# 이중 파티션 플래시 프로그램 메모리

## 주요 내용

이 매뉴얼 섹션에서는 다음과 같은 주요 주제를 다루고 있습니다.


1.0 소개 ..... 2
2.0 프로그램 메모리 아키텍처 ..... 2
3.0 프로그램 메모리 파티션 플래시 작업 ..... 7
4.0 플래시 메모리 프로그래밍 ..... 13
5.0 프로그램 공간 가시성 및 확장 데이터 공간(PSV 및 EDS) ..... 25
6.0 레지스터 맵 ..... 26
7.0 관련 애플리케이션 정보 ..... 27
8.0 개정 내역 ..... 28


© 2012-2017 Microchip Technology Inc.

# 1.0 소개

모든 PIC24 및 dsPIC33 디바이스에는 사용자 코드 실행을 위한 내부 프로그래밍 가능 플래시 어레이가 있습니다. 탁월한 내구성의 플래시 어레이가 코드 개발과 저장에 있어 뛰어난 유연성을 제공하며 긴 보존 기간과 다수의 읽기/쓰기 주기를 갖추었습니다.

이 버전의 플래시 프로그램 메모리는 다음과 같은 새로운 기능을 추가합니다.

- 이중 파티션 플래시 작업 - 코드 보안 강화 옵션과 함께 애플리케이션 코드의 오류 대비용 저장 및 강력한 부팅 로더 시스템 지원
- 라이브 업데이트 작업 - 주 애플리케이션이 계속 실행되는 동안 비활성 코드 세그먼트 수정 또는 완전 지우기 허용
- 데이터 RAM 공간에서 플래시 어레이의 직접 런타임 프로그래밍 및 데이터 RAM 이미지의 선택적 압축

# 2.0 프로그램 메모리 아키텍처

PIC24 및 dsPIC33 디바이스는 그림 2-1과 같이 4M x 24비트 프로그램 메모리 주소 공간의 주소를 지정합니다. 프로그램 메모리 맵은 사용자 프로그램 공간(000000h ~ 7FFFFFh)과 구성 또는 테스트 메모리 공간(800000h ~ FFFFFFh)으로 똑같이 나뉩니다.

사용자 프로그램 공간에는 리셋 벡터, IVT(인터럽트 벡터 테이블) 및 프로그램 메모리가 포함됩니다. 다음과 같은 세 가지 방법으로 프로그램 공간에 액세스할 수 있습니다.

1. 23비트 PC(프로그램 카운터)
2. 테이블 읽기(TBLRD) 및 테이블 쓰기(TBLWT) 명령
3. 프로그램 메모리의 32Kbyte 세그먼트를 데이터 메모리 주소 공간에 매핑

구현된 프로그램 메모리를 리셋 및 인터럽트 벡터를 포함하는 벡터 영역과 플래시 구성 데이터도 포함하는 코드 영역으로 세분할 수 있습니다. 사용자 프로그램 공간의 구현되지 않은 영역(예: 프로그램 메모리의 구현된 상한 위)에 액세스하면 주소 오류 트랩이 발생합니다.

## 2.1 벡터 영역

벡터 영역은 프로그램 메모리 공간의 시작 부분인 000000h에서 시작합니다. 여기에는 구현된 모든 하드웨어 인터럽트에 대한 마스터 리셋 벡터, 하드웨어 트랩 벡터 및 IVT(인터럽트 벡터 테이블)가 들어 있습니다.

아키텍처의 차이와 IVT의 크기 때문에 벡터 영역은 다른 디바이스 제품군에서 다른 메모리 양을 차지합니다. PIC24 디바이스의 경우 벡터 영역은 0000FEh로 확장됩니다. dsPIC33 디바이스의 경우 벡터 영역은 0001FEh로 확장됩니다. 그림 2-2에 서로 다른 디바이스에 대한 IVT 간 차이가 나와 있습니다. 디바이스 제품군에 관계없이 하드웨어 인터럽트 벡터는 항상 000014h에서 인터럽트 벡터 0으로 시작합니다.

CodeGuard™ 보안 구현에서 벡터 영역은 VS(벡터 세그먼트)에 해당합니다. 보안 구성에 따라 벡터 영역이 BS(부팅 세그먼트)나 GS(일반 세그먼트)의 일부로 처리될 수 있습니다.

### 2.1.1 대체 벡터 인터럽트 테이블

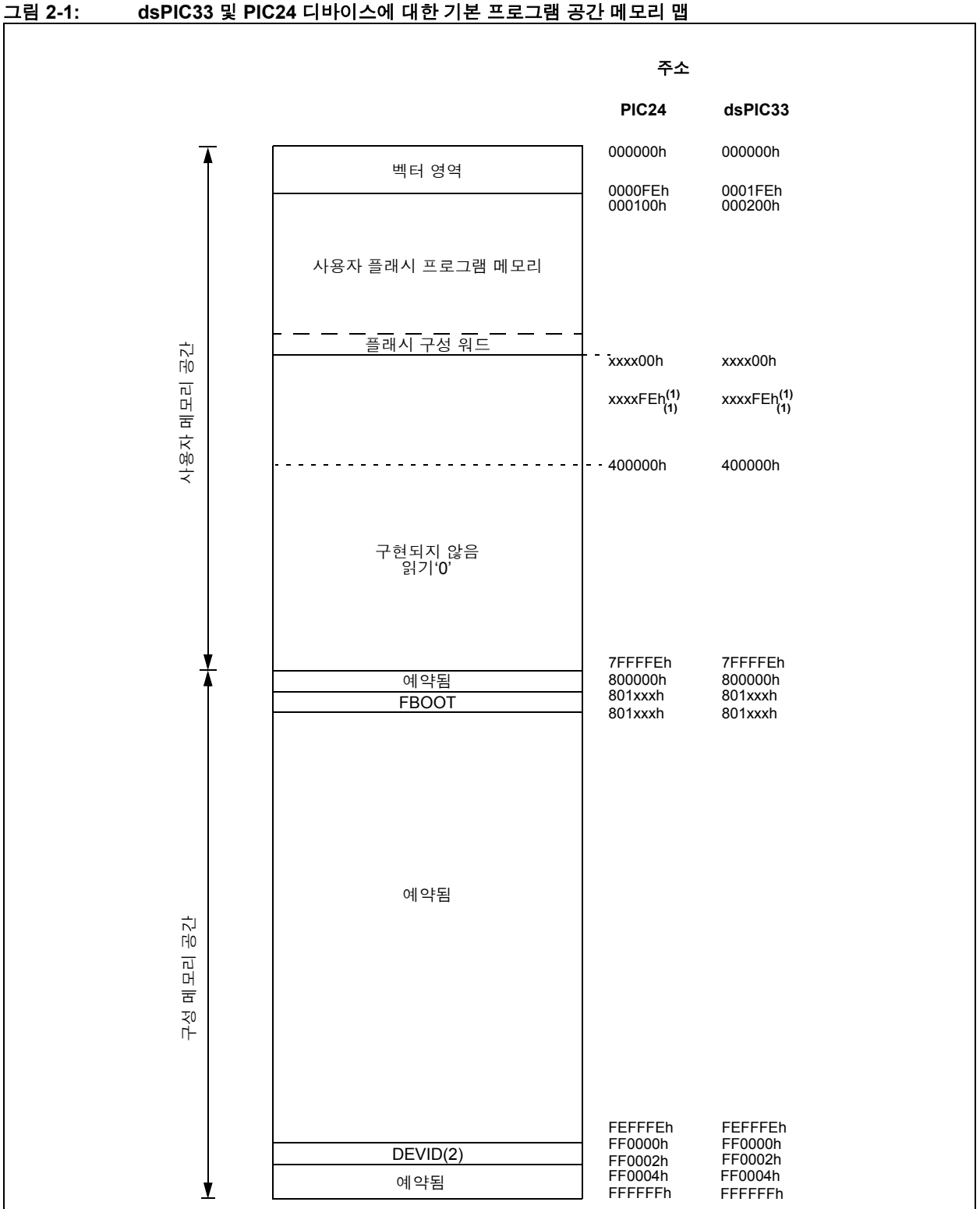
모든 dsPIC33 및 PIC24 디바이스는 높은 수준의 보안 코드 애플리케이션과 대체 예외 처리에 사용할 수 있는 AIVT(대체 IVT) 구현을 제공합니다. 이러한 제품군의 이전 디바이스와 달리 AIVT는 프로그램 메모리의 고정된 주소 범위에 영구 할당되지 않습니다. 대신 AIVT는 다음 조건에 모두 해당하는 경우에만 있습니다.

- 크기가 2페이지 이상(FBSLIM 구성 레지스터에서 세트)인 부팅 세그먼트에 대해 CodeGuard 보안이 구성됨
- AIVTDIS 구성 bit를 '0'으로 프로그래밍하여 AIVT가 활성화됨

AIVT가 활성화된 경우 AIVT는 BS의 마지막 페이지의 시작 부분에서 시작하는 주소 범위에 위치하며, 각 벡터는 페이지 경계로부터 고정 오프셋에 위치합니다. AIVT의 총 크기와 내용(예: 벡터 순서)은 IVT의 총 크기와 내용을 미러링합니다.

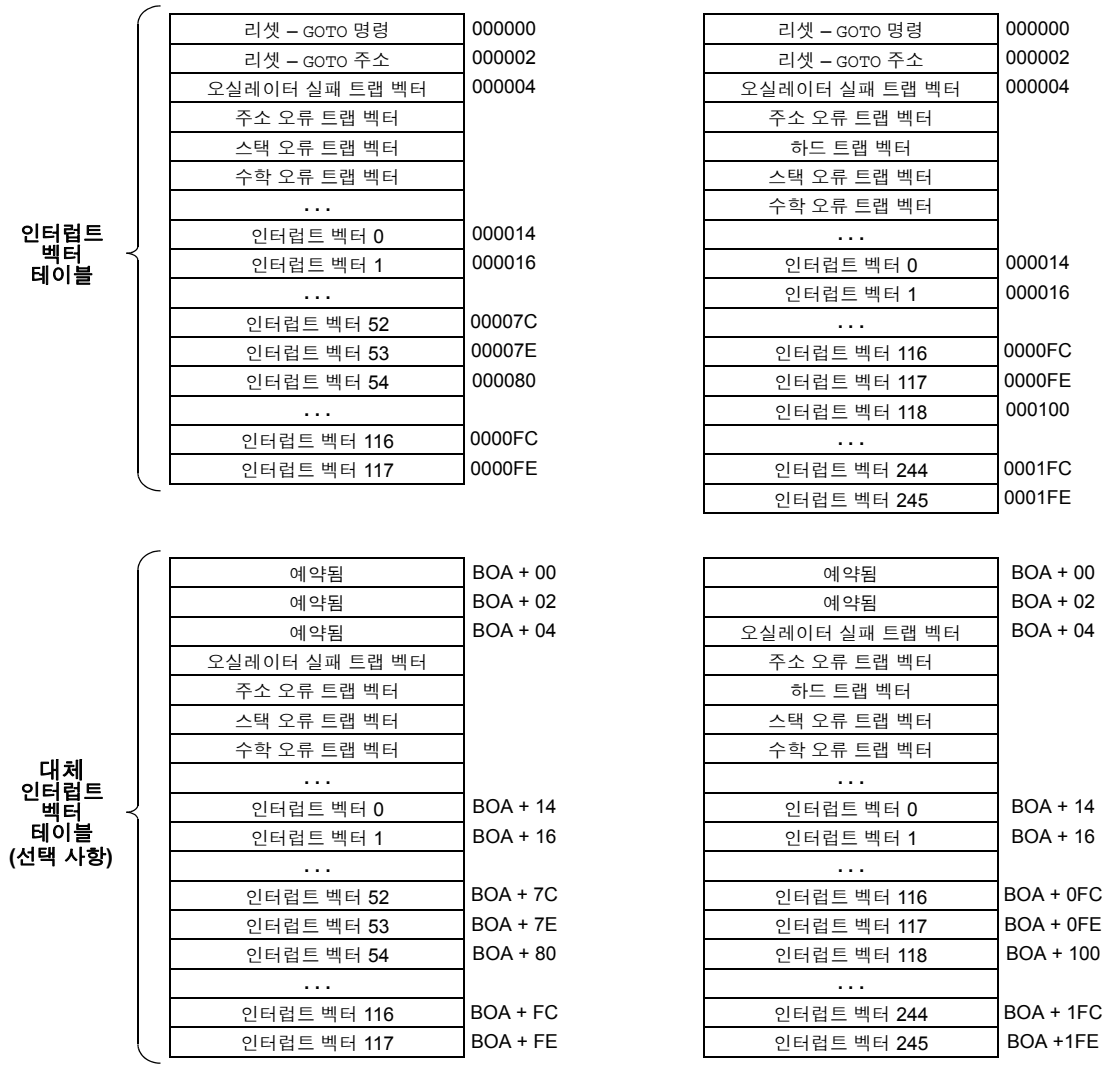

© 2012-2017 Microchip Technology Inc.

**그림 2-1: dsPIC33 및 PIC24 디바이스에 대한 기본 프로그램 공간 메모리 맵**




**Legend:** 메모리 영역은 확장되어 표시되지 않습니다.
**Note 1:** 정확한 경계 주소는 구현된 프로그램 메모리의 크기로 결정됩니다.

**그림 2-2: 벡터 영역 세부 정보**




**Legend:** BOA = 기준 오프셋 주소, 부팅 세그먼트의 마지막 페이지의 시작 주소. 주소는 16진수로 표시됩니다.

**Note:** 표시된 벡터 영역 조직은 지정된 아키텍처에 대한 기본 조직입니다. 디바이스별로 다를 수 있습니다. 디바이스별 세부 정보는 디바이스 데이터 시트를 참조하십시오.


© 2012-2017 Microchip Technology Inc.

## 2.2 코드 영역

코드 영역은 사용자의 애플리케이션 코드가 들어 있는 사용자 프로그램 메모리 영역으로, 벡터 영역의 끝 부분에서 플래시 구성 워드의 시작 부분까지 이어집니다. 부팅 세그먼트가 구현된 경우 벡터 영역의 끝 부분에서 시작하여 미리 결정된 범위로 확장됩니다. 부팅 세그먼트에 없는 코드 영역 부분은 CodeGuard 보안 시스템의 GS(일반 세그먼트)에 해당합니다.

아래 설명과 같이, 구현된 메모리의 끝 부분에 있는 플래시 구성 워드를 제외한 전체 영역을 애플리케이션 코드에 사용할 수 있습니다.

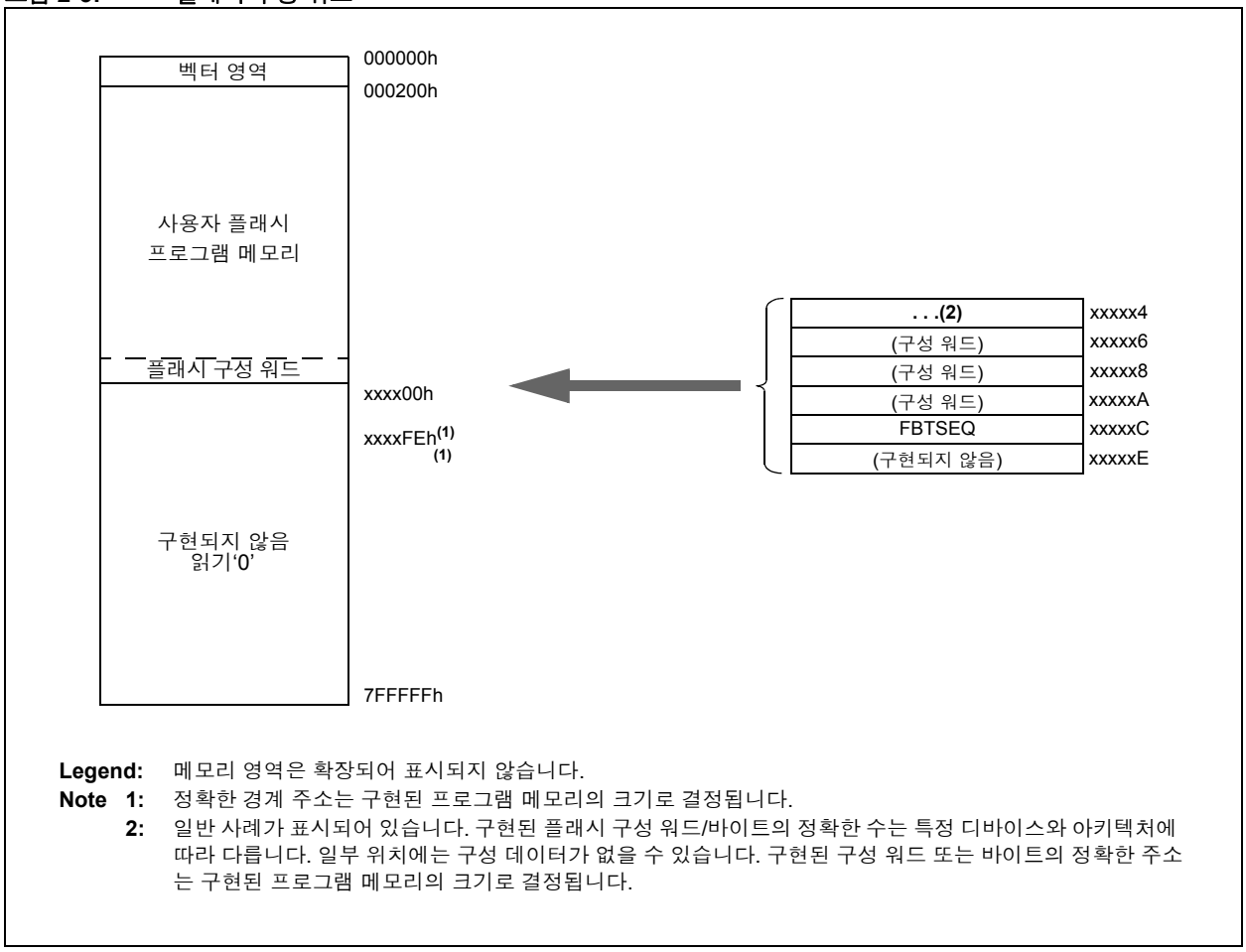
### 2.2.1 플래시 구성 데이터

구현된 플래시 프로그램 메모리의 끝 부분에 있는 영역(대개 마지막 행)은 플래시 구성 데이터용으로 예약되어 있습니다. 디바이스 리셋 시 이 구성 정보는 사용자가 액세스할 수 없는 적절한 디바이스 구성 레지스터로 복사됩니다. 플래시 구성 워드에 원하는 값을 프로그래밍하는 방식으로만 디바이스 구성 데이터를 프로그래밍할 수 있습니다.

구성 bit의 수, 순서 및 조직은 디바이스 아키텍처와 동일한 아키텍처 내의 디바이스 제품군마다 다릅니다. 기능을 기준으로 그룹화되는 16비트 구성 워드로 구성 bit를 조직하는 디바이스가 있고, 개별적으로 주소를 지정할 수 있는 구성 바이트를 기준으로 구성 bit를 조직하는 디바이스도 있습니다. 그림 2-3에는 구성 워드용으로 조직된 영역이 나와 있습니다. 제품군별 정보는 디바이스 데이터 시트를 참조하십시오.

이중 파티션 기능이 있는 디바이스의 경우 FBTSEQ 구성 워드는 대개 구현된 프로그램 메모리의 끝 부분에 위치한, 끝에서 두 번째 구성 워드입니다.

**그림 2-3: 플래시 구성 워드**




**Legend:** 메모리 영역은 확장되어 표시되지 않습니다.

**Note 1:** 정확한 경계 주소는 구현된 프로그램 메모리의 크기로 결정됩니다.

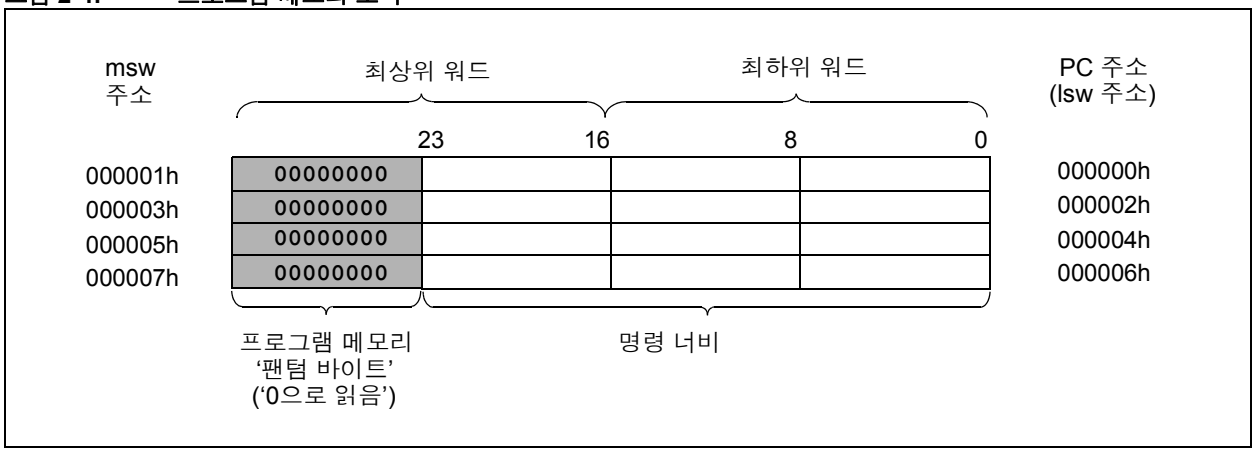
**2:** 일반 사례가 표시되어 있습니다. 구현된 플래시 구성 워드/바이트의 정확한 수는 특정 디바이스와 아키텍처에 따라 다릅니다. 일부 위치에는 구성 데이터가 없을 수 있습니다. 구현된 구성 워드 또는 바이트의 정확한 주소는 구현된 프로그램 메모리의 크기로 결정됩니다.

## 2.3 메모리 조직

프로그램 메모리 공간은 워드 주소 지정 가능 블록으로 조직됩니다. 24비트 너비로 처리되지만 프로그램 메모리의 각 주소를 하위 및 상위 워드(상위 워드의 상위 바이트가 구현되지 않음)로 보는 것이 더 적절합니다. 하위 워드는 항상 짝수 주소를 가지며, 상위 워드는 홀수 주소(그림 2-4)를 가집니다.

프로그램 메모리 주소와 PC는 항상 하위 워드에 워드 정렬됩니다(예: LSb(최하위 bit)는 항상 '0'임). 코드 실행 중 주소가 2씩 증가 또는 감소합니다.

**그림 2-4: 프로그램 메모리 조직**




### 2.3.1 프로그램 메모리 주소 지정

일반 코드 실행의 경우 PC(프로그램 카운터)가 실행을 위한 EA(유효 주소)를 제공합니다. PC는 23비트 너비로 사용자 프로그램 공간의 모든 위치에 대한 직접 액세스를 허용합니다. PC<0>은 프로그램 명령 정렬 유지를 위해 '0'으로 고정됩니다. PC는 PC<1>을 증분하여 PC의 값을 2씩 늘리는 방식을 통해 다음 순차 주소로 증분됩니다.

테이블 읽기 및 테이블 쓰기 작업의 경우 W 레지스터 중 하나의 16비트 주소와 TBLPAG 레지스터의 8비트 주소를 연결하여 EA를 만듭니다. 이를 통해 사용자 공간과 구성 공간 모두에 대한 테이블 작업 액세스가 가능합니다. 테이블 작업을 위한 주소 생성은 **Section 4.2.1 "테이블 작업의 주소 생성"**에서 더 자세히 다룹니다.

EDS(확장 데이터 공간) 및 PSV(프로그램 공간 가시성) 작업의 경우 W 레지스터의 하위 15비트를 DSRPAG/DSWPAG(dsPIC33) 또는 PSVPAG(PIC24F) 레지스터의 8비트 주소와 연결하여 EA를 만듭니다. 확장 데이터 공간과 프로그램 공간 가시성 작업은 *"dsPIC33/PIC24 제품군 참고 자료"*, **"데이터 메모리"**(dsPIC33, DS70595) 및/또는 **"데이터 메모리"**(PIC24, DS39717)에서 다룹니다.


© 2012-2017 Microchip Technology Inc.

## 3.0 프로그램 메모리 파티션 플래시 작업

이중 파티션 플래시 기능이 있는 디바이스의 경우 FBOOT 구성 워드의 BTMODE<1:0> bit를 프로그래밍하여 이중 파티션 프로그램 메모리 모드를 선택합니다. 다른 구성 워드와 달리 FBOOT는 다른 모든 플래시 구성 레지스터와 별도로 구성 메모리 공간에 있습니다. 정확한 주소는 특정 아키텍처(예: PIC24 또는 dsPIC33)와 관련이 있으며 디바이스 제품군에 따라 다를 수 있습니다. 표 3-1에는 다음 섹션에서 다루는 플래시 파티션 옵션이 나열되어 있습니다.

In-Circuit Serial Programming™(ICSP™)을 통해 디바이스가 처음 프로그래밍될 때 프로그래머가 FBOOT를 프로그래밍하여 디바이스 플래시 파티션 모드를 올바르게 세트해야 합니다. RTSP(런타임 셀프 프로그래밍)를 사용하여 실행 시 FBOOT를 다시 프로그래밍할 수는 없습니다. 프로그래머가 ICSP 모드에서 FBOOT bit를 구성해야 합니다. 이는 플래시 구성 워드의 위치가 표준 파티션 모드에서 이중 파티션 모드로 바뀌어 예기치 않은 디바이스 작업을 유발하기 때문입니다.

**표 3-1: 플래시 파티션 옵션**

| BTMODE<1:0> | 파티션 옵션 |
|---|---|
| 11 | 표준 모드(단일 파티션, 기본) |
| 10 | 이중 파티션 모드 |
| 01 | 보호되는 이중 파티션 모드 |
| 00 | 권한 있는 이중 파티션 모드(1) |

**Note 1:** 모든 이중 파티션 디바이스에 구현되어 있지는 않음

### 3.1 표준(단일 파티션) 모드

단일 파티션 또는 표준 파티션 모드라고도 하는 표준 모드는 프로그램 메모리의 기본 작동 모드입니다. BTMODEx 구성 bit가 '11' (프로그래밍되지 않은 구성)일 때 표준 모드가 선택됩니다. 표준 모드는 모든 이전 dsPIC33 및 PIC24 디바이스에서 사용할 수 있는 단일 프로그램 메모리 작동 모드이기도 합니다.

표준 모드에서 전체 사용자 프로그램 메모리는 000000h에서 구현된 플래시 메모리 상한 사이의 플랫 연속 메모리 공간으로 매핑됩니다. 예를 들어, 256Kbyte의 플래시 메모리가 있는 디바이스의 경우 프로그램 메모리 주소 범위는 000000h ~ 02AFFFh이며 이 범위 위의 주소는 구현되어 있지 않습니다. 구현된 전체 메모리 범위(리셋 벡터, IVT 및 플래시 구성 워드용으로 예약된 공간 제외)는 사용자의 애플리케이션에 사용할 수 있습니다. 세그먼트 코드 보안 기능이 있는 디바이스에서는 부팅 세그먼트도 구현할 수 있습니다.

### 3.2 이중 파티션 모드

BTMODEx 구성 bit가 '11' 이외의 값으로 프로그래밍된 경우 디바이스가 세 가지 이중 파티션 모드 중 하나로 작동합니다. 이들 모드에서는 구현된 플래시 메모리가 000000h에서 시작하는 활성 파티션과 400000h에서 시작하는 비활성 파티션의 두 영역으로 대칭 분할됩니다. 위의 예에 나와 있는 디바이스의 경우 256Kbyte의 플래시 메모리가 각각 128Kbyte의 두 영역(주소 000000h ~ 0157FFh 및 400000h ~ 4157FFh)으로 구현됩니다. 두 영역 사이의 주소는 구현되어 있지 않습니다(그림 3-1 참조).

이중 파티션 모드에서는 두 개의 독립 애플리케이션을 디바이스로 프로그래밍할 수 있습니다(파티션 1과 파티션 2라는 두 개의 플래시 메모리 파티션 각각에 하나씩). 디바이스가 초기화되면 이들 중 하나가 활성 파티션에 동적으로 매핑되고 실행됩니다. 다른 애플리케이션은 비활성 파티션에 매핑되어 프로그램 메모리 작업에 사용 가능합니다. 활성 파티션과 비활성 파티션 중 어디에 파티션을 할당하는지는 부팅 시퀀스 번호라는 코드 시그니처에 의해 자동으로 결정됩니다. 소프트웨어 컨트롤하에 실행 중 코드 파티션을 활성 파티션과 비활성 파티션 사이에서 스와핑할 수도 있습니다.

이중 파티션 모드에서는 활성 파티션의 애플리케이션이 비활성 파티션의 프로그램 데이터에 액세스(실행 불가)하거나 비활성 파티션을 다시 프로그래밍할 수 있습니다. 비활성 파티션의 플래시 메모리에 쓰는 경우 플래시 쓰기가 실행되는 동안 CPU를 정지할 필요가 없습니다. 따라서 중요한 컨트롤 기능이나 타이밍이 중요한 통신 실행이 애플리케이션 업데이트와 동시에 발생할 수 있는 라이브 업데이트 기능 실행이 가능합니다. 특정 이중 파티션 모드는 코드 보안과 강력한 작업을 위해 프로세스에 추가 제한을 둡니다. 비활성 파티션에 매핑된 코드는 실행할 수 없습니다. 파티션을 스와핑할 수 있지만 활성 파티션의 코드만 실행할 수 있습니다.

### 3.2.1 이중 파티션 모드

가장 간단한 이중 파티션 모드는 활성 파티션에서 파티션 1 또는 2의 코드로의 작업에 제한을 두지 않습니다. 다른 파티션의 코드 세그먼트 간 상호 작용에 대한 모든 제한은 고급 보안 기능의 구성으로 결정됩니다.

### 3.2.2 보호되는 이중 파티션 모드

보호되는 이중 파티션 모드는 기본 코드 세그먼트(파티션 1)를 플래시 쓰기 또는 지우기 작업으로부터 보호합니다. 이는 유사 시 대기 백업 이미지를 파티션 1에 저장할 수 있게 하여 “공장 기본값” 모드 구현을 가능하게 합니다.

보호되는 이중 파티션 모드를 사용하면 비활성 파티션에 있는 동안 플래시 메모리 작업에서 파티션 1을 쓰거나 지울 수 없습니다. 파티션 1도 구성 bit 세팅을 통해 쓰기 금지된 경우 지우거나 쓸 수 없습니다. 반대로 파티션에서의 작업에서 파티션 2를 지우거나 쓸 수 있습니다.

이는 유사 시 대기 백업 코드 이미지와 함께 유사 시 대기 부팅 로더를 파티션 1에 놓을 수 있게 합니다. 그러면 이 코드 이미지를 기본으로 실행하고 플래시 업데이트에 실패하는 경우 파티션 2를 다시 쓰는 데 사용할 수 있습니다.

### 3.2.3 권한 있는 이중 파티션 모드

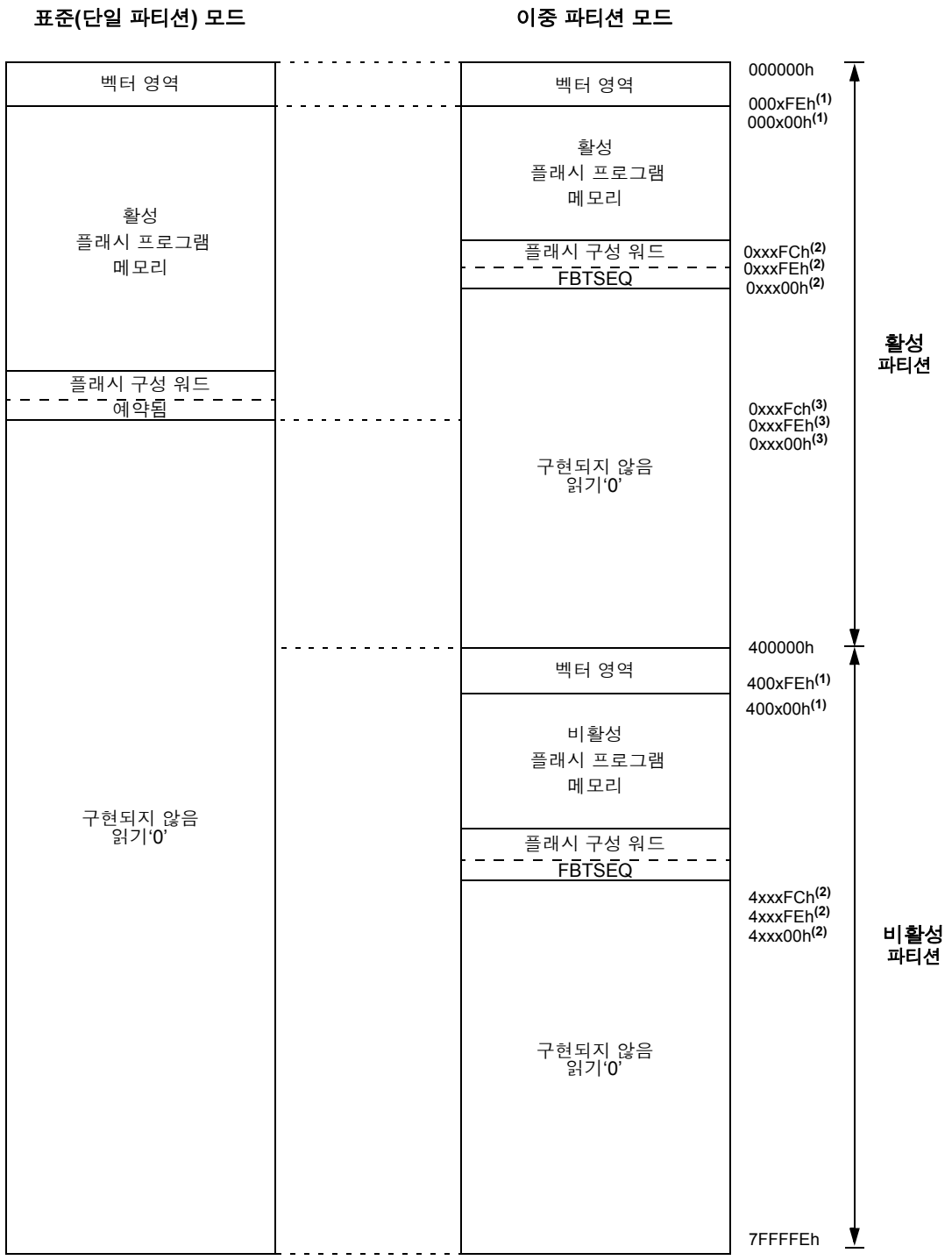
권한 있는 이중 파티션 모드는 다른 작성자가 애플리케이션의 코드 세그먼트를 쓸 수 있으며 이러한 세그먼트 중 하나에 대한 지적 재산 보호를 위한 보안 강화가 필요한 경우 추가 보안 보호를 구현합니다. 하드웨어의 애플리케이션 개발자가 비공개 타사 라이브러리가 포함된 대량의 코드를 쓰는 시스템을 예로 들 수 있습니다. 이 모드는 선택 디바이스에서 프로그램 메모리 공간의 다른 코드 세그먼트를 선택적으로 보호할 수 있는 고급 보안 기능과 작동하도록 설계되었습니다.

권한 있는 이중 파티션 모드는 두 파티션의 BSLIMx 구성 bit에 특별한 보호를 추가한다는 점에서 표준 이중 파티션 모드와 다릅니다. 이 보호 기능은 bit를 효과적으로 잠그며 부팅 세그먼트와 일반 세그먼트의 크기를 변경하지 못하게 합니다. 적절한 보안 세팅과 함께 실행 시 세그먼트를 수정하거나 예기치 않게 읽지 못하게 합니다.

이중 파티션 기능이 있는 디바이스에는 권한 있는 이중 파티션 모드가 구현되어 있지 않습니다. 자세한 내용은 디바이스별 데이터 시트를 참조하십시오.


© 2012-2017 Microchip Technology Inc.

**그림 3-1: 표준 및 이중 파티션 사용자 메모리 공간 맵**




**Legend:** 메모리 영역은 확장되어 표시되지 않습니다.

**Note 1:** 기본 벡터 영역 경계는 PIC24 디바이스의 경우 000100h이고, dsPIC33 디바이스의 경우 000200h입니다.

**2:** 이중 파티션 모드의 메모리 경계 값은 표준 파티션 모드의 값의 반입니다.

**3:** 정확한 프로그램 메모리 경계는 구현된 프로그램 메모리의 크기로 결정됩니다.

### 3.2.4 코드 파티션 선택

이중 파티션 모드에는 활성 파티션에 매핑하고 실행할 파티션을 결정하는 두 가지 방법인 부팅 시퀀스 번호와 BOOTSWP 명령이 있습니다. P2ACTIV bit(NVMCON<10>)를 사용하여 어느 물리적 파티션이 활성 파티션인지 결정할 수 있습니다. P2ACTIV = 1이면 파티션 2가 활성 상태이고, P2ACTIV = 0이면 파티션 1이 활성 상태인 것입니다.

부팅 시퀀스 번호는 디바이스 리셋 시 활성 파티션을 자동으로 결정하는 데 사용되는 12비트 값입니다. 각 파티션마다 FBTSEQ 플래시 구성 워드에 저장되는 고유한 부팅 시퀀스 번호가 필요합니다.

BOOTSWP 명령은 디바이스 리셋 없이 활성 파티션과 비활성 파티션을 스와핑하는 데 사용됩니다.

#### 3.2.4.1 부팅 시퀀스 번호

12비트 부팅 시퀀스 번호는 항상 사용자 프로그램 메모리의 마지막 위치의 다른 플래시 구성 워드 위에 있는 FBTSEQ 플래시 구성 워드에 저장됩니다(그림 3-2 참조). 프로그램 메모리 워드의 하위 16비트만 사용하는 다른 구성 레지스터와 달리 FBTSEQ는 전체 24비트 너비입니다. 일반 작동 조건하에서 각 파티션은 FBTSEQ에 대해 다른 값을 가져야 합니다. 이중 파티션 모드를 사용하지 않으면 FBTSEQ의 값이 무시됩니다.

부팅 시퀀스 번호는 bit 필드 BSEQx(FBTSEQ<11:0>)의 실제 값과 IBSEQx bit 필드(FBTSEQ<23:12>) 값의 1의 보수, 이렇게 두 부분에 저장됩니다. 디바이스 리셋 시 부팅 시퀀스 번호를 읽을 때 BSEQx와 IBSEQx의 값이 자동으로 비교됩니다. 이들 두 값이 서로를 보완하지 않는 경우 부팅 시퀀스 번호가 잘못된 것으로 간주됩니다. 보완 값은 하드웨어에서 자동으로 만들지 않으며 프로그래밍 시 하드웨어에서 확인하지 않습니다. 애플리케이션에서 적절한 값을 계산하고 프로그래밍해야 합니다.

디바이스 리셋 시 두 파티션의 부팅 시퀀스 번호가 비교됩니다. 하위 BSEQx 값의 파티션이 활성 파티션에 매핑되며 해당 코드가 실행됩니다. 부팅 시퀀스 번호 중 하나가 잘못된 경우 어느 부팅 시퀀스 번호가 더 낮은지에 관계없이 디바이스에서 유효한 부팅 시퀀스 번호의 파티션을 활성 파티션으로 선택합니다. 두 부팅 시퀀스 번호 모두 잘못된 경우 파티션 1이 활성 파티션으로 기본 선택됩니다.

프로그래밍 비활성 파티션의 부팅 시퀀스 번호를 더 낮은 값으로 다시 프로그래밍하여 실행 중 스와핑하도록 파티션을 준비할 수 있습니다. 리셋이 실행되면 더 낮은 값의 파티션이 활성화됩니다. 비활성 파티션이 업데이트되고 리셋 후 활성 파티션에 매핑될 때 이 방법이 사용됩니다.

FBTSEQ의 위치 덕분에 플래시 프로그램 메모리의 체크섬 또는 기타 확인에서 해당 파티션을 쉽게 제외할 수 있습니다. FBTSEQ 값은 대개 다른 파티션의 BSEQx를 기준으로 실행 시 결정되기 때문에 CRC 등의 체크섬에 포함할 수 없는 경우가 많습니다.

그림 3-3 위쪽의 시퀀스는 부팅 시퀀스 번호가 수정되고 디바이스 리셋이 실행될 때 코드 파티션 간의 관계를 보여줍니다.

**그림 3-2: FBTSEQ와 다른 구성 워드의 관계(이중 파티션 모드만)**

**Legend:** 음영 처리된 영역은 구현되지 않았으며 항상 '0'으로 읽습니다. 주소는 구현된 프로그램 메모리 상한의 상대 위치를 반영합니다.

**Note:** FBTSEQ와 다른 구성 워드가 꼭 가깝지는 않습니다. 여기 표시된 관계는 비교를 위한 것입니다.


© 2012-2017 Microchip Technology Inc.

#### 3.2.4.2 BOOTSWP 명령

BOOTSWP 명령은 PIC24 및 dsPIC33 명령 세트의 확장으로, 디바이스 리셋 없이도 활성 파티션과 비활성 파티션 사이에서 코드 세그먼트를 스와핑할 수 있게 하여 코드, 라이브 업데이트를 지원합니다. BOOTSWP 명령을 사용한 파티션 스와핑을 "소프트 스와핑"이라고 합니다. BOOTSWP 명령을 실행하려면 구성 bit인 BTSWP(FIDC<25>)를 클리어해야 합니다. BTSWP가 세트된 상태에서 BOOTSWP 명령을 시도하면 NOP 명령이 실행됩니다.

BOOTSWP 명령 뒤에는 항상 PC(예: GOTO W, CALL W 또는 BRA W)를 쓰는 싱글 워드 명령이 와야 합니다. 명령 대상은 현재 주소의 32Kbyte 내 주소에 있어야 합니다. 실행 시 활성 파티션과 비활성 파티션 교환이 이루어지고, PC가 새로운 활성 파티션의 GOTO 명령에서 지정하는 위치를 향합니다.

**Note:** LNK 명령을 사용하여 새 스택 프레임을 만든 기능 내에서 BOOTSWP 명령을 실행할 경우 BOOTSWP 명령 다음에 GOTO가 아닌 CALL을 사용해야 합니다. 그렇지 않으면 디바이스가 스택 오류 트랩을 생성합니다.

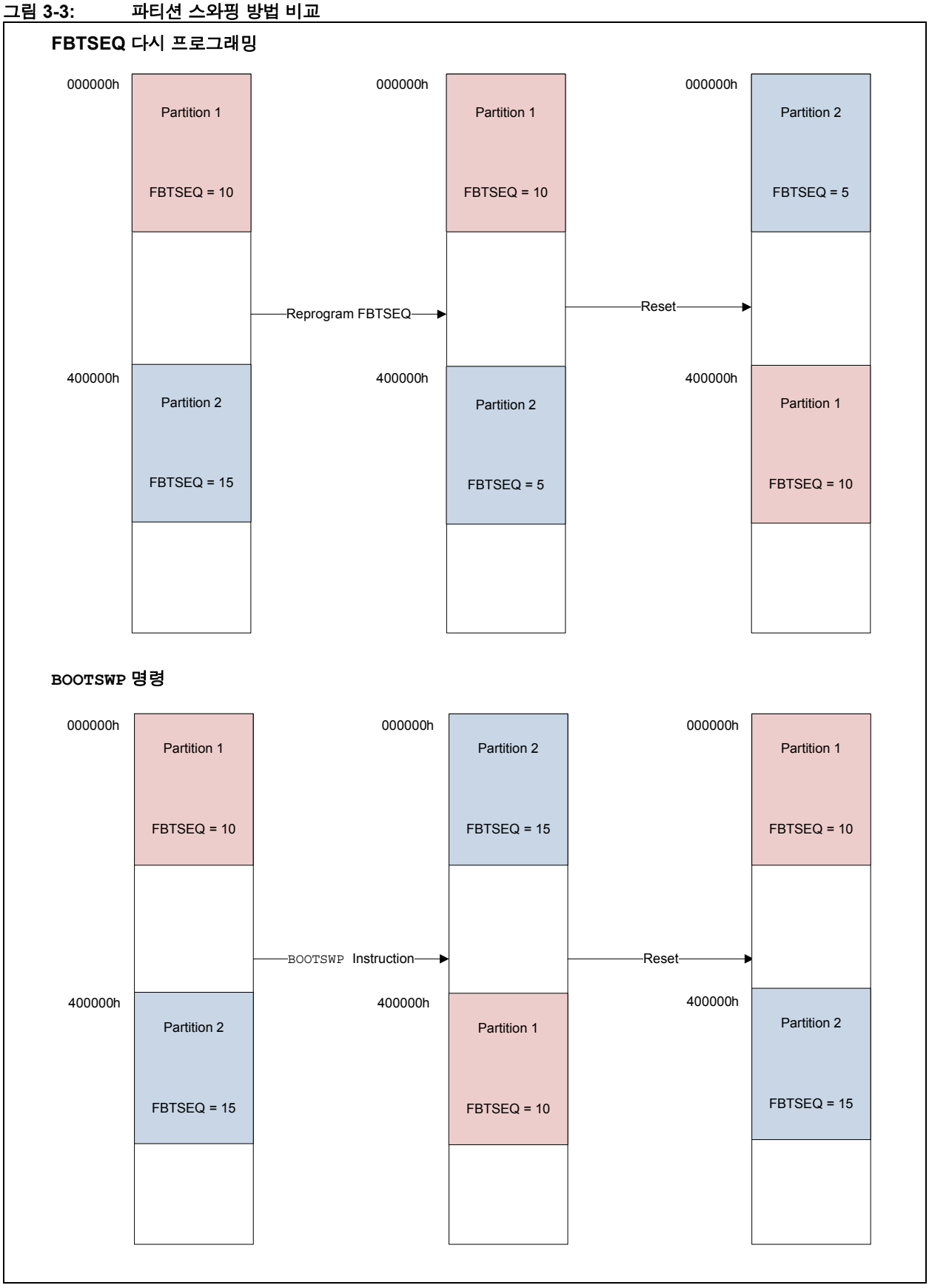
BOOTSWP 명령 실행 후 SFTSWP bit(NVMCON<11>)가 세트됩니다. 이 bit는 BOOTSWP 명령이 제대로 실행되었으며, 디바이스 리셋이 아닌 BOOTSWP를 통해 현재 활성 파티션이 시작되었음을 펌웨어에 알립니다. 상태 bit인 P2ACTIV(NVMCON<10>)를 읽어서 어느 파티션이 활성화되었는지 확인할 수도 있습니다.

파티션 스와핑 후 이전에 활성화된 모든 주변 장치와 인터럽트는 활성 상태로 유지됩니다. RAM과 스택도 스와핑 후 해당 상태를 유지합니다. 소프트 스와핑을 사용하는 애플리케이션의 경우 계속해서 예상대로 작동하려면 디바이스를 다시 초기화하는 루틴으로 애플리케이션이 점프하는 것이 좋습니다.

강력한 작업을 위해 BOOTSWP 명령(2개의 순차적 단계로 NVMKEY 레지스터에 55h 및 AAh 쓰기)을 실행하기 전에 표준 NVM 잠금 해제 시퀀스를 실행해야 합니다. 자세한 내용은 **Section 4.1 "레지스터"**를 참조하십시오. 잠금 해제 시퀀스를 실행하기 전에 인터럽트를 비활성화하는 것도 중요합니다. 잠금 해제 시퀀스를 수행하지 않으면 BOOTSWP가 강제 NOP로 실행됩니다. BOOTSWP 다음에 GOTO 명령이 계속 실행되어 PC가 현재 작동 파티션의 위치로 점프하게 합니다. 마찬가지로 BOOTSWP는 표준 파티션 모드에서 효과가 없습니다.

그림 3-3 아래쪽의 시퀀스는 BOOTSWP 명령이 실행될 때 파티션 간의 관계를 보여줍니다. BOOTSWP 파티션 변경은 일시적입니다. 다음 디바이스 리셋 후 더 낮은 부팅 시퀀스 번호의 파티션이 활성 파티션에 재할당됩니다.

그림 3-3: 파티션 스와핑 방법 비교





© 2012-2017 Microchip Technology Inc.

## 4.0 플래시 메모리 프로그래밍

다음 세 가지 방법 중 하나로 PIC24 및 dsPIC33 디바이스를 프로그래밍할 수 있습니다.

- RTSP(런타임 셀프 프로그래밍)
- ICSP™(회로 내 직렬 프로그래밍™)
- EICSP(고급 회로 내 직렬 프로그래밍)

디바이스에 직렬 데이터 연결을 사용하여 외부 프로그래머가 ICSP 및 EICSP를 수행하는 동안 실행 중에 애플리케이션 소프트웨어가 RTSP를 수행합니다. ICSP와 EICSP는 RTSP보다 훨씬 더 빠른 프로그래밍 시간을 지원합니다.

이 섹션에서는 RTSP 기술을 설명합니다. Microchip 웹 사이트(www.microchip.com)에서 다운로드할 수 있는 해당 디바이스에 대한 프로그래밍 사양 문서에 ICSP 및 EICSP 프로토콜이 정의되어 있습니다.

### 4.1 레지스터

프로그래밍 작업은 6개의 레지스터를 통해 컨트롤됩니다. NVMCON 및 NVMKEY 레지스터는 모든 작업을 활성화하고 선택하는 데 사용됩니다. 나머지 4개의 레지스터는 데이터 및 어드레스 포인터를 정의합니다.

| | |
|---|---|
| **Note:** | 모든 디바이스가 데이터 RAM 버퍼 프로그래밍을 구현하는 것은 아닙니다. 자세한 내용은 디바이스별 데이터 시트를 참조하십시오. |

#### 4.1.1 컨트롤 레지스터

NVMCON 레지스터(Register 4-1)는 모든 플래시 프로그래밍 작업을 컨트롤합니다. NVMOP<3:0> bit(NVMCOM<3:0>)는 수행할 특정 쓰기 또는 지우기 작업을 선택합니다. WR bit(NVMCOM<15>)는 적절한 작업을 트리거하며, 작업이 완료될 때까지 세트된 상태로 있다가 하드웨어에 의해 클리어됩니다. WREN bit(NVMCOM<14>)는 쓰기 및 지우기 작업을 활성화하거나 비활성화합니다. WREN이 클리어되면 작업을 트리거하도록 WR bit를 세트할 수 없습니다.

NVMKEY는 플래시 메모리를 실수로 쓰거나 지우지 않게 예방하는 데 사용되는 쓰기 전용 레지스터로서, 하위 바이트만 구현됩니다. 프로그램을 시작하거나 시퀀스를 지우기 위해 두 쓰기의 "잠금 해제" 시퀀스가 레지스터에서 수행됩니다.

1. NVMKEY<7:0>에 55h를 씁니다.
2. NVMKEY<7:0>에 AAh를 씁니다.

이 시퀀스 후 한 번의 명령 주기 동안 NVMCON 레지스터에 쓰기가 허용됩니다.

#### 4.1.2 주소 레지스터

NVMADRL 및 NVMADRH 레지스터는 쓰기 작업에 대한 시작 어드레스 포인터를 정의합니다. 두 유형의 프로그램 메모리 쓰기(래치 기반 및 RAM 버퍼) 모두 이들 레지스터를 사용하여 대상 주소를 세트합니다.

NVMSRCADRL 및 NVMSRCADRH 레지스터는 RAM 버퍼 프로그래밍을 사용할 때 소스 데이터의 데이터 RAM에 시작 주소를 정의합니다. NVMSRCADRH 레지스터는 EDS(확장 데이터 공간)가 있는 디바이스에서 확장 데이터 공간 메모리의 주소를 가리키는 데 사용됩니다.

**Register 4-1: NVMCON: 플래시 프로그래밍 컨트롤 레지스터**

| R/S-0(1) | R/C-0 | R/C-0 | R/W-0 | R/C-0 | R-0 | R/W-0 | R/C-0 |
|---|---|---|---|---|---|---|---|
| WR | WREN | WRERR | NVMPIDL(5) | SFTSWP | P2ACTIV | RPDF(2) | URERR(2) |
| bit 15 | | | | | | | bit 8 |

| U-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | NVMOP3 | NVMOP2 | NVMOP1 | NVMOP0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | S = 세트 가능 유일한 bit | C = 클리어 가능 유일한 bit | |
|---|---|---|---|
| R = 읽기 가능 bit | W = 쓰기 가능 bit | U = 비구현 bit, '0'으로 읽음 | |
| -n = POR의 값 | '1' = Bit가 세트됨 | '0' = Bit가 클리어됨 | x = 알려지지 않은 bit |

bit 15 **WR:** 쓰기 컨트롤 bit(1)
1 = 플래시 프로그램/지우기 작업 시작
0 = 프로그램/지우기 작업이 완료되고 비활성 상태임

bit 14 **WREN:** 프로그램/지우기 활성화 bit
1 = 프로그램/지우기 주기 허용
0 = 메모리 또는 퓨즈 요소의 프로그래밍/지우기 금지

bit 13 **WRERR:** 시퀀스 오류 플래그 bit
1 = 잘못된 프로그램/지우기 종료가 발생했거나 구현되지 않은 프로그래밍 작업이 선택됨
0 = 프로그램 또는 지우기 작업이 진행 중이거나, 정상적으로 완료되었거나, 이제 시작해야 함

bit 12 **NVMPIDL:** 유휴 활성화 시 NVM 전원 끄기 bit(5)
1 = 디바이스가 유휴 모드로 전환될 때 플래시 어레이에서 전원 제거
0 = 디바이스가 유휴 모드로 전환될 때 대기 모드에서 플래시 어레이의 전원을 켜둠

bit 11 **SFTSWP:** 소프트 스와핑 상태 bit
BTMODE<1:0> = 10 또는 0x일 경우
1 = BOOTSWP 명령을 사용하여 파티션이 성공적으로 스와핑됨
0 = BOOTSWP 명령을 사용하여 파티션 스와핑 성공 대기 중
BTMODE<1:0> = 11일 경우
구현되지 않음, '0'으로 읽음

bit 10 **P2ACTIV:** 이중 파티션 활성 상태 bit
BTMODE<1:0> = 10 또는 0x일 경우
1 = 파티션 2 플래시가 활성 파티션임
0 = 파티션 1 플래시가 활성 파티션임
BTMODE<1:0> = 11일 경우
구현되지 않음, '0'으로 읽음

bit 9 **RPDF:** RAM 버퍼 행 프로그래밍 데이터 형식 컨트롤 bit(2)
1 = 행 데이터가 RAM에 압축된 형식으로 저장됨
0 = 행 데이터가 RAM에 압축되지 않은 형식으로 저장됨

**Note 1:** 이 bit는 BOR(브라운아웃 리셋) 시에도 리셋됩니다.
**2:** 모든 디바이스에서 RAM 버퍼 행 작업을 사용할 수 있는 것은 아닙니다. RAM 버퍼 행 작업을 사용할 수 없는 경우 이들 bit가 구현되지 않은 것이며 '0'으로 읽습니다.
**3:** 이러한 옵션을 선택하면 WRERR bit가 세트되고 WR bit가 클리어됩니다.
**4:** 더블 워드 프로그램 작업에는 4개 명령 워드 단위의 인접한 두 명령 워드(각각 24비트)가 필요합니다.
**5:** 선택 디바이스에만 구현되어 있습니다. 자세한 내용은 디바이스별 데이터 시트를 참조하십시오.


© 2012-2017 Microchip Technology Inc.

**Register 4-1: NVMCON: 플래시 프로그래밍 컨트롤 레지스터 (Continued)**

bit 8 **URERR:** RAM 버퍼 행 프로그래밍 데이터 언더런 오류 플래그 bit(2)
1 = 데이터 언더런 오류로 인해 프로그래밍 작업이 종료됨
0 = 데이터 언더런 오류가 발견되지 않음

bit 7-4 **구현되지 않음:** '0'으로 읽음

bit 3-0 **NVMOP<3:0>:** NVM 작업 선택 bit(WR의 다음 세팅에 의해 시작됨)
1xxx =예약됨 (3)
011x =예약됨 (3)
0101 =예약됨 (3)
0100 =비활성 파티션 지우기 작업 ( 표준 파티션 모드의 예약된 옵션 )
0011 =페이지 지우기 작업
0010 =행 프로그램 작업
0001 =더블 워드 프로그램 작업 (4)
0000 =예약됨 (3)

**Note 1:** 이 bit는 BOR(브라운아웃 리셋) 시에도 리셋됩니다.
**2:** 모든 디바이스에서 RAM 버퍼 행 작업을 사용할 수 있는 것은 아닙니다. RAM 버퍼 행 작업을 사용할 수 없는 경우 이들 bit가 구현되지 않은 것이며 '0'으로 읽습니다.
**3:** 이러한 옵션을 선택하면 WRERR bit가 세트되고 WR bit가 클리어됩니다.
**4:** 더블 워드 프로그램 작업에는 4개 명령 워드 단위의 인접한 두 명령 워드(각각 24비트)가 필요합니다.
**5:** 선택 디바이스에만 구현되어 있습니다. 자세한 내용은 디바이스별 데이터 시트를 참조하십시오.

## 4.2 테이블 작업 명령

테이블 명령은 PIC24 및 dsPIC33 디바이스의 프로그램 메모리 공간과 데이터 메모리 공간 간에 데이터를 전송하는 한 가지 방법을 제공합니다. 이 섹션에는 플래시 프로그램 메모리 프로그래밍 중 사용되는 테이블 명령이 요약되어 있습니다. 다음과 같은 4가지 기본 테이블 명령이 있습니다.

- TBLRDL: 테이블 읽기 하위
- TBLRDH: 테이블 읽기 상위
- TBLWTL: 테이블 쓰기 하위
- TBLWTH: 테이블 쓰기 상위

TBLRDL 및 TBLWTL 명령은 프로그램 메모리 공간의 bit<15:0>을 읽고 쓰는 데 사용됩니다. TBLRDL 및 TBLWTL은 워드 또는 바이트 모드에서 프로그램 메모리에 액세스할 수 있습니다.

TBLRDH 및 TBLWTH 명령은 프로그램 메모리 공간의 bit<23:16>을 읽고 쓰는 데 사용됩니다. TBLRDH 및 TBLWTH는 워드 또는 바이트 모드에서 프로그램 메모리에 액세스할 수 있습니다. 프로그램 메모리는 단지 24비트 너비이므로 TBLRDH 및 TBLWTH 명령은 존재하지 않는 프로그램 메모리의 상위 바이트 주소를 지정할 수 있습니다. 이 바이트를 '팬텀 바이트'라고 합니다. 팬텀 바이트를 읽으면 00h가 반환되며, 팬텀 바이트에 써도 아무 효과가 없습니다.

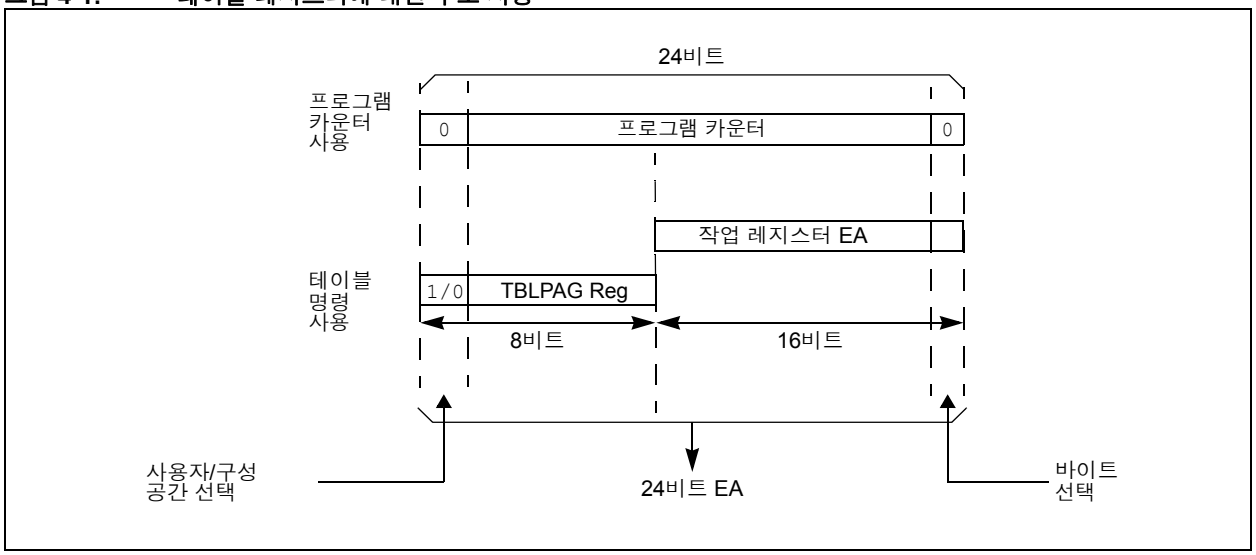
### 4.2.1 테이블 작업의 주소 생성

24비트 프로그램 메모리는 같은 주소 범위를 공유하는 2개의 병렬 16비트 공간으로 간주됩니다. 따라서 TBLRDL 및 TBLWTL 명령은 '하위'프로그램 메모리 공간(PM<15:0>)에 액세스합니다. TBLRDH 및 TBLWTH 명령은 '상위' 프로그램 메모리 공간(PM<31:16>)에 액세스합니다. PM<31:24>에 대한 읽기나 쓰기는 가상(구현되지 않은) 바이트에 액세스합니다. 바이트 모드에서 테이블 명령을 사용하면 테이블 주소의 LSb가 바이트 선택 bit로 사용됩니다. LSb는 상위 또는 하위 프로그램 메모리 공간의 어느 바이트에 액세스하는지를 결정합니다.

그림 4-1에서는 테이블 명령을 사용하여 프로그램 메모리의 주소를 지정하는 방법을 보여줍니다. 24비트 프로그램 메모리 주소는 테이블 명령(24비트 프로그램 카운터가 참조용으로 표시됨)에 지정된 W 레지스터의 TBLPAG<7:0> bit와 EA(유효 주소)로 구성됩니다. EA의 상위 23비트가 프로그램 메모리 위치 선택에 사용됩니다. 바이트 모드 테이블 명령의 경우 W 레지스터 EA의 LSb는 주소를 지정할 16비트 프로그램 메모리 워드의 바이트를 선택하는 데 사용됩니다. '1'은 bit<15:8>을 선택하고, '0'은 bit<7:0>을 선택합니다. 워드 모드의 테이블 명령에 대해서는 W 레지스터 EA의 LSb가 무시됩니다.

프로그램 메모리 주소와 함께 테이블 명령도 쓸 프로그램 메모리 데이터나 프로그램 메모리 읽기 대상의 소스인 W 레지스터나 메모리 위치에 대한 W 포인터를 지정합니다. 바이트 모드에서 테이블 쓰기 작업의 경우 소스 작업 레지스터의 bit<15:8>이 무시됩니다.

**그림 4-1: 테이블 레지스터에 대한 주소 지정**



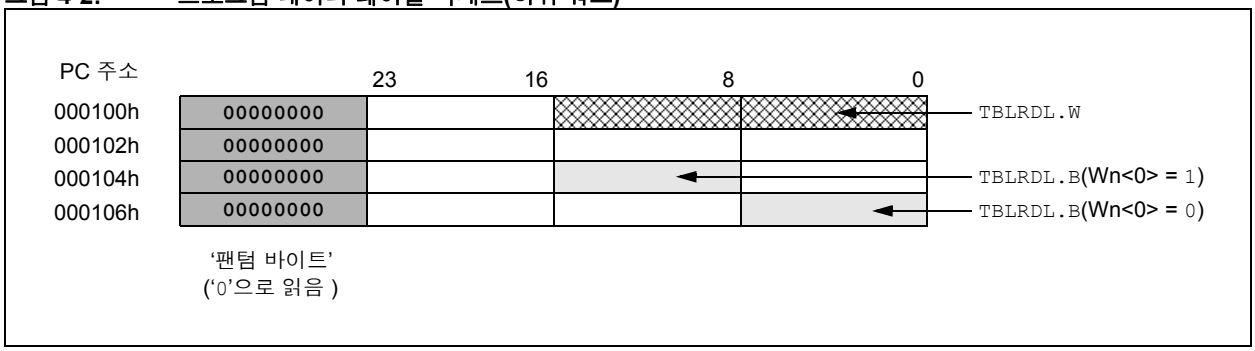


© 2012-2017 Microchip Technology Inc.

### 4.2.2 하위 워드 액세스

TBLRDL 및 TBLWTL 명령은 프로그램 메모리 데이터의 하위 16비트에 액세스하는 데 사용됩니다. 워드 차원의 테이블 액세스에 대해서는 W 레지스터 주소의 LSb가 무시됩니다. 바이트 차원의 액세스의 경우 W 레지스터 주소의 LSb가 읽을 바이트를 결정합니다. 그림 4-2에서는 TBLRDL 및 TBLWTL 명령에서 액세스하는 프로그램 메모리 데이터 영역을 보여줍니다.

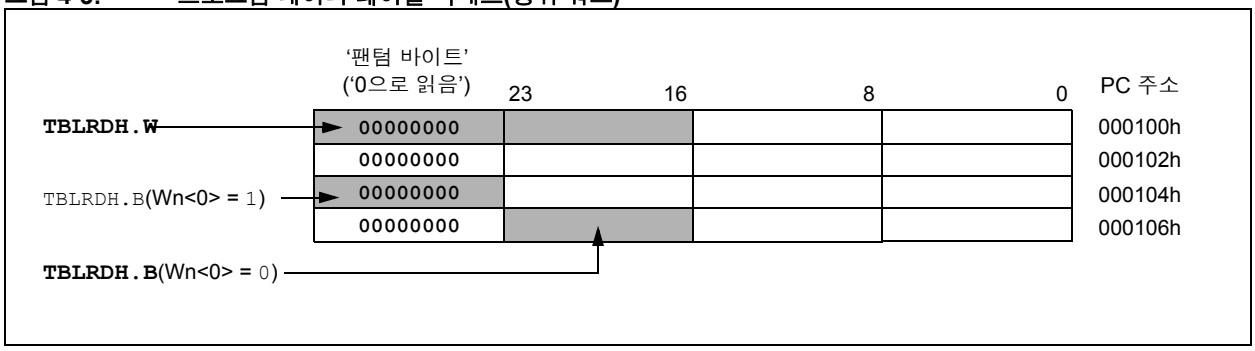
**그림 4-2: 프로그램 데이터 테이블 액세스(하위 워드)**




### 4.2.3 상위 워드 액세스

TBLRDH 및 TBLWTH 명령은 프로그램 메모리 데이터의 상위 8비트에 액세스하는 데 사용됩니다. 이들 명령은 또한 직교에 대해 워드 또는 바이트 액세스 모드를 지원하지만 그림 4-3과 같이 프로그램 메모리 데이터의 상위 바이트는 항상 '0'을 반환합니다.

**그림 4-3: 프로그램 데이터 테이블 액세스(상위 워드)**




### 4.2.4 프로그램 메모리의 데이터 저장소

대부분의 애플리케이션에 대해 상위 바이트(PM<23:16>)가 데이터에 사용되지 않아, 프로그램 메모리가 데이터 저장소에 대해 16비트 너비로 나타난다고 간주됩니다. 디바이스에서 저장된 데이터를 실수로 실행하지 않도록 프로그램 데이터의 상위 바이트를 NOP(00h 또는 FFh)나 유효하지 않은 opcode(3Fh) 값으로 프로그래밍하는 것이 좋습니다. TBLRDH 및 TBLWTH 명령은 주로 어레이 프로그램/확인 및 압축된 데이터 저장소를 필요로 하는 애플리케이션용으로 제공됩니다.

### 4.2.5 프로그램 메모리 BIT 동작

플래시 프로그램 메모리의 bit는 '1'에서 '0'으로만 프로그래밍할 수 있으며, 이후에 '1'로 지울 수 있습니다. 프로그래밍 시퀀스를 사용하여 bit를 세트하는 것은 아무 효과도 없습니다.

### 4.2.6 테이블 읽기 명령 사용

테이블 읽기에는 두 단계가 필요합니다. 먼저 TBLPAG 레지스터와 W 레지스터 중 하나를 사용하여 어드레스 포인터를 세트합니다. 그런 다음 주소 위치의 프로그램 메모리 콘텐츠를 읽을 수 있습니다.

Example 4-1 및 Example 4-2의 코드 예시에서는 워드 모드에서 테이블 명령을 사용하여 프로그램 메모리의 워드를 읽는 방법을 보여줍니다.

**Example 4-1: 워드 읽기 모드(Assembly에서)**

```
; Set up the address pointer to program space
MOV     #tblpage(PROG_ADDR), W0       ; get table page value
MOV     W0,TBLPAG                     ; load TBLPAG register
MOV     #tbloffset(PROG_ADDR), W0     ; load address LS word
; Perform the table writes to load the latch
TBLRDL  [W0], W2
TBLRDH  [W0], W3
```

**Example 4-2: 워드 읽기 모드(C에서)**

```
int addrOffset;
int varWord1;
int varWord2;

TBLPAG = ((PROG_ADDR & 0x7F0000)>>16);
addrOffset = (PROG_ADDR & 0x00FFFE);

varWord1 = __builtin_tblrdl(addrOffset);
varWord2 = __builtin_tblrdh(addrOffset);
```

**Note:** dsPIC33 및 PIC24 디바이스용으로 Microchip 어셈블러에서 `tblpage()` 및 `tbloffset()` 지시문을 제공합니다. 이러한 지시문은 프로그램 메모리 주소 값에서 테이블 명령에 대한 적절한 TBLPAG 및 W 레지스터 값을 선택합니다. 자세한 내용은 *"MPLAB® Assembler, Linker and Utilities for PIC24 MCUs and dsPIC® DSCs User's Guide"* (DS51317)를 참조하십시오.


© 2012-2017 Microchip Technology Inc.

### 4.2.7 테이블 쓰기 홀딩 래치

테이블 쓰기 명령은 플래시 프로그램 어레이에 직접 쓰지 않습니다. 대신 명령에서 프로그래밍할 데이터가 먼저 홀딩 래치로 로드되게 합니다. 이러한 래치는 대개 FA0000h에서 시작하는 구성 메모리 공간에서 메모리 매핑되며 테이블 쓰기 명령으로만 액세스할 수 있습니다. 홀딩 래치가 모두 로드되면 특별 시퀀스의 명령을 실행하여 실제 메모리 프로그램 작업이 시작됩니다.

서로 다른 디바이스가 특정 프로그램 어레이 설계(예: 행 프로그래밍 크기 및 행 프로그래밍 알고리즘)에 따라 다양한 수의 홀딩 래치를 구현합니다. 자세한 내용은 디바이스별 데이터 시트 및/또는 프로그래밍 사양을 참조하십시오.

#### 4.2.7.1 두 워드 쓰기 수행

TBLWTH 및 TBLWTL 명령 쌍을 사용하여 한 번에 두 워드에 대해 워드 쓰기가 수행됩니다. Example 4-3 또는 Example 4-4(C와 동일)의 코드 시퀀스로 워드 쓰기 모드를 사용하여 플래시에 프로그래밍할 두 프로그램 메모리 래치 위치를 쓸 수 있습니다.

**Example 4-3: 두 워드 쓰기 예(Assembly에서)**

```
; Set up the address pointer to 1st write latch
MOV     0xFA, W0                                ; get table page value
MOV     W0,TBLPAG                               ; load TBLPAG register
MOV     0x0, W0                                 ; load address LS word
; Load write data into W registers
MOV     #PROG_LOW_WORD_1, W2
MOV     #PROG_HI_BYTE_1, W3
MOV     #PROG_LOW_WORD_2, W4
MOV     #PROG_HI_BYTE_2, W5
; Perform the table writes to load the latch
TBLWTL  W2, [W0]
TBLWTH  W3, [W0++]
TBLWTL  W4, [W0]
TBLWTH  W5, [W0++]
```

**Example 4-4: 두 워드 쓰기 예(C에서)**

```
int varWord1L = 0xXXXX;
int varWord1H = 0x00XX;
int varWord2L = 0xXXXX;
int varWord2H = 0x00XX;
int addrOffset;
int TargetWriteAddressL;            // bits<15:0>
int TargetWriteAddressH;            // bits<22:16>

NVMCON = 0x4001;                    // Set WREN and word program mode
TBLPAG = 0xFA;                      // write latch upper address
addrOffset = (PROG_ADDR & 0x00FFFE);  // ensure address is properly aligned
NVMADRL = TargetWriteAddressL;      // set target write address
NVMADRH = TargetWriteAddressH;

__builtin_tblwtl(0,varWord1L);      // load write latches
__builtin_tblwth(0,varWord1H);
__builtin_tblwtl(0x2,varWord2L);
__builtin_tblwth(0x2,varWord2H);
__builtin_disi(5);                  // Disable interrupts for NVM unlock
sequence
```

## 4.3 RTSP(런타임 셀프 프로그래밍)

RTSP는 사용자 코드에서 플래시 프로그램 메모리 콘텐츠를 수정할 수 있게 합니다. RTSP는 TBLRD(테이블 읽기) 및 TBLWT(테이블 쓰기) 명령과 NVM 컨트롤 레지스터를 사용하여 수행됩니다. PIC24 및 dsPIC33 디바이스는 다음과 같은 플래시 프로그래밍 작업을 지원합니다.

- 플래시 페이지 지우기
- 행 프로그래밍(래치 기반 또는 RAM 기반)
- 워드 프로그래밍

RTSP를 통한 플래시 프로그래밍은 행이라는 메모리 블록이나 두 워드의 플래시 메모리로 수행됩니다. 프로그래밍 전에 메모리 위치를 지워야 합니다. 지우기 작업은 여러 행으로 구성되는 페이지라는 메모리 블록에서 수행됩니다. 행 크기는 디바이스마다 다릅니다. 자세한 내용은 디바이스 데이터 시트를 참조하십시오. 일반적으로 dsPIC33 및 PIC24 디바이스의 경우 페이지는 8행으로 정의됩니다. 이 문서에서는 행당 64개의 명령(페이지당 512개의 명령)이 있는 예를 사용합니다.

### 4.3.1 쓰기 홀딩 래치를 사용하여 행 프로그래밍

**Section 4.2.7 "테이블 쓰기 홀딩 래치"**에 나와 있듯이 래치 기반 행 프로그래밍을 구현하는 디바이스에는 프로그래밍 데이터를 포함하는 홀딩 래치가 있습니다. 실제 프로그래밍 작업 전에 TBLWT 명령을 통해 순서대로 쓰기 데이터를 래치에 로드해야 합니다. 행 쓰기를 수행하기 전에 명령 워드를 전체 행으로 래치에 로드해야 합니다.

RTSP 프로그래밍의 기본 시퀀스는 테이블 포인터를 세트한 다음 일련의 TBLWT 명령을 수행하여 버퍼를 로드하는 것입니다. 프로그래밍은 NVMCON 레지스터에 컨트롤 bit를 세트하여 수행됩니다. 예를 들어, 64개의 명령 행이 있는 디바이스에서 프로그래밍 주기는 쓰기 래치를 로드하기 위한 64개의 TBLWTL 명령 및 64개의 TBLWTH 명령과 NVMCON을 잠금 해제하고 WR bit를 세팅하는 프로그래밍 시퀀스로 구성됩니다. Example 4-5에 프로세스의 예가 나와 있습니다.

**Example 4-5: 쓰기 래치가 있는 행 프로그래밍(C에서)**

```
int varWordL[64];
int varWordH[64];
int targetWriteAddressL;              // bits<15:0>
int targetWriteAddressH;              // bits<22:16>
int i;

NVMCON = 0x4002;                      // Set WREN and row program mode
TBLPAG = 0xFA;
NVMADRL = targetWriteAddressH;        // set target write address
NVMADRH = targetWriteAddressL;

for(i=0; i<=63; i++)                  // load write latches with data
{                                     // to be written
__builtin_tblwtl(i, varWordL[i]);
__builtin_tblwth((i * 2), varWordH[i]);
}
__builtin_disi(5);                    // Disable interrupts for NVM unlock
sequence
```


© 2012-2017 Microchip Technology Inc.

### 4.3.2 RAM 버퍼를 사용한 행 프로그래밍

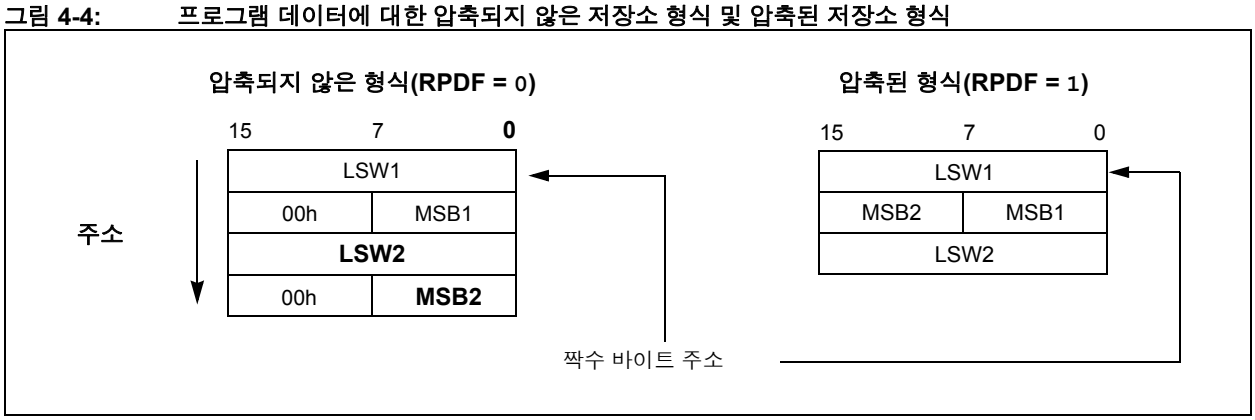
홀딩 래치를 통해 TBLWT 명령으로 데이터를 전송하지 않고 데이터 RAM의 버퍼 공간에서 직접 수행할 dsPIC33 및 PIC24 디바이스 허용 행 프로그래밍을 선택합니다. RAM 버퍼의 위치는 쓸 프로그램 데이터의 첫 워드를 포함하는 데이터 RAM 주소와 함께 로드되는 NVMSRCADR 레지스터로 결정됩니다.

프로그램 작업을 수행하기 전에 프로그래밍할 데이터 행과 함께 RAM의 버퍼 공간을 로드해야 합니다. 압축된 형식이나 압축되지 않은 형식으로 RAM을 로드할 수 있습니다. 압축된 저장소는 하나의 데이터 워드를 사용하여 인접한 두 프로그램 데이터 워드의 MSB(최상위 바이트)를 저장합니다. 압축되지 않은 형식은 각 프로그램 데이터 워드에 대해 두 개의 데이터 워드를 사용하며, 다른 모든 워드의 상위 바이트는 00h입니다. 압축된 형식은 압축되지 않은 형식과 비교하여 데이터 RAM 공간의 약 3/4을 사용합니다. 반대로 압축되지 않은 형식은 24비트 프로그램 데이터 워드의 구조와 비슷하며, 상위 팬텀 바이트로 완료됩니다. 데이터 형식은 RPDF bit(NVMCON<9>)로 선택됩니다. 이들 두 형식이 그림 4-4에 나와 있습니다.

RAM 버퍼가 로드되면 플래시 어드레스 포인터, NVMADRL 및 NVMADRH는 쓸 플래시 행의 24비트 시작 주소와 함께 로드됩니다. 프로그래밍 쓰기 래치와 마찬가지로 프로세스는 NVM 잠금 해제 시퀀스를 쓰고 WR bit를 세트하여 시작됩니다. 프로세스가 시작되면 디바이스는 자동으로 올바른 래치를 로드하고, 모든 바이트가 프로그래밍될 때까지 NVM 주소 레지스터를 증가시킵니다. Example 4-6에 프로세스의 예가 나와 있습니다. 데이터 언더런 오류 조건이 발생하는 값으로 NVMSRCADR이 세트되면 URERR bit(NVMCON<8>)는 조건을 나타내도록 선택됩니다.

또한 RAM 버퍼 행 프로그래밍을 구현하는 디바이스는 1 ~ 2개의 쓰기 래치를 구현합니다. 이들은 TBLWT 명령을 사용하여 로드되며 워드 프로그래밍 작업을 수행하는 데 사용됩니다.

**그림 4-4: 프로그램 데이터에 대한 압축되지 않은 저장소 형식 및 압축된 저장소 형식**




**Example 4-6: 데이터 RAM 버퍼에서 프로그램 메모리 쓰기(C에서)**

```
int data[64];                       // Data to be programmed in RAM
int targetWriteAddressL;            // bits<15:0>
int targetWriteAddressH;            // bits<22:16>

NVMCON = 0x4002;                    // Row programming
NVMCONbits.RPDF = 0;                // Select compressed format
NVMSRCADRL = (int)&data[0];         // Start address of data in RAM
NVMADRL = targetWriteAddressL;
NVMADRH = targetWriteAddressH;

__builtin_disi(5);                  // Disable interrupts for NVM unlock sequence
__builtin_write_NVM();
```

## 4.4 일반 플래시 프로그래밍 알고리즘

플래시 프로그래밍 작업은 다음과 같은 NVM(비휘발성 메모리) 컨트롤 레지스터를 사용하여 컨트롤됩니다.

- NVMCON
- NVMKEY
- NVMADRL/H
- NVMSRCADRL/H(일부 디바이스)

RTSP 모드에서 내부 플래시를 프로그래밍하거나 지우기 위해서는 전체 프로그래밍 시퀀스가 필요합니다. WR bit(NVMCON<15>)를 세트하면 작업이 시작되고, 작업이 완료되면 WR bit가 자동으로 클리어됩니다.

활성 파티션(특히 표준 파티션 모드에서)에서 플래시 프로그래밍 작업을 수행할 때 작업이 완료될 때까지 CPU가 정지합니다. 비활성 파티션을 프로그래밍할 때 CPU가 정지하지 않고 계속 작동할 수 있습니다. 다음 섹션에 CPU 정지 및 정지 안 함을 보이는 프로그래밍 알고리즘에 대한 개요가 있습니다.

### 4.4.1 프로그램 메모리 지우기(활성 파티션)

1. NVMOPx bit(NVMCOM<3:0>)를 '0011'로 세트하여 페이지 지우기에 대해 구성하고 WREN bit(NVMCOM<14>)를 세트합니다.
2. 지울 블록의 시작 주소를 NVMADRL/H 레지스터에 씁니다.
3. 인터럽트를 비활성화합니다.
4. NVMKEY에 55h를 씁니다.
5. NVMKEY에 AAh를 씁니다.
6. WR bit(NVMCOM<15>)를 세트하여 지우기 주기를 시작합니다.
7. 2개의 NOP 명령을 실행합니다.

지우기가 완료되면 WR bit가 자동으로 클리어됩니다.

### 4.4.2 행 프로그래밍(활성 파티션, 표준 파티션 모드)

사용자는 한 번에 한 행씩 프로그램 플래시 메모리를 프로그래밍할 수 있습니다. 이를 위해서는 원하는 행이 들어 있는 페이지를 지워야 합니다.

다음은 활성 파티션으로 행 프로그래밍을 수행하는 일반 프로세스입니다.

1. 8행의 프로그램 메모리(512개의 명령)를 읽고 데이터 RAM에 저장합니다.
2. 원하는 새 데이터로 RAM에서 프로그램 데이터를 업데이트합니다.
3. 블록 지우기:
   a) NVMOPx bit(NVMCOM<3:0>)를 '0011'로 세트하여 페이지 지우기에 대해 구성하고 WREN bit(NVMCOM<14>)를 세트합니다.
   b) 지울 블록의 시작 주소를 NVMADRL/H 레지스터에 씁니다.
   c) NVMKEY에 55h를 씁니다.
   d) NVMKEY에 AAh를 씁니다.
   e) WR bit(NVMCOM<15>)를 세트합니다. 지우기 주기가 시작되고 해당 주기 동안 CPU가 정지합니다. 지우기가 완료되면 WR bit가 자동으로 클리어됩니다.
4. 데이터 RAM에서 프로그램 메모리 버퍼로 처음 64개의 명령을 쓰거나(**Section 4.2.7 "테이블 쓰기 홀딩 래치"** 참조) RAM에 데이터의 시작 주소가 저장된 NVMSRCADR 레지스터를 씁니다.
5. 플래시 메모리에 프로그램 블록을 씁니다.
   a) NVMOPx bit를 '0010'으로 세트하여 행 프로그래밍에 대해 구성하고 WREN bit를 세트합니다.
   b) NVMKEY에 55h를 씁니다.
   c) NVMKEY에 AAh를 씁니다.
   d) WR bit를 세트합니다. 프로그래밍 주기가 시작되고 해당 주기 동안 CPU가 정지합니다. 플래시 메모리에 쓰기가 완료되면 WR bit가 자동으로 클리어됩니다.
6. 512개의 명령이 플래시 메모리에 모두 다시 써질 때까지 NVMADRL/H의 주소를 늘려서 데이터 RAM에 있는 블록의 사용 가능한 다음 64개 명령으로 4-5단계를 반복합니다.


© 2012-2017 Microchip Technology Inc.

**Note:** 쓰기 또는 지우기 주기 동안 모든 디바이스가 CPU 정지를 보이는 것은 아닙니다. 정지를 피하려면 지우거나 쓰고 있는 행에 적용하여 읽기나 쓰기를 방지하는 것이 좋습니다.

잘못된 작업을 방지하기 위해서는 지우기 또는 프로그래밍 작업 전 NVMKEY에 대한 쓰기 시작 시퀀스가 필요합니다. 프로그래밍 명령이 실행된 후 사용자는 프로그래밍이 완료될 때까지 프로그래밍 시간 동안 기다려야 합니다. 프로그래밍 시퀀스 시작 다음의 두 명령은 NOP여야 합니다.

**Note 1:** 행, 블록 및 홀딩 래치 수는 디바이스마다 다를 수 있습니다. 실제 수와 플래시 메모리 프로그래밍의 전체 기준 코드는 디바이스별 데이터 시트를 참조하십시오.

**2:** 단일 홀딩 래치가 있는 디바이스의 경우 워드 프로그래밍으로 플래시 프로그램 메모리를 써야 합니다.

### 4.4.3 비활성 파티션에서 페이지 프로그래밍 (이중 파티션 모드)

이중 파티션 모드에서 프로그래밍할 경우 특별히 고려할 사항이 있습니다. 비활성 파티션을 프로그래밍하는 동안에는 CPU가 계속 실행될 수 있으므로 CPU 정지가 발생하지 않습니다.

이중 파티션 모드 중 하나에서 데이터 페이지를 지우고 다시 프로그래밍하는 알고리즘은 다음과 같습니다.

1. 블록 지우기:
   a) NVMOPx bit(NVMCOM<3:0>)를 '0011'로 세트하여 페이지 지우기에 대해 구성합니다.
   b) WREN bit(NVMCOM<14>)를 세트합니다.
   c) 지울 블록의 시작 주소를 페이지 주소와 함께 NVMADR 레지스터에 씁니다.
   d) NVMKEY에 55h를 씁니다.
   e) NVMKEY에 AAh를 씁니다.
   f) WR bit(NVMCOM<15>)를 세트합니다. 지우기 주기가 시작되고 CPU는 계속 실행됩니다.
   g) 지우기가 완료되면 WR bit가 자동으로 클리어되고 NVM 쓰기 전체 인터럽트 플래그(NVMIF)가 발생합니다.
2. RAM 버퍼를 채워서 프로그래밍할 데이터를 준비합니다. 또는 메모리의 첫 행에 대한 데이터와 함께 TBLWT 명령(64개 명령)으로 쓰기 래치를 로드합니다.
3. 블록 프로그래밍:
   a) NVMOPx bit(NVMCON<3:0>)를 '0010'으로 세트하여 행 프로그래밍에 대해 구성합니다.
   b) WREN(NVMCON<14>) bit를 세트합니다.
   c) 쓸 블록의 시작 주소를 행 시작 주소와 함께 NVMADR 레지스터에 씁니다.
   d) NVMKEY에 55h를 씁니다.
   e) NVMKEY에 AAh를 씁니다.
   f) WR bit(NVMCOM<15>)를 세트합니다. 쓰기 주기가 시작되고 CPU는 계속 실행됩니다.
   g) 지우기가 완료되면 WR bit가 자동으로 클리어되고 NVM 쓰기 전체 인터럽트 플래그(NVMIF)가 발생합니다.
4. 2-3단계를 반복하여 지운 페이지의 남은 각 데이터 행을 프로그래밍합니다.

### 4.4.4 전체 비활성 파티션 프로그래밍 (이중 파티션 모드)

비활성 파티션에서 코드를 전체적으로 업데이트하려면

1. 비활성 파티션 지우기:
   a) NVMOPx bit(NVMCOM<3:0>)를 '0100'으로 세트하여 비활성 파티션 지우기에 대해 구성합니다.
   b) WREN bit(NVMCOM<14>)를 세트합니다.
   c) NVMKEY에 55h를 씁니다.
   d) NVMKEY에 AAh를 씁니다.
   e) WR bit(NVMCOM<15>)를 세트합니다. 지우기 주기가 시작되고 해당 주기 동안 CPU는 계속 실행됩니다. 지우기가 완료되면 WR bit가 자동으로 클리어되고 NVM 쓰기 전체 인터럽트 플래그(NVMIF)가 발생합니다.
2. **Section 4.4.3 "비활성 파티션에서 페이지 프로그래밍 (이중 파티션 모드)"**에 설명된 페이지 쓰기를 사용하여 비활성 파티션의 각 페이지를 씁니다.
3. 쓴 데이터를 확인합니다. 쓸 데이터에 대해 CRC를 수행하고 전체 파티션에서 CRC 값을 확인하여 데이터를 제대로 썼는지 점검합니다.

### 4.4.5 부팅 로더를 사용하여 활성 파티션 업데이트

1. **Section 4.4.4 "전체 비활성 파티션 프로그래밍 (이중 파티션 모드)"**의 설명에 따라 전체 비활성 파티션을 지우고 프로그래밍합니다.
2. 활성 파티션의 FBTSEQ 구성 레지스터를 읽습니다.
3. 값을 1씩 줄이고 비활성 파티션의 FBTSEQ에 씁니다.
4. 강제 파티션 스와핑:
   a) CPU 정지가 별로 중요하지 않을 경우 디바이스 리셋을 수행합니다. 비활성 파티션의 부팅 시퀀스 번호가 더 낮으므로 리셋 후 활성 파티션이 됩니다.
   b) CPU 정지가 허용되지 않는 경우 BOOTSWP 명령을 실행합니다.


© 2012-2017 Microchip Technology Inc.

## 5.0 프로그램 공간 가시성 및 확장 데이터 공간(PSV 및 EDS)

모든 dsPIC33 및 PIC24 디바이스에 대해 테이블 명령(**Section 4.2 "데이블 작업 명령"** 참조)을 사용하여 프로그램 메모리 공간 내의 데이터에 액세스할 수 있습니다. 이는 한 번에 1바이트 또는 워드씩 데이터를 읽거나 써야 할 때 유용합니다. 16,000워드 페이지의 프로그램 메모리 공간을 상위 32Kbyte의 데이터 주소 공간에 매핑하는 것도 가능합니다. 이를 통해 테이블 명령을 사용하지 않은 투명 액세스뿐만 아니라 기본 64Kbyte 주소 지정 제한 이상의 효과적인 데이터 공간 확장도 가능합니다.

dsPIC33 및 PIC24 디바이스 모두 데이터 공간에 구현된 프로그램 메모리 공간의 페이지를 매핑할 수 있습니다. 이 기능을 PSV(프로그램 공간 가시성)라고 합니다.

일부 디바이스는 특정 주변 장치를 가상 프로그램 메모리 페이지의 특정 범위에 메모리 매핑하여 PSV를 확장합니다. 이 기능은 데이터 처리량이 많아야 하는 고급 그래픽 컨트롤러와 같은 주변 장치에 특히 유용합니다. 이러한 PSV 확장을 EDS(확장 데이터 공간)라고 합니다.

PSV와 EDS는 데이터 메모리의 기능으로 구현되며, dsPIC33 및 PIC24 디바이스에 대해 다르게 구현됩니다. 자세한 내용은 *"dsPIC33/PIC24 제품군 참고자료"*, **"데이터 메모리"**(dsPIC33, DS70595) 및/또는 **"데이터 메모리"**(PIC24, DS39717)를 참조하십시오.

### 5.1 PSV 및 명령 정지

PSV를 사용한 명령 정지에 대한 자세한 내용은 *"dsPIC33/PIC24 제품군 참고자료"*, **"dsPIC33E 인핸스드 CPU"**(DS70005158)를 참조하십시오.

# 6.0 레지스터 맵

이중 파티션 플래시 프로그램 메모리와 관련된 SFR이 Table 6-1에 요약되어 있습니다.

**표 6-1: 플래시 프로그램 메모리와 관련된 특수 기능 레지스터(1)**

| 파일 이름 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | 모두 리셋(2) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TBLPAG | — | — | — | — | — | — | — | — | 테이블 페이지 어드레스 포인터 | | | | | | | | 0000 |
| NVMCON | WR | WREN | WRERR | NVMPIDL | SFTSWP | P2ACTIV | RPDF | URERR | — | — | — | — | NVMOP3 | NVMOP2 | NVMOP1 | NVMOP0 | 0000 |
| NVMKEY | — | — | — | — | — | — | — | — | NVMKEY<7:0> | | | | | | | | 0000 |
| NVMSRCADRL | 데이터 RAM 프로그래밍 버퍼 시작 주소 | | | | | | | | | | | | | | | | 0000 |
| NVMSRCADRH | 데이터 RAM 프로그래밍 버퍼 주소(EDS 작업만) | | | | | | | | | | | | | | | | 0000 |
| NVMADRL | 플래시 프로그램 메모리 대상 주소, 하위 바이트(ADDR<15:0>) | | | | | | | | | | | | | | | | 0000 |
| NVMADRH | — | — | — | — | — | — | — | — | 플래시 프로그램 메모리 대상 주소, 상위 바이트(ADDR<23:16>) | | | | | | | | 0000 |

**Legend:** — = 구현되지 않음, '0'으로 읽음. 리셋 값은 16진법으로 표시됩니다.

**Note 1:** 특정 메모리 맵에 대한 자세한 내용은 디바이스 데이터 시트를 참조하십시오.

**2:** 표시된 리셋 값은 POR에만 해당됩니다. 다른 리셋 상태의 값은 리셋 시 메모리 쓰기 또는 지우기 작업의 상태에 따라 달라집니다.



© 2012-2017 Microchip Technology Inc.

## 7.0 관련 애플리케이션 정보

이번 섹션에서는 본 설명서의 이 섹션과 관련된 애플리케이션 정보에 대해 기술합니다. 이 장치 정보는 PIC24 또는 dsPIC33 제품군 전용으로 기술된 것은 아니지만 개념은 연계되어 있으므로 변경 및 가능한 제한 사항을 적용하여 사용할 수 있습니다.

이중 파티션 플래시 프로그램 메모리와 관련된 현재 애플리케이션 정보는 다음과 같습니다.

| 제목 | 애플리케이션 정보 # |
|---|---|

현재 관련 애플리케이션 정보 없음.

**Note:** PIC24 및 dsPIC33 디바이스 제품군에 대한 추가 애플리케이션 정보와 코드 예시는 Microchip 웹 사이트(www.microchip.com)를 참조하십시오.

## 8.0 개정 내역

### 개정 A(2014년 3월)

본 문서의 원래 버전.

### 개정 B(2015년 2월)

제목과 모든 구문에서 “이중 부팅 플래시 프로그램 메모리”가 “이중 파티션 플래시 프로그램 메모리” 또는 “이중 파티션 플래시”로 변경되었습니다.


© 2012-2017 Microchip Technology Inc.

Microchip 디바이스의 코드 보호 기능에 대한 자세한 내용은 다음과 같습니다:

• Microchip 제품은 특정 Microchip 데이터 시트에 포함된 사양을 충족합니다.

• Microchip 은 자사 제품군을 의도한 방식에 따라 정상 조건에서 사용할 경우 현재 시장에 출시된 동종 제품 중 가장 안전한 제품군으로 봅니다.

• 코드 보호 기능을 훼손하는 데 부정 및 심지어 불법적인 수단까지 이용되고 있습니다. Microchip 이 알고 있는 바에 따르면, 이러한 모든 수단을 이용하기 위해서는 Microchip 데이터 시트에 포함된 작동 사양 외의 방법으로 Microchip 제품을 사용해야 합니다 . 대부분의 경우 이러한 행위를 하는 자는 지적 자산 도용과 연관되어 있습니다.

• Microchip 은 코드 안전성에 대해 우려하는 고객과 협력하고자 합니다.

• Microchip 이나 다른 반도체 제조업체도 코드 안전성을 보장할 수 없습니다. 코드 보호는 Microchip 이 제품의 " 침해 불가능성 " 을 보증한다는 의미가 아닙니다.

코드 보호는 지속적으로 발전하고 있습니다. Microchip 은 당사 제품의 코드 보호 기능을 지속적으로 개선하기 위해 노력하고 있습니다.

Microchip 의 코드 보호 기능을 훼손하려는 시도는 디지털 밀레니엄 저작권법 (Digital Millennium Copyright Act) 위반에 해당할 수 있습니다.

이러한 행위에 의해 귀하의 소프트웨어 또는 기타 저작권을 소유한 작업물에 무단 액세스가 이루어졌다면 해당 법률을 근거로 소송을 제기할 수있습니다.

디바이스 애플리케이션 및 유사 내용과 관련하여 본 출판물에 포함된 정보는 귀하의 편의를 위해 제공된 것이며 업데이트로 대체될 수 있습니다. 귀하의 애플리케이션이 해당 사양을 충족하는지 확인하는 것은 귀하의 책임입니다. Microchip은 정보와 관련하여 명시적, 묵시적, 서면, 구두, 법적 또는 다른 보증이나 진술을 하지 않으며, 여기에는 조건, 품질, 성능, 상업성, 특정 목적에의 적합성을 포함하되 이에 제한되지 않습니다 . Microchip은 이 정보 및 이 정보의 사용으로 발생한 모든 책임을 부인합니다. Microchip 디바이스를 생명 유지 및 / 또는 안전 애플리케이션 용도로 사용하는 위험은 전적으로 구매자에게 있으며, 구매자는 그러한 사용으로 인해 발생한 모든 손해, 청구, 소송 또는 비용으로부터 Microchip을 옹호, 보호하며 배상을 약속하는 데 동의합니다. Microchip 지적 재산권 하에 어떠한 라이센스도 묵시적 또는 다른 방식으로 제공되지 않습니다.

**QUALITY MANAGEMENT SYSTEM**
**CERTIFIED BY DNV**
**═ ISO/TS 16949 ═**

**상표**

Microchip 이름과 로고, Microchip 로고, AnyRate, AVR, AVR 로고 , AVR Freaks, BeaconThings, BitCloud, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KEELOQ, KEELOQ logo, Kleer, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST 로고, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 로고, Prochip Designer, QTouch, RightTouch, SAM-BA, SpyNIC, SST, SST 로고, SuperFlash, tinyAVR, UNI/O, XMEGA 는 미국 및 기타 국가에서 Microchip 의 등록 상표입니다.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, Quiet-Wire는 미국에서 Microchip Technology Incorporated의 등록 상표입니다.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, chipKIT, chipKIT 로고, CodeGuard, CryptoAuthentication, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PureSilicon, QMatrix, RightTouch logo, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA는 미국 및 기타 국가에서 Microchip 의 상표입니다.

SQTP는 미국에서 Microchip Technology Incorporated 의 서비스 마크입니다.

Silicon Storage Technology는 기타 국가에서 Microchip Technology Incorporated 의 등록 상표입니다.

GestIC는 기타 국가에서 Microchip Technology Inc.,의 자회사인 Microchip Technology Germany II GmbH & Co.KG 의 등록 상표입니다.

위에 언급된 다른 모든 상표는 각각 해당 회사의 자산입니다.

© 2017, Microchip Technology Incorporated, All Rights Reserved.

ISBN: 978-1-5224-2079-8

*Microchip 은 세계 본사 , 애리조나 주 챈들러 및 템피 , 오리건 주 그레셤의 설계 및 웨이퍼 가공 설비와 캘리포니아 주 및 인도의 설계 센터에 대해 ISO/TS-16949:2009 인증을 획득하였습니다. Microchip 의 고품질 시스템 프로세스 및 절차는 PIC® MCU 및 dsPIC® DSC, KEELOQ® 코드 호핑 디바이스, 직렬 EEPROM, 마이크로 주변장치, 비휘발성 메모리 및 아날로그 제품에 적용되고 있습니다. Microchip 의 개발 시스템의 설계 및 제조를 위한 품질 시스템도 ISO 9001:2000 인증을 획득하였습니다.*

# 각국의 영업소 및 서비스


**미주**

본사
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
기술지원 :
http://www.microchip.com/support
웹사이트 :
www.microchip.com
아틀란타
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455
오스틴 , TX
Tel: 512-257-3370
보스턴
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088
시카고
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075
클리블랜드
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643
달라스
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924
디트로이트
Novi, MI
Tel: 248-848-4000
휴스턴 , TX
Tel: 281-894-5983
인디애나폴리스
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
로스앤젤레스
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
뉴욕 , NY
Tel: 631-435-6000
새너제이 , CA
Tel: 408-735-9110
캐나다 - 토론토
Tel: 905-673-0699
Fax: 905-673-6509

**아시아 / 태평양**

아태평양 지사
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
홍콩
Tel: 852-2943-5100
Fax: 852-2401-3431
호주 - 시드니
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755
중국 - 베이징
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104
중국 - 청두
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889
중국 - 충칭
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500
중국 - 동관
Tel: 86-769-8702-9880
중국 - 항저우
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116
중국 - 홍콩 SAR
Tel: 852-2943-5100
Fax: 852-2401-3431
중국 - 난징
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470
중국 - 칭다오
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205
중국 - 상하이
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066
중국 - 센양
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393
중국 - 센젠
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760
중국 - 우한
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118
중국 - 시안
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

**아시아 / 태평양**

중국 - 샤먼
Tel: 86-592-2388138
Fax: 86-592-2388130
중국 - 주하이
Tel: 86-756-3210040
Fax: 86-756-3210049
인도 - 방갈로르
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123
인도 - 뉴델리
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632
인도 - 푸네
Tel: 91-20-3019-1500
일본 - 오사카
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310
일본 - 도쿄
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771
한국 - 대구
Tel: 82-53-744-4301
Fax: 82-53-744-4302
한국 - 서울
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934
말레이시아 - 쿠알라룸푸르
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859
말레이시아 - 페낭
Tel: 60-4-227-8870
Fax: 60-4-227-4068
필리핀 - 마닐라
Tel: 63-2-634-9065
Fax: 63-2-634-9069
싱가포르
Tel: 65-6334-8870
Fax: 65-6334-8850
대만 - 신주
Tel: 886-3-5778-366
Fax: 886-3-5770-955
대만 - 까오슝
Tel: 886-7-213-7828
대만 - 타이페이
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102
태국 - 방콕
Tel: 66-2-694-1351
Fax: 66-2-694-1350

**유럽**

오스트리아 - 벨스
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393
덴마크 - 코펜하겐
Tel: 45-4450-2828
Fax: 45-4485-2829
프랑스 - 파리
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79
독일 - 뒤셀도르프
Tel: 49-2129-3766400
독일 - 카를스루헤
Tel: 49-721-625370
독일 - 뮌헨
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44
이탈리아 - 밀라노
Tel: 39-0331-742611
Fax: 39-0331-466781
이탈리아 - 베니스
Tel: 39-049-7625286
네덜란드 - 드루넨
Tel: 31-416-690399
Fax: 31-416-690340
폴란드 - 바르샤바
Tel: 48-22-3325737
스페인 - 마드리드
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91
스웨덴 - 스톡홀름
Tel: 46-8-5090-4654
영국 - 워킹엄
Tel: 44-118-921-5800
Fax: 44-118-921-5820


01/27/15


© 2017 Microchip Technology Inc.