



チュートリアル Vertical Tracker Part #3

< 目的 > TTM に強化された拡張機能を紹介。

- Structures (構造体) サポート
- Functions テーブル
 - パラメータで定義する振舞い (入力などに対してではなく)
- INTERM 変数 (テンポラリなローカル変数)
 - Function テーブルで活用
 - 例: `INTERM_x = function(a, b, c)`

演習の目的 :

- 構造体を含むインタフェースの拡張
- nearest altitude 用に新しい function を作成
- vertical_tracker2 のオブジェクトマップファイルを拡張
- C:\TVEC_Tutorial\exercises\test_driver_utilities にある共通スキーマ、オブジェクトマッピング等を使用
- テストドライバジェネレータで C 言語テストドライバを生成
- Vertical_tracker3 用ソースコードに対して、テストドライバを実行
- 期待値と結果出力を比較するテスト結果解析プログラムを実行

< 設定 >

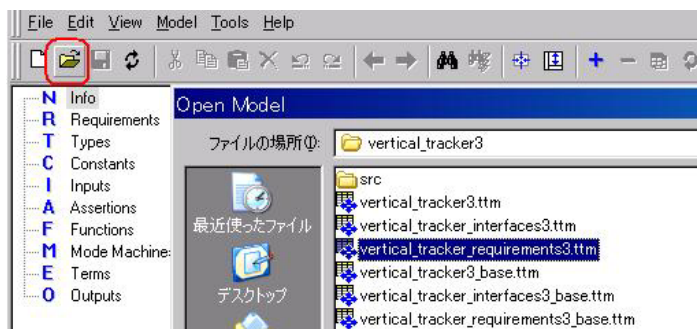
1. ファイルをコピーします。(バックアップ)

* C:\TVEC\examples\ttm\course\exercises\vertical_tracker3 内の ttm ファイルを C:\TVEC_Tutorial\exercises\vertical_tracker3 (新設) へコピーします。

- vertical_tracker_requirements3.ttm を vertical_tracker_requirements3_base.ttm へコピーします。
- vertical_tracker_interfaces3.ttm を vertical_tracker_interfaces3_base.ttm へコピーします。
- vertical_tracker3.ttm を vertical_tracker3_base.ttm へコピーします。

< 要求仕様 >

2. TTM で、**vertical_tracker_requirements3.ttm** を開きます。

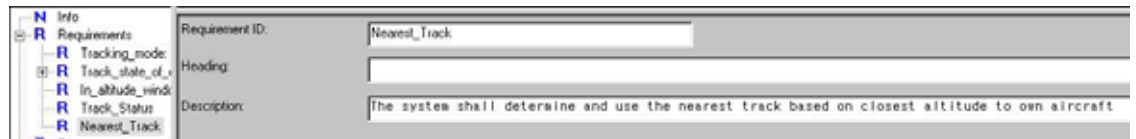


3. Requirements に以下を追加します。

RequirementID と Description 部分に、それぞれ追加します。

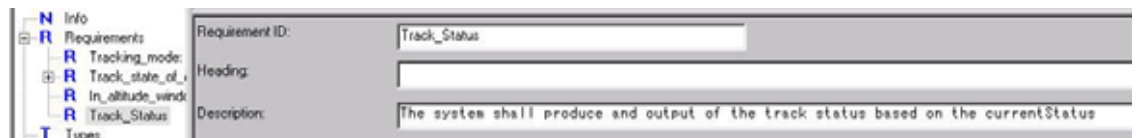
- **Nearest_Track**

The system shall determine and use the nearest track based on closest altitude to own aircraft (Description 欄と Text 欄は日本語対応)



- **Track_Status**

The system shall produce and output of the track status based on the currentStatus (Description 欄と Text 欄は日本語対応)



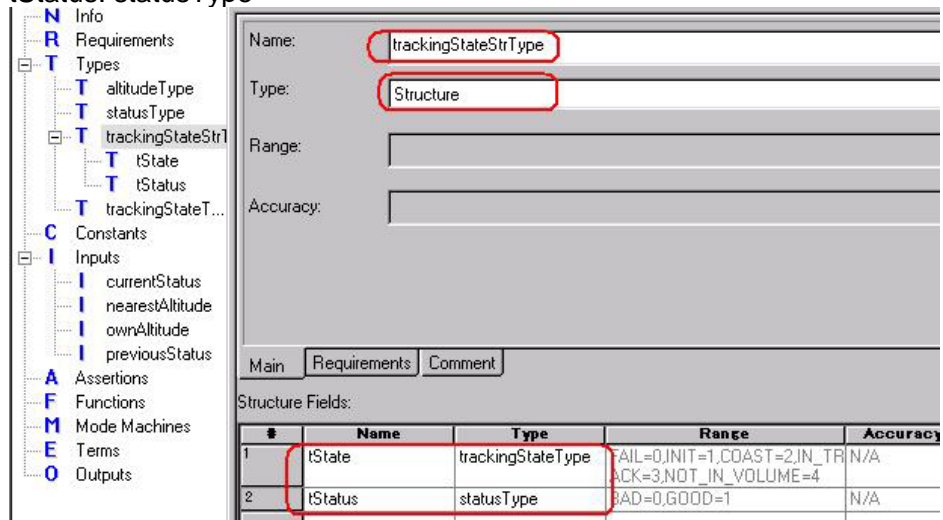
- 保存します。

< インタフェース >

4. TTM で、**vertical_tracker_interfaces3.ttm** を開きます。

5. Types に下記を追加します。

- Structure 型で 2 つのメンバを持つ **trackingStateStrType**
 - tState: trackingStateType
 - tStatus: statusType



- Structure 型で3つのメンバを持つ tracksType
 - a1, a2, a3: altitudeType

Structure Fields:

#	Name	Type	
1	a1	altitudeType	-100
2	a2	altitudeType	-100
3	a3	altitudeType	-100

6. Inputs に以下を追加します。

- **currentTrk, prevTrk** : tracksType

- 保存します。

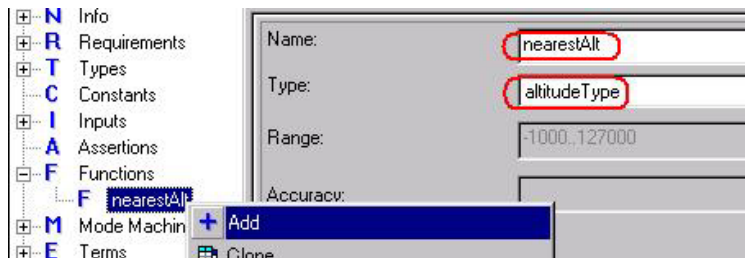
< 振舞い >

vertical_tracker3.ttm を開きます。

Functions : nearestAlt

7. functions で nearestAlt を作成します。(altitudeType)

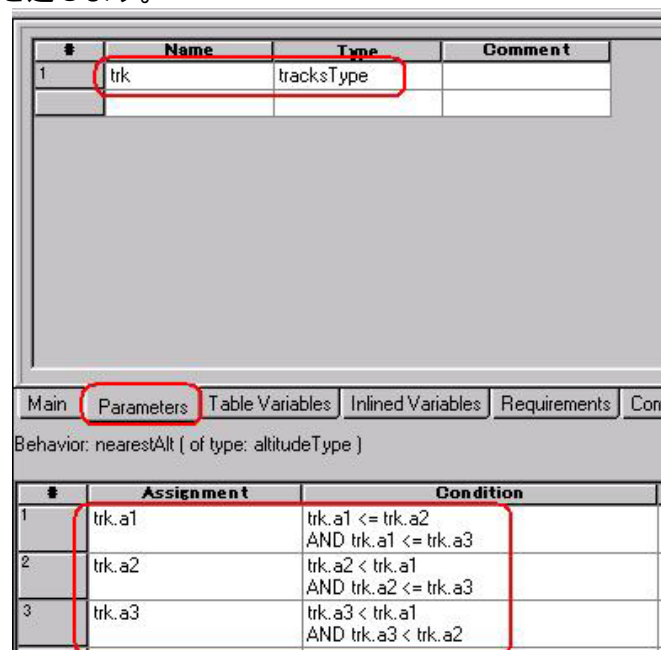
- この function は、要求仕様: Nearest_Track に関連します。



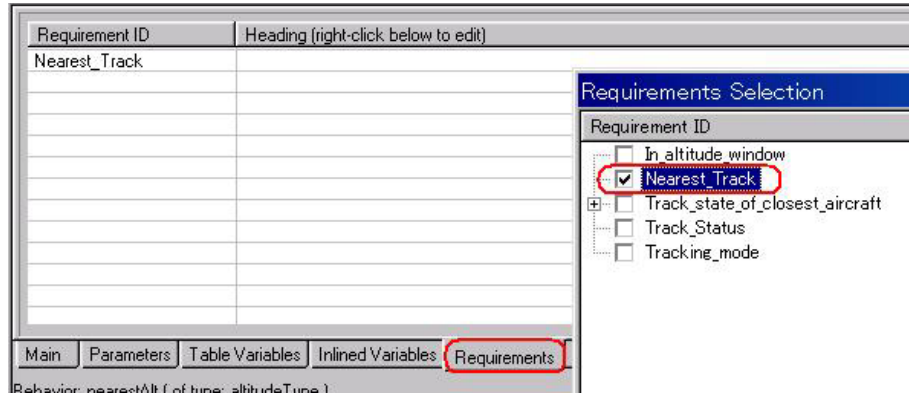
8. 入力引数 (Parameters タブ) を追加します。parameter: **trk** (tracksType)
function のパラメータは、構造化可能です。

9. 振舞いを記述します。

- この function は、3 つの中で最も小さい高度の値を返します。
trk.a1 <= trk.a2 かつ trk.a1 <= trk.a3 の時、
(trk.a1 が a2, a3 に等しい場合も含む) trk.a1 を返します。
trk.a2 < trk.a1 かつ trk.a2 <= trk.a3 の時、
trk.a2 を返します。
trk.a3 < trk.a1 かつ trk.a3 < trk.a2 の時、
trk.a3 を返します。

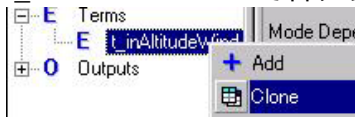


10. モデル化された振舞いへ要求仕様：Nearest_Track をリンクします。
 (Requirements タブ内で右クリックします。)

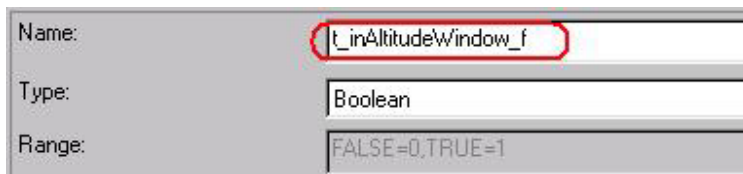


Term での nearestAlt Function 参照

11. t_inAltitudeWindow で右クリックし、Clone を選択します。



12. 複製されて、t_inAltitudeWindow_clone ができますので、名前を変更します。
 name : t_inAltitudeWindow_f



入力変数 nearestAltitude を参照する t_inAltitudeWindow_f のテーブルを修正します。

$\text{abs}(\text{ownAltitude} - \text{nearestAltitude}) \leq 2700$

下記のように、INTERM と function を使用します。

INTERM_nearest:= nearestAlt(currentTrk)

AND $\text{abs}(\text{ownAltitude} - \text{INTERM_nearest}) \leq 2700$

13. t_inAltitudeWindow_f の TRUE と FALSE 両方に対応する状態を作成します。

Behavior: t_inAltitudeWindow_f (of type: Boolean)

#	Assignment	Condition
1	TRUE	INTERM_nearest:=nearestAlt(currentTrk) AND $\text{abs}(\text{ownAltitude} - \text{INTERM_nearest}) \leq 2700$
2	FALSE	INTERM_nearest:=nearestAlt(currentTrk) AND $\text{abs}(\text{ownAltitude} - \text{INTERM_nearest}) > 2700$

< 出力の拡張 >

14. 出力 trackingState で右クリックし、Clone を選択します。

15. 複製されて、trackingState_clone ができますので、名前を trackingStateStr に変更します。（ trackingStateStrType に変更）



16. Term 変数 t_inAltitudeWindow を参照する trackingStateStr の状態を修正し、t_inAltitudeWindow_f を参照するよう変更します。

各 Assignment は、

- Assignment 箇所の tState は、トラッキング状態の値
- Assignment 箇所の tStatus は、入力変数 currentStatus の値

例：

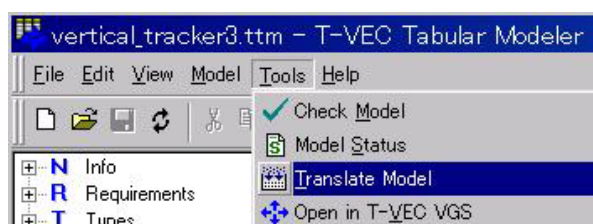
```
tState = INIT;
tStatus = currentStatus;
```

Behavior: trackingStateStr (of type: trackingStateStrType)

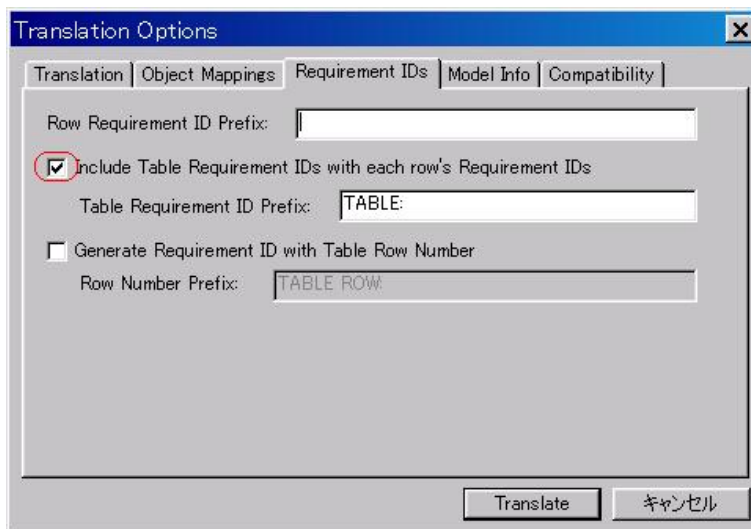
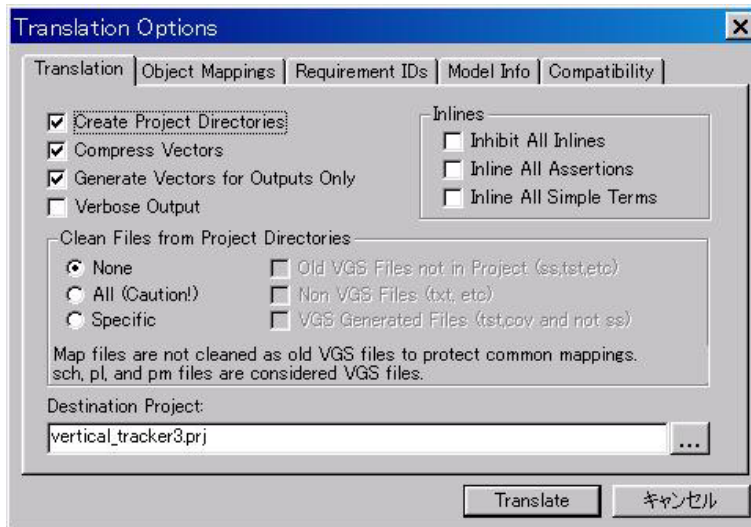
#	Assignment	Condition	Mode	Requirement ID
1	tState = FAIL; tStatus = currentStatus;	TRUE	NOT_TRACKING	Tracking_mode: Altitude >= 10000 feet
2	tState = FAIL; tStatus = currentStatus;	currentStatus = BAD AND previousStatus = BAD	TRACKING	Tracking_state_fail: Current and previous status bad
3	tState = INIT; tStatus = currentStatus;	currentStatus = GOOD AND previousStatus = BAD AND t_inAltitudeWindow_f	TRACKING	Tracking_state_initializing: Previous status bad, but current status g...
4	tState = COAST; tStatus = currentStatus;	previousStatus = GOOD AND currentStatus = BAD AND t_inAltitudeWindow_f	TRACKING	Tracking_state_coast: Previous status good, current status bad
5	tState = IN_TRACK; tStatus = currentStatus;	previousStatus = GOOD AND currentStatus = GOOD AND t_inAltitudeWindow_f	TRACKING	Tracking_state_in_track: Both status good and in altitude window
6	tState = NOT_IN_VOLUME; tStatus = currentStatus;	NOT (t_inAltitudeWindow_f) AND previousStatus = GOOD AND currentStatus = GOOD	TRACKING	Tracking_state_not_in_volume: No aircraft in window

< モデル変換とプロジェクトのビルド >

- Translate Model を実行します。

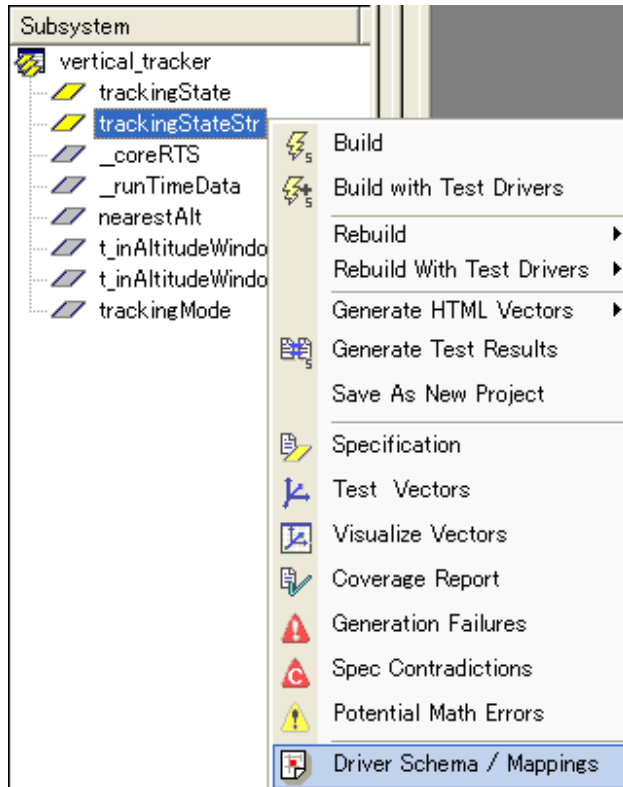


- trackingStateStr のベクタを表示。（モデルの変換、ビルドの実行）



<テストドライバ生成>

- テストドライバ環境設定を使用します。
- **C:-\TVEC_Tutorial\exercises\test_driver_utilities**
- 上記フォルダ内のスキーマとオブジェクトマッピングを、vertical_tracker3 テストドライバにも活用します。
- trackingStateStr のオブジェクトマッピングファイルを参照します。
TrackingStateStr サブシステム上で右クリックし、Driver Schema / Mappings を選択します。



< オブジェクトマッピングの構成 >

17. 空の期待出力スキーマとテストドライバスキーマセクションを削除します。

18. 以下の中身を確認しましょう。

- C:\TVEC_Tutorial\exercises\test_driver_utilities\common.map
- C:\TVEC_Tutorial\exercises\test_driver_utilities\vertical_tracker_interface.map

19. 出力 : trackingStateStr__OUT (tState、 tStatus) 以外の全てのオブジェクトマッピングを削除します。

入力変数全てのオブジェクトマッピングは、以下の map ファイル内にあります。
C:\TVEC_Tutorial\exercises\test_driver_utilities\vertical_tracker_interface.map

20. ユーザ定義変数 <order> を追加します。これは、入力変数設定を制御します。

```
<order> = 'ownAltitude, currentStatus, previousStatus,
          currentTrk.a1, currentTrk.a2, currentTrk.a3';
```

21. 更新したマップファイルを保存します。

```
<order> = 'ownAltitude, currentStatus, previousStatus, currentTrk.a1, currentTrk.a2, currentTrk.a3';
```

```
trackingStateStr__OUT.tState
{
    NAME_DESCRIPTOR 'trackingStateStr__OUT.tState';
    GET_DESCRIPTOR 'trackingStateStr__OUT.tState';
    SET_DESCRIPTOR 'trackingStateStr__OUT.tState = <binding->value>';
    TYPE_DESCRIPTOR 'trackingStateType';
    USER_DESCRIPTOR["1"] 'OUTPUT';
    USER_DESCRIPTOR["2"] '';
    USER_DESCRIPTOR["3"] '';
    USER_DESCRIPTOR["4"] '';
}

trackingStateStr__OUT.tStatus
{
    NAME_DESCRIPTOR 'trackingStateStr__OUT.tStatus';
    GET_DESCRIPTOR 'trackingStateStr__OUT.tStatus';
    SET_DESCRIPTOR 'trackingStateStr__OUT.tStatus = <binding->value>';
    TYPE_DESCRIPTOR 'statusType';
    USER_DESCRIPTOR["1"] 'OUTPUT';
    USER_DESCRIPTOR["2"] '';
    USER_DESCRIPTOR["3"] '';
    USER_DESCRIPTOR["4"] '';
}
```

22. ユーザ定義変数 <SCHEMA_HOME>を追加します。

```
<SCHEMA_HOME> = 'C:\TVEC_Tutorial\exercises\test_driver_utilities';
```

23. Perl への参照を追加します。

```
EMBED_PERL '<SCHEMA_HOME>\perl.pl';
```

24. テスト環境設定を提供する inits_and_declaration.map への参照を追加します。

```
INCLUDE '<SCHEMA_HOME>\inits_and_declarations.map';
```

25. ユーザ定義変数 <global_declarations>を追加します。

```
<global_declarations> = '
#include <stdio.h>
#include <stdlib.h>
#include "vert_tracker.h"
';
```

これは、inits_and_declaration.map の後にこなければなりません。なぜなら、inits_and_declarations.map でのデフォルト定義位置を無効にしてしまうからです。

26. ユーザ定義変数 <local_declarations>を追加します。

```
<local_declarations> = '
enum track_state_type state = fail;
int ownAltitude;
enum status_type currentStatus, previousStatus;
struct track_state_str_type track_state_str;
int currentTrk[3];<nl>';
```

これは、inits_and_declaration.map の後にこなければなりません。なぜなら、inits_and_declarations.map でのデフォルト定義位置を無効にしてしまうからです。

27. スキーマへの参照を追加します。

```
INCLUDE '<SCHEMA_HOME>\schema.sch';
```

28. マッピングへの参照を追加します。

```
INCLUDE '<SCHEMA_HOME>\common.map';
```

29. 更新したマップファイルを保存します。

```
<order> = 'ownAltitude, currentStatus, previousStatus, currentTrk.a1, currentTrk.a2, currentTrk.a3';
<SCHEMA_HOME> = '#mbd_course#exercises#test_driver_utilities';
EMBED_PERL '<SCHEMA_HOME>#perl.pl';
INCLUDE '<SCHEMA_HOME>#inits_and_declarations.map';

<global_declarations> = '
#include <stdio.h>
#include <stdlib.h>
#include "vert_tracker.h" ';

<local_declarations> = '
enum track_state_type state = fail;
int ownAltitude;
enum status_type currentStatus, previousStatus;
struct track_state_str_type track_state_str;
int currentTrk[3];<nl>';

INCLUDE '<SCHEMA_HOME>#schema.sch';
INCLUDE '<SCHEMA_HOME>#common.map';

trackingStateStr__OUT.tState
{
    NAME_DESCRIPTOR 'trackingStateStr__OUT.tState';
    SET_DESCRIPTOR 'trackingStateStr__OUT.tState';
}
```

30. 以下のように、trackingStateStr__OUT.tState の GET_DESCRIPTOR を置き換えます。

4 つの引数を持つ関数の呼び出し

```
track_state_str = vert_tracker3(ownAltitude, currentTrk,
                                currentStatus, previousStatus);
```

- 実行中、出力をプリントする printf 文を追加します。
printf("%s\n", get_state_string(track_state_str.tstate));
- <subsystem>.OUT ファイルにテスト結果を出力する fprintf 文を追加します。
fprintf(ofstream, "%s", get_state_string(track_state_str.tstate));
- 文字列内の tStatus をマップするために、コードを追加します。fprintf 文で <subsystem>.OUT ファイルに出力します。
if (track_state_str.tstatus == good) {
 fprintf (ofstream, "%s\n", "good");
}
else {
 fprintf (ofstream, "%s\n", "bad");
}

31. SET_DESCRIPTOR を空にします。

```
SET_DESCRIPTOR "";
```

32. trackingStateStr__OUT.tStatus の GET_DESCRIPTOR、SET_DESCRIPTOR を空にします。

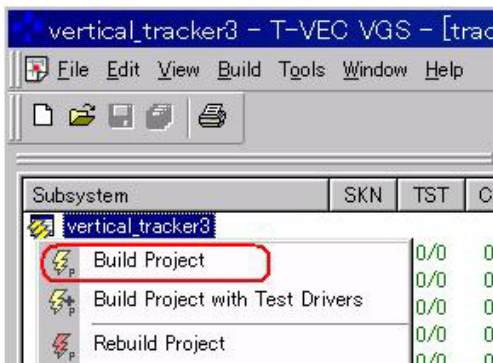
33. 更新したマップファイルを保存します。

```
trackingStateStr__OUT.tState
{
  NAME_DESCRIPTOR 'trackingStateStr__OUT.tState';
  GET_DESCRIPTOR
  track_state_str = vert_tracker3(ownAltitude, currentTrk,
                                currentStatus, previousStatus);
  printf("%s\n", get_state_string(track_state_str.tstate));
  fprintf(ofstream, "%s,", get_state_string(track_state_str.tstate));
  if (track_state_str.tstatus == good) {
    fprintf(ofstream, "%s\n", "good");
  }
  else {
    fprintf(ofstream, "%s\n", "bad");
  }
};
SET_DESCRIPTOR ``;
TYPE_DESCRIPTOR 'trackingStateType';
USER_DESCRIPTOR["1"] 'OUTPUT';
USER_DESCRIPTOR["2"] ``;
USER_DESCRIPTOR["3"] ``;
USER_DESCRIPTOR["4"] ``;
}

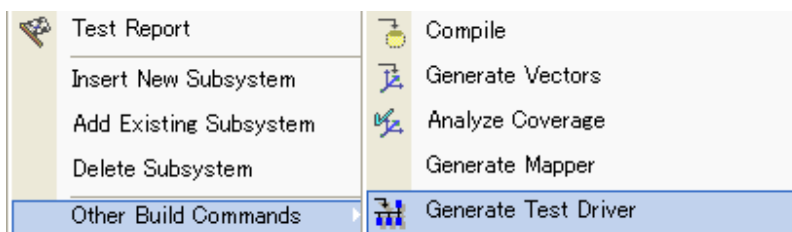
trackingStateStr__OUT.tStatus
{
  NAME_DESCRIPTOR 'trackingStateStr__OUT.tStatus';
  GET_DESCRIPTOR ``;
  SET_DESCRIPTOR ``;
  TYPE_DESCRIPTOR 'statusType';
  USER_DESCRIPTOR["1"] 'OUTPUT';
  USER_DESCRIPTOR["2"] ``;
  USER_DESCRIPTOR["3"] ``;
  USER_DESCRIPTOR["4"] ``;
}
}
```

<テストドライバの生成>

34. ビルドを行います。



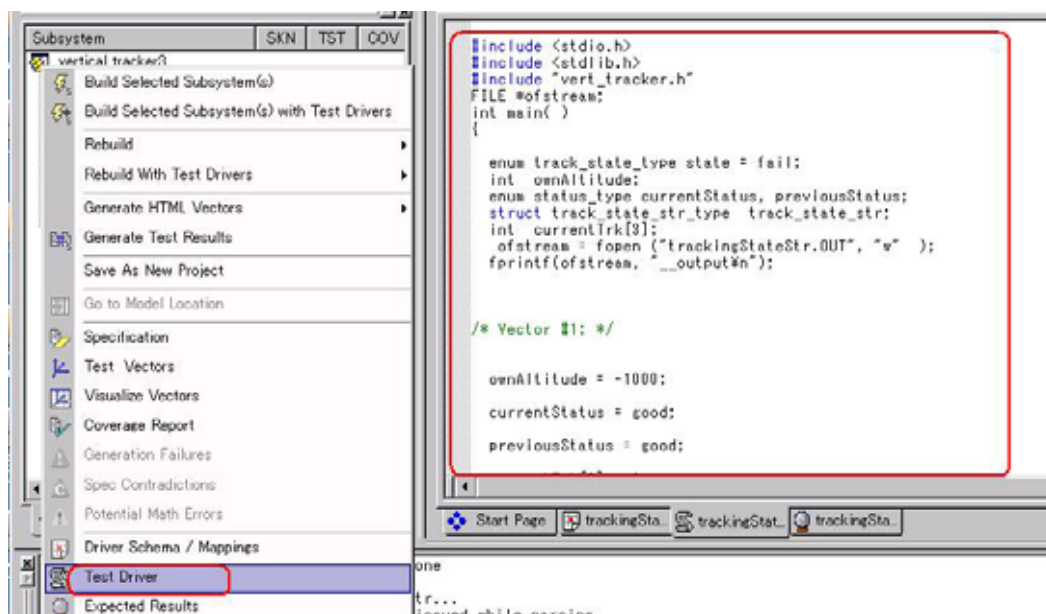
35. trackingStateStr サブシステムで右クリックし、
Other Build Commands で、Generate Test Driver を選択します。



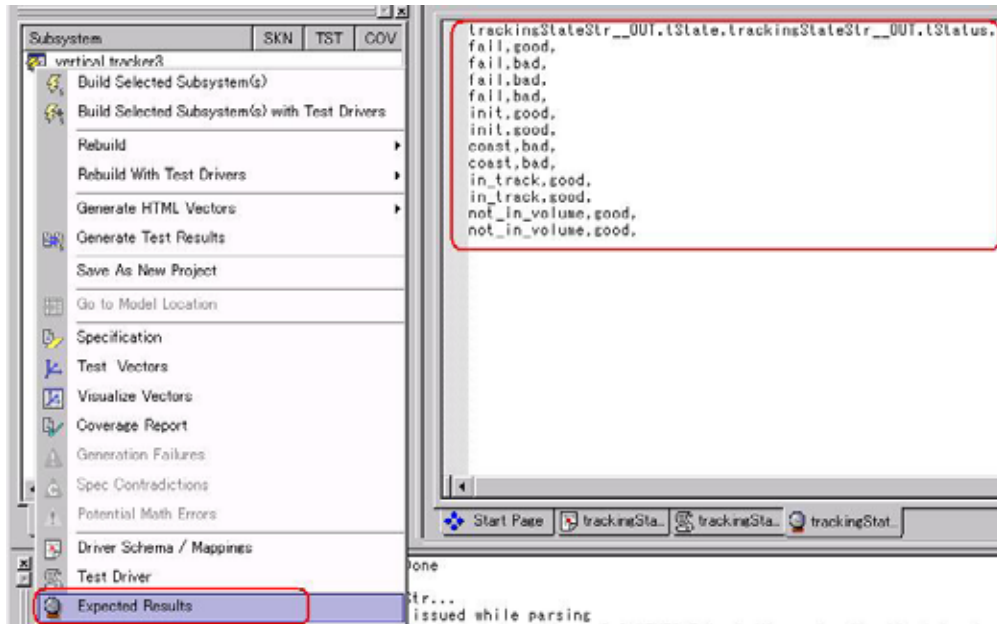
- テストドライバジェネレータはテストベクタ、オブジェクトマッピングファイル、期待出力、テストドライバスキーマを用います。
- test_drivers ディレクトリに以下が生成されます。
 - **trackingStateStr.eot** 期待出力値のリスト
 - **trackingStateStr.drv** 各テストケースの実行とファイルに結果を保存する C 言語テストドライバ

36. 期待出力とテストドライバファイルは、以下から見るができます。

- trackingStateStr で右クリックし、Test Driver を選択します。



- trackingStateStr で右クリックし、Expected Results を選択します。



<テスト実行と解析結果>

37. DOS プロンプトを開きます。

38. vertical_tracker3 の test_drivers ディレクトリに移動します。

- `cd C:\TVEC_Tutorial\exercises\vertical_tracker3\test_drivers`

39. vert_tracker.c と vert_tracker.h を test_drivers ディレクトリにコピーします。

- `copy ..\src\vert_tracker.*`

40. テストドライバを C ファイルにコピーします。

- `copy trackingStateStr.drv trackingStateStr.c`

41. テストドライバと vert_tracker.c をビルドし、実行形式ファイルを作成します。

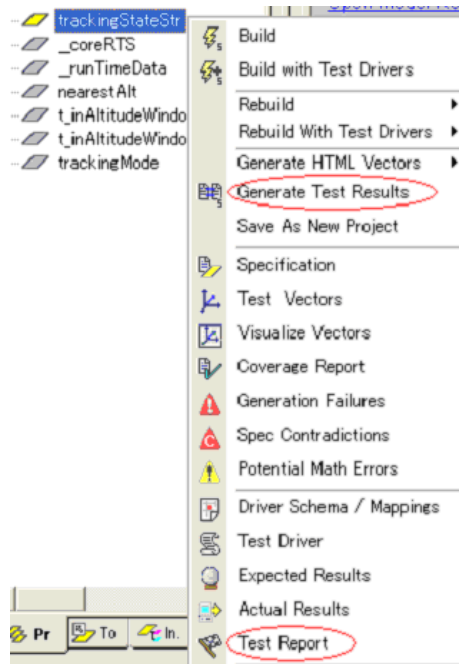
- `lc trackingStateStr.c vert_tracker.c -o trackingStateStr.exe`

42. trackingStateStr.exe ファイルを実行します。

43. 結果レポートを生成するために、trackingStateStr サブシステム上で右クリックし、Generate Test Results を選択し、結果を比較します。

44. 結果レポートを参照するために、trackingStateStr サブシステム上で右クリックし、Test Report を選択します。

Test Report の結果



Test	Object	Expected	Actual	Result
Errors		0		
Test Status		PASSED		
1	trackingStateStr__OUT.tState	fail	fail	OK
1	trackingStateStr__OUT.tStatus	good	good	OK
2	trackingStateStr__OUT.tState	fail	fail	OK
2	trackingStateStr__OUT.tStatus	bad	bad	OK
3	trackingStateStr__OUT.tState	fail	fail	OK
3	trackingStateStr__OUT.tStatus	bad	bad	OK
4	trackingStateStr__OUT.tState	fail	fail	OK
4	trackingStateStr__OUT.tStatus	bad	bad	OK
5	trackingStateStr__OUT.tState	init	init	OK
5	trackingStateStr__OUT.tStatus	good	good	OK
6	trackingStateStr__OUT.tState	init	init	OK
6	trackingStateStr__OUT.tStatus	good	good	OK
7	trackingStateStr__OUT.tState	coast	coast	OK
7	trackingStateStr__OUT.tStatus	bad	bad	OK
8	trackingStateStr__OUT.tState	coast	coast	OK
8	trackingStateStr__OUT.tStatus	bad	bad	OK
9	trackingStateStr__OUT.tState	in_track	in_track	OK
9	trackingStateStr__OUT.tStatus	good	good	OK

以上

ツールのデモンストレーションを行っています。
ご興味いただける場合は、お手数ですが下記までご依頼頂けると幸いです。



富士設備工業株式会社 電子機器事業部
〒591-8025 大阪府堺市北区長曾根町1928-1
Tel: 072-252-2128 www.fuji-setsu.co.jp