Lab 4 – FreqDiv

Frequency/Voltage table

|  |  |  |
| --- | --- | --- |
| Port | Index | Voltage/Frequency |
| 5V | No | 5.09Vrms DC |
| 3V | No | 3.29Vrms DC |
| GND | No | 0.01Vrms DC |
| P4.0 | 0 | 1.38KHz |
| P4.1 | 1 | 694Hz |
| P4.2 | 2 | 346.9Hz |
| P4.3 | 3 | 173.3Hz |
| P4.4 | 4 | 86.7Hz |
| P4.5 | 5 | 43.3Hz |
| P4.6 | 6 | 21.6Hz |
| P4.7 | 7 | 10.8Hz |

Frequency/Voltage chart

Questions

1. When the GPIO port is high, the rms voltage is 3.3V
2. It is not exactly 0 because of the noise in the system. It’s very small and can be ignored in most circumstances.
3. The relationship between the frequencies measured is that as you measure from P4.0 to P4.7, the next frequency is about half from the previous. In other words, the frequency is divided with each pin moving down the list. Therefore, the period for each cycle is proportionally doubled in the same manor. The reason this happened is because the LSB is assigned to P4 of index 0, and as you move bit by bit to the MSB, the index of each bit increases by one, and therefore takes the pin ID of the same number as the char bit index. In the loop, the char is incremented by one during each pass, then the function to set selected pins high is executed. Then, the delay code happens, then the functions to set the pins low is executed on a bitwise inversion of the char, effectively completing the cycle. Therefore, it would take 2 loop executions to make the square wave for the last pin tied to the LSB. Additionally, since it’s binary incrementation, it would take 4 loop executions to make the square wave for the second to last pin, tied to the second to LSB. This is what creates the division of the frequency form the LSB all the way to the MSB.
4. To get the number of times per second the while loop ran a complete cycle, you only need to look at the LSB, since it would be changing the most. In one cycle in the loop, it would also set the LSB pin high and then low, therefore also a cycle. Pin 4.0 shows the LSB, taking 1.38KHz, therefore 1,380 times per second.

Code: (main.c)

////////////////////////////////////////////////////////

//Willard Wider

//Lab 04 - FreqDiv

//05/17/18

//Fequency division

////////////////////////////////////////////////////////

**#include** <ti/devices/msp432p4xx/driverlib/driverlib.h>

**#define** DELAY 100

**int** **main**(**void**)

{

**volatile** uint32\_t i, j;

**unsigned** **char** incriment = 0;

// Stop watchdog timer

WDT\_A\_hold(WDT\_A\_BASE);

//set the pins for the fewq div

**GPIO\_setAsOutputPin**(GPIO\_PORT\_P4,GPIO\_PIN0);

**GPIO\_setAsOutputPin**(GPIO\_PORT\_P4,GPIO\_PIN1);

**GPIO\_setAsOutputPin**(GPIO\_PORT\_P4,GPIO\_PIN2);

**GPIO\_setAsOutputPin**(GPIO\_PORT\_P4,GPIO\_PIN3);

**GPIO\_setAsOutputPin**(GPIO\_PORT\_P4,GPIO\_PIN4);

**GPIO\_setAsOutputPin**(GPIO\_PORT\_P4,GPIO\_PIN5);

**GPIO\_setAsOutputPin**(GPIO\_PORT\_P4,GPIO\_PIN6);

**GPIO\_setAsOutputPin**(GPIO\_PORT\_P4,GPIO\_PIN7);

//DEBUG: show at least something is working (like the LED)

//show that at least the code is being entered

**GPIO\_setAsOutputPin**(GPIO\_PORT\_P1,GPIO\_PIN0);

**while**(1)

{

//set them high

**GPIO\_setOutputHighOnPin**(GPIO\_PORT\_P4, ++incriment);

//delay of pause

**for**(i = 0; i < DELAY; i++)

{

}

//set the inverts back to low

**GPIO\_setOutputLowOnPin**(GPIO\_PORT\_P4, ~incriment);

//DEBUG: show at least something is working

**GPIO\_toggleOutputOnPin**(GPIO\_PORT\_P1,GPIO\_PIN0);

}

}