

> I am using /IRQ2 for a data acq application at a fairly high duty cycle and it works fine. At 16MHz, how long should it normally take for the CPU housekeeping prior to entering and after finishing an ISR?

The basic no-work interrupt overhead is on the order of 10us.

Long answer:

Interrupt latency from IRQx to the prefetch of the first instruction of the interrupt handler is about 42 clocks or 2.6us at 16MHz and zero waits, to which you have to factor in the worst-case instruction length in clocks which must complete before interrupt handling begins and this is about 61 clocks or 3.8us for the DIVS.L instruction. This latter number is also the minimum IRQ low width to prevent a spurious interrupt exception.

This means the worst case latency from IRQx assertion to the start of an assembly language routine which does not attempt to save any register context is about 6.4us. The corresponding RTE instructions adds about 26 clocks or 1.6us for a basic assembly language overhead of about 8.1us.

If your handler is coded in C and you're using the BIOS IEV_C_FUNCT wrapper macro, it saves and restores the five standard Metrowerks C registers D0, D1, D2, A0, and A1 along with a pointer to the exception stack context in about 90 clocks or 5.6us.

This means that if you had a C interrupt handler that did nothing, the worst case interrupt overhead is about 13.7us or 73kHz and the best case timing would be about 9.9us or 101kHz. A do nothing assembly language handler would have worst case 8.1us or 123kHz, and best case 4.3us or 232kHz timing.