- Date : 2023 년 07 월 20 일
- 제목 : dsPIC33CKMP 의 AIVT 를 이용한 부트로더와 애플리케이션 모두에서 인터럽트 사용방안, 설정 유의사항, 예제 프로젝트 공유
- 문서번호 : KR_ES_01034
- 작성자 : 김기남 책임 (chris.kim@microchip.com, Sr.ESE)

본 문서는 dsPIC33CKMP 패밀리 MCU 에서 부트로더 구현 시, 부트로더와 애플리케이션 모두에서 인터럽트를 사용하는 방안에 대해 논의한다. 해당 디바이스의 AIVT 기능을 이용하면 동일 인터럽트를 부트로더와 애플리케이션에 대해 각각 다른 테이블 지정이 가능하다. 그러나 MCC 부트로더 생성 시 AIVT 기능 미지원으로 AIVT enable 시 빌드 에러가 발생하며, 정상적인 동작을 위해 MCC 생성 코드에 관련 코드 변경이 필요하다. 또한 부트로더의 경우 링커 추가 옵션을 설정해야 boot 영역에 코드 배치가 가능하다. 본 문서는 MCC 를 이용하여 부트로더 및 애플리케이션 기본 코드를 생성하고, AIVT 고려 시 필요한 추가 코드를 수정한다. 그리고 예제 프로젝트를 제공한다.

1. 예제 프로젝트 설명

예제 프로젝트는 dsPIC33CK256MP508 디바이스 기준으로 작성되었으며, dsPIC33CK Curiosity 보드와 MCP2221A USB-to-UART bridge IC 를 사용하여 테스트 되었다.
예제는 다음 폴더구조로 압축되어 있다.

- 0_UnifiedHost-1.19.1
  - 부트로더 Host 프로그램이며, PC 는 MCP2221A 와 연결한다
- 1_Default
  - MCC 기본 부트로더와 애플리케이션 프로젝트이다
- 2_AIVT_enabled
  - MCC 기본 부트로더에 AIVT enable 하고, 관련 부분이 수정된 프로젝트이다 (변경점은 1_Default 프로젝트와 Compare)
  - 추가로 Timer1 인터럽트가 부트로더, 애플리케이션 프로젝트에서 모두 enable 되어있다.

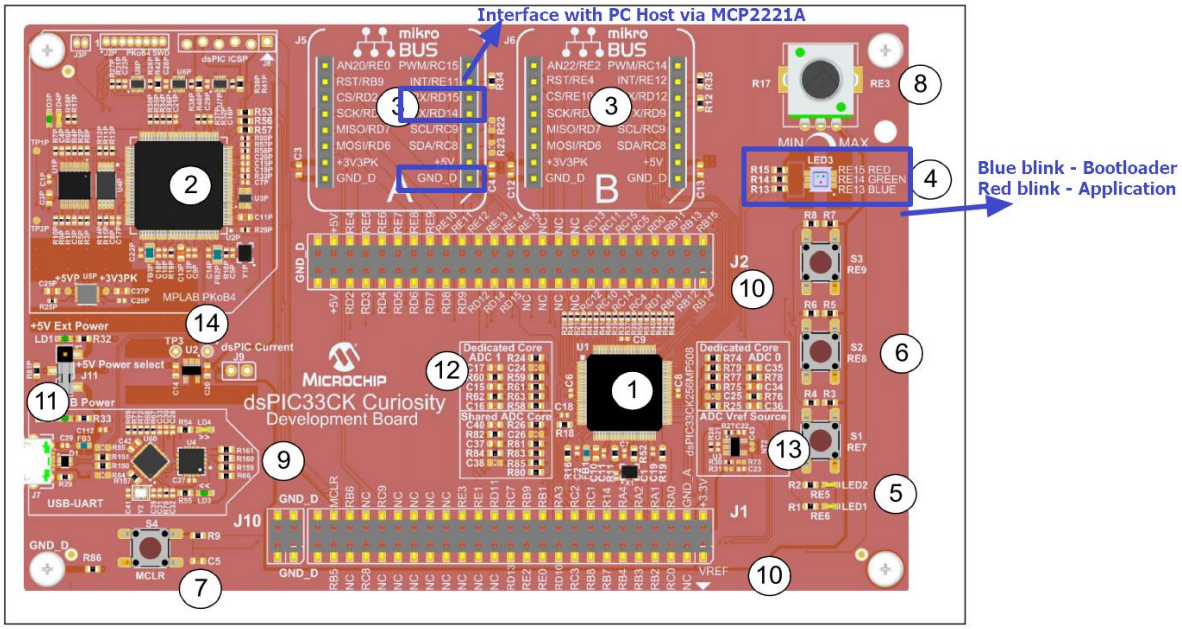
환경

MPLAB X IDE v6.05 (v6.10 도 상관없음)
dsPIC33CK-MP_DFP (1.11.346)
XC16 (v2.00)
MCC 5.3.0
16-bit Bootloader 1.23.0

2. 동작 설명

2_AIVT_enabled 폴더의 부트로더를 프로그램하면 1 초 간격으로 Curiosity 보드의 RGB LED 의 Blue 가 토글되어 깜빡인다. 애플리케이션을 부트로딩하여 애플리케이션이 실행되면 1 초 간격으로 Red 가 토글되어 깜빡인다.

FIGURE 1-1: dsPIC33CK CURIOSITY DEVELOPMENT BOARD

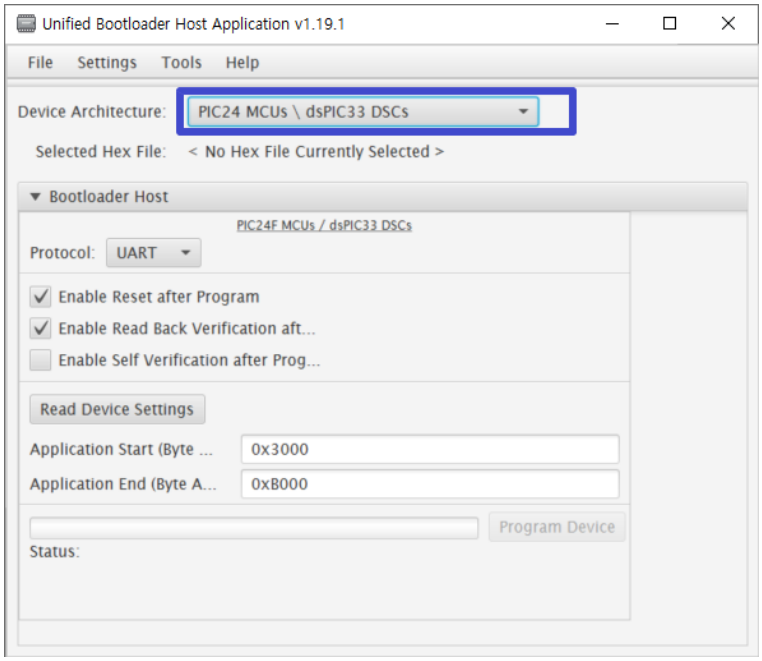


3. 실행 방법

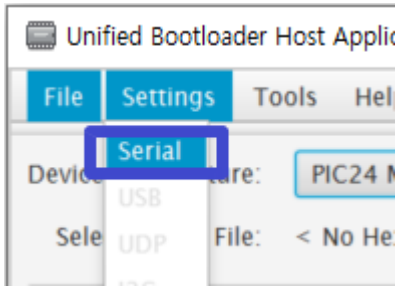
다음에서는 MCC UnifiedHost PC 툴을 가지고, MCC 부트로더 동작 과정에 대해 설명한다.

Bootloader 프로젝트를 Curiosity 보드에 다운로드 한다.
Blue LED 가 Blink 됨을 확인한다.

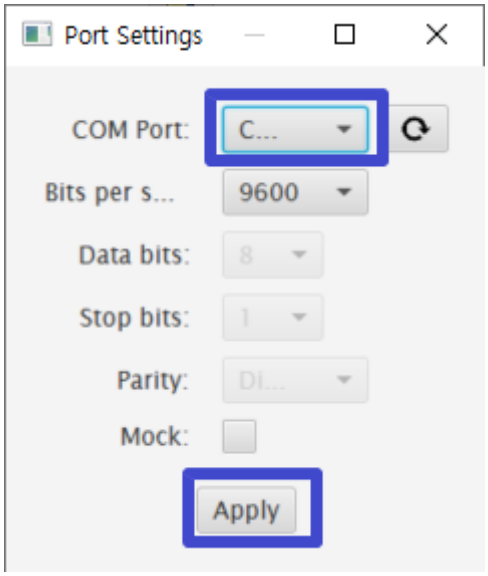
0_UnifiedHost-1.19.1 폴더에서 UnifiedHost 를 아래 명령으로 실행한다.
0_UnifiedHost-1.19.1>**java -jar UnifiedHost-1.19.1.jar**
dsPIC33 DSCs 디바이스를 선택한다



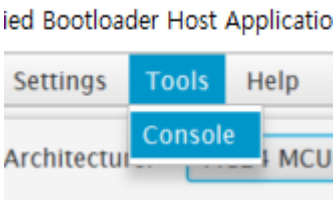
Settings > Serial 클릭하여 시리얼 설정



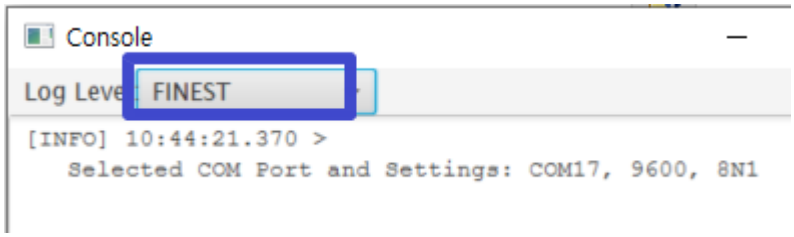
Comport 를 선택 후 Apply 한다.



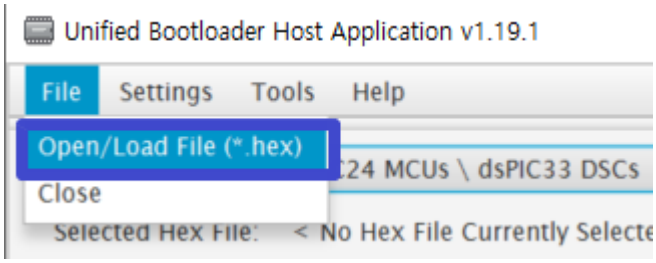
로그를 보기위해 Tools > Console 을 클릭하여 Console 을 활성화 한다.



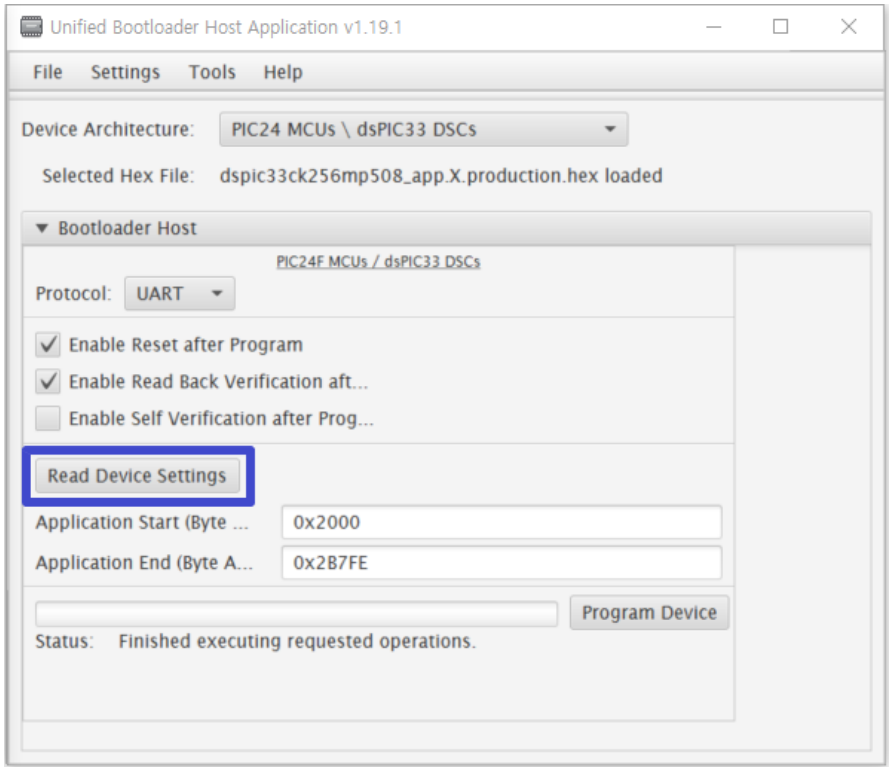
로그레벨을 최저로 모든 로그 출력 가능하게 변경한다.



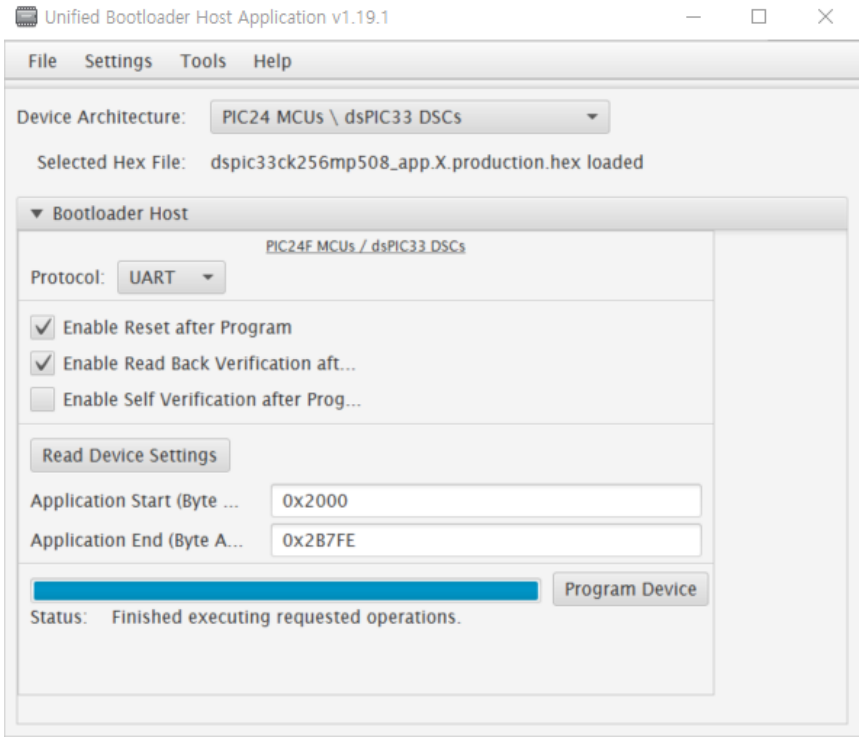
Open File 을 클릭하여 Hex 파일을 로딩한다.
dspic33ck256mp508_app.X₩dist₩default₩production₩dspic33ck256mp508_app.X.production.hex

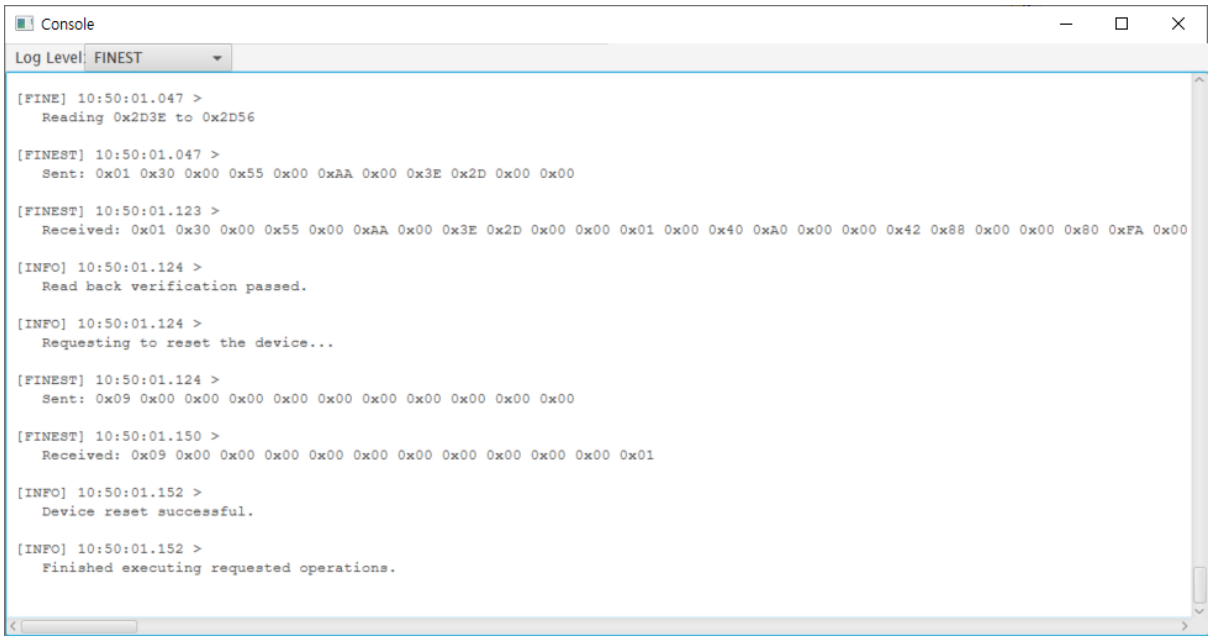


Read Device Settings 버튼을 클릭하여 Application START/END 주소 변경을 확인한다.



Program Device 버튼을 클릭하여 Application 을 writing 한다.
Write 완료 후 MCU 는 RESET 하여 application 쪽으로 Jump 되고, RED LED 가 점멸된다.



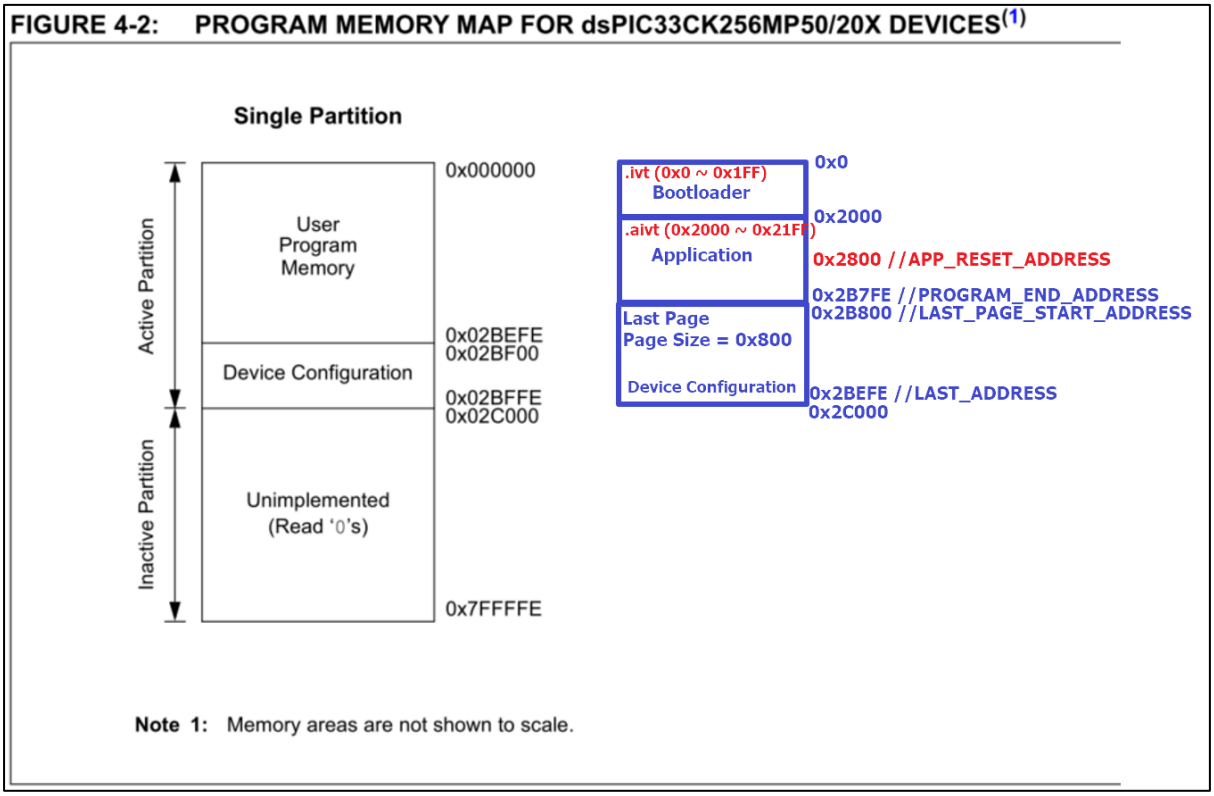


4. 메모리 맵

아래는 dsPIC33CK256MP50x Device 에 대한 Single Partition Memory Map 보여준다.

우측은 해당 디바이스의 Memory Map 을 0x2000 전까지는 부트로더가 0x2000 후부터는 애플리케이션이 사용하는 것을 보여준다. 또한 AIVT vector 는 0x2000 번지에 위치하기 되고, Application Reset 코드는 0x2800 에 위치되어 있다.

FIGURE 4-2: PROGRAM MEMORY MAP FOR dsPIC33CK256MP50/20X DEVICES(1)



Note 1: Memory areas are not shown to scale.

5. Interrupt Vector Table

다음은 사용할 Timer1 에 IVT 주소를 보여준다.

BSLIM 은 4. 메모리 처럼 배치를 위해 0x1FFA 로 설정한다.

아래 공식은 BSLIM 설정값을 통해 AIVT 테이블 주소 계산을 보여준다.

$[(\overline{BSLIM[12:0]} - 1) \times 0x800] + Offset.$

AIVT Address = (NOT(**BSLIM**) – 1) x 0x800 + Offset

= (NOT(**0x1FFA**) – 1) x 0x800 + Offset

= (5 – 1) x 0x800 + Offset

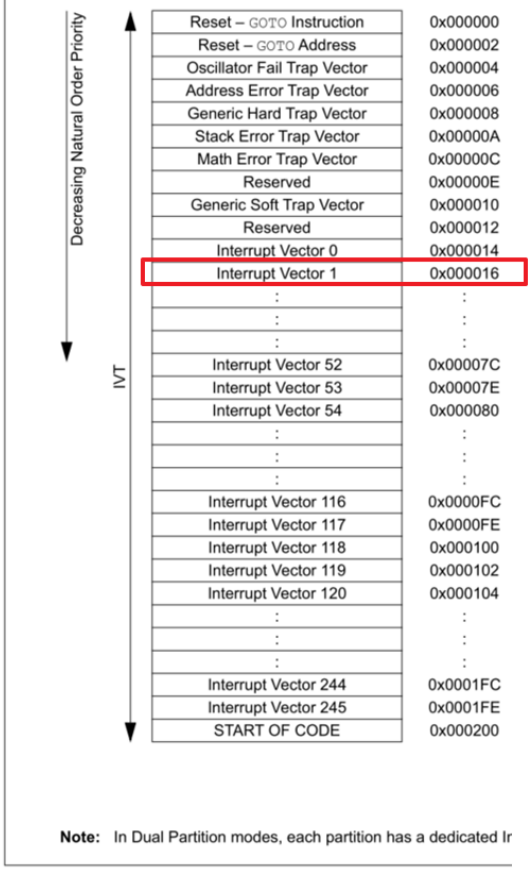
= **0x2000** + Offset

즉, BSLIM 0x1FFA 설정은 0x2000 에 AIVT 주소이고, 0x2800 부터 애플리케이션 코드가 배치된다.

TABLE 7-2: INTERRUPT VECTOR DETAILS

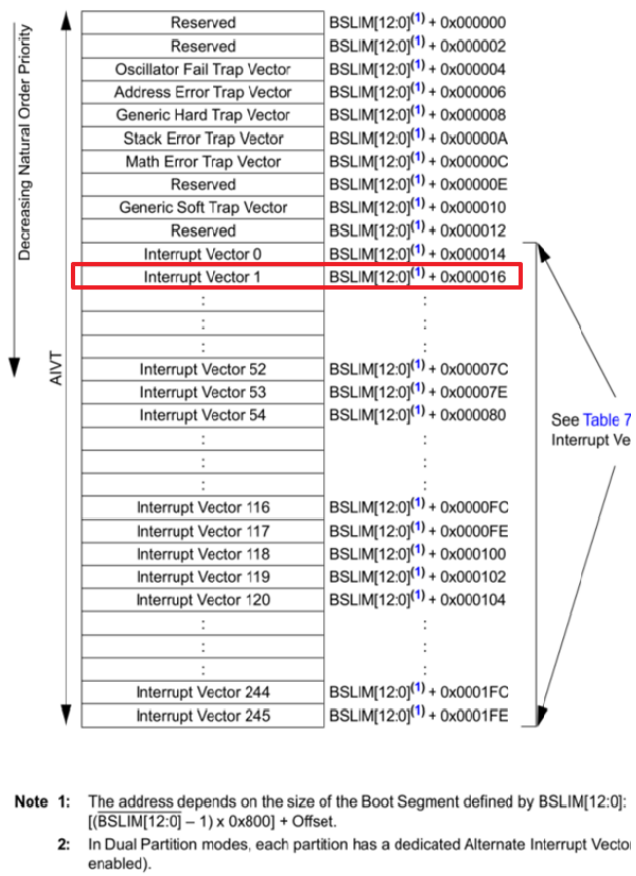
Interrupt Source	MPLAB® XC16 ISR Name	Vector #	IRQ #	IVT Address	Interrupt Bit Location		
					Flag	Enable	Priority
External Interrupt 0	_INT0Interrupt	8	0	0x000014	IFS0[0]	IEC0[0]	IPC0[2:0]
Timer1	_T1Interrupt	9	1	0x000016	IFS0[1]	IEC0[1]	IPC0[6:4]

FIGURE 7-1: dsPIC33CK256MP508 FAMILY INTERRUPT



Note: In Dual Partition modes, each partition has a dedicated Ir

RE 7-2: dsPIC33CK256MP508 ALTERNATE INTERRUPT VECTOR TABLE



Note 1: The address depends on the size of the Boot Segment defined by BSLIM[12:0]: [(BSLIM[12:0] – 1) x 0x800] + Offset.

2: In Dual Partition modes, each partition has a dedicated Alternate Interrupt Vecto enabled).

6. 부트로더와 애플리케이션 병합된 Hex 만들기

다음은 부트로더와 애플리케이션을 한 번에 Write 하기 위한 병합된 Hex 만드는 방법에 대한 가이드이다.

Hexmate 툴을 통해 4. 메모리맵을 참고하여 아래의 명령과 같이 병합된 Hex 파일을 만들 수 있다.
Hexmage 는 MPLABX 하위 폴더에 있다.
("C:₩Program Files₩Microchip₩MPLABX₩v6.05₩mplab_platform₩bin₩hexmate.exe")

명령은 아래와 같이 argument 로 Byte 주소,hex 파일 형태로 입력하고 마지막 출력파일을 지정한다. 하단 입력 명령어처럼 여러 인자들을 space 구분하여 한 줄로 cmd 에 입력한다.

<명령어 형식>

hexmate.exe

r0-3FFF,dspic33ck256mp508_boot.X.production.hex

r4000-56FFF,dspic33ck256mp508_app.X.production.hex

r57000-FFFFFFFF,dspic33ck256mp508_boot.X.production.hex

-Ocombined.hex

<CMD 에 입력 명령어>

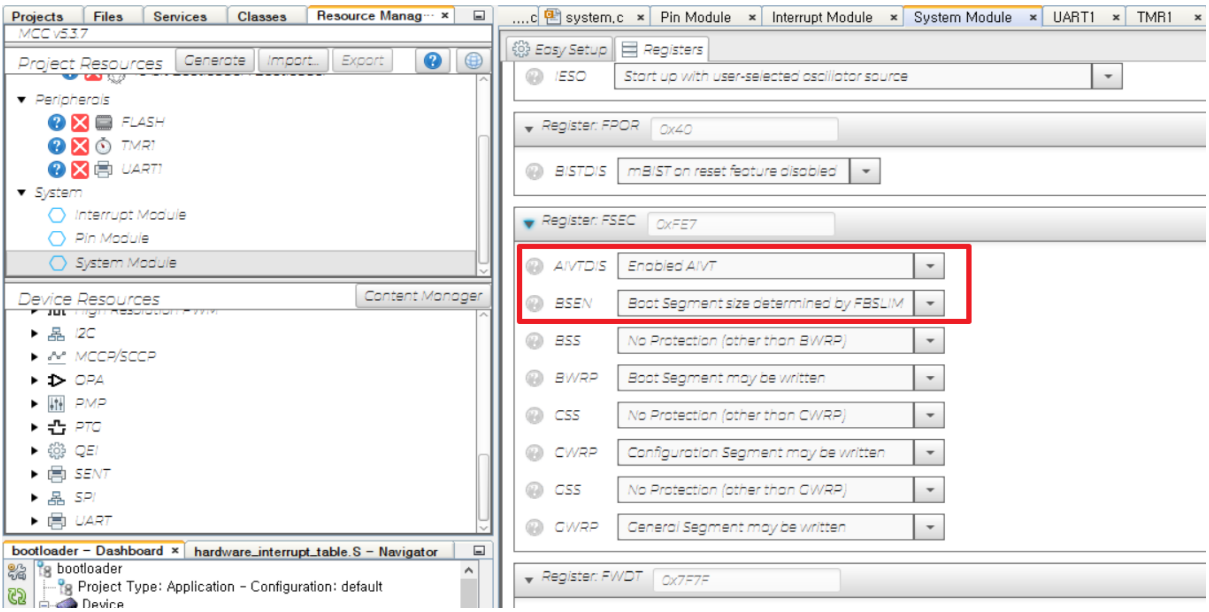
hexmate.exe r0-3FFF,dspic33ck256mp508_boot.X.production.hex r4000-56FFF,dspic33ck256mp508_app.X.production.hex r57000-FFFFFFFF,dspic33ck256mp508_boot.X.production.hex -Ocombined.hex

7. 코드 변경 점

이번 장에서는 MCC 기본 코드 생성할 때 AIVT configuration register 설정 방법과, 생성된 코드에서 AIVT 운용을 위해 필요한 수정들에 대해 다룬다. 제공된 예제 코드의 AIVT 적용 전/후 프로젝트들이 있으니 이를 참고하면서 다음의 설명을 참고하면 도움이 될 것 입니다.

1) MCC 를 통한 AIVT Enable 설정

아래는 AIVT 를 설정을 위한 Configuration Register 설정이다. FSEC reg 의 AIVTDIS 과 BSEN 을 ON 하고, FBSLIM reg 의 BSLIM 을 설정한다. 먼저 AIVT 사용을 위한 Configuration 구성을 먼저 한 후 RESET 후에는 기본적으로 Interrupt IVT 로 동작하기 때문에 부트로더는 IVT 테이블을 사용한다. 이후 애플리케이션으로 Jump 시 INTCON2.AIVTEN=1 을 설정하여, Interrupt Controller 가 AIVT 주소를 이용하도록 컨트롤 한다.



2) 인터럽트 핸들링 코드 수정

부트로더에서 애플리케이션 진입과정에서는 인터럽트를 disable 에 애플리케이션 초기화 중에 인터럽트 발생으로 인한 잠재적 이슈를 미연 방지한다.

dspic33ck256mp508_boot.X₩mcc_generated_files₩boot₩boot_demo.c

```
void BOOT_DEMO_Tasks(void)
{
    //생략 application jump 조건에서 interrupt disable 추가
    __builtin_disable_interrupts();
    BOOT_StartApplication();
}
```

애플리케이션에서는 인터럽트 활성화 전 INTCON2 reg 에 AIVTEN 를 설정하여 AIVT 테이블로 동작하도록 설정해야 한다.

dspic33ck256mp508_app.X₩mcc_generated_files₩system.c

```
void SYSTEM_Initialize(void)
{
    PIN_MANAGER_Initialize();
    CLOCK_Initialize();
    INTERRUPT_Initialize();
    TMR1_Initialize();
    //Chris Enable AIVTEN
    INTCON2bits.AIVTEN = 1;
    INTERRUPT_GlobalEnable();
    SYSTEM_CORCONModeOperatingSet(CORCON_MODE_PORVALUES);
}
```

3) AIVT Enable 후 Linker 에게 다음의 additional option 을 통해 알려줘야 한다.

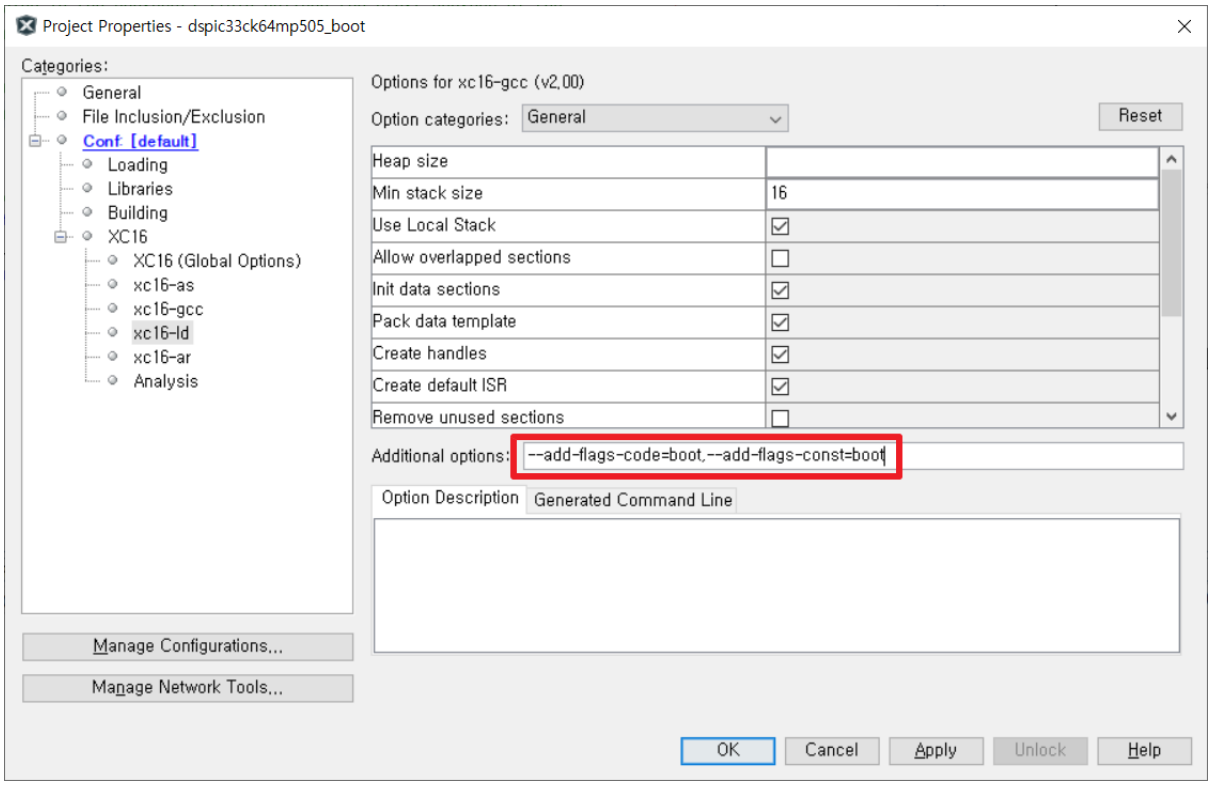
<bootloader 의 경우>

--add-flags-code=boot,--add-flags-const=boot,-D__USE_BFA

boot 는 BS 영역에 배치한다는 의미이다, 해당 option 미 적용 시 BSLIM 에 코드가 배치된다.

__USE_BFA define 은 GLD script 에 정의되어 있는 .text optimization 을 하지 않는다는 의미이다.

해당 옵션 미적용 시 .text 영역이 BSLIM 쪽에 배치된다.



<application 의 경우>

추가 옵션 설정 안해도 되지만, 부트로더 처럼 앞쪽 주소로 코드를 배치하기 위해 아래 define 은 적용한다.

-D__USE_BFA

4) AIVT enable 시 부트로더와 애플리케이션은 독립적으로 동작해야 한다. 서로의 주소 영역을 reserve 하는 .space linker script 를 비활성화 한다.

해당 수정이 없으면 링커쪽에서 allocation 에러가 발생한다.

dspic33ck256mp508_boot.X₩mcc_generated_files₩boot₩memory_partition.S

```
#if 0//chris
#include "boot_config.h"

    .section *, code, address(BOOT_CONFIG_PROGRAMMABLE_ADDRESS_LOW), noload, keep
reserved_application_memory:
    .space 0x2BEFE   -   BOOT_CONFIG_PROGRAMMABLE_ADDRESS_LOW, 0x00
#endif
```

dspic33ck256mp508_app.X₩mcc_generated_files₩boot₩memory_partition.S

```
#if 0//Chris
    /* Reserve the memory used by the bootloader */
    .section *, code, address(PROGRAM_MEMORY_ORIGIN), noload, keep
boot_loader_memory:
    .space (BOOT_CONFIG_PROGRAMMABLE_ADDRESS_LOW   - PROGRAM_MEMORY_ORIGIN),
0x00
#endif
```

5) AIVT 사용으로 부트로더와 인터럽트는 별도의 Interrupt Vector Table 을 사용한다. 때문에 Interrupt remapping 을 위한 script 는 더 이상 사용하지 않는다.

dspic33ck256mp508_boot.X₩mcc_generated_files₩boot₩hardware_interrupt_table.S

```
#if 0//Chris START
#include "boot_config.h"



... 생략 ...

/*
 * IVT section information.
 */
    .section .ivt, code, address(0x4), keep

    #define REMAP_TABLE
    .include "mcc_generated_files/boot/interrupts.S"
#endif
```

dspic33ck256mp508_app.X₩mcc_generated_files₩boot₩hardware_interrupt_table.S

```
#if 0//Chris START
#include "boot_config.h"

... 생략 ...

.include "mcc_generated_files/boot/interrupts.S"
#endif//Chris END
```

dspic33ck256mp508_app.X₩mcc_generated_files₩boot₩user_interrupt_table.S

```
#if 0//Chris START
#include "boot_config.h"

... 생략 ...

.section .ivt, code, address(0x4), keep

    #define REMAP_TABLE
    .include "mcc_generated_files/boot/interrupts.S"
#endif //Chris END
```

6) 애플리케이션 RESET 번지 주소를 수정

앞서 메모리 맵에서 본 것과 같이 AIVT 테이블은 0x2000 번지에 배치되고, Application 의 진입점은 0x2800 이다. 때문에 Application Reset Address 를 0x2800 번지로 수정한다.

dspic33ck256mp508_boot.X₩mcc_generated_files₩boot₩boot_config.h

```
#define BOOT_CODE_OFFSET_AIVT_ENABLED 0x800//Chris
//#define BOOT_CONFIG_APPLICATION_RESET_ADDRESS (BOOT_CONFIG_PROGRAMMABLE_ADDRESS_LOW +
BOOT_CONFIG_VERIFICATION_APPLICATION_HEADER_SIZE)
#define BOOT_CONFIG_APPLICATION_RESET_ADDRESS (BOOT_CONFIG_PROGRAMMABLE_ADDRESS_LOW +
BOOT_CONFIG_VERIFICATION_APPLICATION_HEADER_SIZE + BOOT_CODE_OFFSET_AIVT_ENABLED)//Chris
```

dspic33ck256mp508_app.X₩mcc_generated_files₩boot₩boot_config.h

```
#define BOOT_CODE_OFFSET_AIVT_ENABLED 0x800//Chris
//#define BOOT_CONFIG_APPLICATION_IMAGE_APPLICATION_HEADER_ADDRESS (BOOT_CONFIG_PROGRAMMABLE_ADDRESS_LOW)
#define BOOT_CONFIG_APPLICATION_IMAGE_APPLICATION_HEADER_ADDRESS (BOOT_CONFIG_PROGRAMMABLE_ADDRESS_LOW +
BOOT_CODE_OFFSET_AIVT_ENABLED)//Chris
```