Bilkent University

CS Department

CS 224 - Digital Design and Computer Architecture

Preliminary Design Report Lab 07

Section 04

Okan Şen 21202377

25/12/2019

1)

- TRISx: These register control bits decide if each pin related to the I/O port is an input or output. If TRIS is 1, the pin is input, else the pin is an output.

- PORTx: The PORTx register functions differently depending on whether a pin is set to input or output. The simpler case is if a pin is set to output. Then, the PORTC register, for example, controls the value at the physical IO pins on Port

- LATx: This is used for writing data. A read of the LATx register returns the values held in the port output latches, on the contrary, the values on the I/O pins.

- ODCx is associated with each of the ports. Setting any of the design of the bit the equivalent pin to act as an open-drain output.

2) Rotation Code:

Part a:

```
#include <p32xxxx.h>

// This code shows and rotates the pattern (10001000) right or stops based on the
//input coming from the user. The pattern is to be shown on the LEDs.
int stop = 0;
int initial = 0b01110111; //Initial pattern. Note that 0 means on, while 1 means off.
int right = 1;
int temp; // New variable to store the last pattern in case the user freezes the loop. So that we
can continue from the last pattern.
void main(){
TRISD = 0x0; // All bits of PORTD are output. ~0 means output~
// Three bits of PORTA are inputs but only one of them is used in this example as a
//stop button, others are redundant. ~1 means input~
TRISA = 0b111;
// From PORTD, outputs will be sent to LEDs. Make sure that you physically connected
//them by looking at Figure 1, in the directives document.
// Initial pattern is sent to the LEDs through PORTD.
PORTD = initial;
while(1){
int lsb; //least significant bit
int msb; // most significant bit
int mask;

// Stop button is the push-button which is labeled as 1 on the board.
```

```
if(PORTABits.RA1 == 0){ // If stop button clicked
stop = !stop;
if (stop) {
        temp = PORTD; // We set temp to PORTD value in case we freeze it.
}
if(!stop){
// If process restarted, copy temp pattern into PORTD.
PORTD = temp;
}
}
if(!stop){
        //Rotate right
        if (right) {
                lsb = PORTD & 0x1; // Extract least significant bit
                mask = lsb << 7; // Least significant bit will be the msb of the shifted pattern
                PORTD = (PORTD >> 1) | mask; // Paste lsb to the leftmost bit the right shifted
portd
        }
        //Rotate left
        else if (!right) {
                msb = (PORTD & 0x80) >> 7 // Take PORTD's 8th bit and shift right by 7
                PORTD = (PORTD << 1) | msb; // Shift portd left by 1 bit and or with msb which is
00..0 or 00..1
        }


} else {
//Do not shift anything, that is, stop.
PORTD = 0b11111111;
}
delay_ms(1000); // Wait 1 second.
}
}
// Rotation ends here
```

3)

7 segment display

Part b:

```
#include <p32xxxx.h>
```

// This code starts x from 0 and calculates its cube, then display it on the 7 segment display area, until it reaches 21.
// The display method is missing the connections with the ports and 32 bit areas only. Otherwise, all the logic and coding is set.

```
int d0, d1, d2, d3; // variables to store how many digits we will need for x's current value.

void main(){
	// Always keep this loop running
	while(1){
		// Make a counter so that loop starts from 0
		// and resets when it hits 22.
		for (int i = 0; i <= 21; i++) {
			int val = i * i * i; // get x^3
			int number = val;
			int digits = 0;

			// This is a loop to get the separate digits
			// of current X, to call a display method.
			while (number != 0) {
				number /= 10;
				digits++;

				if (digits == 1) {
					d3 = number%10;
				}
				else if (digits == 2) {
					d2 = number%10;
				}
				else if (digits == 3) {
					d1 = number%10;
				}
				else if (digits == 4) {
					d0 = number%10;
				}
			}
			// Then call display method to display the value.
			display (d0, d1, d2, d3);
		}
	}
```

```
}

void display (int d0, int d1, int d2, int d3) {

	// In the display method, we add waits because the 7 segment cannot display everything
at once
	// so we use a little trick which actually doesn't show everything at once but the human
eye
	// can not perceive this quick differences, so we see it as simultaneously displayed.

	TRISD = 0x0; // All bits of PORTD are output. ~0 means output~
	// From PORTD, outputs will be sent to 7 segment display. Make sure that you physically
connected
	//them.
	//
	PORTD = 00000000; // I do not know what to write here since I couldn't find a good
	// documentation of 7 segment display bits on Unilica.


		// DO DISPLAY FUNCTIONS HERE AT D3rd DIGIT.

		delay_ms(5); // Wait 5 miliseconds.


		// DO DISPLAY FUNCTIONS HERE AT D2nd DIGIT.

		delay_ms(5); // Wait 5 miliseconds.

		// DO DISPLAY FUNCTIONS HERE AT D1st DIGIT.

		delay_ms(5); // Wait 5 miliseconds.

		// DO DISPLAY FUNCTIONS HERE AT D0th DIGIT.

		delay_ms(5); // Wait 5 miliseconds.


}
```