シリアルフラッシュＲＯＭ対応のローダプログラムを追加する場合の説明（ブートモード１／３用）

Ｒｅｖ１．３０　２０１２／１０／４
ＤＥＦバージョン１１．００Ａより
ＤＥＦバージョン１２．１０Ａ変更
ＤＥＦバージョン１２．２０Ａ変更

【対象ＣＰＵ】

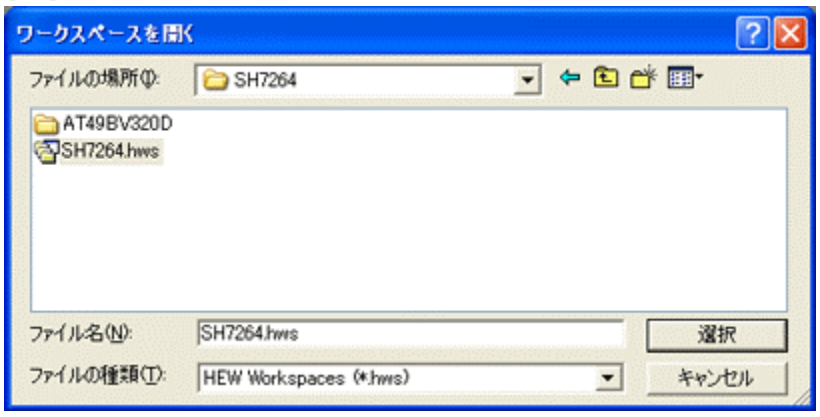
１．ＲＯＭレス品種が対象になります。（ＳＨ７２６２／４／６／７）

【機能】

１．サンプルで「ＡＴ２５ＤＦ０４１Ａ」の対応ソフトを用意してあります。（ルネサスＣのみ対応）【ＳＨ７２６２／４】
２．サンプルで「ＳＳＴ２５ＶＦ０１６Ｂ／Ｍ２Ｐ１６」の対応ソフトを用意してあります。（ルネサスＣのみ対応）【ＳＨ７２６６／７】
３．ＳＨ７２６２／４／６／７【ＲＳＰＩ０】に接続されたシリアルフラッシュＲＯＭが対象になります。
４．ＨｅｗにてＦｌａｓｈＲＯＭの品種追加が出来るよう対応する。
５．シリアルＦｌａｓｈＲＯＭ仕様にあわせたバイトリードをプログラミングするだけで追加が可能になります。

【品種追加前の準備】

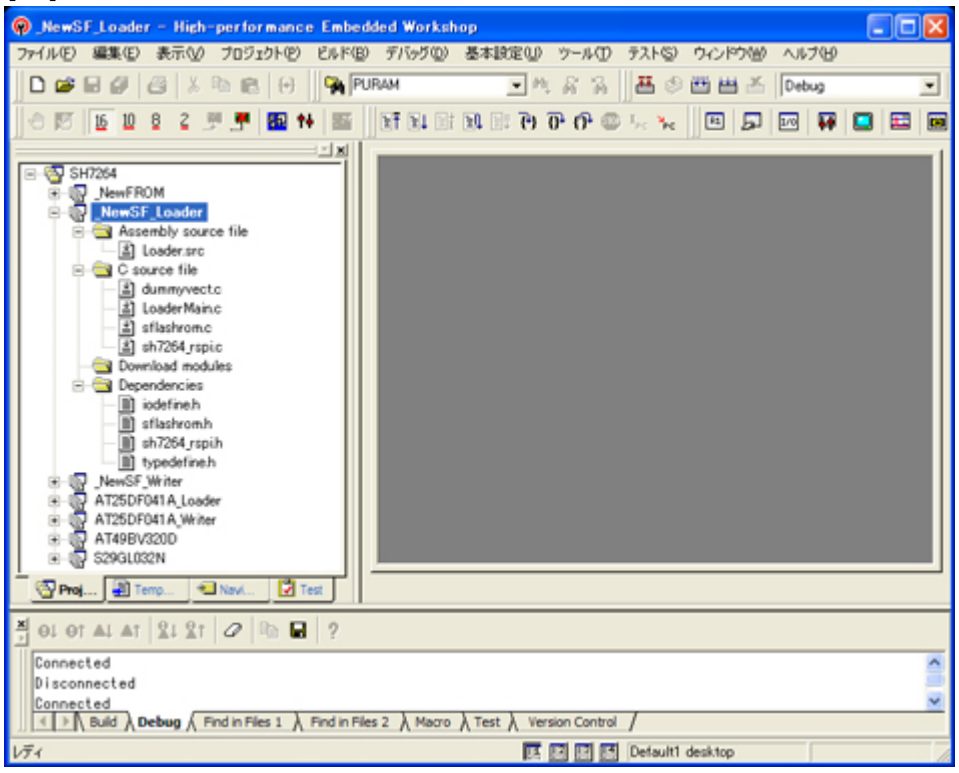
１．ＡＨ７０００コントロールソフトのインストールＤＩＲにあるワークスペースを開きます。（追加例）
［1-1］



＜デフォルトディレクトリ＞
"c:¥Program Files¥Aone¥DEF¥rom-custom¥SH7264"

＜ワークスペース＞
"SH7264.hws"

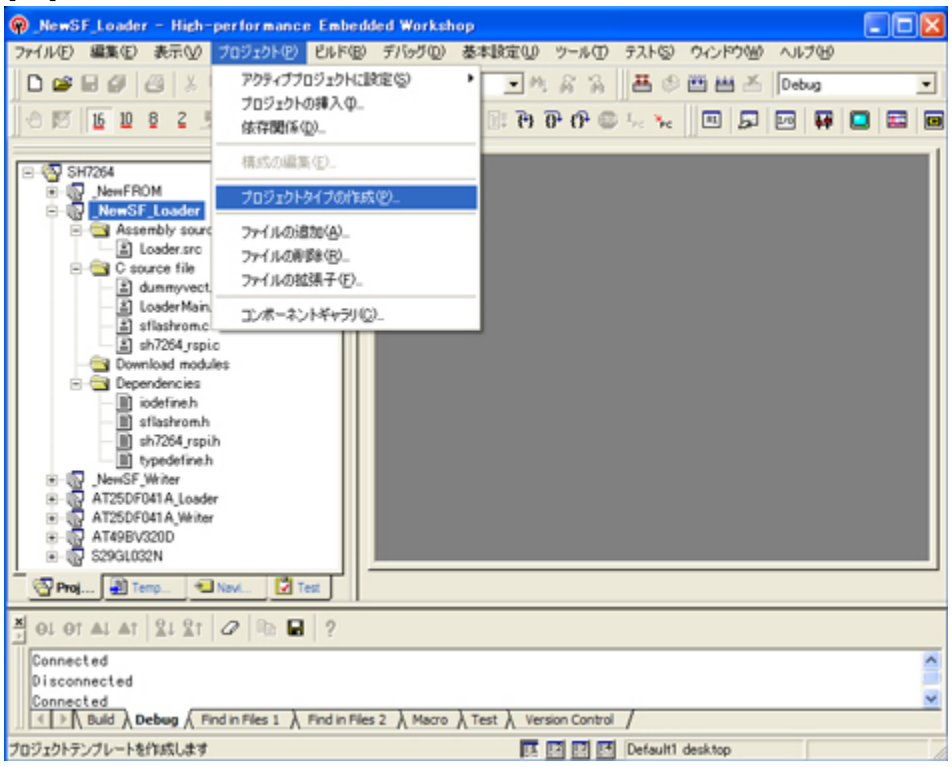
２．プロジェクト名「_ＮｅｗＳＦ__Ｌｏａｄｅｒ」をアクティブプロジェクトに設定します。
［1-2］



＜_ＮｅｗＳＦ__Ｌｏａｄｅｒ＞をマウスクリックし、右クリックのポップアップメニューから選択します。

3．「プロジェクトタイプの作成」をします。

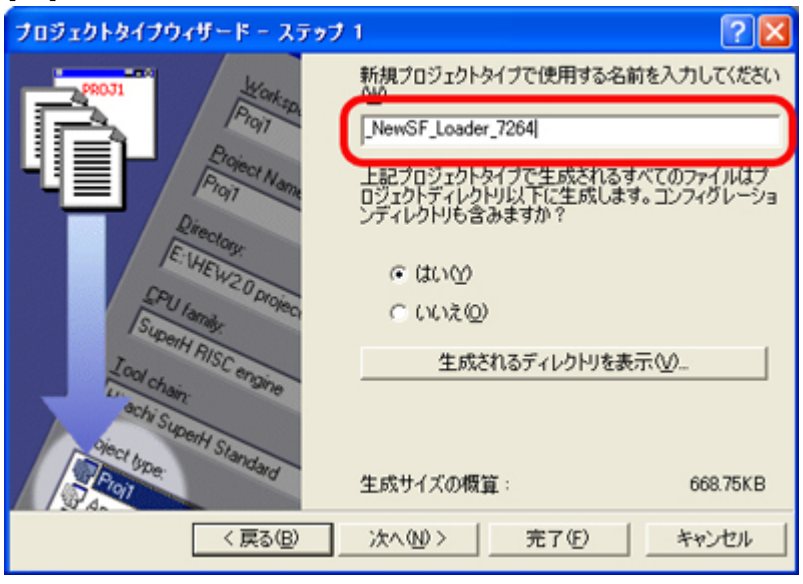
［1-3］



Ｈｅｗメニュー
＜プロジェクト＞ー＜プロジェクトタイプの作成＞をクリックします。

4．新規プロジェクトタイプで使用する名前を指定します。

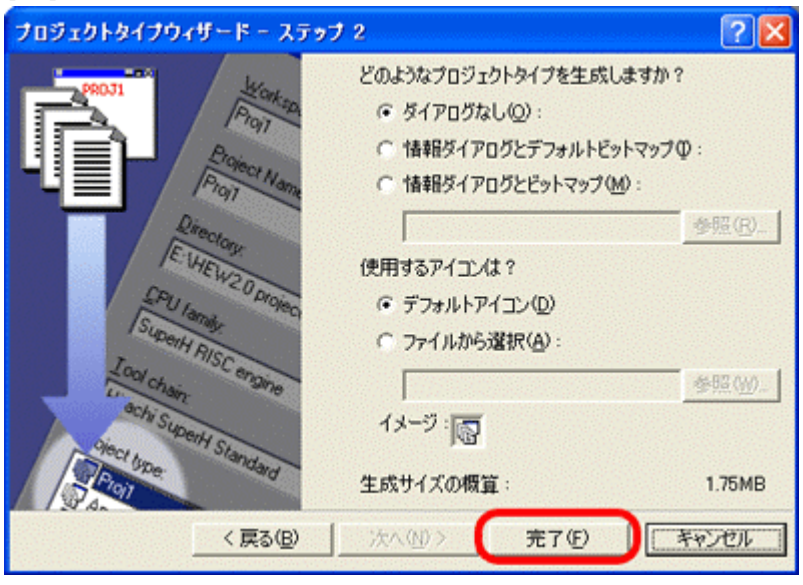
［1-4］



名前は、重複しないようにして下さい。
例として「ＮｅｗＳＦ＿Ｌｏａｄｅｒ＿７２６４」としておきます。

＜ー「次へ」をクリックします。

［1-5］



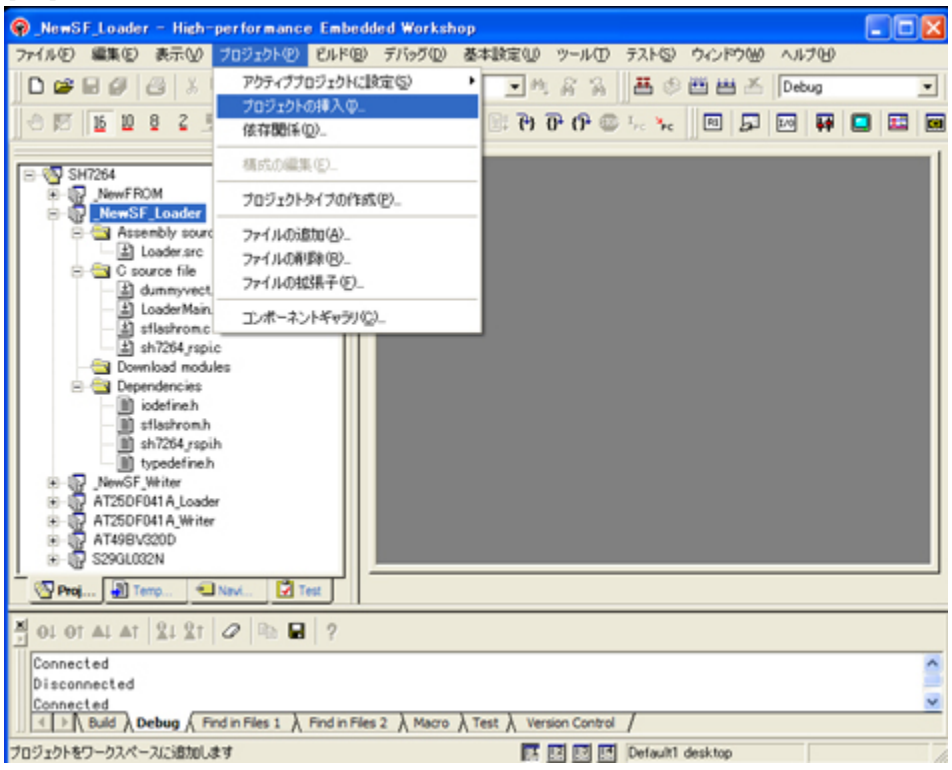
＜ー完了をクリックします。

この操作で「品種追加前の準備」は完了です。

【新規シリアルＦｌａｓｈＲＯＭ用ローダの追加】

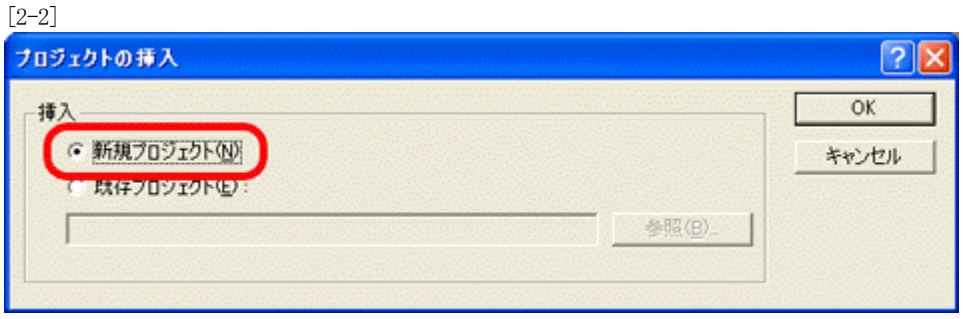
1．新規シリアルＦｌａｓｈＲＯＭ用ローダを追加するため、「プロジェクトの挿入」をします。

[2-1]



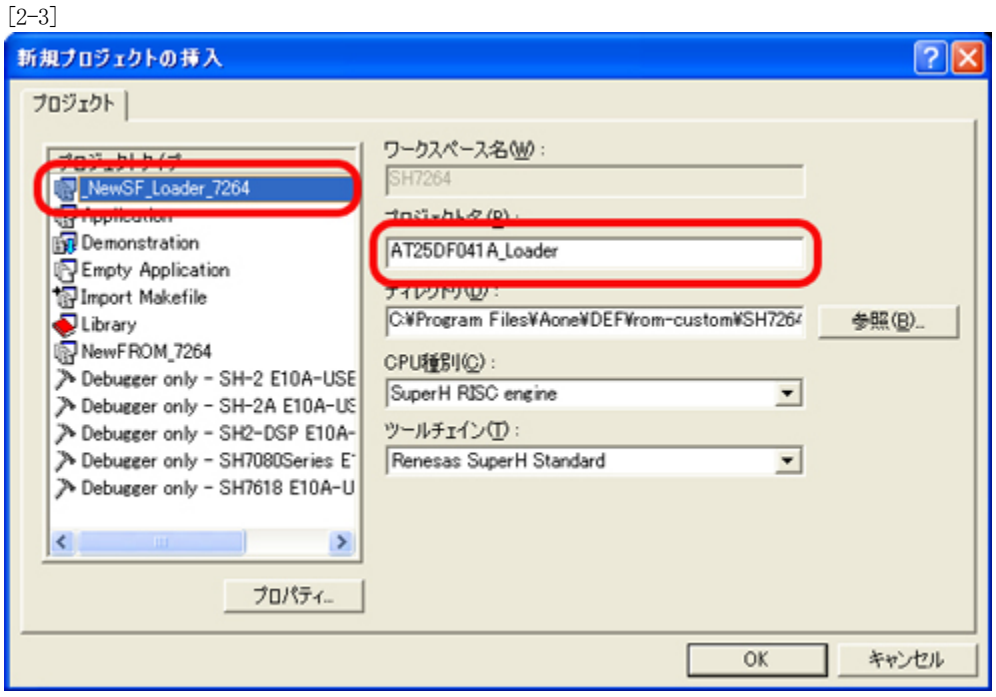
Ｈｅｗメニュー
＜プロジェクト＞ー＜プロジェクトの挿入＞をクリックします。

[2-2]



「新規プロジェクト」を指定して、「OK」をクリックします。

[2-3]



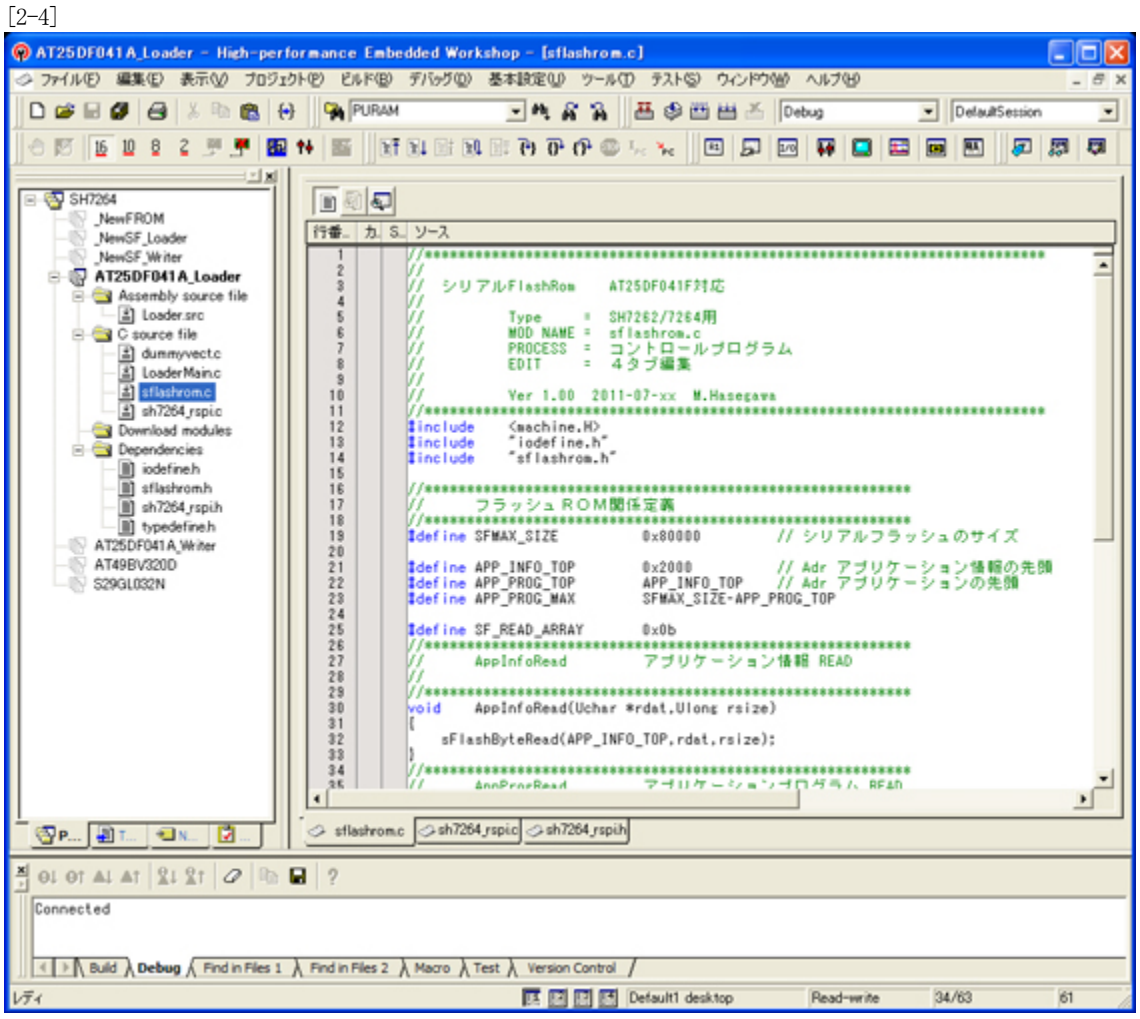
1）プロジェクトタイプに作成した
「ＮｅｗＳＦ＿Ｌｏａｄｅｒ＿７２６４」を指定します。

2）プロジェクト名を指定します。FlashROM 名を追加した
「ＡＴ２５ＤＦ０４１Ａ＿Ｌｏａｄｅｒ」にします。

＜ー「OK」をクリックします。

２．新規シリアルＦｌａｓｈROM用ローダの定義およびプログラムを作成します。

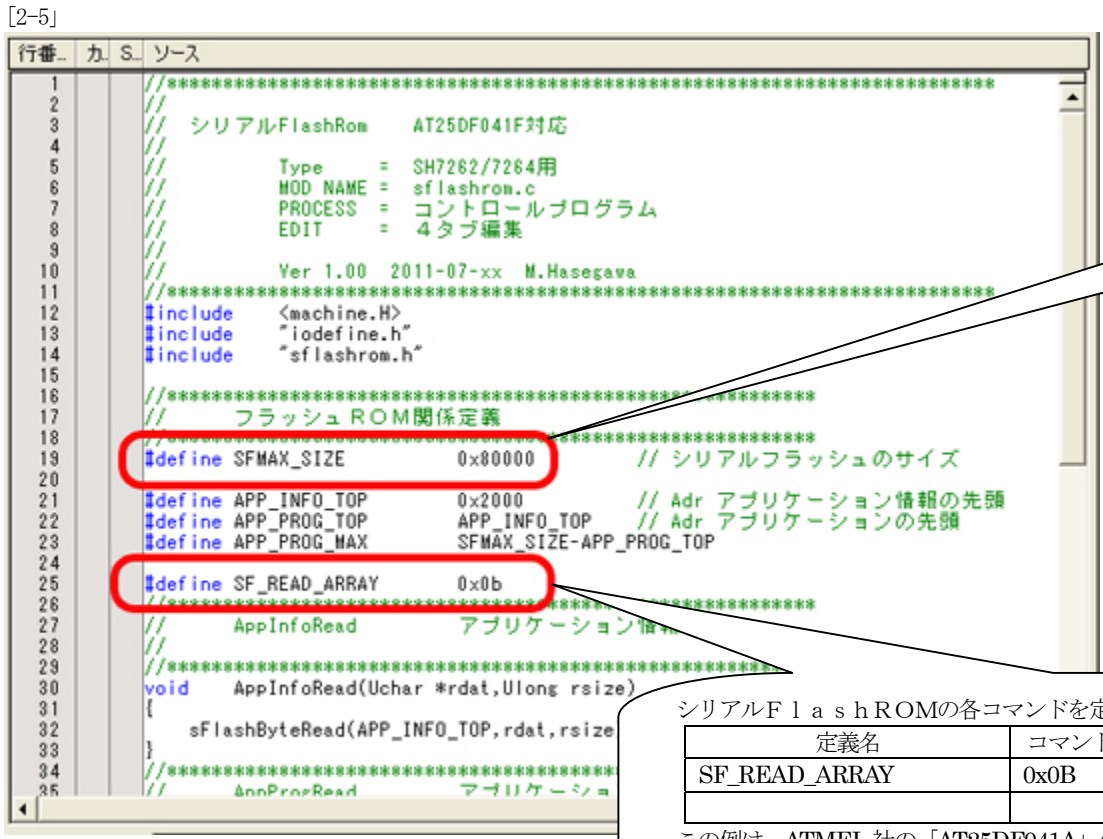
[2-4]



品種追加用ファイル
「sflashrom.c」を開きます。

１）「sflashrom.c」に新規シリアルＦｌａｓｈROMの情報を定義します。

[2-5]



「SFMAX_SIZE」に使用するシリアルＦｌａｓｈROMの最大サイズ（バイト）を定義しま

シリアルＦｌａｓｈROMの各コマンドを定義します。

定義名	コマンド	機能
SF_READ_ARRAY	0x0B	データの読み出し

この例は、ATMEL 社の「AT25DF041A」のコマンドです。コマンド仕様は各デバイスメーカーによって異なります。使用するデバイスのデータシートを参照して下さい。

２）「sflashrom.c」に新規シリアルＦｌａｓｈROMに依存したコマンドプログラムを作成します。

⚠

下記の参考プログラム例は、ATMEL 社の「AT25DF041A」用コマンドプログラムです。コマンドパラメータ仕様は各デバイスメーカーによって異なります。使用するデバイスのデータシートを参照してから作成して下さい。

[2-6]

```
行番  S  ソース
 26     //*********************************************************
 27     //      AppInfoRead         アプリケーション情報 READ
 28     //
 29     //*********************************************************
 30     void    AppInfoRead(Uchar *rdat,Ulong rsize)
 31     {
 32         sFlashByteRead(APP_INFO_TOP,rdat,rsize);
 33     }
 34     //*********************************************************
 35     //      AppProgRead         アプリケーションプログラム READ
 36     //
 37     //      return(0)  : 正常終了
 38     //      return(-1) : 異常終了
 39     //*********************************************************
 40     int     AppProgRead(Uchar *rdat,Ulong rsize)
 41     {
 42         if (rsize <= APP_PROG_MAX) {
 43             sFlashByteRead(APP_PROG_TOP,rdat,rsize);
 44             return(0);
 45         }
 46         return(-1);
 47     }
 48     //*********************************************************
 49     //      sFlashByteRead      指定バイト数の読み込み
 50     //
 51     //*********************************************************
 52     void    sFlashByteRead(Ulong adr,Uchar *rdat,Ulong rsize)
 53     {
 54         Uchar   cmd[5];
 55
 56         cmd[0] = SF_READ_ARRAY;
 57         cmd[1] = (Uchar)((adr >> 16) & 0xff);
 58         cmd[2] = (Uchar)((adr >> 8)  & 0xff);
 59         cmd[3] = (Uchar)( adr        & 0xff);
 60         cmd[4] = 0;
 61         IoCmdExeRead(cmd,5,rdat,rsize);
 62     }
 63
```

関数「AppInfoRead(...)」は、アプリケーションのローダ情報をシリアルＦｌａｓｈＲｏｍの「APP_INFO_TOP(0x2000)」番地より読み出します。
＜引数の仕様＞
Uchar *rdat; // 読み出したデータを格納する先頭アドレス
Ulong　rsize; // 読み出しデータのサイズ（バイト）

関数「AppProgRead(...)」は、アプリケーションプログラムをシリアルＦｌａｓｈＲｏｍの「APP_PROG_TOP(0x2000)」番地より読み出します。
＜引数の仕様＞
Uchar *rdat; // 読み出したデータを格納する先頭アドレス
Ulong　rsize; // 読み出しデータのサイズ（バイト）

関数「sFlashByteRead(....)」は、バイト読み出しします。

＜引数の仕様＞
Ulong　adr ; // 読み出しするデバイス内の先頭アドレス
Uchar *rdt; // 読み出したデータを格納する先頭アドレス
Ulong　rsize; // 読み出しデータのサイズ（バイト）

[IoCmdExeRead()¥sh7264_rspi.c] RSPI0 コマンド(リードデータありタイプ）送信関数です。

[2-6-1]

```
行  S/Wブ  ソース
 40         //*********************************************************
 41         //      LoaderMain
 42         //      ローダープログラムからJmp
 43         //*********************************************************
 44         void    LoaderMain(void)
 45         {
 46             set_vbr((void *)&DummyVectors);
 47             set_fpscr(FPSCR_Init);
 48             set_cr(SR_Init);
 49
 50             SoftTimer();                    // 4ms Timer  Point!!
 51                                             // BootMode1/3の状態でデバッグする場合、デバッガの
 52                                             // Reset->Breakの間にローダが動作しないようにする
 53                                             // ためにココにSoftTimerを入れる
 54
 55
 56             CPG.SYSCR3.BYTE = 0xff;         // VRAME5->0 アクセス有効 大容量内蔵RAM
 57                                             //           0x3C00_0000 -> 0x3C17_FFFF
 58             CPG.SYSCR4.BYTE = 0xff;         // VRAMWE5->0 ライト有効/無効 大容量内蔵RAM
 59                                             //           0x3C04_0000 -> 0x3C17_FFFF(有効)
 60             CPG.SYSCR5.BYTE = 0x0f;         // ERAMWE8->0 ライト有効 保持用大容量内蔵RAM
 61                                             //           0x3c00_0000 -> 0x3c01_ffff
 62
 63             IoInitRSPI0();                  // RSPIの初期化
 64
 65             AppInfoRead((Uchar *)AppProgInfo,sizeof(AppProgInfo));
 66                                             // MAX 16MBまで
 67             Size = (AppProgInfo[1] - AppProgInfo[0]) & 0xffffff;
 68             Stat = AppProgRead((Uchar *)AppProgInfo[0],Size);
 69             LastAdr = AppProgInfo[1]-1;
 70             if((LastAdr >= 0xfff80000) && (LastAdr <= 0xfff8ffff)){
 71                                             // アドレスが内臓高速RAMの場合、ポート内容を登録させるため
 72                                             // ページごとにポートが存在しているため無条件で二度書きする
 73                 Stat = AppProgRead((Uchar *)AppProgInfo[0],Size);
 74             }
 75             if (Stat == 0) {
 76
 77                 CPG.SYSCR5.BYTE = 0x0;      // ERAMWE8->0 ライト無効 保持用大容量内蔵RAM
 78                                             //           0x3c00_0000 -> 0x3c01_ffff
 79                                             //           リセット状態に戻す
 80                 JmpAppProg();               // アプリケーションへ
 81             }
```

⚠ポイント！！

ブートモード１／３の設定のままでデバッグする場合のポイントです。
この位置にソフトタイマー(約 4msec)を入れて下さい。

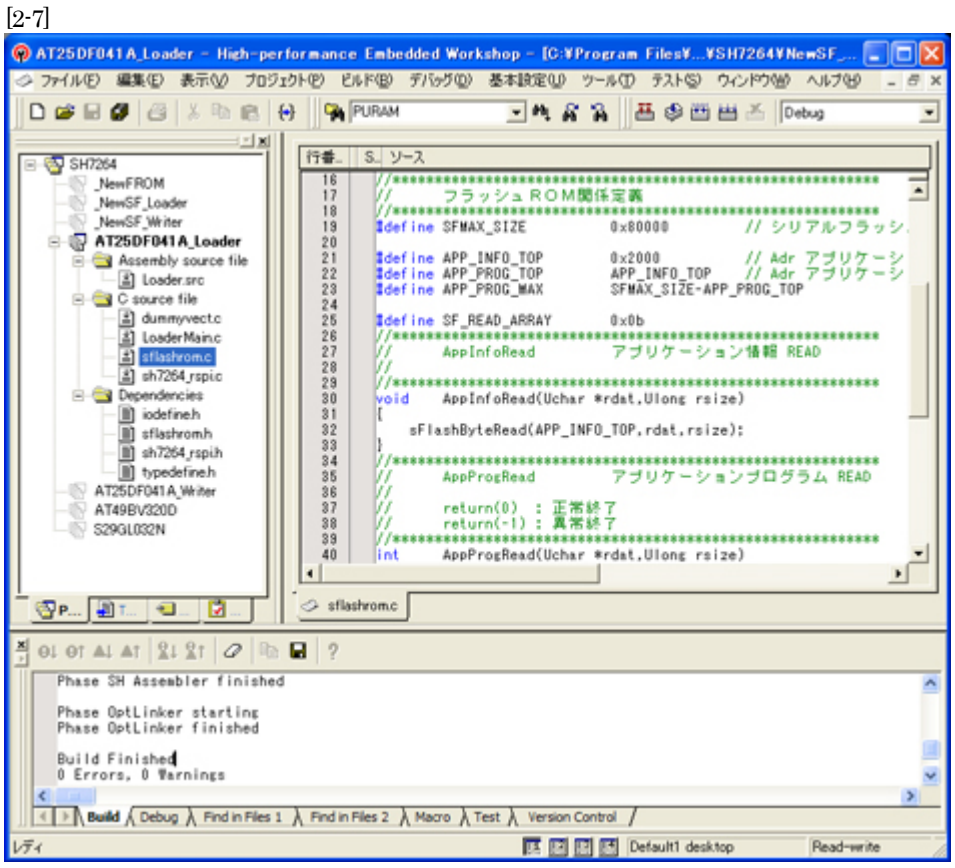
＜理由＞
シリアルフラッシュ ROM に旧バージョンのソフトウェアが書き込みしてある状態で新バージョンのソフトウェアを作成するためにデバッグしている時にデバッガ操作で「RstMon」Reseet+MoniterStart をした場合、この位置にソフトウェアタイマー(約 4msec)を入れときますとシリアルフラッシュ ROM に登録されているプログラムのロードを防ぐことが出来ます。

＜RstMon の動作説明＞
RstMon をクリックしますと、
①ターゲットを Reset します。
②リセット解除後、強制ブレークを発行します。
①から②の間の強制ブレークは、H-UDI(J-TAG)通信でコマンド発行しますので、ブレーク有効になるまでは CPU は動作します。

以上で、関数の作成は終了です。

3．新規シリアルＦｌａｓｈROM用ローダプログラムをビルドします。

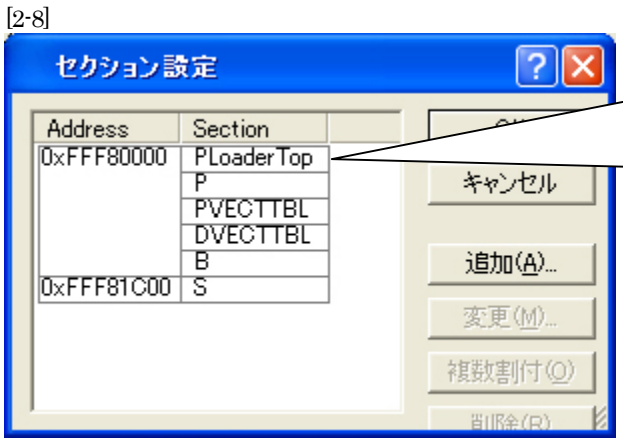
[2-7]



Ｈｅｗメニュー＜ビルド＞－＜すべてをビルド＞で、
「0 Errors 0 Warnings」になったことを確認します。

1）新規シリアルＦｌａｓｈROMライタ用プログラム作成上のルール

[2-8]



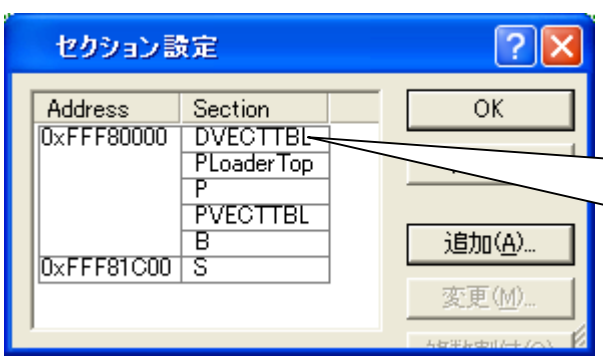
＜ルール１＞
セクション名「PLoaderTop」の、ロケートは必ず、シリコン内部のブートローダが転送する先頭アドレス「0xFFF80000」に割付けて下さい。

＜ルール２＞
スタックまで含めたオブジェクトサイズは「0x2000」8KB までです。

以上のルールは厳守して下さい。

2）ローダプログラムをデバッグする場合は、セクションの配置換えが必要です。

[2-9]



＜デバッグ時＞
セクション名「DVECTTBL」の、ロケートは必ず、シリコン内部のブートローダが転送する先頭アドレス「0xFFF80000」に割付けて下さい。

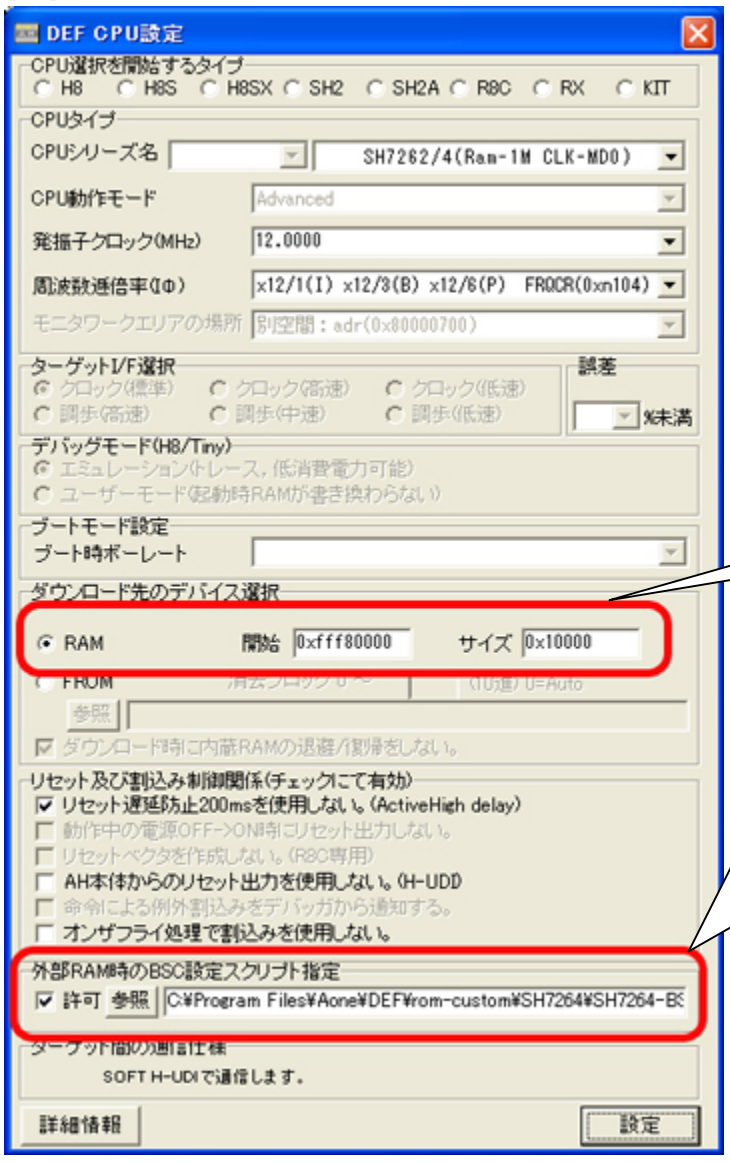
⚠
デバッグが終了しましたら、１）のロケーションに戻して下さい。

【作成したＦｌａｓｈROM用ローダプログラムのデバッグ方法】

⚠デバッグ中は、「ブートモード０」にするか、もしくは「ブートモード０」に出来ない回路構成（アドレス・データバスを汎用Ｉ／Ｏに使用）の場合は、プログラムの開始アドレスを「0xFFF8_2000」にロケートしなおし DEF 設定も「0xFFF8_2000」に設定してデバッグして下さい。デバッグ終了後は「0xFFF8_000」に戻して下さい。
理由：
「ブートモード１・３」のままですとリセット解除後、ブートプログラムが起動され内蔵高速ＲＡＭの先頭から「0x2000」エリアにローダプログラムがロードされてしまいます。

１．デバッガ用コントロールソフト「ＤＥＦ」にてデバッグする為の設定をします。

［3-1］



＜ＣＰＵ設定＞

作成した「FlashROM」プログラムをターゲット側の内部ＲＡＭに転送して実行させますので、「ＲＡＭ」にチェック後、先頭アドレスとサイズを指定します。

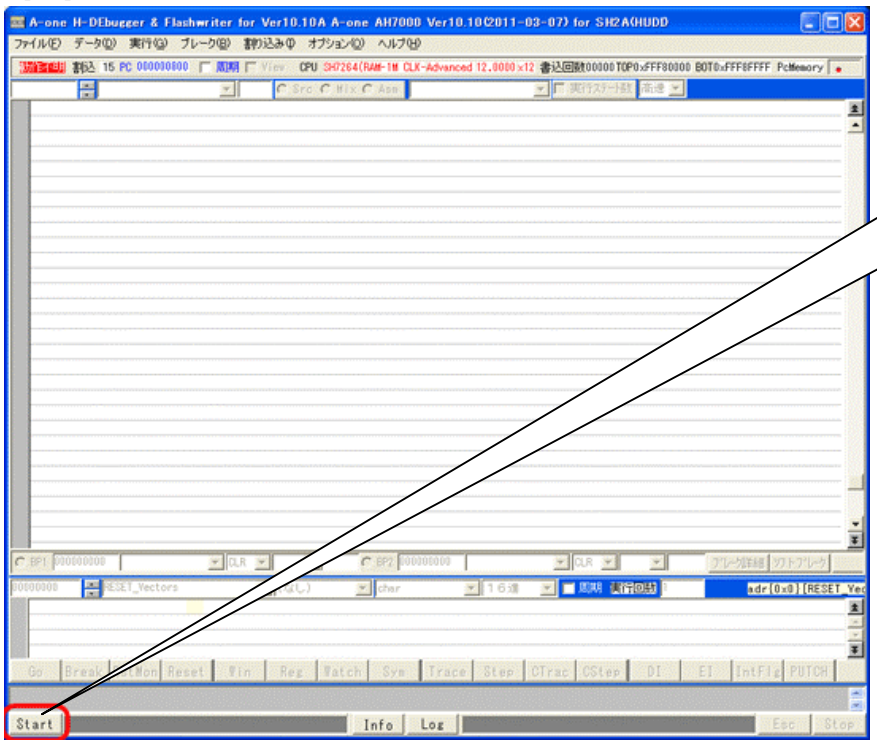
ＳＨ７２６４の場合
開始　0xfff80000　　サイズ　0x10000
になります。

作成した「FlashROM」プログラムのデバッグにＢＳＣ設定が必要な場合は、スクリプトファイルを指定します。

ＳＨ７２６４の場合
インストールディレクトリ
"c:¥Program Files¥Aone¥DEF¥rom-custom¥SH7264"
に、例として「SH7264-BSC-HSB.log」が用意してありますので、目的ハードにカスタマイズして下さい。

＜ー「設定」をクリックします。

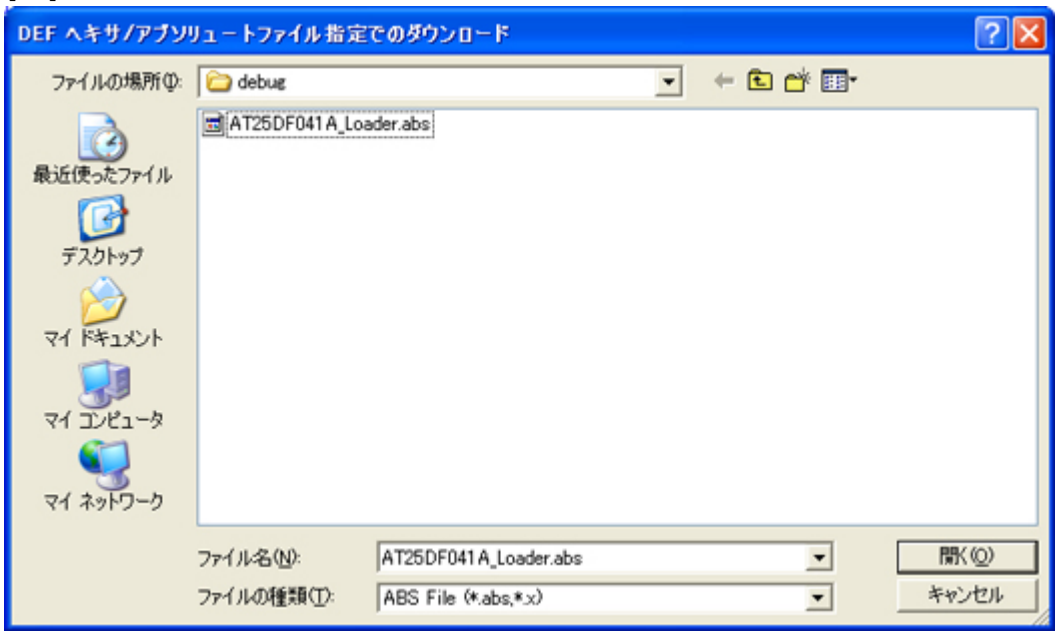
［3-2］



ＤＥＦ画面、左下隅の「Start」をクリックします。

２．作成したＦｌａｓｈROMソフトのデバッグを開始する準備をする。

[3-3]

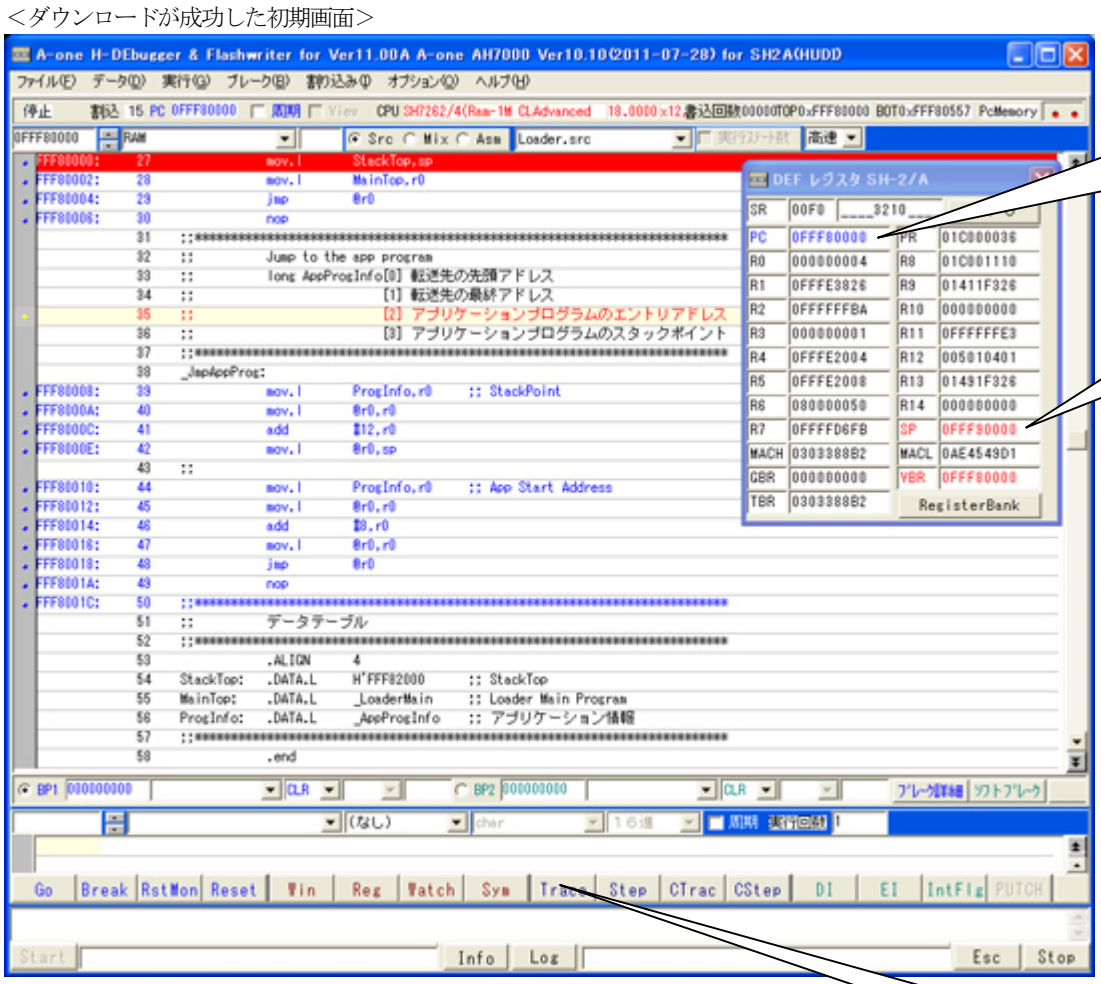


ＤＥＦメニュー
＜ファイル＞－＜ダウンロード＞
で、ダウンロードします。

インストールディレクトリ
"c:¥Program Files¥Aone¥DEF¥rom-custom¥SH7264"
下の"$(ProjectName)¥Debug"
に作成したアブソリュートファイルがありますので指定します。
例）$(ProjectName):AT25DF041A_Loader
「AT25DF041A_Loader.abs」

[3-4]

＜ダウンロードが成功した初期画面＞

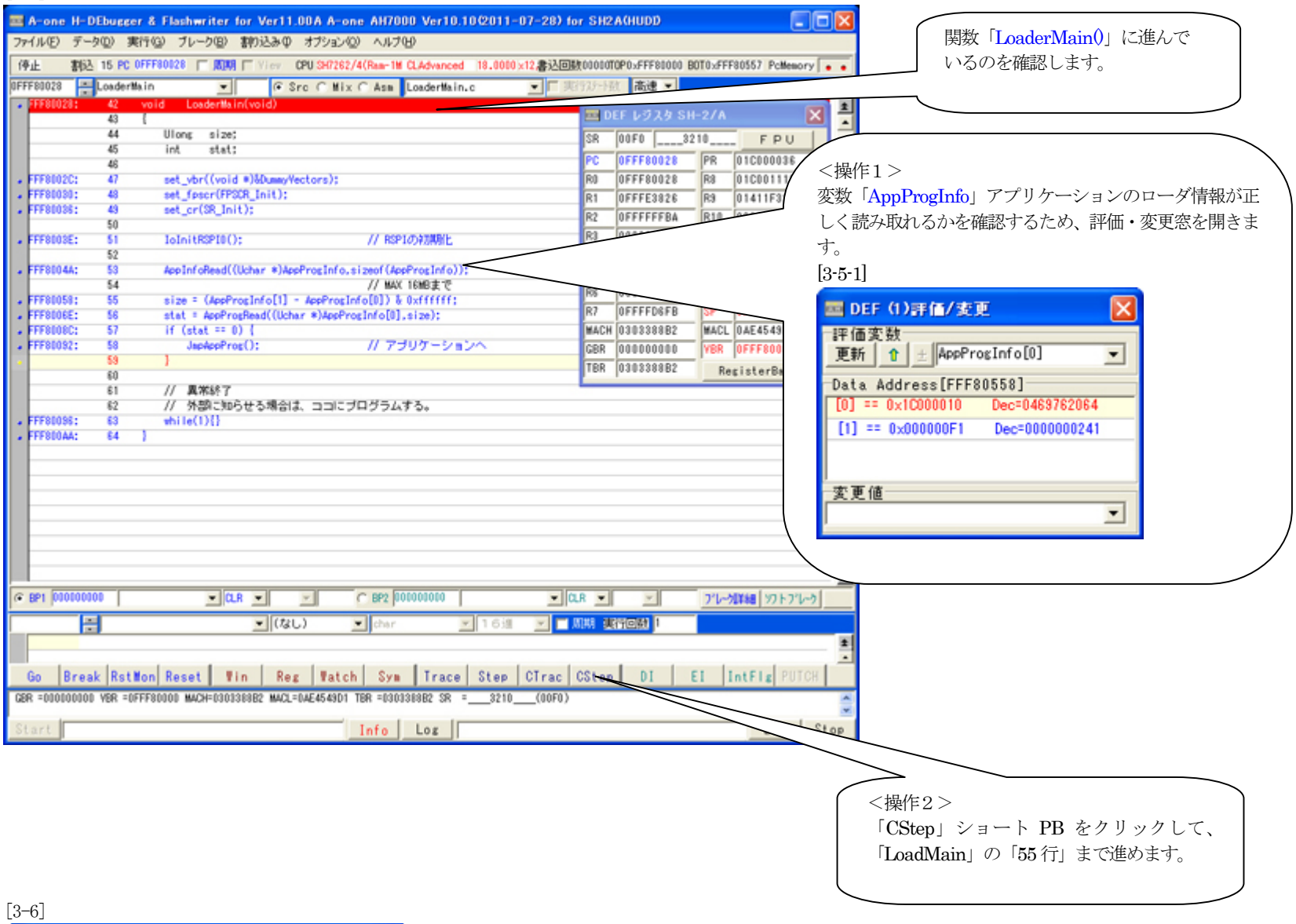


PC レジスタ値が
先頭アドレス[「0xFFF80100」になっていることが確認出来ます。

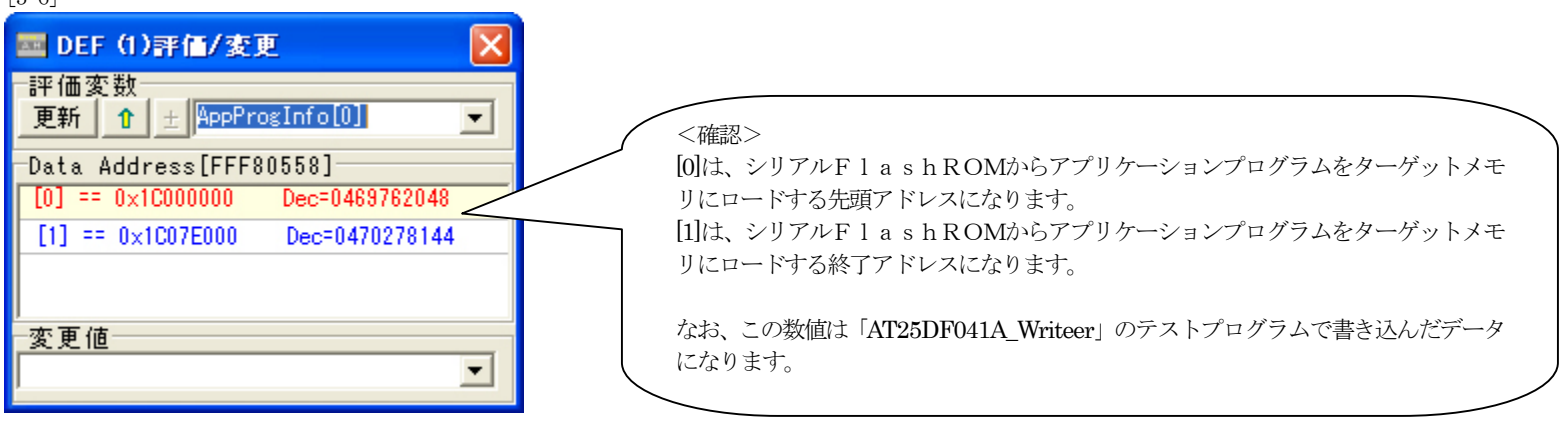
SP レジスタ値が内蔵 RAM のボトムアドレスになっていることが確認できます。

＜操作＞
「Trace」ショート PB をクリックして、
関数「LoaderMain()」まで進めます。

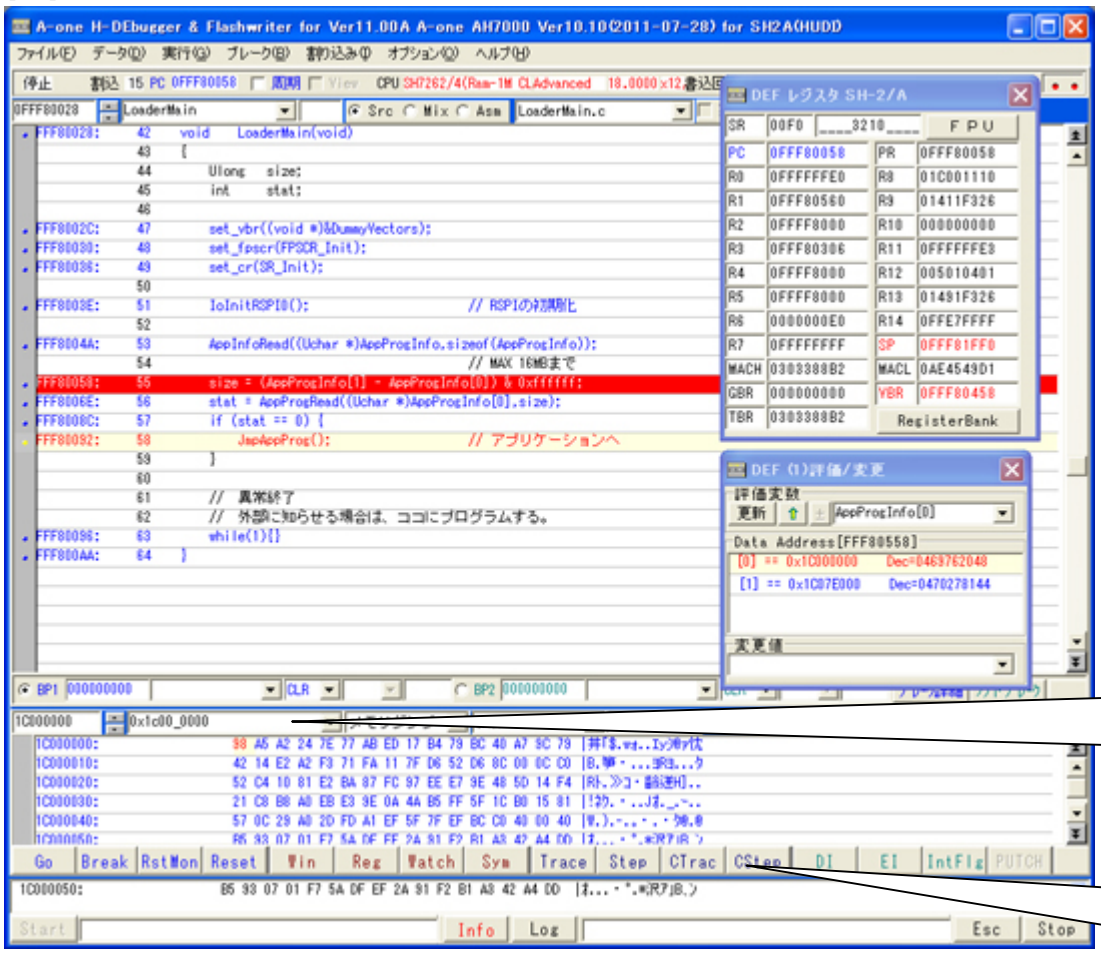
[3-5]


関数「LoaderMain()」に進んでいるのを確認します。
＜操作１＞
変数「AppProgInfo」アプリケーションのローダ情報が正しく読み取れるかを確認するため、評価・変更窓を開きます。
[3-5-1]
DEF (1)評価/変更
評価変数
更新
AppProgInfo[0]
Data Address[FFF80558]
[0] == 0x1C000010 Dec=0469762064
[1] == 0x000000F1 Dec=0000000241
変更値
＜操作２＞
「CStep」ショート PB をクリックして、「LoadMain」の「55 行」まで進めます。

[3-6]


DEF (1)評価/変更
評価変数
更新
AppProgInfo[0]
Data Address[FFF80558]
[0] == 0x1C000000 Dec=0469762048
[1] == 0x1C07E000 Dec=0470278144
変更値
＜確認＞
[0]は、シリアルＦｌａｓｈROMからアプリケーションプログラムをターゲットメモリにロードする先頭アドレスになります。
[1]は、シリアルＦｌａｓｈROMからアプリケーションプログラムをターゲットメモリにロードする終了アドレスになります。
なお、この数値は「AT25DF041A_Writeer」のテストプログラムで書き込んだデータになります。

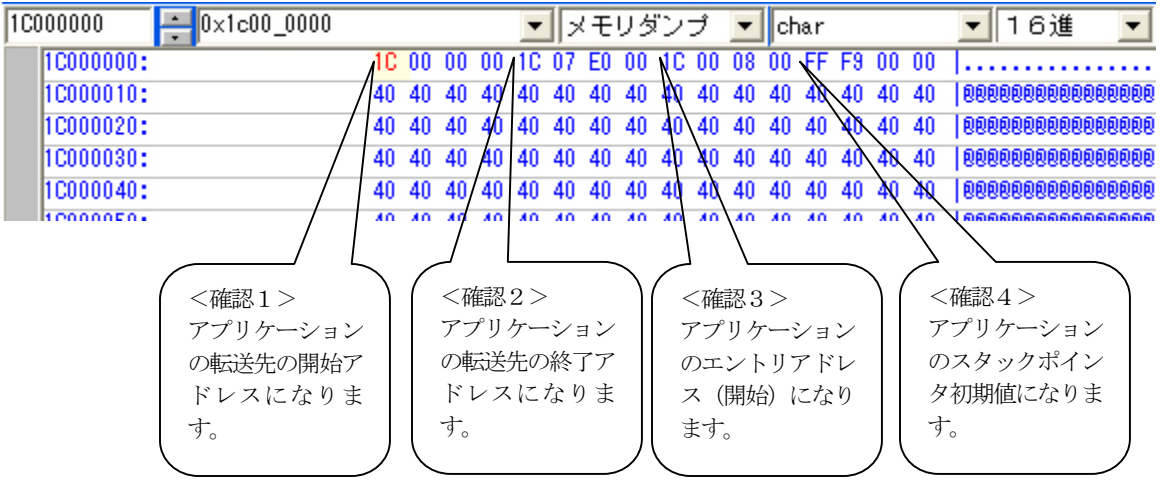
[3-7]



＜操作１＞
ローダプログラムが正しく読み出せたかを確認するため、ダンプアドレスに「0x1C00_0000」を指定しておきます。

＜操作２＞
「CStep」ショート PB をクリックして、「LoadMain」の「58 行」まで進めます。

[3-7-1]



＜確認１＞
アプリケーションの転送先の開始アドレスになります。

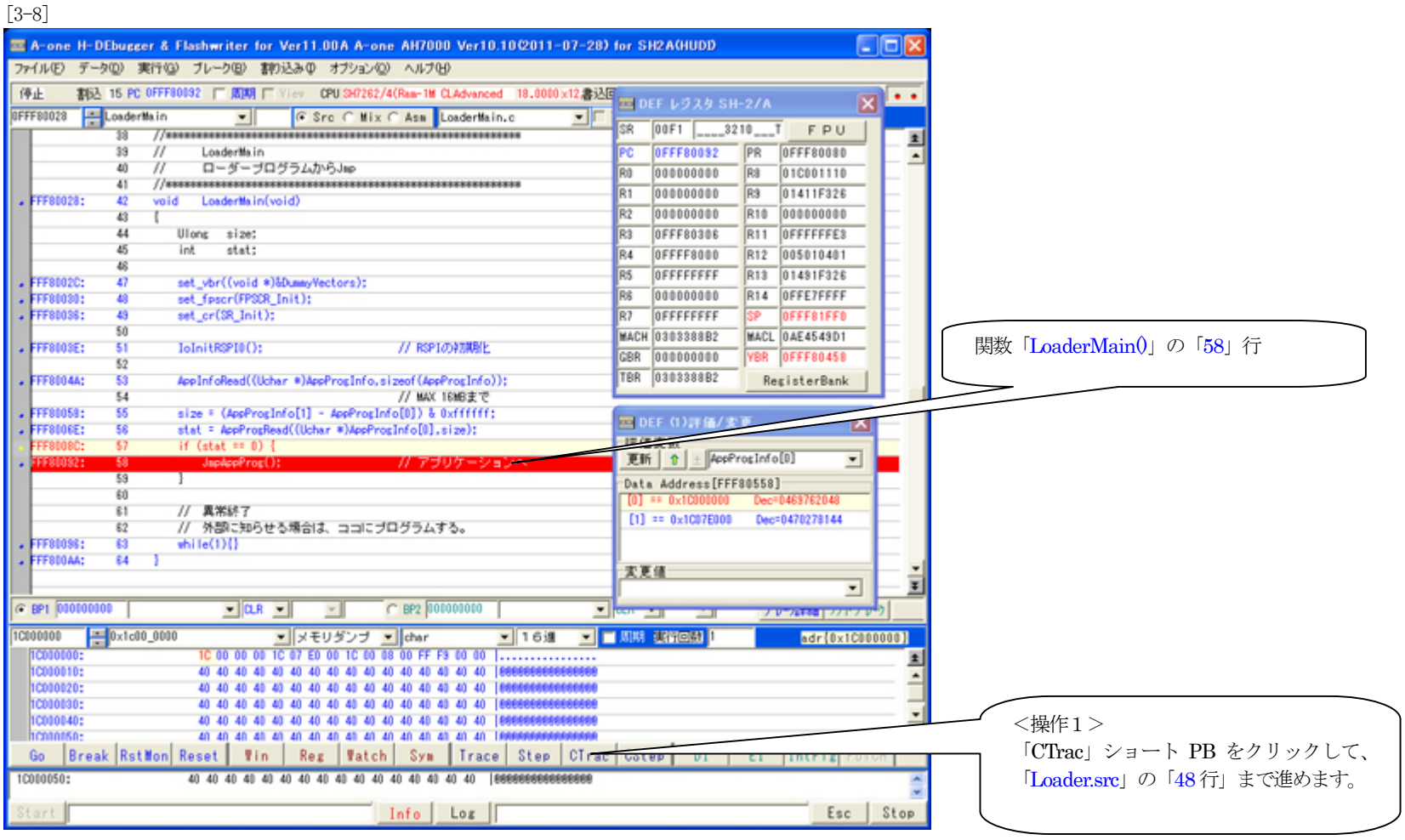
＜確認２＞
アプリケーションの転送先の終了アドレスになります。

＜確認３＞
アプリケーションのエントリアドレス（開始）になります。

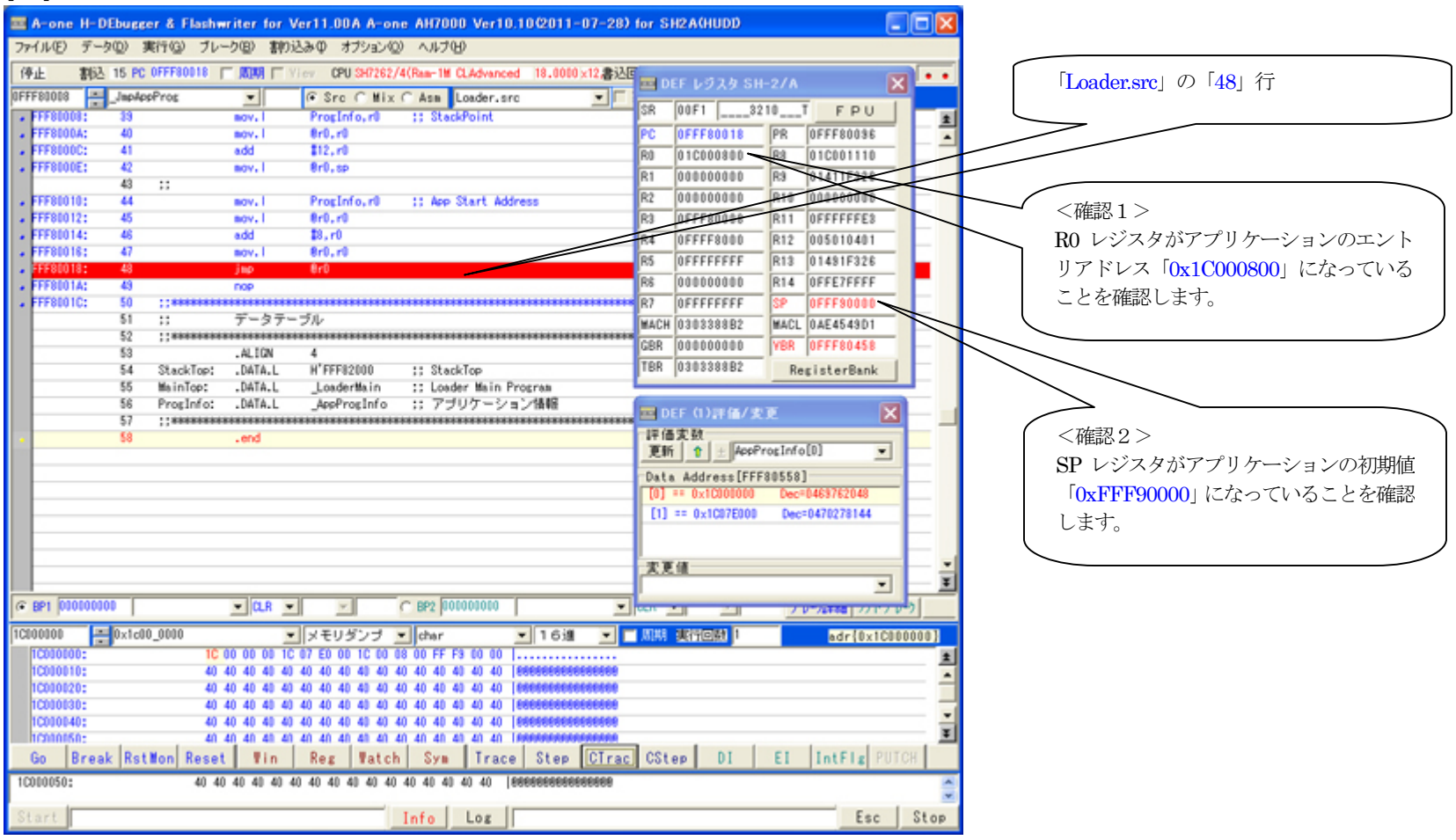
＜確認４＞
アプリケーションのスタックポインタ初期値になります。

この数値は「AT25DF041A_Writeer」のテストプログラムで書き込んだデータになります。

[3-8]



[3-9]



ローダプログラムの確認は終了です。

⚠ 3．御願い

本説の方法で、フラッシュＲＯＭ用ローダプログラムを追加した場合、必ず、プロジェクトのバックアップすることを御願いします。
今回は「c:¥Program Files¥Aone¥DEF¥rom_custom」で作成する例で記述しましたが「¥rom_custom」をホルダごと別のディレクトリに貼り付けても作成できます。
つまり、ユーザーアプリのプロジェクトごとに管理するのも一案かもしれません。追加作成したプロジェクトは、ユーザー様の責任のもとで管理願います。

以上で、シリアルＦｌａｓｈＲＯＭ用ローダプログラムの追加作業が終了です。