Design manual for windows simulation

Author: Yuhang Yang

General information:

This program simulates the window control operations in an airplane. a window has four states or four opaque levels. For individual control, the number of '2' and '3' on the keypad is used to increase and decrease the window one's opaque level. The number of '5' and '6' on the keypad is used to increase and decrease the window two's opaque level. The number of '8' and '9' on the keypad is used to increase and decrease the window three's opaque level. The '0' and '#' on the keypad is used to increase and decrease the window four's opaque level.

For central control, the '7' on the keypad is used to set all windows to clear. And the '*' on the keypad is used to set all windows to dark.

It takes one second for a window to change from one state to another for local control and central control.

Push button is used as emergency status and set all windows as clear.

Function: Keypad Scan

The keypad on the board consists of two layers: a layer of horizontal lines which connected with 3 rows, and another layer of vertical lines which connected with 3 columns. In order to detect which key has been pressed, the program needs to scan columns from right to left. And for each column, the rows will be scanned from top to down.

To realize this function, the program selects the leftmost column as the first column and set the value related with this column as 0, and other columns set as 1. For each column, a row mask is used to detect which related row value is 0. The row mask is a 4-bit constant whose 1 bit will be set as 0 every time and other bits will be set as 1. The row mask will 'logical and' with the value of row pins. If the one row value is 0, the result of 'logical and' will be 0 as well. And this can figure out which row has been pressed.

There are two registers recording which column and row is scanning now. Because in this program, the control keypad just 3 columns. Every time when scanned the last column, the scan needs to back to first column.

If one pressed key is detected, the program will get into the function related with that key. This will be introduced later. For the local control, the program will continue to scan next row rather than return to scan the first row of the first column. This ensure the program can detect several keys are pressed at once.

Function: PWM Analog Output

To simulate the brightness, one function of this program is that outputs light of different levels on LEDs. This requires PWM analog output. Each window, in another words, every two LED bars is connected with two PWM signals in this design which are generated by one timer.

Timer1, timer3, timer4 and timer5 are used to generate PWM signals. And for one timer, OCnA and OCnB (n represents 1, 3, 4 or 5) are used. The timer mode is Phase Correct PWM mode, and no prescaling. The compare output mode is that clear OCnA and OCnB on compare match when up-counting, and set on compare match when down-counting. The PWM duty cycle will be introduced later.

Function: interrupt

Three interrupts are used to simulate central control and automatic emergency response. For these three interrupts, the mode is falling edge triggered interrupt. Because the emergency response has a highest priority, according to interrupt vectors, the external interrupt 0 pin is connected to a push button. When the button pressed, the simulation will set on emergency. For the central control, the interrupt 1 and interrupt 2 are used. To trigger the interrupt 1 and 2, the related pin will be set from 1 to 0 to create the falling edge when the related keys pressed.

Function: Timer

There are two timers used for controlling in this program. One timer is timer0 for the individual control, and another timer is timer2 for the central control. These timers are generally used to count time that a window changes from one state to another. The prescaling for the two timer is 1024. Therefore, 62 interrupts = 1,000,000μs/ (256 * 1024 / 16Mhz). That means every 62 timer interrupts is 1 second. Two 2-byte data memory are used to store the value of Counter for timer0 and timer2 to count time. Every time when the timer interrupt occurs, the counter will be increased by one. When the value of the counter match to 62, that is one second.

Function: LCD Display

There are two registers to communicate with LCD, one is Instruction register (IR), and another one is Data Register (DR).

Before the use of LCD, it should be initialized by software. When the power on, no data should be sent to LCD until Vcc reach to 4.5V. After that, it needs to check the busy flag. After the busy flag is ready, the commands can be sent to IR to set function. And the data can be also sent to LCD. Since the next instruction can be only written after the busy flag is set to 0, every time the code needs to check the busy flag and then send to the instruction. it needs to check the busy flag every time before the next instruction.

Design: Initialization

In the initialization part, the program set interrupts 0, 1 and 2 as falling edge triggered interrupt and enable interrupt 0. Then set the prescaling of timer0 and timer2 as 1024. After that, the program initializes the LCD and load the initial state on screen.

Then the all states of these four windows are set to 0, and these values are stored in a 4-byte data memory Wins. Four timers are used to generate PWM signals, and all PWM duty cycles are set to 0. Finally, the pins of columns are set as output, while those of rows are set as inputs.

Design: local control

The second column of keypad is set to increase the window opaque level and the third column for decreasing the level. Once any key of these two columns is pressed, the program will show 'L : ' on the LCD.

Each row controls one window. The key value will be set as row * 3 if the second column is pressed. The key value will be set as row * 3 + 15 if the third key is pressed. These values will be used to tell which column is pressed. If the value is greater than 10, it is decreasing function, otherwise increasing function.

Then it will read the status of the current window from 4-byte data memory Wins by adding the value of row to Wins address. For increasing, the value of status will be increased by one, and then it will be stored back to Wins. After that, the program will subtract the status value with -'0' to get the ASCII code, and then display the number on LCD. However, the maximal status value is 3. If the status value is equaled to 3, the program will not increase the value. That means this is not a valid key press and the program will do nothing but skip to scan next row.

To display the opaque level of each window at correct position, the DD RAM address will

be a value equal to 0xC4 add the key value. For decreasing, the process is similar, but the minimal value is 0. If the status value equals to 0, the program will not decrease the value anymore. Besides, the DD RAM address will be 0xB5 add the key value.

After that, the program will enable the timer0.

As the delay time of a window changes from one state to another is 1 second in this program, if the following valid key press is less than 1 second after last key press, the keypress time will be increased by one. The keypress time will be set as 0 only if the following valid keypress' time interval is greater than 1 second after last key press.

The value of row which is pressed every time will be stored in a chunk of data memory called Holder. This value is used for timer0 to know which key has been pressed one second ago. The storing address is the Holder address adds keypress time. In this case, during a continuing keypress (start with the first valid keypress and end with a following valid keypress' time interval is greater than 1 second after last key press.), the value of row will be stored in a continuing data memory address in sequence.

Another value will be stored is the match value of timer0. During a continuing keypress, every time of a valid keypress, a value equal to the current counter value adds 62 (delay another one second) will be stored in a data memory called Pointer. The storing address is the Pointer address adds keypress time. This value is used for timer0 to know what is the next match number. And then the program will also add another value of zero to the next address (the Pointer address adds keypress time adds one), to indicate it is the end of the match number this time.

The value of row and the match value of timer0 which are stored in the sequence of memory will be read by the timer0 later. Then the program will continue to scan next row.

At the meantime, during a continuing keypress, every time when the counter of timer match to a value from Pointer, a value called mark will be increased by one. Therefore, every time when the timer0 interrupt occurs, it will load the value from Pointer. To load the correct value form Pointer, the address will be the Pointer address adds the value of mark.

If the value of the counter dose not match to the value loaded from the Pointer, it will continue to increase the value of the counter, and then return from interrupt. Otherwise, that means it is an executing time. It will load the row value of the key has been pressed one second ago from Holder by adding the Holder address with the value of mark. According to this row value, it can tell which window has been changed one second ago (the first row controls the first window, the second row controls the second window, etc.). Then it will load the current status of that window from data memory by adding Wins address with the value of row to get correct address. After that, it will change the relative PWM duty cycle as 85 times the current states of this window. Hence, it can change the

bright level of LED bars of one window.

After that, the program will load the value of next executing time. If the value is zero, that means it is the end of one continuing keypress, which means the timer0 no need to increase counter to match another number. Therefore, the program will disable the timer0 and set the value of press and mark as 0. And continuing to scan the keypad. Otherwise, the timer0 will increase mark by one and continue to increase the value of counter to match another number load form Pointer.

Design: central control

If the keypress is '7' or '*' on the keypad, this is central control for setting all windows to dark or clear. For clearing windows, the program will enable the interrupt2. To trigger the interrupt, the program will create a falling edge by setting the output of the interrupt2 pin from 1 to 0.

During the interrupt2, the program will disable timer0 to overwrite the local control. Then the program will enable timer2. It will also change the display with 'C :' and store a value of 0 in a register. Then set all current states of all windows as 0. Then, it returns from the interrupt and jump to a waiting loop.

If the counter of timer2 match to 62, which is one second, the program will compare the value in the register with 0. If it is 0, then it is clear the window. The program will set all PWM duty cycle as 0, and set all status of window as 0. Then disable the timer2, and continue to scan keypad.

The central control of setting to dark has a similar process.

Design: emergency state

Every time press the push button, the interrupt0 will be triggered. During the interrupt 0, it will disable timer0 and timer2 to overwrite the local control and central control. It will change the display on LCD as '! ! : ' and change all the number of status as 0. After that the program will set all value in Wins as 0, and aet the counters of timer2 and timer0 as 0. Then it continues to scan keypad.