お客様各位

ZUD－F35－10－0208
2011 年 03 月 31 日

ルネサス エレクトロニクス株式会社
MCU 事業本部
ソフトウェア統括部
MCU ツール技術部

# RL78 マイクロコントローラ
# フラッシュ・セルフ・プログラミング・ライブラリ
# Type01 Ver.2.00
# 使用上の留意点
# (暫定)

## ご使用前に必ずお読みください

# 【 目 次 】



この添付資料では，本製品のバージョンによる差分などについて記載しています。ご使用の前に必ずお読みください。

# 1 制限事項

フラッシュ・セルフ・プログラミング・ライブラリのご使用時において以下の制限があります。

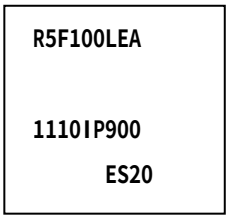
<table>
<tr><th>No.</th><th>内容</th></tr>
<tr><td>1</td><td>・割り込みベクタ変更処理の制限について<br><br>【制限内容】<br>下記に示す「表 1．割り込みベクタ変更機能 - 制限事項対応表」の「制限該当サンプル」一覧に記載のある RL78 マイクロコントローラでは，フラッシュ・セルフ・プログラミング・ライブラリの制限により，割り込みベクタ変更処理を正常に動作させる事が出来ません。従いまして，フラッシュ・セルフ・プログラミング・ライブラリで提供している以下のフラッシュ関数を使用しないで下さい。<br><br>・FSL_ChangeInterruptTable<br>・FSL_RestoreInterruptTable<br><br>表 1．割り込みベクタ変更処理 - 制限事項対応表
<table>
<tr><th>RL78/G13 のピン数：型名</th><th>フラッシュ・メモリのサイズ</th><th>制限該当サンプル(旧バージョン)</th><th>非該当サンプル(新バージョン)</th></tr>
<tr><td>20 ピン：R5F10x6<br>24 ピン：R5F10x7<br>25 ピン：R5F10x8</td><td>16KB～64KB</td><td>Ver.2.1 以前</td><td>Ver.3 以降</td></tr>
<tr><td rowspan="2">30 ピン：R5F10xA<br>32 ピン：R5F10xB<br>36 ピン：R5F10xC</td><td>16KB～64KB</td><td>Ver.2.1 以前</td><td>Ver.3 以降</td></tr>
<tr><td>96KB，128KB</td><td>Ver.1</td><td>Ver.2 以降</td></tr>
<tr><td rowspan="2">40 ピン：R5F10xE</td><td>16KB～64KB</td><td>Ver.2.1 以前</td><td>Ver.3 以降</td></tr>
<tr><td>96KB～192KB</td><td>Ver.1</td><td>Ver.2 以降</td></tr>
<tr><td rowspan="3">44 ピン：R5F10xF<br>48 ピン：R5F10xG<br>52 ピン：R5F10xJ<br>64 ピン：R5F10xL</td><td>16KB～64KB</td><td>Ver.2.1 以前</td><td>Ver.3 以降</td></tr>
<tr><td>96kB～256KB</td><td>Ver.1</td><td>Ver.2 以降</td></tr>
<tr><td>384KB～512KB</td><td>非該当</td><td>全バージョン</td></tr>
<tr><td rowspan="2">80 ピン：R5F10xM<br>100 ピン：R5F10xP</td><td>96kB～256KB</td><td>Ver.1</td><td>Ver.2 以降</td></tr>
<tr><td>384KB～512KB</td><td>非該当</td><td>全バージョン</td></tr>
<tr><td>128 ピン：R5F10xS</td><td>192KB～512KB</td><td>非該当</td><td>全バージョン</td></tr>
</table>
</td></tr>
</table>

**・サンプル・バージョンの判別方法**

サンプル・バージョンは捺印で判別できます。

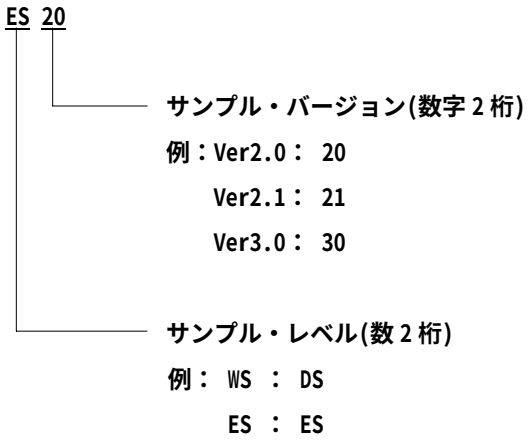
以下の例のようにサンプルには製品のサンプル・レベル，サンプル・バージョンが **4 文字**で記載されています。サンプル・レベル，サンプル・バージョンの判別用捺印がされていないものは CS 品または量産品となります。

例． R5F100LEAFB(64 ピン LQFP/64KB) でバージョン 2.0 の ES サンプルの場合




1 段目：製品型名
2 段目：製造ロット番号
3 段目：サンプル・レベル(ES) ，サンプル・バージョン(2.0)

サンプル・レベル，サンプル・バージョンの捺印構成




備考． WS はワーキング・サンプル，ES はエンジニアリング・サンプル，CS はコマーシャル・サンプルを表します。

# 2 アプリケーション・ノート

本バージョンは下記のアプリケーション・ノートに対応しています。

| マニュアル名 | 資料番号 |
|---|---|
| RL78 マイクロコントローラ<br>フラッシュ・セルフ・プログラミング・ライブラリ Type01 | R01AN0350JJ0100_RL78 |

# 3 動作環境

IBM PC/AT™互換機版（Windows®ベース）

OS ： WindowsXP，WindowsVista，Windows7

# 4 対応ツール

RL78 マイクロコントローラ フラッシュ・セルフ・プログラミング・ライブラリ Type01 Ver2.00 を使用する場合には，下記のバージョンを使用してください。

| ツール名 | バージョン |
|---|---|
| 統合開発環境：CubeSuite | E1.50 |

# 5 ソフトウェア・リソース

RL78 マイクロコントローラ用 フラッシュ・セルフ・プログラミング・ライブラリ Type01 Ver2.00 を使用する場合には，該当プログラムをユーザ・プログラム領域に配置する必要がありますので，そのコード・サイズ分のプログラム領域を消費します。

また，フラッシュ・セルフ・プログラミング・ライブラリを動作させるためには，スタック，データ・バッファ，ワーク・エリア(セルフ RAM 注)，を必要とします。

各デバイスで使用するソフトウェア・リソースを表 5-1～表 5-5 に示します。一覧の各項目に対する概要についてはアプリケーション・ノートの「2.2 ソフトウェア環境」をご参照ください。

表 5-1 フラッシュ・セルフ・プログラミング・ライブラリ Type01 ソフトウェア・リソース

| 項目 | 容量(バイト) | 配置制限・使用制限 | |
|---|---|---|---|
| セルフ RAM 注 | 0 ～ 1024 | RL78/G13 ： RAM 4KB、 ROM 64KB 製品 | FEF00H ～ FF2FFH |
| | | RL78/G13 ： RAM 20KB、ROM 256KB 製品<br>※128pin 製品は対象外 | FAF00H ～ FB2FFH |
| | | RL78/G13 ： RAM 32KB、ROM 512KB 製品 | F7F00H ～ F82FFH |
| | | RL78/G12 ： RAM 768B 以上の製品 | FF900H ～ FFC7FH |
| | | RL78/F12 ： RAM 4KB、 ROM 64KB 製品 | FEF00H ～ FF2FFH |
| | | 上記製品以外 | なし |
| スタック 表 5-2 | 0 ～ 46 | セルフ RAM，FFE20H-FFEFFH 以外の RAM 領域に展開可能 | |
| データ・バッファ 表 5-3 | 0 ～ 256 | | |
| ライブラリ・サイズ 表 5-4,表 5-5 | ROM:MAX 1245 | セルフ RAM，FFE20H-FFEFFH 以外のプログラム領域 | |
| | RAM:0 ～ 447 | セルフ RAM，FFE20H-FFEFFH，内蔵 ROM 以外のプログラム領域に展開可能 | |

注． フラッシュ・セルフ・プログラミング・ライブラリで固定的にワーク・エリアとして使用する領域を本書，及びアプリケーション・ノートではセルフ RAM と呼びます。
マッピングはされず，フラッシュ・セルフ・プログラミング実行時に自動的に使用される(以前のデータは破壊される)領域のため，ユーザ設定等は必要ありません。フラッシュ・セルフ・プログラミング・ライブラリを使用していない状態の場合は，通常の RAM 空間として使用できます。

表 5-2 フラッシュ関数のスタック・サイズ

| 関数名 | バイト | 関数名 | バイト |
|---|---|---|---|
| FSL_Init | 40 | FSL_GetBlockEndAddr | 36 |
| FSL_Open | 0 | FSL_GetFlashShieldWindow | 46 |
| FSL_Close | 0 | FSL_SwapBootCluster | 38 |
| FSL_PrepareFunctions | 10 | FSL_SwapActiveBootCluster | 42 |
| FSL_PrepareExtFunctions | 10 | FSL_InvertBootFlag | 42 |
| FSL_ChangeInterruptTable | 30 | FSL_SetBlockEraseProtectFlag | 42 |
| FSL_RestoreInterruptTable | 30 | FSL_SetWriteProtectFlag | 42 |
| FSL_BlankCheck | 42 | FSL_SetBootClusterProtectFlag | 42 |
| FSL_Erase | 42 | FSL_SetFlashShieldWindow | 42 |
| FSL_IVerify | 42 | FSL_StatusCheck | 30 |
| FSL_Write | 42 | FSL_StandBy | 30 |
| FSL_GetSecurityFlags | 46 | FSL_WakeUp | 42 |
| FSL_GetBootFlag | 46 | FSL_ForceReset | 0 |
| FSL_GetSwapState | 36 | FSL_GetVersionString | 0 |

表 5-3 フラッシュ関数のデータ・バッファ使用サイズ

| 関数名 | バイト | 関数名 | バイト |
|---|---|---|---|
| FSL_Init | 0 | FSL_GetBlockEndAddr | 4 |
| FSL_Open | 0 | FSL_GetFlashShieldWindow | 4 |
| FSL_Close | 0 | FSL_SwapBootCluster | 0 |
| FSL_PrepareFunctions | 0 | FSL_SwapActiveBootCluster | 0 |
| FSL_PrepareExtFunctions | 0 | FSL_InvertBootFlag | 0 |
| FSL_ChangeInterruptTable | 0 | FSL_SetBlockEraseProtectFlag | 0 |
| FSL_RestoreInterruptTable | 0 | FSL_SetWriteProtectFlag | 0 |
| FSL_BlankCheck | 0 | FSL_SetBootClusterProtectFlag | 0 |
| FSL_Erase | 0 | FSL_SetFlashShieldWindow | 4 |
| FSL_IVerify | 0 | FSL_StatusCheck | 0 |
| FSL_Write 注 | 4 ～ 256 | FSL_StandBy | 0 |
| FSL_GetSecurityFlags | 2 | FSL_WakeUp | 0 |
| FSL_GetBootFlag | 1 | FSL_ForceReset | 0 |
| FSL_GetSwapState | 1 | FSL_GetVersionString | 0 |

注． FSL_Write 関数は書き込みサイズ分(ワード単位)の容量が必要です。

例：2 ワード（1 ワード = 4 バイト）分の書き込み = 2 × 4 = 8 バイト

表 5-4 フラッシュ・セルフ・プログラミング・ライブラリのコード・サイズ 1 (全て ROM に配置する場合)

・　フラッシュ・セルフ・プログラミング・ライブラリを全て ROM に配置して使用する場合のコード・サイズです。RAM にコードを配置する必要はありませんが，使用方法に制限があり，一部の関数が使用できなくなります。詳細についてはアプリケーション・ノートを参照してください。

| コード・サイズの条件 | RAM 容量(バイト) | ROM 容量(バイト) |
|---|---|---|
| 全ての関数を登録した場合のコード・サイズ<br>※一部の関数は使用できません。 | 0 | 1245 |
| 以下の関数を使用した場合のコード・サイズ<br>・FSL_Init<br>・FSL_Open<br>・FSL_Close<br>・FSL_PrepareFunctions<br>・FSL_BlankCheck<br>・FSL_Erase<br>・FSL_IVerify<br>・FSL_Write<br>・FSL_StatusCheck | 0 | 498 |

表 5-5 フラッシュ・セルフ・プログラミング・ライブラリのコード・サイズ 2 (BGO を使用する場合)

・　フラッシュ・セルフ・プログラミング中に BGO(バック・グラウンド・オペレーション)機能を使用する場合のコード・サイズです。BGO 機能を使用する場合は FSL_RCD セグメントを RAM に配置する必要があります。また，FSL_RCD セグメントを RAM に展開するためにはプログラムの ROM 化処理を行う必要があるため，別途 ROM 化用として FSL_RCD セグメント分の ROM 容量が必要となります。

| コード・サイズの条件 | RAM 容量(バイト) | ROM 容量(バイト) |
|---|---|---|
| 全ての関数を使用した場合のコード・サイズ | 447 (FSL_RCD) | 798<br>+<br>ROM 化が必要な容量 (447) |
| 以下の関数を使用した場合のコード・サイズ<br>・FSL_Init<br>・FSL_Open<br>・FSL_Close<br>・FSL_PrepareFunctions<br>・FSL_BlankCheck<br>・FSL_Erase<br>・FSL_IVerify<br>・FSL_Write<br>・FSL_StatusCheck | 66 (FSL_RCD) | 432<br>+<br>ROM 化が必要な容量 (66) |

備考．本表はフラッシュ・セルフ・プログラミング・ライブラリ分のコード・サイズのみについて記載しています。BGO を行う場合はユーザ・プログラムも RAM 上に配置する必要があるため，別途ユーザ・プログラム分のコード・サイズの容量も必要となります。また，ROM 化したプログラムを RAM に展開するプログラム分のコード・サイズも必要になります。ROM 化機能の詳細については使用する予定の開発ツール等のユーザーズ・マニュアルを参照してください。

# 6 フラッシュ・セルフ・プログラミング中の処理時間

この節ではフラッシュ・セルフ・プログラミング・ライブラリの処理時間について記載します。

## 6.1 フラッシュ関数処理時間

フラッシュ関数がユーザ・プログラムから実行された後，処理を終了してユーザ・プログラムに戻るまでの時間です。ステータス・チェックのモード設定によって，時間に含まれる処理の内容が異なります。

### 6.1.1 ステータス・チェック・ユーザ・モード

このモードのフラッシュ関数処理時間の概念を図6-1-1，処理時間を表6-1-1～6-1-2に示します。

図 6-1-1 ステータス・チェック・ユーザ・モードにおけるフラッシュ関数処理時間の概念図

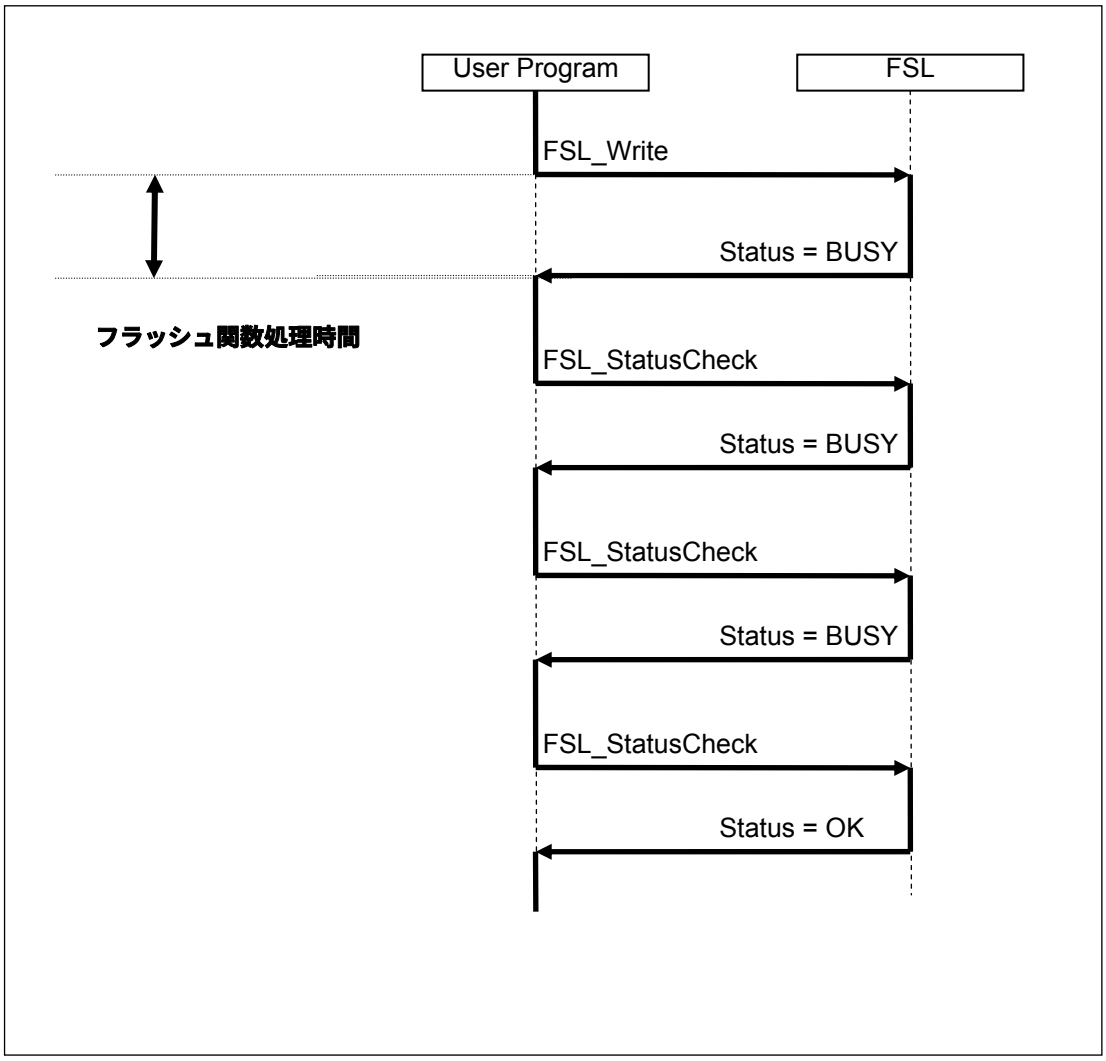


表 6-1-1 ステータス・チェック・ユーザ・モードのフラッシュ関数処理時間（フル・スピード・モード）

| FSL_Function | Max | |
|---|---|---|
| FSL_Init | 4540 /fclk | μs |
| FSL_PrepareFunctions | 2465 /fclk | μs |
| FSL_PrepareExtFunctions | 1221 /fclk | μs |
| FSL_ChangeInterruptTable | 253 /fclk | μs |
| FSL_RestoreInterruptTable | 229 /fclk | μs |
| FSL_Open | 10 /fclk | μs |
| FSL_Close | 10 /fclk | μs |
| FSL_BlankCheck | 2080 /fclk+ | 30 μs |
| FSL_Erase | 2195 /fclk+ | 30 μs |
| FSL_IVerify | 2099 /fclk+ | 30 μs |
| FSL_Write | 2460 /fclk+ | 30 μs |
| FSL_GetSecurityFlags | 331 /fclk+ | 0 μs |
| FSL_GetBootFlag | 328 /fclk+ | 0 μs |
| FSL_GetSwapState | 206 /fclk+ | 0 μs |
| FSL_GetBlockEndAddr | 368 /fclk+ | 0 μs |
| FSL_GetFlashShieldWindow | 307 /fclk+ | 0 μs |
| FSL_SetBlockEraseProtectFlag | 2351 /fclk+ | 30 μs |
| FSL_SetWriteProtectFlag | 2350 /fclk+ | 30 μs |
| FSL_SetBootClusterProtectFlag | 2350 /fclk+ | 30 μs |
| FSL_InvertBootFlag | 2345 /fclk+ | 30 μs |
| FSL_SetFlashShieldWindow | 2168 /fclk+ | 30 μs |
| FSL_SwapBootCluster | 431 /fclk+ | 32 μs |
| FSL_SwapActiveBootCluster | 2328 /fclk+ | 30 μs |
| FSL_ForceReset | － | |
| FSL_StatusCheck | 802 /fclk+ | 18 μs |
| FSL_StandBy | | |
| (Erase) | 935 /fclk+ | 31 μs |
| (except Erase)<br>in case of FSL_SetXXX are suported | 140004 /fclk+ | 513812 μs |
| (except Erase)<br>in case of FSL_SetXXX are not suported | 76101 /fclk+ | 35952 μs |
| FSL_WakeUp | | |
| (Suspended Erase) | 2144 /fclk+ | 30 μs |
| (except Erase) | 148 /fclk+ | 0 μs |
| FSL_GetVersionString | 10 /fclk | μs |

備考． fclk：CPU／周辺ハードウエア・クロック周波数( 例：20Mhz 時の fclk = 20 )

表 6-1-2 ステータス・チェック・ユーザ・モードのフラッシュ関数処理時間（ワイド・ボルテージ・モード）

| FSL_Function | MAX | |
|---|---|---|
| FSL_Init | 4540 /fclk | μs |
| FSL_PrepareFunctions | 2465 /fclk | μs |
| FSL_PrepareExtFunctions | 1221 /fclk | μs |
| FSL_ChangeInterruptTable | 253 /fclk | μs |
| FSL_RestoreInterruptTable | 229 /fclk | μs |
| FSL_Open | 10 /fclk | μs |
| FSL_Close | 10 /fclk | μs |
| FSL_BlankCheck | 2079 /fclk+ | 30 μs |
| FSL_Erase | 2195 /fclk+ | 30 μs |
| FSL_IVerify | 2099 /fclk+ | 30 μs |
| FSL_Write | 2460 /fclk+ | 30 μs |
| FSL_GetSecurityFlags | 331 /fclk+ | 0 μs |
| FSL_GetBootFlag | 328 /fclk+ | 0 μs |
| FSL_GetSwapState | 206 /fclk+ | 0 μs |
| FSL_GetBlockEndAddr | 368 /fclk+ | 0 μs |
| FSL_GetFlashShieldWindow | 307 /fclk+ | 0 μs |
| FSL_SetBlockEraseProtectFlag | 2351 /fclk+ | 30 μs |
| FSL_SetWriteProtectFlag | 2350 /fclk+ | 30 μs |
| FSL_SetBootClusterProtectFlag | 2350 /fclk+ | 30 μs |
| FSL_InvertBootFlag | 2345 /fclk+ | 30 μs |
| FSL_SetFlashShieldWindow | 2168 /fclk+ | 30 μs |
| FSL_SwapBootCluster | 431 /fclk+ | 32 μs |
| FSL_SwapActiveBootCluster | 2328 /fclk+ | 30 μs |
| FSL_ForceReset | － | |
| FSL_StatusCheck | 802 /fclk+ | 18 μs |
| FSL_StandBy | | |
| (case: Erase) | 935 /fclk+ | 44 μs |
| (case: except Erase)<br>in case of FSL_SetXXX are suported | 122911 /fclk+ | 53801 μs |
| (case: except Erase)<br>in case of FSL_SetXXX are not suported | 73221 /fclk+ | 69488 μs |
| FSL_WakeUp | | |
| (case: Erase Suspended) | 2144 /fclk+ | 30 μs |
| (case: except Erase) | 148 /fclk+ | 0 μs |
| FSL_GetVersionString | 10 /fclk | μs |

備考． fclk：CPU／周辺ハードウエア・クロック周波数( 例：20Mhz 時の fclk = 20 )

### 6.1.2　ステータス・チェック・インターナル・モード

このモードのフラッシュ関数処理時間の概念を図6-1-2，処理時間を表6-1-3～6-1-4に示します。

図 6-1-2 ステータス・チェック・インターナル・モードにおけるフラッシュ関数処理時間の概念図

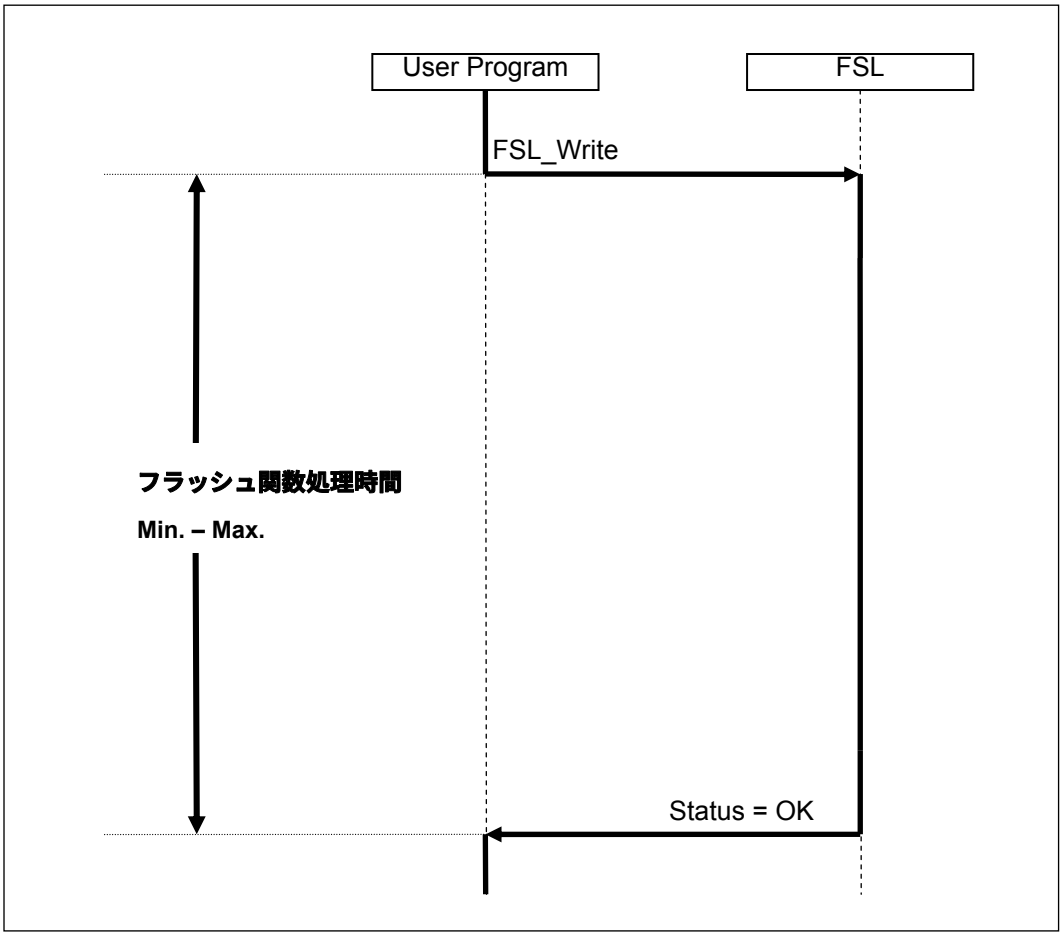


表 6-1-3 ステータス・チェック・インターナル・モードのフラッシュ関数処理時間(フル・スピード・モード)

| FSL_Function | Min | Max |
|---|---|---|
| FSL_Init | － | 4540 /fclk μs |
| FSL_PrepareFunctions | － | 2465 /fclk μs |
| FSL_PrepareExtFunctions | － | 1221 /fclk μs |
| FSL_ChangeInterruptTable | － | 253 /fclk μs |
| FSL_RestoreInterruptTable | － | 229 /fclk μs |
| FSL_Open | － | 10 /fclk μs |
| FSL_Close | － | 10 /fclk μs |
| FSL_BlankCheck | 3313 /fclk+ 84 μs | 4844 /fclk+ 164 μs |
| FSL_Erase | 4880 /fclk+ 163 μs | 73342 /fclk+ 255366 μs |
| FSL_IVerify | － | 10476 /fclk+ 1107 μs |
| FSL_Write | 3129 /fclk+ 66 +( 595 /fclk+ 60 )*W μs | 3130 /fclk+ 66 +( 1153 /fclk+ 561 )*W μs |
| FSL_GetSecurityFlags | － | 331 /fclk+ 0 μs |
| FSL_GetBootFlag | － | 328 /fclk+ 0 μs |
| FSL_GetSwapState | － | 206 /fclk+ 0 μs |
| FSL_GetBlockEndAddr | － | 368 /fclk+ 0 μs |
| FSL_GetFlashShieldWindow | － | 307 /fclk+ 0 μs |
| FSL_SetBlockEraseProtectFlag | 1574 /fclk+ 18 μs | 140950 /fclk+ 513830 μs |
| FSL_SetWriteProtectFlag | 1573 /fclk+ 18 μs | 140949 /fclk+ 513830 μs |
| FSL_SetBootClusterProtectFlag | 1574 /fclk+ 18 μs | 140950 /fclk+ 513830 μs |
| FSL_InvertBootFlag | 1569 /fclk+ 18 μs | 140945 /fclk+ 513830 μs |
| FSL_SetFlashShieldWindow | 1385 /fclk+ 18 μs | 140768 /fclk+ 513830 μs |
| FSL_SwapBootCluster | － | 431 /fclk+ 32 μs |
| FSL_SwapActiveBootCluster | 1950 /fclk+ 50 μs | 141326 /fclk+ 513862 μs |
| FSL_ForceReset | － | － |
| FSL_StatusCheck | － | － |
| FSL_StandBy | － | － |
| FSL_WakeUp | － | － |
| FSL_GetVersionString | － | 10 /fclk μs |

備考 1. fclk：CPU／周辺ハードウエア・クロック周波数( 例：20Mhz 時の fclk = 20 )

2. W：書き込みデータのワード数（1 ワード = 4 バイト）

表 6-1-4 ステータス・チェック・インターナル・モードのフラッシュ関数処理時間(ワイド・ボルテージ・モード)

| FSL_Function | Min | Max |
|---|---|---|
| FSL_Init | － | 4540 /fclk μs |
| FSL_PrepareFunctions | － | 2465 /fclk μs |
| FSL_PrepareExtFunctions | － | 1221 /fclk μs |
| FSL_ChangeInterruptTable | － | 253 /fclk μs |
| FSL_RestoreInterruptTable | － | 229 /fclk μs |
| FSL_Open | － | 10 /fclk μs |
| FSL_Close | － | 10 /fclk μs |
| FSL_BlankCheck | 3309 /fclk+ 124 μs | 4585 /fclk+ 401 μs |
| FSL_Erase | 4678 /fclk+ 401 μs | 64471 /fclk+ 266193 μs |
| FSL_IVerify | － | 7661 /fclk+ 7534 μs |
| FSL_Write | 3129 /fclk+ 66 +( 591 /fclk+ 112 )*W μs | 3130 /fclk+ 66 +( 1108 /fclk+ 1085 )*W μs |
| FSL_GetSecurityFlags | － | 331 /fclk+ 0 μs |
| FSL_GetBootFlag | － | 328 /fclk+ 0 μs |
| FSL_GetSwapState | － | 206 /fclk+ 0 μs |
| FSL_GetBlockEndAddr | － | 368 /fclk+ 0 μs |
| FSL_GetFlashShieldWindow | － | 307 /fclk+ 0 μs |
| FSL_SetBlockEraseProtectFlag | 1574 /fclk+ 18 μs | 123857 /fclk+ 538032 μs |
| FSL_SetWriteProtectFlag | 1573 /fclk+ 18 μs | 123856 /fclk+ 538032 μs |
| FSL_SetBootClusterProtectFlag | 1574 /fclk+ 18 μs | 123857 /fclk+ 538032 μs |
| FSL_InvertBootFlag | 1569 /fclk+ 18 μs | 123852 /fclk+ 538032 μs |
| FSL_SetFlashShieldWindow | 1385 /fclk+ 18 μs | 123675 /fclk+ 538032 μs |
| FSL_SwapBootCluster | － μs | 431 /fclk+ 32 μs |
| FSL_SwapActiveBootCluster | 1950 /fclk+ 50 μs | 124233 /fclk+ 538064 μs |
| FSL_ForceReset | － | － |
| FSL_StatusCheck | － | － |
| FSL_StandBy | － | － |
| FSL_WakeUp | － | － |
| FSL_GetVersionString | － | 10 /fclk μs |

備考 1. fclk：CPU／周辺ハードウエア・クロック周波数( 例：20Mhz 時の fclk = 20 )

2. W：書き込みデータのワード数（1 ワード = 4 バイト）

## 6.2 FSL_StatusCheck(ステータス・チェック)推奨間隔時間

ステータス・チェック・ユーザ・モードで処理を行う場合，FSL_StatusCheck関数でステータス・チェックを行いますが，シーケンサの制御が終了する前にFSL_StatusCheck関数を実行してもあまり意味がないため，各フラッシュ関数で実行している処理毎に，一定の間隔を空ける事によってステータス・チェック処理の効率が向上します。また，FSL_Write関数による書き込みは，4byte毎にステータス・チェック処理によるトリガーが必要になるため，4byte単位でのステータス・チェック処理が必要となります。

図 6-2-1 FSL_Write 関数のステータス・チェック間隔時間の概念図(12byte 書き込みの場合)

・ ステータス・チェック・ユーザ・モードで 12byte の書き込みを行う場合，シーケンサは 4byte 毎に書き込みを行うため，4byte の書き込み終了時に FSL_StatusCheck 関数によって次の書き込みへのトリガーが必要となります。4byte 以降の書き込み処理が残っている状態で FSL_StatusCheck 関数を実行しない場合，次の書き込み処理に遷移できないため，書き込み処理が終了しません。

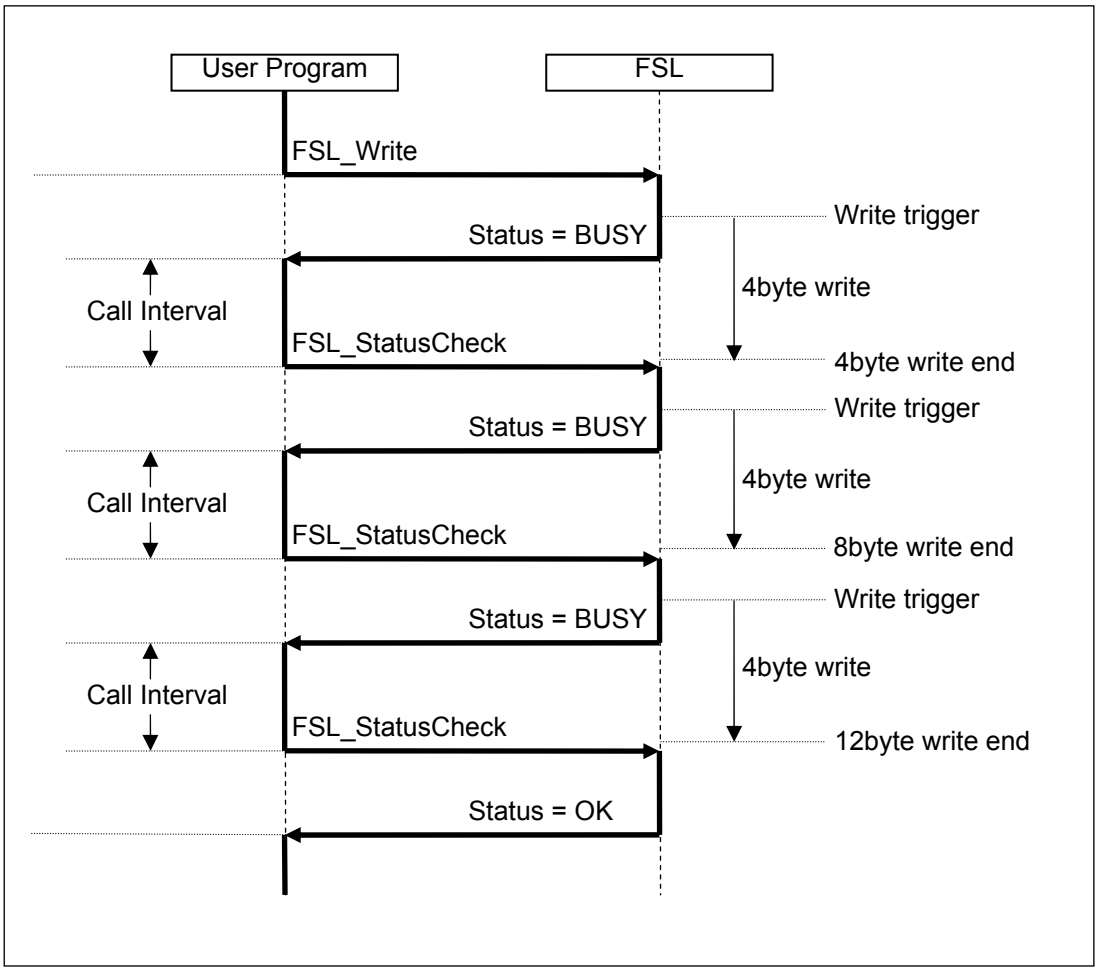


図 6-2-2　FSL_Write 関数以外のステータス・チェック間隔時間の概念図(消去の場合)

・　ステータス・チェック・ユーザ・モードで FSL_Write 以外の処理を行う場合，シーケンサは全ての処理を終了すまで Busy 状態となり，FSL_StatusCheck 関数によるトリガーは必要ありません。

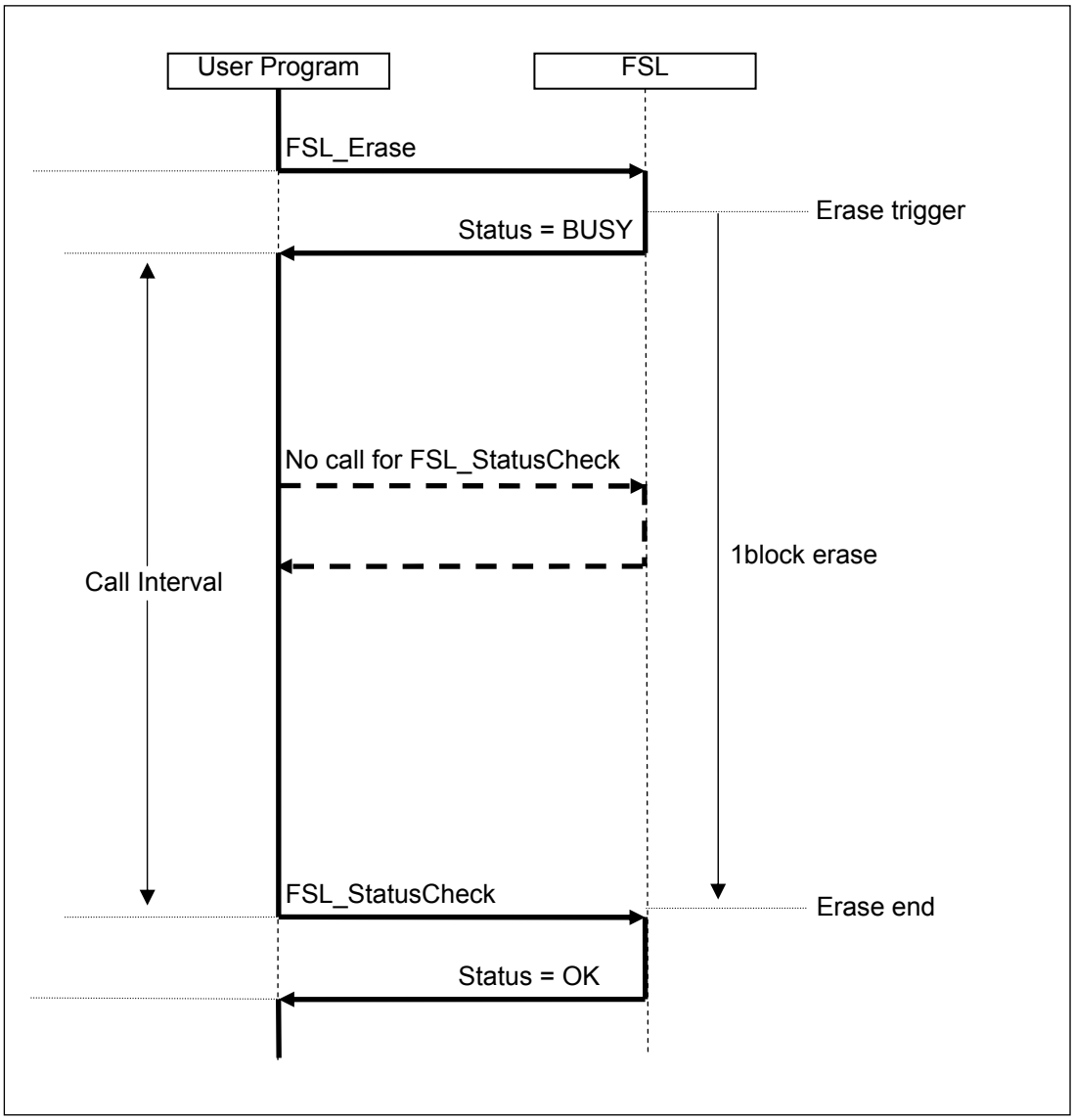


表 6-2-1　FSL_StatusCheck(ステータス・チェック)推奨間隔時間(フル・スピード・モード)

| FSL_Function | Call Interval |
|---|---|
| FSL_Init | － |
| FSL_PrepareFunctions | － |
| FSL_PrepareExtFunctions | － |
| FSL_ChangeInterruptTable | － |
| FSL_RestoreInterruptTable | － |
| FSL_Open | － |
| FSL_Close | － |
| FSL_BlankCheck | 1569 /fclk+ 98 μs |
| FSL_Erase | |
| (case: block is blanked) | 1490 /fclk+ 97 μs |
| (case: block is not blanked) | 3092 /fclk+ 6471 μs |
| FSL_IVerify | 7181 /fclk+ 1041 μs |
| FSL_Write注 | 72 /fclk+ 60 μs |
| FSL_GetSecurityFlags | － |
| FSL_GetBootFlag | － |
| FSL_GetSwapState | － |
| FSL_GetBlockEndAddr | － |
| FSL_GetFlashShieldWindow | － |
| FSL_SetBlockEraseProtectFlag | 6431 /fclk+ 7053 μs |
| FSL_SetWriteProtectFlag | 6431 /fclk+ 7053 μs |
| FSL_SetBootClusterProtectFlag | 6431 /fclk+ 7053 μs |
| FSL_InvertBootFlag | 6431 /fclk+ 7053 μs |
| FSL_SetFlashShieldWindow | 6431 /fclk+ 7053 μs |
| FSL_SwapBootCluster | － |
| FSL_SwapActiveBootCluster | 6431 /fclk+ 7053 μs |
| FSL_ForceReset | － |
| FSL_StatusCheck | － |
| FSL_StandBy | － |
| FSL_WakeUp | |
| (case: block is blanked) | 1490 /fclk+ 97 μs |
| (case: block is not blanked) | 3092 /fclk+ 6471 μs |
| FSL_GetVersionString | － |

備考.　fclk：CPU／周辺ハードウエア・クロック周波数( 例：20Mhz 時の fclk = 20 )

注.　FSL_Write 関数は 4 バイト毎の推奨間隔時間

表 6-2-2　FSL_StatusCheck(ステータス・チェック)推奨間隔時間(ワイド・ボルテージ・モード)

| FSL_Function | Call Interval |
|---|---|
| FSL_Init | － |
| FSL_PrepareFunctions | － |
| FSL_PrepareExtFunctions | － |
| FSL_ChangeInterruptTable | － |
| FSL_RestoreInterruptTable | － |
| FSL_Open | － |
| FSL_Close | － |
| FSL_BlankCheck | 1310 /fclk+ 335 μs |
| FSL_Erase | |
| (case: block is blanked) | 1289 /fclk+ 335 μs |
| (case: block is not blanked) | 2689 /fclk+ 6959 μs |
| FSL_IVerify | 4366 /fclk+ 7468 μs |
| FSL_Write注 | 67 /fclk+ 112 μs |
| FSL_GetSecurityFlags | － |
| FSL_GetBootFlag | － |
| FSL_GetSwapState | － |
| FSL_GetBlockEndAddr | － |
| FSL_GetFlashShieldWindow | － |
| FSL_SetBlockEraseProtectFlag | 5728 /fclk+ 8445 μs |
| FSL_SetWriteProtectFlag | 5728 /fclk+ 8445 μs |
| FSL_SetBootClusterProtectFlag | 5728 /fclk+ 8445 μs |
| FSL_InvertBootFlag | 5728 /fclk+ 8445 μs |
| FSL_SetFlashShieldWindow | 5728 /fclk+ 8445 μs |
| FSL_SwapBootCluster | － |
| FSL_SwapActiveBootCluster | 5728 /fclk+ 8445 μs |
| FSL_ForceReset | － |
| FSL_StatusCheck | － |
| FSL_StandBy | － |
| FSL_WakeUp | |
| (case: block is blanked) | 1289 /fclk+ 335 μs |
| (case: block is not blanked) | 2689 /fclk+ 6959 μs |
| FSL_GetVersionString | － |

備考.　fclk：CPU／周辺ハードウエア・クロック周波数( 例：20Mhz 時の fclk = 20 )

注.　FSL_Write 関数は 4 バイト毎の推奨間隔時間

# 7 インストール

この節では，フラッシュ・セルフ・プログラミング・ライブラリのインストールとアンインストールの手順について説明します。

## 7.1 インストール方法

フラッシュ・セルフ・プログラミング・ライブラリのインストールは次の手順で行います。

(1) Windowsを起動します。

(2) フラッシュ・セルフ・プログラミング・ライブラリが入っている圧縮ファイルを解凍し，展開されたフォルダをユーザ任意の場所に配置します。

## 7.2 アンインストール方法

フラッシュ・セルフ・プログラミング・ライブラリのアンインストールは次の手順で行います。

(1) Windowsを起動します。

(2) ユーザ任意の場所に配置したフラッシュ・セルフ・プログラミング・ライブラリが入っているフォルダを削除します。

## 7.3 ファイル構成

フラッシュ・セルフ・プログラミング・ライブラリが入っている圧縮ファイルを解凍し，展開されるファイルの構成は，次のとおりです。

```
配置先フォルダ
└─ FSLRL78 Type01
    └─ Vx.xx
        ├─ doc
        │   ├─ R01AN0350JJ0100.pdf          : アプリケーション・ノート
        │   ├─ ZUD-F35-10-0208.pdf          : 使用上の留意点文書(本書)
        │   └─ ソフトウェアのご使用条件.pdf : ソフトウェアのご使用条件
        │
        ├─ librl78
        │   └─ fsl.lib  : フラッシュ・セルフ・プログラミング・ライブラリ
        │
        └─ incrl78
            ├─ fsl.h       : C言語用インクルード・ファイル
            ├─ fsl.inc     : アセンブラ用インクルード・ファイル
            └─ fsl_types.h : C言語用定義設定インクルード・ファイル
```

# 8 ビルド方法

この節では，フラッシュ・セルフ・プログラミング・ライブラリを用いたプログラムのビルドの手順を説明します。

## 8.1 使用するソフトウェア

フラッシュ・セルフ・プログラミング・ライブラリを用いたプログラムをビルドする際には，次の統合開発環境が必要です。

・統合開発環境 CubeSuite E1.50

## 8.2 CubeSuite でのビルド方法

CubeSuite を用いてフラッシュ・セルフ・プログラミング・ライブラリをユーザ・プログラムに組み込んで，ビルドする手順を説明します。

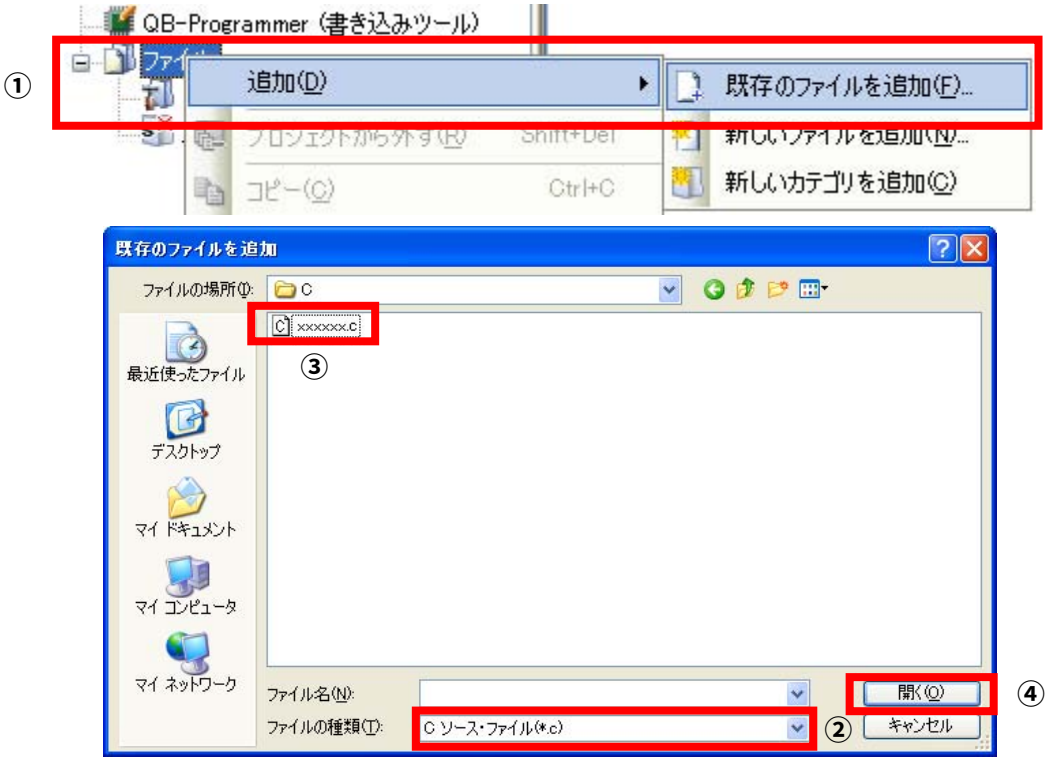
### 8.2.1 C 言語の場合

#### (1) プロジェクトの作成とソース・ファイルの設定

CubeSuite でプロジェクトを作成し，表示された画面の左側にある[ プロジェクト・ツリー ]から，[ ファイル ]を右クリックして表示されるリストの[ 追加 ]を選択し，[ 既存のファイルを追加 ]をクリックすると，[ 既存のファイルを追加 ]画面が表示されます（図 8－1）。

次に画面中にある[ ファイルの種類 ]のプルダウンメニューをクリックすると，ファイルの種類の一覧が表示されるので，その中にある[ C ソース・ファイル(*.c) ]を選択し，ソース・ファイルとしてユーザ・プログラムのファイルを登録してください。

図 8-1 ソース・ファイルの指定

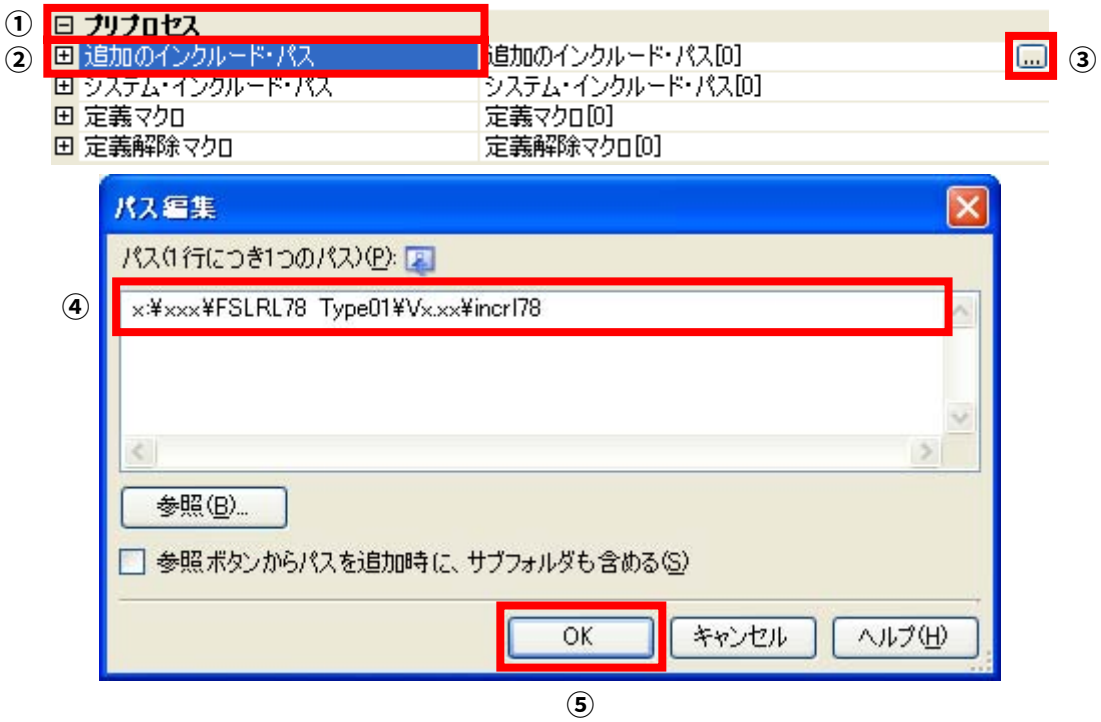


## (2) インクルード・ファイル・パスの設定

CubeSuite の[ プロジェクト・ツリー ]にある,[ CA78K0R(ビルド・ツール) ]をクリックして CA78K0R のプロパティ画面を開き， 下部にある[ コンパイル・オプション ]タブをクリックすると，コンパイル・オプション画面が表示されます。

次に画面中にある[ プリプロセス ]をクリックし,項目を表示させて[ 追加のインクルード・パス ]の欄を選択し，右に表示される[ ... ]ボタンをクリックすると，[ パス編集 ]ダイアログ画面が表示されるので，画面中のテキスト・ボックスに，フラッシュ・セルフ・プログラミング・ライブラリ用のヘッダ・ファイルがあるフォルダのパスを入力します（図 8－2）。 [ 参照 ] ボタンをクリックして，マウス操作により指定することもできます。

図 8-2 インクルード・ファイルの指定




## (3) ライブラリ・ファイルの設定

ライブラリ・ファイルをプロジェクトで使用するための設定は 2 通りの方法があります。

・プロジェクトにライブラリ・ファイルを登録する方法

CubeSuite の[ プロジェクト・ツリー ]にある,[ ファイル ]を右クリックして表示されるリストから[ 追加 ]を選択し，[ 既存のファイルを追加 ]をクリックすると，[ 既存のファイルを追加 ]画面が表示されます（図 8－3）。

次に画面中にある[ ファイルの種類 ]のプルダウンメニューをクリックすると，ファイルの種類の一覧が表示されるので，その中にある[ ライブラリ・ファイル(*.lib) ]を選択し，フラッシュ・セルフ・プログラミング・ライブラリ（fsl.lib）のファイルを登録してください。

・ライブラリ・ファイルのフォルダ・パスを設定し，使用するライブラリ名を指定する方法

CubeSuite の[ プロジェクト・ツリー ]にある，[ CA78K0R(ビルド・ツール) ]をクリックして CA78K0R のプロパティ画面を開き， 下部にある[ リンク・オプション ]タブをクリックすると，リ

ンク・オプション画面が表示されます。

次に画面中の[ ライブラリ ]の項目にある[ 追加のライブラリ・パス ] の欄を選択し，右に表示される[ ... ]ボタンをクリックすると，[ パス編集 ]ダイアログ画面が表示されるので，画面中のテキスト・ボックスに，フラッシュ・セルフ・プログラミング・ライブラリがあるフォルダのパスを入力します（図 8－4）。[ 参照 ] ボタンをクリックして，マウス操作により指定することもできます。

フォルダ・パスの設定後，[ ライブラリ ]の項目にある[ 使用するライブラリ・ファイル ] の欄を選択し，右に表示される[ ... ]ボタンをクリックすると，[ テキスト編集 ]ダイアログ画面が表示されるので，画面中のテキスト・ボックスに，フラッシュ・セルフ・プログラミング・ライブラリ（fsl.lib）のファイル名を設定してください（図 8－5）。ファイル名はフォルダ・パスを含めて設定することも可能です。

図 8-3 ライブラリ・ファイルの指定

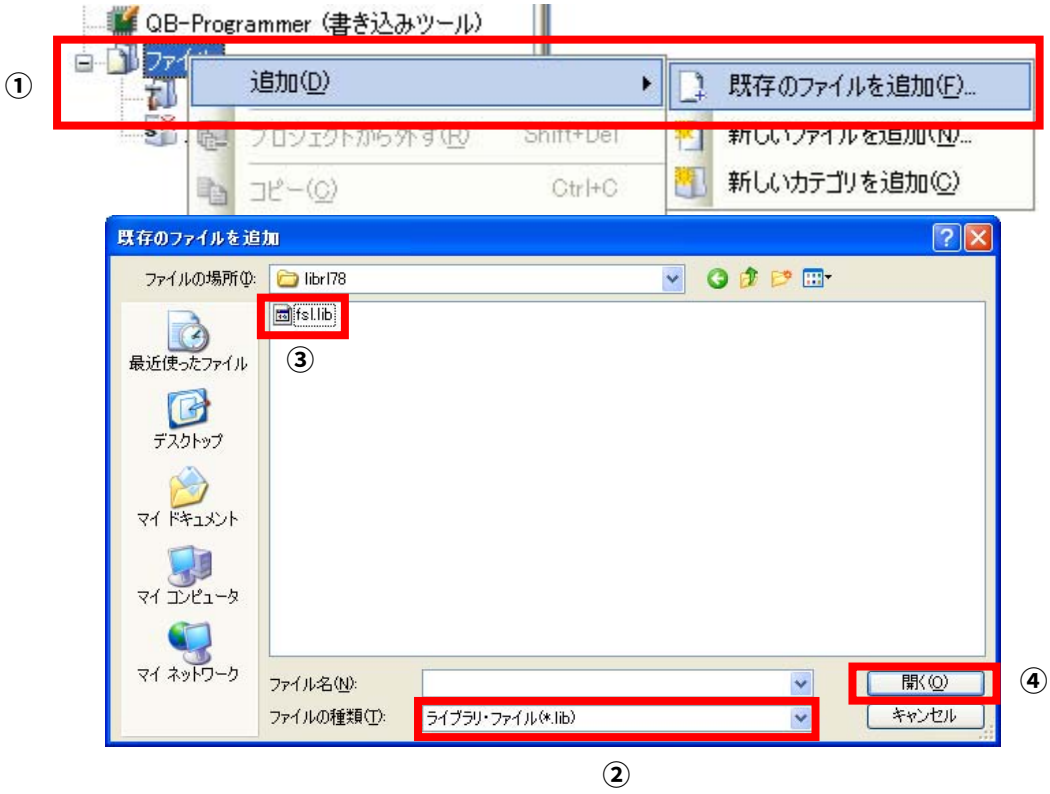


図 8-4 ライブラリ・ファイルのフォルダ・パスを指定

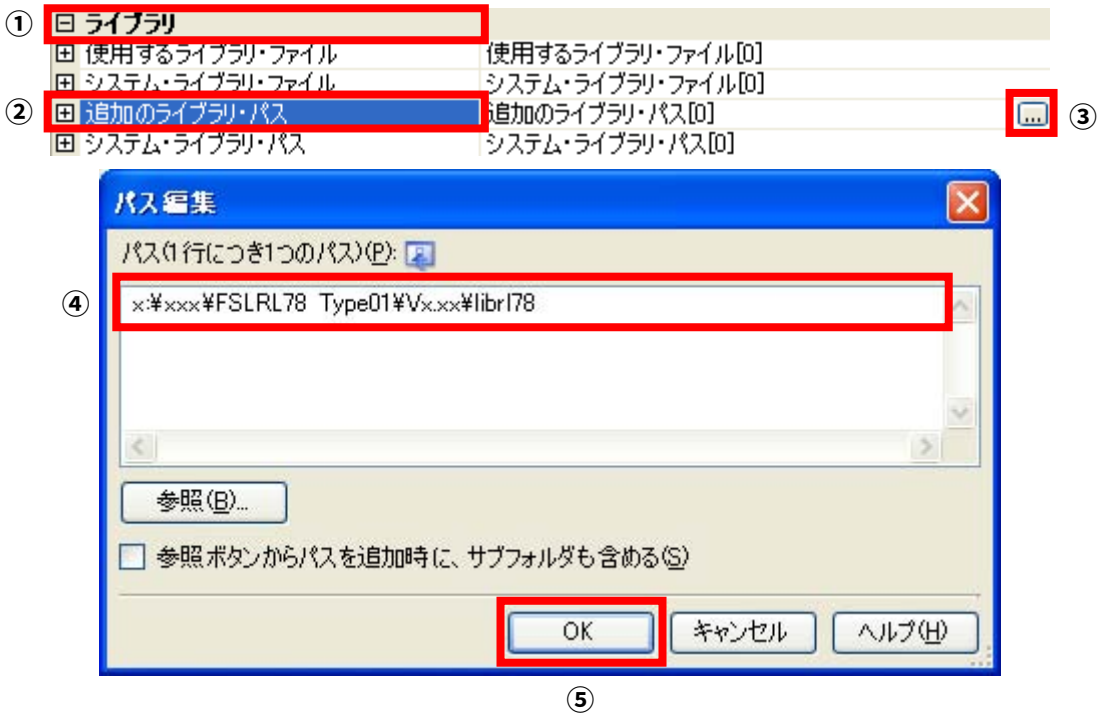
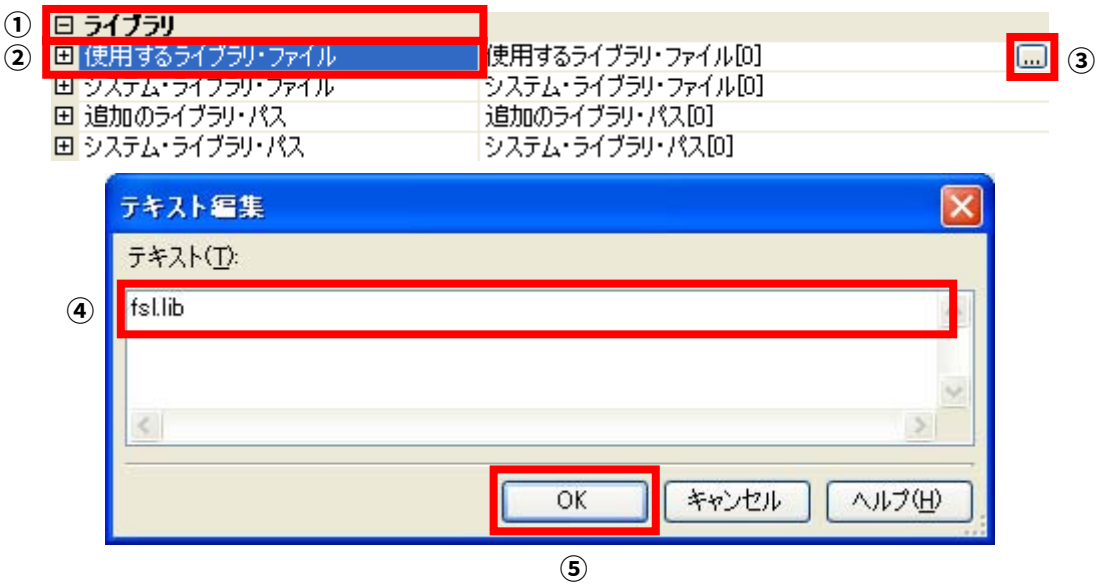



図 8-5 使用するライブラリ・ファイル名の設定

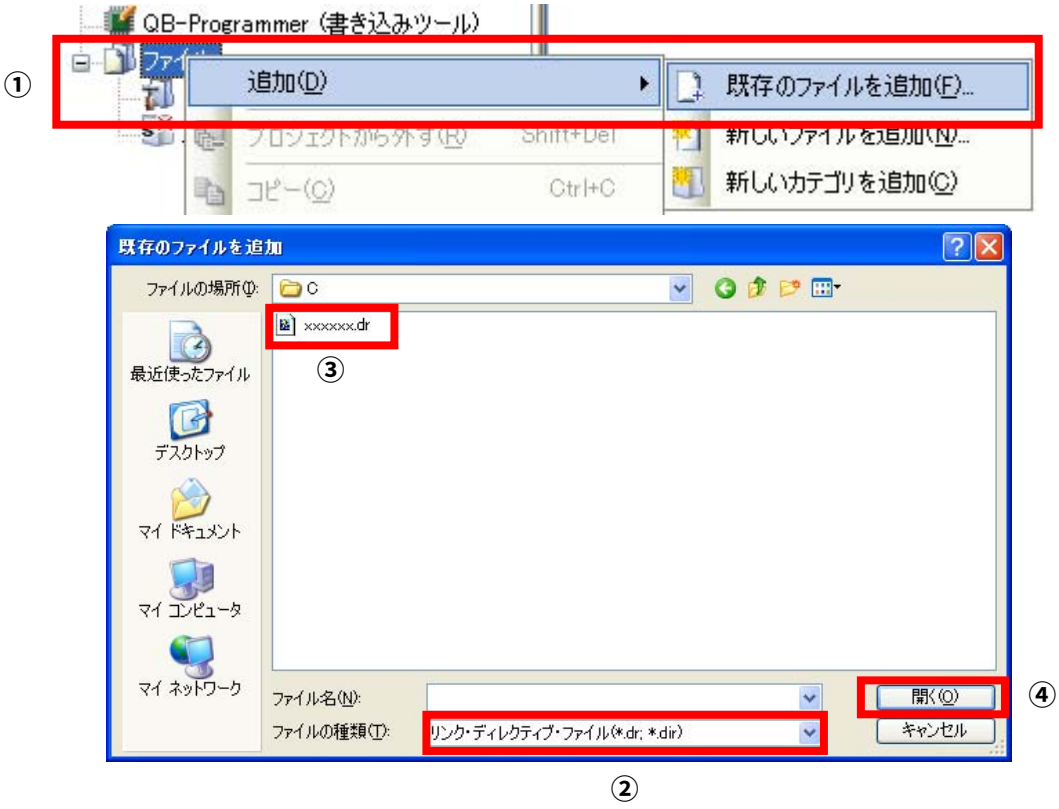


## (4) リンク・ディレクティブの設定

CubeSuite の[ プロジェクト・ツリー ]にある，[ ファイル ]を右クリックして表示されるリストから[ 追加 ]を選択し，[ 既存のファイルを追加 ]をクリックすると，[ 既存のファイルを追加 ]画面が表示されます（図 8－6）。

次に画面中にある[ ファイルの種類 ]のプルダウンメニューをクリックすると，ファイルの種類の一覧が表示されるので，その中にある[ リンク・ディレクティブ・ファイル(*.dr;*.dir) ]を選択し，リンク・ディレクティブ・ファイル名のユーザ・ファイルを登録してください。

図 8-6 リンク・ディレクティブ・ファイルの指定




## (5) ビルド

CubeSuite のメニュー [ ビルド ] → [ ビルド・プロジェクト ] をクリックして，ビルドを実行します。

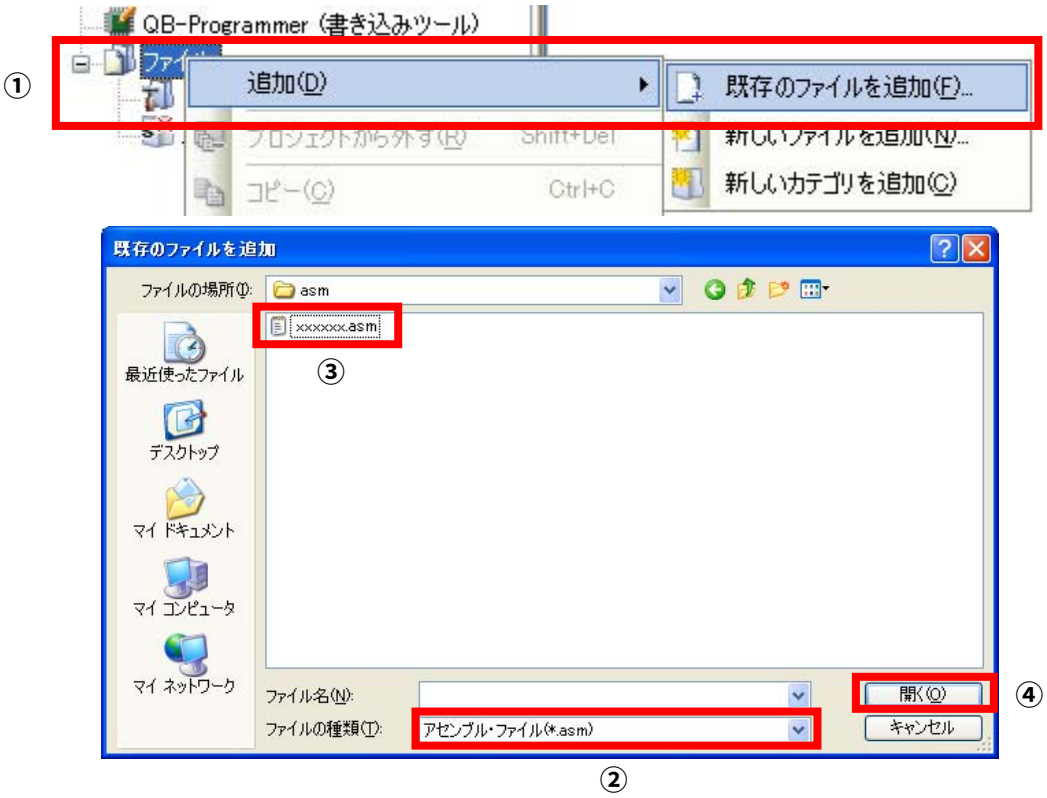
### 8.2.2　アセンブリ言語の場合

#### (1) プロジェクトの作成とソース・ファイルの設定

CubeSuite でプロジェクトを作成し，表示された画面の左側にある[ プロジェクト・ツリー ]から，[ ファイル ]を右クリックして表示されるリストの[ 追加 ]を選択し，[ 既存のファイルを追加 ]をクリックすると，[ 既存のファイルを追加 ]画面が表示されます（図 8－7）。

次に画面中にある[ ファイルの種類 ]のプルダウンメニューをクリックすると，ファイルの種類の一覧が表示されるので，その中にある[ アセンブル・ファイル(*.asm) ]を選択し，ソース・ファイルとしてユーザ・ファイルを登録してください。

図 8-7 ソース・ファイルの指定

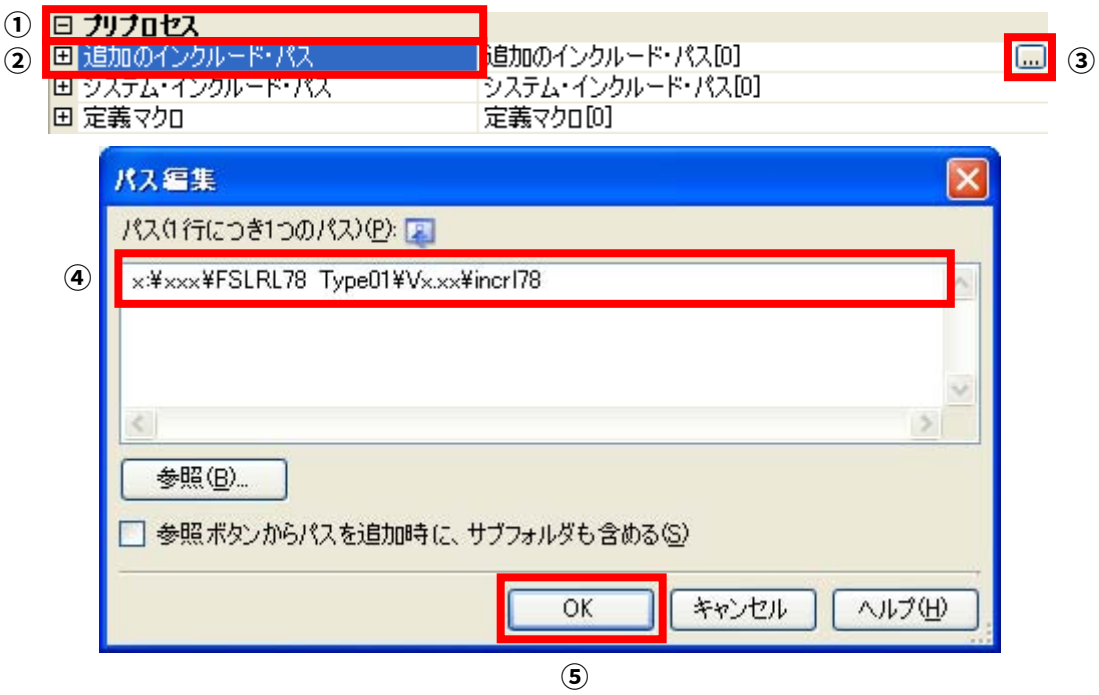



#### (2) インクルード・ファイル・パスの設定

CubeSuite の[ プロジェクト・ツリー ]にある，[ CA78K0R(ビルド・ツール) ]をクリックして CA78K0R のプロパティ画面を開き， 下部にある[ アセンブル・オプション ]タブをクリックすると，アセンブル・オプション画面が表示されます。

次に画面中にある[ プリプロセス ]をクリックし，項目を表示させて[ 追加のインクルード・パス ]の欄を選択し，右に表示される[ … ]ボタンをクリックすると，[ パス編集 ]ダイアログ画面が表示されるので，画面中のテキスト・ボックスに，フラッシュ・セルフ・プログラミング・ライブラリ用のヘッダ・ファイルがあるフォルダのパスを入力します（図 8－8）。 [ 参照 ] ボタンをクリックして，マウス操作により指定することもできます。

図 8-8 インクルード・ファイルの指定




## (3) ライブラリ・ファイルの設定

ライブラリ・ファイルをプロジェクトで使用するための設定は 2 通りの方法があります。

・プロジェクトにライブラリ・ファイルを登録する方法

CubeSuite の[ プロジェクト・ツリー ]にある，[ ファイル ]を右クリックして表示されるリストから[ 追加 ]を選択し，[ 既存のファイルを追加 ]をクリックすると，[ 既存のファイルを追加 ] 画面が表示されます（図 8－9）。

次に画面中にある[ ファイルの種類 ]のプルダウンメニューをクリックすると，ファイルの種類の一覧が表示されるので，その中にある[ ライブラリ・ファイル(*.lib) ]を選択し，フラッシュ・セルフ・プログラミング・ライブラリ（fsl.lib）のファイルを登録してください。

・ライブラリ・ファイルのフォルダ・パスを設定し，使用するライブラリ名を指定する方法

CubeSuite の[ プロジェクト・ツリー ]にある，[ CA78K0R(ビルド・ツール) ]をクリックして CA78K0R のプロパティ画面を開き， 下部にある[ リンク・オプション ]タブをクリックすると，リンク・オプション画面が表示されます。

次に画面中の[ ライブラリ ]の項目にある[ 追加のライブラリ・パス ] の欄を選択し，右に表示される[ ... ]ボタンをクリックすると，[ パス編集 ]ダイアログ画面が表示されるので，画面中のテキスト・ボックスに，フラッシュ・セルフ・プログラミング・ライブラリがあるフォルダのパスを入力します（図 8－10）。[ 参照 ] ボタンをクリックして，マウス操作により指定することもできます。

フォルダ・パスの設定後，[ ライブラリ ]の項目にある[ 使用するライブラリ・ファイル ] の欄を選択し，右に表示される[ ... ]ボタンをクリックすると，[ テキスト編集 ]ダイアログ画面が表示されるので，画面中のテキスト・ボックスに，フラッシュ・セルフ・プログラミング・ライブラリ（fsl.lib）のファイル名を設定してください（図 8－11）。ファイル名はフォルダ・パスを含めて設定することも可能です。

図 8-9 ライブラリ・ファイルの指定

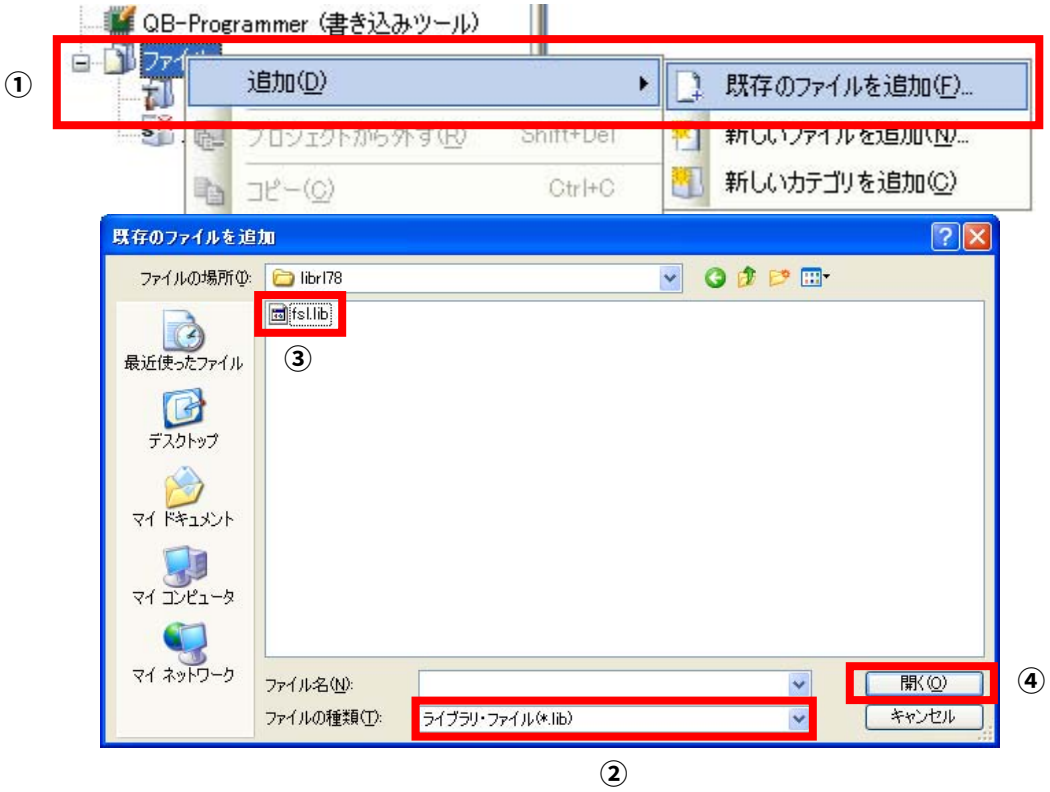



図 8-10 ライブラリ・ファイルのフォルダ・パスを指定

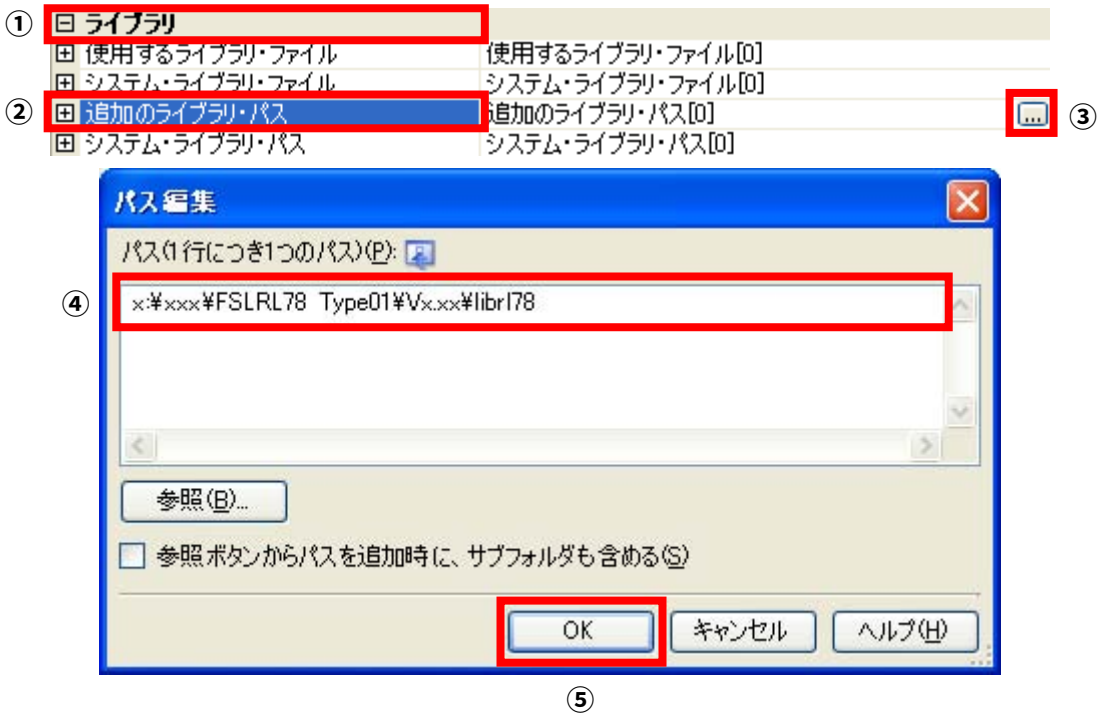
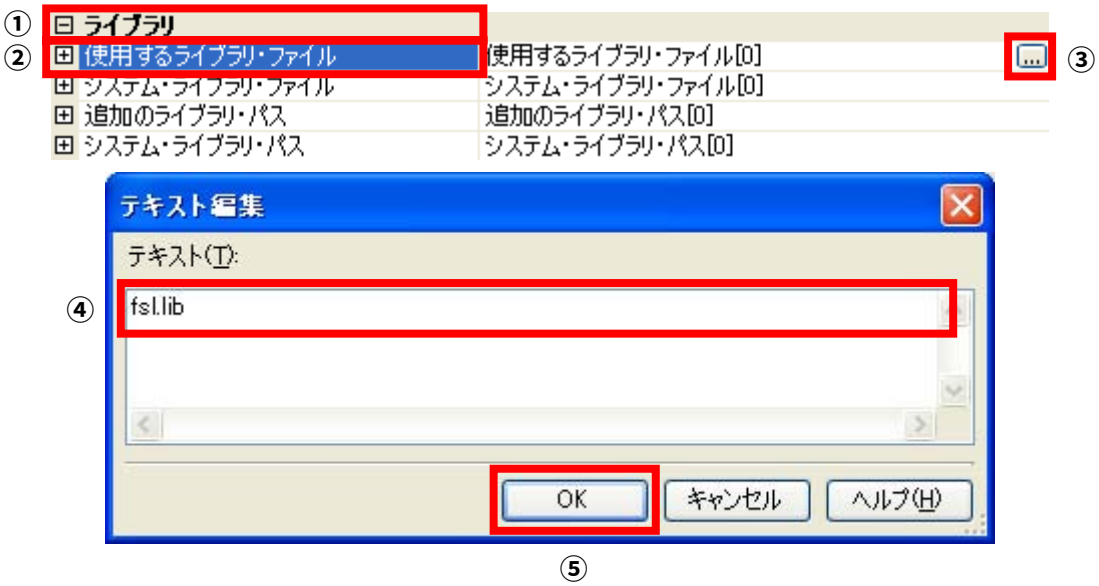


図 8-11 使用するライブラリ・ファイル名の設定

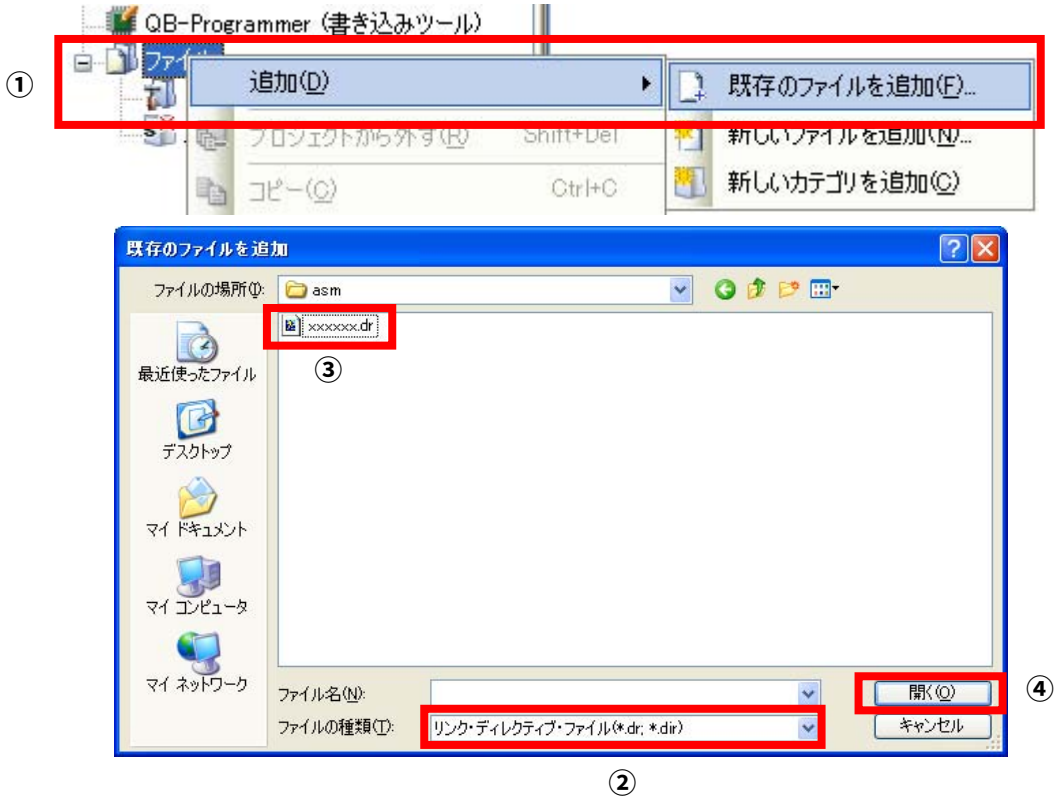



## (4) リンク・ディレクティブの設定

CubeSuite の[ プロジェクト・ツリー ]にある，[ ファイル ]を右クリックして表示されるリストから[ 追加 ]を選択し，[ 既存のファイルを追加 ]をクリックすると，[ 既存のファイルを追加 ]画面が表示されます（図 8－12）。

次に画面中にある[ ファイルの種類 ]のプルダウンメニューをクリックすると，ファイルの種類の一覧が表示されるので，その中にある[ リンク・ディレクティブ・ファイル(*.dr;*.dir) ]を選択し，リンク・ディレクティブ・ファイル名のユーザ・ファイルを登録してください。

図 8-12 リンク・ディレクティブ・ファイルの指定



### (5) ビルド

CubeSuite のメニュー [ ビルド ] → [ ビルド・プロジェクト ] をクリックして，ビルドを実行します。

# 9 デバッグ方法

IECUBE，またはオンチップ・デバッグ・エミュレータ E1 を使用してデバッグを行う場合につきましては以下の資料のご参照ください。

・ ユーザーズ・マニュアル「CubeSuite RL78,78K0R デバッグ編」