STM32CubeIDE를 이용한 STM32 따라하기, 백상호 (250107)

CH.03.1 LED Blink (p.76 ~ p.99)

실습

보드선택

1. **STM32CubeIDE 실행**
   - `[File] -> [New] -> [STM32 Project]`
2. **Board Selector 탭**
   - 카테고리: `{Type -> Nucleo64, MCU -> STM32F4}`
   - `Next -> 프로젝트 이름 설정 -> Yes -> Yes`

HSE LSE 비활성

1. **LED_Blink.ioc 파일**
   - `Pinout & Configuration` 탭
2. **Category 탭**
   - `System core -> RCC -> Disable -> HSE -> Disable`

Clock Configuration 설정 확인

1. **LED_Blink.ioc 파일**
   - `Clock Configuration` 탭
     - 이 탭 전체가 System clock Mux 역할인 듯
2. **클럭소스 비활성화 확인**
   - 앞 단계에서 클럭소스를 비활성화했으므로, PLLCLK이 활성화되어 HCLK가 64MHz로 설정됨

- **STM32F103 시리즈**:
  - 최대 HCLK는 72MHz
  - 내부 클럭으로는 64MHz가 최대

코드생성

1. **코드 생성**
   - `Project -> Generate Code`
2. **디렉토리 구조**

디렉토리 구조

```
LED_Blink.
├ Core
│  ├ Inc
│  ├ Src
│  ├ Startup
├ Drivers
│  ├ CMSIS
│  │  ├ Device
│  │  │  ├ ST
│  │  │  │  ├ STM32F1xx
│  │  │  │  │  ├ Include
│  │  │  │  │  ├ Source
│  │  │  │  │  │  ├ Templates
│  │  ├ Include
│  ├ STM32F1xx_HAL_Driver
│  │  ├ Inc
│  │  │  ├ Legacy
│  │  ├ Src
LED_Blink $
```

1. **함수**
   - `Drivers -> STM32F1xx_HAL_Driver -> Src`
     - **`stm32f1xx_hal_gpio.c`**

        - `HAL_GPIO_TogglePin()`: 해당 포트의 핀 출력을 토글
    - `stm32f1xx_hal.c`
        - `HAL_Delay()`: ms 단위로 딜레이를 줌

2. **파일 구조**

- `Inc/main.h`:
    - LED 포트, 핀 번호 등이 선언되어 있음
- `Src/main.c`:

```
/* USER CODE BEGIN WHILE */
HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin); // 100번째 라인에서 시작
HAL_Delay(500);
/* USER CODE END WHILE */
```

- **주의**:
    - CubeMX에서 생성한:

```
/* USER CODE BEGIN WHILE */
/* USER CODE END WHILE */
```

    - 주석 사이에 사용자 코드를 위치시켜야 `Generate Code`를 다시 실행했을 때 사용자 코드가 사라지지 않음

빌드

1. `Project -> Build All`
2. **툴체인 미설치 에러 해결**

- `Window -> Preferences -> STM32Cube -> Toolchain`
- 툴체인 다운로드 링크

런

1. `Run -> Run` (자동 빌드)

디버깅

1. `Run -> Debug -> switch` (자동 빌드 -> 디버깅 모드 진입)
2. **브레이크 포인트 지정**
    - 라인넘버 위에 마우스 호버 → 오른쪽 클릭 → 메뉴에서 선택

용어정리

----------구분선 - HSE LSE 비활성

serial wire 3핀 (SWO, TCK, TMS)

- 직렬 와이어 / SWD / Serial Wire Debug
- ARM Cortex-M MCU에서
  ARM의 Debug Access Port(DAP)의 일부로 사용되는
  디버그 프로토콜입니다.
- 기존 JTAG 인터페이스보다 핀 수가 적어서,
  공간 제약이 있는 설계에 이상적

SWO, TCK, TMS

- 최소한의 디버그 인터페이스의 핵심을 형성하여
  MCU에 대한 효율적인 프로그램 및 디버그 제어를 수행

SWO

- 직렬 와이어 출력
- 실시간 디버깅 및 추적 출력에 사용
- printf 스타일 디버깅 정보, 카운터, 기타 런타임 데이터, 등의 데이터를 제공
- 일반적으로 ST-Link 및 와 같은 도구와 함께 사용되어 추적 로깅을 수행
  IDE(예: Keil, STM32CubeIDE)

TCK

- 테스트 시계
- Test Clock
- 디버그 인터페이스에서 동기화를 위한 클록 신호를 제공

- 디버거와 MCU 간의 데이터 전송이 적절한 타이밍으로 이루어지도록 보장

TMS

- 테스트 모드 선택
- Test Mode Select
- 디버그 모드를 제어하는 데 사용됩니다
  (예: SWD와 JTAG 프로토콜 중에서 선택)
- 디버그 시스템의 동작을 안내하는 구성라인 역할

USART2 2핀(RX/TX)

- Universal
- Synchronous
- Asynchronous
- Receiver
- Transmitter
- 장치 간 데이터 교환에 사용되는 직렬 통신 프로토콜
- USART2의 경우 기본 비동기 통신에 두 개의 핀만 필요

RX(수신)

- 이 핀은 다른 장치로부터 직렬 데이터를 수신합니다.
- 이는 송신장치의 TX 핀에 연결됩니다.

TX(송신)

- 이 핀은 직렬 데이터를 다른 장치로 전송합니다.
- 수신 장치의 RX 핀에 연결됩니다.

RX/TX 통신의 주요 특징:

- 전이중 통신
  Full-Duplex Communication
  RX와 TX 라인이 동시에 작동하여 양방향 데이터 교환이 가능합니다.
- 통신 속도
  baud rate에 의해 정의됨

예: 9600bps, 115200bps

응용 프로그램

- 주변장치(예: GPS, 블루투스 모듈, 센서)와 통신
- PuTTY나 RealTerm과 같은 직렬 모니터를 통한 디버깅.
- 데이터 로깅 또는 시스템 피드백

크리스탈 = 전자수정

- Electronic Crystal
- 석영을 특정방식으로 절단하여 모양을 잡고, 전기적 진동을 가하면 "정확한 주파수"로 진동함

기능

- 타이밍과 동기화를 위한 안정적인 클록 신호를 제공
- MCU와 같은 장치의 발진기(oscillator) 회로와 함께 사용됨

압전 효과

- Piezoelectric Effect
- 수정에 전압을 가하면 일정한 주파수로 진동
- 이 진동은 클록 소스 역할

RCC

- Reset and Clock Control
- 시스템 주변장치에 대한 클록 및 리셋을 구성하고 제어
- Configures and controls clocks and resets for system peripherals

특징

- 전력을 절약하기 위해 주변 장치 클럭을 활성화하거나 비활성화합니다.
- 고속 외부(HSE), 고속 내부(HSI), 저속 외부(LSE) 등과 같은 클럭 소스를 구성합니다.
- 시스템 재설정 및 클럭 주파수 조정에 대한 제어를 제공합니다.
- **시스템 클록(예: HCLK, PCLK1, PCLK2)과 해당 소스를 구성합니다.**

HSE(RCC_OSC_IN / RCC_OSC_OUT)

- High-Speed External
- 외부 고주파파 크리스탈 발진기
- 클록 소스
- MCU에 연결된, 정밀한 시스템 클록 생성을 위한 고주파 수정/발진기
  High-frequency crystal/oscillator for precise system clock generation
- 일반적으로 MCU에 연결된 외부 크리스털 오실레이터를 포함

특징

- 일반적으로 4MHz~25MHz 범위에서 작동
- 정확하고 안정적인 클록 소스를 제공하므로 타이밍이 중요한 애플리케이션에 적합
- 종종 정밀한 시스템 클록 생성에 사용되거나 더 높은 클록 속도를 달성하기 위한,
  PLL(위상 고정 루프)의 소스로 사용됨

RCC_OSC_IN / RCC_OSC_OUT

- 외부 크리스털을 MCU에 연결하는 데 사용되는 핀
- 둘 사이에 크리스털이 연결됨

RCC_OSC_IN

- 고속 외부 클록의 입력 핀
- 외부 수정 발진기 또는 외부 클록 소스의 한 단자에 연결
- 일부 문서에서는 XTAL_IN 또는 OSC_IN 핀 이라고도 함

RCC_OSC_OUT

- 고속 외부 클록의 출력 핀 입니다.
- 수정 발진기의 다른 단자에 연결합니다.
- 진동이나 안정화를 유지하기 위해 수정에 피드백을 제공

커패시터

- 진동을 안정화하기 위해 종종 "커패시터"가 이러한 핀과 접지 사이에 배치
- 이러한 커패시터의 값은 크리스털의 사양에 따라 달라짐

외부 클록소스

- 크리스털 대신 외부 클록 생성기가 RCC_OSC_IN 핀을 직접 구동할 수 있음
- 이 경우, RCC_OSC_OUT 핀은 연결하지 않은 채로 두거나 테스트/디버깅에 사용할 수 있음

LSE(RCC_OS32_IN / RCC_OSC32_OUT)

- Low-Speed External
- 외부 저주파 크리스털 발진기
- 클록 소스
- RTC 및 저전력 타이밍 요구 사항을 위한 저주파 크리스털(32.768 kHz).
  Low-frequency crystal (32.768 kHz) for RTC and low-power timing needs

특징

- 주로 실시간 클록(RTC) 애플리케이션과 저전력 모드에 사용됨
- 시간 유지 기능을 위해 저주파, 저전력 및 안정적인 클로킹을 제공

LSE 핀

- **RCC_OSC32_IN**
  - 저속 외부 클록의 입력 핀 입니다.
  - 32.768 kHz 크리스털의 한 단자에 연결
- **RCC_OSC32_OUT**
  - 저속 외부 클록의 출력 핀 입니다.
  - 크리스털의 다른 단자에 연결합니다.

GPIO

- General Purpose I/O
- flexible interface
- GPIO 핀은
  센서, LED, 스위치 또는 기타 주변 장치와 같은 외부 장치와
  상호 작용하는 데 사용됩니다.
  이들은 매우 configurable하며 애플리케이션에 따라 입력/출력으로 기능할 수 있음

주요 기능

- 방향설정 가능
  - 입력 모드: GPIO 핀이 외부 장치(예: 버튼이나 센서)의 상태를 읽음
  - 출력 모드: GPIO 핀이 외부 장치를 제어하기 위한 신호를 보냄
- 논리 레벨: 높음 / 낮음
  - 높음(1): 일반적으로 공급 전압 근처의 전압(예: 3.3V 또는 5V)을 나타냄
  - 낮음(0): 0V(접지) 근처의 전압을 나타냄

PA5

- 포트 / A / 핀5

LD2

- LED2
- 초록색임
- 스펙에 따르면, 사용자 LED

B1

- 버튼1

----------구분선 - Clock Configuration 설정확인

System clock Mux

- **SYSCLK MUX**
- **System Clock Multiplexer**
  - MCU의 주 클록 소스를 선택할 수 있는 멀티플렉서
  - 선택된 클록은 시스템 클록이 됩니다.
    - CPU, 버스, 주변 장치를 포함한 나머지 시스템을 구동하는 데 쓰임

SYSCLK

- 시스템 클록

PLL

- **Phase-Locked Loop** (위상 잠금 루프)
- STM32 MCU의 하드웨어 모듈

수행하는 기능

- 입력 클록 주파수를 곱합니다.
- `configurable` 요소를 사용하여 출력 주파수를 나눔

인풋

- **HSI**
  high speed Internal oscillator
- **HSE**
  high speed External oscillator
- **MSI**
  Multi-speed internal oscillator

아웃풋

- **PLLCLK**
  SYSCLK MUX를 통해 선택된 경우 시스템 클록으로 사용됨
- **PLL48CLK**
  48MHz 클록을 필요로 하는 주변 장치(예: USB, RNG)에 사용됨

PLL의 일반적인 사용

- CPU에 더 높은 주파수 클록을 생성합니다(예: 72MHz, 100MHz, 216MHz).
- 주변 장치 요구 사항에 맞게 주파수를 조정합니다.

PLLCLK

- **Phase-Locked Loop Clock** (페이즈 잠금 루프 클록)

HCLK

- **High-Speed Clock**
- AHB를 구동하는 클록 신호
- CPU 코어의 작동 주파수를 결정

CPU 클록

- CPU 코어의 작동 주파수

AHB

- **Advanced High-Performance Bus**
- CPU, 메모리, 고속 주변 장치를 연결함

----------구분선 - 코드생성

HAL

- **Hardware Abstraction Layer**