

데이터 중단점

*이 샘플은 Microsoft 게임 개발 키트와 호환 가능합니다(2020년 6월).*

# 설명

이 샘플은 다른 유형의 메모리 액세스를 탐지하는 데 유용한 하드웨어 데이터 중단점을 생성하는 방법을 보여줍니다. 프로세서 코어에서 완전히 처리되므로 실행 속도에 영향을 주지 않습니다. 사용의 주요 단점은 프로세서 코어에서 사용할 수 있는 슬롯이 4개뿐이라는 것입니다.

Windows에서는 스레드에 대한 로컬 중단점만 허용합니다. 즉, 중단점은 질문 내 해당 스레드가 실행되는 동안에만 활성화됩니다. 하드웨어 중단점은 스레드에 바인딩되므로 다양한 코어에서 실행되는 스레드를 따릅니다. 필요한 경우 각 스레드를 동일한 중단점으로 설정할 수 있습니다.

# 목차

## DataBreak.cpp/h

* 하드웨어 데이터 중단점을 설정하고 지우는 기능을 제공하는 독립 실행형 패키지입니다.
* SetHardwareBreakPointForThread
  + 주소와 질문 내 해당 슬롯이 지정된 중단점을 설정합니다.
  + 슬롯이 이미 사용 중인 경우 덮어씁니다.
* ClearHardwareBreakPointForThread
  + 지정된 슬롯의 중단점을 지웁니다.

# 구현 참고 사항

하드웨어 중단점을 설정하려면 디버그 레지스터의 내용을 조정해야 합니다. 주요 문제는 이러한 레지스터에 대한 액세스는 커널에서만 사용할 수 있다는 것입니다. 디버그 레지스터를 설정하는 트릭은 스레드 컨텍스트를 조정하는 것입니다. 이렇게 하면 컨텍스트 전환 중에 스케줄러가 디버그 레지스터에 콘텐츠를 씁니다.

스레드의 컨텍스트는 스레드가 일시 중단된 경우에만 변경할 수 있습니다. 이 때문에 대상 스레드가 현재 스레드인 경우 작업을 수행하려면 다른 스레드가 필요합니다. 샘플은 컨텍스트 전환을 수행하는 임시 작업자 스레드를 만듭니다.

발생한 예외는 구조적 예외 처리 **(\_\_try**, **\_\_except**) 또는 처리되지 않은 예외 필터를 통해 감지할 수 있습니다. 그러나 **MiniDumpWriteDump**와 함께 사용되는 경우 캡처 예외 레코드는 무시됩니다. 그러나 예외를 발생시킨 스레드에서 **MiniDumpWriteDump**가 호출되는 경우 잘못된 코드는 호출 스택에 계속 있습니다. 단, 더 위에 있습니다. 처리되지 않은 예외 필더가 잘못된 스레드 컨텍스트에서 호출됩니다.

## 중요 참고 사항:

이 시스템에서 발생한 예외는 OS에서 특별히 처리하는 단일 단계 예외입니다. 디버거가 연결되면 디버거를 먼저 감지합니다. 기본적으로 Visual Studio는 이를 무시하며 코드를 한 단계씩 실행하지 않으면 컨트롤이 타이틀 예외 처리기에 전달됩니다. 기본적으로 KD는 예외를 일으키는 코드 줄에서 중단됩니다. 하지만 디버거가 연결되지 않은 경우 **EnableKernelDebugging**은 동작을 정의합니다. 이 기능을 사용하도록 설정하면 타이틀이 중지됩니다. 콘솔에서 디버거가 연결되기를 기다리고 있습니다. **EnableKernelDebugging**을 사용하도록 설정하지 않으면 타이틀 예외 처리기가 호출됩니다.

DataBreak.cpp의 **DataBreakThread** 함수는 디버그 레지스터에 필요한 내용을 문서화합니다.

# 업데이트 기록

초기 릴리스 2020년 8월

# 개인정보처리방침

샘플을 컴파일하고 실행하는 경우 샘플 사용량을 추적할 수 있도록 샘플 실행 파일의 파일 이름이 Microsoft에 전송됩니다. 이 데이터 수집을 옵트아웃하려면 "샘플 사용량 원격 분석"으로 레이블이 지정된 Main.cpp에서 코드 블록을 제거할 수 있습니다.

Microsoft의 개인정보 정책에 대한 자세한 내용은 [Microsoft 개인정보처리방침](https://privacy.microsoft.com/en-us/privacystatement/)을 참조하세요.