2017.04.03_V3.6

埋め込みコードカバレッジ計測
FAQ集

本書は、埋め込みコード（カバレッジ測定用オブジェクト）環境作成時に、手順通り実施して、カバレッジ計測が正しく出来ない場合の対応方法について記載しています。

埋め込みコードの使用方法については、ユーザー向け技術サポート情報から参照できる下記チュートリアルの「【応用編】埋め込みコードによるカバレッジ計測」を参照してください。

http://www.gaio.co.jp/support/user/pdf/CoverageMaster_Tutorial.pdf

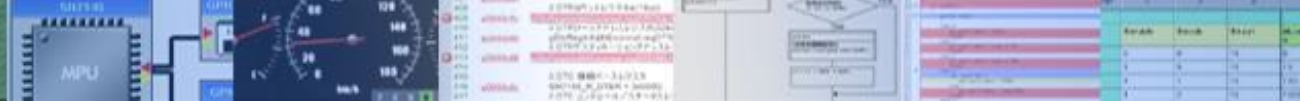
GAIO TECHNOLOGY

目次①

GAIO TECHNOLOGY

目次②

- 【詳細③A,B】
  マイコン、コンパイラ特有の問題
  - V850 / GHS P43～
  - R32C / NC100 P45～

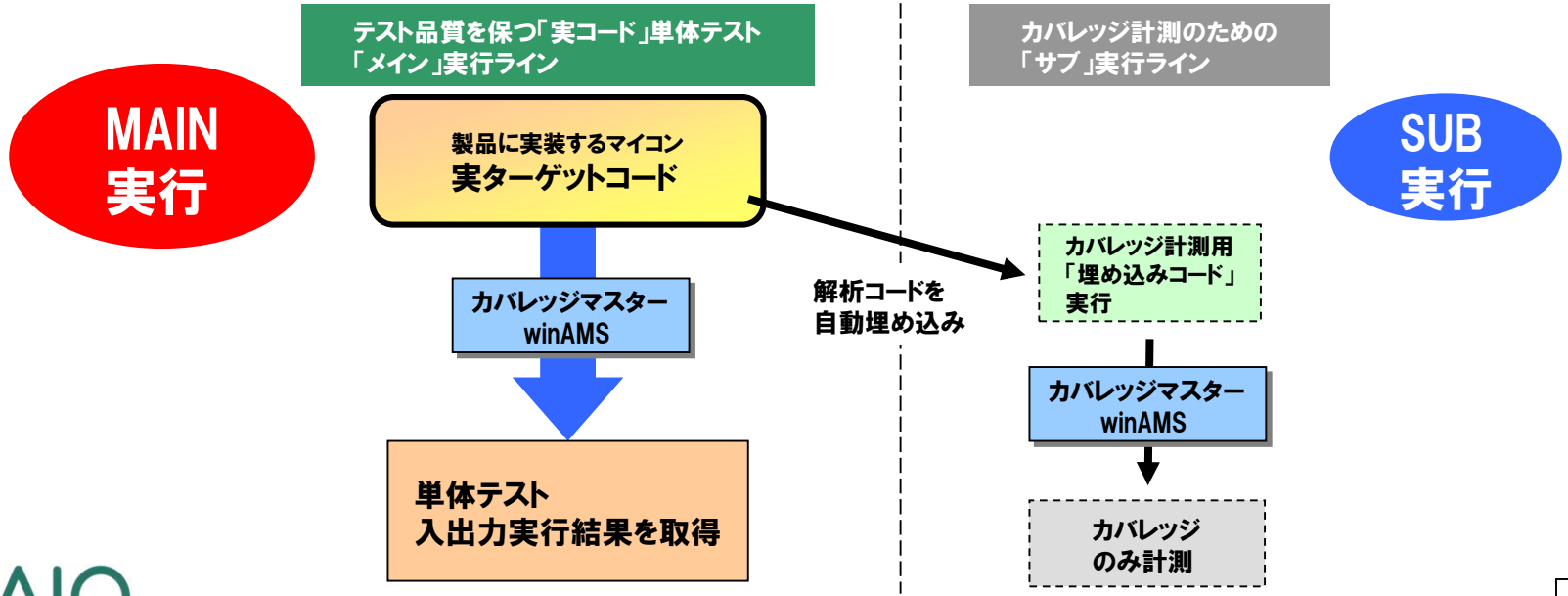


埋め込みコードを利用した
カバレッジ測定機能

GAIO TECHNOLOGY

実コード計測と埋め込みコード計測を併用

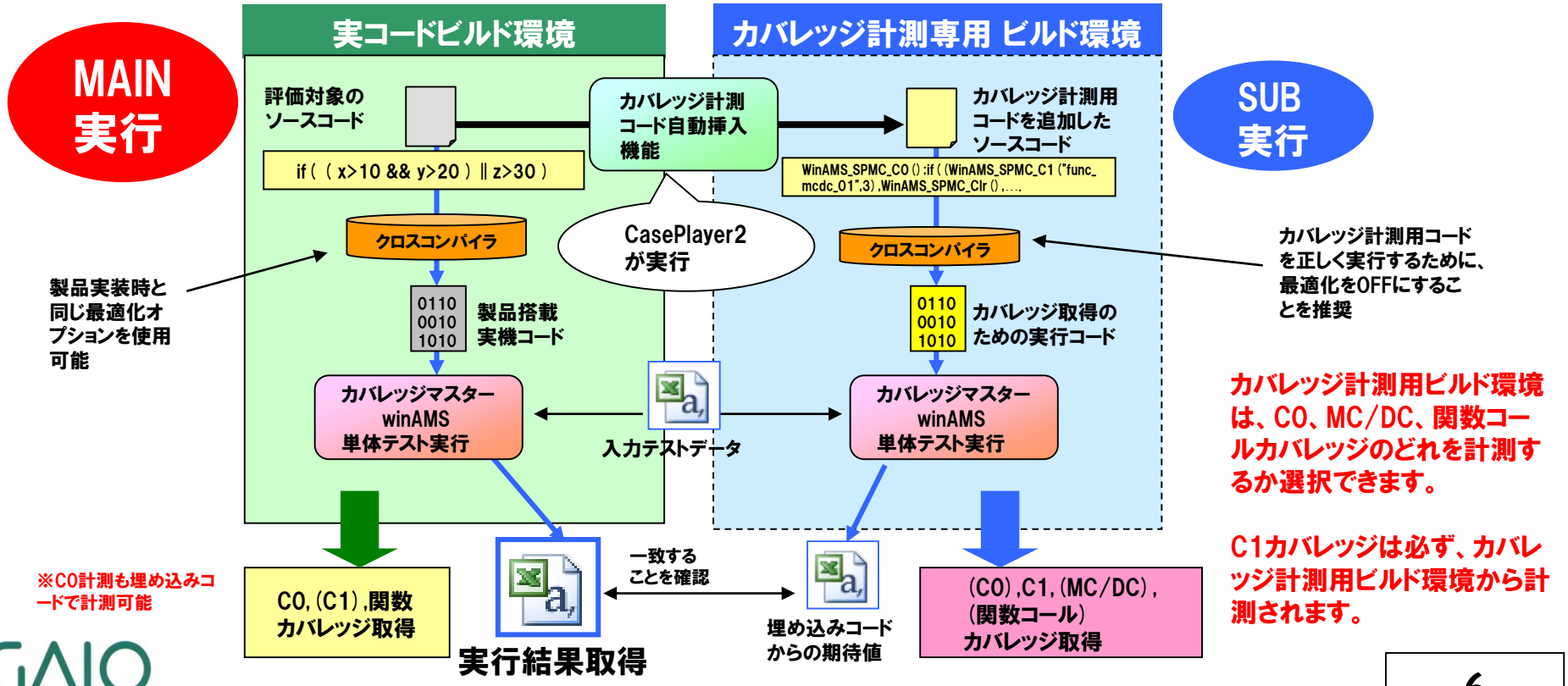
- **カバレッジ計測のために フックコードを埋め込み カバレッジ計測実行**
  - MC/DC、関数コールカバレッジはオプション機能
- **最適化やマシン語展開によるカバレッジ計測不足の影響なし**
- **カバレッジのみ「埋め込みコード」から計測**
  実行結果（期待値比較）は「実コード」から取得し評価 ←重要！
  - 実行結果は実コードから取得することで テスト品質を保つ



埋め込みコードによるカバレッジ計測機能：詳細処理フロー

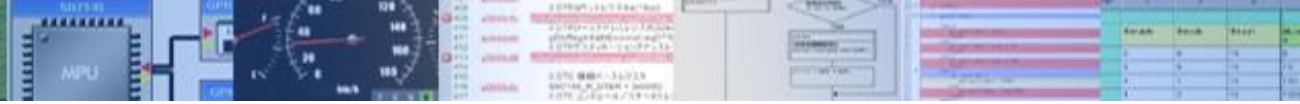
■「実コード」と「埋め込みコード」を並列実行

- カバレッジ計測用（フック）コードを自動挿入し実行
- 入出力テスト、期待値評価には」、製品実装と同じ「実コード」から取得



カバレッジ計測用ビルド環境は、CO、MC/DC、関数コールカバレッジのどれを計測するか選択できます。

C1カバレッジは必ず、カバレッジ計測用ビルド環境から計測されます。

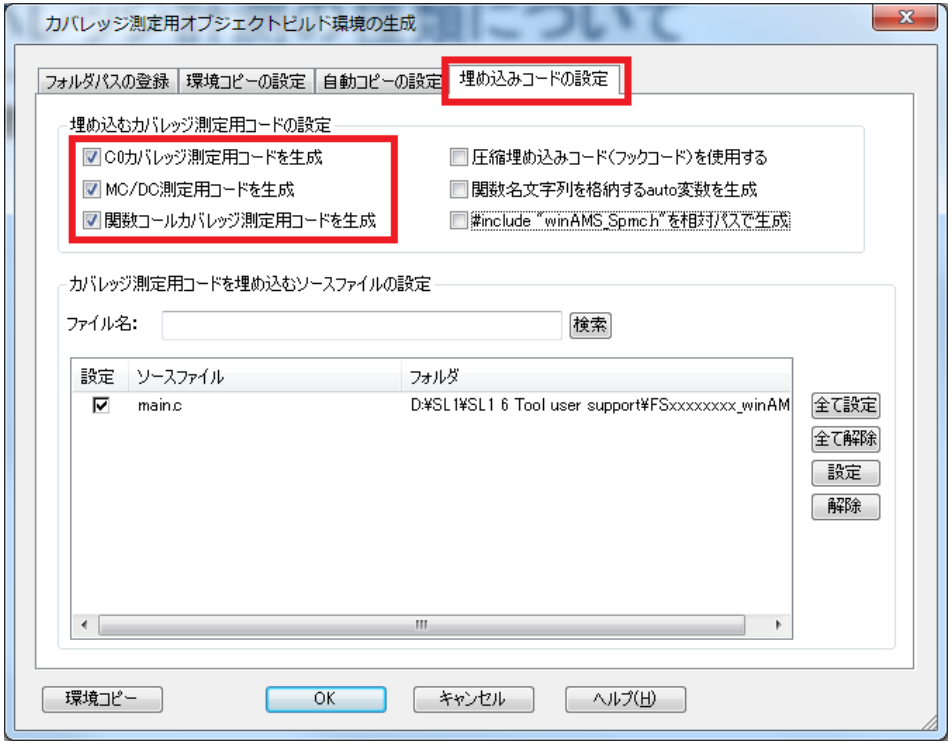


埋め込みコードを利用して計測できる
カバレッジ計測の種類について

GAIO TECHNOLOGY

カバレッジ計測の種類について

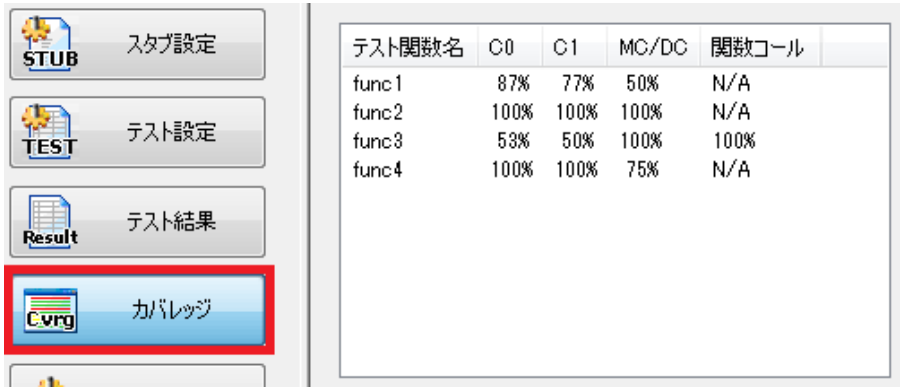
- 「埋め込みコードの設定」で、取得するカバレッジを選択
  - C0,MC/DC（オプション機能）,関数コール（オプション機能）カバレッジ
  - C1カバレッジは、デフォルト計測され、選択不可
  - C0カバレッジが未選択の場合は、実コードビルド環境からカバレッジ計測



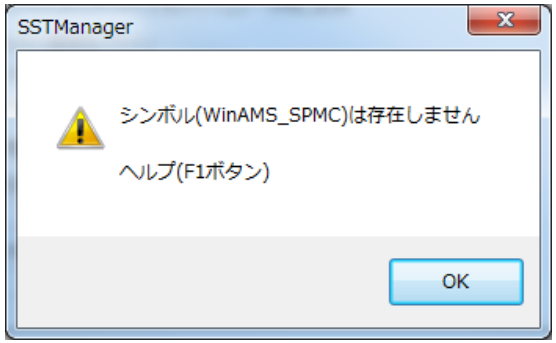
カバレッジ計測が正しく実行されない例

カバレッジ計測が正しく実行されない例

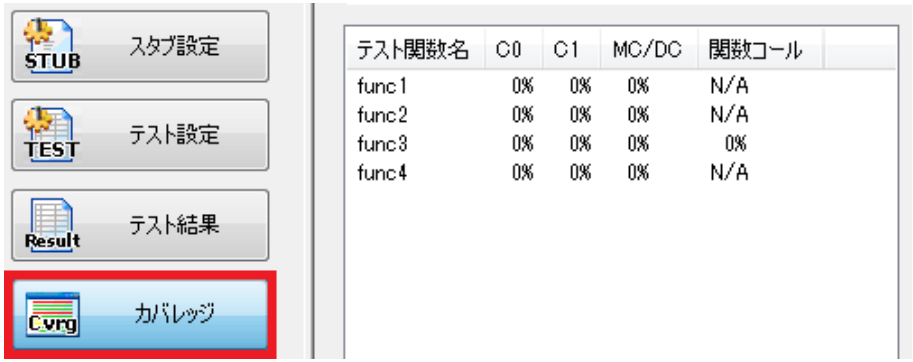
■ 正常にカバレッジ計測できる場合



■ 正常にカバレッジ計測できない場合



「起動設定」-「実行するオブジェクトの指定」-「カバレッジ測定用オブジェクトファイル」にカバレッジ計測専用 ビルド環境のオブジェクトを指定していない場合や、環境作成の手順間違いの場合



カバレッジ計測用関数が参照しているヘッダファイルの定義を使用しているマイコンに合わせて調整していない場合など

カバレッジ計測されない原因概要

カバレッジ計測されない原因概要①

■ 原因は様々、ほとんどの原因は下記①に起因

① カバレッジ計測用関数で使用するテスト対象関数名若しくは、カバレッジ測定用情報ファイルパス名が正しく渡らない場合
※C1,MC/DCカバレッジともに0%

A) マイコンの特性（ミラー機能など）により、文字列型定数のアドレスが正しく渡らない場合

- 関連MPU：
  FMC16LX, FMC16FX, RL78(S2,S3コア), C2000(C28x), M16C,
  TX03(ARM Cortex-M3)

B) テスト対象アプリの初期化が正しく実施されず、ベースアドレスが不正で、文字列型定数のアドレスが正しく渡らない場合

- 関連MPU：
  e200z710/e200z410/e200z425, V850E2M, RH850

② テスト対象関数の実行中に、カバレッジ計測用変数が初期化される場合
※C1カバレッジは計測できるが、MC/DCカバレッジのみ0%

GAIO TECHNOLOGY

カバレッジ計測されない原因概要②

■ 原因は様々、ほとんどの原因は前頁①に起因

③ マイコン、コンパイラ特有の問題

A) LSB/MSBの指定違いに依って、正しい条件分岐ができない場合
※C1カバレッジは計測できるが、MC/DCカバレッジのみ0%

- 関連MPU(コンパイラ):
  V850(GHS), V850E2M(GHS)

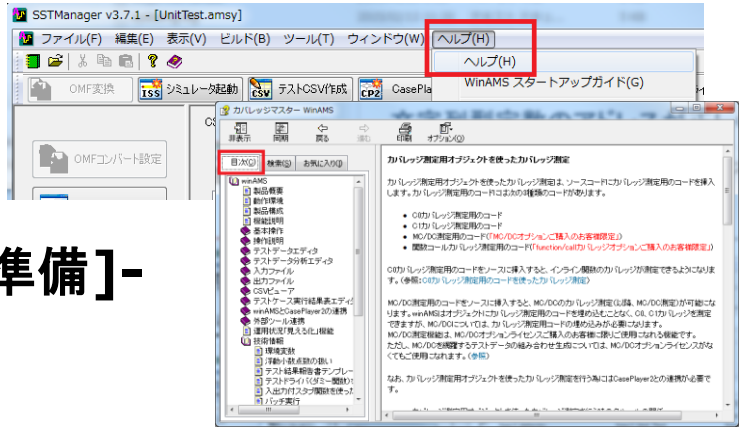
B) 最適化の影響で、実コードビルド環境のカバレッジ計測ができない場合
※C0カバレッジが0%

- 関連MPU(コンパイラ):
  R32C(Hew)

【詳細①A】
文字列型定数のアドレスが
正しく渡らない場合

アドレスが正しく渡らない場合【参考①】

- **文字列型定数の詳細については、カバレッジマスターwinAMSのヘルプ(※)をご参照下さい**
  ※[winAMS]-[技術情報]-[カバレッジ測定用オブジェクトを使ったカバレッジ測定]-[カバレッジ測定用オブジェクトを使ったカバレッジ測定のための準備]-



① 圧縮埋め込みOFF
[カバレッジ測定ソース(ヘッダ)ファイル]
[カバレッジ測定用のコードにおける文字列型の変更]

② 圧縮埋め込みON
[圧縮埋め込みコードにおけるカバレッジ測定ソース(ヘッダ)ファイルと文字列型の変更]

アドレスが正しく渡らない場合【参考②】

■ カバレッジ計測用関数

① 圧縮埋め込みOFF（WinAMS_SPMC.c）

① C0計測用：BOOL WinAMS_SPMC_C0(WinAMS_SPMC_TFUNCNAME funcname,WinAMS_SPMC_U4 line)

② C1計測用：BOOL WinAMS_SPMC_C1(WinAMS_SPMC_TFUNCNAME funcname,U4 blkID)

③ MC/DC計測用：BOOL WinAMS_SPMC_Res(WinAMS_SPMC_TFUNCNAME funcname,U4 resID,BOOL res,U2 expcnt,U4 blkID)

④ 関数コール計測用：void WinAMS_SPMC_CALL(WinAMS_SPMC_TFUNCNAME funcname,WinAMS_SPMC_U4 callID)

② 圧縮埋め込みON（WinAMS_SPMC_Com.c）

① C0計測用：BOOL WinAMS_SPMC_Com(WinAMS_SPMC_TFILENAME_PTR file,U4 index)

② C1計測用：BOOL WinAMS_SPMC_Com(WinAMS_SPMC_TFILENAME_PTR file,U4 index)

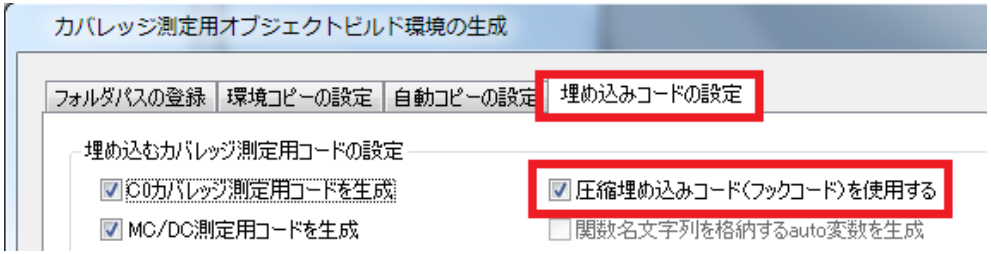
③ MC/DC計測用：BOOL WinAMS_SPMC_Res_Com(WinAMS_SPMC_TFILENAME_PTR file,U4 index,BOOL res)

④ 関数/関数コール計測用：
BOOL WinAMS_SPMC_Com(WinAMS_SPMC_TFILENAME_PTR file,U4 index)

GAIO TECHNOLOGY

文字列型定数のアドレスが正しく渡らない場合①

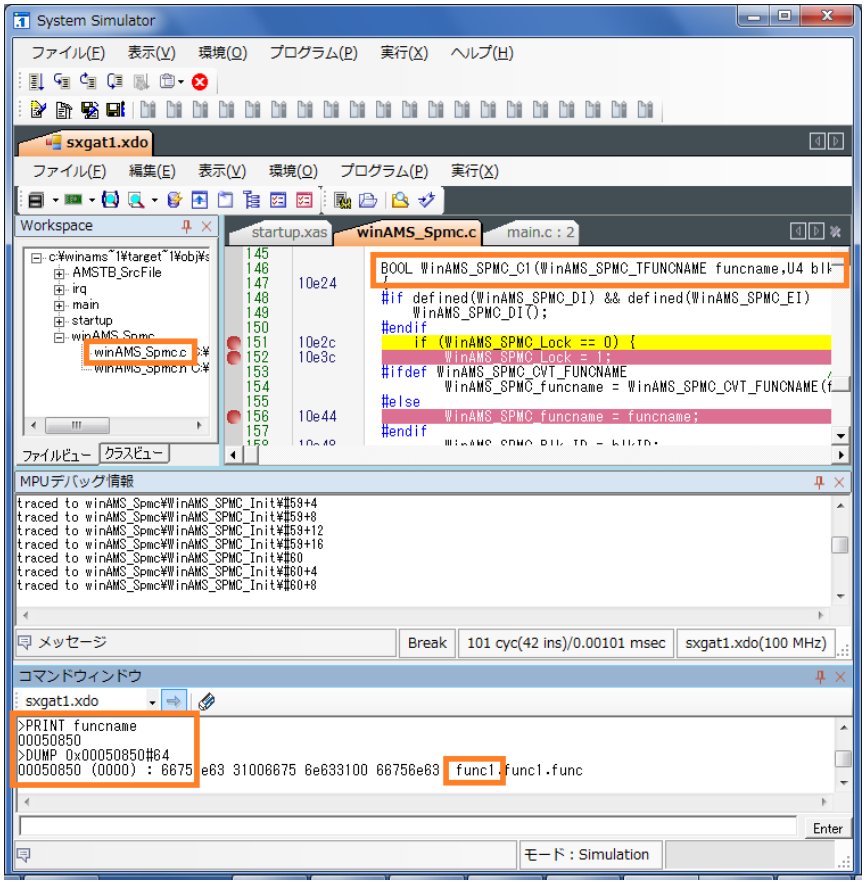
1. **文字列型定数のアドレスが正しく渡っているか確認します**
2. **アドレスが正しく渡っていない場合は、カバレッジ計測用関数が参照しているヘッダファイルを修正します**
   - 圧縮埋め込みコードのON/OFFで、関連関数（ファイル）と修正方法が変わります。
     - 圧縮埋め込みコードOFF：winAMS_Spmc.c, winAMS_Spmc.h, winAMS_SpmcDefine.h
     - 圧縮埋め込みコードON　：winAMS_Spmc_Com.c, winAMS_Spmc_Com.h, winAMS_SpmcDefine_Com.h



   - カバレッジ計測用関数が含まれるファイルは、ツールのインストールフォルダ(テンプレート)から、カバレッジ計測専用 ビルド環境にコピーされます。カバレッジ計測専用 ビルド環境作成の度にファイルの修正をしない場合は、テンプレートの変更が必要になります。
     例）C:¥Program Files¥gaio¥CasePlayer2¥template¥winAMS_Spmc.c
     C:¥Program Files¥gaio¥CasePlayer2¥template¥winAMS_Spmc.h
     C:¥Program Files¥gaio¥CasePlayer2¥template¥winAMS_SpmcDefine.h
3. **上記のファイルを使用して、カバレッジ計測用オブジェクトを再作成します**

文字列型定数のアドレスが正しく渡らない場合②

■ アドレスの確認方法（圧縮埋め込みOFF【デフォルト】の場合）



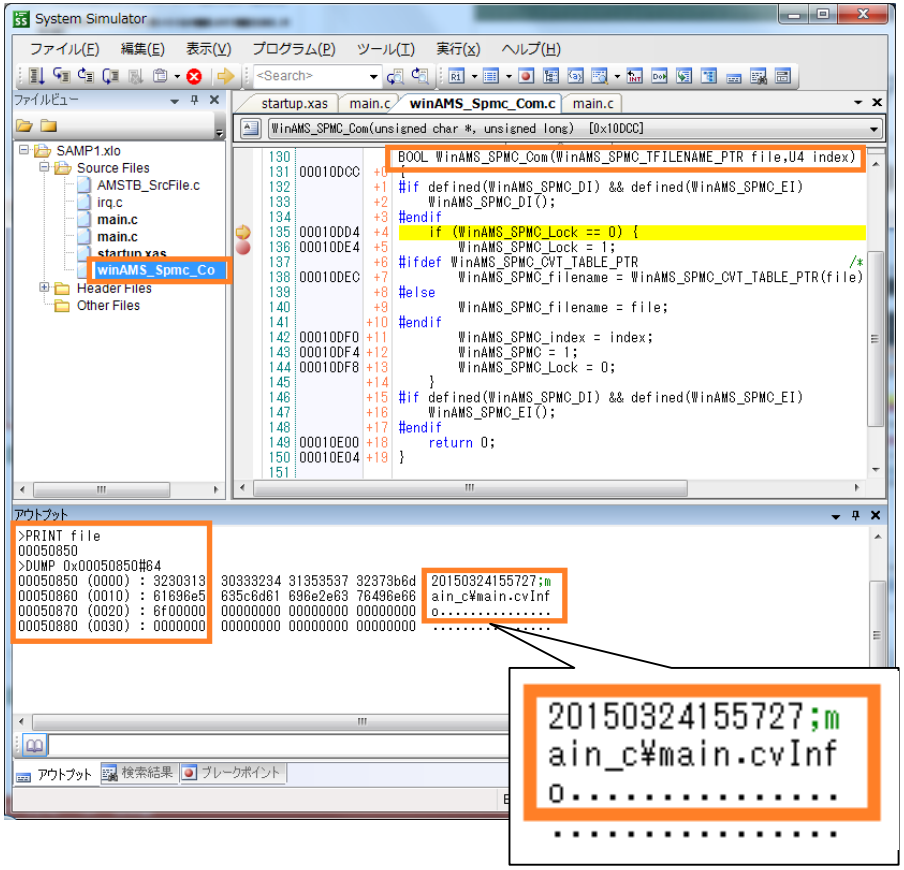
◆カバレッジ計測用関数にテスト対象関数名のアドレスが正常に設定されている かの確認方法

1. カバレッジ測定用オブジェクトで、任意の関数（例：func1）のテストを実行
2. C1計測用の関数（WinAMS_SPMC_C1）にブレークポイント（例：151行目）を設定して、そこまで実行
3. コマンドウィンドウで、「print funcname」を実行し第1引数のアドレス確認
4. 上記3.で確認したアドレスを使って「dump」実行
5. コマンドウィンドウの右側にテスト対象関数名が表示される
   ※表示されなければ、関数名が正常に渡っていない事になる

例)
>print funcname
00050850
>dump 0x00050850#32(or 64, or 128)
00050850(0000):xxx xxx func1.func1.func1

文字列型定数のアドレスが正しく渡らない場合③

■ アドレスの確認方法（圧縮埋め込みONの場合）



◆カバレッジ計測用関数にテスト対象関数名のアドレスが正常に設定されている かの確認方法

1. カバレッジ測定用オブジェクトで、任意の関数（例：func1）のテストを実行
2. カバレッジ計測用の関数（WinAMS_SPMC_Com）にブレークポイント（例：135行目）を設定して、そこまで実行
3. コマンドウィンドウで、「print file」を実行し第1引数のアドレス確認
4. 上記3.で確認したアドレスを使って「dump」実行
5. コマンドウィンドウの右側にカバレッジ測定用情報ファイルパス名（YYYYMMDDHHMM;ファイル名_c¥ファイル名.cvInf）が表示される
   ※表示されなければ、カバレッジ測定用情報ファイルパス名が正常に渡っていない事になる

例)
>print file
00050850
>dump 0x00050850#64(or 32, or 128)
00050850(0000):xxx xxx 20150324155727;m
00050860(0010):xxx xxx ain_c¥main.cvInf
00050870(0020):xxx xxx o

【詳細①A】
文字列型定数のアドレスが
正しく渡らない場合
マイコン：FMC16LX

マイコン：FMC16LX ①

■ 文字列型定数のアドレスが正しく渡らない原因

① スモールモデル若しくはミディアムモデルの場合、カバレッジ計測用関数の第1引数の文字列型定数のアドレスには（A）ROMミラー領域の論理アドレス（0x8000～0xFFFF）が渡る

② 文字列型定数の実体は（B）ROM領域の物理アドレス（0xFF8000～0xFFFFFF）に配置されている

③ カバレッジ計測用関数はデフォルト設定では、上記（A）を参照している為、計測対象の関数名が解析できない

■ 対応策

① カバレッジ計測専用ビルド環境に含まれるカバレッジ計測用関数の定義を変更する

② 上記でコンパイルしたカバレッジ計測用オブジェクトを利用する。

マイコン：FMC16LX ②

■ 対応策（続き①）

① 圧縮埋め込みコードOFF
「winAMS_SpmcDefine.h」を手動修正（赤文字＝デフォルト設定から変更する行）

```
/*【Reference】Add 関数引数の型にconst追加(winAMS_Spmc.hが参照) */
/*【Reference】Add 問題が解決しない場合は、constを追加しない(SW_WinAMS_SPMC_const_funcname 0) */
#define SW_WinAMS_SPMC_const_funcname 1
#if SW_WinAMS_SPMC_const_funcname
#define WinAMS_SPMC_const_funcname 1
#endif

/* A user defines the type of the string of the function name */
/*【Reference】Mod WINAMS_SPMC_USR_DEF_TFUNCNAME を0 から 1に変更*/
#define WINAMS_SPMC_USR_DEF_TFUNCNAME 1                    /* 0:not define, 1:define */
#if WINAMS_SPMC_USR_DEF_TFUNCNAME

/* example */
/*【Reference】Mod WinAMS_SPMC_BASE_TFUNCNAME を char から char __far に変更*/
//#define WinAMS_SPMC_BASE_TFUNCNAME char                  /* base type=char */
/* #define WinAMS_SPMC_BASE_TFUNCNAME signed char */       /* base type=signed char */
/* #define WinAMS_SPMC_BASE_TFUNCNAME unsigned char */     /* base type=unsigned char */
#define WinAMS_SPMC_BASE_TFUNCNAME char __far              /* base type=char __far */

/*【Reference】Mod fname を ROMミラーリング機能に合わせて変更 */
//#define WinAMS_SPMC_CVT_FUNCNAME(fname)  (fname)         /* funcname pointer convert */
#define WinAMS_SPMC_CVT_FUNCNAME(fname) ((WinAMS_SPMC_TFUNCNAME)((unsigned long)(fname) | 0xff0000))
```


sample1

マイコン：FMC16LX ③

■ 対応策（続き②）

② 圧縮埋め込みコードON
「winAMS_SpmcDefine_Com.h」を手動修正（赤文字＝デフォルト設定から変更する行）

```
/* A user defines the type of the string of the file name */
/*【Reference】Mod WinAMS_SPMC_BASE_TFILENAME を char から char __far に変更*/
//#define WinAMS_SPMC_BASE_TFILENAME char                    /* base type=char */
/* #define WinAMS_SPMC_BASE_TFILENAME signed char */        /* base type=signed char */
/* #define WinAMS_SPMC_BASE_TFILENAME unsigned char */      /* base type=unsigned char */
/*【Reference】Add WinAMS_SPMC_BASE_TFUNCNAME を char から char __far に変更*/
#define WinAMS_SPMC_BASE_TFILENAME char __far              /* base type=char __far */

#define WinAMS_SPMC_TABLE_PTR_TATTR1                       /* type tatt1 *name */
#define WinAMS_SPMC_TABLE_PTR_TATTR2                       /* type *tatt2 name */
/* #define WinAMS_SPMC_TABLE_PTR_TATTR1 far */             /* example */
#define WinAMS_SPMC_TABLE_TATTR                            /* type tatt name[] */
/* #define WinAMS_SPMC_TABLE_TATTR far */                 /* example */

/*【Reference】Mod fname を ROMミラーリング機能に合わせて変更 */
//#define WinAMS_SPMC_CVT_TABLE_PTR(fname) (fname)         /* table pointer convert */
#define WinAMS_SPMC_CVT_TABLE_PTR(fname) ((WinAMS_SPMC_TFILENAME_PTR)((unsigned long)(fname) |
0xff0000))          /* example */

#define WinAMS_SPMC_CONST const /* const table */
/* #define WinAMS_SPMC_CONST */ /* not const table */
```


sample2

【詳細①A】
文字列型定数のアドレスが
正しく渡らない場合
マイコン：FMC16FX

マイコン：FMC16FX ①

■ 文字列型定数のアドレスが正しく渡らない原因

① スモールモデル若しくはミディアムモデルの場合、カバレッジ計測用関数の第1引数の文字列型定数のアドレスには（A）ROMミラー領域の論理アドレス（0x8000～0xFFFF）が渡る

② 文字列型定数の実体は（B）ROM領域の物理アドレス（※0xFn8000～0xFnFFFF）に配置されている

※1.n=0x0~0xF, ROMミラー機能選択レジスタ(ROMM)のミラーリングバンク選択ビット(bit7～bit4)によって、バンクの値が変わる

※ROMミラー機能を利用していない場合（ROM ミラー機能許可（bit0＝0））は、上記1.の限りではありません。マップファイルを参照して、const領域の先頭1バイトのアドレスを確認する必要があります。（右の例だと、0xFE）

マップファイルサンプル（*.mp1）

```
00000180-0000018F  00000010  DATA  N RW-- 01 ABS  Register Bank No. 00
                              ~ 省略 ~
00DF2002-.....  00000000  CONST P R--I 02 REL  DCLEAR
00FE0000-00FE0403  00000404  CONST P R--I 02 REL  CONST
00FF0000-00FF0FE8  00000FE9  CODE  P R-XI 01 REL  CODE
```

③ カバレッジ計測用関数はデフォルト設定では、上記（A）を参照している為、計測対象の関数名が解析できない

■ 対応策

① カバレッジ計測専用ビルド環境に含まれるカバレッジ計測用関数の定義を変更する

② 上記でコンパイルしたカバレッジ計測用オブジェクトを利用する。

マイコン：FMC16FX ②

■ 対応策（続き①）

① 圧縮埋め込みコードOFF
「winAMS_SpmcDefine.h」を手動修正（赤文字＝デフォルト設定から変更する行）

```
/*【Reference】Add 関数引数の型にconst追加(winAMS_Spmc.hが参照) */
/*【Reference】Add 問題が解決しない場合は、constを追加しない(SW_WinAMS_SPMC_const_funcname 0) */
#define SW_WinAMS_SPMC_const_funcname 1
#if SW_WinAMS_SPMC_const_funcname
#define WinAMS_SPMC_const_funcname 1
#endif

/* A user defines the type of the string of the function name */
/*【Reference】Mod WINAMS_SPMC_USR_DEF_TFUNCNAME を0 から 1に変更*/
#define WINAMS_SPMC_USR_DEF_TFUNCNAME 1                    /* 0:not define, 1:define */
#if WINAMS_SPMC_USR_DEF_TFUNCNAME

/* example */
/*【Reference】Mod WinAMS_SPMC_BASE_TFUNCNAME を char から char __far に変更*/
//#define WinAMS_SPMC_BASE_TFUNCNAME char                  /* base type=char */
/* #define WinAMS_SPMC_BASE_TFUNCNAME signed char */       /* base type=signed char */
/* #define WinAMS_SPMC_BASE_TFUNCNAME unsigned char */     /* base type=unsigned char */
#define WinAMS_SPMC_BASE_TFUNCNAME char __far              /* base type=char __far */

/*【Reference】Mod fname を ROMミラーリング機能に合わせて変更 */
//#define WinAMS_SPMC_CVT_FUNCNAME(fname)  (fname)         /* funcname pointer convert */
/*【Reference】Mod ROMミラー機能選択レジスタ(ROMM)のミラーリングバンク選択ビット(bit7～bit4)によって、下記の */
/* 0xff0000のff部分がf0～ffに変わります。環境に合わせて変更してください。（下記は、ミラーリングバンク選択ビットが */
/*「1111」(FFhバンク)）の場合 */
#define WinAMS_SPMC_CVT_FUNCNAME(fname) ((WinAMS_SPMC_TFUNCNAME)((unsigned long)(fname) | 0xff0000))
```

マイコン：FMC16FX ③

■ 対応策（続き②）

② 圧縮埋め込みコードON
「winAMS_SpmcDefine_Com.h」を手動修正（赤文字＝デフォルト設定から変更する行）

```
/* A user defines the type of the string of the file name */
/*【Reference】Mod WinAMS_SPMC_BASE_TFILENAME を char から char __far に変更*/
//#define WinAMS_SPMC_BASE_TFILENAME char                    /* base type=char */
/* #define WinAMS_SPMC_BASE_TFILENAME signed char */        /* base type=signed char */
/* #define WinAMS_SPMC_BASE_TFILENAME unsigned char */      /* base type=unsigned char */
/*【Reference】Add WinAMS_SPMC_BASE_TFUNCNAME を char から char __far に変更*/
#define WinAMS_SPMC_BASE_TFILENAME char __far              /* base type=char __far */

#define WinAMS_SPMC_TABLE_PTR_TATTR1                       /* type tatt1 *name */
#define WinAMS_SPMC_TABLE_PTR_TATTR2                       /* type *tatt2 name */
/* #define WinAMS_SPMC_TABLE_PTR_TATTR1 far */             /* example */
#define WinAMS_SPMC_TABLE_TATTR                            /* type tatt name[] */
/* #define WinAMS_SPMC_TABLE_TATTR far */                  /* example */

/*【Reference】Mod fname を ROMミラーリング機能に合わせて変更 */
//#define WinAMS_SPMC_CVT_TABLE_PTR(fname) (fname)         /* table pointer convert */
/*【Reference】Mod ROMミラー機能選択レジスタ(ROMM)のミラーリングバンク選択ビット(bit7～bit4)によって、下記の */
/* 0xff0000のff部分がf0～ffに変わります。環境に合わせて変更してください。（下記は、ミラーリングバンク選択ビットが */
/*「1111」(FFhバンク) ）の場合 */
#define WinAMS_SPMC_CVT_TABLE_PTR(fname) ((WinAMS_SPMC_TFILENAME_PTR)((unsigned long)(fname) |
0xff0000))          /* example */

#define WinAMS_SPMC_CONST const /* const table */
/* #define WinAMS_SPMC_CONST */ /* not const table */
```

【詳細①A】
文字列型定数のアドレスが
正しく渡らない場合
マイコン：RL78（S2,S3コア）

マイコン：RL78（S2,S3コア）①

■ 文字列型定数のアドレスが正しく渡らない原因

① スモールモデル若しくはミディアムモデルの場合、カバレッジ計測用関数の第1引数の文字列型定数のアドレスには（A）ミラー先領域（F0x0000～0xFFFFF）の下位16bitアドレス（0x0000～0xFFFF）が渡る

② 文字列型定数の実体は（B）ミラー元領域のアドレス（※0xn0000～0xnFFFF）に配置されている
※S2,S3コアの場合、n=0x0~0x1,プロセッサ・モード・コントロール・レジスタ（PMC）のMAA(LSB:0ビット)によって、ミラー元領域が変わる
※S1コアは、MAAが0固定の為、ミラー元領域アドレスは、0x0000～0x5EFFで、ミラー先域アドレスが、0xF8000～0xFDEFF固定になります。

③ カバレッジ計測用関数はデフォルト設定では、上記（A）の下位16bitアドレスを参照している為、MAA=1の場合に、計測対象の関数名が解析できない
※MAA=0の場合は、対策しなくても、計測対象の関数名が解析

■ 対応策

① カバレッジ計測専用ビルド環境に含まれるカバレッジ計測用関数の定義を変更する

② 上記でコンパイルしたカバレッジ計測用オブジェクトを利用する。

マイコン：RL78（S2,S3コア）②

■ 対応策（続き①）

① 圧縮埋め込みコードOFF
「winAMS_Spmc.c」を手動修正（赤文字＝デフォルト設定から変更する行）

```
#ifdef WinAMS_SPMC_const_funcname
WinAMS_SPMC_BASE_TFUNCNAME const *volatile WinAMS_SPMC_funcname;
#else
/*【Reference】Mod WinAMS_SPMC_funcname を long にして 3byteのポインタを受けれるように変更*/
//WinAMS_SPMC_BASE_TFUNCNAME *volatile WinAMS_SPMC_funcname;
unsigned long volatile WinAMS_SPMC_funcname;
#endif /* WinAMS_SPMC_const_funcname */
```


sample5

「winAMS_SpmcDefine.h」を手動修正（赤文字＝デフォルト設定から変更する行）

```
/*【Reference】Mod WINAMS_SPMC_USR_DEF_TFUNCNAME を0 から 1に変更*/
#define WINAMS_SPMC_USR_DEF_TFUNCNAME 1                 /* 0:not define, 1:define */
#if WINAMS_SPMC_USR_DEF_TFUNCNAME
/* example */
#define WinAMS_SPMC_BASE_TFUNCNAME char                 /* base type=char */
/* #define WinAMS_SPMC_BASE_TFUNCNAME signed char */    /* base type=signed char */
/* #define WinAMS_SPMC_BASE_TFUNCNAME unsigned char */  /* base type=unsigned char */
/* #define WinAMS_SPMC_BASE_TFUNCNAME char __far */     /* base type=char __far */

/*【Reference】Mod fname をミラーリング機能(PMC = 1 の場合)に合わせて変更 */
//#define WinAMS_SPMC_CVT_FUNCNAME(fname) (fname)       /* funcname pointer convert */
#define WinAMS_SPMC_CVT_FUNCNAME(fname) ((unsigned long)(fname) & 0x00ffff | 0x010000)
```


sample6

マイコン：RL78（S2,S3コア）③

■ 対応策（続き②）

② 圧縮埋め込みコードON
「winAMS_Spmc_com.c」を手動修正（赤文字＝デフォルト設定から変更する行）


sample7

```
#if __COMPILER_FCC907__
volatile WinAMS_SPMC_CONST WinAMS_SPMC_BASE_TFILENAME
WinAMS_SPMC_TABLE_PTR_TATTR1*WinAMS_SPMC_TABLE_PTR_TATTR2 WinAMS_SPMC_filename;
#else
/*【Reference】Mod WinAMS_SPMC_funcname を long にして 3byteのポインタを受けれるように変更*/
/* WinAMS_SPMC_CONST WinAMS_SPMC_BASE_TFILENAME
WinAMS_SPMC_TABLE_PTR_TATTR1*WinAMS_SPMC_TABLE_PTR_TATTR2 volatile WinAMS_SPMC_filename; */
unsigned long      volatile WinAMS_SPMC_funcname;
//volatile long WinAMS_SPMC_filename; ↑上記で問題が解決しない場合は、こちらを利用する
#endif /* __COMPILER_FCC907__ */
```

「winAMS_SpmcDefine_com.h」を手動修正（赤文字＝デフォルト設定から変更する行）


sample8

```
/*【Reference】Mod fname をミラーリング機能(PMC = 1 の場合)に合わせて変更 */
//#define WinAMS_SPMC_CVT_TABLE_PTR(fname) (fname)                    /* table pointer convert */
#define WinAMS_SPMC_CVT_TABLE_PTR(fname) ((unsigned long)(fname) & 0x00ffff | 0x010000)
```

【詳細①A】
文字列型定数のアドレスが
正しく渡らない場合
マイコン：M16C

マイコン：M16C ①

■ 文字列型定数のアドレスが正しく渡らない原因

① 圧縮埋め込みコードがOFFの場合、カバレッジ計測用関数の第1引数の型が char *型で near pointerとして、アドレス（A）が渡る
※圧縮埋め込みコードがONの場合、第1引数の型が const char *型で far pointer

② 文字列型定数の実体は異なるアドレス（B）（0xcnnnn～0xcnnnn）に配置されている

③ カバレッジ計測用関数はデフォルト設定では、上記（A）を参照している為、計測対象の関数名が解析できない

■ 対応策

① カバレッジ計測専用ビルド環境に含まれるカバレッジ計測用関数の定義を変更する

② 上記でコンパイルしたカバレッジ計測用オブジェクトを利用する。

GAIO TECHNOLOGY

マイコン：M16C ②

■ 対応策（続き①）

① 圧縮埋め込みコードOFF
「winAMS_Spmc.h」を手動修正（赤文字＝デフォルト設定から変更する行）

```
/*【Reference】Add WinAMS_SPMC_const_funcname定義 near pointer から far pointer へ */
#define WinAMS_SPMC_const_funcname

#if WINAMS_SPMC_USR_DEF_TFUNCNAME
#ifdef WinAMS_SPMC_const_funcname
typedef const WinAMS_SPMC_BASE_TFUNCNAME * WinAMS_SPMC_TFUNCNAME;
#else
typedef WinAMS_SPMC_BASE_TFUNCNAME * WinAMS_SPMC_TFUNCNAME;
#endif /* WinAMS_SPMC_const_funcname */
#else
#ifdef WinAMS_SPMC_const_funcname
typedef const char * WinAMS_SPMC_TFUNCNAME;     /*【Reference】near pointer から far pointer へ */
#else
typedef char * WinAMS_SPMC_TFUNCNAME;
#endif /* WinAMS_SPMC_const_funcname */
#endif /* WINAMS_SPMC_USR_DEF_TFUNCNAME */
```

sample10

【詳細①A】
文字列型定数のアドレスが
正しく渡らない場合
マイコン：**TX03**（ARM Cortex-M3）

マイコン:TX03(ARM Cortex-M3) ①

■ 文字列型定数のアドレスが正しく渡らない原因

① カバレッジ計測用関数の第1引数の文字列型定数のアドレスには(A)RAM領域のアドレス(0xFFnnnn～0xFFnnnn)が渡る

② 文字列型定数の実体はROM領域のアドレス(0xnnnn～0xnnnn)に配置されている

③ カバレッジ計測用関数はデフォルト設定では、上記(A)を参照している為、計測対象の関数名が解析できない

■ 対応策

① スタートアップコマンドファイルで、ROM領域のデータをRAM領域にコピーする
※シミュレータには、ミラー処理が実装されていない。
カバレッジ計測とは無関係なconst変数の参照時の問題回避も考慮して、コピー実施

■スタートアップコマンドファイル(イメージ)

; メモリ領域 0x1000～0x10ffの内容を0xFF2000～0xFF20ffへコピー
COPY MEMORY 0x1000#0x100 0xFF2000

【詳細①B】
文字列型定数のアドレスが
正しく渡らない場合
マイコン：e200zxxx系

マイコン:e200zxxx系 ①

■ 文字列型定数のアドレスが正しく渡らない原因①

① ベースレジスタ(R2【読み出し専用SDAポインタ(ROM化データ参照)】)が未設定(未初期化)で、カバレッジ計測用関数の第1引数の文字列型定数のアドレスに正しいアドレスが渡らない

② カバレッジ計測用関数はデフォルト設定では、上記アドレスを参照している為、計測対象の関数名が解析できない

■ 文字列型定数のアドレスが正しく渡らない原因②

① ベースレジスタ(R13【読み取り・書き込み可変SDAポインタ(変数データ参照)】)が未設定(未初期化)で、カバレッジ計測用変数のアドレスが不正になる

② カバレッジ計測用関数は上記変数を参照している為、計測対象の関数名が解析できない

■ 対応策

① テスト対象アプリケーションのスタートアップを実行して、ベースレジスタ(R2【読み出し専用SDAポインタ】)、(R13【読み取り・書き込み可変SDAポインタ】)を設定する。

■関連FAQ:【011_03】スタートアップルーチン、マイコン設定
http://www.gaio.co.jp/support/user/faq/winams/faq_011_03.html

【詳細①B】
文字列型定数のアドレスが
正しく渡らない場合
マイコン:V850E2M / RH850

マイコン:V850E2M/RH850 ①

■ 文字列型定数のアドレスが正しく渡らない原因

① ベースレジスタ(R4【GP:グローバル・ポインタ】)が未設定(未初期化)で、カバレッジ計測用変数のアドレスが不正になる

② カバレッジ計測用関数は上記変数を参照している為、計測対象の関数名が解析できない

■ 対応策

① テスト対象アプリケーションのスタートアップを実行して、ベースレジスタ(R4【GP:グローバル・ポインタ】/ R5【TP:テキスト・ポインタ】/ R30【EP:エレメント・ポインタ】)を設定する。

■関連FAQ:【011_03】スタートアップルーチン、マイコン設定
http://www.gaio.co.jp/support/user/faq/winams/faq_011_03.html

【詳細②】
カバレッジ計測用変数が初期化される場合

カバレッジ計測用変数が初期化される場合

■ 初期化される原因
埋め込みC1カバレッジ計測は可能。MC/DCカバレッジ計測が0%

① テストドライバから、テスト対象関数をコールしているような構造で、テスト対象関数コール前にMC/DCカバレッジ計測用の変数（※）を初期化している
※WinAMS_SPMC_maxCondCnt, WinAMS_SPMC_maxCondNest

② テストドライバから、テスト対象関数をコールしているような構造で、テストドライバ中のauto変数が大量で、スタック領域を超えて、MC/DCカバレッジ計測用の変数を破壊する

■ 対応策

① 1. 変数初期化前にMC/DCカバレッジ計測用の変数値を退避して、初期化後に値を戻してから、関数コールする
2. リンクパラメータを調整して、MC/DCカバレッジ計測用の変数値が初期化領域に配置されない埋め込みオブジェクトを生成する

② 十分なスタック領域を確保できるようにSPを設定する

GAIO TECHNOLOGY

【詳細③A】
マイコン、コンパイラ特有の問題
V850 / GHS

マイコン、コンパイラ特有の問題（V850 / GHS）

■ カバレッジ計測ができない原因
埋め込みC1カバレッジ計測は可能。MC/DCカバレッジ計測が不正結果

① ビットフィールドを使用したif文条件判定で、テストCSVから、その変数の値にデータを設定しているが、意図通りの値が設定されずに（OMFコンバート時のLSB/MSB指定間違い）、正しい実行ルートを通過しない

■ 対応策

① コンパイラのバージョンに依って、デバッグ情報のLSB/MSBの捕らえ方が変わるので、正しいマイコンとOMFコンバータオプションを指定する
※MPUが、V850E2M/RH850 のOMFコンバータは、デフォルトが「-MSB」指定で、「-LSB」を指定できない

1. GHS【R3(LSB) / R7(LSB)】　OMF変換オプション:指定なし（-LSB）
    1. コンパイルオプション:-cpu v850f　MPU:v850(GHS) / 型番:V850E/GP1シリーズ
2. GHS【R8.1.3以降（MSB）】OMF変換オプション:-MSB
    1. コンパイルオプション:-cpu v850e1f　MPU:v850(GHS) / 型番:V850E/GP1シリーズ
    2. コンパイルオプション:-cpu v850e　MPU:v850E2M(GHS) / 型番:V850E2Mシリーズ
    3. コンパイルオプション:-cpu v850e2　MPU:v850E2M(GHS) / 型番:V850E2Mシリーズ
    4. コンパイルオプション:-cpu v850e2r　MPU:v850E2M(GHS) / 型番:V850E2Mシリーズ
    5. コンパイルオプション:-cpu v850e2v3　MPU:v850E2M(GHS) / 型番:V850E2Mシリーズ

GAIO TECHNOLOGY

【詳細③B】
マイコン、コンパイラ特有の問題
R32C / NC100

マイコン、コンパイラ特有の問題（R32C / NC100）

■ カバレッジ計測ができない原因

埋め込みC1,MC/DCカバレッジ計測は可能。埋め込みなしC0カバレッジ計測が0%

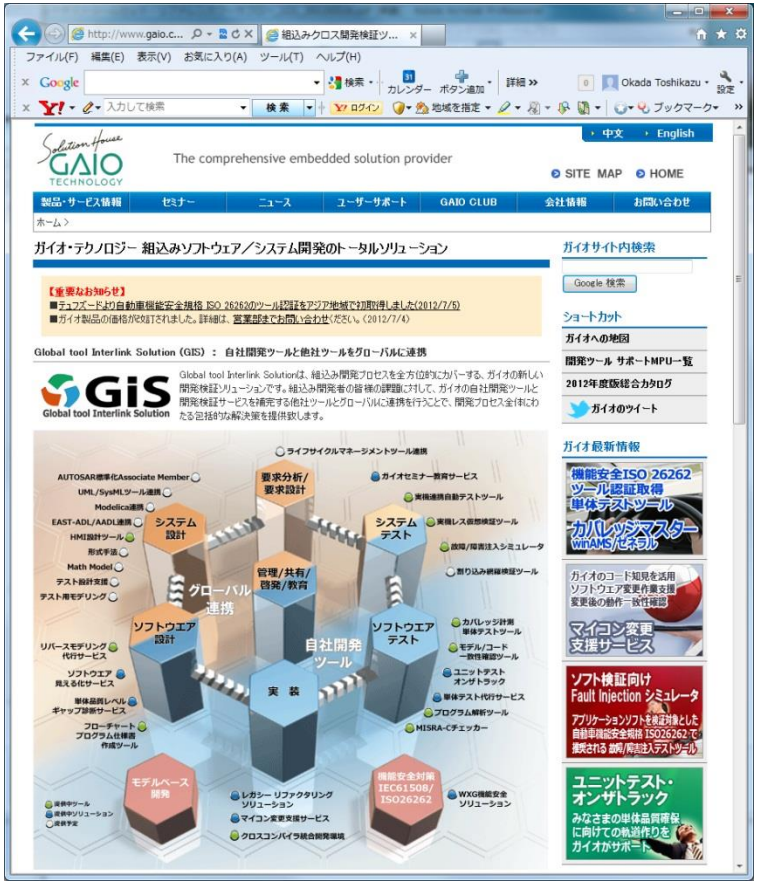
① NC100の最適化の影響で、デバッグ情報が纏まり（※）、テスト対象とは無関係な関数まで計測対象となる
※例）異なる関数名の複数の空関数が、1関数として、デバッグ情報が出力される

■ 対応策

① 1.最適化をOFFにしたオブジェクトを使用する
2.C0埋め込みONのオブジェクトを使用する

END

■ 最新情報はWEBサイトから　www.gaio.co.jp



ガイオ・テクノロジー株式会社

※会社名・商品名は各社の商標または登録商標です。
※本テキストの内容は、予告無く変更される場合があります。
※本書記載の誤りにより生ずる問題や損失に対して弊社は
　責任を負いません。
※本資料の無断転載、複写はお断りします。

ガイオ・テクノロジー株式会社
営業本部　サポート部
〒140-0002
東京都品川区東品川2-2-4　天王洲ファーストタワー25階

ご質問はユーザーサポート 窓口まで
http://www.gaio.co.jp/support/support_entry.html

GAIO TECHNOLOGY