**CG1112 Engineering Principles and Practices II for CEG**

**Week 6 Tutorial Part 2 – Interrupts and Timers**

Question 1.

We wish to read a block of 16384 bytes from a disk drive.

1. We have the following program that reads from the drive using polling.

byteCount=16384;

i=0;

while(byteCount>0)

{

while(!DRIVE\_READY);

buffer[i++]=DRIVE\_DATA\_LATCH;

byteCount--;

}

Assuming that our microprocessor runs at 50 MHz and the drive transfers data at a rate of 1 megabyte per second (1 MBPS), how many clock cycles are spent on polling between each byte? How many clock cycles are spent in total on polling?

We get 1000000 bytes per second, or 1 byte every microsecond. Each clock cycle is 20ns, and hence it takes 50 cycles per byte.

We thus spend 50 cycles stuck in the while(!DRIVE\_READY) loop for a total of 16384 times, giving us 819,200 cycles wasted.

1. We now use interrupt driven I/O instead, and an interrupt is triggered each time a byte arrives from the drive. Assuming that it takes 50ns to process each interrupt, how many clock cycles are spent processing each interrupt? How many clock cycles are spent in total when we transfer 1 block from the drive?

50ns = 2.5 clock cycles. Each byte triggers an interrupt. Time taken to process the entire block is 2.5 \* 16384 = 40,960 cycles.

1. There is a 3rd transfer mode not covered in lectures called Direct Memory Access or DMA, where a separate piece of hardware called the DMA Controller or DMAC completely takes over the transfer of data and sends an interrupt when the transfer is complete. The downside is that it takes a little bit of time to set up the transfer.

Assuming it takes 1000ns to set up a DMA transfer and 200ns to process the DMA interrupt, how many clock cycles are spent in total when transferring one block from the disk drive?

Total = 1000+200 = 1200 ns = 60 cycles

Question 2

Repeat Question 1, but with a block size of 8 bytes instead of 16384 bytes. Comment on the relative efficiencies of polling, interrupt driven I/O and DMA. From your answers comment on why devices like the keyboard, mouse and UART port do not use DMA.

Polling:

Each byte still takes 1 microsecond or 50 cycles. Total number of cycles taken is 50\*8 = 400 cycles.

Interrupts:

Each interrupt still takes 2.5 cycles. Total time = 8\*2.5 = 20 cycles.

DMA:

No matter what the block size is it will always take 60 cycles.

Since DMA set-up and interrupt processing time is independent of block size, DMA is not efficient for small block sizes.  This means that “character devices” like the mouse, keyboard and UART port do not benefit from the advantages of DMA.

Question 3

We are given the following contents of TCCR1A and TCCR1B, and OCR1A holds the value of 247. Given that TCNT1 is initially 0, that we are operating Timer 1 in CTC mode, and that all interrupt flags in TIMSK1 and SREG are properly set:

TCCR1A = 0b01010000;

TCCR1B = 0b00001100;

1. What is the interval between which TIMER1\_COMPA\_vect is triggered?

The prescalar is 0b100 which is 256. At 16MHz, the period is 256/16000000 = 16 us. The period therefore is (247 + 1) \* 16 us = 3.968 ms.

1. Sketch the waveform on OC1A.

There is a match approx. once per 4 ms, and the COM1A0 and COM1A1 bits are 0b01, which is a “toggle on match”. Hence we get a square wave of period of about 8 ms.

Question 4

We want to trigger TIMER0\_COMPA\_vect on Timer 0 once every 650 us (microseconds) in CTC Mode. Complete the following table. Round up any cases where V is not an integer (e.g. 34.3 -> 35, 38.8 -> 39, etc). Assume that the Atmega328P is being clocked at 20 MHz and that CLKIO is identical to the Atmega’s frequency.

Note: If the V value cannot be loaded into OCR0A, write “N/A” in the Actual Period column.

Note: To get the period we take P / 20000000, where P is the prescalar value.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prescalar | Period in us | V (no rounding) | V (rounding) | Actual period |
| CLKIO/1 | 0.05 | N/A | N/A | N/A |
| CLKIO/8 | 0.4 | N/A | N/A | N/A |
| CLKIO/64 | 3.2 | 203.125 | 204 | 652.8 us |
| CLKIO/256 | 12.8 | 50.78125 | 51 | 652.8 us |
| CLKIO/1024 | 51.2 | 12.6953125 | 13 | 665.6 |

Based on the table above comment on which value of V gives the best accuracy. Why do you think this is so?

Prescalar of 64 and 256 give the same accuracy while prescalar of 1024 gives very poor accuracvy.

In general the smaller a prescalar is, the more accurate the timing would be. Of course this will also depend on the rounding error.