

DENSO

Crafting the Core

デンソークリエイト 1DAYインターンシップ マイクロマウス単体テスト手順書

オンライン

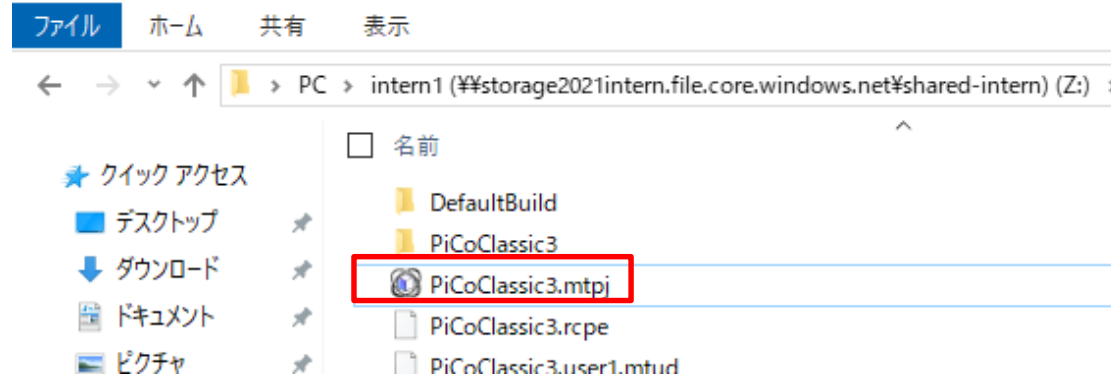
CONFIDENTIAL
関係者外秘

1. テスト用プロジェクトを開く
2. テスト用プログラムをビルドする
3. テスト用プログラムを動作させる
4. 単体テストを実施する

参考. 各種ウィンドウが消えてしまったときは

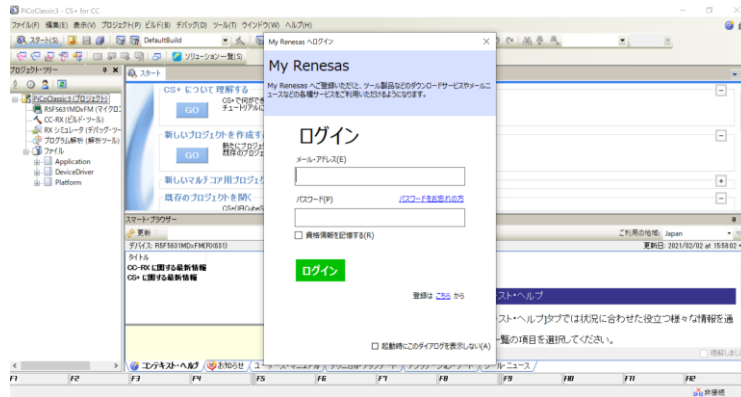
1. テスト用プロジェクトを開く

「Z:¥unit_test」フォルダ内の「PiCoClassic3.mtpj」をダブルクリックしてください。



アプリケーションソフト “CS+ for CC” が起動します。

起動時の画面はプログラム開発時とほぼ同じです。各種ダイアログが出たら、気にせず閉じてください。

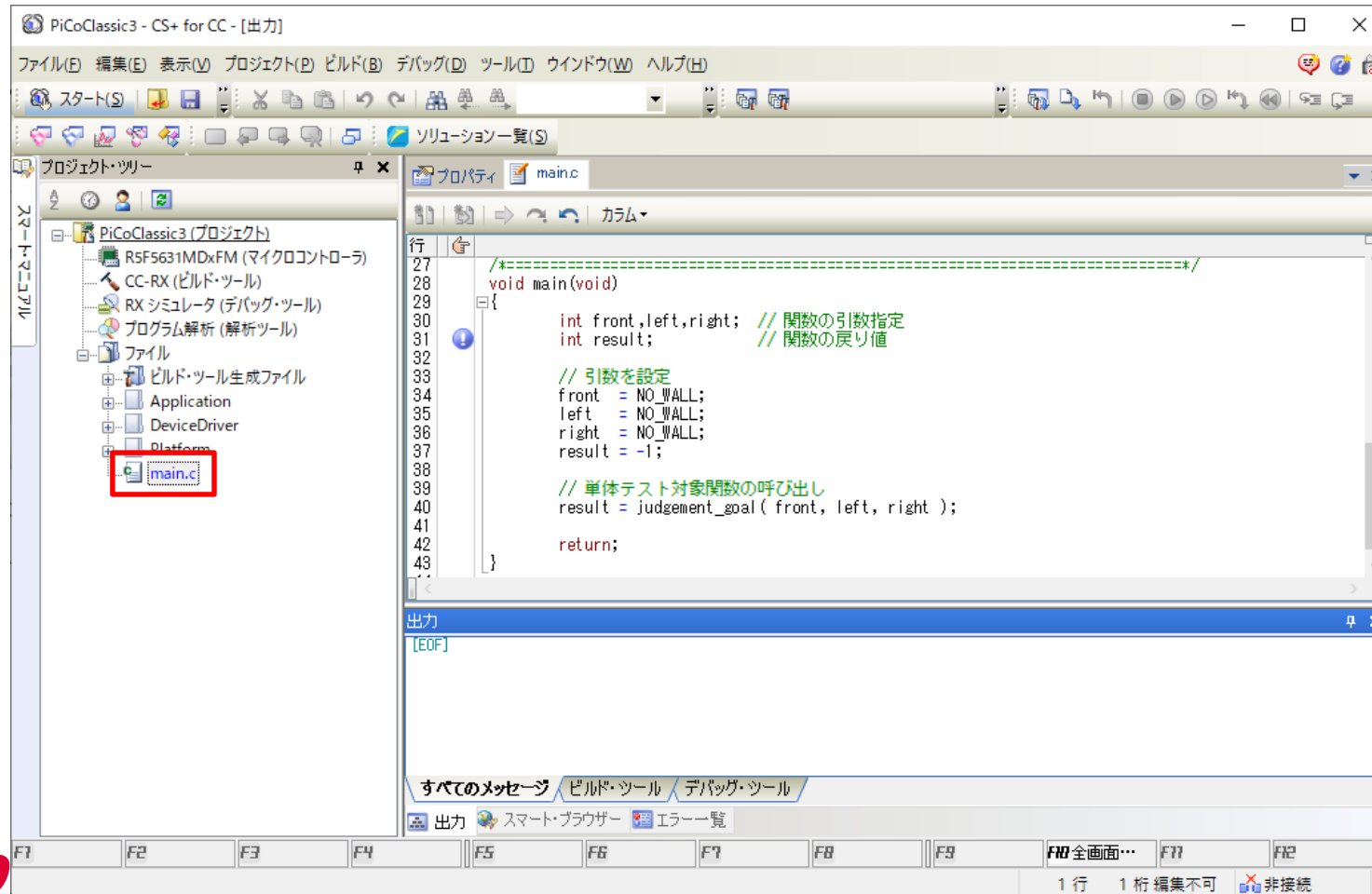


1. テスト用プロジェクトを開く

プロジェクトを開くと、以下のような画面になります。

左側にあるプロジェクト・ツリーからmain.cをダブルクリックして開いてください。

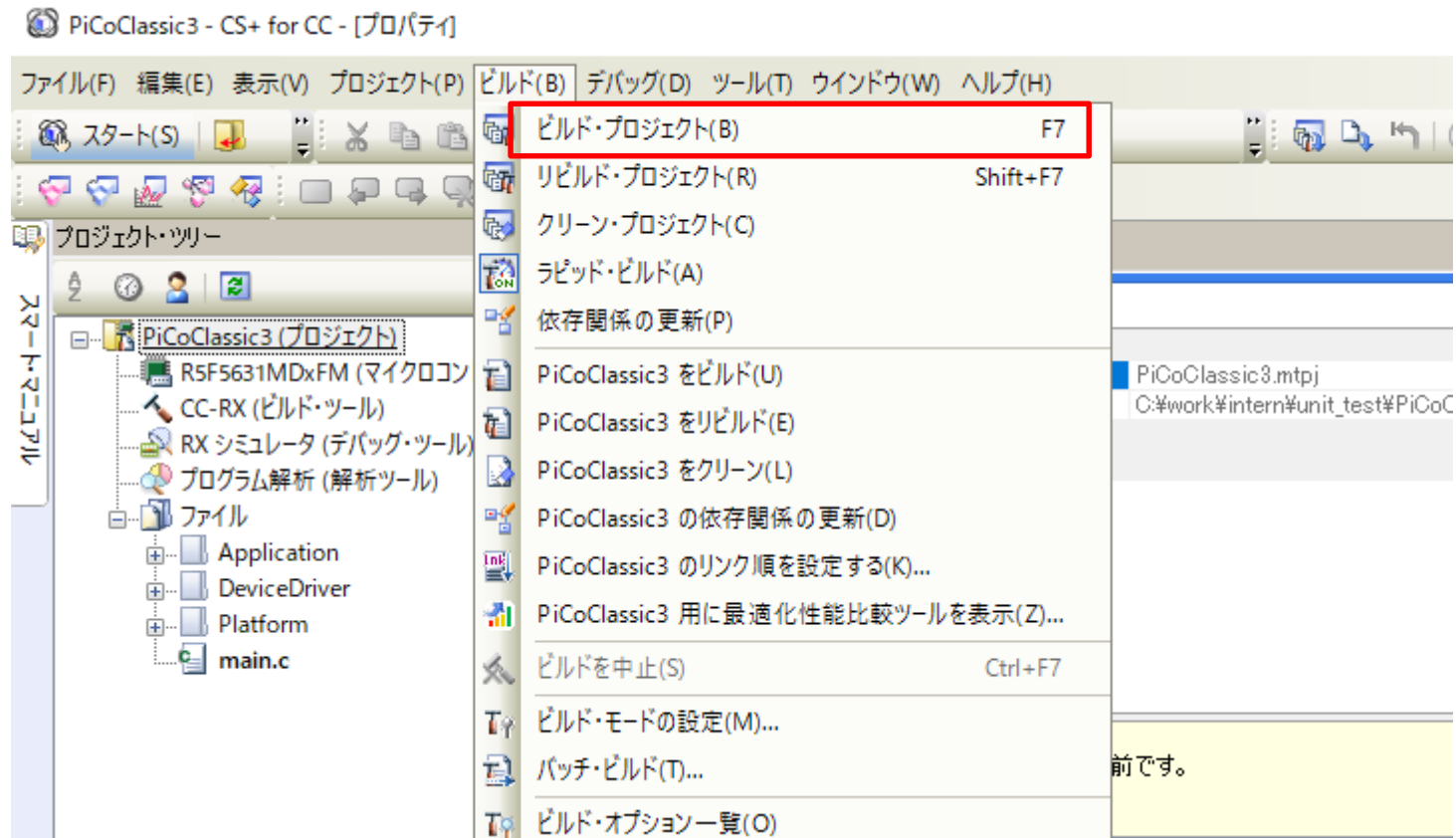
皆さんの作成した関数を呼び出すテスト用プログラムを用意してあります。



2. テスト用プログラムをビルドする

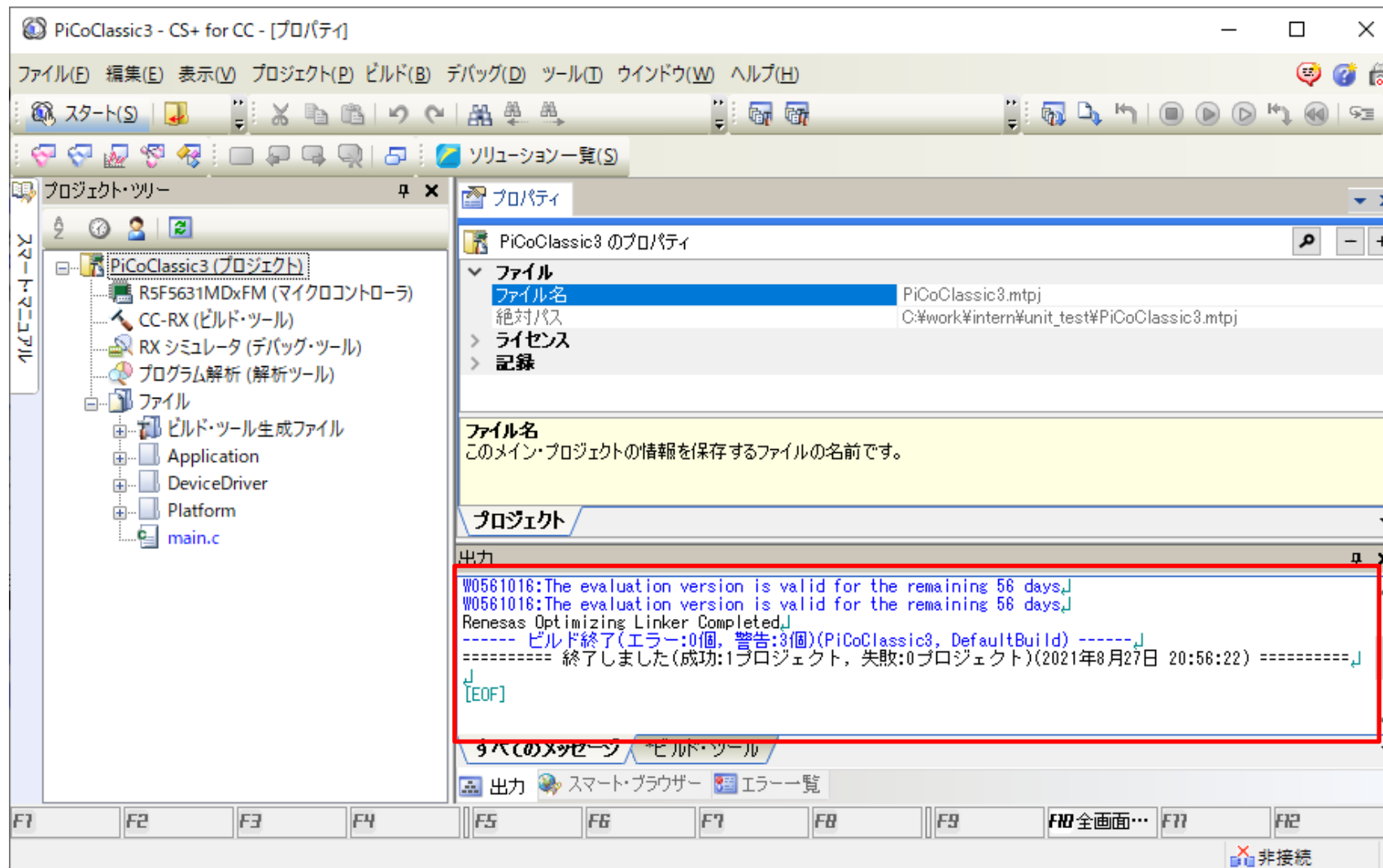
そのままビルドすれば、皆さんの作成したプログラムに対するテスト用プログラムがビルドされます。

メニューバーから「ビルド -> ビルド・プロジェクト(B)」を選択します。



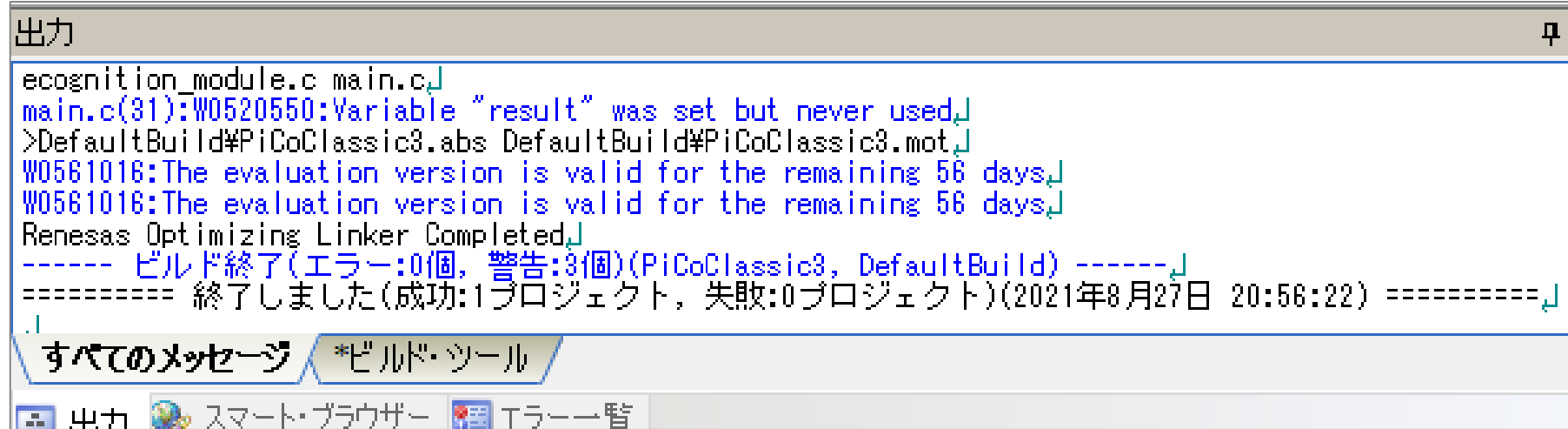
2. テスト用プログラムをビルドする

ビルドを実行すると下記のような画面になります。



2. テスト用プログラムをビルドする

画面の下のメッセージウィンドウの出力に次の文字が表示されたら、ビルドが成功です。



The screenshot shows a software development environment's output window. The title bar is labeled '出力' (Output). The text inside the window is as follows:

```
ecognition_module.c main.c↓  
main.c(31):W0520550:Variable "result" was set but never used↓  
>DefaultBuild#PiCoClassic3.abs DefaultBuild#PiCoClassic3.mot↓  
W0561016:The evaluation version is valid for the remaining 56 days↓  
W0561016:The evaluation version is valid for the remaining 56 days↓  
Renesas Optimizing Linker Completed↓  
----- ビルド終了(エラー:0個, 警告:3個)(PiCoClassic3, DefaultBuild) -----↓  
===== 終了しました(成功:1プロジェクト, 失敗:0プロジェクト)(2021年8月27日 20:56:22) =====↓  
↓
```

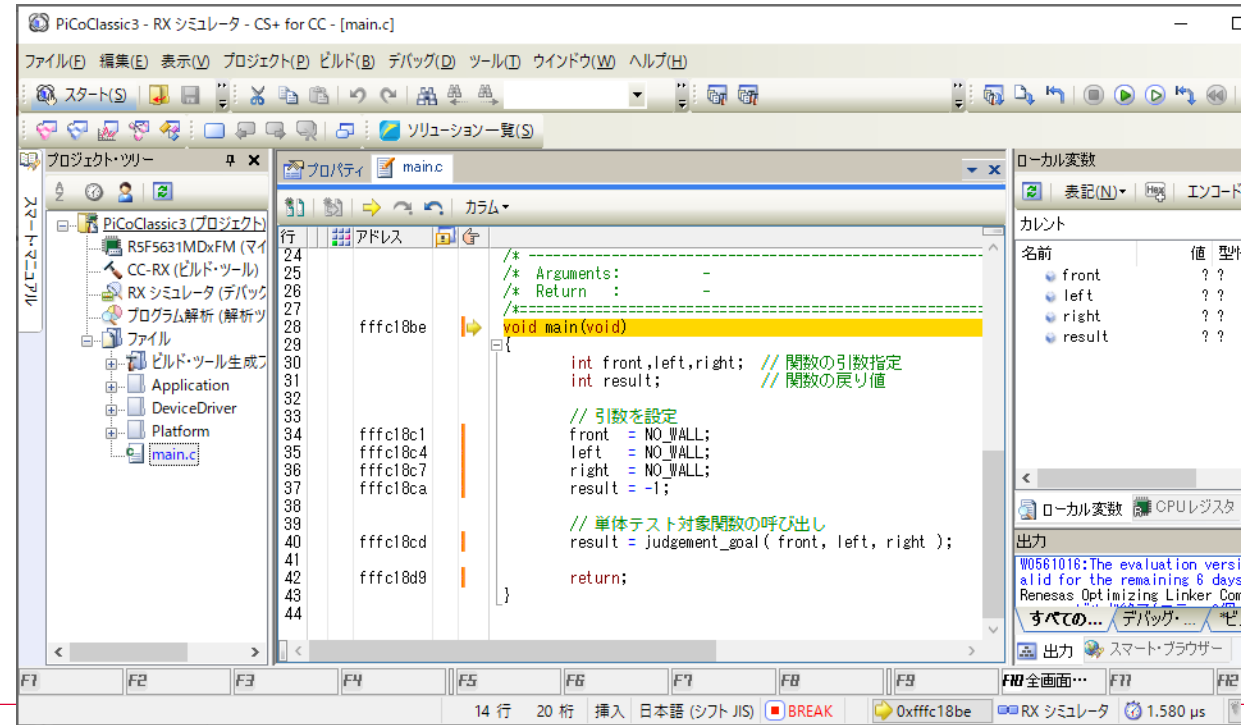
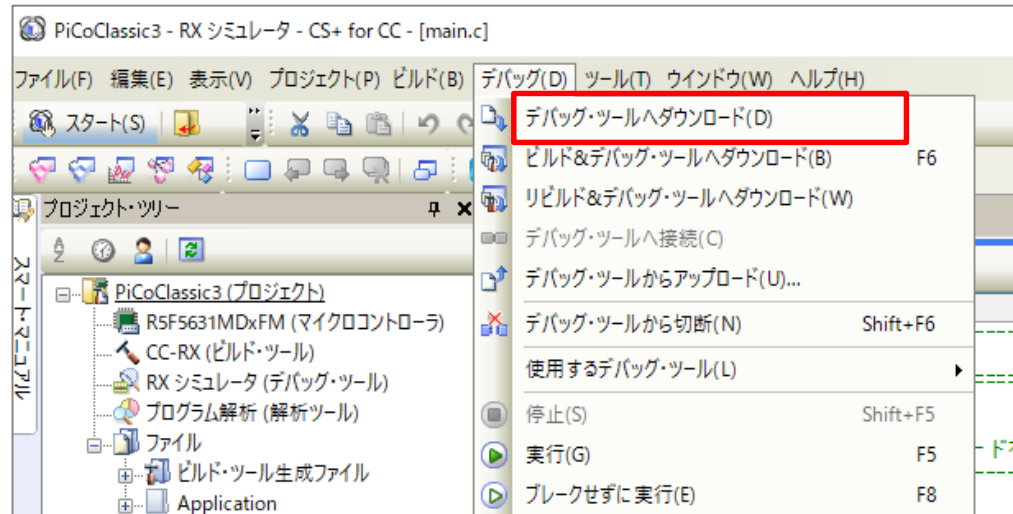
Below the text, there are two tabs: 'すべてのメッセージ' (All Messages) and '*ビルド・ツール' (Build Tools), with the latter being selected. At the bottom of the window, there are three icons with labels: '出力' (Output), 'スマート・ブラウザー' (Smart Browser), and 'エラー一覧' (Error List).

警告が合計3個出ますが問題はありません。

3. テスト用プログラムを動作させる

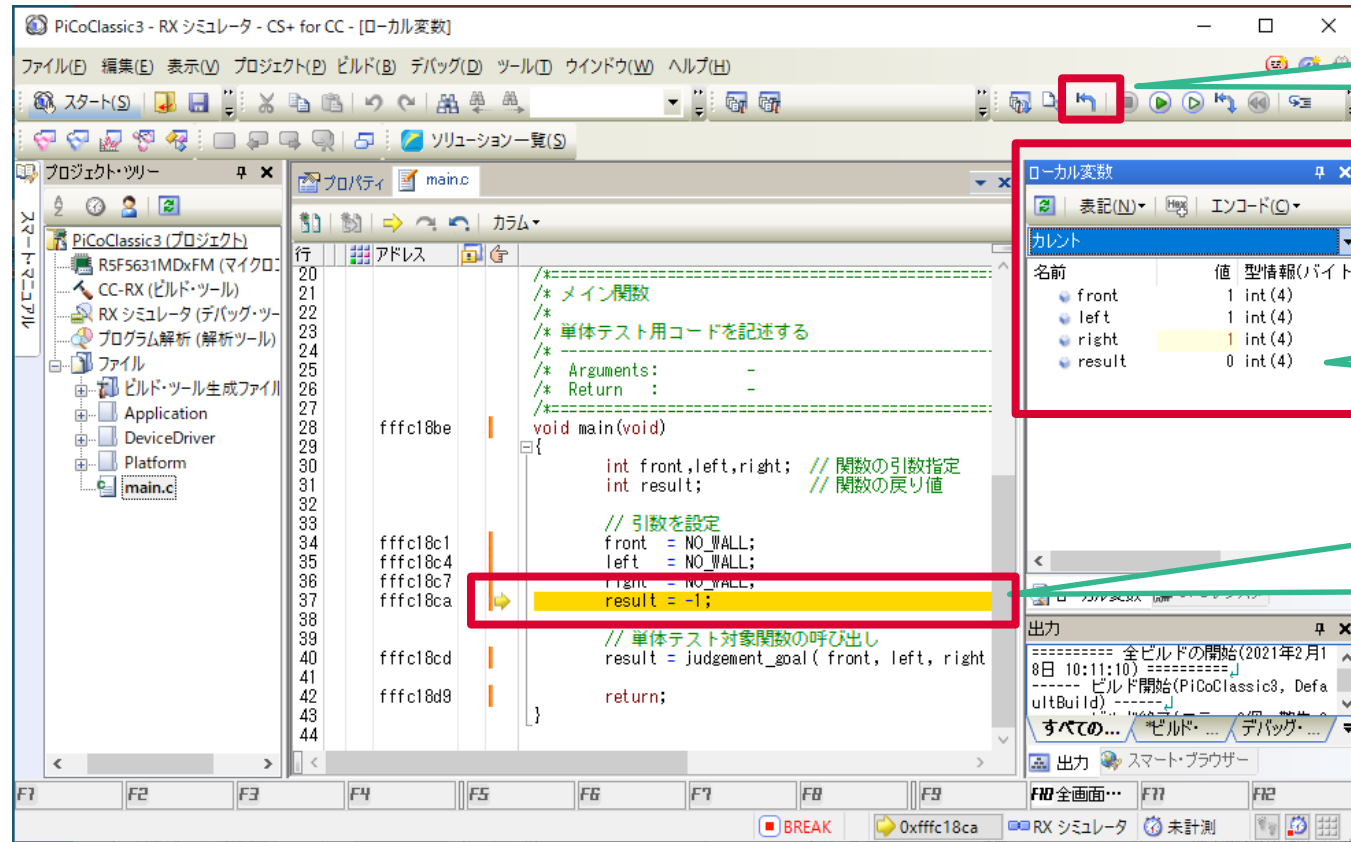
ビルドが成功したら、PC(シミュレータ)上でテスト用プログラムを動作させます。

メニューバーから「デバッグ -> デバッグ・ツールヘダウンロード(D)」を選択します。
ダウンロードが完了すると以下のような画面になり、main関数の開始直前の状態になります。



3. テスト用プログラムを動作させる

これでシミュレータでプログラムを動作させる準備ができました。
シミュレータ動作中の画面は以下のようになっています。



CPUリセットで最初から実行しなおせる

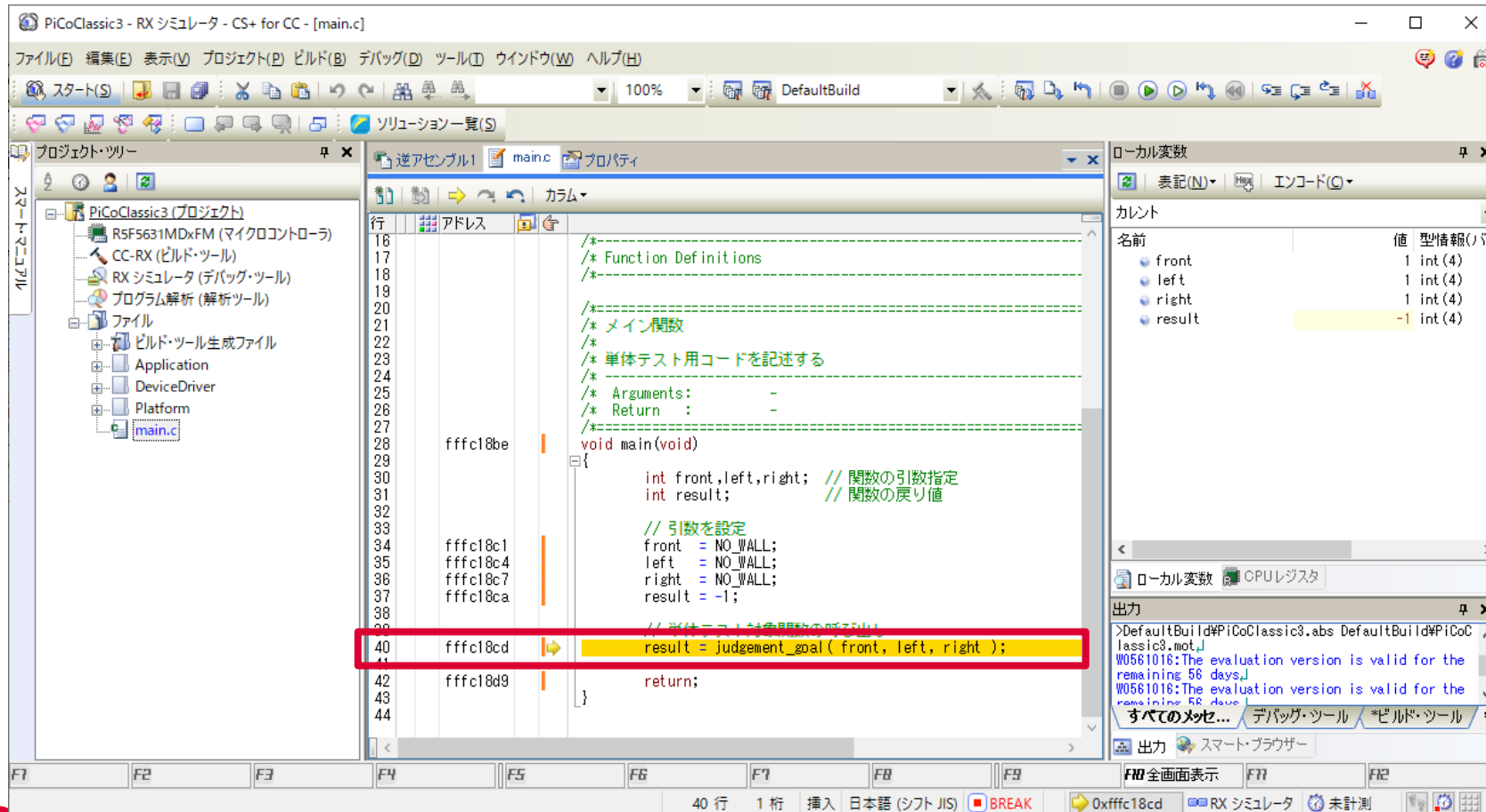
変数の値が確認できる。
また、数字をダブルクリックすると変更もできる。

黄色の行を実行する直前で止まっている。
ステップ・インすると、その行を実行して次の行に移動する

4. 単体テストを実施する

ここから、単体テストを実施します。

メニューバーの「デバッグ(D) -> ステップ・イン(I)」(ショートカットキーはF11)を用いて1行ずつコードを実行し、40行目が黄色くなるまで進めてください。



4. 単体テストを実施する

ここで、設計書に記載された単体テストのテストケースを確認しましょう。
ここから、テストケースをひとつずつ実施します。

まず、実施するテストケースの入力値に合わせて、「ローカル変数」にて対応する変数の値を変更してください。

The screenshot displays the Denso IDE interface. On the left, a table lists test cases for the 'MicroMouse' project. The 'Input Values' column is highlighted with a red box. On the right, the 'Local Variables' window is also highlighted with a red box, showing the values for 'front', 'left', 'right', and 'result'.

No.	テスト内容	入力値	期待値	可否
1.	3面壁ありの場合ゴールと判定する	existence_front_wall=WALL existence_left_wall=WALL existence_right_wall=WALL	goal_judgement_result=0	(未設定)
2.	前壁なしの場合ゴールではないと判定する	existence_front_wall=NO_WALL existence_left_wall=WALL existence_right_wall=WALL	goal_judgement_result=1	(未設定)
3.	左壁なしの場合ゴールではないと判定する	existence_front_wall=WALL existence_left_wall=NO_WALL existence_right_wall=WALL	goal_judgement_result=1	(未設定)
4.	右壁なしの場合ゴールではないと判定する	existence_front_wall=WALL existence_left_wall=NO_WALL existence_right_wall=NO_WALL	goal_judgement_result=1	(未設定)
5.	前壁のみありの場合ゴールではないと判定する	existence_front_wall=WALL existence_left_wall=NO_WALL existence_right_wall=NO_WALL	goal_judgement_result=1	(未設定)
6.	左壁のみありの場合ゴールではないと判定する	existence_front_wall=NO_WALL existence_left_wall=WALL existence_right_wall=NO_WALL	goal_judgement_result=1	(未設定)
7.	右壁のみありの場合ゴールではないと判定する	existence_front_wall=NO_WALL existence_left_wall=NO_WALL existence_right_wall=WALL	goal_judgement_result=1	(未設定)
8.	3面壁なしの場合ゴールではないと判定する	existence_front_wall=NO_WALL existence_left_wall=NO_WALL existence_right_wall=NO_WALL	goal_judgement_result=1	(未設定)

ローカル変数

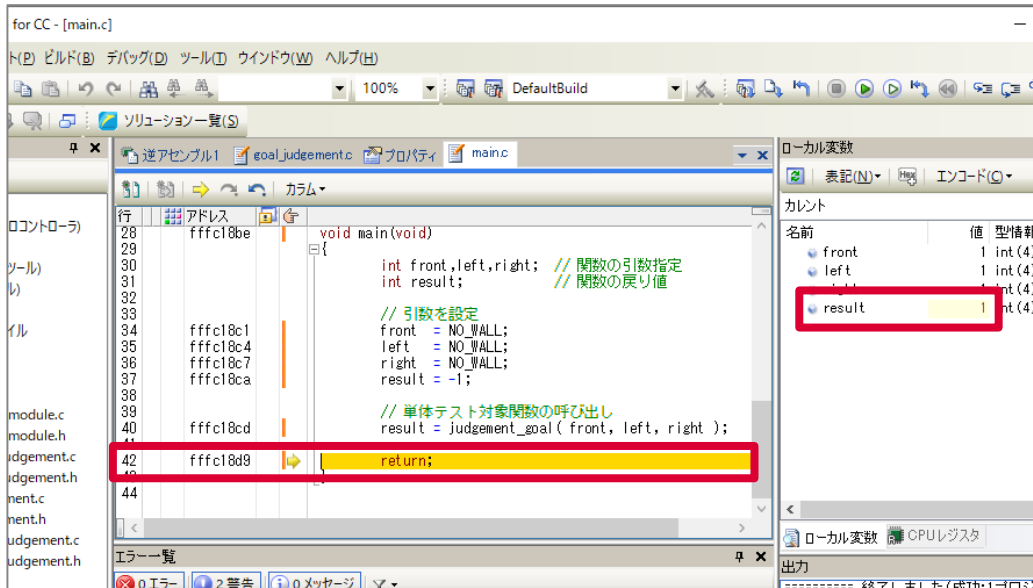
名前	値	型情報(バイ)
front	1	int (4)
left	1	int (4)
right	1	int (4)
result	-1	int (4)

```
/* CS+ for CC - [main.c]
プロジェクト(B) ビルド(B) デバッグ(D) ツール(T) ウィンドウ(W) ヘルプ(H)
100% DefaultBuild
逆アセンブル(main.c) プロパティ
カラム
行 アドレス コード
16 /* Function Definitions
17
18
19
20 /* メイン関数
21
22
23 /* 単体テスト用コードを記述する
24
25 /* Arguments:
26 /* Return:
27
28 void main(void)
29 {
30     int front, left, right; // 関数の引数指定
31     int result;           // 関数の戻り値
32
33     // 引数を設定
34     front = NO_WALL;
35     left = NO_WALL;
36     right = NO_WALL;
37     result = -1;
38
39     // 単体テスト対象関数の呼び出し
40     result = judgement_goal( front, left, right );
41
42     return;
43 }
44
```

4. 単体テストを実施する

「ローカル変数」の値を変更できたら、再びステップ・インを用いて、コードを1行ずつ実行します。
皆さんが作成した関数の中も1行ずつ実行し、想定した通りの箇所を通っているか確認しましょう。

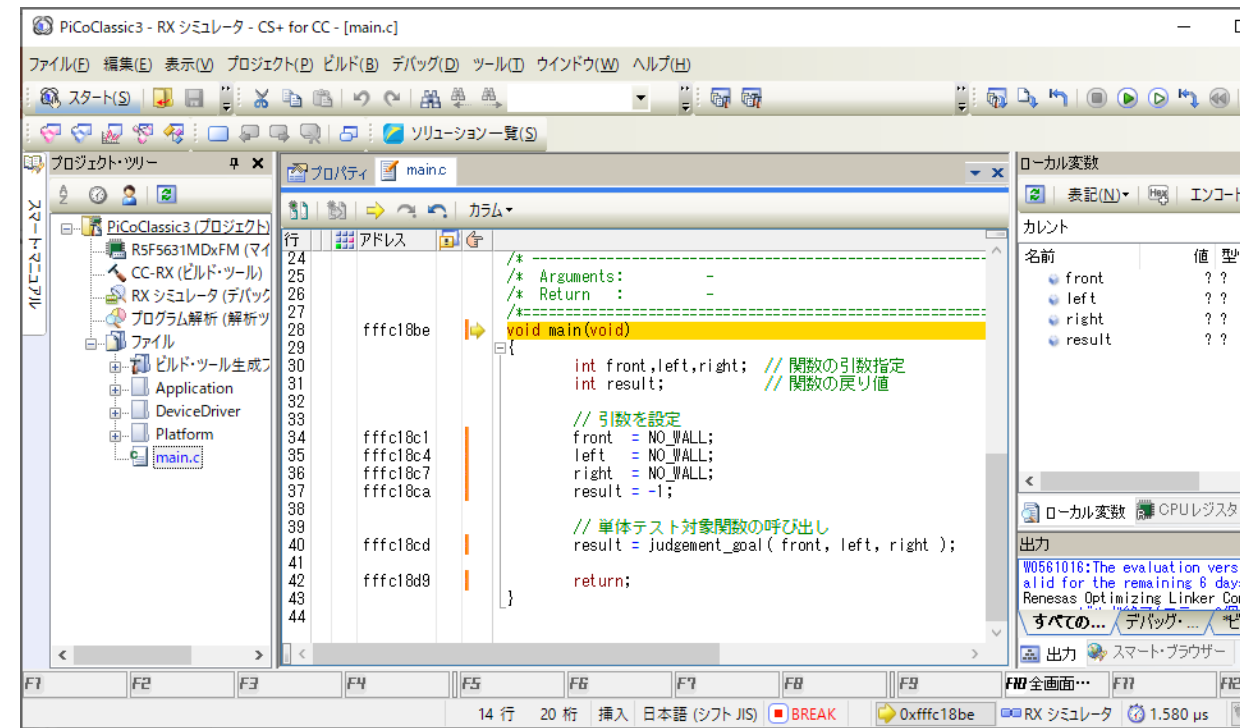
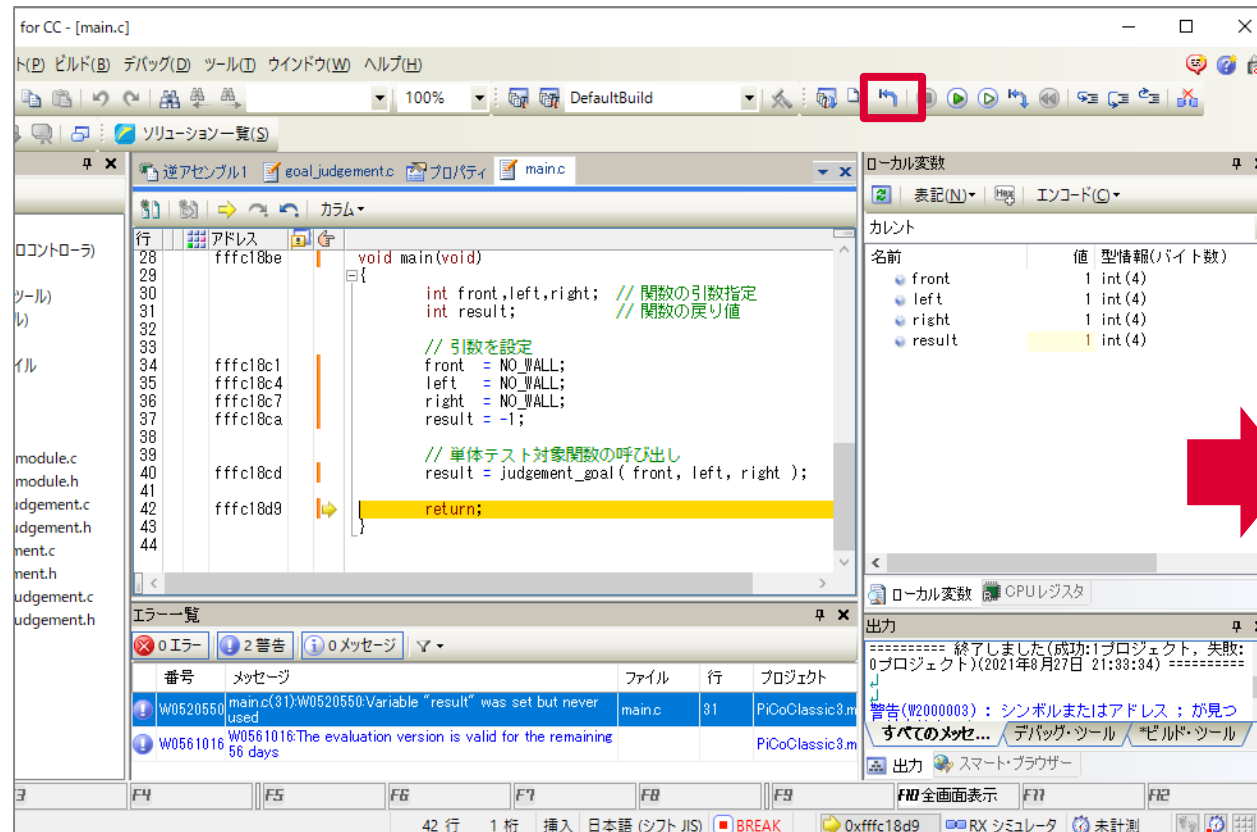
main関数の42行目が黄色くなるまで進めたら、
最後に「ローカル変数」のresultの値が、テストケースの期待値と同じであることが確認できたら、
そのテストケースは実施OKです。



ファイル ホーム 表示 トレーサビリティ プロダクトライン チーム開発 プロファイル ヘルプ				
<div> <div> <div>切り取り</div> <div>コピー</div> <div>貼り付け</div> <div>スタイルのコピー</div> </div> <div> <div>クリップボード</div> <div>モデル</div> <div>エラーチェック</div> </div> </div> <div> <div>前の関連</div> <div>次の関連</div> </div> <div> <div>クイックスタイル</div> <div>線</div> </div> <div> <div>順序</div> <div>品レイアウト</div> </div> <div> <div>サイズ追加</div> <div>ルーティン</div> </div> <div> <div>要素表示</div> </div>				
<div> <div>MicroMouse</div> <div>システム要求</div> <div>ソフトウェア仕様</div> <div>ソフト構造</div> <div>単体テスト</div> <div>ゴール判定</div> <div>外部テスト</div> <div>テストケース</div> <div>データ型</div> <div>int</div> <div>unsigned short</div> </div> <div> <div>キーワード</div> </div> <div> <div>新規</div> <div>参照</div> <div>入力</div> </div> <div> <div>追加できる要素がありません。</div> </div>				
No.	テスト内容	入力値	期待値	可否
1.	3面壁ありの場合ゴールと判定する	existence_front_wall=WALL existence_left_wall=WALL existence_right_wall=WALL	goal_judgement_result=0	(未設定)
2.	前壁なしの場合ゴールではないと判定する	existence_front_wall=NO_WALL existence_left_wall=WALL existence_right_wall=WALL	goal_judgement_result=1	(未設定)
3.	左壁なしの場合ゴールではないと判定する	existence_front_wall=WALL existence_left_wall=NO_WALL existence_right_wall=WALL	goal_judgement_result=1	(未設定)
4.	右壁なしの場合ゴールではないと判定する	existence_front_wall=WALL existence_left_wall=WALL existence_right_wall=NO_WALL	goal_judgement_result=1	(未設定)
5.	前壁のみありの場合ゴールではないと判定する	existence_front_wall=WALL existence_left_wall=NO_WALL existence_right_wall=NO_WALL	goal_judgement_result=1	(未設定)
6.	左壁のみありの場合ゴールではないと判定する	existence_front_wall=NO_WALL existence_left_wall=WALL existence_right_wall=NO_WALL	goal_judgement_result=1	(未設定)
7.	右壁のみありの場合ゴールではないと判定する	existence_front_wall=NO_WALL existence_left_wall=NO_WALL existence_right_wall=WALL	goal_judgement_result=1	(未設定)
8.	3面壁なしの場合ゴールではないと判定する	existence_front_wall=NO_WALL existence_left_wall=NO_WALL existence_right_wall=NO_WALL	goal_judgement_result=1	(未設定)

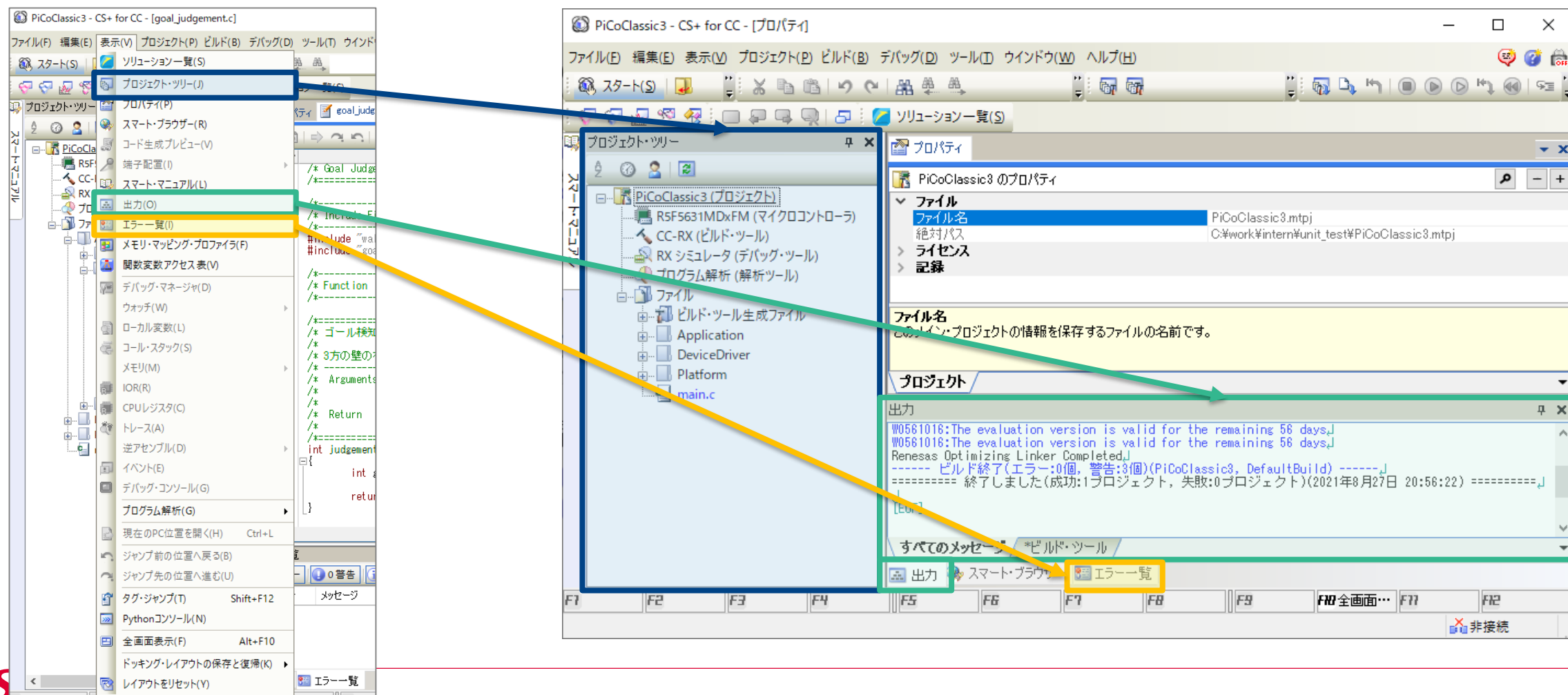
4. 単体テストを実施する

一つのテストケースを実施し終わったら、
「デバッグ(D) -> CPUリセット(T)」または、ツールバー上のCPUリセットボタンを押し、
テストケース実施前の状態に戻します。
そしてそこから、一つ目のテストケースと同じ手順で以降のテストケースを実施していきます。



参考. 各種ウィンドウが消えてしまったときは (ソースコード作成時)

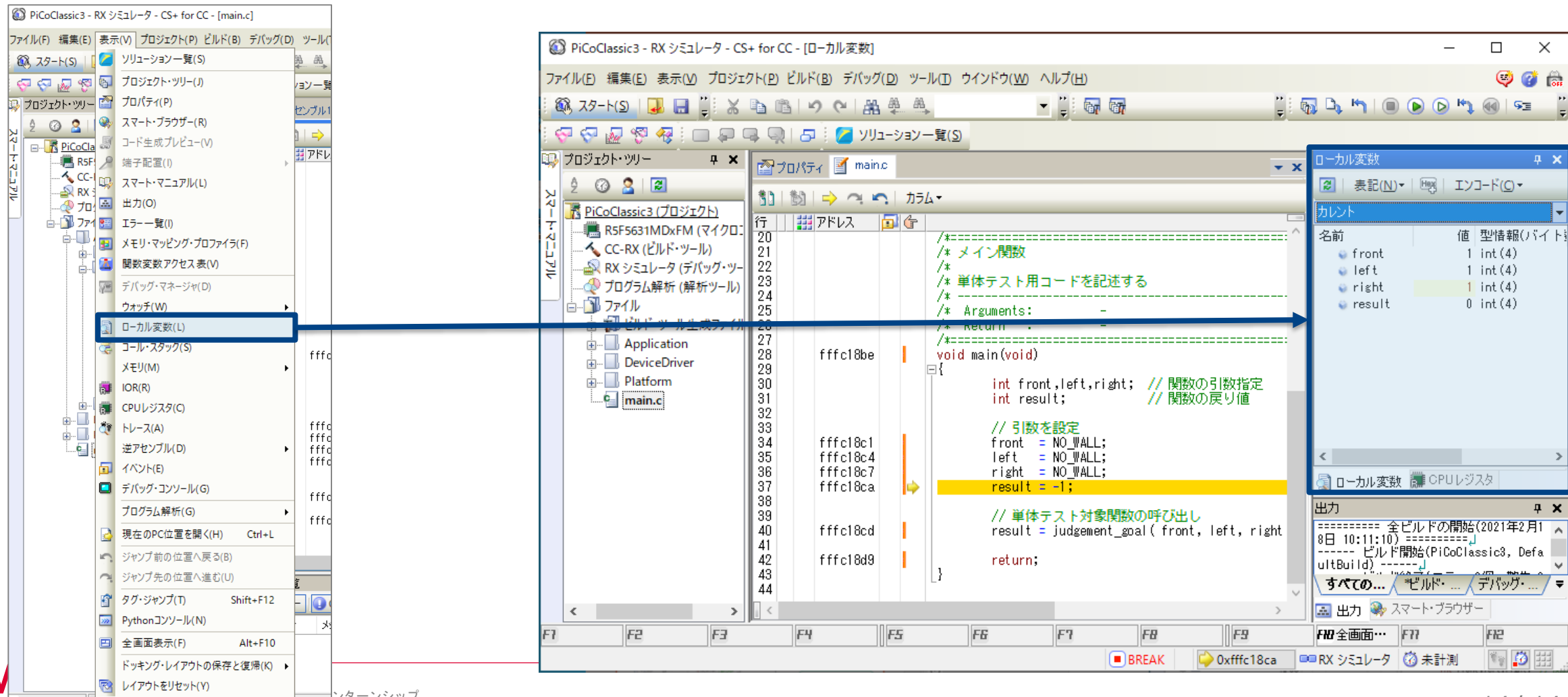
ツール内の各種ウィンドウが消えてしまったときは、
「表示(V)」メニューから表示したいウィンドウを選択すると再表示されます。



参考. 各種ウィンドウが消えてしまったときは（単体テスト実行時）

CONFIDENTIAL
関係者外秘

ツール内の各種ウィンドウが消えてしまったときは、
「表示(V)」メニューから表示したいウィンドウを選択すると再表示されます。



DENSO

Crafting the Core