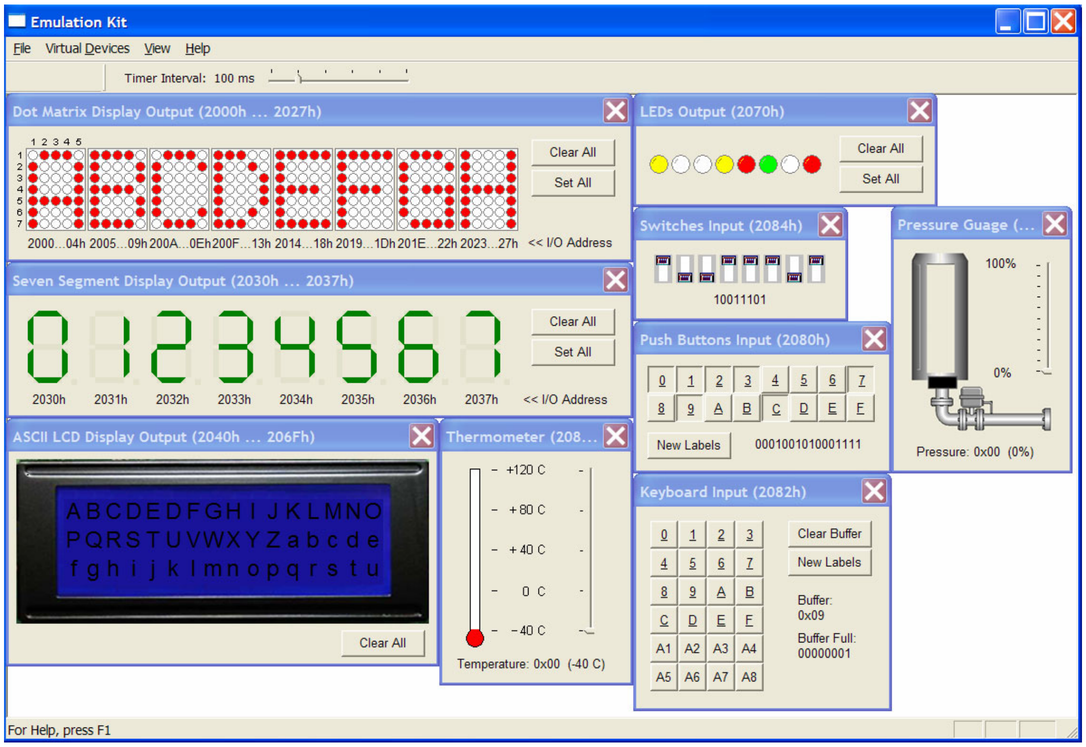
I/O Emulation Kit[1]

The **I/O Emulation Kit** is a collection of virtual I/O devices that are built to work with Emu8086. Emu8086 can access the different devices within I/O Emulation Kit using Assembly IN/OUT instructions at the I/O address range from 2000h to 2FFFh (8192 to 12287). The addresses corresponding to each device are listed in Table 1 below.



Timer Interval: 100 ms

Temperature: 0x00 (-40 C)

Pressure: 0x00 (0%)

Buffer: 0x09

Buffer Full: 00000001

Table 1. Available Devices

Device	I/O Addresses	Number of Addresses	Register Type
Dot Matrix Display Output	2000h … 2027h	40	8 bit
Seven Segment Display Output	2030h … 2037h	8	8 bit
ASCII LCD Display Output	2040h … 206Fh	48	8 bit
LEDs Output	2070h	1	8 bit
Push Buttons Input	2080h	1	16 bit
Keyboard Input	2082h … 2083h	2	8 bit
Switches Input	2084h	1	8 bit
Thermometer Input	2086h	1	8 bit
Pressure Gauge Input	2088h	1	8 bit

[1] This help file is a work in progress. More material will be added in the near future.

The devices in “I/O Emulation Kit” communicate with Emu8086 through the text file called “EmuPort.io" which is located in Windows "Temp" directory (this directory can be accessed by GetTempPath() function of the Windows API).

All you need is to run Emu8086 and I/O Emulation Kit in a Windows environment. Many Assembly program samples are provided to show you how to access and deal with the different devices within I/O Emulation Kit.

If you wish to see a menu item for “I/O Emulation Kit” from within the "Virtual Devices" menu of Emu8086, then add the binary files of “I/O Emulation Kit” to the “DEVICES” folder of Emu8086 folder (e.g. in “C:\Program Files\Emu8086\DEVICES”). You can however run “I/O Emulation Kit” separately from Emu8086 and the two applications will still be able to communicate.

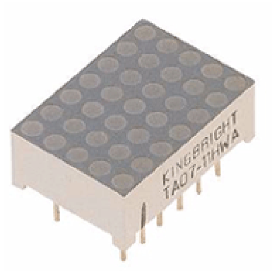
I/O Emulation Kit provides an extra flexibility in which the I/O devices can be fast or slow (i.e., respond quickly or slowly to data coming from Emu8086). This is controlled by the Timer interval tool bar which allows you to set the Update frequency of data arriving to each device to one of the following values:
10, 100, 200, 300, 400 and 500 ms

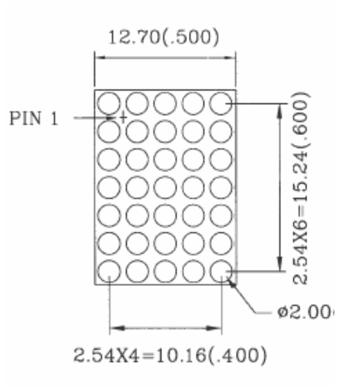
Notice that Emu8086 has a similar strategy in adjusting the speed of the microprocessor, but the range of values Emu8086 supports is:
0, 1, 100, 200, 300 and 400 ms

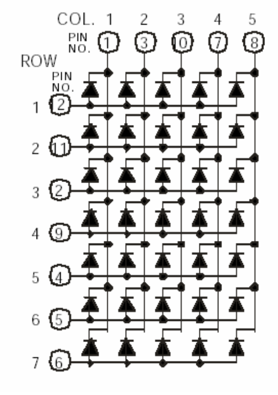
You may need to adjust the update interval depending on the value selected by the Emu8086 software and the speed of your PC for optimal performance.

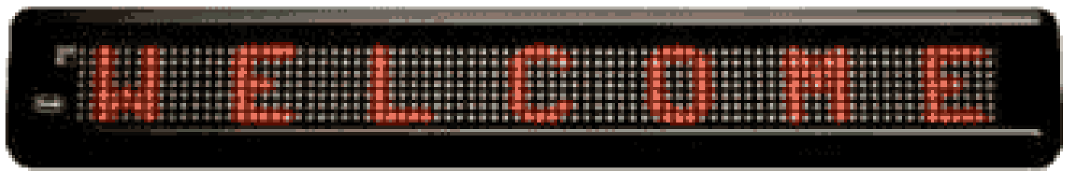
Dot Matrix Display Output

This output device represents a block of eight mono-color anode-row 5x7 Dot Matrix Displays, such as the LMD07057B shown below.









Each 5 x 7 dot matrix display consists of 14 pins. Five pins are designated for addressing the columns and seven pins are designated to addressing the rows. The remaining two pins are designated for power and ground.

In Type B dot matrix displays, the rows are connected to the anodes (+) of the 35 LEDs, while the columns are connected to the cathodes (-) of these LEDs. Rows are numbered from 1 up to 7 down, while columns are numbered 1 left to 5 right.

To light a certain LED (dot) in the matrix, the row of that dot should be connected to logic 1 (+5 volts) and its corresponding column to logic 0 (0 volts). The controller in this device lights the first column, the second column and so on all the way to the seventh column, and then the process is repeated.

Each dot matrix display within this device is controlled by five 8-bit interface registers to light the five columns, each one of these interface registers controls one column of the dot matrix display. The first register (least I/O address) controls column 1, the second register controls column 2, and so on.

If the bit is logic 1 in the interface register, it tells the dot matrix controller to light the corresponding dot (LED) ON in a particular column. If the bit is logic 0, the

LED is turned OFF. The 8-bit interface register of each column is interpreted as follows,

7	6	5	4	3	2	1	0
X	R7	R6	R5	R4	R3	R2	R1

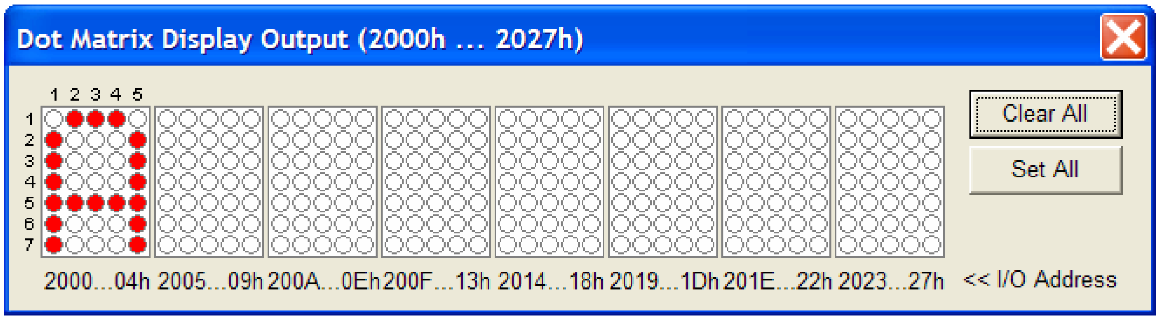
Since the device contains eight dot matrix displays, the device contains a total of forty 8-bit Interface registers which are assigned the I/O addresses 2000h to 2027h.

An example is shown below which displays the character 'A' on the first Dot Matrix display on the device (see the figure below):

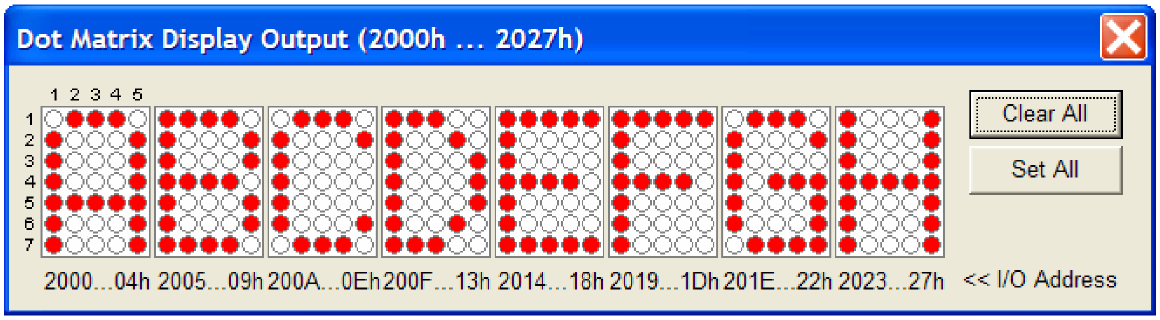
```
Dots  DB    01111110b, 00010001b, 00010001b, 00010001b, 01111110b ; A

      MOV DX, 2000h      ; first DOT MATRIX column
      MOV SI, 0
      MOV CX, 5

NEXT: MOV AL, Dots[SI]
      OUT DX, AL
      INC SI
      INC DX
      LOOPNE NEXT
```

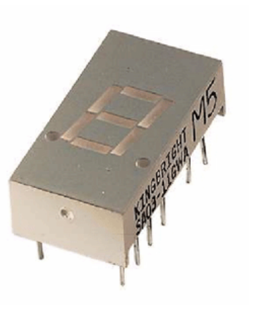

Dot Matrix Display Output (2000h ... 2027h)
1 2 3 4 5
1
2
3
4
5
6
7
Clear All
Set All
2000...04h 2005...09h 200A...0Eh 200F...13h 2014...18h 2019...1Dh 201E...22h 2023...27h
<< I/O Address

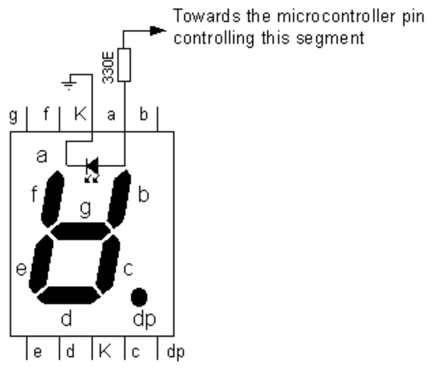
You can run the example "dot_matrix.asm" to display the first six character of the alphabet on the Dot Matrix Display Output as the figure shown below.


Dot Matrix Display Output (2000h ... 2027h)
1 2 3 4 5
1
2
3
4
5
6
7
Clear All
Set All
2000...04h 2005...09h 200A...0Eh 200F...13h 2014...18h 2019...1Dh 201E...22h 2023...27h
<< I/O Address

Seven Segment Output

This output device represents a block of eight Seven Segment Displays, such as the one shown below.





Each seven segment display consists of 10 pins. Eight pins are connected to the anodes of the LEDs in the display, while the cathodes of these LED displays are connected together and connected to the other two pins, which are usually connected to the ground.

Each seven segment display within this device is controlled by one 8-bit interface registers to light the different LEDs of the display. To light a certain LED in the display, the corresponding bit in the interface register should be set to logic 1. If the bit is logic 0, the LED is turned OFF. The 8-bit interface register of each column is interpreted as follows,

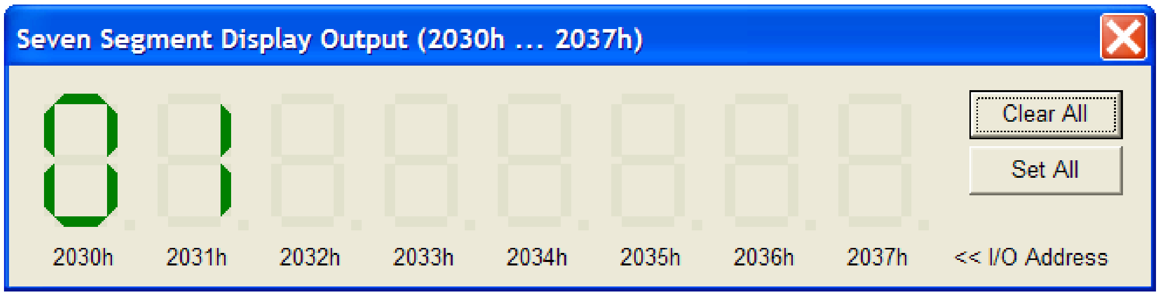
7	6	5	4	3	2	1	0
DP	g	f	E	d	c	b	a

Since the device contains eight dot matrix displays, the device contains a total of eight 8-bit Interface registers which are assigned the I/O addresses 2030h to 2037h.

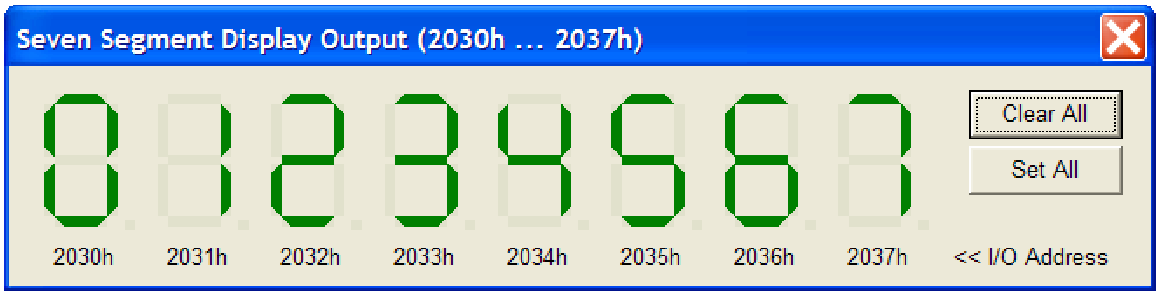
An example is shown below which displays the numbers 0 and on the first two seven segment displays of the device (see the figure below):

```
MOV DX, 2030h     ; first seven segment display
MOV AL, 00111111b
OUT DX, AL

MOV DX, 2031h     ; second seven segment display
MOV AL, 00000110b
OUT DX, AL
```



You can run the example “seven_segment.asm” to display the first eight numbers on the Seven Segment Display Output as the figure shown below.



ASCII LCD Output

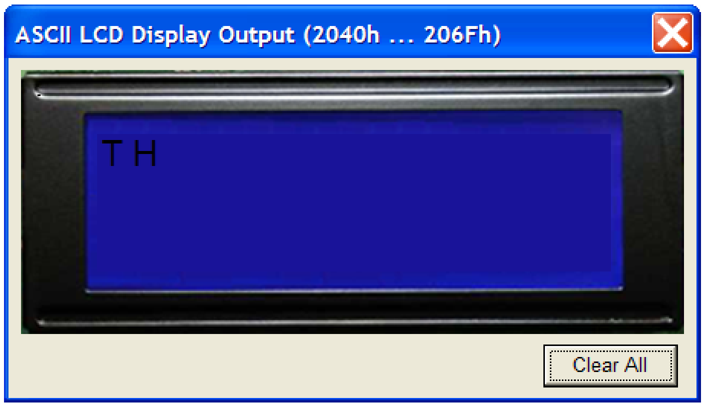
This output device represents a mono-color LCD that is capable of displaying ASCII characters (3 rows x 16 columns). A typical mono-color LCD display is shown below.

The LCD display implemented here supports only ASCII characters. To display a character on one of the 3 x 16 available locations on the display, you have to send an 8-bit value (between 0 and 255) representing the ASCII code of the character you want to display to the appropriate I/O address. The first location of the first row has the address 2040h, while the second location of the first row has the address 2041h and so on.

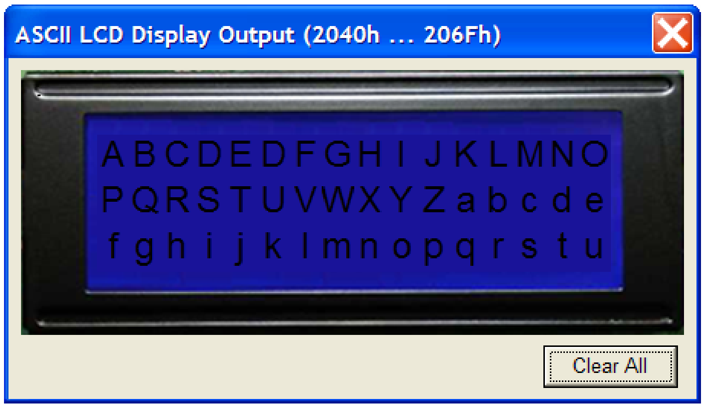
An example is shown below which displays the two characters ‘T’ and ‘H’ on the first row of the ASCII LCD display (see the figure below):

```
MOV DX, 2040h
MOV AL, 'T'
OUT DX, AL

MOV DX, 2041h
MOV AL, 'H'
OUT DX, AL
```



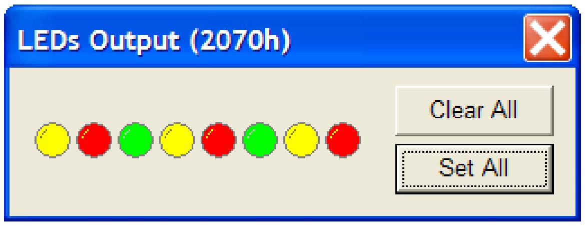
You can run the example “ASCII.asm” to see how you can write on all the rows available on the ASCII LCD Display. The figure below shows the outcome of running this sample Assembly code.



```
ABCDEDFGHI JKLMNO
PQRSTUVWXYZabcde
 fghijklmnopqrstu
```

LED Display Output

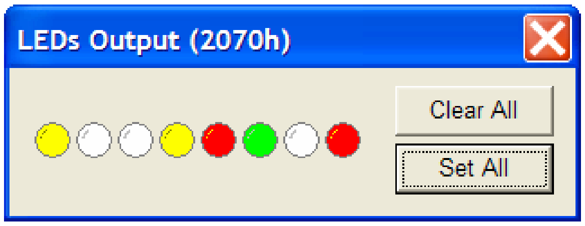
This output device represents a group of eight colored Light Emitting Diodes (LEDs), such as the ones shown below.



The LEDs are controlled by an 8-bit interface register at I/O address 2070h. Each bit of this register controls one LED. To light a certain LED, the corresponding bit of the interface register should be set to logic 1.

An example is shown below which lights a group of LEDs in the output device (see the figure below):

```
MOV DX, 2070h
MOV AL, 10011101b
OUT DX, AL
```



You can run the examples “LED_sequence.asm” and “LED_switches.asm” to see other ways of suing the LED display output device.