## Assignment 1

1. *Generate a square wave in the frequency range 100 Hz to 5 kHz*

To generate the square wave, we decided to use one of the hardware timers to create an interrupt signal at a set interval. The ADuC841 provides a 16-bit up-counter at 11.0592 MHz. The timer counter triggers an interrupt service routine (ISR) every time it overflows. This ISR complements the output value of a particular pin.

In order to generate a signal of 100 Hz, we need to trigger the ISR at 200 Hz.

Hence, for 200 Hz;

Therefore, the counter needs to count 55296 times. Since the counter is counting up, the initial (reload) value that needs to be loaded is 10240.

Again, for our example;

Similarly, to generate the frequency of 5 kHz, the counter needs to count approximately 1106 times (rounded to an integer) which can be represent by using 11 bits, with the initial value 64430. These initial values can now be calculated by:

We know that the counter uses at least 11 bits. Thus, we choose to use Timer 2 because we need to reload a value larger than 8 bits while Timer 0/1 only allows 8-bit up-counter with auto reload.

In this assignment, we use 4 of the switches (P2.0-3) to select the required frequency of 16 generated frequencies.

|  |  |
| --- | --- |
| Switches | Frequency (Hz) |
| 0000 | 100 |
| 0001 | 200 |
| 0010 | 400 |
| 0011 | 800 |
| 0100 | 1600 |
| 0101 | 2000 |
| 0110 | 2200 |
| 0111 | 2500 |

|  |  |
| --- | --- |
| Switches | Frequency (Hz) |
| 1000 | 2800 |
| 1001 | 3000 |
| 1010 | 3400 |
| 1011 | 3800 |
| 1100 | 4200 |
| 1101 | 4400 |
| 1110 | 4800 |
| 1111 | 5000 |

We also use another switch (P2.4) to enable or disable the interrupt so that the sound can be turned on or off.

The bit value of P2.4 is used directly to control IE.7, the interrupt enable flag for Timer 2.

The program is based on the use of hardware timer interrupts to generate the square wave, while using the software to flash the LED and alter the frequency as a result of switch positions.

First, we set Timer 2 to run and enable the Timer 2 Interrupt ISR and create the LUT (words) for the frequencies. Then we retrieve the switch value from P2 and update the frequencies as per the value. The software then enters the main loop.

The LED is flashed via the use of a software delay. To make the most use of the cpu cycles, we incorporated the switch values into the delay cycle. Similarly, to the timer0.asm example given, we run through two loops to create a delay that is a multiple of 10 ms and is called as a subroutine.

By including several instructions to check the value of P2 and compare it against a previously found value in memory, we can decide whether to overwrite the reload values of Timer 2. As this check is performed during the delay loops, the effect of changing a switch is almost instantaneous, and the impact on the delay time is minimal. The LED output bit is set to be complemented in between the delays.

The subroutine that alters the frequency of the T2 ISR first retrieves the 4-bit value that corresponds to our switch positions, then multiplies it by two as we are dealing with 16-bit words and not bytes in our LUT. We then use the data pointer DTPR to access the first byte of the word using the offset from the switches and the LUT label, which corresponds to the high byte of the reload value. The next byte in memory corresponds to the low byte of the reload value. These values are set, after which we check the interrupt pin and set the interrupt enable respectively.

The functionality of our code was checked via the evaluation board and the switches, with the help of the on-board piezo-electric transducer. When the first four switches are in the off position, a low tone is heard. When they are in the on position, a high-pitched tone is heard. When P2.4 is in the off position, the tone stops, otherwise on.