ARM Coretex-M

펌웨어 설계

단위별 디바이스 드라이버 설계

한국기술교육대학교
온라인평생교육원

학습목표

- HAL 드라이버 중 시스템 초기화에 필요한 API에 대해 설명할 수 있다.
- HAL 시스템 드라이버에 대해 설명할 수 있다.

학습내용

- HAL 시스템 초기화 드라이버
- HAL 시스템 드라이버

HAL 시스템 초기화 드라이버

시스템 구동용 HAL API 함수

클럭

HAL API

시스템을 초기에 구동시키기 위한 HAL API 함수
제일 중요한 클럭 관련 HAL API 함수를 정리해야 함

- HAL_StatusTypeDef HAL_RCC_OscConfig (RCC_OscInitTypeDef *RCC_OscInitStruct)
  - 여러 개의 클럭 소스 (HSE, HSI, LSE, LSI, PLL)의 동작 조건을 설정하는 함수

- HAL_StatusTypeDef HAL_RCC_ClockConfig (RCC_ClkInitTypeDef *RCC_ClkInitStruct, uint32_t FLatency)
  - 시스템의 클럭 소스를 선택
  - AHB, APB1, APB2 클럭의 분주비를 설정
  - 플래시 메모리의 대기 상태 숫자를 설정
  - HCLK 클럭이 변경될 때 SysTick설정을 업데이트

HAL 시스템 초기화 드라이버

시스템 구동용 HAL API 함수

GPIO

시스템 구동 시 사용하는 GPIO용 HAL API 함수

- HAL_GPIO_Init() / HAL_GPIO_DeInit()
- HAL_GPIO_ReadPin() / HAL_GPIO_WritePin()
- HAL_GPIO_TogglePin ()

▼

더불어 GPIO 모드 (input, output, analog)와
EXTI (외부 인터럽트)용으로 쓸 것인지 설정 가능

- HAL_GPIO_Init() / HAL_GPIO_DeInit()
- HAL_GPIO_ReadPin() / HAL_GPIO_WritePin()
- HAL_GPIO_TogglePin ()

▼

외부 인터럽트 용도로 쓰려면 stm32f4xx_it.c의
HAL_GPIO_EXTI_IRQHandler() 에서 호출하는
HAL_GPIO_EXTI_Callback() 콜백함수를 사용자가 구현

HAL 시스템 초기화 드라이버

시스템 구동용 HAL API 함수

NVIC와 SysTick 타이머

NVIC와 SysTick 타이머 구동을 위한 API 함수

- HAL_NVIC_SetPriorityGrouping()
- HAL_NVIC_SetPriority()
- HAL_NVIC_EnableIRQ()/HAL_NVIC_DisableIRQ()
- HAL_NVIC_SystemReset()
- HAL_NVIC_GetPendingIRQ() / HAL_NVIC_SetPendingIRQ () /
- HAL_NVIC_ClearPendingIRQ()
- HAL_SYSTICK_Config()
- HAL_SYSTICK_CLKSourceConfig()

함수 이름을 통해 용도는 직관적으로 이해 가능

HAL 시스템 초기화 드라이버

시스템 구동용 HAL API 함수

Power

1. 전원 관련된 HAL API 함수는 HAL PWR Driver로 전원 관리를 위해 사용

2. 모든 STM32시리즈는 다음과 같은 HAL PWR 함수를 가짐

PVD 설정, 활성화/비활성화 및 인터럽트 제어

- HAL_PWR_PVDConfig()
- HAL_PWR_EnablePVD() / HAL_PWR_DisablePVD()
- HAL_PWR_PVD_IRQHandler()
- HAL_PWR_PVDCallback()

Wakeup 핀 설정

- HAL_PWR_EnableWakeUpPin()
  / HAL_PWR_DisableWakeUpPin()

저전력 모드 진입

- HAL_PWR_EnterSLEEPMode()
- HAL_PWR_EnterSTOPMode()
- HAL_PWR_EnterSTANDBYMode()

HAL 시스템 초기화 드라이버

시스템 구동용 HAL API 함수

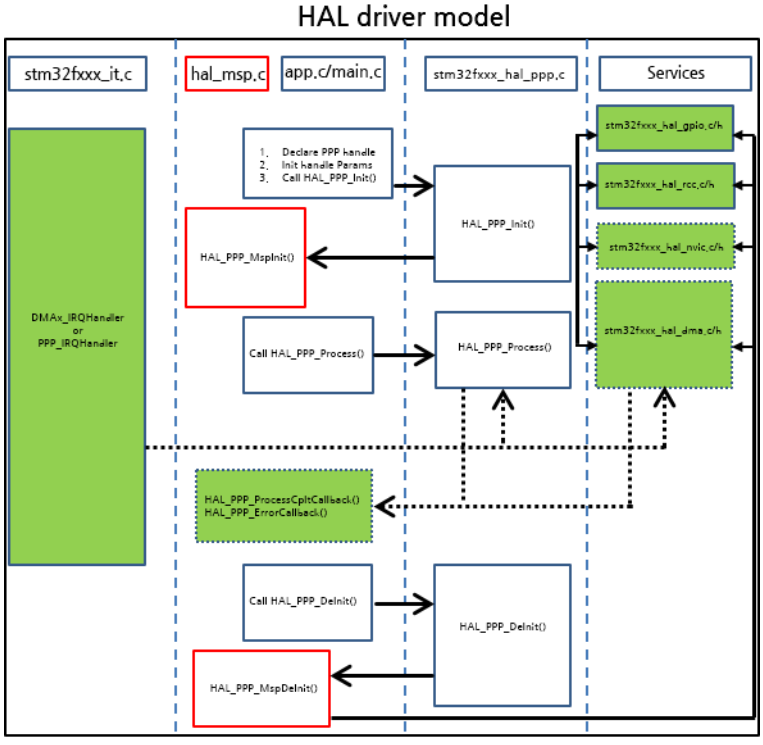
외부 인터럽트 (EXTI)

- 외부 인터럽트는 주변장치가 사용하는 서비스로 여러 개의 EXTI가 각각 설정이 가능하도록 되어있음
- GPIO를 통해 총 16개의 EXTI 라인이 연결되어 있고 GPIO 드라이버에 의해 관리됨
- GPIO_InitTypeDef 구조체에 의해 GPIO로 설정할지 외부 인터럽트로 설정할지 외부 이벤트로 설정할지 결정됨
- EXTI 인터럽트 모드로 설정되며 사용자 프로그램에서 stm32f4xx_it.c 에 존재하는 HAL_PPP_FUNCTION_IRQHandler()가 호출되며 사용자가 필요한 기능은 HAL_PPP_FUNCTIONCallback() 콜백 함수에 구현함
- 예를 들어 HAL_PWR_PVDCallback()와 같은 콜백 함수를 사용함

HAL 시스템 초기화 드라이버

HAL 드라이버 사용법

HAL 사용 모델

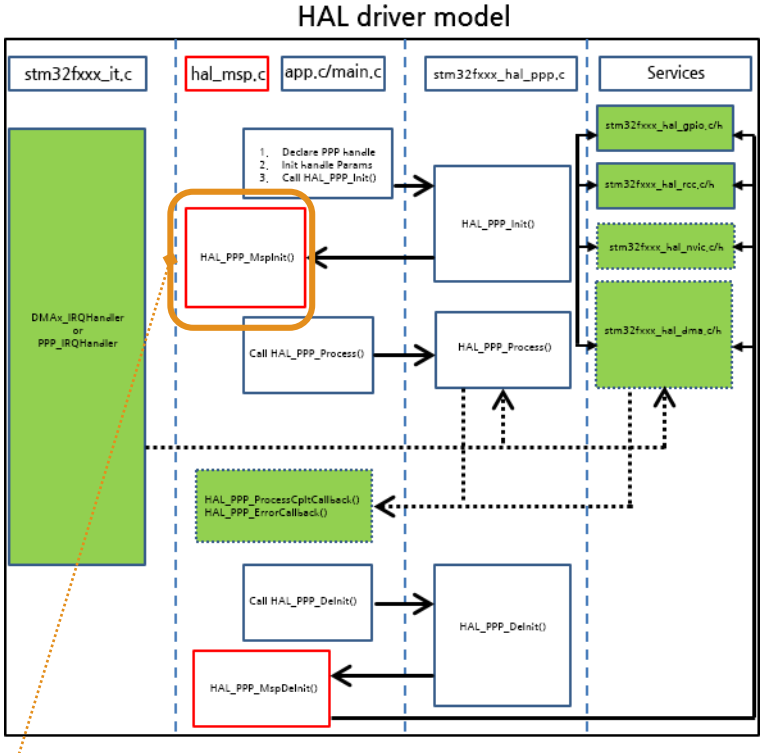


- 사용자 응용 프로그램, HAL 드라이버와 인터럽트 간의 상호 관계와 HAL 드라이버의 전형적인 사용 모델
- 기본적으로 HAL 드라이버 API는 사용자 파일들에서 호출되거나 DMA나 사용하는 장치의 인터럽트에서 기인한 인터럽트 핸들러에서 호출

HAL 시스템 초기화 드라이버

HAL 드라이버 사용법

HAL 사용 모델



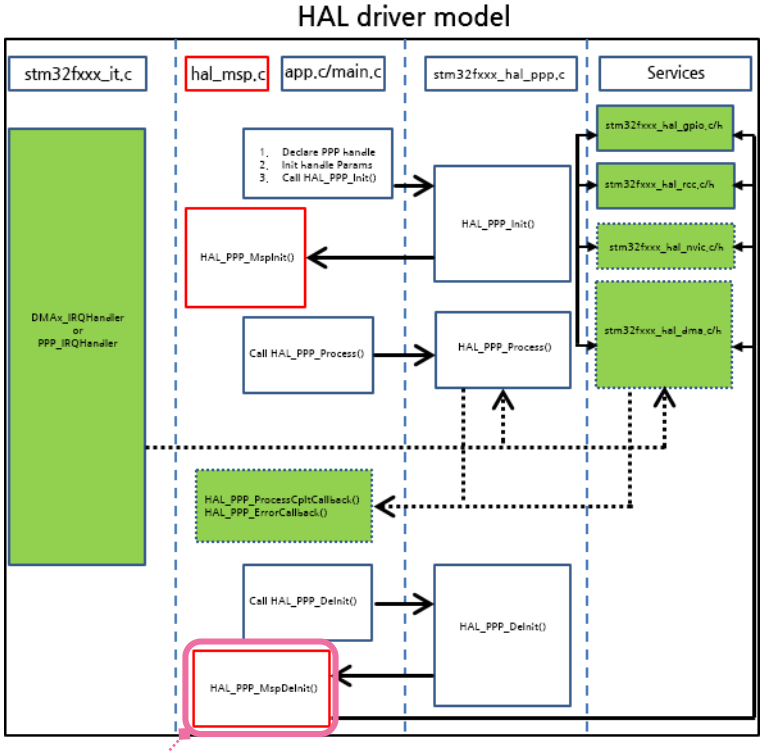
HAL_PPP_MspInit()

···› 결과적으로 사용자가 필요한 기능을 구현할 때 초기화에 필요한 내용

HAL 시스템 초기화 드라이버

HAL 드라이버 사용법

HAL 사용 모델



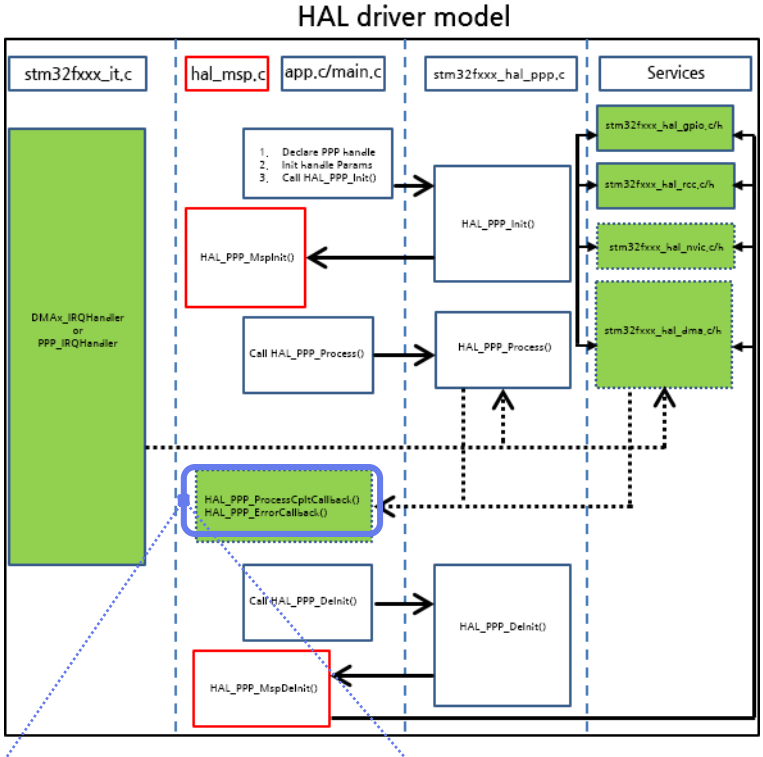
HAL_PPP_MspDeInit()

⋯› 구현하고 초기화를 해제할 때 필요한 기능

HAL 시스템 초기화 드라이버

HAL 드라이버 사용법

HAL 사용 모델



HAL_PPP_ProcessCpltCallback()

··› 인터럽트 활성화 때 필요한 기능

HAL_PPP_ErrorCallback()

··› 에러 발생시 구현할 기능

HAL 시스템 초기화 드라이버

HAL 드라이버 사용법

HAL 초기화

HAL 코어를 초기화하기 위한 함수

HAL_Init()

- 시스템 시작 시 반드시 호출되어야 함
- 수행하는 동작은 다음과 같음
- 데이터 캐시와 명령어 캐시 초기화
- Systick 타이머를 1ms마다 인터럽트가 발생하도록 설정
- 클럭, GPIO, DMA, 인터럽트 등의 시스템 레벨 초기화
- HAL_MspInit() 함수 호출

HAL_DeInit()

- 초기화를 해제하는 기능
- 모든 주변 장치 리셋
- HAL_MspDeInit() 함수 호출

HAL_GetTick()

- 현재의 SysTick 카운터의 값을 읽어옴

HAL_Delay()

- SysTick 타이머를 사용하여 시간지연을 구현하는 함수로 단위는 msec

HAL 시스템 초기화 드라이버

HAL 드라이버 사용법

시스템 클럭 초기화

플래시 메모리

1

시스템 클럭 설정은
사용자 코드의
시작부분에 해야함

2

사용자가 사용 조건에 따라
클럭 설정을
수정할 수 있음

```
static void SystemClock_Config(void)
{
RCC_ClkInitTypeDef RCC_ClkInitStruct;
RCC_OscInitTypeDef RCC_OscInitStruct;
/* Enable HSE Oscillator and activate PLL with HSE as source */
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
RCC_OscInitStruct.HSEState = RCC_HSE_ON;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
RCC_OscInitStruct.PLL.PLLM = 25;
RCC_OscInitStruct.PLL.PLLN = 336;
RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
RCC_OscInitStruct.PLL.PLLQ = 7;
HAL_RCC_OscConfig(&RCC_OscInitStruct);
/* Select PLL as system clock source and configure the HCLK, PCLK1 and PCLK2 clocks
dividers */
RCC_ClkInitStruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK |
RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5); }
```

시스템 클럭 설정의 예

HAL 시스템 초기화 드라이버

HAL 드라이버 사용법

HAL의 입출력 프로세스

HAL 함수가 데이터를 입출력하는 모드 3가지

1	2	3
폴링 (Polling) 모드	인터럽트 (Interrupt) 모드	DMA 모드

! UART를 사용하여 데이터를 입출력할 때 CubeMX를 통해 3가지 모드 중 하나 선택

1	2	3
폴링 (Polling) 모드	인터럽트 (Interrupt) 모드	DMA 모드
아무것도 선택하지 않은 기본 상태	NVIC를 통해 인터럽트를 활성화한 상태	DMA를 활성화하여 UART를 사용한 상태

! 3가지 설정은 모두 CubeMX를 통해 설정 가능

HAL 시스템 초기화 드라이버

HAL 드라이버 사용법

폴링(Polling) 모드

- 폴링 모드에서 기본적으로 블로킹 모드이며 작업이 완료되면 결과가 리턴됨
  - 블로킹 모드는 그 함수의 수행이 끝날 때까지 계속 멈춰있다는 의미
  - 인터럽트나 DMA 모드와는 달리 해당 HAL 함수는 타임 아웃(Timeout)으로
    지정된 값 이상의 시간이 지나지 않으면 계속 해당 함수를 수행함

- 성공적으로 완료되면 HAL_OK 값을 리턴함
- 에러가 발생하면 에러의 상태 값을 반환함
- HAL 함수 내에서의 무한 대기 방지를 위해 타임 아웃 값을 사용함

전형적인 폴링 모드 HAL API 함수

- HAL_StatusTypeDef HAL_PPP_Transmit
  (PPP_HandleTypeDef * phandle,
  uint8_t pData,int16_t Size,uint32_t Timeout)
  - 마지막 매개 변수
  - msec 단위

HAL 시스템 초기화 드라이버

HAL 드라이버 사용법

인터럽트(Interrupt) 모드

- 인터럽트 모드는 인터럽트를 사용하는 모드
- 인터럽트가 발생하면 인터럽트 핸들러에서 호출된 콜백 함수가 호출되어 실행됨

HAL_PPP_Process_IT() — 프로세스를 시작

HAL_PPP_IRQHandler() 핸들러 — PPP장치의 인터럽트 핸들러

__weak HAL_PPP_ProcessCpltCallback () — 인터럽트 핸들러가 호출하는 콜백 함수

__weak HAL_PPP_ProcessErrorCallback() — 수행 시 에러가 발생하면 호출되는 에러 콜백 함수

인터럽트 모드의 4개의 함수

사용자가 구현하고자 하는 기능을 코딩

HAL 시스템 초기화 드라이버

HAL 드라이버 사용법

인터럽트(Interrupt) 모드

인터럽트 모드의 전형적인 사용의 예

```
UART_HandleTypeDef UartHandle;

int main(void)
{
                    /* Set User Parameters */
                    UartHandle.Init.BaudRate = 9600;
                    UartHandle.Init.WordLength = UART_DATABITS_8;
….
                    UartHandle.Init.Instance = USART3;
                    HAL_UART_Init(&UartHandle);
                    while (1);
}
void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)
{
 // 원하는 작업을 여기에 코딩
}
void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart)
{
}
```

main.c

- UART를 인터럽트 모드로 사용한다면 main.c에 UART 핸들러와 HAL_UART_TxCpltCallback와 같은 콜백 함수를 구현

```
extern  UART_HandleTypeDef UartHandle;

void USART3_IRQHandler(void)
{
  HAL_UART_IRQHandler(&UartHandle);
}
```

stm32f4xx_it.c

- stm32f4xx_it.c파일에 UART 인터럽트 핸들러를 호출
- stm32f4xx_it.c파일에 UART 인터럽트 핸들러는 CubeMX를 통해 자동 생성됨

HAL 시스템 초기화 드라이버

HAL 드라이버 사용법

DMA 모드

- DMA는 CPU를 대신하여 정해진 일을 수행하는 하드웨어 블록
- DMA 모드는 DMA를 사용한 주변장치 사용을 위해 DMA 인터럽트를 사용

HAL_PPP_Process_DMA() — 프로세스를 시작

HAL_PPP_DMA_IRQHandler() — PPP장치의 DMA 인터럽트 핸들러

__weak HAL_PPP_ProcessCpltCallback () — 인터럽트 핸들러가 호출하는 콜백 함수

__weak HAL_PPP_ProcessErrorCallback() — 수행 시 에러가 발생하면 호출되는 에러 콜백 함수

DMA 모드의 4개의 함수

사용자가 구현하고자 하는 기능을 코딩

HAL 시스템 초기화 드라이버

HAL 드라이버 사용법

DMA 모드

```
int main(void)
{
                    /* Set User Parameters */
                    UartHandle.Init.BaudRate = 9600;
                    UartHandle.Init.WordLength = UART_DATABITS_8;
....
                    UartHandle.Init.Instance = USART3;
                    HAL_UART_Init(&UartHandle);
                    HAL_UART_Send_DMA(&UartHandle, TxBuffer,
sizeof(TxBuffer));
                    while (1);
}
void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)
{
}
void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart)
{
}
```

main.c

- UART를 DMA 인터럽트 모드로 사용한다면 CubeMX에서 DMA를 활성화

```
extern UART_HandleTypeDef UartHandle;
void DMA1_Stream3_IRQHandler(void)
{
  HAL_DMA_IRQHandler(&hdma_usart3_tx);
}
```

stm32f4xx_it.c

```
HAL_UART_Transmit_DMA (UART_HandleTypeDef *huart, uint8_t *pData,
uint16_t Size)
{
(...)
huart->hdmatx->XferCpltCallback = UART_DMATransmitCplt;
huart->hdmatx->XferHalfCpltCallback = UART_DMATxHalfCplt;
(...)
}
```

stm32f4xx_hal_uart.c

- 인터럽트 모드와 비슷하게 콜백 함수를 호출하여 구현

HAL 시스템 드라이버

HAL 초기화 및 해제 함수

시스템 전체를 초기화하거나 초기화를 해제하는 함수

stm32f4xx_hal.c 파일에 소스가 존재

HAL_Init()

- 플래시 메모리의 인터페이스, NVIC 할당 및 클럭 동작 설정 등을 초기화
- 리셋 후에 주변 장치를 클럭 설정하기 이전에 이 함수가 먼저 호출되어야 함
- 즉, main 함수에서 가장 먼저 호출되어야 함

HAL_DeInit()

- 모든 주변 장치를 리셋

HAL_MspInit()

- MSP(MCU Specific Package)를 초기화하는 함수

HAL 시스템 드라이버

HAL 초기화 및 해제 함수

시스템 전체를 초기화하거나 초기화를 해제하는 함수

stm32f4xx_hal.c 파일에 소스가 존재

HAL_MspDeInit()

- MSP 초기화를 해제하는 함수

HAL_InitTick()

- SysTick 타이머를 초기화하는 함수
- 초기화하면 SysTick 인터럽트가 1msec마다 발생함

HAL 시스템 드라이버

HAL 제어용 함수

HAL 제어용 함수

HAL_IncTick (void)	• 전역 변수 “uwTick”을 증가시키는 함수
uint32_t HAL_GetTick (void)	• 현재의 Tick값을 반환하는 함수, 단위는 msec
HAL_Delay (__IO uint32_t Delay)	• 시간 지연을 하는 함수, 단위는 msec
HAL_SuspendTick (void)	• SysTick 타이머 인터럽트 발생을 유보시키는 함수
HAL_ResumeTick (void)	• SysTick 타이머 인터럽트 발생을 재개시키는 함수
uint32_t HAL_GetHalVersion (void)	• HAL API Driver의 버전을 읽어오는 함수
uint32_t HAL_GetREVID (void)	• 디바이스의 revision ID를 읽어오는 함수
uint32_t HAL_GetDEVID (void)	• 디바이스의 ID를 읽어오는 함수

요점노트

1. HAL 시스템 초기화 드라이버

- HAL 시스템 초기화 드라이버
  - 초기 시스템구동을 위해 시스템 구동용 HAL API 함수를 사용함
  - 시스템 구동용 HAL API 함수는 클럭, GPIO, NVIC, Power, 외부 인터럽트 등을 초기화함
  - HAL 사용 모드에서 HAL 드라이버 API는 사용자 파일들에서 호출되거나 DMA나 인터럽트에서 호출됨
  - DMA모드는 DMA를 사용한 주변장치 사용을 위해 DMA인터럽트를 사용함

요점노트

2. HAL 시스템 드라이버

- HAL 시스템 드라이버
  - HAL 시스템 드라이버는 초기화 및 해제함수, 제어용 함수로 구성
  - HAL 시스템 초기화 및 해제 함수는 시스템 전체를 초기화하거나 초기화를 해제하는 함수
  - HAL 제어용 함수는 시간을 제어하거나 MCU의 ID 등을 읽어오는 함수