CG2271 Real-Time Operating Systems

Tutorial 2 Suggested Solutions

1. Which modules generate the IRQ0, IRQ10 and IRQ31 interrupt requests, and what are their CMSIS typedef enumeration labels? Examine the Interrupt Vector Assignments table in the KL25Z subfamily reference manual and the MKL25Z4.h file (or appropriate *device*.h file for a different MCU device).

Answers:

IRQ0: DMA0_IRQn – Direct memory access unit 0.

IRQ10: SPI0_IRQn – Serial peripheral interface unit 0.

IRQ31: PORTD_IRQn - Port D interrupt.

You can refer to Reference Manual Pg 53.

0x0000_0040	16	0	0	DMA	DMA channel 0 transfer complete and error
0x0000_0068	26	10	2	SPI0	Single interrupt vector for all sources
0x0000_00BC	47	31	7	Port control module	Pin detect (Port D)

2. This question involves configuring registers so that if interrupts IRQ0, IRQ10 and IRQ31 are requested simultaneously, the CPU responds as requested. For each question, explain what values must be loaded into which registers, and then write C code which uses the CMSIS functions to perform that operation.

   a. Interrupts are serviced in order IRQ10, IRQ0, IRQ31.

Answers:

Priorities must be in increasing in the above order. For example, 64 (1), 128(2), 192(3), or 0 (0), 64 (1), 128 (2).

```
Set NVIC_IPR2[23:22] to 01      // IRQ10
Set NVIC_IPR0[7:6] to  10       // IRQ0
Set NVIC_IPR7[31:30] to  11     // IRQ31

NVIC_SetPriority(SPI0_IRQn, 1); // IRQ10
NVIC_SetPriority(DMA0_IRQn, 2);         // IRQ0
NVIC_SetPriority(PORTD_IRQn, 3);        // IRQ31
```

b. We wish to enable IRQ13 but disable IRQ24. What value needs to be loaded into which register bits, and what is the CMSIS code call to accomplish the same?

Answers:

Set NVIC_ISER[13] to 1
NVIC_EnableIRQ(13);

Set NVIC_ICER[24] to 1
NVIC_DisableIRQ(24);

3. Consider a system built on a 48 MHz KL25Z with a main loop and an interrupt which occurs at a 10 kHz frequency. Assume the ISR take 14.9 us to execute and there is a total of 1 us of response and return time overhead.

a. What percentage of the processor's time is spent servicing the Interrupts including the overheads?

Answers:

Interrupt Frequency: 10kHz -> Interrupt Period: 100 us
Total Interrupt Processing Time: 14.9 + 1 -> 15.9us

% of time spent servicing interrupts: 15.9us / 100us -> 15.9%

b. What percentage of the processor's time is left for the main loop to execute?

Answers:

% of time for main loop: 100% - 15.9% -> 84.1%

c. If the main loop requires 37 ms of computation to execute one iteration, what is the minimum main loop update rate (in Hz)?

Answers:

48MHz -> 48000000 cycles/s
Main Loop Utilization -> 84.1% -> 84.1% * 48000000 -> 40368000

Period for 1 clock cycle -> 1 / 48M -> 20.83ns
Number of Cycles for 1 iteration -> 37m / 20.83n -> 1776000

% of cycles in Main Loop -> 1776000 / 40368000 -> 0.043995

Frequency -> 1 / 0.043995 -> 22.73Hz

4. Consider the system above again, but now with the interrupt running at 25 kHz and the ISR taking 34 us to execute with 1 us of overheads.

a. What percentage of the processor's time is spent in interrupt response and return overhead?

Answers:

Interrupt Frequency: 25kHz -> Interrupt Period: 40 us
Total Interrupt Processing Time: 34 + 1 -> 35us

% of time spent servicing interrupts: 35us / 40us -> 87.5%

b. What percentage of the processor's time is left for the main loop to execute?

Answers:

% of time for main loop: 100% - 87.5% -> 12.5%

c. If the main loop requires 37 ms of computation to execute one iteration, what is the minimum main loop update rate (in Hz)?

Answers:

48MHz -> 48000000 cycles/s
Main Loop Utilization -> 12.5% -> 12.5% * 48000000 -> 6000000

Period for 1 clock cycle -> 1 / 48M -> 20.83ns
Number of Cycles for 1 iteration -> 37m / 20.83n -> 1776000

% of cycles in Main Loop -> 1776000 / 6000000 -> 0.296

Frequency -> 1 / 0.296 -> 3.38Hz