The approach we took to change the frequency output by the transducer was to change the high byte reload value for Timer2 (RCAP2H) based on the configuration of three switches. By changing the reload value for Timer2, we change the rate at which interrupts happen. If the input to drive the transducer is toggled in the Interrupt Service Routine (ISR), then the speaker frequency is the interrupt frequency divided by 2.

Timer2 is a 16-bit counter, which allows us to count as high as 65536. At a clock frequency of 11.0592 MHz, to have an interrupt frequency of 200 Hz (i.e. a speaker frequency of 100 Hz), the timer would have to be able to count to 55296. A 16-bit counter can easily handle this, which is why Timer2 was chosen.

Our program borrows elements from both the blink.asm and timer0.asm files. Like in blink.asm: the main loop blinks an LED with a fixed frequency by means of a delay subroutine. From timer0.asm: at the program address for the Timer2 interrupt, an (ISR) for Timer2 is jumped to, as there is not enough space in 8 bytes to handle what we wish to do in the ISR.

A lookup table is defined in the setup section of the program. This table holds the values to be loaded into RCAP2H for the switch configurations. A table showing how these values are calculated can be seen below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Switch  Configuration | Desired Frequency (Hz) | Interrupt Frequency (Hz) | Required no. of counts | Counter Start Value | RCAP2H Reload Value |
| 000 | 100 | 200 | 55296 | 10240 | 40 |
| 001 | 250 | 500 | 22119 | 43417 | 170 |
| 010 | 500 | 1000 | 11060 | 54476 | 213 |
| 011 | 1000 | 2000 | 5530 | 60006 | 235 |
| 100 | 1500 | 3000 | 3687 | 61849 | 242 |
| 101 | 2000 | 4000 | 2765 | 62771 | 246 |
| 110 | 3000 | 6000 | 1844 | 63692 | 249 |
| 111 | 4000 | 8000 | 1383 | 64153 | 251 |

From the table, the interrupt frequency is twice the desired transducer frequency. The required number of counts is the clock frequency divided by the interrupt frequency, rounded up to the nearest integer. The counter start value is the max counter value (65536) minus the required number of counts. Finally, the reload value is the start value divided by 256, rounded up to the nearest integer. The reason for this division is to fit the counter start value into one byte. The low reload byte (RCAP2L) is loaded with 128 in the setup section and remains constant for the remainder of the program as it has less impact on the interrupt frequency. Although this does mean a loss in accuracy of the output frequency, it does mean that we can update the reload value in one instruction. The loss in accuracy is negligible for this application, however in other use cases it may be unacceptable.

The estimated output frequencies with the above approximations and the measured frequencies can be seen in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| Desired Frequency (Hz) | RCAP2H Reload Value | Calculated Frequency  (Hz) | Measured Frequency (Hz) |
| 100 | 40 | 100.3 | 100.2 |
| 250 | 170 | 252.7 | 252.5 |
| 500 | 213 | 508.3 | 507.5 |
| 1000 | 235 | 1053.7 | 1050 |
| 1500 | 242 | 1600 | 1590 |
| 2000 | 246 | 2273.7 | 2260 |
| 3000 | 249 | 3323.1 | 3290 |
| 4000 | 251 | 4800 | 4740 |

The main loop in the program toggles the LED and checks the state of switch 7. If the switch is high, then the interrupt for Timer2 is enabled, i.e. the speaker will make a tone. If the switch is low, then the speaker will be silent. The LED will toggle regardless.

In the ISR, the state of switches 0-2 are checked and based on their configuration, a value from the lookup table is loaded into RCAP2H. The input to the transducer is also toggled in the ISR which creates the tone.