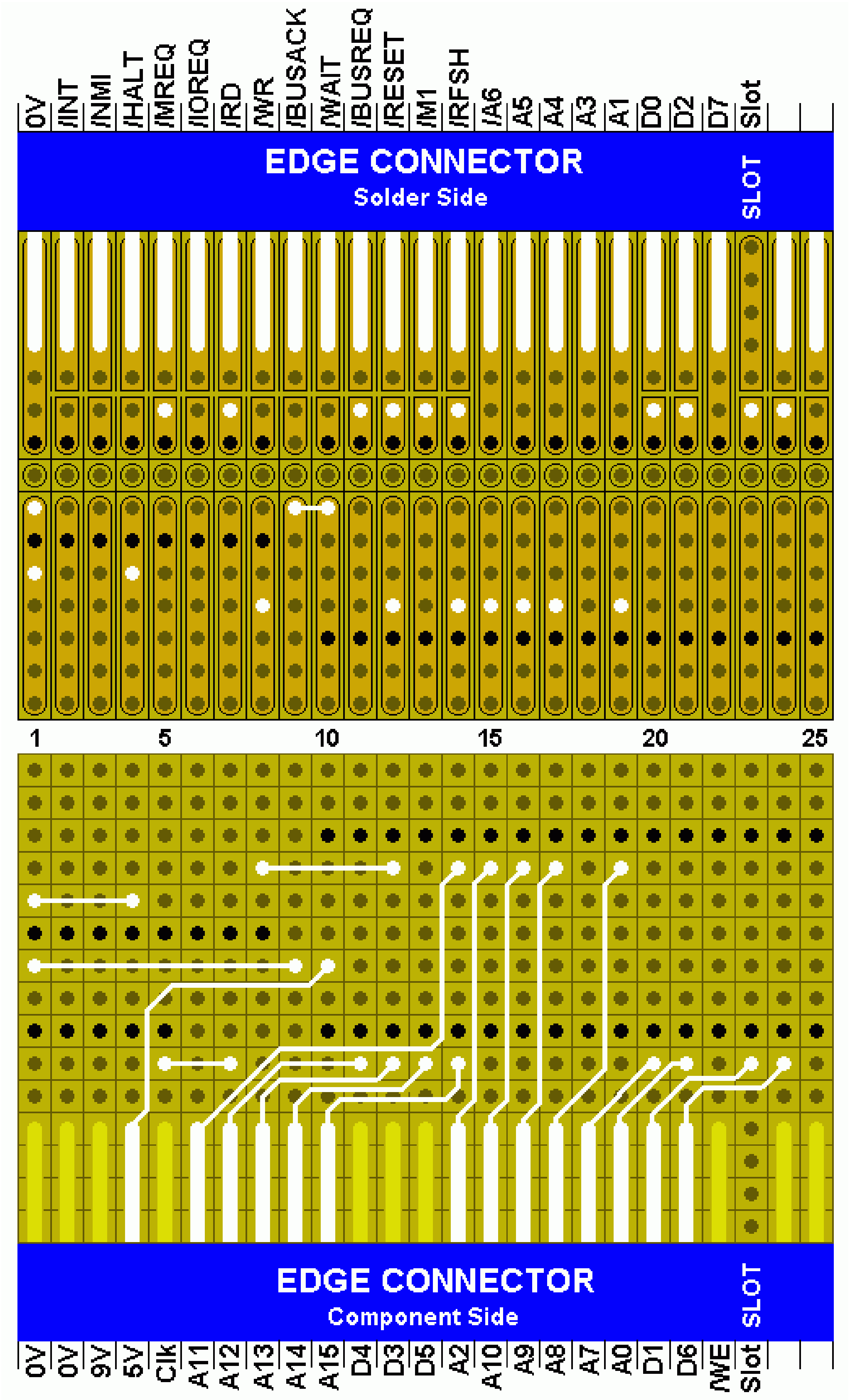
```
; 96K RAMpack check 3
;
;Assemble with TASM 3.2 with
;Tasm -80 -b -l 9kRC_3.asm RC3.bin
;Program to check the memory locations
;in each 48K page. If you get the No Errors
;message then your 96KRAMpack is working
;correctly.
;To set the Aces RAMTOP to 16000 do
;16000 15384 ! quit
;load code to 16000 with 16000 250 bload RC3
;16000 call to run.
;load into EightyOne by importing to address 16000
;file RC3.bin then 16000 CALL to run.
;EB/SPT 2006

                org 16000
start           .equ $
padaddress              .equ 27ffh                              ;end address of PAD
                call 0A24h                              ;call cls in ROM
                call s_print                    ;call print string
                .db 13                          ;print CR to screen
                .text "96K RAMpack Diagnostic Check"    ;start message
                .db 13                          ;print CR to screen
                .text "Page 0 - RAM  16384 to 65535"    ;message
                .db 13
                .text "Page 1 - RAM  16384 to 65535"
                .db 13
                .db 0                           ;end of string marker
                di
                ld (SPST),SP                    ;------?
                LD SP,padaddress                        ; store bad ram address in PAD?
                xor  a                          ;a=0 for page 0
                call pagecheck                  ;start checking PAGE 0
                call errorcheck                 ;check hl for error
                call s_print                    call print string
                .db 13                          ;No error message
                .text "No Errors in page 0"
                .db 13,0
                ; ex   de,hl                    ;save address in DE ?
                ld a,128                                ;a=128 for page 1
                call pagecheck
                call errorcheck
                call s_print
                .db 13
                .text "No Errors in page 1"
                .db 13,0
SPST            .EQU $+1                        ;point to addr inside LD SP,nn
retrun_to_forth LD SP,0
                ei
                call s_print
                .db 13
                .text "Finished!"               ;end message
                .db 0
                jp (iy)                         ;return to FORTH
;--------------------------------------------------------
errorcheck              xor a                           ;load 0 into A
                cp (hl)                         ;compare A with contents in hl address
                ret nz                          ;return
                call s_print                    ;call print string
                .text "Error in RAM"            ;error message
                .db 13, 0                       ;CR and end message marker
                jr retrun_to_forth              ;return to Forth
;--------------------------------------------------------
;
;pagecheck code by Edwin
;
;entry:
;       A=0:   RAM page 0
;       A=128: RAM Page 1
;exit:
;       Z,HL=0 ram ok
;       NZ,HL=bad ram address
```

```
;
pagecheck                ld r,a          ;select page in A
                ld hl,16384     ;test ram range-16384-65535
repeat          ld c,(hl)
                xor a
test            ld (hl),a
                cp (hl)
                ret nz          ;return error in ram
                dec a
                jr nz,test
                ld (hl),c
                inc hl
                ld  a,h         ;test H is at end of 64K range
                and a
                jr nz,repeat
                ret
;-------------------------------------------------
;s_print - print string message by spt
;text message must be placed after the call to
;s_print, and end with a 0 byte marker.
;
;entry: none
;exit   : none
;
s_print         pop hl          ;retrieve return address
                ld a,(hl)               ;into hl
                inc hl          ;increase by 1
                push hl         ;store address
                and a           ;does hold 0
                ret z           ;if so, z flag set and return
                rst 08h         ;print contents in A reg
                jr s_print              ;repeat until end marker 0 is found
                ret             ;return
;-----------------------------------------------
end             .equ $
length          .equ end-start
                .end
```

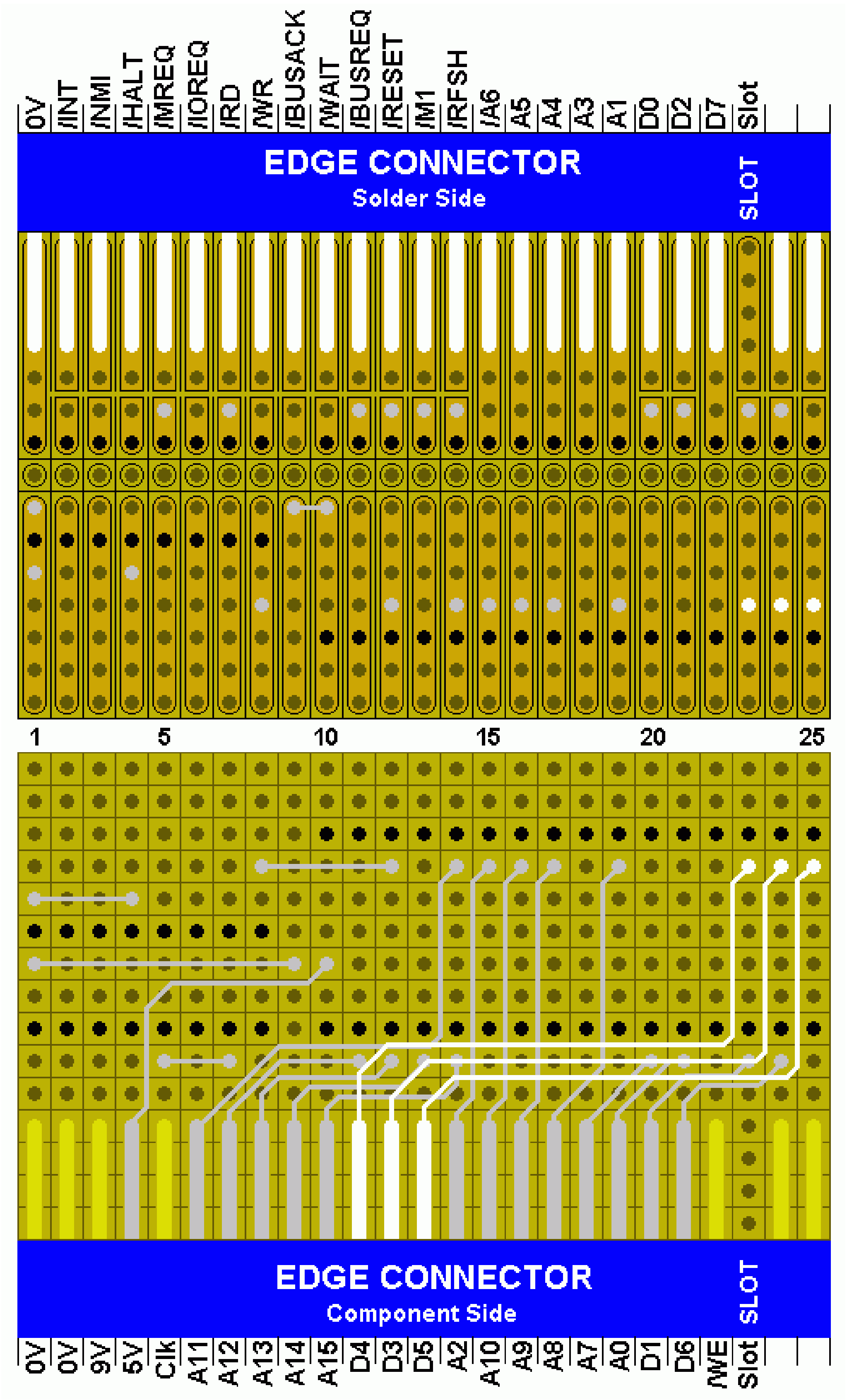


```
96K RAMpack Diagnostic Check
Page 0 - RAM  16384 to 65535
Page 1 - RAM  16384 to 65535

No Errors in page 0

No Errors in page 1

Finished! OK
```

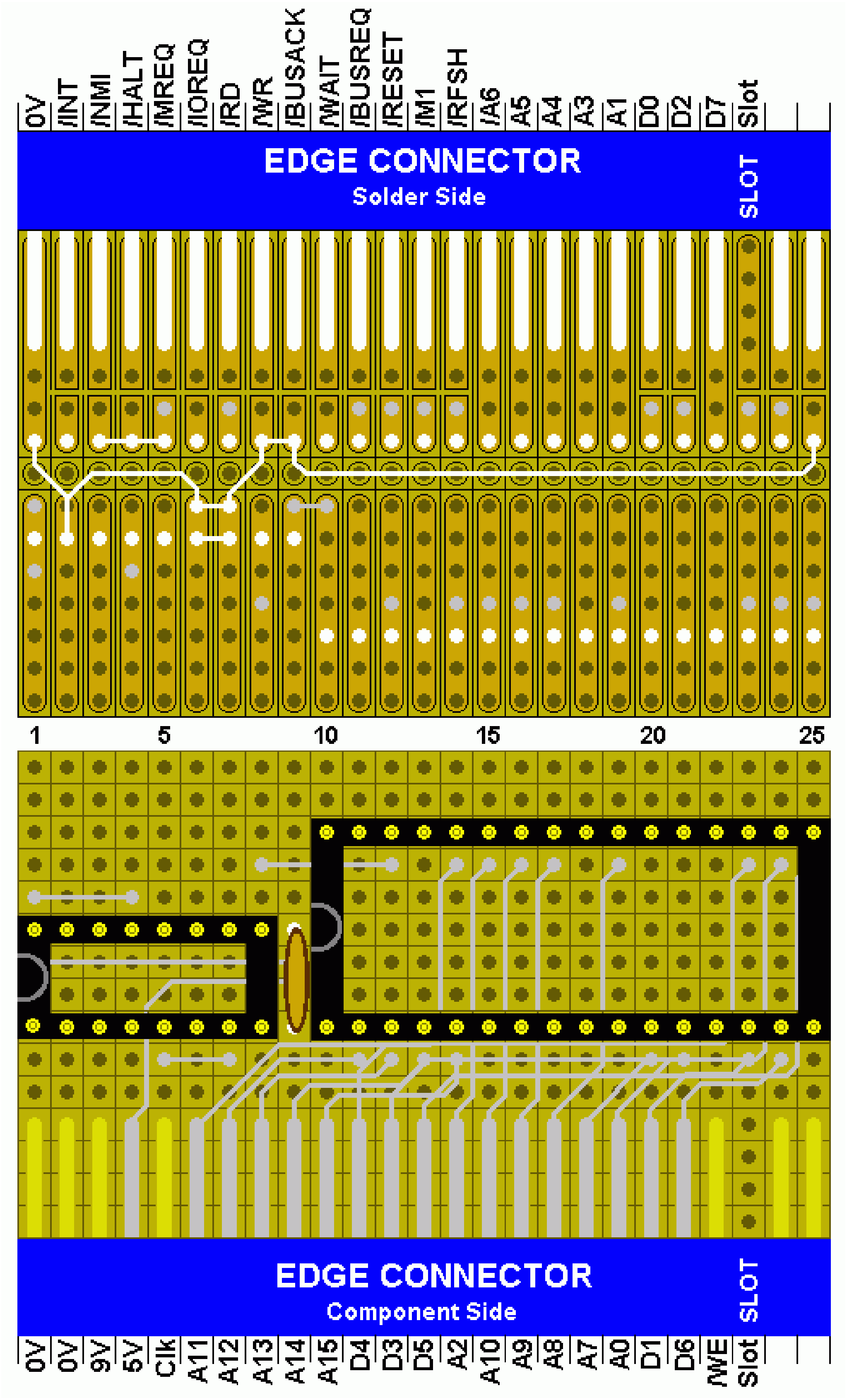
You can down load the code here in the zip file is a EightyOne TZX file, Ace snapshot file and a wav file.



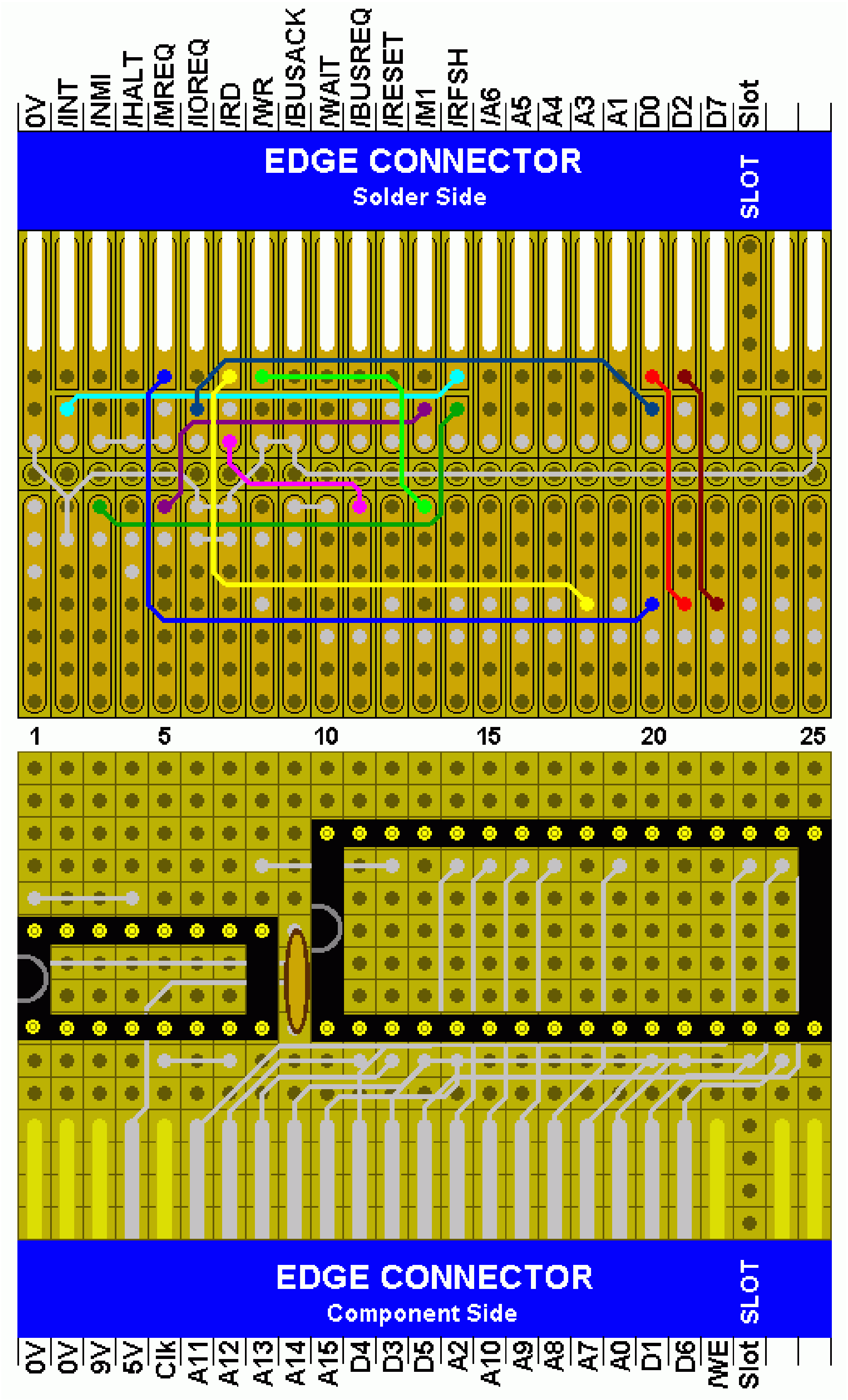
cutting tracks, soldering connector, solder wires at component side.



Component side 2nd layer wirering.



Adding components, solder side wirering



Solder side 2nd layer wirering