Traffic Controller

Laboratory Report No. 1
Final Report

Submitted by:

Lab Partner: None

Submitted in partial fulfillment of the requirements for ELET3155, Electronics Laboratory VI, Section 080.

Submitted to Steve Kuyath

Date Performed: N/A
Date Due: 06/09/2007
Date Submitted: 06/07/2007

Software Used
Microsoft Word 2003, Microsoft Excel 2003,
Microsoft Project 2000, PSpice Student Version Release 9.1
I. Summary

The purpose of this lab was to become familiar with the specifications required for senior design. Several requirements must be met. First, the design was broken into smaller parts so that the process is manageable. Second, a schedule was derived from all the parts. This helps in tracking the progress of the project. The software package called Microsoft Project was used as the scheduling tool. Third, the circuit and program were designed and debugged. Finally, the circuit was built, tested and demonstrated.

Another purpose for developing this project was to learn about a new microcontroller, the AVR ATmega 128. The ATmega 128 was used to design a traffic light controller. The traffic light controller required the use of the different ports that the AVR has to offer. This helped to become familiar with the instruction set as well as the port operations.

The design of the traffic light controller was broken down into several parts. After breaking down the design, it was determined that the design had six parts. First, the type of intersection and how to control the flow of traffic was identified. Second, a schedule was derived for all the tasks. Third, a block diagram was created for this circuit. Fourth, the flow chart for a program that was to control the sequence of lights for the intersection was created. Fifth, the code needed to operate the ATmega 128 microcontroller was designed and then debugged. Finally, the schematic of the circuit was minimized. After these six steps, the circuit can be built and tested.

II. Parts List

1 – AVR ATmega 128 Microcontroller
1 – 7432
1 – 7476
4 – Red LED’s
4 – Yellow LED’s
4 – Green LED’s
4 – 4.9k Ohms Resistors
1 – 500 Ohm Resistor

III. Procedure

1. Identify Criteria

A visit to the intersection in question revealed how the traffic flows through the intersection. The intersection that was observed has two main flows of traffic; one north/south and the other east/west. It was determined that the north/south and east/west traffic light will be set to stay green or red for a predetermined amount of time. At times, the traffic is too heavy and prevents vehicles from making left turns. Therefore, the traffic light will have left turn arrows that will be triggered by sensors when a vehicle is waiting in the left turn lane.

2. Schedule

When creating a schedule for the project, the design process must be divided into several tasks. These tasks will be placed in a chart for tracked. Microsoft Project is a software package that helps manage these tasks. Each task is to be assigned a start date and a time frame that the task must be completed for completion. This will enable the project leader to track the progress and determine if any adjustments need to be made. Figure 1.1 below is a capture of Microsoft Project for this particular design. A spreadsheet in Microsoft Excel was also created as a comparison. Figure 1.2 below is the comparison spreadsheet. After the dates were set in Microsoft Project, there was no good way to determine if a task was completed on time or if it went over schedule. On the other hand, the Excel spreadsheet allowed for separate columns for the actual start and completion dates, allowing for a better analysis of the project’s status. The designing of the program was given the longest duration to allow for learning a new instruction.
set and the inner workings of the AVR ATMega 128. Tasks three and four went over the allotted time because the tasks were not started on time. The debugging of the program took much longer than anticipated, which was due to the unfamiliarity with the instruction set which pushed out the rest of the tasks. In the end, this project was completed on time for the demonstration.

Figure 1.1 Traffic Light Controller Schedule in Microsoft Project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Status</th>
<th>Start Date</th>
<th>Complete Date</th>
<th>Actual Start Date</th>
<th>Actual Complete Date</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Identify Criteria</td>
<td>comp</td>
<td>19-May</td>
<td>19-May</td>
<td>19-May</td>
<td>19-May</td>
<td>met</td>
</tr>
<tr>
<td>2</td>
<td>Determine Inputs &amp; Outputs</td>
<td>comp</td>
<td>20-May</td>
<td>20-May</td>
<td>20-May</td>
<td>20-May</td>
<td>met</td>
</tr>
<tr>
<td>3</td>
<td>Block Diagram</td>
<td>comp</td>
<td>19-May</td>
<td>20-May</td>
<td>20-May</td>
<td>21-May</td>
<td>over</td>
</tr>
<tr>
<td>4</td>
<td>Schematic</td>
<td>comp</td>
<td>21-May</td>
<td>21-May</td>
<td>21-May</td>
<td>22-May</td>
<td>over</td>
</tr>
<tr>
<td>5</td>
<td>Flow Chart</td>
<td>comp</td>
<td>21-May</td>
<td>22-May</td>
<td>21-May</td>
<td>22-May</td>
<td>met</td>
</tr>
<tr>
<td>6</td>
<td>Write Proposal &amp; Submit</td>
<td>comp</td>
<td>23-May</td>
<td>23-May</td>
<td>23-May</td>
<td>23-May</td>
<td>met</td>
</tr>
<tr>
<td>7</td>
<td>Design Program</td>
<td>comp</td>
<td>24-May</td>
<td>27-May</td>
<td>23-May</td>
<td>26-May</td>
<td>met</td>
</tr>
<tr>
<td>8</td>
<td>Test Program</td>
<td>comp</td>
<td>27-May</td>
<td>29-May</td>
<td>26-May</td>
<td>31-May</td>
<td>over</td>
</tr>
<tr>
<td>9</td>
<td>Load Program</td>
<td>comp</td>
<td>29-May</td>
<td>29-May</td>
<td>26-May</td>
<td>30-May</td>
<td>over</td>
</tr>
<tr>
<td>10</td>
<td>Write Brief Status Report &amp; Submit</td>
<td>comp</td>
<td>30-May</td>
<td>30-May</td>
<td>30-May</td>
<td>30-May</td>
<td>met</td>
</tr>
<tr>
<td>11</td>
<td>Build Circuit</td>
<td>comp</td>
<td>30-May</td>
<td>31-May</td>
<td>25-May</td>
<td>1-Jun</td>
<td>over</td>
</tr>
<tr>
<td>12</td>
<td>Test Circuit</td>
<td>comp</td>
<td>31-May</td>
<td>1-Jun</td>
<td>29-May</td>
<td>2-Jun</td>
<td>over</td>
</tr>
<tr>
<td>13</td>
<td>Demonstrate Project</td>
<td>comp</td>
<td>2-Jun</td>
<td>2-Jun</td>
<td>2-Jun</td>
<td>2-Jun</td>
<td>met</td>
</tr>
<tr>
<td>14</td>
<td>Write Final Report</td>
<td>ip</td>
<td>2-Jun</td>
<td>9-Jun</td>
<td>30-May</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1.2 Traffic Light Controller Schedule in Excel
3. Block Diagram

The block diagram shows a rough drawing of how inputs and outputs are interfaced with the microcontroller. The traffic light controller has four sensors which indicate when a vehicle is waiting to make a left turn. The signal from the sensors will flow through a signal conditioning circuit that transforms it to a signal that the ATMega 128 can recognize. The output has four sets of lights, each set having the colors green, yellow, and red. Each set controls a different flow of traffic. The signal conditioning circuit was not needed for the output since the ATMega 128 has pull-up resistors built into the microcontroller. The pull-up resistors allow the LED’s to operate without any other components. If the light were operated by a 120 volt AC source, then a transistor would be needed to drive a relay to operate the lights. This transistor and any other components would be part of the signal conditioning circuit for the output.

![Traffic Light Controller Block Diagram]

**Figure 1.3 Traffic Light Controller Block Diagram**
4. Flow Chart

The flow chart is a legible representation of the program that will control the intersection. This helps the programmer with the sequential order in which the program needs to flow. In this case, the north and south bound traffic will start with a green light while all the other lanes of traffic are red. After a predetermined amount of time, the north/south traffic light turns yellow and then red. When the north/south light turns red, the east/west light turns green and follows the same sequence as the north/south bound traffic light. The system remains in this loop until an indication that a vehicle is waiting in a left turn lane. When the light turns yellow, the controller goes out and scans the inputs. If any of the inputs are high, then the program will jump to a subroutine. This subroutine has a different light sequence. The new light sequence controls the left turn lane arrows as well as the main lights. After the subroutine is complete, the program jumps back to the main loop.
Figure 1.4 Traffic Light Controller Flow Chart
5. Design Code and Debug

The program has four parts. The main part is the main loop which will jump to one of three subroutines. The main loop allows the two main flows of traffic to run the light sequence from green to yellow then red. Between each light change the program jumps to the first subroutine which is a delay subroutine that counts the amount of time needed between light changes. The green light will stay green for 5 seconds and the yellow for 2 seconds. While the one flow of traffic is green then yellow, the others will remain red. After the one set of lights turn red, then the other main flow of traffic will turn green and run the same amount of time as the other. The second and third subroutines have a different light sequence. These two sequences control the left turn lane lights. In the main loop, when the lights turn yellow, the controller scans the inputs for any data. If there is data, then the program will jump to the appropriate subroutine. The turn lane stays green only for 2 seconds before changing yellow. Appendix A shows the entire program for this project with each command line commented. Some of the commands in the instruction set do not use hexadecimal numbers, commands like clear bit (CBI) and set bit (SBI). The command syntax is “sbi A,b” where A is the I/O location address and b is the bit in the status register. The bit in the status register really refers to the pin on the port. These commands only allow one bit to be set or cleared at a time. If multiple bits need to be set or cleared, then the In and Out commands are recommended. These commands require that a register be used. For instance, if port b needed bits 1 and 4 to be high, then a register can be loaded with hex 09. After the register is loaded then the Out command can be used. The two commands would look like the following:

```
LDI R20,$09 ;Load register 20 with hex 09
OUT PORTB,R20 ;Output register 20 to port B
```
6. Schematic

Figure 1.5 represents the input circuit to the microcontroller. Both the north and south bound left turn lane sensors were placed as inputs to an OR gate. The out of the OR gate was tied to the J side of a J/K flip flop which made the output Q go high and stay high. The K side was tied to the yellow light of the turn lane that was presently green. If J is high when K goes high, then the flip flop will trigger and set Q to low. Q will also go low if J is low when K goes high. The output of Q was tied to port A so that the program could check the status of the left turn lane sensors. Both the clear and preset of the flip flop was tied to 5 volts. The clear lead also had a push button tied to it in order to reset the flip flop. When the clear goes low, Q is set low, and this is the state at which the controller needs to start. Figure 1.6 represents the rest of the circuit. Because the ATMega 128 ports have pull-up resistors built in, the LED’s could be tied directly to the microcontroller. Ports B and D are the outputs to the LED’s where port B controls the north and south bound traffic, while port D controls the east and west bound traffic.

Attached to the microcontroller were two other critical components. First, a 5 volts DC source was attached to pin 52 and pin 53 to ground. The 5 volts and ground source provides the power the microcontroller needs to operate. Second, a 16MHz crystal and two capacitors were attached to pins 23 and 24. The two capacitors values were 22pF. The crystal and capacitors provides the clocking of the microcontroller.
Figure 1.5 Traffic Light Controller Input Circuit

Figure 1.6 Traffic Light Controller Schematic of the Microcontroller and Outputs
IV. Conclusion

In preparation for senior design, this project has helped in understanding the design process. The idea of dividing the project into manageable tasks helps the individual from becoming overwhelmed. Creating a schedule after the design is divided into tasks helps focus on only one part of the design and not concentrate on any other task until this task is complete. The tasks of the flow chart and block diagram are no more than a starting point for the design. The block diagram is a rough sketch of how the circuit will work without any specific details of how the circuit is actually going to work. The same can be said about the flow chart. The only difference is that the flow chart is laid out with more detail of how the program will operate. When designing the code, the programmer can follow the flow chart and convert it to match the instruction set. The schematic can be designed after every little detail has been worked out. The circuit can be built from the schematic and the program loaded into ATMega 128. With the program loaded and the circuit built, the design can be tested and debugged.

The AVR ATMega 128 microcontroller has many advantages. First, all the ports can be either input or output by setting appropriate bits in the data direction register (DDRx). Second, each pin attached to the ports has a built in pull-up resistor which provides 5 volts to the pin when written to. In some cases this eliminates the need for a signal conditioning circuit on the outputs, which will save on final costs. Finally, the AVR ATMega 128 has a reduced instruction set. Comparing to the 68HC11 microcontroller, the ATMega 128 allows for clearing or setting a single bit on a port with one instruction. There has been a learning curve with this instruction set. In some cases an instruction is looking for the decimal representation of the pin to be controlled and not hexed. When the hex value was entered, a different pin was activated than was expected.
V. References


VI. Appendix

.include "m128def.inc"
ستراتيج
çección
Define Register
. . .
def counter1 = R16
def counter2 = R17
def counter3 = R18
def temp = R19
def counter = R20
Main Program
.cseg
.org 0
rjmp INIT ; Initialize

INIT:
ldi temp,low(RAMEND); Initialize Stack
out spl,temp
ldi temp,high(RAMEND)
out sph,temp
ldi r21,$ff
out ddrb,r21
out ddrd,r21
ldi r21,$00
out porta,r21
ldi r21,$0c
out portb,r21
ldi r21,$09
out portd,r21
MAINLP:  
rcall  DELAY5  
cbi  portb,2  ;North/South green off  
sbi  portb,6  ;North/South yellow on  
rcall  DELAY2  
in  r22,pina  ;Read input on port a  
andi  r22,$01  ;Filter bits North/South sensor  
brne  EWLT  
cbi  portb,6  ;North/South yellow off  
sbi  portb,0  ;North/South red on  
cbi  portd,0  ;East/West red off  
sbi  portd,2  ;East/West green on  
  
RET1:  
rcall  DELAY5  
cbi  portd,2  ;East/West green off  
sbi  portd,1  ;East/West yellow on  
rcall  DELAY2  
in  r22,pina  ;Read input on port a  
andi  r22,$02  ;Filter bits for East/West sensor  
brne  NSLT  
cbi  portd,1  ;East/West yellow off  
sbi  portd,0  ;East/West red on  
cbi  portb,0  ;North/South red off  
sbi  portb,2  ;North/South green on  
  
RET2:  
rjmp  MAINLP  

;******************************  
;* Delay Time  
;******************************  

DELAY_1ms:  ldi  counter1,16  ;\  
DELAY_1ms_1:  ldi  counter2,250  ;\  
DELAY_1ms_2:  nop  ;  | 2 loops make up  
dec  counter2  ;  |  a 1msec delay  
brne  DELAY_1ms_2  ;  |  
dec  counter1  ;  /  
brne  DELAY_1ms_1  ;  /  
ret  

DELAY_200ms:  ldi  counter3,200  ;\  
DELAY_200ms_1:  rcall  DELAY_1ms  ;  | 200msec delay  
dec  counter3  ;  |  
brne  DELAY_200ms_1  ;  /  
ret
DELAY5:
  ldi counter,25 ];
DELAY5_1:
  rcall DELAY_200ms ; | 5 second delay
dec counter ; |
brne DELAY5_1 ; /
ret

DELAY2:
  ldi counter,10 ];
DELAY2_1:
  rcall DELAY_200ms ; | 2 second delay
dec counter ; |
brne DELAY2_1 ; /
ret

; East/West Left Turn

EWLT:
  cbi portb,6 ; North/South yellow off
  sbi portb,0 ; North/South red on
  ldi r23,$24
  out portd,r23 ; E/W red off green & green LT on
  rcall DELAY2
cbi portd,5 ; East/West left turn green off
  sbi portd,4 ; East/West left turn yellow on
  rcall DELAY2
cbi portd,4 ; East/West left turn yellow off
  sbi portd,3 ; East/West left turn red on
  rjmp RET1

; North/South Left Turn

NSLT:
  cbi portd,1 ; East/West yellow off
  sbi portd,0 ; East/West red on
  ldi r23,$24
  out portb,r23 ; N/S red off green & green LT on
  rcall DELAY2
cbi portb,5 ; North/South left turn green off
  sbi portb,4 ; North/South left turn yellow on
  rcall DELAY2
cbi portb,4 ; North/South left turn yellow off
  sbi portb,3 ; North/South left turn red on
  rjmp RET2