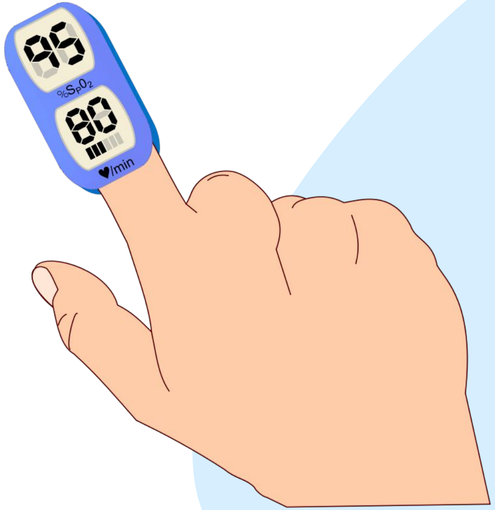
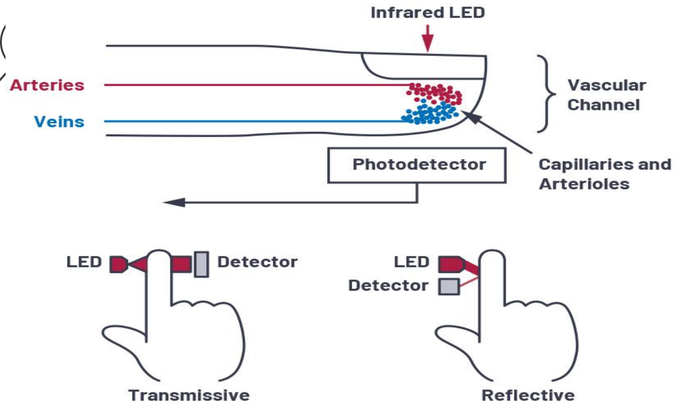
Developed by Cole J, Dhayalan B, Luke T, Nicholas K

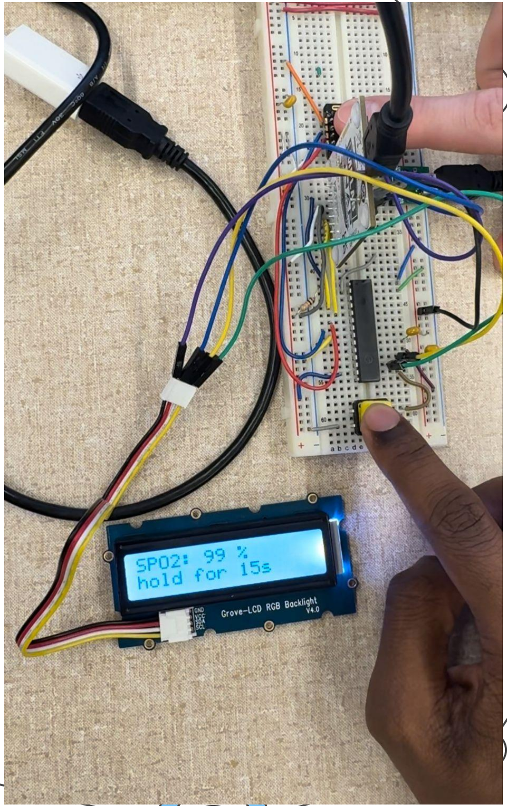


Blood Oxygen Monitor

Using MAX30102 sensor, Grove-LCD-backlight display, and a PIC24 microcontroller

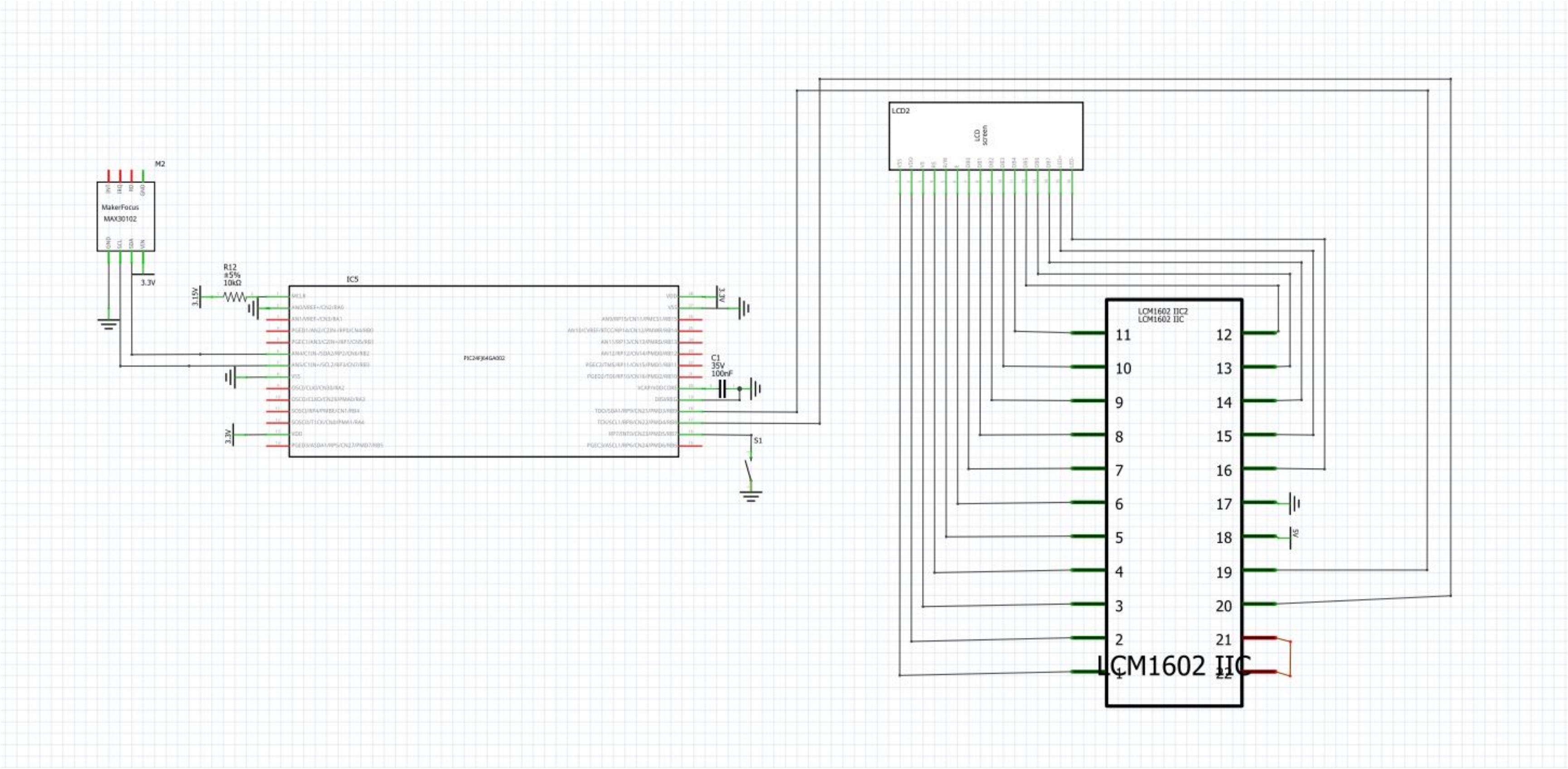


Purpose

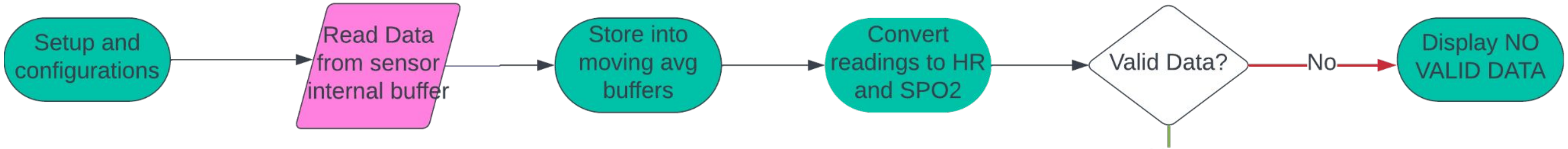


- Health Monitoring: The project aims to provide individuals with a convenient and accessible means of monitoring their vital signs, specifically blood oxygen saturation (SpO2) and heart rate.
- Real-Time Feedback
- Promoting Awareness: With increasing interest in personal health monitoring, this project fosters awareness of vital signs.
- How does it work? The MAX30102 combined two LEDS, a photodetector, Optimized optics, and signal processing to to detect SPO2 and HR. The sensor uses photodiodes to measure the light reflected by the blood represented by analog sensors.

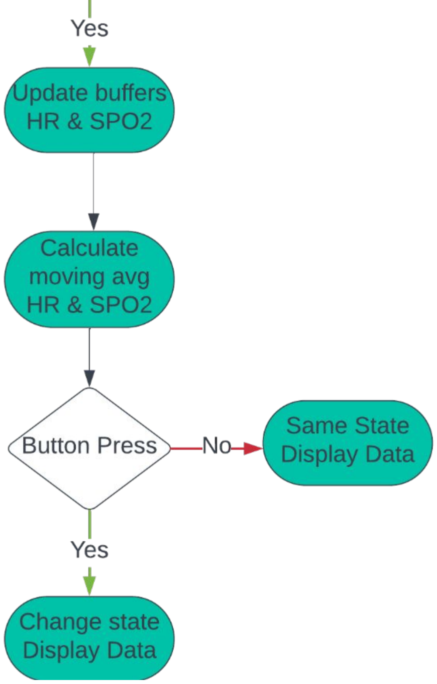
Monitor Schematic

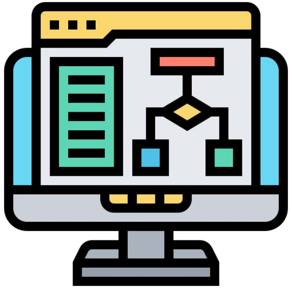


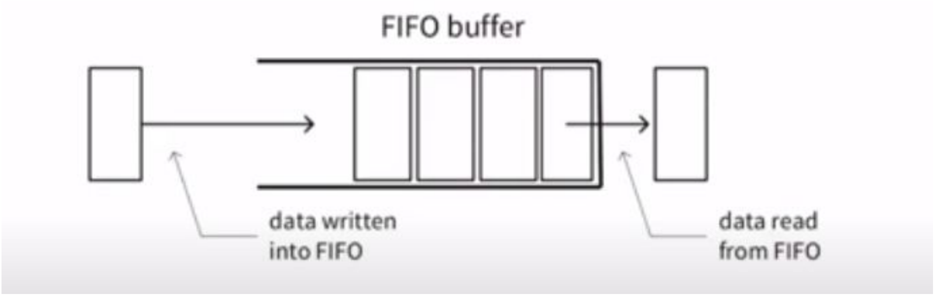
Block Diagram/Algorithm



Setup and configurations
Read Data from sensor internal buffer
Store into moving avg buffers
Convert readings to HR and SPO2
Valid Data?
No
Display NO VALID DATA



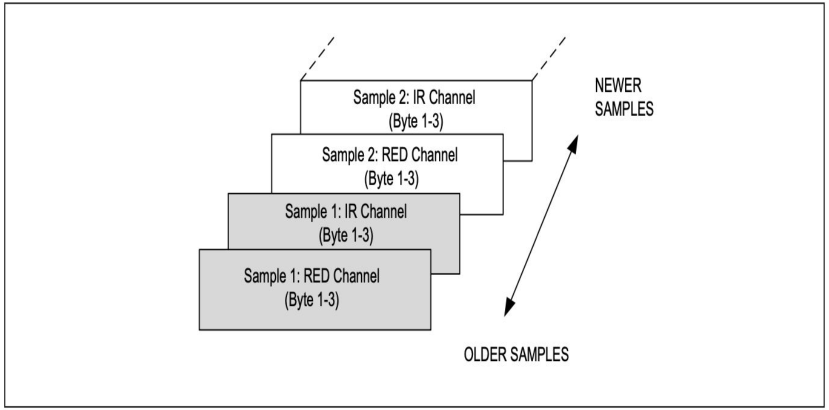




MAX30102 Library

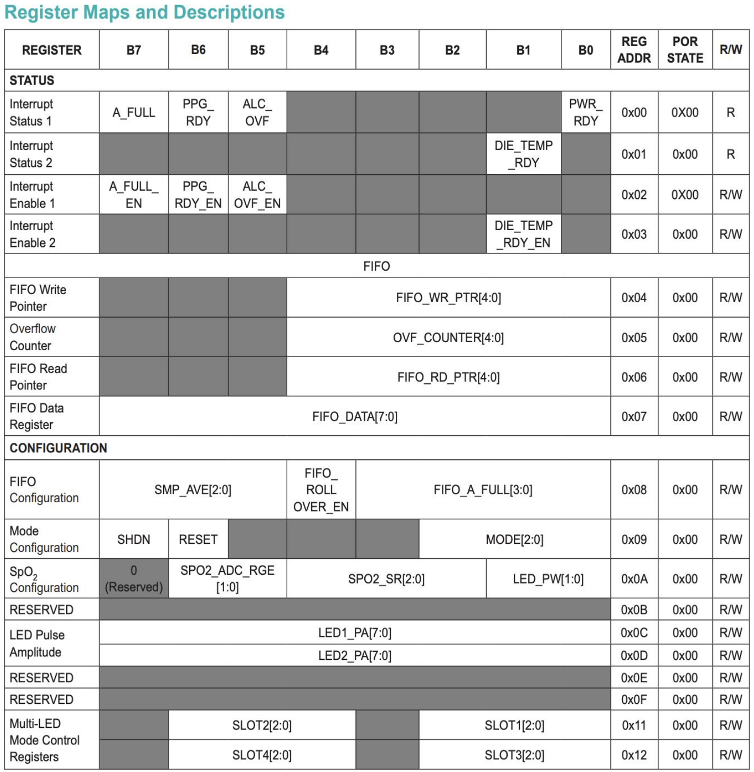
Consists of:

- Configuration.
- Reading in data from the FIFO buffer.
- Processing the data into a readable spo2 and heart rate values.
    - Spo2 algorithm: peak detections in PPG cycle/calculations(Arduino)



Sample 2: IR Channel (Byte 1-3)
Sample 2: RED Channel (Byte 1-3)
Sample 1: IR Channel (Byte 1-3)
Sample 1: RED Channel (Byte 1-3)
NEWER SAMPLES
OLDER SAMPLES

Figure 2. Graphical Representation of the FIFO Data Register. It shows IR and Red in SpO$_2$ Mode.

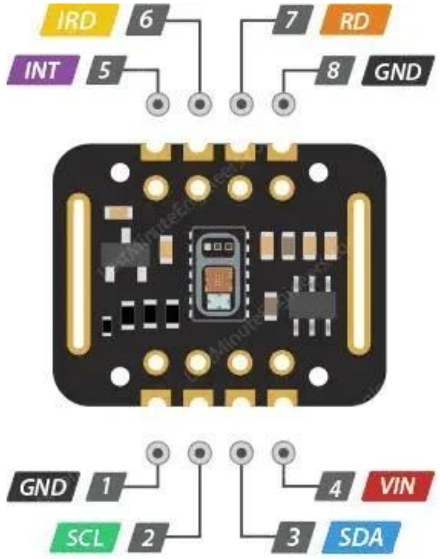


Register Maps and Descriptions

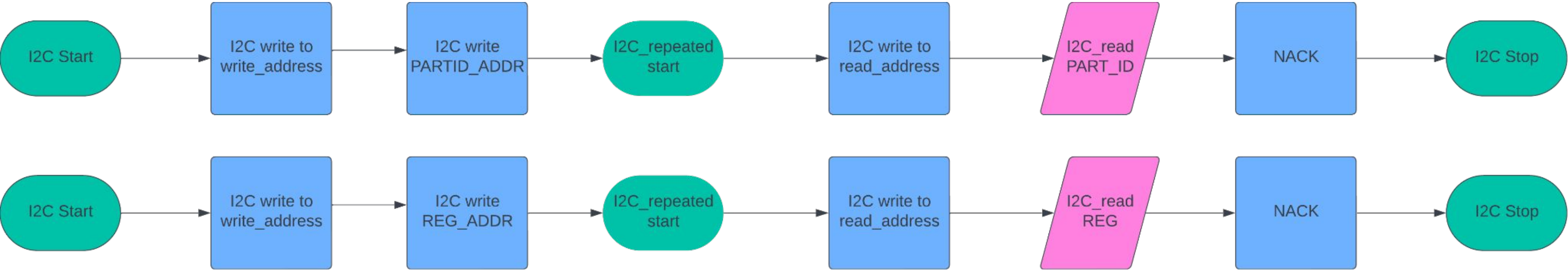
REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
STATUS											
Interrupt Status 1	A_FULL	PPG_RDY	ALC_OVF					PWR_RDY	0x00	0X00	R
Interrupt Status 2							DIE_TEMP_RDY		0x01	0x00	R
Interrupt Enable 1	A_FULL_EN	PPG_RDY_EN	ALC_OVF_EN						0x02	0X00	R/W
Interrupt Enable 2							DIE_TEMP_RDY_EN		0x03	0x00	R/W
FIFO											
FIFO Write Pointer				FIFO_WR_PTR[4:0]					0x04	0x00	R/W
Overflow Counter				OVF_COUNTER[4:0]					0x05	0x00	R/W
FIFO Read Pointer				FIFO_RD_PTR[4:0]					0x06	0x00	R/W
FIFO Data Register	FIFO_DATA[7:0]								0x07	0x00	R/W
CONFIGURATION											
FIFO Configuration	SMP_AVE[2:0]			FIFO_ROLL OVER_EN	FIFO_A_FULL[3:0]				0x08	0x00	R/W
Mode Configuration	SHDN	RESET				MODE[2:0]			0x09	0x00	R/W
SpO$_2$ Configuration	0 (Reserved)	SPO2_ADC_RGE [1:0]		SPO2_SR[2:0]			LED_PW[1:0]		0x0A	0x00	R/W
RESERVED									0x0B	0x00	R/W
LED Pulse Amplitude	LED1_PA[7:0]								0x0C	0x00	R/W
	LED2_PA[7:0]								0x0D	0x00	R/W
RESERVED									0x0E	0x00	R/W
RESERVED									0x0F	0x00	R/W
Multi-LED Mode Control Registers		SLOT2[2:0]				SLOT1[2:0]			0x11	0x00	R/W
		SLOT4[2:0]				SLOT3[2:0]			0x12	0x00	R/W

MAX30102 and the PIC24 Micro

- Uses I2C communication
- Setting up configurations of SPO2, FIFO, MODE… etc.
  - Used bitmask to config the sensor
- Uses buffer to store readings from the sensor
  - RED and IR LED buffers
- Algorithm to read data from the FIFO (implemented by the group)
- Utilizes Arduino SPO2 Algorithm to process data
- Shortfalls:
  - 4 extra pins not used
  - Choosing perfect configuration
  - Using NACK and ACK correctly
  - Reading Data from FIFO
  - Heart Rate not entirely accurate due to sensor



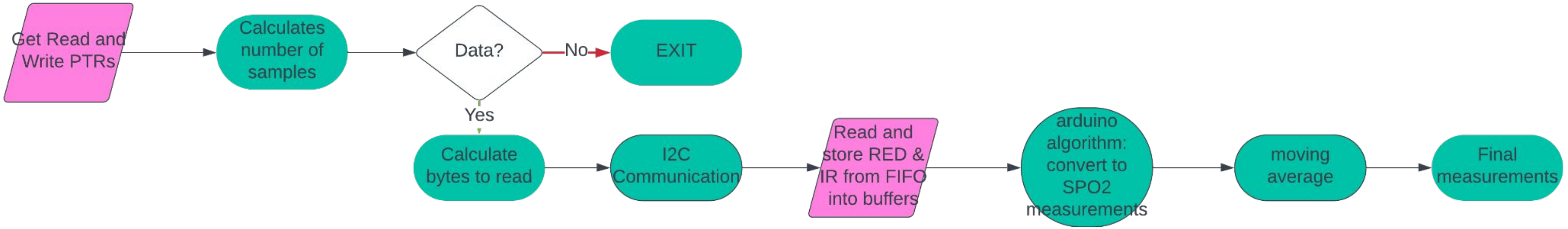
MAX30102 Algorithms



I2C Start → I2C write to write_address → I2C write PARTID_ADDR → I2C_repeated start → I2C write to read_address → I2C_read PART_ID → NACK → I2C Stop

I2C Start → I2C write to write_address → I2C write REG_ADDR → I2C_repeated start → I2C write to read_address → I2C_read REG → NACK → I2C Stop

Reading data algorithm:

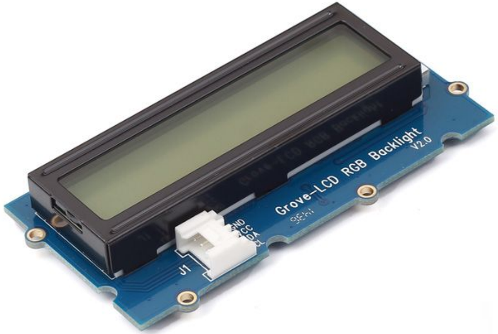


Get Read and Write PTRs → Calculates number of samples → Data?

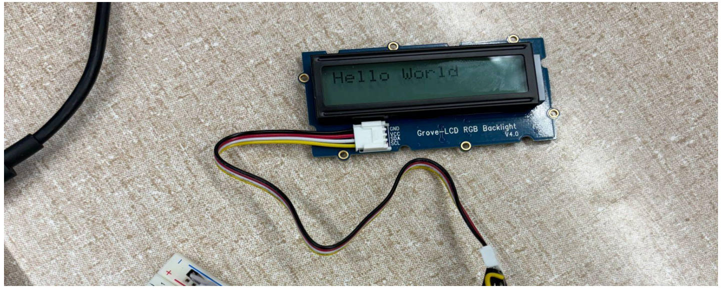
No → EXIT

Yes → Calculate bytes to read → I2C Communication → Read and store RED & IR from FIFO into buffers → arduino algorithm: convert to SPO2 measurements → moving average → Final measurements

Grove – LCD RGB Backlight

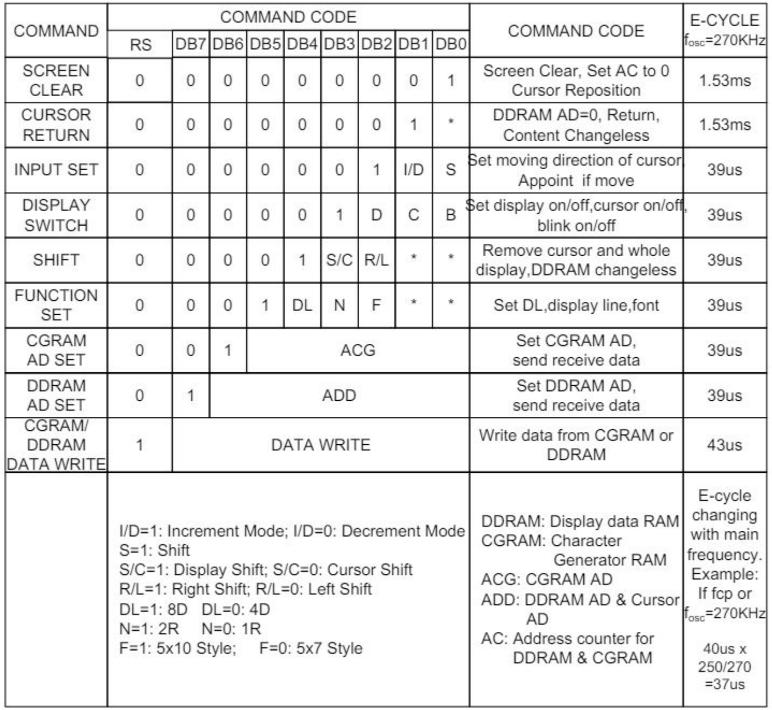


- Why the Grove?
  - 16x2 Display
  - RGB backlight – more fun with color right?!
  - I2C ⟶ Only 2 pins
- Trials and Tribulations
  - Datasheet – not much help, however lab5 and arduino clutched
  - Delayed Update Time – Interfered with sensor
  - Power Resets – Sticky screen!?
  - Lack of space ⟶ Button!



LCD/RGB Library

- #include
  - “xc.h”, <string.h>, “rgb_lcd_display.h”
- Functions
  - init_I2C1() – baud rate, enable I2C
  - grovergb_init() – initialize display - datasheet
  - setRGB(r,g,b) – set desired color value 0-255
  - Helper functions: clear, home, display(on/off), blink(on/off), setColorAll/White/Red, setCursor, printChar, printStr



COMMAND	COMMAND CODE									COMMAND CODE	E-CYCLE f_{osc}=270KHz
	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
SCREEN CLEAR	0	0	0	0	0	0	0	0	1	Screen Clear, Set AC to 0 Cursor Reposition	1.53ms
CURSOR RETURN	0	0	0	0	0	0	0	1	*	DDRAM AD=0, Return, Content Changeless	1.53ms
INPUT SET	0	0	0	0	0	0	1	I/D	S	Set moving direction of cursor, Appoint if move	39us
DISPLAY SWITCH	0	0	0	0	0	1	D	C	B	Set display on/off,cursor on/off, blink on/off	39us
SHIFT	0	0	0	0	1	S/C	R/L	*	*	Remove cursor and whole display,DDRAM changeless	39us
FUNCTION SET	0	0	0	1	DL	N	F	*	*	Set DL,display line,font	39us
CGRAM AD SET	0	0	1	ACG						Set CGRAM AD, send receive data	39us
DDRAM AD SET	0	1	ADD							Set DDRAM AD, send receive data	39us
CGRAM/ DDRAM DATA WRITE	1	DATA WRITE								Write data from CGRAM or DDRAM	43us
	I/D=1: Increment Mode; I/D=0: Decrement Mode S=1: Shift S/C=1: Display Shift; S/C=0: Cursor Shift R/L=1: Right Shift; R/L=0: Left Shift DL=1: 8D DL=0: 4D N=1: 2R N=0: 1R F=1: 5x10 Style; F=0: 5x7 Style									DDRAM: Display data RAM CGRAM: Character Generator RAM ACG: CGRAM AD ADD: DDRAM AD & Cursor AD AC: Address counter for DDRAM & CGRAM	E-cycle changing with main frequency. Example: If fcp or f_{osc}=270KHz 40us x 250/270 =37us

Switching via Button

-Simple Button used to switch between displaying Blood Oxygen and Heart Rate

-Initialization: Sets button and Timer 2 interrupts

-Input Capture used to detect button press

-Switch states using bool “displayHeartRate”

-Get function getButtonState for usage in other files

Contributions/ Bigger Brains

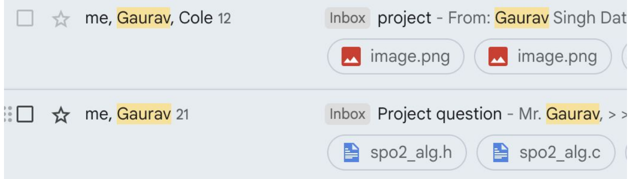
Dhayalan - wiring, max30102 library(configurations, I2C, reading algorithms/data processing, integration of all libraries), and testing.

Luke – Button, LCD, Moving Average, documentation, good vibes, ect.

Nicholas - Sensor Configs, PART_ID/Read Alg, I2C comm, documentation.

Cole – LCD text and color functionality/code, documentation, etc.

The majority of our learning came through a much deeper understanding of I2C and how to use other devices with the PIC24.

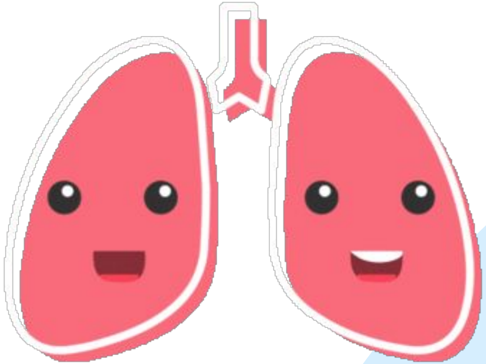


□ ☆ me, Gaurav, Cole 12 Inbox project - From: Gaurav Singh Dat

image.png image.png

□ ☆ me, Gaurav 21 Inbox Project question - Mr. Gaurav, > >

spo2_alg.h spo2_alg.c



THANK YOU!

Any Questions?