ARM® コンパイラ

バージョン 6.02

エラーおよび警告リファレンスガイド

ARM

ARM DUI0807CJ

ARM® コンパイラ

エラーおよび警告リファレンスガイド

リリース情報

ドキュメント履歴

発行	日付	機密保持ステータス	変更点
A	14 3 月 2014	非機密扱い	ARM コンパイラ v6.00 リリース
B	15 12 月 2014	非機密扱い	ARM コンパイラ v6.01 リリース
C	30 6 月 2015	非機密扱い	ARM コンパイラ v6.02 リリース

非機密著作権情報

LES-PRE-20349

機密保持ステータス

本書は非機密扱いであり、本書を使用、複製、および開示する権利は、ARM および ARM が本書を提供した当事者との間で締結した契約の条項に基づいたライセンスの制限により異なります。

無制限アクセスは、ARM 社内による分類です。

製品ステータス

本書の情報は最終版であり、開発済み製品に対応しています。

Web アドレス

http://www.jp.arm.com

目次
ARM® コンパイラ エラーおよび警告リファレンスガイド

序章

この前書きでは、次について紹介します。*ARM® コンパイラ エラーおよび警告リファレンスガイド*.

このドキュメントは、次で構成されています。

- *本書について*(6 ページ).

本書について

『ARM® コンパイラエラーおよび警告リファレンスガイド』には、各コンパイルツールによって生成されるエラーと警告のリストが記載されています。armclang によって生成されるエラーと警告は記載されていません。

本書の構成

本書は以下の章から構成されています。

第 章 1 アセンブラのエラーおよび警告
アセンブラ（armasm）のエラーおよび警告メッセージについて説明します。

第 章 2 リンカのエラーおよび警告
リンカ（armlink）のエラーおよび警告メッセージについて説明します。

第 章 3 ELF イメージ変換ユーティリティのエラーおよび警告
ELF イメージ変換ユーティリティ（fromelf）のエラーおよび警告メッセージについて説明します。

第 章 4 ライブラリアンのエラーおよび警告
ARM ライブラリアン（armar）のエラーおよび警告メッセージについて説明します。

第 章 5 その他のエラーおよび警告
ツールのいずれかによって表示されるその他のエラーおよび警告メッセージについて説明しています。

用語集

「ARM 用語集」は、ARM マニュアルで使用されている用語とその定義のリストです。一般に認められている意味と ARM での意味が異なる場合を除いて、「ARM 用語集」に業界標準の用語は含まれていません。

詳細については、「*ARM 用語集*」を参照して下さい。

表記規則

italic
重要用語、相互参照、引用箇所を示します。

bold
メニュー名などのユーザインタフェース要素を太字で記載しています。また、必要に応じて記述リスト内の重要箇所、ARM プロセッサの信号名、重要用語、および専門用語にも太字を使用しています。

`monospace`
コマンド、ファイル名、プログラム名、ソースコードなど、キーボードから入力可能なテキストを示しています。

`monospace`
コマンドまたはオプションに使用可能な略語を示しています。コマンド名またはオプション名をすべて入力する代わりに、下線部分の文字だけを入力することができます。

`monospace italic`
引数が特定の値で置き換えられる場合のモノスペーステキストの引数を示しています。

`monospace bold`
サンプルコード以外に使用される言語キーワードを示しています。

`<` および `>`
コードまたはコードの一部のアセンブラ構文で置換可能な項が使用されている場合に、その項を囲みます。以下はその一例です。

```
MRC p15, 0, &lt;Rd&gt;, &lt;CRn&gt;, &lt;CRm&gt;, &lt;Opcode_2&gt;
```

スモールキャピタル
「*ARM 用語集*」で定義されている専門的な意味を持つ用語について、本文中で使用されます。例えば、IMPLEMENTATION DEFINED、IMPLEMENTATION SPECIFIC、UNKNOWN、UNPREDICTABLE などです。

ご意見、ご感想

本製品に関するフィードバック

本製品についてのご意見やご提案がございましたら、以下の情報を添えて購入元までお寄せ下さい。
- 製品名
- 製品のリビジョンまたはバージョン
- 説明にはできるだけ多くの情報を含めて下さい。適宜、症状と診断手順も含めて下さい。

内容に関するフィードバック

内容に関するご意見につきましては、電子メールを *errata@arm.com* まで送信して下さい。その際には、以下の内容を記載して下さい。

- タイトル
- 文書番号（ARM DUI0807CJ）。
- 問題のあるページ番号
- 問題点の簡潔な説明

また、補足すべき点や改善すべき点についての全般的なご提案もお待ちしております。

—————— **注** ——————

ARM では、この PDF を Adobe Acrobat および Acrobat Reader でのみテストしており、その他の PDF リーダーを使用した場合の表示品質については、保証いたしかねます。

————————————

その他の情報

- *ARM Information Center*.
- *ARM Technical Support Knowledge Articles*.
- *Support and Maintenance*.
- *ARM Glossary*.

第 1 章
アセンブラのエラーおよび警告

アセンブラ（armasm）のエラーおよび警告メッセージについて説明します。

以下のセクションから構成されています。

- *1.1 armasm のエラーおよび警告メッセージのリスト*(1-9 ページ).

1.1 armasm のエラーおよび警告メッセージのリスト

armasm で生成されるエラーおよび警告メッセージのリスト

A1017E : :INDEX:PC 相対式では使用できません
:INDEX:式演算子が、PC 相対式(通常はプログラムラベル)に適用されました。:INDEX:は、レジスタ相対式での、ベースレジスタからのオフセットを返します。

<areaname> というエリア内の <label> というラベルのオフセットを取得する場合は、<label> - <areaname> という形式で指定します。

『*armasm ユーザガイド*』の以下のセクションを参照して下さい。

単項演算子。

A1020E : 不正な事前定義:<directive>
--predefine(-pd)コマンドラインオプションのオペランドが認識されませんでした。ディレクティブは、スペースを含んでいる場合は引用符で囲む必要があります。例えば、Windows では次のようにします。

```
--predefine "versionnum SETA 5"
```

SETS ディレクティブを使用する場合は、ディレクティブの引数も引用符で囲む必要があります。引用符は、オペレーティングシステムおよびシェルに応じてエスケープしなければなりません。以下に例を示します。

```
--predefine "versionstr SETS \"5A\""
```

A1021U : 入力ファイルがありません
コマンドラインで、入力ファイルが指定されませんでした。原因としては、引数を囲む引用符のうち、終了側の引用符がなかったことが考えられます。

A1023E : ファイル "<filename>" を開けませんでした:<reason>

A1024E : ファイル "<filename>" をロードできませんでした:<reason>

A1042E : 認識されない APCS 修飾子 '<qualifier>'
--apcs コマンドラインオプションに指定された引数にエラーがあります。<qualifier> のスペルを確認して下さい。

A1051E : --depend ファイル '<filename>' を開けません:<reason>

A1055E : --errors ファイル '<filename>' を開けません:<reason>

A1056E : ターゲットの cpu '<cpu>' が認識されません
--cpu コマンドラインオプションで指定された名前が、認識されているプロセッサ名ではありません。引数のスペルを確認して下さい。

サポートされているプロセッサおよびアーキテクチャのリストを表示するには、--cpu=list を使用します。

A1067E : 出力ファイルが '<filename1>' として指定されましたが、既に '<filename2>' として指定されています
コマンドラインで、複数の出力ファイル(-o filename)が指定されました。コマンドラインオプションのスペルの間違いが原因である可能性があります。

A1071E : リストファイル '<filename>' を開けません:<reason>
--list <filename> コマンドラインオプションで指定されたファイルが開けませんでした。この例外は、以下のいずれかの理由で発生することがあります。

- 指定された名前が無効
- スペースがない
- 同じ名前の読み取り専用ファイルが既に存在する
- ファイルが別のプロセスによって使用されている

ファイルのパスが正しく指定されていることを確認して下さい。

A1072E : リストファイル '<filename>' として .s ファイルまたは .o ファイルを指定することはできません
--list コマンドラインオプションの filename 引数が、ソースファイルまたはオブジェクトファイルであることを示す拡張子を持っています。原因としては、コマンドラインで filename 引数を指定し忘れたことが考えられます。--list コマンドラインオプションに正しい引数が指定されていることを確認して下さい。

A1073E : 出力ファイル '<filename>' としてソースファイルを指定することはできません
コマンドラインで指定されたオブジェクトファイルが、ソースファイルであることを示すファイル名拡張子を持っています。原因としては、コマンドラインでオブジェクトファイル名を指定し忘れたことが考えられます。

A1074E : 依存ファイル '<filename>' としてソースファイルを指定することはできません
--depend コマンドラインオプションの filename 引数が、ソースファイルであることを示す拡張子(.s)を持っています。原因としては、コマンドラインで filename 引数を指定し忘れたことが考えられます。正しい引数が指定されていることを確認して下さい。

A1075E : エラーファイル '<filename>' としてソースファイルを指定することはできません
--errors コマンドラインオプションの filename 引数が、ソースファイルであることを示す拡張子(.s)を持っています。原因としては、コマンドラインで filename 引数を指定し忘れたことが考えられます。正しい引数が指定されていることを確認して下さい。

A1085W : 強制ユーザモード LDM/STM の後にバンク付きの R8-R14 を使用することはできません
ARM アーキテクチャでは、ユーザレジスタ LDM または STM のすぐ後に続く命令でバンクレジスタにアクセスすることはできません。NOP を LDM または STM の直後に追加すると、このエラーを回避することができます。

以下に例を示します。

```
stmib sp, {r0-r14}^ ; Return a pointer to the frame in a1. mov r0, sp
```

変更後:

```
stmib sp, {r0-r14}^ ; Return a pointer to the frame in a1. nop mov r0, sp
```

A1088W : エリア AREA の宣言が正しくありません |$$$$$$$|
このメッセージは AREA ディレクティブが指定されていないと生成されます。メッセージ番号 A1105E も参照して下さい。

A1099E : 構造体スタックが最大スタックサイズ <max> をオーバーフローしています

A1100E : 構造体スタックがアンダーフローしています

A1105E : Area ディレクティブがありません
このメッセージは AREA ディレクティブが指定されていないと生成されます。メッセージ番号 A1088W も参照して下さい。

A1106E : コンマがありません

A1107E : 不正なシンボル型。ラベルが期待されます。

A1108E : シンボル '<name>' が複数回定義されています

A1109E : 不正な式の型

A1110E : 定数式が期待されます
例えば SETA の後には、定数式を指定する必要があります。

『armasm ユーザガイド』の以下のセクションを参照して下さい。

数値式。

A1111E : 定数式またはアドレス式が期待されます

A1112E : アドレス式が期待されます

A1113E : ストリング式が期待されます
例えば SETS の後には、ストリング式を指定する必要があります。

『armasm ユーザガイド』の以下のセクションを参照して下さい。

文字列式。

A1114E : レジスタ相対式が期待されます

A1116E : ストリングのオペランドは DCB に対してのみ指定できます

A1117E : レジスタシンボル '<name>' は既に定義されています
A1118E : 現在のマクロ拡張がありません
A1119E : 条件内では MEND を使用できません
MEND は "END of Macro" の意味であり、英語の単語 "mend" ではありません。

『armasm ユーザガイド』の以下のセクションを参照して下さい。

マクロについて。

A1120E : 不正なグローバル名
A1121E : グローバル名 '<name>' は既に存在しています
A1122E : ローカル変数は、マクロ外では使用できません
A1123E : 不正なローカル名
A1125E : グローバル/ローカルシンボル '<name>' の型が不明または間違っています
A1126E : 不正なアライメント境界です。2 の倍数にする必要があります
A1127E : 不正な IMPORT/EXTERN 名
A1128E : 共通名 '<sym>' は既に存在しています
A1129E : インポートされた名前 '<sym>' は既に存在しています
A1130E : エクスポートされた名前が不正です
A1131E : エクスポートされたシンボル '<sym>' のシンボル型が不正です
A1132E : REQUIRE ディレクティブは <entity> 形式の出力ではサポートされていません
A1133E : 必須のシンボル名が不正です
A1134E : 必須のシンボル型が不正です。シンボルが外部またはラベルで、かつシンボルが再配置可能および絶対であることが期待されます。
A1135E : エリア名がありません
アルファベット以外の文字で始まる AREA 名は縦棒で囲む必要があります。例えば、次のような名前の場合、

```
AREA 1_DataArea, CODE, READONLY
```

次のように変更します。

```
AREA |1_DataArea|, CODE, READONLY
```

A1136E : エントリアドレスは既に設定されています
A1137E : 行の終わりに予期しない文字があります
この問題は、命令の行に、命令の一部ではない余分な文字が発見された場合に発生します。

以下に例を示します。

```
ADD r0, r0, r1 comment
```

以下のように変更できます。

```
ADD r0, r0, r1 ; comment
```

A1138E : 文字列 "<string>" は操作に対して短すぎます。長さは <oplength> より大きくする必要があります。
A1139E : 文字列がオーバーフローしています。文字列が <max> 文字を超えています。
A1140E : 不正なオペランド型
A1141E : 再配置された式のみを加算または減算できます
A1142E : 減算による再配置は <entity> 形式の出力ではサポートされていません
この問題は、次のように、減算シンボルが別のエリアにある場合に発生します。

```
IMPORT sym1
IMPORT sym2
DCD (sym2 - sym1)
```

A1143E : COMMON ディレクティブは %s 形式の出力ではサポートされていません
A1144E : DCDO ディレクティブは %s 形式の出力ではサポートされていません
A1145E : エクスポートされたシンボル '<sym>' は定義されていません
A1146E : 出力ファイル <filename> を開けません:<reason>
A1147E : 不正なシフト名

A1148E : 不明なシフト名 <name>。LSL、LSR、ASR、ROR、RRX のいずれかが期待されます。

A1150E : 不正なシンボル。定義されていないか、外部です。

この問題は、通常、以下の場合に発生します。

- 現在のファイルが、いくつかのシンボルを定義するために、他のファイルの INCLUDE を必要とする場合。以下に例を示します。

```
"init.s", line 2: Error: A1150E: Bad symbol
2 00000000 DCD EBI_CSR_0
```

この場合、必須の定義ファイルを含めることで解決できます。以下に例を示します。

```
INCLUDE targets/eb40.inc
```

- 現在のファイルが、いくつかのシンボルのために、IMPORT を必要とする場合。以下に例を示します。

```
"init.s", line 4: Error: A1150E: Bad symbol
4 00000000 LDR r0, =||Image$$RAM$$ZI$$Limit||
```

この場合、必須のシンボルをインポートすることで解決できます。以下に例を示します。

```
IMPORT ||Image$$RAM$$ZI$$Limit||
```

A1151E : 不正なレジスタ名シンボル

例:

```
MCR p14, 3, R0, Cr1, Cr2
```

コプロセッサレジスタ CR は、ビルドのコードでは小文字(c)でラベル付けする必要があります。ARM レジスタは、以下のように、r でも R でもかまいません。

```
MCR p14, 3, r0, c1, c2
```

または

```
MCR p14, 3, R0, c1, c2
```

A1152E : 予期しない演算子

A1153E : 未定義のシンボル

A1154E : 予期しないオペランド。演算子が期待されます。

A1155E : <operator> と等しい予期しない単項演算子

A1156E : 開き括弧がありません

A1157E : ディレクティブの後に構文エラーがあります

A1158E : 行の先頭が不正です。先頭は空白にする必要があります。

一部のディレクティブ(ENTRY、IMPORT、EXPORT、GET など)は、行の先頭にラベルがない行でしか使えません。ラベルが存在する場合は、このエラーが発生します。

A1159E : 行の先頭にラベルがありません

一部のディレクティブ(FUNCTION、SETS など)は、行の先頭にラベルが必要です。以下に例を示します。

```
my_func FUNCTION
```

または

```
label SETS
```

ラベルが存在しない場合は、このエラーが発生します。

A1160E : 不正なローカルラベル番号

数値のローカルラベルは範囲 0 ～ 99 の数値で、その後に名前を付けることもできます。

*『armasm ユーザガイド』*の以下のセクションを参照して下さい。

数値のローカルラベル。

A1161E : ローカルラベル定義の後に構文エラーがあります

A1162E : 不正なルーチン名 '<name>'
A1163E : 不明なオペコード <name> です。オペコードまたはマクロが期待されます。
この最も一般的な原因は、以下のとおりです。

- 命令の前の左余白にホワイトスペースを入れるのを忘れている。例えば、次の命令は、

```
MOV PC,LR
```

次のように変更します。

```
MOV PC,LR
```

- --fpu スイッチを使わずに、ハードウェア浮動小数点命令を使用している。以下に例を示します。

```
FMXR FPEXC, r1 ;
```

これは、 armasm --fpu=vfp でアセンブルする必要があります。

- オペコードの入力を間違えた。例えば、

```
ADDD
```

以下は、正しくない例です。

```
ADD
```

A1164E : オペコードは選択されたプロセッサではサポートされていません
armasm コマンドラインで選択されたプロセッサでは、この命令はサポートされていません。

以下を参照してください。

ARM アーキテクチャリファレンスマニュアル。

A1165E : 実際のパラメータが多すぎます。<actual> パラメータが期待されます。
A1166E : ラベルの後に構文エラーがあります
A1167E : 行の先頭が無効です
A1168E : プレインデクス形式では変換できません
A1169E : 閉じ角括弧がありません
A1170E : イミディエート 0x<adr> がこの操作の範囲外です。(0x<adr>)未満である必要があります。
このエラーは、DCB、DCW、または DCWU ディレクティブを大きすぎるイミディエートと共に使用した場合に発生します。
*『armasm ユーザガイド』*の以下のセクションを参照して下さい。

- *DCB*。
- *DCW および DCWU*。

A1171E : 閉じ括弧がありません
A1172E : 不正なローテータ <rotator> です。ローテータは偶数かつ 0 以上 30 以下である必要があります。
A1173E : ADR/L は外部シンボルでは使用できません
ADR および ADRL 疑似命令は、同じコードエリア内のラベルと共に使う場合のみ使用できます。レジスタにエリア外のアドレスをロードするには、代わりに LDR を使用して下さい。
A1174E : データ転送オフセット 0x<val> が範囲外です。使用可能な値は 0x<min> ～ 0x<max> です
A1175E : 不正なレジスタ範囲
A1176E : 分岐のオフセット 0x<val> が範囲外です。使用可能な値は 0x<min> ～ 0x<max> です
分岐は PC 相対で、範囲の制限があります。数値のローカルラベルを使用している場合は、ROUT ディレクティブを使うとその有効範囲が制限できます。これにより、誤って違うラベルが参照されるのを防ぐことができます。

*『armasm ユーザガイド』*の以下のセクションを参照して下さい。

数値のローカルラベル。

A1179E : 不正な 16 進数
A1180E : 閉じ引用符がありません
A1181E : 不正な演算子
A1182E : ベースとなる <base> 数が不正です
A1183E : 数値オーバーフロー
A1184E : 外部シンボルは式内では有効ではありません
A1185E : シンボルがありません
A1186E : データエリア内で生成されたコード
命令がデータエリア内にアセンブルされました。この問題は、AREA ディレクティブで CODE 属性を省略した場合に発生することがあります。

*『armasm ユーザガイド』*の以下のセクションを参照して下さい。

AREA。

A1187E : マクロのパラメータにエラーがあります
A1188E : レジスタ値 <val> は範囲外です。使用可能な値は <min> ～ <max> です
A1189E : '#' がありません
A1190E : 予期しない '<entity>'
A1191E : 浮動小数点レジスタ番号が範囲(0 ～ <maxi>)を超えています
A1192E : コプロセッサレジスタ番号が範囲(0 ～ 15)を超えています
A1193E : コプロセッサ番号が範囲(0 ～ 15)を超えています
A1194E : 不正な浮動小数点数
A1195W : 小さな浮動小数点値は 0.0 に変換されます
A1196E : 浮動小数点数を禁止するには遅すぎます
A1198E : 不明なオペランド
この問題は、オペランドが誤って入力された場合に発生します。

以下に例を示します。

```
armasm --cpu=8-A.64 init.s -g -PD "ROM_RAM_REMAP SETL {FALS}"
```

これは、以下のようにする必要があります。

```
armasm --cpu=8-A.64 init.s -g -PD "ROM_RAM_REMAP SETL {FALSE}"
```

*『armasm ユーザガイド』*の以下のセクションを参照して下さい。

アセンブリ時の変数代入。

A1199E : コプロセッサ命令が範囲(0 ～ <maxi>)を超えています
A1200E : 構造体が一致しません。While/Wend が期待されます。
A1201E : 代入された行が長すぎます。最大長は <max> です。
A1202E : 代入されたシンボル '<name>' の宣言がありません
*『armasm ユーザガイド』*の以下のセクションを参照して下さい。

アセンブリ時の変数代入。

A1203E : マクロプロトタイプのラベルパラメータの先頭が不正です
A1204E : 不正なマクロパラメータのデフォルト値
A1205E : レジスタ <reg> がリスト内に複数あります
A1206E : レジスタは、レジスタ番号の昇順で表示されている必要があります
この警告は、例えば LDM や STM 命令で、レジスタが昇順で指定されておらず、--checkreglist オプションが使用されている場合に発生します。

A1207E : 不正または不明な属性
このエラーは、AREA ディレクティブで無効な属性が指定された場合に発生します。以下に例を示します。

```
AREA test,CODE,READONLY,HALFWORD
```

HALFWORD は無効なので、削除します。

*『armasm ユーザガイド』*の以下のセクションを参照して下さい。

AREA。

A1209E : ADRL は PC と共にデスティネーションとして使用することはできません
A1210E : 初期化されていないエリア '<name>' にゼロ以外のデータがあります
A1211E : 開き角括弧がありません
A1212E : ゼロによる除算
A1213E : 属性 <entity> を属性 <entity> と共に使用することはできません
A1214E : シンボル '<sym>' をレジスタリストとして定義するには遅すぎます
A1215E : 不正なレジスタリストシンボル
A1216E : 不正なストリングエスケープシーケンス
A1217E : コードファイル '<codeFileName>' への書き込み中にエラーが発生しました:<reason>
A1219E : APSR 指定子、CPSR 指定子、または SPSR 指定子が不正です
以下に例を示します。

```
MRS r0, PSR
```

この場合、以下のように、どのステータスレジスタ(CPSR または SPSR)を使用するかを指定する必要があります。

```
MRS r0, CPSR
```

A1220E : BLX <address> は無条件にする必要があります
A1221E : エリアの属性 '<entity>' は <entity> オブジェクトのファイル形式ではサポートされていません
A1223E : Comdat シンボル '<name>' は定義されていません
A1224E : <entity> 形式では、エリア間の PC 相対データ転送は実行できません
A1225E : ASSOC 属性は非 comdat エリアでは使用できません
A1226E : SELECTION 属性は非 comdat エリアでは使用できません
A1227E : Comdat 関連エリア '<name>' はこの時点でファイル内に定義されていません
A1228E : Comdat 関連エリア '<name>' はエリア名ではありません
A1229E : COMDAT シンボルがありません
A1230E : Missing '}' after COMDAT symbol
A1234E : シンボル '<sym>' の未定義または未エクスポートの Weak エイリアス
A1237E : レジスタまたはレジスタの組み合わせがこの操作に対して無効です
A1238E : この操作で使用する場合、イミディエート値はワード境界で整列している必要があります
A1240E : イミディエート値はこの操作では使用できません
A1241E : この操作ではイミディエート値を使用する必要があります
A1242E : この操作で使用する場合、オフセットはワード境界で整列している必要があります
A1243E : この操作では、オフセットはハーフワード境界で整列している必要があります
A1244E : '!' がありません
A1245E : Thumb コードから ARM コードへの B または BL です
A1246E : ARM コードから Thumb コードへの B または BL です
A1247E : ARM コードから ARM コードへの BLX です。BL を使用してください。
この問題は、このアセンブラファイル内に A32 コードから A32 コードへの BLX *label* 分岐がある場合に発生します。これが許可されないのは、BLX *label* では常に命令セットの状態変更が起きるためです。通常は、代わりに BL を使用することで解決できます。
A1248E : Thumb コードから Thumb コードへの BLX です。BL を使用してください。
この問題は、このアセンブラファイル内に T32 コードから T32 コードへの BLX *label* 分岐がある場合に発生します。これが許可されないのは、BLX *label* では常に命令セットの状態変更が起きるためです。通常は、代わりに BL を使用することで解決できます。

A1249E : ポストインデクスアドレッシングモードは使用できません
A1250E : プレインデクスアドレッシングモードはこの命令では使用できません。[Rn, Rm] を使用してください。
A1253E : 外部シンボルへの Thumb 分岐は再配置できません:<fmt> では表現できません
A1254E : ハーフワードリテラル値はサポートされていません

次の例では、LDRH を LDR に変更します。LDR は、レジスタに定数をロードする標準的な方法です。

```
LDRH R3, =constant
```

A1256E : DATA ディレクティブは CODE エリアでのみ使用できます
A1259E : 無効な PSR フィールド指定子です。構文は <PSR>_ で、<PSR> には CPSR または SPSR が入ります。
A1260E : PSR フィールド '<entity>' が複数回指定されています
A1261E : MRS ではフィールドを選択できません。直接 APSR、CPSR、または SPSR を使用してください

この問題は、MRS 命令と共に CPSR または SPSR のフィールドを使おうとした場合に発生します。以下に例を示します。

```
MRS r0, CPSR_c
```

A1262U : 式記憶エリアアロケータがエラーになりました
A1265U : 構造体が一致しません:INCLUDE ファイルの最後で IF または WHILE が一致しません
A1267E : ファイル <filename> に対して不正な GET または INCLUDE です
A1268E : 条件またはマクロが一致しません
A1269U : 構造体スタック上に予期しない GET があります
A1270E : ファイル "<entity>" が見つかりません
A1271E : 行が長すぎます。最大行長は <MaxLineLength> です。
A1272E : 入力ファイルの終わり
A1273E : '\\' はストリングの分割には使用できません
A1274W : コメントの最後に '\\' があります
A1283E : リテラルプールが離れすぎています。LTORG を使用して 1KB 以内になるようにアセンブルしてください。

T32 コードでは、リテラルプールを、アクセスを試みる LDR 命令の 1 KB 以内にしなければなりません。メッセージ A1284E および A1471W も参照して下さい。

A1284E : リテラルプールが離れすぎています。LTORG を使用して 4KB 以内になるようにアセンブルしてください。

A32 コードでは、リテラルプールを、アクセスを試みる LDR 命令の 4 KB 以内にしなければなりません。この問題を解決するには、LTORG ディレクティブをアセンブリソースコード内の適切な場所に追加します。
『*armasm ユーザガイド*』の以下のセクションを参照して下さい。

- *LDR Rd, =label を使用したレジスタへのアドレスのロード*。
- *LTORG*。

A1285E : 不正なマクロ名
A1286E : マクロは既に存在しています
A1287E : マクロプロトタイプのパラメータの先頭が不正です
A1288E : マクロプロトタイプのパラメータが不正です
A1289E : マクロプロトタイプのパラメータの区切り文字が無効です
A1290E : マクロ定義が大きすぎます。最大長は <max> です。
A1291E : マクロ定義を入れ子にすることはできません
A1310W : シンボル属性が認識されません
A1311U : 拡張内でマクロ定義が試行されました
A1312E : アサートがエラーになりました
A1313W : ファイルの終わりに END ディレクティブがありません

アセンブラは、ファイル内のコードがどこで終了するかを知るために、END ディレクティブを必要とします。このディレクティブの後には、自由な形式でコメントなどの情報を追加できます。

A1314W : 予約されている命令(NV 条件を使用)
A1315E : NV 条件はターゲットの CPU ではサポートされていません
A1316E : MSR に対するシフトされたレジスタオペランドの効果が定義されていません

A1319E : 影響が定義されていません(PC を Rs として使用)
A1320E : 影響が定義されていません(レジスタ指定シフトで PC を Rn または Rm として使用)
A1321E : 影響が定義されていません(PC をオフセットレジスタとして使用)
A1322E : PC の非境界整列転送です。転送先アドレスは 4 バイト境界で整列されている必要があります。それ以外の場合、予測不能な結果になります

このエラーは、LDR 命令を使用してワード境界で整列されていないアドレスから PC をロードしようとすると報告されます。以下に例を示します。

```
AREA Example, CODE LDR pc, [pc, #6] ; Error – オフセットは 4 の倍数である必要があります END
```

このコードよって、予期していなかった結果が生じます。

A1323E : 予約済みの命令(Rm = Rn とポストインデクスの併用)
A1324E : 影響が定義されていません(PC + ライトバック)
A1327E : 移植不可能命令(ライトバックとベースを使用する LDM がレジスタリストで指定されている場合、ベースの最終的な値は予期できません)

LDM では、ベースレジスタ <Rn> が <registers> で指定されていて、ベースレジスタのライトバックが指定されている場合、<Rn> の最終的な値は UNKNOWN です。

A1328E : 移植不可能命令(ライトバックとベースを使用する STM がレジスタリストの最初にない場合、ベースの格納値は予期できません)

STM 命令では、<Rn> が <registers> で指定されていて、ライトバックが指定されている場合、次のようになります。

- <Rn> が <register_list> で指定されている最も小さな番号のレジスタであるならば、<Rn> の元の値が格納されます。
- そうでないならば、<Rn> の格納値は UNKNOWN です。

A1329E : 予測不能な命令(ベースへのライトバックを使用した強制ユーザモード転送)

この問題は、PUSH {r0}^ のような命令によって発生します(^ は、ユーザレジスタへのアクセスを示します)。この命令ではベースレジスタへのライトバックはできません。ベースレジスタは、別途更新する必要があります。以下に例を示します。

```
SUB sp, sp,#4
STMID sp, {r0}^
```

また、STMFD R0!, {r13, r14}^ を次のように置き換える方法もあります。

```
SUB r0, r0,#8
STM r0, {r13, r14}^
```

メッセージ番号 A1085W も参照して下さい。

A1331E : 予測不能な命令(ソースまたはデスティネーションとしての PC)
A1332E : 影響が予測不能です(PC 相対 SWP)
A1334E : 影響が定義されていません(PC/PSR の使用)
A1335E : 意味のない命令(PC にライトバックすることはできません)
A1337E : 意味のない命令(PC はデスティネーションです)
A1338E : 疑わしい命令(PC がオペランドとして使用されています)
A1339E : RdLo と RdHi が同じレジスタの場合予測できません
A1341E : 境界整列されていないデスティネーションへの分岐です。<max> バイト境界で整列されているデスティネーションが期待されます。
A1342W : シンボルがこの命令と離れすぎている場合、別の AREA 内のシンボルの <name> はリンク時のエラーの原因になります
A1344I : ホストエラー:メモリが不足しています

A1355U : AREA 内にないラベルが見つかりました
この問題は、アセンブラディレクティブの前にホワイトスペースがない場合に発生します。ARM アセンブラディレクティブは、次のようにインデントされている必要があります。

```
    IF :DEF: FOO
    ; code
    ENDIF
```

以下は、正しくない例です。

```
IF :DEF: FOO
 ; code
ENDIF
```

最初の列で始まるシンボルは、ラベルであると見なされます。

A1356E : 命令がターゲットの CPU でサポートされていません
この問題は、選択されたアーキテクチャまたはプロセッサでサポートされていない命令を使おうとした場合に発生します。

以下に例を示します。

```
SMULBB r0,r0,r1 ;
```

これは、次のようにアセンブルできます。

```
armasm --cpu 5TE
```

以下を参照してください。

ARM アーキテクチャリファレンスマニュアル。

A1406E : 不正な 10 進数
A1407E : 大きすぎる浮動小数点値
A1408E : 大きすぎる(単精度)浮動小数点値
A1409W : 小さな(単精度)浮動小数点値は 0.0 に変換されます
A1411E : ベクタ指定子に閉じの '>' がありません
A1412E : ベクタ長が不正です。<min> と <max> の間にする必要があります。
A1413E : ベクタのストライドが不正です。<min> と <max> の間にする必要があります。
A1414E : ベクタはそれ自身を循環して上書きされます。長さ * ストライドが <max> 以下である必要があります。
A1415E : VFPASSERT の後に 'VECTOR' または 'SCALAR' を指定する必要があります
A1416E : ベクタ長が現在のベクタ長 <len> と一致しません
A1417E : ベクタのストライドが現在のベクタのストライドと一致しません
A1418E : レジスタの型 '<type>' が命令に対して不正です。浮動小数点/倍精度型のレジスタが期待されます。
A1419E : スカラオペランドがスカラバンク内にありません
A1420E : ベクタオペランドの長さが異なっています
A1421E : ベクタオペランドのストライドが異なっています
A1422E : このベクタオペランドとスカラオペランドの組み合わせは使用できません
A1423E : この操作はベクタ化できません
A1424E : この命令に対するオペランドではベクタ指定子は使用できません
A1425E : デスティネーションのベクタをスカラバンク内に置くことはできません
A1426E : ソースのベクタをスカラバンク内に置くことはできません
A1427E : オペランドが部分的にオーバラップしています
A1428E : レジスタリストに型の異なるレジスタが含まれています
A1429E : 期待されるレジスタリスト
この問題は、FRAME SAVE および FRAME RESTORE ディレクティブにレジスタリストが指定されていない場合に、アセンブラによって通知されます。
*『armasm ユーザガイド』*の以下のセクションを参照して下さい。

- *FRAME RESTORE*。
- *FRAME SAVE*。

A1430E : 不明な frame ディレクティブ
A1431E : frame ディレクティブは PROC/FUNCTION の外部では使用できません
*『armasm ユーザガイド』*の以下のセクションを参照して下さい。

Frame ディレクティブ。

A1432E : 浮動小数点レジスタ型は、選択した浮動小数点アーキテクチャと互換性がありません
A1433E : この命令のライトバック形式のみが存在します
命令で指定されたアドレッシングモードは、ライトバック指定子(つまり、ベースレジスタの後の '!')を含んでいませんでしたが、命令セットは命令のライトバック形式だけをサポートしています。ライトバック形式を使用するか、要求された動作をする命令に置き換えて下さい。
A1434E : アーキテクチャ属性 '<attr1>' と '<attr2>' が競合しています
A1435E : アーキテクチャ用にアセンブルする場合、{PCSTOREOFFSET} は定義されていません
{PCSTOREOFFSET} は、プロセッサ用にアセンブルする場合にだけ定義され、アーキテクチャ用の場合は定義されません。
A1437E : {ARCHITECTURE} が定義されていません
{ARCHITECTURE} は、アーキテクチャ用にアセンブルする場合にだけ定義され、プロセッサ用の場合は定義されません。
A1446E : 不正または不明な属性 '<attr>' です。代わりに --apcs /interwork を使用してください
以下に例を示します。

```
AREA test1, CODE, READONLY

AREA test, CODE, READONLY, INTERWORK
```

このコードは、元はレガシー ARM ソフトウェア開発ツールキット(STD)と共に動作することが想定されていたと考えられます。INTERWORK エリア属性は、現在では廃止されています。このエラーが表示されないようにするには、以下の処理を行います。

- AREA の行から ", INTERWORK" を削除します。
- 代わりに、 armasm --apcs /interwork foo.s でアセンブルします。

A1447W : ファイルの終わりに END ディレクティブがありませんが、END というラベルが見つかりました
この問題は、インデントされていない END ディレクティブによって発生します。
A1448E : 廃止予定の形式の PSR フィールド指定子が使用されています(_f を使用してください)
A1449E : 廃止予定の形式の PSR フィールド指定子が使用されています(_c を使用してください)

A1450E : 廃止予定の形式の PSR フィールド指定子が使用されています(将来の互換性のために _cxsf を使用してください)

armasm では、以下の形式で、MRS および MSR 命令のすべてがサポートされています。

```
 MRS(cond) Rd, CPSR   MRS(cond) Rd, SPSR   MSR(cond) CPSR_fields, Rm   MSR(cond)
SPSR_fields, Rm   MSR(cond) CPSR_fields, #immediate   MSR(cond) SPSR_fields,
#immediate
```

ここで、fields は、cxsf の任意の組み合わせです。

アセンブラの従来のバージョンでは、制御フィールドとフラグフィールドを変更するために、MSR 命令で次のような他の形式も使用できました。

- cpsr または cpsr_all、制御およびフラグフィールド
- cpsr_flg、フラグフィールドのみ
- cpsr_ctl、制御フィールドのみ

同様の制御およびフラグ設定が、SPSR にも適用されます。

これらの形式は、廃止予定になっているため、使用しないで下さい。これらの形式が従来のコードに含まれている場合、アセンブラによって以下のメッセージが通知されます。

```
廃止予定の形式の PSR フィールド指定子が使用されています(_cxsf を使用してください)
```

警告が表示されないようにするには、多くの場合、代わりに _c、_f、_cf、または _cxsf を使うようにコードを変更します。

『*armasm ユーザガイド*』の以下のセクションを参照して下さい。

- *A32 状態での条件付き実行*。
- *T32 状態での条件付き実行*。
- *AArch32 状態の汎用レジスタ*。
- *AArch32 状態でのインラインバレルシフタへのアクセス*。

以下の FAQ を参照して下さい。

armasm :use of MRS and MSR instructions ('Deprecated form of PSR field specifier') も参照して下さい。

A1454E : FRAME STATE RESTORE ディレクティブに対応する FRAME STATE REMEMBER がありません

『*armasm ユーザガイド*』の以下のセクションを参照して下さい。

- *Frame ディレクティブ*。
- *FRAME STATE REMEMBER*。
- *FRAME STATE RESTORE*。

A1456W : INTERWORK エリアディレクティブは現在サポートされていません。--apcs /inter が選択された場合と同様に続行されます。

例えば、次のコードはこの警告を生成します。

```
AREA test, CODE, READONLY, INTERWORK
```

このコードは、元はレガシー ARM ソフトウェア開発ツールキット(STD)と共に動作することが想定されていたと考えられます。INTERWORK エリア属性は、現在では廃止されています。この警告が表示されないようにするには、以下の処理を行います。

1. AREA の行から ", INTERWORK" を削除します。
2. 代わりに、 armasm --apcs /interwork foo.s でアセンブルします。

メッセージ番号 A1446E も参照して下さい。

A1457E : INTERWORK と NOINTERWORK コードエリアを同じファイル内に混在させることはできません。
INTERWORK と(既定の)NOINTERWORK コードエリアは、同じファイル内に混在させることができません。このコードは、元来、ARM ソフトウェア開発ツールキット(SDT)で使用することを目的としている可能性があります。INTERWORK AREA 属性は、ARM コンパイラツールチェーンでは廃止されています。

以下に例を示します。

```
AREA test1, CODE, READONLY    …  AREA test2, CODE, READONLY, INTERWORK
```

このエラーが表示されないようにするには、以下の手順を実行します。

1. 2 つの AREA を、例えば test1.s と test2.s のような個別のアセンブリファイルに移動します。
2. , INTERWORK を test2.s の AREA の行から削除します。
3. test1.s を armasm --apcs /nointerwork でアセンブルします。
4. test2.s を armasm --apcs /interwork でアセンブルします。
5. リンク時に、リンカによって必要なインターワーキングベニアが追加されます。

メッセージ番号 A1446E も参照して下さい。

A1458E : fpu が None の場合、DCFD または DCFDU は使用できません。

A1459E : レジスタに対する B または BL は実行できません。
命令のこの形式は使用できません。使用できる形式については、以下を参照して下さい。

ARM アーキテクチャリファレンスマニュアル。

A1461E : 指定されたプロセッサまたはアーキテクチャは Thumb 命令をサポートしていません
--cpu オプションを使用して特定のアーキテクチャまたは CPU を指定した後で、このエラーを生成した AREA 内に Thumb コードを組み込んだと考えられます。

例えば、次のコマンドラインでは、選択されたアーキテクチャ ARMv4 は、Thumb コードをサポートしていません。

```
armasm --cpu 4 code.s
```

A1462E : 指定されたメモリ属性はこの命令をサポートしていません

A1463E : SPACE ディレクティブは大きすぎるためこのエリアに収まりません。エリアのサイズの制限は 2^32 です。

A1464W : ENDP/ENDFUNC に対応する PROC/FUNC がありません

A1466W : 演算子の優先順位とは、C では式が異なる方法で評価されることを意味します
armasm は常に、特定の式を C とは異なる順序で評価してきました。この警告は、C プログラマがアセンブリ言語で記述する際の手助けになります。
警告が表示されないようにするには、以下のいずれかの手順を実行します。

- 括弧を追加することによって、評価の順序が明示的になるようにコードを変更します。
- --unsafe スイッチで、警告を非表示にします。

*『armasm ユーザガイド』*の以下のセクションを参照して下さい。

演算子の優先順位。

A1467W : 負のオフセット <offset> を使用した FRAME ADDRESS は推奨されません

A1468W : 標準構造フレームアドレスより上にレジスタを保存する FRAME SAVE は推奨されません

A1469E : FRAME STATE REMEMBER ディレクティブに対応する FRAME STATE RESTORE がありません
*『armasm ユーザガイド』*の以下のセクションを参照して下さい。

- *Frame ディレクティブ*。
- *FRAME STATE REMEMBER*。
- *FRAME STATE RESTORE*。

A1471W : ディレクティブ <directive> は実行可能な位置にある可能性があります
この問題は、例えば LTORG ディレクティブで発生する場合があります（メッセージ A1283E および A1284E を参照）。LTORG は、この位置でのリテラルプール DCD データをダンプするようにアセンブラに指示します。

この警告が発生しないようにするには、データをプロセッサが命令として実行できない場所に配置する必要があります。LTORG の位置として適切なのは、無条件分岐の直後、またはサブルーチンの最後にある復帰命令の後です。

最後の手段として、データが実行されるのを防ぐために、LTORG の部分に分岐を追加することもできます。以下に例を示します。

```
  B unique_label
  LTORG
unique_label
```

A1475E : 少なくとも 1 つのレジスタを転送する必要があります。転送しないと結果を予測できません。
A1476E : ワード境界で整列されていないアドレスでの BX r15 は予測できません
A1477E : このレジスタの組み合わせは予測不能な動作になります
このエラーは、実行時に予測不可能な結果が生じる命令をアセンブリしている場合に生成されます。この 予測不可能 な結果を回避するには、コードを記述し直す必要があります。例えば、次の命令は Thumb 命令のアセンブル時にこのエラーを必ず発生し、ターゲットアーキテクチャは ARMv6T2 以降です。

```
ADD sp, r0, #100 ; error - 予測不要な SP の使用 CMP pc, #1 ; error - 予測不要な PC の使用 PUSH    {r0, pc} ; error - 予測不要なレジスタの組み合わせの使用
```

A1479W : 要求されたアライメント <alignreq> がエリアアライメント <align> より大きいため、エリアアライメントを大きくしました
これは、ALIGN ディレクティブの整列境界が、それを含む AREA よりも粗い場合の警告です。これは使用できません。この問題を解消するために、AREA の側のアライメントがアセンブラによって自動的に増やされます。この警告を発生させる単純な例は、次のようなものです。

```
AREA test, CODE, ALIGN=3  ALIGN 16  mov pc, lr  END
```

この例では、AREA (ALIGN=3) のアライメントが 2^3=8 バイト境界ですが、mov pc,lr 命令は 16 バイト境界になっているため、エラーになります。

——————— 注 ———————
2 つのアライメントの種類が、異なる方法で指定されています。
———————————————

この警告は、AREA ... ALIGN=0 を使ってコードセクションをバイト境界に整列させるときにも発生します。そのような設定はできません。コードセクションは、次のようにだけ整列できます。

- A32 コードでは、4 バイト境界。つまり、"ALIGN=2" を使用します。
- T32 コードでは、2 バイト境界。つまり、"ALIGN=1" を使用します。

*『armasm ユーザガイド』*の以下のセクションを参照して下さい。

- *ALIGN*。
- *AREA*。

A1480W : ディレクティブまたは命令と同じ名前をマクロに付けることはできません
A1481E : オブジェクトファイル形式はこのエリアアライメントをサポートしていません
A1482E : シフトオプションが範囲外です。使用可能な値は <min> ～ <max> です。

A1484W : 現在サポートされていないシフト名 'ASL' です。代わりに LSL を使用してください。
ARM アーキテクチャには、ASL シフト演算がありません。ARM バレルシフタには、以下のシフトの種類だけがあります。

- ROR
- ASR
- LSR
- LSL

算術(符号付き)左シフトは、論理左シフトと同じです。符号ビットは、常にシフトアウトされるためです。

名前 ASL が使用されている場合、アセンブラはこの警告を発して ASL を LSL に変換します。

*『armasm ユーザガイド』*の以下のセクションを参照して下さい。

- *--unsafe*。
- *ASR*。

A1485E : LDM/STM 命令が --split_ldm で許可されている最大レジスタ数 <max> を超えています
A1486E : 別の AREA のシンボルの ADR/ADRL は ELF ではサポートされていません。
ADR および ADRL 疑似命令は、同じコードセクション内のラベルと共に使う場合のみ使用できます。レジスタにエリア外のアドレスをロードするには、代わりに LDR を使用して下さい。
A1487W : 現在サポートされていない命令名 'ASL' です。代わりに LSL を使用してください。
この警告は、--16 コマンドラインオプションを使用してアセンブルする場合、CODE16 ディレクティブを使用してアセンブルする場合、UAL 以前の Thumb コードで ASL 命令が使用された場合に生成されます。対応する ARM ASL メッセージ A1484W を参照して下さい。
A1488W : 行 <lineno> の PROC/FUNC に対応する ENDP/ENDFUNC がありません(ファイル '<filename>')
A1489E : <FPU> が定義されていません
A1490E : <CPU> が定義されていません
{CPU} は、プロセッサ用にアセンブルする場合にだけ定義され、アーキテクチャ用の場合は定義されません。
A1491W : 内部エラー:不正なアライメントを使用した再配置がオフセット <offset> で見つかりました
この問題は、アセンブラのエラーを示している場合があります。購入元にお問い合わせ下さい。
A1492E : イミディエート 0x<val> がこの操作の範囲外です。使用可能な値は 0x<min> ～ 0x<max> です
A1493E : REQUIRE は AREA 内にある必要があります
A1495W : 分岐のターゲットがデータアドレスです
armasm はシンボルの種類を判断し、データへの分岐を検出します。この警告が表示されないようにするには、--diag-suppress 1495 を指定します。
A1496W : シンボル '<symbol>'(オフセット <offset>)に関する ROPI アドレスの絶対再配置を実行すると、リンクエラーが発生する場合があります
例えば、--apcs /ropi で次のコードをアセンブルするときに、この警告が発生します。これは、PI コードシンボルへの絶対再配置(R_ARM_ABS32)が生成されるためです。

```
 AREA code, CODE
codeaddr DCD codeaddr
 END
```

A1497W : シンボル '<symbol>'(オフセット <offset>)に関する RWPI アドレスの絶対再配置を実行すると、リンクエラーが発生する場合があります
例えば、--apcs /rwpi で次のコードをアセンブルするときに、この警告が発生します。これは、PI データシンボルへの絶対再配置(R_ARM_ABS32)が生成されるためです。

```
 AREA data, DATA
dataaddr DCD dataaddr
 END
```

A1498E : Thumb 命令の後に予期しない文字があります
例えば、次の命令は UAL と UAL 以前のコードで有効です。

```
ADD r0, r0, r1
```

ただし、次の命令は UAL 以前の Thumb コードでは無効です。予期しない文字は、, ASR #1 です。

```
ADD r0, r0, r1, ASR #1
```

A1499E : レジスタの組み合わせは有効な連続ペアではありません
A1500E : '<eword>' が期待する場所に他の文字が指定されています
A1501E : シフトオプションが範囲外です。使用可能な値は 0、8、16、または 24 です。
A1502W : レジスタ <reg> は caller-save レジスタであり、この操作では有効ではありません
A1505E : 不正な式の型。論理式が期待されます。
A1506E : 累算器は accx の形式である必要があります。x の範囲は 0 ～ <max> です。
A1507E : レジスタリストの 2 番目のパラメータは最初のパラメータ以上である必要があります。
A1508E : 構造体が一致しません。条件が期待されます。
A1509E : 不正なシンボル型です。ラベルまたは weak 外部シンボルが期待されます。
A1510E : イミディエート 0x<imm> は、0 ～ 255 およびローテーションで表すことはできません
A1511E : イミディエートは 2 つのデータ処理命令の組み合わせで表すことはできません
A1512E : イミディエート 0x<val> がこの操作の範囲外です。使用可能な値は <min> ～ <max> です
A1513E : シンボルが見つからないか、シンボル型が '<name>' に対して互換性がありません
A1514E : 不正なグローバル名 '<name>'
A1515E : 不正なローカル名 '<name>'
A1516E : 不正なシンボル '<name>'。定義されていないか、外部です。
A1517E : <operator> と等しい予期しない演算子
A1539E : リンク順序の依存関係 '<name>' がエリアではありません
A1540E : self にリンク順序の依存関係を指定することはできません
A1541E : <code> は有効な条件コードではありません
A1542E : マクロ名 <name1> と <name2>[パラメータ] が矛盾しています
A1543W : マクロパラメータのデフォルト値が空です
A1544E : 空の PSR フィールド指定子は無効です。フィールドには c、x、s、f のいずれか 1 つを指定する必要があります。
A1545U : 1 つの <objfmt> ファイルにセクションが多すぎます
A1546W : スタックポインタの更新によって、8 バイトのスタックのアライメントが分断される可能性があります
スタックは外部境界上の 8 バイト境界で整列されていなければならないので、奇数のレジスタをプッシュすると、この警告が発生します。以下に例を示します。

```
PUSH {r0}
```

この警告はデフォルトでは非表示になっています。この警告が表示されるようにするには、--diag_warning 1546 を使用します。

*『armasm ユーザガイド』*の以下のセクションを参照して下さい。

--diag_warning=tag{, tag}。

A1547W : PRESERVE8 ディレクティブが自動的に設定されました
例:

```
PUSH {r0,r1}
```

この警告は、PRESERVE8 ディレクティブがユーザによって明示的に設定されず、アセンブラによって自動的に設定されたために発生します。この警告はデフォルトでは非表示になっています。これを有効にするには、--diag_warning 1547 を使用します。

*『armasm ユーザガイド』*の以下のセクションを参照して下さい。

- *--diag_warning=tag{, tag}*。
- *REQUIRE8 および PRESERVE8*。

A1548W : コードに SP のインデクス/オフセットされた LDRD/STRD が含まれていますが、REQUIRE8 が設定されていません

この警告は、REQUIRE8 ディレクティブが必要であるのに設定されなかった場合に発生します。以下に例を示します。

```
PRESERVE8 STRD r0,[sp,#8]
```

『armasm ユーザガイド』の以下のセクションを参照して下さい。

REQUIRE8 および PRESERVE8。

A1549W : PRESERVE8 を設定せずに REQUIRE8 を設定するのは異常です。

例:

```
PRESERVE8 {FALSE}
REQUIRE8
STRD r0,[sp,#8]
```

A1550U : 入力と出力のファイル名が同じです。

A1551E : Comdef エリア <name> を非 comdat グループに追加することはできません

A1560E : 定数以外のバイトリテラル値はサポートされていません

A1561E : MERGE セクションと STRING セクションはデータセクションである必要があります

A1562E : Merge セクションのエントリサイズは 0 より大きくする必要があります

A1563W : 命令によって CPU が <stalls> サイクルの間ストールします

--cpu オプションによって選択されたプロセッサのパイプラインが原因で、コード内で発生する可能性があるインターロックに関する情報を、アセンブラから取得できます。そのためには、armasm --diag_warning 1563 を使用してアセンブルします。

——— 注 ———

--cpu オプションで Cortex-A8 などのマルチイッシュープロセッサが指定されている場合、インターロック警告は正しくないことがあります。

———————————

警告 A1746W も参照して下さい。

A1572E : 演算子 SB_OFFSET_11_0 は、LDR/STR 命令でのみ使用できます

A1573E : 演算子 SB_OFFSET_19_12 は、データ処理命令でのみ使用できます

A1574E : "<str>" の 1 つ以上のフラグ文字が期待されます

A1575E : bit[0] が 1 に等しい BLX はアーキテクチャ上定義されていません

A1576E : 不正なコプロセッサレジスタ名シンボル

A1577E : 不正なコプロセッサ名シンボル

A1578E : 不正な浮動小数点レジスタ名シンボル '<sym>'

A1581W : <no_padbytes> バイトのパディングがアドレス <address> で追加されました
生成されるコードにパディングバイトが追加される場合、デフォルトではアセンブラによって警告が生成されます。この問題は、より大きな境界が必要なアドレスで命令またはディレクティブが使用されると、常に発生します。たとえば、いくつかの T32 命令の後で確実に A32 命令が 4 バイト境界で開始される必要がある場合や、DCB の後に DCD が続く場合です。

以下に例を示します。

```
 AREA Test, CODE, READONLY  THUMB ThumbCode  MOVS r0, #1  ADR r1, ARMProg  BX r1  ;
ALIGN ; &lt;&lt;&lt; 最初の警告を回避するためにコメントを解除  ARM ARMProg  ADD r0,r0,#1
BX LR  DCB 0xFF  DCD 0x1234  END
```

このコードでは、次の警告が生成されます。

```
A1581W:2 バイトのパディングが 0x6 で追加されました 8 00000008 ARM A1581W:3 バイトのパディングがアドレス 0x11 で追加されました 13 00000014 DCD 0x1234
```

この警告は、T32 専用コードで ADR を使用した場合にも発生します。ADR T32 疑似命令は、ワード境界で整列されたアドレスしかロードできませんが、T32 コード内のラベルはワード境界で整列されていない場合があります。ALIGN を使用して、T32 コード内のアドレスを 4 バイト境界調整で整列させることができます。

*『armasm ユーザガイド』*の以下のセクションを参照して下さい。

- *ADR (PC 相対)*。
- *ADR (レジスタ相対)*。
- *ALIGN*。
- *DCB*。
- *DCD および DCDU*。

A1582E : リンク順序エリア '<name>' が定義されていません
A1583E : グループシンボル '<name>' が定義されていません
A1584E : モード <mode> はこの命令では使用できません
A1585E : オペランド型(<typ1>)は演算子 <op> に対して不正です
A1586E : オペランド型(<typ1>、<typ2>)は演算子 <op> に対して不正です
A1587E : レジスタリスト内のレジスタ数 <count> が多すぎます。最大数は <max> です。
A1588E : Align は VLD 命令および VST 命令でのみ使用できます
A1589E : エレメントインデクスはすべてのレジスタで一定である必要があります
A1590E : サブスクリプトのエレメントとサブスクリプト以外のエレメントを組み合わせることはできません
A1593E : 不正なアライメント。転送サイズ UIMM * <dt> と一致している必要があります。
A1595E : 不正なアライメント。<st> * <dt> と一致しているか、<st> が 4 の場合、64 である必要があります
A1596E : dt st の組み合わせに対してアライメント <align> が不正です
A1597E : dt が 8 の場合、レジスタを 2 ずつインクリメントすることはできません
A1598E : 不正なレジスタリストの長さ
A1599E : サブスクリプトが範囲外です。0 と <max_index> の間にある必要があります。
A1600E : セクション型は、SHT_LOOS と SHT_HIUSER の範囲内にある必要があります。
A1601E : イミディエートを表現できません
A1603E : IT ブロック内のこの命令は、予測不能な結果になります
A1604W : デスティネーションへの Thumb 分岐で、<max> バイトに整列されていません
A1606E : シンボル属性 <attr1> は、属性 <attr2> とともに使用できません
A1607E : Thumb-2 ワイド分岐命令が使用されましたが、Thumb-1 ナロー分岐命令にフィットするオフセットです
A1608W : MOV pc,<rn> 命令が使用されましたが、BX <rn> の方が適しています

A1609W : MOV <rd>,pc 命令ではビット 0 に設定されないため、復帰アドレスは作成されません
この警告は、Thumb 状態での実行中に PC の現在値がレジスタにコピーされる場合に発生します。この方法で復帰アドレスを作成しようとすると、ビット[0] が設定されないため、失敗します。この命令に BX しようとすると、状態が ARM に切り替えられます。

復帰アドレスを作成するには、以下のコードを使用します。

```
MOV r0, pc
ADDS r0, #1
```

その後は、以下のようにしてこの警告を非表示にしても安全です。

```
--diag_suppress 1609
```

A1611E : この命令では、レジスタリストを 2 ずつインクリメントすることはできません
A1612E : <type> アドレシングは <instr> では使用できません
A1613E : この操作では無効なレジスタまたはレジスタの組み合わせ <registers> です。<expected> のうちの 1 つが必要です
A1614E : dt が 64 の場合、スカラアクセスは実行できません
A1615E : 全レーンへの単一の要素または構造体のストアが定義されていません
A1616E : 命令、オフセット、イミディエート、またはレジスタの組み合わせが、現在の命令セットでサポートされていません
このエラーは、オペランドの無効な組み合わせを使おうとした場合に発生することがあります。例えば、Thumb での次のようなコードです。

```
MOV r0, #1 ; /* 不可 */ MOVS r0, #1 ; /* Ok */
```

『armasm ユーザガイド』の以下のセクションを参照して下さい。

A32 命令と T32 命令。

A1617E : 指定した幅は、現在の命令セットでサポートされていません
A1618E : 指定した命令は、現在の命令セットでサポートされていません
A1619E : 指定した条件が、前の IT と整合性がありません
A1620E : ファイル '<filename>' への書き込み中にエラーが発生しました:<reason>
A1621E : Thumb コードから ARM コードへの CBZ または CBNZ です
A1622E : 負のレジスタオフセットは、現在の命令セットでサポートされていません
A1623E : オフセットは、現在の命令セットでサポートされていません
A1624W : Thumb コードから ARM コードへの分岐です
A1625W : ARM コードから Thumb コードへの分岐です
A1626W : Thumb コードから ARM コードへの BL です
A1627W : ARM コードから Thumb コードへの BL です
この問題は、このファイル内に ARM コードから Thumb コードへの（またはその逆への）分岐がある場合に発生します。通常は、Thumb コードを別のアセンブラファイルに移動することで解決できます。その後、リンク時に、リンカによって必要なインターワーキングベニアが追加されます。
A1630E : 指定されたプロセッサまたはアーキテクチャは ARM 命令をサポートしていません
Cortex-M3 および Cortex-M1 などの ARM M プロファイルプロセッサでは、Thumb 命令セットだけが実装されており、ARM 命令セットは実装されていません。アセンブリファイルにいくつかの ARM 固有の命令が含まれており、これらのいずれかのプロセッサ用にビルドされていると考えられます。
A1631E : ロード/ストアでは、左へ 1、2、または 3 しかシフトできません
A1632E : IT AL ブロックでは、ELSE を使用できません
A1633E : LDR rx,= pseudo 命令は、ロードワード形式でしか使用できません
A1634E : Thumb の LDRD/STRD では、レジスタオフセットアドレシングモードはありません
A1635E : CBZ/CBNZ は、条件付きにすることはできません
A1636E : Thumb では、フラグ設定 MLA はサポートされていません
A1637E : 行の読み出しでエラーが発生しました:<reason>
A1638E : Thumb 内のレジスタオフセットのロード/ストアでは、ライトバックは使用できません
A1639E : 条件付きの DCI は Thumb モードでのみ使用できます

A1640E : オフセットは 4 の倍数である必要があります
A1641E : Thumb では、強制ユーザモード LDM/STM は使用できません
A1642W : 再配置されたナロー分岐は推奨されません
A1643E : 単精度/倍精度値に対して命令を使用できるかどうか判断できません
A1644E : 単精度レジスタを FLDMX/LSTMX とともに使用することはできません
A1645W : <old> に <new> を代入しました

armasm は、命令を置換する場合に警告を発行するように設定できます。以下に例を示します。

- ADD *negative_number* は SUB *positive_number* と同じです。
- MOV *negative_number* は MVN *positive_number* と同じです。
- CMP *negative_number* は CMN *positive_number* と同じです。

T32 命令セットでは、予測不能な単一レジスタ LDM が LDR に変換されます。

この警告はデフォルトでは非表示になっていますが、--diag_warning 1645 を使って、表示されるようにすることができます。

--diag_warning 1645 を使用して、次のコードがアセンブルされた場合、

```
AREA foo, CODE  ADD r0, #-1  MOV r0, #-1  CMP r0, #-1
```

アセンブラによって次のように通知されます。

```
警告:A1645W:Substituted ADD with SUB 3 00000000 ADD r0, #-1 Warning:A1645W:
Substituted MOV with MVN 4 00000004 MOV r0, #-1 Warning:A1645W:Substituted CMP with
CMN 5 00000008 CMP r0, #-1
```

生成されたコードは、次のようになります。

```
foo 0x00000000:e2400001 ..@.SUB r0,r0,#1 0x00000004:e3e00000 ....MVN r0,#0
0x00000008:e3700001 ..p.CMN r0,#1
```

A1647E : 不正なレジスタ名シンボルです。整数レジスタが必要です

構文のこの位置には、整数(コア)レジスタを指定する必要があります。

A1648E : 不正なレジスタ名シンボルです。ワイヤレス MMX SIMD レジスタが必要です
A1649E : 不正なレジスタ名シンボルです。ワイヤレス MMX ステータス/制御または汎用レジスタが必要です
A1650E : 不正なレジスタ名シンボルです。任意のワイヤレス MMX レジスタが必要です
A1651E : デスティネーションレジスタが R15 でない場合の TANDC、TEXTRC、TORC 命令が定義されていません
A1652W : FLDMX/FSTMX 命令は ARMv6 で廃止される予定です。精度が不明な値を保存および復元するには、FLDMD/FSTMD 命令を使用して下さい。
A1653E : ステータス/制御レジスタを使ったシフト命令が定義されていません
A1654E : バイトまたはハーフワードのロード/ストア時には、外部シンボルにアクセスできません
A1655E : ハーフワード/ワード/ダブルワードが整列されていない場合、命令は予測不能です
A1656E : この命令を使用する場合、ターゲットは少なくともワード境界で整列している必要があります
A1657E : WLDRB/WLDRH = 定数を使用して、バイト/ハーフワードリテラルをロードできません
A1658W : <opt> のサポートは廃止される予定です

armasm に渡されたオプションは、廃止される予定になっています。

*『armasm ユーザガイド』*の以下のセクションを参照して下さい。

アセンブラコマンドラインオプション。

A1659E : ARM/Thumb と Thumb-2EE 間で、B/BL/BLX を実行できません
A1660E : このレジスタタイプではスカラインデクスを指定できません
A1661E : このレジスタではアライメントを指定できません
A1662E : このレジスタタイプではデータ型を指定できません
A1663E : このレジスタでは既にデータ型が指定されています
A1664E : データ型指定子を認識できません
A1665E : データ型のサイズは、8、16、32、64 のいずれかである必要があります
A1666E : 浮動小数点型のサイズは、32 または 64 である必要があります
A1667E : 多項式型のサイズは、8 または 16 である必要があります

A1668E : 命令に指定されているデータ型の数が多すぎます
A1669E : この命令ではデータ型指定子を使用できません
A1670E : 64 ビットダブルワード型のレジスタ式が必要です
A1671E : 128 ビットクワッドワード型のレジスタ式が必要です
A1672E : 64 ビットまたは 128 ビットのレジスタ式が必要です
A1673E : ソースの 2 つのデータ型は、型とサイズが同じである必要があります
A1674E : ソースの第 1 オペランドは、第 2 オペランドの 2 倍のサイズの整数型である必要があります
A1675E : デスティネーションのデータ型とサイズは、ソースと同じである必要があります
A1676E : デスティネーションは、ソースの 2 倍のサイズの整数型である必要があります
A1677E : デスティネーションは、ソースと同じ型で、サイズは 1/2 にする必要があります
A1678E : デスティネーションは、ソースと同じサイズの型なしである必要があります
A1679E : デスティネーションは、ソースと同じ型で、サイズは 2 倍にする必要があります
A1680E : デスティネーションは、符号なし型で、サイズはソースの符号付き型の 1/2 である必要があります
A1681E : デスティネーションは、符号なし型で、サイズはソースの符号付き型と同じである必要があります
A1682E : デスティネーションが符号なし/符号付き型でソースが浮動小数点型であるか、ディスティネーションが浮動小数点型でソースが符号なし/符号付き型である必要があります。サイズは両方とも 32 ビットにしてください
A1683E : データ型指定子が、この命令の有効なエンコーディングと一致していません
A1684E : ソースオペランドは、サイズが <min> ～ <max> の、符号付き型または符号なし型である必要があります
A1685E : ソースオペランドは、サイズが <min> ～ <max> の、符号付き型、符号なし型、または浮動小数点型である必要があります
A1686E : ソースオペランドは、サイズが <min> ～ <max> の、符号付き型または浮動小数点型である必要があります
A1687E : ソースオペランドは、サイズが <min> ～ <max> の、整数型または浮動小数点型である必要があります
A1688E : ソースオペランドは、サイズが <min> ～ <max> の型なしである必要があります
A1689E : ソースオペランドは、<n> ビットの浮動小数点型である必要があります
A1690E : ソースオペランドは、サイズが <min> ～ <max> の符号付き型である必要があります
A1691E : ソースオペランドは、サイズが <min> ～ <max> の、整数型、浮動小数点型、または多項式型である必要があります
A1692E : ソースオペランドは、サイズが <min> ～ <max> の、符号付き型、符号なし型、または多項式型である必要があります
A1693E : ソースオペランドは、サイズが <min> ～ <max> の、符号なし型または浮動小数点型である必要があります
A1694E : 現在の命令セットでは、命令を条件付きにすることはできません
条件付き命令は、指定された命令セットでは使用できません。例えば、命令 MOVEQ は、A32 コード、および IT 命令が利用可能な T32 コードアーキテクチャでは使用できます。
A1695E : この命令ではスカラインデクスを使用できません
A1696E : 32、64、または 128 ビットのレジスタ式が必要です
A1697E : 32 または 64 ビットの VFP レジスタ式が必要です
A1698E : 32 ビットの VFP レジスタ式が必要です
A1699E : これらのレジスタでは、64 ビットのデータ型は使用できません
A1700E : ソースオペランドは、サイズが <min> ～ <max> の整数型である必要があります
A1701E : ソースオペランドでは、16 ビットの多項式型は使用できません
A1702E : この命令では、レジスタ Dm はスカラにできません
A1704E : このデータ型では、レジスタ Dm は D0 ～ D<upper> の範囲内である必要があります
A1705W : アセンブラが Qm レジスタを D<rnum>[<idx>] に変換しました
A1706E : レジスタ Dm はスカラである必要があります
A1708E : この命令の第 3 オペランドは、定数式である必要があります
A1709E : ARM またはスカラレジスタ式が必要です
A1710E : 現在のレジスタと以前のレジスタの差は <diff> である必要があります
A1711E : この命令では、レジスタリスト内にスカラレジスタを含めることはできません
A1712E : LSB と WIDTH を組み合わせて使用すると、動作が不安定になります
A1713E : APSR のフィールド指定子が無効です。APSR_ の後に n、z、c、v、q、g のいずれかを指定してください
A1714E : APSR のフィールド指定子の組み合わせが無効です

A1715E : PSR がターゲットアーキテクチャで定義されていません
A1716E : VMOV 命令のデスティネーションは、ARM 整数レジスタ、32 ビット単精度レジスタ、64 ビットダブルワードレジスタ、または 64 ビットダブルワードスカラレジスタである必要があります
A1717E : ソースレジスタは、ARM 整数レジスタ、32 ビット単精度レジスタ、または 64 ビットダブルワードスカラレジスタである必要があります
A1718E : ソースレジスタは、ARM 整数レジスタまたはデスティネーションと同じレジスタである必要があります
A1719W : この PSR 名は今後のリリースで廃止される予定です
A1720E : ソースレジスタは、64 ビットダブルワードスカラレジスタである必要があります
A1721E : デスティネーションレジスタには、all-lanes 指定子を使用できません
A1722E : IT ブロック内ではラベルは使用できません
A1723W : __RELOC は廃止される予定です。新しい RELOC ディレクティブを使用してください
A1724E : RELOC は、命令やデータ生成ディレクティブの直後にのみ使用できます
A1725W : コマンドラインでの 'armasm inputfile outputfile' という形式の使用は、今後サポートされなくなる予定です
A1726W : --max_cache を 8MB 未満にすることは、推奨されません
A1727W : 16 ビットの Thumb MOVS 命令の使用によりイミディエートが生成された可能性があります
A1728E : ソースレジスタは、デスティネーションレジスタと同じ型である必要があります
A1729E : レジスタリストには、32 ビット単精度レジスタまたは 64 ビットダブルワードレジスタのみを含めることができます
A1730E : これらの命令では、IA または DB アドレシングモードのみを使用できます
A1731E : クワッドワードレジスタでは、レジスタリストのインクリメントを 2 以上にすることはできません
A1732E : レジスタリストには、1 ～ 4 個の連続するダブルワードレジスタを含める必要があります
A1733E : レジスタリストには 2 個または 4 個のダブルワードレジスタを含める必要があります。4 個含める場合は、インクリメントを 2 にすることはできません
A1734E : レジスタリストには、インクリメント 1 または 2 の <n> 個のダブルワードレジスタを含める必要があります
A1735E : ポストインデクスオフセットは、ロード/ストアされたバイト数(<n>)と同じにして下さい
A1736E : リスト内のレジスタ数と要素数を同じにしてください
A1737E : PC や SP はオフセットレジスタとして使用できません
A1738E : イミディエート値の大きさが、この操作で対応している範囲を超えました
A1739W : 単一の VMOV 命令により定数が生成されました。2 番目の命令は NOP です
A1740E : FRAME PUSH ディレクティブや FRAME POP ディレクティブでは、0 未満のバイト数を指定できません
A1741E : 命令を条件付きにすることはできません
A1742E : LSL #Imm が必要です
A1744E : レジスタのアライメントは、16 ～ 256 の範囲内の 2 の累乗にする必要があります
A1745W : このレジスタの組み合わせは廃止される予定なので、アークテクチャの今後のリビジョンでは機能しない可能性があります

この警告は、以下の条件がすべて満たされた場合に生成されます。

- 廃止予定のレジスタの組み合わせを使用している場合(例:

```
PUSH    {r0, pc}
```

)。
- 32 ビット T32 命令をサポートするターゲットアーキテクチャ、つまり ARMv6T2 以降のターゲットアーキテクチャに対してアセンブルしている場合。
- A32 コードにアセンブルしている場合。

注

- A32 コードではなく T32 にアセンブルしており、ターゲットアーキテクチャが ARMv6T2 以降のときには、代わりにエラー A1477E が生成されます。
- 32 ビット T32 命令をサポートしていないアーキテクチャまたはプロセッサに対して、つまり、ARMv6T2 以前のすべての ARM® アーキテクチャに対してアセンブルを行うときには、デフォルトで、診断は表示されません。

A1746W : この CPU では、命令ストールが正しく診断されない可能性があります
この警告は、アセンブラが正確にモデル化していないプロセッサでメッセージ A1563W を有効にしていると示されます。この警告は、ユーザがコードを改善する際に A1563W の出力に依存できないことを示しています。

警告 A1563W も参照して下さい。

A1753E : メモリバリアのオプションを認識できません
A1754E : スカラレジスタの型を変更できません
A1755E : このレジスタには、既にスカラインデクスが指定されています
A1756E : すべてのレジスタのデータ型を指定する必要があります
A1757W : シンボル属性は角括弧内になければなりません。その他の構文は廃止されます
A1758W : このディレクティブを使用した複数のシンボルのエクスポートは廃止されます
A1759E : 指定されたプロセッサまたはアーキテクチャは Thumb-2EE 命令をサポートしていません
A1760W : ビルド属性 <from> は '<attr>' です
A1761W : <from> の '<diff>' とのビルド属性の差
A1762E : 分岐のオフセット 0x<val> が 16 ビット Thumb 分岐の範囲外ですが、32 ビット Thumb 分岐でエンコードできます。
この問題は、分岐命令へのオフセットが 16 ビットの分岐に収まらない大きさである場合に、T32 用にアセンブルすると発生します。アセンブラに 32 ビット分岐を生成するように指示するために、命令に .W 接尾文字を追加することができます。
A1763W : この命令に IT ブロックが挿入されました
このことは、T32 コードの多くの条件付き命令を使用できるように、アセンブラによって IT ブロックが挿入されたことを示しています。例えば、

```
MOVEQ r0,r1
```

この警告はデフォルトではオフになっています。有効にするには、--diag_warning A1763 を使用します。

A1764W : <name> 命令はアーキテクチャ <arch> 以上では廃止されます
A1765E : ALIGN のパディング値のサイズは 1、2、または 4 バイトである必要があります
この問題は、任意に指定できる padsize 属性が ALIGN ディレクティブで使用され、そのサイズが不適切であった場合に発生します。整列先のパラメータは参照されません。パラメータには、2^0 から 2^31 の範囲で、2 の累乗値を指定できます。
A1766W : コードのパディング値のサイズは最低でも <size> バイト必要です。データとして処理します
A1767E : 属性の後に予期しない文字があります
A1768E : '=' がありません
A1769E : 不正な NEON または VFP システムレジスタ名シンボル
A1771E : <exp> ビットのビットパターンが期待される場所の不正な浮動小数点ビットパターン
A1772E : デスティネーションは符号付きまたは符号なしの整数型、ソースは 32 ビットまたは 64 ビットの浮動小数点型である必要があります
A1773E : 浮動小数点変換は 32 ビット単精度型と 64 ビット倍精度型の間でのみ可能です
A1774E : 固定小数点変換は 16 ビットまたは 32 ビットの符号付き型または符号なし型でのみ可能です
A1775E : これらの型の間で変換することはできません
A1776E : この演算は 32 ビット単精度浮動小数点型では使用できません
A1777E : <n> がシンボル型の範囲外です。値は <min> と <max> の間にある必要があります
A1778E : <n> がシンボルのバインディングの範囲外です。値は <min> と <max> の間にある必要があります
A1779E : DCDO は READONLY シンボル '<key>' には使用できません
A1780E : 不明な ATTR ディレクティブ
A1781E : タグ #<id> は ATTR を使用して設定することはできません
A1782E : タグ #<id> は ATTR <cmd> を使用して設定する必要があります
A1783E : 属性スコープはラベルかセクションの名前である必要があります
A1784W : weak 定義 '<sym>' への参照は再配置されません
A1785E : マクロ '<macuse>' が見つかりませんが、'<macdef>' は存在しています

A1786W : SP を使用するこの命令は廃止される予定なので、アークテクチャの今後のリビジョンでは機能しない可能性があります

この警告は、以下の条件がすべて満たされた場合に生成されます。

- 廃止予定の方法で SP を明示的に使用している場合。以下に例を示します。

```
ADD sp, r0, #100
```

- 32 ビット T32 命令をサポートするターゲットアーキテクチャ、つまり ARMv6T2 以降のターゲットアーキテクチャに対してアセンブルしている場合。
- A32 コードにアセンブルしている場合。

ARM では、対応する T32 命令で使用できない A32 命令での SP の明示的な使用を廃止する予定です。下位互換性のためにこのような廃止予定のレジスタを A32 命令でまだ使用できます。また、アセンブラのコマンドラインオプション --diag_suppress=1786 を使用してこの警告を非表示にできます。ただし、アーキテクチャの今後のリビジョンでは機能しない可能性があるので、ARM ではコードを変更することをお勧めします。

サンプルに示されている廃止予定の SP の使用を以下のようなシーケンスに置き換えられます。

```
ADD r1, r0, #100
MOV sp, r1
```

注

- A32 コードではなく T32 にアセンブルしており、ターゲットアーキテクチャが ARMv6T2 以降のときには、代わりにエラー A1477E が生成されます。
- 32 ビット T32 命令をサポートしていないアーキテクチャまたはプロセッサに対して、つまり、ARMv6T2 以前のすべての ARM アーキテクチャに対してアセンブルを行うときには、デフォルトで、診断は表示されません。

A1787W : VFP ベクタモードの使用は ARMv7 で廃止される予定です

A1788W : この命令での PC の明示的使用は廃止される予定なので、アークテクチャの今後のリビジョンでは機能しない可能性があります

この警告は、以下の条件がすべて満たされた場合に生成されます。

- 廃止予定の方法で PC を明示的に使用している場合。以下に例を示します。

```
CMP pc, #1
```

- 32 ビット T32 命令をサポートするターゲットアーキテクチャ、つまり ARMv6T2 以降のターゲットアーキテクチャに対してアセンブルしている場合。
- A32 コードにアセンブルしている場合。

ARM では、対応する T32 命令で使用できない A32 命令での SP の明示的な使用を廃止する予定です。下位互換性のためにこのような廃止予定のレジスタを A32 命令でまだ使用できます。また、アセンブラのコマンドラインオプション --diag_suppress=1786 を使用してこの警告を非表示にできます。ただし、アーキテクチャの今後のリビジョンでは機能しない可能性があるので、ARM ではコードを変更することをお勧めします。

注

- A32 コードではなく T32 にアセンブルしており、ターゲットアーキテクチャが ARMv6T2 以降のときには、代わりにエラー A1477E が生成されます。
- 32 ビット T32 命令をサポートしていないアーキテクチャまたはプロセッサに対して、つまり、ARMv6T2 以前のすべての ARM アーキテクチャに対してアセンブルを行うときには、デフォルトで、診断は表示されません。

A1789W : この命令での PC の明示的使用は廃止される予定なので、アークテクチャの今後のリビジョンでは機能しない可能性があります。ただし、デスティネーションレジスタでは例外です。

A1790W : ベースレジスタをロードする Thumb LDM でライトバックは無視されます
この問題は、ベースレジスタのライトバックを示す感嘆符を誤って追加したことによって発生します。

以下に例を示します。

```
LDM r0!, {r0-r4}
```

これは、r0 がベースレジスタであり、デスティネーションレジスタリストにも含まれているため、正しい命令ではありません。この場合、アセンブラはライトバックを無視して、次のコードを生成します。

```
LDM r0, {r0-r4}
```

A1791W : タグ #<id> の以前の値はオーバーライドされます
A1792E : 未定義のビルド属性のタグ
A1793E : 変換は 16 ビット浮動小数点と 32 ビット浮動小数点の間でのみ可能です
A1794E : 変換演算には 2 つのデータ型が必要です
A1795E : ソースおよびデスティネーションベクタには <n> 個の要素を含める必要があります
A1796E : レジスタ型はデータ型と互換性がありません
A1797E : 指定された FPU は CPU アーキテクチャと互換性がありません
A1798W : 出力は WYSIWYG(<output>)ではありません
A1799W : 出力の WYSIWYG プロパティはチェックされていません
A1800W : 行に対する出力はありません
A1801E : 命令は現在の命令セットでは予測不能です
A1803E : 不正なシステム命令名
A1804E : 命令に対する不正な CP14 または CP15 レジスタ名
A1805E : レジスタは読み出し専用です
A1806E : レジスタは書き込み専用です
A1807W : 命令はターゲットの CPU で NOP として実行されます
A1808E : 生成されたオブジェクトファイルは破損している可能性があります(<reason>)
A1809W : 命令によって PC は使用前に整列されます。セクションは少なくとも 4 バイト境界で整列されている必要があります

この警告は、以下のすべての条件が満たされた場合に適用されます。これらの条件がすべて満たされた場合、かつ、この命令を含んでいるコードセクションがリンク時に 4 バイト整列のアドレスに配置されなかった場合、命令は実行時に誤ったアドレスで演算される可能性があります。これは、命令によって PC が使用前に 4 バイト整列したアドレスに整列されることが原因です。

- PC 相対オフセットをワード境界で整列された PC を必要とする T32 命令で使用している。
- この命令を含んでいるコードセクションに 4 バイト以下の境界整列が使用されている。
- 命令がリンク時に再配置されなかった(アセンブラが生成する再配置が原因)。

以下の例では、セクション内の整列が 2 バイトで行われているためこの警告によって診断された T32 内の LDR 命令を示します。

```
AREA ||.text||, CODE, READONLY, ALIGN=1
THUMB
LDR r0, [pc, #8]  ; gives warning A1809W
```

A1810E : ベースレジスタのライトバック値が不明確です。'[rn,#n]!' または '[rn],#n' 構文を使用してください
A1811E : フィル値のサイズは 1、2、または 4 バイトであると同時に、フィルサイズの因数である必要があります。
A1812W : ARM 命令と Thumb 命令のそれぞれ逆の命令セットでは命令をアセンブルできません
A1813W : 16 ビットの命令が使用された可能性のある箇所で 32 ビットの命令が使用されました
A1814E : 出力ファイルがありません
A1815E : SHT_ARM_EXIDX セクションには、リンク順序の依存関係が設定されている必要があります
A1816E : CODE16 では不明なオペコード '<name>' ですが、THUMB には存在します。
A1817W : ATTR タグ #<id> 設定は <scope> で無視されます
A1818W : ATTR COMPAT フラグ <flag> とベンダ '<vendor>' の設定は、<scope> で無視されます

A1819W : タグ #<id> と互換性のある ATTR 設定は、<scope> で無視されます
A1820E : レジスタとプロセッサモードが命令に対して有効ではありません
A1821E : 定数式またはレジスタ式が必要です
A1822E : 32 ビット拡張レジスタリストが必要です
A1823E : 64 ビット拡張レジスタリストが必要です
A1824E : コアレジスタまたは 32、64、または 128 ビットの拡張レジスタが必要です
A1825E : 定数または 32 ビット拡張レジスタが必要です
A1826E : 定数または 64 ビット拡張レジスタが必要です
A1827E : 定数または 128 ビット拡張レジスタが必要です
A1828E : コアレジスタまたは 32 ビット拡張レジスタが必要です
A1829E : コアレジスタまたは 64 ビット拡張レジスタが必要です
A1830E : コアレジスタまたは 128 ビット拡張レジスタが必要です
A1831E : 定数、浮動小数点定数、コアレジスタ、または 64 ビット拡張レジスタが必要です
A1832E : 浮動小数点定数、コアレジスタ、または 32 ビット拡張レジスタが必要です
A1833E : 定数または '{option}' が必要です。オプションは 0 ～ 255 の定数です
A1834E : レジスタまたはアドレス式が必要です
A1835E : 命令に指定されているデータ型の数が少なすぎます
A1836E : デスティネーションの '<dt>' データ型が必要です
A1837E : 最初のソースの '<dt>' データ型が必要です
A1838E : '<word1>' または '<word2>' が必要な場所に他の文字が指定されています
A1839E : デスティネーションレジスタは、スカラである必要があります
A1840E : 最初のソースレジスタは、スカラである必要があります
A1841E : ベースレジスタで指定されたアライメントは、命令に対して有効ではありません
A1842E : pseudo 命令では構文は使用できません
A1843E : 命令にリテラルロードはサポートされていません
A1844E : リテラル型はサポートされていません
A1845E : 現在の命令セットではレジスタ型を使用できません
A1846E : CPSR または SPSR に無効なフィールドが指定されました: 少なくとも c、x、s、または f のうちいずれかが直後に続く必要があります
A1847E : 再配置を 2 つ以上必要とする式は使用できません

この問題は、A32 命令のアセンブリ時に、他の領域のデータにアクセスしようとすると、発生することがあります。例えば、

```
LDR r0, [pc, #label - . - 8]
```

またはこれと同等の

```
LDR r0, [pc, #label-{PC}-8]
```

で、label が別の AREA で定義されている場合です。

コードを、次のように、より単純で同等の構文を使うように変更して下さい。

```
LDR r0, label
```

このようにすれば、label が同じ領域にあっても、別の領域にあっても、正しく動作します。

A1848W : IT ブロックの状態の変化
A1849E : オペランドのスカラインデックスが、データ型の範囲外です
A1850E : 幅はどのデータ型修飾子よりも前に配置する必要があります
A1851E : 行の先頭が無効です - このディレクティブではローカルラベルはサポートされていません
A1852E : ターゲット FPU では VFP ベクタモードの使用はサポートされていません
A1853E : 命令またはターゲットの境界整列は、少なくとも <n> バイトで整列されている必要があります
A1854E : 不明なオペコード '<name>' です。ターゲット CPU が違う可能性があります
A1856E : シフトされたレジスタオペランドは使用できません
A1857E : 指定されたシフトは使用できません
A1858E : この命令のフラグ設定形式は使用できません
A1859E : この命令のフラグ保存形式は使用できません
A1860E : レジスタオペランドはこの命令の R0 ～ R7 である必要があります

A1861E : オプション '<opt>' は廃止されました。
A1862E : 浮動小数点型のサイズは、16 または 32 である必要があります
A1863E : 浮動小数点型のサイズは、32 である必要があります
A1864E : 浮動小数点型のサイズは、16、32、または 64 である必要があります
A1865W : 定数式の前に '#' がありません。
A1866E : ワイヤレス MMX コントロールレジスタは、ターゲット CPU で定義されていません
A1867E : イミディエート 0x<val> がこの操作の範囲外です。使用可能な値は 0x<min> ～ 0x<max> の <mult> の倍数です
A1868E : ビットフィールド LSB は範囲外です。使用可能な値は 0 ～ <max> です
A1869E : レジスタ <field> に PC は指定できません
A1870W : '<attrs>' を除き、領域 '<name>' に正しくない属性があります
A1871E : イミディエート 0x<imm> は、0 ～ 255 の範囲で 0 ～ 23 だけ左シフトして表現することも、全バイト、奇数バイト、および偶数バイトで重複することもできません

このエラーは armasm が指定されたイミディエートで命令をエンコードできない場合に発生します。

例えば、ARM コンパイラ 4.1 以上では、Thumb 状態の次の命令がこのエラーの原因となります。

```
ADDS    r0, r1, #-20
```

これを回避するには、代わりに同等の SUBS 命令を使用します。

```
SUBS    r0, r1, #20
```

A1872E : レジスタによるシフトは使用できません
A1873E : この命令の非ライトバック形式のみが存在します
A1874E : 指定されたレジスタリストをターゲット命令セットにロード、格納できません
A1875E : レジスタ Rn はこの命令の R0 ～ R7 である必要があります

指定されたレジスタが R0 ～ R7 の範囲内になるように変更します。

A1876W : '|' を :OR: 演算子の同義語として使用することは廃止される予定です。
A1877E : 現在の命令セットでは、<field> に対して指定されたレジスタを使用できません
A1878E : この操作で使用する場合、オフセットは <reqalign> バイトで整列している必要があります
A1879E : 指定されたアドレッシングモードは使用できません
A1880E : データ転送サイズは使用できません
A1881E : <mode> ロード/保存モードは使用できません
A1882E : デスティネーションと最初のソースのレジスタは、同じである必要があります
A1883E : デスティネーションと 2 番目のソースのレジスタは、同じである必要があります
A1884E : 指定された AIF ビットはターゲット CPU で使用できません
A1885E : 現在の命令セットでは、プロセッサモードを変更できません
A1886E : carry-in を使用した追加のオペランドのサイズが無効です: 16 または 32 である必要があります
A1887E : 指定されたソースデータ型は使用できません。次のいずれかである必要があります:<str>
A1888E : 指定されたデスティネーションデータ型は使用できません。次のいずれかである必要があります:
<str>
A1889E : 指定されたレジスタ型はターゲットアーキテクチャで使用できません
A1890E : 指定されたシフトを実行すると、予測不能な動作になります
A1891E : このレジスタの組み合わせでライトバックすると、予測不能な動作になります
A1892W : このレジスタの組み合わせでライトバックすることは廃止される予定なので、アークテクチャの今後のリビジョンでは機能しない可能性があります
A1893E : 指定されたフラグを実行すると、予測不能な動作になります
A1894E : 指定されたイミディエートを実行すると、予測不能な動作になります
A1895E : 指定された条件を実行すると、予測不能な動作になります
A1896E : この命令では指定されたアライメントはサポートされていません
A1897E : ビットフィールド幅は範囲外です。使用可能な値は 1 ～ <max> です
A1898E : ターゲットを再配置できません。この命令に適した再配置はありません
A1899E : 指定された演算子は次の命令でのみ使用できます:<instrs>
A1900W : 廃止されるシステム命令名
A1901E : 指定されたシステム命令は、ターゲットアーキテクチャではサポートされていません

A1902E : 指定された特殊レジスタは、ターゲットアーキテクチャではサポートされていません

A1903E : 最初のパスに行がありません。アセンブルできません

命令や指令が、アセンブラのパス 2 に表示されてパス 1 では表示されない場合に発生します。

以下の例は、行がパス 1 に表示されない場合を示しています。

```
    AREA x,CODE
    [ :DEF: foo
num EQU 42 ; Assembler does not see this line during pass 1 because
           ; foo is not defined at this point during pass 1
    ]
foo DCD num
    END
```

A1905U : '<filename>' のプリプロセッサ処理ができませんでした

A1906W : 未完了の IT ブロック

A1907W : このシンボルのテストは確認済みです。これは、第 2 パスのエラーの原因となる可能性があります。この診断はデフォルトでは非表示になっています。A1903E、A1908E、または A1909E エラーが発生する可能性がある状況を特定するには、これを有効にして下さい。

A1908E : ラベル '<name>' の値が一致していません: パス 1 では <val1>、パス 2 では <val2> でした

以下の例では、パス 1 では x の値が 0x0004+r9、パス 2 では x の値が 0x0000+r0 であるため、このエラーが生成されます。

```
    map 0, r0
    if :lnot: :def: sym
        map 0, r9
        field 4
    endif
x   field 4
sym LDR r0, x
```

A1909E : 第 2 パスに行がありません。アセンブルできません

命令や指令が、アセンブラのパス 1 に表示されてパス 2 では表示されない場合に発生します。

以下の例は、行がパス 2 に表示されない場合を示しています。

```
    AREA x,CODE
    [ :LNOT: :DEF: foo
    MOV r1, r2  ; Assembler does not see this line during pass 2 because
                ; foo is already defined
    ]
foo MOV r3, r4
    END
```

A1911E : イミディエート 0x<val> がこの操作の範囲外です。イミディエート値は 0 である必要があります。

A1912E : この命令では、フラグ設定の指定時にレジスタ <field> に PC を指定できません

A1913E : 指定されたオペランド型はこの位置では使用できません

A1914E : 式が必要です

A1915W : この命令またはディレクティブでは再配置は推奨されません

A1916E : 不明な組み込み変数 '<name>'

A1917E : ベクタレジスタ式が必要です

A1921E : 8 ビットのバイトレジスタ式が必要です

A1922E : 16 ビットハーフワード型のレジスタ式が必要です

A1923E : ベクタレジスタのリストが必要です

A1925E : このアーキテクチャではコプロセッサ番号を 14 または 15 にしてください

A1927E : コアレジスタ、64 ビット拡張レジスタ、またはベクタレジスタが必要です

A1931W : IT ブロック内のこの命令は廃止される予定です

A1932E : このアーキテクチャにはレジスタ型を使用できません

A1933E : オプション '<opt>' は、ターゲットアーキテクチャではサポートされていません

A1934E : <shift> によるシフトは使用できません。使用可能な値は <allowed> です

A1936E : リテラルプールが離れすぎています。LTORG を使用して <distance> 以内になるようにアセンブルして下さい

A1937E : 丸めモード Z のみをサポートする固定小数点への変換
A1938E : このアーキテクチャではコプロセッサ番号を 14 にしてください
A1939W : このニーモニックの使用は廃止される予定です
A1940E : 実行専用は <option> と互換性がありません
A1941E : NOP 命令を使用して <nopsize> の非倍数に整列させることができません
A1942E : データ宣言は、実行専用セクションでは使用できません
A1943E : INCBIN は実行専用セクションでは使用できません
A1944E : 実行専用セクションでは、リテラルプールエントリを生成できません
A1992E : 外部シンボルの MOVT は、対応する MOVW 命令に従う必要があります
A1993E : この演算子では <objfmt> でサポートされない再配置が必要です
A1994E : このディレクティブは、<objfmt> ではサポートされていません
A1995E : weak 定義は、<objfmt> ではサポートされていません
A1996E : TYPE は、IMPORT の WEAK の後以外では使用できません
A1997E : weak 外部シンボルの予想されるエイリアス
A1998E : Comdat 関連エリアには Comdat 関連セレクション型が必要です
A1999E : Comdat 関連エリアを別の Comdat 関連エリアにすることはできません
A9511E : 製品定義ファイルが見つかりませんでした

armasm で、必要な製品ライセンスのマッピング（.elmap）ファイルが見つかりません。
これは、以下が原因である可能性があります。

- .elmap ファイルが欠落している。例えば、インストールが破損している場合などです。
- ARM_PRODUCT_PATH 環境変数が間違って設定されているため、armasm が間違った場所で .elmap ファイルを探している。
- ARM_TOOL_VARIANT 環境変数が間違って設定されているため、armasm が存在しない .elmap ファイルを探している。

第 2 章
リンカのエラーおよび警告

リンカ（armlink）のエラーおよび警告メッセージについて説明します。

以下のセクションから構成されています。

2.1 armlink のエラーおよび警告メッセージの非表示

コマンドラインオプションを使用して、リンカで生成される一部の診断メッセージを非表示またはダウングレードできます。

リンカのすべての警告は、コンパイラの警告の場合と同様に、`--diag_suppress` を使って非表示にすることができます。以下に例を示します。

```
--diag_suppress 6306
```

一部のエラー（L6220E、L6238E、L6784E など）は、次のようにして警告に降格できます。

```
--diag_warning
```

2.2 armlink のエラーおよび警告メッセージのリスト

armlink によって生成されるエラーメッセージと警告メッセージのリスト。

L6000U : メモリが不足しています。
このエラーは、RVCT v4.0 以前によって報告されます。このエラーと対処方法を表示する理由の詳細については、エラー L6815U の説明を参照して下さい。

L6001U : ファイル <filename> から読み出すことができませんでした。

L6002U : ファイル <filename> を開けませんでした:<reason>
このメッセージは、リンカのコマンドラインで指定されたファイルをリンカが開けなかったことを示しています。ファイルへのアクセスに問題があるか、コマンドラインに誤りがあることを示している場合もあります。このメッセージの一般的な例として、以下のようなものがあります。

- L6002U:ファイル /armlib/{libname} を開けませんでした:そのようなファイルまたはディレクトリはありません

  --libpath でライブラリのパスを指定します。
- エラー:armlink:L6002:ファイル errors=ver.txt を開けませんでした

  errors=ver.txt の前に二重ダッシュ(--)がないことが原因です。オプションに接頭文字 -- または - を付けないと、リンカはオプションを入力ファイルとして扱い、指定されたすべてのファイルをロードすることができなくなるので、リンク手順が失敗します。

『*armlink ユーザガイド*』の以下のセクションを参照して下さい。

--libpath=pathlist。

『*スタート ガイド*』の以下のセクションを参照して下さい。

ツールチェーンの環境変数。

L6003U : ファイル <filename> に書き込むことができませんでした。
指定されたファイルを読み出すとき、開くとき、またはそのファイルに書き込むときに、ファイル I/O エラーが発生しました。

L6004U : <library> のライブラリメンバリスト <list> が不完全です。
この問題は、ライブラリオブジェクトのリスト内にホワイトスペースがある場合に発生することがあります。

以下の例は、致命的なエラー:L6004U:x.lib のメンバリストのライブラリメンバが足りません:

```
armlink x.lib(foo.o, bar.o)
```

以下のサンプルが成功します。

```
armlink x.lib(foo.o,bar.o)
```

他の一般的な原因として、ライブラリが破損していること、ライブラリの形式がサポートされていないものであることなども考えられます。

L6005U : <library> のメンバリストの最後に余分な文字があります。

L6006U : 実行領域 <regionname> の OVERALIGN 属性にオーバーアライメント値が指定されていません。
『*armlink ユーザガイド*』の以下のセクションを参照して下さい。

L6007U : ファイル <filename> の形式を認識できません。
リンカが認識できる形式は、オブジェクトファイルでは ELF、ライブラリファイルでは AR です。指定されたファイルは、破損しているか、リンカが認識できない形式のファイルです。

L6008U : メンバ <mem>(<lib>)の形式を認識できません。
リンカが認識できるライブラリメンバオブジェクトのファイル形式は、ELF です。指定されたライブラリメンバは、破損しているか、リンカが認識できない形式のファイルです。

L6009U : ファイル <filename>: エンディアンが一致しません。
指定されたファイルまたはオブジェクトのエンディアンが他の入力ファイルのエンディアンと一致しませんでした。リンカは 1 つのリンク手順内でビッグエンディアンとリトルエンディアンのいずれのオブジェクトの入力も処理できますが、両方が混在する入力は処理できません。

L6010U : ファイル <filename> に対して stderr を再オープンできませんでした:<reason>
指定されたファイルを読み出すとき、開くとき、またはそのファイルに書き込むときに、ファイル I/O エラーが発生しました。

L6011U : 無効な整数定数:<number>.
この問題は、不正な整数定数を指定すると発生します。整数は、接頭文字 &、0x、または 0X を付けて、16 進形式で入力できます。

L6015U : リンクする入力ファイルが見つかりませんでした。
リンカには、リンクするオブジェクトファイルを少なくとも 1 つは指定しなければなりません。

例えば、次のリンクを実行しようすると、

```
armlink lib.a -o foo.axf
```

リンカはこのエラーを報告します。

代わりに、次のようにする必要があります。

```
armlink foo_1.o foo_2.o lib.a -o foo.axf
```

L6016U : オブジェクト/ライブラリ <object> でシンボルテーブルが見つからないか破損しています。
この問題は、GNU ツールでビルドしたライブラリとリンクする場合に発生します。GNU ar は、互換性のない情報を生成する場合があるためです。

この問題を解決するには、ar を armar に置き換えて、同じコマンドライン引数を使用します。また、エラーから回復するために、armar -s を使用してシンボルテーブルを再ビルドすることもできます。

L6017U : ライブラリ <library> のシンボルテーブルに無効なエントリが含まれています。オフセット 0x<offset> にメンバがありません。
ライブラリが破損している可能性があります。再ビルドしてみて下さい。

L6018U : <filename> は有効な ELF ファイルではありません。

L6019U : <filename> は有効な 64 ビット ELF ファイルではありません。

L6020U : <filename> は有効な 32 ビット ELF ファイルではありません。

L6022U : オブジェクト <objname> に複数の <table> が含まれています。
オブジェクトファイルにエラーがあるか、破損しています。この問題は、コンパイラのエラーを示している場合があります。購入元にお問い合わせ下さい。

L6024U : ライブラリ <library> に無効なメンバ名が含まれています。
指定されたファイルが有効なライブラリファイルではありません。エラーが含まれているか、破損しています。再ビルドしてみて下さい。

L6025U : 非ライブラリファイル <library> からメンバを抽出することができません。
指定されたファイルが有効なライブラリファイルではありません。エラーが含まれているか、破損しています。再ビルドしてみて下さい。

L6026U : ELF ファイル <filename> にはリトルエンディアンまたはビッグエンディアンのエンコーディングが含まれていません
ELF ファイルが無効です。再ビルドしてみて下さい。

L6027U : 再配置 #<rel_class>:<rel_number>(<objname>(<secname>)内)は無効または不明な型です。
この問題は、コンパイラのエラーを示している場合があります。購入元にお問い合わせ下さい。

L6028U : 再配置 #<rel_class>:<rel_number>(<objname>(<secname>)内)に無効なオフセットが含まれています。
この問題は、コンパイラのエラーを示している場合があります。購入元にお問い合わせ下さい。

L6029U : 再配置 #<rel_class>:<rel_number>(<objname>(<secname>)内)に無効または不明なシンボルが関係しています。
再配置が以下のいずれかのシンボルに関係しています。

- 無効か、オブジェクトシンボルテーブルに存在しない
- 再配置で使用するのに適していない

この問題は、コンパイラのエラーを示している場合があります。購入元にお問い合わせ下さい。

L6030U : 領域 <regname> のオーバーアライメント <overalignment> は、4 以上で 2 の累乗である必要があります

『armlink ユーザガイド』の以下のセクションを参照して下さい。

- 実行領域の属性。
- 入力セクション記述の構文。
- 実行領域と入力セクションのオーバーアライメント。

L6031U : スキャッタ記述ファイル <filename> をオープンできませんでした:<reason>

指定されたファイルを開こうとしたときに、I/O エラーが発生しました。この問題は、無効なファイル名が原因である場合があります。

L6032U : <object> に無効な <text> <value> (最大 <max_value>)があります

L6033U : シンボル <symbolname>(<objname>)は、無効なセクションに対して相対的に定義されています。

L6034U : シンボル <symbolname>(<objname> 内)に無効な値が含まれています。

この問題の最も一般的な原因は、セクション相対シンボルの値がセクション境界を超えていることです。

L6035U : 再配置 #<rel_class>:<rel_number>(ZI セクション <objname>(<secname>)内)は無効または不明な型です。

ZI セクションは R_ARM_NONE 型以外の再配置を持つことはできません。

L6036U : ファイル <filename> を閉じることができませんでした:<reason>

指定されたファイルを閉じるときに、I/O エラーが発生しました。

L6037U : '<arg>' は、オプション '<option>' では有効ではありません。

この引数は、このオプションでは有効ではありません。この問題の原因は、スペルの間違い、または引数のサポートされていない省略形の使用である可能性があります。

L6038U : 更新された SYMDEFS を書き込むための一時ファイルを作成できませんでした。

SYMDEFS 出力の保存に必要な一時ファイルの作成中に、I/O エラーが発生しました。

L6039W : #<rel_class> からの再配置:<rel_number>(<objname>(<secname>)内)は <symname> に関係しています。R-Type の再配置の作成がスキップされます。<rel_type> 型には対応する R-type の再配置はありません。

--reloc は、対応する R-Type 再配置を持たない再配置を含むオブジェクトと共に使用されます。

L6041U : 内部エラーが発生しました(<clue>)。

購入元にお問い合わせ下さい。

L6042U : 再配置 #<rel_class>:<rel_number>(<objname>(<secname>)内)にマッピングシンボル(#<idx>、最後のマッピング シンボル = #<last>)が関係しています。

マッピングシンボルに関係している再配置は使用できません。この問題は、コンパイラのエラーを示している場合があります。購入元にお問い合わせ下さい。

L6043U : 再配置 #<rel_class>:<rel_number>(<objname>(<secname>)内)に、範囲外のシンボル(#<val>、範囲 = 1-<max>)が関係しています。

再配置は、範囲(1 - n)内のシンボルに関してのみ実行できます(n はシンボルの数)。

L6047U : このイメージのサイズ(<actual_size> バイト)がこのバージョンのリンカに許可されている最大サイズを超過しています

L6048U : リンカがリンク手順(<id>)を続行できません。このバージョンのリンカではこのイメージは作成されません。

L6049U : リンカがリンク手順(<id>)を続行できません。このバージョンのリンカでは指定された 1 つ以上のライブラリとリンクできません。

L6050U : このイメージのコードサイズ(<actual_size> バイト)が、このバージョンのリンカに許可されている最大サイズを超過しています。

L6058E : リンカスクリプト <script> を行 <lineno> で解析しているときに構文エラーが発生しました:<token>

リンク ld スクリプトには、行番号に構文エラーがあります。

L6064E : ELF 実行可能ファイル <filename> がコマンドラインで入力として指定されました
原因として、-c コンパイラオプションを指定せずに、コンパイラからオブジェクトファイルを出力として指定したことが考えられます。以下に例を示します。

```
armclang --target aarch64-arm-none-eabi file.c -o file.o

armlink file.o -o file.axf
```

L6065E : ロード領域 <name>(サイズ <size>)が 0x80000000 の書き込み可能な連続ブロックの最大サイズを超えています。
リンカが 2GB を超えるセグメントの書き込みを試みました。セグメントの最大サイズは 2GB です。

L6175E : EMPTY 領域 <regname> にセクションセレクタを含めることはできません

L6176E : EMPTY 属性のない領域 <regname> に対して負の max_size を使用することはできません。
EMPTY 属性のある領域だけが、負の max-size を持つことができます。

L6177E : +offset 形式のベースアドレスを使用する領域 <regname> に対して負の max_size を使用することはできません。
+offset 形式のベースアドレスを使用する領域は、負の max-size を持つことができません。

L6188E : 特別なセクション <sec1> が、<obj1> と <obj2> によって複数回定義されています。
特別なセクションは、"Veneer$$Code" のように、1 回だけ使用できるセクションです。

L6195E : '<attr1>' と '<attr2>' の両方を領域 <regname> に対して指定することはできません
『armlink ユーザガイド』の以下のセクションを参照して下さい。

- *ロード領域の属性*。
- *実行領域の属性*。
- *ロード領域と実行領域のアドレス属性*。
- *ロード領域のアドレス属性の継承規則*。
- *実行領域のアドレス属性の継承規則*。
- *RELOC アドレス属性の継承規則*。

L6200E : シンボル <symbolname> が <object1> と <object2> によって複数回定義されています。
この問題の一般的な例には、以下があります。

```
シンボル __stdout が、retarget.o と stdio.o によって複数回定義されています。
```

これは、__stdout の競合する 2 つの定義が retarget.o と stdio.o にあるということを意味しています。retarget.o 内の定義は、ユーザ自身が記述したものです。stdio.o 内の定義は、デフォルトの実装であり、誤ってリンクされた可能性があります。

stdio.o には、多くのシンボル定義と、ファイル関数(fopen、fclose、fflush など)の実装が含まれています。

stdio.o は、いくつかの未解決の参照を解決するためにリンクされています。

stdio.o がリンクされている理由を特定するには、--verbose リンクオプションスイッチを使う必要があります。以下に例を示します。

```
armlink [... your normal options...]--verbose --list err.txt
```

その後、err.txt を調べて、リンカが何のために何をどこからリンクしているかを詳しく確認します。

以下の作業が必要になる場合もあります。

- fopen、fclose、fflush などの呼び出しを削除する。
- 関数の _sys_xxxx ファミリを再実装する。

『ARM C および C++ ライブラリと浮動小数点サポートユーザガイド』の以下のセクションを参照して下さい。

C および C++ ライブラリでの入出力関数のカスタマイズ。

L6201E : オブジェクト <objname> に複数のエントリセクションが含まれています。
入力オブジェクトで、エントリポイントを複数指定しています。使用するエントリポイントを選択するには、--entry コマンドラインオプションを使用して下さい。

『armlink ユーザガイド』の以下のセクションを参照して下さい。

--entry=location。

L6202E : <objname>(<secname>)を非ルート領域 '<regionname>' に割り当てることはできません
ルート領域とは、その実行アドレスがロードアドレスと同じである領域のことです。そのため、この領域は、スキャッタロード初期化コードによる移動やコピーが必要ありません。
特定のセクションは、次のようなイメージのルート領域内になければなりません。

- __main.o。
- リンカによって生成されるテーブル(Region$$Table)
- ライブラリからのスキャッタロード(__scatter*.o)オブジェクト
- ライブラリからのデコンプレッサ(__dc*.o)オブジェクト

必要なセクションがルート領域に配置されていない場合は、リンカは次のようなメッセージを表示します。

```
anon$$obj.o(Region$$Table) cannot be assigned to a non-root region 'RAM'.
```

InRoot$$Sections を使用すると、ルート領域にすべての必要なセクションを含めることができます。

```
ROM_LOAD 0x0000 0x4000
{
  ROM_EXEC 0x0000 0x4000 ; root region
  {
    vectors.o (Vect, +FIRST) ; Vector table
    * (InRoot$$Sections) ; All library sections
                         ; that must be in a root region
                         ; for example, __main.o, __scatter*.o,
                         ; dc*.o and * Region$$Table
  }
  RAM 0x10000 0x8000
  {
    * (+RO, +RW, +ZI) ; all other sections
  }
}
```

L6203E : エントリポイント(<address>)が非ルート領域 <regionname> 内にあります。
イメージのエントリポイントは、イメージのルート領域内の有効な命令に対応している必要があります。

L6204E : エントリポイント(<address>)が命令を指していません。
--entry コマンドラインオプションで指定した、イメージのエントリポイントは、イメージのルート領域内の有効な命令に対応している必要があります。

『armlink ユーザガイド』の以下のセクションを参照して下さい。

--entry=location。

L6205E : エントリポイント(<address>)は、ARM 命令用にワード境界で整列している必要があります。
このメッセージは、--entry コマンドラインオプションで指定したイメージのエントリポイントが、ワード境界で整列されていないため、表示されます。例えば、--entry=0x8000 ではなく、--entry=0x8001 と指定した場合です。

『armlink ユーザガイド』の以下のセクションを参照して下さい。

--entry=location。

L6206E : エントリポイント(<address>)がイメージの外部にあります。
--entry コマンドラインオプションで指定したイメージのエントリポイントが、イメージの外部にあります。例えば、次のように、エントリアドレスとして 0x8000 ではなく 0x80000 を指定した場合です。

```
armlink --entry=0x80000 test.o -o test.axf
```

『armlink ユーザガイド』の以下のセクションを参照して下さい。

--entry=location。

L6208E : --entry コマンドの無効な引数:'<arg>'
『armlink ユーザガイド』の以下のセクションを参照して下さい。

--entry=location。

L6209E : --entry に対して指定された無効なオフセット定数(<arg>)
『armlink ユーザガイド』の以下のセクションを参照して下さい。

--entry=location。

L6210E : イメージに複数のエントリポイントを含めることはできません(<address1>,<address2>)
1 つまたは複数の入力オブジェクトで、イメージのエントリポイントを複数指定しています。使用するエントリポイントを選択するには、--entry コマンドラインオプションを使用して下さい。

『armlink ユーザガイド』の以下のセクションを参照して下さい。

--entry=location。

L6211E : セクションの選択があいまいです。オブジェクト <objname> に複数のセクションが含まれています。
この問題は、複数の AREA を含むアセンブラオブジェクトに対してリンカオプション --keep を使用している場合に発生します。リンカで、どの AREA を保持するのかを認識できる必要があります。
この問題を解決するには、次のように、複数の --keep オプションを使って、保持する各 AREA の名前を指定します。

```
--keep boot.o(vectors) --keep boot.o(resethandler) …
```

注

複数の AREA を持つアセンブラファイルを使用すると、他の問題も発生する場合があるので、使用しないことを推奨します。

L6213E : 複数の First セクション <object2>(<section2>)は使用できません。<object1>(<section1>)は既に存在します。
FIRST セクションは 1 つしか使用できません。
L6214E : 複数の Last セクション <object2>(<section2>)は使用できません。<object1>(<section1>)は既に存在します。
LAST セクションは 1 つしか使用できません。
L6215E : --First/--Last に対するシンボルの選択があいまいです。シンボル <symbol> に複数の定義があります。
『armlink ユーザガイド』の以下のセクションを参照して下さい。

L6216E : 連続しないセクション <secname> に対してベース/リミットシンボルを使用することはできません
コンパイラによって生成される例外処理インデクステーブルは、セクション名が .ARM.exidx になります。詳細については、*「Exception Handling ABI for the ARM Architecture」*を参照して下さい。

リンク時には、これらのテーブルは同じ実行領域に連続して配置されていなければなりません。スキャッタファイル内で特定のセレクタパターンを使ってこれらのセクションを明示的に連続しないように配置している場合は、このエラーメッセージが発生する可能性が高くなります。以下に例を示します。

```
LOAD_ROM 0x00000000 {   ER1 0x00000000   {     file1.o (+RO) ; C++ ソースから     *
(+RO)   }   ER2 0x01000000   {     file2.o (+RO) ; C++ ソースから   }   ER3 +0
{     * (+RW, +ZI)   } }
```

例外処理インデクステーブルが file1.o と file2.o の両方に含まれる場合、リンカはこれらを個別の領域に配置できないため、これによって次のエラーが生成される可能性があります。

```
エラー: L6216E:連続しないセクション .ARM.exidx に対してベース/リミットシンボルを使用することはできません
```

また、.init_array セクションは、同じ領域内に連続して配置されていないとベースおよびリミットシンボルにアクセスできない場合もあります。

正しいコードを以下に示します。

```
LOAD_ROM 0x00000000 {   ER1 0x00000000   {     file1.o (+RO)   ; C++ ソースから     *
(.ARM.exidx)  ; セクション .ARM.exidx は明示的に配置する必要があります。                    ; そうしないと、2 つの領域で共有され、                    ; リンカは配置先を                    ; 決定できなくなります。*(.init_array)  ; セクション .init_array は明示的に配置する必要があります。                    ; そうしないと、2 つの領域で共有され、                    ; リンカは配置先を                    ; 決定できなくなります。*
(+RO)   }   ER2 0x01000000   {     file2.o (+RO)   ; from a C++ source   }   ER3
+0   {     * (+RW, +ZI)   } }
```

この例では、ベースシンボルとリミットシンボルが 1 つの領域内の .init_array に含まれています。

*『ARM C および C++ ライブラリと浮動小数点サポートユーザガイド』*の以下のセクションを参照して下さい。

C++ の初期化、構築、および破棄。

L6217E : 再配置 #<rel_class>:<rel_number>(<objname>(<secname>)内)は <symbol> に関係しています。インポートされたシンボルへの R_ARM_SBREL32 再配置

L6218E : シンボル <symbol>(<objname> から参照)が定義されていません。
この一般的な例として、以下のようなものがあります。

- ユーザエラー。未定義または間違った定義のシンボルへの参照があります。
- `Undefined symbol __ARM_switch8 or __ARM_ll_<xxxx> functions`

  ヘルパ関数は、コンパイラによって自動的にオブジェクトファイル内に生成されます。

  ——————— 注 ———————

  しかし、ヘルパ関数が h_*xxx* ライブラリ(h は、これらが標準 C ライブラリコードではなく、コンパイラヘルパライブラリであることを示しています)内にあった従来のプロジェクトのオブジェクトをリンクしている場合は、未定義参照エラーが発生します。

  オブジェクトを再コンパイルするか、これらのライブラリをリンカが見つけられるようにして下さい。

  ——————————————

- C の関数またはエンティティを C++ の関数またはエンティティから参照しようとしています。この問題は C++ の名前のマングリングによって発生します。C 関数を `extern ""C""` とマークすることで対処できます。
- `シンボル thunk{v:0,-44} to Foo_i::~Foo_i()(Bar_i.o から参照)が定義されていません。`

  シンボル `thunk{v:0,-44} to Foo_i::~Foo_i()` は、標準の `Foo_i::~Foo_i()` 関数のラッパ関数です。
  `Foo_i` は、他のベースクラスからの派生クラスです。そのため、
  — そのベースクラスへのポインタによって参照される場合のために、ベースクラスの vtable を持っています。
  — ベースクラスの vtable は、thunk のためのエントリを持っています。
  — デストラクタ thunk は、実際の(派生クラスの)デストラクタが出力の場合は、出力になります。

  このため、エラーを回避するには、このデストラクタを定義する必要があります。
- `シンボル main が定義されていません (kernel.o から参照)。`

  このエラーによって、リンカは、アプリケーションに `main()` 関数が含まれていないことをレポートしています。

L6219E : <type> セクション <object1>(<section1>) 属性 {<attributes>} は、隣接するセクション <object2>(<section2>) と互換性がありません。
このエラーは、リンカによって使用されているデフォルトの順序付け規則(RO、次に RW、次に ZI)の違反があった場合に発生します。一般的には、例えばスキャッタファイルで `+FIRST` または `+LAST` を使って、RW が RO の前になるように強制しようとした場合に発生します。

L6220E : <type> 領域 <regionname> のサイズ (<size> バイト) が、制限値 (<limit> バイト) を超えています。
例:

```
Execution の領域 ROM_EXEC のサイズ(4208184 バイト)が制限(4194304 バイト)を超えています。
```

この問題は、領域がスキャッタファイルで最大長を指定されたものの(指定は任意)、領域内のコードおよびデータのサイズが指定された制限を超えている場合に発生します。このエラーは、`--diag_suppress 6220` で非表示にすることができます。

例えば、スキャッタファイルで `ALIGN` ディレクティブと共に `.ANY`*num* セレクタを使用して、リンカにパディングを挿入させる場合に、このエラーが発生する可能性があります。この問題を解消するには、`--any_contingency` オプションを使用します。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

- *.ANY モジュールセレクタによる未割り当てセクションの配置*。
- *--any_contingency*。
- *--diag_suppress=tag[,tag,...]*。

L6221E : <type1> の領域 <regionname1> (<addrtype1>、範囲 [<base1>,<limit1>)) が <type2> の領域 <regionname2> (<addrtype2>、範囲 [<base2>,<limit2>)) と重複しています。

このエラーは、スキャッタロード記述ファイルの情報およびリンカが生成するマップ情報で実行領域の重複がないと示された場合でも発生する可能性があります。

RVCT v4.0 以前では、次に示すように、リンカから多くの情報が得られずにメッセージの内容を理解するのは困難でした。

警告 L6221E:<type1> 領域 <regionname1> は <type1> 領域 <regionname2> と重複しています

test.s サンプルファイル:

```
    AREA area1, CODE
    BX lr

    AREA area2, READWRITE, NOINIT
    SPACE 10

    AREA area3, READWRITE
    DCD 10
    END
```

scatter.txt サンプルファイル:

```
LR1 0x8000
{
  ER1+0
  {
    *(+ro)
  }
  ER2+0
  {
    *(+zi)
  }
  ER3+0
  {
    *(+rw)
  }
}
```

次を使用して構築します。

```
armasm test.s
armlink -o test.axf --scatter scatter.txt test.o
```

次のメッセージを生成します。

警告:L6221E:実行領域 [0x00008004,0x00008010) を持つ実行領域 ER2 は、ロード領域 [0x00008004,0x00008008) を持つ実行領域 ER3 と重複しています。

実行領域のベースアドレスが別の領域のロードアドレスと重複していると、警告メッセージ L6221E が表示される場合があります。これは不正なスキャッタファイルが原因である可能性があります。イメージのメモリマップには、スキャッタロードファイルによって記述される、ロードビューと実行ビューがあります。非 ZI セクションは、一意のロードアドレスを持っている必要があり、多くの場合は一意の実行アドレスも持っている必要があります。RVCT v3.1 以降、リンカによって領域が ZI 実行領域に割り当てられなくなりました。つまりこの警告の原因として、相対ベースアドレスを持つロード領域 LR2 がロード領域 LR1 の ZI 実行領域の直後に配置されていることが考えられます。

重複部分に実際のコードはなく、内部にデータも存在しないため、この警告は無害である可能性があります。

RVCT v4.0 ビルド 821 以降では、次のリンカオプションを使って各領域のアドレスと、ロード領域と重複する領域を見つけることができます。

```
--load_addr_map_info --map --list=map.txt
```

以下のいずれかを実行できます。

- 解析後に、実行領域のアドレスにコピーされていないロード領域を実行領域が壊さないことが判断できた場合にのみ、警告を無視します。アプリケーションをデバッグし、正常に初期化して実行することも確認します。
- 実行領域のベースアドレスを調整します。
- FIXED スキャッタロード属性を使用して、ロード領域と実行領域に同一のベースアドレスを指定します。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

- *相対ベースアドレスのロード領域と ZI 実行領域を含んだスキャッタファイル*。
- *実行領域の属性*。
- *ルート実行領域と FIXED 属性*。

L6222E : 部分的にリンクされたオブジェクトに複数の ENTRY セクション(<e_oname>(<e_sname>)および <oname>(<sname>))を含めることはできません。

複数のオブジェクトが部分的にリンクされた 1 つのオブジェクトになっている場合、全オブジェクトのうちの 1 つのセクションだけがエントリポイントを持つことができます。

注

この場合は、リンカオプション --entry を使ってエントリポイントの 1 つを選択することはできません。

L6223E : <objname>(<secname>)のあいまいなセレクタが、実行領域 <region1> および <region2> で見つかりました。

この問題は、スキャッタファイルで <objname>(<secname>)を複数の実行領域に配置するように指定した場合に発生します。ワイルドカード(*)を使用している場合に誤って発生することもあります。この問題を解決するには、スキャッタファイルでの選択をより限定的に行います。

L6224E : <objname>(<secname>)を実行領域内に配置できませんでした。

この問題は、リンカが入力セクションをスキャッタファイルのいずれのセレクタとも一致させられなかった場合に発生します。スキャッタファイルを修正して、適切なセレクタを追加する必要があります。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

リンカによるセクションの配置。

L6225E : 数値 <str...> が長すぎます。

L6226E : 領域 <regname> のベースアドレスがありません。

L6227E : --split を使用せずに -reloc と --rw-base を併用することはできません。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

- *--reloc*。
- *--rw_base=address*。
- *--split*。

L6228E : '<str1>' が予期されましたが、'<str2>' が見つかりました。

L6229E : スキャッタ記述 <file> が空です。

L6230E : 複数の実行領域(<region1>、<region2>)で <secname> を選択することはできません。

L6231E : モジュールセレクタがありません。

L6232E : セクションセレクタがありません。

L6233E : 不明なセクションセレクタ '+<selector>' です。

L6234E : <ss> は単一のセレクタの後に指定する必要があります。

例えば、スキャッタファイルで次のような記述があるとします。

```
:
* (+FIRST, +RO)
:
```

+FIRST は、この(単一の)セクションを先頭に配置することを意味します。複数のセクションと一致する可能性があるセレクタ(例えば +RO や +ENTRY)は、+FIRST(または +LAST)と共に使用することはできません。一緒に使用すると、エラーメッセージが生成されます。

L6235E : 複数のセクションがセレクタと一致します。すべてのセクションを FIRST/LAST にすることはできません。

『armlink ユーザガイド』の以下のセクションを参照して下さい。

- *FIRST 属性と LAST 属性を使用したセクションの配置*。
- *入力セクション記述の構文*。

L6236E : セレクタと一致するセクションがありません。FIRST/LAST になるセクションがありません。
スキャッタファイルでは +FIRST または +LAST になるセクションが指定されていますが、そのセクションが存在しないか、リンカがそのセクションを使われないと判断して削除しました。どのオブジェクトがプロジェクトから削除されたかを表示するには、リンカオプション --info unused を使用して下さい。例:

```
ROM_LOAD 0x00000000 0x4000 {  ROM_EXEC 0x00000000  {    vectors.o (Vect, +First)
&lt;&lt; ここでエラー     * (+RO)  }  RAM_EXEC 0x40000000  {    * (+RW, +ZI)  } }
```

次のような対処方法があります。

- リンカコマンドラインで vectors.o が指定されていることを確認します。
- --keep vectors.o を使ってリンクし、これをリンカが削除しないようにします。--no_remove を使ってこの最適化全体をオフにすることは ARM ではお勧めしていません。
- ARM では、ENTRY ディレクティブを vectors.s に追加して、ここがアプリケーションのエントリポイントであることをリンカに示すことを勧めています。例えば、

  ```
  AREA Vect, CODE ENTRY ; これをエントリポイントとして定義 Vector_table ...
  ```

  次に、--entry Vector_table を使ってリンクし、コードの実際の開始位置を定義します。

『armlink ユーザガイド』の以下のセクションを参照して下さい。

- *FIRST 属性と LAST 属性を使用したセクションの配置*。
- *--entry=location*。
- *--info=topic[,topic,...]*。
- *--keep=section_id*。
- *--remove*、*--no_remove*。
- *入力セクション記述の構文*。

『armasm ユーザガイド』の以下のセクションを参照して下さい。

ENTRY。

L6237E : <objname>(<secname>)に整列されていないデータへの再配置が含まれています。

L6238E : <objname>(<secname>)に '<attr1>' 関数から '<attr2>' 関数 <sym> への無効な呼び出しが含まれています。

このリンカエラーは、オブジェクトコード内でスタックのアライメントの競合が検出された場合に発生します。『ARM アーキテクチャ用 ABI』では、コードがインタフェースで 8 バイト境界のスタック整列を保持することを推奨しています。これによって、8 バイト境界で整列されている double および long long データ型にアクセスする LDRD および STRD 命令の効率的な使用が可能になります。

~PRES8 や REQ8 のようなシンボルは、以下のように、オブジェクトのビルド属性です。

- PRES8 は、オブジェクトがスタックの 8 バイト境界の整列を保持することを意味します。
- ~PRES8 は、オブジェクトがスタックの 8 バイト境界の整列を保持しないことを意味します(~ は NOT の意味)。
- REQ8 は、オブジェクトがスタックの 8 バイト境界の整列を要求することを意味します。

このリンクエラーは、通常、以下の 2 つの場合に発生します。

- アセンブラコード(8 バイト境界のスタック整列を保持しない)が、コンパイルされた C/C++ コード(8 バイト境界のスタック整列を必要とする)を呼び出す。
- 旧式のツールでコンパイルされた従来のオブジェクトと、最近のツールでコンパイルされたオブジェクトをリンクしようとする。これらの属性を持っていない従来のオブジェクトは、実際には 8 バイト境界の整列が保持できたとしても、~PRES8 として処理されます。

以下に例を示します。

```
エラー:L6238E:foo.o(.text)に '~PRES8' 関数から 'REQ8' 関数 foobar への無効な呼び出しが含まれています。
```

これは、オブジェクト foo.o に(.text という名前のセクション内に)、8 バイト境界のスタック整列を保持しない関数があり、それが 8 バイト境界のスタック整列を必要とする関数 foobar を呼び出そうとしていることを意味しています。

同様の警告に、次のようなものがあります。

```
警告:L6306W:'~PRES8' セクション foo.o(.text)では、'REQ8' 関数 foobar のアドレスを使用できません
```

この場合、外部シンボルのアドレスが参照されています。

この問題の解決策としては以下の 2 つの方法が考えられます。

- すべてのオブジェクトとライブラリを再ビルドします。

  アセンブラファイルがある場合は、すべての命令が 8 バイト境界のスタック整列を保持することを確認し、必要に応じて修正する必要があります。

  例えば、次のようなコードは、

  ```
  STMFD sp!, {r0-r3, lr} ; 奇数のレジスタをプッシュ
  ```

  次のように変更します。

  ```
  STMFD sp!, {r0-r3, r12, lr} ; 偶数のレジスタをプッシュ
  ```

  すべての命令が 8 バイト境界のスタック整列を保持する場合、アセンブラが自動的にオブジェクトを PRES8 属性でマークするので、各アセンブラファイルの先頭に PRESERVE8 ディレクティブを追加する必要がなくなります。
- 従来のオブジェクトまたはライブラリが、ソースコードがなかったり、資格や認証の問題などのために再ビルドできない場合は、そのオブジェクトを調べて、8 バイト境界の整列が保持されるかどうかを確認する必要があります。

  オブジェクトコードを逆アセンブルするには、fromelf -c を使って下さい。ADS 1.1 以降でコンパイルされた C/C++ コードは、通常、8 バイト境界の整列を保持しますが、アセンブルされたコードは保持しません。

オブジェクトが確実に 8 バイト境界の整列を保持している場合は、リンカコマンドラインで `--diag_suppress 6238` を使って、リンカエラー L6238E を非表示にすることができます。

このエラーメッセージを利用すると、オブジェクトが `PRES8` であることを確実にチェックできます。

リンカ警告 L6306W は、`--diag_suppress 6306` で非表示にできます。

以下の FAQ を参照して下さい。

8 バイト境界のスタック整列

`L6239E : 非インターワーキング <t2> シンボル '<sym>'(<obj2> 内)を、<t1> コード(<obj1>(<sec1>)内)から呼び出すことはできません。`

例:

```
Cannot call non-interworking ARM symbol 'ArmFunc' in object foo.o from THUMB code in
bar.o(.text)
```

この問題は、オプション `--apcs /interwork` を使ってコンパイルされなかった `foo.c` によって発生した可能性があります。このオプションは、リンカによって生成されるインターワークベニアを使って A32 コードから T32 コード(または T32 から A32)を呼び出すことを可能にします。

`L6241E : <objname>(<secname>)は、'<attr1>' 関数 <sym> のアドレスを使用できません。これは、イメージに '<attr2>' 関数が含まれているためです。`

`'--strict'` を使ってリンクした場合は、失敗してエラーになる条件が、次のようにリンカによって報告されます。

```
エラー:L6241E:foo.o(.text)は、'~IW' 関数 main のアドレスを使用できません。これは、イメージに
'IW' 関数が含まれているためです。
```

`IW` はインターワークを意味し、`~IW` は非インターワークを意味しています。

L6242E : オブジェクトの属性がイメージの属性と互換性がないため、オブジェクト <objname> をリンクできません。

コンパイルツールによって生成された各オブジェクトファイルには、構築する際に使用されたオプションを示す属性のセットが含まれています。リンカは、処理する各オブジェクトファイルの属性をチェックします。リンカによってそれ以前にロードされたオブジェクトファイルの属性と互換性のない属性が検出された場合、リンカはエラーを生成します。
このエラーには 以下の 3 つの一般的な原因があり、それぞれの原因に対して異なるメッセージが表示されます。

- エラー: L6242E:オブジェクトの属性がイメージの属性と互換性がないため、オブジェクト foo.o をリンクできません。8 バイトデータ型に対する 4 バイト境界整列の要求が、8 バイトデータ型に対する 8 バイト境界整列の要求と競合しています。

  この問題は、ADS または RVCT 1.2 を使用してビルドされたオブジェクトに RVCT 2.0 以降を使用してビルドされたオブジェクトをリンクしようとすると発生する場合があります。ADS および RVCT 1.2 では、double および long long データ型は、4 バイト境界で整列されます (-Oldrd コンパイラオプションまたは __align キーワードを使用した場合以外)。RVCT 2.0 では、ABI が変更されたため、double および long long データ型が 8 バイト境界で整列されます。

  この変更は、double または long long データ型を使用している ADS および RVCT 1.2 のオブジェクトおよびライブラリが RVCT 2.0 以降を使用してビルドされたオブジェクトおよびライブラリとの直接的な互換性を失ったことにより、リンカによる属性の不調和がレポートされるようになったことを意味します。

  古い ADS C オブジェクトと同時に RVCT 2.x または 3.0 C オブジェクトを使用するには、RVCT 2.x または 3.0 C コードを --apcs /adsabi コマンドラインオプションを使用してコンパイルします。このオプションは RVCT 2.2 では廃止予定、RVCT 3.1 では廃止されています。

- エラー: L6242E:オブジェクトの属性がイメージの属性と互換性がないため、オブジェクト foo.o をリンクできません。... 純粋エンディアン double 型は、複合エンディアン double と対立します。

  これは、ARM コンパイラツールチェーンを使用してビルドされたオブジェクト、古い SDT オブジェクトのある RVCT または ADS、あるいは、コンパイラオプション --fpu softfpa または --fpu fpa のいずれかを使用してビルドされたオブジェクトをリンクしようとすると発生する場合があります。SDT は、標準的でないリトルエンディアン double 型とビッグエンディアン long long を使用していました。ただし、ADS と RVCT は、--fpu softfpa オプションまたは --fpu fpa オプションが使用された場合以外は、業界標準である double 型と long long 型を使用します (これらのオプションは RVCT 2.1 以前のバージョンでのみサポートされます)。リトルエンディアン double 型とビッグエンディアン long long 型とは異なる形式を使用するオブジェクトファイルをリンクしようとすると、リンカによってエラーが報告されます。

  RVCT または ARM コンパイラツールチェーンを使用してプロジェクト全体を再ビルドすることを推奨します。オブジェクトまたはライブラリのソースコードがない場合、--fpu softfpa を使用してコードをコンパイルし直して下さい。

- エラー: L6242E:オブジェクトの属性がイメージの属性と互換性がないため、オブジェクト foo.o をリンクできません。...VFP と FPA の不調和

  このエラーは、通常、異なる --fpu オプションを使用してビルドされたオブジェクトをリンクしようとすると発生します。同じ --fpu オプションを使用してプロジェクト全体を再ビルドすることを推奨します。

以下の FAQ を参照して下さい。

Are legacy objects and libraries compatible with my project?

L6243E : セレクタは削除された未使用のセクションとのみ一致します。FIRST/LAST になるセクションがありません。

このセレクタと一致するすべてのセクションが、使用されないため、イメージから削除されました。詳細については、--info unused を使用して下さい。

L6244E : <type> の領域 <regionname>のアドレス(<addr>)が、<align> バイト境界に整列されていません。

L6245E : 要求された ZI セクション '<name>' を作成できませんでした。

L6248E : <objname>(<secname>)(<attr1>、領域 '<r1>' 内)で、<rtype> 再配置を <symname>(<attr2>、領域 '<r2>' 内)に対して指定することはできません。

このエラーは、位置非依存(PI)コードをビルドしようとすると発生する場合があります。以下のサンプルコードを考えてみます。

```
#include <stdio.h> char *str = "test"; int main(void) {     printf ("%s",str); }
```

これを以下のようにコンパイルおよびリンクしてみます。

```
armcc -c --apcs /ropi/rwpi pi.c
armlink --ropi --rwpi pi.o
```

リンカによって以下のエラーメッセージが表示されます。

```
エラー: L6248E:PI 領域 'ER_RW' 内の pi.o(.data) は、PI 領域 'ER_RO' 内の .conststring にアドレス型の再配置を持つことはできません。
```

これは、コンパイラが、.conststring セクション内の文字列のアドレスに初期化される必要のあるグローバルポインタ str を生成するためです。ただし、絶対アドレスは、PI システム内のアドレスを使用できないため、リンク手順が失敗します。

このエラーを解消するには、明示的なポインタを回避するようにコードを記述し直す必要があります。このために、以下のいずれかの方法を実行します。

- 以下のようにグローバルポインタの代わりにグローバル配列を使用します。

  ```
  #include <stdio.h> const char str[] = "test"; int main(void) {     printf ("%s",str); }
  ```

- 以下のようにグローバルポインタの代わりにローカルポインタを使用します。

  ```
  #include <stdio.h> int main(void) {     char *str = "test";     printf ("%s",str); }
  ```

注

以下のようなポインタ配列を使用する場合

```
char * list[] = {"zero", "one", "two"};
```

リンカは、配列内の各要素に対して別々のエラーを報告します。この場合、リストに対して、以下のような、最初の次元に配列内の要素の数があり、2 番目の次元に配列内の要素の最大サイズがある 2 次元配列を宣言することをお勧めします。

```
char list[3][5] = {"zero", "one", "two"};
```

以下のように printf() ステートメントを変更する必要があります。

```
printf("%s", list[1]);
```

コンパイラエラー番号 1359 を参照して下さい。

L6249E : エントリポイント(<address>)が複数のセクション内にあります。

L6250E : オブジェクト <objname> に、特別なシンボル <symbol> の不正な定義が含まれています。

L6251E : オブジェクト <objname> に、特別なシンボル <symbol> への不正な参照が含まれています。

L6252E : --xreffrom/--xrefto コマンドの無効な引数:'<arg>'

L6253E : 無効な SYMDEF アドレス:<number>.

L6254E : 無効な SYMDEF 型:<type>.
symdefs ファイルの内容が無効です。

『armlink ユーザガイド』の以下のセクションを参照して下さい。

symdefs ファイルの形式。

L6255E : ファイル <filename> を削除できませんでした:<reason>
指定されたファイルを削除しようとしたときに、I/O エラーが発生しました。このファイルが読み出し専用であるか、見つかりませんでした。

L6257E : <object>(<secname>)をオーバーレイされた実行領域 '<ername>' に割り当てることができません。
このメッセージは、スキャッタファイルに問題があることを示しています。

『armlink ユーザガイド』の以下のセクションを参照して下さい。

スキャッタファイルの構文。

L6258E : エントリポイント(<address>)がオーバーレイされた実行領域内にあります。
このメッセージは、スキャッタファイルに問題があることを示しています。

『armlink ユーザガイド』の以下のセクションを参照して下さい。

スキャッタファイルの構文。

L6259E : 予約されているワード '<name>' を <type> の領域名として使用することはできません。
<name> は予約語なので、領域には別の名前を選択して下さい。

L6260E : 同じ名前(<regionname>)を持つ複数のロード領域を使用することはできません。
このメッセージは、スキャッタファイルに問題があることを示しています。

『armlink ユーザガイド』の以下のセクションを参照して下さい。

スキャッタファイルの構文。

L6261E : 同じ名前(<regionname>)を持つ複数の実行領域を使用することはできません。
このメッセージは、スキャッタファイルに問題があることを示しています。

『armlink ユーザガイド』の以下のセクションを参照して下さい。

スキャッタファイルの構文。

L6263E : <regionname> の <addr> アドレスを <regtabregionname> 内の <pi_or_abs> 領域テーブルから指定することはできません。
Region Table には、ZI をコピー、伸張、または作成する C ライブラリ初期化コードによって使用される情報が含まれています。このエラーメッセージは、Region Table で記述できないイメージ構造がスキャッタファイルで指定されている場合に生成されます。

このエラーメッセージは、多くの場合、PI と非 PI ロード領域が同じイメージ内に混在しているときに表示されます。

L6265E : 非 PI セクション <obj>(<sec>)を PI 実行領域 <er> に割り当てることはできません。
この問題は、リンカコマンドラインで誤った ARM ライブラリを明示的に指定することによって発生する場合があります。以下のいずれかを実行して下さい。

- ARM ライブラリの明示的な指定を削除します。
- ライブラリ(例えば c_t.l)を正しいライブラリに置き換えます。

L6266E : RWPI セクション <obj>(<sec>) を非 PI 実行領域 <er>に割り当てることはできません。
--apcs=/rwpi でコンパイルされたファイルが、PI 属性を持たない実行領域に配置されています。

L6271E : ロード領域 <regname> に対して複数の相互に排他的な属性が指定されています
このメッセージは、スキャッタファイルに問題があることを示しています。

L6272E : 実行領域 <regname> に対して複数の相互に排他的な属性が指定されています
このメッセージは、スキャッタファイルに問題があることを示しています。

L6273E : セクション <objname>(<secname>)に相互に排他的な属性(READONLY と ZI)が含まれています
このメッセージは、オブジェクトファイルに問題があることを示しています。
L6275E : COMMON セクション <obj1>(<sec1>)は、<sym>(<obj2>(<sec2>)で定義)を定義しません
同名の COMMON セクションがいくつかある場合、リンカはそのうちの 1 つをイメージに追加するものとして選択し、他のすべてを破棄します。選択された COMMON セクションには、選択されなかったすべての COMMON セクションで定義されているすべてのシンボルを定義する必要があります。そうしない場合、選択されなかったセクションに定義されているシンボルは再び未定義になります。選択されなかったコピーで定義されているシンボルが、選択されたコピーで定義されていない場合、リンカによってエラーが生成されます。このエラーは、通常、コンパイラのエラーによって発生します。購入元にお問い合わせ下さい。
L6276E : アドレス <addr> が <s1>(<sp1>(<obj1>)から <src1> を経由)と <s2>(<sp2>(<obj2>)から <src2> を経由)の両方としてマークされています。
イメージは、指定されたアドレスに、矛盾する複数のマッピングシンボルを持つことはできません。イメージ内の各ワードの内容は、A32(`$a`)または T32(`$t`)コード、DATA(`$d`)、または NUMBER として、一意に型を決められているためです。ワードが A32 コードと DATA の両方であることはできません。この問題は、コンパイラのエラーを示している場合があります。購入元にお問い合わせ下さい。
L6277E : 不明なコマンド '<cmd>' です。
L6278E : 予期される <str> がありません。
L6279E : <sym> のあいまいなセレクタ('<sel1>' と '<sel2>')が見つかりました。
L6280E : 指定されたパターンを使用して <sym> の名前を変更することはできません。
*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

RENAME ステアリングファイルコマンド。

L6281E : <sym1> と <sym2> の両方の名前を <newname> に変更することはできません。
*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

RENAME ステアリングファイルコマンド。

L6282E : <sym> の名前を <newname> に変更することはできません。これは、その名前のグローバルシンボルが存在しているためです(<obj> 内で定義)。
*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

RENAME ステアリングファイルコマンド。

L6283E : オブジェクト <objname> に、シンボル <symbolname> への不正なローカル参照が含まれています。
ローカルシンボルは、常にオブジェクト自体の内部で定義されるため、オブジェクトには、ローカルシンボルへの参照を含めることはできません。
L6285E : 再配置不能なロード領域 <lr_name> に R-Type のダイナミックな再配置が含まれています。最初の R-Type のダイナミックな再配置が、<object> (<secname>)内のオフセット 0x<offset> で見つかりました。
このエラーは、2 つのセグメントが実行時に離れた場所に移動される場合に、各セグメント間に PI 参照があると発生します。リンカが 2 つのセクションを実行時に離れた場所に移動できると判断した場合、リンカはセクションが静的にリンクされたアドレスから移動されても解決できる再配置(R-Type の再配置)を生成します。しかし、PI 領域には位置に依存してはならないという制約があり、他のセクションと関係がある再配置を持つことはできないため、リンカはこの再配置に失敗します(エラー L6285E を生成)。

L6286E : 再配置 #<rel_class>:<rel_number>(<objname>(<secname>)内)に {symname|%s} が関係しています。値(<val>)が(<rtype>)の範囲(<range>)を超えています。

このエラーは、通常、以下の状況下で発生します。

- 手書きのアセンブリコードで、シンボルにオフセットを保持するための命令オペコード内のビット数が不足している場合。

  たとえば、A32 状態の LDR または STR 命令のオフセット範囲は ±4095 です。
- リンカがイメージの大きなコードセクションの周りにベニアを配置することが難しい場合。

  リンカは、非常に大きなセクションの近くにベニアを配置する場合、大きなセクションの前と後のどちらにベニヤを配置するかを決める必要があります。リンカがベニアを配置してから、さらにベニアを配置する必要がある場合は、元のベニヤとそのターゲット間に配置されます。これにより、ベニアとそのターゲットとの間の距離が離れます。

  リンカは自動的にベニアとそのターゲット間の距離を少し離すことができます。ただし、元のベニヤとそのターゲット間に配置されるベニアの数が多いと、ターゲットが範囲外に移動する可能性があります。この場合は、リンカはメッセージ L6286E を生成します。

  この問題の回避策として、リンカが多くのベニアを配置している領域から、大きなコードセクションの距離を離すことをお勧めします。これは、大きなセクションをそれ自体の領域に配置するか、スキャッタロード記述ファイルの +FIRST ディレクティブを使用して位置する領域に最初に配置することによって実行できます。

  以下に例を示します。

  ```
  LOAD 0x0A000000 0x1000000 {    ROM1 +0x0    {       *(+RO)    } }
  ```

  これを次のように変更します。

  ```
  LOAD 0x0A000000 0x1000000 {    ROM1 +0x0    {       *(+RO)    }    ROM1A +0x0
  {       large.o (+RO)    } }
  ```
- .ARM.exidx 例外処理インデクステーブルが、異なる実行領域または例外処理コードからかなり離れた領域に配置されている場合。

  .ARM.exidx 例外処理インデクステーブルが単一の実行領域に配置されている必要があります。また、これらのテーブルと C++ 例外処理を使用する C++ コードの間の距離も -0x40000000 ～ 0x3fffffff の範囲内でなければなりません。そうしないと、リンカによって以下のエラーが表示されます。

  ```
  L6286: Value(0x9ff38980) out of range(-0x9ff38980) out of range(-0x40000000
  - 0x3fffffff) for relocation #0 (R_ARM_PREL31), wrt symbol xxx in
  XXXX.o(.ARM.exidx)
  ```

  この動作は、*『ARM アーキテクチャ用例外処理 ABI』*(EHABI)で指定されています。EHABI では、.ARM.exidx が使用する R_ARM_PREL31 再配置は、再配置を計算するために最も高いビット(ビット 31)を使用しないことが定義されています。

  多くの場合、この原因は、.ARM.exidx セクションにアクセスする必要がある C++ コードが分割されて別々の実行領域に配置され、有効な範囲(-0x40000000 ～ 0x3fffffff)以外になっているためです。

  このエラーを解決するには、分割されている実行領域間にメモリがある場合は、.ARM.exidx セクションをセレクタ *(.ARM.exidx) で配置することです。以下に例を示します。

  ```
  LOAD_ROM 0x00000000 {    ER1 0x00000000   ; ER2 ～ ER1 の距離は
  {                ; 範囲外です。         file1.o (+RO) ; C++ ソースから。*
  (+RO)    }    ERx 0x30000000    {       *(.ARM.exidx) ; ARM.exidx からの ER1 および
  ER2 までの距離は両方とも範囲内です。}    ER2 0x60000000     {       file2.o (+RO) ; C+
  + ソースから。}    ER3 +0    {       * (+RW, +ZI)    } }
  ```

  別の方法としては、テーブルに十分近い実行領域 (-0x40000000 ～ 0x3fffffff の範囲内)にコードを配置して下さい。

それ以外の場合は、最新のパッチを*[Downloads]*からインストールして下さい。

詳細については、以下を参照して下さい。

「What does "Error: L6286E: Value out of range for relocation" mean?」

http://infocenter.arm.com/help/topic/com.arm.doc.ihi0038-/index.html

L6287E : 不正な配列制約(<align>)が <objname>(<secname>)に対して指定されています。
不正なアライメントが ELF オブジェクトに対して指定されました。

L6291E : 固定実行領域 <ername> のロードアドレス <addr> を割り当てられません。ロードアドレスは、使用できる次のロードアドレス <load_addr> 以上である必要があります。
*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

実行領域の属性。

L6292E : 不明な属性 '<attr>' が領域 <regname> に対して指定されていますが、無視されます。
このエラーメッセージは、FIXED 属性を持つ実行領域に固有のものです。FIXED は、ロードアドレスを実行アドレスと同じにすることを意味します。リンカがこのようにできるのは、実行アドレスがロード領域内で次に使用可能なロードアドレス以上である場合だけです。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

L6294E : <type> の領域 <regionname> は、32 ビットアドレス空間(ベース <base>、サイズ <size> バイト)を超えています。
このエラーメッセージは、スキャッタファイルの問題に関連しています。

L6295E : 再配置 #<rel_class>:<rel_number><symname> SBREL 再配置に関する再配置 #<rel_class>:<rel_number>(<objname>(<secname>)内)では、イメージが RWPI である必要があります。

L6296E : 特別なシンボル <sym1> の定義が不正です。これは、シンボル <sym2> が絶対であるためです。
L6188E を参照して下さい。

L6300W : Common セクション <object1>(<section1>)がセクションの定義 <object2>(<section2>)より大きくなっています。
この問題は、コンパイラのエラーを示している場合があります。購入元にお問い合わせ下さい。

L6301W : ファイル <filename> が見つかりませんでした:<reason>
デフォルトディレクトリで、指定されたファイルが見つかりませんでした。

L6302W : 複数の SHLNAME エントリは無視されます。
エディットファイル内で使用できる SHLNAME エントリは 1 つだけです。最初のエントリだけがリンカに認識されます。後のすべての SHLNAME エントリは、無視されます。

L6304W : 重複する入力ファイル <filename> は無視されます。
指定されたファイル名が、入力ファイルのリスト内に複数あります。

L6305W : イメージにエントリポイントが含まれていません(指定されていないか、複数選択されているために設定されていません)。
ELF イメージのエントリポイントが指定されていないか、リンクされたエントリポイントを持つセクションが複数あるために設定されていません。リンカオプション --entry を使用して、単一の一意のエントリを指定する必要があります。以下に例を示します。

```
--entry 0x0
```

または

```
--entry <label>
```

組み込みシステムの場合は、ラベル形式が一般的です。

L6306W : '<attr1>' セクション <objname>(<secname>)では、'<attr2>' 関数 <sym> のアドレスを使用できません。
L6238E を参照して下さい。

L6307W : 再配置 #<rel_class>:<rel_num>(<objname>(<secname>)内)は <sym> に関係しています。境界整列されていないデスティネーションへの分岐です。

L6308W : <membername> と一致するオブジェクトがライブラリ <libraryname> 内にありません。
ライブラリ内のオブジェクトの名前がリンクラインで指定されましたが、ライブラリ内にその名前のオブジェクトがありませんでした。

L6309W : ライブラリ <libraryname> にメンバが含まれていません。
リンカコマンドラインでライブラリが指定されましたが、ライブラリ内にメンバがありませんでした。

L6310W : ARM ライブラリが見つかりません。
この問題は、多くの場合、--libpath の引数が正しくないことが原因です。

--libpath リンカオプションを使用して正しいパスを設定します。Windows インストールのデフォルトのパスは、次のようになります。

install_directory\lib

このパスに、以下のものが含まれていないことを確認して下さい。

- \armlib
- \cpplib
- 末尾の円記号(\)。これらは、リンカによって自動的に追加されます。

リンカがライブラリを探す場所を表示するには、--verbose または --info libraries を使用します。

『*armlink ユーザガイド*』の以下のセクションを参照して下さい。

- *--info=topic[,topic,...]*。
- *--libpath=pathlist*。
- *--verbose*。

『*スタート ガイド*』の以下のセクションを参照して下さい。

- *ツールチェーンの環境変数*。

L6311W : シンボル <symbol>(<objname> から参照)が定義されていません。
L6218E を参照して下さい。

L6312W : <type> 領域の説明(領域 <region>)が空白です。

L6313W : 現在は <oldname> をセクションセレクタとして使用することはできません。代わりに <newname> を使用して下さい。
例えば、スキャッタファイルで IWV$$Code を使用することはできません。IWV$$Code を Veneer$$Code と置き換えて下さい。

L6314W : パターン <module>(<section>)に一致するセクションがありません。
以下に例を示します。

```
パターン foo.*o(ZI)に一致するセクションがありません。
```

これは、次のいずれかの理由による可能性があります。

- ファイル foo.o がスキャッタファイルで指定されていますが、リンカコマンドラインのリストには含まれていません。この問題を解決するには、リンクラインに foo.o を追加して下さい。
- foo.o の ZI データをスキャッタファイルを使って配置しようとしましたが、foo.o に ZI データが含まれていませんでした。この問題を解決するには、スキャッタファイル内の foo.o の行から +ZI 属性を削除します。
- コードとデータを特定のアドレスに配置するために、ソースコードで __attribute__((at(*address*))) を使用しました。また、スキャッタファイルで *(.ARM.__AT_*address*) を使用していますが、アドレスを 8 桁の 16 進数として指定していません。例えば、ソースコードで __attribute__((at(0x10000))) を使用する場合は、セクション名をスキャッタファイルで *(.ARM.__AT_0x00010000) として指定する必要があります。

『*armlink ユーザガイド*』の以下のセクションを参照して下さい。

- *特定アドレスへの関数とデータの配置方法*。
- *__at を使用した特定のアドレスへのセクションの配置*。

L6315W : オブジェクト <objname> 内の複数のビルド属性シンボルは無視されます。
オブジェクトには、最大で 1 つの絶対的な BuildAttribute$$... シンボルを含めることができます。オブジェクトシンボルテーブルの、このような最初のシンボルだけがリンカに認識されます。後のすべてのシンボルは、無視されます。

L6316W : セクション <sec_no> のオブジェクト <objname> 内の複数のビルド属性シンボルは無視されます
オブジェクトには、特定のセクションに適用される、最大で 1 つの BuildAttribute$$... シンボルを含めることができます。オブジェクトシンボルテーブルの、このような最初のシンボルだけがリンカに認識されます。後のすべてのシンボルは、無視されます。

L6317W : <objname>(<secname>)は、'<attr1>' 関数 <sym> のアドレスを使用できません。これは、イメージに '<attr2>' 関数が含まれているためです。

L6318W : <objname>(<secname>)に、コードではないシンボル <sym> への分岐が含まれています。
この警告は、通常はアセンブラファイルに、同じファイル内の(別の AREA 内の)コードではないシンボルへの分岐があることを意味しています。これは、多くの場合、コードではなくデータがある場所のラベルまたはアドレスへの分岐です。

以下に例を示します。

```
AREA foo, CODE
B bar
AREA bar, DATA
DCD 0
END
```

この場合、次のようなメッセージが表示されます。

```
init.o(foo) contains branch to a non-code symbol bar.
```

次のように、デスティネーションの名前がない場合は、

```
BL 0x200 ; Branch with link to 0x200 bytes ahead of PC
```

次のメッセージが表示されます。

```
bootsys.o(BOOTSYS_IVT) contains branch to a non-code symbol <Anonymous Symbol>.
```

この警告は、GCC によって生成されたオブジェクトをリンクした場合にも表示されることがあります。GCC は、各オブジェクトへの内部的な参照に、リンカの再配置を使用します。これらの再配置のターゲットは、通常、ターゲットがコードであるかデータであるかをリンカが判断するための適切なマッピングシンボルを持たないため、警告が生成されます。一方、armcc は、このようなすべての参照をコンパイル時に解決します。

L6319W : <cmd> コマンドは無視されます。セクション <objname>(<secname>)が見つかりません。
例えば、Linux アプリケーションをビルドするとき、

```
--keep *(.init_array)
```

をメイクファイルのリンカコマンドラインで指定したとしても、このセクションは C++ なしでビルドした場合には存在しないため、次の警告が報告されます。

```
Ignoring --keep command. Cannot find section *(.init_array)
```

通常は、この警告を無視するか、--diag_suppress 6319 を使って非表示にすることができます。

L6320W : <cmd> コマンドは無視されます。引数 '<argname>' が見つかりません。

L6323W : 再配置 #<rel_class>:<rel_number>(<objname>(<secname>)内)は <sym> に関係しています。複数のバリアントが存在します。<type> バリアントを使用してあいまいさを解決します

L6324W : ロード領域 <regname> に指定された <attr> 属性は無視されます。
この属性は、実行領域のみに適用されます。ロード領域に対して指定された場合、リンカはこの属性を無視します。

L6325W : ベースアドレスからの +offset を使用する <attr> 属性が領域 <regname> に対して指定されていますが、無視されます。

この属性は、+offset 形式のベースアドレスを使用する領域には適用されません。+offset 形式を使用する領域に対して指定された場合、リンカはこの属性を無視します。

+offset 形式のベースアドレスを使用する領域は、PI、RELOC、および OVERLAY 属性を以下のいずれかから継承します。

- 記述内の前の領域。
- ロード領域内の最初の実行領域である場合は、親ロード領域。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

- *ロード領域のアドレス属性の継承規則*。
- *実行領域のアドレス属性の継承規則*。
- *RELOC アドレス属性の継承規則*。

L6326W : 非ルート実行領域 <ername> の ZEROPAD 属性は無視されます。

ZEROPAD は、ルート実行領域のみに適用されます。ルート領域は、その実行アドレスがロードアドレスと同じである領域です。そのため、実行時に移動やコピーが必要ありません。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

実行領域の属性。

L6329W : パターン <module>(<section>)は、削除された未使用のセクションとのみ一致します。

このパターンと一致するすべてのセクションが、使用されないため、イメージから削除されました。詳細については、--info unused を使用して下さい。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

- *未使用セクションの削除*。
- *--info=topic[,topic,...]*。

L6330W : シンボル <symbol>(<objname> から参照)が定義されていません。未使用のセクションが削除されました。

これは、未使用のセクション内のベースシンボルとリミットシンボルが参照されていたことを意味しています。詳細については、--info unused を使用して下さい。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

L6331W : パターン <pat> と一致する有効なグローバルシンボルがありません。

L6332W : シンボル <sym1>(<obj1> から参照)が定義されていません。シンボル <sym2> に解決されました。

L6334W : 領域 <regname> のオーバーアライメント <overalignment> は負にできません。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

実行領域と入力セクションのオーバーアライメント。

L6335W : <objname>(<secname>)内の ARM インターワーキングコードに、ARM 非インターワーキングコードに対する無効な末尾の呼び出しが含まれている可能性があります。

コンパイラは、コードのサイズとパフォーマンスを向上させるために、末尾呼び出しの最適化を行うことができます。しかし、ARMv4T コードでは、T32 インターワーク(IW)関数が(ベニアによって)A32 IW 関数を呼び出すシーケンスで、A32 非インターワーク(~IW)関数の末尾呼び出しが行われる問題があります。A32 非 IW 関数からの復帰では、適切な BX 命令を使わずに、復帰アドレスをスタックから PC にポップする場合があります。リンカはこの状況を検出してこの警告を報告できます。

T32 IW からの T32 非 IW の末尾呼び出しは発生しません。T32 の B での末尾呼び出しは範囲が狭く、同じ ELF セクション内の関数として生成されるだけであり、それらも T32 になるためです。

この警告は、オブジェクトに無効な末尾呼び出しが含まれている*可能性がある*という点で問題ですが、リンカはオブジェクトの属性を調べているだけであり、そのセクションの内容を調べてはいないので、確実ではありません。

警告を回避するには、コードベース全体をユーザライブラリも含めて、--apcs /interwork で再コンパイルするか、手動で A32 IW 関数を調べて末尾呼び出し(通常の分岐 B 命令を使って関数呼び出しが行われている個所)を確認し、この警告が実際に問題であるかどうかをチェックします。この警告は、--diag_suppress L6335W を使用して非表示にできます。

L6337W : Common コードセクション <o1>(<s1>)および <o2>(<s2>)に互換性のない浮動小数点リンケージが含まれています

L6339W : 実行領域 <er_name> の RELOC 属性は無視されます。

実行領域には、明示的に RELOC 属性を指定することはできません。この属性になるのは、+offset 形式のアドレシングを使用している場合に、親ロード領域または前の実行領域から属性を継承するときだけです。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

実行領域の属性。

L6340W : リンク型 <linktype> の場合、first オプションと last オプションは無視されます

--first および --last オプションは、部分的にリンクされたオブジェクトを作成する場合には無意味です。

L6366E : <object> 属性 <attr> は、指定された CPU と FPU の属性 <cli> <diff> と互換性がありません。

L6367E : <object>(<section>)属性 <attr> は、指定された CPU と FPU の属性 <cli> <diff> と互換性がありません。

L6368E : <object>(<section>)で定義された <symbol> の属性 <attr> は、指定された CPU と FPU の属性 <cli> <diff> と互換性がありません。

L6369E : <object>(ABSOLUTE)で定義された <symbol> は、指定された CPU と FPU の属性 <cli> <diff> と互換性がありません。

L6370E : CPU <cpu> は FPU <fpu> と互換性がありません

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

- *--cpu=name*。
- *--fpu=name*。

L6371E : CPU と FPU の属性を追加しています:<attrs>

L6372E : イメージには少なくとも 1 つのロード領域が必要です。

L6373E : libattrs.map ファイルがシステムライブラリディレクトリ <dir> で見つかりません。ライブラリの選択が不適切な可能性があります。

L6384E : 名前 <region> のロード実行領域がまだ行 <line> に見つかりません。

原因として、スキャッタファイルでの制限計算で現在のベースアドレスを使用したことが考えられます。以下に例を示します。

```
ER_foo 0 ImageBase(ER_foo)
```

L6385W : 行 <line> で加算オーバーフローが発生しました

L6386E : 実行領域の式は、行 <line> のベースアドレス計算でのみ使用できます
L6387E : ロード領域の式は、行 <line> の ScatterAssert 式でのみ使用できます
『armlink ユーザガイド』の以下のセクションを参照して下さい。

ScatterAssert 関数とロードアドレスに関連する関数。

L6388E : 行 <line> で ScatterAssert 式 <expr> のエラーが発生しました
『armlink ユーザガイド』の以下のセクションを参照して下さい。

ScatterAssert 関数とロードアドレスに関連する関数。

L6389E : 行 <line> のロード領域 <name> が未完了で、領域の長さに依存する演算を使用できません
L6390E : 条件演算子 (expr) ?行 <line> の(expr):(expr)に(expr)がありません。
『armlink ユーザガイド』の以下のセクションを参照して下さい。

- *スキャッタファイル内の式の評価について*。
- *スキャッタファイル内の式の規則*。

L6404W : 実行領域 <name> には、EMPTY、ZEROPAD および PADVALUE の組み合わせよりも FILL 値を使用して下さい。
『armlink ユーザガイド』の以下のセクションを参照して下さい。

実行領域の属性。

L6405W : どの .ANY セレクタもセクション <name>(<objname>)に一致しません。
『armlink ユーザガイド』の以下のセクションを参照して下さい。

.ANY モジュールセレクタによる未割り当てセクションの配置。

L6406W : セクション <name>(<objname>)と一致する .ANY セレクタがある実行領域のスペースがありません。
この問題は、.ANY を含むスキャッタファイル領域に、リストされているセクションを配置するための十分なスペースがない場合に発生します。セクションのための十分なスペースがあるように、スキャッタファイルを変更する必要があります。

『armlink ユーザガイド』の以下のセクションを参照して下さい。

.ANY モジュールセレクタによる未割り当てセクションの配置。

L6407W : 集合サイズ 0x<size> バイトのセクションは .ANY セレクタに収まりません。
この警告は、どの .ANY セレクタにも配置できないイメージデータの総量を特定します。

例えば、次のように、ZI データの量に対して小さすぎる実行領域に .ANY(+ZI) が配置される場合です。

```
ROM_LOAD 0x8000 {     ROM_EXEC 0x8000     {          .ANY(+RO,+RW)     }     RAM +0
0x{...}&lt;&lt;&lt; 領域の最大長が小さすぎる     {          .ANY(+ZI)     } }
```

『armlink ユーザガイド』の以下のセクションを参照して下さい。

.ANY モジュールセレクタによる未割り当てセクションの配置。

L6408W : 出力は --fpic ですが、<obj> のセクション <sec> には FPIC 属性がありません。
L6409W : 出力は --fpic ですが、オブジェクト <obj> には FPIC 属性がありません。
L6410W : STV_DEFAULT の可視性 <vis> がないシンボル <sym> は、静的に解決される必要があり、<lib> の定義は使用できません。
L6411W : 起動シンボル <name> の定義と互換性のあるライブラリが存在しません。
L6412W : オブジェクト <obj> のセクション <sec> に対して、オブジェクト <srcobj> のセクション <srcsec> の非 R_ARM_ABS32 再配置のマージを無効にします。
L6413W : オブジェクト <obj> のセクション <sec> のマージを無効にします。セクションに、配置が間違った文字列が含まれています。
L6414E : --ropi が --rwpi または --rw-base なしで使われています。
このエラーは、ARM コンパイラ 6.0 でサポートされていないオプションに関するものです。

L6415E : このイメージと互換性のあるライブラリの一意のセットが見つかりませんでした。--cpu オプションを使用して特定のライブラリを選択することを推奨します。
*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

--cpu=name。

L6416E : <objname>(<secname>)内の <relclass>:<idx> にある再配置 <type> は、そのターゲットからのオフセット <offset> がある状態ではベニアを挿入することができません。
L6417W : 再配置 #<rel_class>:<rel_number>(<objname>(<secname>)内)は、予約済みのタグ付けシンボル(#<idx>)と関係しています。
L6418W : <objname>(<secname>)で定義されたタグ付けシンボル <symname> は、認識されません。
L6419W : 未定義のシンボル <symbol>(<objname> から参照)がインポートされました。
L6420E : 認識された型ではないため、<oepname>(<secname>:<secnum>)は無視されます。
L6422U : PLT の生成には ARM 命令をサポートしているアーキテクチャが必要です。
プロシージャリンクテーブル(PLT)を生成するリンカの場合は、A32 命令セットをサポートするターゲットを使用する必要があります。たとえば、Cortex-M3 ターゲットに対して、リンカは PLT を生成できません。
L6423E : 同じコレクション内では、セクション <secname> に種類の異なるソート属性を含めることはできません。
L6424E : 同じコレクション内では、セクション <secname1> およびセクション <secname2> を別の実行領域に分けることはできません。
L6425E : 同じコレクション内では、セクション <secname> に長さの異なるセクション名を付けることはできません。
L6426E : 同じコレクション内では、セクション <secname> は重複した名前を持つことはできません。
L6427E : <sym> の名前は既に <name> に変更されているため、<newname> に変更できません。
L6429U : オープンするファイルの最大数を <val> に設定しようとしましたが、エラーコード <error> により失敗しました。
armlink がいつでも開いておけるファイルハンドルの数を増やそうとしましたが、失敗しました。
L6431W : <object>(<section>)で定義されたシンボル <symbol> の互換性のない列挙型サイズ属性は無視されます。
L6432W : オブジェクト <object>(<section>)の互換性のない列挙型サイズ属性は無視されます。
L6433W : オブジェクト <object> の互換性のない列挙型サイズ属性は無視されます。
L6434W : <object>(<section>)で定義されたシンボル <symbol> の互換性のない wchar_t サイズ属性は無視されます。
L6435W : セクション <object>(<section>)の互換性のない wchar_t サイズ属性は無視されます。
L6436W : オブジェクト <object> の互換性のない wchar_t サイズ属性は無視されます。
L6437W : 再配置 #<rel_class>:<idx>(<objname>(<secname>)内)は <armsym> に関係しています。オブジェクト <armobjname> の型なしシンボルへの分岐の再配置。ターゲットの状態が不明です。
L6438E : __AT セクション <objname>(<secname>)のアドレス <address> は、少なくとも 4 バイト境界で整列されている必要があります。
L6439W : <objname>(<secname>)で複数回定義されているグローバルシンボル <sym> が、<selobj>(<selsec>)で定義されているシンボルのために拒否されました。
L6440E : リンク時のコード生成で予期しないエラーが発生しました
L6441U : 開いているファイルの最大数を取得するためのシステムコールがエラー <error> のために失敗しました。
L6442U : リンカには開かれているファイルが最低でも <min> 個必要です。現在のシステムの上限は <max> ファイルです。
L6443W : 領域 <region> のデータ圧縮が無効になっています。領域には、圧縮されたアドレスに依存しているシンボル <symname> への参照が含まれています。
リンカでは、領域の内容が圧縮前に固定される必要があり、圧縮後は内容を変更できません。そのため、圧縮可能領域は圧縮処理に依存するメモリ位置を参照できません。
L6444I : シンボルの可視性:<symname> は <visibility> に設定されています。
L6445I : シンボルの可視性:<symname> は、既存の <old_vis> および新しい <new_vis> から <set_vis> にマージされました。
L6447E : SHT_PREINIT_ARRAY セクションは共有オブジェクトでは使用できません。
L6448W : <filename> の処理中:<message>

L6449E : <filename> の処理中:<message>
L6450U : ライブラリ <libname> が見つかりません。
L6451E : Thumb を許可のうえ作成された <object> は、ARM 専用のリンクでは使用できません。
L6452E : Thumb を許可のうえ作成された <object>(<section>)は、ARM 専用のリンクでは使用できません。
L6453E : Thumb を許可のうえ作成された <object>(<section>)で定義された <symbol> は、ARM 専用のリンクでは使用できません。
L6454E : Thumb を許可のうえ作成された <object>(ABSOLUTE)で定義された <symbol> は、ARM 専用のリンクでは使用できません。
L6455E : シンボル <symbolname> には、<object1> と <object2> による廃止予定の ARM/Thumb 同義語定義が含まれています。
L6459U : テンポラリファイルを作成できませんでした。
L6462E : 共有ライブラリから <sym> への参照は、オブジェクト <obj> で Visibility が Hidden または Protected に設定された定義にのみ一致します。
L6463U : 入力オブジェクトに <archtype> 命令が含まれていますが、オブジェクト属性に基づく <archtype> アーキテクチャの有効なターゲットが見つかりません。--cpu オプションを使用して特定の CPU を選択することを推奨します。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

- *--cpu=name*。

L6464E : --dynamic_debug、--emit-relocs、および --emit-debug-overlay-relocs のうちの 1 つだけを選択できます。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

- *--dynamic_debug*。
- *--emit_debug_overlay_relocs*。
- *--emit_relocs*。

L6467W : ライブラリから注釈がレポートされています:<msg>
L6468U : --base_platform とコードを含む複数のロード領域では、--pltgot=direct および --pltgot=none のみがサポートされています。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

- *--base_platform*。
- *--pltgot=type*。

L6469E : --base_platform は、非 RELOC 実行領域を含む RELOC ロード領域をサポートしていません。ロード領域 <lrname> の実行領域 <ername> のベースアドレスには、+0 を使用して下さい。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

- *--base_platform*。
- *RELOC アドレス属性の継承規則*。

L6470E : PLT セクション <secname> をロード領域 <lrname> 外に移動できません。
L6471E : 分岐の再配置 <rel_class>:<idx>(セクション <secname>、オブジェクト <objname>)は、オブジェクト <armobjname> の ARM 絶対 <armsym> シンボルを参照します。Thumb アドレスとして扱う場合は、エラーを非表示にします。
L6475W : IMPORT/EXPORT コマンドは、--override_visibility が指定されていない場合は無視されますステアリングファイル内の EXPORT コマンドを使うか、--undefined_and_export コマンドラインオプションを使ってエクスポートしようとしたシンボルは、可視性が低いため、エクスポートされませんでした。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

- *--override_visibility*。
- *--undefined_and_export=symbol*。
- *EXPORT*。

L6616E : RegionTable <sec_name>(<obj_name>)のサイズを増やすことができません
L6617E : ZISectionTable <sec_name>(<obj_name>)のサイズを増やすことができません

L6629E : 括弧が一致しません。)が予期されましたが、<character> が、位置 <col>、行 <line> で見つかりました

このメッセージは、スキャッタファイルの解析エラーを示しています。

L6630E : トークンの先頭が無効です。数値または(が予期されましたが、<character> が、位置 <col>、行 <line> で見つかりました。

このメッセージは、スキャッタファイルの解析エラーを示しています。

L6631E : 行 <line> にゼロによる除算があります

このメッセージは、スキャッタファイルの式評価エラーを示しています。

L6632W : 行 <line> で除算がアンダーフローしています

このメッセージは、スキャッタファイルの式評価エラーを示しています。

L6634E : '<filename>' のプリプロセッサコマンドが長すぎます。最大長は <max_size> です。

このメッセージは、スキャッタファイルのプリプロセッシングに問題があることを示しています。

L6635E : プリプロセッサによって生成された中間ファイル '<filename>' を開くことができません:<reason>

このメッセージは、スキャッタファイルのプリプロセッシングに問題があることを示しています。

L6636E : '<filename>' のプリプロセッサ処理ができませんでした。

このメッセージは、スキャッタファイルのプリプロセッシングに問題があることを示しています。

L6637W : 入力オブジェクトが指定されていません。少なくとも 1 つの入力オブジェクトまたはライブラリ(オブジェクト)を指定する必要があります。

少なくとも 1 つの入力オブジェクトまたはライブラリ(オブジェクト)を指定する必要があります。

L6638U : オブジェクト <objname> には、リンク順序の依存関係サイクルがあります。SHF_LINK_ORDER を使用してセクションをチェックして下さい。

L6640E : PDTTable セクションは最小のスタティックデータアドレスではありません。最小のスタティックデータセクションは <secname> です。

RWPI の共有ライブラリを実装するシステムは、*プロセスデータテーブル*(PDT)を使用します。これはリンカによって静的なリンク時に作成され、イメージのデータ領域の先頭に配置される必要があります。

このメッセージは、PDT をイメージのデータ領域の先頭に配置することをスキャッタファイルが許可していないことを示しています。

メッセージが表示されないようにするには、スキャッタファイルを調整して、PDT が正しく配置されるようにします。このメッセージは、間違ってオブジェクトファイルを --apcs=/rwpi でビルドした場合にも表示されます。

L6642W : <obj_name> が --vfe を使用してコンパイルされていないため、未使用の仮想関数の削除は正しく機能しない可能性があります

L6643E : セクション <sectionname> 内の仮想関数の削除情報は、間違ったセクションを参照しています。

このメッセージは、コンパイラのエラーを示している場合があります。購入元にお問い合わせ下さい。

L6644E : セクション <oepname>(<sectionname>)内の仮想関数の削除情報の読み出し中に、予期せずにバッファの最後に到達しました。

このメッセージは、コンパイラのエラーを示している場合があります。購入元にお問い合わせ下さい。

L6645E : セクション <oepname>(<sectionname>)内の仮想関数の削除情報が正しくありません:オフセット <offset> での再配置があるはずです。

このメッセージは、コンパイラのエラーを示している場合があります。購入元にお問い合わせ下さい。

L6646W : セクション <oepname>(<sectionname>)内の仮想関数の削除情報に、オフセット <offset> より前からのガベージが含まれています。

このメッセージは、コンパイラのエラーを示している場合があります。購入元にお問い合わせ下さい。

L6647E : <vcall_objectname>(<vcall_sectionname>)の仮想関数の削除情報は、セクション <curr_sectionname>(オブジェクト <curr_objectname>)、オフセット <offset> が仮想関数または RTTI への再配置であるのに、そのオフセットに再配置がないことを示していますが、これは誤りです。

このメッセージは、コンパイラのエラーを示している場合があります。購入元にお問い合わせ下さい。

L6649E : EMPTY 領域 <regname> に最大サイズを指定する必要があります。
*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

実行領域の属性。

L6650E : オブジェクト <objname> の Group セクション <sectionidx> に無効なシンボルインデクス <symidx> が含まれています。
L6651E : セクション <secname>(オブジェクト <objname> 内)に SHF_GROUP フラグがありますが、どのグループのメンバにもなっていません。
L6652E : データセクションのバイト順序を逆にすることができません。入力オブジェクトは <inputendian> で、要求されたデータバイト順序は <dataendian> です。
L6654E : 拒否されたローカルシンボル <symname> は、グループメンバ <objname>(<nongrpname>)以外から参照されています

このメッセージは、コンパイラのエラーを示している場合があります。購入元にお問い合わせ下さい。

L6656E : 内部エラー:vfe セクションのリストに <oepname>(<secname>)という非 vfe セクションが含まれています。

このメッセージは、コンパイラのエラーを示している場合があります。購入元にお問い合わせ下さい。

L6664W : 再配置 #<rel_class>:<rel_number>(<objname>(<secname>)内)は、シンボル(最後のマップシンボル #<last> の前の #<idx>)に関係しています。
L6665W : Lib$$Request$$armlib Lib$$Request$$cpplib が定義されていません。ARM ライブラリを検索しません。

この警告は、次のコードによって生成されます。

```
  AREA Block, CODE, READONLY
  EXPORT func1
  ;IMPORT ||Lib$$Request$$armlib||
  IMPORT printf
func1
  LDR r0,=string
  BL printf
  BX lr
  AREA BlockData, DATA
string DCB "mystring"
  END
```

リンカはライブラリを参照するようには指示されていないため、シンボル printf を見つけられません。

この問題は、以下のエラーも発生させます。

```
L6218E: Undefined symbol printf (referred from L6665W.o).
```

ライブラリが必要ない場合は、このメッセージを無視して下さい。そうでない場合は、以下の行のコメント化を解除して、エラーと警告が出ないようにして下さい。

```
IMPORT ||Lib$$Request$$armlib||
```

L6679W : 出力 ELF セクション #<sec> '<secname>' のデータは、圧縮に適していませんでした(<data_size> バイトが <compressed_size> バイトになりました)。
L6682E : マージセクション <oepname>(<spname>)はコードセクションです
L6683E : マージセクション <oepname>(<spname>)にサイズ 0 の要素があります
L6684E : セクション <spname>(オブジェクト <oepname> 内)に SHF_STRINGS フラグがありますが、SHF_MERGE フラグがありません
L6685E : セクション <spname>(オブジェクト <oepname> 内)に SHF_MERGE セクションへの分岐再配置 <rel_idx> があります
L6688U : 再配置 #<rel_class>:<rel_idx>(<oepname>(<spname>)内)は、負の要素を参照しています。
L6689U : 再配置 #<rel_class>:<rel_idx>(<oepname>(<spname>)内)。デスティネーションがマルチバイト文字の中間にあります
L6690U : マージセクション <spname>(オブジェクト <oepname> 内)にシンボルがありません
L6703W : セクション <er> は圧縮不能として暗示的にマークされています。

L6707E : 実行領域 <regionname> のパディング値は PADVALUE 属性を使用して指定されていません。
*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

実行領域の属性。

L6708E : <secname>(オブジェクト <oepname> 内)のデバッグフレームを処理できませんでした。
L6709E : <secname>(オブジェクト <oepname> 内)の fde を関連付けることができませんでした。
L6713W : オフセット <offset>(<oepname>(<secname>)内)の関数にはシンボルがありません。
L6714W : オブジェクト <oepname> の例外インデクステーブルセクション .ARM.exidx にデータが含まれていません。
L6720U : イメージ内に存在する例外テーブル <spname>(オブジェクト <oepname> 内)に --noexceptions が指定されています。
*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

--exceptions、*--no_exceptions*。

L6721E : セクション #<secnum> '<secname>' (<oepname> 内)は認識されないため、総称的に処理できません。
L6725W : ダイナミックな再配置があるため、未使用の仮想関数の削除は正しく機能しない可能性があります
L6728U : リンク順序の依存関係がセクション番号 <from> から無効なセクション番号 <to> に指定されています。
L6730W : 再配置 #<rel_class>:<index>(<objname>(<secname>)内)は <name> に関係しています。シンボルに ABI 型 <type>、以前の型 <legacy_type> があります。
この警告は、RVCT 2.0 と 2.1 間でのリンカの動作の変更に関係しています。

—————— **注** ——————

以下の例は、--strict_relocations を使用した場合、または入力オブジェクトが RVCT 2.0 以前の場合にのみ警告メッセージを生成します。

————————————

例:

```
    AREA foo, CODE, READONLY
    CODE32
    ENTRY
    KEEP
func proc
    NOP
    ENDP
    DCD foo
    END
```

RVCT 2.0 以前では、内容(この例では ARM コード)に基づきインターワーキングが必要かどうかがリンカで決定されます。RVCT 2.1 以降では、リンカは ABI に従います。ABI では、インターワーキングを適用するかどうかを決定するのはシンボルのタイプ(この例では STT_SECTION (データとして解釈される))であると定義されています。

最も単純な解決方法は、データをアセンブリソースファイル内の別のデータ領域へ移動することです。

また、--diag_suppress 6730 を使って、この警告を非表示にすることもできます。

L6731W : <secname> から参照されているセクションが存在しないため、未使用の仮想関数の削除は正しく機能しない可能性があります。
L6733W : <objname>(<secname>)は <lr1name> から <lr2name> へのオフセットの再配置を含んでいます。ロード領域は厳密に相対的である必要があります。
L6738E : 再配置 #<rel_class>:<relocnum><wrtsym> に関する再配置 #<rel_class>:<relocnum> (<oepname>(<secname>)内)は、GOT 相対再配置ですが、_GLOBAL_OFFSET_TABLE_ が未定義です。
GNU によって生成される一部のイメージは、_GLOBAL_OFFSET_TABLE_ という名前のシンボルを参照できます。再配置を生成する GOT Slot がなく、リンカが GOT base の適切なアドレスを取得できない場合は、リンカによってこのエラーメッセージが生成されます。
L6739E : バージョン '<vername>' と未定義のバージョン '<depname>' に依存関係があります。

L6740W : '<symverscr>' にバージョン '<vername>' のシンボル '<symname>' が定義されていますが、入力オブジェクト内にありません。
L6741E : バージョンが付与されたシンボルのバインディングは 'local:' または 'global:' である必要があります。
L6742E : '<oepname>' によって定義されたシンボル '<symname>' です。デフォルトのバージョンシンボル '<defversym>' と一致しません。
L6743E : 再配置 #<rel_class>:<index>(<oepname>(<spname>)内)に、別の定義を持つ <symname> が関係しています。内部の整合性チェックが失敗しました
L6744E : 再配置 #<rel_class>:<index>(<oepname>(<spname>)内)に未定義のシンボル <symname> が関係しています。内部の整合性チェック:
L6745E : ターゲット CPU <cpuname> は ARM をサポートしておらず、<objname>(<secname>)に ARM コードが含まれています
L6747W : ターゲットアーキテクチャを <oldversion> から <newversion> へ移行しています。
リンカは、廃止された ARMv3 を指定するオブジェクトを検出した場合、そのオブジェクトを ARMv4 にアップグレードして、ARM ライブラリで使用できるようにします。
L6748U : ファイル <oepname> にダイナミック配列、シンボルテーブル、またはストリングテーブルがありません。
L6751E : ソートアルゴリズム <str> を使用できません。
L6753E : CallTree ソートでは、CallTree ソート ER 内にエントリポイントが必要です。
L6761E : シンボル <symname> を削除します。
L6762E : <imgtype> のビルド時には '<type>' PLT エントリをビルドできません。
L6763W : 共有オブジェクトまたは DLL のビルド時には、'<optname>' を使用できません。オフに切り替えています
L6764E : Thumb シンボル <symname> をコールするターゲットアーキテクチャ 4T に対する PLT エントリを作成できません
L6765W : アーキテクチャ 4T オブジェクトのリンク中には、共有オブジェクトのエントリポイントは ARM 状態である必要があります。
この問題は、GNU C ライブラリとリンクする場合に発生することがあります。GNU スタートアップコード crt1.o にはエントリポイントに関するビルド属性がないため、リンカはどの実行状態(A32 または T32)でコードを実行すればよいのかを特定できません。GNU の C ライブラリスタートアップコードは A32 コードであるため、この警告は無視しても問題ありません。--diag_suppress 6765 を使用して、警告を非表示にすることもできます。
L6766W : アーキテクチャ 4T の PLT エントリは、インクリメンタルリンキングをサポートしていません。
L6769E : 再配置 #<rel_class>:<relocnum>(<oepname>(<secname>)内)は <wrtsym> に関係しています。シンボルに GOTSLOT がありません。
L6770E : ダイナミック配列のサイズおよび内容を変更するタイミングが遅すぎて、変更を確定できません。
L6771W : <oepname>(<secname>)は RO データに 1 つ以上のアドレス型再配置を含んでいます。セクション RW を、実行時にダイナミックに再配置されるように生成しています。
L6772W : --sysv でのビルド中にインポート <symname> コマンドが無視されました。
L6774W : <objname>(<secname>)に長さが正しくないデバッグフレームエントリがあります。
L6775W : <objname>(<secname>)内に、このセクションにない CIE を使用する FDE があります。
L6776W : <objname>(<secname>)のデバッグフレーム内に実行セクションが記述されていません。
L6777W : <objname>(<secname>)内のデバッグフレームに <actual> 個の再配置があります(<expected> 個を期待)
L6778W : <objname>(<secname>) 内のデバッグフレームでは、64 ビット DWARF を使用します。
L6780W : <origvis> の可視性は、<impexp> を介してシンボル '<symname>' から削除されました。
L6781E : <objname>(<secname>)内の再配置番号 <rel_class>:<rel_number>(<rtype>、wrt シンボル <symname>)のパーティション番号 <part> では値(<val>)を表せません。
L6782W : 再配置 #<rel_class>:<relnum><oepname> の再配置番号 <rel_class>:<relnum> '<rtype>' では、<pltgot_type> PLT エントリと共に正しくデータにアクセスできない可能性があります。

L6783E : <oepname> (<secnum>:<secname>)のマッピングシンボル番号 <symnum> '<msym>' が、セクションサイズ(シンボルオフセット = 0x<moffset>、セクションサイズ = 0x<moffset>、セクションサイズ = 0x<secsize>)の末尾または末尾以降で定義されています。

このメッセージは、セクションのアドレスが ELF セクションの末尾またはその外の位置を指していることを示しています。この問題は、インライン展開された空のデータセクションによって発生する場合があり、オブジェクトファイルに問題がある可能性を示しています。--diag_warning 6783 を使って、このエラーを非表示にすることもできます。

L6784E : 値 <value> を持つ <oepname> (<secnum>:<secname>)のシンボル番号 <symnum> '<symname>' が、サイズ 0x<size> であり、セクションの範囲を超えています。

リンカが、セクションの外まで拡がるサイズのシンボルを検出しました。このメッセージは、RVCT 2.2 ビルド 503 以降のツールチェーンでは、デフォルトでただの警告です。--diag_warning 6784 を使って、このエラーを非表示にできます。

L6785U : '<libname>' からのインポート用にマークされたシンボル '<symname>' は既に '<oepname>' によって定義されています。

L6786W : <oepname> (<secnum>:<secname>)のマッピングシンボル番号 <symnum> '<msym>' が非境界整列オフセット = 0x<moffset> で定義されています

L6787U : エントリが <regionname> に対して必要とする領域テーブルハンドラ '<handlername>' が見つかりませんでした。

L6788E : 実行範囲 <er1name> を [<base1>,<limit1>) にスキャッタローディングすると、[<base2>,<limit2>) にある実行領域 <er2name> の内容が実行時に壊れます。

この問題は、スキャッタローディングが行われ、実行領域が他の実行領域(それ自体または別の領域)を部分的または全体的に上書きしてしまう場所に配置される場合に発生します。

例えば、次のコードはこのエラーを生成します。

```
LOAD_ROM 0x0000 0x4000 {  EXEC1 0x4000 0x4000  {  * (+RW,+ZI)  }  EXEC2 0x0000
0x4000  {  * (+RO)  } }
```

そして、次のメッセージが表示されます。

```
エラー:L6788E:実行範囲 EXEC2 のスキャッタローディングを行うと、実行時に実行範囲 EXEC2 のコンテンツが壊れる可能性があります。
```

次のコードからはエラーは生成されません。

```
LOAD_ROM 0x0000 0x4000 {  EXEC1 0x0000 0x4000  {  * (+RO)  }  EXEC2 0x4000 0x4000
{  * (+RW,+ZI)  } }
```

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

スキャッタファイルに関する情報。

L6789U : ライブラリ <library> メンバ <filename>:エンディアンが一致しません。

L6790E : 再配置 #<rel_class>:<relnum>(<objname>(<secname>)内)は <symname> に関係しています。GOT 生成配置を介して、ウィーク参照をインポートできない可能性があります

L6791E : 0x<offset> <oepname> (<secname>)に不明なパーソナリティルーチン <pr> があります。

L6792E : オフセット 0x<offset> <oepname> (<secname>)の記述子です。

L6793E : クリーンアップ記述子 <oepname> (<secname>)のオフセット 0x<offset> で、ランディングパッド参照が必要です。

L6794E : キャッチ記述子 <oepname> (<secname>)のオフセット 0x<offset> で、ランディングパッド参照が必要です。

L6795E : キャッチ記述子 <oepname> (<secname>)のオフセット 0x<offset> で、RTTI 参照が必要です。

L6796E : オフセット 0x<offset> <oepname> (<secname>)に位置する記述子がセクションの終了位置を超えました。

L6797E : 例外テーブル <oepname> (<secname>)内のオフセット 0x<offset> に位置するデータがセクションの終了位置を超えました

L6798E : 関数指定記述子 <oepname> (<secname>)のオフセット 0x<offset> で、RTTI 参照が必要です。
L6799E : 関数指定記述子 <oepname> (<secname>)のオフセット 0x<offset> で、ランディングパッド参照が必要です。

ランディングパッドは、例外が発生した後でクリーンアップを行うコードです。リンカが古い形式の例外テーブルを検出した場合は、これが自動的に新しい形式に変換します。

このメッセージは、新しいバージョンのリンカと古いバージョンのコンパイラを使用していない限り、表示されません。

L6800W : 0x<offset> <oepname> (<secname>)で、一般モデルパーソナリティルーティンを変換できません。

パーソナリティルーティンは、例外処理スタックをアンワインドします。リンカが古い形式の例外テーブルを検出した場合は、これが自動的に新しい形式に変換します。このメッセージは、コンパイラでのエラーを示しています。購入元にお問い合わせ下さい。

L6801E : <secarmthumb> コードを含む <objname>(<secname>)は、'~IW(全コードに対しては意図されていないユーザはインターワークする必要があります)' <funarmthumb> 関数 <sym> のアドレスを使用できません。

リンカは、非インターワーキング(~IW)関数が他の状態のコードによってアドレスを取得される個所を診断できます。このエラーはデフォルトでは無効になっていますが、--strict を使ってリンクすることで有効にできます。このエラーは --diag_warning 6801 を使って警告に降格することができます。後で、必要であれば、--diag_suppress 6801 を使って完全に非表示にすることもできます。

例えば、次のように、a.c 内のコードが t.c 内の非インターワーキング関数のアドレスを使用したとします。

```
armcc -c a.c
armcc --thumb -c t.c
armlink t.o a.o --strict
```

次のメッセージが表示されます。

```
エラー:L6801E:ARM コードを含む a.o(.text)には、'~IW' Thumb 関数 foo のアドレスを使用できません。
```

L6802E : 再配置 #<rel_class>:<idx>(<objname>(<secname>)内)は <armsym> に関係しています。<armobjname>(<armsecname>)での非 Thumb シンボルに対する Thumb 分岐。
L6803W : 再配置 #<rel_class>:<idx>(<objname>(<secname>)内)は <armsym> に関係しています。Thumb 分岐が <armobjname>(<armsym>)にあるターゲットに到達することはほとんどありません。
L6804W : 古いシンボル型 STT_TFUNC の使用が検出されました
L6805E : 再配置 #<rel_class>:<idx>(<objname>(<secname>)内)は <armsym> に関係しています。<armobjname> 内の型なし絶対シンボルへの分岐です。ターゲットの状態が不明です。
L6806W : 再配置 #<rel_class>:<idx>(<objname>(<secname>)内)は <othersym> に関係しています。<otherobjname>(<othersecname>)内の型なしシンボルへの分岐です。ABI では、外部コードシンボルが STT_FUNC 型である必要があります。
L6807E : 再配置 #<rel_class>:<idx>(<objname>(<secname>)内)は <othersym> に関係しています。同じセクションの型なしシンボルへの分岐です。状態の変更が必要です。
L6809W : 再配置 <rel_class>:<idx>(<objname>(<secname>)内)は、廃止された型 <rtype> です。ABI 準拠の型については ARMELF を参照して下さい。

L6810E : 再配置 <rel_class>:<idx>(<objname>(<secname>)内)は、廃止された型 <rtype> です。
多くの場合、再配置のエラーと警告は、前のバージョンの ARM ツールでビルドされたオブジェクトファイルをリンクしている場合に発生します。

再配置のエラーと警告を表示するには、--strict_relocations スイッチを使用して下さい。このオプションを使用すると、オブジェクトの ABI 準拠を確認できます。デフォルトではオフになっており、廃止されたり廃止予定だったりする再配置は、リンカによって自動的に処理されます。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

--strict_relocations、*--no_strict_relocations*。

L6812U : 不明なシンボルアクション型です。購入元にお問い合わせください。

L6813U : <newname> に名前変更するためのシンボル <symname> が見つかりませんでした。
*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

RENAME ステアリングファイルコマンド。

L6815U : メモリが不足しています。割り当てサイズ:<alloc_size>システムサイズ:<system_size>
このエラーは、ARM コンパイラ v4.1 以降で報告されます。ここでは、利用可能なメモリの量と、リンク手順を実行する必要があるメモリの量についての情報が提供されます。

このエラーが発生する原因として、リンカに、ターゲットオブジェクトにリンクするための十分なメモリがないことが考えられます。これは一般的ではありませんが、以下のような多くの理由でトリガされる可能性があります。

- 非常に大きなオブジェクトやライブラリにリンクしています。
- 大量のデバッグ情報を生成しています。
- スキャッタファイル内に定義された大領域があります。

このような場合、ワークステーションは仮想メモリ不足に陥る可能性があります。

この問題は、FIXED スキャッタロード属性を使用した場合にも起こります。FIXED 属性は、実行領域を固定アドレスで ROM 内のルート領域にします。ROM イメージを生成するには、リンカは、前の実行領域の末尾と FIXED 領域の間にパディングバイトを追加する必要があります。FIXED 領域のアドレスが実行領域の末尾から遠いときに大量のパディングを追加すると、リンカはメモリ不足に陥る可能性があります。間隔を短くすると、リンク手順は成功します。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

- *実行領域の属性*。
- *ルート実行領域と FIXED 属性*。

リンカはほとんどすべてのサイズのイメージを生成することができますが、リンクの実行と終了には大量のメモリが必要になります。リンク時のパフォーマンスを向上させ、メモリ不足エラーを回避するために、以下のソリューションを試してみて下さい。

1. リンク時に、すべての不要なアプリケーションとプロセスをシャットダウンして下さい。

   例えば、Eclipse のもとで実行している場合は、コマンドラインからリンカを実行するか、ビルド間の Eclipse を終了して再起動してみて下さい。
2. --no_debug リンカオプションを使用します。

   このコマンドは、デバッグ情報を含めずにオブジェクトを作成するようにリンカに指示します。

   *『armlink ユーザガイド』*の以下のセクションを参照して下さい。

   --debug、*--no_debug*。

   ———— 注 ————

   このオプションを使用すると、ソースレベルのデバッグを実行することはできません。

   ————————————
3. デバッグ情報を削減します。

   --no_debug オプションを使用しない場合は、デバッグ情報を削減するための別の方法があります。

   fromelf ユーティリティを使用して、デバッグ情報をデバッグする必要がないオブジェクトとライブラリから削除することもできます。

   *『fromelf ユーザガイド』*の以下のセクションを参照して下さい。

   --strip=option[,option,...]。
4. 部分リンクを使用します。

   部分リンクを使用すると、リンクステージを少数の小さな演算に分割することができます。分割すると、最終的なリンクでメモリ内のオブジェクトファイルが重複しないようにすることもできます。

   *『armlink ユーザガイド』*の以下のセクションを参照して下さい。

--partial。

5. Windows オペレーティングシステム上でメモリのサポートを増やします。

   一部の Windows オペレーティングシステムでは、仮想アドレス空間を 2GB（デフォルト）から 3GB に増やすことができます。

   詳細については、以下の Microsoft の記事を参照してください。

   メモリサポートと Windows オペレーティングシステム
6. `--no_eager_load_debug` リンカオプションを使用します。

   このオプションは、RVCT 4.0 ビルド 697 以降で使用できます。このオプションを使用すると、リンカはオブジェクトのロード後にメモリからデバッグセクションデータを削除します。これにより、最終イメージを書き込む際にデバッグデータのほとんどを再度ロードする必要が生じるため、リンカのパフォーマンスは低下しますが、リンカのピーク時のメモリ使用量が抑えられます。

   *『armlink ユーザガイド』*の以下のセクションを参照して下さい。

   --eager_load_debug、*--no_eager_load_debug*。

まだ同じ問題が起こる場合は、サポートケースを作成して下さい。

`L6828E` : 再配置 `#<rel_class>:<idx><symname>` に関する再配置 `#<rel_class>:<idx>(<objname>(<secname>)内)`。分岐のソースアドレス `<srcaddr>` は、次の使用可能な `[<pool_base>,<pool_limit>)` のプールに到達できません。`--veneer_pool_size` オプションを使用してコンティンジェンシを増やしてください。

`--veneer_inject_type=pool` ベニア生成モデルでは、プール内のベニアへの分岐が、ベニアに設定できる最大アドレスであるプール制限に届くことが要求されています。後で、分岐がプール制限に到達できないことがわかった場合、および `armlink` がプール制限を下げてもすべてのベニアを格納できる場合は、`armlink` によって分岐に合わせてプール制限が下げられます。エラーメッセージ L6828 は、`armlink` がプール制限を下げられない場合のみ発生します。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

--veneer_inject_type=type。

`L6898E` : 再配置 `#<rel_class>:<idx>(<objname>(<secname>)内)` は `<armsym>` に関係しています。`<armobjname>(<armsecname>)`での非 `ARM/Thumb` シンボルに対する `ARM` 分岐。
`L6899E` : 既存の `SYMDEFS` ファイル `'<filename>'` は、読み出し専用です。
`L6900E` : `AND` および `OR` 演算子の優先順位を特定するために、括弧が必要です。
`L6901E` : シンボル名が必要です。
`L6902E` : ストリングが必要です。
`L6903E` : スクリプトの `'<clause>'` 節内で `'<text>'` を実行できません。
`L6904E` : 名前変更後のシンボルが、別の名前変更処理と競合します。
`L6905E` : 名前変更前のシンボルが、別の名前変更処理と競合します。
`L6906E` : （これは、名前変更の競合相手です。）
`L6907E` : 式が必要です。
`L6910E` : フェーズ名が必要です。
`L6912W` : インデックス `<idx>(<oepname>(<secname>)内)`にあるシンボル `<symname>` には、`ABI` シンボル型 `<symtype>` がありますが、これはマッピングシンボル型 `<maptype>` と矛盾しています。
`L6913E` : 実行領域名が必要です。
`L6914W` : `--<memoption>` の使用時には、オプション `<spurious>` は無視されます。

L6915E : ライブラリからエラーがレポートされています:<msg>
メッセージは通常、以下のいずれかです。

- エラー:L6915E:ライブラリからエラーがレポートされています:スキャッタロードファイルがヒープまたはスタック領域を宣言しておらず、__user_initial_stackheap が定義されていません。

  または

  エラー:L6915E:ライブラリからエラーがレポートされています:スキャッタロードの使用中には、使用可能なヒープ領域をセミホスティング __user_initial_stackheap で確実に設定することはできません

  多くの場合、__user_setup_stackheap() が再実装されていなかったり、ARM_LIB_STACK または ARM_LIB_HEAP 領域がそれぞれのスキャッタファイルで定義されていなかったりします。

  ———— 注 ————

  __user_setup_stackheap() は、非推奨の関数 __user_initial_stackheap() よりも優先されます。

  ————————

  『*ARM C および C++ ライブラリと浮動小数点サポートユーザガイド*』の以下のセクションを参照して下さい。
  — *__user_setup_stackheap()*。
  — *古い関数 __user_initial_stackheap()*。

  『*armlink ユーザガイド*』の以下のセクションを参照して下さい。

  空き領域の予約。
- エラー:L6915E:ライブラリからエラーがレポートされています:__use_no_semihosting が要求されていましたが、<function> が参照されました。

  <function> は __user_initial_stackheap、_sys_exit, _sys_open、_sys_tmpnam、_ttywrch, system、remove、rename、_sys_command_string、time、または clock を表します。

  このエラーは、SVC または BKPT 命令が C ライブラリからリンクされるのを避けるために、セミホスティング使用関数をターゲット変更した場合に表示されることがあります。

  次のコードを使用して、セミホスティング使用関数が C ライブラリからリンクされないようにして下さい。

  ```
  #pragma import(__use_no_semihosting)
  ```

  『*ARM C および C++ ライブラリと浮動小数点サポートユーザガイド*』の以下のセクションを参照して下さい。

  非セミホスティング環境でのライブラリの使用。

  セミホスティング使用関数がまだリンクされている場合は、リンカによってこのエラーが報告されます。

  この問題を解決するには、これらの C ライブラリ関数の独自の実装を用意する必要があります。

  emb_sw_dev ディレクトリには、一般的ないくつかのセミホスティング使用関数を再実装する方法の例が保存されています。ファイル retarget.c を参照して下さい。

  C ライブラリ関数を使用したセミホスティングの詳細については、以下の

  ARM C ライブラリ、C++ ライブラリ、および浮動小数点サポートユーザガイド。

——— 注 ———

リンカは、ユーザ独自のアプリケーションコード内の、例えば __semihost() というセミホスティング使用関数については、報告しません。

———————

どのセミホスティング使用関数がまだ C ライブラリからリンクされているかを特定するには、次のようにします。

1. armlink --cpu=8-A.32 --verbose --list err.txt でリンクします。
2. err.txt で __Iusesemihosting の出現個所を探します。

以下に例を示します。

```
...Loading member sys_exit.o from c_4.l. reference :__I$use$semihosting
definition:_sys_exit ...
```

これは、セミホスティング使用関数 _sys_exit が C ライブラリからリンクされていることを示しています。この問題を避けるには、この関数の独自の実装を用意する必要があります。

- エラー:L6915E:ライブラリからエラーがレポートされています:__use_no_heap が要求されていましたが <reason> が参照されました

<reason> が malloc、free、__heapstats、または __heapvalid を表している場合、__use_no_heap を使用すると、これらの関数と競合します。

- エラー:L6915E:ライブラリからエラーがレポートされています:__use_no_heap_region が要求されていましたが <reason> が参照されました

<reason> が malloc、free、__heapstats、__heapvalid、または __argv_alloc を表している場合、__use_no_heap_region を使用すると、これらの関数と競合します。

L6916E : 再配置 #<rel_class>:<idx>(<oepname>(<spname>)内)。条件付き BL 命令に対する R_ARM_CALL です。
L6917E : 再配置 #<rel_class>:<idx>(<oepname>(<spname>)内)。BLX 命令に対する R_ARM_JUMP24 です。
L6918W : <oepname> (<spname>)のアライメント <spalign> を行うには、0x<eraddr> の実行領域 <ername> に対するパディングが必要です。
L6922E : セクション <objname>(<secname>)に相互に排他的な属性(READONLY と TLS)が含まれています
L6923E : 再配置 #<rel_class>:<idx>(<oepname>(<spname>)内)は <symname> に関係しています。<symobjname>(<symsecname>)での非 TLS シンボルへの TLS 再配置 <type>。
L6924E : 再配置 #<rel_class>:<idx>(<oepname>(<spname>)内)は <symname> に関係しています。<symobjname>(<symsecname>)での STT_TLS シンボルへの非 TLS 再配置 <type>。
L6925E : 領域 <region> の <token> 属性を無視します。MemAccess はサポートされません。
L6926E : <oepname> (<spname>)の再配置番号 <rel_class>:<idx> が、命令エンコーディング 0x<bl> に対して不正な再配置型 <rtype> です。
L6927E : <oepname> (<spname>)の再配置番号 <rel_class>:<idx> が、命令エンコーディング 0x<bl1><bl2> に対して不正な再配置型 <rtype> です。
L6932W : ライブラリから警告がレポートされています:<msg>
L6935E : オブジェクト(<new>)と(<old>)のデバッググループの内容が互いに異なります(<name>、署名シンボル <sig>)
L6936E : シンボル '<sym>' のライブラリスクリプトの複数 RESOLVE 節。
L6937E : ライブラリスクリプト関数 '<func>' の複数定義。
L6939E : 領域 <regname> のアライメントがありません。
L6940E : 領域 <regname> のアライメント <alignment> は、2 の累乗(4 以上)か、最大値である必要があります。
L6941W : ファイル <filename> の chmod システムコールで、エラー <perr> が発生しました
L6942E : 実行領域 <ername> に複数の <type> が含まれています。セクション:
L6966E : 領域 <regname> のアライメント <alignment> は負にできません。
L6967E : エントリポイント(<address>)は THUMB 命令を指していますが、有効な THUMB コードポインタではありません。

L6968E : Linux カーネルバージョン \"<kernel>\" を解析できませんでした。
L6969W : <ername> で AT セクション <name> のタイプを RW から RO に変更しています。
L6971E : <objname>(<secname>)の <type> 型は、er <ername> の <prevobj>(<prevname>)の <prevtype> 型と互換性がありません。

スキャッタファイルで __at セクションを配置すると、このメッセージが表示されることがあります。例えば、main.c の次のコードと関連のスキャッタファイルにより、このエラーが生成されます。

```
int variable __attribute__((at(0x200000)));
```

```
LR1 0x0000 0x20000 {     ER1 0x0 0x2000     {         *(+RO)     }     ER2 0x8000
0x2000     {         main.o     }     RAM 0x200000 (0x1FF00-0x2000)
{         *(+RW, +ZI)     } }
```

変数の型は ZI です。リンカはこれをアドレスに配置しようとします。0x200000 .ところが、このアドレスはスキャッタファイルにより RW セクション用に予約されています。そのため、エラーが生成されます。

エラー: L6971E:RW 型の stdio_streams.o(.data) は er RAM の ZI 型の main.o(.ARM.__AT_0x00200000) と互換性がありません。

この問題に対処するには、ソースコードでアドレスを次のように変更します。

```
int variable __attribute__((at(0x210000)));
```

『armlink ユーザガイド』の以下のセクションを参照して下さい。

- *特定アドレスへの関数とデータの配置方法*。
- *__at を使用した特定のアドレスへのセクションの配置*。

L6972E : 必要なベース 0x<required> を持つ <objname> (<secname>)がベースアドレス 0x<actual> に割り当てられました。
L6973E : オーバーレイ ER <ername> のアドレス 0x<addr> での AT セクションの配置中にエラーが発生しました。

例えば、DLL またはアプリケーションをオーバーレイ領域を使用してビルドするときに、セクションを配置するために __attribute__((at(*address*))) を使用しようと試みました。__attribute__((at(*address*))) では、.ARM.__at_*address* を使用してスキャッタファイルに固定の位置を指定する必要があります。この場合、--no_autoat リンカオプションも指定する必要があります。
『armlink ユーザガイド』の以下のセクションを参照して下さい。

- *__at を使用した特定のアドレスへのセクションの配置*。
- *--autoat、--no_autoat*。

L6974E : AT セクション <name> にベースアドレスがありません。
『armlink ユーザガイド』の以下のセクションを参照して下さい。

__at を使用した特定のアドレスへのセクションの配置。

L6975E : <objname>(<secname>)は必要なベースと SHF_MERGE を持つことはできません。
L6976E : <objname>(<secname>)は必要なベースと SHF_LINK_ORDER を持つことはできません。
L6977E : <objname>(<secname>)は複数セクションの連続ブロックになることはできません
L6978W : <objname>(<secname>)にはユーザ定義セクション型と必要なベースアドレスがあります。
L6979E : 必要なベースアドレスを持つ <objname>(<secname>)を位置非依存の ER <ername> に配置できません。
L6980W : FIRST および LAST が、必要なベースアドレスを持つ <objname>(<secname>)で無視されました。

『armlink ユーザガイド』の以下のセクションを参照して下さい。

FIRST 属性と LAST 属性を使用したセクションの配置。

L6981E : __AT は BPABI と SystemV のイメージタイプと互換性がありません
『armlink ユーザガイド』の以下のセクションを参照して下さい。

__at セクションの配置に関する制限。

L6982E : ベース <base>、リミット <limit> の AT セクション <objname>(<spname>)が、ベース <base2>、リミット <limit2> の AT セクション <obj2name>(<sp2name>)のアドレス領域と重複しています。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

__at を使用した特定のアドレスへのセクションの配置。

L6983E : 必要なベースアドレス <base> を持つ AT セクション <objname>(<spname>)が、ベース <erbase>、リミット <erlimit> の ER <ername> の範囲外です。

この問題は、コードに __attribute__((at(*address*))) を、スキャッタファイルに .ARM.__at_*address* を、リンカコマンドラインに --no_autoat オプションを指定する場合に発生することがあります。この場合、.ARM.__at_*address* のアドレス部分は 8 桁の 16 進数として指定される必要があります。以下に例を示します。

```
int x1 __attribute__((at(0x4000)));  // function.c で定義 ; スキャッタファイル LR1 0x0
{     ...          function.o(.ARM.__at_0x00004000)     ...}
```

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

- *__at を使用した特定のアドレスへのセクションの配置*。
- *--autoat、--no_autoat*。

L6984E : AT セクション <objname>(<spname>)が、セクションアライメント <alignment> に整列していないベースアドレス <base> を要求しました。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

__at を使用した特定のアドレスへのセクションの配置。

L6985E : 必要なベースアドレス <base> を持つ AT セクション <objname>(<spname>)を自動的に配置できません。--no_autoat オプションを使用して手動でスキャッタファイルに配置してください。

*『armlink ユーザガイド』*の以下のセクションを参照して下さい。

- *__at を使用した特定のアドレスへのセクションの配置*。
- *--autoat、--no_autoat*。

L9511E : 製品定義ファイルが見つかりませんでした

armlink で、必要な製品ライセンスのマッピング(.elmap)ファイルが見つかりません。これは、以下が原因である可能性があります。

- .elmap ファイルが欠落している。例えば、インストールが破損している場合などです。
- ARM_PRODUCT_PATH 環境変数が間違って設定されているため、armlink が間違った場所で .elmap ファイルを探している。
- ARM_TOOL_VARIANT 環境変数が間違って設定されているため、armlink が存在しない .elmap ファイルを探している。

第 3 章
ELF イメージ変換ユーティリティのエラーおよび警告

ELF イメージ変換ユーティリティ（`fromelf`）のエラーおよび警告メッセージについて説明します。

以下のセクションから構成されています。

- *3.1 fromelf のエラーおよび警告メッセージのリスト*(3-80 ページ).

3.1 fromelf のエラーおよび警告メッセージのリスト

fromelf で生成されるエラーおよび警告メッセージをリストします。

Q0105E : ロード領域 #<segindex> が先頭のアドレス空間の範囲を越えています。
Q0106E : メモリが不足しています。
Q0107E : 出力ファイル '<filename>' の書き込みに失敗しました:<reason>
Q0108E : 出力ファイル '<filename>' を作成できませんでした:<reason>
Q0119E : 出力ファイルが指定されていません。
Q0120E : 入力ファイルが指定されていません。
Q0122E : ファイル '<filename>' を開けませんでした:<reason>
<reason> が無効な引数の場合、コマンドラインに無効な文字が含まれることが原因である可能性があります。

例えば、Windows の場合、アーカイブファイルと共にフィルタを指定するときにエスケープ文字 \ を使用した場合などです。

```
fromelf --elf --strip=all t.a\(test*.o\) -o filtered/
```

Windows の場合、以下のように指定します。

```
fromelf --elf --strip=all t.a(test*.o) -o filtered/
```

*『fromelf ユーザガイド』*の以下のセクションを参照して下さい。

input_file。

Q0128E : ファイル入出力エラーです。
このエラーは、ディレクトリの末尾がパスの区切り文字ではない場合に、--output コマンドラインオプションでディレクトリを指定したときに発生する可能性があります(--output=my_elf_files/ など)。

*『fromelf ユーザガイド』*の以下のセクションを参照して下さい。

--output=destination。

Q0129E : 32 ビット ELF ファイルではありません。
Q0130E : 64 ビット ELF ファイルではありません。
Q0131E : 無効な ELF 識別番号が見つかりました。
ELF 形式でないか破損しているファイルに対して fromelf を使おうとすると、このエラーが発生します。オブジェクト(.o)ファイルおよび実行可能(.axf)ファイルは、ELF 形式です。
Q0132E : 無効な ELF セクションインデクス <idx> が見つかりました。
Q0133E : 無効な ELF セグメントインデクス <idx> が見つかりました。
Q0134E : 無効な ELF ストリングテーブル <idx> が見つかりました。
Q0135E : 無効な ELF セクションエントリサイズが見つかりました。
Q0136E : ELF ヘッダに無効なファイルタイプが含まれています。
Q0137E : ELF ヘッダに無効なコンピュータ名が含まれています。
Q0138E : ELF ヘッダに無効なバージョン番号が含まれています。
Q0131E を参照して下さい。
Q0147E : ディレクトリ <dir> の作成に失敗しました:<reason>
<reason> が既存のファイルである場合、既存のファイルと同じ名前を持つディレクトリを指定したことが原因である可能性があります。例えば、filtered という名前のファイルが既に存在している場合、次のコマンドを実行するとこのエラーが生成されます。

```
fromelf --elf --strip=all t.a(test*.o) -o filtered/
```

パスの区切り文字 / により、fromelf は filtered がディレクトリであると判断します。

*『fromelf ユーザガイド』*の以下のセクションを参照して下さい。

--output=destination。

Q0171E : ストリングテーブル <idx> への st_name インデックスが無効です。
Q0131E を参照して下さい。
Q0172E : シンボルテーブル <idx> へのインデックスが無効です。
Q0131E を参照して下さい。
Q0186E : このオプションでは、デバッグ情報が存在している必要があります
--fieldoffsets オプションでは、イメージが dwarf デバッグテーブル付きでビルドされている必要があります。
Q0425W : ファイル内に正しい形式でない仮想関数の削除ヘッダがあります
この問題は、コンパイラのエラーを示している場合があります。購入元にお問い合わせ下さい。
Q0426E : ファイルからの vtable 情報の読み出し中にエラーが発生しました
この問題は、コンパイラのエラーを示している場合があります。購入元にお問い合わせ下さい。
Q0427E : vtable 内のシンボルのストリングを取得中にエラーが発生しました
この問題は、コンパイラのエラーを示している場合があります。購入元にお問い合わせ下さい。
Q0433E : 診断スタイル <style> は認識されません
Q0440E : <secname> の再配置セクションがありません
Q0447W : 不明な診断番号(<num>)
Q0448W : <file> のセクション番号 <secnum> '<secname>' の展開中に、圧縮データの末尾以降を読み出そうとしています
この問題は、内部エラーを示している場合があります。購入元にお問い合わせ下さい。
Q0449W : <file> のセクション番号 <secnum> '<secname>' の展開中に、サイズ <bufsize> の展開データバッファの末尾以降に書き込もうとしています
この問題は、内部エラーを示している場合があります。購入元にお問い合わせ下さい。
Q0450W : ファイル <file> のセクション番号 <secnum> '<secname>' に、ABI 配置の以前の型と現在の型が混在しています。
Q0451W : オプション '--strip symbols' が、デバッグ情報を持つ ELF ファイルに対して '--strip debug' を指定せずに使用されました。
Q0452W : オプション '--strip filesymbols' が、デバッグ情報を持つ ELF ファイルに対して '--strip debug' を指定せずに使用されました。
Q0453W : '<path1>' および '<path2>' からパス名を取り除くと重複したファイル名 '<filename>' になります。
Q0454E : ELF ファイル内: <details>
Q9511E : 製品定義ファイルが見つかりませんでした
fromelf で、必要な製品ライセンスのマッピング(.elmap)ファイルが見つかりません。
これは、以下が原因である可能性があります。

- .elmap ファイルが欠落している。例えば、インストールが破損している場合などです。
- ARM_PRODUCT_PATH 環境変数が間違って設定されているため、fromelf が間違った場所で .elmap ファイルを探している。
- ARM_TOOL_VARIANT 環境変数が間違って設定されているため、fromelf が存在しない .elmap ファイルを探している。

第 4 章
ライブラリアンのエラーおよび警告

ARM ライブラリアン（armar）のエラーおよび警告メッセージについて説明します。

以下のセクションから構成されています。

- *4.1 armar のエラーおよび警告メッセージのリスト*(4-83 ページ).

4.1 armar のエラーおよび警告メッセージのリスト

armar で生成されるエラーおよび警告メッセージをリストします。

L6800U : メモリが不足しています
L6825E : アーカイブ '<archive>' の読み出し中:<reason>
L6826E : '<archive>' はアーカイブ形式になっていません
L6827E : '<archive>':不正な形式のシンボルテーブル
L6828E : '<archive>':不正な形式のストリングテーブル
L6829E : '<archive>':不正な形式のアーカイブ(オフセット <offset>)
L6830E : アーカイブ '<archive>' の書き込み中:<reason>
L6831E : '<member>' がアーカイブ '<archive>' 内に存在しません
L6832E : アーカイブ '<archive>' が見つかりません:<reason>
L6833E : ファイル '<filename>' が存在しません
L6835E : ファイル '<filename>' の読み取り中:<reason>
L6836E : '<filename>' は既に存在しているため、抽出されません
L6838E : アーカイブが指定されていません
L6839E : いずれかのアクション -[<actions>] を指定する必要があります
L6840E : 1 つのアクションオプションのみを指定できます
L6841E : 位置 '<position>' が見つかりません
L6842E : ファイル名 '<filename>' がファイルシステムに対して長すぎます
L6843E : ファイル '<filename>' の書き込み中:<reason>
L6874W : アーカイブメンバ '<member>' のマイナーバリアントにベースバリアントが含まれていません
同じ関数のマイナーバリアントがライブラリ内に存在します。2 つの同じオブジェクトを見つけて、一方を削除して下さい。
L6875W : ELF オブジェクトではないファイル '<filename>' をアーカイブ '<name>' に追加中
L9511E : 製品定義ファイルが見つかりませんでした
armar で、必要な製品ライセンスのマッピング(.elmap)ファイルが見つかりません。
これは、以下が原因である可能性があります。

- .elmap ファイルが欠落している。例えば、インストールが破損している場合などです。
- ARM_PRODUCT_PATH 環境変数が間違って設定されているため、armar が間違った場所で .elmap ファイルを探している。
- ARM_TOOL_VARIANT 環境変数が間違って設定されているため、armar が存在しない .elmap ファイルを探している。

第 5 章
その他のエラーおよび警告

ツールのいずれかによって表示されるその他のエラーおよび警告メッセージについて説明しています。

以下のセクションから構成されています。

5.1 内部エラーとその他の予期しないエラー

内部エラーはとは、ツールが内部の整合性チェックに失敗したか、処理できない、予想外の入力があったことを示します。ツール自体に問題がある可能性を示唆している場合もあります。

例えば、

```
内部エラー:[0x76fd03:600448]
```

これには、以下の情報が含まれています。

- メッセージの説明(内部エラー)。
- 発生したエラーを示す、6 桁の 16 進数のエラーコード(`0x76fd03`)。
- バージョン番号(60 は ARM コンパイラ 6.0 です)。
- ビルド番号(この例では 0448)。

内部エラーが表示された場合は、購入元にお問い合わせ下さい。

調査を容易にするために、エラーを発生させたソースファイルまたは関数のみを送るようにして下さい。また、使用したコマンドラインオプションもお知らせ下さい。

アセンブラによって内部エラーが発生し、ソースコードにプリプロセッサマクロが含まれている場合は、ソースを購入元に送信する前に、コンパイラでのソースの前処理が必要になることがあります。

5.2 その他のエラーおよび警告メッセージのリスト

ARM コンパイラツールチェーンのツールによって生成されるエラーおよび警告メッセージのリスト。

——— 注 ———

When the message is displayed, the *X* prefixing the message number is replaced by an appropriate letter relating to the tool. For example, the code `X3900U` is displayed as `L3900U` by the linker when you have specified an unrecognized option.

———————————

`X3900U` : 認識されないオプション `'<dashes><option>'` です。
`<option>` をツールが認識できません。この問題の原因は、スペルの間違い、またはオプションのサポートされていない省略形の使用である可能性があります。

`X3901U` : オプション `'<option>'` の引数がありません。

`X3902U` : 再帰的 `via` ファイルのインクルードの深さがファイル `'<file>'` 内で `<limit>` に達しました。

`X3903U` : 引数 `'<argument>'` は、オプション `'<option>'` には使用できません。
原因としては、不正な形式の整数や不明な引数などが考えられます。

`X3904U` : `via` ファイル `'<file>'` を開けませんでした。

`X3905U` : `via` ファイル `'<file>'` の読み出し中にエラーが発生しました。

`X3906U` : `via` ファイル `'<file>'` の形式が間違っています。

`X3907U` : `via` ファイル `'<file>'` コマンドはバッファには長すぎます。

`X3908U` : オーバーフロー:`'<string>'` は整数に収まりません。

`X3910W` : 古い構文です。`'<hyphens><option><separator><parameter>'` を使用して下さい。

`X3912W` : オプション `'<option>'` は廃止される予定です。

`X3913W` : `via` ファイル `'<file>'` を閉じることができませんでした。

`X3915W` : オプション `'<option>'` の引数 `'<argument>'` は廃止される予定です

`X3916U` : オプション `'<dashes><option>'` に、期待されない引数が指定されています

`X3917U` : 連結されたオプションには引数を指定できません:`-<option> <arg>`

`X9905E` : 浮動小数点ハードウェアでない場合、`--apcs=/hardfp` は使用できません

`X9906E` : `fpu <fpu_option>` では、`--apcs=/hardfp` は使用できません

`X9907E` : 浮動小数点のサポートなしは選択できません

`X9908E` : `--fpmode=none` は `--fpu choice` をオーバーライドします