COMP9032 Project Report – Clinic Management Kiosk

z5565813 Tianhao Luo
z5503515 Yang Zhang
z5526409 Runze Guo
z5563975 Wen Ma

Board setting

AVR Pins(top and bottom row)		**Input/Output Device Pins(middle row)**	
Port Group	**Pin**	**Port Group**	**Pin**
PORT F	PF0~7	LCD DATA	D0~7
PORT K	PK8	MOTOR	Mot
PORT D	RDX3,4	INPUTS	PB1,0
PORT A	PA4~7	LCD CTRL	BE, RW, E, RS
PORT C	PC0~7	LED BAR	LED0~7
PORT G	PG2~3	LED BAR	LED8~9
PORT L	PL0~7	KEYPAD	C3~0, R3~0
PORT B	PB1	MOTOR	LED

First Part: Product manual

Patients use the keypad while the doctor uses PB0 and PB1. After the lab board is powered on, “Next Patient:” is displayed on LCD the 1st line. Because there is no patient in the waiting queue, the LCD display “None” on the LCD 2nd line, PB0 and PB1 does nothing, only A on the keypad works.

Patients use keypad:

1.Patients need to press A before starting to enter their name.

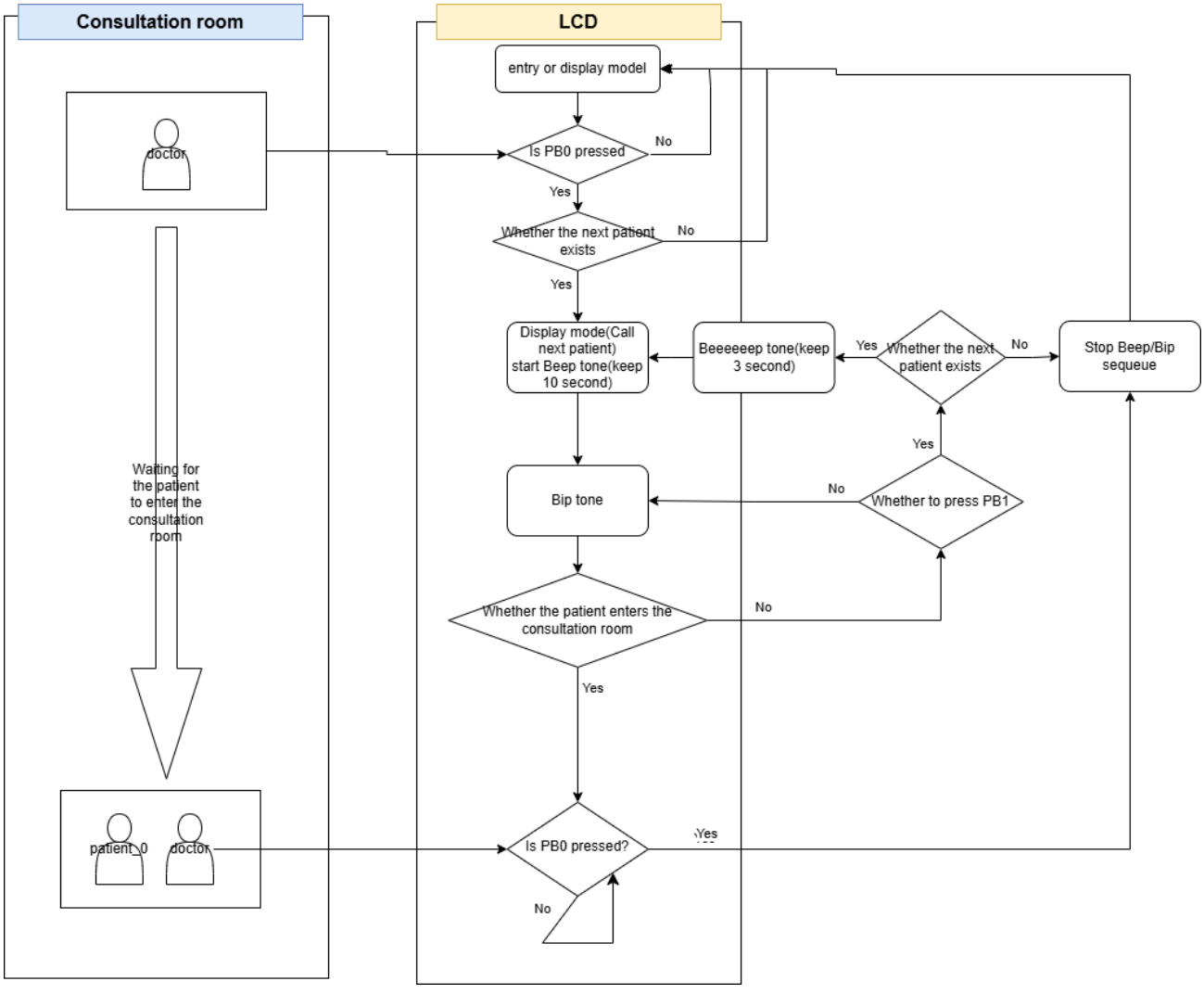
2.After pressing A, “Enter First Name:” is displayed on the LCD 1st line. Patient input displayed on the LCD 2nd line.

3.Patients can only press 2-9, B, C on the keypad before confirming their name with D.
4.The patient enters the name using a method similar to the nine-key input method on a cell phone. If a key is pressed once and then not pressed again for a short period of time, the character entered is confirmed and the cursor moves to the next one on the right.
5.B for backspace, C for clear the whole 2nd line
6.Patients can only input up to 15 characters.

7.After confirming with pressing D, “Your Name is:” is displayed on the LCD

1st line. Patient name and number are on the left and right of the LCD 2nd line. Patient number starts from 1. For each new patient, his patient number is the previous patient's plus one.
8. Patients press D again to confirm their numbers. Only D can be pressed in this state.

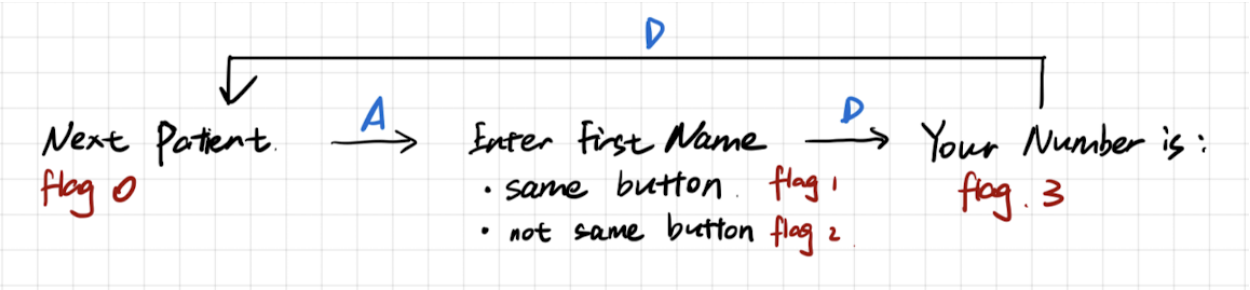
The Doctor uses PB0 and PB1:



PB0:
The doctor presses PB0 during non-Beep/Bip ringing periods to call the next patient into the consultation room. During the ringing period (indicating that the patient is waiting), press PB0 to turn off the ringing, indicating that the patient has arrived in the consultation room.
PB1:
After the doctor waits for the patient for a period of time (during the ringing period), if the patient does not come, he can press PB1 to switch to the next patient.
Tone:
Beep sounds for ten seconds, which represents the normal waiting time for the patient.
Bip is more rapid and loops infinitely, which means that the doctor has timed out waiting for the patient (exceeding the normal waiting time of ten seconds).
Beeeep is a continuous tone for 3 seconds. When the doctor presses PB1 beeps, indicating that the current patient's consultation number has been invalidated and the next patient will be switched.

Second Part: Appendices

Overall Process:



Depending on the state of the main program LCD when we return to the main program from interrupt, we divide the program into 4 parts. During setting the flag, use cli and sei to lock interrupt.
1. LCD display “Next Patient:” (Display mode)
2. LCD display “Enter First Name:” Loop characters with pressing the same button. (Entry mode)
3. LCD display “Enter First Name:” Confirm the character with pressing another button. (Entry mode)
4. LCD display “Your Number is:” (Entry mode)

Input Method:

The keypad input for the program uses a similar method as the demo code given in the lab. Every column and row is scanned to check if any key is pressed. If no key is

pressed, the program will check the column and row from the start repeatedly. If a key press is detected, the program will execute based on which key is pressed. The difficult part of the keypad input is how to implement the mobile phone character input. When button 2 – 9 is pressed, the LCD will display the first character corresponding to the key press, and store the key as a previous value. After that, the program will no longer scan the keypad unstopped, but a timer will be setup and the loop will discontinued after that amount of time. The timer is not implemented by the internal timer of the board, but rather just a simple counter that will jump out of the loop if reaches zero. If a key is pressed when the timer does not run out, it will be compared with the previous value. If it is equal to the previous value, the counter will be re-initialized, and the next character of that key will be displayed on the LCD at the same position. Otherwise, the key pressed will be ignored, and the timer will continue to count down. After the timer reaches zero, the cursor will move to the next left position, and the keypad will be scanned unstopped again. The characters for each key are stored in a 2D array called “key_map” (which is a string), with each subarray having a length of 4 bytes since each button can have a maximum of 4 characters. Thus, if we want to access the i-th subarray, key_map[i], we can go to the address corresponding to &key_map + 4*i. If the same button is pressed multiple times, a pressed button counter will store the times it is pressed. We can locate the character by finding which key is pressed and how many times that key is pressed. For example, if key “2” is pressed 2 times, we first locate the array corresponding to “2” which is key_map[0] (since character input starts from key “2”), and then we find the index by looking at the pressed time, which is 1 (in 0 indexed). Thus key_map[0][1] will be the character we want to find. If press button counter is more than the characters mapped to each key, we will take the mod value of the character mapped to that key. For example, if button “2” is pressed 4 times, which mean the button count will be 3 (starting from 0), and after taking mod 3, we will get 0 index thus allowing the character to be circulated.

Storage Method of patient names and patient numbers:

There are two spaces for storage names, buffer and string_arr

1. buffer is used during the patient input process [1,2]. (max 16 bytes)
    a. Move buffer pointer to do backspace (B), clear (C) process.

b. Each time a patient presses a key (including the same key situation and another key situation), store the current character in the buffer.

c. After finishing the input process (D) [3], copy buffer characters to string_arr and add a 0.

d. When an interrupt ends and it comes back to the input process [1,2], use the buffer to reload the current input on the LCD.

2. string_arr is used for all patients in queue, if overflow, store from index 0 again. (max 256 bytes)

a. After confirming with (D) [3], a new patient will be added to string_arr from the arr_tail pointer with a 0 end.

b. In display mode [0], the “Next Patient” is the name starting from the arr_head pointer to the first 0.

c. After calling a patient, arr_tail pointer moves to the start of the next name.

Each pointer (head and tail) also corresponds with an independent space for patients. (head_num and tail_num)

1. Patient numbers start from 1, head_num start from 1, tail_num start from 0.
2. Each num is constructed with 3 bytes, 1 byte for hundred, 1 byte for ten, 1 byte for unit. This makes it easier to display on the LCD.
3. In confirm process [3], tail_num + 1 then display tail_num
4. After calling the next patient, display head_num then head_num + 1
5. The number_plus_1 macro and the display_num macro are the logic that operates on numbers.

Display Method:

The character will normally be displayed from left to right and the cursor will move to the next right position after displaying one character. Delete is implemented by writing space from right to left 2 times and moving the cursor to left once. Clear line is implemented by moving the cursor to the leftmost position, then writing 16 spaces to the right and moving it to the leftmost position again. To circulate between characters mapped to the same key, after the character is displayed on the screen, the cursor is moved to the left.

Interrupt method:

All the operations from the doctor’s side like calling patients and canceling appointments are handled by interrupt, because we always need to handle the doctor’s operation first even if there is a patient currently using the keypad, and we need to restore the state and hand the control back to the patient whenever doctor’s operation is finished. PB0 is connected to INT0 and PB1 is connected to INT1, with both triggered with failing edge setup. If PB0 (“next patient button”) is pressed once and there are patients in the queue, the interrupt routine will be triggered and a state flag, named “pb0_toggle”, will be set from 0 to 1 indicating it is the first time we press the PB0

button. The first patient in the queue will be displayed on the LCD, whose name is obtained by traversing the string array using the head pointer and the id is obtained by looking at the head num. After that, we set the global interrupt to allow the PB0 button to be pressed second time, and go into the sequence of displaying different motor and LED patterns. The sequence will continue indefinitely if the state flag, "pb0_toggle", remains as 1 by checking the flag consistently during the sequence loop. If the PB0 is pressed second time for 1 second, another interrupt will be triggered and the state flag, "pb0_toggle" will be set to 0, indicating that the button is pressed second time. The interruption triggered by the second PB0 pressed will be returned immediately after that. Once the state flag is set to 0, the interruption triggered by the first PB0 pressed will stop the current LED and motor display and try to restore the state and give control back to the patient side. The state flag of the input and output section, "entry_flag", is checked. If it is in display mode, the interruption will hand the control back immediately after displaying the next patient's name and ID, by reading from the string array and the head num, on the LCD. If it is in entry mode, after displaying the next patient's info for 5 seconds, the current patient's name will be read from the buffer and displayed on LCD, and if an id is already assigned to the patient(after the patient enters the name and press "D" one time), the id will be retrieved by looking at the tail num. The PB1 (cancel appointment) button only activates after PB0 is pressed once and "pb0_toggle" is set to 1. If PB1 is pressed for 1 second, PB1's interrupt routine will be triggered, and "pb0_toggle" will be set to 2, indicating that the "cancel appointment" button is pressed. PB1's interrupt will return immediately afterward. After the "pb0_toggle" is set to 2, the sequence loop in the first PB0 interrupt will discontinued. If there are still patients in the queue, the head pointer in the string array will continue to move and the patient name and ID will be displayed in the LCD. The motor and LED sequence will be restarted as if PB0 is pressed one time and "pb0_toggle" is set to 1, indicating that the doctor cancels the current appointment and the next patient is getting called. If there are no patient in the queue, the PB0 interrupt will execute the routine that restores and gives control back to the patient, which is already explained previously. "pb0_toggle" is reset to 0 to allow the button to be re-pressed, and the interrupt stops afterward.

Timer method:

The timer will start by PB0 interrupt when the PB0 button is pressed two times. The timer uses a similar method as the timer code discussed in the lecture. A "TempCounter" that counts how many overflow, and a "SecondCounter" that counts how many seconds have passed (we count 1000 overflow as 1 second). "SecondCounter" is initialized to 20 every time the timer is triggered and decreases by 1 for every 1 second passed. In order to implement the LED light turn off one by one every two seconds, we

need to count how many 2 seconds have passed. A 0XFF mask is used to achieve the desired effect. If the first two seconds have passed, the mask will right shift once, and if the second two seconds have passed, it will right shift twice and vice versa. Since the mask can only control 8 lights, the remaining 2 are controlled manually in a similar manner. The timer will turn off itself when “SecondCounter” reaches zero.

Limitation:

Since we use 256 bytes static array to store the patient’s name, if the queue is too long and 256 bytes are all used up, the earliest patient’s name in the queue will be overwrite and result in error. If we assume the length of the name will be average 8 characters, the array can withstand up to 32 patients in the queue, and cannot correctly call the patient if patient in the queue exceed that limit.

Pseudo code

1. Definitions of variable, function, and constant

1.1. macro

Clear: clear two consecutive registers

num_plus_1: considering the three numbers (unit, ten, hundred) as a whole, this number + 1.

storeChar: store a char in buffer and buffer pointer + 1

display_num: display unit, ten, hundred of patient number from right of the 2nd line.

add_index_to_Z: store address + index in Z register

do_lcd_command: need 1 8 bits constant as parameter, execute that value as LCD command

do_lcd_immediate: need 1 8 bits constant as parameter, display the value as character in LCD

do_lcd_register: need 1 r16-r31 register as parameter, display the value in that register as character in LCD

clear_line_macro: clearing LCD 2nd line, and move cursor to the left of the 2nd line

clear_buffer_index: clearing buffer_index

1.2. Variable

(Counter)

tail_num_unit / *tail_num_ten* / *tail_num_hundred:* used to indicate the lastest ID for new incoming patient.

head_num_unit / *head_num_ten* / *head_num_hundred:* used to indicate the first patient waiting to see the doctor.

TempCounter: used in timer, count how many overflow interrupt is triggered.

SecondCounter: used in timer, a 20-second count down.

lightOn: used in timer, indicate how many LED lights are currently on

twoSecondCounter: used in timer, check if 2 seconds is passed.

(Address)

buffer_address: a 16 bytes space used to store the current patient's input in entry mode

string_arr_address: a 256 bytes space used to store all patients' names. The names are copied from the buffer and separated by 0.

(Pointer)

arr_head: init as 0, point to the first character of the first patient's name in the queue in the string_arr.

arr_tail: init as 0, point to the first position after the 0 symbol used to separate the last patient's name in the string_arr

buffer_index: init as 0, point to the next position after the last input character in the buffer.

(Flag)

entry_flag: used to indicate which state the LCD display are currently in, 0 as display mode, 1 - 3 as entry mode, where 1 indicates the patient is typing their name but not in the loop that checks if the same button is pressed, 2 indicates the patient is typing their name and is in the loop that checks if the same button is pressed, 3 indicate patient finished typing their name, pressed button 'D' once and is assigned an ID.

pb0_toggle: used in interrupt, 0 indicate PB0 is pressed first time, 1 indicate PB0 is pressed second time, 2 indicate PB1 is pressed once.

empty_flag: used in interrupt, 1 indicate that the patient queue is empty, 0 indicate the queue is not empty.

1.3. Constant

key_map: used in keypad input, store characters from "A" to "Z" as a 2D array.

2. Main function

2.1. input and output

Main Loop:

- While true:
  - set entry_flag to 0 (indicate it is in Display Mode for interrupt):
    - Display "Next Patient:" on LCD.
    - If patient queue is not empty:
      - Display next patient's name and number.
    - Else:
      - Display "None" on LCD.
    - Wait for keypad input:
      - If 'A' is pressed:
        - Call entry_mode().

Entry Mode (entry_mode()):

- Display "Enter First Name" on LCD.
- Set entry_flag to 1. (indicate it is in Entry Mode for interrupt)
- While true:
  - Wait for keypad input.
  - If 'B' is pressed (Delete):
    - Remove last character from buffer and LCD.
  - If 'C' is pressed (Clear Line):
    - Clear buffer and LCD's second line.
  - If 'D' is pressed (Confirm Input):
    - If buffer_index > 0:
      - Add name from buffer to patient queue.
      - Display "Your Number is:" and the patient number.
      - Increment tail patient number.
      - Set entry_flag to 3.
      - Wait for 'D' to confirm, then return to main loop.
  - If key '2' to '9' is pressed:
    - Map key presses to letters (handle multiple presses for same key).
    - Add character to buffer.
    - Display character on LCD.

2.2. interrupt

Interrupt Service Routine for INT0 (BUTTON_ISR_0):

- If pb0_toggle == 0 and patient queue is not empty: (“Next patient” button is pressed the first time”)
    - Display "Call Patient:" with patient's name and number on LCD.
    - Remove patient from queue (increment arr_head).
    - Set pb0_toggle to 1.
    - Enable global interrupt
    - Start LED and motor sequence

While true:

-if pb0_toggle == 1: continue sequence

- if pb0_toggle == 0: restore display mode or entry mod depends on the entry _flag, start timer, and exit
- if pb0_toggle == 2:

- if the patient queue is not empty:
    - cancel the current patient
    - call the next patient
    - Set pb0_toggle to 1
    - restart the loop

-else: restore display mode or entry mod depends on the entry_flag and exit

- Else if pb0_toggle == 1: (“Next patient is pressed second time”)
    - Set pb0_toggle to 0 and exit.
- Else if patient queue is empty:
    - Do nothing.

Interrupt Service Routine for INT1 (BUTTON_ISR_1):

- If pb0_toggle == 1 (if pb0 is pressed one time):
    - Play 3-second Beeeep sound.
    - Set pb0_toggle to 2 to restart the call sequence after Beeeep.

Timer0 Overflow Interrupt (Timer0OVF):

- Every 1 second:
    - Decrement consultation time counter (SecondCounter).
    - Update LEDs to indicate remaining time.
    - If consultation time reaches zero:
        - Disable timer.
        - Turn off LEDs.