お客様各位

資料中の「ラピスセミコンダクタ」等名称の ラピステクノロジー株式会社への変更

2020 年 10 月 1 日をもって、ラピスセミコンダクタ株式会社の LSI 事業部門は、ラピステクノロジー株式会社に分割承継されました。従いまして、本資料中にあります「ラピスセミコンダクタ株式会社」、「ラピスセミ」、「ラピス」といった表記に関しましては、全て「ラピステクノロジー株式会社」に読み替えて適用するものとさせていただきます。なお、会社名、会社商標、ロゴ等以外の製品に関する内容については、変更はありません。以上、ご理解の程よろしくお願いいたします。

2020年10月1日
ラピステクノロジー株式会社

Dear customer

LAPIS Semiconductor Co., Ltd. ("LAPIS Semiconductor"), on the 1st day of October, 2020, implemented the incorporation-type company split (shinsetsu-bunkatsu) in which LAPIS established a new company, LAPIS Technology Co., Ltd. (“LAPIS Technology”) and LAPIS Technology succeeded LAPIS Semiconductor’s LSI business.

Therefore, all references to "LAPIS Semiconductor Co., Ltd.", "LAPIS Semiconductor" and/or "LAPIS" in this document shall be replaced with "LAPIS Technology Co., Ltd."

Furthermore, there are no changes to the documents relating to our products other than the company name, the company trademark, logo, etc.

Thank you for your understanding.

LAPIS Technology Co., Ltd.
October 1, 2020

ROHM GROUP
LAPIS
SEMICONDUCTOR

FJXT_MCU_HOWTO_CREATE_LIBRARY-01

機種に依存しない
ライブラリファイルの作成方法

初版　発行日 2019 年 7 月 22 日

ROHM
SEMICONDUCTOR

ご注意

1) 本資料の記載内容は改良などのため予告なく変更することがあります。

2) ラピスセミコンダクタは常に品質・信頼性の向上に取り組んでおりますが，半導体製品は種々の要因で故障・誤作動する可能性があります。
万が一，本製品が故障・誤作動した場合であっても，その影響により人身事故，火災損害等が起こらないようご使用機器でのディレーティング，冗長設計，延焼防止，バックアップ，フェイルセーフ等の安全確保をお願いします。定格を超えたご使用や使用上の注意書が守られていない場合，いかなる責任もラピスセミコンダクタは負うものではありません。

3) 本資料に記載されております応用回路例やその定数などの情報につきましては，本製品の標準的な動作や使い方を説明するものです。したがいまして，量産設計をされる場合には，外部諸条件を考慮していただきますようお願いいたします。

4) 本資料に記載されております技術情報は，本製品の代表的動作および応用回路例などを示したものであり，それをもって，当該技術情報に関するラピスセミコンダクタまたは第三者の知的財産権その他の権利を許諾するものではありません。したがいまして，上記技術情報の使用に起因して第三者の権利にかかわる紛争が発生した場合，ラピスセミコンダクタはその責任を負うものではありません。

5) 本製品は，一般的な電子機器（AV機器，OA機器，通信機器，家電製品，アミューズメント機器など）および本資料に明示した用途への使用を意図しています。

6) 本資料に掲載されております製品は，耐放射線設計はなされておりません。

7) 本製品を下記のような特に高い信頼性が要求される機器等に使用される際には，ラピスセミコンダクタへ必ずご連絡の上，承諾を得てください。
・ 輸送機器（車載，船舶，鉄道など），幹線用通信機器，交通信号機器，防災・防犯装置，安全確保のための装置，医療機器，サーバー，太陽電池，送電システム

8) 本製品を極めて高い信頼性を要求される下記のような機器等には，使用しないでください。
・ 航空宇宙機器，原子力制御機器，海底中継機器

9) 本資料の記載に従わないために生じたいかなる事故，損害もラピスセミコンダクタはその責任を負うものではありません。

10) 本資料に記載されております情報は，正確を期すため慎重に作成したものですが，万が一，当該情報の誤り・誤植に起因する損害がお客様に生じた場合においても，ラピスセミコンダクタはその責任を負うものではありません。

11) 本製品のご使用に際しては，RoHS 指令など適用される環境関連法令を遵守の上ご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して，ラピスセミコンダクタは一切の責任を負いません。本製品の RoHS 適合性などの詳細につきましては，セールス・オフィスまでお問合せください。

12) 本製品および本資料に記載の技術を輸出又は国外へ提供する際には，「外国為替及び外国貿易法」，
「米国輸出管理規則」など適用される輸出関連法令を遵守し，それらの定めにしたがって必要な手続を行ってください。

13) 本資料の一部または全部をラピスセミコンダクタの許可なく，転載・複写することを堅くお断りします。

ラピスセミコンダクタ株式会社
〒222-8575 神奈川県横浜市港北区新横浜 2-4-8
http://www.lapis-semi.com

目次

1. はじめに

本書は，機種に依存しないライブラリの作成方法について記載しています。
本書に記載の方法にしたがってライブラリを作成することにより，特定の機種に限定することなく使用することができるようになります。
なお，本書に記載のライブラリ作成方法は，SFR の情報がない機種情報ファイルを使うため，ペリフェラルを使用するプログラムは機種に依存しないライブラリとして作成することはできませんので，予めご了承ください。

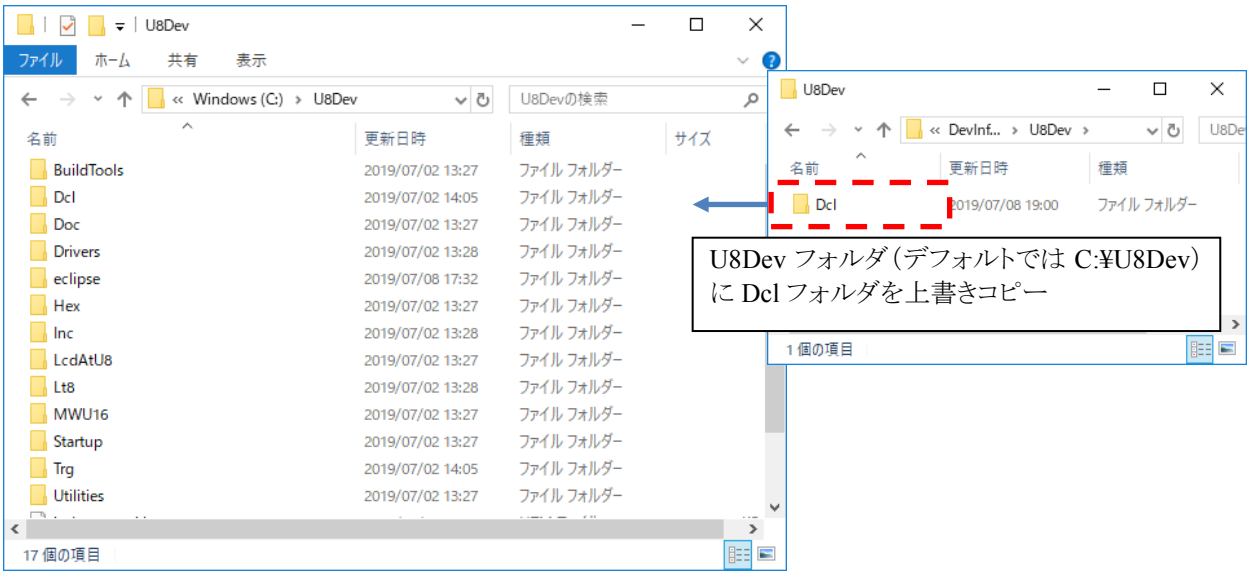
1.1. 関連するマニュアル

本書に関連するマニュアル類を以下に示します。併せて参照してください。

マニュアル名称	説明	備考
MACU8 アセンブラパッケージ ユーザーズマニュアル	アセンブラ，リンカ，オブジェクトコンバータ，ライブラリアンの使用方法を記載したマニュアルです。 アセンブリファイルの疑似命令やマップファイルの見方などを確認したい場合に参照ください。	Windows の スタートメニュー [U8 Tools] > [V2_xx_xx] > [ドキュメント] から参照いただけます。
CCU8 ユーザーズマニュアル	CCU8 C コンパイラの使用方法を記載したマニュアルです。 プラグマや組み込み関数の記述の仕方などを確認したい場合に参照ください。	
LEXIDE-U16 ユーザーズマニュアル	統合開発環境 LEXIDE-U16 の使用方法を記載したマニュアルです。 LEXIDE-U16 のオプションとコンパイラなど各ツールのオプションの対応を確認したい場合に参照ください。	Windows の スタートメニュー [U8 Tools] > [nX-U8 ドキュメント] から参照いただけます。

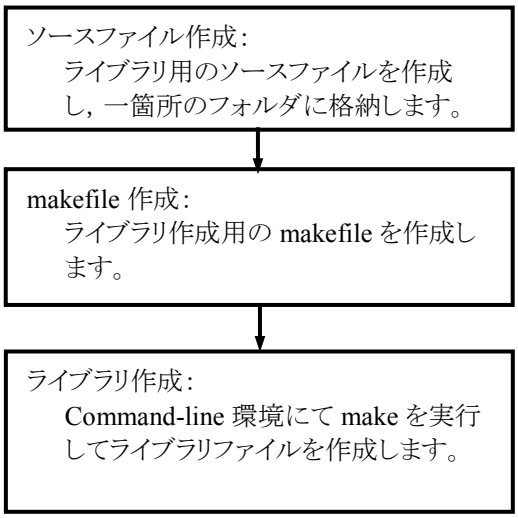
1.2. 準備

機種に依存しないライブラリを作成するには，専用の機種情報ファイルを事前にインストールしておく必要があります。
本ドキュメントに併せてご提供の DevInf_for_Lib_vxxx.zip（vxxx はバージョン番号を示す文字列）を展開してできたフォルダの下にある U8Dev フォルダ以下の内容を，U8/U16 Development Tools をインストールしたフォルダ（デフォルトでは C:¥U8Dev フォルダ）に上書きコピーしてください。



2. ライブラリの作成手順

ライブラリは以下の手順にしたがって作成します。
LEXIDE-U16 にはライブラリファイルを作成する機能はないため，ライブラリ作成用の makefile を作成し，Command-line 環境にて make を起動してライブラリファイルを作成します。

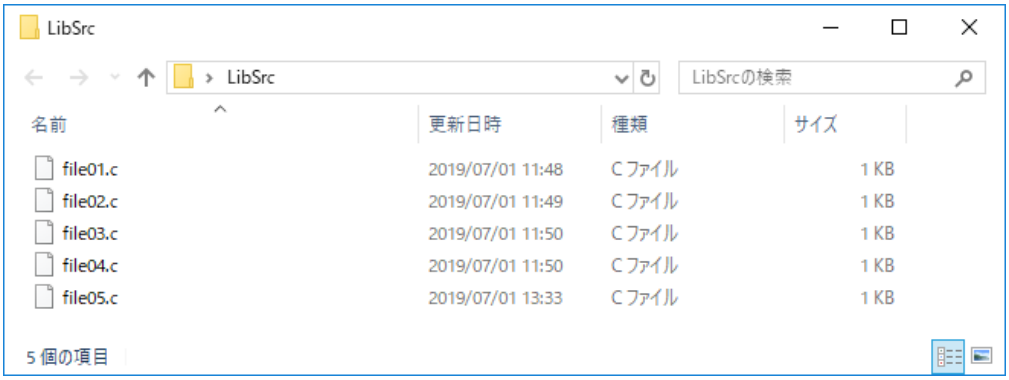


2.1. ソースファイル作成

機種に依存しないライブラリを作成する前段階として，ソースファイルを作成します。
ソースファイルを作成する場合，以下に注意してください。

- ソースファイルには，ペリフェラルを使う記述はできません。
- アセンブリ言語でソースファイルを作成する場合，スモールモデルの場合には TYPE 疑似命令に“ML621LIBS”を，ラージモデルの場合には“ML621LIBL”を指定してください。
- ライブラリ用のソースファイルは，一箇所のフォルダに格納しておいてください。
- アドレスを直接指定する記述はしないでください。

以降は，デスクトップに LibSrc フォルダを作成し，LibSrc フォルダに 5 つのライブラリ用のソースファイル（file01.c～file05.c）を格納したものとして，説明を進めていきます。



2.2. makefile 作成

テキストエディタを使ってライブラリ作成用の makefile を作成します。
以下に, makefile の記述例を示します。

記述例	参照先

# Compiler 	
CC = ccu8
CFLAGS = -TML621LIBS -MS -near -Om -Oa
	「2.2.1. コンパイラのマクロ定義」を参照してください。

# Assembler 	
RAS = rasu8
RASFLAGS = -MS -NPR
	「2.2.2. アセンブラのマクロ定義」を参照してください。

# Target file 	
LIBFILE = LIBS.lib
	「2.2.3. ライブラリファイルのマクロ定義」を参照してください。

# Object file list 	
OBJS = ¥
file01.obj¥
file02.obj¥
file03.obj¥
file04.obj¥
file05.obj
 	
ASMS := $(OBJS:.obj=.asm)
	「2.2.4. ライブラリ用オブジェクトファイルのマクロ定義」を参照してください。

# Add operation list for library 	
ADD_LIST = ¥
+file01.obj¥
+file02.obj¥
+file03.obj¥
+file04.obj¥
+file05.obj
	「2.2.5. ライブラリ作成用オペレーションリストのマクロ定義」を参照してください。

# Invoke librarian LIBFILE: $(OBJS) del $(LIBFILE) libu8 $(LIBFILE) $(ADD_LIST); del $(OBJS) del $(ASMS)	「2.2.6. ライブラリアン起動ルールの定義」を参照してください。
# Invoke assebler %.obj: %.asm $(RAS) $(RASFLAGS) $<	「2.2.7. アセンブラ起動ルールの定義」を参照してください。
# Invoke compiler %.asm: %.c $(CC) $(CFLAGS) $<	「2.2.8. コンパイラ起動ルールの定義」を参照してください。

各項目の説明を以下に示します。
CCU8 コンパイラの詳細については,『CCU8 ユーザーズマニュアル』の「3. CCU8 の起動とコマンドラインオプション」を参照してください。
RASU8 アセンブラの詳細については,『MACU8 アセンブラパッケージ ユーザーズマニュアル』の「6.3 RASU8 の操作方法」および「6.5 オプション」を参照してください。
LIBU8 ライブラリアンの詳細については,『MACU8 アセンブラパッケージ ユーザーズマニュアル』の「8.2 LIBU8 の実行」および「8.5 LIBU8 の操作」を参照してください。

2.2.1. コンパイラのマクロ定義

```
# Compiler
CC = ccu8
CFLAGS = -TML621LIBS -MS -near -Om -Oa
```

CC には ccu8 を, CFLAGS にはコンパイラのオプションを定義しています。
CFLAGS には以下を定義してください。インクルードパス指定などは, 必要に応じて追加してください。
スモールモデル用(パック機能非対応)の場合: -TML621LIBS –MS –near –Om –Oa
ラージモデル用(パック機能非対応)の場合: -TML621LIBL –ML –near –Om –Oa
スモールモデル用(パック機能対応)の場合: -TML621LIBS –MS –near –Om –Oa -Zpack
ラージモデル用(パック機能対応)の場合: -TML621LIBL –ML –near –Om –Oa -Zpack
パック機能については,『CCU8 ユーザーズマニュアル』の「11. 構造体/共用体のパック機能」を参照してください。

2.2.2. アセンブラのマクロ定義

```
# Assembler
RAS = rasu8
RASFLAGS = -MS -NPR
```

RAS には rasu8 を, RASFLAGS にはアセンブラのオプションを定義しています。
RASFLAGS には以下を定義してください。
スモールモデル用の場合: –MS –NPR
ラージモデル用の場合: –ML –NPR

2.2.3. ライブラリファイルのマクロ定義

```
# Target file
LIBFILE = LIBS.lib
```

LIBFILE には以下を定義してください。
スモールモデル用(パック機能非対応)の場合: LIBS.lib
ラージモデル用(パック機能非対応)の場合: LIBL.lib
スモールモデル用(パック機能対応)の場合: LIBSPK.lib
ラージモデル用(パック機能対応)の場合: LIBLPK.lib

2.2.4. ライブラリ用オブジェクトファイルのマクロ定義

```
# Object file list
OBJS = ¥
       file01.obj¥
       file02.obj¥
       file03.obj¥
       file04.obj¥
       file05.obj

ASMS := $(OBJS:.obj=.asm)
```

OBJS には，ライブラリ作成に依存するオブジェクトファイルのリストを定義しています。複数の行にファイルを指定する場合は，行末に‘¥’を指定してください。最後のファイルの行末には‘¥’をつける必要はありません。
ASMS には，アセンブリファイルのリストを定義しています。前述のオブジェクトファイルのリスト OBJS の拡張子を asm に置き換えたものをアセンブリファイルのリストとしています。

2.2.5. ライブラリ作成用オペレーションリストのマクロ定義

```
# Add operation list for library
ADD_LIST = ¥
       +file01.obj¥
       +file02.obj¥
       +file03.obj¥
       +file04.obj¥
       +file05.obj
```

ADD_LIST には，ライブラリ作成時のオペレーションのリストを定義しています。複数の行にオペレーションリストを指定する場合は，行末に‘¥’を指定してください。最後のファイルの行末には‘¥’をつける必要はありません。

2.2.6. ライブラリアン起動ルールの定義

```
# Invoke librarian
LIBFILE: $(OBJS)
       if exist $(LIBFILE) del $(LIBFILE)
       libu8 $(LIBFILE) $(ADD_LIST);
       del $(OBJS)
       del $(ASMS)
```

LIBU8 ライブラリアンの起動ルールを定義しています。LIBU8 を実行した後に，make 実行時に生成されたオブジェクトファイルとアセンブリファイルを削除しています。

2.2.7. アセンブラ起動ルールの定義

```
# Invoke assebler
%.obj: %.asm
       $(RAS) $(RASFLAGS) $<
```

RASU8 アセンブラの起動ルールを定義しています。フォルダに存在するアセンブリファイルを RASU8 アセンブラにてアセンブルします。

2.2.8. コンパイラ起動ルールの定義

```
# Invoke compiler
%.asm: %.c
       $(CC) $(CFLAGS) $<
```

CCU8 コンパイラの起動ルールを定義しています。フォルダに存在する C ソースファイルを CCU8 コンパイラにてコンパイルします。

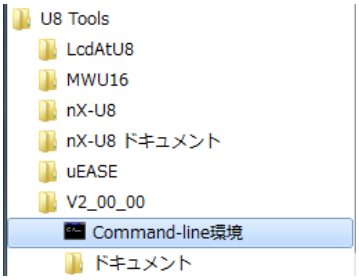
2.2.9. makefile の保存

ソースファイルと同じフォルダ（ここでは LibSrc フォルダ）に，名前を付けて makefile を保存します。

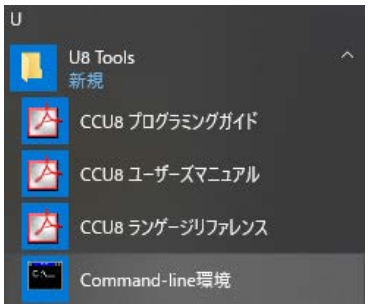
2.3. ライブラリ作成

Command-line 環境を用いてライブラリファイルを作成します。
Command-line 環境は,Windows のスタートメニューから開きます。

Windows7 の場合：[U8 Tools] > [V2_xx_xx] > [Command-line 環境]を選択
（xx はツールのバージョンによって異なります）
Windows10 の場合：[U8 Tools] > [Command-line 環境]を選択

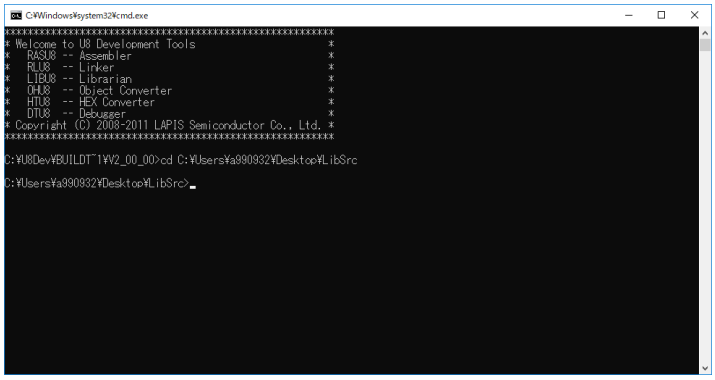


Windows7 の場合



Windows10 の場合

Command-line 環境にて，ライブラリ作成対象のフォルダに移動します。



Command-line 環境にて，make.exe を起動してライブラリファイルを作成します。
make.exe は，U8/U16 Devleopment Tools をインストールしたフォルダ（デフォルトでは C:¥U8Dev）の下の Utilities¥Bin フォルダに格納されています。
作成した makefile の名前が LIB.mk の場合，以下のようにタイプして起動してください。

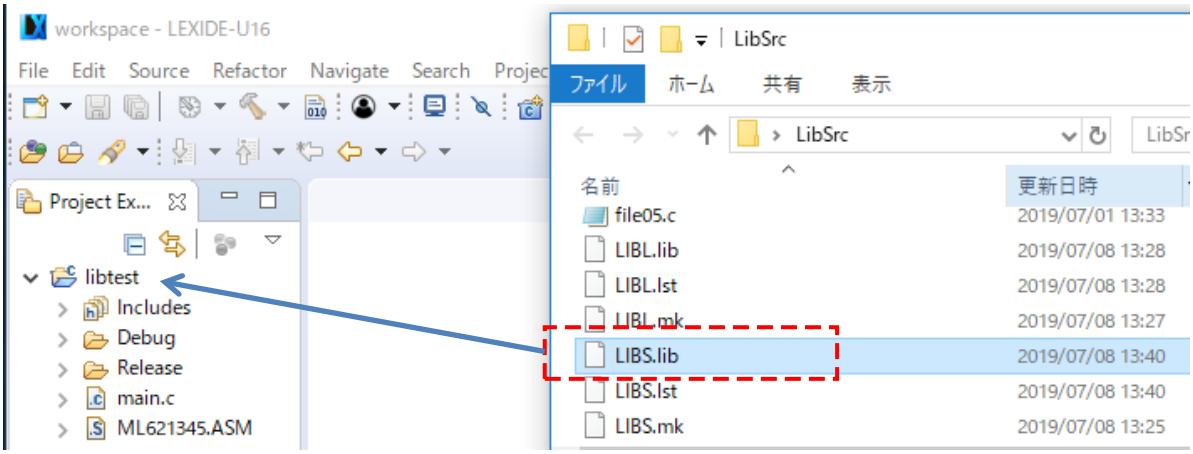
C:¥U8Dev¥Utilities¥Bin¥make –fLIB.mk

No warnings，No errors が表示されれば，ライブラリの作成完了です。

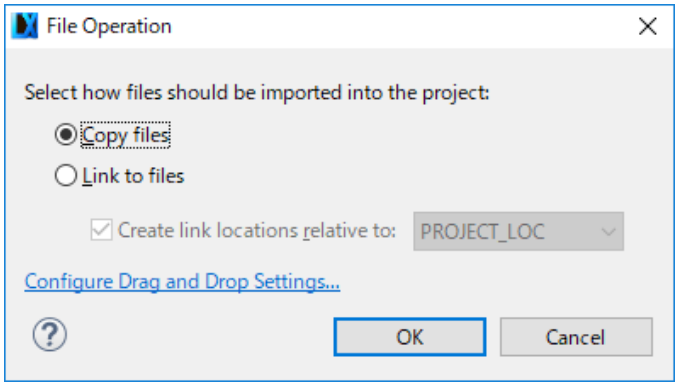
3. 作成したライブラリの使用方法

作成したライブラリを使用するには，ライブラリを使用するプロジェクトのワークスペースにライブラリファイルをコピーし，Linker のオプションでそのライブラリを指定します。

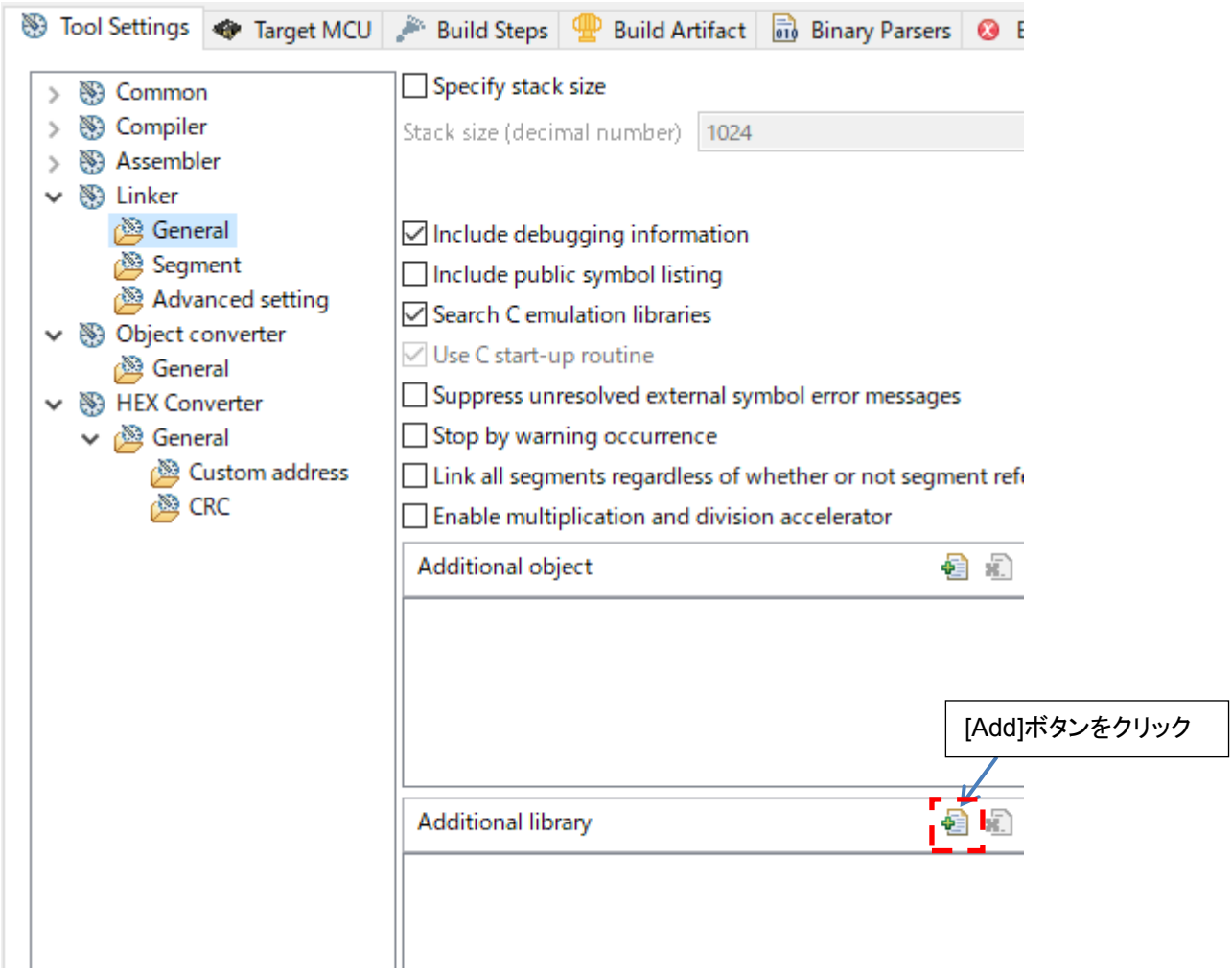
作成したライブラリを対象のプロジェクトにドラッグ＆ドロップしてコピーします。
メモリモデルおよびパック機能あり・なしに応じて，適切なライブラリを指定してください。



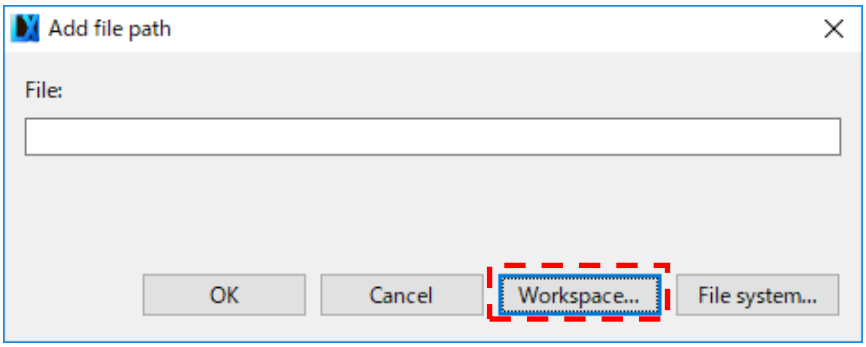
下記ダイアログが表示されますので，[Copy files]を選択して[OK]ボタンをクリックします。



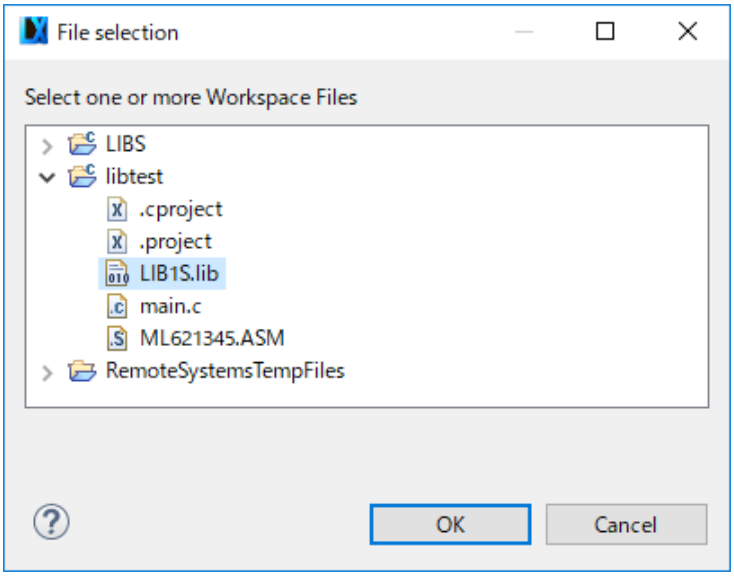
次に，プロジェクトの[Properties]ダイアログの[Tool Settings]タブの[Linker]>[General]>[Additional library]にてライブラリファイルを指定します。
[Additional library]の[Add]ボタン をクリックしします。



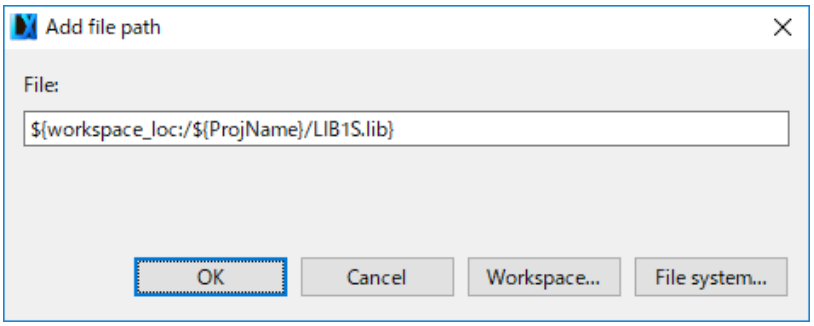
表示された[Add file path]ダイアログにて[Workspace…]ボタンをクリックします。



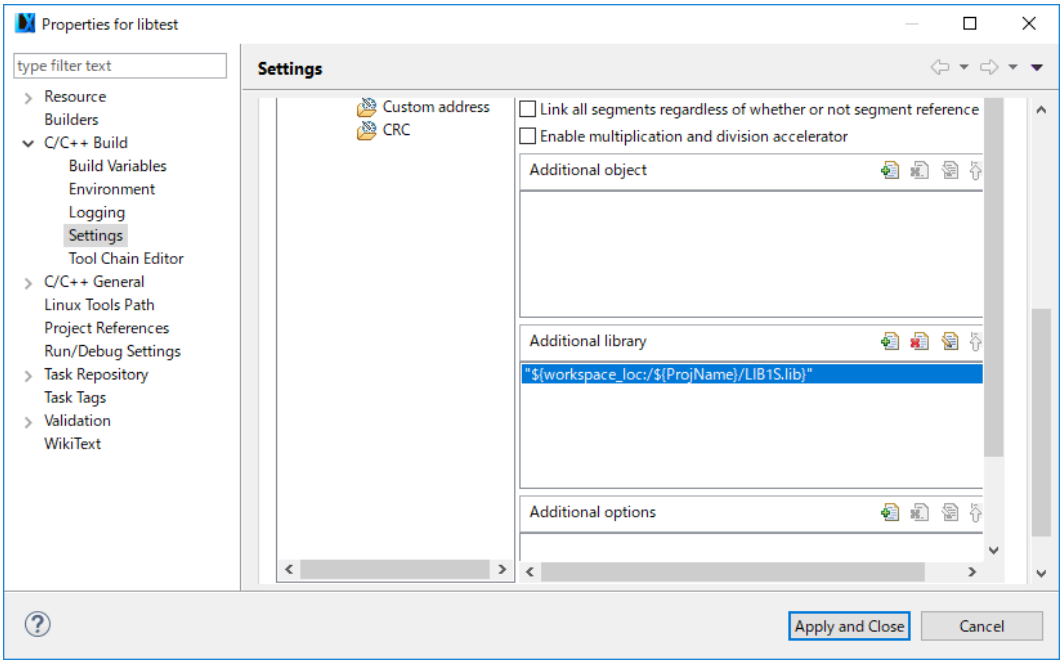
[File selection]ダイアログにて，ワークスペースにコピーしたライブラリを選択し，[OK]ボタンをクリックします。



下記ダイアログの[OK]ボタンをクリックします。

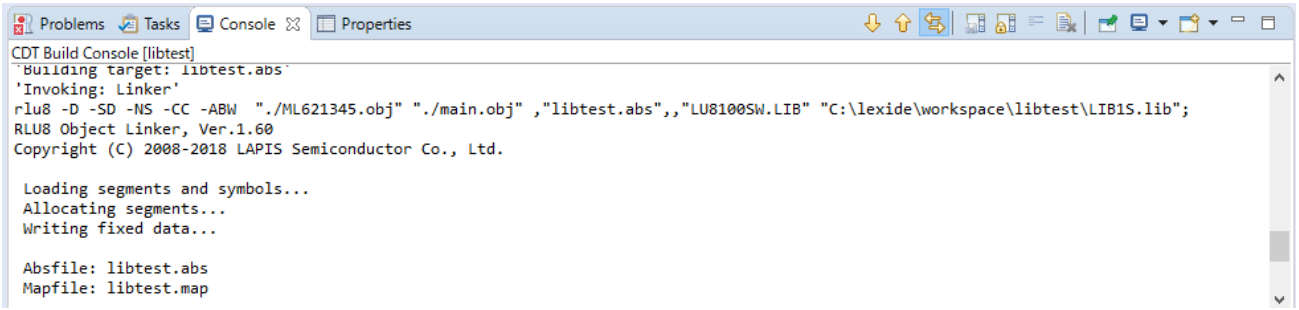


[Additional library]に指定したライブラリが追加されたことを確認し，[Apply and Close]ボタンをクリックします。



以上で，作成したライブラリファイルがプロジェクトに登録されました。

ツールバーの アイコンをクリックすると，ビルドが開始します。
指定したライブラリ（ここでは LIB1S.lib）がリンク時に参照されていることが[Console]にて確認できます。



改版履歴

ドキュメント No.	発行日	ページ		変更内容
		改版前	改版後	
FJXT_MCU_HOWTO_CREATE_LIBRARY-01	2019.7.22	ー	ー	初版発行