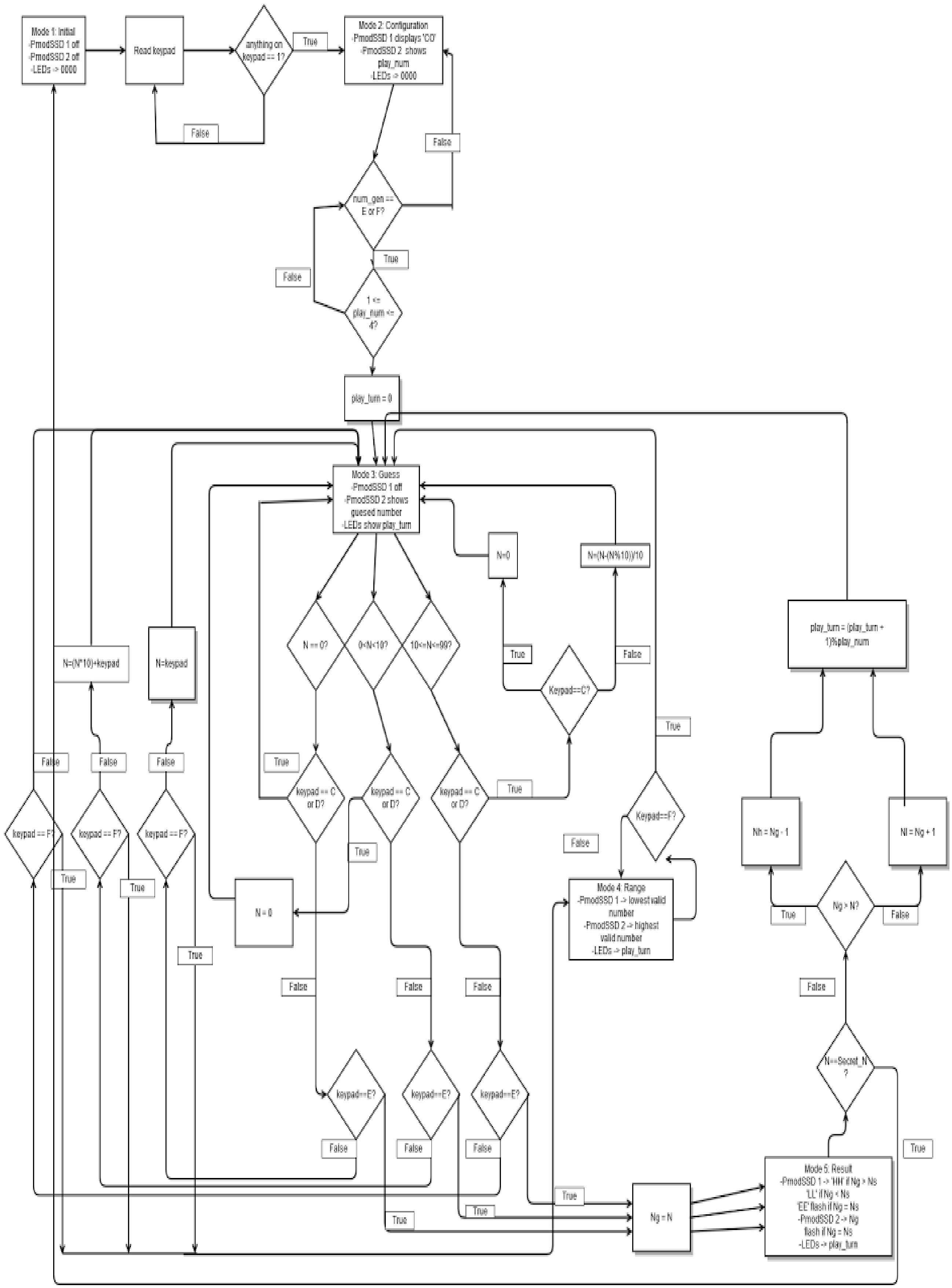
CPEG 222

Project 2 Flow Chart

Katie Black

Byron Lambrou

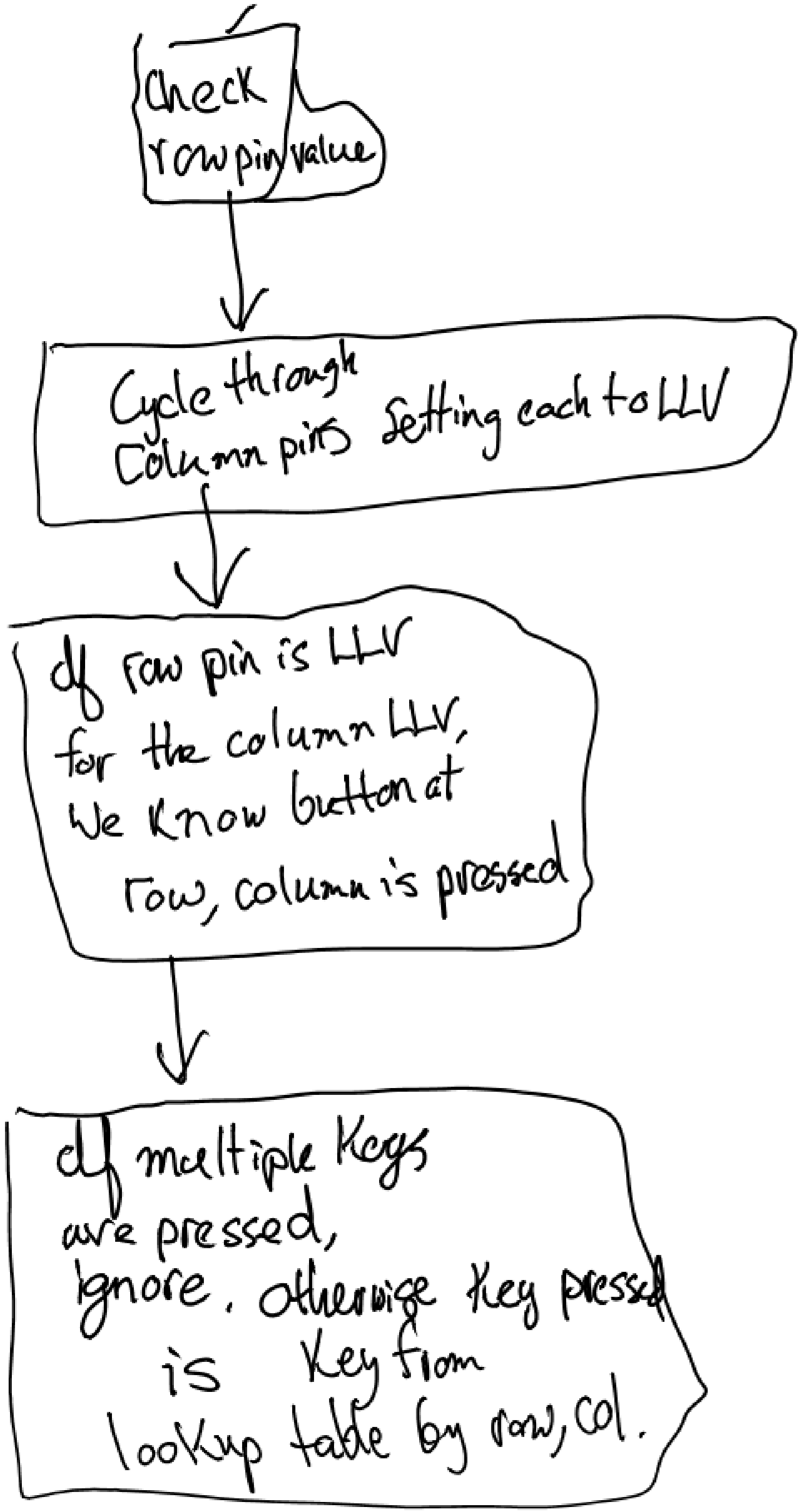


Mode 1: Initial
-PmodSSD 1 off
-PmodSSD 2 off
-LEDs -> 0000
Read keypad
anything on keypad == 1?
True
False
Mode 2: Configuration
-PmodSSD 1 displays 'CO'
-PmodSSD 2 shows play_num
-LEDs -> 0000
False
num_gen == E or F?
True
False
1 <= play_num <= 4?
play_turn = 0
Mode 3: Guess
-PmodSSD 1 off
-PmodSSD 2 shows guesed number
-LEDs show play_turn
N=0
N=(N-(N%10))/10
N == 0?
0<N<10?
10<=N<=99?
True
False
Keypad==C?
N=(N*10)+keypad
N=keypad
play_turn = (play_turn + 1)%play_num
keypad == C or D?
keypad == C or D?
keypad == C or D?
True
True
True
True
Keypad==F?
False
False
False
False
keypad == F?
keypad == F?
keypad == F?
True
True
True
N = 0
Mode 4: Range
-PmodSSD 1 -> lowest valid number
-PmodSSD 2 -> highest valid number
-LEDs -> play_turn
Nh = Ng - 1
Nl = Ng + 1
Ng > N?
True
False
False
N==Secret_N ?
True
False
False
False
keypad==E?
keypad==E?
keypad==E?
False
False
False
True
True
True
Ng = N
Mode 5: Result
-PmodSSD 1 -> 'HH' if Ng > Ns
'LL' if Ng < Ns
'EE' flash if Ng = Ns
-PmodSSD 2 -> Ng
flash if Ng = Ns
-LEDs -> play_turn

Questions to Address:

Question 1: In Lab1, a timer interrupt was used. A timer interrupt occurs after a certain number of clock cycles (specified by the timer parameters) have passed. In this lab we are using a Change_Notice interrupt. A Change_Notice interrupt is generated in response to a change of state on selected input pins. When we press a button the keypad, this will change the value of a keypad pin, which will generate an interrupt corresponding to keypad input, which we can then process accordingly.

Question 2: The keypad takes inputs on the column pins, and gives outputs on the row pins. In order to determine which key is pressed, we will output logic low to all of the column pins. Then, in the event that a button is pressed, we will receive a change_notice interrupt, because the value of the corresponding row pin will have changed. Once we have received the interrupt, we can sweep through the column pins, putting a logic low voltage to only one pin at a time. If only one button is pressed, then the row pin will have a logic low voltage only when the column pin corresponding to the button is also at logic low.


Check
row pin value
Cycle through
Column pins Setting each to LLV
If row pin is LLV
for the column LLV,
We know button at
row, column is pressed
If multiple Keys
are pressed,
ignore. otherwise Key pressed
is Key from
lookup table by row, col.

Question 3:

Question 4: Given some random number between 0 and 99, we can always guarantee success in at most 7 guesses. The strategy is a simple binary search algorithm. First, guess 50. If the random number was 50, then we are immediately successful. Elsewise, we are told whether we should go higher or lower. We then know the new bounds where the random number may lie are between 0 and 49, or 51 and 99. In either case, simply add the bounds together, then divide by two, and take the ceiling of this number. If the initial case were that the number were lower than 50, then the next guess would be 25=ceil(24.5=49/2). Supposing the secret number is less than 25, the new bounds are 0-24. If we continue with this approach we will always find the number with at most ceil(log_2(100)) comparisons, which is 7. This works because we can treat the possible values of the secret number as an ordered array from 0-99 for which we are searching for some value N_s, the secret number value.