Heart Rate sensor and Pulse Oximeter

Introduction:

The project that will be discussed in the report, is an embedded project that I did for a client on Fiverr. The premise of the project is to create a heart rate sensor and a pulse oximeter (detects the oxygen level in the blood), displays this information in an LCD, and sends it to a smartphone via a bluetooth module.

Concept:

When blood enters tissue vessels it carries hemoglobin which carries oxygen with it, that gives blood and red blood cells their red colour. Hemoglobin is found in two forms. The first form the hemoglobin carries oxygen (HbO2), and the second one is without oxygen (Hb). Oxygen Saturation levels (SpO2) can be calculated from these 2 values.

HR and Pulse Oximeters take advantage of how hemoglobin reflects and absorbs light. Hb absorbs red light more than HbO2. HbO2 absorbs infrared light more than Hb. Both LEDs are needed to measure the oxygen level.

To measure the heart rate, when the heart pumps blood through the body, the volume of caterpillars and the blood vessels inside the tissue expand (HbO2 is present and not Hb), and when it is not pumping the they retain their original volume (Hb is present and not HbO2). Only the red LED is needed to detect the heart rate.

The heart rate and pulse oximeter sensor works based on this concept, it has two LEDs, a red LED and an infrared LED, and an The LEDs emit light into the tissue, and IR LEDs detect the reflected light, according to the internal circuitry of the IC, it detects

Overview:

I am using an MSP430 development board (MSP430FR6989) to communicate with a MAX30102 HR and SPO2 sensor from MAXIM.

The MAXIM chip have two LEDs one is red and the other is infrared, emits light into the tissue which is controlled internally by the IC LED driver, the reflected light is detected by an photodiode which converts it into an electrical signal that goes to an internal ADC, and stores it in the IC memory (FIFO).

The MPU interfaces with IC via I2C communication protocol, the MPU configures the control registers of the IC as per the requirement of the application, and reads samples from the FIFO.

The data from the MAXIM IC, are ADC output values that needs to be filtered and HR and SpO2 values calculated before displaying it, various DSP and filtering algorithms needs to implemented in the MPU, and we need to ,

When we have the correct HR and SpO2 values, the MPU shall send these data to a 2x16 Dot-Matrix LCD. The MPU has a parallel interface with the LCD.

Also, the MPU sends the data to a bluetooth module, which displays it onto a smartphone. The Bluetooth module is an HC05 module, the MPU interfaces with it via UART.

I used the MSP430 platform, because of its ultra low power consumption, if the prototype is to be embedded in a PCB circuit, and operated from a battery then it is better to have a microcontroller that doesn’t consume a lot of power.

MAXIM30102:

As explained in the section above, the MAX30102 has two LEDs, one red and the other is infrared, the reflected signal is detected by the photodiode. The photodiode detects the reflected signal of each LED and outputs it into the IC's internal ADC.

The ADC has a resolution of 18 bits, it converts the signal from the IR LED first, stores it in a memory location of size 3 bytes (24 bits), converts the next signal from the RED LED, and stores it in the next 3 bytes.

These 2 ADC output values are one sample, which is stored in the FIFO (memory) of the IC. The FIFO can store 32 samples (each consists of IR and RED LED values), and has a width of 3 bytes.

The ADC value is stored in the FIFO memory of the IC. The FIFO can hold 32 samples, each sample has a width of 6 bytes, 3 bytes for the IR LED channel and 3 bytes for the red LED channel

I programmed and configured the I2C module of the MSP430, and created an I2C library that has all the functions relevant to I2C communication (start, stop, repeated start, read, and write) of the MSP430.

To interfacing with the MAX30102 consists of initializing and resetting the chip, setting up the MAX30102 by writing to the control registers (resolution of the ADC, which mode to use HR or SpO2,). It configures the IC through typical I2C communication, first the MPU starts the transmission, sends the address of the IC, along with a write bit (command), when the IC acknowledges it, it sends the control register address,

I created 2 32 bit wide arrays of 100 elements, one stores 100 infrared LED samples from the chip, and the other to store 100 red LED samples.

After initializing and setting the MAXIM chip, the MPU starts collecting 100 samples (red and infrared LEDs). Inside the function that reads fifo data.

The width of the data transmitted and/or received by the I2C is 8 bits, hence when we read data from the MAXIM chip, we can only read 8 bits of data, where we need to read a 32 bit data, to get around this, I created an 8 bit integer array of 6 elements, elements 0, 1, and 2 store the infrared LED sample, and sample 3, 4, and 5 store the red LED sample.

Through bit arithmetics, I store the values of the arrays into the red and move to the next element.

When the MPU gets the initial 100 samples of red LED and infrared LED, it gets the average value of the heart rate and SpO2.

Then it goes into a loop that loops forever, where it deletes the oldest 25 samples (element 0 to 24), shifts samples of both arrays (elements 25 to 99 occupies 0 to 74), gets new 25 samples and stores them in elements 75 to 99.

Filter the signal and get the average of the 100 samples, repeat the process again.

Code for MAX30102, *Main code*, *MAX30102*, *MAX30102 datasheet*

Algorithm:

When the MPU gets the data from the MAXIM chip, it has raw ADC output data of the infrared LED and the red LED stored in 32 bit wide integer variables.

We get values of IR and Red integer variables in the range of 50,000+, we need to remove this offset, when we remove the DC signal, we are left with the AC part.

Even after having an AC signal, it is still hard to detect the peaks of the signal, hence we need to “clean” the signal a bit more by using a Mean Signal Filter, and passing it through a Low Pass Filter.

Calculating the heart rate (Beats Per Minute) we get the time the current peak occurs, and the time of the previous preak and apply this equation: $BMP = 60000 / (CurrentTimeStamp - PreviousTimeStamp)$

To calculate the oxygen level (SpO2), it is the ratio of oxygenated hemoglobin to total hemoglobin:
$SpO2 = HbO2 / (Hb + HbO2)$

As mentioned in the overview since these two forms of hemoglobin affect how red and infrared waves are absorbed we can use this formula to find the ratio: $R = (ACrmsred)/(ACrmsir)$

AC RMS Red is the RMS value of the Red LED, and AC RMS IR is the RMS value of the IR LED.

SpO2 is calculated using this formula: $SpO2 = 110 - 25 \times R$

LCD:

The LCD is a 16x2 Dot-Matrix, it consists of 80 segments (80 dots, with 8 columns x 5 rows), that consist of 16 characters and 2 lines. The MSP430 uses a parallel interface with the LCD, 11 lines, 8 for data (DB0 to DB7) and 3 for control: Enable (E), Read/Write (R/W), and Register Select (RS), in addition to the power lines.

The working concept of the LCD is when I want to display something, I clear the Enable pin, wait for a certain period of time as specified by the datasheet, map the character (a letter or an instruction) into the DB0 to DB7 pins, and select the suitable register with the RS pin, either to to write to an instruction register, or to a data register.

To help me with the code I created an LCD library that has all the necessary functions to display a string into the LCD such as write data, write command, delay, and display string.

Code for LCD

HC05 Bluetooth Module:

The HC05 interfaces with the MSP430 via UART, it has 2 pins (Receive and Transmit) and operates in 3.3V, and a baud rate of 9600.

To interface with HC05, I configured one of MSP430 UART modules baud rate to 9600, data of size 8 bits, and no parity. I created a UART library to help with programming the interface, it consists of initializing the UART to have the same Baud Rate (9600) as the HC05, sending data, and displaying strings.

In order for the HC05 to communicate with the smartphone (or a PC), I downloaded an app called “Bluetooth Terminal HC-05”, paired the device with the HC05 and started receiving messages from the MSP430.

The code consists of configuring the UART module (peripheral pins, baud rate), sending a string through the UART, waiting for one second and sending the next string.

Code for HC05

Conclusion:

The project allowed me to use my C programming and embedded skills for medical applications, which was more challenging than an. The new skill sets I learned from the project was learning about I2C communication, UART, and improving my C programming skills, such as passing arrays to functions using pointers.