

EMK310 Home Practical Assignment

Am I Breathing?

FINAL REPORT

|  |  |
| --- | --- |
| Name: Mitchell Luke Williams | Student Number: u18013555 |

# 0. Introduction

The Peanut Butter Respiration Detection system utilizes several store-bought and home-sourced components to measure the breathing rate of a person and provide feedback on the rate of that breathing to potentially aide in the COVID-19 pandemic. The system is implemented using a PIC18F45K22, the lid of a peanut butter jar and standard electronic components like LEDS, resistors and capacitors. The breathing sensor is a simple affair constructed from a push-button switch and is activated every time the user inhales. The practical demonstrates the use of an ADC, table writing and reading in flash memory, sensor design, PWM control of LED outputs as well as the fundamental use of interrupts in a system such as this.

Disparate parts of the system are integrated together using a state machine with the three states of STARTUP, CALIBRATION and BREATHDETECT. The respective operations occur in these three states. These are namely the user initiation of the system, calibration of the sensor’s low and high voltage thresholds as well as the actual measuring of the breath rate and LED output.

The purpose of writing medical software in assembly is often because life-critical devices like pacemakers require the utmost efficiency and speed. Following in this vein, the system clock is set to 4MHz because although it is unnecessarily high for the ADC conversion the averaging of recorded breathes makes use of a flash memory table which can have a delay of up to 6ms when written to and so a high clock frequency is selected to combat any detrimental effects this may have.

The system accurately measures breath rate of the user and displays this to the user thus the practical is deemed a success. A broad overview of the entire system and visual confirmation of the working subsystems is present in the embedded video. The details of each individual subsystem follow below.

# Respiratory sensor

[Peer Review Criteria 1 - 4]

Describe and justify the design and implementation of your sensor in detail. Demonstrate the performance of the sensor and discuss.

The sensor of the system merely needs to differentiate between a user’s exhalation and inhalation. This equates to a HIGH and LOW voltage that need to be measured by the ADC against the low and high threshold voltages defined in the CALIBRATION state. The fundamental operation of the ADC is to convert analog voltages into a digital binary representation. Because the PIC outputs a steady current and the voltage is what is to be measured it follows from Ohm’s Law that the resistance of the load attached to the ADC need only be adjusted to construct a sensor as required.

Thus a simple push button switch is used to create large resistance to the current arriving at the ADC input port when not pressed and when it is pressed provide a close to V­­DD voltage value. These are the two LOW and HIGH voltage values that can be used by the firmware to count breathes and breathing rate.

The switch is thus placed between the positive and negative rails along with an RC circuit to debounce the switch and provide clean transitions between the HIGH and LOW voltage states. The switch is then attached to the belt that goes around the user’s waist for convenience and consistent functioning. The sensor subsystem’s circuit diagram is shown below with the respective component values required for an RC time period suitable to a sensor that can operate to more than 30 times per minute.

# Implementation of ADC

[Peer Review Criteria 5 - 7]

Describe the implementation of the ADC in detail and provide results.

## Description of implementation

The sensor sub-circuit is attached to port RA0 of the PIC and ADC conversions continuously convert the analog voltage provided by the sensor into a binary value for use in the firmware controlling the system. Left justification is used as there is a large difference between the HIGH and LOW voltage states and the two LSBs are thus unnecessary for delineation of an open or closed switch. The ADC references are set to VDD and V­SS as this is what the switch changes – a voltage either close to the VDD entering it or ground.

The acquisition time is set to 2TAD and the A/D conversion clock to in order to allow for the RC debouncing circuit to discharge properly and measure a voltage change when the switch is pressed. The ADC conversion takes place in the BREATHDETECT state where it is continuously repeated and when a conversion is completed an interrupt occurs and all the ADC handling code is run in and called from the interrupt service routine.

This converted voltage value is compared to the afore-mentioned low and high threshold voltages upon which it is decided whether the current voltage is a LOW or HIGH one and calls a LOW or HIGH subroutine accordingly.

## Evidence of correct functioning

# Calibration

[Peer Review Criteria 8 -10]

Describe the implementation of the calibration in detail and provide results.

## Description of implementation

The calibration of the sensor subsystem is very simple since the only part of it that the user interacts with is the push button. Physically, the belt that the switch is attached to must be tightened enough so that the user’s stomach contacts the switch and pushes it in when they fill their lungs and their diaphragm expands.

In the firmware, the CALIBRATION state is a linear state that runs once upon initiation of the system and sets up the ADC as it is used for the breath detection. An ADC conversion is done immediately but the result discarded as this first value is often a junk value. A time delay of around 1.2s (30 iterations x 0.04 seconds) then occurs before the ADC value of the open switch voltage is measured and has 25 subtracted from it in order to get the HIGH threshold value that a signal must exceed in order to be classified as a HIGH value.

The 25 value is chosen due to variations in measured open switch voltages (noise) identified in testing and to account for these. LED number 6 is lit to show the user they must now press the switch button in. The same time delay occurs, after which the voltage is measured again and this time 25 is added to the measured value and the result stored as the LOW threshold value that a signal must be less than in order to be classed as a LOW value.

The calibration is now complete and the state machine transitions to the BREATHDETECT state. The calibration was necessary because the LOW and HIGH threshold voltage values could change dramatically depending on the type of wire and component impedance present in the RC debouncing circuit.

## Evidence of correct functioning

# Respiration rate detection

[Peer Review Criteria 11 - 14]

Describe the implementation of the respiration rate detection in detail and provide results.

## Description of implementation

Every time the ADC finishes a conversion an interrupt occurs and calls the ADCSR subroutine. In this, the signal is classified as either LOW or HIGH compared to the thresholds set in the CALIBRATION routine and the SRLOW or SRHIGH subroutine called. These two routines control the counting and timing of successive breaths by the user. When the program first enters SRLOW it is checked if it has reached SRHIGH yet – if it has not the timer is started if it hasn’t been already.

Timer4 is used for this application and has both its pre-scaler and post-scaler set to 16 to stretch over the maximum time duration. At a FOSC of 4MHz the time it takes for one timer interrupt to occur is thus .

If the program is in SRLOW and has reached SRHIGH one whole breath has occurred and so the timer is switched off. Every time the timer rolled over an interrupt occurred and incremented an interrupt counter TIMERINTERRUPTS. Now the amount of interrupts that have occurred for this one breath is written to flash memory using the TBLWT\*+ command. The counter NUMBREATHSCOUNTED is checked to see if eight breaths have occurred yet – if they have not, the entire process above is repeated, starting with the ADC conversion.

If eight breaths have occurred the SRAVERAGEBREATHS subroutine is called. This is where the calculations are performed to determine the average breath rate. Because all the table writes above were short writes the long write to flash memory must be initiated and so this is done – storing the eight counts of timer interrupts into a table in the flash memory starting at address 0x1000.

The program then reads out these values from the table using the TableLoop loop and the TBLRD\*+ command. As it reads them out the program adds them all together in the two registers TOTALINTERRUPTSLOW and TOTALINTERRUPTSHIGH – representing a 16 bit number. The flash memory is then erased so the address space can be used again for the next eight breaths.

Now that the total amount of timer interrupts for eight breaths has been counted the program divides both TOTALINTERRUPTSLOW and TOTALINTERRUPTSHIGH by 8 by rotating the individual bits right three times (23 = 8) and applying masks to the registers to clear the rotated-in values. The final value of TOTALINTERRUPTSLOW is placed in INTERRUPTSPERBREATH.

We can disregard TOTALINTERRUPTSHIGH now because any value larger than the maximum value of TOTALINTERRUPTSLOW (256 x each interrupt representing 0.065536s = 16s) means the user is taking 16s to take one breath and therefore must be dead already and has larger problems to deal with than the workings of The Peanut Butter Respiration Detection system. Finally, the PICLEDS subroutine is called to determine which LEDs to turn on and represent the breathing rate of the user now present in INTERRUPTSPERBREATH in terms of timer interrupts. All values are cleared at the end of this subroutine so that the next eight breaths can be counted and averaged in the same way.

## Evidence of correct functioning

# Breathing difficulty alert

[Peer Review Criterium 15]

Not implemented. However, if it was to be, the difference between successive average breath rates could be ignored and instead, simply check the value of TOTALINTERRUPTSHIGH and if it is anything larger than 0, trigger an LED that represents a breathing difficulty alert.

# LED output

[Peer Review Criteria 16 - 18]

Describe the implementation of the LED driving algorithm. Show and discuss results obtained.

## Description of implementation

The PICLEDS subroutine handles all the code required to set the PIC’s onboard LEDs based on what the calculated breath rate is – or rather, the average number of timer interrupts occurring each breath. Following from this, the algorithm uses a fall-through approach to compare the value of INTERRUPTSPERBREATH to that computed beforehand for the different categories of breath. These categories are:

* Less than 12 breaths per minute = 0 LEDs on
* 12-16 breaths per minute = 1 LED on
* 16-20 breaths per minute = 2 LEDs on
* 20-30 breaths per minute = 3 LEDs on
* Above 30 breaths per minute = 4 LEDs on

## Evidence of correct functioning

# PWM output

[Peer Review Criteria 19 - 21]

Describe the implementation of the PWM in detail. Provide and discuss results.

## Description of implementation

## Evidence of correct functioning

# Interrupts

[Peer Review Criteria 22 - 24]

Describe the implementation of the two different interrupts in detail. Provide and discuss evidence of at least three successive events for both interrupts.

## Description of implementation

## Evidence of correct functioning

# Touch start/stop

[Peer Review Criteria 25 - 27]

Describe the implementation of the touch start/stop sensor and algorithm in detail. Provide results and discuss results.

## Description of implementation

## Evidence of correct functioning

# Integration

[Peer Review Criterium 28]

Describe the integration of the system and provide a functional block diagram. Provide evidence that successful integration occurred.

# Conclusion

Provide a brief overview of the strengths and shortcomings of the system. Use the evidence in the report to state whether the system has been implemented successfully. Also mention future work that can be done to improve the system.

# EEPROM

[Peer Review Criteria 30 - 31]

Not applicable.