**X-CUBE-SBSFU V2.6.1 NUCLEO-F429ZI포팅 절차ver.1**

2022-09-27 이지훈

(https://blog.naver.com/eziya76)

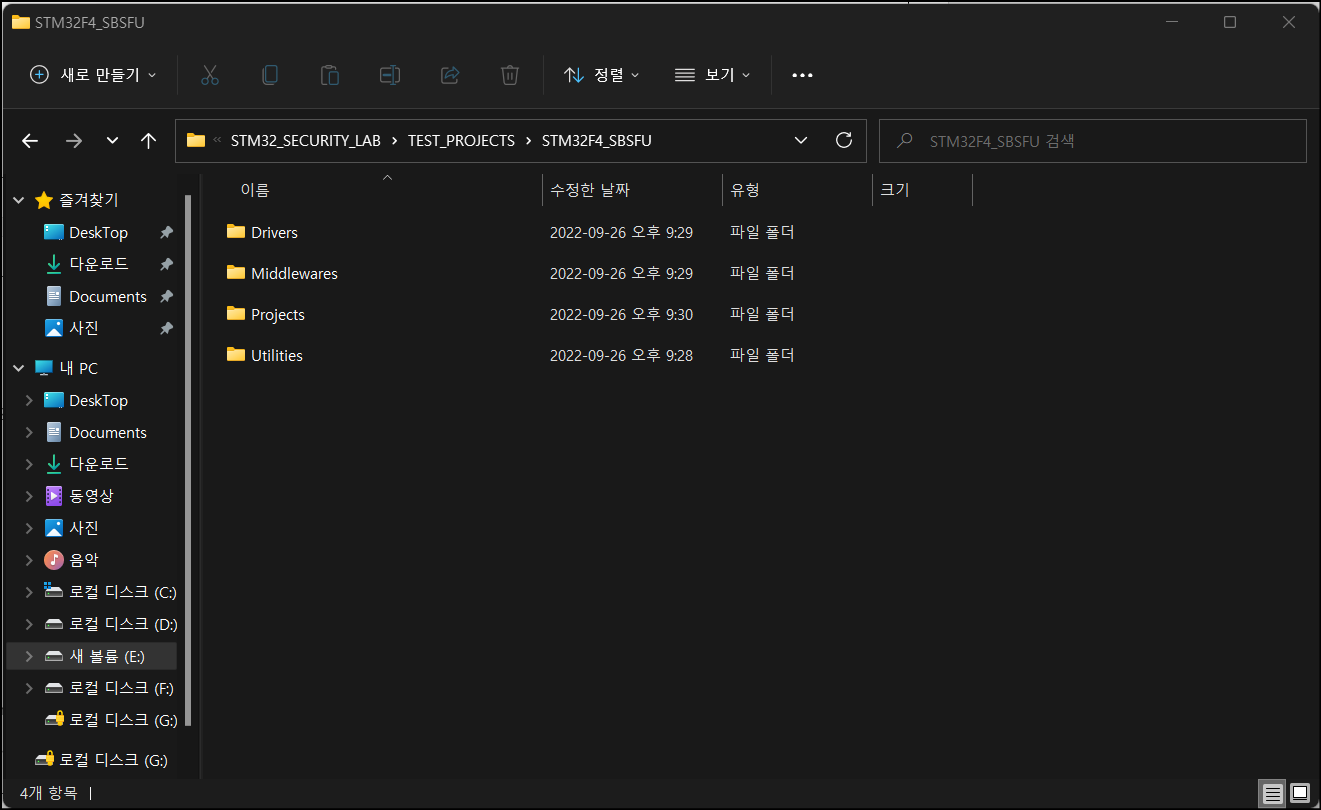
* 본 문서는 STM32F413H-Discovery 프로젝트를 NUCLEO-F429ZI (STM32F429ZITx) 프로젝트로 포팅하는 내용을 기술하고 있습니다.
* 본 문서는 개인적으로 작업한 내용을 정리하는 용도이며 다른 MCU 포팅에는 적합하지 않거나 오류를 포함할 수 있습니다.
* 참고 용도로만 사용하시기 바라며 다른 MCU 로 포팅 시에는 이 문서의 내용과 상이할 수 있습니다.

1. 준비물 및 필수 문서

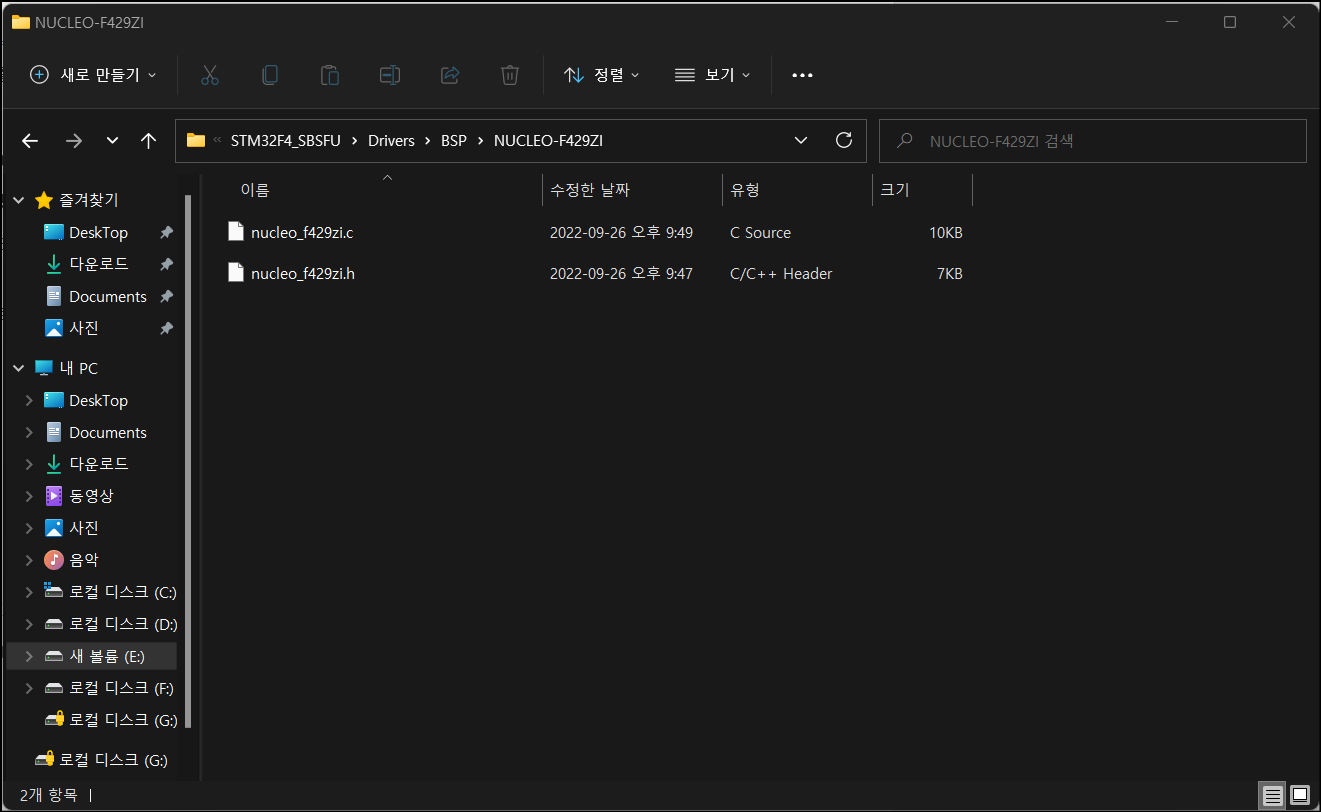
* X-CUBE-SBSFU V2.6.1
* STM32CubeIDE V1.10.1
* STM32CubeProgrammer V2.11.0
* TeraTerm V4.106
* Notepad++
* AN5056: Integration guide for the X-CUBE-SBSFU STM32Cube Expansion Package
* UM2262: Getting started with the X-CUBE-SBSFU STM32Cube Expansion Package
* Github: https://github.com/eziya/STM32\_SECURITY\_LAB/tree/main/TEST\_PROJECTS/STM32F4\_SBSFU

1. 프로젝트 폴더 구성
   1. 패키지 파일 압축 해제

* STM32CubeExansion\_SBSFU\_V2.6.1 파일을 압축해제 한다. (폴더명은 STM32F4\_SBSFU 로 설정)

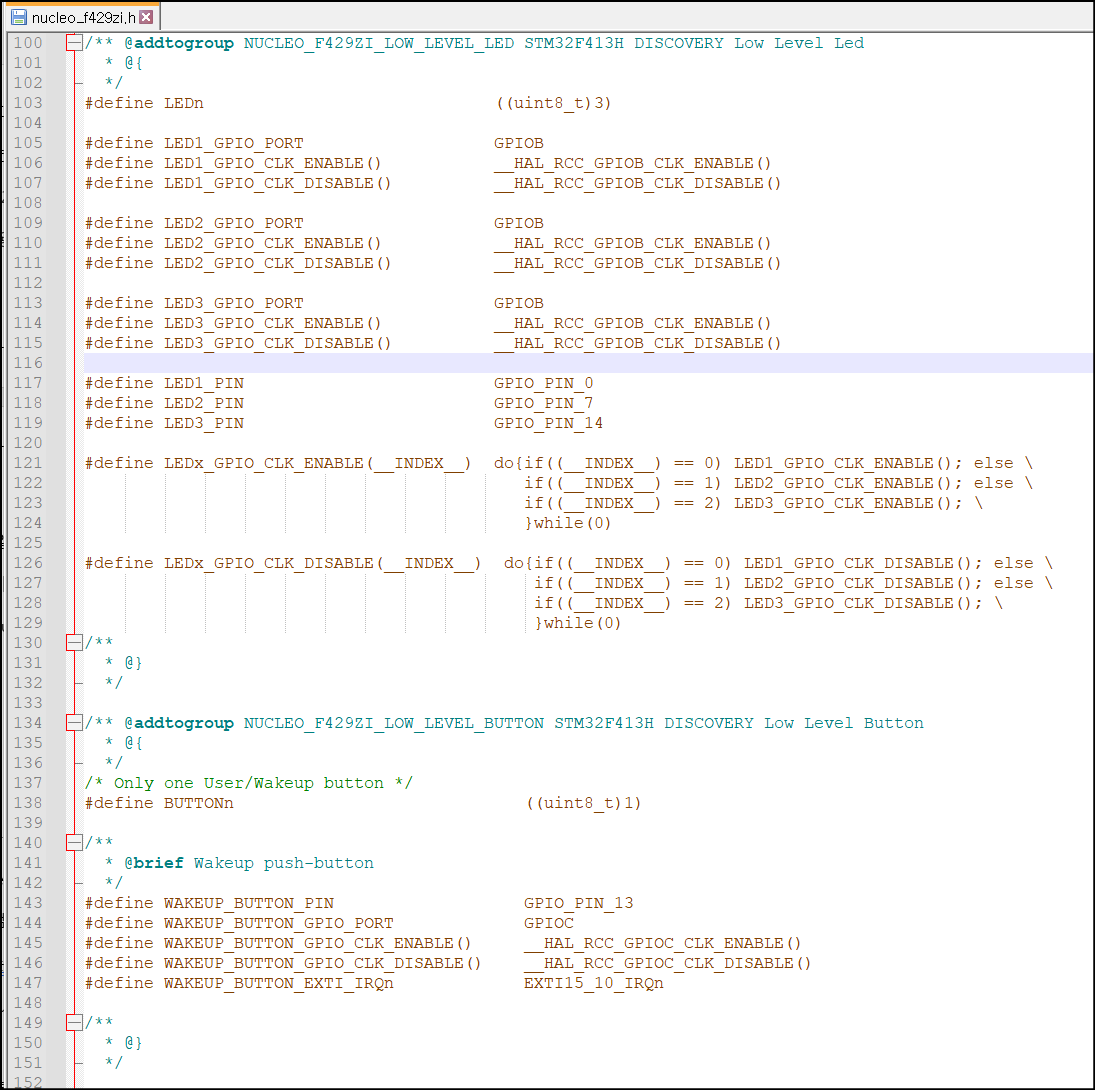


* 1. 불필요 파일 삭제
* STM32F413H-Discovery 프로젝트를 기본 프로젝트로 사용하며 다른 프로젝트는 사용하지 않기 때문에 불필요한 폴더를 삭제한다.
* /\_htmresc 폴더 삭제
* /Drivers/BSP 폴더에서 Components 폴더와 STM32F413H-Discovery 폴더를 제외한 나머지 폴더 삭제
* /Drivers/CMSIS/docs 폴더 삭제
* /Drivers 폴더에서 STM32F4xx\_HAL\_Driver 를 제외한 나머지 드라이버 폴더 삭제
* /Middlewares/ST/STM32\_Cryptographic 폴더와 /Middlewares/ST/STM32\_Secure\_Engine 폴더를 제외한 나머지 /Middlewares 하위 폴더 삭제
* /Projects/ 폴더에서 STM32F413H-Discovery를 제외한 나머지 프로젝트 폴더 삭제
* /Projects/STM32F413H-Discovery/Applications 내에서 2\_Images 폴더를 제외한 나머지 폴더 삭제
  1. 프로젝트 폴더 및 파일 명칭 변경
* /Projects/STM32F413H-Discovery 폴더의 명칭을 /Projects/STM32F429ZI 로 변경
* /Projects/NUCLEO-F429ZI 폴더 내 stm32f413h\_discovery.c/h 파일을 제외한 나머지 파일을 모두 삭제하고 stm32f413h\_discovery.c/h 파일명을 nucleo\_f429zi.c/h 로 변경
* 위 작업까지 완료하면 프로젝트 폴더 구성 완료

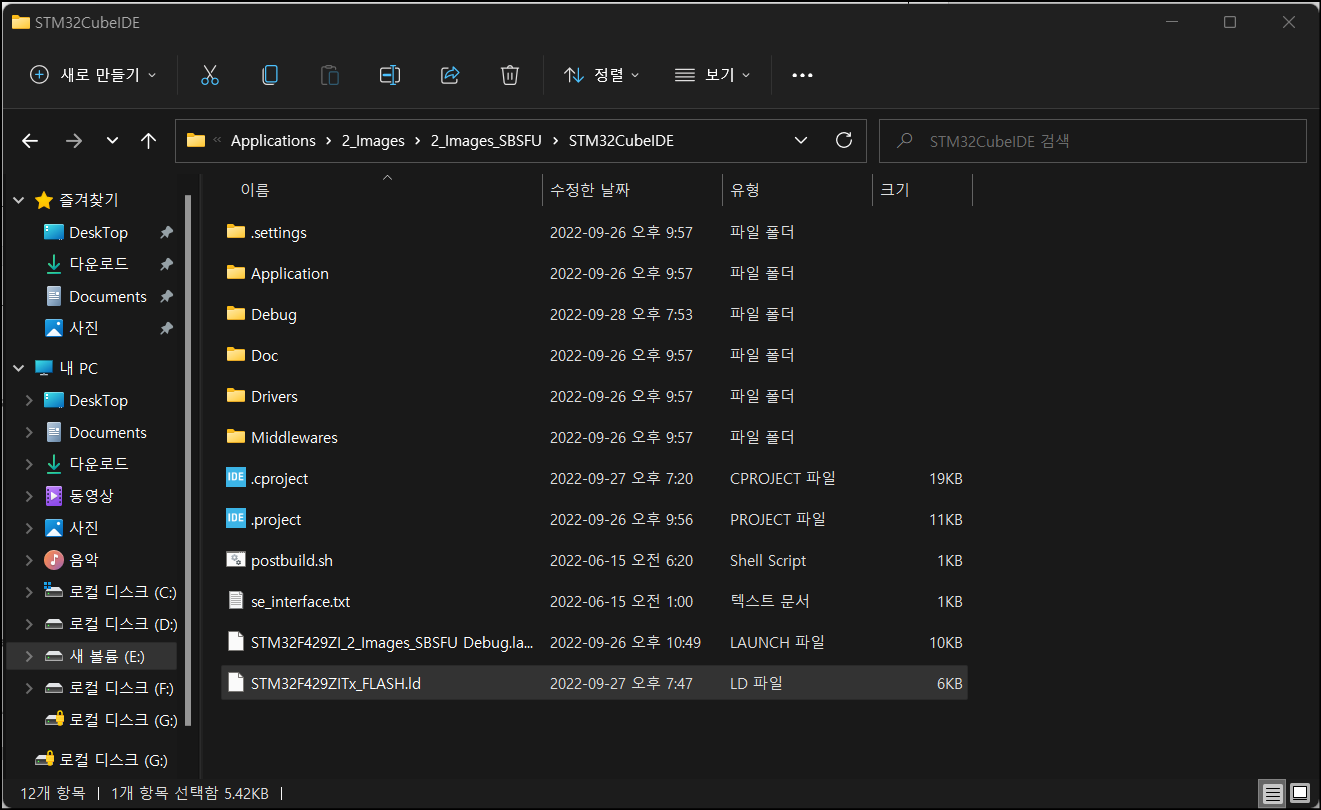


1. 포팅 절차
   1. BSP 파일 수정

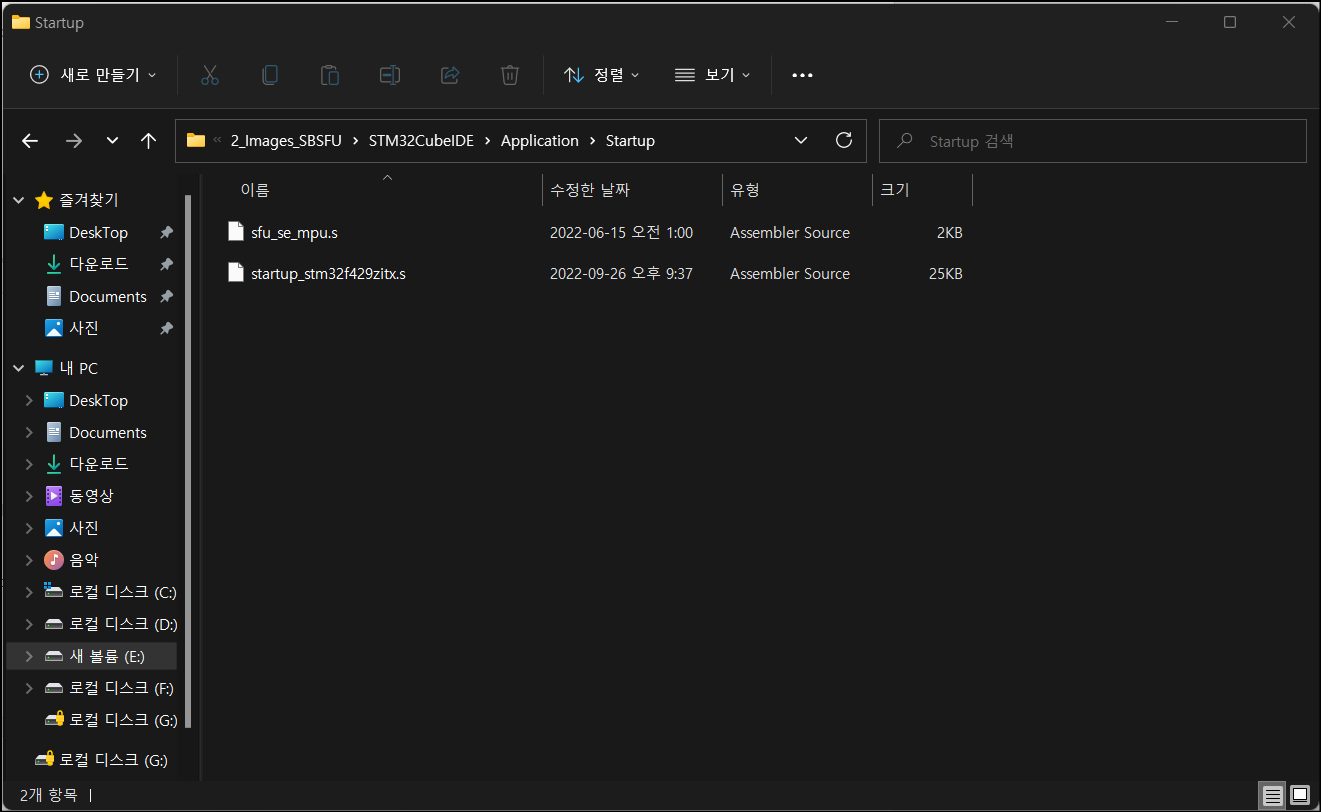
* nucleo\_f429zi.c/h 파일을 NUCLEO-F429ZI 보드에 맞게 Led 와 Button 정보를 수정한다.
* Notepad++ 검색 변경 기능을 이용해서 STM32F413-DISCO 관련 문자열을 NUCELO-F429ZI 로 교체한다.
* Led 는 NUCLEO-F429ZI 보드에 맞게 PB0, PB7, PB14 에 맞게 총 3개로 포트와 핀 정보를 수정한다.
* 버튼은 PC13 을 사용하므로 그에 맞게 수정한다.
* nucleo\_f429zi.c 파일도 동일하게 LED 와 Button 관련 내용을 수정한다.
* Led 와 Button 외 다른 주변 장치 관련 코드는 사용하지 않으므로 삭제한다.
* 세부 수정 내용은 nucleo\_f429zi.c/h 파일을 참고한다.



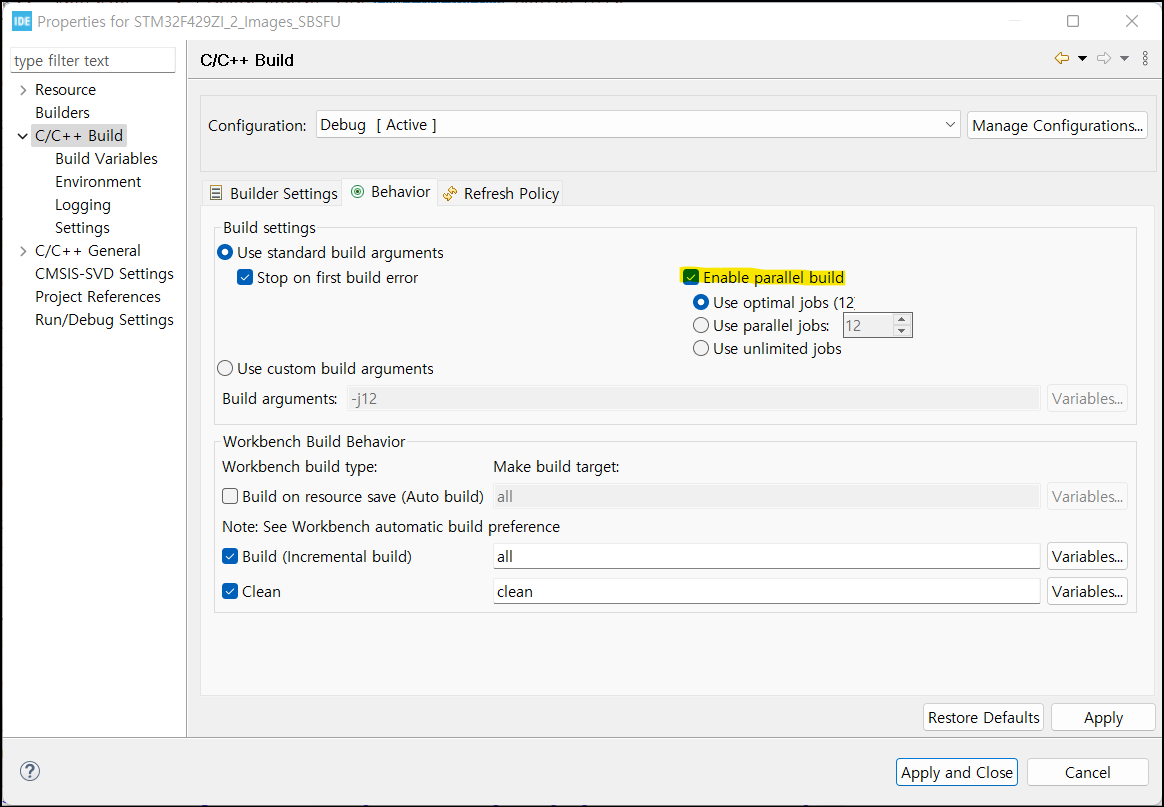
* 1. 프로젝트 파일 (.cproject / .project ) 수정
* 수정에는 notepad++ 를 사용하면 손쉽게 한번에 변경 작업을 할 수 있다.
* /2\_Images/2\_Images\_SBSFU/STM32CubeIDE 폴더 내부 .cproject 파일을 오픈한다.
* STM32F413 으로 검색하여 관련 정보들을 STM32F429 로 변경한다.
* STM32F413ZHTx 는 STM32F429ZITx 로 변경한다.
* target\_board 는 NUCLEO-F429ZI 로 변경한다.
* 프로젝트 명칭 관련 STM32F413H\_DISCOVERY\_ 는 STM32F429ZI\_ 로 교체한다.
* BSP 디렉토리 정보를 /Drivers/BSP/STM32F413H-Discovery 에서 /Device/BSP/NUCLEO-F429ZI 로 교체한다.
* STM32F413xx 는 STM32F429xx 로 변경한다.
* STM32F429ZHTx\_FLASH.ld 는 STM32F429ZITx\_FLASH.ld 로 변경한다.
* /2\_Images/2\_Images\_SBSFU/STM32CubeIDE 폴더 내부 .project 파일을 오픈한다.
* .cproject 파일과 마찬가지로 STM32F413\_DISCOVERY\_ 는 STM32F429ZI\_ 로 교체한다.
* link 태그에서 BSP 관련 파일을 stm32f413h\_discovery.c 파일을 nucleo\_f429zi.c 로 교체한다.
* 2\_Images\_SBSFU 에서 수행한 작업을 2\_Image\_SECoreBin 프로젝트와 2\_Images\_UserApp 프로젝트의 .cproject 와 .project 파일에 동일하게 적용하여 모든 STM32F413 관련 정보를 STM32F429 로 변경한다.
  1. Linker Script 파일명 변경
* 앞선 프로젝트 파일에서 linker script 파일 명칭을 변경하였으므로 실제 프로젝트 폴더에서 파일명을 STM32F469NIHX\_FLASH.ld 로 변경하여 준다.



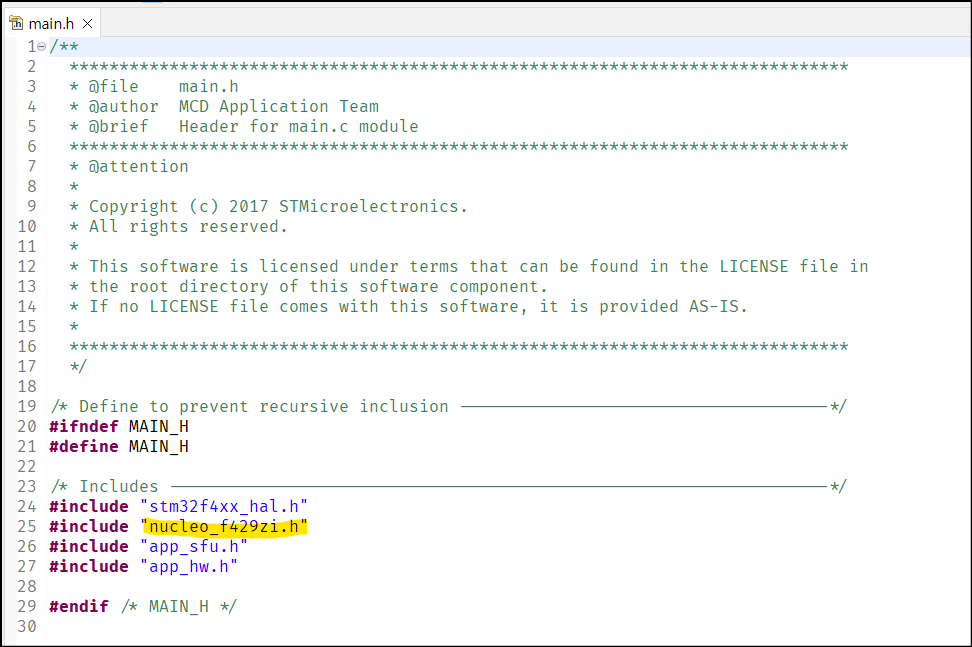
* 1. Startup 파일 변경
* MCU 를 변경하였기 때문에 startup 어셈블리 파일을 변경해 주어야 한다. 타 예제 프로젝트에서 startup\_stm32f429zitx.s 파일을 복사하여 사용한다.
* /2\_Images\_SBSFU/STM32CubeIDE/Application/Startup/ 폴더에 startup\_stm32f413xx.s 를 삭제하고 startup\_stm32f429zitx.s 를 추가한다.
* /2\_Images\_UserApp/STM32CubeIDE/Application/Startup/ 폴더에 startup\_stm32f413xx.s 를 삭제하고 startup\_stm32f429zitx.s 를 추가한다.



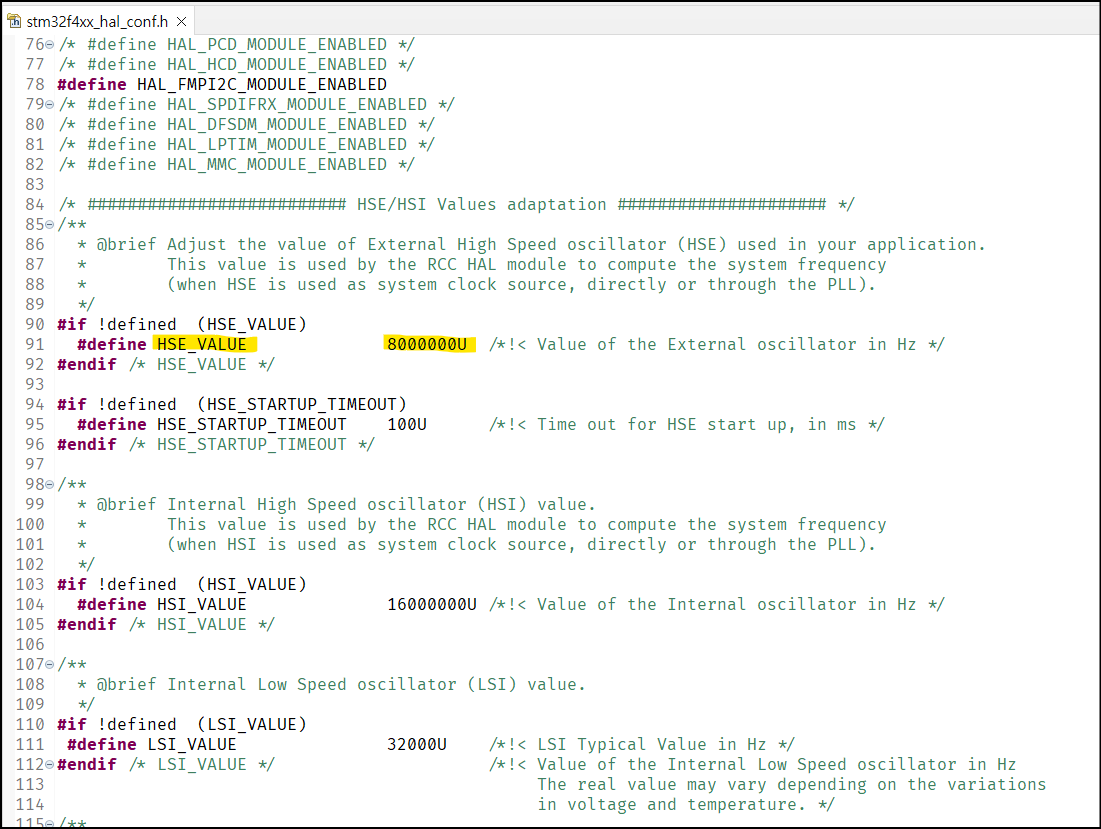
* 1. STM32CubeIDE 프로젝트 Import 하기
* 프로젝트 수정이 완료되면 STM32CubeIDE를 실행시키고 workspace 를 /STM32F429ZI/Applications/2\_Images/ 폴더로 설정하고 실행한다.
* STM32CubeIDE 에서 Import – Existing Projects into Workspace 기능을 사용하여 3개 프로젝트를 Import 한다.
* 빠른 빌드를 위해서 Preference – C/C++ Build – Behavior 에서 Enable parallel build 옵션을 3개 프로젝트 모두 enable 한다.

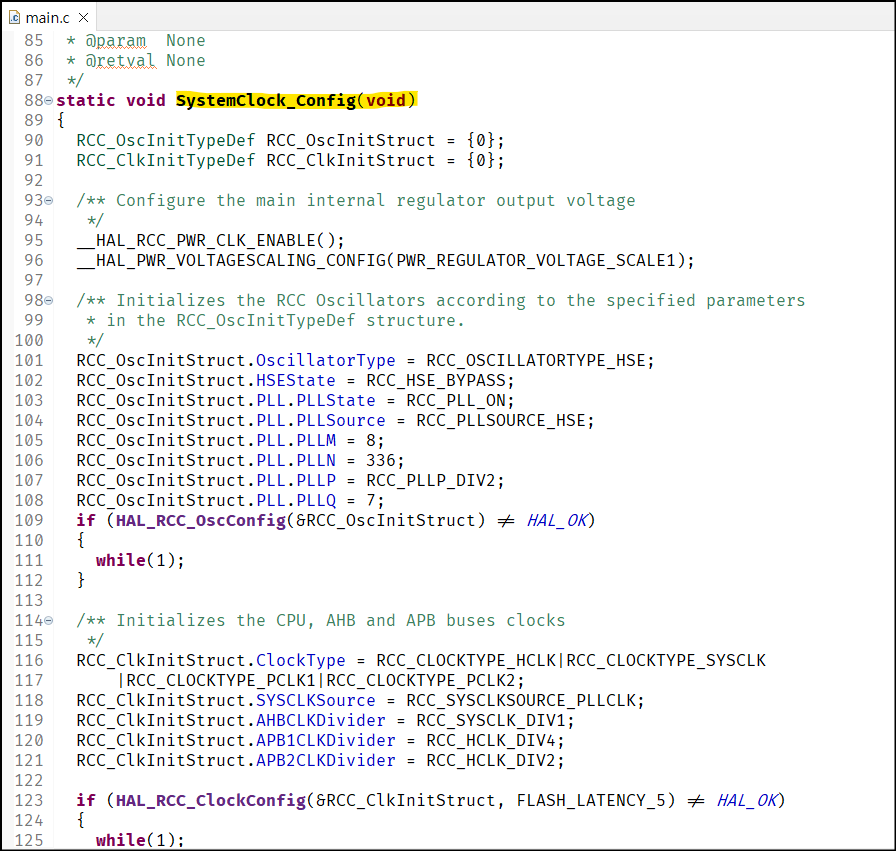


* 1. 빌드하며 컴파일 오류 수정하기
* SECoreBin 프로젝트는 빌드하면 오류 없이 빌드가 된다.
* SBSFU 프로젝트를 빌드하면 BSP include 파일 오류가 발생한다. include 파일을 stm32f413h\_discovery.h 에서 nucleo\_f429zi.h 로 변경한다.
* UserApp 프로젝트를 빌드하면 BSP include 파일 오류가 발생한다. include 파일을 stm32f413h\_discovery.h 에서 nucleo\_f429zi 로 변경한다.
* Include 파일 오류를 수정하면 아마 컴파일 오류는 없는 상태가 될 것이다. (하지만 실제 다운로드 하면 동작하지는 않는다.)

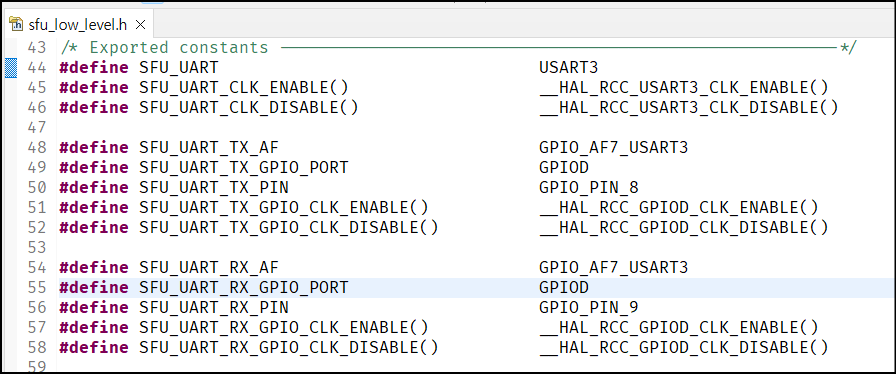


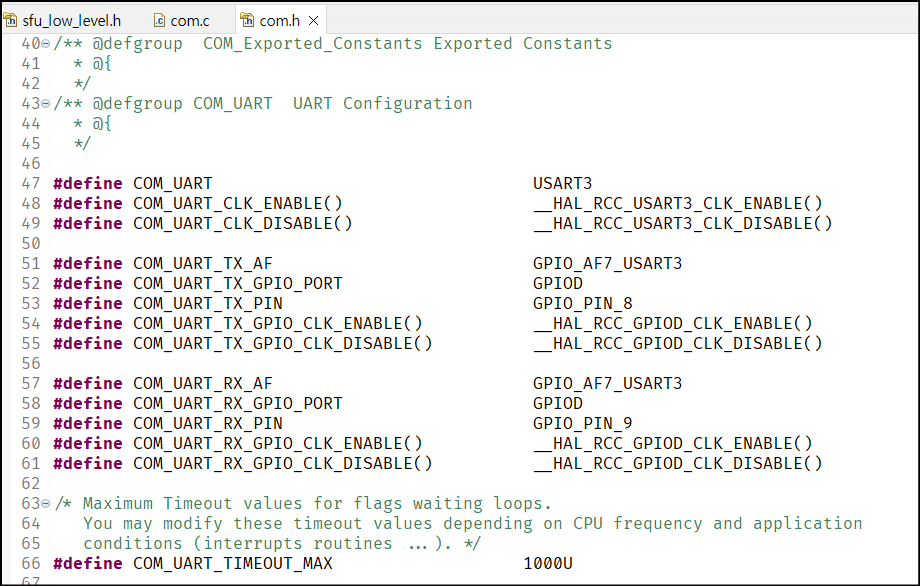
* 1. 시스템 Clock 변경 수정
* 빌드가 되더라도 동작하지 않는 이유는 STM32F413-DISCO 와 NUCLEO-F429ZI 의 회로 구성이 상이하기 때문이다.
* STM32F413-DISCO 과 NUCLEO-F429ZI 보드는 시스템 Clock 설정이 상이하기 때문에 HSE Value 와 System Clock 설정 코드를 변경해 주어야 한다.
* 먼저 HSE clock은 stm32f4xx\_hal\_conf.h 파일에 HSE\_VALUE 를 변경해 주어야 한다.
* NUCLEO-F429ZI 는 STM32F413-DISCO 와 동일하게 8MHz 오실레이터를 사용하지만 STM32F469 보드의 경우에는 25MH를 사용하기 떄문에 25000000U 로 HSE\_VALUE 를 변경해야 한다.
* SBSFU 프로젝트 main.c 파일 내 SystemClock\_Config 함수를 수정하여 준다. (코드는 CubeMx 를 이용해 Default 설정으로 생성한 코드를 참조한다.)
* UserApp 프로젝트 main.c 파일 내 SystemClock\_Config 함수를 수정해 준다. (코드는 CubeMx 를 이용해 Default 설정으로 생성한 코드를 참조한다.)
* 3개 프로젝트 모두 HSE\_VALUE 변경 시 stm32f4xx\_hal\_conf.h 파일을 변경해 주어야 하고 System Clock\_Config 는 SBSFU 프로젝트와 UserApp 프로젝트만 변경해주면 된다.



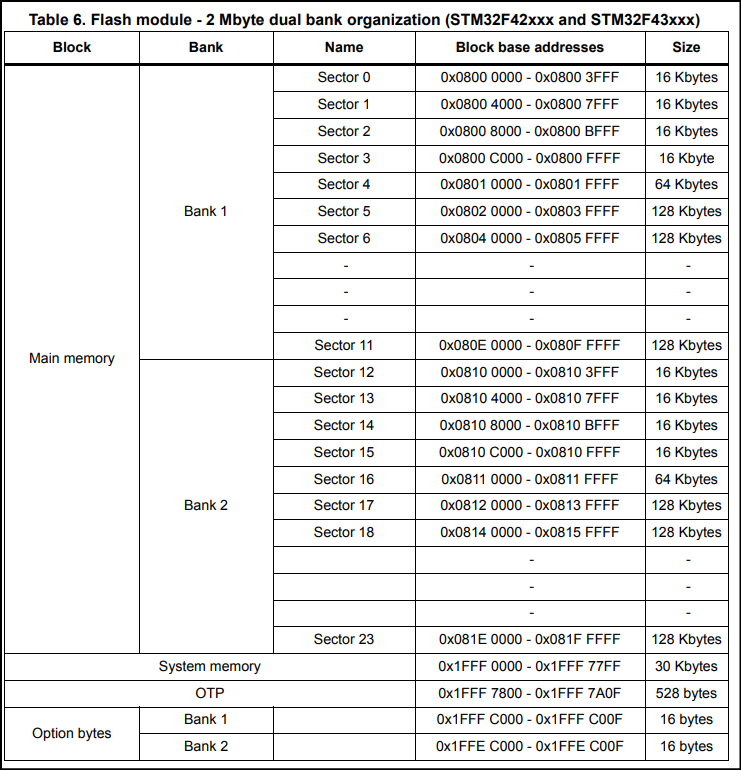


* 1. UART 정보 변경
* SBSFU 는 통신 및 디버그 메시지에 UART 를 사용한다. 따라서 NUCLEO-F429ZI 보드에 맞게 UART 포트를 변경해 주어야 한다.
* NUCLEO-F429ZI 보드는 reference 매뉴얼을 참고하면 ST-Link 와 UART3 가 연결되어 있다.
* SBSFU 프로젝트의 sfu\_low\_level.h 파일을 오픈하면 SFU\_UART 관련 설정을 확인할 수 있다. 이 정보들을 NUCLEO-F429ZI 보드에 맞게 수정한다.
* NUCLEO-F429ZI 보드의 UART3 은 PD8(TX), PD9(RX) 를 사용하며 Alternate Function 값은 GPIO\_AF7\_USART3 이다.   
  (이 정보를 쉽게 확인하려면 CubeMx 를 이용해서 Default 설정으로 생성된 프로젝트의 UART3 초기화 코드를 확인하면 된다)
* 동일하게 UserApp 프로젝트의 com.h 파일의 COM\_UART 관련 설정을 변경해 주어야 한다. SFU\_UART 와 동일한 정보로 설정해 주면 된다.

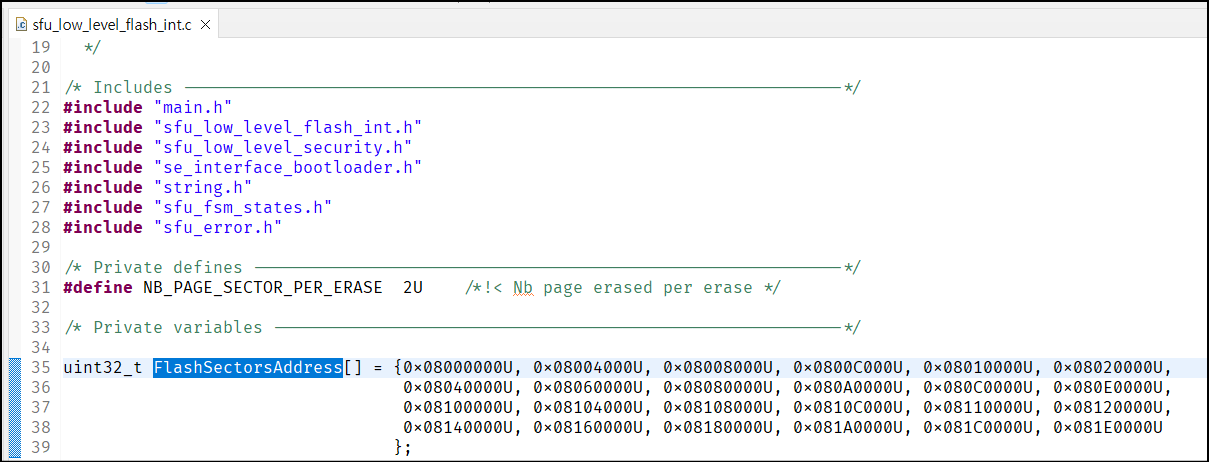


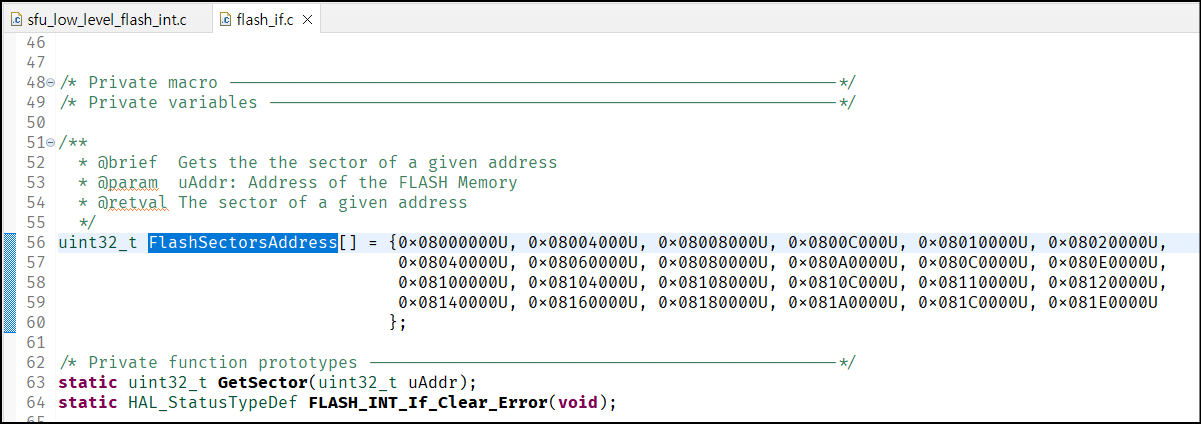


* 1. Flash 정보 변경
* STM32F429ZI 의 flash 영역 구성에 맞게 flash 영역 정보를 수정한다.
* 2개 뱅크에 24개의 섹터로 구성되어 있는 것을 확인할 수 있다.

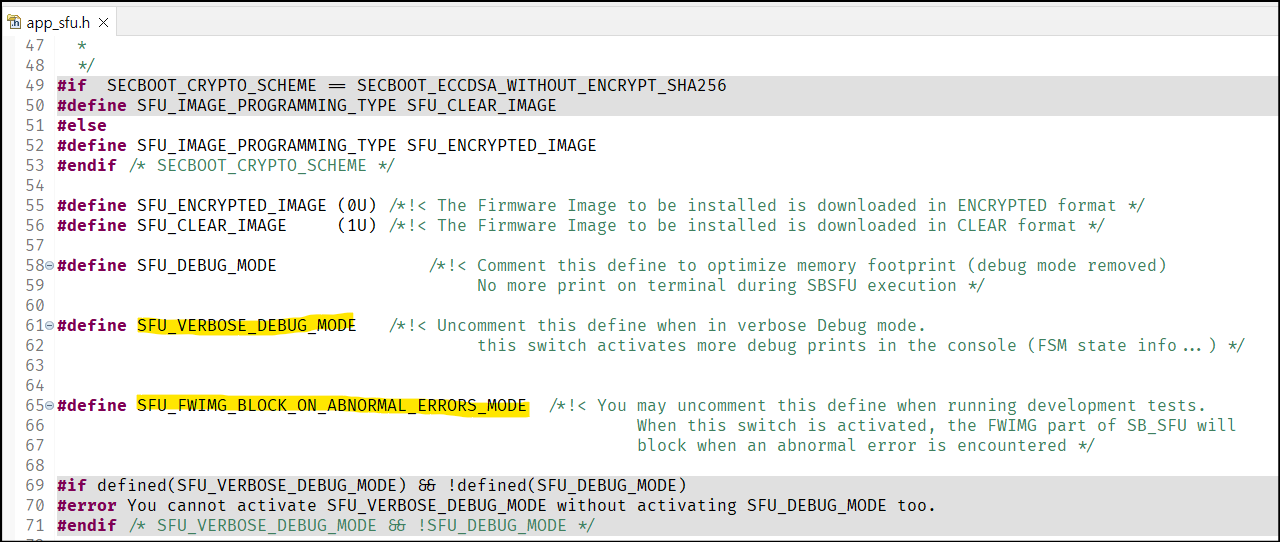


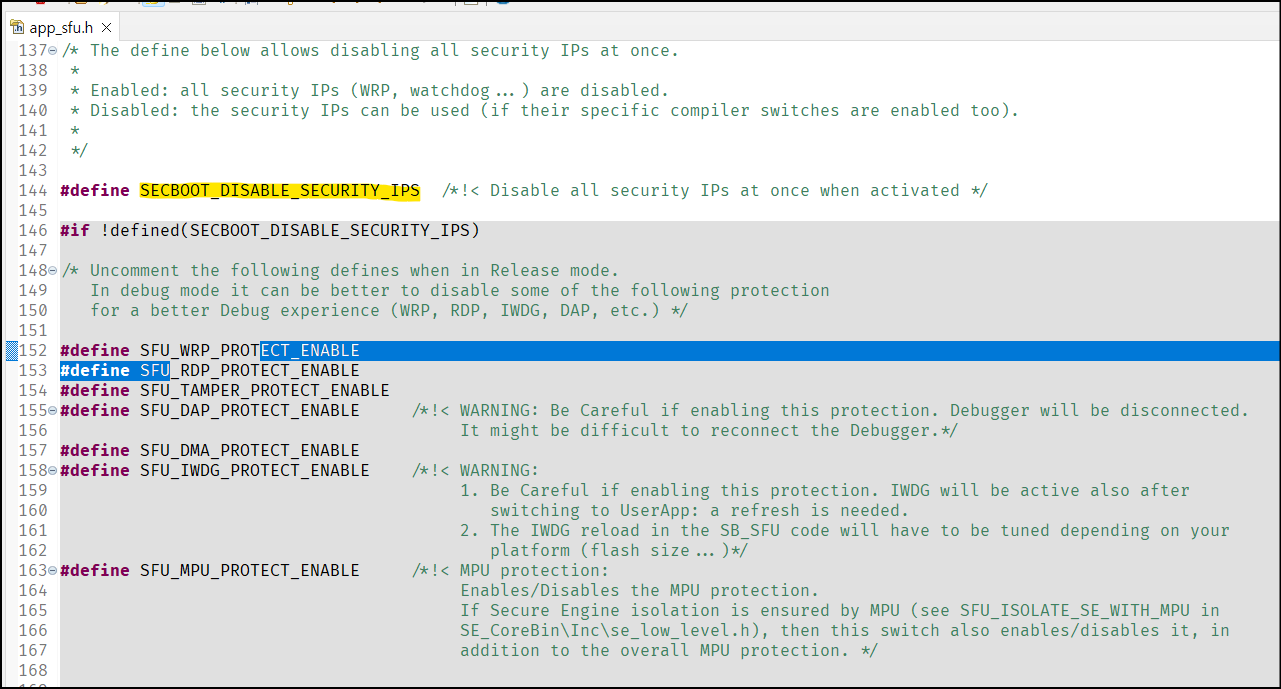
* SBSFU 프로젝트의 sfu\_low\_level\_flash\_int.c 파일을 수정한다.
* UserApp 프로젝트의 flash\_if.c 파일을 수정한다.
* 각각 파일의 FlashSectorsAddress 배열을 24개 섹터의 시작 주소로 설정한다.



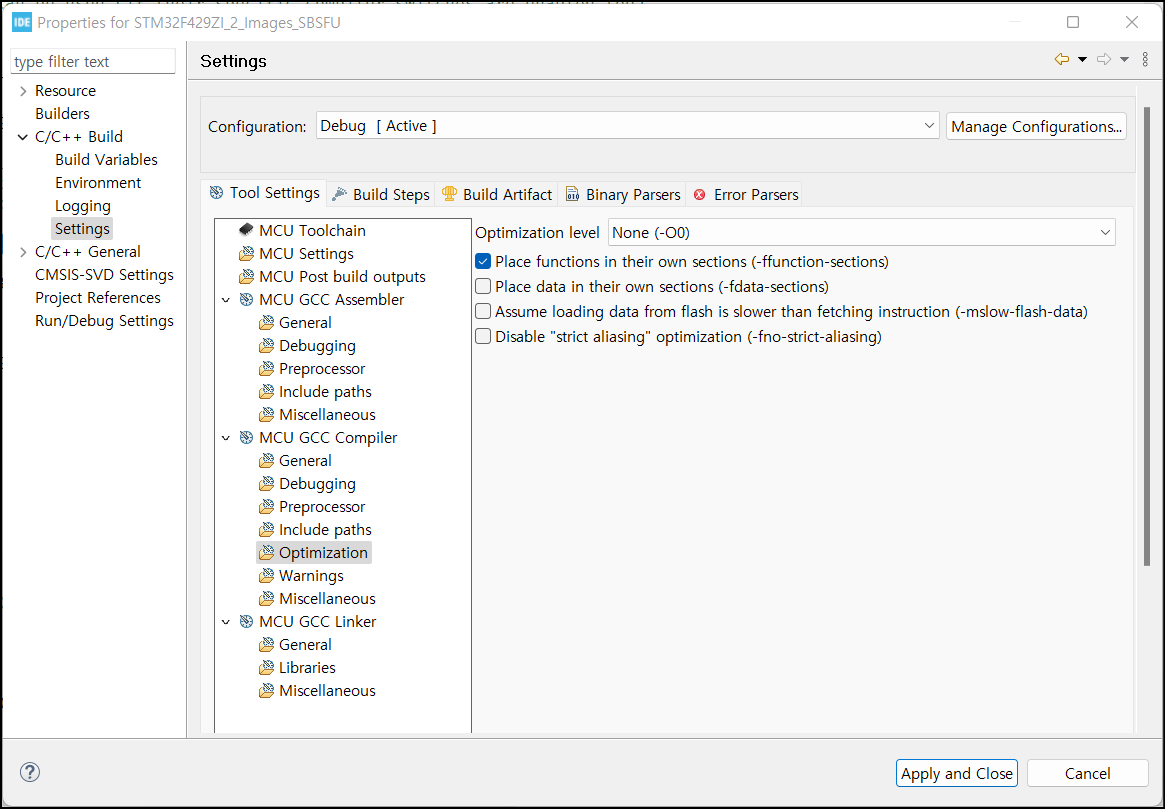


* 1. 개발을 위한 SBSFU 보호 기능 비활성화 및 옵션 변경
* SBSFU 프로젝트의 app\_sfu.h 파일을 오픈한다.
* SFU\_VERBOSE\_DEBUG\_MODE 주석을 해제한다. (더 많은 디버그 메시지를 볼 수 있다)
* SFU\_FWIMG\_BLOCK\_ON\_ABNORMAL\_ERRORS\_MODE 주석을 해제한다. (에러 발생시 중지)
* SECBOOT\_DISABLE\_SECURITY\_IPS 주석을 해제한다. (보안기능이 비활성화되어 디버깅 용이)
* SWAP 기능을 미사용 하고자 하는 경우 SFU\_NO\_SWAP 주석을 해제한다. (특별히 롤백 등의 기능이나 Partial FW 기능이 필요 없는 경우)

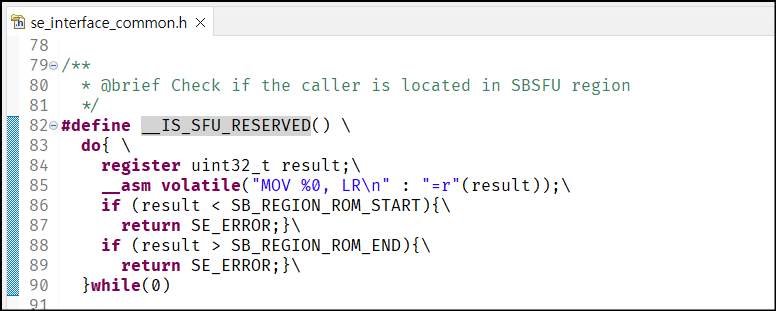




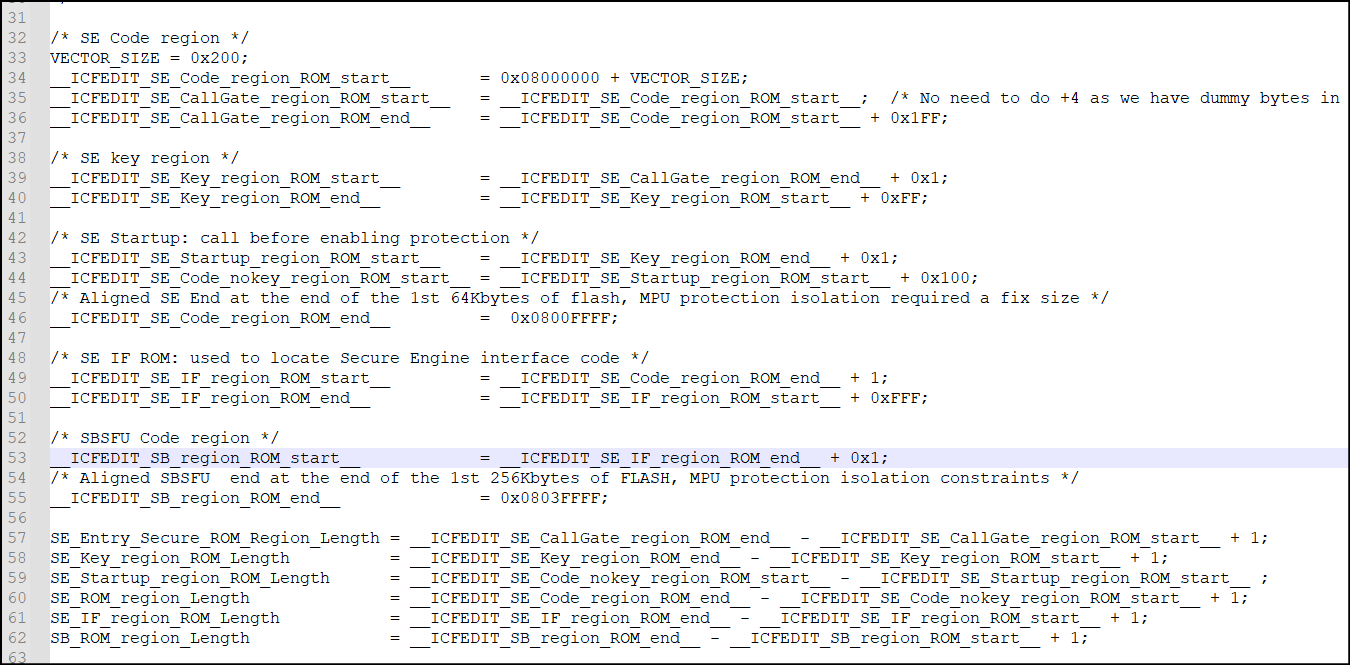
* 1. 디버깅을 위한 optimization 옵션 비활성화
* 현재 X-CUBE-SBSFU 프로젝트는 용량 최소화optimization 옵션이 설정되어 있다.
* 이 경우 디버깅이 불편한데 debugging 편의성을 위하여 optimization 옵션을 none 으로 설정한다.



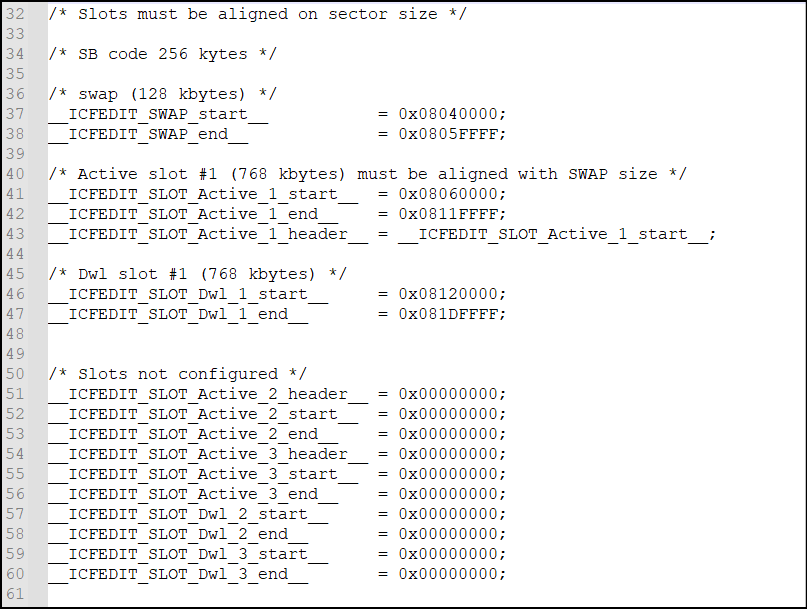
* 이렇게 최적화 옵션을 수행 후 실행하면 secure engine 시작 코드 내부의 \_IS\_SFU\_RESERVED 에서 지정된 영역 외 접근으로 인식하며 동작이 중단되는데 이유는 \_\_IS\_SFU\_RESERVED 내부에 inline 함수가 사용되고 있어 최적화 비활성화 시 inline 함수가 일반 함수로 동작하면서 LR 레지스터를 의도하지 않은 주소로 반환하기 때문이다.
* 따라서 \_\_IS\_SFU\_RESERVED 함수 내부를 아래와 같이 수정하여야 inline 함수가 최적화 되지 않아 LR 레지스터가 잘못 반환되는 문제를 회피할 수 있다.



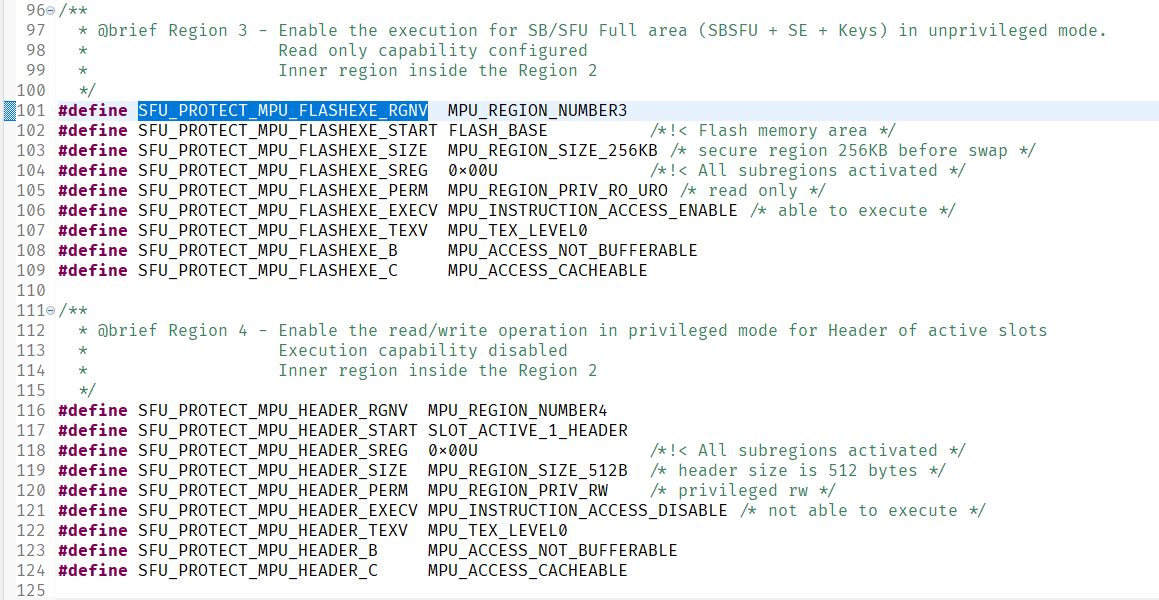
* 1. 디버깅을 위한 Linker script 변경 및 슬롯 크기 변경
* 디버깅을 위해 optimization 옵션을 변경 시, 코드의 크기가 증가하기 때문에 기본적으로 설정된 secure engine 과 secure boot 의 영역이 부족하게 된다. 따라서 linker script 내 두 영역의 크기를 확장 시켜 주어야 한다.
* /Linker\_Common/STM32CubeIDE/mapping\_sbsfu.ld 파일을 notepad++ 로 오픈한다.
* STM32F413-DISCO 예제의 SE\_Code\_region 의 설정값은 32KB(0x7FFF) 이다. 이 크기를 64KB(0xFFFF) 로 증가시킨다.
* STM32F413-DISCO 예제의 SE\_IF\_region 의 설정값은 2KB(0x8FF) 이다. 이 크기를 4KB(0xFFF) 로 증가시킨다.
* STM32F413-DISCO 예제의 SB\_region 의 설정값은 128KB(0x1FFFF) 이다. 이 크기를 256KB(0x3FFFF) 로 증가시킨다.



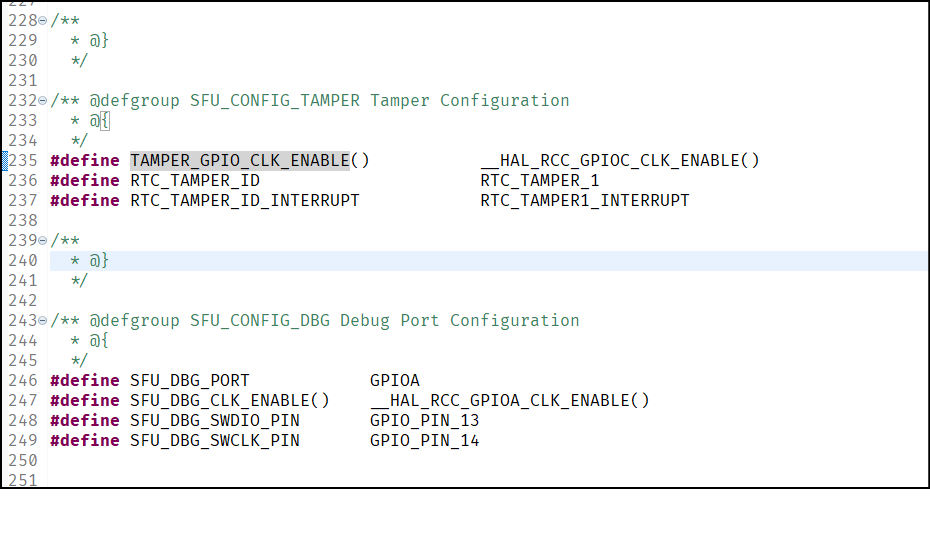
* STM32F413H-DISCO 보드는 flash 용량이 1536KB 이고 NUCLEO-F429ZI 보드는 2048KB 이다.
* 따라서 액티브 슬롯과 다운로드 슬롯의 크기를 늘릴 수 있다.
* /Linker\_Common/STM32CubeIDE/mapping\_fwimg.ld 파일을 로드한다.
* 부트로더 코드가 증가한 만큼 SWAP 영역의 시작 위치를 0x8040000 으로 조정한다.
* (2048 – 256 – 128) / 2 = 1664KB 이므로 슬롯마다 각각 최대 832KB 를 할당할 수 있지만, SWAP 크기인 128KB 과 align 되어야 하기 때문에 최대 슬롯 크기는 768KB 가 된다.
* 액티브 슬롯과 다운로드 슬롯의 시작과 끝 주소를 768KB 에 맞게 수정한다.



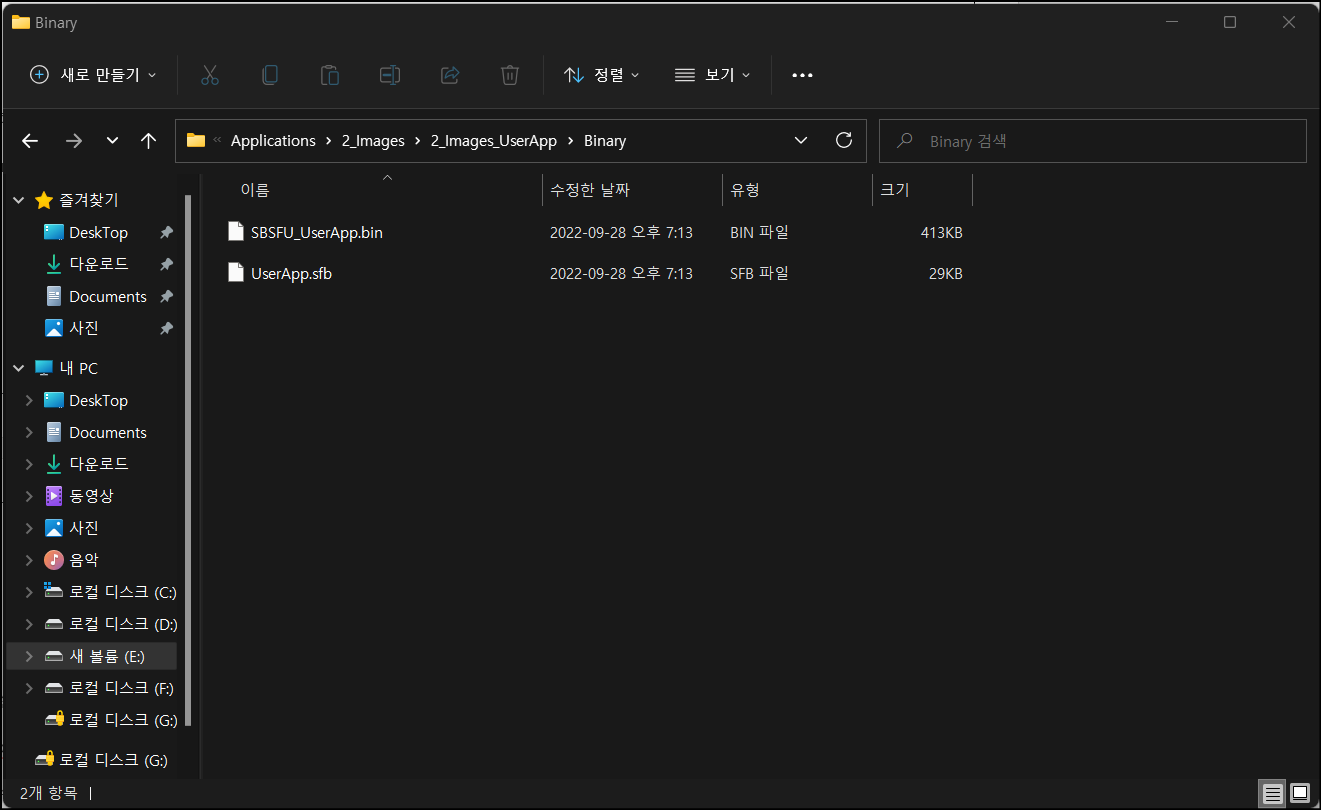
* 1. MPU 설정
* 각 영역의 크기를 변경하였기 때문에 MPU 설정의 영역 크기를 변경해 주어야 한다.
* Sfu\_low\_level\_security.h 파일의 MPU 설정값들을 변경해 준다.
* 디버깅을 위해서 SB region 의 크기를 증가시켰으므로 SFU\_PROTECT\_MPU\_FLASHEXE\_RGNV 의 크기를 128KB 에서 256KB 로 늘려준다.
* SFU\_PROTECT\_MPU\_SRAMACC\_RGNV 에서 SRAM 의 크기가 192KB 이므로 SFU\_PROTECT\_MPU\_SRAMACC\_SREG 값을 0xF8 로 변경한다.
* SFU\_PROTECT\_MPU\_EXEC\_SE\_RGNV 에서 SE region 의 크기를 64KB 로 늘려준다.
* APP\_PROTECT\_MPU\_FLASHEXE\_RGNV 에서 각 영역의 크기가 바뀌었으므로 APP\_PROTECT\_MPU\_FLASHEXE\_SREG 값을 변경 해 준다.
* 각각의 차이점은 STM32F413-DISCO 프로젝트와 포팅된 프로젝트의 파일을 비교해서 확인한다.



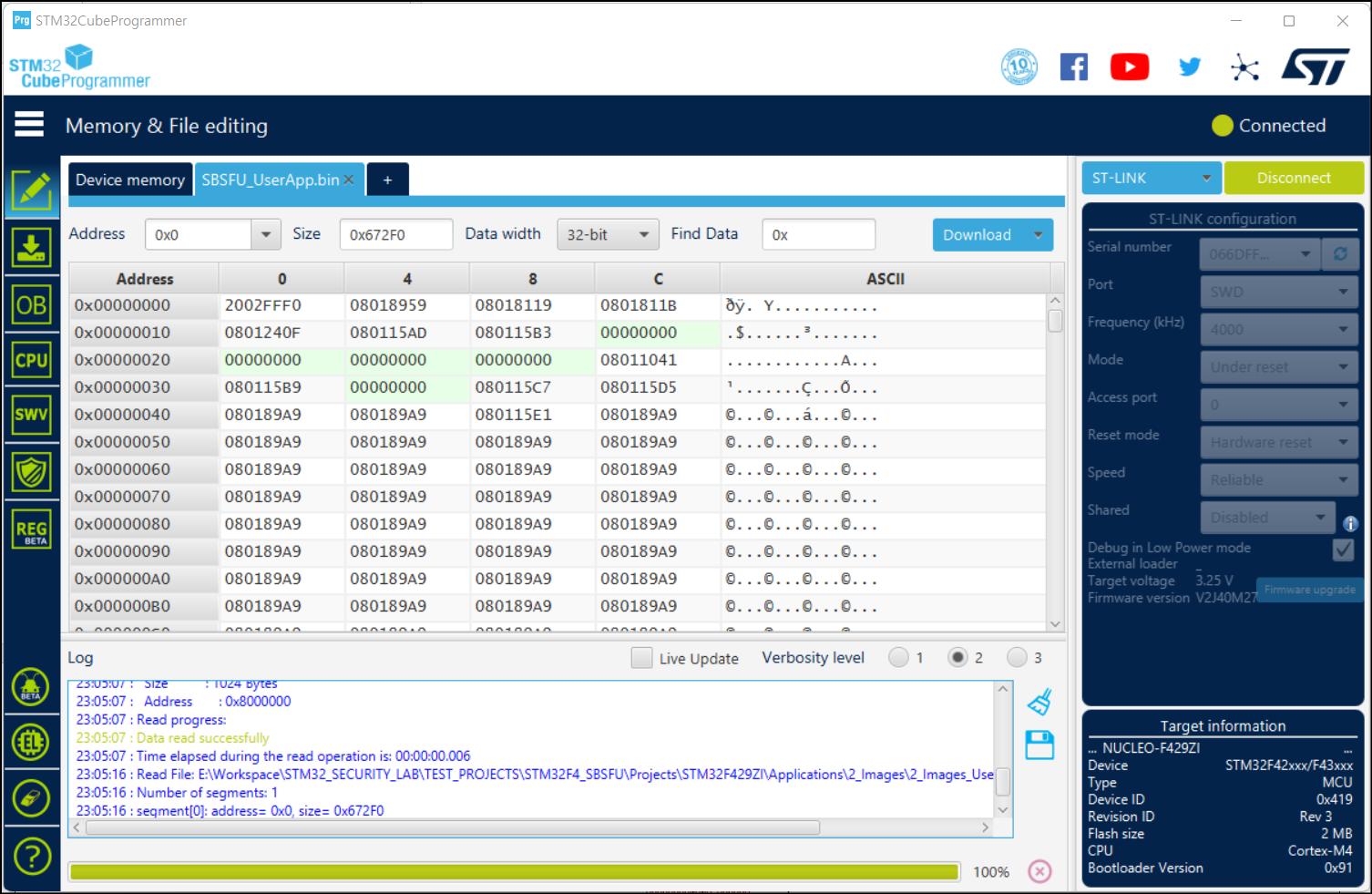
* 1. Tamper 설정
* STM32F413 과 STM32F429 의 RTC Tamper 기능은 특별히 차이가 없어서 수정할 필요가 없지만 다른 MCU 는 핀과 설정 방식이 동일한지 확인해 볼 필요는 있다.



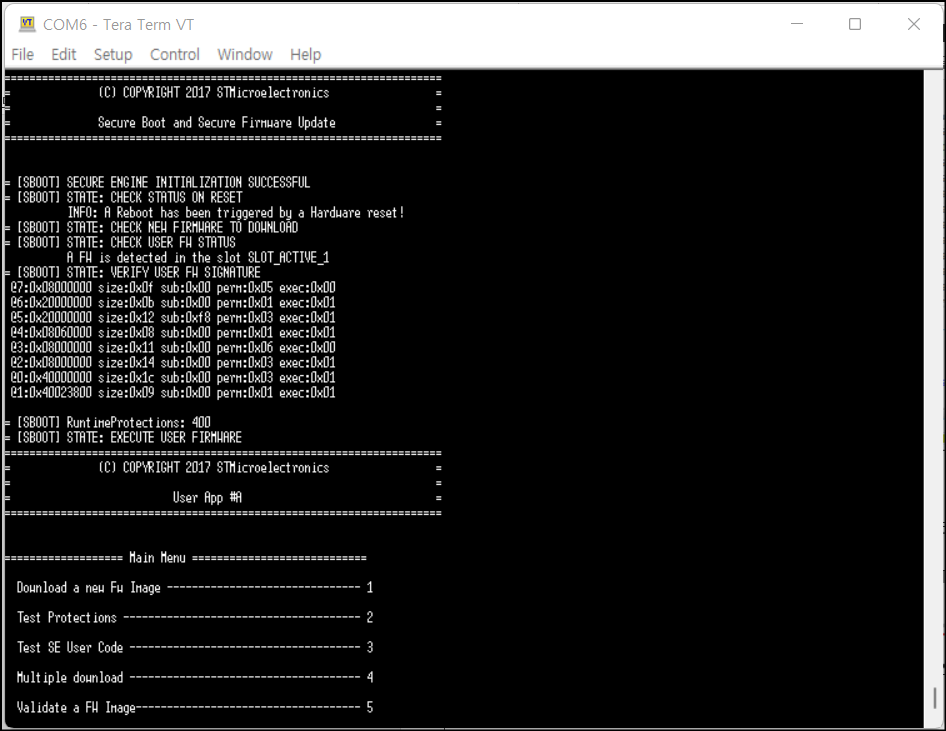
* 1. 테스트
* 이제 빌드 된 SBSFU 를 다운로드 하여 정상 동작을 확인해볼 필요가 있다.
* Postbuild 작업 후에 결과물은 /2\_Images\_UserApp/Binary 폴더에 저장된다.
* SBSFU\_UserApp.bin 파일은 SBSFU 와 Application 이 merge 된 파일이고, UserApp.sfb 는 업데이트용 순수 application 파일이다.



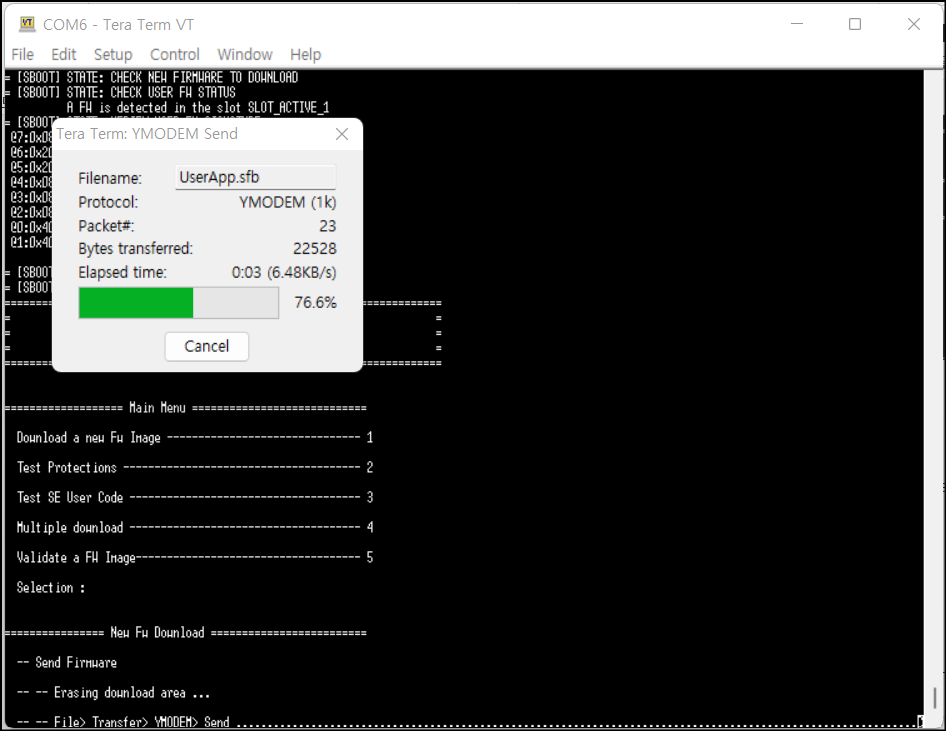
* 우선 STM32CubeProgrammer 를 실행한다.
* Writing 을 위해서 SBSFU\_UserApp.bin 파일을 로드한다.



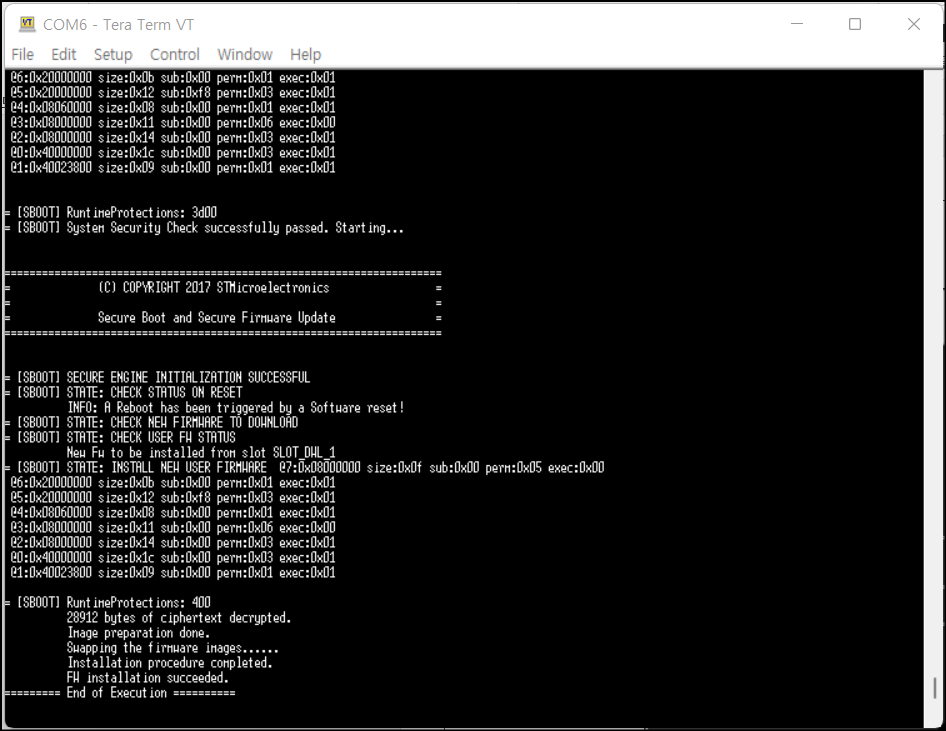
* Mass erase 로 전체 flash 영역을 삭제 후 download 를 클릭하여 writing 을 수행한다.
* 정상적으로 flash 가 writing 된 후, ST-Link 를 disconnect 한다.
* PC 내 ST-Link VCP 포트를 확인 후 TeraTerm 을 이용하여 115200-8-N-1 설정으로 접속한다.
* 보드 reset 버튼을 클릭하면 정상적으로 SBSFU 가 수행되고 UserApp 으로 Jump 하는지 확인한다.



* 메뉴에서 1번 “Download a new Fw Image” 를 선택하면 다운로드 대기 상태가 된다.
* 이 때 TeraTerm 메뉴의 File – Transfer - Y-MODEM - Send 메뉴를 선택하고 UserApp.sfb 파일을 선택하여 전송한다.



* TeraTerm 출력을 확인하여 정상적으로 다운로드 후 Installation 프로세스가 수행되는지 확인한다.



* app\_sfu.h 파일의 보호 기능 활성화 후 secure boot 및 업데이트를 다시 테스트하여 모두 정상 수행되는지 확인한다.
* 앞선 설정의 문제가 있는 경우, 다운로드 과정이나 설치 과정에서 오류가 발생할 것이다.