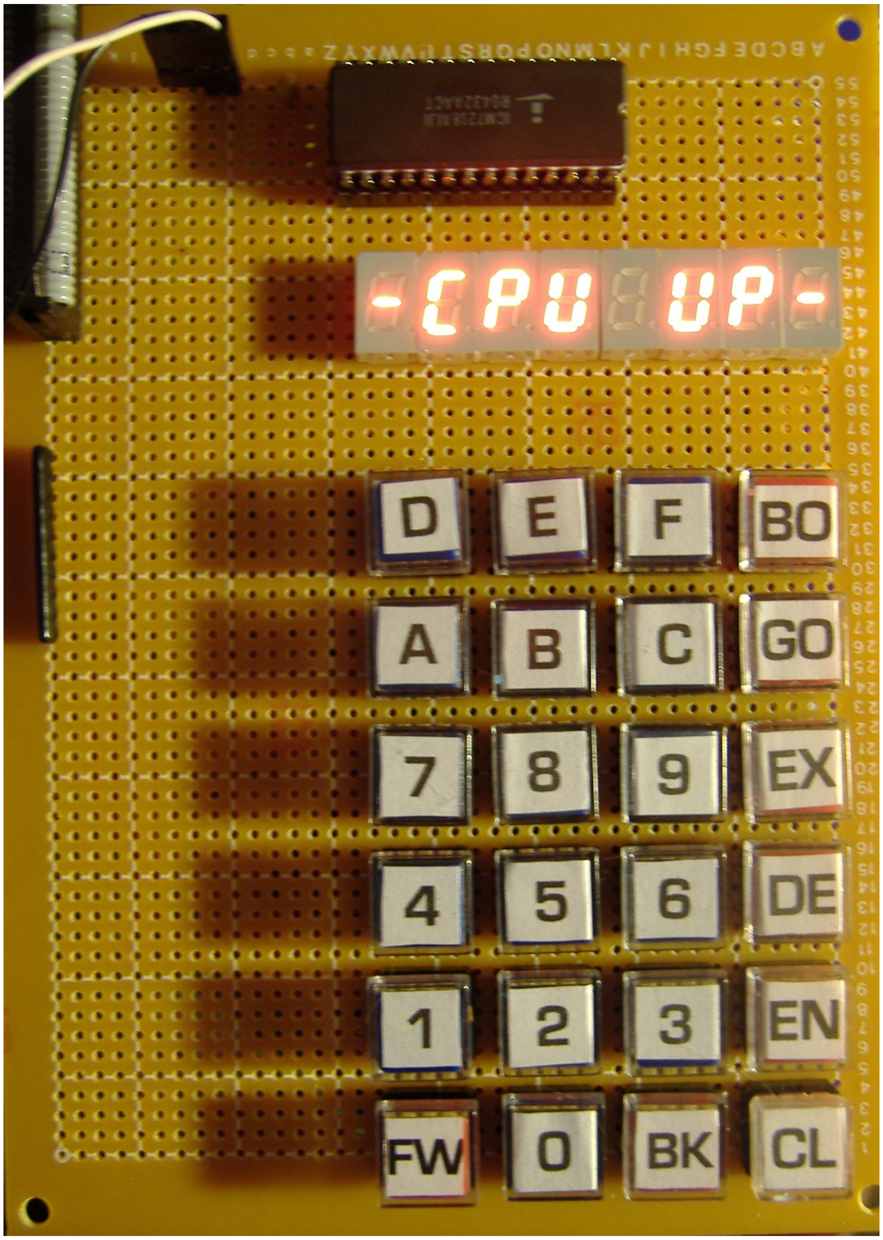
# Console Panel for the ***N8VEM*** sbc

Board design and Operation



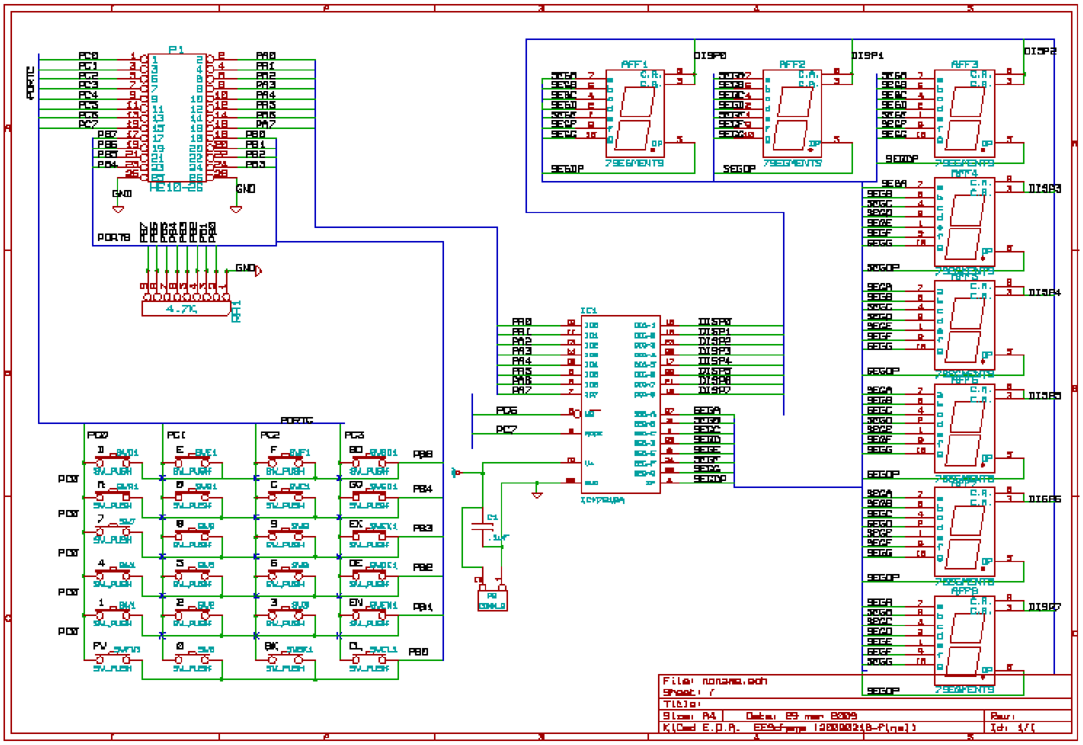
# I. Purpose

The purpose of this board is to allow a simple, low-level interface to the N8VEM hardware. The user will be able to inspect/modify ports, inspect/modify memory, execute programs, or boot the system from various ROM images or various other devices (IDE, ATAPI, Floppy, etc. . ). This can all be done without any other console hardware.

# II. Hardware Design

The hardware design of the board is very simple. Port B and the low nibble of port C of the 8255 PIO on the N8VEM card is used for an X/Y matrix Keyboard. Port B is the row sense, and port C bits 0-3 are the column strobes. The software will set port C bits 0-3 high in sequence, while port B is set to input. The LED interface uses an Intersil 7218A chip. This chip connects to port A for data input and uses bits 6 and 7 of port C as control inputs. The chip can be configured to either decode the hex digits and display them, or can be set to allow the program to directly access the segments. In the monitor software I have chosen to directly drive the segments to allow a selection of letters to be displayed as well as the standard hex numbers.

**Figure 1. Schematic**



# III. Console Panel Operation

Operation of the console panel is very simple. The user begins by pressing one of the function keys, [PW],[PR],[DE],[EX],[GO] or [BO]. The system will then prompt for required parameters, then execute the function.

## [PW] function – Port Write

- At the -CPU UP- prompt the user presses the [PW] key.
- System prompts for the port to output to: PORT . .
- The user enters the two digit port on the keypad, then presses [EN] (enter). The [CL] key will clear the input if the user enters the port incorrectly (before pressing [EN]
- System will the prompt for the value to send to the port: P0xx .. (xx is the port that was entered)
- The user then enters the byte value to send to the port, then presses [EN] (enter). The [CL] key will clear the input if the user enters the port incorrectly (before pressing [EN]
- System will respond with -CPU UP- to indicate that it is ready for next command.

## [PR] function – Port Read

- At the -CPU UP- prompt the user presses the [PR] key.
- System prompts for the port to output to: PORT . .
- The user enters the two digit port on the keypad, then presses [EN] (enter). The [CL] key will clear the input if the user enters the port incorrectly (before pressing [EN]
- System will then display the port and the input value from that port: P0xx yy (xx is the port that was entered, yy is the inputted value)
- The user then can press the [CL] key to return to the -CPU UP- prompt.

## [DE] function – Deposit to Memory

- At the -CPU UP- prompt the user presses the [DE] key.
- System prompts for the address to write to: ADDR . . . .
- The user enters the four digit address on the keypad, then presses [EN] (enter). The [CL] key will clear the input if the user enters the address incorrectly (before pressing [EN]
- System will the prompt for the value to write to the address: xxxx .. (xxxx is the address that was entered)
- The user then enters the byte value to write to the address, then presses [EN] (enter). The [CL] key will clear the input if the user enters the port incorrectly (before pressing [EN]
- The user can then press the [CL] key to return to the -CPU UP- prompt, or can press [EN] and the system will prompt the user to enter a value to the next consecutive address. The user can also press the [DE] key to enter a new address.

## [EX] function – Examine Memory

- At the -CPU UP- prompt the user presses the [EX] key.
- System prompts for the address to write to: ADDR . . . .
- The user enters the four digit address on the keypad, then presses [EN] (enter). The [CL] key will clear the input if the user enters the address incorrectly (before pressing [EN]
- System will then display the address and the value.: xxxx yy (xxxx is the address that was entered, yy is the value at that address)
- The user can then press the [CL] key to return to the -CPU UP- prompt, or can press [EN] and the system will display the next consecutive address and value. The user can also press the [EX] key to enter a new address.

## [GO] function – Execute Program

- At the -CPU UP- prompt the user presses the [GO] key.
- System prompts for the address to write to: ADDR . . . .
- The user enters the four digit address on the keypad, then presses [EN] (enter). The [CL] key will clear the input if the user enters the address incorrectly (before pressing [EN]
- System will then execute the code at that address.

## [BO] function – Boot System

- At the -CPU UP- prompt the user presses the [BO] key.
- System prompts for the boot device/location: BOOT . .
- The user enters the one digit boot identifier
- System will then boot up using that device/location

  Currently supported:

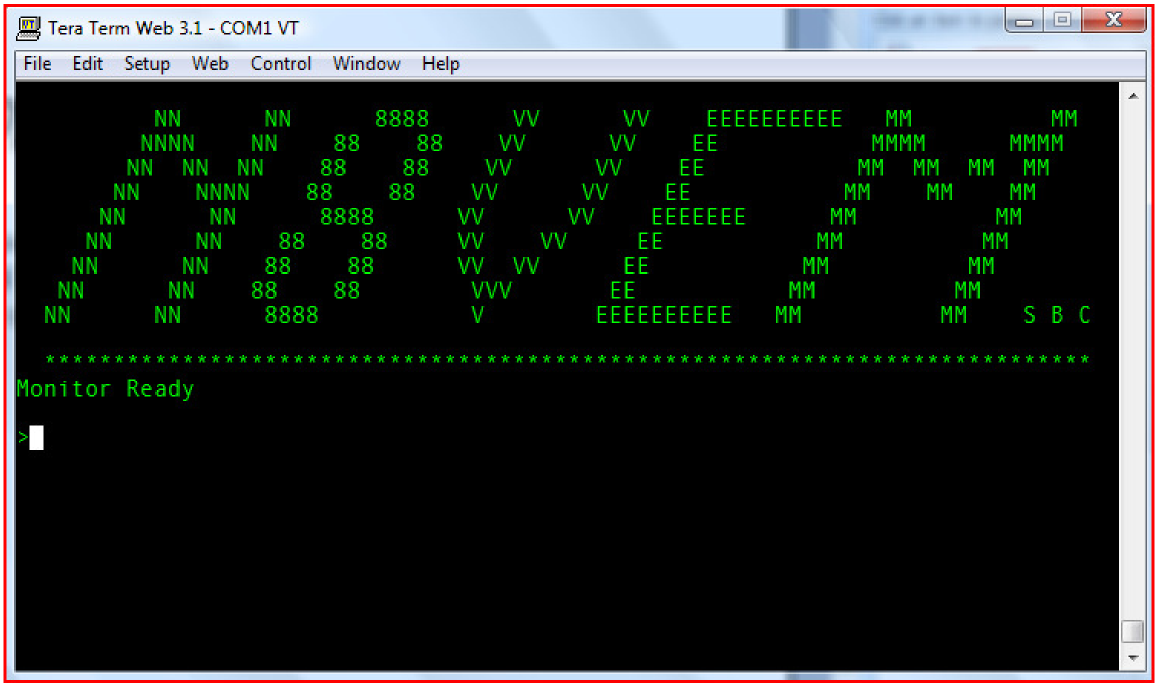
  0 Boot to Serial Console Debug Monitor
  1 Boot CPM from Rom

  Planned:

  2 Boot from Primary IDE Device
  3 Boot from Primary ATAPI Device
  4 Boot from Floppy A

# *IV. Serial Console Operation*

Boot code 0 will bring up the serial console based debug monitor. This monitor is the same monitor that is currently used with the N8VEM, except that a robust line editor has been added to allow easier interaction with the monitor. Otherwise the code is (more or less) the same code that was written by Andrew Lynch.

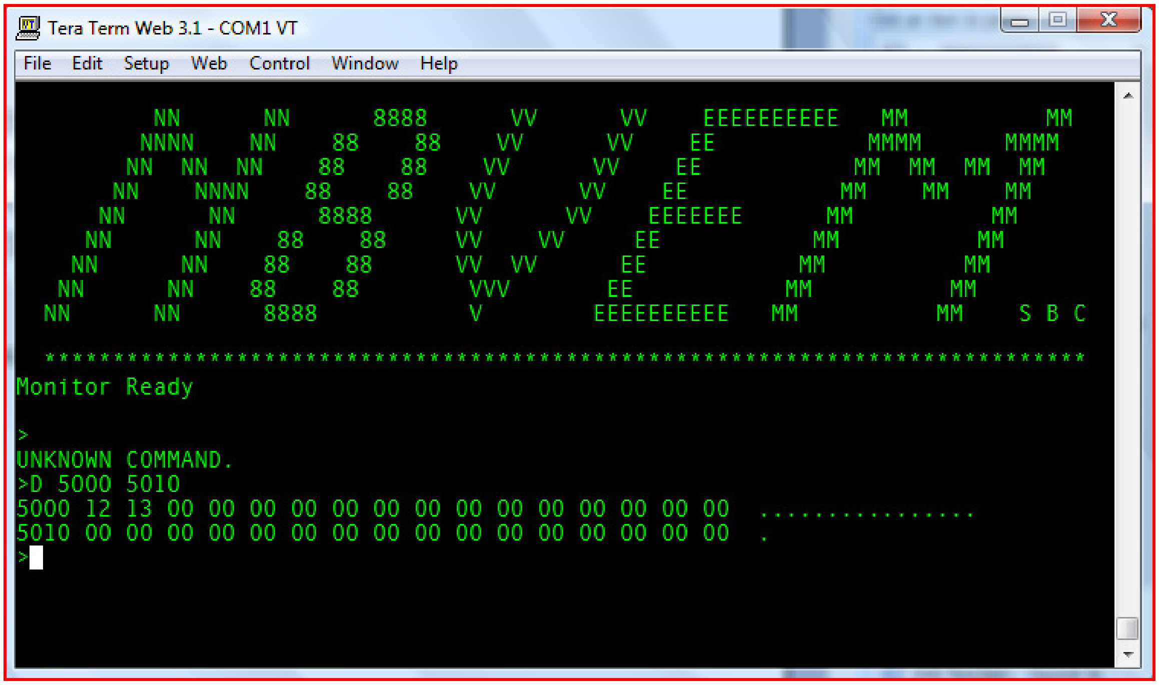


```
          NN      NN      8888      VV      VV    EEEEEEEEEE   MM          MM
         NNNN    NN    88    88    VV      VV    EE           MMMM      MMMM
        NN  NN  NN    88    88    VV      VV    EE           MM  MM  MM  MM
       NN    NNNN    88    88    VV      VV    EE           MM    MM    MM
      NN      NN      8888      VV      VV    EEEEEEE      MM          MM
     NN      NN    88    88     VV    VV     EE           MM          MM
    NN      NN    88    88      VV  VV      EE           MM          MM
   NN      NN    88    88        VVV       EE           MM          MM
  NN      NN      8888           V        EEEEEEEEEE   MM          MM    S B C

  *****************************************************************************
Monitor Ready

>
```

Supported Commands are:

**C** BOOT CP/M FROM ROM DRIVE  
**D** XXXXH YYYYH DUMP MEMORY FROM XXXX TO YYYY  
**F** XXXXH YYYYH ZZH FILL MEMORY FROM XXXX TO YYYY WITH ZZ  
**H** LOAD INTEL HEX FORMAT DATA  
**I** XX INPUT FROM PORT XX AND SHOW HEX DATA  
**K** ECHO KEYBOARD INPUT  
**M** XXXXH YYYYH ZZZZH MOVE MEMORY BLOCK XXXX TO YYYY TO ZZZZ  
**O** XXH YYH OUTPUT TO PORT XX HEX DATA YY  
**P** XXXXH YYH PROGRAM RAM FROM XXXXH WITH VALUE IN YYH, WILL PROMPT FOR NEXT LINES FOLLOWING UNTIL CR  
**R** RUN A PROGRAM FROM CURRENT LOCATION

An Example of one of these commands would be:


Tera Term Web 3.1 - COM1 VT
File Edit Setup Web Control Window Help

```
          NN      NN      8888      VV      VV    EEEEEEEEEE   MM          MM
         NNNN    NN    88    88    VV      VV    EE           MMMM      MMMM
        NN  NN  NN    88    88    VV      VV    EE           MM  MM  MM  MM
       NN    NNNN    88    88    VV      VV    EE           MM    MM    MM
      NN      NN      8888      VV      VV    EEEEEEE      MM          MM
     NN      NN    88    88     VV    VV     EE           MM          MM
    NN      NN    88    88      VV  VV      EE           MM          MM
   NN      NN    88    88        VVV       EE           MM          MM
  NN      NN      8888           V        EEEEEEEEEE   MM          MM    S B C

  *****************************************************************************
Monitor Ready

>
UNKNOWN COMMAND.
>D 5000 5010
5000 12 13 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
5010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .
>
```

# V. Summary

Please note that this is still a work in progress,  it currently runs as a COM file in CP/M, and I have no plans to test it in a ROM image until I get the remainder of the boot options working.  There are possibly (likely) bugs in the code, and I am sure that it could be optimized much more than it is.  Due to the size of this code I also plan on changing the CP/M boot so that CP/M over-writes the monitor on boot up, allowing as much RAM as possible to be free for the TPA.