LAB REPORT 3

Muhammad Anza Khan

260618490

Ayden Aamir Malik

260627064

Slider switches and LEDs

In this part of the lab we simply light up whichever led is on, if it’s slider switch is high.

ledss.s

We simply write the address of the slider switch into the led register which turns on the led for that specific slider.

Basic I/O

In this part, we use the state of the slider switches to display a value on the HEX displays. Each of the 4 pushbutton maps to a specific HEX display.
Since, the last two HEX displays i.e. HEX4 and HEX5 do not having any mapping push buttons, we simply flood their displays i.e. set to 1 all their display segments.

The first four slider switches are used to display a value between 0 to F on one of the four HEX displays.
Slider 9 is used to clear whatever is on the display.

File description

1.HEX_display.s
The subroutine in this assembly file called HEX_clear_ASM will turn off all the segments of all the HEX displays passed in the argument.

2.HEX_display_flood.s
The subroutine in this assembly file called HEX_flood_ASM will turn on all the display segments of whichever HEX display is called in the parameter. We do this by displaying 8 for each display.

3.HEX_displays_write_ASM.s
The subroutine in this assembly file is called HEX_write_ASM which displays a number between 0 to F to whichever HEX display is passed to it parameter.

In addition, we also use a subroutine from the pushbutton.s assembly file called read_PB_data_ASM which simply checks which button is pressed and displays to the corresponding HEX display.

Polling based stopwatch

Here we made a stopwatch by implementing a simple I/O based polling system. We made two HPS_TIM_config_t variables t0 which was the main timer for our stopwatch and the second variable

named poll which was used to poll the edgecapture registers every 5milliseconds.
Our main timer was incremented after every 10 milliseconds.
Within the main body of the code we check PB0, PB1 and PB2 which map to start, pause and reset respectively for the stopwatch.
When we press PB0 it sets all respective fields of t0 to 1 which starts the stopwatch.
PB1 pausing it by resetting all the fields to 0.
PB2 sets all the fields to 0 and also the counts to 0 which ends the timer and floods the HEX displays.

Interrupt based stopwatch

Here we make a stopwatch by implementing interrupts.
We enable interrupts for all the pushbuttons and whenever a specific pushbutton is pressed we initiate a particular interrupt service routine for that specific pushbutton. The field PB_int_flag acts as the flag for the ISR for the push buttons, anytime that this particular field is true, we service the ISR for whichever button. To do this, we test for certain values of the PB_int_flag which map to specific flags for the push buttons and in turn if true, we execute ISR for that specific push button.
As mentioned above, we check for specific values of PB_int_flag, the three values are 0, 1 and 2.
If 0, we execute the ISR for PB0. When this happens we start the HPS_TIM_config_t t0 by setting all its fields to 1's and once we do this we reset the PB_int_flag to 4 which we determined to be the reset value for it.
When it is 1, we simply pause the timer by setting all fields of t0 to be 0, and finally if it is 2, that means we kill the timer by setting all fields of t0 to be 0 and than resetting all count fields to be 0 and also within this we reset PB_int_flag to be 4, which resets the flag.