SOUND-CONTROLLED STOPWATCH

Problem Statement:

The goal of the project is to implement the sound controlled stopwatch using the given hardware PSoC1. There are several modes that is represented using the finite state machine in the stopwatch which is controlled by the switches (long press of the in-built button) and the short press corresponds to the same state. The accuracy mode which is one of the functional description determines the precision of the stop watch. The memory mode is built in the system that saves the previous ‘x' recordings of the stop watch at different accuracy modes. The complete system control is handled by the C program that is being dumped into the PSoC board.

Functional flow and state machine transition:

The functional flow is handled by the finite state machines diagram. There are five different modes. This system works by taking the advantage of the system resources such clock, timers, ADC’s, filters mainly and to takes support of led's and LCDs. The toggling of the states depends on the type of button being pressed. The long press indicates to toggle to the another state and the short press indicates that toggling/ resetting to the same state

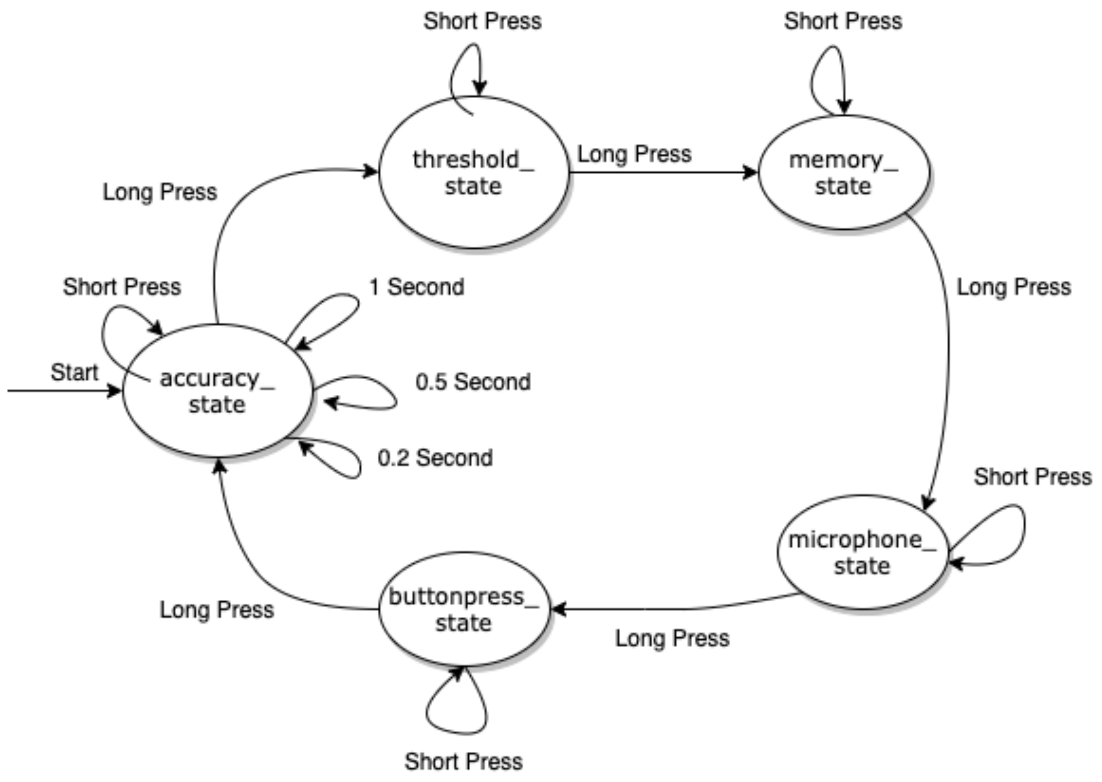


Fig 1. Finite State Machine

Information Processing in the system - The project works by taking advantage of various digital and analog components present in the system. We attach an external microphone sensor to the SOC which basically acts as the input to the system. The signal we have at this stage is very weak and hence we use an amplifier(Programmable Gain Amplifier) to amplify. Low pass filter is used to eliminate noise. In the further step, an ADC is used to get the digital samples of the signal. This sampled signal is used to control the stop watch with the help of a push button switch.The stopwatch is designed using a state machine and timers. Long press corresponds to the main state and short press corresponds to the sub state of the system and the system is explained in greater detail above.

Finite State Machines :

(a) Accuracy State :

This mode is the initial phase of all the process. This mode is further split into three sub modes for the precision of the timer. 1 second, 1/2 (0.5 second) and 1/10 (0.1 second). This mode is the deciding factor of how accurate the timer should run on push button mode (or) sound controlled mode displaying the timer in its output. If the precision is set for 1 Second, then the timer will display the approximate time HH:MM:SS, if the precision is set for 0.5 Sec, the LCD will display the approximate the time HH:MM:SS:MS which is approximated to (1/2) seconds and if the accuracy mode is set for (1/10) Sec, the LCD will display most appropriate time if the timer is stopped at any instance. Pressing the button for more time will switch the state from one mode to another if is represented in the Fig 1.

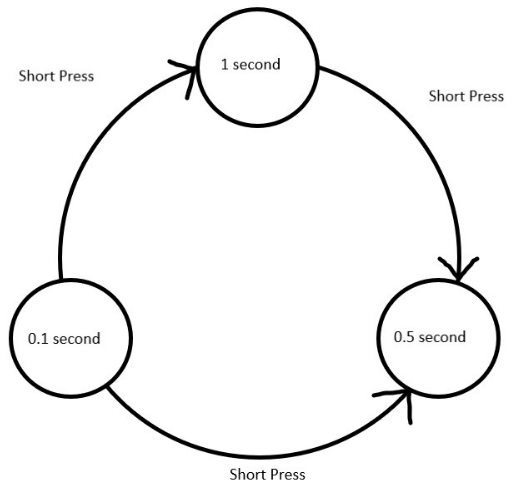


Fig 2. Accuracy modes state change

(b) Threshold State :

This state helps in determining the threshold for the sound(whistle) being recorded which acts as a trigger to start and stop the timer. The microphone senses the external noise as well, the Filter and amplifier plays a role in eliminating the external noise and makes the data available to control the timer. The mode when activated, sound disturbances(whistle) will makes the timer to start and stop. This operation is pretty much similar to the button-press mode as the external parameters to stop and starting of the timer differs.

(c) Memory State :

In memory state the system computes the average, shortest and longest values of the time intervals which are stored in the memory. The long press from this state will go to the next state i.e microphone state.

(d) Microphone State :

The basic intention of this state is to control the stop watch using the sound (example whistle). The stop watch is designed with a suitable threshold value making it immune to the background noise and responding to only sounds within the limits. The corresponding length of the duration is displayed on the LCD and long press in the state makes the system transition into the button press state.

(e) Button Press State :

The button will act as a trigger to start the PSoC timer and will run based on the accuracy mode being set. The timer for this button press is activated (start) on a short press and will be stopped for the next short press. The button presses will be counted on the subsequent switches of the states till the toggle_state (variable used in the code) reaches 5, if the toggle_state reaches 5 it gets shifted to the initial state and this way it works. The reading of each timer is stored in the memory which uses the memory mode to see those recordings.

In-built functionalities :

LCD_1_Start() : This function is used to initialise the LCD

LCD_Position(x, y) : This function is used position the display in the LCD display, x and y corresponds to co-ordinates

LCD_PrCString() : This function is used to print the characters to the LCD display

M8C_EnableIntMask : This is a macro that enables the interrupts masking through the configuration. There might be 2 inputs for these macros which depends on the applications that we will be using

PGA_1_Start() : The programmable Gain Amplifier, used to handle the weak signals thats been taken through the microphone

LPF2_3_Start() : There might be a chance of external noise during the threshold state, so in order to eliminate the external noise the Low pass filter is used.

DUALADC_Start() : This is to start the analog selection which helps to perform the

DUALADC_SetResolution(x) : This is to set the resolution to x bits

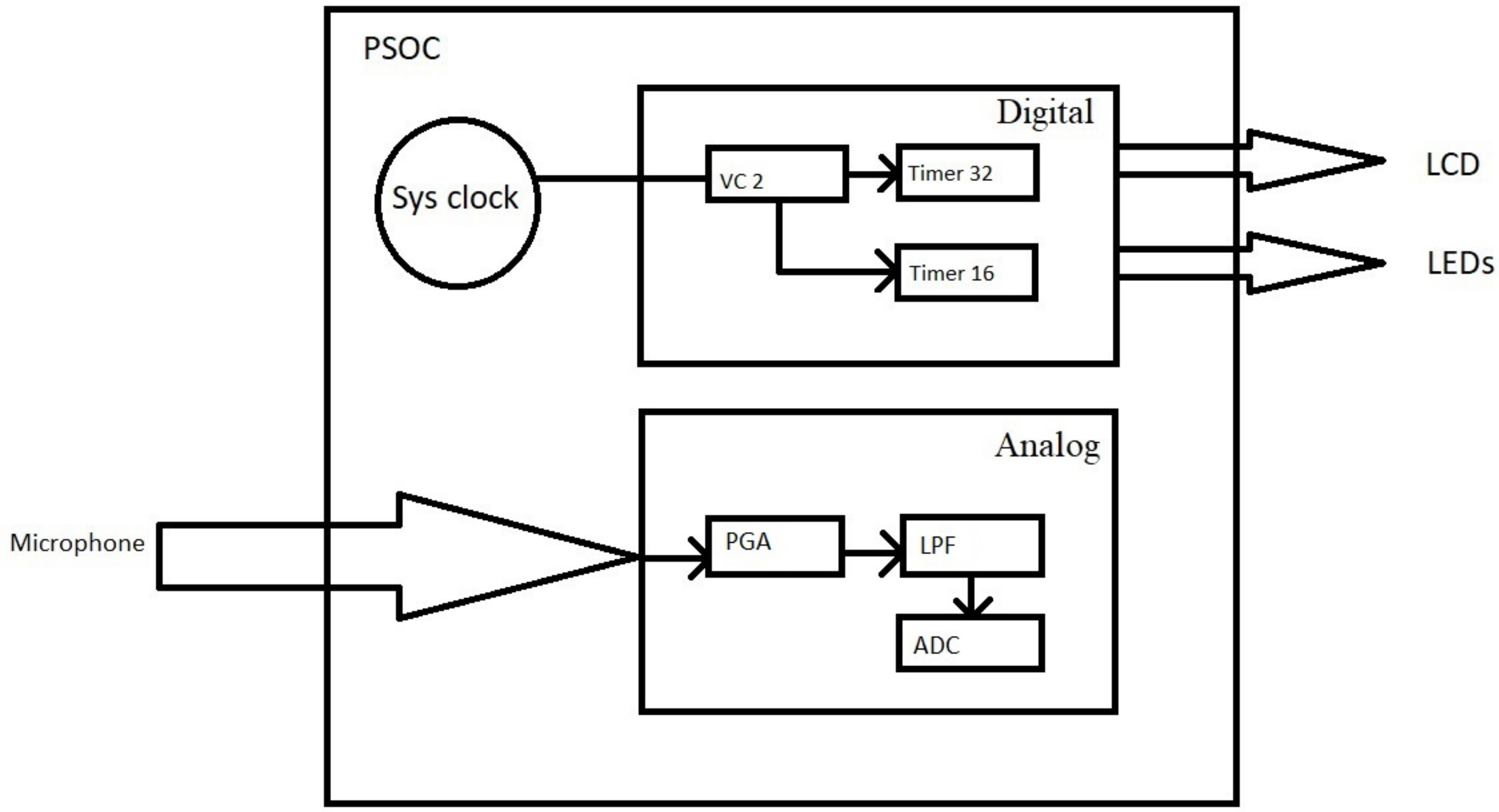
DUALADC_GetSamples() : This is to start ADC to read continuously

Timer16_1_Start() : To initialise the timer 16

Timer32_1_Start() : To initialise the timer 32

PRT0DR() **:** This is a data register used to control the external pins in the board, to be precise this corresponds to the P0[0], Port0

System Design :



PSoC system resources :

The OnChip oscillator generates 24 MHz frequency, using this calculation we achieved the "Period" to feed the Timers. Here, 2 numbers of Timer16 and 1 number of Timer32 is used

(i) Timer8 = 0 - 255

(ii) Timer16 = 0 - 65536

(iii) Timer32 = 0 - 4294967295

Computations for the "Period" :

(i) Vcc / SysClk = 5.0 V / 24 MHz

(ii) VC1 = SysClk / N = 24 / 16 = 1.5MHz (ADC)

(iii)VC2 = VC1/N = 1.5/10 = 0.1MHz = 100 KHz (We use this for the timers.

When applied to the below formula :

*Output Period = Source Clock Period * (PeriodRegisterValue + 1)*

(a) 1 Second precision = 10 ^(-5) * (PeriodRegisterValue + 1)

PeriodRegisterValue for 1 Second = 99999

(b) 1/2 Second precision = 49999

(c) 1/10 Second precision = 9999

The 8 bit timer cannot accommodate this output period as it can max support 255. Similarly 16 bit timer can be used for ½ and 1/10 precision but can accommodate the 1 sec precision. For this, we need higher period supporting timers. Hence we use 2 units of 16 bit timers and one 1 unit 32 bit timer, to get ½, 1/10 and 1 sec precision.

I/O Resources consumed :

(i) 1 Push Buttons
(ii) 8 pins (1 port) LCD
(iii) 4 LED
(iv) 1 Microphone (2 pins used; 1 - Input and 1- Ground)

Data structures and algorithms :

We used Array to collect the samples. In the process of determining the range of threshold, we sorted the values using efficient sorting algorithm to get the samples in an increasing order. The lowest 5 values average is set to lower boundary and average of last 5 values is used to set the higher boundary. The time complexity of the sorting algorithm is O(n log n)

Testing and Debugging :

The testing and debugging is done using the on board output resources i.e LEDs and LCD. Initially, we started programming the led to check basic functionality (for example- functioning of push button). Once we were sure about the button, we started to work with lcd and led in combination to make sure the the state machine transition is as intended. During the initial designing of the each state and substate, we had many number of lcd print informations . As we progressed and are confident about the state transitions, we reduced the print info's and using this mechanism, we were able to identify the progress and made sure it was in accordance with the problem statement

Challenges Faced :

(i) One of the greatest challenges that we faced was, there were 3 different chips and it was difficult for us to figure out which chip would be more comfortable for the SystemOnChip (SOC). So this made us to burn the program into the controller which consumed some days researching about the device catalog

(ii) As the PSoC designer is OS Compatibility (runs good with Windows), we(3 in a team) only one has Windows machine and it was difficult for us to implement the project within the given period of time. If we were provided the labs (with proper machines) it should have helped us to learn a lot at the initial phase

Functionalities :

(i) main() - This function determines which state to toggle based on the button press, there are totally 5 states and the main state will jump to its corresponding state function

(ii) accuracy_state() - The accuracy state determines the precision of the timer in seconds, there are 3 different modes. If any specific mode is set, the memory state will store the timer accordingly to the accuracy state

(iii) threshold_state() - This function determines the threshold of the sound level based on the number of recordings being done. The minimum is found and maximum is found, so if any whistle is made to trigger the timer, if checks whether the whistle units falls greater than the minimum and lesser than the maximum recorded value

(iv) button_press state() - This function starts and stops the timer based on the type of button presses. The timer which runs will be displayed in the LCD screen. For each short presses, the timer will start (or) stop

(v) microphone_state() - This is similar to the button_press state where the timer will be started and stopped based on the threshold_state. The whistle will initiate the timer and the next whistle will stop the timer

(vi) run_timer() - This is the inner function and its being branched inside the threshold_state() and microphone_state(), this function actually runs the timer.

Conclusion and future scope :

We were able to successfully implement the sound based stopwatch containing various modes of operation. Though the system runs perfectly with the switch button, the sound based control mechanism still throws challenges. The system can be further improved to consume less system resources and design in a lesser hardware budget and optimised software. In terms of hardware, we should have investigated ways to depend on using lesser timer units, depending only on the output resources. In terms of software, we should have come up with a lesser number of variables to control which would have consumed lesser memory of the system. We did not think of any parallelism in the system while implementing which would have reduced much runtime efficiency. Improving microphone sensitivity, optimising code in terms of runtime using better data structure and algorithms, consuming lesser hardware resources should be emphasised in the future work of our system.

It was a different learning experience for us to design hardware and software parallelly and help us understand real time complexities involved in designing an application. We would like to thank professor Doboli for providing us with enormous resources which helped us in getting started with the project.

References :

(i) https://www.cypress.com/documentation/user-module-datasheets/user-module-datasheet-16-bit-timer-datasheet-timer16-v-11

(ii) https://www.tutorialspoint.com/cprogramming/index.htm

(iii) https://www.cypress.com/products/psoc-1

(iv) https://www.cypress.com/documentation/development-kitsboards/psoc-1-kits

(v) https://en.wikipedia.org/wiki/Cypress_PSoC#:~:text=PSoC%20(programmable%20system%20on%20a,integrated%20analog%20and%20digital%20peripherals.