SOUND CONTROLLED STOPWATCH

ESE 566 - Hardware Software Co-Design of Embedded Systems

Project Members:

Anubhav Tomar (112268905)

Rahul P Suresh (112270636)

Shahzad Patel (112390824)

Stony Brook University

Department of Electrical and Computer Engineering

Contents

List of Figures

Algorithm and Design

The system is implemented by using Finite State Machines. All the states of the finite machine signifies the different modes of the system. The inputs of the Push button determines the change of mode and different actions within each mode. Toggling between the modes is implemented by long button presses ($> 1sec$). Within each mode different functionalities are selected by different short button presses ($< 1sec$).

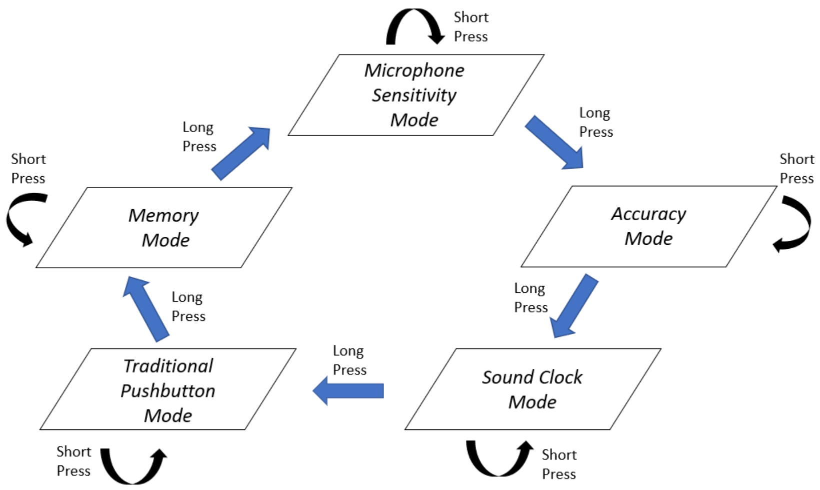


Figure 1: Finite State Machine of the System Modes

The different modes of the system are as follows:

1. **Microphone Sensitivity Mode:**
   This is the first mode of the system. In this mode, we use the background noise to calibrate the trigger threshold for when the sound mode is used. The system records the current "noise" level and when used in stopwatch mode, it will only start recording for levels which are twice the threshold value. The calibration mechanism works by recording sample for some time and then considering the average value as the threshold. A long button press causes the system to transition to Accuracy Mode.

2. **Accuracy Mode:**
   In this mode of the system we set the accuracy of the stopwatch from the given available accuracy options of 1 second, 1/2 seconds and 1/10 seconds. In this mode we have set the default accuracy of 1 second. A short button press will transition to the different accuracy options and a long button press will transition to Sound Clock Mode.

3. **Sound Clock Mode:**
   In this mode of the system the stopwatch is immune to background noise and starts only at the person's whistle blows detected by the microphone. The LCD will display $00:00:00$:($selected$ $accuracy$). Once the stopwatch is running, the systems takes a whistle or a short button press as

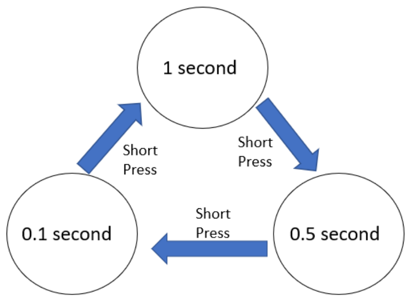


Figure 2: Finite State Machine within the Accuracy Mode

an input to stop the stopwatch. After this the system is programmed to display the clock time for 10 seconds and then transition back to its idle state where the user could either use the clock again or transition to the Traditional Push Button Mode by a long button press.

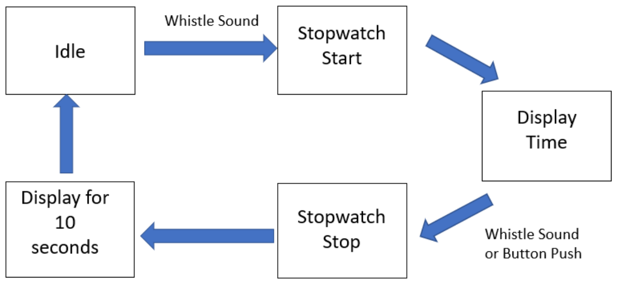


Figure 3: Finite State Machine within the Sound Clock Mode

4. **Traditional Push Button Mode:**
   In this mode of the system the stopwatch starts only on a short button press. The LCD will display $00:00:00$:($selected\ accuracy$). Once the stopwatch is running, the systems takes a short button press as an input to stop the stopwatch. After this the system is programmed to display the clock time for 10 seconds and then transition back to its idle state where the user could either use the clock again or transition to the Memory Mode by a long button press.

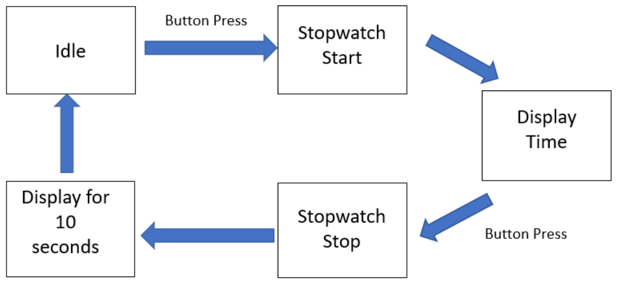


Figure 4: Finite State Machine within the Traditional Stopwatch Mode

5. **Memory Mode:**
   In memory mode, the system computes the average, shortest and longest duration of the time samples stored in the memory at run time. The values are displayed on the LCD. A long button press causes the system to transition back to the Microphone Sensitivity Mode.

Assembly Routines and Data Structure

We make use of two 32 bit Timers in the project. They are named as $stateTimer$ and SW. The $stateTimer$ is used to check and toggle between modes of the stopwatch, check for long press and short press. The SW is used to check for short press of the button to start and stop the display stopwatch clock. The following are the routines that are used to desgin the system.

1. $\_check$: This routine is executed whenever the input from the push button is low. It checks whether our $stateTimer$ is running or not using $stateTimerFlag$ which is set once we get a HIGH input from the push button. If it is running then it jumps to $\_checkLongPress$ routine.

2. $\_poll$: This routine is used to keep checking the state of the push button. If the input is HIGH then it starts the $stateTimer$ and sets $stateTimerFlag$ to HIGH.

3. $\_checkLongPress$: This routine is used by $\_check$ to stop the $stateTimer$ and compares the duration of the press. If the duration is > 1 second then it jumps to $\_goToNextState$ otherwise it jumps to $\_checkShortPressState$.

4. $\_checkShortPressState$: This routine is called whenever the push button is pressed for < 1 second. If the system is in the stopwatch made, then the short press will stop the clock by executing $\_checkSWState$ routine. And in the accuracy mode it is used to change the accuracy of the stopwatch.

5. $\_goToNextState$: This routine is used to toggle between the different states on long press based on the current state of the system.

6. $\_checkSWState$: This routine is used to check the current state of the SW timer. If the $SWFlag$ is HIGH then it executed $\_SWstop$. Otherwise it executes $\_SWstart$.

7. $\_SWstart$: This routine is used to start the SW Timer on detecting short press of the button and sets $SWFlag$ to HIGH.

8. $\_SWstop$: This routine is used to stop the SW Timer on detecting short press of the button. After stopping the timer it executes $\_delayDisplaySWTime$ routine and resets $SWFlag$ to LOW.

9. $\_delayDisplaySWTime$: It runs the stateTimer for 5 seconds wherein it displays the stopwatch time and revert back to initial state by clearing the hours, minutes, second and accuracy values on the LCD.

10. $\_setSensitivity$: This routine is used to read the digital signal from the ADC, consecutively for 4 times, to calibrate the threshold value of the sound input depending on the background noise. This threshold value is then stored in the memory as a comparison in sound mode.

11. $\_setAccuracy$: This routine is used to set the accuracy of the stopwatch to 1, 0.5 or 0.1 seconds.

12. $\_soundSW$: This routine is used to start or stop the stopwatch when the input sound signal received by the microphone is twice the threshold value.

13. $\_pushBtnSW$: This routine is used to display the current state of the timer./

14. $\_displayMemory$: This routine is used to compute and display the average, longest, and shortest times.

15. $\_SW\_ISR$: This is the interrupt service routine used by the SW timer block to run the stopwatch.

16. $\_stateTimer\_ISR$: This is the interrupt service routine used by the stateTimer digital block.

The following are the data structures used in the routine implementation

1. String: These are used to display the current mode of the system and the running clock.

2. Stack: We use the stack to set the period of the timer.

Interfacing Signals

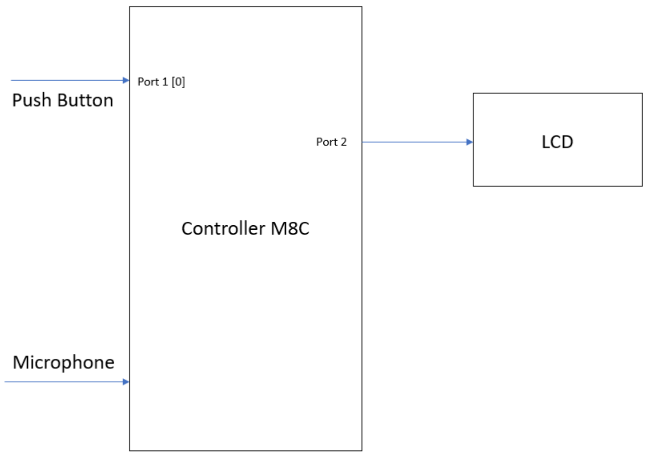


Figure 5: Interfacing Block Diagram

Input Interface:
SW input switch » P1[0], 0^{th} bit (LSB)

Output Interface:
LCD: For displaying the information in different modes of the system » P2
LEDs: For displaying current mode of the system. » P0 [1 ··· 4]
Microphone Sensitivity Mode P0 [0] » LED1
Accuracy Mode P0 [1] » LED2
Stopwatch Mode P0 [2] » LED3
Memory Mode P0 [2] » LED4

Computational Complexity

Number of Clock Cycles Per Routine:

_check: 17
_poll: 71
_checkLongPress: 45
_checkShortPress: 113
_goToNextState: 159
_checkSWState: 31
_SWStart: 43
_SWstop: 65
_delayDisplaySWTime: 488
_setSensitivity: 63
_setAccuracy: 105
_soundSW: 63
_pushBtnSW: 63 _displayMemory: 63
_SW_ISR: 425
_stateTimer_ISR: 98

Total number of routine instructions = 1814

Hardware Resources

Hardware Resoucres
The following is the snapshot of the configured global resources and the various analog and digital hardware programmable blocks are used in designing the system.

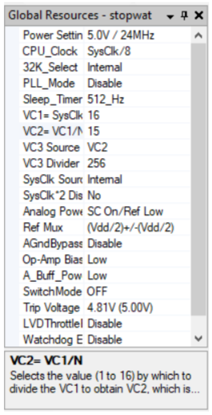
1. *32 bit timers*: Timer digital blocks are used to design an interrupt depending on the accuracy of the stopwatch. The minimum accuracy of the stopwatch in the system is 1 second.

$$Period = \frac{1}{24MHz * 16 * 15} = 99999$$

   For 99999 we need 32 bit timer.

2. *Low Pass Filter*: It uses the signal of the Microphone and removes all high frequency noises. We only allow the whistle sound as input to the ADC.

3. *Programmable Gain Amplifier*: To Amplify the output of Low Pass Filter before feeding it to ADC.



Global Resources - stopwat

Power Settin	5.0V / 24MHz
CPU_Clock	SysClk/8
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk	16
VC2= VC1/N	15
VC3 Source	VC2
VC3 Divider	256
SysClk Sourc	Internal
SysClk*2 Dis	No
Analog Powe	SC On/Ref Low
Ref Mux	(Vdd/2)+/-(Vdd/2)
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Pow	Low
SwitchMode	OFF
Trip Voltage	4.81V (5.00V)
LVDThrottleI	Disable
Watchdog E	Disable

VC2= VC1/N
Selects the value (1 to 16) by which to divide the VC1 to obtain VC2, which is...

Figure 6: Global Resoucres

4. *Analog to Digital Converter*: It samples the analog input to a digital output which is read by the PSoC.

Testing and Debugging

Procedure:
An LCD and LEDs were used to debug the program. Individual register values were displayed on the LCD. Partial code was executed to display the intermediate values. Boolean values were displayed as ’ON’ or ’OFF’ states of LEDs. Normal mobile phone stopwatch was used for verifying the output of our stopwatch. The inputs were based on the implemented functionality. Following are some of the possible test cases:

1. **Case 1**:
   1^{st} Long Press
   LCD Display: ’Microphone Sensitivity’
   LED1: ’ON’

2. **Case 2**:
   2^{nd} Long Press
   LCD Display: ’Accuracy Mode 1 sec’
   LED2: ’ON’

3. **Case 3**:
   1^{st} Short Press after 2^{nd} Long Press
   LCD Display: ’Accuracy Mode 0.5 sec’
   LED2: ’ON’

4. **Case 4**:
   3^{nd} Long Press
   LCD Display: ’Sound Mode’
   LED3: ’ON’

5. **Case 5**:
   Whistle Sound
   LCD Display: Stopwatch starts from ’00:00:00:00’
   LED3: ’ON’

6. **Case 6**:
   1^{st} Short Press after 3^{nd} Long Press
   LCD Display: Stopwatch starts from ’00:00:00:00’
   LED3: ’ON’

Waveforms

The following are the waveform plots in different test cases.

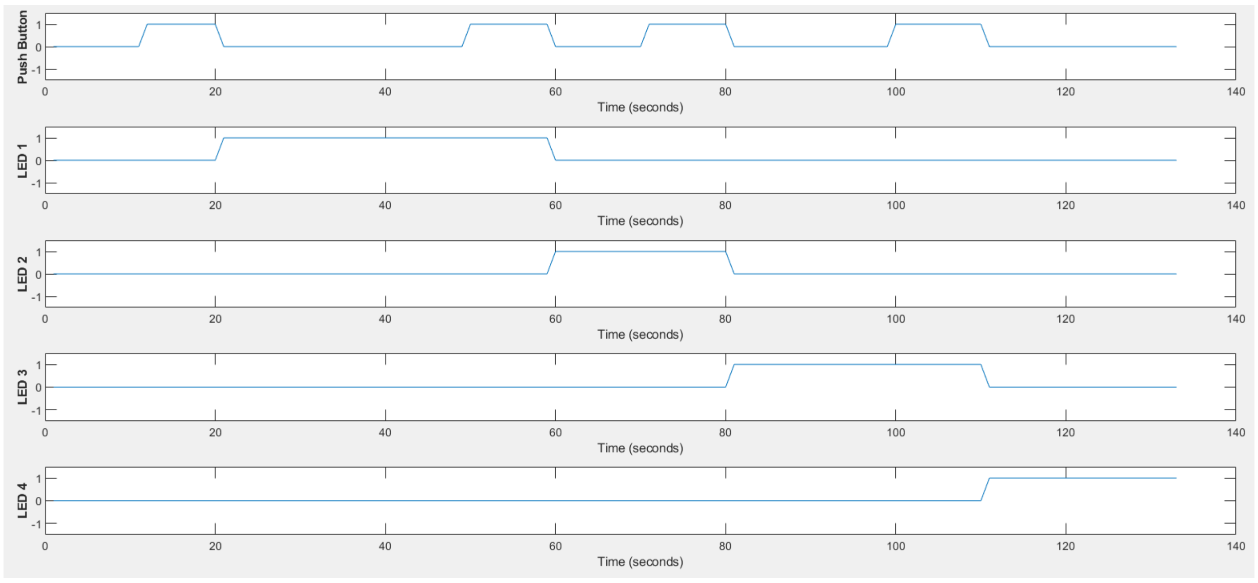


Figure 7: Wave-forms of Different Modes

Future Scope to Improve Design and Reduce Cost

1. A capacitor sensitive switch can be used in place of a mechanical switch as it would be more efficient and less prone to error.
2. We can add Lap functionality in the design to record the snapshot while stopwatch keep running.
3. We can minimize the hardware resources by using only 16 bit timers. For this we have to set VC3 as the clock source.

Bibliography and Web Site References

1. PSoC Designer Assembly Language User Guide:<http://www.cypress.com/file/72341/download>
2. Getting Started With PSoC 1:<http://www.cypress.com/file/45491/download>
3. PSoC Implementation of a Newspaper Vending Machine Controller:<http://www.cypress.com/file/71186/download>