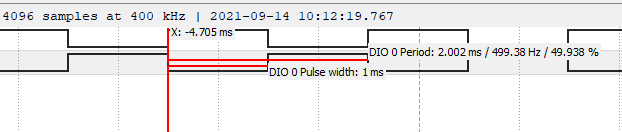
|  |  |  |
| --- | --- | --- |
| Desired Delay | Measured Delay (SW) | Measured Delay (HW) |
| 1 ms | 1 ms | 1 ms |
| 10 ms | 10 ms | 10 ms |
| 100 ms | 100 ms | 100 ms |
| 1000 ms | 1000 ms | 1000 ms |

1. I would want to use a software delay when my while(1) loop is small and I don’t anticipate on receiving any signals while waiting. A software delay is also useful if you want to halt your program for a specified time. The advantage of a hardware delay is that the PIC32 can also be doing other stuff in the background and check in with the core timer every once in a while.

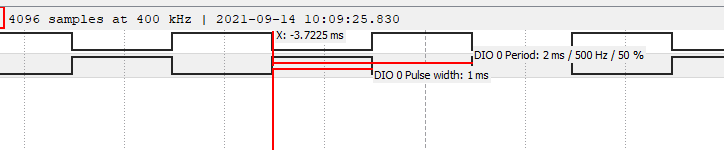
The minimum and maximum of the hardware delay is however long it takes to check the core timer, and check it again to see if it has incremented by at least one. The maximum time a hardware assisted delay can have is however long it takes to iterate through the core timer -1 iteration. The minimum time a software delay could wait is 1 millisecond. The maximum software delay is [11111111111111111111111111111111]2 milliseconds (the maximum value that an unsigned int variable can be.

The hardware assisted delay can be extended by using another register to store how many times the sore timer has returned to its starting value. This is electively adding another bit to the end of the core timer. [Binary representation of how many times the core timer has returned to the same position] + [what the core timer is reading]. The pure software delay can be increased by multiplying how many ticks happen per millisecond. For example, if you want to double the maximum time you can wait you take the number of ticks for 1 ms and multiply by 2.

***Figure 1: Hardware Assisted 1ms Delay***

******

***Figure 2: Software Assisted 1ms Delay***

******