Under CCES, the easiest way to find an EUNIT command is to start typing it, then type cntrl-space and let CCES find the assert for you.

E-Unit functions and constants

Header information

```
#define EMBEDDEDBEDUNIT_LITE
#include <EmbeddedUnit/EmbeddedUnit.h>
```

Basic EmbeddedUnit

```
TEST_CONTROL(TEST_GROUP_NAME);
TEST(TEST_NAME) ;
TEST_FILE_RUN_NOTIFICATION(TEST_GROUP_NAME);
```

```
CHECK(expression)  CHECK(a == b);
XF_CHECK(expression)
CHECK_EQUAL(expected, actual)
XF_CHECK_EQUAL(expected, actual)
CHECK_CLOSE(expected, actual, tolerance)
CHECK_ARRAY_EQUAL(expected, actual, count)
CHECK_ARRAY_CLOSE(expected, actual, count, tolerance)
CHECK_ARRAY2D_CLOSE(expected, actual, rows, columns, tolerance)
CHECK_THROW(expression, ExpectedExceptionType)
CHECK_ASSERT(expression)
REPORT(msg)
```

```
TIME_CONSTRAINT(ms), TIME_CONSTRAINT_US
TIME_CONSTRAINT_EXEMPT()
MEASURE_EXECUTION_TIME(time)
HARD_TIME_CONSTRAINT_TRY(ms), HARD_TIME_CONSTRAINT_CATCH()
HARD_TIME_CONSTRAINT_END()
```

```
MEMORY_CONSTRAINT(maxChange) MEMORY_CONSTRAINT_EXEMPT( )
```

```
void CodeCoverageStartLogging(int loopCompress = 2)
void CodeCoverageStopLogging(void)
```

TEST DRIVEN DEVELOPMENT SYNTAX EXAMPLES

// This function is developed bool WaitForAWhileASM(short int time_wanted);
// Assume this function is already written long int CalculateTwiceAccuracy(short int time1, short int time2);   This function returns the timing accuracy 100 * (2* time1 –time2) / time2

```
TEST_CONTROLT(TEST_GROUP_NAME);

TEST(Q2_TESTS, DEVELOPER_TEST) {
//  If the parameter time is less than 0 return false  (because it can't be done);
      CHECK( WaitForAWhileASM(-1) = = false );
      CHECK( WaitForAWhileASM(1000) = = true );
//    When you make the parameter time bigger, then the subroutine takes longer to return
//     Needs to be checked for both a long and a short time
//  2 * Time for WaitForAWhileASM(X) = time for  WaitForAWhileASM(2X) ;
// The accuracy of timing should be better than 2% or 2 part in 100.
              unsigned long int time1, time2;
      time1 = MEASURE_EXECUTION_TIME(WaitForAWhileASM(400));
      time2 = MEASURE_EXECUTION_TIME(WaitForAWhileASM(800));
      CHECK( CalculateTwiceAccuracy(time1,  time2) < 2);    // 1 is 1%, 2 is 2%
      time1 = MEASURE_EXECUTION_TIME(WaitForAWhileASM(1));
      time2 = MEASURE_EXECUTION_TIME(WaitForAWhileASM(2));
     CHECK( CalculateTwiceAccuracy(time1,  time2) < 2);    // 1 is 1%, 2 is 2%
}
```

```
TEST(Q4_TEST, DEVELOPER_TEST) {
    StopCoreTimer( );                                        // Stop the timer
    InitializeCoreTimer(0x200000, 0x200000, 1);              // Set some sensible values into the core timer
registers
    StartCoreTimer(3 );                          // Start the timer
    long int time1 = ReadCoreTimerAndResetASM(0x2000, 0x2000);
// Assert statement #1 – check that the core timer value has got smaller since started
    CHECK(time1 < 0x200000);
// Assert statement #2 – From the code we expect 2 writes and 1 read (in total) to occur
// when we run the ReadCoreTimerAndResetASM( ) function
   WatchDataClass<unsigned long int> coretimer_access( 2,
                              (unsigned long int *) pTCOUNT, (unsigned long int *) pTPERIOD);
// NOTE that the 2 in this line means we are specifying 2 memory locations for the
// WatchDataClass to keep track on.
// NOTE that the WatchDataClass and WATCH_MEMORY_RANGE keep track of ALL
//  memory read and write operations that occur in the memory locations between the
// addresses for TCOUNT and TPERIOD
   WATCH_MEMORY_RANGE(coretimer_access,
                         ReadCoreTimerAndResetASM (0x3000, 0x3000));
    CHECK(coretimer_access.getReadsWrites() = = 3);
// Assert statement #3 – get the final value of the core-timer registers
//  getFinalValue(0) would get the final value of the TCOUNT register
//  getFinalValue(1) would get the final value of the TPERIOD register
// It is the order in the WatchDataClass line that counts, not the parameter order in
// ReadCoreTimerAndResetASM( )
    CHECK(coretimer_access.getFinalValue(1) = = 0x3000);
}

TEST_FILE_RUN_NOTIDICATION(TEST_GROUP_NAME);
```