PCB Trace Width Calculator

Melikşah Sagun

30/05/2021

İçindekiler

Purpose of This Program

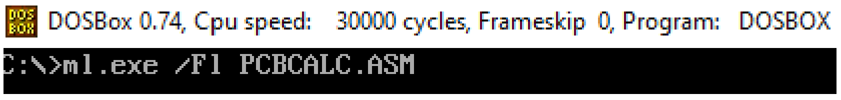
This program is designed for calculating the width of a trace on a PCB thats required to carry a given current while keeping the resulting increase in trace temperature below a specified limit. The main operations are:

1- Maximum current calculation from trace width in internal layer
2- Maximum current calculation from trace width in external layer
3- Minimum trace width calculation from currentin internallayer
4- Minimum trace width calculation from currentin external layer

User can choose the desired operation by entering corresponding number of the options on the screen. After that in each section the user will be asked the enter the width value in mils or the current value in miliamperes in 4 digits.

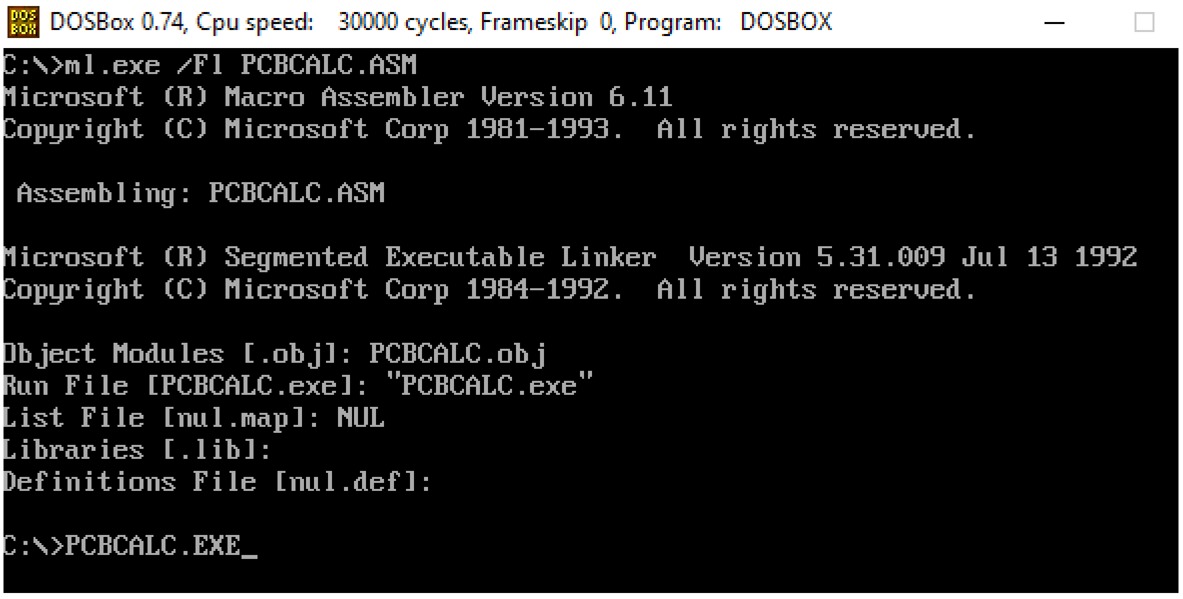
After that the program will do the necessary computations and will output the wanted values in either mils or miliamperes.

Compiling



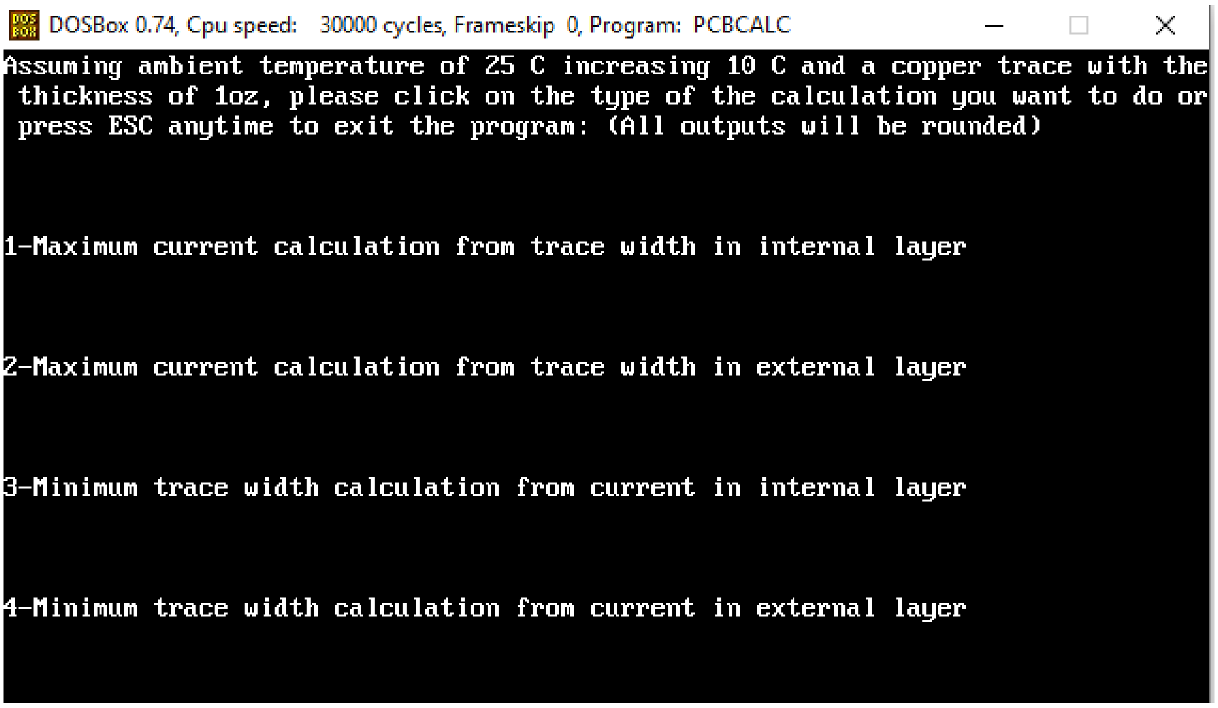
```
DOSBox 0.74, Cpu speed:    30000 cycles, Frameskip  0, Program:   DOSBOX
C:\>ml.exe /Fl PCBCALC.ASM
```

After running this command we can run the .exe;

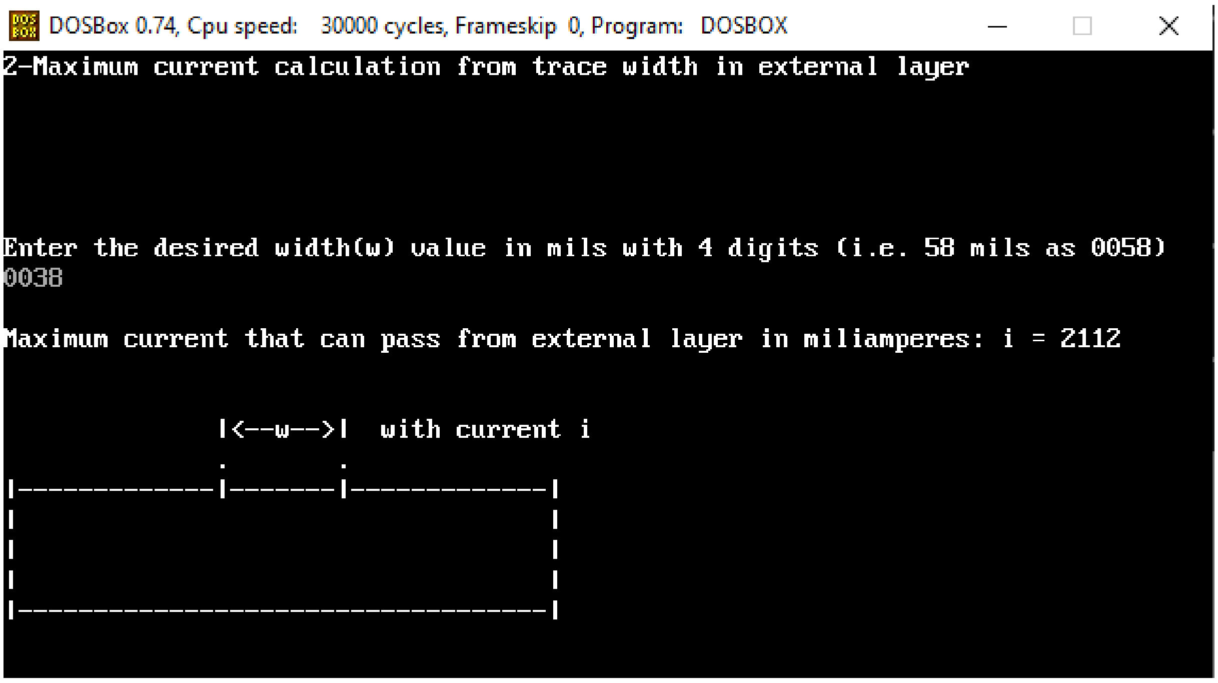


```
DOSBox 0.74, Cpu speed:    30000 cycles, Frameskip  0, Program:   DOSBOX
C:\>ml.exe /Fl PCBCALC.ASM
Microsoft (R) Macro Assembler Version 6.11
Copyright (C) Microsoft Corp 1981-1993.  All rights reserved.

 Assembling: PCBCALC.ASM

Microsoft (R) Segmented Executable Linker  Version 5.31.009 Jul 13 1992
Copyright (C) Microsoft Corp 1984-1992.  All rights reserved.

Object Modules [.obj]: PCBCALC.obj
Run File [PCBCALC.exe]: "PCBCALC.exe"
List File [nul.map]: NUL
Libraries [.lib]:
Definitions File [nul.def]:

C:\>PCBCALC.EXE_
```

Main screen;



```
DOSBox 0.74, Cpu speed:    30000 cycles, Frameskip  0, Program:  PCBCALC
Assuming ambient temperature of 25 C increasing 10 C and a copper trace with the
 thickness of 1oz, please click on the type of the calculation you want to do or
 press ESC anytime to exit the program: (All outputs will be rounded)



1-Maximum current calculation from trace width in internal layer



2-Maximum current calculation from trace width in external layer



3-Minimum trace width calculation from current in internal layer



4-Minimum trace width calculation from current in external layer
```



```
DOSBox 0.74, Cpu speed:    30000 cycles, Frameskip  0, Program:  DOSBOX
2-Maximum current calculation from trace width in external layer





Enter the desired width(w) value in mils with 4 digits (i.e. 58 mils as 0058)
0038

Maximum current that can pass from external layer in miliamperes: i = 2112


               |<--w-->|  with current i
               .       .
|-------------|-------|-------------|
|                                   |
|                                   |
|                                   |
|-----------------------------------|
```

Macros

clear_screen

```
clear_screen macro

    mov AL, 00h ; Clear screen by
scrolling
    int 10h
    mov AH,00      ;set video mode
    mov AL,12H     ;640x480 graphic mode
    int 10H
    int 10h
endm
```

Mainly used for clearing the screen

take_input

```
take_input macro
    mov AH,01H
    int 21H      ;takes the input from
keyboard
    cmp AL, 27
    jz exit

endm
```

Takes input from the user and stores in the AL register. Checks for the ESC key. If ESC is detected exits the program otherwise returns to the main program.

four_digit_input

```
four_digit_input macro  ;takes each digit and multiply it by 100, 10 and 1
and adds them together
LOCAL not_a_number, repeat, finished ; local labels
print new_line
repeat:
    mov BX, 0 ; Reset BX
    take_input
    cmp AL, 58
    jnc not_a_number
    cmp AL, 47
    jc not_a_number  ; Check if input is a number
    sub AL, 48 ; ASCII to number
    mov AH, 0
    mov CX, AX
    mov AX, 1000 ; Input 2564 = 2000 + 500 + 60 + 4
    mul CX
    add BX, AX
    ;-----------------------------------------------
    ; Other sections are same but for other digits.
    ;-----------------------------------------------
    take_input
    cmp AL, 58
    jnc not_a_number
    cmp AL, 47
    jc not_a_number
    sub AL, 48
    mov AH, 0
    mov CX, AX
    mov AX, 100
    mul CX
    add BX, AX
    ;----------
    take_input
    cmp AL, 58
    jnc not_a_number
    cmp AL, 47
    jc not_a_number
    sub AL, 48
    mov AH, 0
    mov CX, AX
    mov AX, 10
    mul CX
    add BX, AX
    ;----------
    take_input
    cmp AL, 58
    jnc not_a_number
    cmp AL, 47
    jc not_a_number
    sub AL, 48
    mov AH, 0
    mov CX, AX
    mov AX, 1
    mul CX
    add BX, AX
    ;-------------
    jmp finished

    not_a_number: ; If input is not a number print error message
        print error

    jmp repeat

    finished:

endm
```

Takes a 4 digit input from user. Multiplies each digit by the value of the position of the said digit. If one of the inputs are not a number the macro resets and asks again.

print

```
print macro message ; loads the message and
prints it to screen
    mov AH,09H
    mov DX,offset message
    int 21H
endm
```

Takes the specified string(message) from memory and prints it to the screen.

current_calc

```
current_calc macro base2, power2, coeff ; round(CX) = temp_bcd =
coeff*(base2)^power2

    ; This macro uses the x87 FPU commands.

    ;While using the 8086 is possible to calculate real powers of real
numbers,
    ;calculating with prescision is extremely hard.

    ; Even the x87 FPU can't calculate the powers of any number in one
command because
    ; the range of base2 must be between +1.0 and -1.0.

    ; The main idea of behind this calculation is that 2^(y*log2(x)) =
x^y

    fld power2
    fild base2 ; loading the variables into FPU stack

    fyl2x ;Y*Log2X
    fld1 ; Load 1
    fld st(1) ; Load st1 to st0
    fprem ; ST(0) modulo ST(1)
    f2xm1 ; 2 to the X power minus 1
    fadd ; st0 = st1 + st0
    fscale ; Scale ST(0) by ST(1)
    fxch st(1)  ;Exchange st0 with st1
    fistp temp_bcd
    fmul coeff ; st0 = st0 * coeff
    fistp temp_bcd  ; Round st0 and store it as integer to memory
    mov cx, temp_bcd ; variable to register

endm
```

Takes an integer dword(base2), float dquad(power2) and another float dquad(coeff); calculates the rounded result of $coeff * base2^{power2}$ and stores in temp_bcd and CX.

We can't take directly x^y in a x87 system so instead we are using the mathematical equation of $2^{y\,log_2 x} = x^y$. And in the f2xm1 operation, the base is limited by the range of -1.0 +1.0, we need to scale the base into required region before doing the f2xm1 operation.

Complete Code

```
.model small
.stack 64
.data
buffer    db  16 dup(' ')             ; buffer to hold chars
bufferEnd label byte
xtable    db  '0123456789ABCDEF'      ; translate table


message1 db "Assuming ambient temperature of 25 C increasing 10 C and a
copper trace with the thickness of 1oz, please click on the type of the
calculation you want to do or press ESC anytime to exit the program: (All
outputs will be rounded)" ,13,10, 13,10, 13,10,13,10, '$'
option1 db "1-Maximum current calculation from trace width in internal
layer" ,13,10, 13,10,13,10,13,10, '$'
option2 db "2-Maximum current calculation from trace width in external
layer" ,13,10, 13,10,13,10, 13,10, '$'
option3 db "3-Minimum trace width calculation from current in internal
layer" ,13,10, 13,10,13,10, 13,10, '$'
option4 db "4-Minimum trace width calculation from current in external
layer" ,13,10, 13,10,'$'
message2 db 13,10,13,10,"Enter the desired width(w) value in mils with 4
digits (i.e. 58 mils as 0058)", "$"
message4 db 13,10,13,10,"Enter the desired current(i) value in miliamperes
with 4 digits (i.e. 58 miliampere as 0058)", "$"
message5 db 13,10,13,10,"Maximum current that can pass from internal layer
in miliamperes: i = ", "$"
message6 db 13,10,13,10,"Maximum current that can pass from external layer
in miliamperes: i = ", "$"
message7 db 13,10,13,10,"Minimum trace width that can support the current
in internal layer in mils: w = ", "$"
message8 db 13,10,13,10,"Minimum trace width that can support the current
in external layer in mils: w =  ", "$"
exit_message db 13,10,13,10,"Thank you for using my program. Made by
Meliksah Sagun 21828786 for ELE338 MICROPROCESSOR ARCHITECTURE and
PROGRAMMING LAB", "$"

error db 13,10,13,10,"You entered a non number character. Please only enter
numbers",13,10,13,10, "$"


external1 db 13,10, "              |<--w-->|  with current i     " ,13,10
external2 db        "              .       ." ,13,10
external3 db        "|-------------|-------|-------------|" ,13,10
external4 db        "|                                   |" ,13,10
external5 db        "|                                   |" ,13,10
external6 db        "|                                   |" ,13,10
external7 db        "|-----------------------------------|" ,13,10, '$'

internal1 db 13,10,"              |<--w-->|   with current i " ,13,10
internal2 db       "              .       .     " ,13,10
internal3 db       "|-------------.-------.-------------|" ,13,10
internal4 db       "|             .       .             |" ,13,10
internal5 db       "|             .       .             |" ,13,10
internal6 db       "|             |-------|             |" ,13,10
internal7 db       "|-----------------------------------|" ,13,10, '$'

new_line db 13, 10, '$'
```

```
temp_bcd dw 2
base dw 16 dup(0)


clear_screen macro

    mov AL, 00h ; Clear screen by scrolling
    int 10h
    mov AH,00     ;set video mode
    mov AL,12H    ;640x480 graphic mode
    int 10H
    int 10h
endm

take_input macro
    mov AH,01H
    int 21H     ;takes the input from keyboard
    cmp AL, 27
    jz exit

endm

four_digit_input macro  ;takes each digit and multiply it by 100, 10 and 1
and adds them together

; input = BX

LOCAL not_a_number, repeat, finished ; local labels

print new_line

repeat:
    mov BX, 0 ; Reset BX
    take_input

    cmp AL, 58
    jnc not_a_number
    cmp AL, 47
    jc not_a_number  ; Check if input is a number

    sub AL, 48 ; ASCII to number
    mov AH, 0
    mov CX, AX
    mov AX, 1000 ; Input 2564 = 2000 + 500 + 60 + 4
    mul CX
    add BX, AX
    ;-----------------------------------------------
    ; Other sections are same but for other digits.
    ;-----------------------------------------------
    take_input
    cmp AL, 58
    jnc not_a_number
    cmp AL, 47
    jc not_a_number
    sub AL, 48
    mov AH, 0
    mov CX, AX
    mov AX, 100
    mul CX
    add BX, AX
    ;----------
    take_input
    cmp AL, 58
    jnc not_a_number
    cmp AL, 47
    jc not_a_number
    sub AL, 48
    mov AH, 0
    mov CX, AX
    mov AX, 10
    mul CX
    add BX, AX
    ;----------

    take_input
    cmp AL, 58
    jnc not_a_number
    cmp AL, 47
    jc not_a_number
    sub AL, 48
    mov AH, 0
    mov CX, AX
    mov AX, 1
    mul CX
    add BX, AX
    ;-------------

    jmp finished

    not_a_number: ; If input is not a number print error message
        print error

    jmp repeat

    finished:

endm

print macro message ; loads the message and prints it to screen
    mov AH,09H
    mov DX,offset message
    int 21H
endm



print_int MACRO num, numradix
LOCAL L1,L2
    pusha

    mov ax,num
    mov bx,numradix

    mov   cx,0
    mov   di,offset bufferEnd

L1: mov   dx,0             ; clear dividend to zero
    div   bx               ; divide AX by the radix
    xchg  ax,dx            ; exchange quotient, remainder
    push  bx
    mov   bx,offset xtable; translate table
    xlat                  ; look up ASCII digit
    pop   bx
    dec   di              ; back up in buffer
    mov   [di],al         ; move digit into buffer
    xchg  ax,dx           ; swap quotient into AX

    inc   cx              ; increment digit count
    or    ax,ax           ; quotient = 0?
    jnz   L1              ; no: divide again

     ; Display the buffer using CX as a counter.

L2: mov   ah,2            ; function: display character
    mov   dl,[di]         ; character to be displayed
    int   21h             ; call DOS
    inc   di              ; point to next character
    loop  L2

    popa

    print new_line
    print new_line
ENDM

current_calc macro base2, power2, coeff ; round(CX) = temp_bcd =
coeff*(base2)^power2

    ; This macro uses the x87 FPU commands.

    ;While using the 8086 is possible to calculate real powers of real
numbers,
    ;calculating with prescision is extremely hard.

    ; Even the x87 FPU can't calculate the powers of any number in one
command because
    ; the range of base2 must be between +1.0 and -1.0.

    ; The main idea of behind this calculation is that 2^(y*log2(x)) = x^y

    fld power2
    fild base2 ; loading the variables into FPU stack

    fyl2x ;Y*Log2X
    fld1 ; Load 1
    fld st(1) ; Load st1 to st0
    fprem ; ST(0) modulo ST(1)
    f2xm1 ; 2 to the X power minus 1
    fadd ; st0 = st1 + st0
    fscale ; Scale ST(0) by ST(1)
    fxch st(1)  ;Exchange st0 with st1
    fistp temp_bcd
    fmul coeff ; st0 = st0 * coeff
    fistp temp_bcd  ; Round st0 and store it as integer to memory
    mov cx, temp_bcd ; variable to register

endm

.code
start:
```

```
mov ax,@data
mov ds,ax
mov es,ax

clear_screen
print message1

print option1

print option2

print option3

print option4


take_input
cmp AL, '1'
jz opt1
cmp AL, '2'
jz opt2
cmp AL, '3'
jz opt3
cmp AL, '4'
jz opt4

opt1:
    clear_screen

    print option1

    print message2

    four_digit_input

    mov base, BX  ; Input (BX) to a variable

    ; Miliamperes for internal layers: current = 83*(width)^0.725

    power1 dq 0.725
    coeff1 dq 83.0

    current_calc base, power1, coeff1 ; coeff1*base^(power1)

    push CX   ; Store the CX for later
    print message5  ; Print message5
    pop CX

    print_int CX, 10 ; Print CX in decimal to the screen
    print internal1

    jmp exit
opt2:
    clear_screen
    print option2
    print message2
    four_digit_input
    mov base, BX

    ; Miliamperes for external layers: current = 165.5*(width)^0.7
    power2 dq 0.7
    coeff2 dq 165.5
    current_calc base, power2, coeff2
    push CX
    print message6
    pop CX
    print_int CX, 10
    print external1
    jmp exit
opt3:
    clear_screen
    print option3
    print message4
    four_digit_input
    mov base, BX

    ; Mils for internal layers(mA input): width = 0.0022*(current)^1.38

    power3 dq 1.38
    coeff3 dq 0.0022
    current_calc base, power3, coeff3
    push CX
    print message7
    pop CX
    print_int CX, 10
    print internal1
    jmp exit
opt4:
    clear_screen
    print option4
    print message4
    four_digit_input
    mov base, BX

    ; Mils for external layers(mA input): width = 0.0007*(current)^1.41

    power4 dq 1.41
    coeff4 dq 0.0007
    current_calc base, power4, coeff4
    push CX
    print message7
    pop CX
    print_int CX, 10
    print external1
    jmp exit

exit:

print exit_message
mov ah, 4Ch
int 21h
end start
```