Arm のキーテクノロジ TrustZone セキュリティの研究

IoT注目マイコン Cortex-M23実験室

最終回 第3回

TrustZone セキュリティ・プログラムの実力を試す

中森 章

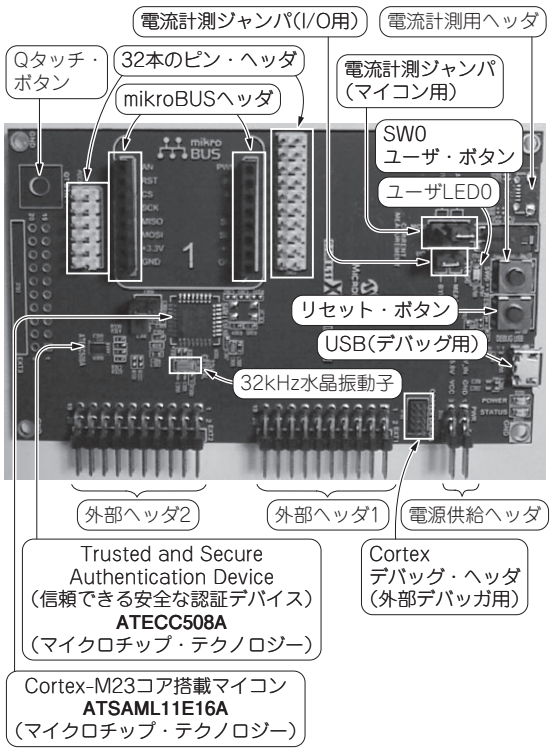

写真1　Arm のセキュリティ TrustZone 内蔵 IoT マイコン Cortex-M23 ボード
SAM L11 XplainedPro Evaluation Kit

やること

今回は，**写真1**の Arm Cortex-M23マイコン・ボードをターゲットにして，セキュアなGetStart-Sプログラムと，ノンセキュアなGetStart-NSプログラム（プロジェクト）を改造して，セキュア領域からノン・セキュア領域の関数を呼び出してセキュア領域に返ってくるTrustZoneプログラムを作ってみます．

ノン・セキュア領域に，引き数の値を2乗して返す関数である，

```
int func_square(int x)
{
    return x * x;
}
```

を置いて，セキュア領域から，この関数を呼び出すことにします．そのためにはセキュア領域に関数のアドレスをもらう関数，

```
void func_addr(int x)
```

を定義して，そのアドレスをセーブし，セーブしたアドレスの関数を呼び出します．

この場合の，GetStart-Sプロジェクトと GetStart-NSプロジェクトの`main.c`と GetStart-Sプロジェクトの`trustzone_veneer.c`はそれぞれ，**リスト1**と**リスト2**のようになります．

ノン・セキュア側では`func_square`という関数を，セキュア側では`func_square`という関数のポインタを宣言していますが，ビルドを行っても型の不整合のエラーは出ません．基本的に，セキュア領域とノン・セキュア領域は別世界だからです．

全体の動作としては，ノン・セキュア領域から`nsc_func_addr()`関数によって，ノン・セキュア領域にある`func_square()`関数のアドレスがセキュア領域に渡され，セキュア領域にある関数へのポインタ`func_square`に格納されます．セキュア領域では関数へのポインタ`func_square`で示される関数を呼び出して，結果を変数`y`に格納します．

実行してみる

それでは，ビルドしたソリューションを実行します．セキュア領域の`ns_func_addr()`関数の入り口にブレーク・ポイントを置いて，ステップ実行してみます．

`ns_func_addr()`関数の入り口では，r0の値が0x8237となっています．これは，ノン・セキュア領域にある関数`func_square()`の先頭アドレスが0x8236であることを示しています（**図1**，**図2**，**図3**）．

ここで，ステップ実行すると，0x8236番地の関数，すなわち`func_square()`を呼び出して戻ってきます．r0には`func_square()`関数の戻り値である2×