

# Microprocessor Project Report

CSHT-503

# Traffic Light

*Interfacing 82C55 to implement traffic light using 8086 Kit*

**Submitted to :**

Ms. Venika Gupta

**Presented By:**

1. **Charu Munakhiya**( 14066270010 )
2. **Deepti Sharma**( 14066270012 )
3. **Mansi Shukla**( 14066270024 )
4. **Nikita Kumari**( 14066270032 )

# Acknowledgment

We are thankful to our respected lecturer

**Ms. Venika Gupta**

for guiding us to the best of her knowledge.

We are also extend our heartfelt gratitude towards

the lab staff for providing us with all basic kits( 8086,8255)

and guiding us throughout our project.

# Certificate

This is to certify that the students have prepared the report on the project

titled

***“Interfacing 82C55 to implement traffic light with 8086 Kit”***.

The report is the result of their effort and endeavors. The report is found worthy of acceptance as the final project report for the subject of

***Microprocessor***

at graduation level.

They have prepared the report under my guidance and

have

***successfully***

accomplished the task.

|  |
| --- |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Ms. Venika Gupta** |
| Department Of Electronics |

**Index**

|  |  |
| --- | --- |
| **HEADING** | **PAGE NUMBER** |
| *Title* | (i) |
| *Acknowledgement* | (ii) |
| *Certificate* | (iii) |
| *Index* |  |
| *Aim* | 1 |
| *Requirements* | 2 |
| *Theory* | 3 |
| *Procedure* | 11 |
| *Program and Memory Contents* | 11,12 |
| *Observations and Photographs* | 13 |
| *Result* | 15 |

# Aim

To program

**82C55 (PPI DEVICE)**

for interfacing it with 8086 and hence transmit control words for preparing a

**Traffic light**.

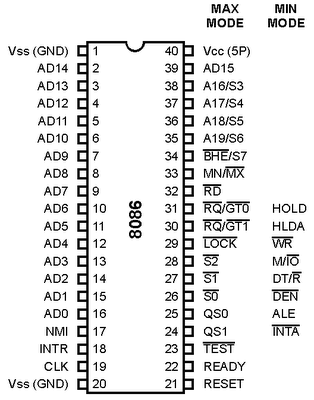
Also to study 82C55 and 8086 in theory as well as practice on trainer kit.

# Requirements

1. Battery
2. 8086 Trainer Kit
3. 8SC55 Trainer Kit
4. Dyna Bus
5. Keyboard

# Theory

# 8086 Microprocessor:

8086 is a 16-bit microprocessor with 16-bit data bus.

**PIN CONNECTIONS:**

**AD15–AD8:**

The 8086 address/data bus lines compose the upper multiplexed address/data bus on the 8086. These lines contain address bits A15–A8 whenever ALE is a logic 1, and data bus connections D15–D8 when ALE is logic 0. These pins enter a high-impedance state when a HLDA occurs.

**A19/S6–A16/S3:**

The address/status bus bits are multiplexed to provide address signals A19–A16 and also status bits S6–S3. These pins also attain a high-impedance state during the hold acknowledge.

**RD:**

Whenever the read signal is logic 0, the data bus is receptive to data from the memory or I/O devices connected to the system. This pin floats to its high-impedance state during a hold acknowledge.

**READY:**

The READY input is controlled to insert wait states into the timing of the microprocessor. If the READY pin is placed at a logic 0 level, the microprocessor enters into wait states and remains idle. If the READY pin is placed at a logic 1 level, it has no effect on the operation of the microprocessor.

**INTR:**

Interrupt request is used to request a hardware interrupt. If INTR is held high when IF = 1, the 8086/8088 enters an interrupt acknowledge cycle (becomes active) after the current instruction has completed execution.

**TEST:**

The Test pin is an input that is tested by the WAIT instruction. If is a logic 0, the WAIT instruction functions as an NOP and if is a logic 1, the WAIT instruction waits for to become a logic 0. The pin is most often connected to the 8087 numeric coprocessor.

**NMI:**

The non-maskable interrupt input is similar to INTR except that the NMI interrupt does not check to see whether the IF flag bit is logic 1

**RESET**:

The reset input causes the microprocessor to reset itself.

**CLK**:

The clock pin provides the basic timing signal to the microprocessor.

**VCC:**

This power supply input provides a +5.0 V, ±10 % signal to the microprocessor.

**GND:**

The ground connection is the return for the power supply. The 8086 microprocessors have two pins labeled GND—both must be connected to ground for proper operation.

**MN/MX**:

The minimum/maximum mode pin selects either minimum mode or maximum mode operation for the microprocessor. If minimum mode is selected, the MN/MX pin must be connected directly to +5.0 V.

**BHE/S7:**

The bus high enable pin is used in the 8086 to enable the most-significant data bus bits (D15–D8) during a read or a write operation. The state of S7 is always a logic 1.

**Minimum Mode Pins:**

**M/IO:**

The M/IO (8086) pin selects memory or I/O. This pin indicates that the microprocessor address bus contains either a memory address or an I/O port address. This pin is at its high-impedance state during a hold acknowledge.

**WR:**

The write line is a strobe that indicates that the 8086 is outputting data to a memory or I/O device. During the time that the is a logic 0, the data bus contains valid data for memory or I/O. This pin floats to a high impedance during a hold acknowledge. **INTA:**

The interrupt acknowledge signal is a response to the INTR input pin. The pin is normally used to gate the interrupt vector number onto the data bus in response to an interrupt request.

**ALE:**

Address latch enable shows that the 8086 address/data bus contains address information. This address can be a memory address or an I/O port number.

**DT/ R:**

The data transmit/receive signal shows that the microprocessor data bus is transmitting or receiving data. This signal is used to enable external data bus buffers.

**DEN:**

Data bus enable activates external data bus buffers.

**HOLD:**

The hold input requests a direct memory access (DMA). If the HOLD signal is a logic 1, the microprocessor stops executing software and places its address, data, and control bus at the high-impedance state.

**HLDA:**

Hold acknowledge indicates that the 8086 has entered the hold state.

**SS0:**

The status line is equivalent to the S0 pin in maximum mode operation of the microprocessor. This signal is combined with IO/M and DT/R to decode the function of the current bus cycle.

**Maximum Mode Pins**:

In order to achieve maximum mode for use with external coprocessors, connect the MN/MX pin to ground.

**S2, S1, S0:**

The status bits indicate the function of the current bus cycle. These signals are normally decoded by the 8288 bus controller. These lines are bidirectional and are used to both request and grant a DMA operation.

**LOCK:**

The lock output is used to lock peripherals off the system. This pin is activated by using the LOCK: prefix on any instruction.

**QS1 and QS0:**

The queue status bits show the status of the internal instruction queue. These pins are provided for access by the numeric coprocessor (8087).

**82C55: PPI DEVICE**

The *82C55 programmable peripheral interface (PPI)* is known for the following key points:

1. It is a low-cost interfacing component.

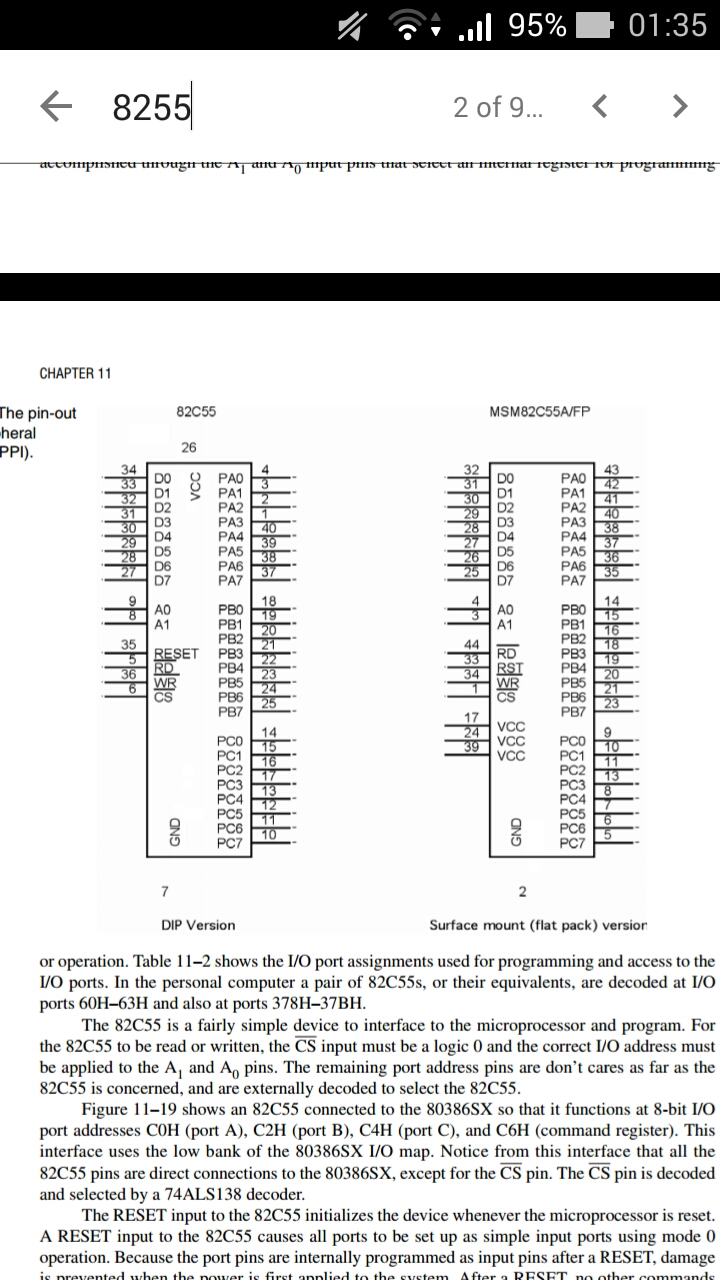
2. Has 24 pins for I/O which are programmable in groups of 12 pins

3. The groups can operate in three distinct modes of operation.

4. It can interface any TTL-compatible I/O device to the microprocessor.

5. It is used for interface to the keyboard and the parallel printer port in many PCs, but it is found as a function within a interfacing chip set.

The chip set also controls the timer and reads data from the keyboard interface.

**PIN CONNECTIONS:**

**PORTS (PA7-PAO; PB7-PB0; PC7-PC0):**

Its three I/O ports (labeled A, B, and C) are programmed as groups.

1. Group A consists of port A (PA7–PA0) and the upper half of port C (PC7-PC4).
2. Group B consists of port B (PB7–PB0) and the lower half of port C (PC3–PC0).

**A1 & A0:**

A1 and A0 input pins select an internal register for programming or operation. For the 82C55 to be read or written the input must be logic 0 and the correct I/O address must be applied to the A1 and A0 pins. The remaining port address pins are don’t cares as far as the 82C55 is concerned, and are externally decoded to select the 82C55.

**RESET:**

The RESET input to the 82C55 initializes the device whenever the microprocessor is reset. A RESET input to the 82C55 causes all ports to be set up as simple input ports using mode 0 operations.

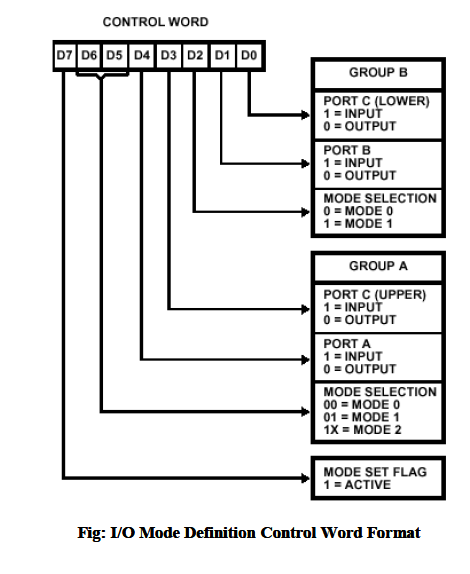
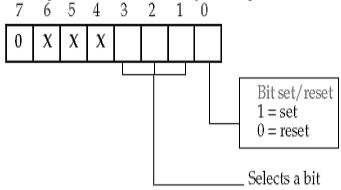
**PROGRAMMING 82C55:**

**COMMAND BYTES:**

The bit position 7 selects either command byte A or command byte B.

1. Command byte A programs the function of group A and B.
2. Command byte B sets (1) or resets (0) bits of port C only if the 82C55 is programmed in mode 1 or 2.The bit set/reset feature is often used in a control system to set or clear a control bit at port C. The bit set/reset function is glitch-free, which means that the other port C pins will not change during the bit set/reset command.

|  |  |  |
| --- | --- | --- |
| **A1** | **A0** | **Function** |
| **0** | **0** | **Port A** |
| **0** | **1** | **Port B** |
| **1** | **0** | **Port C** |
| **1** | **1** | **Command register** |

****

# Describing the control word: 80H

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |

**Mode set:** 1; that is; this is the command byte A and will be used to configure all the ports.

**GROUP A:**

**Mode Selection:** 00 which implies mode 0 is selected. In mode 0, simple I/O takes place through the Port A and also the upper half of Port C.

**Port A:** 0 which means it is an output port.

**Port C (Upper):** 0 which means it is an output port.

**GROUP B:**

**Mode Selection**: 0 means mode 0 is selected for group B pins, again which indicates a simple I/O.

**Port B:** 0 which means it is an output port.

**Port C (lower):** 0 which implies it is an output port.

**Traffic Light Program explained:**

The program we have developed indicates a red light on Port B, green light on Port A and yellow light on Port C.(according to day to day life) But here, the red light signal indicates to go the user and the green light indicates to stop and yellow light indicates that a wait of few cycles is left.

We have identified four directions, north way, east way, south way, west way. At a time only one side’s traffic is allowed to move (red) the other three have to stop (green). After a certain amount of delay (delay procedure) and then one of the sides (north, east, south, west) receives a yellow signal(whichever is the next clockwise). When one of them receives a yellow signal, again a delay (delay2 procedure) is added which is of lesser time, finally when the delay gets over , yellow light switches off and green light is switched on to indicate that side to go and the other one is stopped.

**Counting Delay :**

The code segment that causes the longer delay is as follows:

|  |  |  |
| --- | --- | --- |
| **DELAY:** | **Clock Cycles** | **Cumulative Total** |
| PUSH CX | 11 | 11 |
| MOV CX, 000FH | 4 | 15 |
| A2: |  |  |
| MOV DX, FFFFH | 4\*15 | 75 |
| A3: |  |  |
| DEC, DX | (2\*65536)\*15 | 1966155 |
| JNZ A3 | (16\*65536+4)\*15 | 17694855 |
| DEC CX | 2\*15 | 17694885 |
| JNZ A2 | 16\*15+4 | 17695129 |
| POP CX | 8 | 17695137 |
| RET | 8 | 17695145 |

|  |  |
| --- | --- |
| **TOTAL CLOCK STATES** | 17695145 |
| **THE FREQUENCY APPLIED** | 24 MHz |
| **CLOCK** | 8 MHz |
|  | 1/8\*106 seconds |
| **TIME DELAY** | 17695145/8\*106 |
|  | **2.212 seconds** |

The code segment that causes the smaller delay is as follows:

|  |  |  |
| --- | --- | --- |
| **DELAY2:** | **Clock Cycles** | **Cumulative Total** |
| PUSH CX | 11 | 11 |
| MOV CX, 0005H | 4 | 15 |
| A4: |  |  |
| MOV DX, FFFFH | 4\*5 | 35 |
| A5: |  |  |
| DEC, DX | (2\*65536)\*5 | 655395 |
| JNZ A5 | (16\*65536+4)\*5 | 5898295 |
| DEC CX | 2\*5 | 5898305 |
| JNZ A4 | 16\*5+4 | 5898389 |
| POP CX | 8 | 5898397 |
| RET | 8 | 5898405 |

|  |  |
| --- | --- |
| **TOTAL CLOCK STATES** | 5898405 |
| **TIME DELAY** | 5898405/8\*106 |
|  | **0.737 seconds** |

**Procedure**

1. Connect battery with the power supply of 8086 (VCC).
2. Connect the keyboard with 8086 kit port.
3. Connect the dyna bus with 8086 and 8255.
4. Finally switch it on. Now press A and enter the address to feed the program.
5. When you complete your program, press U to un-assemble and check your program.
6. Now press E to edit the memory location to feed any data into the memory.
7. Finally press G to run the program.

**PROGRAM:**

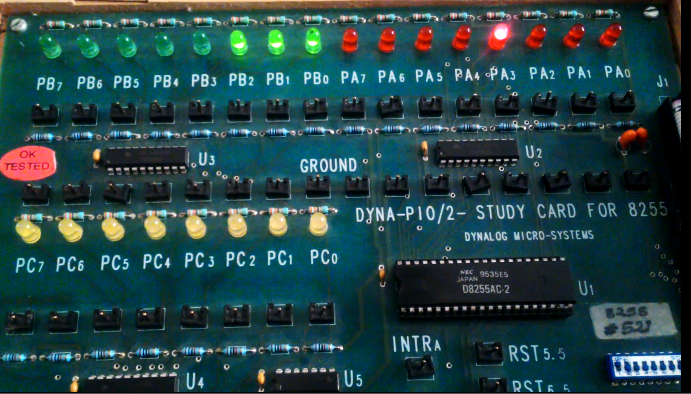
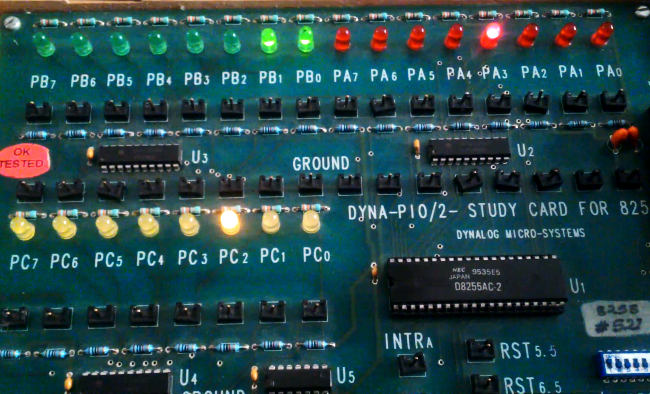
|  |  |
| --- | --- |
| **1100** | MOV BX , 1200H |
| **1103** | MOV AL, [BX] |
| **1105** | MOV DX, 33 |
| **1108** | OUT DX,AL |
| **1109** | A6:  MOV CX, 0004H |
| **110C** | MOV BX, 1201H |
| **110F** | A1:  MOV AL, [BX] |
| **1111** | MOV DX ,31H |
| **1114** | OUT DX, AL |
| **1115** | INC BX |
| **1116** | MOV AL, [BX] |
| **1118** | MOV DX, 32H |
| **111B** | OUT BX, AL |
| **111C** | INC BX |
| **111D** | MOV AL, [BX] |
| **111F** | MOV DX, 30H |
| **1122** | OUT DX, AL |
| **112B** | INC BX |
| **1124** | CALL DELAY |
| **1127** | MOV AL, [BX] |
| **1129** | MOV DX, 31H |
| **112C** | OUT DX, AL |
| **112D** | INC BX |
| **112E** | MOV AL,[BX] |
| **1130** | MOV DX, 32H |
| **1133** | OUT DX, AL |
| **1134** | INC BX |
| **1135** | CALL DELAY2 |
| **1138** | DEC DX |
| **1139** | JNZ A1 |
| **113B** | JMP A6 |
| **113D** | INT 3 |
| **113E** | DELAY :  PUSH CX |
| **113F** | MOV CX , 000FH |
| **1142** | A2:  MOV DX, FFFFH |
| **1145** | A3:  DEC DX |
| **1146** | JNZ A3 |
| **1148** | DEC CX |
| **1149** | JNZ A2 |
| **114B** | POP CX |
| **114C** | RET |
| **114D** | DELAY 2:  PUSH CX |
| **114E** | MOV CX , 0005H |
| **1151** | A4:  MOV DX, FFFFH |
| **1154** | A5:  DEC DX |
| **1155** | JNZ A5 |
| **1157** | DEC CX |
| **1158** | JNZ A4 |
| **115A** | POP CX |
| **115B** | RET |

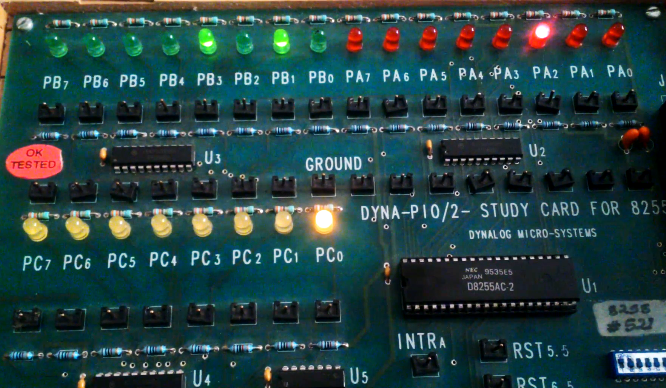
**LOOKUP TABLE:**

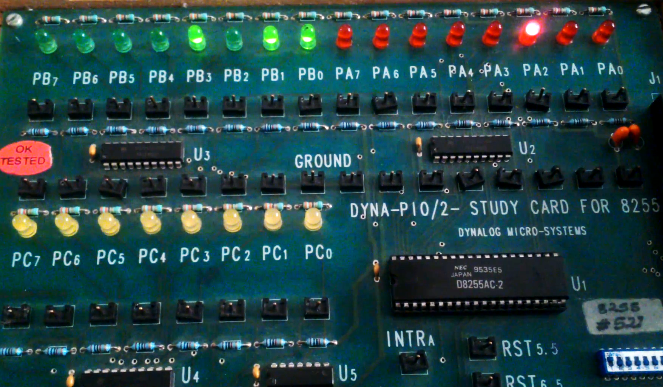
|  |  |
| --- | --- |
| **1200** | 80 |
| **1201** | 0E 00 01 0C |
| **1205** | 02 0D 00 02 |
| **1209** | 05 08 07 00 |
| **120D** | 08 03 04 0B |
| **1211** | 00 04 0A 01 |

**Observations and Photographs**

1. **NORTH WAY**



1. **EAST WAY**



We observe that when we run the program the 8255 first get initialized and then starts giving the output on port A, B and C.

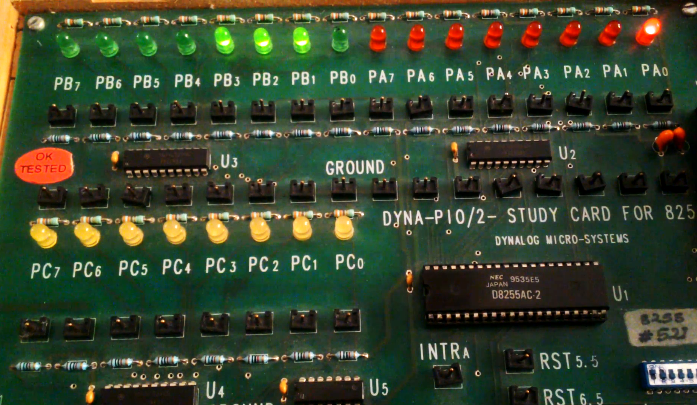
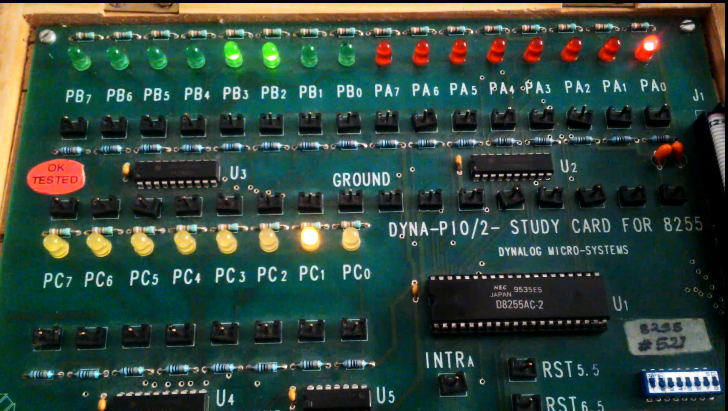
When a certain direction receives the signal to move the other three get a stop signal from port B.

After a delay of merely 2 seconds, yellow light switches on port for the direction which comes next clockwise.

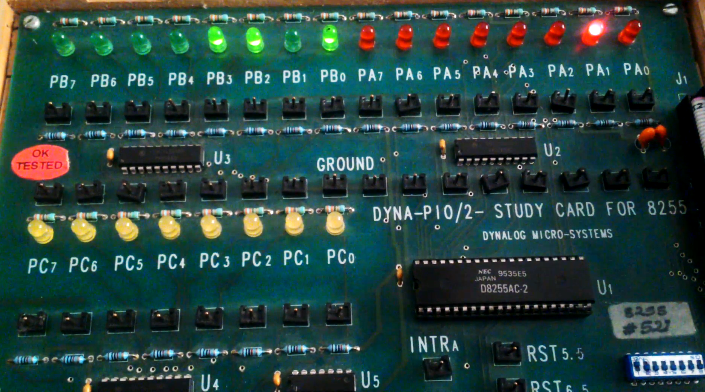
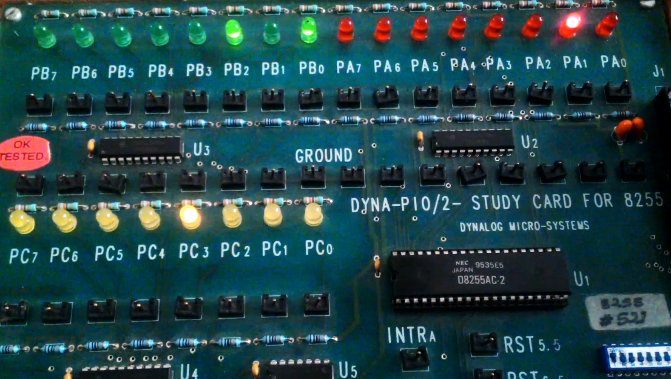
Finally the next direction receives the go signal and the yellow signal goes off and the previous direction receives a stop.

The program runs infinitely.

1. **SOUTH WAY**



1. **WEST WAY**



**Result**

**And**

**Possible developments using the program**

**Result:**

* Traffic light using 82C55 interfacing was successfully developed.
* 82C55 was successfully interfaced with the 8086 kit.

**Possible Developments:**

* We can increase the delay time a little more to give the feel and look of the real world traffic light system
* We can design more type of light designs like Christmas lights etc using this kind of programming
* We can interface some other input and output with 82C55 and try to manipulate the device through programming