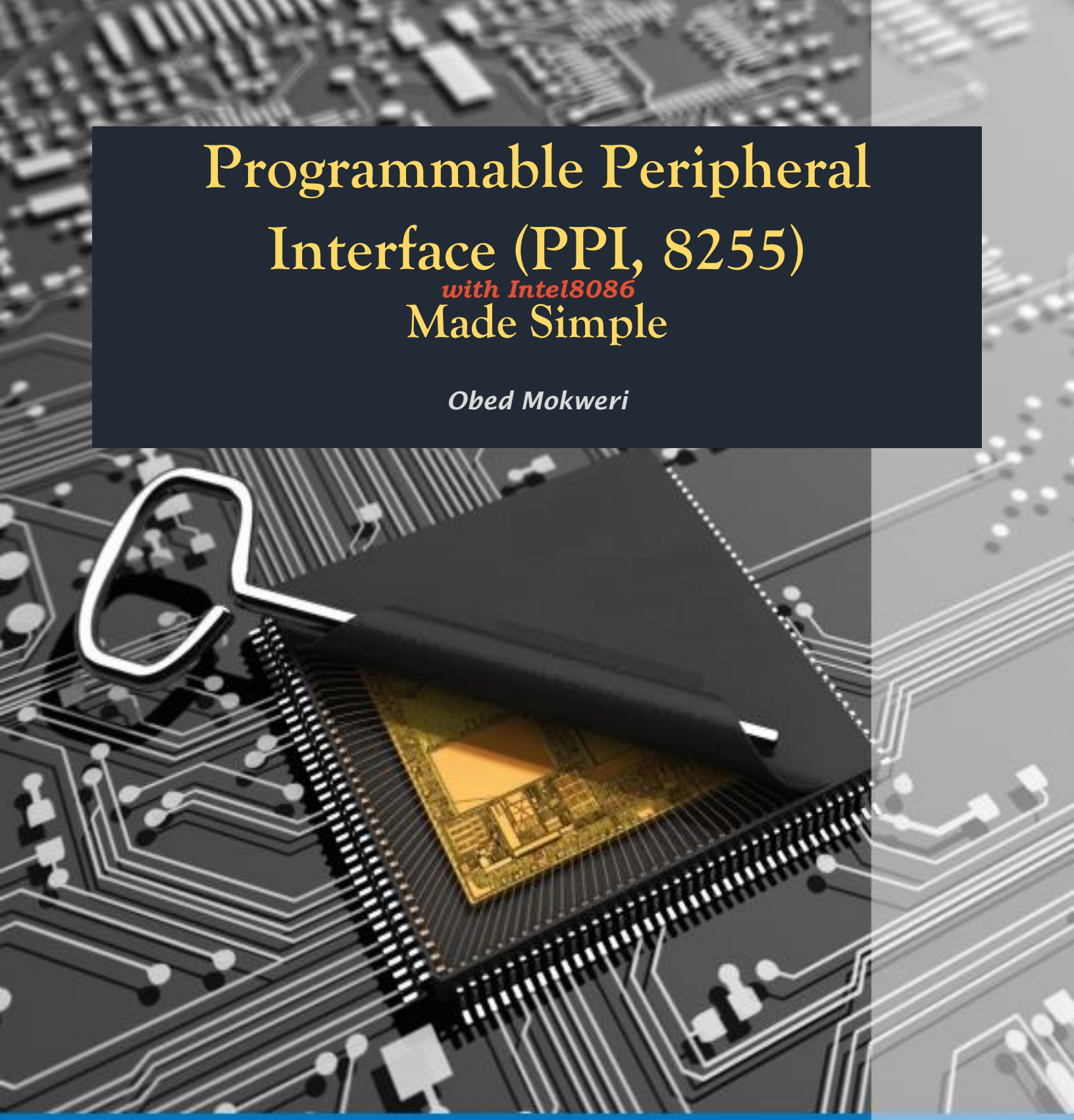
Programmable Peripheral Interface (PPI, 8255)

with Intel8086

Made Simple

Obed Mokweri



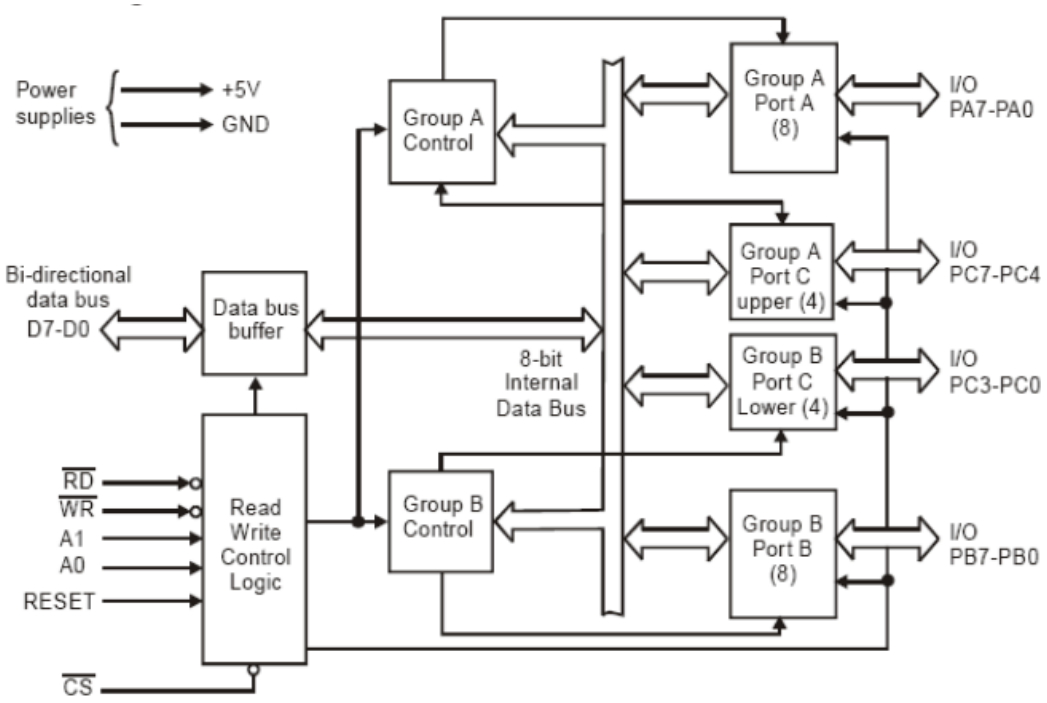
Contents;

Compiled by: Eng. Obed © 2018
(Embedded Systems Developer)

8255 Architecture

The 8255A is a programmable peripheral interface (PPI) device designed for use in Intel microcomputer systems. Its function is that of a general purposes I/O component to Interface peripheral equipment to the microcomputer system bush. The functional configuration of the 8255A is programmed by the systems software so that normally no external logic is necessary to interface peripheral devices or structures.
There are 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation.
The high performance and industry standard configuration of the ***8255A*** make it compatible with the ***8086***.

8255 Internal Architecture



Functional Diagram

8255 Functional Description

Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 8255A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

- ✓ **A0 and A1**: ***Port Select 0 and Port Select 1***
  These input signals, in conjunction with the $\overline{RD}$ and $\overline{WR}$ inputs, control the selection of one of the three ports or the control word register. They are normally connected to the least significant bits of the address bus **(A0 and A1).**
- ✓ ***$\overline{CS}$: Chip Select***
  A "low" on this input pin enables the communication between the 8255A and the CPU.
- ✓ ***$\overline{RD}$: Read***
  A "low" on this input pin enables 82C55A to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to "read from" the 8255A
- ✓ ***$\overline{WR}$: Write***
  A "low" on this input pin enables the CPU to write data or control words into the 8255A.
- ✓ ***RESET:***
  A "high" on this input initializes the control register to 9Bh and all ports (A, B, C) are set to the input mode. "Bus hold" devices internal to the 8255A will hold the I/O port inputs to a logic "1" state with a maximum hold current of 400mA.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 8255A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 8255A.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A - Port A and Port C upper (C7–C4)
Control Group B - Port B and Port C lower (C3–C0)

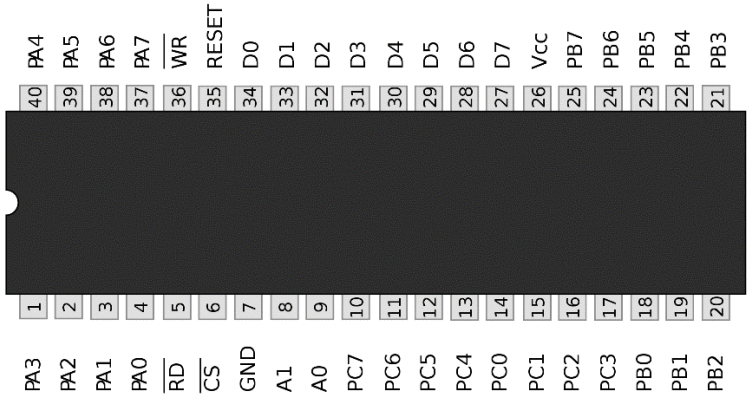
The control word register can be both written and read as shown in the address decode table in the pin descriptions. Figure 6 shows the control word format for both Read and Write operations. When the control word is read, bit D7 will always be a logic "1", as this implies control word mode information.

Ports A, B, and C

The 8255A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or ''personality'' to further

enhance the power and flexibility of the 82C55A.
Port A: One 8-bit data output latch/buffer and one 8-bit input latch buffer. Both ‘‘pull-up’’ and ‘‘pulldown’’ bus hold devices are present on Port A.
Port B: One 8-bit data input/output latch/buffer. Only ‘‘pull-up’’ bus hold devices are present on Port B.
Port C: One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B. Only “pull-up” bus hold devices are present on Port C.

8255 Pin Configuration



Working Modes of 8255

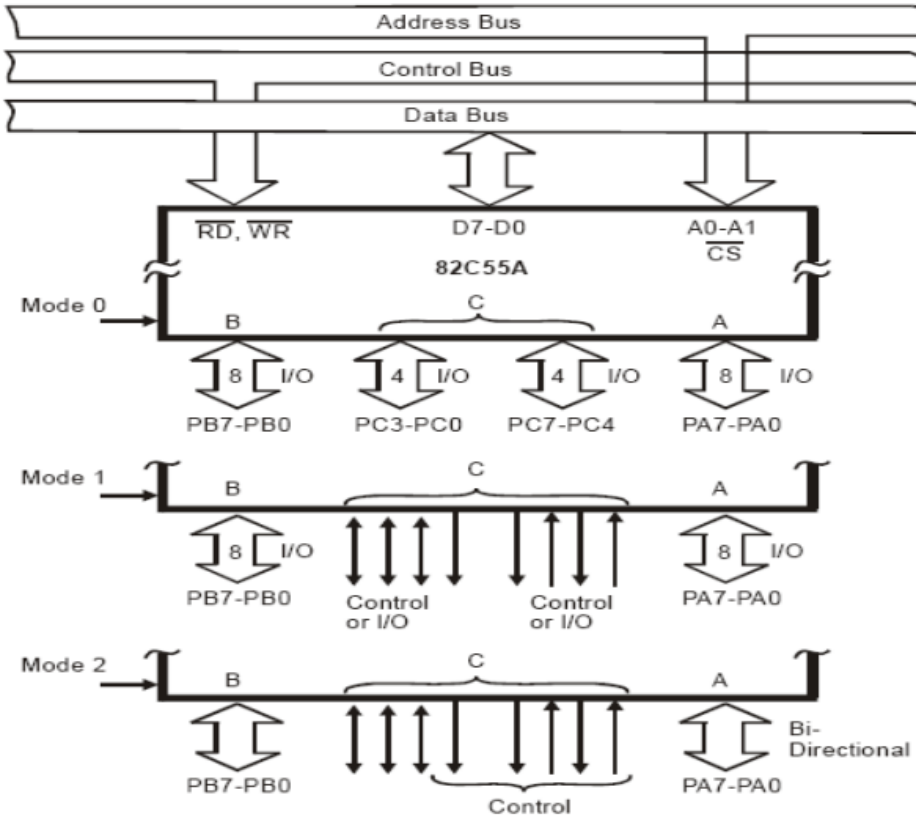
Mode Selection;

There are three basic modes of operation that can be selected by the system software:

Mode 0 – Basic Input/Output.
Mode 1 – Strobed Input/Output.
Mode 2 – Bi-Directional Bus.

A1	A2	Selects
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Control

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions.
For instance: Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard on an interrupt-driven basis.



Working Modes of 8255

Control Word for 8255 PPI

D7	D6	D5	D4	D3	D2	D1	D0
Mode set Flag	**GROUP A**				**GROUP B**		
	Mode selection		**Port A**	**Port CL**	**Mode selection**	**Port B**	**Port CU**
1 = Active	00 = Mode 0		1=input	(PC7-PC4)	Selection	1=input	(PC3-PC0)
	01 = Mode 1		0=output	1=input	0=Mode 0	0=output	1=input
	1X = Mode 2			0=output	1=Mode 1		0=output

BSR Mode for 8255

Single Bit Set/Reset Feature;

Any of the eight bits of Port C can be **Set** or **Reset** using a single Output instruction. This feature reduces software requirements in control based applications. operation just as if they were output ports.
When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset.

BSR Control Word

D7	D6	D5	D4	D3			D2				D1		D0
Bit Set/Reset Flag				**Bit Select**									**Bit Set/Reset**
0 = Active	x	x	x	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**		1=Set
	Don't Care			0	1	0	1	0	1	0	1	**B0**	0=Reset
				0	0	1	1	0	0	1	1	**B1**	
				0	0	0	0	1	1	1	1	**B2**	

Operation of Different Modes

Mode 0: (Basic Input/Output):

This functional configuration provides simple input and output operations for each of the three ports. No handshaking is required, data is simply written to or read from a specific port.

Basic Functional Definitions:

- ✓ Two 8-bit ports and two 4-bit ports
- ✓ Any Port can be input or output
- ✓ Outputs are latched
- ✓ Input are not latched
- ✓ 16 different Input/output configurations possible.

Mode 1: (Strobed Input/Output):

This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "hand shaking" signals. In mode 1, port A and port B use the lines on port C to generate or accept these "hand shaking" signals.

Mode 1 Basic Function Definitions:

- ✓ Two Groups (Group A and Group B)
- ✓ Each group contains one 8-bit port and one 4-bit control/data port
- ✓ The 8-bit data port can be either input or output. Both inputs and outputs are latched.
- ✓ The 4-bit port is used for control and status of the 8-bit port.

Mode 2:

Basic Functional Definitions:

- ✓ Used in Group A only
- ✓ One 8-bit, bi-directional bus Port (Port A) and a 5-bit control Por t (Port C)
- ✓ Both inputs and outputs are latched
- ✓ The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A)

Hands-on

Okay guys, enough of theory, lets get the hands-on skills (Most Important): If you would ever say you understand microprocessors then you must show it by making them do something for you. So, let's check if we are really anywhere with skills.
Have fun!!!!!!!!

I/O Interfacing (LED's Interfaced with 8086)

Task;

Interface an 8255 chip with 8086 to work as an I/O port. Initialize port A as an output port. Write an Assembly Language Program to light an array of 8 LEDs such that one LED lights at a time from LED1 to 8 and back (Running LEDs). Simulate your work in Proteus.

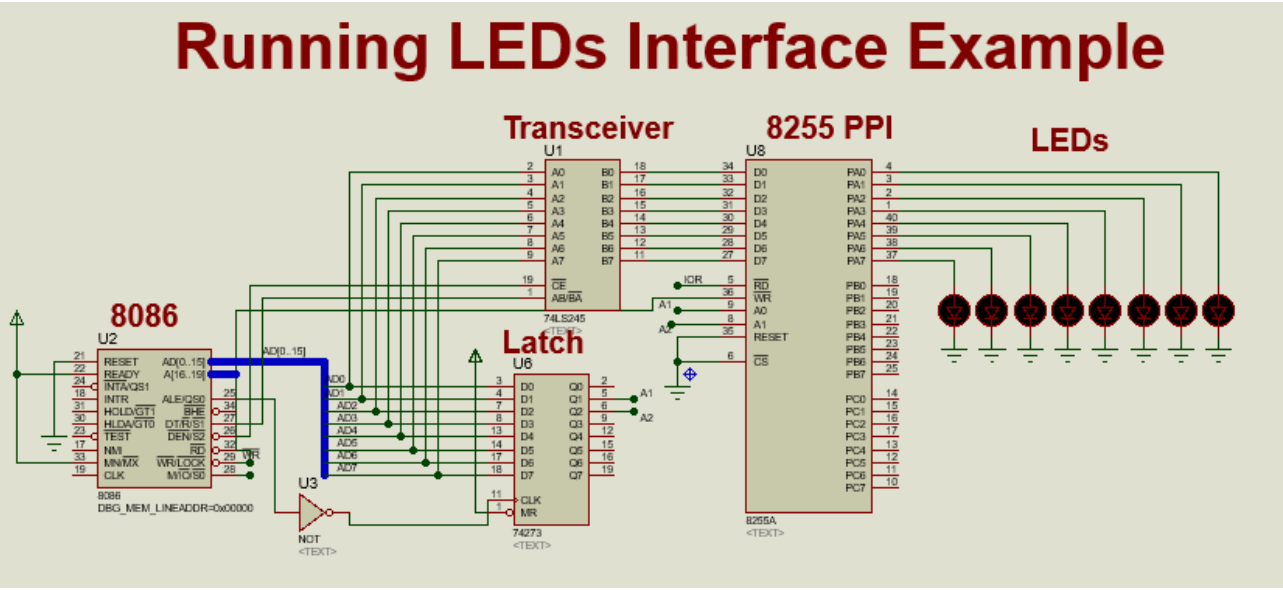
Solution:

The control word is decided as given as follows;

D7	D6	D5	D4	D3	D2	D1	D0	Control Word
Mode set Flag	GROUP A				GROUP B			
	Mode selection		*Port A*	*Port CL*	*Mode selection*	*Port B*	*Port CU*	
1	0	0	0	0	0	0	0	80H

The control word is therefore **80H.**

Circuit diagram



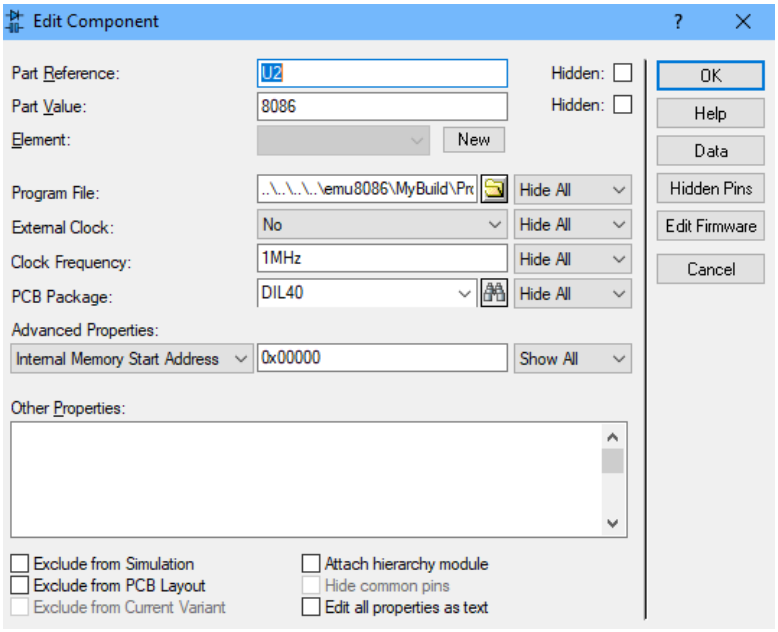
I would prefer you create the circuit yourself on Proteus (Its real important), because its then that you get the important skills. So, that you also feel the little pain of doing it, ha-ha. But its real fun.

Assembly Code

```
01
02 data segment           ; Initialize the data segment
03
04    PORTA EQU 00H
05    PORTB EQU 02H
06    PORTC EQU 04H
07    PCW   EQU 06H
08
09 ends
10
11 stack segment
12     dw   128  dup(0)
13 ends
14
15 code segment
16 start:
17 ;
18
19     mov ax, data         ; enter data to AX
20     mov ds, ax           ; move ax into cs
21     mov es, ax           ; move ax into es
22
23     MOV DX,PCW           ; move PWC to DX
24     MOV AL,10000000B     ; Control Word
25     OUT DX,AL            ; give this mode to IC I/O
26
27    Main:
28     MOV CX,8             ; loop 3 times
29     MOV AL,00000001B     ; Set LED 1 on
30
31    Right:                 ; LED Moving to the right
32     MOV DX,PORTA         ;
33     OUT DX,AL            ; Turn on LED in portA
34     SHL AX,1             ; Slide the LED Live bit to the right
35     CALL DELAY           ; Delay
36     LOOP Right           ; loop three times
37
38     MOV CX,8             ; Initialize loop counter to 8 again
39     MOV AL,10000000B     ; Bit for LED 8 turns on
40                          ; LED Moving to left
41    Left:
42     MOV DX,PORTA         ; move PORTA to DX
43     OUT DX,AL            ; Turn on the LED
44     SHR AX,1             ; Slide the LED Live bit to the right
45     CALL DELAY           ; Delay
46     LOOP Left            ;
47
48     JMP Main            ; Repeat the process from the led to the right
49
50
51     delay proc near      ; Procedure delay
52         push cx          ; hold cx
53         mov cx,2fffh     ; fill cx with delay value
54         loop $           ; looping until cx=0
55         pop cx           ; re-release cx
56         ret              ; back to the main program
57     delay endp           ; end procedure delay
58
59
60
61 end start
62
```

And there you go.
Compile the code in your emu8086 emulator to generate an executable file which you then load into the 8086 in the Proteus environment.

Still wondering how that is done? Double click the 8086 component in Proteus, a window like the one below should pop;



On the ***Program File:*** field provide a link to the executable file you compiled earlier. That's rather simple, right? Nice, then you are good. Now hit the play ***Simulate*** button at the bottom of the Proteus windows.

Woooow…Mine just worked. And yours?

Reference;

Members, Google is a good place. I wonder how we would survive without it. We would survive however. I always find help in there, you could spare a few of your MBs or even stand the cold breeze outside Assembly hall to utilize the really slow JKUAT WIRELESS STUDENTS - WiFi and search these things yourself. They are all online.

Challenge;

Well, that was nice. I think the only was of being a developer is by coming up with solution and working on problems. Just to get you going, here is a little challenge. Its not due like the assignments, so you don't have to call me *mtiaji* but one thing is that you got to be zealous. Its really simple for you guys.

Interface an 8255 chip with 8086 to work as an I/O port. Initialize port A as an output port, Port B as I/P port and Port C as O/P port. Write an Assembly Language Program to sense switch positions SW0–SW7 connected at port B. The sensed pattern is to be displayed on port A, to which 8 LED's are connected, while port C lower displays number of on switches out of the total eight switches.
Simulate your work in Proteus.

NOTE: This document and its accompanying files are also available for download in Github, a tool I recommend to you guys. The particular link will be provided.

Look forward to the next document release. I will always share any good stuff I manage to chew as long as God gives breathe. Should you find any challenges, I would be glad to help or we can Google it together. But always try finding a solution yourself first.
mogsobd@gmail.com

NEXT Release: Keypad Interface

HAVE FUN MEMBERS!!!!!