International Journal of Latest Trends in Engineering and Technology
Vol.(9)Issue(2), pp.025-034
DOI: http://dx.doi.org/10.21172/1.92.06
e-ISSN:2278-621X

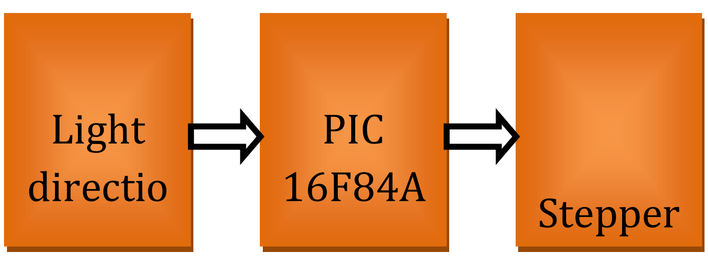
SOLAR TRACKING SYSTEM USING PIC16F84A STEPPER MOTOR AND 555TIMER

Amey Arvind Madgaonkar[1], Sumit Dhere[2] & Rupesh Ratnakar Kadam[3]

Abstract-Sometimes we come across such a situation where we need to move an object in response to the direction of light. E.g.to track the solar rays to increase the efficiency of power generation through solar panel. Keeping this idea of light direction sensing in the mind, we designed a device which tracks the light source. Today, microcontrollers have become an integral part of all automatic & semi-automatic machines. The architectural changes in instrumentations & control systems were & are due to the computing & communication capability of the microcontroller devices. In this paper, we choose 16F84A which is a microcontroller of PIC (Peripheral Interface Controller) family. It is programmed so as to move the stepper motor which holds the sensors (LDRs) to sense the intense light source and move it accordingly. The exception of the stepper motor is that; it only operates under software control. It needs a specific computer program to complete program to complete its operation. This main requirement of the stepper motor can be overcome by using PIC 16F84A. To sense the light source, we have used a photo sensor i.e. Light Dependent Resistor (LDR). When the LDR gets exposed to the light source, its resistance decreases further that leads to decrease in the voltage across it. This change in the status of LDR is sensed by the IC 555 timer circuit which is an interfacing circuit between LDR and PIC 16F84A. Here, IC 555 timer circuit obtains its input from the LDR and converts it into a digital output signal which serves as an input for PIC 16F84A. According to change in this input signal; PIC 16F84A decides the movement of the stepper motor. But in case, the source is not available then it will rotate the stepper motor in clockwise and anticlockwise direction from its reference position. This operation is called as search mode operation of the device.

KEYWORDS:LDR,PIC16F84A,stepper moter,555timer

1. Block diagram



The system can be divided in 3 main blocks:

1) Light direction sensing unit.
2) PIC 16F84A microcontroller.
3) Stepper motor.

The PIC is programmed to sense the movement of light source & manipulate the stepper motor as per the digital signal acquired from the light direction sensing unit.

2. LIGHT DIRECTION SENSING UNIT

For light sensing purpose, we have designed a circuit using light dependent resistor (LDR) as a light sensor followed by 555 timer IC which converts the output of LDR into digital signal. This unit mainly works into two parts. A] Light Dependent Resistor (LDR) B] IC 555 timer

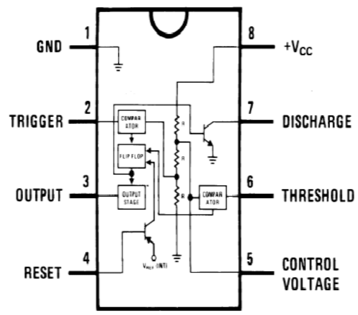
2.1 Light Dependent Resistor (LDR)

1 Humanities and basic science department, Lecturer, Yashwantrao Bhonsale polytechnic, Sawantwadi.
2 Physics department
3 Humanities and basic science department, First year academic co-coordinator, Yashwantrao Bhonsale polytechnic, Sawantwadi.

A photo resistor, light dependent resistor (LDR) or cadmium sulfide **(**CdS**)** cell is a resistor whose resistance decreases with increasing incident light intensity. It can also be referred to as a photoconductor.

2.2 IC 555 timer

IC 555 is a highly stable timing circuit or oscillator. In the timer mode of operation, the time is precisely controlled by one external resistor and capacitor. In stable oscillator mode, the free running frequency and the duty cycle are both accurately controlled with two external resistors and one capacitor



LM555 Timer

Figure1.The functional block diagram of the IC 555 timer is as shown above.

There is a potential divider 5kΩ, 5kΩ, 5kΩ which provide fixed reference voltage to the comparator in the IC .Because of internal potential divider.

$V_{CONTROL}$=2/3V_{CC}

CONTROL terminal is the inverting input of the upper comparator. Non-inverting terminal to the lower comparator 1/3V_{CC}. Output of upper comparator is HIGH drives SET input of flip flop. Output of lower comparator drives the RESET input of flip flop. Q_{output} of flip flop drives the discharge transistor which acts as a switch. When the flip flop is SET Q is HIGH. The transistor is ON and provides path for the capacitor (connected externally) to discharge. When flip flop is RESET Q is LOW. The transistor is OFF and the capacitor is allowed to charge.

The Q output of flip flop provides the output of timer IC. When RESET (pin4) is grounded it inhabits the operation of IC. This ON-OFF feature is sometimes useful .In most of application the RESET pin not used and pin 4 is tied directly to the supply voltage (pin8).

Pin 1 in the chip is GROUND. Pin 6 is NON-INVERTING input of UPPER comparator. Voltage across timing capacitor is provided at this pin. It is called as THRESHOLD. Pin 5 is the CONTROL pin. It is INVERTING input of UPPER comparator .Due to internal potential $V_{control}$= 2/3V_{CC} .

To modify CONTROL voltage externally the CONTROL pin is made available. Pin 2 is TRIGGER input. It is INVERTING input of LOWER comparator .TRIGGER is provided at this pin. The TRIGGER is negative one and the TRIGGER voltage should fall below [$V_{CONTROL}$/2]. PIN 7 is the collector of discharge transistor .The 555 timer works with any supply voltage between 4.5V to 16V.

3. PIC 16F84A MICROCONTROLLER :

The PIC16F84A belongs to the mid-range family of the PIC microcontroller devices.

The program memory contains 1K words, which translates to 1024 instructions, since each 14-bit program memory word is the same width as each device instruction. The data memory (RAM) contains 68 bytes. Data EEPROM is 64 bytes.There are also 13 I/O pins that are user-configured on a pin-to-pin basis. Some pins are multiplexed with other device functions. These functions include External interrupt ,Change on PORTB interrupts, Timer0 clock input

4. STEPPER MOTOR

The stepper motor is an electromagnetic device that converts digital pulses into mechanical shaft rotation. The shaft or spindle of a stepper motor rotates in discrete step increments when electrical command pulses are applied to it in the proper sequence. The sequence of the applied pulses is directly related to the direction of motor shafts rotation. The speed of the motor shafts rotation is directly related to the frequency of the input pulses and the length of rotation are directly related to the number of input pulses applied. Many advantages are achieved using this kind of motors, such as higher simplicity, since no brushes or contacts are present, low cost, high reliability, high torque at low speeds, and high accuracy of motion. Many systems with stepper motors need to control the acceleration/ deceleration when changing the speed.

5. THE ROTATING MAGNETIC FIELD

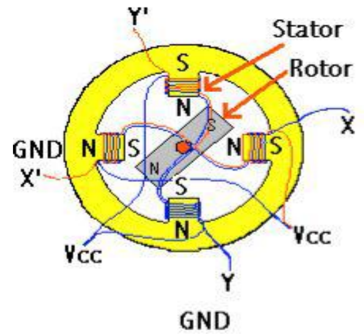


Figure2.Magnetic field rotation in Stepper Motor

When a phase winding of a stepper motor is energized with current a magnetic flux is developed in the stator. The direction of this flux is determined by the "Right Hand Rule" .The rotor then aligns itself so that the flux opposition is minimized. In this case the motor would rotate clockwise so that its south pole aligns with the north pole of the stator and its north pole aligns with the south pole of stator. To get the motor to rotate we must provide a sequence of energizing the stator windings in such a fashion that provides a rotating magnetic flux field which the rotor follows due to magnetic attraction.

5.1 Working

The LDR is used as the light source. To sense the direction of light , we have used a pair of LDRs with an obstacle in between them, so that when the source changes its position, one of the LDRs comes under shadow. As a result its resistance increases.

The 555 timer circuits are also designed separately for these two LDRs. When one of the LDRs becomes inactive due to movement of the source, voltage across it, which is the voltage of pin 2 and 6 of the IC, gets increased. When it crosses 2/3 rd VCC, the output of the Threshold comparator becomes high which sets the flip-flop. As a result output of the IC goes low.

The outputs of the two timer ICs are connected to pins RB0 and RB1 of the PIC. The software gets the information about the movement of the source by reading the output of the light direction sensing unit. When the source moves towards the left of the sensor, the right sensor becomes inactive lowering the voltage of the pin RB0. The software reads this input and moves the stepper motor anticlockwise to get the source. Similarly when the source moves towards the right side it is sensed by the software and the stepper motor is moved clockwise till both the sensors face the source.

When the source is not available to both the sensors, i.e. when both the pins RB0 and RB1 are low, the system enters in search mode. The software moves the stepper motor once clockwise and then anticlockwise to search the source. If source is still unavailable, then the stepper motor stops moving keeping its reference position. The system will again start working only when the key connected to pin RB2 of the PIC is pressed.

5.2 Software

In this system we have used pins RA0 to RA3 of PORTA of the PIC as output pins to drive the stepper motor and pins RB0 and RB1 of PORTB as input pins to read the status of both the sensors. A key is connected to pin RB2 for starting the working of the system. So at the beginning of the program we have configured PORTA as an output port and PORTB as an input port by writing proper bytes in the registers TRISA and TRISB respectively.

We are moving the stepper motor in half stepping mode. The patterns corresponding to this movement are stored at the memory locations 0x20 to 0x27 in the data memory. The data are being outputted to PORTA through the register INDF. For this purpose the file selection register (FSR) is initialized to 0x20. The 0th bit of the register 0x0f serves as direction flag by setting for clockwise movement. The stepper motor completes one rotation in 64 half steps. The hex equivalent of 64 which is 0x40 stored in the register 0x18. Initially the pattern in 0x20 is outputted to PORTA.

The software waits for the key to start working. Initially the number 0x02 is stored as rotation counter in the register 0x14. When the key is pressed, it checks whether both the pins RB0 and RB1 are high. If they are high indicating that both the sensors are facing the source, the program retains the position of the stepper motor and keeps checking continuously the status of pins RB0 and RB1.

If this is not the case, the program first checks the status of pin RB0. If it is high, indicating that the source is towards the left, the direction flag gets cleared and the subroutine for anticlockwise movement is called. At the same time the data byte 0x03 is stored at the location 0x14 as rotation counter. If RB0 is low, the status of pin RB1 is checked.

If the pin RB1 is high, the stepper motor should move clockwise in order to track the source. Before calling the subroutine for clockwise movement, the direction flag gets set and 0x03 is stored as rotation counter. If source is not available, both the input pins remain low. In this case the program enters in search mode.

In the search mode the rotation counter is 0x02.The status of the direction flag decides the direction of rotation. If source is not available even after one clockwise and one anticlockwise rotation, the system stops working till the key connected to the pin RB2 is pressed.

In the subroutine for anticlockwise movement we have first checked whether the sensors are already at the reference position. If not, the pattern counter 0x18 and FSR are increased till they become 0x08 and 0x28 respectively, every time outputting the INDF to PORTA. After completing one rotation, the counter 0x14 is decremented and direction flag is set for clockwise rotation before returning to main program.

A similar subroutine is written for clockwise rotation. The pattern counter and FSR are decremented till FSR become 0x19 instead of incremented. Also the step counter 0x18 is decremented to reach the reference position. The direction flag is cleared before returning from subroutine. A simple delay routine is written using three registers 0x10,0x11 and 0x12.

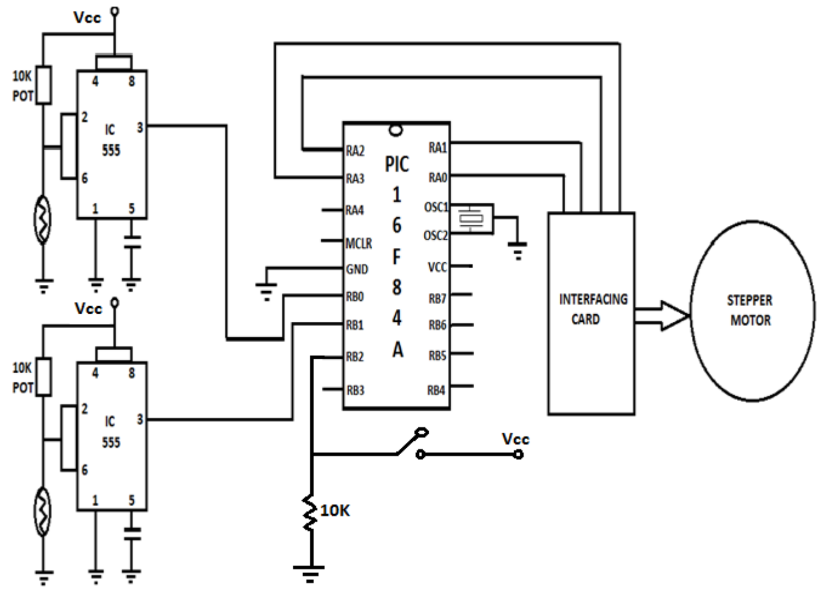


Figure3. Circuit diagram of solar tracking system using pic16f84A and stepper motor

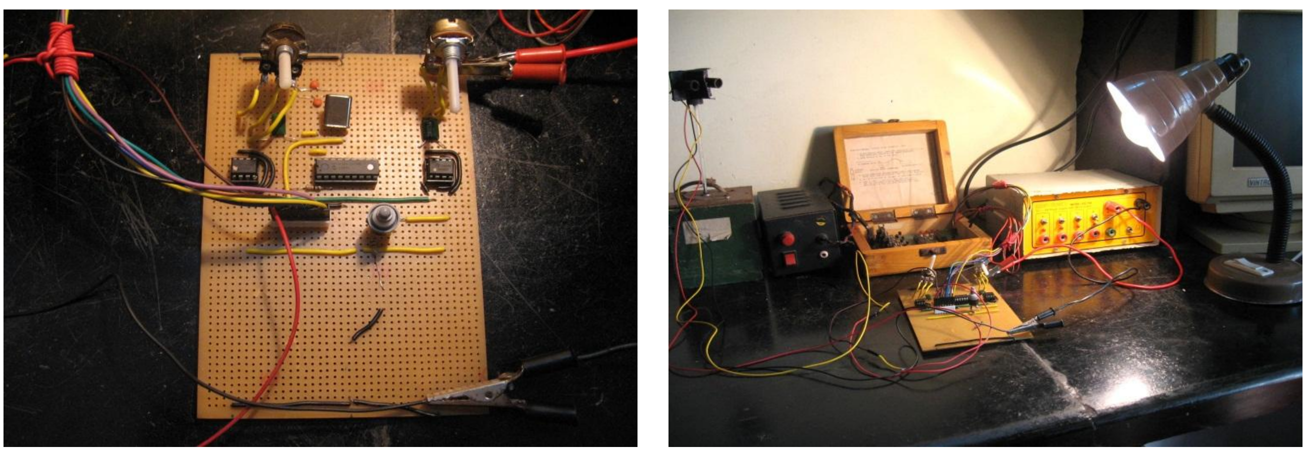
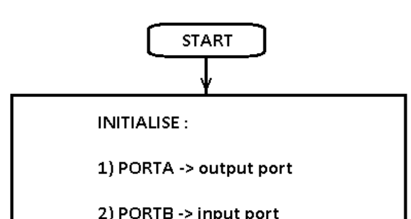
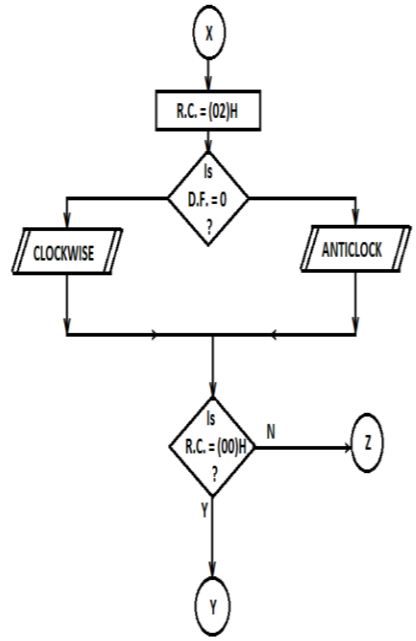


Figure4.Photograph of solar tracking system using pic16f84A , stepper motor and 555timer

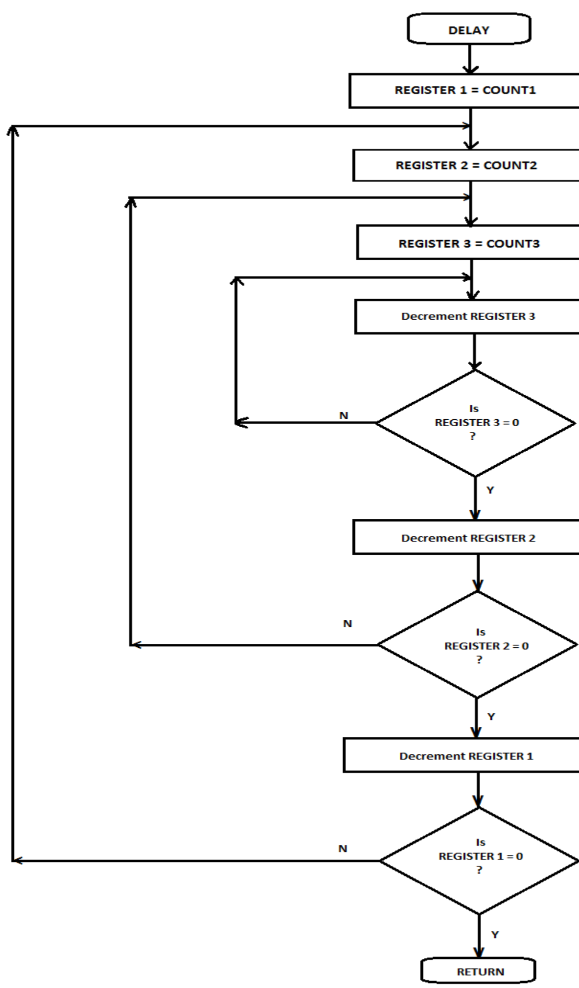
6. FLOW CHART

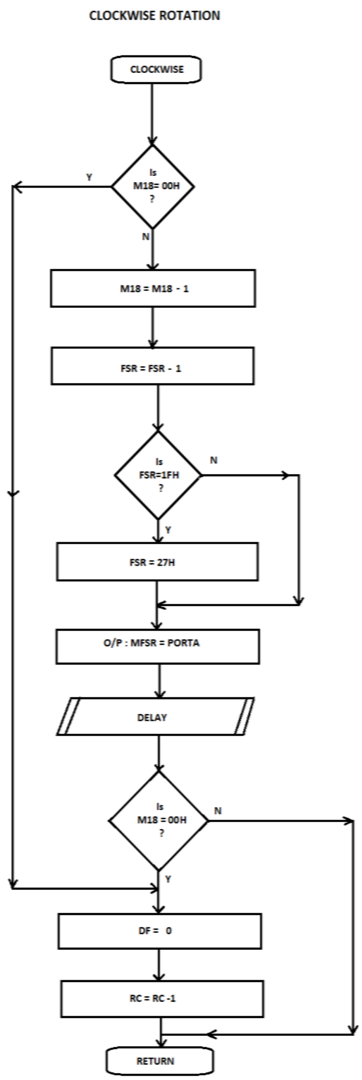
MAIN PROGRAM

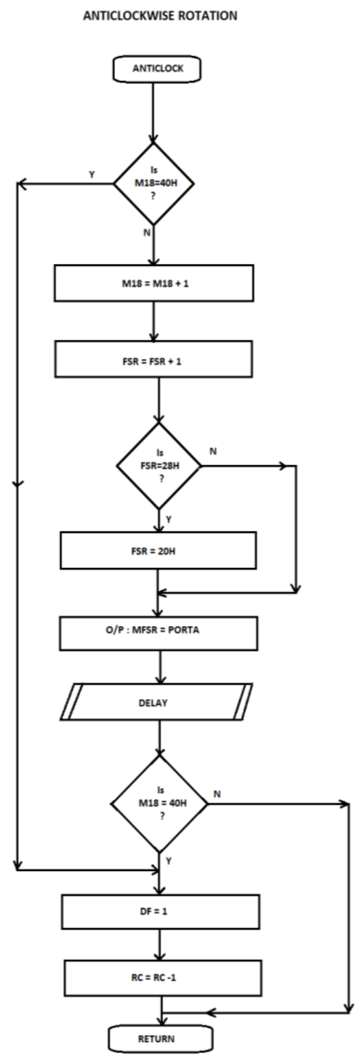




DELAY ROUTINE







Program

```
list      p=16F84A                                    ; list directive to define processor
          #include <p16F84A.inc>          ; processor specific variable definitions

          __CONFIG   _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC

;***** VARIABLE DEFINITIONS*******

STATUS          EQU 0x03
RP0             EQU 0x05
TRISB           EQU 0x86
TRISA   EQU 0x85
PORTB   EQU 0x06
PORTA   EQU 0x05
OPTON   EQU 0x81
TMR0    EQU 0x01
ITCON   EQU 0x0b
PCL             EQU 0x02
INDF    EQU 0x00
```

```
FSR             EQU 0x04
;***********************************************************

                ORG 0x000
                goto MAIN
              ORG 0x020

MAIN            bsf STATUS,RP0                  ;select bank1
                movlw 0x00
                movwf TRISA             ;configure PORTA as output port
                movlw 0xff
                movwf TRISB             ;configure PORTB as input port
                bcf STATUS,RP0                  ;select bank0
                movlw 0x02              ;save the look-up table from
                movwf 0x20              ;0x20 to 0x27
                movlw 0x06
                movwf 0x21
                movlw 0x04
                movwf 0x22
                movlw 0x0c
                movwf 0x23
                movlw 0x08
                movwf 0x24
                movlw 0x09
                movwf 0x25
                movlw 0x01
                movwf 0x26
                movlw 0x03
                movwf 0x27

                movlw 0x20
                movwf FSR                       ;initialise FSR to 0x20
                bsf 0x0f,0x00           ;set direction flag for clockwise rotation
                movlw 0x40
                movwf 0x18              ;set step count=(40)H
                movf INDF,w
                movwf PORTA             ;out pattern in INDF to PORTA
                call DELAY
KEY             btfss PORTB,0x02                ;check whether the key is pressed
                goto KEY
                movlw 0x02
                movwf 0x14              ;set the count for no. of rotations
BACK            movlw 0x03
                andwf PORTB,w           ;mask the bits 2-7 of PORTB
                movwf 0x0e
                movlw 0x03              ;check whether both the LDRS
                xorwf 0x0e,w            ;are facing the source of light
                btfss STATUS,0x02       ;if both the LDRs are not active
                goto TEST0                      ;check the status of left LDR
                movlw 0x03
                movwf 0x14              ;set the count for no. of rotation
                goto BACK                       ;keep tracking the source

TEST0   btfss PORTB,0x00                ;check whether left LDR is active
                goto TEST1                      ;check status of right LDR
                movlw 0x03
                movwf 0x14              ;set the count for no. of rotations
                bcf 0x0f,0x00           ;reset direction flag for anticlockwise
                call ANTICLOCK                  ;move anticlockwise to get the source
                goto BACK
```

```
TEST1   btfss PORTB,0x01              ;check whether right LDR is active
        goto TEST2                            ;check the direction flag
        movlw 0x03                    ;set the count for no. of rotation
        movwf 0x14
        bsf 0x0f,0x00                 ;set direction flag for clockwise rotation
        call CLOCK                            ;move clockwise to get the source
        goto BACK
        movlw 0x02                    ;set count for no.of rotation
        movwf 0x14

TEST2   btfss 0x0f,0x00       ;check whether direction flag is set
        goto LABEL
        call CLOCK
        goto TEST

LABEL   call ANTICLOCK

TEST            movlw 0x00
                xorwf 0x14,w          ;check the rotation counter is zero

                btfss   STATUS,0x02
                goto BACK
                goto KEY                      ;stop rotating if count is zero



ANTICLOCK       movlw 0x40
                xorwf 0x18,w          ;check whether reference position
                btfsc STATUS,0x02     ;is reached
                goto AHEAD
                incf 0x18,f
                incf FSR,f                    ;increment FSR
                movlw 0x28
                xorwf FSR,w
                btfss STATUS,0x02
                goto FWD
                movlw 0x20            ;reinitialise FSR to 0x20
                movwf FSR

FWD             movf INDF,w
                movwf PORTA           ;output next pattern to PORTA
                call DELAY
                movlw 0x40            ;check whether reference position
                xorwf 0x18,w          ;is reached
                btfss STATUS,0x02
                goto RET1

AHEAD   bsf 0x0f,0x00         ;set direction flag for clockwise rotation
                decf 0x14,f                   ;decrement rotation counter
RET1            return


CLOCK movlw 0x00
                xorwf 0x18,w          ;check whether reference position
                btfsc STATUS,0x02     ;is reached
                goto AHEAD1
                decf 0x18,f
                decf FSR,f                    ;decrement FSR
                movlw 0x1f
                xorwf FSR,w
                btfss STATUS,0x02
```

```
                goto FWD1
                movlw 0x27          ;reinitialise FSR to 0x27
                movwf FSR

FWD1    movf INDF,w
                movwf PORTA         ;output previous pattern to PORTA
                call DELAY
                movlw 0x00          ;check whether reference position
                xorwf 0x18,w        ;is reached
                btfss STATUS,0x02
                goto RET2

AHEAD1          bcf 0x0f,0x00       ;reset direction flag for anticlockwise
                decf 0x14,f                 ;decrement FSR
RET2            return


DELAY           movlw 0x01
                movwf 0x10
loop3           movlw 0xff
                movwf 0x11
loop2           movlw 0xff
                movwf 0x12
loop1           decfsz 0x12,F
                goto loop1
                decfsz 0x11,F
                goto loop2
                decfsz 0x10,F
                goto loop3
                return
            END                      ; directive 'end of program'
```

7. CONCLUSION

The light tracking system using PIC 16F84A is designed and tested successfully at laboratory level. This PIC is the brain of the entire tracking system, and it is programmed to detect the sunlight through the sensors and then activate the motor to position so as to obtain maximum sun light .Despite of it complexity, this device is very easy to handle & less costs as compare to the fixed panel solar tracking system.

8. REFERENCE

[1] The 8051 Microcontroller & Embedded Systems-Dr. Rajiv Kapadia (Jaico Pub.House)

[2] 8051 Micro-controller, K.J.Ayala., Penram International.

[3] Design with PIC microcontrollers by John B. Peatman, Pearson Education Asia.

[4] Programming & customizing the 8051 microcontroller By Myke Predko, TMH.

[5] Electric Machines – D.P.Kothari, I.J. Nagrath – Tata Mc.Graw Hill Education

[6]PIC16F84A Data Sheet 18-pin Enhanced FLASH/EEPROM 8-bit Microcontroller