LABORATORY

Microcontrollers

LAB 5

NAME	KELVIN MAKAKA
MATRICULATION NUMBER	26219
STUDY COURSE	MECHATRONIC SYSTEMS ENGINEERING
PRESENTATION DATE	15.01.2021
SUBMISSION DATE	20.01.2021

Task 1.

- Key 1 (Button Read): PB0
- Key 2 (Button Write): PB1
- LCD Display add-on
- EEPROM add-on
- RTC add-on
- ADMUX = (1 << REFS0)|(1 << REFS1)|(1 << MUX3); used to enable temperature sensor in the microcontroller with 1.1v as reference.
- ADC pre-scale of 64 is selected.
- Key 1 shows the stored data and the program prints Output End if all the data has been displayed
- Key2 records the data .

CHANGES

- Numbering orientation used to represent byte slots in the EEPROM memory array: slot/position 0 = byte 0, position 1 = byte 1……
- Each memory array has 8 bits.

log_data function

- Used to write data into the memory array.
- save_value8bit(count,0); // count value written in position 0 in the EEPROM memory array : byte 0
- save_value8bit(year,1); // year written in position 1 in the EEPROM memory array : byte 1

byte 2 :

- temp integer used to hold data temporarily.
- month value which is takes 4 bits is to be saved in the right most nibble of byte 2. This is achieved by using the bitwise operator << (bitwise left shift) . It is shifted 4 times to the left side in byte 2 and the other remaining 4 bits are used to store the value for day
- *(day>>1)* – day with values 1 -31 occupy 5bits (31_{10}=11111_2) . Only 4 bits of memory space is left on the current memory slot thus the bits for day are shifted once to the right so that only the 4 left most bits are saved on the current byte.
- *temp = (month<<4) | (day>>1);* the bitwise or is used to store the two bytes together into one byte so as to ensure efficient use of memory.

byte 3:

- temp is used again to store data temporarily
- *(1&day)* – Bitwise & is used here to extract/mask the LSB of the byte with the day value. Example (00000001 & 00011111) = 00000001 – only the 1 in red is extracted. This ensures that bit that was shifted out in byte 2 above is recovered.
- *((1&day) << 7)* – the extracted bit is stored as the MSB in byte 3 buy it 7 times to the left. Example (00000001 << 7)= 10000000.
- The value for hour is stored in the remaining 7 bits.
- *temp = ((1&day) << 7) | (hour);* the bitwise or is used to store the two bytes together into one byte so as to ensure efficient use of memory.

show_data function

- The function is used to read data from the memory and display on the LCD.
- Here the reverse of what was done in the *log_data* function is done.
- temp1 and temp2 is introduced to temporarily hold data.
- uint8_t temp1 = load_value8bit(2); //reads from byte 2 stores in temp1

- month = (temp1 >> 4); month is read from the 4 most significant bits of byte 2/temp1.
- temp2 = load_value8bit(3); // reads from byte 3 and saves as temp2
- *day = ((15 & temp1) << 1) | (temp2 >> 7);* extracts the least significant nibble of temp1 (byte 2) and the MSB of temp2 (byte 3), combines the bits and saves as day.
- *hour = (127 & temp2);* // reads the last 7bits of from temp2(byte 3) and saves as hour

void nexttime(void) function

- The jump from one recording to the next is 10 seconds.
- In this function we start by adding 10 seconds to the real time so that when the logged data is displayed the time for following consecutive recordings is shown to be at intervals for 10 seconds from the starting time of the first recording.