**Universidad Autónoma de Guadalajara**

Ingeniería Electrónica Biomédica

System designing with Microprocessors

*“*Practice 2: Driver Development and Big Main Loop”

Jesús Arnoldo Zerecero Núñez

2885993

Andrea Alejandra Mondragón Olivos

2915351

Practice 2. Driver Development and Big Main Loop

**Introduction**

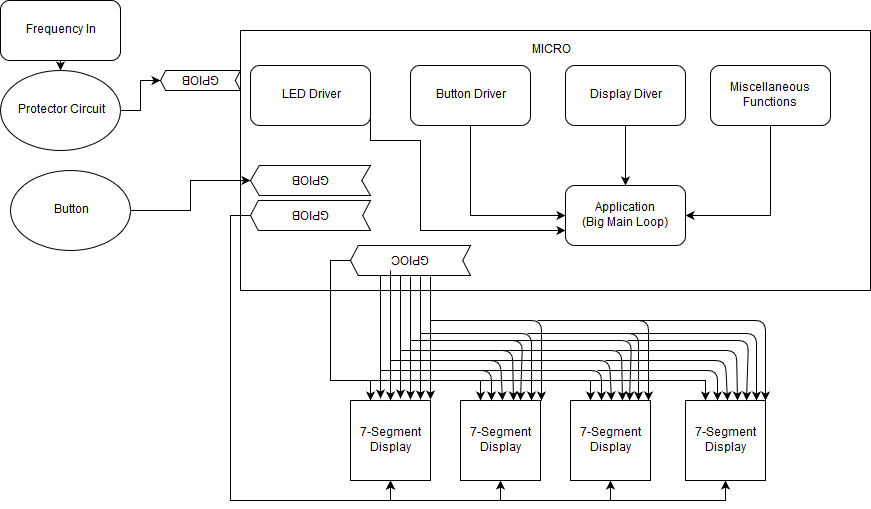
Drivers in applications are used to add an extra layer of abstraction to the code’s functioning, leaving the user with a much easier to use interface, without having him knowing what the code’s internal functioning is. This is useful when trying to make a code private for security reasons (perhaps there’s no use in modifying the code as it may lead to disastrous results). Drivers contain the function’s internal workings, leaving the user only with the function prototype and an idea of what the function does, but not exactly telling him how the function does it.

A Big Main Loop is a format where the application never ends as it’s included in an infinite cycle. Each needed function is called every cycle, while refreshing all the data and reading all conditions. This format is very user friendly, and can help flow the code’s execution without many time constraints.

In this practice, an application for a frequency meter will be built based on a Big Main Loop Format, using no peripheral modules or timers except for GPIO. Drivers to control the microcontroller’s leds, as well as drivers to control external push buttons and seven segment displays will also be created and implemented.

**Development.**

**Block diagram/modules implemented:**

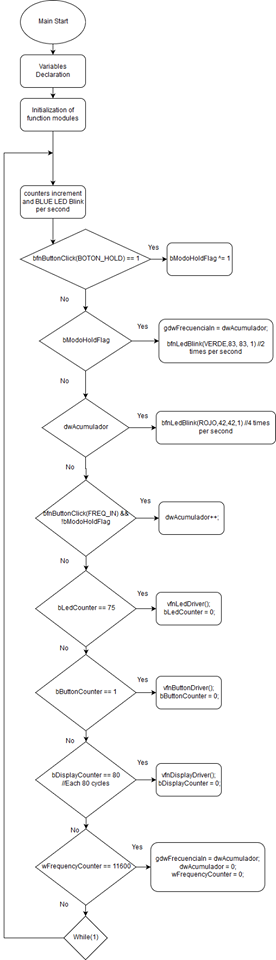


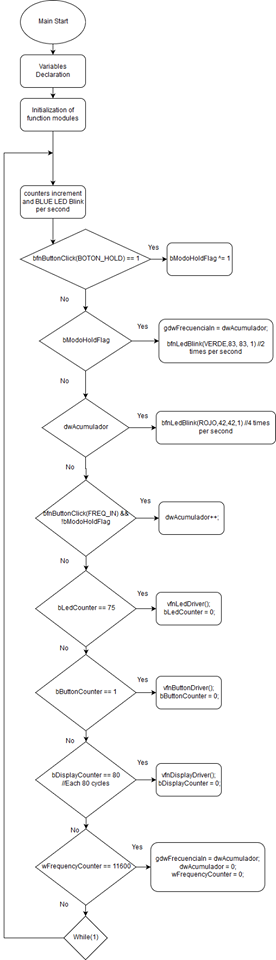
Protection Circuit

The application has the following features:

* The blue led will blink every second to show the system is on and functioning.
* The red led will blink 4 times every whenever an input frequency is detected.
* The external button may be pressed in order to make the system enter “Hold mode”
  + While in this mode, the last frequency detected freezes and the green led will blink 2 times every second.
  + To exit this mode, the button must be pressed again.
* While outside hold mode, the frequency detected will be shown in the 4 external seven segment displays and refreshed every second.
* If the frequency detected is greater than 5KHz, a “1---“ will be written to the displays

**Big Main Loop Flow chart:**





The following is a table analyzing how many assembler cycles it takes to complete every function used in the drivers, as well as the big main loop’s execution time:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module** | **Function** | **Estimated number of instructions** | **Real number of instructions\*** | **CPU usage (Bus clock = 10.5MHz)** |
| Led Driver | vfnLedInit | 21 | 66 | 324/10.5M  (>0.00003085%) |
| vfnLedOn | 5 | 28 |
| vfnLedOff | 5 | 28 |
| vfnLedToggle | 5 | 27 |
| bfnLedBlink | 20 | 78 |
| bfnLedBusy | 4 | 20 |
| vfnLedDriver | 30\*each led | 77\*each led |
| Button Driver | vfnButtonInit | 29 | 63 | 241/10.5M  (>0.0000295%) |
| bfnButtonState | 9 | 30 |
| bfnButtonClick | 15 | 43 |
| vfnButtonDriver | 22\*each button | 105\*each button |
| Display Driver | vfnDisplayInit | 52 | 137 | 506/10.5M  (0.00004819%) |
| vfnDisplayDriver | 94 | 643 |
| Miscellaneous functions | vfnDelay | 6\*asigned delay | 15\* asigned delay | 118/10.5M  (>0.000123%) |
| wfnBCD | 82 | 103 |
| Big Main Loop | Aplicación con contadores y condicionales. | 47 each cycle + asigned vfnDelay | 140 each cycle + asigned vfnDelay | 140/10.5M  (>0.0000133%) |

\*The “real number of instructions” were measured by looking at the C language to assembler language code conversion made by CodeWarrior Development Studio and counting the number of instructions it generated.

As shown in the table, this simple functions are far from exceeding the KL25z’s execution capacity, and were executed with no problem. A smaller and less powerful microcontroller would had sufficed as well.

**Conclusion:**

**Arnoldo:**

From this practice I learned the importance of making a generic driver for any application. This driver must be as simple as possible, while consuming the lowest execution time. I also learned that drivers are used to refresh the values and states of the rest of the module’s function, while not receiving nor returning any parameters. This pushes the programmer to use global variables, arrays or to create and share pin position and control structures. The hardest task we faced was the implementation of the frequency meter in the big main loop without altering the base delay/timer and the rest of the drivers’ execution. As higher frequencies weren’t able to be measured fast enough before many other functions had to be executed before the current pulse was able to be registered. After hours of tuning some of the delays and even removing the base delay, the best result we could reach was measuring up to 5KHz with a good degree of precision.

**Alejandra:**

For this practice, we used our knowledge about the microcontroller’s I/Os to analyze an input frequency and show it in external displays, while implementing a button which made the system enter HOLD mode. We had some problems with the big main loop structure, as it is a format not very know to us and visualizing the functions as generic drivers was a bit of a challenge. After this practice we have better understanding and more resources for building codes in a more optimal way, by implementing drivers.