Autonomous Connect Four

Bradley University
Department of Electrical and Computer Engineering

Wayne Bogart Justin Middleton

Advisor: Dr. James Irwin
Co-Advisor: Nick Schmidt

May 8, 2007

Abstract

The Autonomous Connect Four system is a proof-of-concept project that uses image processing and control algorithms to automate play of a Connect Four game using the Micropac 535 development kit. The paper covers the design process of electrical and mechanical systems, menu systems used, automated and non-automated game modes, image processing objectives related to A/D converter timing and interrupt latency of a Siemens 80C535 microprocessor, explanation of a queue system, and a matrix search algorithm are discussed. Results are discussed throughout the paper and in the conclusion section. Appendices show references used, schematics and design equations, software, and a data sheet for the product.

Table of Contents

Acknowledgements

Justin would like to thank his family for their help with this project. He would also like to thank them for allowing him to spend countless hours working on this project, thus sacrificing time spent with his family. He would also like to thank Dr. James Irwin, Jr. for all his hard work and ideas associated with this project. Justin thanks Dr. Irwin for the required weekly presentations to help in the development of creating interesting and professional Power Point presentations. Justin also thanks him for allowing Justin and Wayne to be creative this project. Justin thanks Mr. Nick Schmidt for dedicating his time, efforts, and tools for ideas and construction of this project. Mr. Dave Miller is thanked for allowing Justin and Wayne to use the ME shop to construct the Autonomous Connect Four. Justin thanks Mr. Chris Mattus for his time, ideas, and allowing the purchase of the necessary tools needed for this project.

Wayne would like to thank his friends and family for their contributions to this project. He would also like to thank Dr. James Irwin, Jr. and Mr. Nick Schmidt for all the time and effort they devoted to this project. Wayne also thanks Mr. Chris Mattus and Mr. Dave Miller for the use of their tools and time.

Introduction

Connect Four has been marketed by companies such as Milton Bradley for many years now. But, it can be traced back to Captain James Cook and his voyages, where he and his subordinates would play quite often. The captain played so often that his crew gave it the name “Captain’s Mistress.” Connect Four is a two player game, each player having their own checkers of a unique color. The object of the game is to place four of the same color checkers in a row before the opponent does. Vertically, horizontally, and diagonally are all ways in which a player can have four in row and win the game. Figure 1 shows the Connect Four game.

Figure 1

Connect Four Gam

The goals for the project were to have three different modes of game play:

- Human vs. Human
- Computer vs. Human
- Computer vs. Computer

Human vs. human and computer vs. human modes are currently functional. Computer vs. computer and computer vs. human are the two modes in which the game is automated. When the project began, the only item we had to work with was a Connect Four game. In order to automate the game we needed to construct a system that would automate the game play processes. These processes include:

- Determining which column to place the checker and placing the checker
- Determining which color checker needs to be played
- Determining when the game has been won and which player won
- Emptying the board when a winner has taken place

Figure 2 shows the system that was constructed to automate the Connect Four.

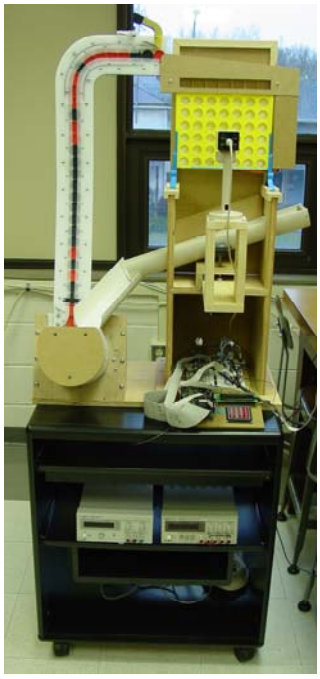

Figure 2

Autonomous Connect Four

System Description

A Micropac 535 development kit was used as the computer for this project. The Micropac 535 contains an 80C535 processor, external RAM, onboard A/D and D/A converters, analog I/O pins, digital input pins, a keypad, and an LCD display. The software written for the development board performs all the controls required to operate the game system.

Base

The base, seen in Figure 3, is the backbone of the Autonomous Connect Four. It is responsible for housing the Connect Four board, the return track, solenoids, and the camera. It provides stability for the previously mentioned items and serves as a way to attach the top track. The return track routes rejected checkers and checkers dumped at the end of a game back to the hopper so they may be used again.

Figure 3

System Base

Hopper

The hopper, designed by all, but constructed by Mr. Nick Schmidt and seen in Figure 4, is the main collection area for the checkers and also pushes the checkers into the feed track. Design of the hopper started with TurboCAD drawings to give a visual representation of what it would look like as well as the size. From the drawings, prototypes were constructed. These prototypes were useful in determining problems with

Figure 4

Hopper

the design. Once the problems were identified and resolved, construction of the final version began. A 110 VAC motor was used to turn the wheel of the hopper, which can be seen in Figure 5. An AC motor was chosen over a DC motor because the motor came

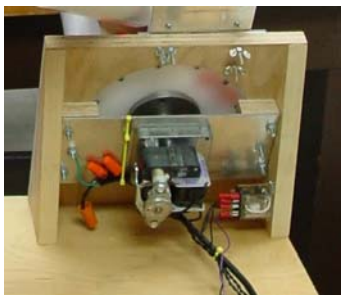

Figure 5

Back of Hopper Showing Motor and Relay

with an electric brake attached to the motor. If a DC motor were used braking software would need to be written to stop the motor. Another reason the AC motor was chosen was that the motor has a fixed speed of 7 RPM. Using a DC motor would have required speed control techniques to be utilized to run the motor at the desired speed. A 12VDC relay was used to control the motor. The relay was chosen for this application because the output signals of the Micropac 535 are DC signals, which are typically 5 Volts. One advantage of the Micropac is that the port 4 inputs and outputs are configured to be open collector which allows applications requiring more than 5 Volts to work with the system. The relay, which was connected to one of the port 4 output pins and the collector of this port pin was pulled up to 12 Volts using a resistor, allowed for the control of an AC motor using DC signals.

Feed Track

The feed track routes the checkers from the hopper to a position where the checkers are ready to be put in play. A prototype of the feed track was constructed using wood to determine and resolve design

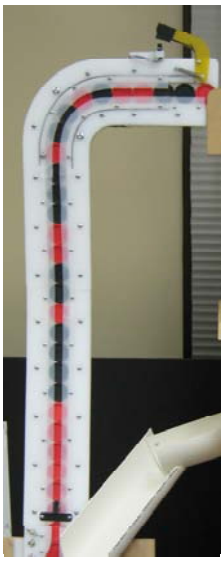

Figure 6

Feed Track

issues. It was determined that the feed track should not be constructed with wood because this resulted in excessive friction as the checkers move through it. The final product was constructed with UHMW polyehtylene; a slippery plastic material; and aluminum. These materials have lower coefficients of friction and allowed the checkers to move freely through the track. Since the checkers enter the feed track in a random fashion, color detection of the checkers became necessary.

Color Detection

The device that detects the color of the checkers is a QRB1134 optical switch. The optical switch is mounted on the backside of the feed track, at the eighth checker location as seen in Figure 7. As the checkers pass by the switch, light is reflected

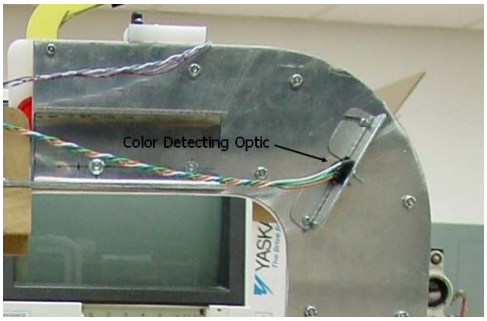

Figure7

Backside of Feed Track showing color detecting optic

from the emitter side of the switch to the detector side. Figure 8 illustrates how the optical switch works. The amount of light reflected by the object corresponds to a voltage level.

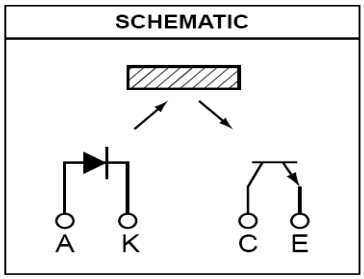

Figure 8

Schematic of QRB1134 Optical Switch

When there is not an object in front of the switch, there is 5 VDC at the collector of the detector. About the same voltage is present when a dark (not reflective) object is in front of the detector. When bright (reflective) objects pass in front of the switch the voltage level at the collector approaches 0 VDC. The DC voltage level corresponding to the checker color is the input to the onboard A/D converter. The volatage level for the

black checkers was 4.0 Volts and the red checkers were 1.0 Volts. Once the checker color has been converted to a digital representation, it is then stored in the queue_reg.

Queue

When the system is first powered, eight checkers are dispensed. The purpose for this is to load a queue in software with the colors of the first eight checkers that are ready to be dispensed. The reason for storing eight checker colors at a time is that the processor is an 8-bit processor. This allowed for each checker color to be stored in one 8-bit register called queue_reg with a logic 1 being a black checker and a logic 0 red. The software for the queue can be referenced in Appendix C. Utilizing a queue allows the system to operate more efficiently since the checkers are fed up a track in random order -- see Figure 6. The hopper motor will continue to run until the correct color checker has been dispensed while continuously updating the queue with new checker values as they passed up the feed track.

Checker Dispensing

As the checkers are pushed out the feed track by the hopper motor a trigger arm is actuated which breaks the beam of a H21A optical switch. Figure 9 shows the optical switch and its location. Once the beam of the optical switch is broken, the processor monitors it until the beam is no longer broken. If the dispensed checker was the incorrect color the hopper motor continues to run dispensing another checker. When a correctly colored checker is dispensed the hopper motor is turned off and the *checker stop* arm prevents any checkers from rolling out.

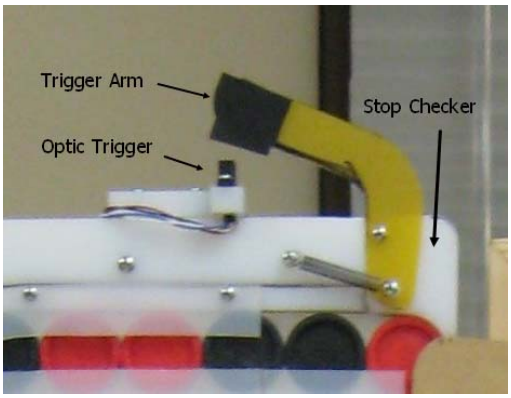

Figure 9

Checker Dispensing Mechanism

Top Track

The top track routes checkers that were dispensed into the desired column. Figure 10 shows the top track. The cut out window above each column permits a human to place their piece in the game board. This can be done either in conjunction with the

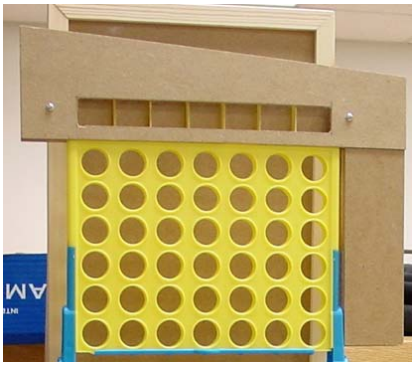

Figure 10

Top Track

system, or two people can play Connect Four without the system. These windows are required for the human vs. human game mode and the human portion of the human vs. computer game mode. An explanation is given later. The column to the right of the game board seen in Figure 10 is the reject track. When checkers of the wrong color are dispensed into the top track they travel through the reject track and then are returned to the hopper so they may be used later. In Figure 11, a side view of the top track shows the solenoids and trap doors used to divert a checker into the desired column of the game board. The home position of the trap doors does not allow access to the columns of

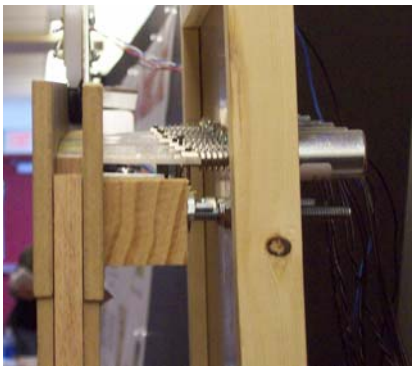

Figure 11

Side view of Top Track showing solenoids and trap doors

the game board through the top. This is the reason for the cut out windows used by humans. When the solenoids are energized, the trap door is pulled open and access to a column achieved. The solenoids used are 24 VDC pull-type. They are engaged using a TIP 32 PNP transistor as switch. Schematics for the transistor switch circuit can be referenced in Appendix B.

Menu System

Once the queue has been loaded with the colors of the first eight checkers, a menu system is displayed that guides the user through the program. From this menu the user has three options:

- Demo Mode
- Game Mode
- Test Mode

Currently, the Demo Mode performs no functions. The original idea of the demo mode was to dispense checkers into the game board in patterns to show that the system could detect the colors of the checkers correctly and place them in desired locations in the game board. Due to time constraints, this feature was not implemented but the system is capable of color detection and checker placement.

Game Mode

If the user enters this mode, a new menu is displayed which gives the user three options:

1. Return to Main Menu
2. Automatic Game
3. Manual Game

A message on the display lets the user know that they may return to the main menu by pressing the A key on the keypad.

Automatic Game

This game mode is the human vs. computer mode in which a player competes against the computer. A player starts a game by choosing which color checker to play. The player then will take a correctly colored checker out of the hopper and place it in the desired column. After the checker is played a button is pressed on the keypad indicating that their turn is done. Ideally the player would not have to press any keys after playing a checker. The location of the last checker would be determined through signal processing techniques applied to the video signal from the camera. Currently the image processing portion is not fully implemented resulting in the player needing to indicate to the computer in what column the checker was placed. Once the player's turn is finished, the computer will use the location of the last checker in an algorithm designed to determine whether this play has won the game. The algorithm checks to see if the last checker has completed a set of four in-a-row of the same color. If there is a winner, the winner light will flash, a message will be displayed indicating which color checker won, and the computer will clear the game board of any checkers. The same process will occur in the Manual Game when there is a winner. If there is no winner, the computer will calculate where to make its move. Offense and defense algorithms would determine where the next play should occur, had they been implemented in software. Currently the only algorithm programmed is a random play algorithm. The computer randomly selects one of the seven columns to play its checker in. This is done by using Timer 0 of the 80C535 processor. The timer is continually running in the background until it is time for the computer to play its checker. The timer is stopped at this point. Timer 0 is a 16 bit timer with its count values stored in two 8 bit registers, TH0 and TL0. The count value in TH0 then has its

upper 4 bits masked off which results in a binary number between 0 and 7. If the number is a 0 or 1, the checker will be placed in column 1. For the other numbers, 2 to 7, the checker will be placed in the corresponding column.

Once the computer has made its play, it will then check for a winner. If no winner has been detected the play returns to the human and the process will be repeated until a winner has occurred or the board becomes full. If the board fills without a winner, a message will be sent to the display indicating that the board is full and the computer will automatically empty it.

Image Processing

After the Player places a checker, the computer needs a way of determining where the checker was placed. Using a video camera the game board is checked. Only the position above any previous checkers, one in each column, need be checked. The new checker location is determined as is its color. The Achiever Black and White Security Camera is a wide angle analog video camera. A webcam was not used because the Micropac 535 does not have a USB interface. While the analog video camera is easier to interface with our system, the information stored in the signal is much harder to interpret.

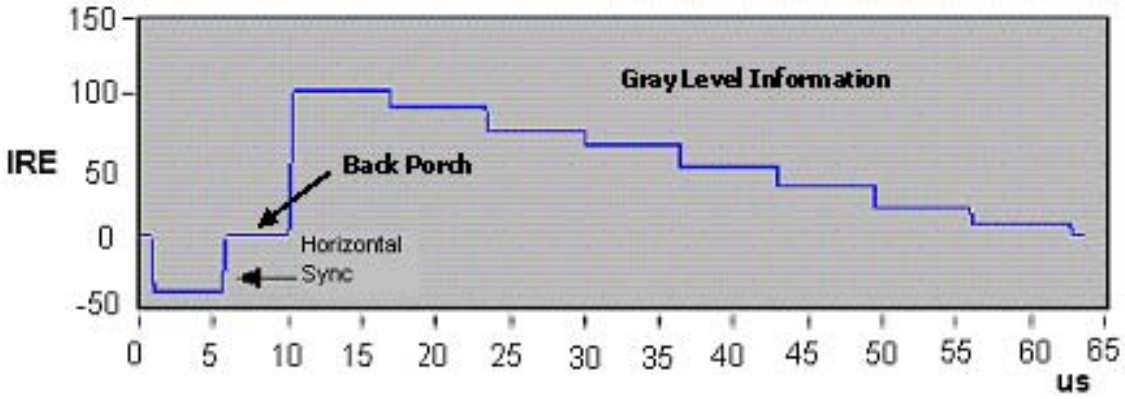

Figure 12

Black and White Video Signal Example

Shown in Figure 12 is a sample horizontal line that would be drawn across the screen of a TV. The horizontal sync is a sign to the TV that a new line is starting. This is followed by the back porch which is used as a black level reference for the rest of the line. From 10us until the next horizontal sync is all the gray level information.

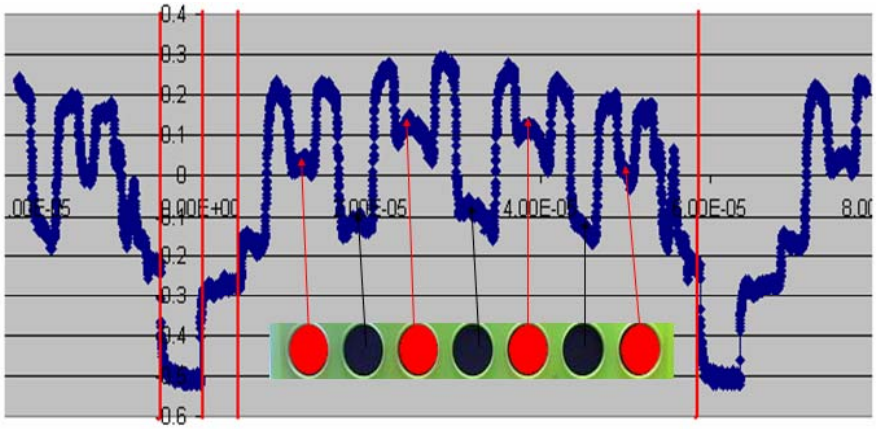

Figure 13

Sample Video Line from Project

Figure 13 shows a slice of a specially set up game board and the corresponding video signal line. There is a noticeable difference between the reds and the blacks on the board. While humans can visually see the differences, the hardware is not as capable. With the use of a LM1881 the luminance of the videosignal is extracted to a form that the hardware can use. The LM1881 is a video sync separator. It takes in a composite video signal and breaks it into TTL outputs for the vertical retrace, horizontal refresh, back porch, and the odd/even fields.

Signal	Indicates
Vertical Retrace	Start from top of screen
Horizontal Refresh	Start new line
Back Porch	Black level reference
Odd/Even	Which field the signal is in

Table 1

Video Signal Definitions

Vertical retrace and horizontal refresh are both used as interrupts in the hardware system. The vertical retrace resets the line count and the horizontal refresh increments the line count. With the help of the LM1881, the hardware can now count to a desired line. Once the hardware counts to its desired line, timing loops are used to wait a specific amount of time to sample and convert the information from a certain point in the line signal. The line signal itself runs for about 64 microseconds, but the information for one checker on

the board is available for only 3 microseconds. Because the on-board A/D os too slow to complete a conversion in 3 microseconds a fast A/D, ADC0820CCN, was used to convert the information into a digital format that the processor could read. The external A/D converts the signal in 1microsecond.

Only seven samples are needed to determine the change in the board, one per column. The software already knows the state of the board from all previous moves. For example, if there are already 3 checkers in column 4, then the software knows that it needs to sample the third row from the top in column 4. This is where each row has a specific line count that it has to count up to. Once the desired count equals the actual count the program waits a certain time to reach column 4 then turns on the fast A/D. After the conversion, the result is sampled to determine if there is a red, black, or no checker. If there is no checker then the program moves to the next column. If there is a red or black, the program stores its place and color value into a matrix format. This matrix is used by the search algorithm to determine winners and by the algorithms for game play to determine computer checker placement.

Manual Game

This game is a version of the human vs. human mode where the players play against one another without having to manually place the checkers in the game board. Each player uses the keypad to instruct the processor where to play the checker. Each player may choose any of the seven columns by pressing the corresponding key, 1 through 7, on the keypad. The camera is not utilized due to the computer knowing where it placed the checker. After each turn the computer will search for four in a row to

determine if one of the players has won the game. If a player has won the game, a message on the LCD display will appear showing which color checker has won. If no winner has been detected the computer queues up the next players checker in the first position of the feed track.

Search Method

The search method, used to check for a winner, must be completed after each player's turn and in all modes. This is performed by the computer searching the Color/Place matrix. The algorithm used searches around the checker that was last placed. First it checks downward to find a winner; if no winner is found the algorithm switches to checking to the right, then left, then the diagonals. If a winner is found, the computer will run the winner routine, which displays the winning color, flashes the winner light, empties the board, and resets the game. If it does not find a winner game play continues.

Test Mode

The test mode was created for the purpose of testing and debugging software and testing the hardware incorporated in the project. In this mode, software was written to drop a checker in each of the seven columns ignoring the color of the checker, reject a checker, dump the checkers from the game board, display a calibration program for the camera on a TV, and load the queue. A keypad interface is used in this mode. The assignments of the keys used are shown in Table 1.

Keys 0 and 9 load the queue. The technique of loading the queue was discussed in the Queue section. Keys 1 through 7 dispense a checker into their respective columns.

The B key dispenses a checker without activating any of the trap doors. Once the checker is dispensed, it travels down the top track and falls through the reject column. Once it passes through the reject column, the checker is funneled back into the hopper.

Key	Function
0	Loads queue
1	Drop checker in first column
2	Drop checker in second column
3	Drop checker in third column
4	Drop checker in fourth column
5	Drop checker in fifth column
6	Drop checker in sixth column
7	Drop checker in seventh column
9	Loads queue
A	Return to main menu
B	Rejects checker
C	Run calibration program
D	Dump checkers from board
E	Queues black checker
F	Queues red checker

Table 2

Keypad functions in Test Mode

When the C key is pushed, a calibration program for the camera is executed. The calibration program is used to align the camera with the game board to ensure that the correct data is sampled by the A/D converter. During the calibration program, white lines are written across the screen of a monitor which allows the user to visually position the camera correctly. The white lines correspond to the areas where checkers may be placed in the game board.

Pushing D results in engaging a linear actuator to move the release such that the checkers fall out of the board and back into the hopper. Once the checkers have fallen out of the board, the actuator moves the release back to its home position so more checkers may be played. The E key queues a black checker to the first position in the feed track, which is the position of the next checker ready to be played. The F key performs the same process as the E key but queues up a red checker instead of black to the first position. If none of the before mentioned commands are desired, the A key will take the user back to the main menu.

Conclusion

All mechanical systems currently function as desired and the human vs. human and human vs. computer game modes were achieved. While the end product did not have all the functionality envisioned, much was learned in the process. The heavy use of TurboCAD, to lay out the design of the mechanical system, proved invaluable for the manufacturing of individual parts. How NTSC analog video signals work and are broken up was a huge challenge. Understanding the signal, and the use of the LM1881, allowed the continued use of the readily available EMAC development kit. Planning out the project progress went smooth but was soon realized that many elements weren't accommodated. For example, a section of the project may have taken two months total. But, there were points where the section had to be put on hold to wait for a necessary part or even to wait for another section to be completed.

Appendix A: References

Data sheets for the following components can be found at the Autonomous Connect Four website: http://cegt201.bradley.edu/projects/proj2007/autocon4/datasheets.dwt

QRB1134 optic switch

LM1881 video sync separator

Micropac 535

Siemens 80C535 microprocessor

H21A optic switch

LMD18200 H-bridge

ADC0820 A/D converter

TIP32A PNP silicon transistor

Microelectronic Circuits 4th Edition, by Sedra, Adel S. / Smith, Kenneth C., Oxford University Press, June 1997

Construction materials: www.mcmaster.com

Hopper motor: www.digikey.com

Appendix B: Circuit Schematics and Design Equations

Circuits used in this project are shown in the following figures. Design equations to determine component values are also listed for the respective schematic.

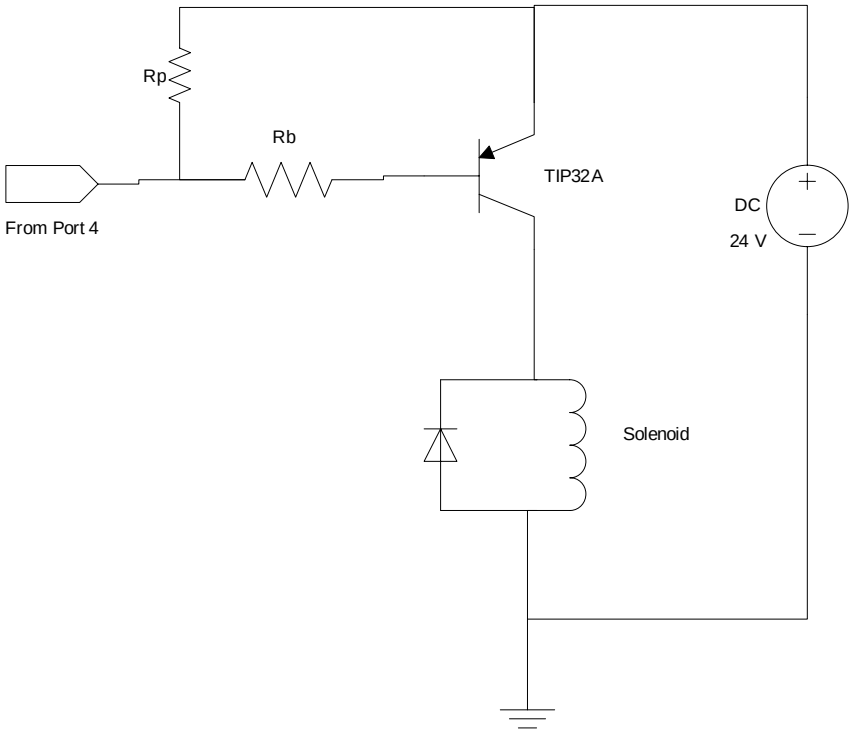

Figure B.1

Transistor Switch Circuit

Figure B.1 is the circuit used to switch the solenoids attached to the column trap doors. This circuit is replicated seven times as there are seven columns to choose from to place a checker. Each circuit is connected to port 4 of the Micropac using port pins 4.0 to 4.6 for

columns 1 to 7 of the game board, respectively. To activate the solenoid in Figure B.1, 1 Amp of current is required. The component values that need to be determined are Rp and Rb. Using standard design equations for bipolar junction transistors, the values of these resistors are determined. Rp was chosen to be 10kOhm to limit current drawn from power supply while the solenoids are not active. Using Equation B.1, the base current needed to drive the transistor into saturation was found to be 50 mA. The hfe value was obtained from the data sheet for the transistor Using this current in Equation B.2, the value of Rb was found to be 448 Ohms. The actual resistor used for Rb was 470 Ohm since there is not a 448 Ohm stock resistor.

$$Ib = Ic/hfe$$

Equation B.1

Transistor Base Current and Collector Current Relationship

$$Vcc = Vbe + Rb*Ib + Vce(port\ 4)$$

Equation B.2

KVL Equation from Supply Voltage to Ground of Port 4 Transistor

Figure B.2 shows the schematic for the circuit used to run the hopper motor. No design equations were needed for this circuit. The coil of the relay was allowed to draw as much current from the power supply as needed to switch the contact side of the relay. The relay was used because the hopper motor is an AC motor. Port 4.7 was used to activate the relay for the motor.

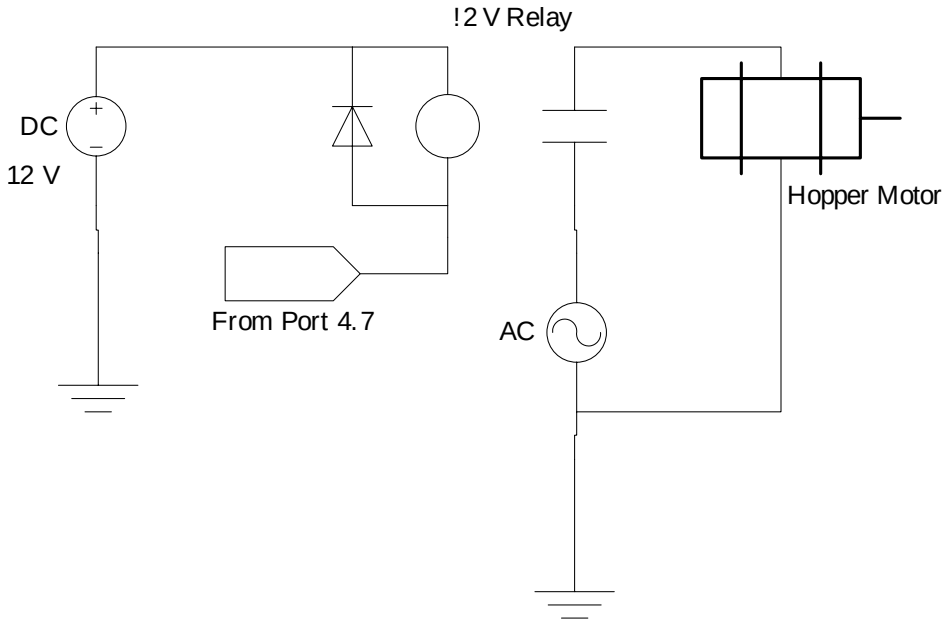

Figure B.2

Circuit used to Switch the Hopper Motor

The schematic for the QRB1134 optical switch and the H21A optical switch is illustrated in Figure B.3. For the QRB1134, Rd was chosen to be 10kOhm to limit the current of the detector side of the switch. Re was found based on the forward current, If, and the forward voltage, Vf, required to saturate the transistor of the detector side of the switch. If and Vf were found to be 35mA and .7V, respectively, from the data sheet for

the QRB1134 optical switch. Using Equation B.3, the value of Re was determined to be 120 Ohms. The output is connected to channel 0 of the A/D converter of the Micropac.

For the H21A, Rd was chosen to be 10kOhm. From the data sheet, If and Vf were found to be 40mA and .7V. Equation B.3 was used to find the value for Re which was determined to be x Ohms. The output of the optical switch is connected to Port 1.3 of the Micropac.

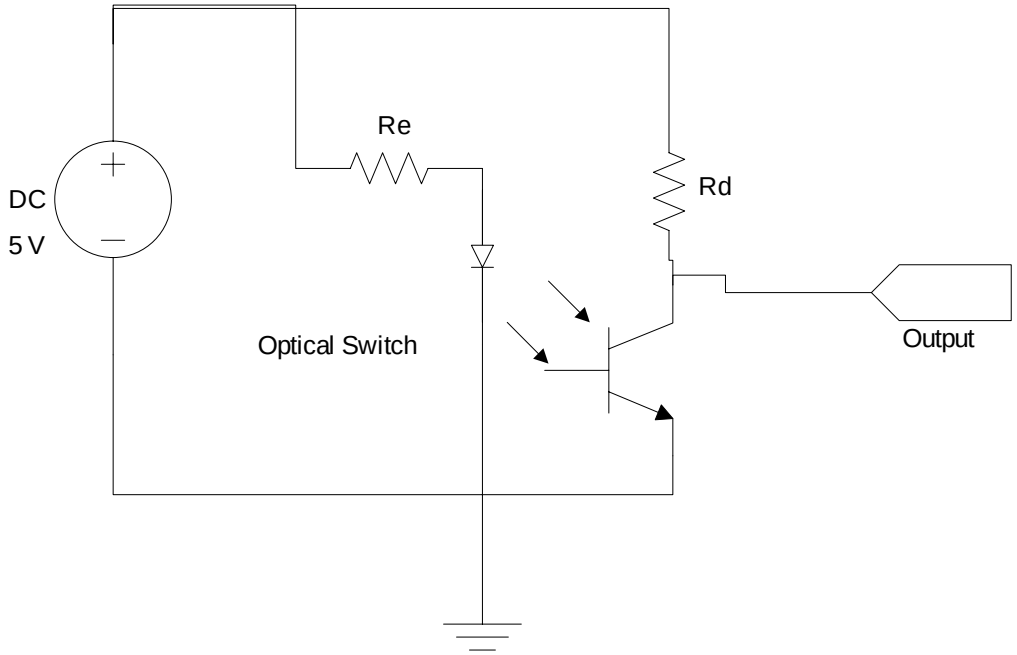

Figure B.3

Schematic for QRB1134 Optical Switch

$$Vs = If*Re + Vf$$

Equation B.3

KVL from Supply Voltage to Ground of Emitter Side of Switch

Figure B.4 is a block diagram of the LMD18200 H-bridge used to drive the actuator that empties the game board. The supply voltage, Vs, connected to pin 6 is 12 VDC. The actuator inputs are connected to the bidirectional output pins 2 and 10. Pin 3 is connected to Port 1.x of the Micropac which controls the polarity of the output pins 2 and 10. Port 1.x of the Micropac is connected to the input, pin 5, of the H-bridge. When the input is high, the output is active and when the input is low the output is inactive. Pins 4 and 7 of the H-bridge are tied to ground and pins 1, 8, 9, and 11 are not used.

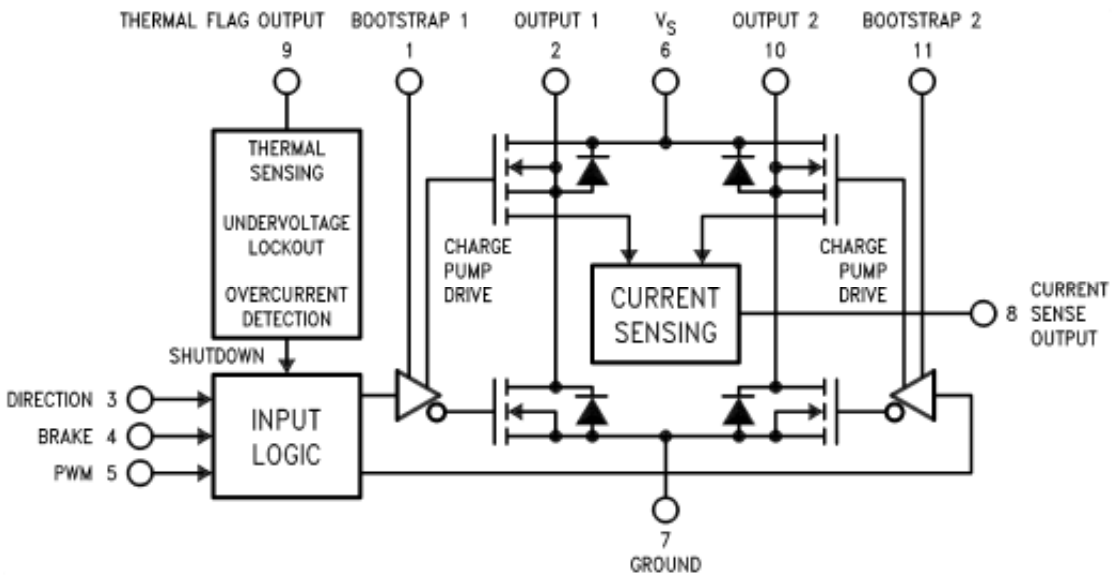

Figure B.4

Block Diagram of LMD18200 H-bridge

**Insert schematics and design equations for the LM1881 and the Op-Amp and the flash A/D.

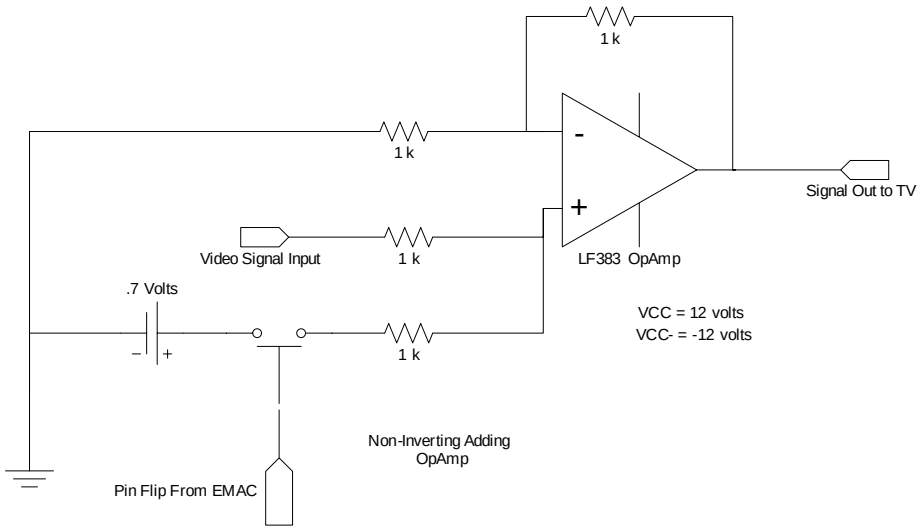

Figure B.5

Non-Inverting OpAmp circuit used for white level

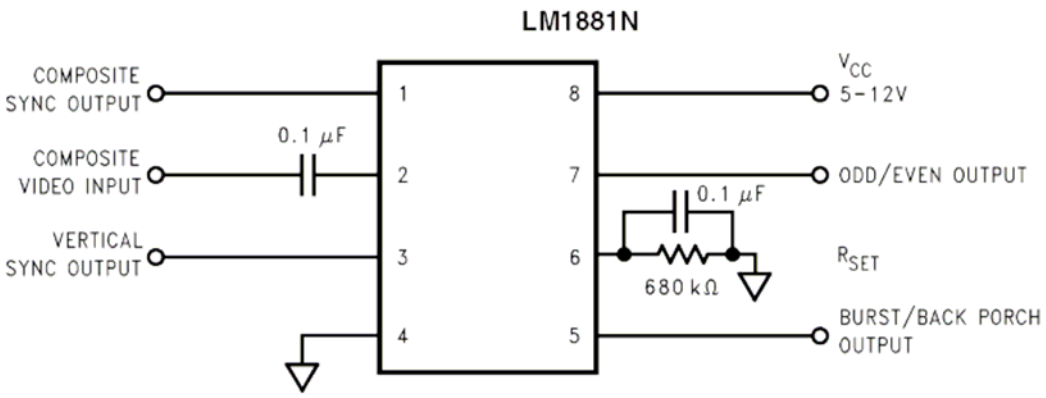

Figure B.5

Video Sync Separator Pin Out

Appendix C: Software

```
;*****************************************************************
;
;Justin Middleton
;Bradley University
;
;*****************************************************************


$SAVE
$NOLIST

; Define temp storage locations for registers
; using bank 3 data locations (bank 3 mem range)

tmpa       DATA          018h
descnt     DATA          019h
tmpq       DATA          020h
opset      DATA          0eh
opreset    DATA          0fh




;define byte variables for my use (range 30h-60h)
col          DATA        030h
testcnt      DATA        031h
linecnt      DATA        032h
dcount1              DATA        033h
dcount2      DATA        034h
color        DATA        035h
queuecnt     DATA        036h
queue_reg    DATA        037h
switchcnt    DATA        038h
loopcnt      DATA        039h
colwait      DATA        03ah
col1cnt      DATA        047h
col2cnt      DATA        048h
```

```
col3cnt     DATA        049h
col4cnt     DATA        04ah
col5cnt     DATA        04bh
col6cnt     DATA        04ch
col7cnt     DATA        04dh
startmeml    DATA       04eh
startmemh    DATA       04fh
samecolorcnt DATA       050h
present1     DATA       051h
present2     DATA       052h
present3     DATA       053h
present4     DATA       054h
present5     DATA       055h
present6     DATA       056h
color1       DATA       057h
color2       DATA       058h
color3       DATA       059h
color4       DATA       05ah
color5       DATA       05bh
color6       DATA       05ch


; Define a bit addressable variables for my use
brdfull      BIT        03h
col1         BIT 04h
col2         BIT  05h
col3         BIT  06h
col4         BIT  07h
col5         BIT  08h
col6         BIT  09h
next_sample  BIT        0ah
p1color      BIT        0bh
player1      BIT        0ch
col7         BIT  0ah
newcnt       BIT          20h
vertref      BIT 21h
checker      BIT        22h
qflag        BIT        23h
drop1        BIT        24h
disp         BIT        24h
strtgme      BIT        25h
colorbit     BIT        26h
bwinner      BIT        27h
rwinner      BIT        28h
col1ful      BIT        29h
col2ful      BIT        2ah
```

```
col3ful     BIT     2bh
col4ful     BIT     2ch
col5ful     BIT     2dh
col6ful     BIT     2eh
col7ful     BIT     2fh



$RESTORE


;*************************************************************************
;Justin Middleton and Wayne Bogart
;Bradley University
;Electrical Engineering Dept.
;
;init.A51
;Created: 3-16-07
;Last Updated: 4-12-07
;
;Input:
;Output:
;Corrupted Registers:  r0
;
;*************************************************************************
*
$nomod51
$Include(reg515.inc)
$Include(myvars.inc)

name    init_mod
extrn   code(lcdinit,main,nopower,dlytime,lcdout)
start   EQU   8000h
cseg    at      8000h
jmp     init

        CSEG  AT start + 005Bh
        INC   testcnt               ;increments line counter
        RETI

        CSEG  AT start + 0063H
        MOV   testcnt,#00h          ;resets line counter
        RETI
```

```
init_seg        segment         code
rseg    init_seg

;***********************************************
;
;       Initialization Module
;
;***********************************************


init:           call    lcdinit
                mov     a, #1bh         ; turns cursor off
                call    lcdout
                call    dlytime
                mov     a, #0ch
                call    lcdout

                call    nopower
                setb    p3.3
                mov     r0, #0ffh
clr_ram:        mov     @r0, #0
                djnz    r0, clr_ram

;***********************************************
;
;     Set solenoids to 0 volts
;
;***********************************************
          mov p4, #255
          clr   p1.5
                clr     p1.7

                mov     descnt, #9ch
                mov     colwait, #01h

                mov     a, tmod
                orl     a, #00000001b
                anl     a, #11110001b
                mov     tmod, a
                setb    tr0
                ;clr    op2


        MOV   40h,#40h
```

```
        MOV   41h,#5dh
        MOV   42h,#7ch
        MOV   43h,#9Ch
        MOV   44h,#0Bdh
        MOV   45h,#0Dah             ;preset values for line count
        MOV   46h,#00h

              mov       loopcnt, #60
iloop:          call    dlytime
                dec     loopcnt
                mov     a, loopcnt
                jnz     iloop

                jmp     main
end


;*************************************************************************
;Justin Middleton and Wayne Bogart
;Bradley University
;Electrical Engineering Dept.
;
;rmain.A51
;Created: 3-16-07
;Last Updated: 4-12-07
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name   main_mod
public  main
extrn
        code(keypad,dlytime,queue,calib,display1,menudisp,lcdout,test,game,powerok,ql
oading,load,qloaded)
```

```
main_seg        segment         code
rseg    main_seg

;************************************************
;
;       Main Module
;
;************************************************


;************************************************
;
;       Main program that links modules together
;       to run program correctly
;
;************************************************

main:       setb        eal

                call    powerok
            mov         loopcnt, #60
mloop:          call    dlytime
                dec     loopcnt
                mov     a, loopcnt
                jnz     mloop


                call    display1

            mov         loopcnt, #100
m1loop:                 call    dlytime
                dec     loopcnt
                mov     a, loopcnt
                jnz     m1loop

                call    qloading
                call    load
                call    qloaded

            mov         loopcnt, #60
m2loop:                 call    dlytime
                dec     loopcnt
                mov     a, loopcnt
                jnz     m2loop
```

```
main2:          call    menudisp


main5:          call    keypad


check_1:        cjne    a, #31h, check_2     ; checks for 1 key

            ;call demo
                jmp     main2


check_2:       cjne     a, #32h, check_3     ; checks for 2 key

               call     test
                jmp     main2


check_3:        cjne    a, #33h, check_return
                call    game
                jmp     main2


check_return:  cjne     a, #61h, nodata      ; checks for a key

nodata:         jmp     main5

end



$NOMOD51
$Include(reg515.inc)

Name  lcd
PUBLIC        lcdinit

lcdinit_seg     SEGMENT     CODE
                RSEG  lcdinit_seg
                USING         0

;*****************************************************************************
*
;
```

```
; 20X2 character LCD drivers for the MICROPAC 535
;
; COPYRIGHT 1993-1995 EMAC INC.
;
; ESCflag is a bit field that must be declared.
; DLAYA is a 5ms delay routine with no registers affected.
; LCDINIT should be executed before any other LCD subroutines.
;
;*************************************************************************
*

ESCflag         equ     psw.5           ; LCD equate
lcdcmd          equ     28h     ; value for P2 to select lcd command port

initdata:
        db      38h,08,01,06,0eh,80h,0


;
; LCDINIT: Init the LCD
;
LCDINIT:
        clr     ESCflag         ; indicate no esc found
        MOV     P2,#LCDCMD      ; POINT TO COMMAND PORT
        LCALL   DLAYA           ; 5MS DELAY
        LCALL   DLAYA           ; 5MS DELAY
        LCALL   DLAYA           ; 5MS DELAY
        LCALL   DLAYA           ; 5MS DELAY
        MOV     A,#30H
        MOVX    @R1,A           ; OUT TO LCD COMMAND PORT
        LCALL   DLAYA           ; 5MS DELAY
        MOVX    @R1,A           ; OUT TO LCD COMMAND PORT
        LCALL   DLAYA           ; 5MS DELAY
        MOVX    @R1,A           ; OUT TO LCD COMMAND PORT

        MOV     DPTR,#INITDATA  ; POINT TO INIT DATA
                ; the last command should take no more than 40 uS.
        mov     b,#80           ; for timeout of 80*3 * (12/clock)
lcdnit2:
        movx    a,@r1           ; read lcd command port
        jnb     acc.7,lcdnit1   ; exit if not busy
        djnz    b,lcdnit2       ; loop till timeout
        sjmp    lcdexit ; exit if timeout

LCDNIT1:
        MOVX    A,@R1           ; READ LCD COMMAND PORT
```

```
        JB      ACC.7,LCDNIT1    ; LOOP IF BUSY FLAG SET
        CLR     A
        MOVC        A,@A+DPTR       ; GET BYTE FROM INIT TABLE
        JZ      LCDEXIT         ; EXIT IF 0
        INC     DPTR            ; POINT TO NEXT BYTE
        MOVX        @R1,A           ; OUTPUT BYTE
        SJMP        LCDNIT1         ; LOOP

LCDEXIT:
        RET


;
; MISCELLANEOUS DELAYS
;
DLAYA:      PUSH  ACC
        MOV         A,#100
        AJMP        DLAYA2

DLAYB:      PUSH        ACC
        MOV         A,#128
        AJMP        DLAYA2

DLAYC:      PUSH  ACC
        MOV         A,#255
        AJMP        DLAYA2
dlayd:  push    acc
        mov     a,#8


DLAYA2:
        PUSH  ACC
        MOV         A,#0FFH
DLAYA1:
        MOV         A,#0FFH
        DJNZ  ACC,$         ; LEVEL 3 LOOP
        POP     ACC
        DJNZ  ACC,DLAYA2        ; LEVEL 1 LOOP

        POP     ACC
        RET


        END
```

```
;*************************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;keypad.A51
;Created: 11-27-04
;Last Updated: 11-27-04
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*

$nomod51
$Include(reg515.inc)
$Include(myvars.inc)

name   keypad_mod
public  keypad
extrn   code(main)




keypad_seg    segment         code
rseg    keypad_seg


;*****************************************************
;
;      Keypad Module
;
;      Detrmines which key is pressed and returns
;      the value in the accumulator
;
;*****************************************************



get_key:        db      '123c456d789ea0bf'
```

```
keypad:         jnb     tcon.3, keypad
        mov     dph, #30h
        movx    a, @dptr
        anl     a, #00011111b
        mov     dptr, #get_key
        movc    a, @a+dptr
        clr     tcon.3
        ret

end
;Justin Middleton and Wayne Bogart
;Bradley University
;Electrical Engineering Dept.
;
;automat.A51
;Created: 4-12-07
;Last Updated: 4-12-07
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name    automat_mod
public  automat
extrn
        code(keypad,dlytime,queue,qred,qblack,p1select,strtgmem,player1d,player2d,gm
eover,dumprout,dispense,storemem,search,blackwin,redwin,compturn,humand,fullbrd,no
twin)


automat_seg    segment         code
rseg    automat_seg

;*********************************************
;
```

```
;         Automatic Module
;
;************************************************


;************************************************
;
;         Plays automatic game
;
;************************************************


automat:        mov     dptr, #8000h
                mov     loopcnt, #72
goaloop:        mov     a, #0ffh
                movx    @dptr, a
                inc     dptr
                dec     loopcnt
                mov     a, loopcnt
                jnz     goaloop

                call    strtgmem
                clr     player1
                clr     strtgme
                mov     col1cnt, #0
            mov col2cnt, #0
                mov     col3cnt, #0
                mov     col4cnt, #0
                mov     col5cnt, #0
                mov     col6cnt, #0
                mov     col7cnt, #0
                clr     brdfull
                clr     col1ful
                clr     col2ful
                clr     col3ful
                clr     col4ful
                clr     col5ful
                clr     col6ful
                clr     col7ful


automat1:       call    keypad

check_0a:       cjne    a, #30h, check_aa       ; checks for 0 key
                                ; go to start display
```

```
                call    p1select
                jmp     automat1

check_aa:       cjne    a, #61h, check_ba       ; checks for a key
                                                ; returns to main menu
                jnb     strtgme, manout
                call    dumprout
manout:         ret
                jmp     automat

check_ba:       cjne    a, #62h, check_ca       ; checks for b key
                                ; sets player 1 color to black
                jb      strtgme, automat1
                setb    p1color
                setb    strtgme
                setb    player1
                jmp     human       ;

check_ca:       cjne    a, #63h, check_1a       ; checks for c
                                        ; sets player1 color to red
                jb      strtgme, automat1
                clr     p1color
                setb    strtgme
                setb    player1
                jmp     human

;check_da:      jmp     automat1



partway:        jmp     automat1

check_1a:       cjne    a, #31h, check_2a       ; checks for 1 key


                jnb     player1, automat1
                mov     a, col1cnt
                cjne    a, #6, go1
                jmp     partway
go1:            setb    col1
                ;inc    col1cnt
                jmp     checkwin
```

```
check_2a:     cjne    a, #32h, check_3a     ; checks for 2 key

              jnb   player1, automat1
                    mov     a, col2cnt
                    cjne    a, #6, go2
                    jmp     partway
go2:                setb    col2
                    ;inc    col2cnt
                    jmp     checkwin


check_3a:     cjne    a, #33h, check_4a     ; checks for 3 key

              jnb   player1, automat1
                    mov     a, col3cnt
                    cjne    a, #6, go3
                    jmp     partway1
go3:                setb    col3
                    ;inc    col3cnt
                    jmp     checkwin

partway1:     jmp     partway


check_4a:     cjne    a, #34h, check_5a     ; checks for 4 key

                     jnb    player1, automat1
                    mov     a, col4cnt
                    cjne    a, #6, go4
                    jmp     partway1
go4:                 setb   col4
                    ;inc    col4cnt
                    jmp     checkwin

check_5a:     cjne    a, #35h, check_6a     ; checks for 5 key

              jnb   player1, partway1
                    mov     a, col5cnt
                    cjne    a, #6, go5
                    jmp     partway2
go5:                setb    col5
                    ;inc    col5cnt
                    jmp     checkwin

partway2:     jmp     partway1
automat2:     jmp     automat
```

```
check_6a:     cjne    a, #36h, check_7a     ; checks for 6 key

              jnb     player1, partway1
                mov   a, col6cnt
                cjne  a, #6, go6
                jmp   partway2
go6:            setb  col6
                ;inc  col6cnt
                jmp   checkwin



check_7a:     cjne    a, #37h, check_da
                jnb   player1, partway2
                mov   a, col7cnt
                cjne  a, #6, go7
                jmp   partway2
go7:            setb  col7
                ;inc  col7cnt
                jmp   checkwin          ; must be 7 if here

                jmp   partway

check_da:     jnb     player1, partway2
                call  gmeover
                call  dumprout
                jmp   automat2

human:      call      humand
                jmp   automat1


checkwin:       call  storemem
                call  search


                jnb   bwinner, redwinnera
                mov   dph, #opreset     ; turn on winner light
                mov   a, #4
                movx  @dptr, a
                call  blackwin
                clr   bwinner

                mov   loopcnt, #100
go1loopa:       call  dlytime
```

```
                dec     loopcnt
                mov     a, loopcnt
                jnz     go1loopa
                call    dumprout
                clr     strtgme
                mov     dph, #opset         ; turn off winner light
                mov     a, #4
                movx    @dptr, a

                call    gmeover             ; game over display
                mov     loopcnt, #100
go2loopa:       call    dlytime
                dec     loopcnt
                mov     a, loopcnt
                jnz     go2loopa

                jmp     automat
human1:         jmp     human

redwinnera:     jnb     rwinner, nowinnera
                call    redwin
                clr     rwinner
                mov     dph, #opreset       ; turn on winner light
                mov     a, #4
                movx    @dptr, a

                mov     loopcnt, #100
go3loopa:       call    dlytime
                dec     loopcnt
                mov     a, loopcnt
                jnz     go3loopa
                call    dumprout
                clr     strtgme
                mov     dph, #opset         ; turn off winner light
                mov     a, #4
                movx    @dptr, a

                call    gmeover             ; game over display
                mov     loopcnt, #100
go4loopa:       call    dlytime
                dec     loopcnt
                mov     a, loopcnt
                jnz     go4loopa

                jmp     automat
```

```
nowinnera:

                call    fullbrd         ; insert each column full check
                jnb     brdfull, nope   ; if all columns full and no winner, dump board
and reset game
                call    notwin
                mov     loopcnt, #60
gonoloopa:      call    dlytime
                dec     loopcnt
                mov     a, loopcnt
                jnz     gonoloopa
                call    gmeover
                call    dumprout
                jmp     automat
nope:           cpl     player1
                clr     col1
                clr     col2
                clr     col3
                clr     col4
                clr     col5
                clr     col6
                clr     col7
                jb      player1, human1
                call    compturn

                mov     loopcnt, #100
gocloopa:       call    dlytime
                dec     loopcnt
                mov     a, loopcnt
                jnz     gocloopa
                jb      p1color, compred
                call    qblack
                jmp     compplay

compred:        call    qred

compplay:       clr     tr0
                mov     a, th0
                anl     a, #0000111b
                setb    tr0

number7:        cjne    a, #7, number6

                mov     a, col7cnt
                cjne    a, #6, play7a
                setb    col7ful
```

```
                mov     a, #6
                jmp     number6
play7a:         setb    col7
                ;inc    col4cnt
                clr     p4.7
                call    dlytime
wait7a:         jnb     p1.3, wait7a
                call    dispense
                call    queue
                mov     a, queue_reg
                mov     c, next_sample
                rrc     a
                mov     queue_reg, a
                jmp     checkwin

number6:        cjne    a, #6, number5

                mov     a, col6cnt
                cjne    a, #6, play6a
                setb    col6ful
                mov     a, #5
                jmp     number5
play6a:         setb    col6
                ;inc    col4cnt
                clr     p4.7
                call    dlytime
wait6a:         jnb     p1.3, wait4a
                call    dispense
                call    queue
                mov     a, queue_reg
                mov     c, next_sample
                rrc     a
                mov     queue_reg, a
                jmp     checkwin

number5:        cjne    a, #5, number4

                mov     a, col5cnt
                cjne    a, #6, play5a
                setb    col5ful
                mov     a, #4
                jmp     number4
play5a:         setb    col5
                ;inc    col4cnt
                clr     p4.7
                call    dlytime
```

```
wait5a:     jnb     p1.3, wait5a
            call    dispense
            call    queue
            mov     a, queue_reg
            mov     c, next_sample
            rrc     a
            mov     queue_reg, a
            jmp     checkwin

number4:    cjne    a, #4, number3

            mov     a, col4cnt
            cjne    a, #6, play4a
            setb    col4ful
            mov     a, #3
            jmp     number3
play4a:     setb    col4
            ;inc    col4cnt
            clr     p4.7
            call    dlytime
wait4a:     jnb     p1.3, wait4a
            call    dispense
            call    queue
            mov     a, queue_reg
            mov     c, next_sample
            rrc     a
            mov     queue_reg, a
            jmp     checkwin

number3:    cjne    a, #3, number2

            mov     a, col3cnt
            cjne    a, #6, play3a
            setb    col3ful
            mov     a, #2
            jmp     number2
play3a:     setb    col3
            ;inc    col4cnt
            clr     p4.7
            call    dlytime
wait3a:     jnb     p1.3, wait3a
            call    dispense
            call    queue
            mov     a, queue_reg
            mov     c, next_sample
            rrc     a
```

```
                mov     queue_reg, a
                jmp     checkwin

number2:        cjne    a, #2, number1

                mov     a, col2cnt
                cjne    a, #6, play2a
                setb    col2ful
                mov     a, #1
                jmp     number1
play2a:         setb    col2
                ;inc    col4cnt
                clr     p4.7
                call    dlytime
wait2a:         jnb     p1.3, wait2a
                call    dispense
                call    queue
                mov     a, queue_reg
                mov     c, next_sample
                rrc     a
                mov     queue_reg, a
                jmp     checkwin

number1:        mov     a, col1cnt
                cjne    a, #6, play1a
                setb    col1ful
                mov     a, #7
                jmp     number7
play1a:         setb    col1
                ;inc    col4cnt
                clr     p4.7
                call    dlytime
wait1a:         jnb     p1.3, wait1a
                call    dispense
                call    queue
                mov     a, queue_reg
                mov     c, next_sample
                rrc     a
                mov     queue_reg, a
                jmp     checkwin


end

;*****************************************************************************
;Justin Middleton
```

```
;Bradley University
;Electrical Engineering Dept.
;
;manual.A51
;Created: 4-2-07
;Last Updated: 4-2-07
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name   manual_mod
public  manual
extrn
        code(keypad,dlytime,queue,qred,qblack,p1select,strtgmem,player1d,player2d,gm
eover,dumprout,dispense,storemem,search,blackwin,redwin)


manual_seg    segment         code
rseg    manual_seg

;************************************************
;
;       Manual Module
;
;************************************************


;************************************************
;
;       Plays game manually through keypad
;
;************************************************
```

```
manual:     mov     dptr, #8000h
            mov     loopcnt, #72
gomloop:    mov     a, #0ffh
            movx    @dptr, a
            inc     dptr
            dec     loopcnt
            mov     a, loopcnt
            jnz     gomloop

            call    strtgmem
            clr     player1
            clr     strtgme
            mov     col1cnt, #0
        mov col2cnt, #0
            mov     col3cnt, #0
            mov     col4cnt, #0
            mov     col5cnt, #0
            mov     col6cnt, #0
            mov     col7cnt, #0

manual1:    call    keypad

check_0m:   cjne    a, #30h, check_am     ; checks for 0 key
                            ; go to start display

            call    p1select
            jmp     manual1

check_am:   cjne    a, #61h, check_bm     ; checks for a key
                                ; returns to main menu
            jnb     strtgme, manout
            call    dumprout
manout:     ret
            jmp     manual1

check_bm:   cjne    a, #62h, check_cm     ; checks for b key
                        ; sets player 1 color to black
            setb    p1color
            setb    strtgme
            setb    player1
            jmp     firstplay     ;

check_cm:   cjne    a, #63h, check_dm     ; checks for c
                                ; sets player1 color to red
            clr     p1color
            setb   strtgme
```

```
                setb    player1
                jmp     firstplay


check_dm:       cjne    a, #64h, check_1m      ; checks for d key

                jnb     strtgme, manual1       ; dump board
            call  dumprout
                clr     strtgme

                call    gmeover              ; game over display
                mov     loopcnt, #100
goloop:                 call     dlytime
                dec     loopcnt
                mov     a, loopcnt
                jnz     goloop

                jmp     manual


check_1m:       cjne    a, #31h, check_2m      ; checks for 1 key

                jnb     strtgme, manual1
                mov     a, col1cnt
                cjne    a, #6, play1
                jmp     manual1
play1:          setb    col1
                ;inc    col1cnt
                clr     p4.7
                call    dlytime
wait1m:                 jnb      p1.3, wait1m
                call    dispense
                call    queue
                mov     a, queue_reg
                mov     c, next_sample
                rrc     a
                mov     queue_reg, a

                jmp     nextplay

break4:         jmp     manual

check_2m:        cjne   a, #32h, check_3m      ; checks for 2 key

                jnb     strtgme, manual1
                mov     a, col2cnt
```

```
                cjne    a, #6, play2
                jmp     manual1
play2:          setb    col2
                ;inc    col2cnt
                clr     p4.7
                call    dlytime
wait2m:                 jnb     p1.3, wait2m
                call    dispense
                call    queue
                mov     a, queue_reg
                mov     c, next_sample
                rrc     a
                mov     queue_reg, a
            jmp nextplay

break:          jmp     manual1

check_3m:       cjne    a, #33h, check_4m      ; checks for 3 key

            jnb strtgme, break
                mov     a, col3cnt
                cjne    a, #6, play3
                jmp     break
play3:          setb    col3
                ;inc    col3cnt
                clr     p4.7
                call    dlytime
wait3m:                 jnb     p1.3, wait3m
                call    dispense
                call    queue
                mov     a, queue_reg
                mov     c, next_sample
                rrc     a
                mov     queue_reg, a
            jmp nextplay

check_4m:       cjne    a, #34h, check_5m      ; checks for 4 key

                jnb     strtgme, break
                mov     a, col4cnt
                cjne    a, #6, play4
                jmp     break
play4:          setb    col4
                ;inc    col4cnt
                clr     p4.7
                call    dlytime
```

```
wait4m:         jnb     p1.3, wait4m
                call    dispense
                call    queue
                mov     a, queue_reg
                mov     c, next_sample
                rrc     a
                mov     queue_reg, a
                jmp     nextplay

check_5m:       cjne    a, #35h, check_6m     ; checks for 5 key

                jnb     strtgme, break
                mov     a, col5cnt
                cjne    a, #6, play5
                jmp     break
play5:          setb    col5
                clr     p4.7
                call    dlytime
wait5m:         jnb     p1.3, wait5m
                call    dispense
                call    queue
                mov     a, queue_reg
                mov     c, next_sample
                rrc     a
                mov     queue_reg, a
                jmp     nextplay

break1:         jmp     break
break3:         jmp     break4

check_6m:       cjne    a, #36h, check_7m     ; checks for 6 key

                jnb     strtgme, break1
                mov     a, col6cnt
                cjne    a, #6, play6
                jmp     break1
play6:          setb    col6
            ;   inc     col6cnt
                clr     p4.7
                call    dlytime
wait6m:         jnb     p1.3, wait6m
                call    dispense
                call    queue
                mov     a, queue_reg
                mov     c, next_sample
                rrc     a
```

```
                mov     queue_reg, a
                jmp     nextplay

check_7m:                       ; must be 7 if here
                jnb     strtgme, break1
                mov     a, col7cnt
                cjne    a, #6, play7
                jmp     break1
play7:          setb    col7
                ;inc    col7cnt
                clr     p4.7
                call    dlytime
wait7m:                 jnb     p1.3, wait7m
                call    dispense
                call    queue
                mov     a, queue_reg
                mov     c, next_sample
                rrc     a
                mov     queue_reg, a
                jmp     nextplay

;*****************************************************************
;
;  Queue's correct checker for Player 1 first move
;
;*****************************************************************

firstplay:      call        player1d
                  jnb       p1color, p1red
                  call      qblack
                  jmp       break1

p1red:          call        qred
                  jmp       break1


nextplay:       call    storemem
                call    search
                jnb     bwinner, redwinner
                mov     dph, #opreset           ; turn on winner light
                mov     a, #4
                movx    @dptr, a
                ;setb   op2
                call    blackwin
                clr     bwinner
```

```
                mov     loopcnt, #100
go1loop:        call    dlytime
                dec     loopcnt
                mov     a, loopcnt
                jnz     go1loop
                call    dumprout
                clr     strtgme
                ;clr    op2
                mov     dph, #opset         ; turn off winner light
                mov     a, #4
                movx    @dptr, a

                call    gmeover             ; game over display
                mov     loopcnt, #100
go2loop:        call    dlytime
                dec     loopcnt
                mov     a, loopcnt
                jnz     go2loop

                jmp     break3

redwinner:      jnb     rwinner, nowinner
                mov     dph, #opreset         ; turn on winner light
                mov     a, #4
                movx    @dptr, a
                call    redwin
                clr     rwinner

                mov     loopcnt, #100
go3loop:        call    dlytime
                dec     loopcnt
                mov     a, loopcnt
                jnz     go3loop
                call    dumprout
                clr     strtgme
                mov     dph, #opset         ; turn off winner light
                mov     a, #4
                movx    @dptr, a

                call    gmeover             ; game over display
                mov     loopcnt, #100
go4loop:        call    dlytime
                dec     loopcnt
                mov     a, loopcnt
                jnz     go4loop
```

```
                jmp     break3

nowinner:       cpl     player1
                jnb     player1, player2
                jmp     firstplay


player2:        call    player2d
                jnb     p1color, p2black
            call  qred
                jmp     break

p2black:       call     qblack
                jmp     break


end

;*************************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;test.A51
;Created: 11-29-04
;Last Updated: 12-7-04
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)

name   test_mod
public  test
extrn
        code(keypad,dlytime,queue,calib,display1,menudisp,lcdout,qloading,qloaded,feed
back,load,dumprout,dispense,scan)
```

```
test_seg        segment         code
rseg    test_seg

;***********************************************
;
;       Test Module
;
;***********************************************


;***********************************************
;
;       Tests components of system
;
;
;***********************************************

test:           ;call   qloading
                ;mov    queuecnt, #8
                ;call   load

;               clr     disp            ; loads intial queue
;           setb qflag
;               mov     queue_reg, #0
;               mov     queuecnt, #8
;               call    queue
;               mov     a, queue_reg
;               rrc     a
;               mov     queue_reg, a
;               dec     queuecnt
;               clr     p4.7            ; start motor
;               call    dlytime
;waita:         jnb     p1.3, waita
;here1:         call    dispense


;here:          jnb     disp, here
;               clr     disp
;               call    queue             ; load queue

;               mov     a, queue_reg
;               rrc     a
;               mov     queue_reg, a
;               dec     queuecnt
;               mov     a, queuecnt
                ;clr    iex6
```

```
                ;setb   ex6
;               jnz     waita
;               setb    p4.7
;               ;clr    ex6

;           call    queue           ; load queue

;               mov     a, queue_reg
;               rrc     a
;               mov     queue_reg, a
;               clr     qflag

;           mov     loopcnt, #100
;tloop:         call    dlytime
;               dec     loopcnt
;               mov     a, loopcnt
;               jnz     tloop
;               call    qloaded
;           mov     loopcnt, #100
;tloop1:                call    dlytime
;               dec     loopcnt
;               mov     a, loopcnt
;               jnz     tloop1
feedback1:     call    feedback

main1:         call    keypad

                cjne    a, #61h, check_b    ; checks for a key
                                            ; returns to main menu
        ret
                jmp     main1

check_b:       cjne    a, #62h, check_d    ; checks for b key
                                ; start motor
                clr     p4.7
                call    dlytime
waitb:         jnb     p1.3, waitb
                call    dispense
                jmp     main1

check_d:       cjne    a, #64h, check_e    ; checks for d key
                                ; dump board

                call    dumprout
                jmp     main1
```

```
check_e:        cjne    a, #65h, check_c     ; checks for e key
                                             ; queue black
                clr     disp
                setb    qflag
tryagain:       mov     a, queue_reg         ; check value in queue
                clr     c
                rrc     a
                jc      isblack              ; determine color
                clr     p4.7
                call    dlytime
waite:          jnb     p1.3, waite
                call    dispense             ; wait for checker to dispense

;               mov     loopcnt, #5
;loopb:        call    dlytime
;               dec     loopcnt
;               mov     a, loopcnt
;               jnz     loopb


            clr   disp
                call    queue                ; sample data
                mov     a, queue_reg         ; store on queue
                rrc     a
                mov     queue_reg, a
                clr     p1.7                 ; for test purpose
                jmp     tryagain

isblack:        setb    p4.7                 ; stop motor

                clr     qflag
                ;call   queue
                ;mov    a, queue_reg
                ;rrc    a
                ;mov    queue_reg, a

;             mov       queuecnt, #0
;               clr     c
;               mov     a, queue_reg
;nextcnt:       rrc     a
;               jnc     notblack
;               mov     tmpq, a
;               call    load
;               mov     a, tmpa
;               orl     a, queue_reg
```

```
;               mov     queue_reg, a

                jmp     main1           ; exit

;notblack:      inc     queuecnt
;               jmp     nextcnt

check_c:        cjne    a, #63h, check_f    ; checks for c
                                    ; run calibration
                lcall   CALIB
                jmp     main1

check_f:        cjne    a, #66h, check_0    ; checks for f key
                                ; queue red checker
                clr     disp
                setb    qflag
tryagain1:      mov     a, queue_reg        ; look at data in queue
                clr     c
                rrc     a
                jnc     isred           ; is next chaecker red
                clr     p4.7
                call    dlytime
waitf:          jnb     p1.3, waitf
                call    dispense            ; wait for checker to dipense


;               mov     loopcnt, #5
;loopr:         call    dlytime
;               dec     loopcnt
;               mov     a, loopcnt
;               jnz     loopr

                clr     disp
                call    queue           ; sample data
                mov     a, queue_reg        ; store in queue
                rrc     a
                mov     queue_reg, a
                clr     p1.7
                jmp     tryagain1

isred:          setb    p4.7            ; stop motor

                clr     qflag
;               call
;               call    queue
;               mov     a, queue_reg
```

```
;               rrc     a
;               mov     queue_reg, a

;           mov         queuecnt, #0
;               clr     c
;               mov     a, queue_reg
;nextcnt1:      rrc     a
;               jc      notred
;               mov     tmpq, a
;               call    load
;               mov     a, tmpa
;               orl     a, queue_reg
;               mov     queue_reg, a
;               jmp     main1
;notred:                inc     queuecnt
;               jmp     nextcnt1

                jmp     main1               ; exit

check_0:        cjne    a, #30h, check_8    ; checks for 0 key
                                            ; not used
                mov     queuecnt, #8
                call    load
                jmp     feedback1

check_8:        cjne    a, #38h, check_9    ; checks for 8 key
                                            ; not used

                ;call   scan
                jmp     main1

check_9:        cjne    a, #39h, check_1    ; checks for 9 key
                                            ; not used
                mov     queue_reg, #0
                mov     queuecnt, #8
                setb    qflag
keepon:                 clr     p4.7
                call    dlytime
wait9:          jnb     p1.3, wait9
                call    dispense            ; wait for checker to dispense
                clr     disp
                call    queue               ; sample data
                mov     a, queue_reg        ; store on queue
                mov     c, next_sample
                rrc     a
                mov     queue_reg, a
```

```
                clr     p1.7
                ;setb   p4.7
                dec     queuecnt
                mov     a, queuecnt
                jnz     keepon
                setb    p4.7
                clr     qflag
                jmp     main1

check_1:        cjne    a, #31h, check_2    ; checks for 1 key

                setb    col1
                clr     p4.7
                call    dlytime
wait1:          jnb     p1.3, wait1
                call    dispense
                call    queue
                mov     a, queue_reg
                mov     c, next_sample
                rrc     a
                mov     queue_reg, a
            clr col1
                jmp     main1


check_2:      cjne      a, #32h, check_3    ; checks for 2 key

                setb    col2
                clr     p4.7
                call    dlytime
wait2:          jnb     p1.3, wait2
                call    dispense
                call    queue
                mov     a, queue_reg
                mov     c, next_sample
                rrc     a
                mov     queue_reg, a
            clr col2
                jmp     main1


check_3:        cjne    a, #33h, check_4    ; checks for 3 key

                setb    col3
                clr     p4.7
                call    dlytime
```

```
wait3:      jnb     p1.3, wait3
            call    dispense
            call    queue
            mov     a, queue_reg
            mov     c, next_sample
            rrc     a
            mov     queue_reg, a
        clr col3
            jmp     main1

check_4:    cjne    a, #34h, check_5    ; checks for 4 key

            setb    col4
            clr     p4.7
            call    dlytime
wait4:      jnb     p1.3, wait4
            call    dispense
            call    queue
            mov     a, queue_reg
            mov     c, next_sample
            rrc     a
            mov     queue_reg, a
        clr col4
            jmp     main1

check_5:    cjne    a, #35h, check_6    ; checks for 5 key

            setb    col5
            clr     p4.7
            call    dlytime
wait5:      jnb     p1.3, wait5
            call    dispense
            call    queue
            mov     a, queue_reg
            mov     c, next_sample
            rrc     a
            mov     queue_reg, a
        clr col5
            jmp     main1

check_6:    cjne    a, #36h, check_7    ; checks for 6 key

            setb    col6
            clr     p4.7
            call    dlytime
wait6:      jnb     p1.3, wait6
```

```
                call    dispense
                call    queue
                mov     a, queue_reg
                mov     c, next_sample
                rrc     a
                mov     queue_reg, a
            clr col6

                jmp     main1

check_7:                        ; must be 7 if here
                setb    col7
                clr     p4.7
                call    dlytime
wait7:          jnb     p1.3, wait7
                call    dispense
                call    queue
                mov     a, queue_reg
                mov     c, next_sample
                rrc     a
                mov     queue_reg, a
            clr col7
                jmp     main1



end

;*********************************************************************
;Justin Middleton and Wayne Bogart
;Bradley University
;Electrical Engineering Dept.
;
;search.A51
;Created: 4-12-07
;Last Updated: 4-12-07
;
;Input:
;Output:
;Corrupted Registers:
;
;*********************************************************************
*


$nomod51
```

```
$Include(reg515.inc)
$Include(myvars.inc)



name   search_mod
public  search
extrn
        code(keypad,dlytime,queue,qred,qblack,p1select,strtgmem,player1d,player2d,gm
eover,dumprout,dispense,storemem)


search_seg      segment         code
rseg    search_seg

;************************************************
;
;       Search Module
;
;************************************************


;************************************************
;
;       Searches for 4 in a row
;
;************************************************


search:         mov     samecolorcnt, #1
                mov     dph, startmemh
                mov     dpl, startmeml

search1:        clr     c               ; searches down
                mov     a, dpl
                subb    a, #9
                mov     dpl, a
                mov     a, dph
                subb    a, #0
                mov     dph, a
                movx    a, @dptr
                jnb     colorbit, comparered
                clr     c
                subb    a, #2
                jnz     leftright
                inc     samecolorcnt
```

```
                mov     a, samecolorcnt
                cjne    a, #4, search1
                setb    bwinner
                jmp     comparedone


comparered:     clr     c
                subb    a, #1
                jnz     leftright
                inc     samecolorcnt
                mov     a, samecolorcnt
                cjne    a, #4, search1
                setb    rwinner
                jmp     comparedone

leftright:      mov     samecolorcnt, #1        ; searches left and right
                mov     dph, startmemh
                mov     dpl, startmeml

search2:        inc     dptr
                movx    a, @dptr
                jnb     colorbit, comparered1
                clr     c
                subb    a, #2
                jnz     compareleft
                inc     samecolorcnt
                mov     a, samecolorcnt
                cjne    a, #4, search2
                setb    bwinner
                jmp     comparedone

comparered1:    clr     c
                subb    a, #1
                jnz     compareleft
                inc     samecolorcnt
                mov     a, samecolorcnt
                cjne    a, #4, search2
                setb    rwinner
                jmp     comparedone

compareleft:    ;mov    tempcnt, samecolorcnt    ; searches left
                mov     dph, startmemh
                mov     dpl, startmeml

search3:        clr     c
                mov     a, dpl
```

```
                subb    a, #1
                mov     dpl, a
            mov a, dph
                subb    a, #0
                mov     dph, a
            movx        a, @dptr
                jnb     colorbit, compareredl
                clr     c
                subb    a, #2
                jnz     negdiag
                inc     samecolorcnt
                mov     a, samecolorcnt
                cjne    a, #4, search3
                setb    bwinner
                jmp     comparedone

compareredl:    clr     c
                subb    a, #1
                jnz     negdiag
                inc     samecolorcnt
                mov     a, samecolorcnt
                cjne    a, #4, search3
                setb    rwinner
                jmp     comparedone

negdiag:        mov     samecolorcnt, #1      ; searches negative diagonal
                mov     dph, startmemh
                mov     dpl, startmeml

search4:        clr     c
                mov     a, dpl
                subb    a, #8
                mov     dpl, a
            mov a, dph
                subb    a, #0
                mov     dph, a
                movx    a, @dptr
                jnb     colorbit, compared2
                clr     c
                subb    a, #2
                jnz     compareul
                inc     samecolorcnt
                mov     a, samecolorcnt
                cjne    a, #4, search4
                setb    bwinner
                jmp     comparedone
```

```
comparered2:   clr     c
               subb    a, #1
               jnz     compareul
               inc     samecolorcnt
               mov     a, samecolorcnt
               cjne    a, #4, search4
               setb    rwinner
               jmp     comparedone

compareul:     ;mov    tempcnt, samecolorcnt    ; searches left
               mov     dph, startmemh
               mov     dpl, startmeml

search5:       clr     c
               mov     a, dpl
               addc    a, #8
               mov     dpl, a
          mov a, dph
               addc    a, #0
               mov     dph, a
          movx         a, @dptr
               jnb     colorbit, compareredul
               clr     c
               subb    a, #2
               jnz     posdiag
               inc     samecolorcnt
               mov     a, samecolorcnt
               cjne    a, #4, search5
               setb    bwinner
               jmp     comparedone

compareredul:  clr     c
               subb    a, #1
               jnz     posdiag
               inc     samecolorcnt
               mov     a, samecolorcnt
               cjne    a, #4, search5
               setb    rwinner
               jmp     comparedone

posdiag:       mov     samecolorcnt, #1      ; searches negative diagonal
               mov     dph, startmemh
               mov     dpl, startmeml

search6:       clr     c
```

```
                mov     a, dpl
                subb    a, #10
                mov     dpl, a
            mov a, dph
                subb    a, #0
                mov     dph, a
                movx    a, @dptr
                jnb     colorbit, comparered3
                clr     c
                subb    a, #2
                jnz     compareur
                inc     samecolorcnt
                mov     a, samecolorcnt
                cjne    a, #4, search6
                setb    bwinner
                jmp     comparedone

comparered3:    clr     c
                subb    a, #1
                jnz     compareur
                inc     samecolorcnt
                mov     a, samecolorcnt
                cjne    a, #4, search6
                setb    rwinner
                jmp     comparedone

compareur:      ;mov    tempcnt, samecolorcnt    ; searches left
                mov     dph, startmemh
                mov     dpl, startmeml

search7:        clr     c
                mov     a, dpl
                addc    a, #10
                mov     dpl, a
            mov a, dph
                addc    a, #0
                mov     dph, a
            movx        a, @dptr
                jnb     colorbit, compareredur
                clr     c
                subb    a, #2
                jnz     comparedone
                inc     samecolorcnt
                mov     a, samecolorcnt
                cjne    a, #4, search7
                setb    bwinner
```

```
                jmp     comparedone

compareredur:   clr     c
                subb    a, #1
                jnz     comparedone
                inc     samecolorcnt
                mov     a, samecolorcnt
                cjne    a, #4, search7
                setb    rwinner
                jmp     comparedone

comparedone: ret


end

;*************************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;load.A51
;Created: 11-29-04
;Last Updated: 12-7-04
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)

name   load_mod
public  load
extrn
        code(keypad,dlytime,queue,calib,display1,menudisp,lcdout,qloading,qloaded,feed
back,dispense)


load_seg        segment         code
rseg    load_seg
```

```
;***********************************************
;
;       Load Module
;
;***********************************************


;***********************************************
;
;       Loads Queue
;
;
;***********************************************

load:           call    qloading

                mov     queue_reg, #0
                mov     queuecnt, #8
                setb    qflag
keepon:                 clr     p4.7
                call    dlytime
waitl:          jnb     p1.3, waitl
                call    dispense        ; wait for checker to dispense
                clr     disp
                call    queue           ; sample data
                mov     a, queue_reg    ; store on queue
                mov     c, next_sample
                rrc     a
                mov     queue_reg, a
                ;clr    p1.7
                ;setb   p4.7
                dec     queuecnt
                mov     a, queuecnt
                jnz     keepon
                setb    p4.7
                clr     qflag

                ret

;               clr     disp            ; loads first queue value
;           setb qflag
;               mov     queue_reg, #0
;               ;mov    queuecnt, #8
;               call    queue
;               mov     a, queue_reg
```

```
;               mov     c, next_sample
;               rrc     a
;               mov     queue_reg, a
;               dec     queuecnt
;               ;mov    a, queuecnt
;               ;cjne   a, #255, motoron
;               ;jmp    getout;
;motoron:       clr     p4.7            ; start motor
;               call    dlytime
;waita:         jnb     p1.3, waita
;here1:         call    dispense1        ; wait for checker to dispense
;

;here:          jnb     disp, here
;               clr     disp
;               call    queue           ; sample data

;               mov     a, queue_reg      ; store data
;               mov     c, next_sample
;               rrc     a
;               mov     queue_reg, a
;               dec     queuecnt
;               mov     a, queuecnt       ; do loop 7 times
;               clr     p1.7
;               ;clr    iex6
;               ;setb   ex6
;               jnz     waita
;               setb    p4.7
;               ;clr    ex6
;
;           call        queue           ; sample data
;
;               mov     a, queue_reg      ; store last value
;               mov     c, next_sample
;               rrc     a
;               mov     queue_reg, a
;               clr     qflag

;pgetout:           ret



end

;*******************************************************************************
```

```
;Justin Middleton and Wayne Bogart
;Bradley University
;Electrical Engineering Dept.
;
;calib.A51
;Created: 1-30-07
;Last Updated: 1-30-07
;
;Input:Burst, odd/even, vertretrace, composite sync
;Output:
;Corrupted Registers:R0, R1
;
;************************************************************************
*
$NOMOD51
$Include(reg515.inc)
$Include(myvars.inc)

        NAME        CALIB
        PUBLIC              CALIB
        ;EXTRN CODE     ()
        calib_seg   SEGMENT CODE
        RSEG        calib_seg

        USING 0

calib:
        SETB  EX4
        SETB  EX5
new:
        MOV   linecnt,#40h          ;R1 line index

next:
        MOV   R1,linecnt
        MOV   descnt,@R1
        jnb   p1.6, nope

begin:
        MOV   A,testcnt
        CJNE  A,descnt,begin

        SETB  P1.0                  ;turn on white out first column
        clr   p1.0
        nop
        nop
        nop
```

```
        setb    p1.0            ;second column
        clr     p1.0
        nop
        nop
        nop
        setb    p1.0            ;third column
        clr     p1.0
        nop
        nop
        nop
        nop
        setb    p1.0            ;fourth column
        clr     p1.0
        nop
        nop
        nop
        setb    p1.0            ;fifth column
        clr     p1.0
        nop
        nop
        nop
        setb    p1.0            ;sixth column
        clr     p1.0
        nop
        nop
        nop
        setb    p1.0            ;seventh
        clr     p1.0

        INC     linecnt

nope:
        jb      tcon.3,leave    ;check if button pressed
        MOV     A,linecnt
        CJNE    A,#46h,next
        ljmp    new

leave:
        CLR     EX4
        CLR     EX5
        CLR     P1.0
        RET


END
```

```
;*********************************************************************
;Justin Middleton and Wayne Bogart
;Bradley University
;Electrical Engineering Dept.
;
;int4.A51
;Created: 9-27-06
;Last Updated: 2-6-07
;
;Input:
;Output: newcnt variable change, control pin output for white level
;Corrupted Registers:  r0
;
;*********************************************************************
*
$NOMOD51
$Include(reg515.inc)
$Include(myvars.inc)


        NAME        INT5
        PUBLIC              INT5
        int5_seg    SEGMENT CODE
        RSEG        int5_seg

int5:
        MOV   testcnt,#00h          ;resets the line count to signal beginning of frame
        NOP
        RET
END

;*********************************************************************
;Justin Middleton and Wayne Bogart
;Bradley University
;Electrical Engineering Dept.
;
;int4.A51
;Created: 9-27-06
;Last Updated: 2-6-07
;
;Input:
;Output: newcnt variable change, control pin output for white level
;Corrupted Registers:  r0
;
;*********************************************************************
*
```

```
$NOMOD51
$Include(reg515.inc)
$Include(myvars.inc)


        NAME        INT4
        PUBLIC              INT4
        int4_seg    SEGMENT CODE
        RSEG        int4_seg

int4:
        INC   testcnt
;       MOV   A,testcnt
;       CJNE  A,descnt,nope         ;check if count equals desired count
;       SETB  newcnt
;       SETB  P1.0                  ;turn on white out
;       clr   p1.0
;       nop
;       nop
;       setb  p1.0              ;second column
;       clr   p1.0
;       nop
;       nop
;       nop
;       setb  p1.0              ;third column
;       clr   p1.0
;       nop
;       nop
;       nop
;       nop
;       setb  p1.0              ;fourth column
;       clr   p1.0
;       nop
;       nop
;       nop
;       setb  p1.0              ;fifth column
;       clr   p1.0
;       nop
;       nop
;       nop
;       setb  p1.0              ;sixth column
;       clr   p1.0
;       nop
;       nop
;       nop
;       setb  p1.0              ;seventh
```

```
;       clr     p1.0
;       ljmp    leave
;nope:
;       CLR     P1.0                    ;set white out =0
;leave:
        RET
END

;*************************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;dlytime.A51
;Created: 12-2-04
;Last Updated: 12-5-04
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*

$nomod51
$Include(reg515.inc)
$Include(myvars.inc)

name    dlytime_mod
public  dlytime
extrn   code(main)


dlytime_seg     segment         code
rseg    dlytime_seg

;*************************************
;
;       Delaytime Module
;
;*************************************

;*********************************************
;
;       creates 50ms delay
;
```

```
;************************************************


dlytime:        mov     dcount1, #0ffh
dlytime1:       mov     dcount2, #10
dlytime2:       dec     dcount2
            nop
            nop
            nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                mov     a, dcount2
                jnz     dlytime2
                dec     dcount1
                mov     a, dcount1
                nop
                nop
                nop
                nop
                jnz     dlytime1
                ret

end

;*****************************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;storemem.A51
;Created: 4-11-07
;Last Updated: 4-11-07
;
;Input:
;Output:
;Corrupted Registers:
;
```

```
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name   storemem_mod
public  storemem
extrn
        code(keypad,dlytime,queue,qred,qblack,p1select,strtgmem,player1d,player2d,gm
eover,dumprout,dispense)


storemem_seg segment        code
rseg    storemem_seg

;************************************************
;
;       Store Memory Module
;
;************************************************


;************************************************
;
;       Stores plays in memeory
;
;************************************************

storemem:
                jnb     col1, played2

                mov     dptr, #800ah       ; bottom memory location for column 1
                mov     r5, col1cnt

memcheck1:      cjne    r5, #0, incmem1
                mov     startmeml, dpl
                mov     startmemh, dph
                mov     r1,#50h

                jnb     player1, player2st     ; stores checker color in memory
                jnb     p1color, store1red     ; using memory under SFR's
```

```
                mov     a, #2
                movx    @dptr, a
                setb    colorbit            ; tells search module what color to look for
                inc     col1cnt
                clr     col1
                ret                     ; 1=black  0=red


store1red:      mov     a, #1
                movx    @dptr, a
                clr     colorbit
                inc     col1cnt
                clr     col1
                ret


player2st:      jnb     p1color, store2black
                mov     a, #1
                movx    @dptr, a
                clr     colorbit

                inc     col1cnt
                clr     col1
                ret

store2black:    mov     a, #2
                movx    @dptr, a
                setb    colorbit
                inc     col1cnt
                clr     col1
                ret

incmem1:        mov     a, #9               ; gets to desired memory location
                clr     c
                addc    a, dpl
                mov     dpl,a
                mov     a, dph
                addc    a, #0
                mov     dph, a
                mov     r1, a

                dec     r5
                jmp     memcheck1
```

```
played2:        jnb     col2, played3

                mov     dptr, #800bh        ; bottom memory location for column 2
                mov     r5, col2cnt

memcheck2:      cjne    r5, #0, incmem2
                mov     startmeml, dpl
                mov     startmemh, dph

                jnb     player1, player2st2     ; stores checker color in memory
                jnb     p1color, store1red2     ; using memory under SFR's
                mov     a, #2
                movx    @dptr, a
                setb    colorbit            ; tells search module what color to look for
                inc     col2cnt
                clr     col2
                ret                     ; 1=black  0=red


store1red2:     mov     a, #1
                movx    @dptr, a
                clr     colorbit
                inc     col2cnt
                clr     col2
                ret


player2st2:     jnb     p1color, store2black2
                mov     a, #1
                movx    @dptr, a
                clr     colorbit

                inc     col2cnt
                clr     col2
                ret

store2black2:   mov     a, #2
                movx    @dptr, a
                setb    colorbit
                inc     col2cnt
                clr     col2
                ret

incmem2:        mov     a, #9               ; gets to desired memory location
                clr     c
                addc    a, dpl
```

```
                mov     dpl,a
                mov     a, dph
                addc    a, #0
                mov     dph, a
                mov     r1, a
                dec     r5
                jmp     memcheck2

played3:        jnb     col3, played4

                mov     dptr, #800ch        ; bottom memory location for column 1
                mov     r5, col3cnt
memcheck3:      cjne    r5, #0, incmem3
                ;mov    startmem, r1
                mov     startmeml, dpl
                mov     startmemh, dph

                jnb     player1, player2st3     ; stores checker color in memory
                jnb     p1color, store1red3     ; using memory under SFR's
                mov     a, #2
                movx    @dptr, a
                setb    colorbit            ; tells search module what color to look for
                inc     col3cnt
                clr     col3
                ret                     ; 1=black  0=red


store1red3:     mov     a, #1
                movx    @dptr, a
                clr     colorbit
                inc     col3cnt
                clr     col3
                ret


player2st3:     jnb     p1color, store2black3
                mov     a, #1
                movx    @dptr, a
                clr     colorbit

                inc     col3cnt
                clr     col3
                ret

store2black3:   mov     a, #2
                movx    @dptr, a
```

```
                setb    colorbit
                inc     col3cnt
                clr     col3
                ret

incmem3:        mov     a, #9           ; gets to desired memory location
                clr     c
                addc    a, dpl
                mov     dpl,a
                mov     a, dph
                addc    a, #0
                mov     dph, a
                mov     r1, a
                dec     r5
                jmp     memcheck3

played4:        jnb     col4, played5

                mov     dptr, #800dh    ; bottom memory location for column 1
                mov     r5, col4cnt
memcheck4:      cjne    r5, #0, incmem4
                ;mov    startmem, r1
                mov     startmeml, dpl
                mov     startmemh, dph

                jnb     player1, player2st4     ; stores checker color in memory
                jnb     p1color, store1red4     ; using memory under SFR's
                mov     a, #2
                movx    @dptr, a
                setb    colorbit        ; tells search module what color to look for
                inc     col4cnt
                clr     col4
                ret                     ; 1=black  0=red


store1red4:     mov     a, #1
                movx    @dptr, a
                clr     colorbit
                inc     col4cnt
                clr     col4
                ret


player2st4:     jnb     p1color, store2black4
                mov     a, #1
                movx    @dptr, a
```

```
                clr     colorbit

                inc     col4cnt
                clr     col4
                ret

store2black4:   mov     a, #2
                movx    @dptr, a
                setb    colorbit
                inc     col4cnt
                clr     col4
                ret

incmem4:        mov     a, #9           ; gets to desired memory location
                clr     c
                addc    a, dpl
                mov     dpl,a
                mov     a, dph
                addc    a, #0
                mov     dph, a
                mov     r1, a
                dec     r5
                jmp     memcheck4

played5:        jnb     col5, played6

                mov     dptr, #800eh    ; bottom memory location for column 1
                mov     r5, col5cnt
memcheck5:      cjne    r5, #0, incmem5
                ;mov    startmem, r1
                mov     startmeml, dpl
                mov     startmemh, dph

                jnb     player1, player2st5     ; stores checker color in memory
                jnb     p1color, store1red5     ; using memory under SFR's
                mov     a, #2
                movx    @dptr, a
                setb    colorbit        ; tells search module what color to look for
                inc     col5cnt
                clr     col5
                ret                     ; 1=black  0=red


store1red5:     mov     a, #1
                movx    @dptr, a
                clr     colorbit
```

```
                inc     col5cnt
                clr     col5
                ret


player2st5:     jnb     p1color, store2black5
                mov     a, #1
                movx    @dptr, a
                clr     colorbit

                inc     col5cnt
                clr     col5
                ret

store2black5:   mov     a, #2
                movx    @dptr, a
                setb    colorbit
                inc     col5cnt
                clr     col5
                ret

incmem5:        mov     a, #9           ; gets to desired memory location
                clr     c
                addc    a, dpl
                mov     dpl,a
                mov     a, dph
                addc    a, #0
                mov     dph, a
                mov     r1, a
                dec     r5
                jmp     memcheck5

played6:        jnb     col6, played7

                mov     dptr, #800fh    ; bottom memory location for column 1
                mov     r5, col6cnt
memcheck6:      cjne    r5, #0, incmem6
                ;mov    startmem, r1
                mov     startmeml, dpl
                mov     startmemh, dph

                jnb     player1, player2st6     ; stores checker color in memory
                jnb     p1color, store1red6     ; using memory under SFR's
                mov     a, #2
                movx    @dptr, a
                setb    colorbit        ; tells search module what color to look for
```

```
                inc     col6cnt
                clr     col6
                ret                     ; 1=black  0=red


store1red6:     mov     a, #1
                movx    @dptr, a
                clr     colorbit
                inc     col6cnt
                clr     col6
                ret


player2st6:     jnb     p1color, store2black6
                mov     a, #1
                movx    @dptr, a
                clr     colorbit

                inc     col6cnt
                clr     col6
                ret

store2black6:   mov     a, #2
                movx    @dptr, a
                setb    colorbit
                inc     col6cnt
                clr     col6
                ret

incmem6:        mov     a, #9           ; gets to desired memory location
                clr     c
                addc    a, dpl
                mov     dpl,a
                mov     a, dph
                addc    a, #0
                mov     dph, a
                mov     r1, a
                dec     r5
                jmp     memcheck6

played7:        mov     dptr, #8010h    ; bottom memory location for column 1
                mov     r5, col7cnt
memcheck7:      cjne    r5, #0, incmem7
                ;mov    startmem, r1
                mov     startmeml, dpl
                mov     startmemh, dph
```

```
                jnb     player1, player2st7     ; stores checker color in memory
                jnb     p1color, store1red7     ; using memory under SFR's
                mov     a, #2
                movx    @dptr, a
                setb    colorbit            ; tells search module what color to look for
                inc     col7cnt
                clr     col7
                ret                     ; 1=black  0=red


store1red7:     mov     a, #1
                movx    @dptr, a
                clr     colorbit
                inc     col7cnt
                clr     col7
                ret


player2st7:     jnb     p1color, store2black7
                mov     a, #1
                movx    @dptr, a
                clr     colorbit

                inc     col7cnt
                clr     col7
                ret

store2black7:   mov     a, #2
                movx    @dptr, a
                setb    colorbit
                inc     col7cnt
                clr     col7
                ret

incmem7:        mov     a, #9               ; gets to desired memory location
                clr     c
                addc    a, dpl
                mov     dpl,a
                mov     a, dph
                addc    a, #0
                mov     dph, a
                mov     r1, a
                dec     r5
                jmp     memcheck7
```

```
end

;
;fullbrd.A51
;Created: 4-25-07
;Last Updated: 4-25-07
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name   fullbrd_mod
public  fullbrd
extrn
        code(keypad,dlytime,queue,qred,qblack,p1select,strtgmem,player1d,player2d,gm
eover,dumprout,dispense,storemem,search,blackwin,redwin,compturn,humand)


fullbrd_seg     segment         code
rseg    fullbrd_seg

;************************************************
;
;       Fullboard Module
;
;************************************************


;************************************************
;
;       Checks for full game board
;
;************************************************
```

```
fullbrd:mov    a, col1cnt
               cjne    a, #6, notful
               mov     a, col2cnt
               cjne    a, #6, notful
               mov     a, col3cnt
               cjne    a, #6, notful
               mov     a, col4cnt
               cjne    a, #6, notful
               mov     a, col5cnt
               cjne    a, #6, notful
               mov     a, col6cnt
               cjne    a, #6, notful
               mov     a, col7cnt
               cjne    a, #6, notful
               setb    brdfull
;              jnb     col1ful, notful
;              jmp     is2ful


;is2ful:       jnb     col2ful, notful
;              jmp     is3ful

;is3ful:       jnb     col3ful, notful
;              jmp     is4ful

;is4ful:       jnb     col4ful, notful
;              jmp     is5ful

;is5ful:       jnb     col5ful, notful
;              jmp     is6ful

;is6ful:       jnb     col6ful, notful
;              jmp     is7ful

;is7ful:       jnb     col7ful, notful
;              setb    brdfull

notful:        ret

end

;*******************************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
```

```
;
;game.A51
;Created: 4-2-07
;Last Updated: 4-2-07
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name   game_mod
public  game
extrn   code(main,keypad,dlytime,queue,load,gamemenu,manual,automat)


game_seg       segment         code
rseg    game_seg

;************************************************
;
;       Game Module
;
;************************************************


game:           call    gamemenu

game1:          call    keypad

                cjne    a, #61h, check_g1     ; checks for a key

                ret                     ; returns to main menu

check_g1:       cjne    a, #31h, check_g2     ; checks for 1 key
```

```
                call    manual          ; goes to manual game mode

                jmp     game

check_g2:       cjne    a, #32h, game1  ; checks for 2 key

                call    automat         ; goes to automatic game mode
                jmp     game            ; software not written at this time

end



game_loop:      jmp     game1

;*********************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;rmain.A51
;Created: 11-29-04
;Last Updated: 12-7-04
;
;Input:
;Output:
;Corrupted Registers:
;
;*********************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name    queue_mod
public  queue
extrn   code(main,keypad,dlytime)


queue_seg       segment         code
rseg    queue_seg
```

```
;***********************************************
;
;         Queue Module
;
;***********************************************



queue:
                ;jnb     checker, queue
                clr     adcon.0         ; set up A/D for channel 0 and single conversion
                clr     adcon.1
                clr     adcon.2
                clr     adcon.3
                ;setb    p1.7            ; for testing
                mov     dapr, #0        ; get data
conversion:     jb      bsy, conversion
                mov     color, addat
                clr     c
                mov     a, #120         ; threshold level between black and red

                subb    a, color        ; generate carry bit

                mov     next_sample, c



out:            ret
end

;*************************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;dumprout.A51
;Created: 4-2-07
;Last Updated: 4-2-07
;
;Input:
;Output:
;Corrupted Registers:
;
```

```
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name   dumprout_mod
public  dumprout
extrn   code(dlytime)


dumprout_seg segment        code
rseg    dumprout_seg

;************************************************
;
;       Dump Module
;
;************************************************


;************************************************
;
;       Dumps game board
;
;************************************************



dumprout:       clr     p1.4            ; open dump
                setb    p1.5
                mov     tmpa, #01
dumpm:          call    dlytime
                dec     tmpa
                mov     a, tmpa
                jnz     dumpm
                clr     p1.5

                mov     tmpa, #50
waitm:          call    dlytime
                dec     tmpa
                mov     a, tmpa
```

```
                jnz     waitm
                setb    p1.4            ; close dump
                setb    p1.5

                mov     tmpa, #02
closem:         call    dlytime
                dec     tmpa
                mov     a, tmpa
                jnz     closem
                clr     p1.5
                ret

end
;*************************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;dispense.A51
;Created: 4-2-07
;Last Updated: 4-2-07
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)

name    dispense_mod
public  dispense
extrn
        code(keypad,dlytime,queue,calib,display1,menudisp,lcdout,qloading,qloaded,feed
back)


dispense_seg    segment         code
rseg    dispense_seg

;*********************************************
;
```

```
;       Dispense Module
;
;************************************************


;************************************************
;
;       Dispenses checker
;
;
;************************************************


dispense:   jb      qflag, going
test10:     jb      p1.3, test10
            call    switch
            call    dlytime
            ;call   dlytime
            setb    p4.7        ; stop motor

disp1:      jnb     col1, disp2
            clr     p4.0
            mov     tmpa, #15
test1:      call    dlytime
            dec     tmpa
            mov     a, tmpa
            jnz     test1
            setb    p4.0

            ;clr    col1
            ret

going:      jmp     load5


disp2:      jnb     col2, disp3
            clr     p4.1
            mov     tmpa, #15
test2:      call    dlytime
            dec     tmpa
            mov     a, tmpa
            jnz     test2
            setb    p4.1

            ;clr    col2
            ret
```

```
disp3:      jnb     col3, disp4
            clr     p4.2
            mov     tmpa, #18
test3:      call    dlytime
            dec     tmpa
            mov     a, tmpa
            jnz     test3
            setb    p4.2

            ;clr    col3
            ret

disp4:      jnb     col4, disp5
            clr     p4.3
            mov     tmpa, #20
test4:      call    dlytime
            dec     tmpa
            mov     a, tmpa
            jnz     test4
            setb    p4.3

            ;clr    col4
            ret

disp5:      jnb     col5, disp6
            clr     p4.4
            mov     tmpa, #20
test5:      call    dlytime
            dec     tmpa
            mov     a, tmpa
            jnz     test5
            setb    p4.4

            ;clr    col5
            ret

disp6:      jnb     col6, disp7
            clr     p4.5
            mov     tmpa, #25
test6:      call    dlytime
            dec     tmpa
            mov     a, tmpa
            jnz     test6
            setb    p4.5
```

```
                ;clr    col6
                ret

disp7:          jnb     col7, exit
                call    dlytime
                call    dlytime
                call    dlytime
                clr     p4.6
                mov     tmpa, #25
test7:          call    dlytime
                dec     tmpa
                mov     a, tmpa
                jnz     test7
                setb    p4.6

                ;clr    col7
                ret

exit:

                ret

load5:          jb      p1.3, load5         ; wait for optic to go low
                ;call   dlytime
                call    switch
                setb    disp

                ret

;*************************************************************
;
;  Switch check subroutine
;
;*************************************************************


switch:         mov     switchcnt, #255     ; make sure optic is low for 255 counts
switch1:        jnb     p1.3, decrease
                jmp     switch

decrease:       dec     switchcnt
                mov     a, switchcnt
                jnz     switch1
                ret

end
```

```
;*************************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;notwin.A51
;Created: 4-25-07
;Last Updated: 4-25-07
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name   notwin_mod
public  notwin
extrn   code(automat)


notwin_seg     segment         code
rseg     notwin_seg

;************************************************
;
;         No Winner Display Module
;
;************************************************

;************************************************
;
; Displays  No Winner  Board Full
;
;************************************************
```

```
dis_addrnot:
;line1 addresses
db      80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8ah, 8bh, 8ch, 8dh, 8eh, 8fh,
90h, 91h, 92h, 93h
;line2 addresses
db      0c0h, 0c1h, 0c2h, 0c3h, 0c4h, 0c5h, 0c6h, 0c7h, 0c8h, 0c9h, 0cah, 0cbh, 0cch,
0cdh, 0ceh, 0cfh, 0d0h, 0d1h, 0d2h, 0d3h


dis_charnot:
;line1 characters
db      ' ', ' ', ' ', ' ', ' ', 'N', 'o', ' ', 'W', 'i', 'n', 'n', 'e', 'r', ' ', ' ', ' ', ' ', ' ', ' '
;line2 characters
db      ' ', ' ', ' ', ' ', ' ', 'B', 'o', 'a', 'r', 'd', ' ', 'F', 'u', 'l', 'l', ' ', ' ', ' ', ' ', ' '




notwin:         mov     r0, #0ffh
dis_loopnot:    inc     r0
                mov     dptr, #dis_addrnot
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_outnot
                mov     dph, #20h
                mov     a, r1
                movx    @dptr, a

                mov     dptr, #dis_charnot
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_outnot
                mov     dph, #21h
                mov     a, r1
                movx    @dptr, a
                cjne    r0, #39d, dis_loopnot

                ret
```

```
delay_outnot:  mov    r3, #07fh
               djnz   r3, $
               ret

end

;*******************************************************************************
;Justin Middleton and Wayne Bogart
;Bradley University
;Electrical Engineering Dept.
;
;feedback.A51
;Created:
;Last Updated: 12-7-04
;
;Input:
;Output:
;Corrupted Registers:
;
;*******************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name   feedback_mod
public  feedback
extrn   code(test)


feedback_seg  segment         code
rseg     feedback_seg

;************************************************
;
;         Feedback Display Module
;
;************************************************

;************************************************
;
```

```
; Displays
;
;***********************************************
```

```
dis_addr4:
;line1 addresses
db      80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8ah, 8bh, 8ch, 8dh, 8eh, 8fh,
90h, 91h, 92h, 93h
;line2 addresses
db      0c0h, 0c1h, 0c2h, 0c3h, 0c4h, 0c5h, 0c6h, 0c7h, 0c8h, 0c9h, 0cah, 0cbh, 0cch,
0cdh, 0ceh, 0cfh, 0d0h, 0d1h, 0d2h, 0d3h
```

```
dis_char4:
;line1 characters
db      'A', '=', 'M', 'a', 'i', 'n', ' ', 'M', 'e', 'n', 'u', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '
;line2 characters
db      'T', 'r', 'y', ' ', 'O', 't', 'h', 'e', 'r', ' ', 'K', 'e', 'y', 's', ' ', ' ', ' ', ' ', ' ', ' ', ' '
```

```
feedback:       mov     r0, #0ffh
dis_loop4:      inc     r0
                mov     dptr, #dis_addr4
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_out4
                mov     dph, #20h
                mov     a, r1
                movx    @dptr, a

                mov     dptr, #dis_char4
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_out4
```

```
                mov     dph, #21h
                mov     a, r1
                movx    @dptr, a
                cjne    r0, #39d, dis_loop4

                ret

delay_out4:     mov     r3, #07fh
                djnz    r3, $
                ret

end

;*************************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;qloaded.A51
;Created: 11-29-04
;Last Updated: 12-7-04
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name    qloaded_mod
public  qloaded
extrn   code(test)


qloaded_seg     segment         code
rseg    qloaded_seg

;*********************************************
;
```

```
;          Queue loaded Display Module
;
;************************************************

;************************************************
;
; Displays Queue loaded
;
;************************************************




dis_addr3:
;line1 addresses
db       80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8ah, 8bh, 8ch, 8dh, 8eh, 8fh,
90h, 91h, 92h, 93h
;line2 addresses
db       0c0h, 0c1h, 0c2h, 0c3h, 0c4h, 0c5h, 0c6h, 0c7h, 0c8h, 0c9h, 0cah, 0cbh, 0cch,
0cdh, 0ceh, 0cfh, 0d0h, 0d1h, 0d2h, 0d3h


dis_char3:
;line1 characters
db       'Q', 'u', 'e', 'u', 'e', ' ', 'L', 'o', 'a', 'd', 'e', 'd', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '
;line2 characters
db       ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '




qloaded:        mov     r0, #0ffh
dis_loop3:      inc     r0
                mov     dptr, #dis_addr3
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_out3
                mov     dph, #20h
                mov     a, r1
                movx    @dptr, a
```

```
                mov     dptr, #dis_char3
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_out3
                mov     dph, #21h
                mov     a, r1
                movx    @dptr, a
                cjne    r0, #39d, dis_loop3

                ret

delay_out3:     mov     r3, #07fh
                djnz    r3, $
                ret

end

;*************************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;gmeover.A51
;Created: 4-2-07
;Last Updated: 4-2-07
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name   gmeover_mod
public  gmeover
```

```
gmeover_seg   segment         code
rseg     gmeover_seg

;************************************************
;
;         Game Over Display Module
;
;************************************************

;************************************************
;
; Displays Game Over
;
;************************************************



dis_addrg:
;line1 addresses
db       80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8ah, 8bh, 8ch, 8dh, 8eh, 8fh,
90h, 91h, 92h, 93h
;line2 addresses
db       0c0h, 0c1h, 0c2h, 0c3h, 0c4h, 0c5h, 0c6h, 0c7h, 0c8h, 0c9h, 0cah, 0cbh, 0cch,
0cdh, 0ceh, 0cfh, 0d0h, 0d1h, 0d2h, 0d3h

;dis_char_seg segment code
;rseg dis_char_seg
dis_charg:
;line1 characters
db       'G', 'a', 'm', 'e', ' ', 'O', 'v', 'e', 'r', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '
;line2 characters
db       'T', 'h', 'a', 'n', 'k', 's', ' ', 'F', 'o', 'r', ' ', 'P', 'l', 'a', 'y', 'i', 'n', 'g', ' ', ' '




gmeover:        mov     r0, #0ffh
dis_loopg:      inc     r0
                mov     dptr, #dis_addrg
                mov     a, r0
                movc    a, @a+dptr
```

```
                mov     r1, a

                call    delay_outg
                mov     dph, #20h
                mov     a, r1
                movx    @dptr, a

                mov     dptr, #dis_charg
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_outg
                mov     dph, #21h
                mov     a, r1
                movx    @dptr, a
                cjne    r0, #39d, dis_loopg

                ret

delay_outg:     mov     r3, #07fh
                djnz    r3, $
                ret

end

;*******************************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;strtgmem.A51
;Created: 4-2-07
;Last Updated: 4-2-07
;
;Input:
;Output:
;Corrupted Registers:
;
;*******************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)
```

```
name   strtgmem_mod
public  strtgmem
extrn   code(main)


strtgmem_seg segment         code
rseg    strtgmem_seg

;***************************************************
;
;         Start Game Menu Display Module
;
;***************************************************

;***************************************************
;
; Displays A= Main Menu 0=start game
;
;***************************************************



dis_addrs:
;line1 addresses
db      80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8ah, 8bh, 8ch, 8dh, 8eh, 8fh,
90h, 91h, 92h, 93h
;line2 addresses
db      0c0h, 0c1h, 0c2h, 0c3h, 0c4h, 0c5h, 0c6h, 0c7h, 0c8h, 0c9h, 0cah, 0cbh, 0cch,
0cdh, 0ceh, 0cfh, 0d0h, 0d1h, 0d2h, 0d3h

;dis_char_seg segment code
;rseg dis_char_seg
dis_chars:
;line1 characters
db      'A', '=', 'G', 'a', 'm', 'e', ' ', 'M', 'e', 'n', 'u', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '
;line2 characters
db      '0', '=', 'S', 't', 'a', 'r', 't', ' ', 'G', 'a', 'm', 'e', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '
```

```
strtgmem:       mov     r0, #0ffh
dis_loops:      inc     r0
                mov     dptr, #dis_addrs
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_outs
                mov     dph, #20h
                mov     a, r1
                movx    @dptr, a

                mov     dptr, #dis_chars
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_outs
                mov     dph, #21h
                mov     a, r1
                movx    @dptr, a
                cjne    r0, #39d, dis_loops

                ret

delay_outs:     mov     r3, #07fh
                djnz    r3, $
                ret

end

;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;powerok.A51
;Created: 4-2-07
;Last Updated: 4-2-07
;
;Input:
;Output:
;Corrupted Registers: r0,r1,r3
;
```

```
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name   powerok_mod
public  powerok



powerok_seg  segment        code
rseg     powerok_seg

;************************************************
;
;         Power OK Display Module
;
;************************************************

;************************************************
;
; Displays Software Loaded OK to apply power
;
;************************************************




dis_addr5:
;line1 addresses
db       80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8ah, 8bh, 8ch, 8dh, 8eh, 8fh,
90h, 91h, 92h, 93h
;line2 addresses
db       0c0h, 0c1h, 0c2h, 0c3h, 0c4h, 0c5h, 0c6h, 0c7h, 0c8h, 0c9h, 0cah, 0cbh, 0cch,
0cdh, 0ceh, 0cfh, 0d0h, 0d1h, 0d2h, 0d3h


dis_char5:
;line1 characters
db       'O', 'K', ' ', 'T', 'o', ' ', 'A', 'p', 'p', 'l', 'y', ' ', 'P', 'o', 'w', 'e', 'r', ' ', ' ', ' '
;line2 characters
```

```
db	'','','','','','','','','','','','','','','','','','','',''




powerok:	mov	r0, #0ffh
dis_loop5:	inc	r0
		mov	dptr, #dis_addr5
		mov	a, r0
		movc	a, @a+dptr
		mov	r1, a

		call	delay_out5
		mov	dph, #20h
		mov	a, r1
		movx	@dptr, a

		mov	dptr, #dis_char5
		mov	a, r0
		movc	a, @a+dptr
		mov	r1, a

		call	delay_out5
		mov	dph, #21h
		mov	a, r1
		movx	@dptr, a
		cjne	r0, #39d, dis_loop5

		ret

delay_out5:	mov	r3, #07fh
		djnz	r3, $
		ret

end

;***********************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;gamemenu.A51
;Created: 11-29-04
```

```
;Last Updated: 12-7-04
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name   gamemenu_mod
public  gamemenu
extrn   code(main)


gamemenu_seg        segment        code
rseg    gamemenu_seg

;************************************************
;
;        Game Menu  Module
;
;************************************************

;************************************************
;
; Displays 1=Manual 2=Automatic A=Main Menu
;
;************************************************



dis_addrg:
;line1 addresses
db      80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8ah, 8bh, 8ch, 8dh, 8eh, 8fh,
90h, 91h, 92h, 93h
;line2 addresses
db      0c0h, 0c1h, 0c2h, 0c3h, 0c4h, 0c5h, 0c6h, 0c7h, 0c8h, 0c9h, 0cah, 0cbh, 0cch,
0cdh, 0ceh, 0cfh, 0d0h, 0d1h, 0d2h, 0d3h
```

```
;dis_char_seg segment code
;rseg dis_char_seg
dis_charg:
;line1 characters
db      '1', '=', 'M', 'a', 'n', 'u', 'a', 'l', ' ', 'A', '=', 'M', 'a', 'i', 'n', ' ', 'M', 'e', 'n', 'u'
;line2 characters
db      '2', '=', 'A', 'u', 't', 'o', 'm', 'a', 't', 'i', 'c', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '




gamemenu:   mov     r0, #0ffh
dis_loopg:  inc     r0
            mov     dptr, #dis_addrg
            mov     a, r0
            movc    a, @a+dptr
            mov     r1, a

            call    delay_outg
            mov     dph, #20h
            mov     a, r1
            movx    @dptr, a

            mov     dptr, #dis_charg
            mov     a, r0
            movc    a, @a+dptr
            mov     r1, a

            call    delay_outg
            mov     dph, #21h
            mov     a, r1
            movx    @dptr, a
            cjne    r0, #36d, dis_loopg

            ret

delay_outg: mov     r3, #07fh
            djnz    r3, $
            ret

end
```

```
;*******************************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;p1select.A51
;Created: 4-2-07
;Last Updated: 4-2-07
;
;Input:
;Output:
;Corrupted Registers:
;
;*******************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name   p1select_mod
public  p1select
extrn   code(manual)


p1select_seg   segment         code
rseg     p1select_seg

;************************************************
;
;         Player 1 Select Display Module
;
;************************************************

;************************************************
;
; Displays Player1 Select color  B=Black C=Red
;
;************************************************
```

```
dis_addrp1:
;line1 addresses
db      80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8ah, 8bh, 8ch, 8dh, 8eh, 8fh,
90h, 91h, 92h, 93h
;line2 addresses
db      0c0h, 0c1h, 0c2h, 0c3h, 0c4h, 0c5h, 0c6h, 0c7h, 0c8h, 0c9h, 0cah, 0cbh, 0cch,
0cdh, 0ceh, 0cfh, 0d0h, 0d1h, 0d2h, 0d3h

;dis_char_seg segment code
;rseg dis_char_seg
dis_charp1:
;line1 characters
db      'P', 'l', 'a', 'y', 'e', 'r', '1', ' ', 'S', 'e', 'l', 'e', 'c', 't', ' ', 'C', 'o', 'l', 'o', 'r'
;line2 characters
db      'B', '=', 'B', 'l', 'a', 'c', 'k', ' ', 'C', '=', 'R', 'e', 'd', ' ', ' ', ' ', ' ', ' ', ' ', ' '



p1select:       mov     r0, #0ffh
dis_loopp1:     inc     r0
                mov     dptr, #dis_addrp1
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_outp1
                mov     dph, #20h
                mov     a, r1
                movx    @dptr, a

                mov     dptr, #dis_charp1
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_outp1
                mov     dph, #21h
                mov     a, r1
                movx    @dptr, a
                cjne    r0, #39d, dis_loopp1

                ret
```

```
delay_outp1:  mov    r3, #07fh
              djnz   r3, $
              ret

end

;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;human.A51
;Created: 4-12-07
;Last Updated: 4-12-07
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name   humand_mod
public  humand



humand_seg   segment        code
rseg     humand_seg

;************************************************
;
;        Human Display Module
;
;************************************************

;************************************************
;
; Displays Play Checker, Enter column played
```

```
;
;***********************************************
```

```
dis_addrhu:
;line1 addresses
db      80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8ah, 8bh, 8ch, 8dh, 8eh, 8fh,
90h, 91h, 92h, 93h
;line2 addresses
db      0c0h, 0c1h, 0c2h, 0c3h, 0c4h, 0c5h, 0c6h, 0c7h, 0c8h, 0c9h, 0cah, 0cbh, 0cch,
0cdh, 0ceh, 0cfh, 0d0h, 0d1h, 0d2h, 0d3h

;dis_char_seg segment code
;rseg dis_char_seg
dis_charhu:
;line1 characters
db      ' ', ' ', ' ', 'P', 'l', 'a', 'y', ' ', 'C', 'h', 'e', 'c', 'k', 'e', 'r', ' ', ' ', ' ', ' ', ' '
;line2 characters
db      'E', 'n', 't', 'e', 'r', ' ', 'C', 'o', 'l', 'u', 'm', 'n', ' ', 'P', 'l', 'a', 'y', 'e', 'd', ' '
```

```
humand:     mov   r0, #0ffh
dis_loophu: inc   r0
            mov   dptr, #dis_addrhu
            mov   a, r0
            movc  a, @a+dptr
            mov   r1, a

            call  delay_outhu
            mov   dph, #20h
            mov   a, r1
            movx  @dptr, a

            mov   dptr, #dis_charhu
            mov   a, r0
            movc  a, @a+dptr
            mov   r1, a

            call  delay_outhu
```

```
                mov     dph, #21h
                mov     a, r1
                movx    @dptr, a
                cjne    r0, #39d, dis_loophu

                ret

delay_outhu:    mov     r3, #07fh
                djnz    r3, $
                ret

end

;*************************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;nopower.A51
;Created: 4-2-07
;Last Updated: 4-2-07
;
;Input:
;Output:
;Corrupted Registers: r0,r1,r3
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name    nopower_mod
public  nopower



nopower_seg   segment         code
rseg      nopower_seg

;*********************************************
;
```

```
;          No Power Display Module
;
;************************************************

;************************************************
;
; Displays Do not apply power
;
;************************************************



dis_addr4:
;line1 addresses
db       80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8ah, 8bh, 8ch, 8dh, 8eh, 8fh,
90h, 91h, 92h, 93h
;line2 addresses
db       0c0h, 0c1h, 0c2h, 0c3h, 0c4h, 0c5h, 0c6h, 0c7h, 0c8h, 0c9h, 0cah, 0cbh, 0cch,
0cdh, 0ceh, 0cfh, 0d0h, 0d1h, 0d2h, 0d3h


dis_char4:
;line1 characters
db       'S', 'o', 'f', 't', 'w', 'a', 'r', 'e', ' ', 'L', 'o', 'a', 'd', 'i', 'n', 'g', ' ', ' ', ' ', ' '
;line2 characters
db       'D', 'o', ' ', 'N', 'o', 't', ' ', 'A', 'p', 'p', 'l', 'y', ' ', 'P', 'o', 'w', 'e', 'r', ' ', ' '




nopower:       mov    r0, #0ffh
dis_loop4:     inc      r0
               mov    dptr, #dis_addr4
               mov    a, r0
               movc   a, @a+dptr
               mov    r1, a

               call     delay_out4
               mov    dph, #20h
               mov    a, r1
               movx   @dptr, a
```

```
                mov     dptr, #dis_char4
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_out4
                mov     dph, #21h
                mov     a, r1
                movx    @dptr, a
                cjne    r0, #39d, dis_loop4

                ret

delay_out4:     mov     r3, #07fh
                djnz    r3, $
                ret

end

;*************************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;player2d.A51
;Created: 4-2-07
;Last Updated: 4-2-07
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name    player2d_mod
public  player2d
```

```
player2d_seg   segment         code
rseg     player2d_seg

;************************************************
;
;         Player 2 Display Module
;
;************************************************

;************************************************
;
; Displays Player 2's Turn
;
;************************************************



dis_addrp2d:
;line1 addresses
db       80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8ah, 8bh, 8ch, 8dh, 8eh, 8fh,
90h, 91h, 92h, 93h
;line2 addresses
db       0c0h, 0c1h, 0c2h, 0c3h, 0c4h, 0c5h, 0c6h, 0c7h, 0c8h, 0c9h, 0cah, 0cbh, 0cch,
0cdh, 0ceh, 0cfh, 0d0h, 0d1h, 0d2h, 0d3h

;dis_char_seg segment code
;rseg dis_char_seg
dis_charp2d:
;line1 characters
db       'P', 'l', 'a', 'y', 'e', 'r', ' ', '2', 's', ' ', 'T', 'u', 'r', 'n', ' ', ' ', ' ', ' ', ' ', ' '
;line2 characters
db       'A', '=', 'G', 'a', 'm', 'e', ' ', 'M', 'e', 'n', 'u', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '



player2d:       mov     r0, #0ffh
dis_loopp2d:    inc     r0
                mov     dptr, #dis_addrp2d
                mov     a, r0
                movc    a, @a+dptr
```

```
                mov     r1, a

                call    delay_outp2d
                mov     dph, #20h
                mov     a, r1
                movx    @dptr, a

                mov     dptr, #dis_charp2d
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_outp2d
                mov     dph, #21h
                mov     a, r1
                movx    @dptr, a
                cjne    r0, #39d, dis_loopp2d

                ret

delay_outp2d: mov     r3, #07fh
                djnz    r3, $
                ret

end

;*******************************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;menudisp.A51
;Created: 11-29-04
;Last Updated: 12-7-04
;
;Input:
;Output:
;Corrupted Registers:
;
;*******************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)
```

```
name   menudisp_mod
public  menudisp
extrn   code(main)


menudisp_seg segment        code
rseg    menudisp_seg

;************************************************
;
;         Menu Display Module
;
;************************************************

;************************************************
;
; Displays 1=demo, 2=test, 3=game
;
;************************************************



dis_addr1:
;line1 addresses
db      80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8ah, 8bh, 8ch, 8dh, 8eh, 8fh,
90h, 91h, 92h, 93h
;line2 addresses
db      0c0h, 0c1h, 0c2h, 0c3h, 0c4h, 0c5h, 0c6h, 0c7h, 0c8h, 0c9h, 0cah, 0cbh, 0cch,
0cdh, 0ceh, 0cfh, 0d0h, 0d1h, 0d2h, 0d3h

;dis_char_seg segment code
;rseg dis_char_seg
dis_char1:
;line1 characters
db      '1', '=', 'D', 'e', 'm', 'o', ' ', '2', '=', 'T', 'e', 's', 't', ' ', ' ', ' ', ' ', ' ', ' ', ' '
;line2 characters
db      '3', '=', 'G', 'a', 'm', 'e', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '
```

```
menudisp:       mov     r0, #0ffh
dis_loop1:      inc     r0
                mov     dptr, #dis_addr1
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_out1
                mov     dph, #20h
                mov     a, r1
                movx    @dptr, a

                mov     dptr, #dis_char1
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_out1
                mov     dph, #21h
                mov     a, r1
                movx    @dptr, a
                cjne    r0, #39d, dis_loop1

                ret

delay_out1:     mov     r3, #07fh
                djnz    r3, $
                ret

end

;*************************************************************************
*
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;qblack.A51
;Created: 11-29-04
;Last Updated: 12-7-04
;
;Input:
;Output:
;Corrupted Registers:
```

```
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)

name   qblack_mod
public  qblack
extrn
        code(keypad,dlytime,queue,calib,display1,menudisp,lcdout,qloading,qloaded,feed
back,load,dispense)


qblack_seg     segment         code
rseg    qblack_seg

;************************************************
;
;       Queue Black Module
;
;************************************************


;************************************************
;
;       Queues up black checker
;************************************************



qblack:         clr     disp
                setb    qflag
tryagainq:      mov     a, queue_reg        ; check value in queue
                clr     c
                rrc     a
                jc      isblackq            ; determine color
                clr     p4.7
                call    dlytime
waiteq:         jnb     p1.3, waiteq
                call    dispense            ; wait for checker to dispense
                clr     disp
                call    queue               ; sample data
                mov     a, queue_reg        ; store on queue
```

```
                rrc     a
                mov     queue_reg, a
                clr     p1.7            ; for test purpose
                jmp     tryagainq

isblackq:       setb    p4.7            ; stop motor

                clr     qflag
                ret

end

;*************************************************************************
*
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;qred.A51
;Created: 11-29-04
;Last Updated: 12-7-04
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)

name   qred_mod
public  qred
extrn
        code(keypad,dlytime,queue,calib,display1,menudisp,lcdout,qloading,qloaded,feed
back,load,dispense)


qred_seg        segment         code
rseg    qred_seg

;*********************************************
;
```

```
;         Queue Red Module
;
;************************************************


;************************************************
;
;         Queues up red checker
;************************************************


qred:           clr     disp
                setb    qflag
tryagain1r:     mov     a, queue_reg        ; look at data in queue
                clr     c
                rrc     a
                jnc     isredr              ; is next chaecker red
                clr     p4.7
                call    dlytime
waitfr:         jnb     p1.3, waitfr
                call    dispense            ; wait for checker to dipense

                clr     disp
                call    queue               ; sample data
                mov     a, queue_reg        ; store in queue
                rrc     a
                mov     queue_reg, a
                clr     p1.7
                jmp     tryagain1r

isredr:         setb    p4.7                ; stop motor

                clr     qflag
                ret

end

;*********************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;display1.A51
;Created: 11-29-04
;Last Updated: 12-7-04
;
```

```
;Input:
;Output:
;Corrupted Registers:
;
;************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name   display1_mod
public  display1
extrn   code(main)


display1_seg   segment         code
rseg     display1_seg

;************************************************
;
;         Display1 Module
;
;************************************************

;************************************************
;
; Displays Senior Project 2007
;
;************************************************




dis_addr:
;line1 addresses
db       80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8ah, 8bh, 8ch, 8dh, 8eh, 8fh,
90h, 91h, 92h, 93h
;line2 addresses
db       0c0h, 0c1h, 0c2h, 0c3h, 0c4h, 0c5h, 0c6h, 0c7h, 0c8h, 0c9h, 0cah, 0cbh, 0cch,
0cdh, 0ceh, 0cfh, 0d0h, 0d1h, 0d2h, 0d3h

;dis_char_seg segment code
```

```
;rseg dis_char_seg
dis_char:
;line1 characters
db      ' ', 'A', 'u', 't', 'o', 'n', 'o', 'm', 'o', 'u', 's', ' ', 'C', 'o', 'n', 'n', 'e', 'c', 't', ' '
;line2 characters
db      ' ', ' ', ' ', ' ', 'F', 'o', 'u', 'r', ' ', '2', '0', '0', '7', ' ', ' ', ' ', ' '




display1:       mov     r0, #0ffh
dis_loop:       inc     r0
                mov     dptr, #dis_addr
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_out
                mov     dph, #20h
                mov     a, r1
                movx    @dptr, a

                mov     dptr, #dis_char
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_out
                mov     dph, #21h
                mov     a, r1
                movx    @dptr, a
                cjne    r0, #36d, dis_loop

                ret

delay_out:      mov     r3, #07fh
                djnz    r3, $
                ret

end

;*****************************************************************
;Justin Middleton
```

```
;Bradley University
;Electrical Engineering Dept.
;
;player1d.A51
;Created: 4-2-07
;Last Updated: 4-2-07
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name   player1d_mod
public  player1d



player1d_seg  segment         code
rseg     player1d_seg

;*************************************************
;
;         Player 1 Display Module
;
;*************************************************

;*************************************************
;
; Displays Player 1's Turn
;
;*************************************************



dis_addrp1d:
;line1 addresses
```

```
db      80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8ah, 8bh, 8ch, 8dh, 8eh, 8fh,
90h, 91h, 92h, 93h
;line2 addresses
db      0c0h, 0c1h, 0c2h, 0c3h, 0c4h, 0c5h, 0c6h, 0c7h, 0c8h, 0c9h, 0cah, 0cbh, 0cch,
0cdh, 0ceh, 0cfh, 0d0h, 0d1h, 0d2h, 0d3h

;dis_char_seg segment code
;rseg dis_char_seg
dis_charp1d:
;line1 characters
db      'P', 'l', 'a', 'y', 'e', 'r', ' ', '1', 's', ' ', 'T', 'u', 'r', 'n', ' ', ' ', ' ', ' ', ' ', ' '
;line2 characters
db      'A', '=', 'G', 'a', 'm', 'e', ' ', 'M', 'e', 'n', 'u', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '




player1d:       mov     r0, #0ffh
dis_loopp1d:    inc     r0
                mov     dptr, #dis_addrp1d
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_outp1d
                mov     dph, #20h
                mov     a, r1
                movx    @dptr, a

                mov     dptr, #dis_charp1d
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_outp1d
                mov     dph, #21h
                mov     a, r1
                movx    @dptr, a
                cjne    r0, #39d, dis_loopp1d

                ret

delay_outp1d: mov       r3, #07fh
```

```
                djnz    r3, $
                ret

end

;*************************************************************************
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;qloading.A51
;Created: 11-29-04
;Last Updated: 12-7-04
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name   qloading_mod
public  qloading
extrn   code(test)


qloading_seg  segment         code
rseg    qloading_seg

;************************************************
;
;         Queue loading Display Module
;
;************************************************

;************************************************
;
; Displays Loading Queue...
;
```

```
;***********************************************



dis_addr2:
;line1 addresses
db      80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8ah, 8bh, 8ch, 8dh, 8eh, 8fh,
90h, 91h, 92h, 93h
;line2 addresses
db      0c0h, 0c1h, 0c2h, 0c3h, 0c4h, 0c5h, 0c6h, 0c7h, 0c8h, 0c9h, 0cah, 0cbh, 0cch,
0cdh, 0ceh, 0cfh, 0d0h, 0d1h, 0d2h, 0d3h


dis_char2:
;line1 characters
db      'L', 'o', 'a', 'd', 'i', 'n', 'g', ' ', 'Q', 'u', 'e', 'u', 'e', '.', '.', '.', ' ', ' ', ' ', ' '
;line2 characters
db      'P', 'l', 'e', 'a', 's', 'e', ' ', 'W', 'a', 'i', 't', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '




qloading:       mov     r0, #0ffh
dis_loop2:      inc     r0
                mov     dptr, #dis_addr2
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_out2
                mov     dph, #20h
                mov     a, r1
                movx    @dptr, a

                mov     dptr, #dis_char2
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_out2
                mov     dph, #21h
                mov     a, r1
```

```
                movx   @dptr, a
                cjne   r0, #39d, dis_loop2

                ret

delay_out2:     mov    r3, #07fh
                djnz   r3, $
                ret

end
;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;redwin.A51
;Created: 4-12-07
;Last Updated: 4-12-07
;
;Input:
;Output:
;Corrupted Registers:
;
;*************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name   redwin_mod
public  redwin



redwin_seg     segment         code
rseg    redwin_seg

;************************************************
;
;         Red Wins Display Module
;
;************************************************
```

```
;************************************************
;
; Displays Red WIns
;
;************************************************



dis_addrre:
;line1 addresses
db      80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8ah, 8bh, 8ch, 8dh, 8eh, 8fh,
90h, 91h, 92h, 93h
;line2 addresses
db      0c0h, 0c1h, 0c2h, 0c3h, 0c4h, 0c5h, 0c6h, 0c7h, 0c8h, 0c9h, 0cah, 0cbh, 0cch,
0cdh, 0ceh, 0cfh, 0d0h, 0d1h, 0d2h, 0d3h

;dis_char_seg segment code
;rseg dis_char_seg
dis_charre:
;line1 characters
db      ' ', ' ', ' ', ' ', ' ', 'R', 'e', 'd', ' ', 'W', 'i', 'n', 's', '!', '!', '!', ' ', ' ', ' ', ' '
;line2 characters
db      ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '




redwin:         mov     r0, #0ffh
dis_loopre:     inc     r0
                mov     dptr, #dis_addrre
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_outre
                mov     dph, #20h
                mov     a, r1
                movx    @dptr, a

                mov     dptr, #dis_charre
                mov     a, r0
                movc    a, @a+dptr
```

```
                mov     r1, a

                call    delay_outre
                mov     dph, #21h
                mov     a, r1
                movx    @dptr, a
                cjne    r0, #39d, dis_loopre

                ret

delay_outre:    mov     r3, #07fh
                djnz    r3, $
                ret

end

;Justin Middleton
;Bradley University
;Electrical Engineering Dept.
;
;blackwin.A51
;Created: 4-12-07
;Last Updated: 4-12-07
;
;Input:
;Output:
;Corrupted Registers:
;
;*****************************************************************************
*


$nomod51
$Include(reg515.inc)
$Include(myvars.inc)



name    blackwin_mod
public  blackwin



blackwin_seg  segment         code
rseg     blackwin_seg
```

```
;************************************************
;
;          Black Wins Display Module
;
;************************************************

;************************************************
;
; Displays Black WIns
;
;************************************************



dis_addrbl:
;line1 addresses
db       80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8ah, 8bh, 8ch, 8dh, 8eh, 8fh,
90h, 91h, 92h, 93h
;line2 addresses
db       0c0h, 0c1h, 0c2h, 0c3h, 0c4h, 0c5h, 0c6h, 0c7h, 0c8h, 0c9h, 0cah, 0cbh, 0cch,
0cdh, 0ceh, 0cfh, 0d0h, 0d1h, 0d2h, 0d3h

;dis_char_seg segment code
;rseg dis_char_seg
dis_charbl:
;line1 characters
db       ' ', ' ', ' ', 'B', 'l', 'a', 'c', 'k', ' ', 'W', 'i', 'n', 's', '!', '!', '!', ' ', ' ', ' ', ' '
;line2 characters
db       ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '




blackwin:       mov     r0, #0ffh
dis_loopbl:     inc     r0
                mov     dptr, #dis_addrbl
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_outbl
                mov     dph, #20h
```

```
                mov     a, r1
                movx    @dptr, a

                mov     dptr, #dis_charbl
                mov     a, r0
                movc    a, @a+dptr
                mov     r1, a

                call    delay_outbl
                mov     dph, #21h
                mov     a, r1
                movx    @dptr, a
                cjne    r0, #39d, dis_loopbl

                ret

delay_outbl:    mov     r3, #07fh
                djnz    r3, $
                ret

end
```

Appendix D: Data Sheet

Dimensions:

Height 42 inches

Width 27.25 inches

Depth 20 inches

Power Requirements:

Voltage: 110 Volt AC

Current: 2 Amps