```
;#############################################################################################################################
;#  シミュレータコマンド(マクロ)サンプル【v1.4.2】
;#  Date  ：2012/08/03
;#  UpDate：2014/05/21
;#  UpDate：2014/06/13
;#  UpDate：2014/11/04
;#  UpDate：2016/10/05
;#
;#  Copyright(c) 2012-2016 GAIO TECHNOLOGY CO., LTD. All Rights Reserved
;#############################################################################################################################
; スタートアップコマンドファイルに記述できるコマンドの規格、解説については、下記からご確認下さい。
; [ヘルプ]→[シミュレータマニュアル]→[目次]→[XAIL-Debugger使用手引き]→[デバッガの機能]
;              →[コマンド/メッセージ]→[デバッガカーネルのコマンド/メッセージ]→[SystemG]→[デバッガコマンド一覧]
;   [コマンドの規格] / [コマンド解説]
; 「;」以降の記述はコメントとして扱う事ができます
;-----------------------------------------------------------------------------------------------------------------------------
start log/all                                               ; ユーザがコマンドウィンドウから実行したデバッガコマン
                                                            ; ドおよびウィンドウに表示されるすべてのデータをログフ
                                                            ; ァイルへ書き込みます
                                                            ; シュミレーション実行後のログファイルは、カバレッジマ
                                                            ; スターのテストプロジェクトフォルダに「systemg.log」の
                                                            ; 名称で保存されます
;-----------------------------------------------------------------------------------------------------------------------------
on error then continue                                      ; マクロ実行中にエラーが生じたときにGOコマンドとSTEPコ
                                                            ; マンドの次のデバッガコマンドを実行します
;-----------------------------------------------------------------------------------------------------------------------------
set unit/all                                                ; すべてのユニットについて、シンボル情報を設定するよう
                                                            ; に指定します
;-----------------------------------------------------------------------------------------------------------------------------
@reset                                                      ; リセット状態にします(パワーオンリセット)
;-----------------------------------------------------------------------------------------------------------------------------
; 下記 assign コマンドを使って、実機と同様のメモリレイアウトにする事が可能です
; assignコマンドの代わりに、「起動設定」画面の「全てのメモリ領域をアサイン」のチェックをONにする事で、選択したMPUの全領
; 域をread/write/execの属性にします
assign/read/write 0x04000000:0x04ffffff                     ; メモリのアサインを行います
;-----------------------------------------------------------------------------------------------------------------------------
;set reg pc = 0xffffffff                                    ; プログラムカウンタを設定します
;-----------------------------------------------------------------------------------------------------------------------------
;set mode source                                            ; ソースウィンドウにソースファイルを表
                                                            ; 示します
;-----------------------------------------------------------------------------------------------------------------------------
;set trace/subroutine=yes/display                           ; プログラムの実行トレースを実施します
                                                            ; ログファイルにも保存されます
;-----------------------------------------------------------------------------------------------------------------------------
;set output/message=all none                                ; すべてのレベルのメッセージに対して、ログファイル/ウィ
                                                            ; ンドウに出力しません
;set output/command nowindow                                ; デバッガコマンドをウィンドウへ出力しません
;-----------------------------------------------------------------------------------------------------------------------------
; 対象関数：main.c：Sample_Infinite_loop
; 指定アドレス(IRQ_COUNT)が読み込まれた時(17行目)に値を書き換えてwhile文を抜ける
;
define/address/global ADR_ADIF = 0x04000000                 ; ADR_ADIFはマクロで使用する任意の変数名
define/Global COUNT = 0                                     ; COUNTはマクロで使用する任意の変数名
;-----------------------------------------------------------------------------------------------------------------------------
; マクロ：01_1.w_adif
macro w_adif_V310                                           ; マクロ名は任意で設定
  define/Global COUNT = COUNT + 1
  IF COUNT < 2 THEN goto func_end

  ; 2回目にマクロが実行されると値のセットを行う
  ; 必要が無ければ、回数カウントの処理は入れないでも良いです
  store ADR_ADIF = 1
  define/Global COUNT = 0

  func_end:                                                 ; ラベル名は任意で設定
mend

macro w_adif_V320                                           ; マクロ名は任意で設定
  define/Global COUNT = COUNT + 1

  ; IF ～ THEN ～ ELSE[IF] ～ ENDIF 構文 は、XAIL V3.2.0以降でサポート
  ; 条件式は()で括る必要があります
  IF (COUNT >= 2) THEN

    ; 2回目にマクロが実行されると値のセットを行う
    ; 必要が無ければ、回数カウントの処理は入れないでも良いです
    store ADR_ADIF = 1
    define/Global COUNT = 0

  ELSE

  ENDIF
```

```
mend
; IRQ_COUNT(0x04000000)が読み込まれた時にマクロ(w_adif)を実行する
;set do/read=0x04000000 w_adif_V310
set do/read=0x04000000 w_adif_V320
;-------------------------------------------------------------------------------------------------------------------------
; 対象関数：main.c：Sample_Infinite_loop
; 無限ループの処理(23行目)を強制的に抜ける(PC変更)
;-------------------------------------------------------------------------------------------------------------------------
; マクロ：01_2.pass
macro pass                                                  ; マクロ名は任意で設定
  set reg pc = main¥#33                                     ; 関数内行オフセットは使用出来ません
mend
; main.c / 関数：Sample_Infinite_loop の +20行目に展開されるマシンコードの1バイト目が実行された時に
; マクロ(pass)を実行する
set do/Aexec=main¥Sample_Infinite_loop¥#+20  pass       ; XAIL V3.0.1以降(関数内行オフセット)
;-------------------------------------------------------------------------------------------------------------------------
; 対象関数：main.c：Sample_Infinite_loop
; 無限ループの処理(23行目)を強制的に抜ける(longjump)
;-------------------------------------------------------------------------------------------------------------------------
; マクロ：01_3.setjmp
macro setjmp                                                ; マクロ名は任意で設定
  set reg pc = dummy¥#13                                    ; 関数内行オフセットは使用出来ません
                                                            ; 初期設定終了アドレスは、「dummy_main」
mend
; main.c / 関数：main の +0行目に展開されるマシンコードの1バイト目が実行される前に
; マクロ(setjmp)を実行する
;set do/Bexec=main¥main¥#+0  setjmp                      ; XAIL V3.0.1以降(関数内行オフセット)

; マクロ：01_3.longjmp
macro longjmp                                               ; マクロ名は任意で設定
  set reg pc = dummy¥#31                                    ; 関数内行オフセットは使用出来ません
mend
; main.c / 関数：Sample_Infinite_loop の +20行目に展開されるマシンコードの1バイト目が実行された時に
; マクロ(longjmp)を実行する
;set do/Aexec=main¥Sample_Infinite_loop¥#+20  longjmp       ; XAIL V3.0.1以降(関数内行オフセット)
;-------------------------------------------------------------------------------------------------------------------------
; 対象関数：main.c：Sample_AutoVariables_Evacuation
; Auto変数の値をGlobal変数に退避して、テスト結果CSVに保存する
; ※Auto変数の変遷等をテスト結果CSVなどで使用したい場合に使用します。
; マイコンに依っては、Auto変数をそのまま利用できない場合もあります。
;-------------------------------------------------------------------------------------------------------------------------
; マクロ：02_1.mymacro
macro mymacro                                               ; マクロ名は任意で設定
  ; プログラムで定義されている変数(Macro_Symbol)にAuto変数(InfoPtr[0].value_a)の値を設定する
  STORE/Synchronize Macro_Symbol= InfoPtr[0].value_a
mend
; main.c / 関数：Sample_AutoVariables_Evacuation の +5行目に展開されるマシンコードの1バイト目が実行された時に
; マクロ(mymacro)を実行する
set do/Aexec=main¥Sample_AutoVariables_Evacuation¥#+5 mymacro  ; XAIL V3.0.1以降(関数内行オフセット)

; マクロ：02_2.mymacro2
macro mymacro2                                              ; マクロ名は任意で設定
  ; プログラムで定義されている変数(Macro_Symbol2)にAuto変数(InfoPtr[0].value_a)の値を設定する
  STORE/Synchronize Macro_Symbol2= InfoPtr[0].value_a
mend
; main.c / 関数：Sample_AutoVariables_Evacuation の +9行目に展開されるマシンコードの1バイト目が実行された時に
; マクロ(mymacro2)を実行する
set do/Aexec=main¥Sample_AutoVariables_Evacuation¥#+9 mymacro2 ; XAIL V3.0.1以降(関数内行オフセット)
;-------------------------------------------------------------------------------------------------------------------------
; 対象関数：main.c：Sample_Register_Evacuation
; Registerの値をGlobal変数に退避して、テスト結果CSVに保存する
; ※Registerの変遷等をテスト結果CSVなどで使用したい場合に使用します。
; 使用出来るレジスタ名はマイコンに依って異なりますので、下記からご確認下さい。
; [ヘルプ]→[シミュレータマニュアル]→[目次]→[プロセッサモデル固有の仕様]→→[SystemG]→[～プロセッサモデル]
;  →「アドレス空間・レジスタ」→「レジスタ」
;-------------------------------------------------------------------------------------------------------------------------
; マクロ：03_1.b_reg
macro b_reg                                                 ; マクロ名は任意で設定
  ; プログラムで定義されている変数(Before_Register)にR1レジスタの値を設定する
  STORE/Synchronize Before_Register= %R1
mend
; main.c / 関数：Sample_Register_Evacuation の +0行目に展開されるマシンコードの1バイト目が実行された時に
; マクロ(b_reg)を実行する
set do/Aexec=main¥Sample_Register_Evacuation¥#+0 b_reg          ; XAIL V3.0.1以降(関数内行オフセット)

; マクロ：03_2.a_reg
macro a_reg                                                 ; マクロ名は任意で設定
  ; プログラムで定義されている変数(After_Register)にR1レジスタの値を設定する
  STORE/Synchronize After_Register= %R1
mend
; main.c / 関数：Sample_Register_Evacuation の +8行目に展開されるマシンコードの1バイト目が実行された時に
; マクロ(a_reg)を実行する
```

```
set do/Aexec=main¥Sample_Register_Evacuation¥#+8 a_reg          ; XAIL V3.0.1以降(関数内行オフセット)
;-----------------------------------------------------------------------------------------------------------------
; 対象関数：main.c：Sample_Global_Set
; テストCSVで設定した変数がソースコードで書き換えられる前に退避(A)して、書き換えられた後に
; (A)の値を変数に戻す
; ※テストCSVで入力したデータを確認したい場合に使用します。
;-----------------------------------------------------------------------------------------------------------------
; マクロ：04_1.Evacuation_Variables
define/Global Evacuation = 0                                    ; Evacuationはマクロで使用する任意の変数名
macro Evacuation_Variables                                      ; マクロ名は任意で設定
  ; プログラムで定義されている変数(GlobalA)の値をEvacuationに設定する
  define/Global Evacuation = GlobalA
mend
; main.c / 関数：Sample_Global_Set の +4行目に展開されるマシンコードの1バイト目が実行された時に
; マクロ(Evacuation_Variables)を実行する
set do/Aexec=main¥Sample_Global_Set¥#+4 Evacuation_Variables    ; XAIL V3.0.1以降(関数内行オフセット)

; マクロ：04_2.Set_Variables
macro Set_Variables                                             ; マクロ名は任意で設定
  ; Evacuationの値をプログラムで定義されている変数(GlobalA)に設定する
  store GlobalA = Evacuation
mend
; main.c / 関数：Sample_Global_Set の +7行目に展開されるマシンコードの1バイト目が実行された時に
; マクロ(Set_Variables)を実行する
set do/Aexec=main¥Sample_Global_Set¥#+7 Set_Variables           ; XAIL V3.0.1以降(関数内行オフセット)
;-----------------------------------------------------------------------------------------------------------------
; 対象関数：main.c：Sample_AutoVariables_Set
; グローバル変数の値をAuto変数に設定する。
;-----------------------------------------------------------------------------------------------------------------
; マクロ：05_1.Set_AutoVariables
macro Set_AutoVariables                                         ; マクロ名は任意で設定
  ; プログラムで定義されている変数(Auto_Symbol)の値をAuto変数(InfoPtr[0].value_a)に設定する
  STORE/Synchronize InfoPtr[0].value_a = Auto_Symbol
mend
; main.c / 関数：Sample_AutoVariables_Set の +6行目に展開されるマシンコードの1バイト目が実行された時に
; マクロ(Set_AutoVariables)を実行する
set do/Aexec=main¥Sample_AutoVariables_Set¥#+6 Set_AutoVariables; XAIL V3.0.1以降(関数内行オフセット)
;-----------------------------------------------------------------------------------------------------------------
; 特定範囲のメモリ内容を、初期化する
; 書式　mem_set　初期化開始アドレス, 初期化終了アドレス
; ex) メモリ0x2000～0x3000(0x2fff迄)番地の内容を0FFhで初期化する
;-----------------------------------------------------------------------------------------------------------------
;FILL MEMORY 0x2000#0x1000 = 0xff                               ; XAIL V3.0.1以降

;-----------------------------------------------------------------------------------------------------------------
; 特定範囲のメモリ内容を、特定範囲のメモリにコピーする
; 書式　md_set　コピー元開始アドレス, コピー元終了アドレス,コピー先開始アドレス, コピー先終了アドレス
; ex) メモリ0x2000～0x3000(0x2fff迄)番地の内容を0xff2000～0xff3000(0xff2fff迄)番地へコピーする
;-----------------------------------------------------------------------------------------------------------------
;COPY MEMORY 0x2000#0x3000 0xff2000                             ; XAIL V3.0.1以降

;-----------------------------------------------------------------------------------------------------------------
;対象関数：main.c：Sample_EndFunc
; テスト対象関数途中の無限ループ(123行目)を終了させ、次のテストケースを実行します。
;06_1.TEST_ENDマクロでwhile文を終了する
;-----------------------------------------------------------------------------------------------------------------

macro TEST_END

 END TEST /FUNCTION     ; XAIL V3.3.0以降

mend

set do/AEXECUTE = main¥Sample_EndFunc¥#+4 /COUNT : 2 TEST_END;06_1.TEST_ENDマクロでwhile文を終了する
;-----------------------------------------------------------------------------------------------------------------
; マクロまたはコマンドプロシジャ内から実行される各デバッガコマンドをシミュレータのコマンドウィンドウに表示するようにし
; ます(SET VERIFYコマンドより後のコマンドから表示します)
;SET VERIFY
;-----------------------------------------------------------------------------------------------------------------
```