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して表示します。



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

- ソース名のカラムに「*****Idle*****」と表示されるデータがある。
- 関数名のカラムに「*****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。
- 変数値出カ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、ステップ名のカラムに、「0x01」などの16進数の数値データが表示される。

	テストレ		spo_131206_114253			×	
	No.	37		解散	ステップ	288 *	
		1 (1)	· Ide ·····		0x0000		
		2 (2)	inter ide		0x0000		
		3 (3)	Idle		0x0000		
	-	4 (4)	***** Ide *****		0x0000		
_	_	5 (5)	ide		0x0000		
	-	6 (6)	ata drv.c	AtsDrvInterval	00 Funcin		
はにりまままれる		7 (7)	ata_drv.c	AtaDrvinterval	03 FuncOut		
IDIE C READ & 41/9%	-	8 (8)	ata_drv.c	AtaDrvInterval	00 Funcin		
	1	9 (9)	ata_drv.c	AtaDrvInterval	03 FuncOut		
	×						
	テスト		kpo,140317,191956			×	
	No.	37	7 ソ ー ス	製款	ステップ	288 *	
		1 (1)	DisplayRenderer.c		0x006b	LU I	
		2 (2)	DisplayRenderer.c		0x002b		
		3 (3)	DisplayRenderer.c	DRenderer_EndScene	01 FuncOut		
		4 (4)	DisplayRenderer.c	DRenderer_DrawDeviceFormatImage	03 #		
		5 (5)	DisplayRenderer.c	DRenderer_DrawDeviceFormatImage	04 if & FuncOut		* 16進数の数個が
		6 (6)	DisplayRenderer.c	DRenderer_DrawDeviceFormatImage	04 if & FuncOut		表示される。
		7 (7)	DisplayRenderer.c	DRenderer_DrawDeviceFormatImage	04 if & FuncOut		
		8 (8)	DisplayRenderer.c	DRenderer_DrawDeviceFormatImage	04 if & FuncOut		
		9 (9)	DisplayRenderer.c	DRenderer_DrawDeviceFormatImage	04 if & FuncOut		
	- C						

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。

≌ テストレポートフィルタ:ソース ×							
チェックされたソースファイルのみ表示します							
状態	ソース	ベースアドレス	フォルダ				
	DisplayRenderer.c	0x00000010	C:¥dtxprj_Ver10.13				
	Task_Buzzer.c	0x0000020	C:¥dtxprj_Ver10.13				
	Task_Display.c	0x0000030	C:¥dtxprj_Ver10.13				
	Task_Hardware.c	0x00000040	C:¥dtxprj_Ver10.13				
	Task_LED.c	0x0000060	C:¥dtxprj_Ver10.13				
	Task_RGB-LED.c	0x0000070	C:¥dtxprj_Ver10.13				
	tasks.c	0x0000080	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。

自効	種別	設定値	処理
	ソース	DisplayRenderer.c;Task Buzzer.c;Task Display	非表示
Π	関数		表示
Π	ステップ種別		表示
	経過時間	全てを表示	表示
	モジュール		表示
	タスク		表示
	マーク		表示
	マークタイプ	全てを表示	表示
	ステップ		表示
	インターフェース		非表示
	共有インターフェース		非表示
	モジュールスタート		非表示
	モジュールエンド		非表示
Ц	<u>タスクスタート</u>		非表示
Ц	タスクエンド		非表示
	美行應性		非表示
M	CPU貝何半 赤形 BN P	0%から100%まで;を非表示	非表示
N.	変要タンノ	変数値出力テータのつち、ハイト単位のレホートフ	
	Eventirigger/KernelInfo	Eventirigger/KerneiInfoナータを表示しない	-
H	コア連択		求示
	עניזעאיני		রহনে

3. フィルタ後のテストレポートで判断する。

•	フィ	ルタ設定前	向		
テストレオ	テストレポート: Repo_130327_191707 (変数値書き換え_ON) X				
No.	コア	ソース	関数	ステップ	說月 ^
12 (15)	Task_RGB-LED.c	Task_RGBLED	06 switch	
13 (16)	Task_Hardware.c	setRGBLEDColor	00 FuncIn	
14 (17)	Task_Hardware.c	setRGBLEDColor	01 FuncOut	
15 (19)	DisplayRenderer.c	DRenderer_Present	04 FuncOut	
16 (20)	Task_Hardware.c	procHardware_LED_AD	25 FuncOut	
17 (22)	DisplayRenderer.c	DRenderer_BeginScene	01 FuncOut	
18 (23)	DisplayRenderer.c	DRenderer_DrawDeviceFormatImage	00 FuncIn	
19 (24)	DisplayRenderer.c	DRenderer_DrawDeviceFormatImage	03 if	
20 (25)	DisplayRenderer.c	DRenderer_DrawDeviceFormatImage	05 FuncOut	*
•					

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。

🖭 テストレポートフィルタ				
フィルタ選	択: フィルタ1	~		
有効	種別	設定値	処理	
	ソース	DisplayRenderer.c;Task_Buzzer.c;Task_Display	非表示	
	関数	DRenderer_Init;DRenderer_SetDisplayString;D	非表示	
	ステップ種別	FuncIn;FuncOut;if;for;while;do;switch;try;thro	非表示	
	経過時間	全てを表示	表示	