動的だと、ここが違う
カバレッジ計測、5つのメリット

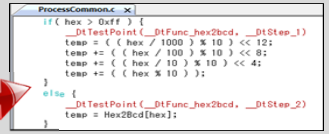
【 動的テストツール DTシリーズ 】

ハートランド・データ株式会社

動的テストツールDT10(ディーティーテン)とは

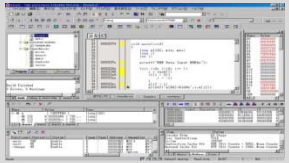
- ソフトウェアの動きを**長時間トレース**
- CPUやOSに依存しない**優れた汎用性**
- 多彩な解析機能で「**ソフトウェアを見える化**」

1 テストポイントを入れて



ソースコードに**自動で挿入**

2 いつも通りコンパイル



ふだんの開発環境をそのまま利用

3 実機を動かしてトレース

実機上のリアルなトレースデータ

動的テストツールDT10はソフトウェアの実行経路を長時間トレースできます。
トレースデータはカンタンに解析できるので、様々な開発フェーズで活用可能です。

< 使用イメージ >

DT10を使用したカバレッジ計測

- **DT10(DT10AE) で計測可能なカバレッジ**
  - **ステートメントカバレッジ(≒C0カバレッジ)**
  - **ブランチカバレッジ(≒C1カバレッジ)**
  - **関数カバレッジ**
  - **コールカバレッジ**
    - 関数コールカバレッジ(Caller)
    - 関数コールカバレッジ(Callee)
  - **MC/DC(Modified Condition/Decision Coverage)**

DT10を使用したカバレッジ計測

- 動的にカバレッジを計測するメリット

1. 事前準備がとても簡単
2. 複数のカバレッジを一挙に計測できる
3. ターゲットを実際に操作しながら計測できる
4. ソースコードの変更があっても大丈夫
5. 複数人でのチーム運用も簡単

DT10を使用したカバレッジ計測

1. 事前準備がとても簡単

- DT10を使えばカバレッジ計測に最適な構成で
  テストポイントを全て自動で挿入

```
void ProcessSubPb( UINT16 unit )
{

    MsgObj msg;
    TSK_Attrs task;
    UINT32 recSize;
    UINT16 j, folder, file;
    BuffmanFileMsg *pFileMsg = ( BuffmanFileMsg * ) msg.Data;

    __DtTestPoint(__DtFunc_ProcessSubPb, __DtStep_0)
    switch ( UnitStatus->CurLoadState ) {
        case Unit_INITIAL_STATE:    // 0x00
            __DtTestPoint(__DtFunc_ProcessSubPb, __DtStep_1)
            break;
        case Unit_EJECT_STATE:  // 0x01
            __DtTestPoint(__DtFunc_ProcessSubPb, __DtStep_2)
            if( UnitStatus->ProcessNo == PROCESS_STOP ) {
                /* EJECTからの遷移処理 */
                __DtTestPoint(__DtFunc_ProcessSubPb, __DtStep_3)
                UnitProcessEJECT(  );
            }
            break;
        case Unit_STOPPOSITION_STATE:   // 0x03
            __DtTestPoint(__DtFunc_ProcessSubPb, __DtStep_4)
            if( UnitStatus->ProcessNo == PROCESS_STOP ) {
                /* STOPからの遷移処理 */
                __DtTestPoint(__DtFunc_ProcessSubPb, __DtStep_5)
                UnitProcessSTOPPOSITION(  );
            }
            break;
        case Unit_NOCD_STATE:   // 0x02
```

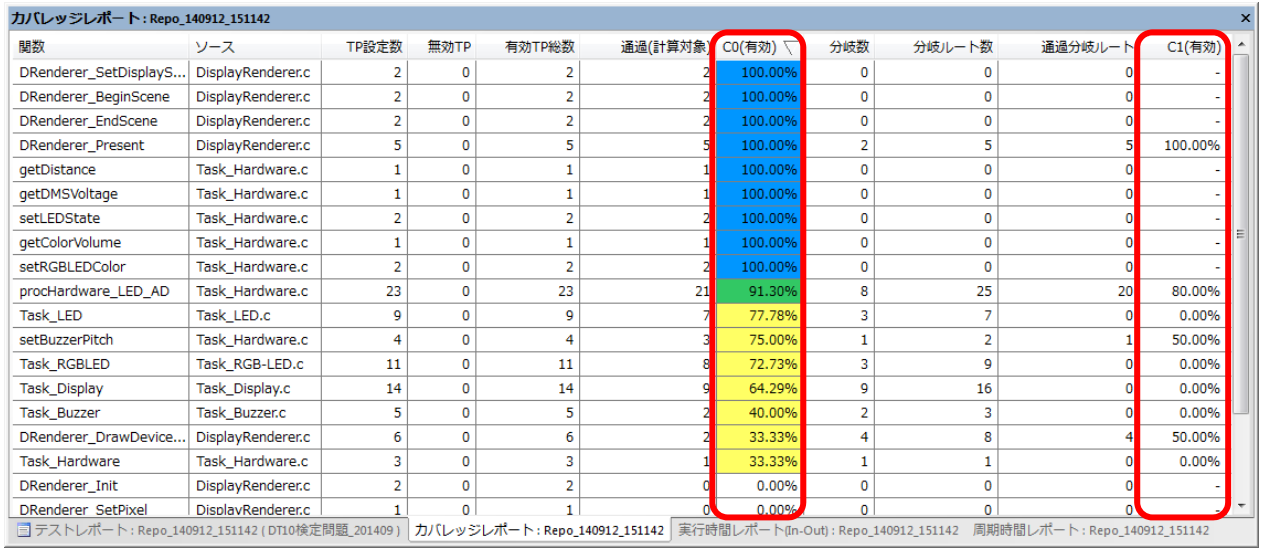
テストポイント

関数入口と出口や
各種条件分岐に対して
全て自動挿入可能

DT10を使用したカバレッジ計測

2. 複数のカバレッジを一挙に計測できる

- 取得したテストレポートを解析することにより
  全関数の各種カバレッジ率を自動算出



カバレッジレポート : Repo_140912_151142

関数	ソース	TP設定数	無効TP	有効TP総数	通過(計算対象)	C0(有効)	分岐数	分岐ルート数	通過分岐ルート	C1(有効)
DRenderer_SetDisplayS...	DisplayRenderer.c	2	0	2	2	100.00%	0	0	0	-
DRenderer_BeginScene	DisplayRenderer.c	2	0	2	2	100.00%	0	0	0	-
DRenderer_EndScene	DisplayRenderer.c	2	0	2	2	100.00%	0	0	0	-
DRenderer_Present	DisplayRenderer.c	5	0	5	5	100.00%	2	5	5	100.00%
getDistance	Task_Hardware.c	1	0	1	1	100.00%	0	0	0	-
getDMSVoltage	Task_Hardware.c	1	0	1	1	100.00%	0	0	0	-
setLEDState	Task_Hardware.c	2	0	2	2	100.00%	0	0	0	-
getColorVolume	Task_Hardware.c	1	0	1	1	100.00%	0	0	0	-
setRGBLEDColor	Task_Hardware.c	2	0	2	2	100.00%	0	0	0	-
procHardware_LED_AD	Task_Hardware.c	23	0	23	21	91.30%	8	25	20	80.00%
Task_LED	Task_LED.c	9	0	9	7	77.78%	3	7	0	0.00%
setBuzzerPitch	Task_Hardware.c	4	0	4	3	75.00%	1	2	1	50.00%
Task_RGBLED	Task_RGB-LED.c	11	0	11	8	72.73%	3	9	0	0.00%
Task_Display	Task_Display.c	14	0	14	9	64.29%	9	16	0	0.00%
Task_Buzzer	Task_Buzzer.c	5	0	5	2	40.00%	2	3	0	0.00%
DRenderer_DrawDevice...	DisplayRenderer.c	6	0	6	2	33.33%	4	8	4	50.00%
Task_Hardware	Task_Hardware.c	3	0	3	1	33.33%	1	1	0	0.00%
DRenderer_Init	DisplayRenderer.c	2	0	2	0	0.00%	0	0	0	-
DRenderer_SetPixel	DisplayRenderer.c	1	0	1	0	0.00%	0	0	0	-

テストレポート : Repo_140912_151142 (DT10検定問題_201409) | カバレッジレポート : Repo_140912_151142 | 実行時間レポート(In-Out) : Repo_140912_151142 | 周期時間レポート : Repo_140912_151142

DT10を使用したカバレッジ計測

2. 複数のカバレッジを一挙に計測できる

- ツール認証取得済みのDT10AEでは
  さらに高度なMC/DCも動的に計測可能

MC/DCレポート : Repo_170712_152232

ソース	関数	ステップ	条件数	網羅数	カバレッジ
Task_RGB-LED.c	Task_RGBLED	05 Decision(if)	1	1	100.00%
Task_RGB-LED.c	joke	01 Decision(if)	4	3	75.00%
Task_Control.c	Task_Control	04 Decision(if)	2	1	50.00%
Task_Control.c	Control_RGBLED	02 Decision(if)	3	0	0.00%
Task_Control.c	Control_RGBLED	08 Decision(if)	2	0	0.00%
Task_Control.c	Control_RGBLED	14 Decision(if)	2	0	0.00%
Task_Control.c	Control_RGBLED	20 Decision(if)	2	0	0.00%
Task_Control.c	Task_Control	02 Decision(while)	1	0	0.00%
Task_RGB-LED.c	Task_RGBLED	04 Decision(while)	1	0	0.00%
Task_RGB-LED.c	Task_RGBLED	10 Decision(if)	1	0	0.00%

使われているコンディションの数

満了したコンディションの数

MC/DC詳細レポート

(A || B) && (C || D)

No.	A	B	C	D	Result
1	T	x	T	x	True
2	T	x	F	T	True
3	T	x	F	F	False
4	F	T	T	x	True
5	F	T	F	T	True
6	F	T	F	F	False
7	F	F	x	x	False

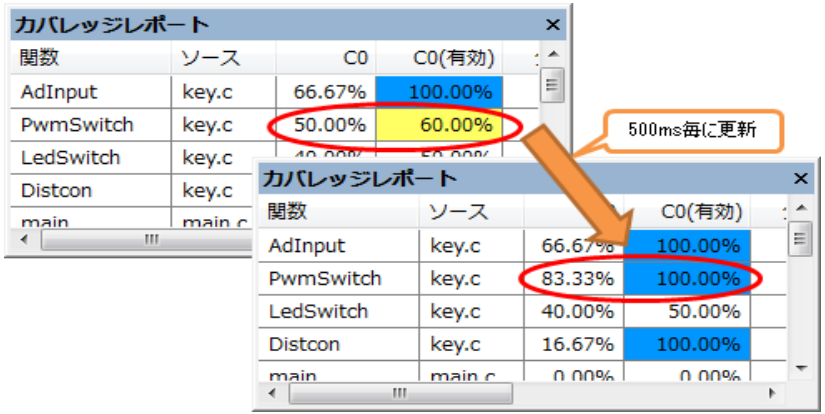
```
A :  ! line1
B :  * line1 <= 0
C :  ! line2
D :  * line2 <= 0
```

各コンディションを満了するために確認しなければならないテストケースのペアをわかりやすく表示

DT10を使用したカバレッジ計測

3. ターゲットを実際に操作しながら計測できる

- テストレポートを取得しながら、リアルタイムに
  カバレッジ率を計測することが可能



DT10を使用したカバレッジ計測

4. ソースコードの変更があっても大丈夫

- ソースコードの変更があっても、変更前の通過情報は
  変更後の環境に引き継ぐことが可能

```
// 値が変わったらスクリーンセーバ終了
if ( abs( d - last_distance ) >= VALID_RANGE )
{
    __DtTestPoint( __DtFunc_Task_Display, __DtStep_2 )
    idle_count = 0;
}
// スクリーンセーバ表示中
if ( idle_count >= MAX_SS_FRAME_COUNT )
{
    __DtTestPoint( __DtFunc_Task_Display, __DtStep_3 )
    DRenderer_BeginScene();
```

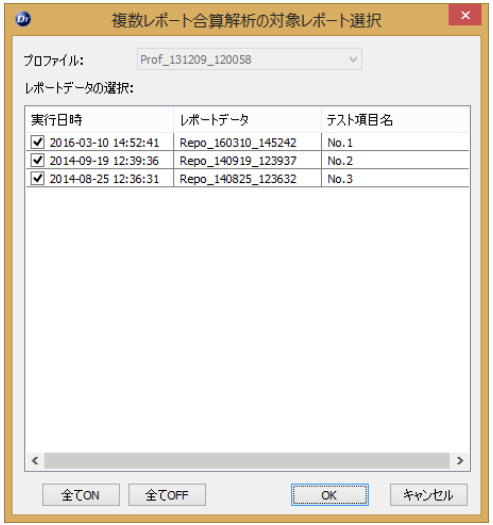
```
// 値が変わったらスクリーンセーバ終了
if ( abs( d - last_distance ) >= VALID_RANGE )
{
    __DtTestPoint( __DtFunc_Task_Display, __DtStep_2 )
    idle_count = 0;
}
// スクリーンセーバ表示中
if ( idle_count >= MAX_SS_FRAME_COUNT )
{
    __DtTestPoint( __DtFunc_Task_Display, __DtStep_3 )
    DRenderer_BeginScene();
```

通過済み

DT10を使用したカバレッジ計測

5. 複数人でのチーム運用も簡単

- プロジェクトファイルを共有することで、複数人で取得したレポートを集計したカバレッジ率を算出



まとめ

- DT10を使用すると、ターゲット機器の動作に伴ったカバレッジをリアルタイムで取得することが出来ます
  - リアルタイムカバレッジの取得
  - C0/C1カバレッジ率の算出
  - 機能安全で要求される関数カバレッジ、コールカバレッジ率、MC/DCの算出
  - テスト未実施箇所もわかりやすく表示
  - ソースコード変更やチーム運用も簡単

⇒ 面倒な事前設定は一切必要ありません

ご清聴ありがとうございました

HEARTLAND.DATA

ハートランド・データ株式会社