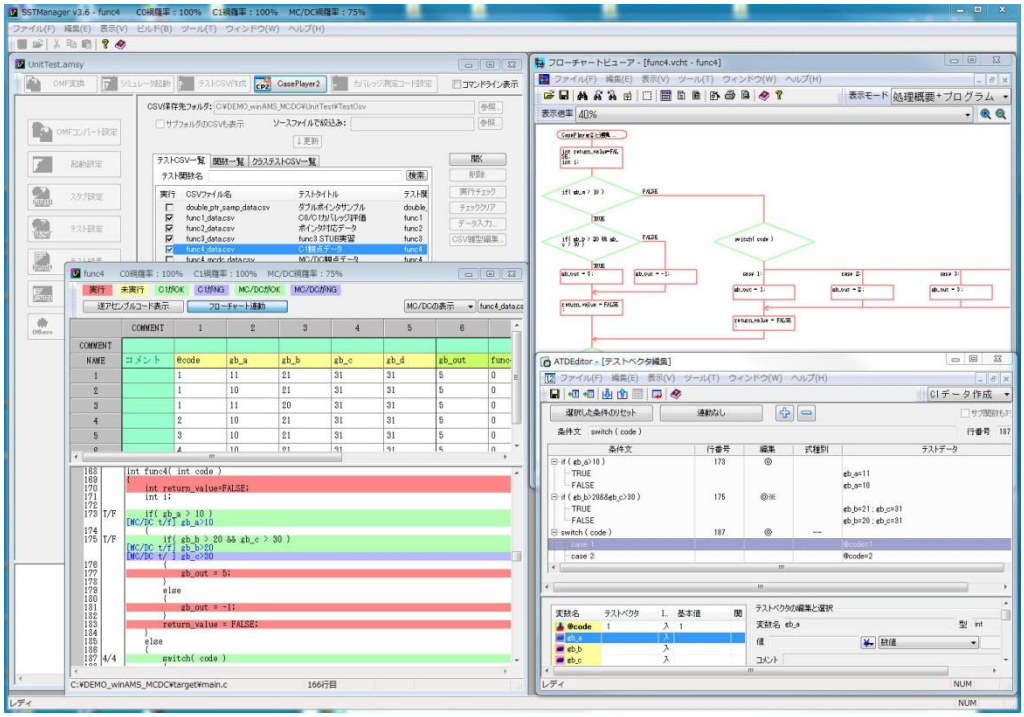
Solution House GAIO TECHNOLOGY



カバレッジマスターwinAMS MBT オプション チュートリアル

2023 年 07 月 18 日
第 2.0.0 版

目次

カバレッジマスターwinAMS MBT オプションチュートリアル

はじめに

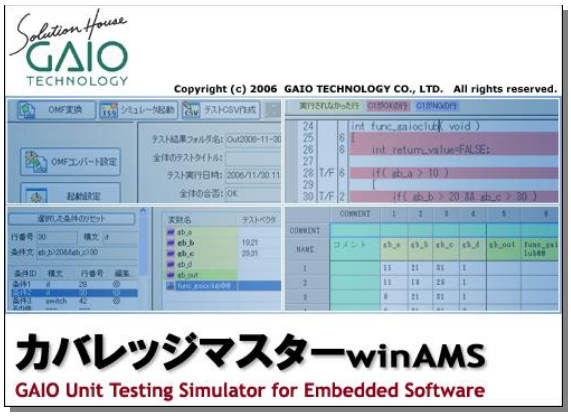
このたびは、ガイオ単体テストツール「カバレッジマスターwinAMS」をご利用いただきまして、誠にありがとうございます。カバレッジマスターwinAMS は、組込みソフト品質評価・改善のための単体テストを、効率化するための検証用ツールです。

他方では、MathWorks 社の MATLAB/Simulink を中心にモデルベース開発の技術や考え方が、組み込み製品の開発にも適用され、その範囲を拡大しています。MathWorks 社が提供するツール等の進化により、カバレッジマスターwinAMS にもモデルベース開発におけるテストの機能追加が求められてきました。

そこで、カバレッジマスターwinAMS V7.0 では、モデルベース開発における Back-to-Back テストのための MBT オプション機能を追加しました。

本チュートリアルは、カバレッジマスターwinAMS をご利用頂いているユーザーに、MBT オプションの使い方や機能を、実習を通して学べるように構成されています。実習は、カバレッジマスターwinAMS、CasePlayer2 の基本的な使い方を習得していることが前提となっています。

尚、本チュートリアルでは、カバレッジマスターwinAMS V7.0 以降の機能を使用しています。V7.0 以前のバージョンでは使用できませんので、V7.0 以降のバージョンをインストールしてご利用ください。



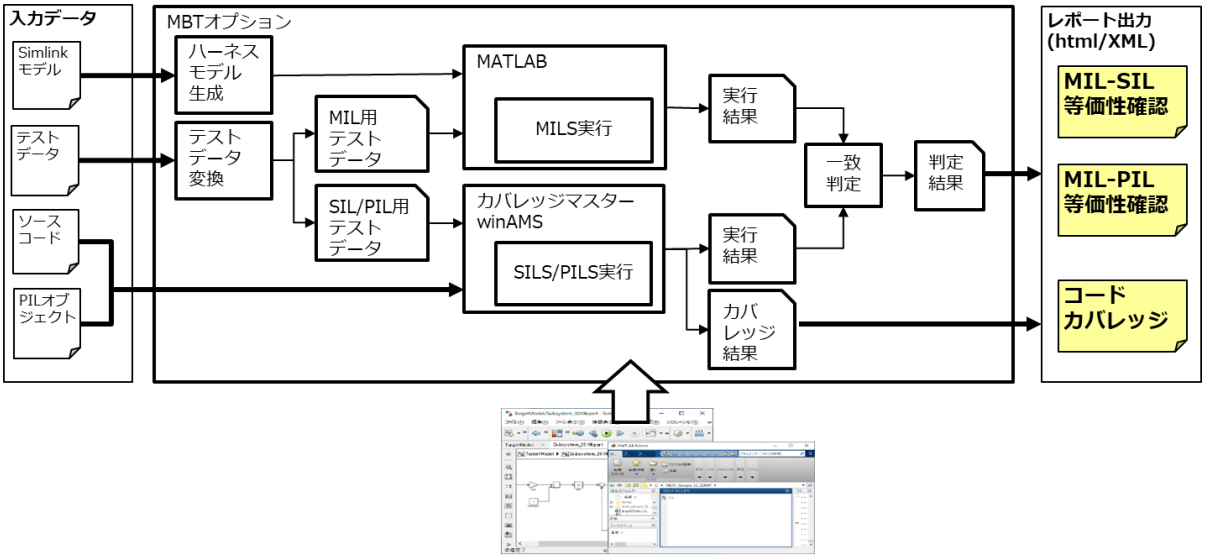
MBT オプション概要

MBT オプションとは

MBT オプションは、モデルベース開発における Back-to-Back テストのための、カバレッジマスターwinAMS のオプションです。MATLAB/Simulink とカバレッジマスターwinAMS/CasePlayer2 を使用して、モデルとソースコード/ターゲットコードの間の動作一致性を確認できます。MBT オプションの主な役割及び機能は以下の通りです。

MBT オプションの各機能は、モデルベース開発における標準ツールである MATLAB/Simulink に統合化されています。MATLAB/Simulink のコマンドラインインターフェース(CLI)にて、カバレッジマスターwinAMS の実行結果とモデルの実行結果の Back-to-Back(BtoB)テスト評価および、テストエビデンス生成を行えます。

また、MATLAB/Simulink のmファイルを使い、各機能を自動化できます。



MATLAB/Simulink

MBT オプション構成

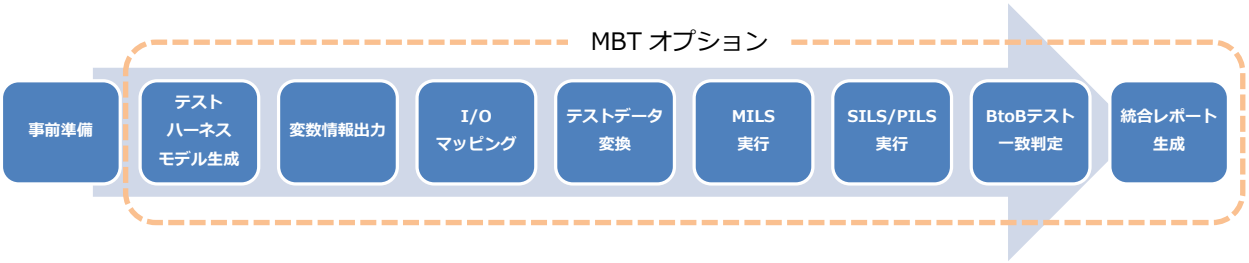
MBT オプション機能一覧

機能	概要
テストハーネスモデル生成機能	サブシステム単体で部分的に検証するためのテストハーネスモデルを作成
変数情報出力機能	I/O マッピングに使用する変数情報を作成
I/O マッピング機能	信号と変数のマッピングを行い、マップリストを作成 マップリスト作成時に許容誤差を設定可
テストデータ変換機能	コードテスト用 CSV ファイルと、モデルテスト用 CSV ファイルを作成 変換の際、LSB の設定に応じて小数点桁数を合わせます
自動テストデータ生成機能	コード情報からテストデータ CSV を作成
MILS/SILS/PILS 実行機能	MILS では、テストハーネスモデルにモデルテスト用 CSV ファイルのデータを入力し、 モデル実行結果ログを作成、 SILS/PILS はソースコードやオブジェクトコードのシミュレーション実行を行い、 コード実行結果ログを作成
コードデバッグ機能	ソースコードやオブジェクトコードのデバッグが可能
Back-to-Back テスト 一致判定機能	コード実行結果ログとモデル実行結果ログを比較して一致判定を行い、 比較結果データファイルを作成
統合レポート生成機能	XML、HTML フォーマットで帳票出力

※グレー行の機能は、カバレッジマスターwinAMS 従来機能

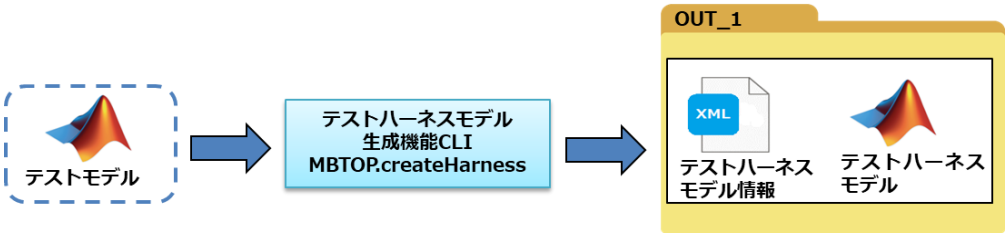
MBT オプション機能説明

MBT オプションを使用した Back-to-Back テスト実行は以下の手順で行います。Back-to-Back テスト実行を行うためには、事前に CasePlayer2、カバレッジマスターwinAMS のテスト実行環境の構築を行う必要があります。
ここでは、MBT オプションのそれぞれの機能を説明します。
機能の詳細については「MBT オプション ユーザーズマニュアル.pdf」を参照ください。



テストハーネスモデル生成機能

Back-to-Back 対象のサブシステムに対して、外部からテストデータを入力し、実行結果取得に必要なポートを接続した単体テスト用のモデル(テストハーネスモデル)を作成します。



変数情報出力機能

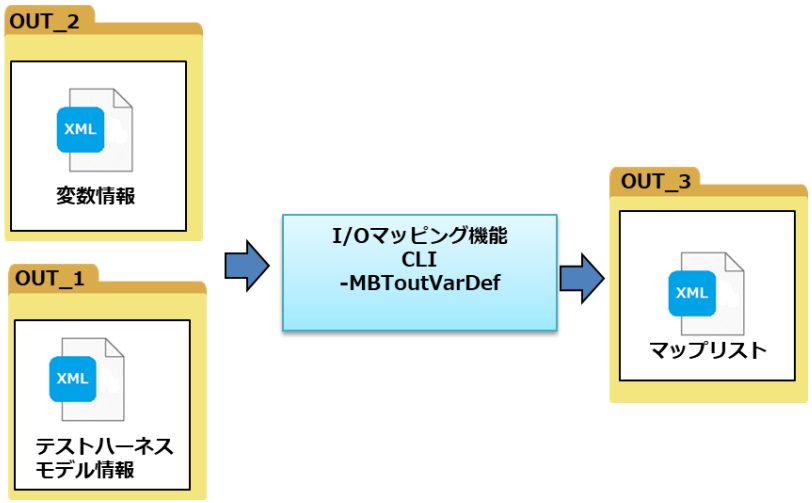
I/O マッピング時に使用するコード側シンボル情報(変数情報)を winAMS プロジェクト情報から取得します。
取得した情報から、指定するテスト対象関数(サブ関数含む)で使用されている変数情報(変数名、型の一覧)をファイルに出力します。

I/O マッピング機能

マッピングとは Back-to-Back 検証時、比較対象となるモデル側シンボル(信号、ブロック)とコード側シンボル(変数)のマッピング(対応付け)を行います。
I/O マッピング機能では、ハーネスモデル生成時に取得したモデル情報と変数情報出力機能取で取得した変数情報のシンボルを、以下の手法でマッピングします。

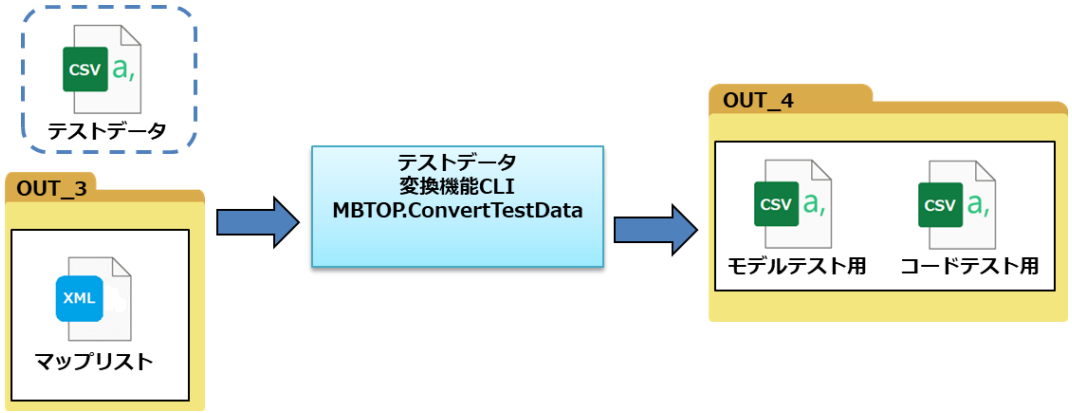
- 名前一致による自動マッピング（完全一致 / 部分一致 / 前方一致 / 後方一致）
- マップリストによる手動マッピング
- DD(TargetLink)ファイルによるマッピング

また、マップリスト作成時に全体の許容誤差を設定できます。信号個別の許容誤差や固定小数点設定、前回値設定は、マップリストを編集することで設定できます。



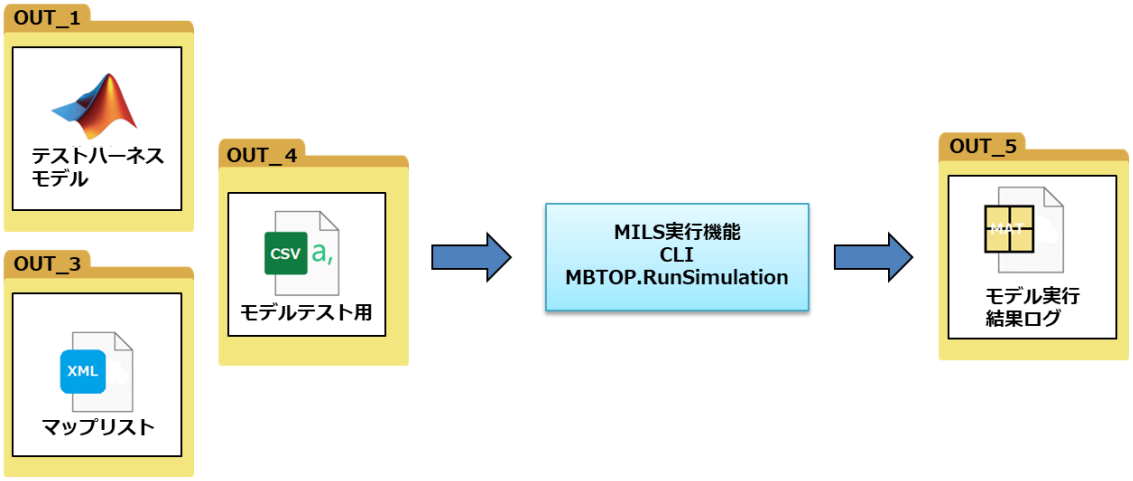
テストデータ変換機能

モデルやソースコードから作成したテスト CSV ファイルから、コードテスト用 CSV ファイルと、モデルテスト用 CSV ファイルを作成できます。
固定小数点設定に従って、モデルから作成した物理値のデータはデジタル値に変換し、コードから作成したデジタル値のデータは物理値に変換します。



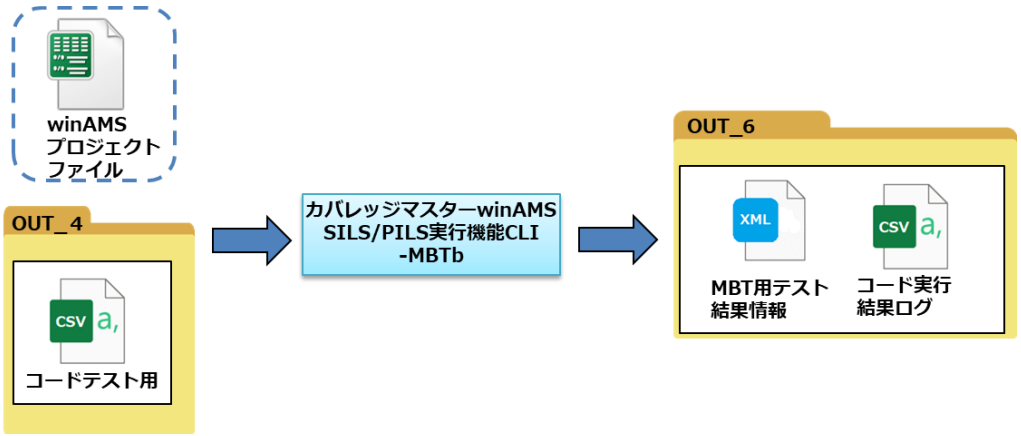
MILS/SILS/PILS 実行機能

MILS 実行機能では、テストハーネスモデルに、モデルテスト用 CSV ファイルのデータを入力し、モデル実行結果ログを作成できます。



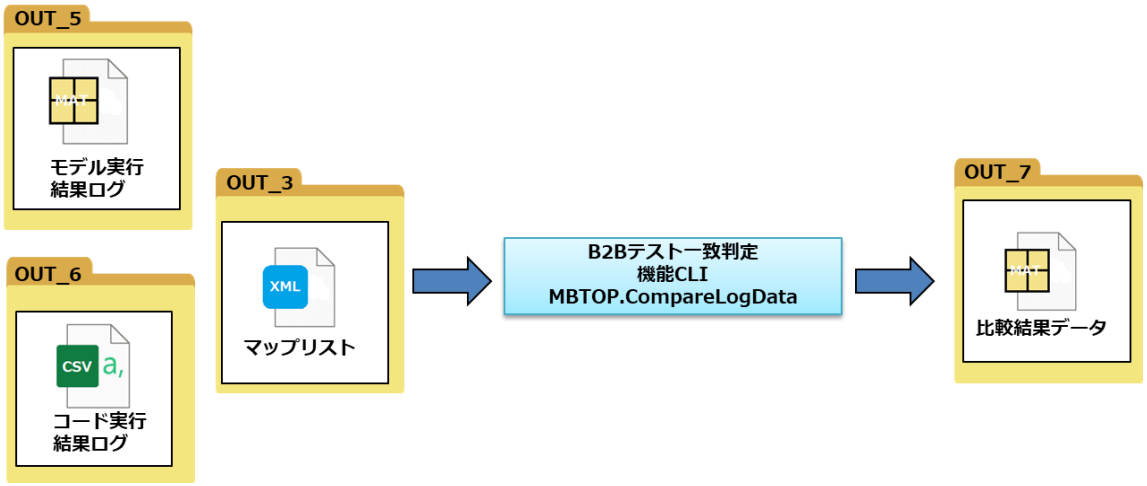
SILS/PILS 実行機能は、カバレッジマスターwinAMS と CasePlayer2 を使用して、ソースコードやオブジェクトコードのシミュレーション実行を行い、コード実行結果ログを作成できます。この時、カバレッジマスターwinAMS の設定によって、コードカバレッジを取得できます。

SILS には、カバレッジマスターwinAMS の「General」マイコンを使用します。PILS には、カバレッジマスターwinAMS の各種マイコンシミュレータを使用します。



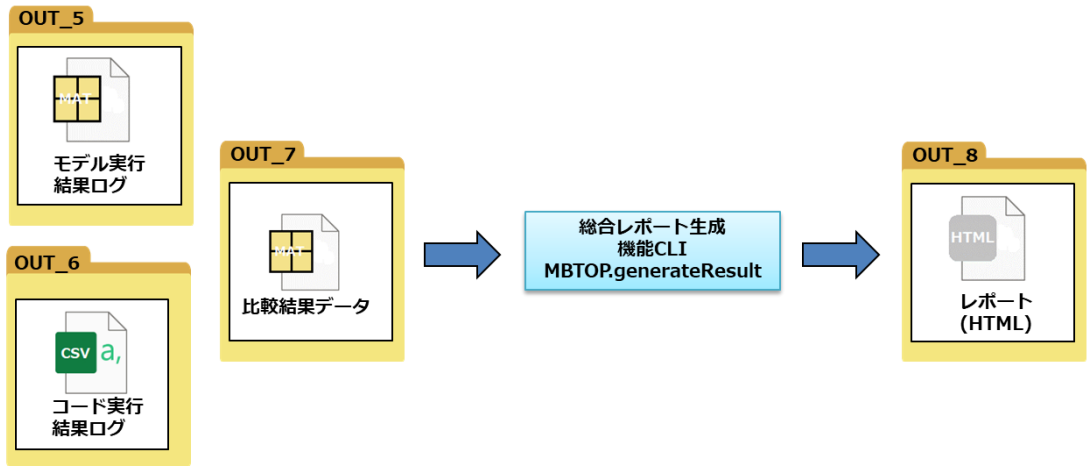
Back-to-Back テスト一致判定機能機能

コード実行結果ログとモデル実行結果ログを比較して、一致判定を行い、比較結果データファイルを作成できます。比較対象の信号と変数ごとに許容誤差を踏まえた誤差評価を行えます。



統合レポート生成機能

比較結果データファイル、モデル実行結果ログ、コード実行結果ログをまとめて、XML、HTML フォーマットで帳票出力ができます。



MBT オプション チュートリアル環境説明

ここでは MBT オプションに対応したチュートリアル環境の構成を説明します。稼働環境については、MBT オプションユーザーズマニュアルを参照してください。

なお、サンプル環境では、「MILS-PILS」の一致判定を行います。

サンプル環境フォルダ構成

サンプル環境のフォルダ構成を示します。

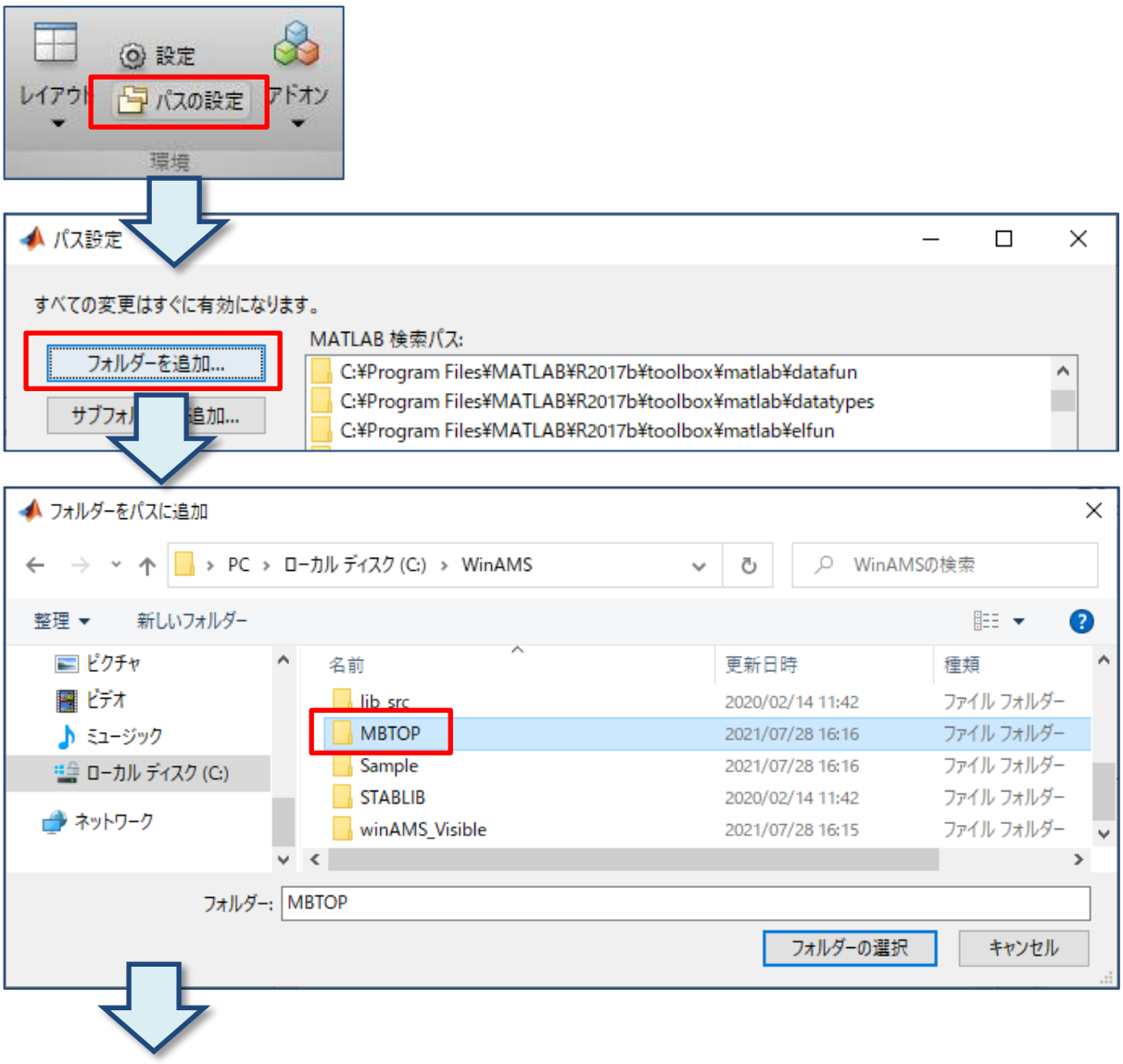
実習1：基本的な MBT オプションコマンドを学習

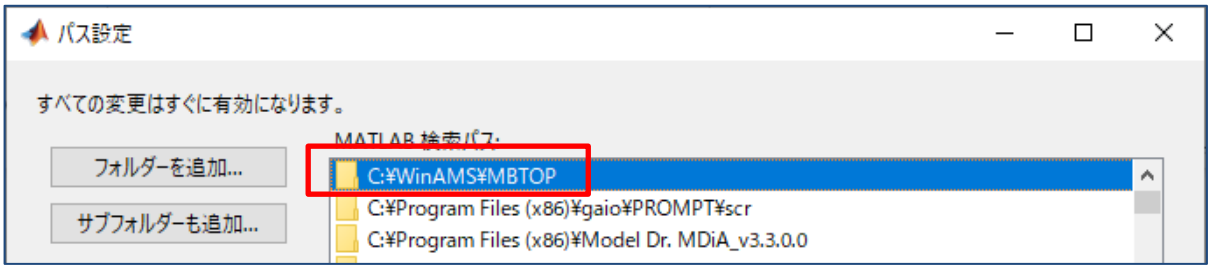
実習1では、MBT オプションの基本的な実行方法を学習します。
本チュートリアルで記載のないコマンドオプション、戻り値の情報は MBT オプションユーザーズマニュアルを参照してください。

1.1. 事前準備

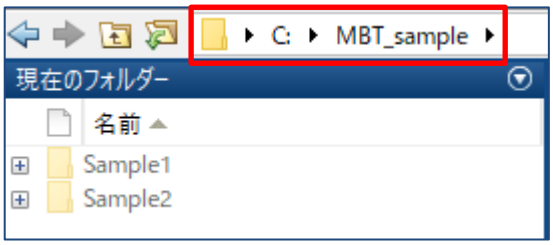
ダウンロードした MBT_sample.zip を展開し、C ドライブ直下へ配置します。

MBT オプションを初めて使用するには、利用する MATLAB/Simulink の「パスの設定」に MBT オプションの実行ファイルのフォルダパスを設定する必要があります。





MATLAB を立ち上げ、現在のフォルダに、展開したサンプル環境のフォルダ MBT_sample を選択します。



浮動小数点精度を統一するため、環境変数を設定するため、MATLAB コマンドウィンドウで以下のコマンドを実行します。

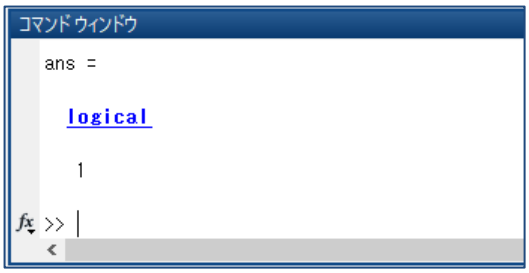
```
setenv('GAIO_CMW_STANDARD_DIGITS', '1');
```

1.2. MBT オプション起動コマンド実行

MBT オプションの各機能を使用するためには、MBT オプションライセンスの確認と取得を行う必要があります。
ライセンス取得は startMBTOP コマンドを使用します。
MATLAB のコマンドウィンドウで MBT オプションの起動コマンドを実行します。

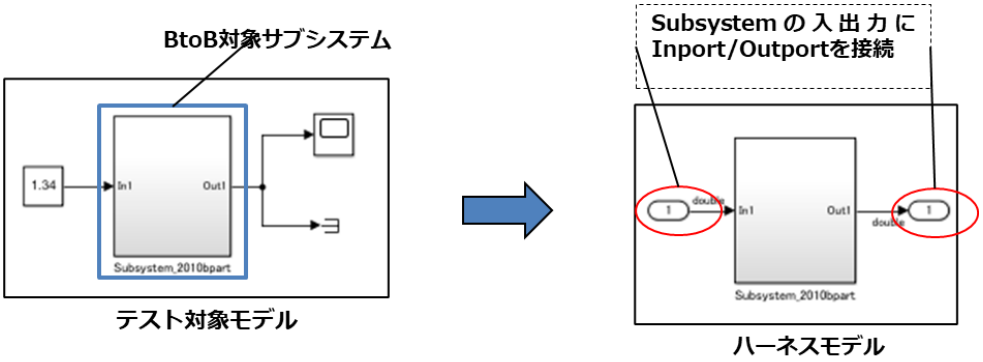
```
MBTOP.startMBTOP()
```

正しく実行されると、コマンドウィンドウに 1 という戻り値が返され、次のコマンドが入力できるようになります。



1.3. テストハーネスモデル生成実行

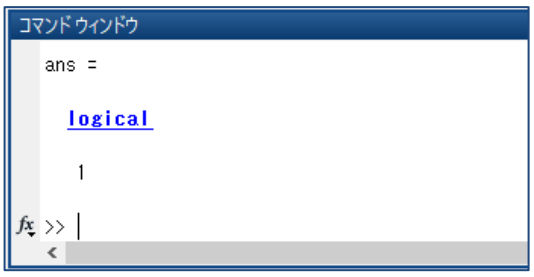
テストハーネスモデル生成機能では、ターゲットモデル内のテスト対象サブシステムを選択し、「テストハーネスモデル」と「テストハーネスモデル情報ファイル」を生成します。



コマンドは createHarness を使用し、ここで指定するオプションは“モデルファイルパス”、“検証対象サブシステム”、“出力先フォルダパス”を指定します。

```
MBTOP.createHarness('Sample1/TargetModel.mdl',
'TargetModel/Subsystem_2010bpart', 'Work1')
```

正しく実行されると、Work1 フォルダが新規に作成され、フォルダ内にハーネスモデル「TargetModel.mdl」が生成されます。また、コマンドウィンドウには 1 という戻り値が返され、次のコマンドが入力できるようになります。



1.4. 変数情報出力

引数に指定した関数名に対して、カバレッジマスターwinAMS のプロジェクト情報から変数情報ファイルを作成します。

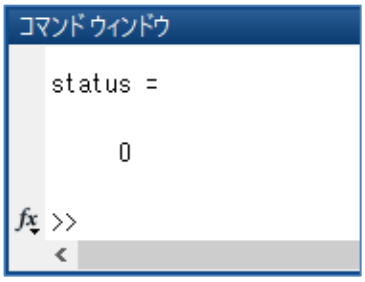
変数情報出力コマンドは、カバレッジマスターwinAMS を実行するため、専用の exe ファイルを実行します。MATLAB コマンドウィンドウ上で CLI を実行する場合は、MATLAB の「system」コマンド または「！」(感嘆符)を使用する必要があります。ここでは、system コマンドを使用した例を紹介します。

変数情報出力は“AmsCommand.exe”＋“-MBToutVarDef”オプションにて実行し、オプションに“テスト対象関数名”、“出力ファイルパス”、“プロジェクトファイルパス”を指定します。system コマンドを使用する場合、各オプション間にはスペースを入れる必要がありますのでご注意ください。

```
status = system(['"C:¥winAMS¥BIN¥AmsCommand.exe"',' -MBToutVarDef',' Subsystem_2010bpart_step', ' -outFile',' ..¥..¥Work1¥varList.xml',' Sample1¥AMS_MBT_DEMO¥AMS_MBT_DEMO.amsy'])
```

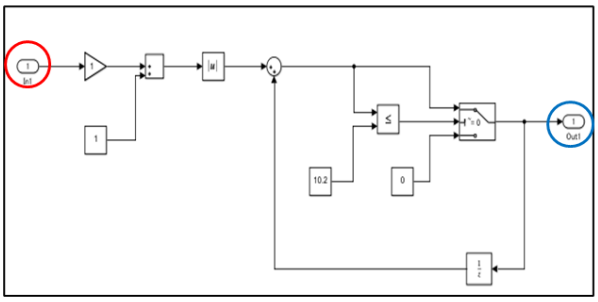
正しく実行されると、Work1 フォルダ内に変数情報ファイル「varList.xml」が生成されます。また、コマンドウィンドウには 0 という戻り値が返され、次のコマンドが入力できるようになります。

戻り値は、0 が正常終了、1、2 がエラー終了、258、260 は警告レベルとなります。



1.5. I/O マッピング実行

モデル情報と変数情報から、名前が一致する情報をマッチングし、「マップリスト」ファイルを作成します。
サンプルモデルのサブシステムの入力、出力に対し、コード側の変数名を紐づけます。ここでは名前一致でのマッピングの後方一致（BackwardMatch）を使用します。



BtoB検証対象のサブシステム

```
void Subsystem_2010bpart_step(void)
{
  real_T rtb_Sum1;

  /* Sum: '<S1>/Sum1' incorporates:
   *  Abs: '<S1>/Abs'
   *  Constant: '<S1>/Constant1'
   *  Inport: '<Root>/In1'
   *  Sum: '<S1>/Sum'
   *  UnitDelay: '<S1>/Unit Delay'
   */
  rtb_Sum1 = fabs(rtU.In1) + rtConstant1) + rtDW.UnitDelay_DSTATE;

  /* Switch: '<S1>/Switch' incorporates:
   *  Constant: '<S1>/Constant2'
   *  Constant: '<S1>/Constant4'
   *  RelationalOperator: '<S1>/Relational Operator'
   */
  if (!(rtb_Sum1 <= rtConstant4)) {
    rtb_Sum1 = rtConstant2;
  }

  /* End of Switch: '<S1>/Switch' */

  /* Outport: '<Root>/Out1' */
  rtY.Out1 = rtb_Sum1;

  /* Update for UnitDelay: '<S1>/Unit Delay' */
  rtDW.UnitDelay_DSTATE = rtb_Sum1;
}
```

BtoB検証対象の関数

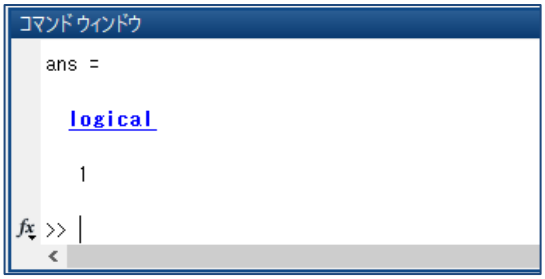
	モデル側	コード側
入力	In1	rtU_In1
出力	Out1	rtY_Out1

Back-to-Back テストの比較対象シンボル

コマンドは AutoMapping を使用し、オプションに“テストハーネスモデル情報”, “変数情報”, “オプション情報”を指定します。また、引数のオプション情報（OptionSettings）は構造体で指定する必要があるため、事前に定義をしています。

```
OptionSettings.mapKind = 'BackwardMatch'
OptionSettings.roundingMethod = 'Round'
OptionSettings.mapListFilePath = 'Work1/mapList.xml'
MBTOP.AutoMapping('Work1/blockInfo.xml', 'Work1/varList.xml', OptionSettings)
```

正しく実行されると、Work1 フォルダ内にマップリストファイル「mapList.xml」、ポインタリストファイル「mapList_p.xml」が生成されます。また、コマンドウィンドウには 1 という戻り値が返され、次のコマンドが入力できるようになります。

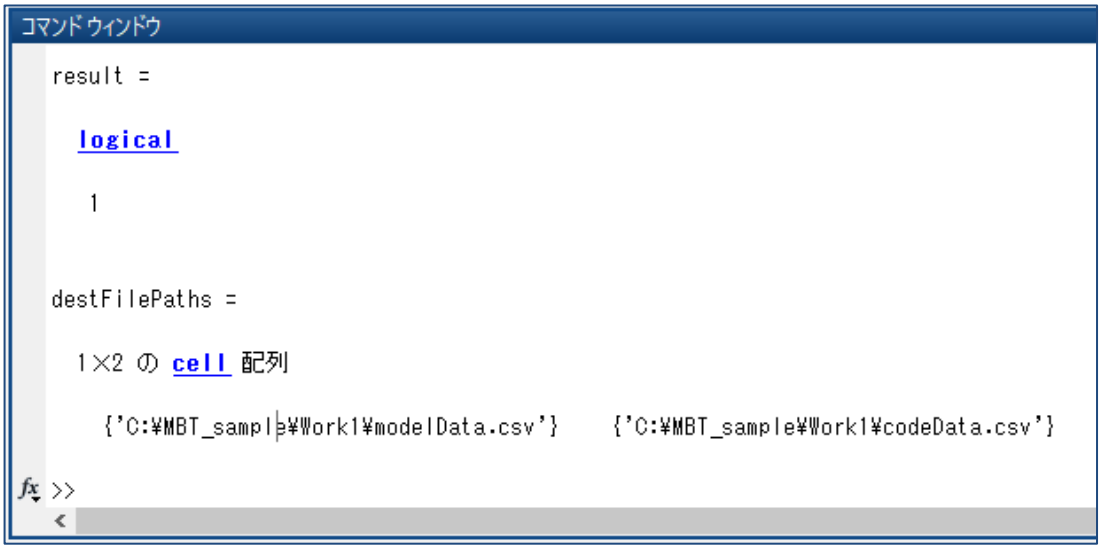


1.6. テストデータ変換実行

入力に指定されたテストデータを MILS 実行、PILS 実行で使用できる形式に変換します。
チュートリアル環境では、Reactis フォーマットのテスト CSV を用意しています。
コマンドは ConvertTestData を使用し、オプションに“入力テストデータ CSV”、“テストデータ CSV 種別”、“テスト対象関数名”、“マップリストファイル”、“ポインタリストファイル”、“出力ファイルパス”、“オプション”を指定します。

```
destFilePaths = { fullfile('Work1', 'modelData.csv'), fullfile('Work1', 'codeData.csv') };
Options.mapKind = 'BackwardMatch'

[result,destFilePaths] = MBTOP.ConvertTestData('Sample1/InputData.csv', 'Reactis',
'Subsystem_2010bpart_step', 'Work1/mapList.xml',
'Work1/mapList_p.xml',destFilePaths,Options)
```

正しく実行されると、Work1 フォルダにモデルテスト用ファイル「modelData.csv」、コードテスト用ファイル「codeData.csv」が作成されます。また、コマンドウィンドウの戻り値 result には 1 が返され、次のコマンドが入力できるようになります。

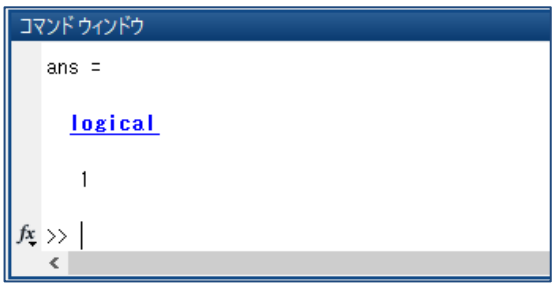


1.7. MILS 実行

テストハーネスモデルと、モデルテスト用 CSV ファイルのデータを使用して MILS 実行を行います。
コマンドは RunSimulation を使用し、オプションに“テスト対象モデル”、“マップリストファイル”、“モデルテスト用ファイル”、“モデル実行結果ログ出力パス”を指定します。

```
MBTOP.RunSimulation( 'Work1/TargetModel.mdl', 'Work1/mapList.xml',
'Work1/modelData.csv', 'Work1/mil')
```

正しく実行されると、Work1¥mil フォルダにモデル実行結果ログファイル「SimulationLog.mat」が作成されます。また、コマンドウィンドウには 1 という戻り値が返され、次のコマンドが入力できるようになります。

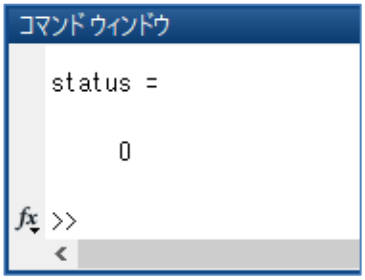


1.8. PILS 実行

カバレッジマスターwinAMS の機能をして使用して、PILS の実行を行います。
PILS 実行も system コマンドを使用します。system コマンドを使用する場合、各オプション間にはスペースを入れる必要がありますのでご注意ください。
PILS 実行は“AmsCommand.exe”＋“-MBTb”オプションにて実行し、オプションに“テスト CSV パス”、“出力ファイルパス”、“変数情報ファイルパス”、“プロジェクトファイルパス”を指定します。

```
status = system(['"C:¥winAMS¥BIN¥AmsCommand.exe"', ' -MBTb ',' -testCsv ',
' ..¥..¥Work1¥codeData.csv ', ' -output ', ' ..¥..¥Work1¥pil ', ' -xmlex ',
' ..¥..¥Work1¥varList.xml ',' Sample1¥AMS_MBT_DEMO¥AMS_MBT_DEMO.amsy'] )
```

正しく実行されると、Work1¥pil フォルダにコード実行結果ログファイル「codeData.csv」が作成されます。また、コマンドウィンドウには 0 という戻り値が返され、次のコマンドが入力できるようになります。
戻り値は、0 が正常終了、1 がエラー終了、260、261、262 は警告レベルとなります。



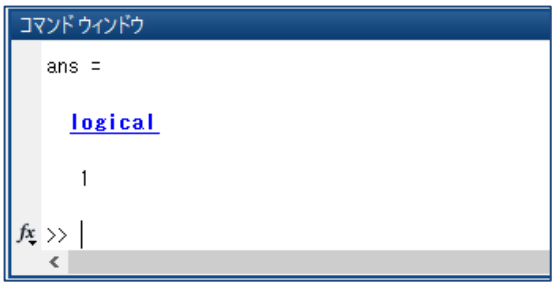
1.9. Back-to-Back テスト一致判定

モデル実行結果ログと、コード実行結果ログから、マッピングされた比較対象のシンボルの一致判定を行い、比較結果データを出力します。

コマンドは CompareLogData を使用し、オプションに“マップリストファイル”、“MIL 実行結果ログ”、“PIL 実行結果ログ”、“出力ファイルパス”を指定します。

```
MBTOP.CompareLogData('Work1/mapList.xml', 'Work1/mil/SimulationLog.mat',
'Work1/pil/codeData.csv', 'Work1/errorResult.mat')
```

正しく実行されると、Work1 フォルダに比較結果データ「errorResult.mat」が作成されます。また、コマンドウィンドウには 1 という戻り値が返され、次のコマンドが入力できるようになります。



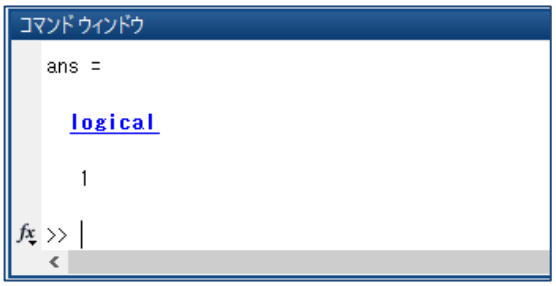
1.10. 統合レポート生成

比較結果データとモデル実行結果ログ、コード実行結果ログをまとめて、XML、HTML フォーマットでエビデンスを出力します。

コマンドは generateResult を使用し、オプションに “MIL 実行結果ログ”、“PIL 実行結果ログ”、“比較結果データ”、“オプション情報”を指定します。オプション情報（OptionStruct）は構造体で指定する必要があるため、事前に定義をしています。

```
OptionStruct.evidenceKind = 'html';
OptionStruct.mappingListPath = 'Work1/mapList.xml';
OptionStruct.codeInfoFilePath = 'Work1/varList.xml';
OptionStruct.evidenceReportFolderPath = 'Work1';

MBTOP.generateResult( 'Work1/mil/SimulationLog.mat', 'Work1/pil/codeData.csv',
'Work1/errorResult.mat', OptionStruct)
```

正しく実行されると、Work1 フォルダにエビデンスファイル「evidence.html」が作成されます。また、コマンドウィンドウには 1 という戻り値が返され、次のコマンドが入力できるようになります。



1.11. MBT オプション終了コマンド

MBT オプション終了時は MBT オプションライセンスを開放するため、MBT オプション終了コマンドを実行します。また、バックグラウンドでカバレッジマスターwinAMS が動作しているため、合わせて終了コマンドを実行します。

```
MBTOP.terminateMBTOP();

system(['"C:¥winAMS¥BIN¥AmsCommand.exe"', ' -endAMS']);
```

1.12. エビデンスの確認

テストが終了しましたので、レポートを確認します。レポートは Work1 フォルダ配下に作成されています。
evidence.html をダブルクリックするとレポートが開きます。

エビデンスの表示では、実行結果欄で Back-to-Back テストの一致判定結果を Passed、Failed で表示します。
また、実行環境、モデル/コード情報、設定情報などを確認することが出来ます。
コードカバレッジ欄では、選択したカバレッジの結果を確認できます。今回は PILS 実行を行っているため、オブジェクトベースでの C0,C1,MC/DC カバレッジの結果を確認することが出来ます。
テストケースリストでは、実行したテスト CSV 毎に合否判定結果を確認できます。このサンプル環境では1つのテストデータのみの実行のため、リストには1つの結果が表示されています。テスト CSV をクリックすると、テストケースの詳細結果を確認することが出来ます。

エビデンス

ターゲットモデル作成者： kobayashi.s
エクスポート日時： 2021/08/13 14:47:41

実行結果	
合否判定	Passed

実行環境	
OS	Microsoft Windows 10 Pro Version 10.0 (Build 19043)
MATLAB version	MATLAB 9.3 (R2017b)
MATLAB option	MC-Verifier 4.1.0.11
	PROMPT 2.1.0.0
	Simulink 9.0 (R2017b)
WinAMSバージョン	V7.0
CasePlayer2バージョン	V7.0
XAILバージョン	V3.7.2.0
マイコンシミュレータバージョン	sxgvt4.xdo V1.09.1
OMFコンバータバージョン	RH850CCRHomf.exe V1, 7, 0, 1
MBTOPバージョン	MBTOP 1.0.0

モデル情報		
モデル	モデルパス	C:\MBT_sample\MbtWork\OUT_1\TargetModel.mdl
	更新日時	2021/08/13 14:47:29
	パラメータMファイル	-
対象サブシステム	TargetModel/Subsystem_2010bpart	

コード設定		
WinAMS プロジェクトファイルパス	C:\MBT_sample\UserInput\AMS_MBT_DEMO\AMS_MBT_DEMO.amsy	
CasePlayer2 プロジェクトファイルパス	C:\MBT_sample\UserInput\CP2_MBT_DEMO\CP2_MBT_DEMO.vproj	
MPU名	RH850	
コンパイラ名	RENESAS (CCRH)	
OMF変換前のオブジェクトファイルパス	C:\MBT_sample\UserInput\CS_RH850\DefaultBuild\test_sample_RH850_prj.abs	
OMF変換後のオブジェクトファイルパス	カバレッジ測定用コードの埋め込み有り	C:\MBT_sample\UserInput\MCDC\CS_RH850\DefaultBuild\test_sample_RH850_prj.abs.xlo
	カバレッジ測定用コードの埋め込み無し	C:\MBT_sample\UserInput\CS_RH850\DefaultBuild\test_sample_RH850_prj.abs.xlo
OMFコンバート設定の変換オプション	-s "C:\MBT_DEMO\UserInput\CS_RH850\SRC"	
スタートアップファイルパス	カバレッジ測定用コードの埋め込み有り	C:\MBT_sample\UserInput\AMS_MBT_DEMO\SS_STARTUP.txt
	カバレッジ測定用コードの埋め込み無し	C:\MBT_sample\UserInput\AMS_MBT_DEMO\SS_STARTUP.txt
初期設定終了アドレス情報	main	

コード情報		
ターゲットコード	ターゲットコードファイルパス	C:\MBT_sample\UserInput\CS_RH850\SRC\Subsystem_2010bpart.c
	ターゲットコード更新日時	2017/03/21 14:58:16
関数	対象関数	Subsystem_2010bpart_step
マッチフラグ	1	

コードカバレッジ	
C0	100%
C1	100%
MC/DC	100%
関数コール	
カバレッジファイル	C:\MBT_sample\MbtWork\OUT_5\pil\TestCoverLog\Subsystem_2010bpart.c\Subsystem_2010bpart_s&0001.txt

テストケースリスト	
テストケース数	1
合否判定	**テストケース名**
Passed	codeData.csv

テストケースの表示では、実行したテスト CSV の合否判定、実行情報、ファイルごとのコードカバレッジ、信号データサマリーを確認できます。

信号データサマリーでは、マッピングされているブロック名と変数名ごとの合否判定を確認できます。合否判定が Failed となった場合は、Failed となっている箇所のブロック名を選択することで、実行時間ごとの結果を確認することが可能です。

テストケース

実行結果	
合否判定	Passed

実行日時	
モデル実行日時	2021/08/13 14:47:34
コード実行日時	2021/08/13 14:47:37
比較実行日時	2021/08/13 14:47:41

実行情報	
モデルテスト用CSVファイル	C:\MBT_sample\MbtWork\OUT_3\modelData.csv
コードテスト用CSVファイル	C:\MBT_sample\MbtWork\OUT_3\codeData.csv
モデル実行結果ログファイル	C:\MBT_sample\MbtWork\OUT_4\SimulationLog.mat
コード実行結果ログファイル	C:\MBT_sample\MbtWork\OUT_5\pil\codeData.csv

コードカバレッジ	
C0	100%
C1	100%
MC/DC	100%
関数コール	

信号データサマリー		
合否判定	ブロック名(モデル)	変数名(コード)
Passed	TargetModel/In1/1/1	rtU.In1
Passed	TargetModel/Out1/1/1	rtY.Out1

信号データの表示では、時間ごとにモデルデータ、コードデータの差分と誤差を確認することが出来ます。

信号データ

実行結果	
合否判定	Passed
最大誤差値	0

信号/変数情報		
信号情報	ブロックパス	TargetModel/Out1/1/1
	信号名	Out1
	データ型	double
変数情報	変数名	rtY.Out1
	データ型	double
固定小数点設定	LSB	
	オフセット	
	丸め設定	
許容誤差設定	計算方式	Local
	許容誤差	0
	丸め方式	Truncate
	桁指定方式	NumberOfSignificantFigures
	桁数	15
備考		

信号データ

時間	モデルデータ	コードデータ	差分	誤差
0	1	1	0	0
0.20000000000000001	3	3	0	0
0.40000000000000002	5.0998334170000001	5.0998334170000001	0	0
0.60000000000000009	7.2985027480000007	7.2985027480000007	0	0
0.80000000000000004	9.5940229549999998	9.5940229549999998	0	0
1	0	0	0	0
1.2000000000000002	2.4794255390000002	2.4794255390000002	0	0
1.4000000000000001	5.0440680120000003	5.0440680120000003	0	0
1.6000000000000001	7.6882856990000006	7.6882856990000006	0	0
1.8	0	0	0	0
2	2.78332691	2.78332691	0	0

1.13. 実習1まとめ

実習1では、MBT オプションを使用した Back-to-Back テストを行うための実行手順を、工程ごとに確認を行いました。チュートリアルで紹介出来ていないオプションも多くありますので、詳細は MBT オプションユーザーズマニュアルを参照してください。

最後に、実習1で実行したコマンドを以下にまとめます。

```
% MBT オプション開始
setenv('GAIO_CMW_STANDARD_DIGITS', '1');
MBTOP.startMBTOP();

% テストハーネスモデル作成
  MBTOP.createHarness('Sample1/TargetModel.mdl',
'TargetModel/Subsystem_2010bpart', 'Work1');

% 変数情報出力
status = system(['"C:¥winAMS¥BIN¥AmsCommand.exe"',' -MBToutVarDef','
Subsystem_2010bpart_step', ' -outFile',' ..¥..¥Work1¥varList.xml','
Sample1¥AMS_MBT_DEMO¥AMS_MBT_DEMO.amsy']);

% I/O マッピング
OptionSettings.mapKind = 'BackwardMatch'
OptionSettings.roundingMethod = 'Round'
OptionSettings.mapListFilePath = 'Work1/mapList.xml'
MBTOP.AutoMapping('Work1/blockInfo.xml', 'Work1/varList.xml', OptionSettings);

% テストデータ変換
destFilePaths = { fullfile('Work1', 'modelData.csv'), fullfile('Work1', 'codeData.csv') };
Options.mapKind = 'BackwardMatch'
[result,destFilePaths] = MBTOP.ConvertTestData('Sample1/InputData.csv', 'Reactis',
'Subsystem_2010bpart_step', 'Work1/mapList.xml',
'Work1/mapList_p.xml',destFilePaths,Options);

% MIL 実行
MBTOP.RunSimulation( 'Work1/TargetModel.mdl', 'Work1/mapList.xml',
'Work1/modelData.csv', 'Work1/mil', false );
```

```
% PIL 実行
status = system(['"C:¥winAMS¥BIN¥AmsCommand.exe"', ' -MBTb ',' -testCsv ',
' ..¥..¥Work1¥codeData.csv ', ' -output ', ' ..¥..¥Work1¥pil ', ' -xmlex ',
' ..¥..¥Work1¥varList.xml ',' Sample1¥AMS_MBT_DEMO¥AMS_MBT_DEMO.amsy']);

% Back-to-Back 一致判定
MBTOP.CompareLogData('Work1/mapList.xml', 'Work1/mil/SimulationLog.mat',
'Work1/pil/codeData.csv', 'Work1/errorResult.mat');

% 統合レポート生成
OptionStruct.evidenceKind = 'html';
OptionStruct.mappingListPath = 'Work1/mapList.xml';
OptionStruct.codeInfoFilePath = 'Work1/varList.xml';
OptionStruct.evidenceReportFolderPath = 'Work1';

MBTOP.generateResult( 'Work1/mil/SimulationLog.mat', 'Work1/pil/codeData.csv',
'Work1/errorResult.mat', OptionStruct);

% MBT オプション終了
MBTOP.terminateMBTOP();
system(['"C:¥winAMS¥BIN¥AmsCommand.exe"', ' -endAMS']);
```

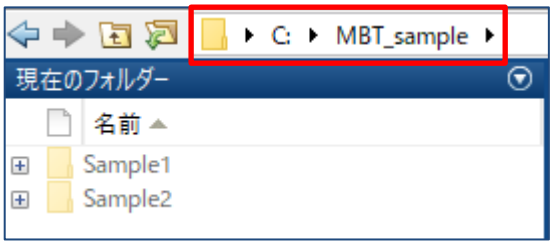
実習2:許容誤差の設定手順を学習

実習2では、許容誤差設定の使用方法を学習します。
MBT オプションの許容誤差設定は、求められる精度での判定を行うために使用します。

実習2の環境は、モデルとオブジェクトファイルで誤差が発生する環境を用意しています。
本環境では、精度が小数点以下 7 桁としたときの許容誤差を加味した検証を実施します。

2.1. 事前準備

MATLAB を立ち上げ、現在のフォルダに、展開したサンプル環境のフォルダ MBT_sample を選択します。



実習1同様、浮動小数点精度を統一するため、環境変数を設定するため、MATLAB コマンドウィンドウで以下のコマンドを実行します。

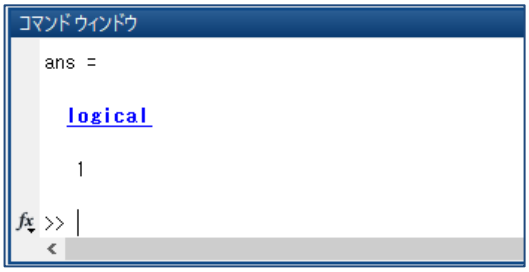
```
setenv('GAIO_CMW_STANDARD_DIGITS', '1');
```

2.2. MBT オプション起動コマンド実行

MBT オプションの各機能を使用するためには、MBT オプションライセンスの確認と取得を行う必要があります。
ライセンス取得は startMBTOP コマンドを使用します。
MATLAB のコマンドウィンドウで MBT オプションの起動コマンドを実行します。

```
MBTOP.startMBTOP()
```

正しく実行されると、コマンドウィンドウに 1 という戻り値が返され、次のコマンドが入力できるようになります。



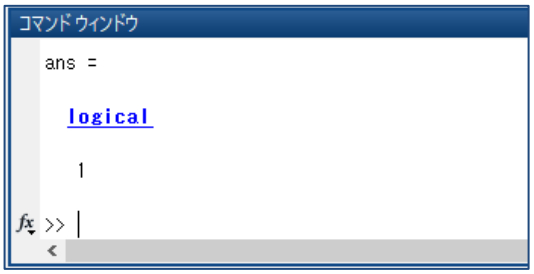
2.3. テストハーネスモデル生成実行

テストハーネスモデル生成機能では、ターゲットモデル内のテスト対象サブシステムを選択し、「テストハーネスモデル」と「テストハーネスモデル情報ファイル」を生成します。

コマンドは createHarness を使用し、ここで指定するオプションは“モデルファイルパス”、“検証対象サブシステム”、“出力先フォルダパス”を指定します。

```
MBTOP.createHarness('Sample2/TargetModel.mdl',
'TargetModel/Subsystem_sample2', 'Work2')
```

正しく実行されると、Work2 フォルダが新規に作成され、フォルダ内にハーネスモデル「TargetModel2.mdl」が生成されます。また、コマンドウィンドウには 1 という戻り値が返され、次のコマンドが入力できるようになります。



2.4. 変数情報出力

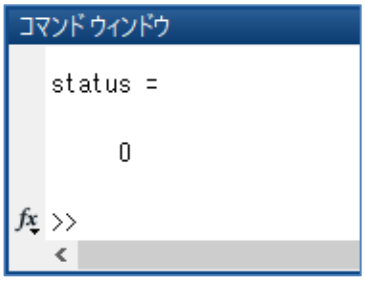
引数に指定した関数名に対して、カバレッジマスターwinAMS のプロジェクト情報から変数情報ファイルを作成します。

変数情報出力は“AmsCommand.exe”＋“-MBToutVarDef”オプションにて実行し、オプションに“テスト対象関数名”、“出力ファイルパス”、“プロジェクトファイルパス”を指定します。system コマンドを使用する場合、各オプション間にはスペースを入れる必要がありますのでご注意ください。

```
status = system(['"C:¥winAMS¥BIN¥AmsCommand.exe"',' -MBToutVarDef','
Subsystem_sample2', ' -outFile',' ..¥..¥Work2¥varList.xml','
Sample2¥AMS_MBT_DEMO¥AMS_MBT_DEMO.amsy'])
```

正しく実行されると、Work2 フォルダ内に変数情報ファイル「varList.xml」が生成されます。また、コマンドウィンドウには 0 という戻り値が返され、次のコマンドが入力できるようになります。

戻り値は、0 が正常終了、1、2 がエラー終了、258、260 は警告レベルとなります。



2.5. I/O マッピング実行

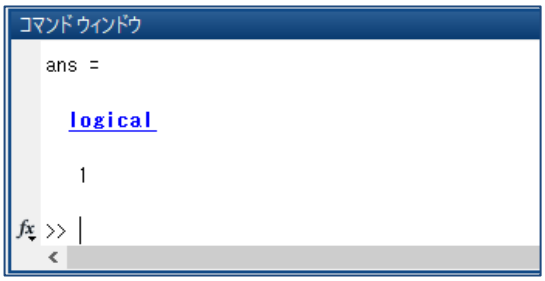
モデル情報と変数情報から、名前が一致する情報をマッチングし、「マップリスト」ファイルを作成します。

サンプルモデルのサブシステムの入力、出力に対し、コード側の変数名を紐づけます。ここでは名前一致でのマッピングの後方一致（BackwardMatch）を使用します。

コマンドは AutoMapping を使用し、オプションに“テストハーネスモデル情報”, “変数情報”, “オプション情報”を指定します。また、引数のオプション情報（OptionSettings）は構造体で指定する必要があるため、事前に定義をしています。

```
OptionSettings.mapKind = 'BackwardMatch'
OptionSettings.mapListFilePath = 'Work2/mapList.xml'
MBTOP.AutoMapping('Work2/blockInfo.xml', 'Work2/varList.xml', OptionSettings)
```

正しく実行されると、Work2 フォルダ内にマップリストファイル「mapList.xml」、ポインタリストファイル「mapList_p.xml」が生成されます。また、コマンドウィンドウには 1 という戻り値が返され、次のコマンドが入力できるようになります。

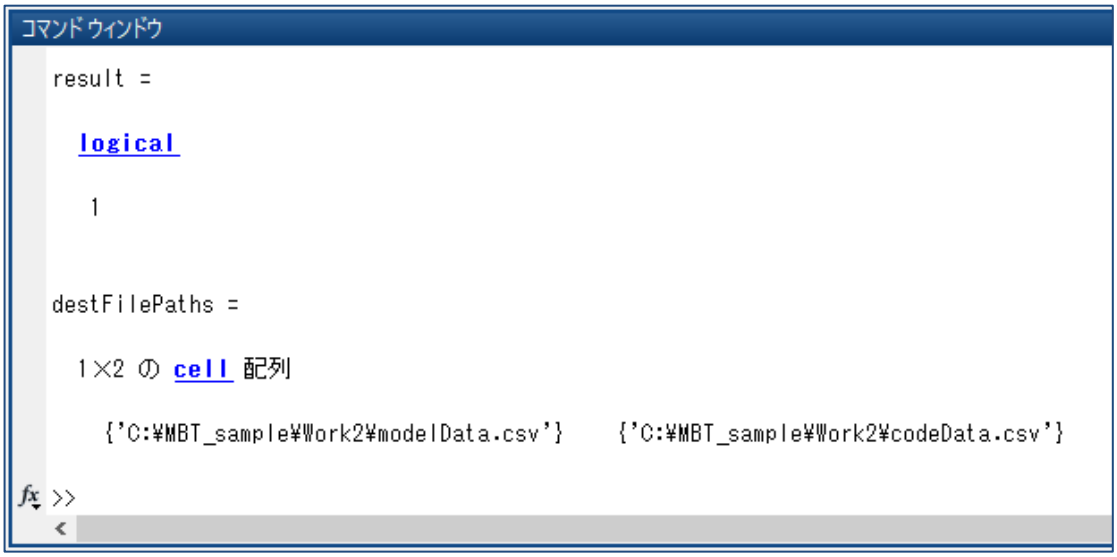


2.6. テストデータ変換実行

入力に指定されたテストデータを MILS 実行、PILS 実行で使用できる形式に変換します。
実習1では Reactis フォーマットのテスト CSV を使用しましたが、実習2では、カバレッジマスターwinAMS 用のテスト CSV を使用します。
コマンドは ConvertTestData を使用し、オプションに“入力テストデータ CSV”、“テストデータ CSV 種別”、“テスト対象関数名”、“マップリストファイル”、“ポインタリストファイル”、“出力ファイルパス”、“オプション”を指定します。

```
destFilePaths = { fullfile('Work2', 'modelData.csv'), fullfile('Work2', 'codeData.csv') };
Options.mapKind = 'BackwardMatch'

[result,destFilePaths] =
MBTOP.ConvertTestData( 'Sample2/cmwOUT/csv/Subsystem_sample2_data.csv',
'winAMS', 'Subsystem_sample2', 'Work2/mapList.xml', 'Work2/mapList_p.xml',
destFilePaths,Options)
```

正しく実行されると、Work2 フォルダにモデルテスト用ファイル「modelData.csv」、コードテスト用ファイル「codeData.csv」が作成されます。また、コマンドウィンドウの戻り値 result には 1 が返され、次のコマンドが入力できるようになります。

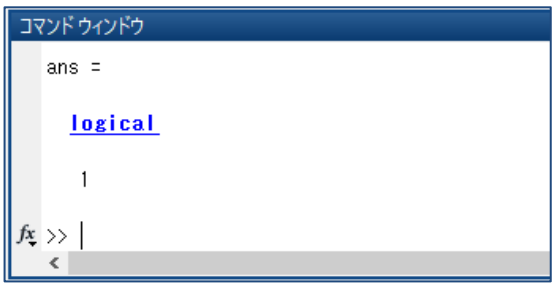


2.7. MILS 実行

テストハーネスモデルと、モデルテスト用 CSV ファイルのデータを使用して MILS 実行を行います。
コマンドは RunSimulation を使用し、オプションに“テスト対象モデル”、“マップリストファイル”、“モデルテスト用ファイル”、“モデル実行結果ログ出力パス”を指定します。

```
MBTOP.RunSimulation( 'Work2/TargetModel.mdl', 'Work2/mapList.xml',
'Work2/modelData.csv', 'Work2/mil')
```

正しく実行されると、Work2¥mil フォルダにモデル実行結果ログファイル「SimulationLog.mat」が作成されます。また、コマンドウィンドウには 1 という戻り値が返され、次のコマンドが入力できるようになります。

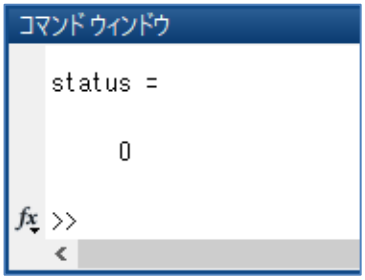


2.8. PILS 実行

カバレッジマスターwinAMS の機能をして使用して、PILS の実行を行います。
PILS 実行も system コマンドを使用します。system コマンドを使用する場合、各オプション間にはスペースを入れる必要がありますのでご注意ください。
PILS 実行は“AmsCommand.exe”＋“-MBTb”オプションにて実行し、オプションに“テスト CSV パス”、“出力ファイルパス”、“変数情報ファイルパス”、“プロジェクトファイルパス”を指定します。

```
status = system(['"C:¥winAMS¥BIN¥AmsCommand.exe"', ' -MBTb ',' -testCsv ',
' ..¥..¥Work2¥codeData.csv ', ' -output ', ' ..¥..¥Work2¥pil ', ' -xmlex ',
' ..¥..¥Work2¥varList.xml ',' Sample2¥AMS_MBT_DEMO¥AMS_MBT_DEMO.amsy'] )
```

正しく実行されると、Work2¥pil フォルダにコード実行結果ログファイル「codeData.csv」が作成されます。また、コマンドウィンドウには 0 という戻り値が返され、次のコマンドが入力できるようになります。
戻り値は、0 が正常終了、1 がエラー終了、260、261、262 は警告レベルとなります。



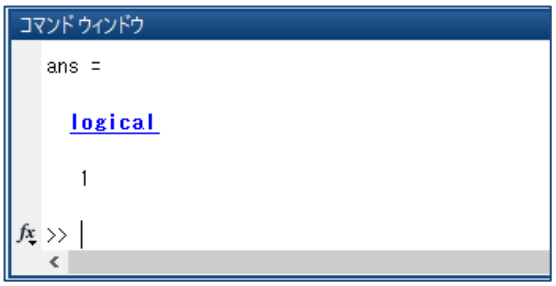
2.9. Back-to-Back テスト一致判定

モデル実行結果ログと、コード実行結果ログから、マッピングされた比較対象のシンボルの一致判定を行い、比較結果データを出力します。

コマンドは CompareLogData を使用し、オプションに“マップリストファイル”、“MIL 実行結果ログ”、“PIL 実行結果ログ”、“出力ファイルパス”を指定します。

```
MBTOP.CompareLogData('Work2/mapList.xml', 'Work2/mil/SimulationLog.mat',
'Work2/pil/codeData.csv', 'Work2/errorResult.mat')
```

正しく実行されると、Work2 フォルダに比較結果データ「errorResult.mat」が作成されます。また、コマンドウィンドウには 1 という戻り値が返され、次のコマンドが入力できるようになります。



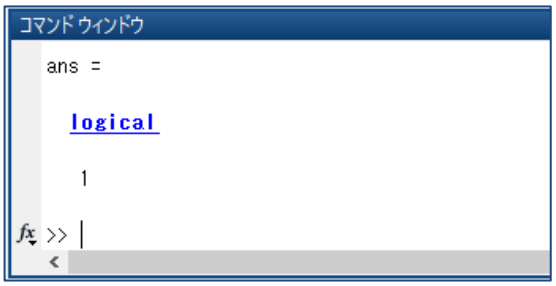
2.10. 統合レポート生成

比較結果データとモデル実行結果ログ、コード実行結果ログをまとめて、XML、HTML フォーマットでエビデンスを出力します。

コマンドは generateResult を使用し、オプションに “MIL 実行結果ログ”、“PIL 実行結果ログ”、“比較結果データ”、“オプション情報”を指定します。オプション情報（OptionStruct）は構造体で指定する必要があるため、事前に定義をしています。

```
OptionStruct.evidenceKind = 'html';
OptionStruct.mappingListPath = 'Work2/mapList.xml';
OptionStruct.codeInfoFilePath = 'Work2/varList.xml';
OptionStruct.evidenceReportFolderPath = 'Work2';

MBTOP.generateResult( 'Work2/mil/SimulationLog.mat', 'Work2/pil/codeData.csv',
'Work2/errorResult.mat', OptionStruct)
```

正しく実行されると、Work2 フォルダにエビデンスファイル「evidence.html」が作成されます。また、コマンドウィンドウには 1 という戻り値が返され、次のコマンドが入力できるようになります。



2.11. 不一致の確認

テストが終了しましたので、レポートを確認します。Work2 フォルダ配下に生成された evidence.html をダブルクリックするとレポートが開きます。

実習2での一致判定結果は Failed となっています。テストケースリストを選択し、Failed となったブロック名を選択すると、信号データサマリーにて、差分が発生している箇所を確認することが出来ます。

エビデンス

ターゲットモデル作成者： kobayashi.s
エクスポート日時： 2021/09/17 11:12:29

実行結果

合否判定	Failed

テストケースリスト

テストケース数	1
合否判定	**テストケース名**
Failed	codeData.csv

信号データサマリー

合否判定	ブロック名(モデル)	変数名(コード)
Passed	TargetModel/In1/1/1	rtU.In1
Passed	TargetModel/In2/1/1	rtU.In2
Failed	TargetModel/Out1/1/1	rtY.Out1

信号データ

実行結果

合否判定	Failed
最大誤差値	1.749243564155835E-14

信号データ

時間	モデルデータ	コードデータ	差分	誤差
0	0	0	0	0
0.200000000000001	0.029999999999999999	0.029999999329447746	6.7055225261292151E-10	2.993535711353991E-08
0.400000000000002	0.080000000000000002	0.079999996349215508	3.6507844941580103E-09	1.6298147355518567E-07
0.600000000000009	0.14999999999999999	0.14999999664723873	3.3527612630646075E-09	1.4967682971021657E-07
0.800000000000000	0.23999999999999999	0.23999999277293682	7.2270631701343	3.2263674123218

結果から、赤枠の時間で差分が許容すべき誤差を超えていることが分かります。
今回の実行は、小数点以下 7 桁以上の誤差を許容する設定せずに実行しています。
許容誤差の設定をしなかった場合（省略時）は、以下の設定となります。

OptionSettings 構造体フィールド許容誤差省略時の設定

引数	省略時の設定	省略時の設定概要
errorMethod	Local	許容誤差計算方式： Local 許容誤差計算の結果、許容誤差の範囲を超えた場合は Failed となる 計算方式の詳細は、ユーザーズマニュアルを参照
toleranceValue	0	許容誤差の範囲： 0
roundingMethod	Truncate	丸め方式： 切り捨て
digitDesignationMethod	NumberOfSignificantFigures	桁数指定方式： 小数点以下
numberOfSignificantFigures	15	桁数： 15 桁

各設定の詳細は「MBT オプションユーザーズマニュアル」にある I/O マッピング機能の MBTOP.AutoMapping 省略時の設定を参照してください。

許容誤差設定省略時は、小数点有効桁 15 桁となっているため、小数点以下 7 桁とする設定を行い、期待した合否判定を行えるようコマンドを変更し実行します。

2.12. 許容誤差設定

許容誤差設定はI/O マッピングコマンドにて実行します。
ここでは、小数点以下7桁とするよう設定します。また、丸め誤差の方式は四捨五入とします。未設定の許容誤差設定は省略時の設定を適用します。

再実行の際、オプションに“上書き許可フラグ”の追加、オプション情報（OptionSettings）へ“許容誤差設定”を追加しています。また、今回はコマンドの末端にセミコロン（;）を追加し、戻り値を非表示としていますので、戻り値やエラーメッセージを確認する場合は、セミコロンを除いて実行してください。

```
% マッピング前に変数情報を更新
status = system(['"C:¥winAMS¥BIN¥AmsCommand.exe"',' -MBToutVarDef','
Subsystem_sample2', ' -outFile',' ..¥..¥Work2¥varList.xml','
Sample2¥AMS_MBT_DEMO¥AMS_MBT_DEMO.amsy']);

% オプション設定
OptionSettings.mapKind = 'BackwardMatch'
OptionSettings.mapListFilePath = 'Work2/mapList.xml'

% 許容誤差設定 追加オプション
%  - 丸め誤差を四捨五入とする設定
OptionSettings.roundingMethod = 'Round'
%  - 桁数指定方式を小数点以下とする設定
OptionSettings.digitDesignationMethod = 'NumberOfSignificantFigures'
%  - 桁数 7 桁とする設定
OptionSettings. numberOfSignificantFigures = 7

% I/O マッピング実行
MBTOP.AutoMapping('Work2/blockInfo.xml', 'Work2/varList.xml', OptionSettings,
true);
```

2.13. Back-to-Back テスト一致判定、レポート出力の再実行

許容誤差設定後は、MILS/PILS の再実行、Back-to-Back テスト一致判定、レポート出力を行います。
変更点は、各コマンドに上書き許可フラグを追加しています。また、「2.12.許容誤差設定」と同様、コマンドの末端にセミコロン(;)を追加し、戻り値を非表示としていますので、戻り値やエラーメッセージを確認する場合は、セミコロンを除いて実行してください。

```
% MILS 再実行
MBTOP.RunSimulation( 'Work2/TargetModel.mdl', 'Work2/mapList.xml',
'Work2/modelData.csv', 'Work2/mil', true);

% PILS 再実行
status = system(['"C:¥winAMS¥BIN¥AmsCommand.exe"', ' -MBTb ',' -testCsv ',
' ..¥..¥Work2¥codeData.csv ', ' -output ', ' ..¥..¥Work2¥pil ', ' -xmlex ',
' ..¥..¥Work2¥varList.xml ',' Sample2¥AMS_MBT_DEMO¥AMS_MBT_DEMO.amsy'] );

% Back-to-Back テスト一致判定
MBTOP.CompareLogData('Work2/mapList.xml', 'Work2/mil/SimulationLog.mat',
'Work2/pil/codeData.csv', 'Work2/errorResult.mat', true);

% レポート出力
MBTOP.generateResult( 'Work2/mil/SimulationLog.mat', 'Work2/pil/codeData.csv',
'Work2/errorResult.mat', OptionStruct, true);
```

実行後、Work2 フォルダに比較結果データ「errorResult.mat」が上書きされます。

2.14. MBT オプション終了コマンド

MBT オプション終了時はMBTオプションライセンスを開放するため、MBT オプション終了コマンドを実行します。
また、バックグラウンドでカバレッジマスターwinAMS が動作しているため、合わせて終了コマンドを実行します。

```
MBTOP.terminateMBTOP();

system(['"C:¥winAMS¥BIN¥AmsCommand.exe"', ' -endAMS']);
```

2.15. エビデンスの確認

再テストが終了しましたので、結果を確認します。
Work2¥evidence.html を開くと、合否判定が「Passed」となっていることが確認できます。

続いて、テスト CSV を選択し、Out の信号データを確認します。実行結果による差分が許容誤差設定の範囲内となり、Failed となっていた行の誤差が0となっていることが確認できます。

信号データ

時間	モデルデータ	コードデータ	差分	誤差
0	0	0	0	0
0.20000000000000001	0.029999999999999999	0.029999999329447746	6.7055225261292151E-10	0
0.40000000000000002	0.080000000000000002	0.079999996349215508	3.6507844941580103E-09	0
0.60000000000000009	0.14999999999999999	0.14999999664723873	3.3527612630646075E-09	0

今回のサンプル環境では、プロジェクト全体に影響する許容誤差設定を行いました。本設定とは別に、信号ごとに許容誤差設定を行うことも可能となっています。

2.16. 実習2まとめ

実習2では、許容誤差設定の手順と、許容誤差設定による結果の違いを確認しました。

最後に、実習2で実行したコマンドを以下にまとめます。
ここでは、事前に許容誤差設定を行った流れを記載しています。

```
% MBT オプション開始
setenv('GAIO_CMW_STANDARD_DIGITS', '0');
MBTOP.startMBTOP();

% テストハーネスモデル作成
  MBTOP.createHarness('Sample2/TargetModel.mdl',
'TargetModel/Subsystem_sample2', 'Work2');

% 変数情報出力
status = system(['"C:¥winAMS¥BIN¥AmsCommand.exe"',' -MBToutVarDef','
Subsystem_sample2', ' -outFile',' ..¥..¥Work2¥varList.xml','
Sample2¥AMS_MBT_DEMO¥AMS_MBT_DEMO.amsy']);

% I/O マッピング
OptionSettings.mapKind = 'BackwardMatch'
OptionSettings.mapListFilePath = 'Work2/mapList.xml'
OptionSettings.errorMethod = 'Local'
OptionSettings.roundingMethod = 'Round'
OptionSettings.digitDesignationMethod = 'NumberOfSignificantFigures'
OptionSettings. numberOfSignificantFigures = 7
MBTOP.AutoMapping('Work2/blockInfo.xml', 'Work2/varList.xml', OptionSettings,
true);

% テストデータ変換
destFilePaths = { fullfile('Work2', 'modelData.csv'), fullfile('Work2', 'codeData.csv') };
Options.mapKind = 'BackwardMatch'
[result,destFilePaths] =
MBTOP.ConvertTestData( 'Sample2/cmwOUT/csv/Subsystem_sample2_data.csv',
'winAMS', 'Subsystem_sample2', 'Work2/mapList.xml', 'Work2/mapList_p.xml',
destFilePaths,Options)

% MIL 実行
```

```
MBTOP.RunSimulation( 'Work2/TargetModel.mdl', 'Work2/mapList.xml',
'Work2/modelData.csv', 'Work2/mil', true);

% PIL 実行
status = system(['"C:¥winAMS¥BIN¥AmsCommand.exe"', ' -MBTb ',' -testCsv ',
' ..¥..¥Work2¥codeData.csv ', ' -output ', ' ..¥..¥Work2¥pil ', ' -xmlex ',
' ..¥..¥Work2¥varList.xml ',' Sample2¥AMS_MBT_DEMO¥AMS_MBT_DEMO.amsy'] );

% Back-to-Back 一致判定
MBTOP.CompareLogData('Work2/mapList.xml', 'Work2/mil/SimulationLog.mat',
'Work2/pil/codeData.csv', 'Work2/errorResult.mat', true);

% 統合レポート生成
OptionStruct.evidenceKind = 'html';
OptionStruct.mappingListPath = 'Work2/mapList.xml';
OptionStruct.codeInfoFilePath = 'Work2/varList.xml';
OptionStruct.evidenceReportFolderPath = 'Work2';

MBTOP.generateResult( 'Work2/mil/SimulationLog.mat', 'Work2/pil/codeData.csv',
'Work2/errorResult.mat', OptionStruct, true);

% MBT オプション終了
MBTOP.terminateMBTOP();
system(['"C:¥winAMS¥BIN¥AmsCommand.exe"', ' -endAMS']);
```

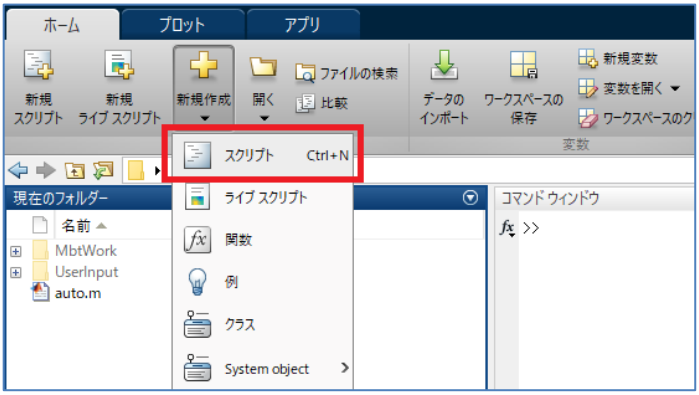
実習3：自動実行環境構築の学習（mファイルによる自動実行）

実習3では、より効率的なmファイルによる実行環境を紹介します。
実習1、実習2では、カバレッジマスターwinAMS、CasePlayer2 のプロジェクト環境は事前に作成していることを前提に実行をしました。実習3では、CLIコマンドを使用して、カバレッジマスターwinAMS、CasePlayer2 のプロジェクトも合わせて作成するmファイルを紹介します。

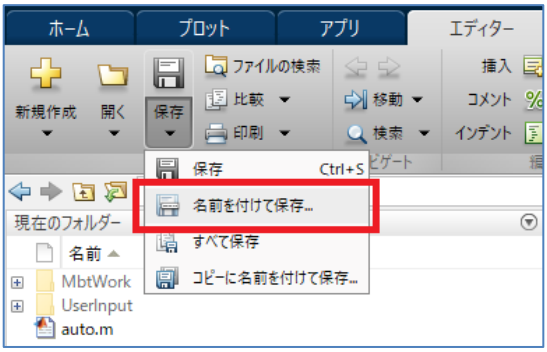
本チュートリアルではオートコード環境、コンパイル環境を提供していないため、MC/DC カバレッジ計測用のオブジェクトは生成しません。実行する内容は CasePlayer2 解析、カバレッジマスターwinAMS の設定、MBT オプションの実行をmファイルにて、まとめて実行する内容を紹介します。
また、実行に必要なモデル、Cコード、オブジェクトファイル、テストCSVは実習1で使用したファイルを使用します。

3.1. mファイル作成

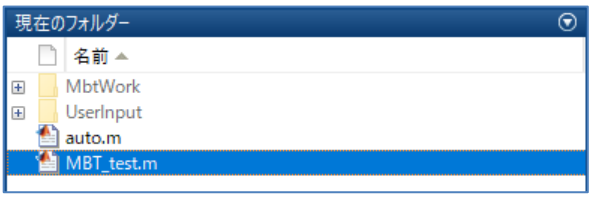
以下の手順でスクリプトを新規に作成します。



作成したスクリプトに名前を付けて保存します。
スクリプトのファイル名は任意に設定します。ここでは、ファイル名を「MBT_test.m」とします。



保存後、フォルダに「MBT_test.m」が作成されています。



作成した MBT_test.m に、以下のコマンドを記載します。

```
%% 初期設定
% 実習３用環境へのコピー（実習３用必須ファイル）
mkdir 'Sample3';
copyfile 'Sample1¥CS_RH850' 'Sample3¥CS_RH850';
copyfile 'Sample1¥InputData.csv' 'Sample3';
copyfile 'Sample1¥TargetModel.mdl' 'Sample3';
mkdir 'Sample3¥CMW';
copyfile 'Sample1¥AMS_MBT_DEMO¥AmsProjSave.amso' 'Sample3¥CMW';
copyfile 'Sample1¥AMS_MBT_DEMO¥SS_STARTUP.txt'    'Sample3¥CMW';

% フォルダの設定
baseDir = fileparts(mfilename('fullpath'));
% ユーザーが準備するファイルが収められたフォルダ（入力ファイル）
userDir = fullfile(baseDir, 'Sample3');
% MBTオプションを実行して出力するファイルを集めるフォルダ
workspaceDir = fullfile(baseDir, 'Work3');
try rmdir(workspaceDir, 's'); catch, end
mkdir(workspaceDir);



try
%% CasePlayer2とカバレッジマスターwinAMSの設定
% CasePlayer2パス設定（インストール先により変更する）
exeCmdCp2 = '"C:¥Program Files
(x86)¥GAIO¥CasePlayer2¥bin¥CaseCommand.exe"';
% プロジェクト出力フォルダ
cp2PrjDir = fullfile(userDir, 'CP2' );
% プロジェクトファイル
cp2Prj = fullfile(cp2PrjDir, 'CP2.vproj' );
% テンプレートファイル
cp2Template = '"C:¥Program Files (x86)¥gaio¥CasePlayer2¥template"';
```

```
% ソースファイルパス
srcPath = fullfile(userDir, 'CS_RH850', 'SRC', 'Subsystem_2010bpart.c');
% システムインクルードパス
incPath1 = 'C:¥MBT_sample¥Sample3¥CS_RH850¥Inc850';

% カバレッジマスターwinAMSパス設定（インストール先により変更する）
exeCmdCmw = '"C:¥winAMS¥BIN¥AmsCommand.exe"';
% プロジェクト出力フォルダ
cmwPrjDir = fullfile(userDir, 'CMW' );
% プロジェクトファイル
cmwPrj = fullfile(cmwPrjDir, 'CMW.amsy' );
% テンプレートファイル
cmwTemplate = fullfile(userDir, 'CMW', 'AmsProjSave.amso' );
% オブジェクトファイルパス
obj_path = fullfile(userDir, 'Sample3', 'CS_RH850', 'DefaultBuild',
'test_sample_RH850_prj.abs' );
% 初期化終了アドレス
init_endaddr='"main"';
% スタートアップコマンドファイル
ss_startup = fullfile(userDir, 'CMW', 'SS_STARTUP.txt' );

% MPU名
cpu_name = '"RH850"';
type_name = '"CCRH"';

%% 環境変数の設定
setenv('GAIO_CMW_STANDARD_DIGITS', '1');



%% CasePlayer2実行
% プロジェクトの新規作成
status = system([ exeCmdCp2, ' -crtprojimport ', cp2Template, ' ', cp2PrjDir] );
```

```
if status ~= 0
error('crtprojimport');
end
system([ exeCmdCp2, ' -saveProj ', cp2Prj] );

% MPU及びコンパイラの設定変更
status = system([ exeCmdCp2, ' -cpucpl ', cpu_name, ' ', type_name, ' ', cp2Prj] );
if status ~= 0
error('cpucpl');
end
system([ exeCmdCp2, ' -saveProj ', cp2Prj] );

% 設定変更の一部([REV])をコマンドラインで直接設定
status = system([ exeCmdCp2, ' -set_rev ', 'MBT_MODE=1;MCDC_CONST_EXPR=1
', cp2Prj] );
if status ~= 0
error('set_rev');
end
system([ exeCmdCp2, ' -saveProj ', cp2Prj] );

% ソース追加
src_set_path = fullfile(workspaceDir, 'source.set' );
fileID = fopen(src_set_path, 'w');
if fileID~= -1
fprintf(fileID, '"%s"', srcPath);
fclose(fileID);
else
error('source.set');
end

status = system([ exeCmdCp2, ' -a ', src_set_path, ' ', cp2Prj] );
if status ~= 0
```

```
error('a');
end
system([ exeCmdCp2, ' -saveProj ', cp2Prj] );

% システムインクルードパス追加
inc_set_path = fullfile(workspaceDir, 'include.set' );
fileID = fopen(inc_set_path, 'w');
if fileID~= -1
fprintf(fileID, '"%s"', incPath1);
fclose(fileID);
else
error('include.set');
end
status = system([ exeCmdCp2, ' -asi ', inc_set_path, ' ', cp2Prj] );
if status ~= 0
error('asi');
end

% 仕様書生成
status = system([ exeCmdCp2, ' -r ', cp2Prj] );
if status ~= 0
error('r');
end



%% カバレッジマスターwinAMS実行
% プロジェクトの新規作成
status = system([ exeCmdCmw, ' -crtprojimport ', cmwTemplate, ' ', cmwPrjDir] );
if status ~= 0
error('crtprojimport');
end
system([ exeCmdCmw, ' -saveProj ', cmwPrj] );
```

```
% MPU及びコンパイラの設定変更
status = system([ exeCmdCmw, ' -mpucpl ', cpu_name, ' ', type_name, ' ',
cmwPrj] );
if status ~= 0
error('mpucpl');
end
system([ exeCmdCmw, ' -saveProj ', cmwPrj] );

% 設定変更(CasePlayer2連携)
system_gCmd = ['Cp2Proj=', cp2Prj];
status = system([ exeCmdCmw, ' -set_system_g ', system_gCmd, ' ', cmwPrj] );
if status ~= 0
error('set_system_g CP2');
end
system([ exeCmdCmw, ' -saveProj ', cmwPrj] );

% 設定変更(ターゲットオブジェクトファイルの設定)
status = system([ exeCmdCmw, ' -obj ', obj_path, ' ', cmwPrj] );
if status ~= 0
error('obj');
end
system([ exeCmdCmw, ' -saveProj ', cmwPrj] );

% 設定変更(初期設定終了アドレス(ターゲットオブジェクト))
status = system([ exeCmdCmw, ' -iendaddr ', init_endaddr, ' ', cmwPrj] );
if status ~= 0
error('iendaddr');
end
system([ exeCmdCmw, ' -saveProj ', cmwPrj] );
```

```
% 設定変更(スタートアップコマンドファイル))
system_gCmd2 = ['Start=', ss_startup];
status = system([ exeCmdCmw, ' -set_system_g ', system_gCmd2, ' ', cmwPrj] );
if status ~= 0
error('set_system_g startup');
end
system([ exeCmdCmw, ' -saveProj ', cmwPrj] );

% 仕様書生成
status = system([ exeCmdCp2, ' -r ', cp2Prj] );
if status ~= 0
error('r');
end

catch ME
disp(getReport(ME));
end



try
%% MBTオプションコマンド
% 開始処理
[status, ME, lastErrorCode] = MBTOP.startMBTOP();
if status ~= true
error('startMBTOP');
end
% ハーネスモデル作成
modelFilePath = fullfile(userDir, 'TargetModel.mdl' );
targetSubsystem = 'TargetModel/Subsystem_2010bpart';
destFolderPath = fullfile(workspaceDir, 'OP1' );
%Options.PreLoadMfile = fullfile(userDir, 'settingM.m' );
overWrite = false;
```

```
[result, harnessModelFilePath, harnessModelInfoFilePath] = MBTOP.createHarness(...
    modelFilePath, targetSubsystem, destFolderPath);
if result ~= true
error('createHarness');
end

% 変数情報作成
functionNames = 'Subsystem_2010bpart_step';
varInfoFilePath = fullfile(workspaceDir, 'varList.xml' );
status = system([ exeCmdCmw, ' -MBToutVarDef ', functionNames, ' -outFile ',
varInfoFilePath,' ', cmwPrj] );
if status ~= 0
error('MBToutVarDef');
end

% I/Oマッピング
modelInfoFilePath = fullfile(workspaceDir, 'OP1', 'blockInfo.xml' );
OptionSettings.mapKind = 'BackwardMatch';
OptionSettings.mapListFilePath = fullfile(workspaceDir, 'OP2', 'mapList.xml' );
overWrite = false;
[ result, varMapStatus, blockMapStatus ] = MBTOP.AutoMapping(...
modelInfoFilePath, varInfoFilePath, OptionSettings, overWrite);
if result ~= true
error('AutoMapping');
end

% テストデータAD/DA変換
atgTestDataCsvPath = fullfile(userDir, 'InputData.csv' );
atgKind = 'Reactis';
mappingListPath = fullfile(workspaceDir, 'OP2', 'mapList.xml' );
pointerListPath = fullfile(workspaceDir, 'OP2', 'mapList_p.xml' );
destFilePaths = { fullfile(workspaceDir, 'OP34', 'modelData.csv' ),
fullfile(workspaceDir, 'OP34', 'codeData.csv' ) };
```

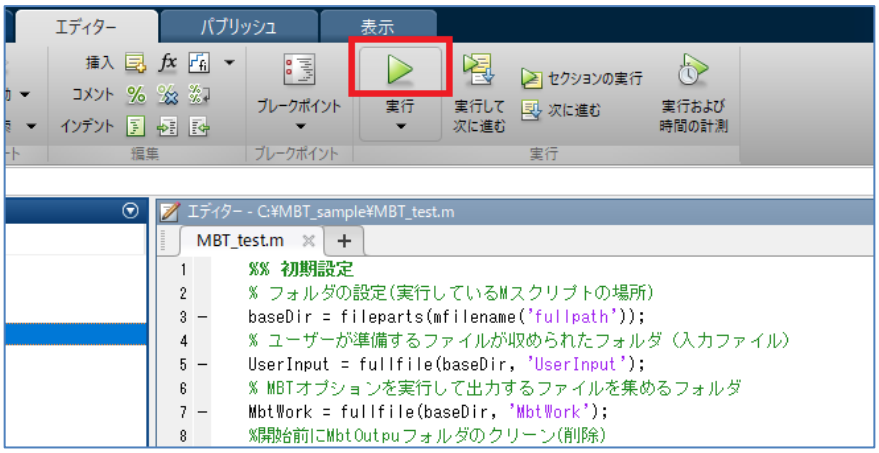
```
Options.mapKind = 'BackwardMatch'
overWrite = false;
[result, destFilePaths] = MBTOP.ConvertTestData(...
atgTestDataCsvPath, atgKind, functionNames, mappingListPath,
pointerListPath,destFilePaths, Options, overWrite );
if result ~= true
error('ConvertTestData');
end

% MILS実行
harnessModelFilePath = fullfile(workspaceDir, 'OP1', 'TargetModel.mdl' );
testDataFilePath = destFilePaths{1};
logDirPath = fullfile(workspaceDir, 'OP5' );
overWrite = false;
[result, logFilePath] = MBTOP.RunSimulation(...
  harnessModelFilePath, mappingListPath, testDataFilePath, logDirPath, overWrite );
if result ~= true
error('RunSimulation');
end

% PILS実行
csvFilePath = destFilePaths{2};
outDir = 'OP5Code';
mbtInfo = fullfile(workspaceDir, 'OP5Code', 'mbtinfo.xml' );
status = system([ exeCmdCmw, ' -MBTb -xmlex ', mbtInfo, ' -testCsv ', csvFilePath, '
-output ', outDir, ' ', cmwPrj] )
if status ~= 0
error('MBTb');
end

% BtoBテスト一致判定
modelLogFilePath = fullfile(workspaceDir, 'OP5', 'SimulationLog.mat' );
[~,name,ext] = fileparts(csvFilePath);
```

```
codeLogFilePath = fullfile(cmwPrjDir, outDir, [name, ext] );
errorResultFilePath = fullfile(workspaceDir, 'OP6', 'errorResult.mat' );
overWrite = false;
[ result, overallResult ] = MBTOP.CompareLogData(...
  mappingListPath, modelLogFilePath, codeLogFilePath, errorResultFilePath,
overWrite );
if result ~= true
error('CompareLogData');
end


% 総合レポート生成
modelLogFilePath = {modelLogFilePath};
codeLogFilePath = {codeLogFilePath};
compareResultDataFilePath = {errorResultFilePath};
OptionStruct.evidenceKind = 'html';
OptionStruct.mappingListPath = mappingListPath;
OptionStruct.codeInfoFilePath = mbtInfo;
OptionStruct.evidenceReportFolderPath = fullfile(workspaceDir, 'OP7' );
overWrite = false;
[ result ] = MBTOP.generateResult(...
  modelLogFilePath, codeLogFilePath, compareResultDataFilePath, OptionStruct,
overWrite);
if result ~= true
error('generateResult');
end

catch ME
disp(getReport(ME));
end
```

```
%% 終了処理
% MBTオプション終了
MBTOP.terminateMBTOP();
% CasePlayer2終了
system([ exeCmdCp2, ' -endCP2'] );
% カバレッジマスターwinAMS終了
system([ exeCmdCmw, ' -endAMS'] );


%EOF
```

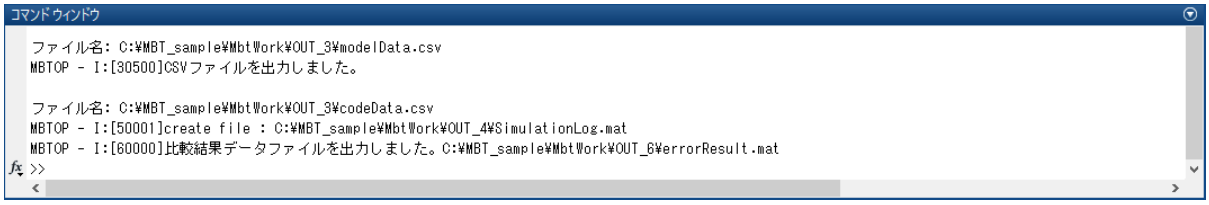
3.2. mファイル実行

作成したmファイルを MATLAB コマンド上で実行します。



MBT オプションが異常終了した時、MBT オプション関連モジュールが起動したままになり、エラー状態が解消しない場合があります。このような場合は MATLAB を再起動してください。MATLAB を終了するとライセンスを開放します。

コマンドウィンドウに「比較結果データファイルを出力しました。」と表示され、操作がアクティブとなると Back-to-Back テストは終了です。エビデンスファイルは「Work3¥OP7」フォルダ内に作成され、確認方法は実習1と同様です。



3.3. 実習3まとめ

実習3では、mファイルを使用し、カバレッジマスターwinAMS、CasePlayer2 のプロジェクト作成から MBT オプション実行までの流れを確認しました。

本サンプル環境をカスタマイズ頂き、ご活用ください。

カバレッジマスターwinAMS
MBTオプション チュートリアル

ガイオ・テクノロジー株式会社

■ユーザーサポートのご案内
http://www.gaio.co.jp/support/support_about.html

■使用方法に関するお問い合わせ方法
お問い合わせは、ユーザーサポート窓口をご利用ください。
http://www.gaio.co.jp/support/support_entry.html
ユーザーサポート窓口へのお問い合わせには、ユーザーIDが必要です。
※保守契約がない場合は、いかなるサポートも提供致しません。