TRACE32활용 SW Run time 측정

MDS테크놀로지
기술지원팀
최종 수정일 : 2017.03

소 속 MDS Technology㈜
주 소 경기도 성남시 분당구 삼평동 676
전화 031-627-3189
FAX 031-627-3100
E-mail trace32@mdstec.com

1. 소개

1.1. Software Run Time 측정

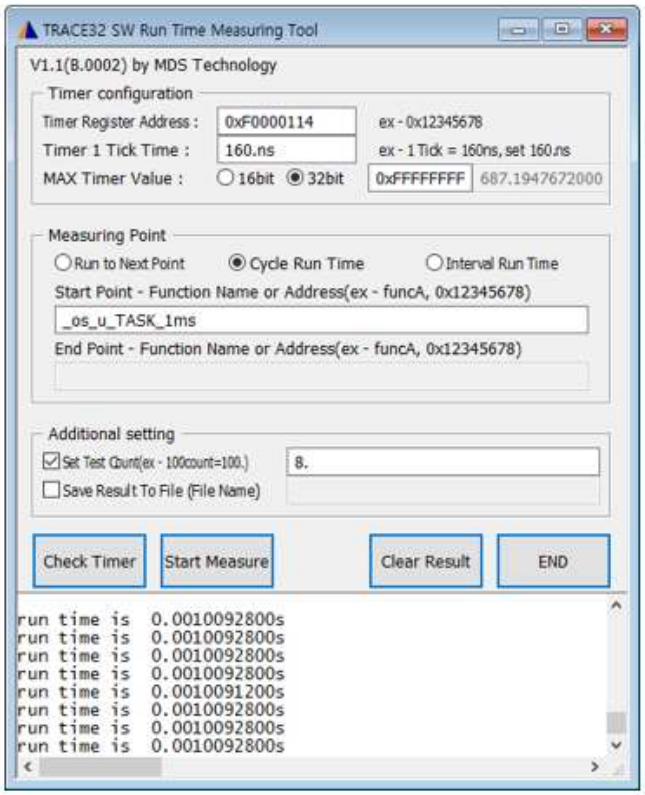
개발된 SW 모듈의 성능을 판단하는 중요한 요소 중 하나는 SW 의 실행 시간이다. 특히 안전과 관련된 기능(SW 모듈)들은 Worst Case 시에도 문제 없이 SW 기능이 동작되도록 설계 및 구현되어야 하기 때문에, 일반적으로 Run time 을 필수로 측정하고 있다

Run Time 을 측정하기 위해 가장 많이 사용되는 방법 중 하나가 측정하고자 하는 SW 의 Entry 와 Exit 부분에 특정 Trigger 를 발생시키는 코드를 삽입 한 후 이를 실행하여 측정하는 방법이다. 만약 HW GPIO 를 Trigger 신호로 이용한다면, 오실로스코프와 같은 장비로 타이밍을 측정할 수 있다. 혹은 CPU 내부 Timer 레지스터를 이용한다면 특정 변수에 Run Time 값을 저장하게 한 후 변수 값을 관찰하여 타이밍 정보를 측정할 수 있다. 하지만 이러한 방법들은 타이밍 측정을 위한 코드가 삽입되기 때문에 발생되는 단점들이 있다.
첫 번째로, 정확한 타이밍 측정이 어렵다. 장시간 측정하는 경우, 코드 삽입에 대한 타이밍 오차율이 거의 없지만 아주 짧은 코드에 대한 수행 시간을 측정하는 경우, 탐침 코드로 인해 발생되는 측정 오차가 클 수 있다.

두 번째로, 다양한 측정 Point 를 설정하기 어렵고 불편하다. 예를 들어, GPIO 를 사용하여 타이밍을 측정하는 경우, 사용할 수 있는 핀 개수가 제한적이기 때문에 다양한 측정 point 를 동시에 측정하기 어렵고 이를 변경하려면 매번 '소스 코드 수정->컴파일->타겟 업로드->재연' 과정을 거쳐야 한다.
세 번째로, 타이밍 오류가 발생되는 경우에 대한 디버깅이 어렵다. 타이밍 위반 사항을 발견했다고 하더라도 어떠한 경우(test case)에 문제가 발생되는지 디버깅 하는 것은 또 다른 문제다. 타이밍 측정과 디버깅에 필요한 정보를 추가로 저장하도록 코드를 수정한다면 이 또한 추가적인 CPU 로드가 사용되므로, 정확한 타이밍 정보를 측정하는 것이 어렵게 된다.

1.2. TRACE32 를 활용한 SW Runtime 측정 방식

TRACE32 는 기본적으로 SW 디버깅 목적으로 많이 사용되지만 SW 타이밍 측정도 TRACE32 기능들을 조합하여 구현이 가능하다.



» **Timer 정보 설정**

» **SW 주기, Run Time 측정**

» **측정 관련 추가 설정**

» **측정 결과**

〈그림 1〉 1ms TASK의 주기 8회 측정 결과

〈그림 1〉 은 1ms 마다 동작되는 TASK(function)의 실행 주기를 측정한 결과이다. 일관성 있고 정확한 측정 결과가 나오는 것을 볼 수 있다.

〈그림 2〉 TRACE32를 활용한 Run Time 측정 방식

TRACE32 SW Run Time 측정 기능의 동작 방식은 타이밍을 측정하고자 하는 곳에 trigger 를 발생 시키도록 한다는 점에서 기존에 사용하던 GPIO 신호 측정 방식이나, 탐침 코드를 이용한 측정

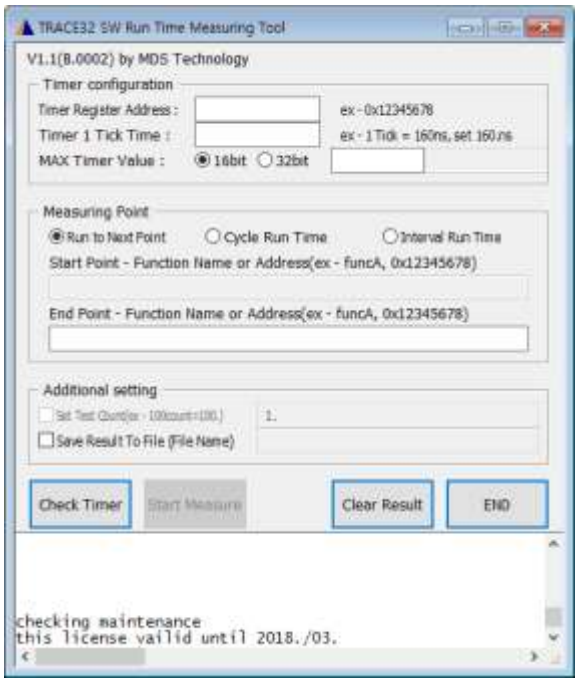
방식과 비슷하다. 다른 점은 <그림 2>와 같이 실제로 측정을 원하는 시점에 break point 를 활용하여 코어와 타이머 레지스터를 동시에 정지시킨 후 타이밍 정보를 측정하는 것이다. 따라서 TRACE32 를 이용하면 위에서 언급한 방식으로 SW Run Time 측정 시 발생되는 단점 및 불편한 사항들을 개선할 수 있다.

1. 정확성: 코어와 타이머 레지스터도 같이 정지 되기 때문에 정확한 Run time 을 측정 가능
2. 편리성: HW pin 연결, 코드 수정이 필요 없기 때문에(Break Point 활용) 원하는 SW Run time 정보를 바로 측정 가능
3. 디버깅: 만약 기준 Run Time 을 초과하는 경우가 발생되면, 디버거는 바로 측정을 멈추고 현재 상황(Test Case)을 디버깅 가능

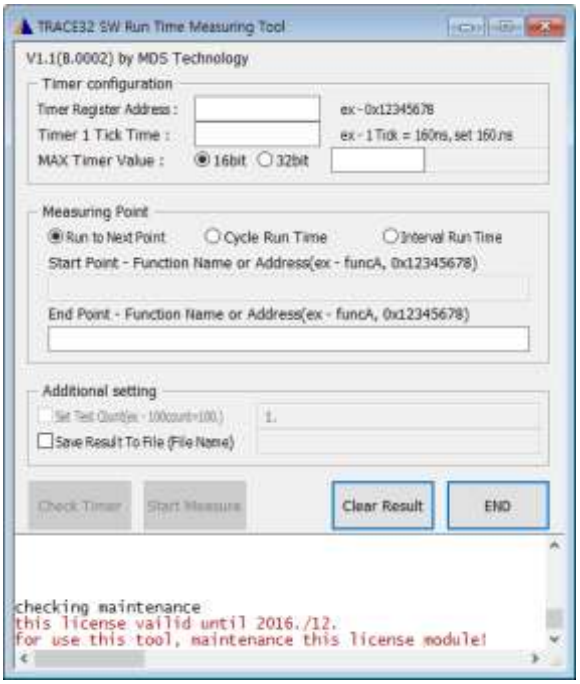
2. 기능 사용 하기

2.1. GUI 소개

실행 방법 : B::Do start_runtime_measure.cmm
실행 하면 연결된 라이선스 모듈의 유지보수 기간을 체크 한다. 유지보수 기간이 종료된 경우 'Run to NEXT Point'기능만 활용 가능하며 10회로 사용이 제한 됨



〈유지보수 기간 유효〉



〈유지보수 기간 종료〉

Timer Configuration - Timer Register Address : 타이밍 측정에 사용할 내부 타이머 레지스터 주소
Timer 1 Tick Time : Timer 1 tick 당 시간 정보
MAX Timer Value : 16bit or 32bit 선택, Max Timer 값 지정

Measuring Point - Run to Next Point : 현재 PC 값부터 End point 까지 실행 시간 측정
Cycle Run Time : Start Point 에 설정된 함수의 실행 주기 측정
Interval Run Time : Start to End Point 까지 실행 시간 측정

Start Point/End Point : Runtime 측정 할 함수 이름 또는 주소 값 입력

Additional Setting - Set Test Count : 지정 된 횟수만큼 반복적으로 측정
Save Result to file : 측정 결과를 지정된 파일로 저장

Check Timer 버튼 : 설정된 타이머가 디버그 모드로 설정 되었는지 확인

Start Measure 버튼 : 타이머 설정이 올바르게 되었다면 버튼 활성화 됨
Measuring Point 에 설정된 SW Runtime 측정

Clear Result 버튼 : 출력 창 메시지 clear

End 버튼 : GUI 종료

타이머 정보나, 타이머 설정을 자동으로 하고자 한다면 아래 파일에 필요한 정보를 입력한다.
set_debug_mode.cmm : 내부 타이머를 debug 모드로 설정 하는 TRACE32 명령어로 사용하는 타이머에 따라서 주소와 write 값이 달라질 수 있다.

set_debug_mode.cmm

```
1  B::
2
3  //Set debug mode for TC277T STM1 module
4  data.set 0xf00001e8--0xf00001eb 0x0 0x0 0x0 0x12
5
6  enddo
```

timer_info.txt : 타이머 레지스터의 주소나, 설정된 상태 정보를 입력

timer_info.txt

```
1  Timer_Regster_Address : 0xFFF3C004
2  Timer_1_Tick_Time : 15.6ns
3  Timer_Bit_Lenth(16_or_32) : 32
4  MAX_Timer_Value : 0xFFFFFFFF
```

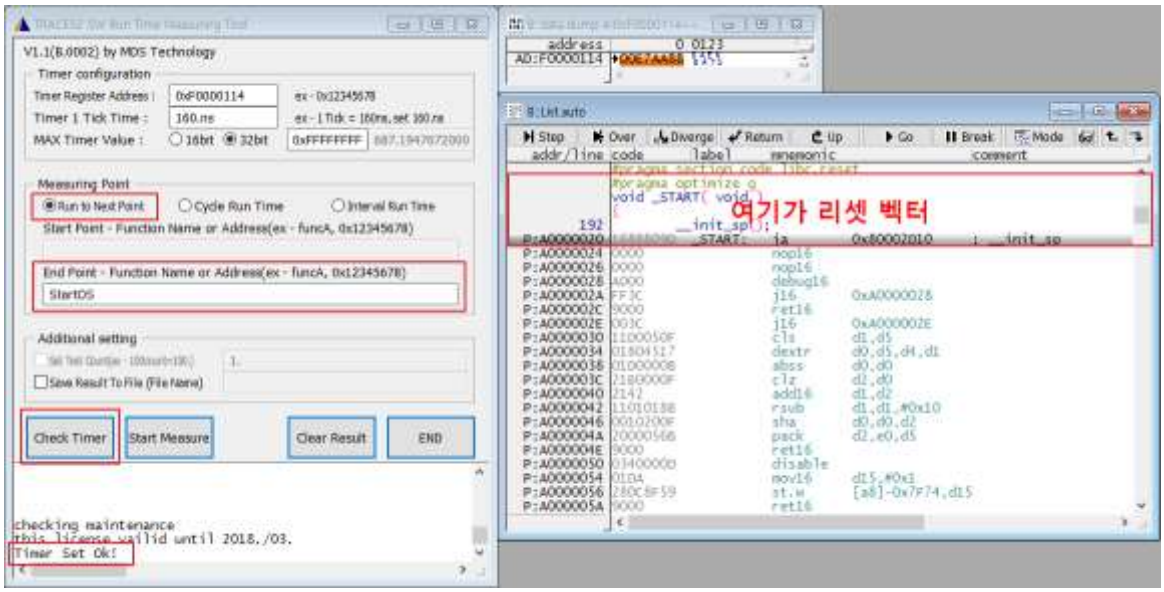
2.2. TRACE32 Run Time 측정 기능 사용 예제

2.2.1. 부팅 시간 측정

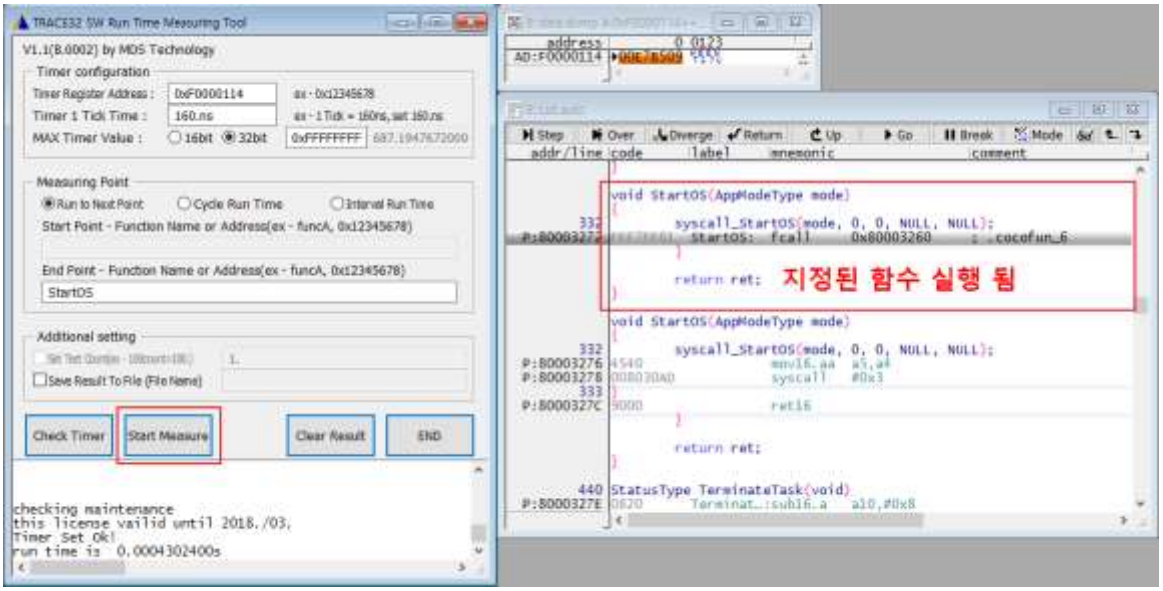
-현재 PC가 reset vector에 있고, StartOS함수가 호출 될 때까지의 run time을 측정해보는 예제

-타이밍 측정을 하기 전 타이머 레지스터 설정을 마치고 check timer를 실행 해 준다.

-Run to Next Point기능을 이용하여 측정



-Start measure를 실행하여 Run Time 측정

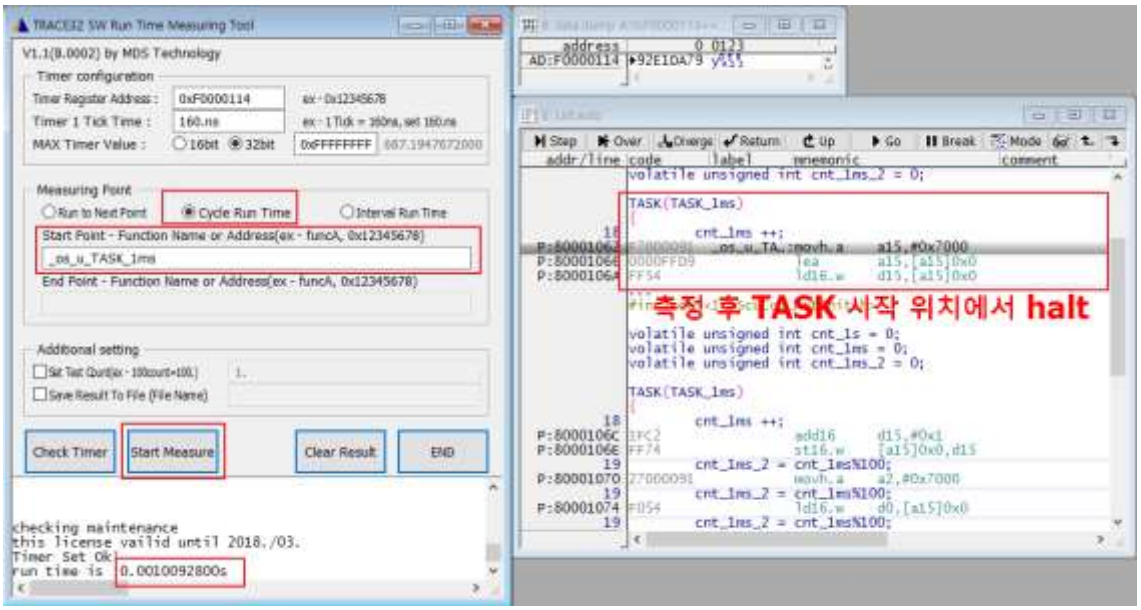


2.2.2 주기적으로 수행되는 TASK 의 실행 주기 측정

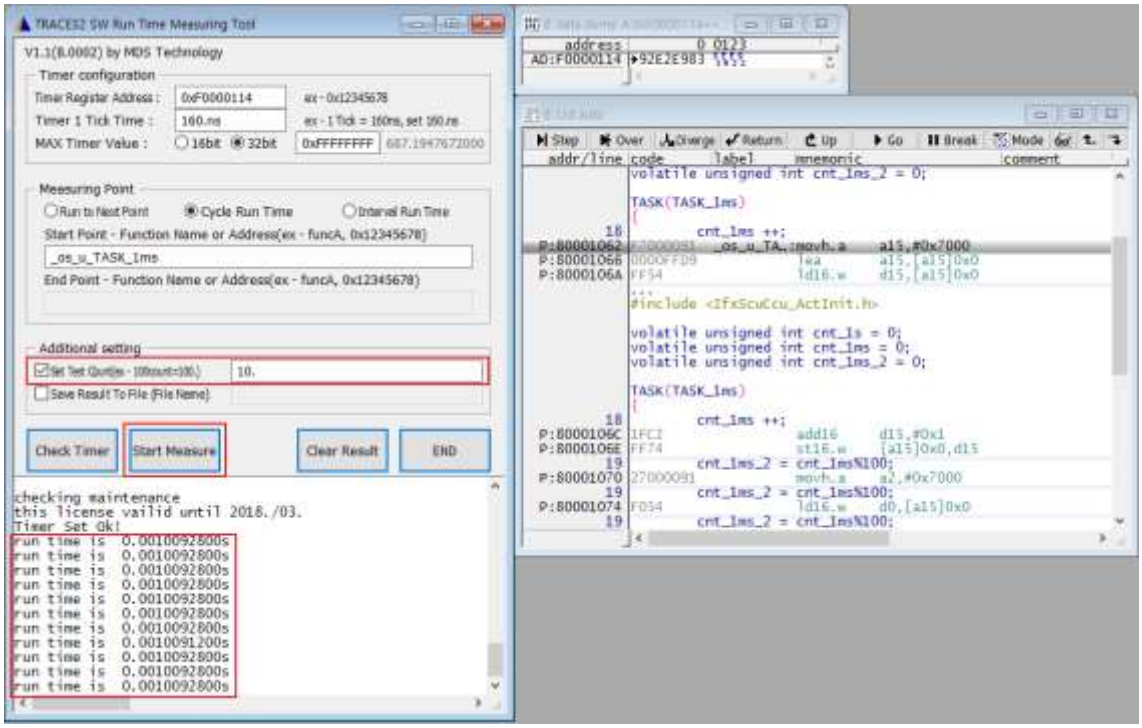
-1ms 마다 실행되는 TASK 의 주기를 측정 해보는 예제로 Cycle run Time 측정 기능 이용

-마찬가지로 측정을 하기 전에 Check Timer 를 통해 타이머 설정이 잘 되었는지 확인 필요

-타이밍 측정을 시작하는 PC 의 위치는 관계 없다



-반복해서 여러 번 측정 하고자 한다면 Additional Setting 의 'Set Test Count' 메뉴를 사용

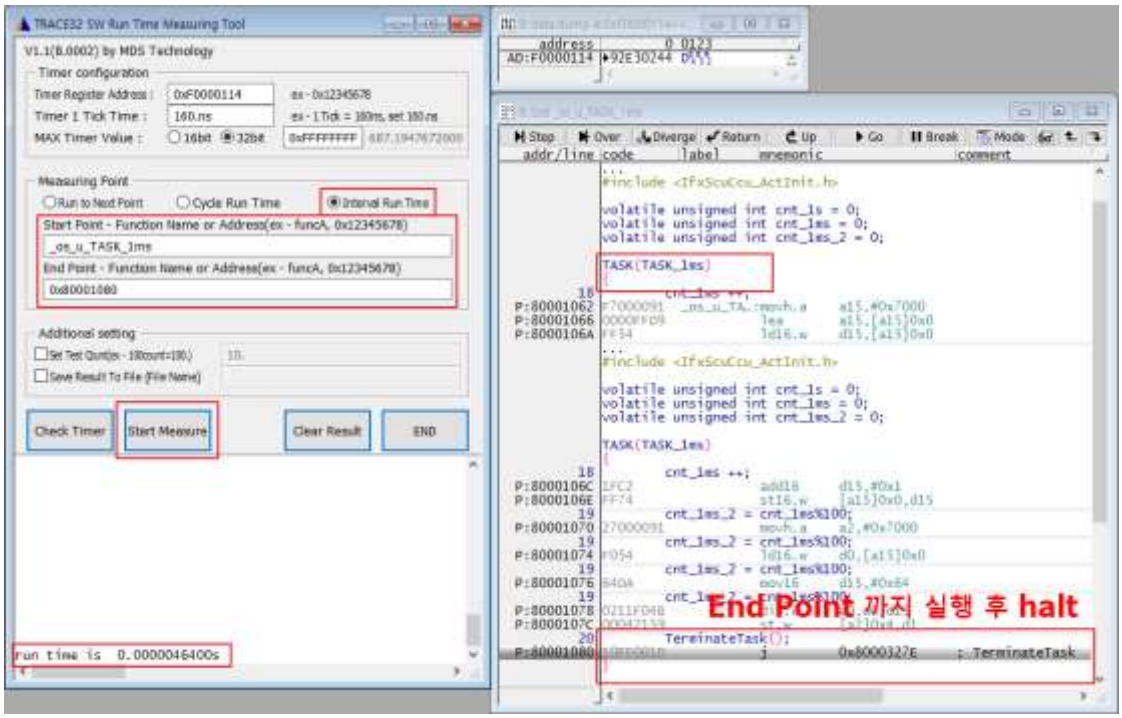


2.2.3. 특정 구간에 대한 Run Time 측정

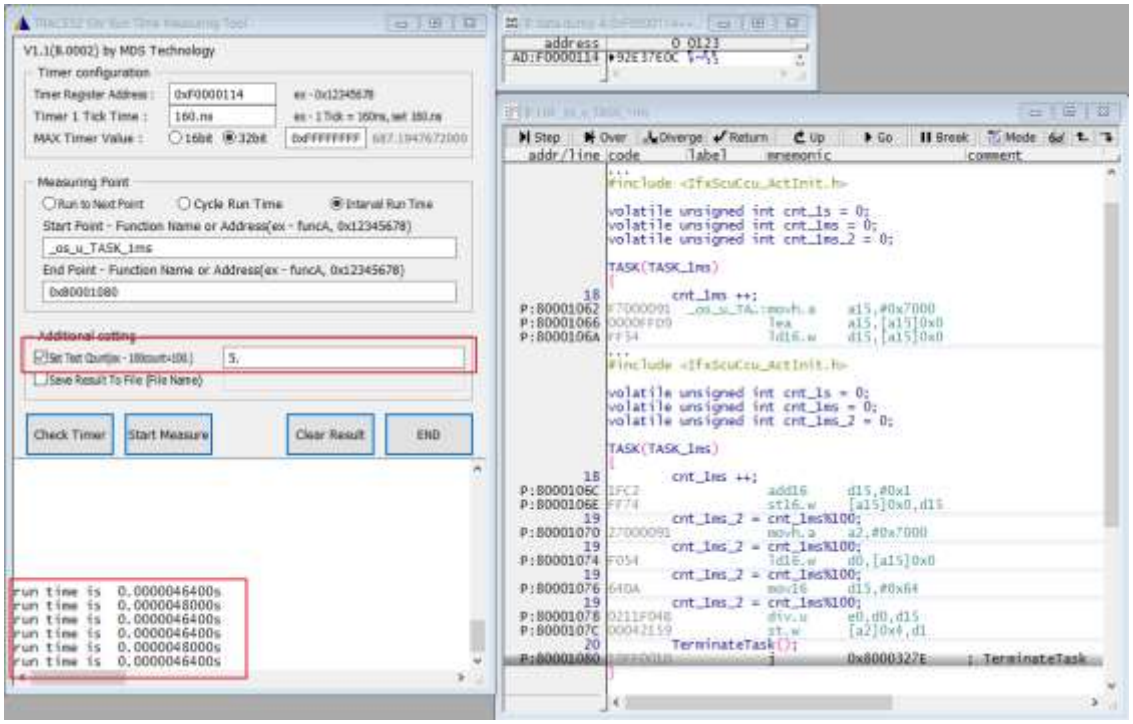
-TASK_1ms 의 시작 위치부터 종료 위치까지 Run Time 을 측정 해보는 예제로 Interval Run Time 측정 기능 이용

-Start Point(TASK 함수 시작)와 End Point(TerminateTask 함수 호출) 설정이 정확한지 확인

-측정을 시작하는 위치(PC)는 무방



-반복해서 여러 번 측정하고자 한다면 마찬가지로 'Set Test Count' 설정 이용

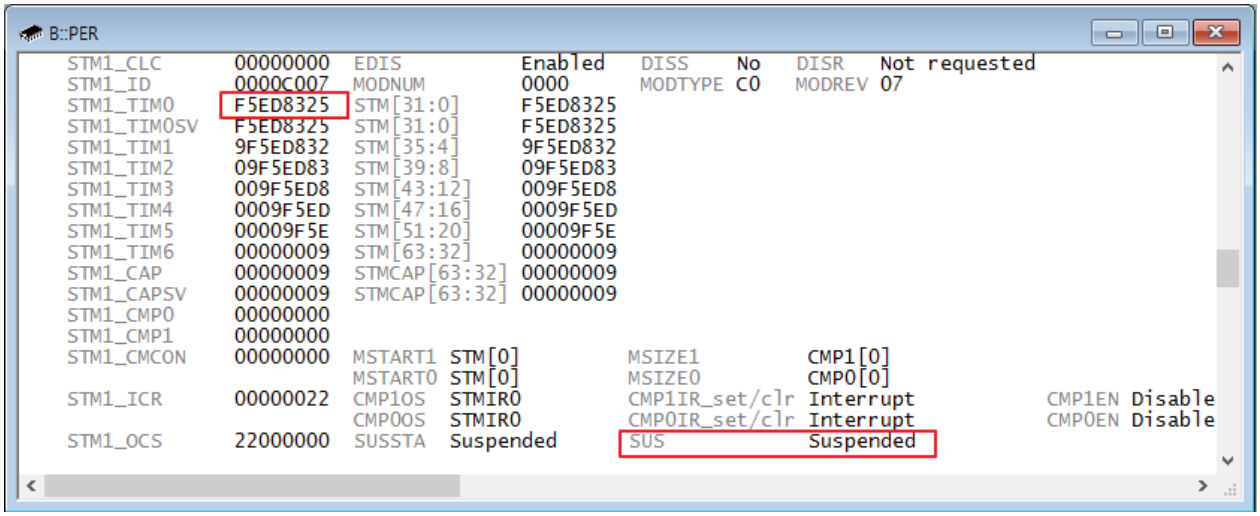


3. 유의 사항

3.1 타이머 레지스터 디버그 모드 설정

정확한 타이밍을 측정하기 위해서는 타겟 코어 제어(run/stop)시에 동기화 되어 Timer 레지스터도 제어(run/stop)가 되어야 한다. 이를 가능하게 하는 것이 타이머 레지스터를 디버그 모드로 설정하는 것이다. 디버그 모드로 Timer 를 설정하기 위해서는 TRACE32 를 활용하거나 startup 코드를 활용하는 방법이 있다. 만약 TRACE32 에서 타이머를 디버그 모드로 설정 했으나 중간에 내부 SW 가 실행되면서 다시 정상 모드로 타이머를 초기화 하면 안된다.

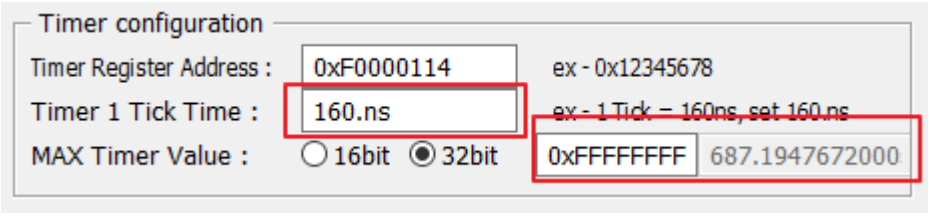
아래는 AURIX TC277T MCU 의 타이머 레지스터 및 디버그 모드 설정의 예이다. SUS field 에 Suspended 로 설정하면 코어의 동작에 따라 같이 동기화 되어 run/stop 된다.



부팅 시간을 측정하는 경우에는 reset vector 부터 startos 까지 실행되는 동안 초기화 되지 않는 Timer 를 사용해야 한다.

3.2 Timer 를 이용한 최대 측정 시간

만약 측정하고자 하는 코드가 타이머 레지스터의 1cycle 시간내에 실행되지 않는다면 overflow 가 발생 되고 정확한 시간을 측정 할 수 없다. 예를 들어 타이머 레지스터 1cycle 이 1ms 라면 이 타이머로는 1ms 이상으로 동작되는 SW run time 을 측정 하면 안 된다. 따라서 설정된 타이머의 최대 측정 가능 시간을 확인하고 그 이하의 시간을 측정 해야 한다. 아래 설정을 예로 들자면 약 687(160ns * 0xFFFFFFFF)초 이내로 실행되는 SW 를 측정 할 수 있다.



3.3. Run Time 중 Timer 설정 변경

타겟 SW 가 동작 하면서 타이밍 측정에 사용되는 타이머의 설정이나 값을 변경하면 정확한 타이밍 측정이 어렵다. 이런 경우에는 다른 타이머를 run time 측정에 활용하는 것을 권장한다.

4. TRACE32 Run Time 측정 기능 응용

TRACE32 의 장점은 강력한 스크립트 기능을 이용하여 반복되는 상황이나, 특정 시나리오 상황에서 원하는 동작을 자동으로 수행하도록 구현 가능하다는 점이다. 이 스크립트 자동화 기능과 Run time 측정 기능을 함께 사용하면 예를 들어 아래와 같은 기능도 구현할 수 있다.

Run Time 측정 기능 응용 예)

1) 디버깅: 타이밍 조건을 설정하여 조건에 fail 되는 경우 테스트를 중지하고 현재 타겟 상태를 확인하여 디버깅 할 수 있다.
2) 타이밍 측정 결과 저장: SW 타이밍 측정 후, 이를 TRACE32 file I/O 기능을 이용하여 측정 결과를 파일로 저장할 수 있다.
3) 타이밍 측정 자동화: 다양한 point 의 SW 타이밍 측정이 필요하다면 시나리오를 만들고 이를 TRACE32 스크립트로 구현하여 자동화 할 수 있다.
4) SW 유닛 테스트에 활용: TRACE32 기능 중 가장 많이 사용 되는 것 중 하나가 원하는 시점에 변수에 값을 read/write 하는 기능일 것이다. 이 기능과 SW Run Time 측정을 같이 사용한다면 특정 조건(변수 값)일 때 SW Run Time 측정하는 것이 가능하다.
5) Data Access Timing 측정: Program Break Point 를 이용하면 SW 타이밍을 측정 가능하고, 마찬가지로 Data Break Point 를 이용하면 Data 가 read/write 되는 타이밍을 측정할 수 있다.