

SOLID IDEのコードカバレッジ機能  
を使ってみよう！

2018.06.11

京都マイクロコンピュータ

# コードカバレッジの機能

何ができるか

# コードカバレッジの機能

- ソースコード中の各ベーシックブロック単位ごとに、実行された回数を記録、表示する機能です。

ファイル名	関数カバレッジ	行カバレッジ	範囲カバレッジ
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\ag903_sample\kopenvg\scissor.c	0% 0/4	0% 0/80	0% 0/8
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\ag903_sample\kopenvg\stroke.c	0% 0/4	0% 0/65	0% 0/7
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\ag903_sample\kosp\osp_sample.c	0% 0/15	0% 0/625	0% 0/554
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\ag903_sample\kspiflash\spi_sample.c	0% 0/1	0% 0/51	0% 0/51
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\ag903_sample\kssp\pccm5100a.c	0% 0/25	0% 0/1058	0% 0/773
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\ag903_sample\kssp\ssp_sample.c	0% 0/9	0% 0/92	0% 0/45
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\ag903_sample\ktmr\ktmr_sample.c	0% 0/19	0% 0/901	0% 0/630
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\ag903_sample\kuart\kuart_sample.c	0% 0/23	0% 0/719	0% 0/506
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\ag903_sample\kusbh\sample_app_cdc.c	0% 0/13	0% 0/705	0% 0/378
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\ag903_sample\kusbh\sample_app_hid.c	0% 0/27	0% 0/888	0% 0/450
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\ag903_sample\kusbh\sample_app_msc.c	42% 3/7.50%	138/272	46% 48/103
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\ag903_sample\kusbh\sample_util.c	33% 3/9	29% 60/205	28% 26/90
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\ag903_sample\kusbh\usb_sample.c	21% 4/19.36%	131/362	16% 26/154
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\ag903_sample\kwdt\kwdt_sample.c	0% 0/8	0% 0/139	0% 0/70
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\demo-AXELL-sample\com.c	66% 8/12.58%	105/178	33% 78/235
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\demo-AXELL-sample\file.c	0% 0/11	0% 0/457	0% 0/276
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\demo-AXELL-sample\fsif.c	0% 0/6	0% 0/127	0% 0/61
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\demo-AXELL-sample\kernel_cfg.c	50% 1/2	50% 3/6	50% 1/2
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\demo-AXELL-sample\malloc_check.c	0% 0/4	0% 0/52	0% 0/25
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\demo-AXELL-sample\sample.h	0% 0/1	0% 0/5	0% 0/1
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\demo-AXELL-sample\sample_common.c	10% 2/19	10% 27/247	7% 11/140
C:\DriveD\CURRENT_WORK\apps\DB51903PF-demo-AXELL-sample\demo-AXELL-sample\sample_main.c	63% 7/11.71%	168/267	63% 76/120
C:\DriveD\CURRENT_WORK\apps\inc\solid_cs_assert.h	0% 0/2		0% 0/3
C:\DriveD\CURRENT_WORK\apps\rtos\toppers_asp3\asp3\include\queue.h	0% 0/4		0% 0/6
C:\DriveD\CURRENT_WORK\apps\rtos\toppers_asp3\asp3\include\syslog.h	0%		0% 0/7
C:\DriveD\CURRENT_WORK\apps\rtos\toppers_asp3\asp3\kernel\arch\arm_gcc\common\arm.h	72%		0% 0/12
C:\DriveD\CURRENT_WORK\apps\rtos\toppers_asp3\asp3\kernel\arch\arm_gcc\common\arm_insn.h	0/60		0% 0/14
C:\DriveD\CURRENT_WORK\apps\rtos\toppers_asp3\asp3\kernel\arch\arm_gcc\common\core_kernel_impl.h	0/68		0% 0/11
C:\DriveD\CURRENT_WORK\apps\rtos\toppers_asp3\asp3\kernel\wait.h	0/60		0% 0/7

ソースコード単位のカバレッジサマリーを一覧表示

```

131 .....文字列出力
132 .....*/
133 void COM_PutStr(int P, t* str)
134 17 {
135 17 → static uint8_t
136 17 → ComMsg-> * dtq;
137 17 → ER → -ercd;
138 17 → uint32_t size;
139 17 → uint8_t * buf;
140 17 →
141 17 → size = (uint32_t)strlen((char const*)str);
142 17 → if(COM_PUTBUF_SIZE < size) {
143 0 → → size = COM_PUTBUF_SIZE; /* Bufサイズで切る */
144 0 → }
145 17 → if(COM_PUTBUF_CNT <= cnt) {
146 2 → → cnt = 0;
147 2 → }
148 17 → dtq = &ComPutMsgBuf[cnt];
149 17 → buf = &ComPutBuf[cnt++][0];
150 17 → sys_memcpy(buf, str, size);
151 17 →
152 17 → dtq->cmd = COMCMD_TX_STRING;
153 17 → dtq->size = size;
154 17 → dtq->buf = (void*)buf;
155 17 → ercd = tsnd_dtq(ComDtqID, (intptr_t)dtq, 100*1000);
156 17 → if(E_OK != ercd) {
157 0 → → ComInternalError((int32_t)ercd);
158 0 → }
159 17 → return;
160 17 }

```

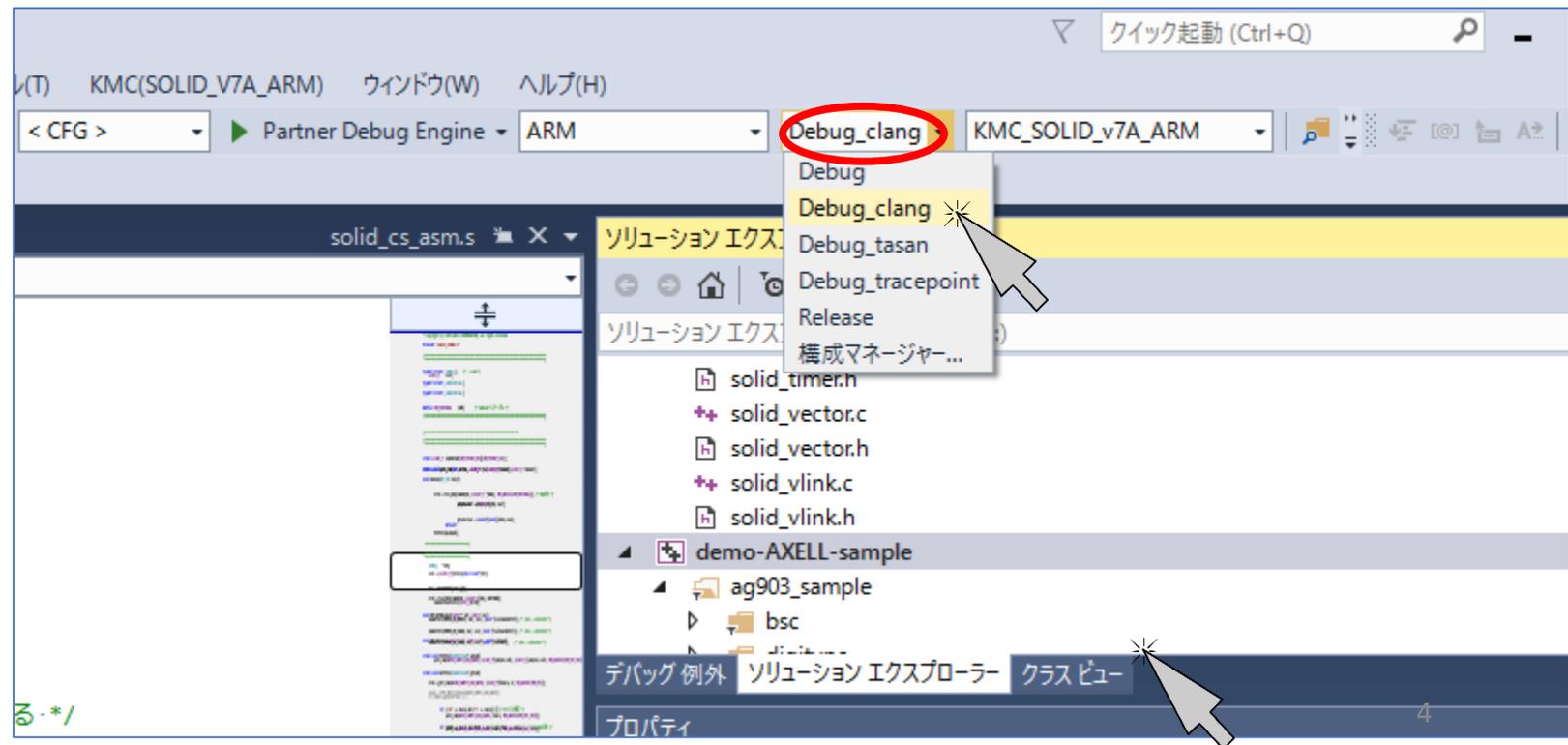
ソースコード上に実行回数をグラフィカルに表示

# コードカバレッジを使うための準備

使うために前もって設定すべき内容

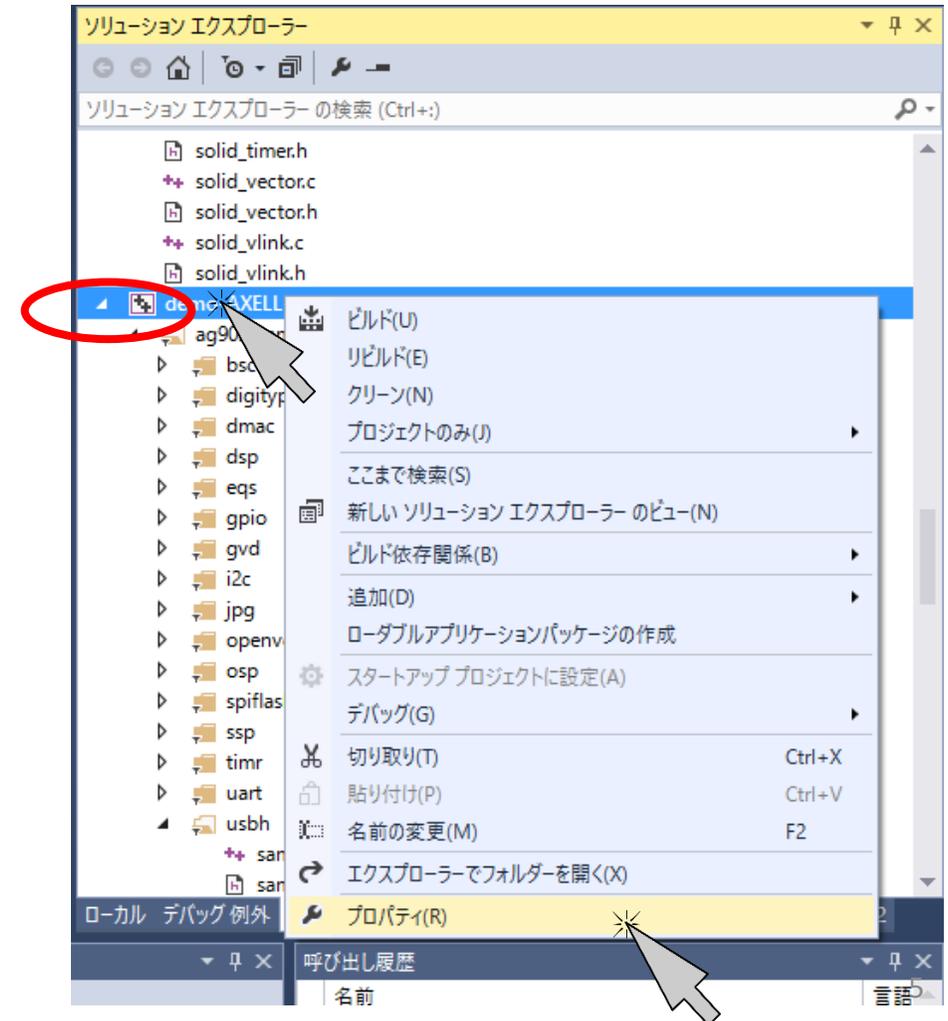
# コードカバレッジを使うための準備

- コードカバレッジは Clangコンパイラが必要  
「ソリューション構成」で、「Debug\_clang」を選択



# コードカバレッジを使うための準備

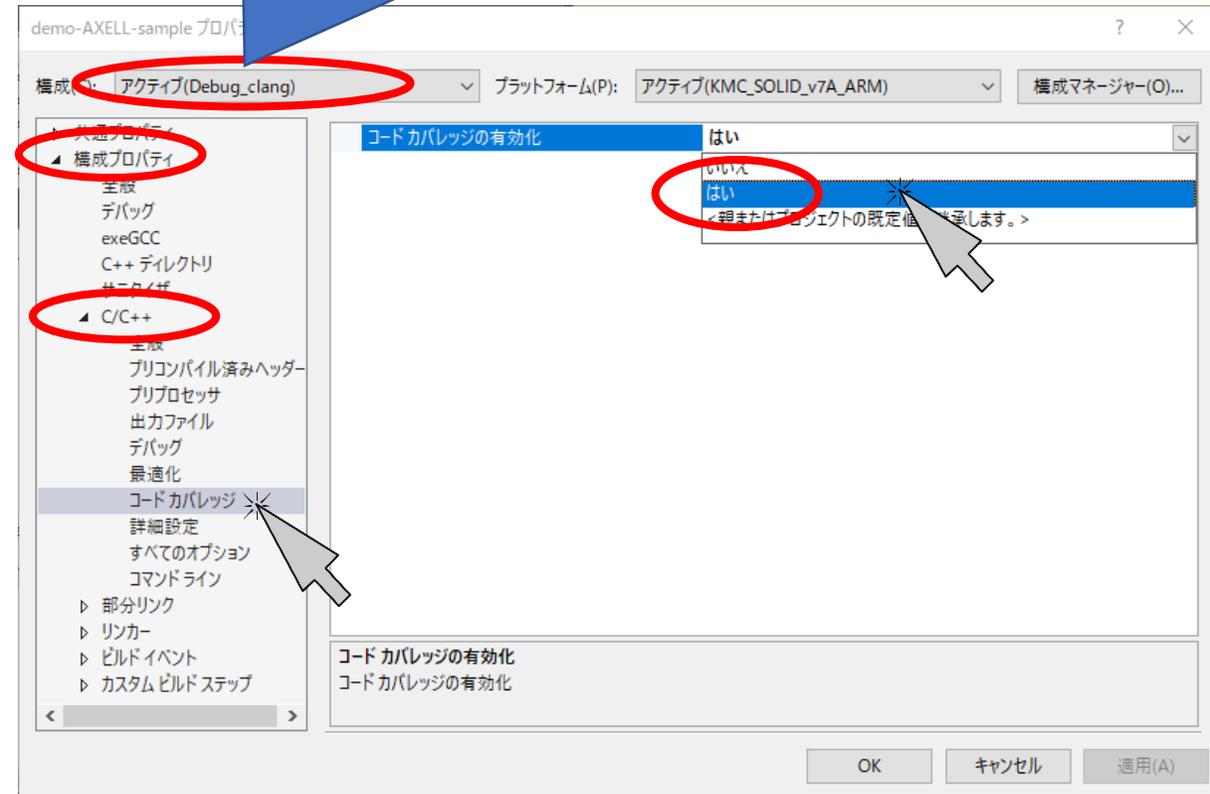
- プロジェクト単位で有効化
  1. ソリューションエクスプローラ  
のプロジェクトで右クリック
  2. ポップアップメニューからプロ  
パティを選択



# コードカバレッジを使うための準備

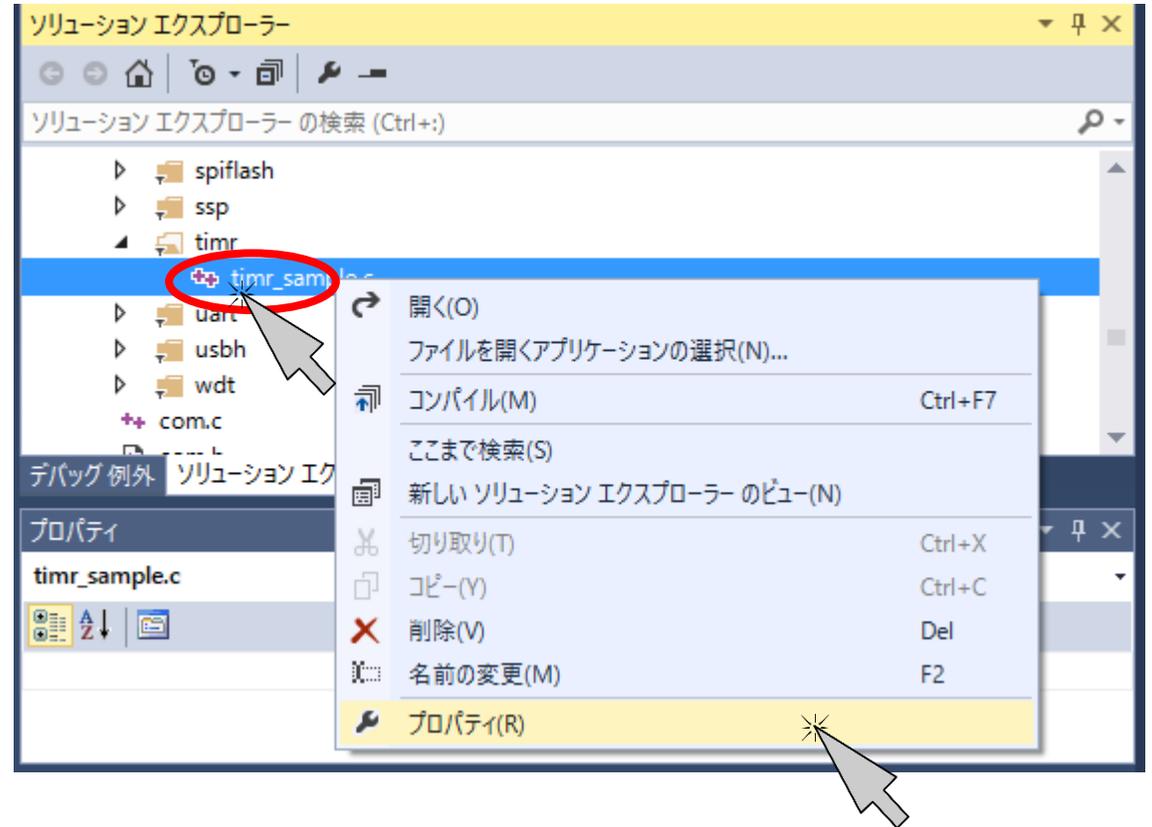
- プロジェクト単位で有効化
  - [構成プロパティ]-[C/C++]-[コードカバレッジ]を選択
  - [コードカバレッジの有効化]を、「はい」に設定

設定プロパティの構成が[Debug\_clang] or [アクティブ(Debug\_clang)]なのを確認



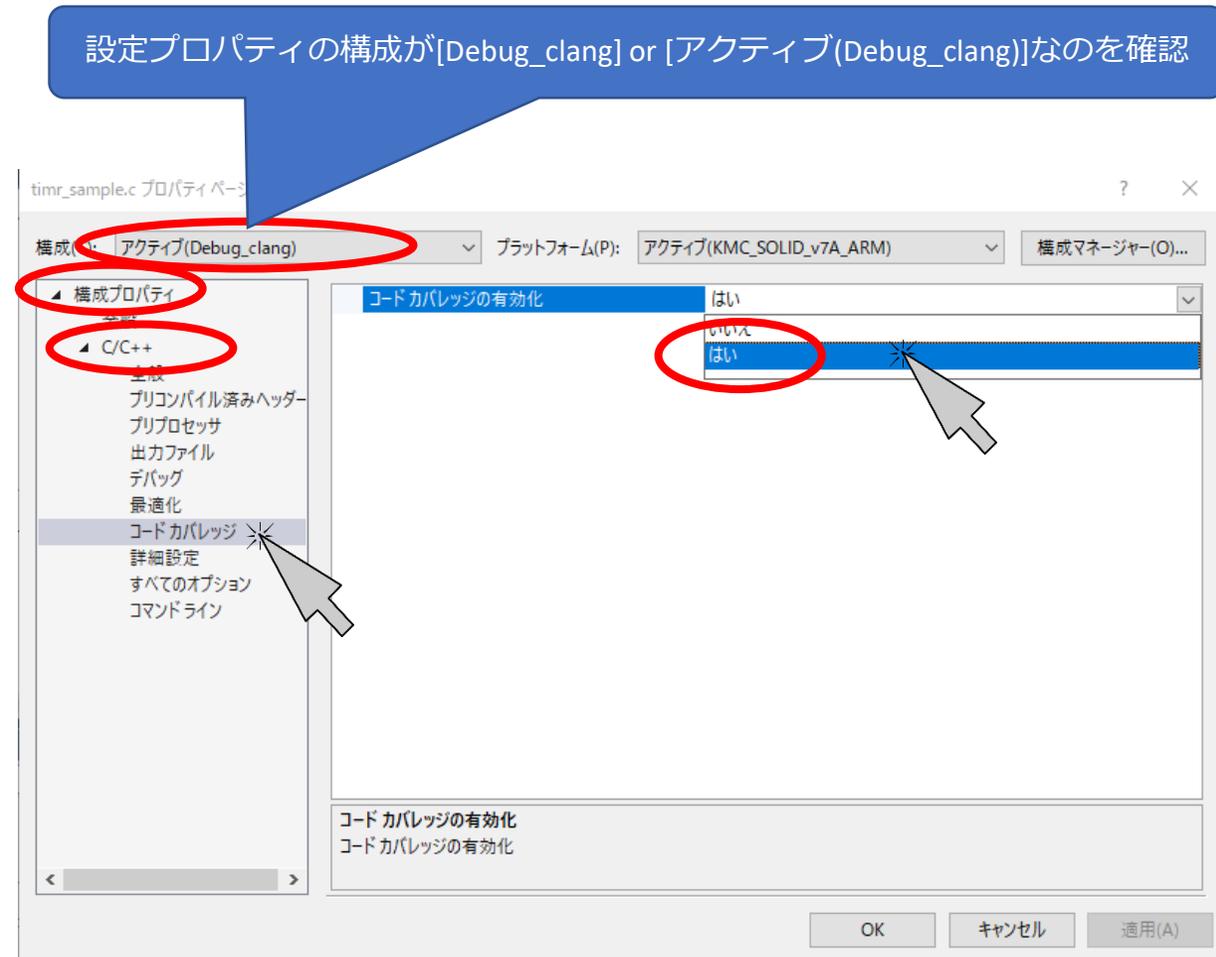
# コードカバレッジを使うための準備

- ソースファイル単位で有効化
  1. ソリューションエクスプローラ  
の各ソースで右クリック
  2. ポップアップメニューからプロ  
パティを選択



# コードカバレッジを使うための準備

- ソースファイル単位で有効化
  3. [構成プロパティ]-[C/C++]-[コードカバレッジ]を選択
  4. [コードカバレッジの有効化]を、「はい」に設定

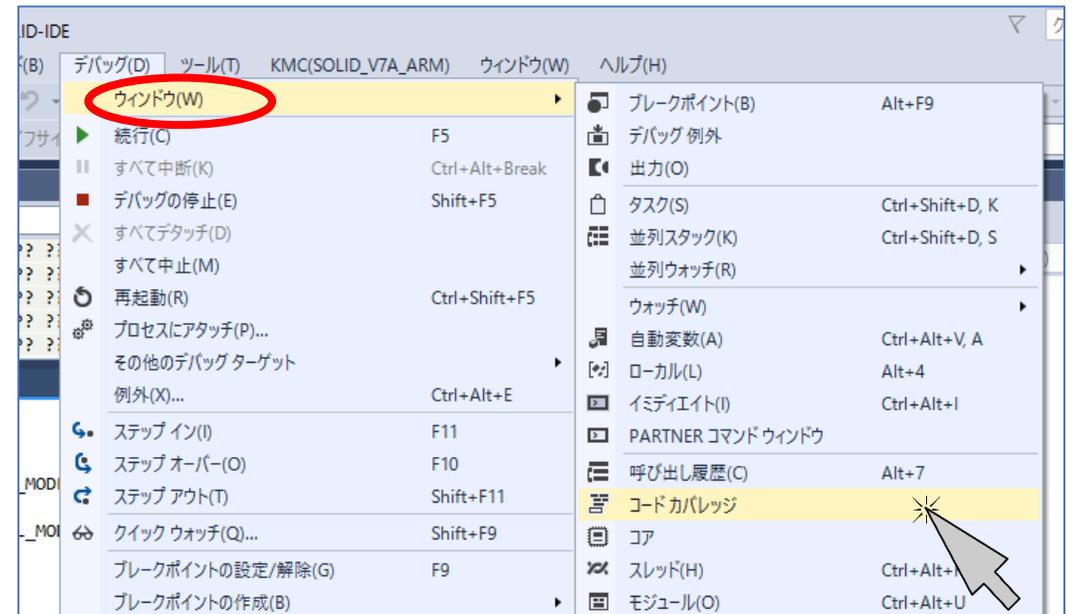


# コードカバレッジの使い方

実行する際の手順

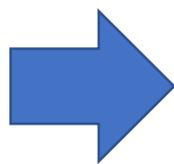
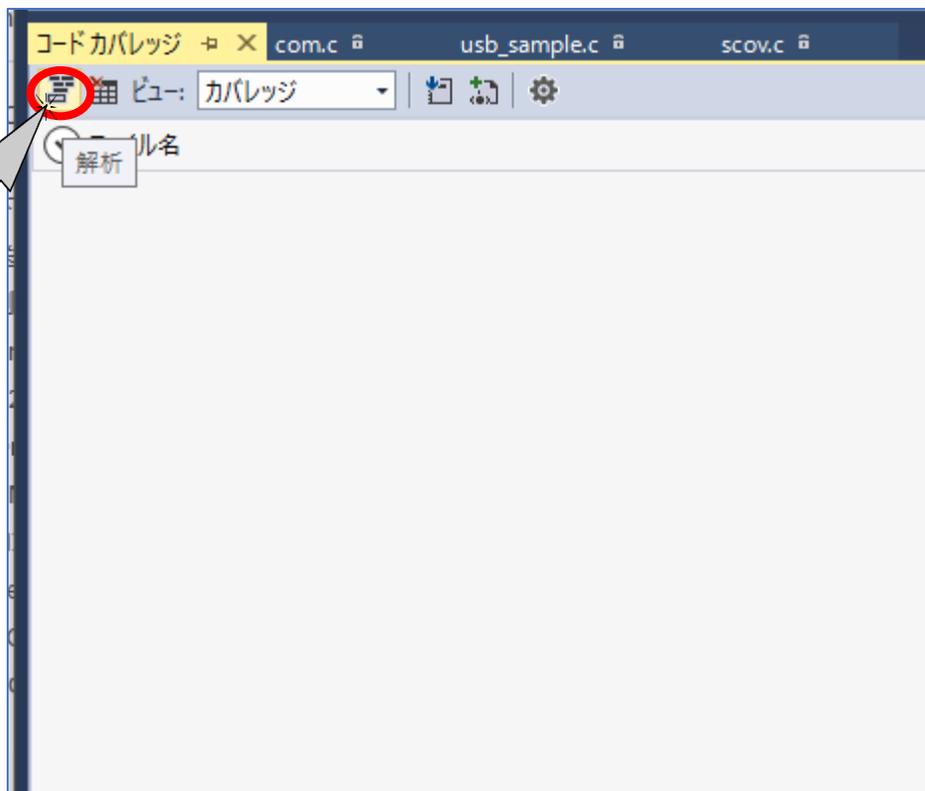
# コードカバレッジの使い方

1. コードカバレッジを有効化した後、有効化したプロジェクトと有効化したソースファイルを再ビルド
2. ビルドが成功したら、デバッグ実行。カバレッジを確認したい時点まで実行したら「すべて中断」
3. メニューの[デバッグ(D)]
  - [ウインドウ(W)]
  - [コードカバレッジ]を選択



# コードカバレッジの使い方

1. コードカバレッジ左上のアイコン  「解析」をクリック



	関数カバレッジ	行カバレッジ	範囲カバレッジ
h\$ag903_sample\$timr\$timr_sample.c	0% 0/19	0% 0/901	0% 0/630
h\$ag903_sample\$uart\$uart_sample.c	0% 0/23	0% 0/719	0% 0/506
h\$ag903_sample\$usbh\$sample_app_cdc.c	0% 0/13	0% 0/705	0% 0/378
h\$ag903_sample\$usbh\$sample_app_hid.c	0% 0/27	0% 0/888	0% 0/450
h\$ag903_sample\$usbh\$sample_app_msc.c	42% 3/7	50% 138/272	46% 48/103
h\$ag903_sample\$usbh\$sample_util.c	33% 3/9	29% 60/205	28% 26/90
h\$ag903_sample\$usbh\$usb_sample.c	21% 4/19	36% 131/362	16% 26/154
h\$ag903_sample\$wdt\$wdt_sample.c	0% 0/8	0% 0/139	0% 0/70
h\$demo-AXELL-sample\$com.c	66% 8/12	58% 105/178	33% 78/235
h\$demo-AXELL-sample\$file.c	0% 0/11	0% 0/457	0% 0/276
h\$demo-AXELL-sample\$fsif.c	0% 0/6	0% 0/127	0% 0/61
h\$demo-AXELL-sample\$kernel_cfg.c	50% 1/2	50% 3/6	50% 1/2
h\$demo-AXELL-sample\$malloc_check.c	0% 0/4	0% 0/52	0% 0/25
h\$demo-AXELL-sample\$sample.h	0% 0/1	0% 0/5	0% 0/1
h\$demo-AXELL-sample\$sample_common.c	10% 2/19	10% 27/247	7% 11/140
h\$demo-AXELL-sample\$sample_main.c	63% 7/11	71% 166/233	63% 76/120
h\$demo-AXELL-sample\$sample_main.h	0% 0/2	0% 0/2	0% 0/3
h\$demo-AXELL-sample\$ueue.h	0% 0/6	0% 0/36	0% 0/6
h\$demo-AXELL-sample\$syslog.h	0% 0/7	0% 0/63	0% 0/7
h\$demo-AXELL-sample\$arm_gcc\$common\$arm.h	0% 0/12	0% 0/72	0% 0/12

# コードカバレッジの使い方

2. カバレッジのサマリーが表示されます。

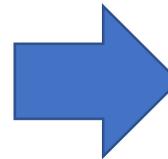
関数単位の  
カバレッジ行単位の  
カバレッジベーシックブロック単位  
のカバレッジ

	関数カバレッジ	行カバレッジ	範囲カバレッジ
sample%ag903_sample%timr%timr_sample.c	0% 0/19	0% 0/901	0% 0/630
sample%ag903_sample%uart%uart_sample.c	0% 0/23	0% 0/719	0% 0/506
sample%ag903_sample%usbh%sample_app_cdc.c	0% 0/13	0% 0/705	0% 0/378
sample%ag903_sample%usbh%sample_app_hid.c	0% 0/27	0% 0/888	0% 0/450
sample%ag903_sample%usbh%sample_app_msc.c	42% 3/7	50% 138/272	46% 48/103
sample%ag903_sample%usbh%sample_util.c	33% 3/9	29% 60/205	28% 26/90
sample%ag903_sample%usbh%usb_sample.c	21% 4/19	36% 131/362	16% 26/154
sample%ag903_sample%wdt%wdt_sample.c	0% 0/8	0% 0/139	0% 0/70
sample%demo-AXELL-sample%com.c	66% 8/12	58% 105/178	33% 78/235
sample%demo-AXELL-sample%file.c	0% 0/11	0% 0/457	0% 0/276
sample%demo-AXELL-sample%fsif.c	0% 0/6	0% 0/127	0% 0/61
sample%demo-AXELL-sample%kernel_cfg.c	50% 1/2	50% 3/6	50% 1/2
sample%demo-AXELL-sample%malloc_check.c	0% 0/4	0% 0/52	0% 0/25
sample%demo-AXELL-sample%sample.h	0% 0/1	0% 0/5	0% 0/1
sample%demo-AXELL-sample%sample_common.c	10% 2/19	10% 27/247	7% 11/140
sample%demo-AXELL-sample%sample_main.c	63% 7/11	71% 166/233	63% 76/120
	0% 0/2	0% 0/2	0% 0/3
queue.h	0% 0/6	0% 0/36	0% 0/6
syslog.h	0% 0/7	0% 0/63	0% 0/7
h%arm_gcc%common%arm.h	0% 0/12	0% 0/72	0% 0/12

# コードカバレッジの使い方

3. サマリのソースコードをダブルクリックするとカバレッジの詳細表示になります

	関数カバレッジ	行カバレッジ	範囲カバレッジ
g903_sample%timr%timr_sample.c	0% 0/19	0% 0/901	0% 0/630
g903_sample%uart%uart_sample.c	0% 0/23	0% 0/719	0% 0/506
g903_sample%usbh%sample_app_cdc.c	0% 0/13	0% 0/705	0% 0/378
g903_sample%usbh%sample_app_hid.c	0% 0/27	0% 0/888	0% 0/450
g903_sample%usbh%sample_app_msc.c	42% 3/7	50% 138/272	46% 48/103
g903_sample%usbh%sample_util.c	33% 3/9	29% 60/205	28% 26/90
g903_sample%usbh%usb_sample.c	21% 4/19	36% 131/362	16% 26/154
g903_sample%wdt%wdt_sample.c	0% 0/8	0% 0/139	0% 0/70
emo-AXELL-sample%com.c	66% 8/12	58% 105/178	33% 78/235
emo-AXELL-sample%file.c	0% 0/11	0% 0/457	0% 0/276
emo-AXELL-sample%fsif.c	0% 0/6	0% 0/127	0% 0/61
emo-AXELL-sample%kernel_cfg.c	50% 1/2	50% 3/6	50% 1/2
emo-AXELL-sample%malloc_check.c	0% 0/4	0% 0/52	0% 0/25
emo-AXELL-sample%sample.h	0% 0/1	0% 0/5	0% 0/1
emo-AXELL-sample%sample_common.c	10% 2/19	10% 27/247	7% 11/140



```

sample_app_msc.c  コードカバレッジ  com.c  usb_sar
133  1  →
134  1  → /*-----*/
135  1  → /*バッファ設定*/
136  1  → /*-----*/
137  1  → /*書き込みデータ設定*/
138  3  → for(i=0; i<USBH_MAX_CLS_MSC_NUM; i-
139  2  → → sample_memset(g_MscTaskInfoTbl[i].p
140  2  → → writeBuff[6]=('1'+i);
141  2  → → sample_memcpy(g_MscTaskInfoTbl[i].p
142  2  → → }
143  1  →
144  1  → /*-----*/
145  1  → /*メールボックス生成*/
146  1  → /*-----*/
147  1  → /*パラメータ設定*/
148  1  → ... cmbx.mbxatr⇒ = TA_TFIFO | TA_MFIFO;
149  1  → ... cmbx.maxmpri⇒ = 1;
150  1  → ... cmbx.mprihd⇒ = (intptr_t)NULL;
151  1  →
152  1  → /*接続切断タスク用メールボックス生成*/
153  1  → g_mbxid_msc_conn = acre_mbx(&cmbx);
154  1  → if(g_mbxid_msc_conn < 0){
155  0  → → /*エラー*/
156  0  → → return;
157  0  → → }
158  1  →
159  1  → /*-----*/
160  1  → /*接続/切断タスク生成*/
161  1  → /*-----*/
162  1  → /*パラメータ設定*/
  
```

# コードカバレッジの使い方

```
sample_app_msc.c  x コードカバレッジ  com.c  usb_sar
133 1 →
134 1 → /*-----*/
135 1 → /*パッファ設定*/
136 1 → /*-----*/
137 1 → /*書き込みデータ設定*/
138 3 → for(i = 0; i < USBH_MAX_CLS_MSC_NUM; i++)
139 2 →     sample_memset(g_MscTaskInfoTbl[i], 0, sizeof(g_MscTaskInfoTbl[i]));
140 2 →     writeBuff[6] = ('1' + i);
141 2 →     sample_memcpy(g_MscTaskInfoTbl[i], writeBuff, sizeof(writeBuff));
142 2 → }
143 1 →
144 1 → /*-----*/
145 1 → /*メールボックス生成*/
146 1 → /*-----*/
147 1 → /*パラメータ設定*/
148 1 →     cmbx.mbxatr = TA_TFIFO | TA_MFIFO;
149 1 →     cmbx.maxmpri = -1;
150 1 →     cmbx.mprihd = (intptr_t) NULL;
151 1 →
152 1 → /*接続切断タスク用メールボックス生成*/
153 1 →     g_mbxid_msc_conn = acre_mbx(&cmbx);
154 1 →     if(g_mbxid_msc_conn < 0){
155 0 →         /*エラー*/
156 0 →         return;
157 0 →     }
158 1 →
159 1 → /*-----*/
160 1 → /*接続/切断タスク生成*/
161 1 → /*-----*/
162 1 → /*パラメータ設定*/
```

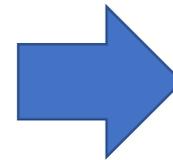
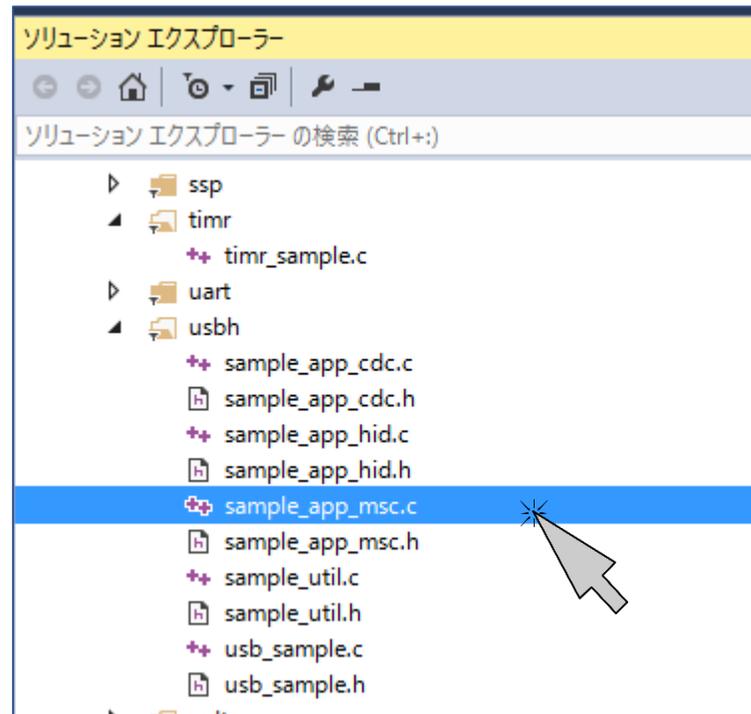
その行が実行された  
回数

薄緑は  
「一回以上実行された」  
ベーシックブロック

薄赤は  
「一度も実行されていない」  
ベーシックブロック

# コードカバレッジの使い方

解析実行後は、ソリューション  
エクスプローラでダブルクリックでも  
カバレッジの詳細が表示されます

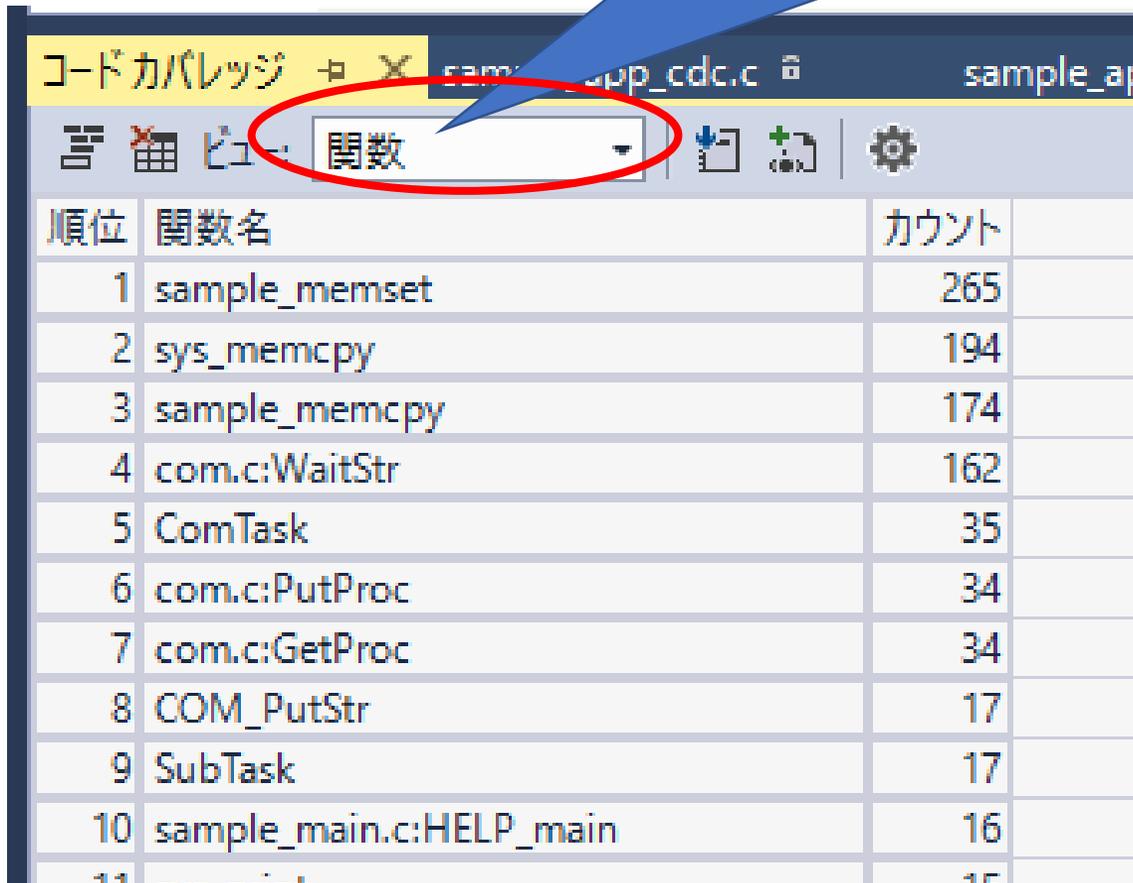


```
sample_app_msc.c  コードカバレッジ  com.c  usb_sar
133  1  →  /*-----*/
134  1  →  /* バッファ設定 */
135  1  →  /*-----*/
136  1  →  /* 書き込みデータ設定 */
137  1  →  /*-----*/
138  3  →  for(i = 0; i < USBH_MAX_CLS_MSC_NUM; i++)
139  2  →  → sample_memset(g_MscTaskInfoTbl[i].p
140  2  →  → writeBuff[6] = ('1' + i);
141  2  →  → sample_memcpy(g_MscTaskInfoTbl[i].p
142  2  →  → }
143  1  →  /*-----*/
144  1  →  /* メールボックス生成 */
145  1  →  /*-----*/
146  1  →  /* パラメータ設定 */
147  1  →  /*-----*/
148  1  →  cmbx.mbxattr = TA_TFIFO | TA_MFIFO;
149  1  →  cmbx.maxmpri = 1;
150  1  →  cmbx.mprihd = (intptr_t) NULL;
151  1  →  /*-----*/
152  1  →  /* 接続切断タスク用メールボックス生成 */
153  1  →  g_mbxid_msc_conn = acre_mbx(&cmbx);
154  1  →  if(g_mbxid_msc_conn < 0) {
155  0  →  → /* エラー */
156  0  →  → return;
157  0  →  → }
158  1  →  /*-----*/
159  1  →  /*-----*/
160  1  →  /* 接続/切断タスク生成 */
161  1  →  /*-----*/
162  1  →  /* パラメータ設定 */
```

15

# コードカバレッジの使い方

[解析]後、「ビュー」を「カバレッジ」->「関数」に切り替える



順位	関数名	カウント
1	sample_memset	265
2	sys_memcpy	194
3	sample_memcpy	174
4	com.c:WaitStr	162
5	ComTask	35
6	com.c:PutProc	34
7	com.c:GetProc	34
8	COM_PutStr	17
9	SubTask	17
10	sample_main.c:HELP_main	16
11	...	15

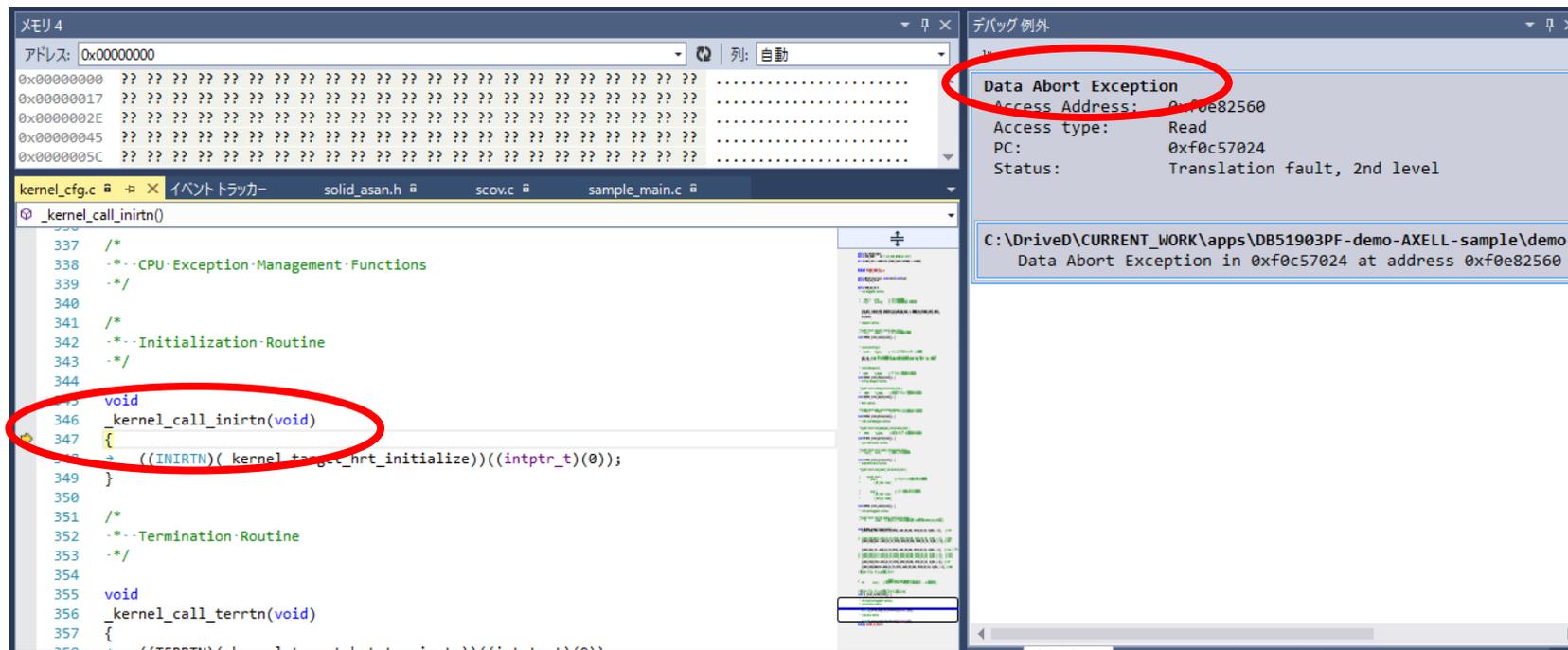
- 関数単位のサマリ表示  
関数単位のカバレッジを、カウント数の順位で一覧表示
- 関数名でソートすることもできます。

# 問題が起きたときは...

トラブルシューティング

# 問題が起きたとき

- kernel\_cfg.c の \_kernel\_call\_inirtn() で “Data Abort Exception” が発生した場合、solid\_cs.ldリンカスクリプトに、コードカバレッジ用の変更が入っていない為起きています。付録で修正すべき点を説明しています。

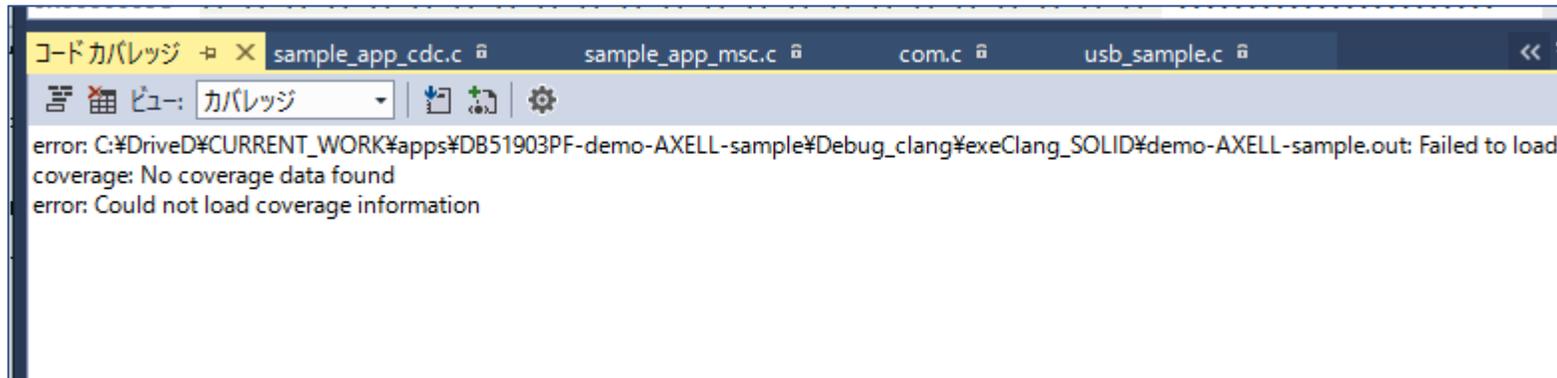


# 問題が起きたとき

- メニューの [デバッグ]-[ウインドウ]の先のメニューに [コードカバレッジ]が無い。
  - SOLID-IDE のバージョンがVer 1.1.11.0 以降か確認
  - [すべて中断]状態か確認。[デバッグ開始]前であったり、[デバッグの終了]をした後では、コードカバレッジは機能しません。

# 問題が起きたとき

- [解析]  をクリックしたら、サマリが表示されずエラー  
カバレッジのデータが無かった場合に起きます。  
少なくとも一つのプロジェクトがソースのカバレッジを有効に  
なっているか確認してください。



# メモリ使用量、処理時間 オーバーヘッド

# コードカバレッジのオーバーヘッド

※対象コードの複雑さ(ベーシックブロックの量)に依存します

- メモリ使用量  
ベーシックブロック1カ所につき、以下のサイズが
  - 記録用カウンタ 64bit (8bytes)
  - 記録用実行コード6~8 step (24~32bytes) ※上記カウンタの加算
- 処理時間オーバーヘッド  
ベーシックブロック実行時に記録用実行コードの実行

# 一例として dhrystone

- dhrystoneアプリでの処理時間測定結果 (Cortex-A5 400MHz)

カバレッジ	Dhrystone 一回の処理時間 (μ秒)	
	最適化無し (-O0)	最適化有り (-O2)
無効	585	205
有効	742	357

- 開発者が以下の用途に使うのには最適
  - 期待した部分が実行されているかどうかの確認
  - 資料が少ない他人が書いたコードの解析時
  - コードの効率化の為、実行回数の多い部分の抽出

# 付録



# リンカスクリプト solid\_cs.ld の修正点

- .dataセクションに以下の記述を追加して下さい

```
.data ALIGN(4K) : {
    _solid_data_start = .;
    *(.data .data.*)
    *(.gnu.linkonce.d.*)

    /* ****追加ここから**** */
    /* Append the LLVM profiling */
    . = ALIGN(8);
    PROVIDE(__start__llvm_prf_cnts = .);
    KEEP(*(__llvm_prf_cnts))
    PROVIDE(__stop__llvm_prf_cnts = .);

    . = ALIGN(8);
    PROVIDE(__start__llvm_prf_data = .);
    KEEP(*(__llvm_prf_data))
    PROVIDE(__stop__llvm_prf_data = .);

    /* ****追加ここまで**** */

    _solid_data_end = .;
}
```

以上です

