Final Project

Final Projects will be graded in lab on May 6, 2008. Projects are intended to be completed by individuals.

The lab instructor will assign a project to you; however:

a. you may *request* a project from those listed below, or
b. you may make a *written suggestion* describing an alternative project. Suggestions for projects other than those listed below must be as detailed and explicit as those below before they will be considered.

The lab instructor will determine whether your request or suggestion is appropriate. Although requests are often honored, there is no guarantee that the lab instructor will grant your request or suggestion. Reasons for not honoring your request may include (but are not limited to):

too many people wanting to do the same project,
the project is probably too easy for you, or
the project is probably too difficult for you.

Furthermore, your lab instructor may add or subtract requirements from any project. It is the responsibility of the student to resolve possible interpretations of the following descriptions *before* the projects are graded.

Rules applying to all projects

All projects must start up in a well-defined state when the project box is turned on. No project should require the project box ON switch to be toggled as part of its subsequent operation.

Your grade will be partially based on neatness. This includes using wires of the appropriate lengths and colors. Although you may assign colors of your choice to various other tasks or concepts, all Vcc wires must be red, and all ground wires must be black.

The Descriptions

In the following descriptions, switches will be designated as momentary (i.e., spring loaded like the set of two in the PAD trainers), or double-throw (i.e., can remain in either of two positions, like the set of eight in the PAD trainers). Note that the double-throw switches in the PAD trainers are *not* debounced.

Note that projects have different *estimated* difficulty ratings. In grading, projects with lower difficulty ratings will be held to higher standards of neatness, ease of use, clarity of function, etc. Also, the estimated difficulties assume that you will not be using the BASIC stamp. You may be permitted to use the BASIC stamp on your project, subject to restrictions specified by the instructor. BASIC Stamp projects will eventually be added to the end of this document.

Project Descriptions Page

1. Traffic Light (Difficulty: 8)

A traffic light is at the intersection of a relatively large highway and a small residential street. Generally, the light is green for vehicles traveling along the highway, and red for the side street. When a car needs to enter the highway from the side street, a sensor in the road triggers (treat as a pulse of unknown width). Generally, when the sensor is tripped, the lights will both change. However:

a. Before changing to red, each light must be yellow for ~2 seconds.
b. A light will become green only after the opposite light becomes red.
c. When either light becomes green, it must *stay* green for a minimum of 20 seconds.
d. The sensor may be triggered or retriggered at any time during the red-green-yellow cycle. If a new trigger occurs while the side street has a green light, there is no effect. If a trigger occurs while the side street is red, the highway will go immediately yellow to red (expect that rule c above may delay it). If a car re-triggers the detector while the side street is "yellow", assume that it will not enter the highway. So, in this circumstance, the yellow cycle will finish normally, then the main highway will be green for ~20 seconds, the highway will be yellow for ~2 seconds, and then the main highway will be red for ~22 seconds.

Input summary: 1 momentary switch (road sensor)
Output summary: 6 LEDs (2 traffic lights; note: we have colored LEDs)

2. Railroad Crossing (Difficulty: 6)

A traffic light is located at a railroad crossing. The track is used by both eastbound and westbound trains. There are three sensors along the railroad track: one at the crossing, and one on either side of it, ½ mile from the crossing. At all times, each sensor indicates whether some portion of the train is at that location. Trains may be up to 1¼ mile long, so that at any moment 0, 1, 2, or 3 of the sensors may be active.

a. Normally, the traffic light is green. However, when a train approaches within ½ mile of the crossing, the traffic light must go yellow for ~2 seconds, and then stay red until the train is completely past the crossing.
b. Trains do not reverse direction while in the vicinity of the crossing.

Input summary: 3 double-throw switches (the sensors at the crossing and at ± ½ mile)
Output summary: 3 LEDs (traffic light; note: we have colored LEDs)

2c. Enhanced Railroad Crossing (Car Warning Lights) (Difficulty: 7)

There is also a sensor in the road at the crossing, which at all times indicates whether there is a car crossing the tracks. On either side of the crossing, ¼ mile from it along the track, there are two warning lights (which warn trains if there is a car on the crossing).
Additional Input summary: 1 double-throw switches (the car sensor at the crossing)
Additional Output summary: 2 LEDs (warning lights)

3. Advanced Vending Machine (Difficulty: 10)

A vending machine has two selections: pretzels (20¢), and snack cakes (40¢). Coins are entered into the machine (nickels, dimes, and quarters only) first. When a selection is made, if funds are sufficient, the item is dispensed and change is returned.

a. A LED will indicate when a successful sale is made. That is, one LED for each product. This LED should light for ~1 second.
b. If a selection is made, but funds are insufficient, a LED should light for ~1 second. During this time, the selections should be locked out. During this time, you may assume that the user will not input new coins.
c. After a sale, use a 7-segment display to indicate the number of *nickels* returned as change. This value should be displayed for ~1 second before the vending machine resets itself for future purchases.
d. Once more than 40¢ is input by a user, the machine does not have to accept more coins. Turn on a LED to indicate when the machine will not accept more money. When this LED is on, we'll assume that new coins are immediately returned.
e. The machine should have a "coin return" switch, which returns all change and resets the machine. All coins returned should be displayed as change as in (c) above.
f. (optional) The vending machine initially contains only 9 of each item. When an item becomes unavailable, an "empty" light comes on. If the user attempts to purchase such an item, a "make another selection" LED lights for ~ 1 second. You may implement resetting the counters by turning the power off and on.

Input summary: 6 momentary switches (3 coin sensors, 2 product selectors, and 1 coin return)

Output summary: 4 LEDs (sale of pretzels, sale of snack cakes, insufficient funds, and maximum funds)
1 7-segment display (change)
3 optional LEDs (out of pretzels, out of snack cakes, and make another selection)

4. Vending Machine (Difficulty: 8)

A vending machine has two selections: pretzels (20¢), and snack cakes (40¢). *First*, a selection is made. *Then*, coins are entered into the machine (nickels, dimes, and quarters only). When the amount of coins is sufficient, the item is dispensed and change is returned.

a. A LED will indicate when a successful sale is made (one LED for each product). This LED should light for ~1 second before the vending machine resets itself for future purchases.
b. After a sale, use a 7-segment display to indicate the number of *nickels* returned as change. This value should be displayed for ~1 second.
c. The machine should have a "coin return" switch that clears the selection, returns all coins as change (see b above), and resets the machine.
d. The vending machine initially contains only 9 of each item. When an item becomes unavailable, an "empty" light comes on. If the user attempts to purchase such an item, a "make another selection" LED lights for ~ 1 second. You may implement resetting the counters by turning the power off and on.

Input summary: 6 momentary switches (3 coin sensors, 2 product selectors, and 1 coin return)

Output summary: 5 LEDs (sale of pretzels, sale of snack cakes, out of pretzels, out of snack cakes, and make another selection)
1 7-segment display (change)

5. Automatic Geiger Counter (Difficulty: 6)

An Automatic Geiger Counter (AGC) has a two-digit 7-segment display showing "counts", which indicate the number of radiation units detected over a 4 second period. The AGC has the following operation:

a. A new radioactive sample enters the detection chamber every 8 seconds.
b. For the first 4 seconds, the AGC counts the number of radiation "hits". The main clock on the project kit will be used to create a pulse train to represent this sample; your instructor will adjust the frequency knob to higher values to simulate a more radioactive sample. As a result, you will not be able to use the main clock for timing purposes. Instead, you will have to use a 555 chip. During these 4 seconds, the counter display must be blank.
c. The final radiation count (from b above) is held and displayed during the final 4 seconds.
d. There must be a manual over-ride switch, which can only activate when the AGC is in the "hold and display" mode. When this switch is toggled, the "hold and display" mode will remain on indefinitely. Normal operation (beginning at (a) above) will resume when this switch is toggled a second time.
e. If the number of counts measured (during (b) above) is greater than 99, then a warning LED will light. When this LED lights, it should have the same effect as toggling of the manual over-ride switch (described in d above).

Input summary: 1 momentary switch (manual over-ride)

Output summary: 1 LED (warning light)
2 7-segment displays (radiation count)

5b. Automatic Geiger Counter (Enhanced Display) (Difficulty: 7)

For easy viewing, there is also a bar graph of 9 LEDs displaying the activity. Each step on the bar graph represents 10 counts: 0–9 counts lights nothing, 10–19 counts lights one LED, 20-29 counts lights two LEDs, etc. This is active during both the accumulation and the display phase.

Additional Output summary: 10 LEDs (bar graph; chip-sized packages with 8 onboard LEDs are available)

6. Drunk Driver Control I (Difficulty: 8)

A car has a circuit to prevent drunk driving. When a user inserts a key into the ignition (i.e., when a switch is thrown), two 1-digit numbers appear on separate seven-segment displays. In addition, a timer begins counting down on a third seven-segment display. The user has 9 seconds to input the absolute value of the difference between the two values displayed.

a. To generate random numbers, use two separate 555 timers, each operating at a very high frequency and connected to a separate 74192 counter chip. It should be possible for either of the random number displays to be the largest.
b. The current value of the user input will be displayed on a fourth seven-segment display. A toggle switch will be used to increment this value, and a second toggle switch to enter this value.
c. If the correct value is entered, then an LED will light to indicate ignition.
d. If the timer runs out (no value is entered), or if an incorrect value is entered, then a warning LED must go on. This LED must stay on for ~10 minutes, during which time the ignition switch should have no effect.
e. When the key is removed after ignition, the ignition light should go off and the system reset. However, if the key is removed during the 9-second entry window or after failure, the event sequence should not be interrupted.

NOTE: Yes, the warning LED must really light for a full 10 minutes. During development and testing, you may shorten this to a few seconds by *temporarily* increasing the clock rate.

Input summary: 1 double-throw switch (ignition/start)
2 momentary switches (increment, and enter)

Output summary: 2 LEDs (incorrect value/wait 10 minutes, and ignition)
4 7-segment displays (2 test values, user input value, and timer)

7. Drunk Driver Control II (Difficulty: 10)

A car has a circuit to prevent drunk driving. When a user inserts a key into the ignition (i.e., when a switch is thrown), two 2-digit numbers appear on separate 7-segment displays. In addition, a timer begins counting down on a fifth 7-segment display. The user has 9 seconds to input the absolute value of the difference between the two values displayed.

a. The current value of the user input will be displayed on two separate seven-segment displays. A toggle switch will be used to increment the ones place, and a separate toggle to increment the tens place. A third switch will be used enter this value.
b. If the timer runs out (no value is entered), or if an incorrect value is entered, then a warning LED must go on. This LED must stay on for ~10 minutes, during which time the driver may not start the car.
c. If the correct value is entered, then an LED will light to indicate ignition.
d. To generate random numbers, use two 555 timers, each operating at a very high frequency and connected to a separate pair of 74192 counter chips.
e. If the key is removed during the 9-second entry window or after failure, the event sequence should not be interrupted.
f. When the key is removed after ignition, the ignition light should go off and the system reset. This includes case (e) if the correct entry was made.
g. If the key is not removed during the 10 minute warning period, then the warning period should continue until the key is removed.

Input summary: 1 double-throw switch (ignition/start)
3 momentary switches (increment ones, increment tens, and enter)

Output summary: 2 LEDs (incorrect value/wait 10 minutes, and ignition)
7 7-segment displays (2 test value pairs, user input value pair, and timer)

8. Stopwatch (Difficulty 8)

A stopwatch is desired that can be used in "count up" mode for things like track meets, but in "count down" mode for things like an egg timer. The output of the stopwatch should be in the form "###.#" (with appropriate blanking). All features of the stopwatch must operate using only two buttons or switches. One of the buttons will act as a "menu select" button. The other "operate" button will cycle through the options for each menu. Features must include:

a. During "normal" operation, the second button acts as a start/stop/continue button.
b. The user can select between "count up" and "count down" mode.
c. Count down mode always stops and holds at zero. If switching to count up mode at that point results in the stopwatch counting up, then an additional "timer active" LED must be included. If switching to count up mode at that point results in the stopwatch holding at zero, no extra LED is needed.
d. The user can load a 4-digit value into memory. The same 7-segment displays must be used to show the value being loaded.
e. There must be a way to reset the timer. While in "count up" mode, this sets the timer to zero. While in "count down" mode, this sets the time to the value loaded in memory.
f. The user can select between "alarm on" and "alarm off" modes, indicated by an LED. Use a second LED to indicate when the alarm is sounding. In count down mode, the alarm sounds when zero is reached. In count up mode, the alarm sounds when the loaded value is reached. When the alarm becomes active, the system mode should automatically be changed to "shut off alarm" mode. The alarm is active until the user turns it off.
g. An LED panel must indicate the currently selected menu. Also, you must provide the instructor with an instruction "manual" so that all features can be verified. Score will be lower if the instructor cannot figure out operation based on manual alone.

Input summary: 2 momentary switches (mode select and mode action)

Output summary: multiple LED panel (mode indicators; chip-sized packages with 8 LEDs are available)
2 LEDs (alarm on, and alarm sounding; buzzers are available for realistic behavior)
1 LED (timer active) *if* count down to zero does not turn off timer.
4 7-segment displays (current time/alarm time)

8b. Stopwatch (Improved Loading) (Difficulty: 9)

Standard project requires the ability to load a value into memory. For this enhancement, it must be possible, starting from the clear state, to load a large value into memory in less time than it takes to count up to that value. Suggested possibilities are:

a. Count up "warp": If the "operate" button is held for a certain time, counting speeds up by a factor near 10×.
b. Set each digit of the load value separately.

No Input/Output changes.

9. Package Sorter (Difficulty: 8)

A mail sorting system has deflector gates along a linear conveyor belt, having stations "ABCDE". While operating, a new package enters the system at point "A" each second; each package has a two bit bar code, indicating its final destination. The bar code is read at point "A" on the conveyor belt. After one second passes, each package moves to the right one station (e.g., a package at C moves to D).

a. The output will consist of 5 seven-segment displays, one for each station. Once the system is started, each display indicates the value of the bar code of the package currently at that station. If there is no package at any particular station, then the display there should be blank.
b. The system must start up indicating that all stations are empty.
c. If a package has the barcode 0_{10} (00_2), it is deflected from the conveyor belt at B. Whenever this happens, a corresponding LED must light up for ½ second to indicate a deflection occurring there. Note that stations CDE will subsequently be empty, as that position moves down the belt.
d. Similarly, packages with barcode 1_{10} should deflect at station C, barcode 2_{10} should deflect at D, and barcode 3_{10} should deflect at E.
e. To simulate packages arriving with random bar codes, generate bar codes using a 555 timer operating at high frequency and connected to a pair of JK Flip flops setup as a 2-bit counter.
f. A switch controls whether the conveyor belt is running or stopped. While stopped, a second switch may be toggled to clear all packages from all stations.

A	B(0)	C(1)	D(2)	E(3)
2				
0	2			
1	0*	2		
2	1		2*	
3	2	1*		
1	3	2		
1	1	3	2*	
0	1	1*	3	
2	0*	1*		3*
2	2			
1	2	2		
3	1	2	2*	

An example output over a few seconds is provided to the right. The first row would occur immediately after start up, or after toggling the clear switch. A "*" indicates a deflection LED being temporarily lit.

Hint for determining when blanking is appropriate: note that no package of type 0_{10} will ever reach station C.

Input summary: 1 double-throw switch (run/stop)
1 momentary switch (clear)

Output summary: 5 7-segment displays (bar code at each station)
4 LEDs (deflection indicator)

10. Combination Lock (Difficulty: 7)

A security lock has 4 buttons. Four buttons must be pressed in the proper sequence, with repeats possible, to open the lock (e.g., ACDB or BBAB or DDAA).

a. The entry code must be programmable using switches. Two switches are needed to represent each digit of the sequence, for a total of 8 switches. In a real application, these switches would be hidden from the person trying to open the lock.
b. There must be a clear switch to restart the sequence if the user accidentally presses a wrong button.
c. If the sequence is incorrect after 4 switches have been pressed, then all further attempts must be locked out, and an alarm activated. The alarm must stay active until the lock is reset.
d. Once the lock is opened, it remains open for 3 or 4 seconds. After this, the system is reset (to the same combination) for the next user.
e. The system must initialize to a pre-determined state when initially powered.

Input summary: 8 double-throw switches (code programming; chip-sized packages of switches, as well as groups of resistors, are available)
6 momentary switches (4 for code entry, clear, and reset)

Output summary: 2 LEDs (open, and alarm)

10b. Combination Lock (with Program Mode) (Difficulty: 8)

Instead of programming the entry code by means of 8 switches, there is a single switch (in a secure area) that controls whether the system is in "program" mode or "entry" mode.

a. While in "program" mode, pushing the four buttons in sequence sets the entry code. The last four buttons pushed, in the order entered, are the entry code. Implementing a reset during programming is optional. You should clearly define the behavior if fewer than 4 buttons are pushed before leaving program mode.
b. While in "entry" mode, behavior is the same as in the unmodified project

***Removed* Input summary**: 8 double-throw switches (code programming)

Additional Input summary: 1 double-throw switch (mode)

11. Serial Data Buffer (Difficulty: 7)

A 6 bit stream of bits is generated in one register, then moved one bit at a time (at ~1 Hz) over a serial wire, to a receiving register. Five of the 6 bits (A_0 through A_4) represent data, and the final bit (A_5) is a "validate" bit. The validate bit is automatically generated so that there is an even number of 1's in the 6 bits sent. Due to noise, any *one* of the 6 bits being transferred across the serial wire may become corrupted (inverted). We'll assume that the noise is never so bad that two bits get corrupted.

c. A "register panel" displays the contents of each of a source register and a receive register. The panel consists of 6 LEDs (including the validate bit) and a pair of 7-segment display showing the value of the data bits only.
d. The user must set the original 5 data bits using 5 switches. While the data is being adjusted, the source register panel must be continuously updated.
e. The receive register must be initialized blank on power up.
f. When a send button is pushed, the receive register panel must go blank and then data transfer must start. The user must be unable to change the original values send is complete, and the source register panel should not change. A "sending" LED must be lit during transfer.
g. Bits are transferred one at a time at ~1 Hz, starting with the LSB. The receive register is a shift register, loading through the MSB (see sample output). LEDs should light or not, as appropriate, as each bit is transferred.
h. When the transfer is complete, the 7-segment displays of the receive register panel should indicate the value of the received data. This, along with the LEDs, should remain unchanged until the next time the send button is pressed.
i. There must be a switch for the instructor to introduce "noise" (that is, invert the transmitted bit) during the transfer process. No more than one bit per transfer will be corrupted.
j. If the signal is corrupt, a warning light should come on. Hint: remember that a multiple input XOR gate can be used as a parity checker.
k. If the received message is corrupt, the message should be resent.
l. Once all bits are successfully sent, the user can then reset the 5 data switches, and send another set.

The following table illustrates what the LEDs might look like after the send button is pushed. No noise error is introduced in this example. A is the source panel, and B is the receive panel.

A_5	A_4	A_3	A_2	A_1	A_0		B_5	B_4	B_3	B_2	B_1	B_0
1	0	1	1	0	1							
1	0	1	1	0	1		1					
1	0	1	1	0	1		0	1				
1	0	1	1	0	1		1	0	1			
1	0	1	1	0	1		1	1	0	1		
1	0	1	1	0	1		0	1	1	0	1	
1	0	1	1	0	1		1	0	1	1	0	1

Input summary: 6 double-throw switches (noise and 5 for data)
1 momentary switches (send)
Output summary: 12 LEDs (6 for original data, 6 for received data; chip-sized packages with 8 onboard LEDs are available)
2 LEDs (sending and corrupt)
4 7-segment displays (two pairs, source data and received data)

12. Adding Machine (Difficulty: 11)

An adding machine adds or subtracts a one-digit number to a bank that can range between -99 and 99. All three digits are shown on 7-segment displays. To operate, a momentary switch is toggled until the desired one digit input is displayed. Then, an "add" or a "subtract" button is pushed, and the results are shown in a two digit display, replacing the previous display. You should consider using "twos complement" numbers for this project. This format, as well as addition and subtraction, will be covered in lecture shortly after the final projects are started.

a. All displays must initialize to zero. Also, there must be a "clear" switch to reset the system to the initial state.
b. There must be a way to quickly preset the 2-digit display to any desired value; a set of switches representing the desired value in any convenient format may be used along with a load button.
c. There must be an "overflow" LED when the result is greater than 99 or less than –99. Once overflow occurs, no more additions or subtractions are allowed.
d. There must be a "negative" LED to represent negative numbers (a minus sign), for use when the result is less than zero. This should be suitably placed next to the bank.

Note that, since zero can be loaded into the bank, the primary purpose of the "clear" switch is to recover from an overflow.

Input summary: 8 double-throw switches (number to load)
5 momentary switches (increment input, add, subtract, clear, and load)
Output summary: 2 LEDs (overflow and negative bank)
3 7-segment displays (1 for input, 2 for bank)

12b. Adding Machine (Positive Bank) (Difficulty: 7)

The bank need only handle positive numbers: 0 through 99. You should **not** use "twos complement" numbers for this project.

Removed Output summary: 1 LED (negative bank)

12c. Adding Machine (Octal Bank) (Difficulty: 9)

All numbers in the project are displayed in octal format. Bank can range from -77_8 to 77_8, and one can add or subtract from 0 to 7.

Modified Input summary: 6 double-throw switches (number to load)

13. Chip Tester (Difficulty: 7)

The circuit must determine whether all 4 gates of three different types of quad-gate chips work (7400, 7408, 7432, or 7486). There is one open "bay" where the chip to be tested will be placed. To operate, the user places a chip into this bay. The user specifies which type of chip it is by throwing one of three SPDT switches. Then, the user presses a toggle "start" switch. If all 4 gates on the chip work perfectly, for all 4 possible input conditions, then an "OK" light should come on. If any of the 4 gates fails to function for any of the 4 possible input combinations, then a "BAD chip" light should come on. There should be 4 additional warning lights that indicate which, if any, of the gates are bad. All LEDs remain lit until the next test is initiated by the user again pressing the "start" switch. Also, the second momentary switch can be used to clear all of the LEDs after a test..

a.

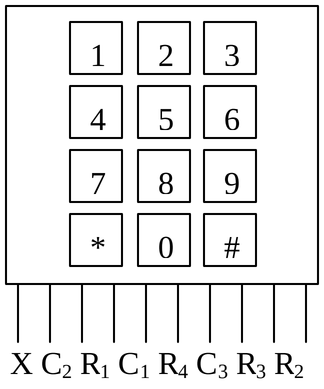
Input summary:	3 switches (test type 1, test type 2, and test type 3)
	2 momentary switches (start, clear)
	1 test bay (for chip to be tested)
Output summary:	6 LEDs (OK, BAD, and 4 gate-specific warnings)

14. Telephone Dialer (Difficulty: 11)

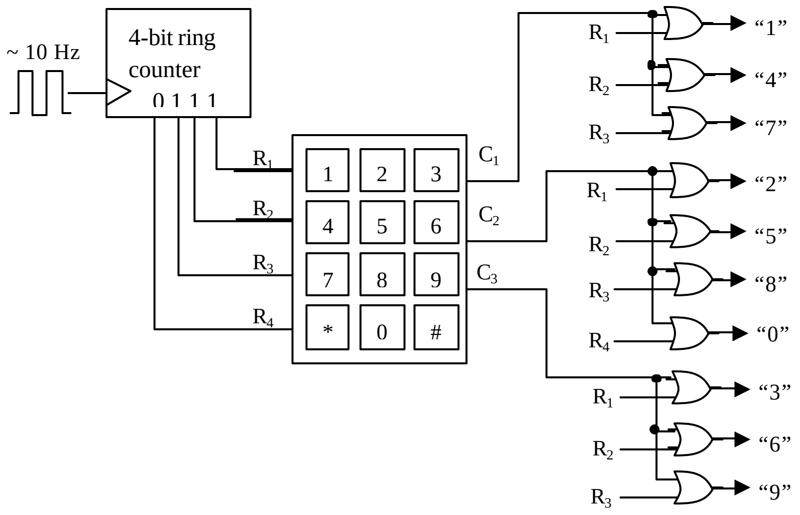
Telephone keypads have 12 keys, arranged in a grid. The keypad has 7 wires connected to it. When a key is pressed, two of the wires are connected together. For example, when the "8" is pressed, wires R_3 ("row 3") and C_2 ("column 2") are connected together. When the user presses 7 numerical keys in any combination, the values of these keys should be displayed in order by a group of 7-segment displays. An active-low suggestion for use of the keypad is provided. Note that the circuit fragment shown has a problem with repeating digits (i.e., when you hold down a key for an extended time) that needs to be corrected.

a. All displays must initialize blank.
b. The first digit pressed should be displayed in the MSD, the seventh in the LSD.
c. There should be a reset switch (hang up).

Input summary: 1 12-button keypad
1 momentary switch (reset)
Output summary: 7 7-segment displays



Unit layout



14b. Telephone Dialer (One Digit) (Difficulty: 9)

Use only one 7-segment display, which should initialize blank. Thereafter, it should display the most recent number pressed. Use the funny shapes which our displays show for 1010 and 1110 to indicate the * and # keys respectively.

In addition, have an LED which lights while any key is being pressed, and one 7-segment display which shows the total number of button presses since the project was turned on. When this display reaches 9, it should roll over to 0 with the next button press.

Modified Input summary: 1 12-button keypad
Modified Output summary: 2 7-segment displays (last key, and number of presses)
1 LED (key pressed)

15. Binary to Decimal Converter (Difficulty: 9)

The binary to decimal converter takes an 8 bit binary number and converts it to decimal form. The user inputs an unsigned binary number by setting eight double throw switches. Calculation takes place each time a momentary switch is thrown. The output is shown in decimal using three 7-segment displays.

a. All displays must initialize blank.
b. The first two digits should be blank as appropriate.
c. Once the user changes any part of the binary number by throwing a switch, the display should blank until the load switch is pressed again.

Input summary: 8 double throw switches (number to load)
1 momentary switch (load)
Output summary: 3 7-segment display (result of previous load)

15. Combat Simulation Game (Difficulty: 10)

There is a single score that indicates the current state of the game. The score begins at 7, and the game ends either when the score reaches 14 (player wins), or when it reaches 0 (player loses). The user selects an action (sword attack, sword defense, magical attack, or magical defense), and then presses "go". The circuit similarly chooses (randomly) one of these 4 choices. The score is adjusted based on the following chart.

a. The current score is displayed on a bar chart of 14 LED's.
b. A 7-segment display will show the most recent choice of the machine player
c. A yellow LED will light if the score reaches 0 (player loses).
d. A green LED will light if the score reaches 14 (player wins).
e. Pressing reset returns the score to 7, and turns off both the "win" and "lose" LED's. It also blanks the 7-segment display.

Input summary: 2 double throw switches (to choose actions 0, 1, 2, or 3)
2 momentary switches (go, restart)
Output summary: 14 LED's (score)
2 LED's (win and loss)
A 7-segment display (machine action indicator)

		Player Choice			
		attack	defend	magic attack	magic defense
Machine Decision		0	1	2	3
Attack	0	0	0	+1	-1
Defend	1	0	0	+2	-2
Magic Attack	2	-1	-2	0	+3
Magic Defense	3	+1	+2	-3	0

BASIC Stamp Proto-ideas

1. Battleship

The BASIC Stamp generates an ocean (a 5×5) grid of spaces, into which it randomly places two ships, one of which is 2 squares long, and the other is 3 squares long. The ships may be oriented horizontally or vertically. The player has 12 shots with which to sink both ships. Remaining ammunition is shown on a pair of 7 segment displays. Momentary switches or potentiometers are used to select the row and column for the player's next shot (values 1 through 5 shown on another pair of 7 segment displays). Another switch is used to confirm/fire the shot. After each shot, the ammo remaining is reduced by one. If the shot is a hit, an LED activates for 1 second. If the shot is a miss, a different LED lights for one second. The user must keep track of which destinations have already been fired upon. Each time a ship is sunk, another "sunk" LED will light and remain lit for the duration of the game. Obviously, there are two such "sunk LEDs. Finally, there will be a "game won" LED (if both ships are sunk before the ammunition is depleted) and a "game lost" LED (if one or more ships survive and the ammo runs out). Finally, there should be a game reset/start switch. This switch will also be used to initialize the random number generator.

Input summary:	Momentary switch for fire
	Momentary switch for start/restart game
	Two Potentiometers or switches for row/column selection.
Output summary:	6 LEDs (hit, miss, win, lose, sunk A, sunk B)
	4 7-segment displays (row, column, $2 \times$ ammo)
Basic Stamp:	No restrictions

2. Slot Machine

Using a potentiometer, the user will place a bet ranging from 1-3 credits. The result is displayed on a 7 segment display. Once a desired bet is selected, the machine is activated by a momentary switch. Three separate 7 segment displays will scroll through numbers from 0-4 at medium frequencies (around 8 to 15 Hz). These 7 segment displays will stop scrolling in sequential order; the first stopping after 3 seconds, the second after 4 seconds and the third after 5 seconds. During this time, the user may not bet nor restart. A winning spin will consist only of three identical numbers showing after all three displays have stopped scrolling. The payouts should be as follows: 000 = 5 times the bet, 111 = 10 times the bet, 222 = 15 times the bet, and 333 = 20 times the bet. Obviously a losing spin will display 0 for the payout.

Input Summary:	1 momentary switches(pull lever/start)
	1 potentiometer (select wager)
Output Summary:	6 7-segment displays(1 for wager, 3 for the symbols, 2 for payoff)

3. Switch-a-mole

This is a game similar to whack-a-mole. After pressing a momentary start switch, one of 6 LED's will light for a 3 second time period. Each of the 6 LED's has a corresponding button. The user must press the correct button before the light turns off, or the game ends. If the user presses the correct button, then another light is randomly activated, but for a slightly shorter duration. A pair of 7-segment displays will show the number of buttons that have been correctly pressed since the start button was pressed. Typical play will result in scores of around 25. The duration for each light will be no more primitive than bilinear.

Input Summary:	1 momentary switches(pull lever/start)
	6 debounced buttons
Output Summary:	2 7-segment displays
	6 LEDs

4. Simon Memory Game

A random array having values from 0 to 3 is generated, having length 32. Each value is represented by a unique colored LED, and by a unique musical tone. At first, the length of the array is 1. The tone and light is displayed for 0.5 seconds. The user then presses buttons corresponding to the generated value, which also light the LED's and play the tone for 0.5 seconds each. If incorrect, the game is over. If correct, the STAMP adds another element to the array, and the user attempts to reproduce the entire array.

Input summary:	Momentary switch "start"
	4 buttons A, B, C, D
Output summary:	4 LEDs (A, B, C, D)
	One speaker
	Two 7 segment displays (number remembered so far)
Basic Stamp:	No restrictions

4. Push Push Revolution

A random value ranging from 0 to 3 is generated. Each value is represented by a unique colored LED, and by a unique musical tone. The tone and light is displayed for "A" seconds. The user then presses the button corresponding to the generated value, before the time interval is over. If incorrect or too late, the game is over. If correct, the STAMP waits for the same time interval "A", then generates a new value. The time interval starts at 2.0 seconds, and gradually decreases to a minimum value of 0.4 seconds.

Input summary:	Momentary switch "start"
	4 buttons A, B, C, D
Output summary:	4 LEDs (A, B, C, D)
	One speaker
	Two 7 segment displays (number of tones pressed so far)
Basic Stamp:	No restrictions

1. Roulette
2. Blackjack

Generating "Random" numbers using the BASIC Stamp

PBASIC has a 'RANDOM' statement that can (sort of) be used to generate random values. RANDOM works by taking a variable and rearranging the bits randomly. It always generates the same random sequence every time the program is run. So that we start in some unknown step in this sequence, we will connect a start button to a pin of the BASIC stamp. The BASIC stamp will continue to quickly generate (unused) elements in the 'random' sequence until this button is pressed, after which the sequence should appear to be more truly random.

```
number VAR word
newvar VAR byte
number = 11000    'initialize 'number' to have 8 on and 8 off bits
DO
    RANDOM number
LOOP UNTIL IN1 = 0
newvar = number/16384    'will convert the range to be 0 to 3 instead of 0 to 65537
```

Number is a variable that will store the 'random' number. It ranges from 0 to 65537. This variable should not be set in any other portion of the program.

The loop cycles through the 'random' sequence until a start signal is received (in this case, on pin 1).