임베디드 시스템 설계 및 실험
수요일 분반

Nov 15, 2021

12주차
DMA

조교
김 준 명

공지

수업 일정

- **2~3주 동안 텀프로젝트 수행**
  - **출석체크** 합니다
- **기말고사 주 끝나고 발표 예정**
  - 교수님이 직접 실험실 각자 자리에서 검사
  - **납땜하세요**. 시연할 때 점퍼선 뽑혀서 시간 지체되면 바로 **완성도 0점** 입니다.

INS LAB
IoT NETWORK & SYSTEMS LAB

Contents

12주차 실험 내용

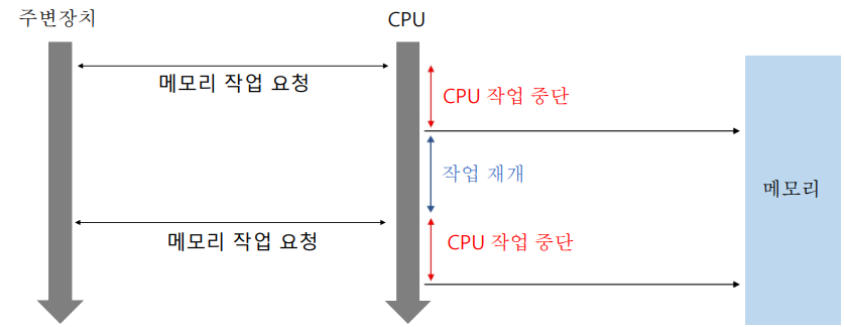
실험 목적

- DMA 동작 방법의 이해
- DMA 구현

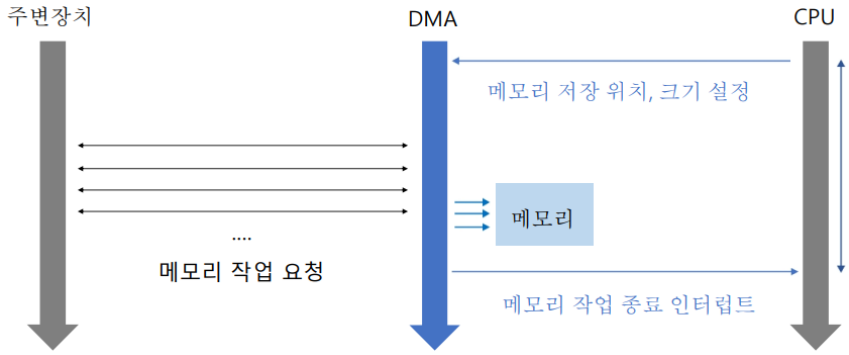
Direct Memory Access (DMA)

- 하드웨어(I/O장치, 기억장치 등)의 하위 시스템이 CPU와 **독립적으로 메모리에 직접** 접근할 수 있게 하는 기능
- DMA는 데이터 이동 완료시 **단 한번 Interrupt를 호출함**
- PIO처리시 **CPU의 시스템 메모리 접근 사이클 만큼** 성능이 향상됨

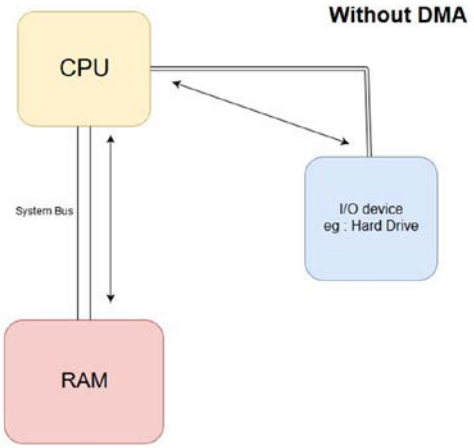
Polling, Interrupt



Direct Memory Access



Direct Memory Access (DMA)



- **일반적인 메모리 접근 방식(Programmed Input/Output)**
  - 주변장치(하드디스크, 그래픽카드 등)데이터가 항상 CPU를 거침
  - CPU를 통해 시스템 메모리에 Data가 전달됨
    - 예시

      CPU: 32bit 기반 3GHz

      I/O device와 32bit bus 연결

      4 bytes * 3 GHz = 12 GB/s (CPU Full Load 시)

- **DMA 방식**
  - RAM이 I/O 장치로부터 데이터가 필요해지면, CPU는 DMA 컨트롤러에게 신호(전송 크기, 주소 등등)를 보냄
  - DMA 컨트롤러가 RAM 주소로 데이터 bus(PCIe 등)를 통해 주고 받음
  - 모든 데이터 전송이 끝나면, DMA Controller가 CPU에게 Interrupt 신호를 보냄

버전	날짜		인코딩	데이터 전송률	대역폭				
	발표	적용			1레인 (×1)	2레인 (×2)	4레인 (×4)	8레인 (×8)	16레인 (×16)
1.0~1.1	2003년	2004년 6월	8b/10b	2.5 GT/s	250 MB/s	500 MB/s	1 GB/s	2 GB/s	4 GB/s
2.0~2.1	2007년 1월	2007년 8월	8b/10b	5 GT/s	500 MB/s	1 GB/s	2 GB/s	4 GB/s	8 GB/s
3.0~3.1	2010년 11월	2011년 7월	128b/130b	8 GT/s	984.6 MB/s	1.969 GB/s	3.938 GB/s	7.877 GB/s	15.754 GB/s
4.0	2017년 6월	2018년	128b/130b	16 GT/s	1.969 GB/s	3.938 GB/s	7.877 GB/s	15.754 GB/s	31.508 GB/s
5.0	2019년 5월	2020년	128b/130b	32 GT/s	3.938 GB/s	7.877 GB/s	15.754 GB/s	31.508 GB/s	63.015 GB/s
6.0	2021년 예정	미정	128b/130b	64 GT/s	7.877 GB/s	15.754 GB/s	31.508 GB/s	63.015 GB/s	126.031 GB/s

PCIe Table 출처: https://namu.wiki/w/PCI%20Express

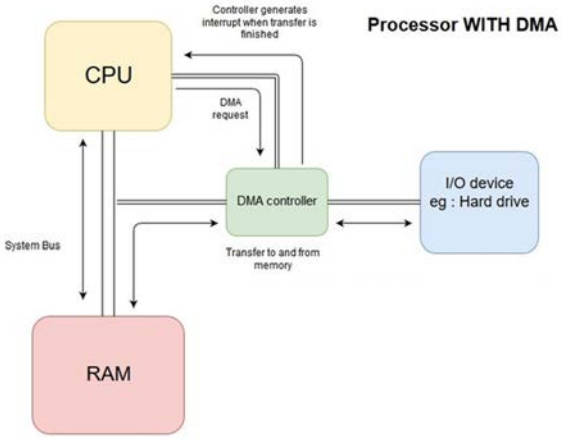


그림 출처: https://www.quora.com/What-is-the-function-of-DMA-in-a-computer

DMA Channel

- **모듈은 DMA Controller 의 DMA 채널을 통해 메모리 R/W**
- **STM32 보드 DMA 채널은 총 12 개**
  - **DMA1** 채널 7개, **DMA2** 채널 5개
- **한 DMA의 여러 채널 사이 요청은 Priority에 따라 동작**
  - DMA_CCRx 레지스터
    - Very high priority
    - High priority
    - Medium priority
    - Low priority
- **Memory-to-memory, Peripheral-to-memory, memory-to-peripheral, and peripheral-to-peripheral 전송**
- 2^{16}**까지 사용가능**

DMA Mode

- **Noncircular(Normal) Mode**
  - DMA Controller 는 데이터를 전송할 때 마다 NDT(Number of Data, DMA_CNDTRx 레지스터) 값을 감소시킴
  - NDT 는 DMA 를 통해 전송할 데이터의 총 용량을 의미하며 레지스터의 값이 0이 되면 데이터 전송 중단
  - NDT를 새로 가져오려면 채널 비활성화 필요
- **Circular Mode**
  - 주기적인 값의 전송(업데이트)이 필요할 때 사용하는 모드
  - NDT 값이 0이 될 경우 설정한 데이터 최대 크기로 재설정됨
- **Memory-to-memory Mode**
  - Peripheral의 trigger없이 작동하는 모드
  - NDT = 0 시 멈춤
  - Circular Mode와 동시 사용 불가능

DMA Controller

Connectivity line devices are STM32F105xx and STM32F107xx microcontrollers.

- **주변 장치의 Request Signal 의 발생**
  - DMA Controller 에서 우선순위 설정 및 요청에 대한 서비스 제공
  - Request / ACK 방식을 통한 주변 장치와 DMA Controller 간 통신

Figure 48. DMA block diagram in connectivity line devices

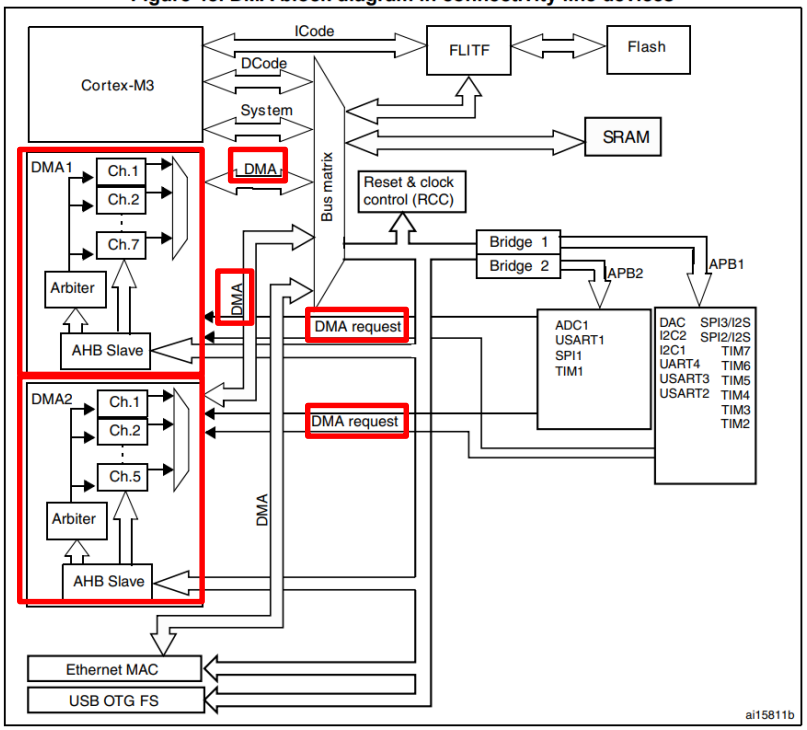
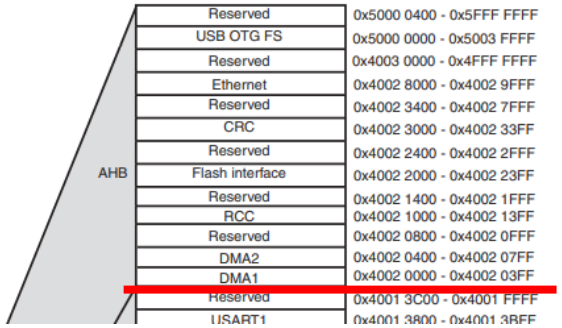


Figure 5. Memory map



- **DMA1 및 DMA2 채널**

Table 78. Summary of DMA1 requests for each channel

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
ADC1	ADC1	-	-	-	-	-	-
SPI/I^2S	-	SPI1_RX	SPI1_TX	SPI2/I2S2_RX	SPI2/I2S2_TX	-	-
USART	-	USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I^2C	-	-	-	I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1		TIM1_CH1	-	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	
TIM2	TIM2_CH3	TIM2_UP	-	-	TIM2_CH1	-	TIM2_CH2 TIM2_CH4
TIM3	-	TIM3_CH3	TIM3_CH4 TIM3_UP	-	-	TIM3_CH1 TIM3_TRIG	-
TIM4	TIM4_CH1	-	-	TIM4_CH2	TIM4_CH3	-	TIM4_UP

Table 79. Summary of DMA2 requests for each channel

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
ADC3(1)					ADC3
SPI/I2S3	SPI/I2S3_RX	SPI/I2S3_TX			
UART4			UART4_RX		UART4_TX
SDIO(1)				SDIO	
TIM5	TIM5_CH4 TIM5_TRIG	TIM5_CH3 TIM5_UP		TIM5_CH2	TIM5_CH1
TIM6/ DAC_Channel1			TIM6_UP/ DAC_Channel 1		
TIM7				TIM7_UP/ DAC_Channel 2	
TIM8	TIM8_CH3 TIM8_UP	TIM8_CH4 TIM8_TRIG TIM8_COM	TIM8_CH1		TIM8_CH2

DMA Configuration

- **Stm320f10x_dma.h**

```
typedef struct
{
  uint32_t DMA_PeripheralBaseAddr; /*!< Specifies the peripheral base address for DMAy Channelx. */

  uint32_t DMA_MemoryBaseAddr;     /*!< Specifies the memory base address for DMAy Channelx. */

  uint32_t DMA_DIR;                /*!< Specifies if the peripheral is the source or destination.
                                        This parameter can be a value of @ref DMA_data_transfer_direction */

  uint32_t DMA_BufferSize;         /*!< Specifies the buffer size, in data unit, of the specified Channel.
                                        The data unit is equal to the configuration set in DMA_PeripheralDataSize
                                        or DMA_MemoryDataSize members depending in the transfer direction. */

  uint32_t DMA_PeripheralInc;      /*!< Specifies whether the Peripheral address register is incremented or not.
                                        This parameter can be a value of @ref DMA_peripheral_incremented_mode */

  uint32_t DMA_MemoryInc;          /*!< Specifies whether the memory address register is incremented or not.
                                        This parameter can be a value of @ref DMA_memory_incremented_mode */

  uint32_t DMA_PeripheralDataSize; /*!< Specifies the Peripheral data width.
                                        This parameter can be a value of @ref DMA_peripheral_data_size */

  uint32_t DMA_MemoryDataSize;     /*!< Specifies the Memory data width.
                                        This parameter can be a value of @ref DMA_memory_data_size */

  uint32_t DMA_Mode;               /*!< Specifies the operation mode of the DMAy Channelx.
                                        This parameter can be a value of @ref DMA_circular_normal_mode.
                                        @note: The circular buffer mode cannot be used if the memory-to-memory
                                              data transfer is configured on the selected Channel */

  uint32_t DMA_Priority;           /*!< Specifies the software priority for the DMAy Channelx.
                                        This parameter can be a value of @ref DMA_priority_level */

  uint32_t DMA_M2M;                /*!< Specifies if the DMAy Channelx will be used in memory-to-memory transfer.
                                        This parameter can be a value of @ref DMA_memory_to_memory */
}DMA_InitTypeDef;
```

각 기능 및 설명은 Reference Manual - DMA 챕터 참고

실험 힌트!!!

- **ADC 설정**
  - Interrupt를 쓰지 말고 DMA를 이용해야 하므로
  - ADC_ITConfig 함수 대신 ADC_DMACmd 함수를 써야 함
- **Volatile**
  - 전역변수로 선언한 ADC 값을 저장할 공간을 항상 참조하도록 volatile 키워드 이용

```
// volatile unsigned 32bits
volatile uint32_t ADC_Value[1];
```

- **a value of type "uint32_t volatile *" cannot be assigned to an entity of type "uint32_t“**
  - 이런 종류의 에러가 뜨면 변수명 앞에 (uint32_t) 로 형 변환하세요

```
(uint32_t)&ADC_Value[0];
```

실험 주의사항

- 실험 장비들을 연결 및 분리할 때 반드시 모든 전원을 끄고 연결해주세요.
- 장비사용시 충격이 가해지지 않도록 주의해주세요.
- 자리는 항상 깔끔하게 유지하고 반드시 정리 후 퇴실해주세요.
- 실험 소스 코드와 프로젝트 폴더는 백업 후 반드시 삭제해주세요.
- 장비 관리, 뒷정리가 제대로 되지 않을 경우 해당 조에게 감점이 주어집니다.

- 동작 중 케이블 절대 뽑지말것
- 보드는 전원으로 USBPort나 어댑터(5V,1A)를 사용할것 (5V 5A 어댑터(비슷하게 생김)와 혼동하지 말 것, 사용시 보드가 타버림 -> 감점)
- 디버깅 모드 중에 보드 전원을 끄거나 연결 케이블을 분리하지 말 것!!!

- ->지켜지지 않을 시 해당 조 감점

미션 ! 별도 미션지 참고

실험 검사

반드시 DMA 사용!!! Interrupt 금지!

이번 주 실험 결과 보고서 및 소스 코드 및 실험 동작 영상

A. 이론부터 실습까지 전반적인 내용을 포함하도록 작성 (실험 과정 사진 찍으시면 좋아요)
B. 다음 실험전날 자정전까지 E-mail 제출
C. 소스 코드는 직접 작성 및 수정한 파일만 제출

나가실 때, 만드신 코드 및 프로젝트 폴더는 모두 백업하시고 삭제해주세요.
다른 분반 파일은 만지지 마시고 조교에게 알려주세요.
자리 정리정돈 안 되어 있으면 감점합니다!!!