

正常なテストレポートが取得されているかを確認する方法

テストレポートが取得できたところで、正常なデータが取得できているか確認してみましょう。

誤ったレポートデータとは？

テストレポートは、ターゲット機器が出力したテストポイントの通過情報に対して、DT+Traceプロジェクトで管理するソースファイル名・関数名・ステップ種別の情報を紐づけて表示します。

No.	コア	ソース	関数	ステップ	説明
22 (22)		DisplayRenderer.c	DRenderer_BeginScene	01 FuncOut	
23 (23)		DisplayRenderer.c	DRenderer_DrawDeviceFormatImage	00 FuncIn	
24 (24)		DisplayRenderer.c	DRenderer_DrawDeviceFormatImage	03 if	
25 (25)		DisplayRenderer.c	DRenderer_DrawDeviceFormatImage	05 FuncOut	
26 (26)		DisplayRenderer.c	DRenderer_DrawDeviceFormatImage	00 FuncIn	
27 (27)		DisplayRenderer.c	DRenderer_DrawDeviceFormatImage	03 if	
28 (28)		ソース名	関数名	ステップ名	
29 (29)		DisplayRenderer.c	DRenderer_DrawDeviceFormatImage	00 FuncIn	
30 (30)		DisplayRenderer.c	DRenderer_DrawDeviceFormatImage	03 if & FuncOut	
31 (31)		DisplayRenderer.c	DRenderer_DrawDeviceFormatImage		
32 (32)		DisplayRenderer.c	DRenderer_DrawDeviceFormatImage		
33 (33)		DisplayRenderer.c	DRenderer_EndScene		
34 (34)		DisplayRenderer.c	DRenderer_EndScene		
35 (35)		tasks.c	***** Event Trigger		
36 (36)		tasks.c	***** Event Trigger		
37 (37)		Task_RGB-LED.c	Task_RGBLED		
38 (38)		Task_Hardware.c	getColorVolume		

しかし、以下のような問題があった場合、取得したデータと識別を管理する情報の不整合により、誤ったレポートデータが取得される可能性があります。

- ターゲットへ組み込むドライバの問題
- ターゲットとの物理的な接続・ノイズの問題
- DT+Traceプロジェクト作成時の誤った設定・手順など。

誤ったレポートデータの具体例

- ソース名のコラムに「*****Idle*****」と表示されるデータがある。
- 関数名のコラムに「*****Event Trigger*****」と表示されるデータがある。
- 変数値出力ポイントを挿入していないのに、「*****Dump Memory *****」と表示されるデータがある。
- 変数値出力ポイントが挿入された処理を通過していないのに、上記の誤ったデータが表示される。
- ソース名、関数名、ステップ名のコラムに、「0x01」などの16進数の数値データが表示される。

Idleと表示される。

16進数の数値が表示される。

テストレポートフィルタ機能を使った確認方法

DT+Traceのテストレポートフィルタ機能を使用することでテストレポート内の誤ったデータを発見することができます。

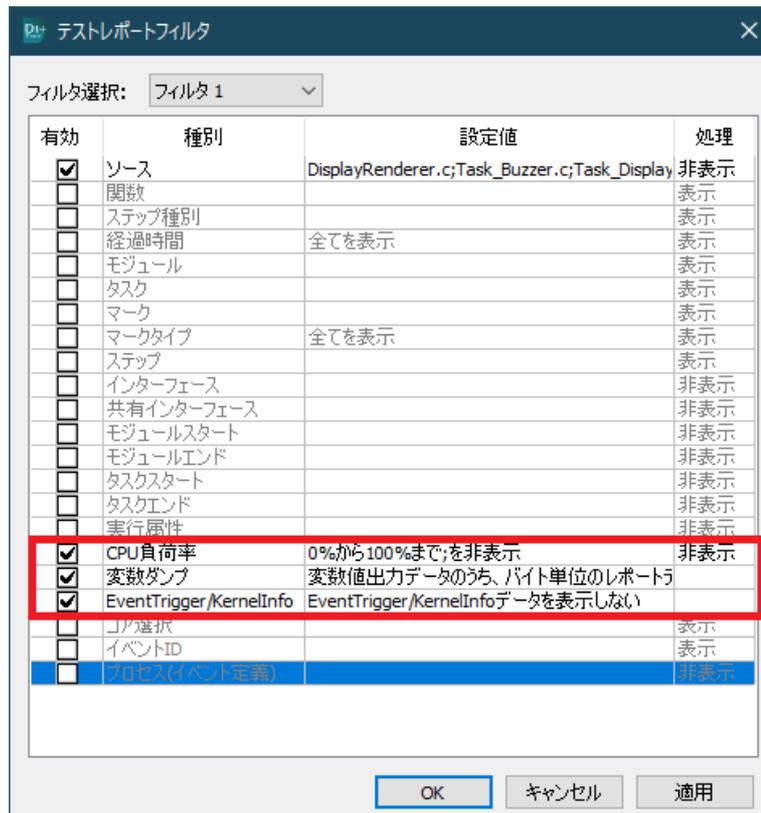
1. ソース名をフィルタの対象にする。
 - チェックしたソース名のデータがフィルタ対象となります。

状態	ソース	ベースアドレス	フォルダ
<input checked="" type="checkbox"/>	DisplayRenderer.c	0x00000010	C:\dtxprj_Ver10.13_...
<input checked="" type="checkbox"/>	Task_Buzzer.c	0x00000020	C:\dtxprj_Ver10.13_...
<input checked="" type="checkbox"/>	Task_Display.c	0x00000030	C:\dtxprj_Ver10.13_...
<input checked="" type="checkbox"/>	Task_Hardware.c	0x00000040	C:\dtxprj_Ver10.13_...
<input checked="" type="checkbox"/>	Task_LED.c	0x00000060	C:\dtxprj_Ver10.13_...
<input checked="" type="checkbox"/>	Task_RGB-LED.c	0x00000070	C:\dtxprj_Ver10.13_...
<input checked="" type="checkbox"/>	tasks.c	0x00000080	C:\dtxprj_Ver10.13_...

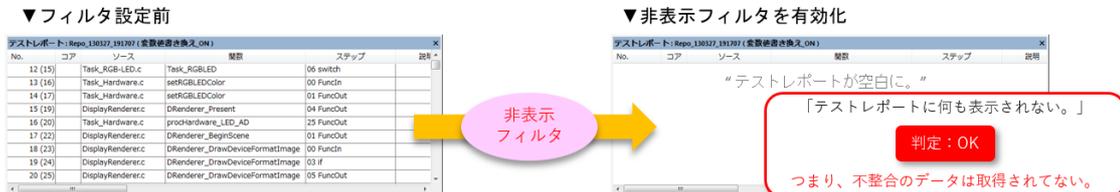
全てチェックをつける 全てチェックを外す OK キャンセル

2. ソース名のフィルタ処理を[非表示]で[有効]にチェックする。

- 選択したソース名を対象に非表示のフィルタとして、有効化します。
- CPU負荷率のデータを無視する場合は、非表示にします。
- 変数ダンプ、EventTrigger/KernelInfoのデータを無視する場合は、有効化します。



3. フィルタ後のテストレポートで判断する。



4. ソース名で非表示のフィルタを設定した時と同様に、関数名・ステップ種別についても、非表示フィルタを有効にして、誤ったデータがテストレポート上に表示されないことを確認していきます。

