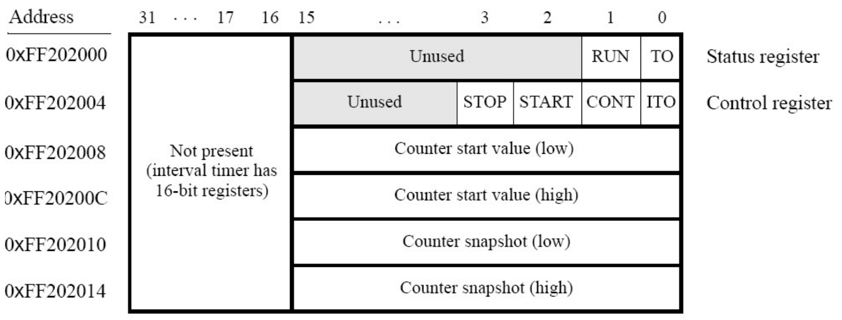
임베디드시스템 설계 및 실습 중간시험

학 번 : **이름:**

1. 임베디드시스템의 특징을 4개 이상 들고 설명하시오. 150자 이내. [15점]

2. 다음은 KAU_Computer에 내장된 Sound Codec의 register map이다. Sound Codec을 예로 들어 Memory Mapped I/O를 설명하시오. 100자 이내. [15점]



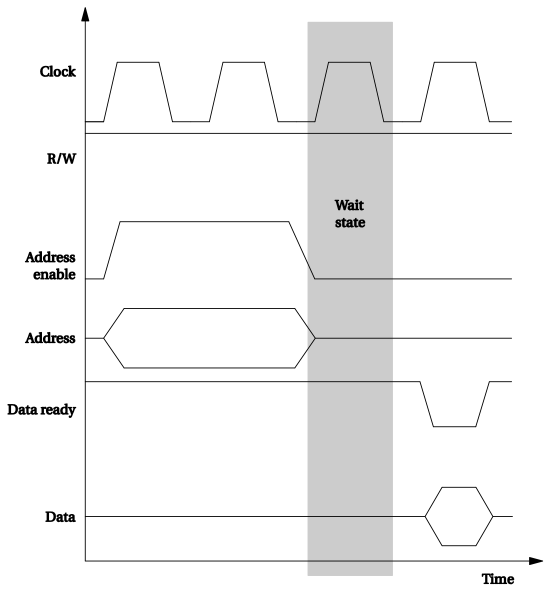
3. 어떤 임베디드시스템 응용에서는 DMA (Direct Memory Access) Controller를 사용하여 시스템의 처리 속도를 높일 수 있고, 어떤 응용에서는 DMA 사용이 시스템의 처리 속도에 거의 영향을 주지 못한다. 다음 물음에 답하시오.
   1) DMA의 구체적인 동작과 그 효용성을 설명하시오. [10점]
   2) DMA 사용이 시스템의 처리 속도를 높일 수 있는 응용 예와 거의 높이지 못하는 응용 예를 들고, 그 이유들을 설명하시오. [10점]

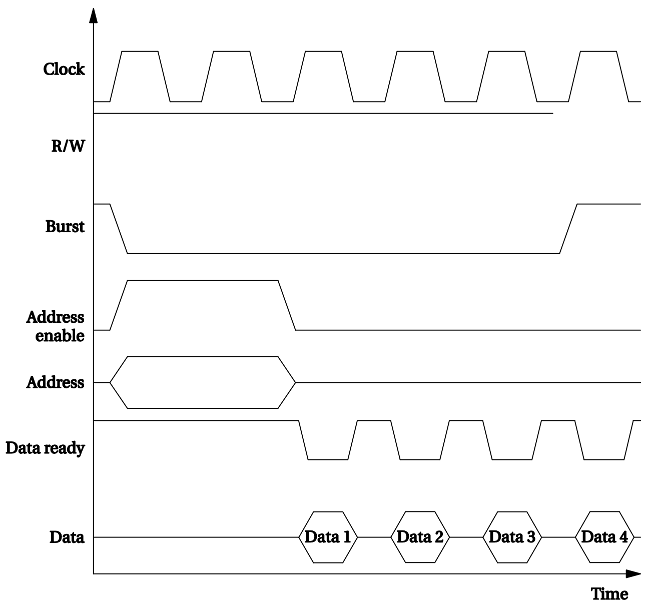
4. NIOS II processor에서 exception 발생시 이를 처리하는 절차를 순서대로 나열하시오. [10점]

1. Writes the address of the instruction after the exception into the *ea* register (*r29*)
2. Saves the existing processor status information by copying the contents of the *status* register (*ctl0*) into the *estatus* register (*ctl1*)
3. Clears the *PIE* bit in the *status* register, thus disabling the additional external processor interrupts
4. Transfers execution to the address of the *exception handler* which determines the cause of the exception and dispatches an appropriate *exception routine* to respond to the exception
5. Clears the *U* bit in the *status* register, to ensure that the processor is in the *Supervisor* mode

----

5. 다음은 어떤 임베디드시스템 내부 bus를 통해 bus master가 bus slave로부터 data를 read하는 timing diagram이다. 단, data read 직후 바로 다음 read transaction을 시작할 수 있다.





1) 다음 표를 채우시오. [15점]

Transaction type	Single word	Single word	Burst word	Burst word	Burst word
wait cycle	0	2	0	4	2
Burst length	1	1	4	4	8
Bus width (bit)	32	32	32	64	32
Bus clock (MHz)	100	100	100	50	50
Throughput (kbps)	①	②	③	④	⑤

2) 임베디드시스템이 HD 이미지 센서 (1280x720 해상도, 화소 하나당 RGB 3bytes)로부터 초당 30장의 화면률로 실시간 입력을 받으려고 한다. 이미지 데이터를 저장하는 내장 메모리와 이미지 센서가 모두 동일한 wait cycle과 최대 burst length를 지원한다고 할 때, DMA를 이용하여 이미지 센서로부터 메모리로 데이터 전송에 가장 적합한 버스 설계를 하나 고르고, 그 이유를 설명하시오. [15점]

6. Big-Endian 임베디드시스템에서 다음 메모리 내용과 C code에 대해 답하시오.

0x40008000	00 01 02 03 04 05 06 07		#define BASE_ADD 0x40008000
0x40008008	08 09 0a 0b 0c 0d 0e 0f		int n;
0x40008010	10 11 12 13 14 15 16 17	①	unsigned char *pA=BASE_ADD;
0x40008018	18 19 1a 1b 1c 1d 1e 1f	②	int *pB=BASE_ADD;
0x40008020	20 21 22 23 24 25 26 27		
0x40008028	28 29 2a 2b 2c 2d 2e 2f	③	for(n=0; n<16; n++) printf(“%x\n”, *(pA+n));
🔔 표시 데이터 주소는 왼쪽에서 오른쪽으로 증가함		④	for(n=0; n<6; n++) printf(“%x\n”, *(pB+n));

1) Displacement addressing mode가 사용된 code line의 번호를 모두 적으시오.[10점]

2) Code line ③의 출력을 적으시오.[10점]

3) Code line ④의 출력을 적으시오.[10점]

mode	**Effective address**
Immediate	operand=value
Register	EA=ri
Register indirect	EA=[ri]
Displacement	EA=[ri]+X
Absolute	EA=LOC

7. KAU_Computer (or DE1-SoC Computer) 내장 타이머를 이용하여 200ms heartbeat를 생성하려고 한다. 뒷장의 timer manual을 참고하여 아래 code를 채우시오. [30점]

```
volatile int timeout;               // used to synchronize with the timer
alt_isr_func interval_timer_ISR( void );

int main(void)  {
    volatile int *pTimer = (int *) INTERVAL_TIMER_BASE;
    int timer_ctxt=0, pb_ctxt=0;

    timeout = 0;                              // synchronize with the timer

    /* set the interval timer period : 200ms */
    int counter = __________①__________;
    *(pTimer + 0x2) = __________②__________;
    *(pTimer + 0x3) = __________③__________;

    alt_irq_enable(INTERVAL_TIMER_IRQ);
    alt_irq_register(INTERVAL_TIMER_IRQ, (int *)&timer_ctxt, interval_timer_ISR);

    /* start interval timer in continuous mode, enable its interrupts */
    *(pTimer + 1) = __________④__________;

    while (1) {
        __________⑤__________;       // wait to synchronize with timer
        timeout = 0;
    }
}

alt_isr_func interval_timer_ISR( void )
{
    volatile int * pTimer = (int *) INTERVAL_TIMER_BASE;
    __________⑥__________;
    timeout = 1;
    return;
}
```

2.6 Interval Timers

The DE1-SoC Computer includes a timer module implemented in the FPGA that can be used by the Nios II processor. This timer can be loaded with a preset value, and then counts down to zero using a 100-MHz clock. The programming interface for the timer includes six 16-bit registers, as illustrated in Figure 14. The 16-bit register at address 0xFF202000 provides status information about the timer, and the register at address 0xFF202004 allows control settings to be made. The bit fields in these registers are described below:

- *TO* provides a timeout signal which is set to 1 by the timer when it has reached a count value of zero. The *TO* bit can be reset by writing a 0 into it.
- *RUN* is set to 1 by the timer whenever it is currently counting.
- *ITO* is used for generating interrupts, which are discussed in section 3.
- *CONT* affects the continuous operation of the timer. When the timer reaches a count value of zero it automatically reloads the specified starting count value. If *CONT* is set to 1, then the timer will continue counting down automatically. But if *CONT* = 0, then the timer will stop after it has reached a count value of 0.
- (*START*/*STOP*) is used to commence/suspend the operation of the timer by writing a 1 into the respective bit.

The two 16-bit registers at addresses 0xFF202008 and 0xFF20200C allow the period of the timer to be changed by setting the starting count value. The default setting provided in the DE1-SoC Computer gives a timer period of 125 msec. To achieve this period, the starting value of the count is 100 MHz × 125 msec = 12.5×10^6. It is possible to capture a snapshot of the counter value at any time by performing a write to address 0xFF202010. This write operation causes the current 32-bit counter value to be stored into the two 16-bit timer registers at addresses 0xFF202010 and 0xFF202014. These registers can then be read to obtain the count value.

A second interval timer, which has an identical interface to the one described above, is also available in the FPGA, starting at the base address 0xFF202020. Each Nios II processor has exclusive access to two interval timers.

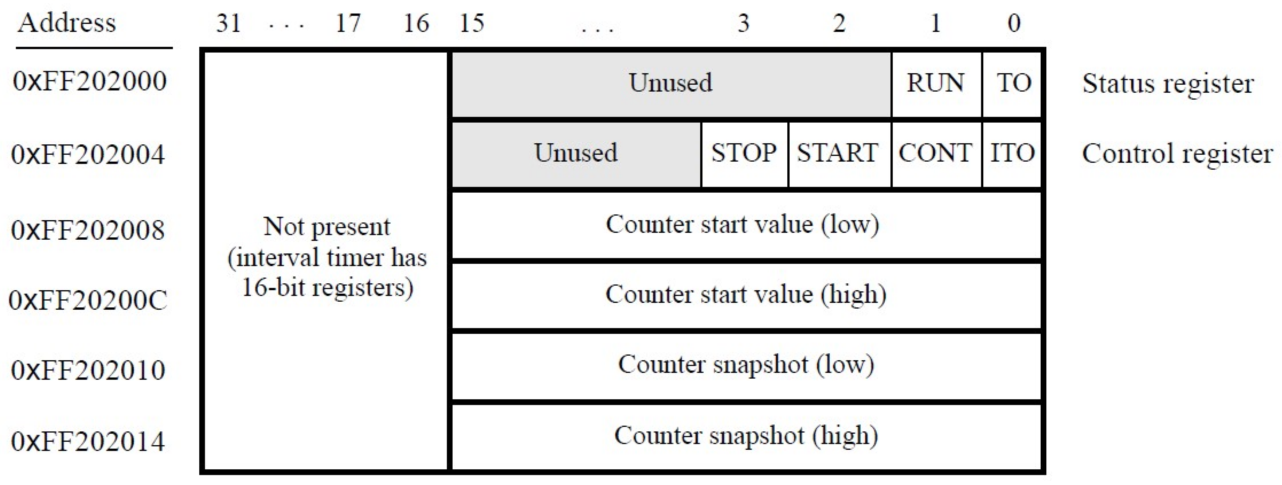


Figure 14. Interval timer registers.