Embedded Firmware Engineer Assignment Report – Q2) C Program in an Online Compiler

Name: Pranav R
Role Applied: Embedded Firmware Engineer

Question 2:

1. Use timer to simulate data generated by external sensor. Setup the timer to trigger every second and generates random number (0 to 5) of random bytes and adds this to a globally accessible data structure.

2. Separately wake up periodically (every 10s), checks if 50 bytes are stored in the globally accessible data structure and prints only the latest 50 bytes (in hex value) and deletes the printed bytes from the data structure.

3. For example, 1st second 4 bytes are added, 2nd second 3 bytes are added and 10th second there are 39 bytes are in the buffer. Thus at 10th second data is not printed. At 20th second, if there are more than 50 bytes in the buffer, the main thread only prints the latest 50 bytes and deletes them

The above was the problem statement for which I have come up with a solution with which I shall explain it in this report as the following.

Understanding of the Problem Statement:

From the problem statement, what I understood is that, generate sensor data(random data) for every second and append it in a ‘buffer’. In our source-code the ‘buffer’ acts as an Array. Then, a separate task checks on every 10 seconds time interval, that, if there are at least 50 bytes in the buffer. If so then it should print it in ‘Hex’ format. If there are fewer than 50 bytes, it does nothing and waits for the next 10 seconds interval. Before getting into the implementations, let’s discuss on the source-code.

Source-Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<unistd.h>
#define MAX_BUFFER 1000
unsigned char buffer[MAX_BUFFER];
int buffer_index=0;
void GenerateSensorData()
{
```

```
        int num_bytes= rand()%6;
        for( int i=0; i<num_bytes &&
buffer_index<MAX_BUFFER; i++){
                buffer[buffer_index++]= rand()%256;
                }
        }
void CheckAndPrintBuffer()
{
        If(buffer_index<=50){
                printf("Printing the latest 50 bytes: \n");
                for(int i= buffer_index-50;
i<buffer_index;i++){
                        printf("%02X", buffer[i]);
                }
printf("\n");
buffer_index-=50;
}
}
int main(){
        srand(time(0));
        for(int sec=1; sec<=30; sec++){
                GenerateSensorData();
```

```
            if(sec%10==0){
                CheckAndPrintBuffer();
            }
        sleep(1);
        }
return 0;
}
```

Now, let's breakdown what is our approach and how the code works.

Code Explanation:

1. We first determine what are all the necessary header files we will be requiring. From the question it is clear that based on "time interval" so basically we shall use - **#include<time.h>**
   Since we are expected to print the sensor values randomly, we will be using,
   #include<stdlib.h>
   and **#include<unistd.h>** for **sleep()** function.
2. After determining the header files, we have defined a Macro named "MAX_BUFFER".

3. Now we will be declaring a **unsigned char variable** named "**buffer"** which can hold the total number of bytes.
4. Now to keep track of how many bytes are currently stored in the buffer, we create a variable named **buffer_index**. Every time new data is added, this value increases.

5. Now let's separate the code into 2 separate functions. The first function is
   i) **GenerateSensorData()**
   ii) **CheckAndPrintBuffer()**

6. **GenerateSensorData():** This particular part in the code,
   "int num_bytes= rand()%6;"
   This randomly generates bytes between 0 to 5 and stores the value in **"num_bytes".**
   ->%6 gives results from 0 to 5.

7. **" for (int i= 0; i<num_bytes && buffer_size<MAX_BUFFER; i++){**
   buffer[buffer_index++]= rand()%256;
   }"

   This particular line of code explains, that, this 'For' loop runs 'num_bytes' times, unless the buffer is already full.

Each time it generates a random byte between 0 and 255 and stores it in the buffer at **“buffer_index”**. Then it increments.

8. As we have programmed the GenerateSensorData() function, now let’s program,
   CheckAndPrintBuffer()
   This line of code,
   “ void CheckAndPrintBuffer(){
   if (buffer_index>=50){
   printf(“The latest 50 bytes are: \n”);
   for(int i= buffer_index – 50;
   i<buffer_index; i++){
   printf(“%02X”, buffer[i]);
   }
   printf(“\n”);
   buffer_index-= 50;
   }”

   This function checks if the buffer has atleast 50 bytes only then it proceeds to print them.
   “%02X” -> represents the Hex form of the output, with 2 digits. After printing, it removes those 50 bytes logically.

9. Now let's build our main() function.

"int main(){
srand(time(0));
for(int sec=1; sec<=30; sec++){
GenerateSensorData();
if(sec%10==0){
CheckAndPrintBuffer();
}
sleep(1);
}
return 0;
}"

Here the srand() seeds the random number generator using the current time. To ensure that, rand() gives different results everytime you run the program.

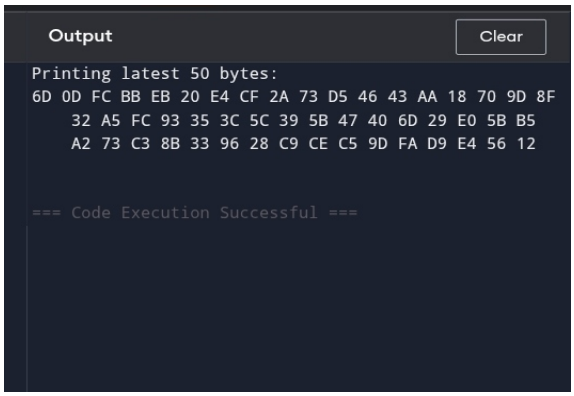
The loop runs for 30 seconds and for every 10 seconds, it calls the function to check if there are 50 bytes to print and remove. This is the code explanation.

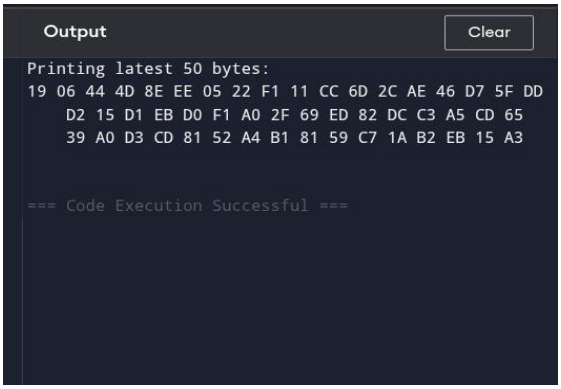
Implementation Highlights:

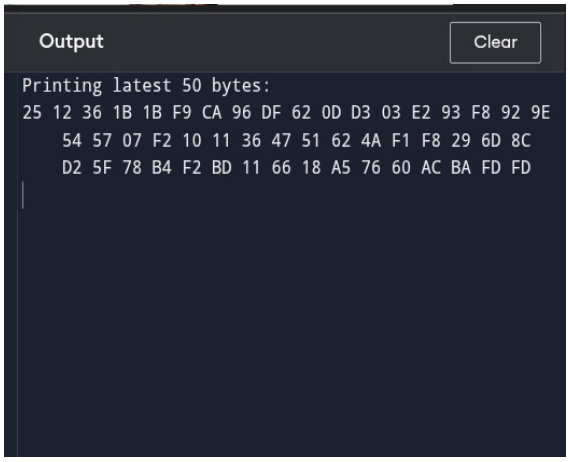
- Used buffer[] array of unsigned char to simulate raw data storage.
- Used rand() to generate random values and sleep() to simulate timer delays.

- GenerateSensorData() function adds new data every second.
- CheckAndPrintBuffer() prints the buffer value.

Output Screenshots:







Conclusion:

The assignment was implemented successfully. The learning from this assignment was very much useful to think about the practical implementation of interrupts and timers in real time projects.