Rev 1.00.00

ハードウェアブレーク(PBC)機能なし MCU 用の RAM でデバッグする場合の説明

1．対象 MCU
・H8/300H、H8S シリーズ、H8SX シリーズ、SH-2 シリーズが対象になります。

2．機能
・BSC（バスステートコントローラ）による拡張 RAM でのデバッグになります。
・PBC/UBC 無しタイプの CPU 品種でもプログラムメモリが RAM の場合、C ソース／Asn ソース上に直接ソフトブレークが張れます。

3．デバッグ開始前の準備

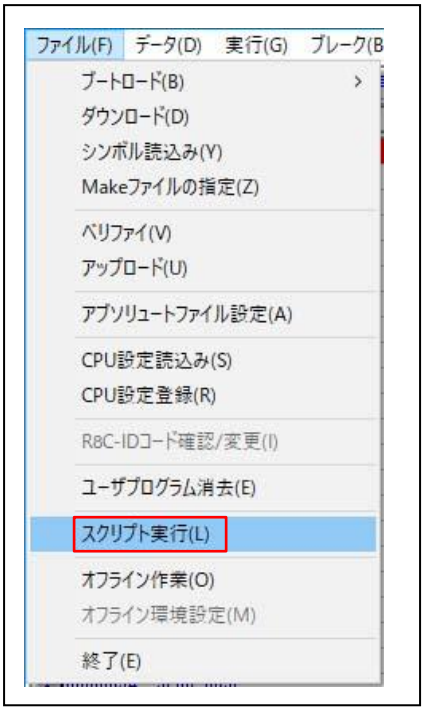
3－1）BSC（バスステートコントローラ）設定のスクリプトファイルを作成する。

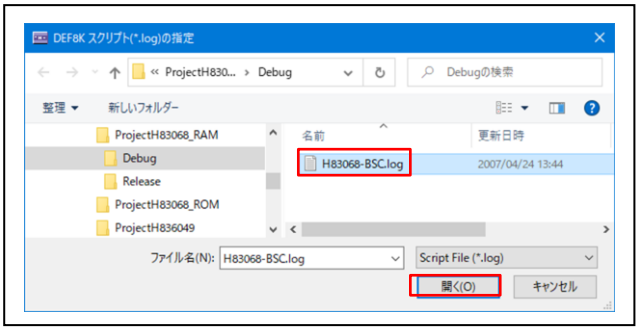
```
例）ファイル名<H83068-BSC.log>

//
// H8 用(H8/3068F)バスステートコントローラ初期設定
//    エリア 1:SRAM 512Kb 8bit 0x200000
// コメントは、コマンド実行ラインに記述しないで下さい。
//

//バス幅コントロールレジスタ CS1 エリア:8bit(default)
<S        ABWCR 0xff

//ポート 1 データディレクションレジスタ A7,A6,A5,A4,A3,A2,A2,A0
<S        P1DDR 0xff

//ポート 2 データディレクションレジスタ A15,A14,A13,A12,A11,A10,A9,A8
<S        P2DDR 0xff

//ポート 5 データディレクションレジスタ A19,A18,A17,A16
<S        P5DDR 0xf

//ポート 8 データディレクションレジスタ CS1 出力端子
<S        P8DDR 0x8
```

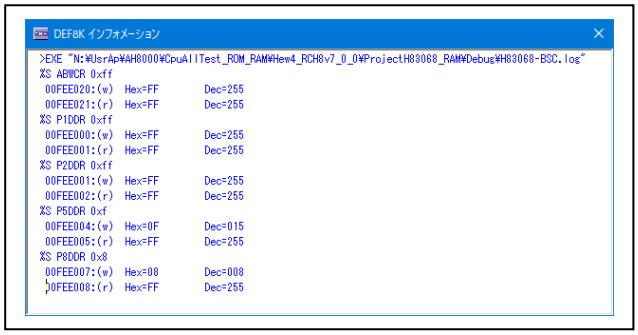
//コメント行
<S ｛8 ビットアクセス｝ ｛レジスタ名｝ ｛データ｝
<SS｛16 ビットアクセス｝｛レジスタ名｝ ｛データ｝
<SL｛32 ビットアクセス｝｛レジスタ名｝ ｛データ｝
【内部登録されているシンボルタイプ（ﾋﾞｯﾄ長）を使用する場合】
<SQ｛8～32 ビットアクセス｝ ｛レジスタ名｝ ｛データ｝

３－２）作成したスクリプトファイルを実行させ確認をする。

【DEF8K メニュー】
＜ファイル＞―＜スクリプト実行＞

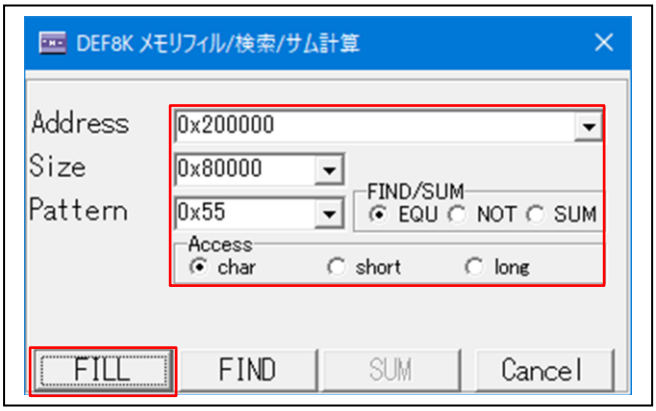


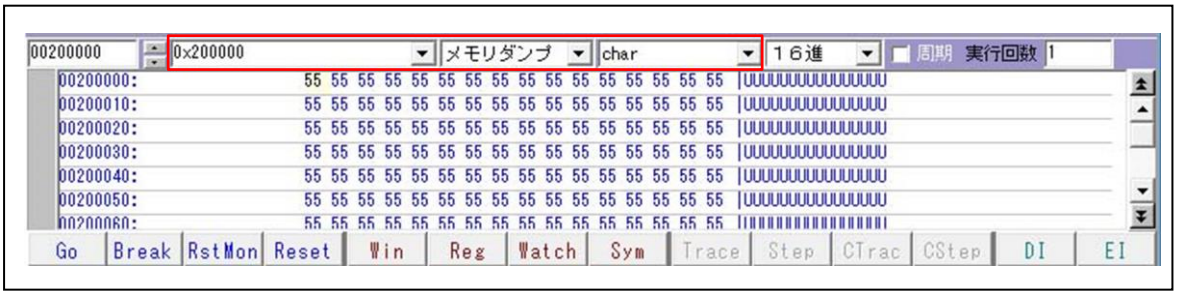




３－３）拡張 RAM が正しく読み書き出来るか「メモリフィル」で確認する。

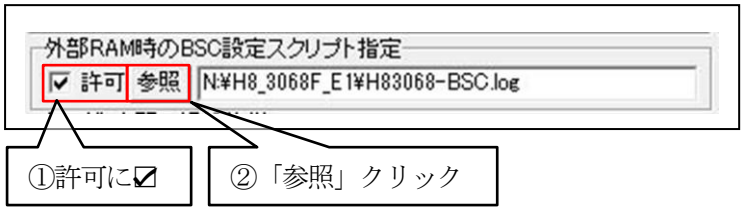
【DEF8K メニュー】
＜データ＞―＜メモリフィル／検索／サム計算＞

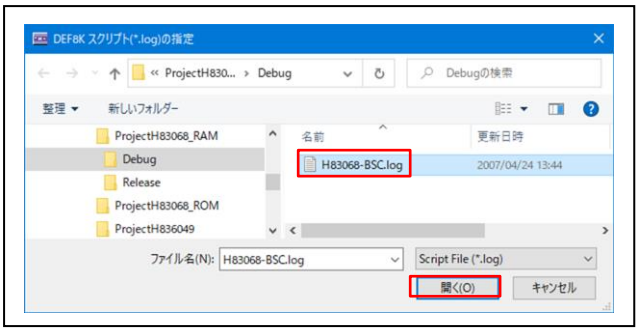




３－４）作成したスクリプトファイルを「CPU 設定」に登録する。

【DEF8K メニュー】
＜オプション＞―＜CPU 設定＞

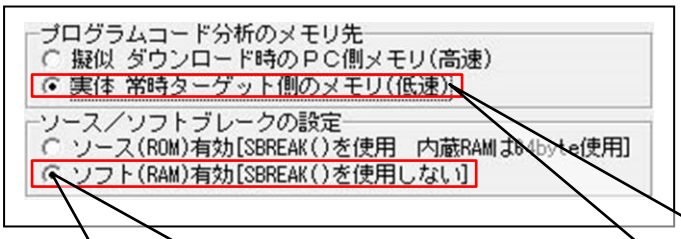




【スクリプトファイル登録による効果】
・ユーザプログラムのダウンロード時の、開始と終了後に登録された内容を実行します。
・【RstMon】と【Reset】を実施後、登録された内容を実行します。

３－５）「環境設定」を設定する。

【DEF8K メニュー】
＜オプション＞―＜環境設定＞



ソフト(RAM)有効[SBREAK()を使用しない]側をチェックする。

プログラムデバッグの初期段階ではバグの原因により、RAM エリアを書き換えてしまう可能性がある場合は、安定するまで「実体」側を指定することを推奨します。

３－６）プログラム側の条件と準備

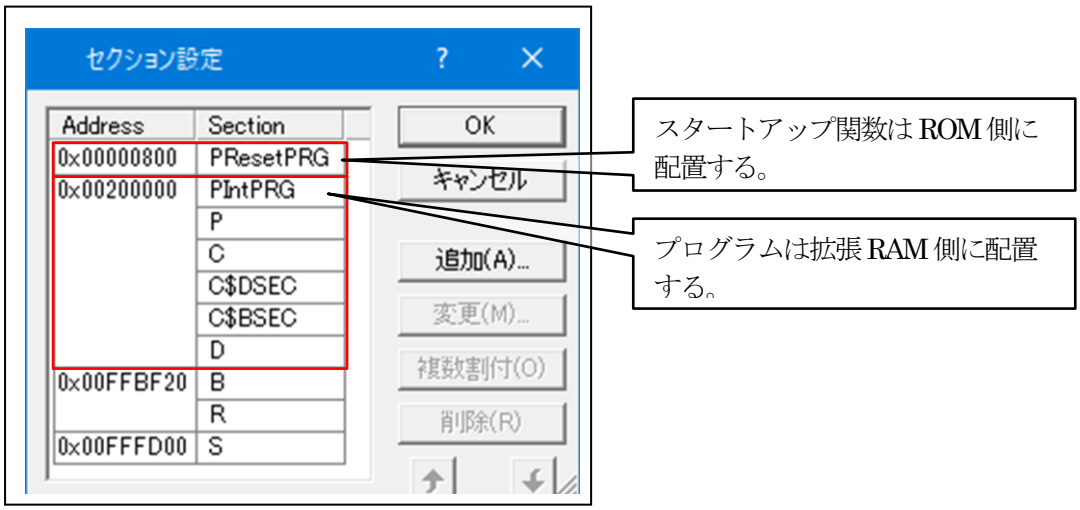
・スタートアップ関数は、必ず、ROM 側に配置して下さい。

・スタートアップ関数処理でスタックポインタ設定後、ソフトタイマ 200ms 以上を実装して下さい。理由は、DEF8K より MCU リセット指示からモニタ起動(NMI)するまでにプログラム未実装の RAM エリアに飛びプログラム暴走を防ぐために必要です。

１）ルネサス C の場合【resetprg.c】

```
#pragma section ResetPRG
//************************************************************
void   Wait1ms(void)     // 1ms ｿﾌﾄﾀｲﾏｰ(25.000MHz)
{
   long cnt;
   cnt = 961;          // 961*26=(25000)
   while(cnt-- != 0) {}   // 26 clock
}
//************************************************************
void   SoftWaitNms(long ms)
{
   while(ms-- != 0) {
      Wait1ms();
   }
}
//    スタートアップ関数
__entry(vect=0) void PowerON_Reset(void)
{
   SoftWaitNms(200); // <----ポイント 200ms Wait(ブート I/F の場合特に必要)
// set_imask_ccr((_UBYTE)1);
   _INITSCT();
// 以下省略
// set_imask_ccr((_UBYTE)0);
   main();
}
```

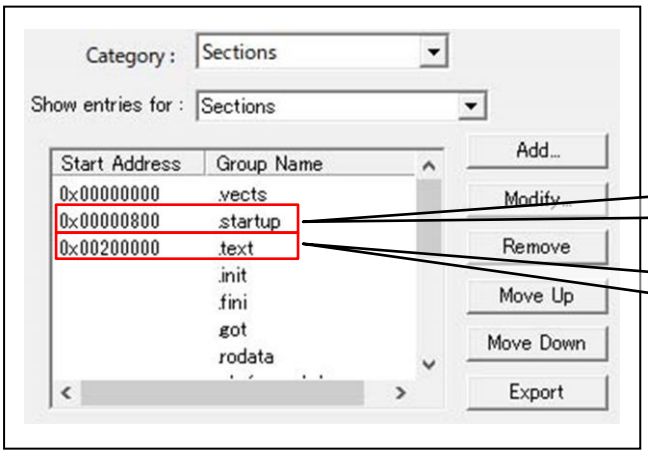
【セクションの設定】



2）KPIT/Gnu-C の場合【start.asm】

```
.global     _start
.section    .startup,"aw"
;-----wait1ms(void)---------------------------------------
      .global   _Wait1ms
_Wait1ms:
      mov.w   #3125,r0   ;; 25000/8(clock)
wait:
      dec.w   #1,r0       ;; 2 clock
      nop                 ;; 2 clock
      bne     wait        ;; 4 clock
      rts
;-----SoftWaitNms(long ms)--------------------------
      .global   _SoftWaitNms
_SoftWaitNms:
      mov.l   er0,er1
loop:
      jsr     @_Wait1ms
      dec.l   #1,er1
      bne     loop
      rts
;-----startup----------------------------------------------------
_start:
      ; initialise the SP for non-vectored code
      mov.l   #_stack,er7
      ; ポイント 200ms Wait(ブート I/F の場合必要)
      mov.l   #200,er0
      jsr     @_SoftWaitNms

      ; call the hardware initialiser
      jsr     @_hw_initialise

      ;;  以下省略
```

【セクションの設定】



スタートアップ関数は ROM 側に配置する。

プログラムは拡張 RAM 側に配置する。

3）IAR-C の場合【cstartup.asm】

```
#define CODESEG STARTUP

 RSEG   CODESEG:CODE:NOROOT(1)
 PUBLIC ?cstart_init_sp

?cstart_init_sp:

      MOV.X  #SFE(CSTACK), SP
      // 200ms wait
      mov.l     #200,er0
      jsr       @SoftWaitNms

// 省略

//-----wait1ms(void)-----------------------------------
Wait1ms:
      mov.w  #3125,r0   // 25000/8(clock)
wait:
      dec.w     #1,r0      // 2 clock
      nop                  // 2 clock
      bne       wait       // 4 clock
      rts
//-----SoftWaitNms(long ms)----------------------------
SoftWaitNms:
      mov.l     er0,er1
loop:
      jsr       @Wait1ms
      dec.l     #1,er1
      bne       loop
      rts
//-----------------------------------------------------
```

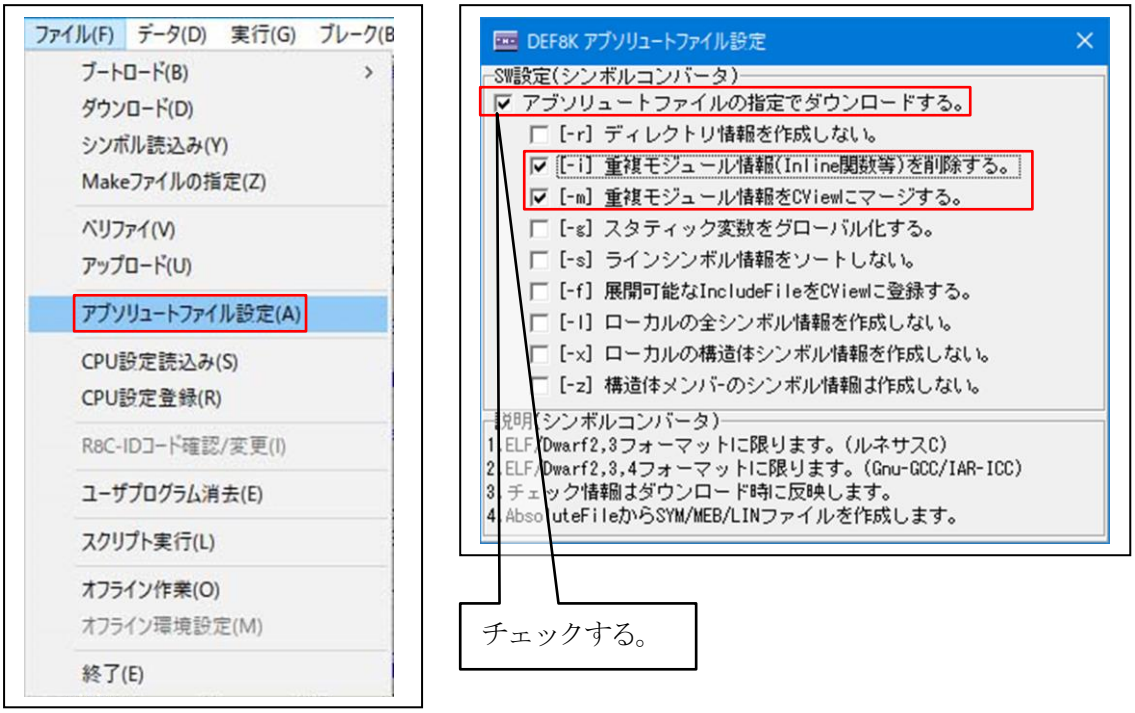
【セクションの設定】【lnk3068flh.xcl】

```
-P(CONST)INTVEC=0-3FF
-P(CONST)FLIST=40-FF
-P(CONST)STARTUP=800-BFF

-P(CONST)DATA16_C=200000-27FFFF
-P(CODE)CODE24=200000-27FFFF
-Z(CONST)DATA8_ID,DATA16_ID,DATA32_ID=200000-27FFFF
-Z(CONST)DIFUNCT=
-P(CONST)DATA32_C=

-Z(CONST)CHECKSUM=200000-27FFFF
```

４．DEF8K でデバッグ

４－１）アブソリュートファイルでのダウンロードを設定する。

【DEF8K メニュー】
＜ファイル＞―＜アブソリュートファイル設定＞



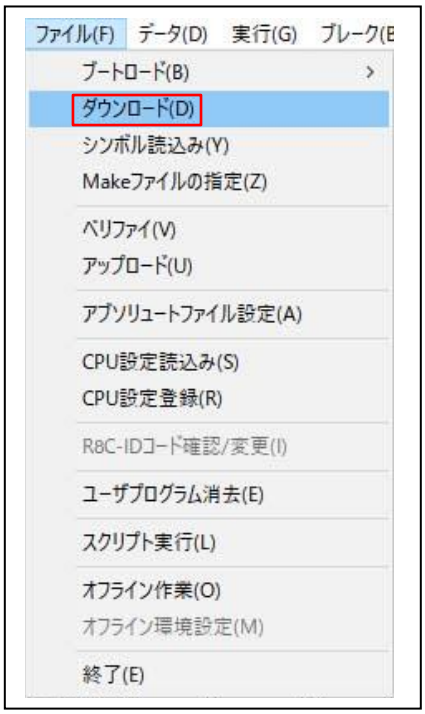
１）シンボルコンバータスイッチの説明

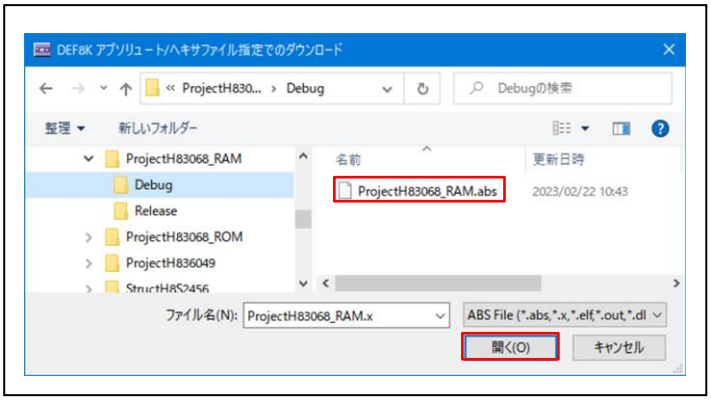
【－ｒ】ディレクトリ情報を作成しない。
【－ｉ】不整合なＩｎｌｉｎｅ情報を削除する。
【－m】重複モジュール情報を C ソースにマージする。
【－ｇ】スタティック変数をグローバル化する。
【－ｓ】ラインシンボル情報をソートしない。
【－ｆ】使用インクルードファイルを CView に登録する。
【－ｌ】ローカル変数情報を作成しない場合はチェックします。
【－ｘ】ローカルの構造体シンボル情報を作成しない場合はチェックします。
【－ｚ】構造体メンバのシンボル情報を作成しない場合はチェックします。

◎[-l] [-x] [-z]のオプションスイッチの使用目的は、シンボル数が制限数を超えてしまいグローバルシンボルを優先したい場合にチェックします。また、コンパイラ等のバージョンアップに伴い ELF/Dwarf 情報に不具合がありデバッグ作業が継続できなくなった場合の一時的な退避策として使用する。

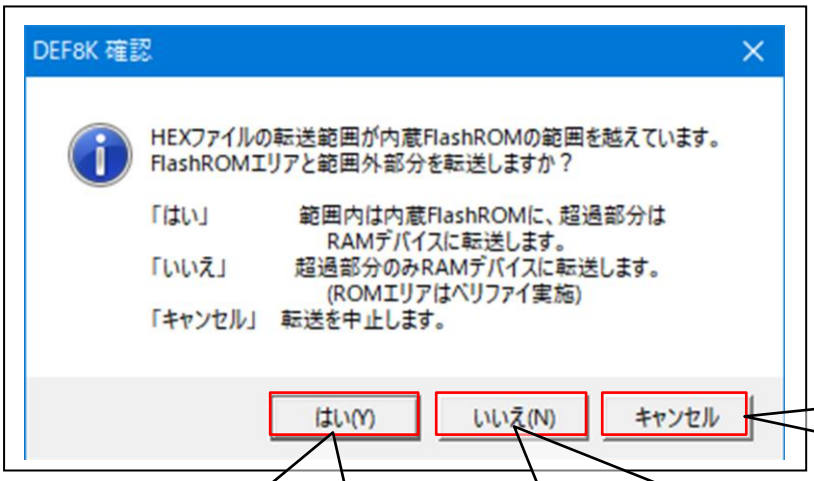
4－2）プログラムをダウンロードする。

【DEF8K メニュー】
＜ファイル＞―＜ダウンロード＞





1）ダウンロード方法の確認

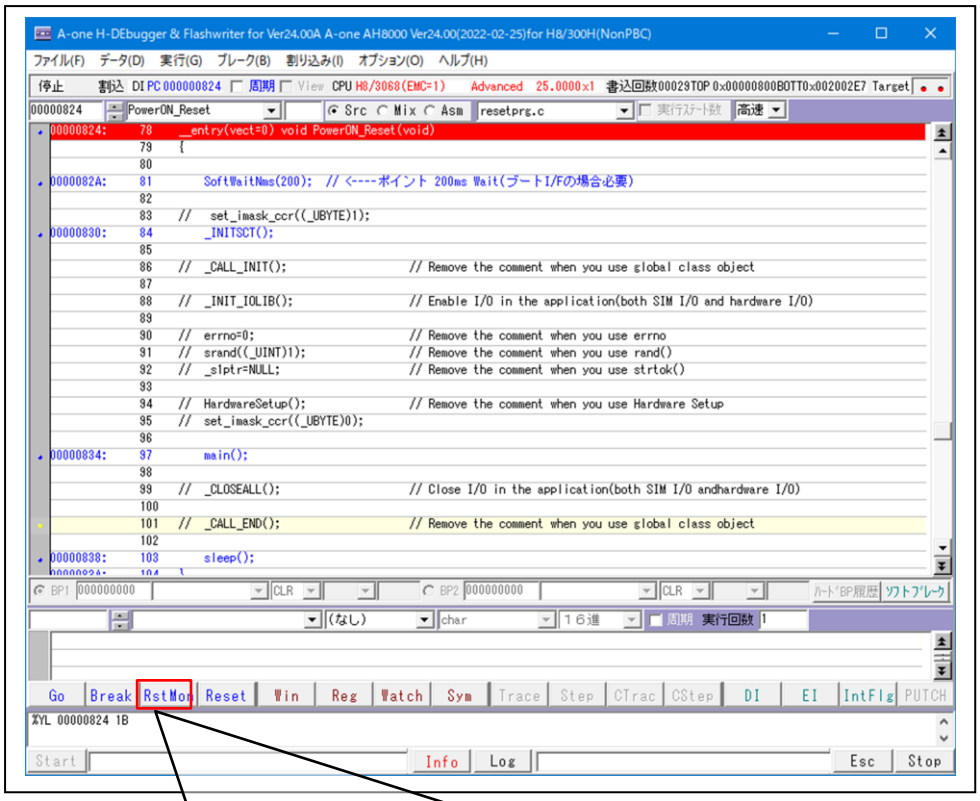


ダウロードを中止します。

フラッシュ ROM と RAM エリアに転送します。

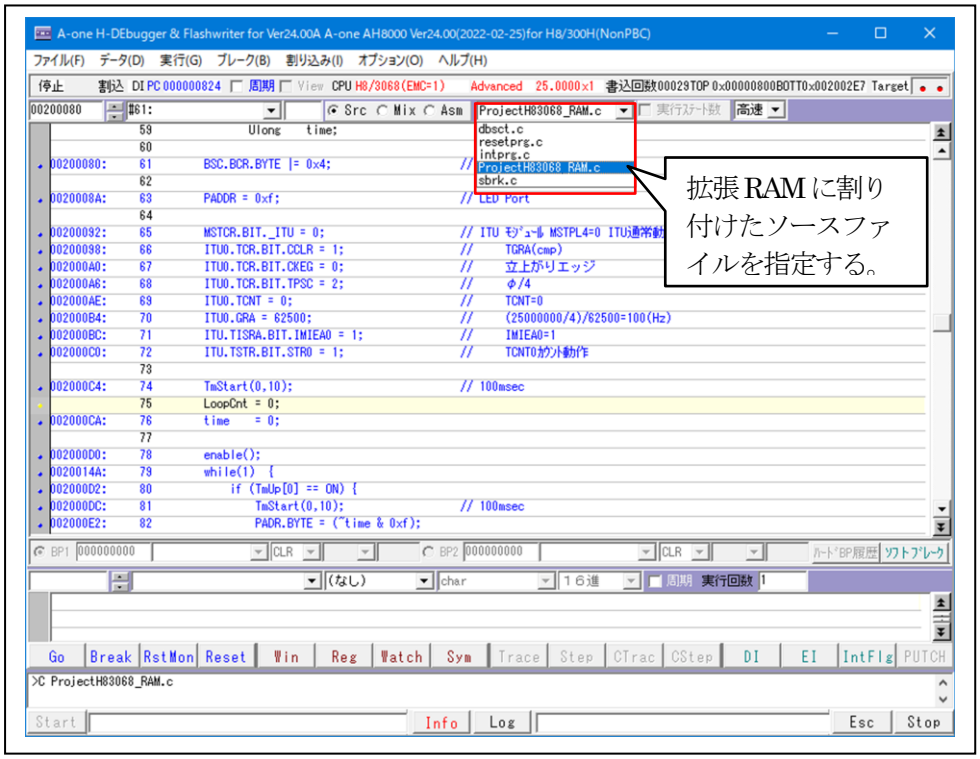
フラッシュ ROM エリアは「ベリファイ」を実施し、RAM エリアのみに転送します。

4－3）ダウンロード後の DEF8K 画面

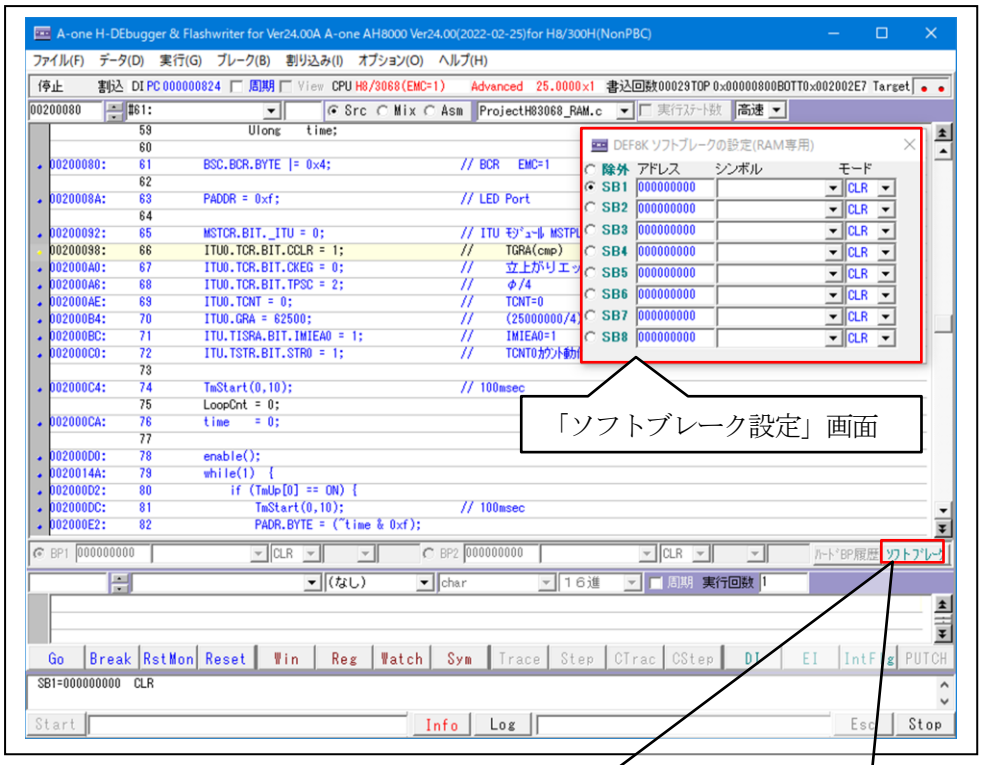


【RstMon】をクリックするとスタートアップ関数の先頭に View 表示が切り替わります。

1）拡張 RAM エリアのプログラムソースに画面を切り替える

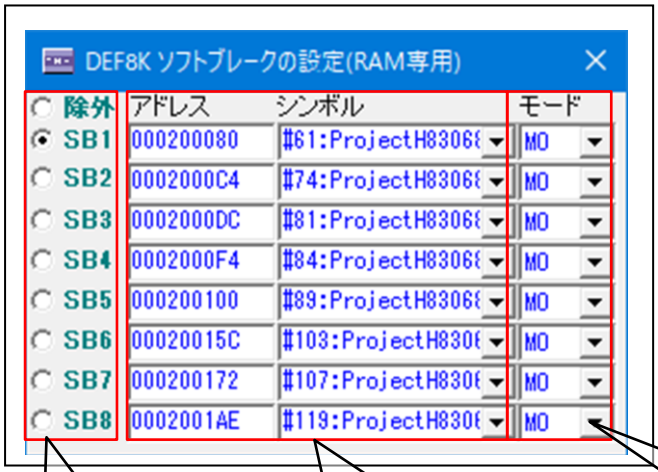


4－4）ソフトブレーク設定画面を開く



【ソフトブレーク】PB をクリックすると、「ソフトブレーク設定」画面が開きます。

1）ソフトブレーク設定の説明



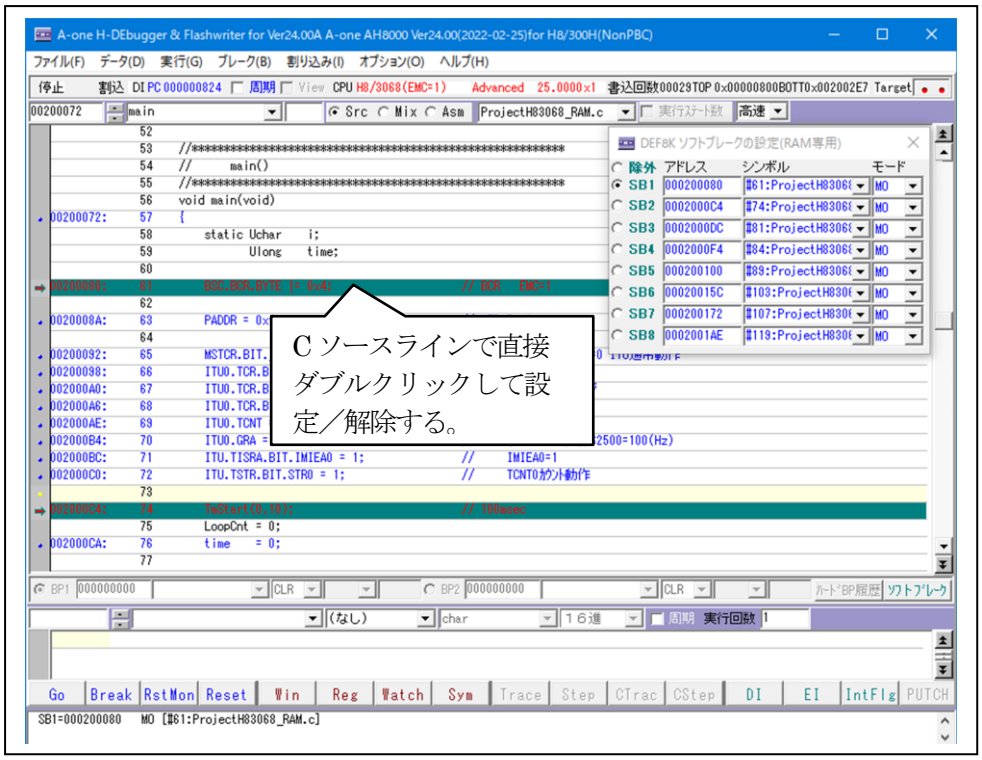
【SBx】
8 ポイントの
設定／選択

【除外】
View 上でのダブルクリック設定を除外する。

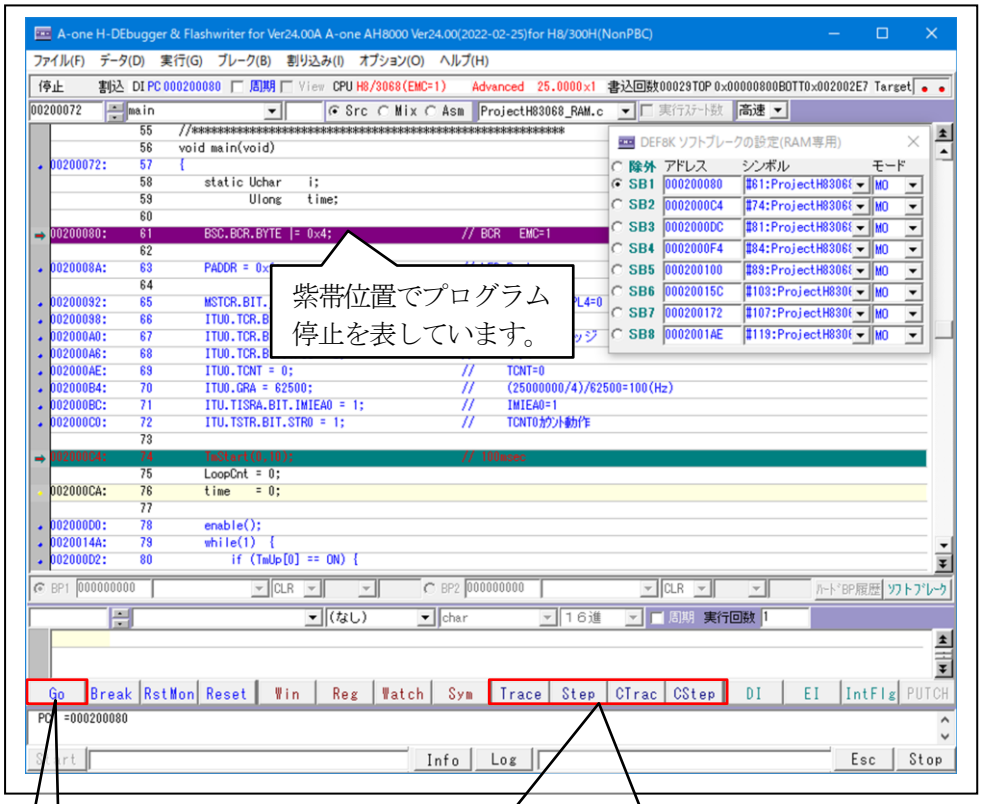
ブレークポイントの
「アドレス」と
「ラインシンボル」を表記
ダブルクリックするとブレークポイント位置の View 画面に切り替わります。

【MO】ブレークポイント有効
【CLR】ブレークポイント消去

2）Cview 画面で直接ブレークポイントの設定



3）ソフトブレークポイントまで実行



以上

５．注意事項

・本文書の著作権は、エーワン（株）が保有します。
・本文書を無断での転載は一切禁止します。
・本文書に記載されている内容についての質問やサポートはお受けすることが出来ません。
・本文章に関して、ルネサス　エレクトロニクス社への問い合わせは御遠慮願います。
・本文書の内容に従い、使用した結果、損害が発生しても、弊社では一切の責任は負わないものとします。
・本文書の内容に関して、万全を期して作成しましたが、ご不審な点、誤りなどの点がありましたら弊社までご連絡くだされば幸いです。
・本文書の内容は、予告なしに変更されることがあります。

〒486-0852
愛知県春日井市下市場町 6-9-20
エーワン株式会社
https://www.aone.co.jp