Solar Tracker

Bill Lane*
Department of Electrical and Computer Engineering
Cleveland State University
Cleveland, Ohio 44115

EEC 517
April 30, 2008

* welane3@hotmail.com or wlane@republicengineered.com

Abstract

Solar energy is rapidly gaining notoriety as an important means of expanding renewable energy resources. As such, it is vital that those in engineering fields understand the technologies associated with this area. My project will include the design and construction of a microcontroller-based solar panel tracking system. Solar tracking allows more energy to be produced because the solar array is able to remain aligned to the sun. This system builds upon topics learned in this course. A working system will ultimately be demonstrated to validate the design. Problems and possible improvements will also be presented.

1. Introduction

Renewable energy solutions are becoming increasingly popular. Photovoltaic (solar) systems are but one example. Maximizing power output from a solar system is desirable to increase efficiency. In order to maximize power output from the solar panels, one needs to keep the panels aligned with the sun. As such, a means of tracking the sun is required. This is a far more cost effective solution than purchasing additional solar panels. It has been estimated that the yield from solar panels can be increased by 30 to 60 percent by utilizing a tracking system instead of a stationary array [1]. This project develops an automatic tracking system which will keep the solar panels aligned with the sun in order to maximize efficiency.

This paper begins with presenting background theory in light sensors and stepper motors as they apply to the project. The paper continues with specific design methodologies pertaining to photocells, stepper motors and drivers, microcontroller selection, voltage regulation, physical construction, and a software/system operation explanation. The paper concludes with a discussion of design results and future work.

2. Background Information

This section presents background information on the main subsystems of the project. Specifically, this section discusses photocell and stepper motor theory in order to provide a better understanding as to how they relate to the solar tracker.

2.1. Light Sensor Theory

Light sensors are among the most common sensor type. The simplest optical sensor is a photoresistor which may be a cadmium sulfide (CdS) type or a gallium arsenide (GaAs) type [2]. The next step up in complexity is the photodiode followed by the phototransistor [2].

The sun tracker uses a cadmium sulfide (CdS) photocell for light sensing. This is the least expensive and least complex type of light sensor [2]. The CdS photocell is a passive component whose resistance in inversely proportional to the amount of light intensity directed toward it. To utilize the photocell, it is placed in series with a resistor. A voltage divider is thus formed and the output at the junction is determined by the two resistances. Figure 1 illustrates the photocell circuit. In this project, it was desired for the output voltage to increase as the light intensity increases, so the photocell was placed in the top position.

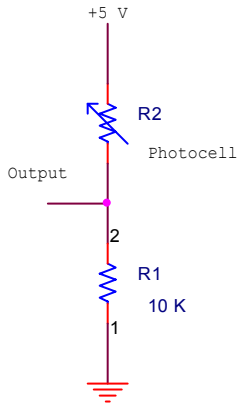

Figure 1 – CdS Photocell Circuit

2.2. Stepper Motor and Driver Theory

Stepper motors are commonly used for precision positioning control applications. All stepper motors possess five common characteristics which make them ideal for this application. Namely, they are brushless, load independent; have open loop positioning capability, good holding torque, and excellent response characteristics. [3].

There are three types of stepper motors: permanent magnet, variable reluctance, and hybrid [3]. The arrangement of windings on the stator is the main distinguishing factor between the three types [3]. Permanent magnet motors may be wound either with unipolar or bipolar windings [3].

The sun tracker uses a unipolar step motor. As such, discussion will be limited to this type of stepper motor. Unipolar motors have two windings with each having a center tap as shown in Figure 2 from [4].

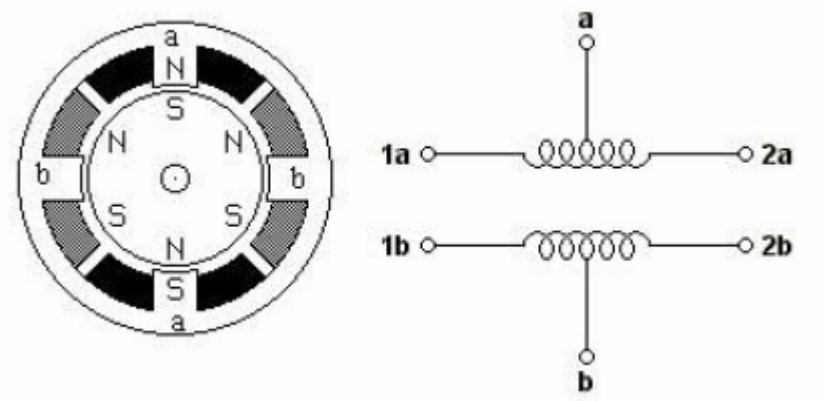

Figure 2 – Unipolar Stepper Motor Coils

The center taps are connected to a positive voltage while the coil ends are alternately grounded to cause a reversal of the field direction in that winding [3]. Figure 2 shows a 4-phase motor. The number of phases is equal to two times the number of coils. The motor is rotated by applying power to the windings in a sequence as shown in Figure 3 from [4].

Clockwise Rotation ➡

Index	1a	1b	2a	2b
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	1	0	0	0
6	0	1	0	0
7	0	0	1	0
8	0	0	0	1

Figure 3 – Standard Drive Sequence Example

The motor may also be half-stepped. Half-stepping is achieved by first energizing one coil, then two coils, then one coil, etc., in a sequence as shown in Figure 4 from [4]. This project utilizes half-stepping and is further discussed in Section 3.5.

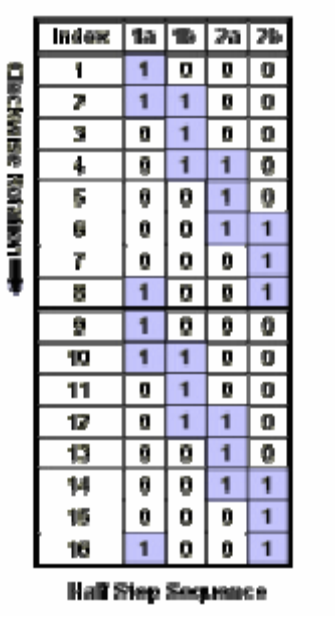

Figure 4 – Half-Step Drive Sequence Example

Lastly, a control circuit is needed to drive the stepper motor. The basic control circuit for a unipolar stepper motor is shown in Figure 5 from [3]. The motor driving circuit specific to this project is explained in Section 3.5.

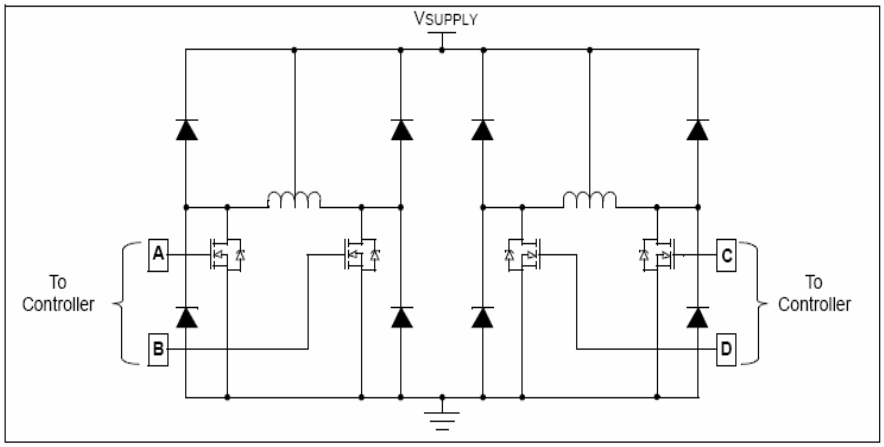

Figure 5 – Unipolar Motor Control Circuit

3. Project Design Methodology

This section will discuss the methodology involved in the design of the solar tracker. The project was divided into parts to make the design process modular. The project consists

of reading a series of light sensor values, comparing them, and then positioning a motor to align with the greatest value which corresponds to the sun's position. Follow-on sections discuss hardware and software design considerations.

3.1. Light Sensor Design

As presented in Section 2.1, the sun tracker uses a CdS photocell for light detection. A complementary resistor value of 10 KΩ was used to construct the circuit shown in Figure 1 in Section 2.1. In this configuration, the output voltage will increase as light intensity increases.

The complementary resistor value should be chosen such as to achieve the widest output range possible. Photocell resistance was measured under dark conditions, average light conditions, and bright light conditions. The results are listed in Table 1.

Measured Resistance	Comments
50 KΩ	Dark (black vinyl tape placed over cell)
4.35 KΩ	Average (normal room lighting level)
200 Ω	Bright (flashlight directly in front of cell)

Table 1 – Photocell Resistance Testing Data

The selected 10K complementary resistor resulted in the following minimum and maximum voltages.

Minimum = 5 V × (10 KΩ / (10 KΩ + 50 KΩ)) = 0.83 V
Maximum = 5 V × (10 KΩ / (10 KΩ + 4.35 KΩ)) = 3.48 V

Thus, an output swing of 2.65 V results. While this is not ideal, it was determined to be sufficient for the project and additional amplification was not pursued.

3.2. Microcontroller

Since the project's focus is on embedded software control, the microcontroller is the heart of the system. The microcontroller selected for this project had to be able to convert the analog photocell voltage into digital values and also provide four output channels to control motor rotation. The PIC16F877 was selected as it satisfies these requirements in addition to already being provided with the class lab kit. Specifically, it possesses the following three features to satisfy the specific project goals [5].

- 10 bit multi-channel analog-to-digital converter

- 5 input/output ports
- 256 x 8 bytes of data EEPROM memory

A 4 MHz crystal oscillator was also used in conjunction with the PIC16F877 to provide the necessary clock input. This speed is sufficient for the application. A pin diagram of the PIC16F877 is provided in Figure 6 from [5].

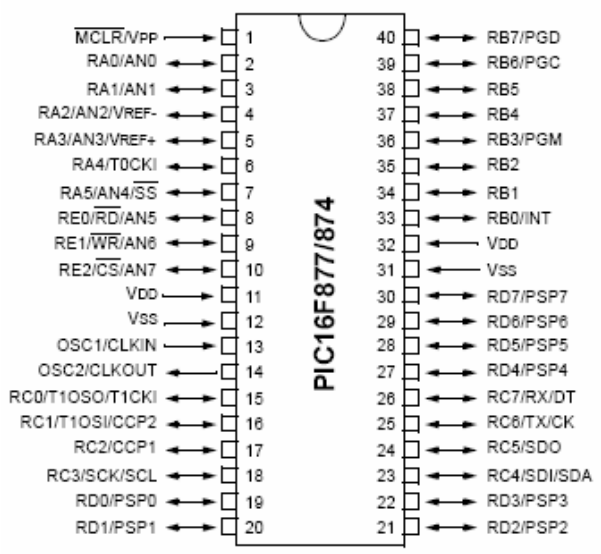

Figure 6 – PIC16F877 Pinout

3.3. Motor Driver and Stepper Motor

A single unipolar stepper motor was chosen to position the tracking sensor. A stepper motor was selected because of the precision it offers in positioning applications such as this. Additionally, complicated drive circuitry is not required with the unipolar type motor. The motor specifically used in the project was a 5 volt, 7.5 degree-per-step, 4 phase, unipolar motor. It was decided to half-step the motor in order to provide greater positioning accuracy. This results in 3.75 degrees-per-step. The drive sequence used in this design is shown in Figure 7.

Index Position	Y2	Y1	X2	X1
1	0	1	0	1
2	0	0	0	1
3	1	0	0	1
4	1	0	0	0
5	1	0	1	0
6	0	0	1	0
7	0	1	1	0
8	0	1	0	0

Figure 7 – Actual Half-Step Sequence Utilized

Figure 8 provides a schematic of the motor drive circuit design. This design and associated motor drive software code was based on course Lab 8. Darlington transistors were selected to be used for the motor drive circuitry. Each transistor is matched with a 3.3 KΩ resistor to an output that is used to switch the current to the motor winding on and off. This provides the proper drive sequence to rotate the motor either clockwise or counterclockwise. The diode connecting the collector to the positive supply voltage protects the transistor from inductive kickback.

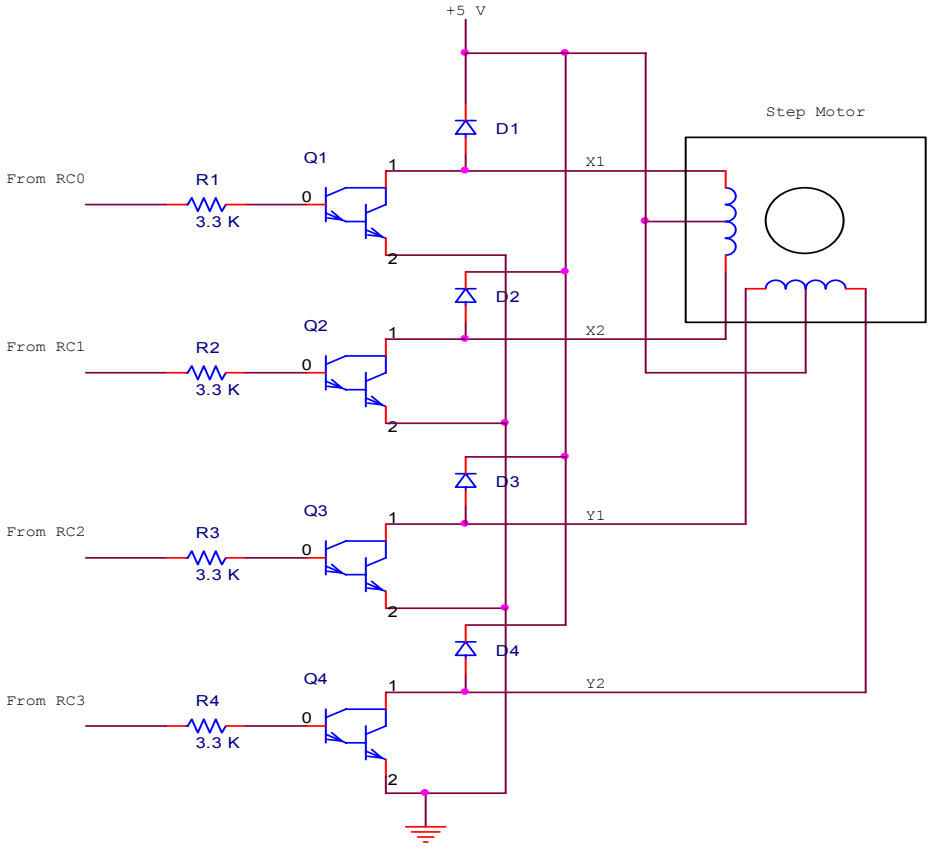

Figure 8 – Motor Drive Circuit

3.4. Voltage Regulation

The PIC16F877 requires a regulated 5 volt supply voltage. The 7805 voltage regulator was used to provide for that. The circuit shown in Figure 9 converts an unregulated supply of 9 volts to 5 volts for use by the microcontroller.

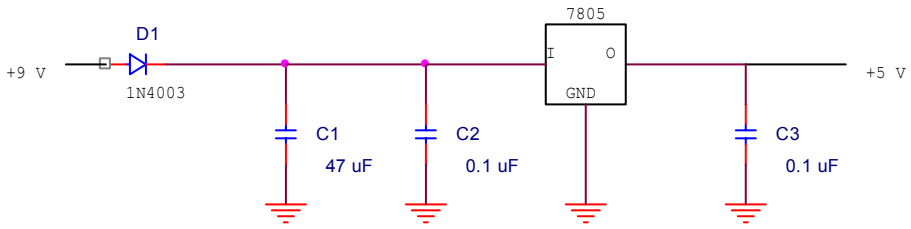

Figure 9 – Voltage Regulator Circuit

3.5. Construction

Ultimately the subparts of the project discussed in Sections 3.1 through 3.5 were consolidated to construct a complete project. Figure 10 provides a block diagram of the project while Figure 11 provides a complete hardware schematic of the project.

Some additional construction details worth mentioning deal with the motor and photocell. The motor was mounted to a plastic perforated board using standoffs to provide a stable base for it. The photocell was mounted on a small balsa wood platform which was secured to the motor shaft.

Lastly, a reset switch was added to allow for the microcontroller to be reset after it enters sleep mode.

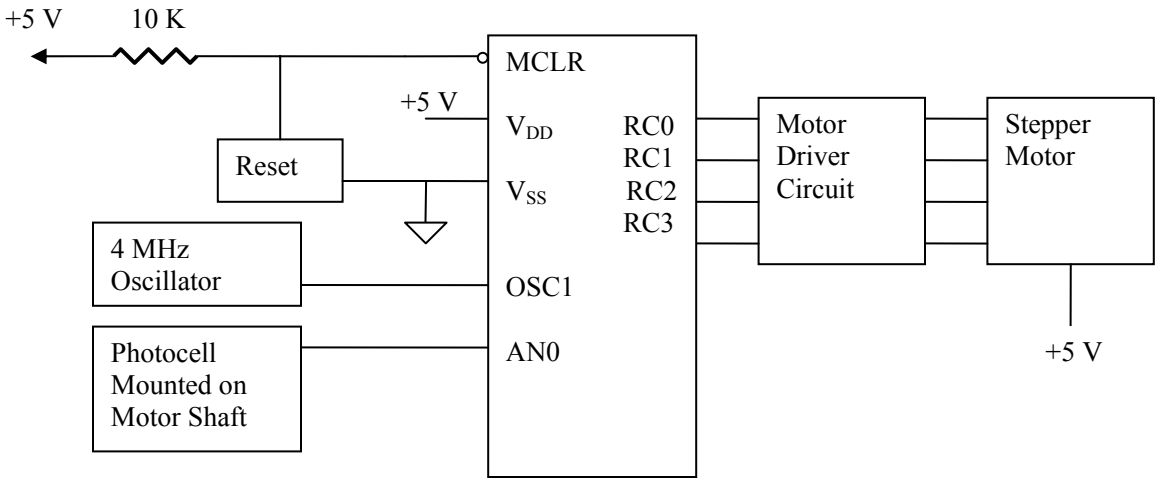

Figure 10 – Hardware Block Diagram

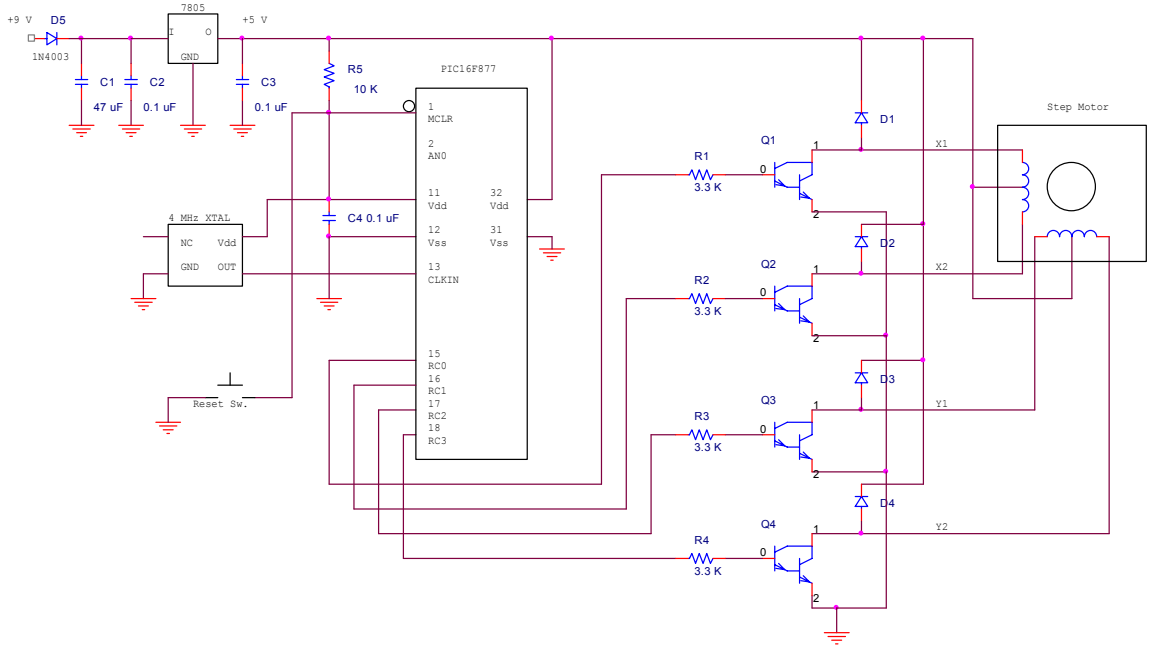

Figure 11 – Hardware Schematic Diagram

Table 2 lists the major components utilized in the project.

Item	Size or Part No.	Qty.
Microchip microcontroller	PIC16F877	1 ea.
Oscillator, crystal	4 MHz	1 ea.
Voltage regulator	7805	1 ea.
Photocell	Cadmium sulfide	1 ea.
Step motor, unipolar	5 V	1 ea.
Capacitor	0.1 μF	3 ea.
Capacitor	47 μF	1 ea.
Resistor	10 KΩ	1 ea.
Resistor	3.3 KΩ	4 ea.
Diode	1N4003	4 ea.
Transistor, Darlington	2SD1276A	4 ea.
Switch, momentary, pushbutton	Normally open	1 ea.
Motor/sensor mounting accessories	Various	Various

Table 2 – Parts List

3.6. Software/System Operation

As was fundamental to the course, the assembly language was utilized for the project. It was more than adequate to satisfy design objectives while enhancing level of understanding of the programming language.

Software operation can be divided into four main parts. The first part is initial positioning. Prior to powering up the system, the photocell must be manually set to a starting point (east). Once manually positioned, the tracking sensor will move one 3.75 degree step per second in the clockwise direction until a value of light intensity greater than the preset threshold is measured. The threshold has been set as a constant in program code to equal a voltage level of 4.60 volts. This level was selected to correspond to what was measured with the shielded photocell pointed directly at the sun. This level ensures that the tracker will seek out only an extremely bright source of light (i.e. the sun or the flashlight used for testing).

The second part of the system code deals with light tracking. This is the heart of the program. Once the tracker has set its initial position to a bright source of light (sun), it is ready to align itself more precisely and continue tracking the light. The tracker first measures light intensity at its present location. It then moves counterclockwise (left) by one 3.75 degree step and takes another measurement. Next, it moves clockwise (right) two 3.75 degree steps and takes a final measurement. The software comparison subroutines compare these values and position the tracker at the point of greatest measurement. If any of the values are equal, the tracker will return to the center position and check again later. The tracker will wait four minutes (four seconds for classroom

demonstration) before checking the three positions again. The four minute interval is based on the fact that the sun moves one degree every four minutes [6].

Low light detection is the third portion of the software routine. This works in conjunction with the tracking routine discussed in the previous paragraph. If light intensity below the low light threshold level, the tracker will keep measuring at whatever position it is at until the threshold is reached. The threshold for this portion has been assigned a constant in software equal to 3.70 volts. This level corresponds to what was measured with the shielded photocell during daytime overcast conditions.

The last portion of the software routine allows the tracker to reset itself at the end of a day. After every motor movement, a register is incremented or decremented so that the net position of the tracker can be known at any given time. Once the tracker has moved 180 degrees, light intensity is checked. If light intensity is below the 3.70 volt threshold, the tracker will return to its starting point and enter sleep mode. If above, it will keep checking until the measurement drops below the threshold. Once the tracker enters sleep mode, it can be reset manually with a momentary switch.

Figure 12 outlines the software operation. The entire code listing is provided in the appendix.

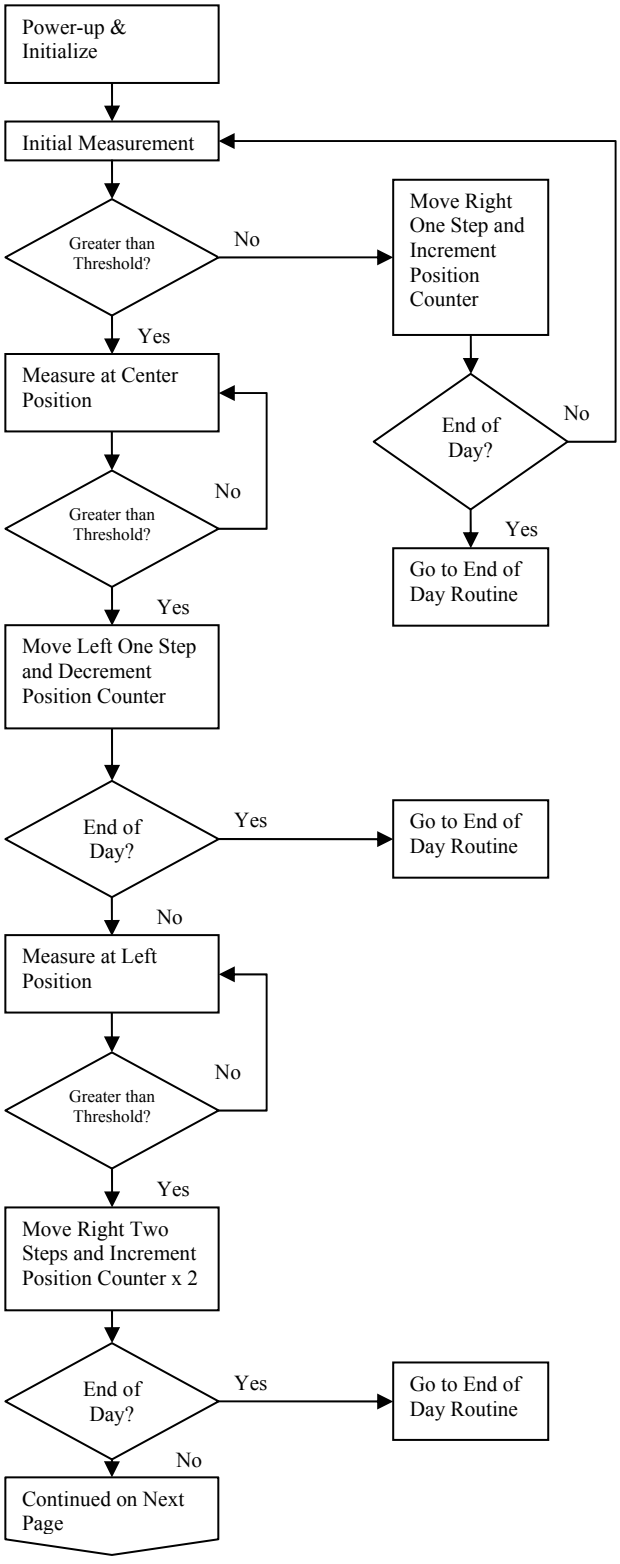

Figure 12 – Software Flow Diagram

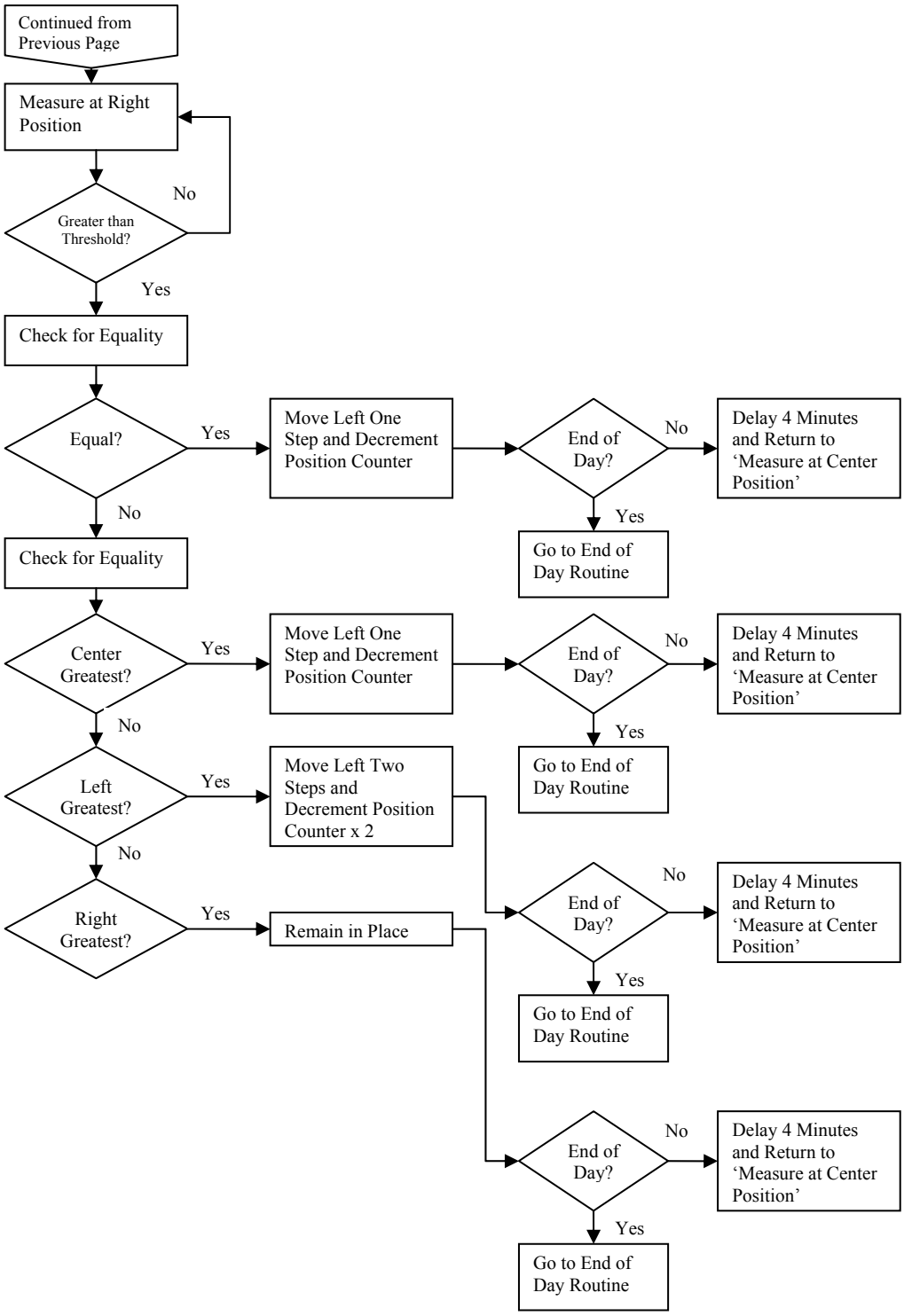

Figure 12 (Cont.) – Software Flow Diagram

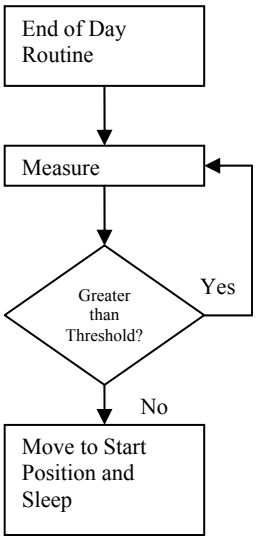

Figure 12 (Cont.) – Software Flow Diagram

4. Design Analysis and Results

Hardware and software portions of the project were separated into stages while developing the overall system. The portions consisted of light detection, motor driving, software tracking, and software enhancements. Building and testing smaller sections of the system made the project more manageable and increased efficiency by decreasing debugging time.

The project performs the required functions envisioned at the proposal phase. However, while satisfied with software operation and simulation, less satisfaction was obtained from two hardware areas. First, there is a potential for problems with motor/photocell movement due to the photocell wires creating binding issues. There are two wires attached to the photocell mounted on the motor shaft. Once the tracker has moved approximately 30 to 45 degrees, the wires place a counter torque on the motor and the motor slips. This creates positioning error. The present workaround for this is to hold the photocell wires in a way as to keep them perpendicular to the rear of the photocell as the tracker moves. This problem will be discussed further in Section 5.

The second issue deals with the photocell. It was discovered that the photocell needs to be shielded such that light can be directed narrowly to its surface. This was done by placing a black vinyl tube around the photocell to create a tunnel and help shield it from light that is not directly in its direct path. This dilemma is discussed further in Section 5.

5. Future Work

The goals of this project were purposely kept within what was believed to be attainable within the allotted timeline. As such, many improvements can be made upon this initial design. That being said, it is felt that this design represents a functioning miniature scale model which could be replicated to a much larger scale. The following recommendations are provided as ideas for future expansion of this project:

- Remedy the motor binding problems due to the photosensor leads. This could be done with some sort of slip ring mechanism, smaller gauge wire, a larger motor with more torque, or a combination of some or all of these ideas.
- Increase the sensitivity and accuracy of tracking by using a different light sensor. A phototransistor with an amplification circuit would provide improved resolution and better tracking accuracy/precision.
- Use a UCN5804 Darlington transistor array to reduce the number of discrete components used.
- Utilize a dual-axis design versus a single-axis to increase tracking accuracy.

6. Conclusion

This paper has presented a means of controlling a sun tracking array with an embedded microprocessor system. Specifically, it demonstrates a working software solution for maximizing solar cell output by positioning a solar array at the point of maximum light intensity. This project presents a method of searching for and tracking the sun and resetting itself for a new day. While the project has limitations, particularly in hardware areas discussed in Section 4 and Section 5, this provides an opportunity for expansion of the current project in future years.

References

[1] A.K. Saxena and V. Dutta, "A versatile microprocessor based controller for solar tracking," in Proc. IEEE, 1990, pp. 1105 – 1109.

[2] T.A. Papalias and M. Wong, "Making sense of light sensors," http://www.embedded.com, 2006.

[3] R. Condit and D. W. Jones, "Stepping motor fundamentals," Microchip Inc. Publication AN907, pp. 1 – 22, 2004.

[4] S. J. Hamilton, "Sun-tracking solar cell array system," University of Queensland Department of Computer Science and Electrical Engineering, Bachelors Thesis, 1999.

[5] Microchip Inc., "PIC16F87X Datasheet," www.microchip.com, 2001.

[6] M. F. Khan and R. L. Ali, "Automatic sun tracking system," presented at the All Pakistan Engineering Conference, Islamabad, Pakistan, 2005.

Appendix: Software Listing

```
;*******************************************************
; Term Project.asm
; Bill Lane
;
;
; Solar Tracker
;*******************************************************

        list p=16f877
        include "p16f877.inc"
        __CONFIG    _CP_OFF & _CPD_OFF & _LVP_OFF & _WDT_OFF &
_BODEN_OFF & _PWRTE_OFF & _XT_OSC

;**Initial assignments**

        cblock  0x20
                Direction           ;Motor direction register (left or right)
                Position0           ;Center reference position register
                Position1           ;Left reference position register
                Position2           ;Right reference position register
                Time1               ;Timer register for 1 msec delay function
                Time2               ;Timer register for 250 msec delay function
                Time3               ;Timer register for 1 sec delay function
                Time4               ;Timer register for 1 min delay function
                PositionCount       ;Register to store net value of steps taken
                Temp                ;Register for return to east decrement position loop
        endc

Left            equ d'2'            ;Left direction = 2
Right           equ d'1'            ;Right direction = 1
Index1          equ b'0101'         ;Step index position 1
Index2          equ b'0001'         ;Step index position 2
Index3          equ b'1001'         ;Step index position 3
Index4          equ b'1000'         ;Step index position 4
Index5          equ b'1010'         ;Step index position 5
Index6          equ b'0010'         ;Step index position 6
Index7          equ b'0110'         ;Step index position 7
Index8          equ b'0100'         ;Step index position 8
Threshold1      equ b'10111101'     ;Threshold level for minimum light detection
Threshold2      equ b'11101011'     ;Threshold level for light search subroutine

        org     0x000               ;Reset vector
        nop

;**Initial setup**

Initial
        banksel PORTC               ;Select Bank 0
```

```
        clrf    PORTC               ;Clear PORTC
        movlw   B'01000001'         ;Configure ADCON0
        movwf   ADCON0              ;ADCO0 = Fosc/8, Ch0, AD converter on
        banksel OPTION_REG          ;Select Bank 1
        movlw   B'10000110'         ;Configure OPTION_REG
        movwf   OPTION_REG          ;TMR0 prescaler = 1:128
        clrf    TRISC               ;Set PORTC as all outputs
        movlw   B'00000110'         ;Configure ADCON1
        movwf   ADCON1
        banksel PORTC               ;Select Bank 0
        movlw   Index1              ;Set initial step motor position at index position
        movwf   PORTC
        clrf    PositionCount       ;Zero out position

;**Search for brightest point (find sun) following initial startup**

Search
        btfss   INTCON,T0IF         ;Checks if Timer0 interrupt flag is set
        goto    Search              ;Loops until set
        bcf     INTCON,T0IF         ;Clears Timer0 interrupt flag
        bsf     ADCON0,GO           ;Sets GO bit in ADCON0 to start ADC
Wait2
        btfss   PIR1,ADIF           ;Checks if AD interrupt flag is set
        goto    Wait2               ;Loops until conversion is complete
        bcf     PIR1,ADIF           ;Clears AD interrupt flag
        movlw   Threshold2          ;Moves minimum brightness value to W for comparison
        subwf   ADRESH,W
        btfsc   STATUS,C            ;See if value in ADRESH is greater than threshold value
        goto    Main                ;If ADRESH > min threshold, continue to main routine
        movlw    Right              ;If ADRESH < min value, move motor right
        movwf    Direction
        call    StepControl
        incf    PositionCount,1     ;Increment position count for right movement
        call    PositionCountCheck  ;Check location to see if at end of day
        goto    Search              ;Keep looking for brightest point (sun)

Delay
        ;call   Delay1s             ;4 sec delay for class demo
        ;call   Delay1s
        ;call   Delay1s
        ;call   Delay1s
        call    Delay1m             ;4 minute delay for actual implementation
        call    Delay1m
        call    Delay1m
        call    Delay1m

;**Main move/measure/compare routine**

Main
        call    ADCStart            ;Call ADC to get measurement at center position
        movf    ADRESH,W
```

```
        movwf   Position0               ;Store center position measurement
        movlw    Left                   ;Move motor left 1 step
        movwf    Direction
        call    StepControl
        decf    PositionCount,1         ;Decrement position count for left movement
        call    PositionCountCheck      ;Check location to see if at end of day

        call    ADCStart                ;Call ADC subroutine to get measurement at left position
        movf    ADRESH,W
        movwf   Position1               ;Store left position measurement
        movlw   Right                   ;Move motor right 2 steps
        movwf   Direction
        call    StepControl
        incf    PositionCount,1         ;Increment position count for right movement
        call    PositionCountCheck      ;Check location to see if at end of day
        movlw   Right
        movwf   Direction
        call    StepControl
        incf    PositionCount,1         ;Increment position count for right movement
        call    PositionCountCheck      ;Check location to see if at end of day

        call    ADCStart                ;Call ADC to get measurement at right position
        movf    ADRESH,W
        movwf   Position2               ;Store right position measurement

        movf    Position0,W             ;Check if center and left positions are equal
        subwf   Position1,W
        btfsc   STATUS,Z
        goto    ReturnToPosition0       ;If equal, return to center position

        movf    Position0,W             ;Check if center and right positions are equal
        subwf   Position2,W
        btfsc   STATUS,Z
        goto    ReturnToPosition0       ;If equal, return to center position

        movf    Position1,W             ;Check if left and right positions are equal
        subwf   Position2,W
        btfsc   STATUS,Z
        goto    ReturnToPosition0       ;If equal, return to center position

        movf    Position1,W             ;Check if center position is greater than left position
        subwf   Position0,W
        btfsc   STATUS,C
        goto    NextCheck               ;If center is greater than left, compare to right position
        goto    NextMajorCheck          ;If center is less than left, check left position vs. others

NextCheck
        movf    Position2,W             ;Check if center position is greater than right position
        subwf   Position0,W
        btfsc   STATUS,C
        goto    ReturnToPosition0       ;If center position is greatest, return to center position
```

```
        goto    NextMajorCheck      ;If center position is not greatest, check left vs. others

ReturnToPosition0
        movlw   Left                ;Moves motor one step left to center if center is greatest
        movwf   Direction
        call    StepControl
        decf    PositionCount,1     ;Decrement position count for left movement
        call    PositionCountCheck  ;Check location to see if at end of day
        goto    Delay

NextMajorCheck
        movf    Position0,W         ;Check if left position is greater than center position
        subwf   Position1,W
        btfsc   STATUS,C
        goto    NextCheck2          ;If left position is greatest, compare to right position
        goto    NextMajorCheck2     ;If left is less than center, check right position vs. others

NextCheck2
        movf    Position2,W         ;Check if left position is greater than right position
        subwf   Position1,W
        btfsc   STATUS,C
        goto    ReturnToPosition1   ;If left position is greatest, return to left position
        goto    NextMajorCheck2     ;If center in not greatest, check right position vs. others

ReturnToPosition1
        movlw   Left                ;Moves motor two steps left to left if left is the greatest
        movwf   Direction
        call    StepControl
        decf    PositionCount,1     ;Decrement position count for left movement
        call    PositionCountCheck  ;Check location to see if at end of day
        movlw   Left
        movwf   Direction
        call    StepControl
        decf    PositionCount,1     ;Decrement position count for left movement
        call    PositionCountCheck  ;Check location to see if at end of day
        goto    Delay

NextMajorCheck2
        movf    Position0,W         ;Check if right position is greater than center position
        subwf   Position2,W
        btfsc   STATUS,C
        goto    NextCheck3          ;If right position is greatest, compare to left position
        goto    ReturnToPosition0   ;If right is less than center, return to center position

NextCheck3
        movf    Position1,W         ;Check if right position is greater than left position
        subwf   Position2,W
        btfsc   STATUS,C
        goto    DoNothing           ;If right position is greatest, remain in place
        goto    Delay               ;If not, start over
```

```
DoNothing
        goto    Delay

;**Step motor control**

ADCStart
        btfss   INTCON,T0IF     ;Checks if Timer0 interrupt flag is set
        goto    ADCStart        ;Loops until set
        bcf     INTCON,T0IF     ;Clears Timer0 interrupt flag
        bsf     ADCON0,GO       ;Sets GO bit in ADCON0 to start ADC
Wait
        btfss   PIR1,ADIF       ;Checks if AD interrupt flag is set
        goto    Wait            ;Loops until conversion is complete
        bcf     PIR1,ADIF       ;Clears AD interrupt flag
        movlw   Threshold1      ;Moves min light intensity value to W for comparison
        subwf   ADRESH,W
        btfsc   STATUS,C        ;See if value in ADRESH is greater than threshold value
        return                  ;If ADRESH > min threshold, continue w/ main routine
        goto    ADCStart        ;If ADRESH < min value, keep checking

StepControl
        call    Delay1s         ;Delay between step movements
        movf    Direction,W     ;Find out direction to move
        movf    PORTC,W         ;Read PORTC
        sublw   Index1          ;Compare PORTC to index position
        btfss   STATUS,Z        ;If no match, check another against another position
        goto    StepControl2
        movf    Direction,W     ;Find out direction to move
        sublw   Right           ;See if right is the direction called for
        btfsc   STATUS,Z        ;If right, proceed to next  position in the right direction
        goto    StepControl1
        movlw   Index2          ;If not right, proceed to next  position in the left direction
        goto    MoveMotor

StepControl1
        movlw   Index8          ;Next index position in the right direction
        goto    MoveMotor

StepControl2
        movf    PORTC,W         ;Read PORTC
        sublw   Index8          ;Compare PORTC to index position
        btfss   STATUS,Z        ;If no match, check another against another position
        goto    StepControl4
        movf    Direction,W     ;Find out direction to move
        sublw   Right           ;See if right is the direction called for
        btfsc   STATUS,Z        ;If right, proceed to next position in the right direction
        goto    StepControl3
        movlw   Index1          ;If not right, proceed to next position in the left direction
        goto    MoveMotor

StepControl3
```

```
        movlw   Index7
        goto    MoveMotor               ;Next index position in the right direction

StepControl4
        movf    PORTC,W                 ;Read PORTC
        sublw   Index7                  ;Compare PORTC to index position
        btfss   STATUS,Z                ;If no match, check another against another position
        goto    StepControl6
        movf    Direction,W             ;Find out direction to move
        sublw   Right                   ;See if right is the direction called for
        btfsc   STATUS,Z                ;If right, proceed to next position in the right direction
        goto    StepControl5
        movlw   Index8                  ;If not right, proceed to next position in the left direction
        goto    MoveMotor

StepControl5
        movlw   Index6                  ;Next index position in the right direction
        goto    MoveMotor

StepControl6
        movf    PORTC,W                 ;Read PORTC
        sublw   Index6                  ;Compare PORTC to index position
        btfss   STATUS,Z                ;If no match, check another against another position
        goto    StepControl8
        movf    Direction,W             ;Find out direction to move
        sublw   Right                   ;See if right is the direction called for
        btfsc   STATUS,Z                ;If right, proceed to next  position in the right direction
        goto    StepControl7
        movlw   Index7                  ;If not right, proceed to next position in the left direction
        goto    MoveMotor

StepControl7
        movlw   Index5                  ;Next index position in the right direction
        goto    MoveMotor

StepControl8
        movf    PORTC,W                 ;Read PORTC
        sublw   Index5                  ;Compare PORTC to index position
        btfss   STATUS,Z                ;If no match, check another against another position
        goto    StepControl10
        movf    Direction,W             ;Find out direction to move
        sublw   Right                   ;See if right is the direction called for
        btfsc   STATUS,Z                ;If right, proceed to next position in the right direction
        goto    StepControl9
        movlw   Index6                  ;If not right, proceed to next position in the left direction
        goto    MoveMotor

StepControl9
        movlw   Index4                  ;Next index position in the right direction
        goto    MoveMotor
```

```
StepControl10
        movf    PORTC,W         ;Read PORTC
        sublw   Index4          ;Compare PORTC to index position
        btfss   STATUS,Z        ;If no match, check another against another position
        goto    StepControl12
        movf    Direction,W     ;Find out direction to move
        sublw   Right           ;See if right is the direction called for
        btfsc   STATUS,Z        ;If right, proceed to next position in the right direction
        goto    StepControl11
        movlw   Index5          ;If not right, proceed to next position in the left direction
        goto    MoveMotor

StepControl11
        movlw   Index3          ;Next index position in the right direction
        goto    MoveMotor

StepControl12
        movf    PORTC,W         ;Read PORTC
        sublw   Index3          ;Compare PORTC to index position
        btfss   STATUS,Z        ;If no match, check another against another position
        goto    StepControl14
        movf    Direction,W     ;Find out direction to move
        sublw   Right           ;See if right is the direction called for
        btfsc   STATUS,Z        ;If right, proceed to next position in the right direction
        goto    StepControl13
        movlw   Index4          ;If not right, proceed to next position in the left direction
        goto    MoveMotor

StepControl13
        movlw   Index2          ;Next index position in the right direction
        goto    MoveMotor

StepControl14
        movf    PORTC,W         ;Read PORTC
        sublw   Index2          ;Compare PORTC to index position
        btfss   STATUS,Z        ;If no match, check another against another position
        goto    StepControl16
        movf    Direction,W     ;Find out direction to move
        sublw   Right           ;See if right is the direction called for
        btfsc   STATUS,Z        ;If right, proceed to next position in the right direction
        goto    StepControl15
        movlw   Index3          ;If not right, proceed to next position in the left direction
        goto    MoveMotor

StepControl15
        movlw   Index1          ;Next index position in the right direction
        goto    MoveMotor

StepControl16                   ;Otherwise return to index position 1
        movlw   Index1
```

```
MoveMotor
        movwf   PORTC               ;Move motor to desired index position
        return

;**End of day checking/reset/sleep routine**

PositionCountCheck
        movf    PositionCount,W
        sublw   d'48'
        btfsc   STATUS,Z            ;If equal to 48, go to end of day brightness check,
                                    otherwise, see if equal to 208
        goto    EndOfDayBrightCheck
        movf    PositionCount,W
        sublw   d'208'
        btfsc   STATUS,Z            ;If equal to 208, go to end of day brightness check,
                                    otherwise, return to main routine
        goto    EndOfDayBrightCheck
        return
EndOfDayBrightCheck
        btfss   INTCON,T0IF         ;Checks if Timer0 interrupt flag is set
        goto    EndOfDayBrightCheck ;Loops until set
        bcf     INTCON,T0IF         ;Clears Timer0 interrupt flag
        bsf     ADCON0,GO           ;Sets GO bit in ADCON0 to start ADC
Wait3
        btfss   PIR1,ADIF           ;Checks if AD interrupt flag is set
        goto    Wait3               ;Loops until conversion is complete
        bcf     PIR1,ADIF           ;Clears AD interrupt flag
        movlw   Threshold1          ;Moves minimum light intensity value to W for
                                    comparison w/ ADRESH
        subwf   ADRESH,W
        btfsc   STATUS,C            ;See if value in ADRESH is greater than threshold value
        goto    EndOfDayBrightCheck ;If ADRESH > minimum threshold, continue checking
        call    ReturnToEast        ;Otherwise, return to east and go to sleep
        sleep
        goto    Initial
ReturnToEast
        movlw   d'48'               ;Repeat 48 times to return motor to start (east)
        movwf   Temp
Loop
        movlw    Left               ;Move motor left 1 step
        movwf    Direction
        call     StepControl
        decfsz   Temp,F
        goto     Loop
        return

;**Delay routines**

Delay1ms
        movlw   d'250'              ;1 msec delay
        movwf   Time1
```

```
Loop1
        nop
        decfsz  Time1,F
        goto    Loop1
        return

Delay250ms                      ;250 msec delay
        movlw   d'250'
        movwf   Time2
Loop2
        call    Delay1ms
        decfsz  Time2,F
        goto    Loop2
        return

Delay1s                         ;1 sec delay
        movlw   d'4'
        movwf   Time3
Loop3
        call    Delay250ms
        decfsz  Time3,F
        goto    Loop3
        return

Delay1m                         ;1 min delay
        movlw   d'240'
        movwf   Time4
Loop4
        call    Delay250ms
        decfsz  Time4,F
        goto    Loop4
        return

    end
```