【講座＋実習】
T-Kernel 2.0入門

T-Engineフォーラム

2013年11月7日(木)～8日(金)

全体目次

- １日目 / 午前 (10:00～12:00)
  - T-Kernel 2.0の概要
  - T-Kernel 2.0の追加機能
- １日目 / 午後 (13:00～16:30)
  - T-Kernel 2.0 Software Packageの概要
  - エミュレータを使った実習と解説（１）
- ２日目 (10:00～16:30)
  - エミュレータを使った実習と解説（２）

T-Kernel 2.0
Software Package

T-Kernel 2.0 Software Package

- T-Kernel 2.0を利用した開発に必要となるソフトウェアを一括して提供
  - T-Engineフォーラムのwebページから入手可能

- パッケージ内容
  - ソースコード
  - 開発環境
  - 各種資料
  - ucode

パッケージ内容 / ソースコード

- T-Kernel 2.0
- T-Monitor
- 各種デバイスドライバ
  - 時計 (RTC)
  - シリアルコンソール
  - タッチパネルおよび押しボタンスイッチ
  - LCD
  - システムディスク (microSD用)

パッケージ内容 / 開発環境

- Eclipse
- Cygwin
  - Windows上で動作
- GNU開発環境
  - Cygwin用
  - Linux用
- エミュレータ

パッケージ内容 / その他

- 実装仕様書
- 開発環境セットアップ用ドキュメント
- ターゲット関連資料
  - T-Engineリファレンスボード
  - エミュレータの説明
- T-License 2.0
- ucode

Installing T-Kernel 2.0
（Version 2.01.03対応）

Installing T-Kernel 2.0

- T-Kernel 2.0 Software Packageの設定
  - Cygwin + Eclipse or Linux (CLI)
  - T-Kernel 2.0ソースコード
  - Qemu
- 参考
  - TRONWARE Vol.140
    - 特集　春から始めるT-Kernel 2.0
      - T-Kernel 2.01.03 対応のインストール手順の説明
  - Interface 2012年5月号
    - 特集　第3章　QEMUを使ったT-Kernel 2.0とLCDのエミュレート

T-Kernel 2.0 Software Packageの設定

- **ReadMe.txt を確認**
  - T-Kernel 2.0全体の説明
  - 4.1 開発環境のインストール
    - (1) Eclipseで開発する場合

→ srcpkg/doc/ja/gcc_setup_guide_cygwin.txt

- GNU開発環境・インストール手順(Cygwin)
  1. Cygwinシステムのインストール
  2. Cygwinインストール後の設定

→ srcpkg/doc/ja/eclipse_setup_guide.txt

- Eclipse開発環境・インストール手順

  - 4.2 T-Kernelソースコードパッケージの展開
    - srcpkg/tkernel_source.tar.gz の展開
  - 4.3 エミュレータでの実行
    - emulator/tef_em1d/readme.txt

Cygwinのインストール

- srcpkg/doc/ja/gcc_setup_guide_cygwin.txt

1. Cygwinシステムのインストール
   - develop/cygwin_d-1.7.7-1.zipをC:¥cygwin_installなどに展開
   - setup.exeを実行
2. Cygwinインストール後の設定
   - Cygwin.batの設定
   - perl用のシンボリックリンクの設定
   - .bashrcの設定
   - Tips
     - Windowsで環境変数HOMEを使っている場合
     - ショートカットの設定
     - コマンドライン用の環境変数も .bashrc に追加
3. ディレクトリ作成
   - mkdir /usr/local/tef_em1d
4. 開発環境パッケージの展開
   - tar zxvf te.Cygwin-i686.*.tar.gz
   - Tips
     - *.tar.gz は Cygwinコンソールで展開
5. 環境変数の設定
   - rom-dbg.bin の生成に必要なので設定しておく → .bashrc

Cygwinの環境変数の設定

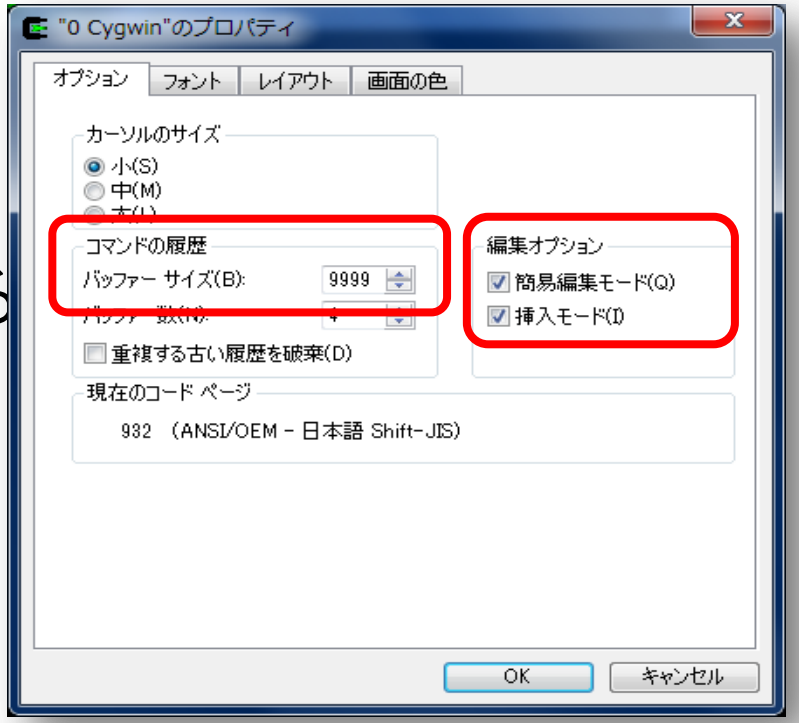
- T-Kernel 2.0用の環境変数を設定
  - /home/[ユーザ名]/.bashrc で以下を定義

```
# for T-Kernel 2.0
export GNU_BD=/usr/local/tef_em1d/tool/Cygwin-i686
export GNUARM_2=$GNU_BD/arm_2-unknown-tkernel
export BD=/usr/local/tef_em1d/tkernel_source
```

- Cygwinでのビルドが可能となる。
  - Eclipseでのビルドで発生した問題の切り分けに利用可能
  - rom-dbg.bin の生成(準備)に必要

Cygwinコンソールの設定

- 簡易編集モードを有効にしておくことで操作性を改善
  → Copy & Paste が可能
- 設定手順
  1. ショートカットのプロパティを開く
  2. 「オプション」タブを選択
  3. 「編集オプション」のに含まれている
     「簡易編集モード」にチェックを追加
  4. [　OK　]
- その他
  - 「画面バッファのサイズ」も拡大



Eclipseのインストール

- srcpkg/doc/ja/eclipse_setup_guide.txt

1. Cygwin環境のインストール ← 済
2. Javaのインストール
   - http://www.java.com/ja/download/windows_xpi.jsp
3. Eclipseのインストール
   - 以下のzipファイルをc:¥eclipseに展開
     - eclipse-cpp-indigo-SR2-incubation-win32.zip (Eclipse + CDT)
     - pleiades_x.x.x.zip (Eclipse日本語化プラグイン)
4. T-Kernel開発用プラグインのインストール
   - 以下のzipファイルをc:¥eclipseに展開
     - develop/org.t_engine.te.x.x.x.zip
     - develop/org.t_engine.tl.tef_em1d.x.x.x.zip
   - Tips : ZIPの展開はWindowsで普通に操作可能
5. eclipse.iniの変更 (日本語化)
   - c:¥eclipse¥eclipse.ini の最後に以下を追加

```
-javaagent:plugins/jp.sourceforge.mergedoc.pleiades/pleiades.jar
```

T-Kernelソースのインストール

- 4.2 T-Kernelソースコードパッケージの展開
  - /usr/local/tef_em1d に srcpkg/tkernel_source.tar.gz を展開

- ファイル構成
  - /usr/local/tef_em1d
    - C:¥cygwin¥usr¥local¥tef_em1d
  - /usr/local/tef_em1d/tool
    - C:¥cygwin¥usr¥local¥tef_em1d¥tool
  - /usr/local/tef_em1d/tkernel_source
    - C:¥cygwin¥usr¥local¥tef_em1d¥tkernel_source

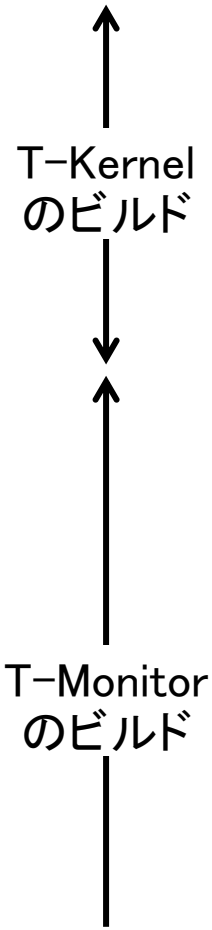
エミュレータのインストール

- 4.3 エミュレータでの実行
  - emulator/tef_em1d/readme.txt
- 以下を C:¥qemu にコピー
  - emulator/tef_em1d/bin
- ショートカットを作成 ← 引数を追加
  - C:¥qemu¥qemu.bat rom.bin sd.img

Eclipse用に rom-dbg.bin を生成

- QEMU-tef_em1d のビルド
  - rom-dbg.bin の生成
- 生成手順
  1. Cygwinのコンソールを起動
  2. 次ページからのビルド用コマンドを順に入力
     - T-Kernelをビルドする。
     - T-Monitorをビルドする。
     - デバッグ用のROMイメージを生成する。
  3. 生成されたrom-dbg.binをC:¥qemuにコピー
  4. qemu.batのショートカットの引数を変更
     - C:¥qemu¥qemu.bat rom-dbg.bin sd.img

ビルド手順 (1/2)




```
cd ${BD}/lib/build/tef_em1d
make                    ← rominfoをビルド
cd ${BD}/config/build/tef_em1d
make                    ← ライブラリをビルド
cd ${BD}/kernel/sysmain/build/tef_em1d
make                    ← T-Kernel本体をビルド
cd ${BD}/monitor/cmdsvc/build/tef_em1d
make                    ← コマンド・SVC処理部をビルド
make install
cd ${BD}/monitor/hwdepend/tef_em1d/build
make                    ← システム依存部をビルド
make install
cd ${BD}/monitor/driver/flash/build/tef_em1d
make                    ← FlashROMドライバのビルド
make install
```

ビルド手順（2/2）

T-Monitor
のビルド
（続き）

```
cd ${BD}/monitor/driver/memdisk/build/tef_em1d
make                          ← memdiskドライバのビルド
make install
cd ${BD}/monitor/driver/sio/build/tef_em1d
make                          ← SIOドライバのビルド
make install
cd ${BD}/monitor/tmmain/build/tef_em1d
make                          ← T-Monitor本体のビルド
make install
```

ロードイメージ
のビルド

```
cd ${BD}/kernel/sysmain/build/tef_em1d
make emu                      ← RAM版kernelの生成
cp rom-dbg.bin /cygdrive/c/qemu/bin
                              ← Qemuのディレクトリへ
```

準備完了

↓

実行

ビルド、ロード、実行

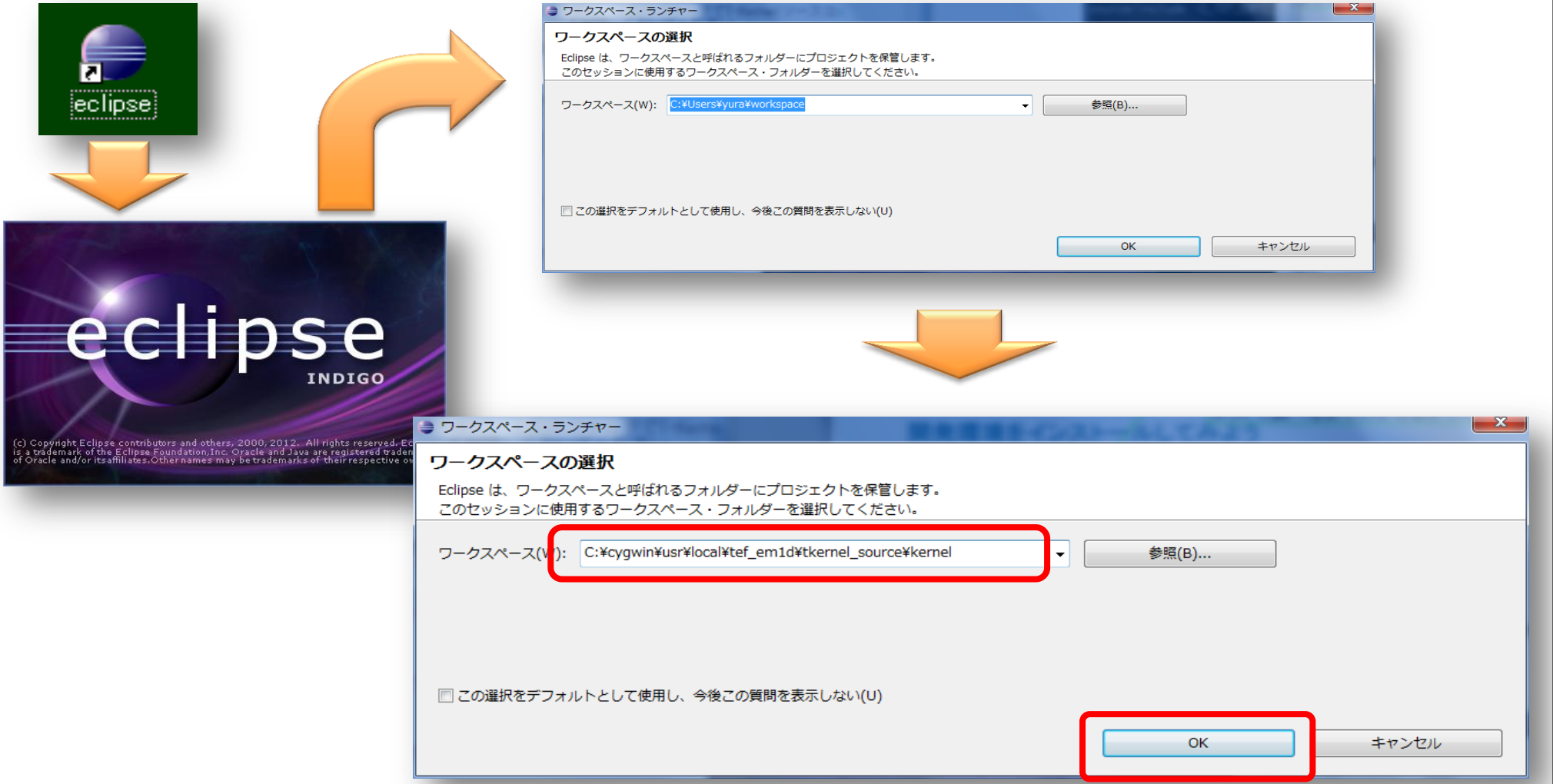
- Eclipseの起動
- ビルド
- エミュレータ用の設定 → Eclipse
- エミュレータの起動
- Eclipseをエミュレータに接続
  - T-Monitorのコマンドを試してみる。
- ロードして実行
- デバッグ

Eclipseの起動

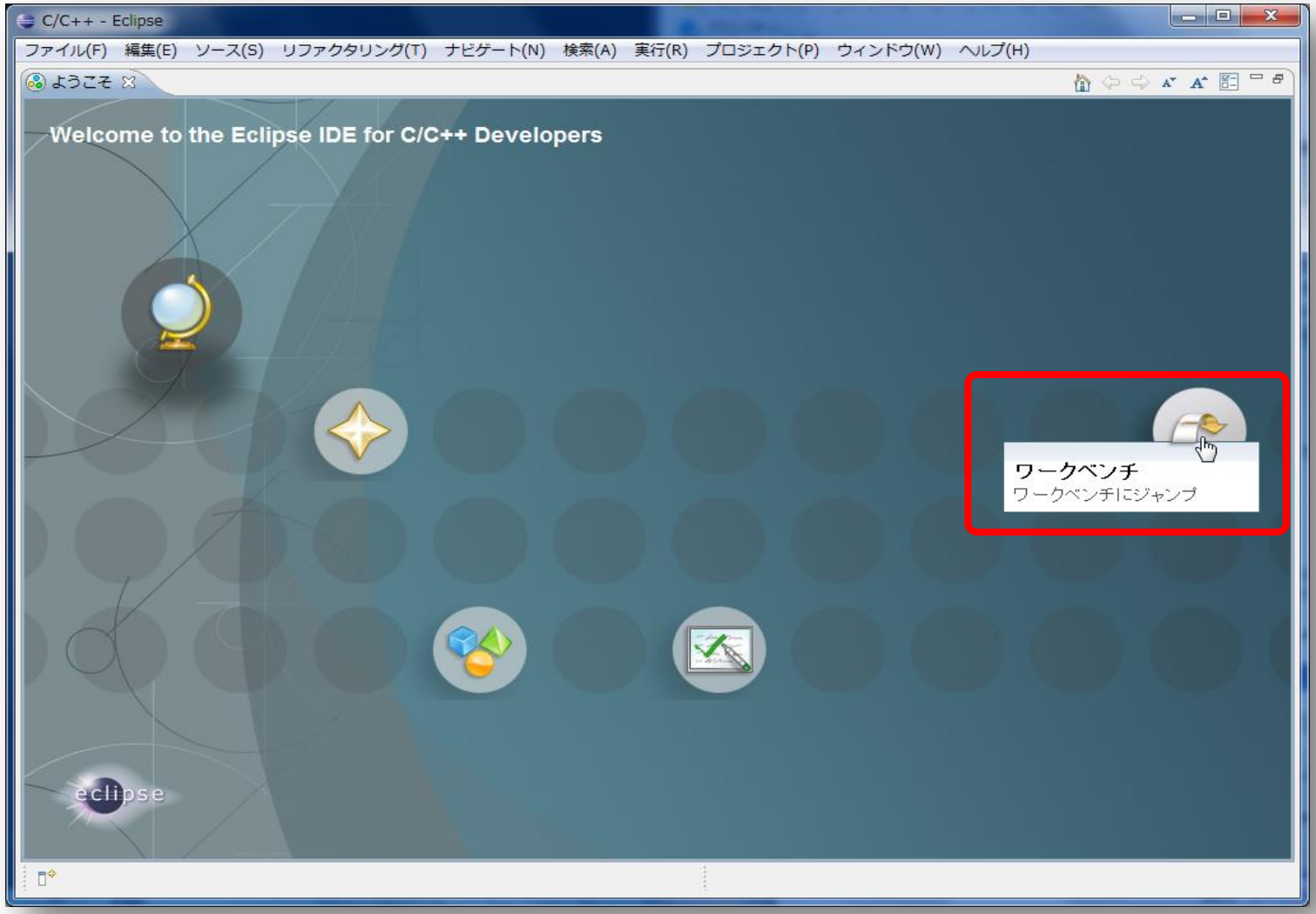
Eclipseの起動

- Eclipseを起動
- ワークスペースとしてkernelを設定
  - その他は直接利用しないのでとりあえずは設定不要
    - rom-dbg.bin生成時にCygwinでコンパイル済
- ビルドの設定
- ビルド
- ロードして実行
  - 「デバッグの構成」

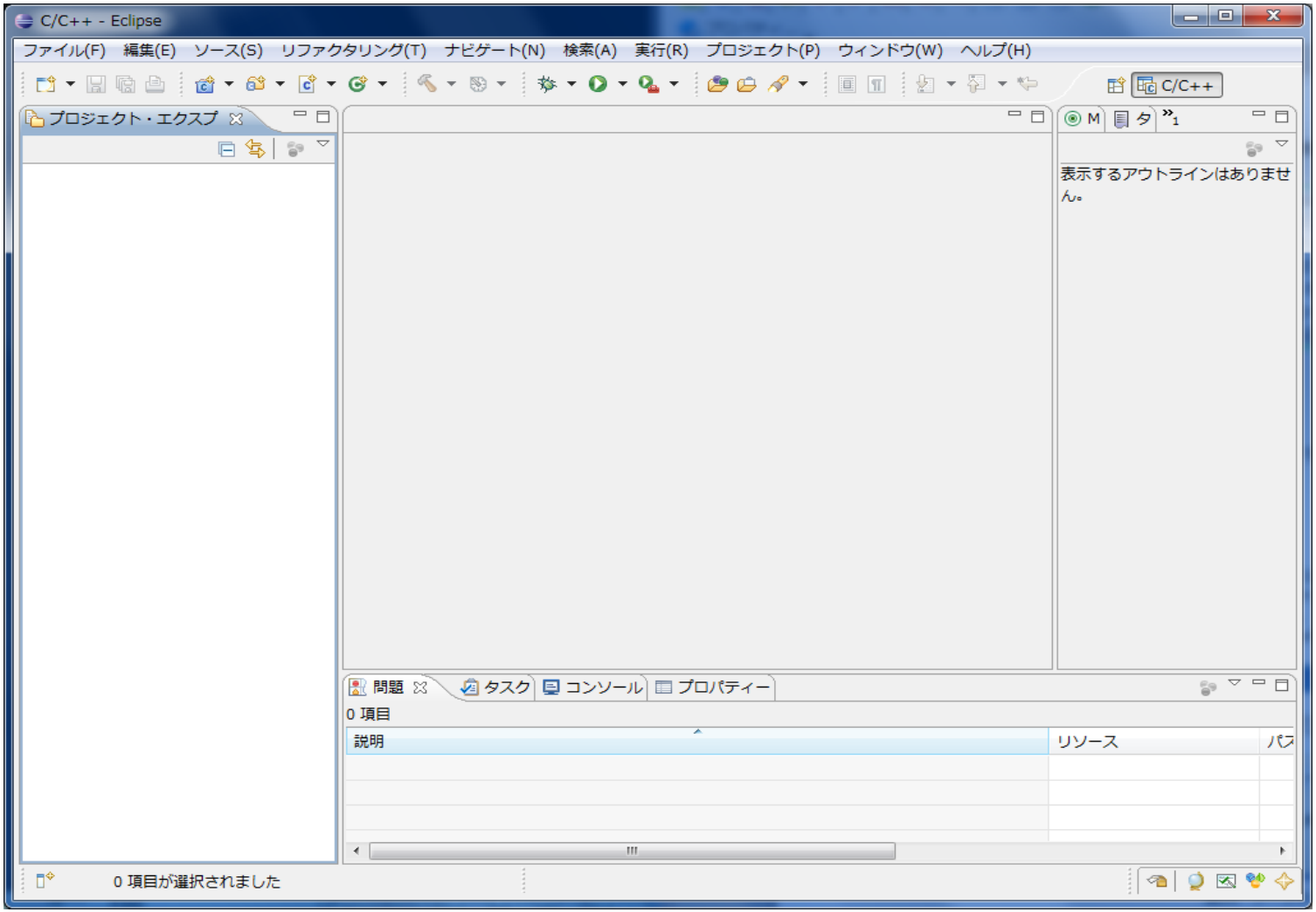
Eclipseの起動 (1/15)



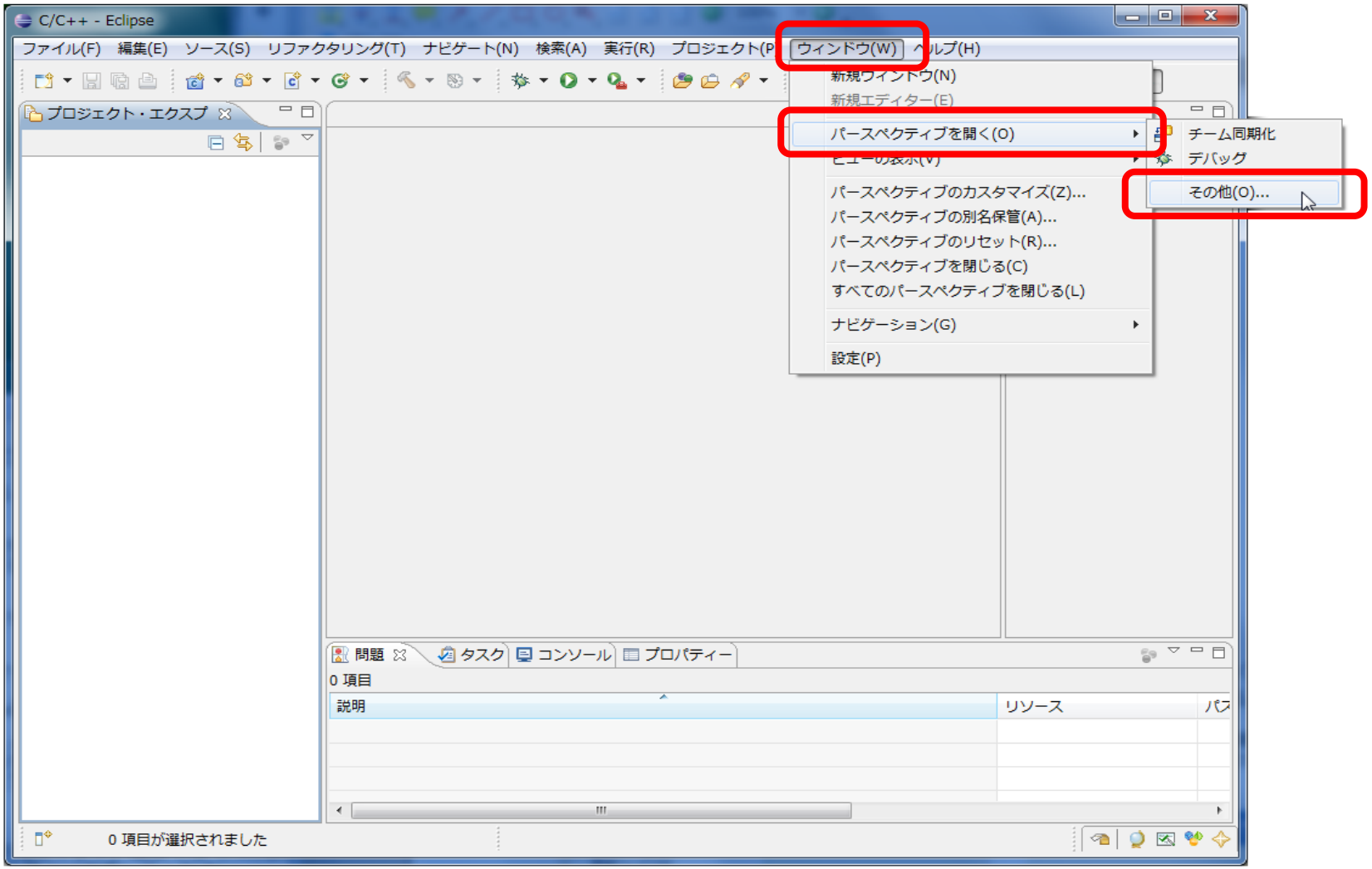
Eclipseの起動 (2/15)



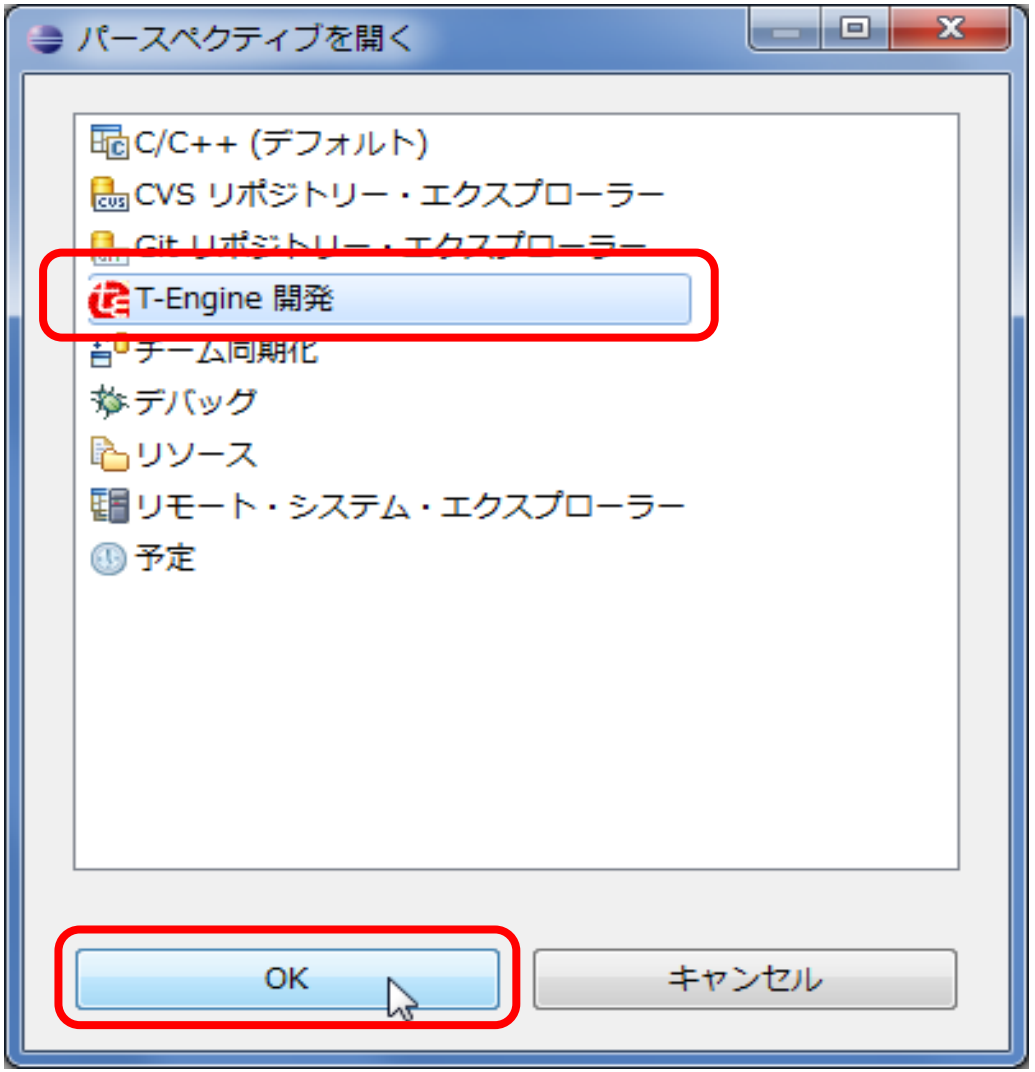
Eclipseの起動 (3/15)



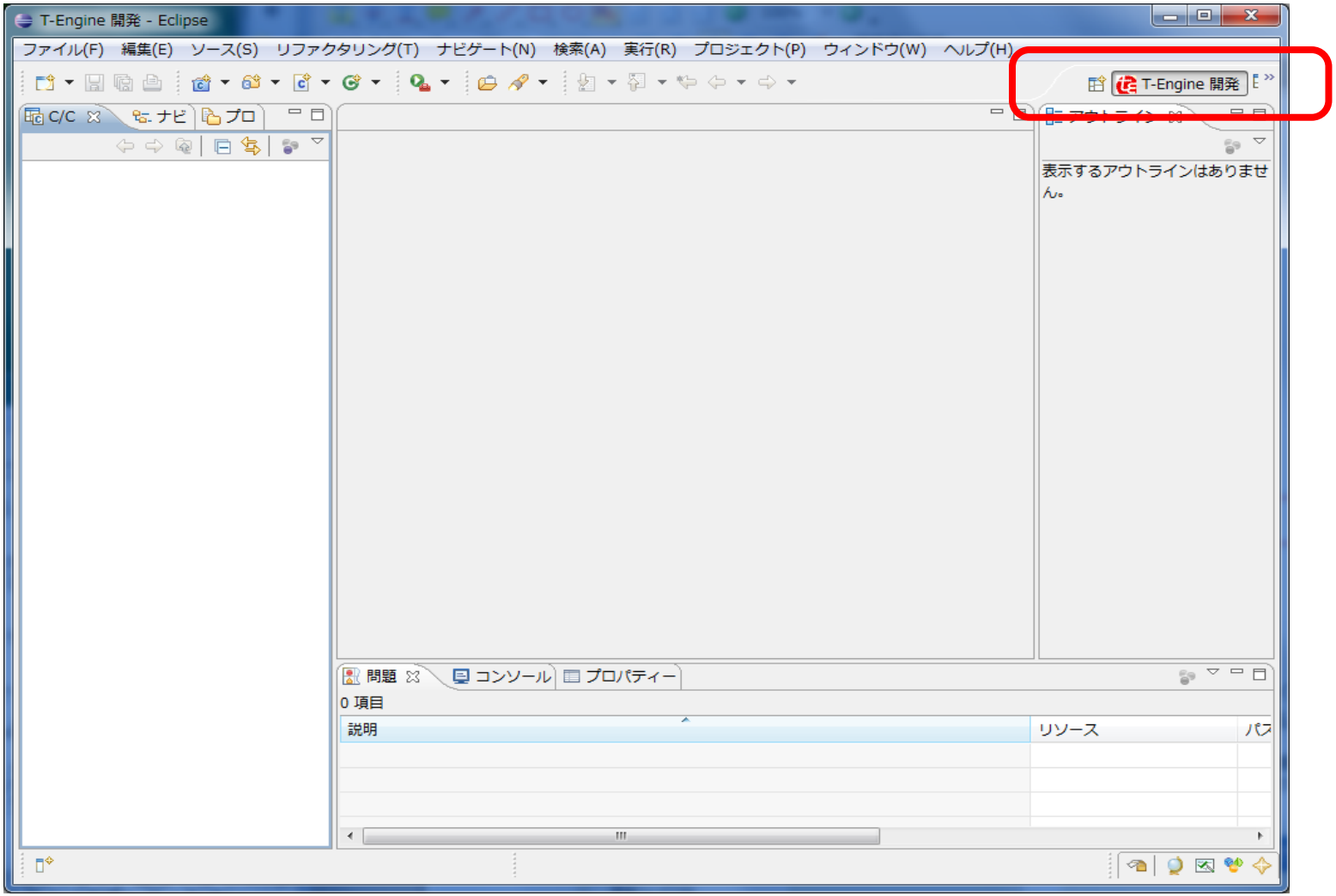
Eclipseの起動 (4/15)



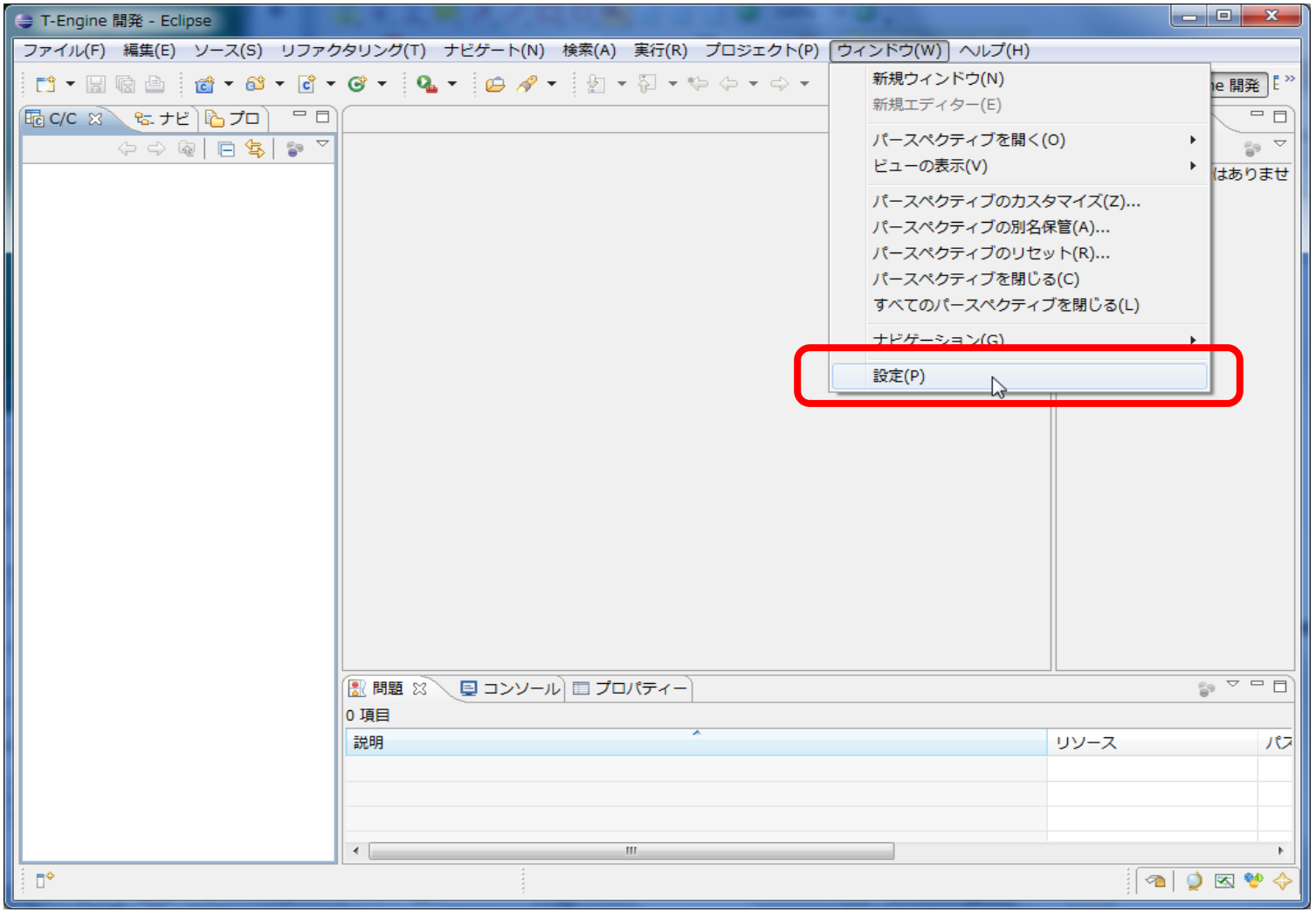
Eclipseの起動 (5/15)



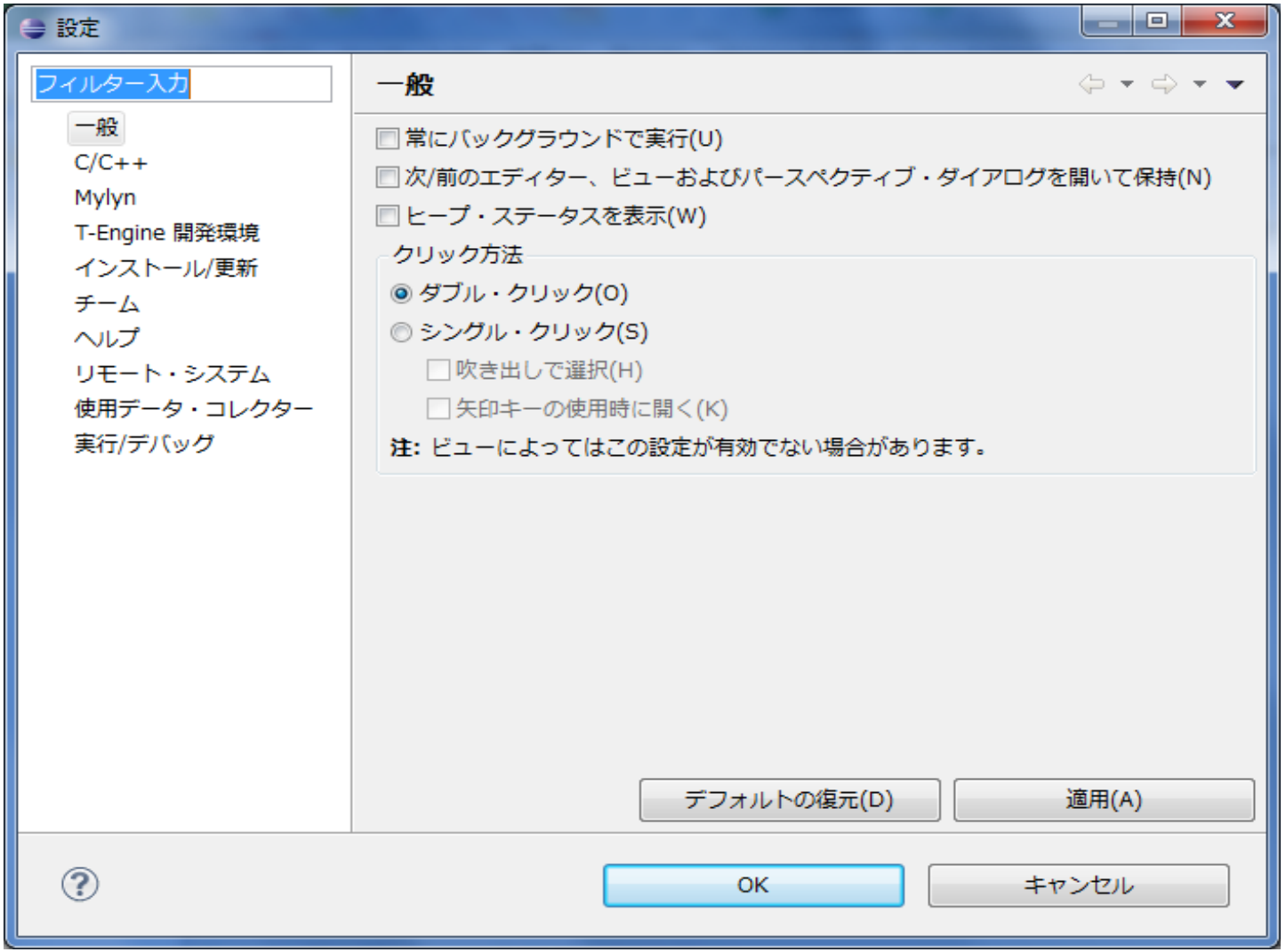
Eclipseの起動 (6/15)



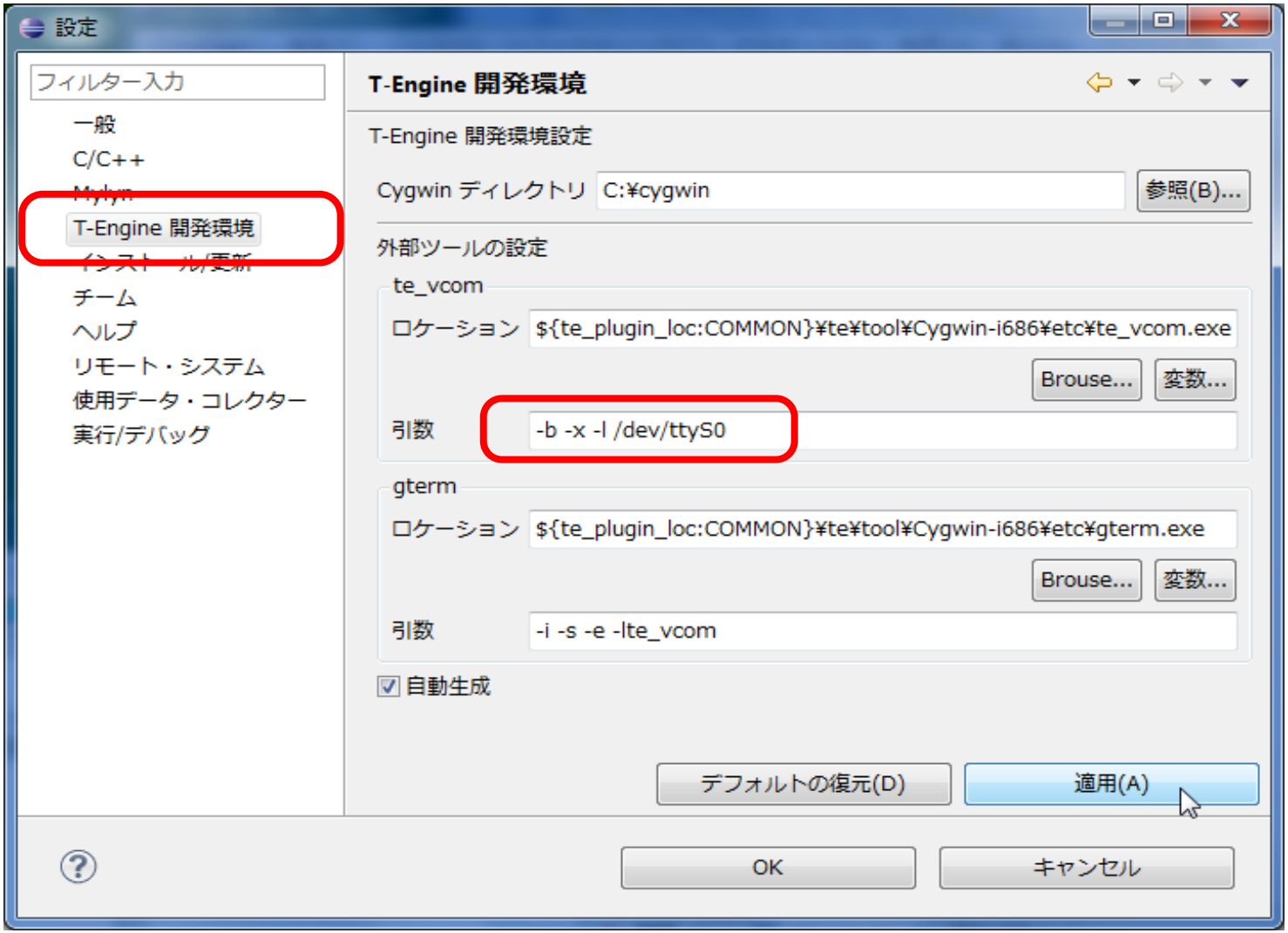
Eclipseの起動 (7/15)



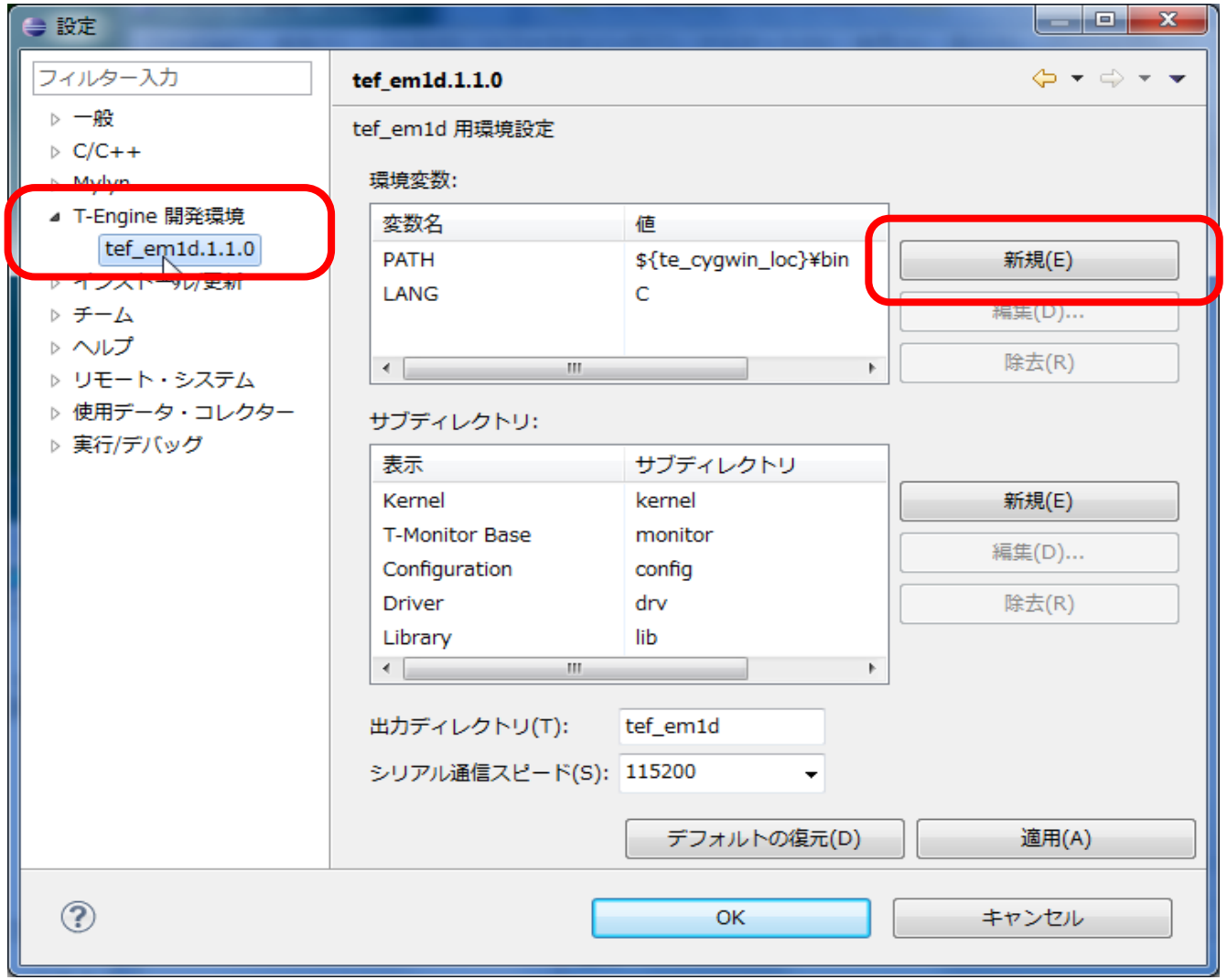
Eclipseの起動 (8/15)



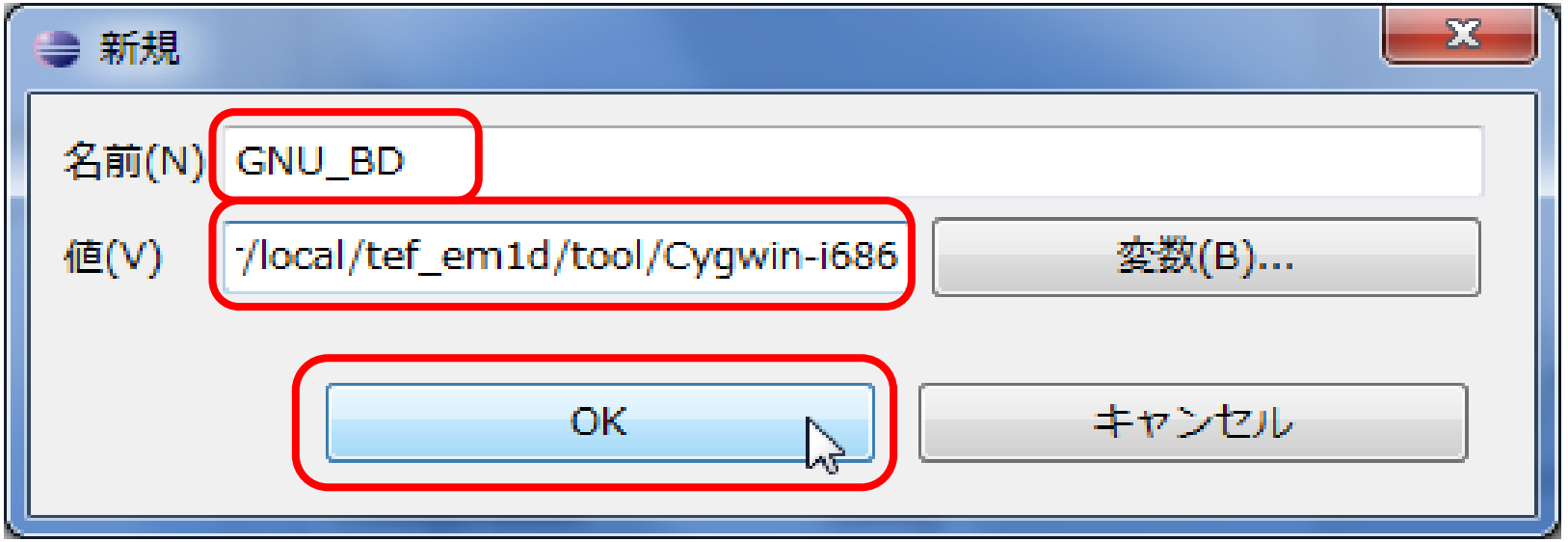
Eclipseの起動 (9/15)



Eclipseの起動（10/15）

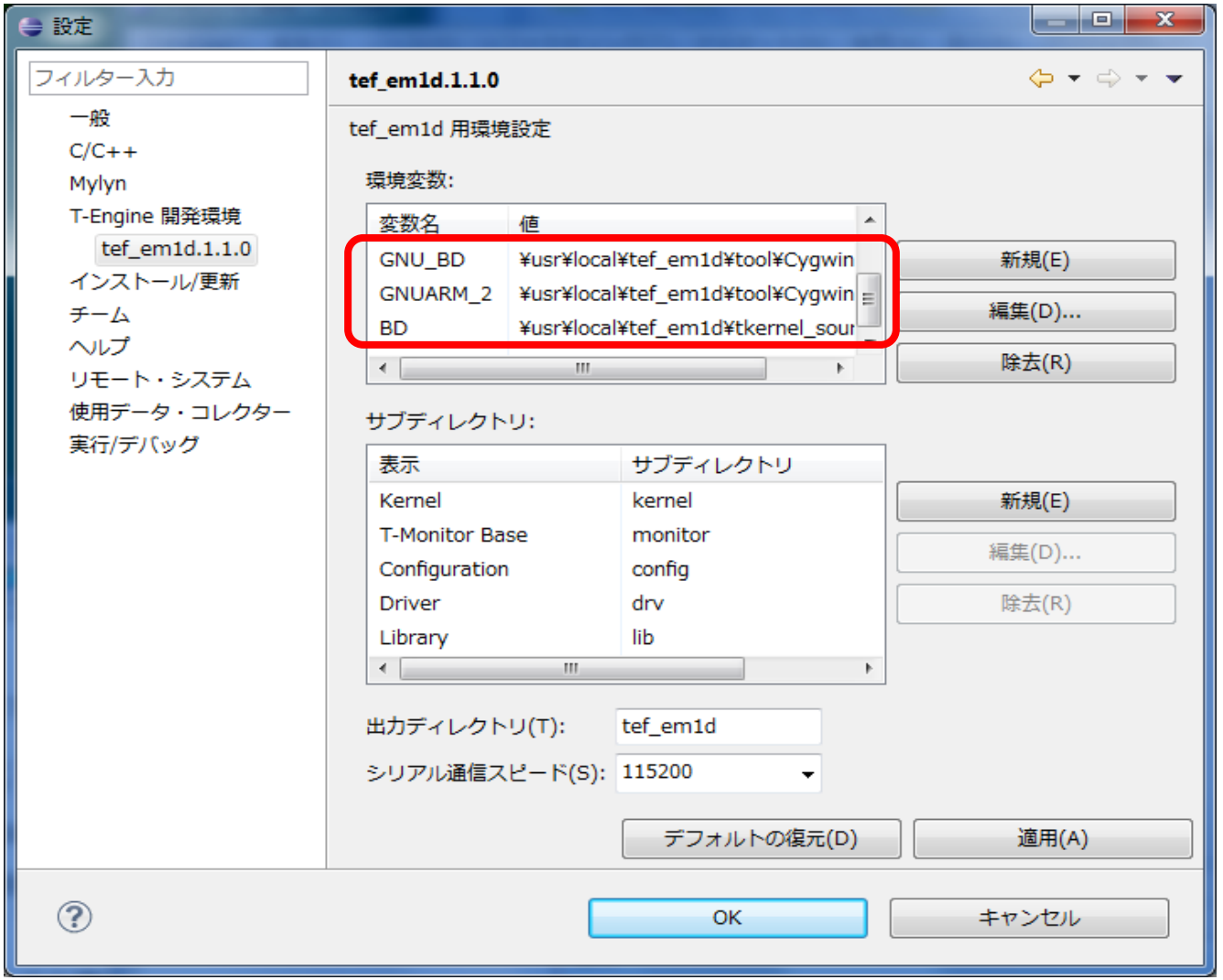


Eclipseの起動（11/15）

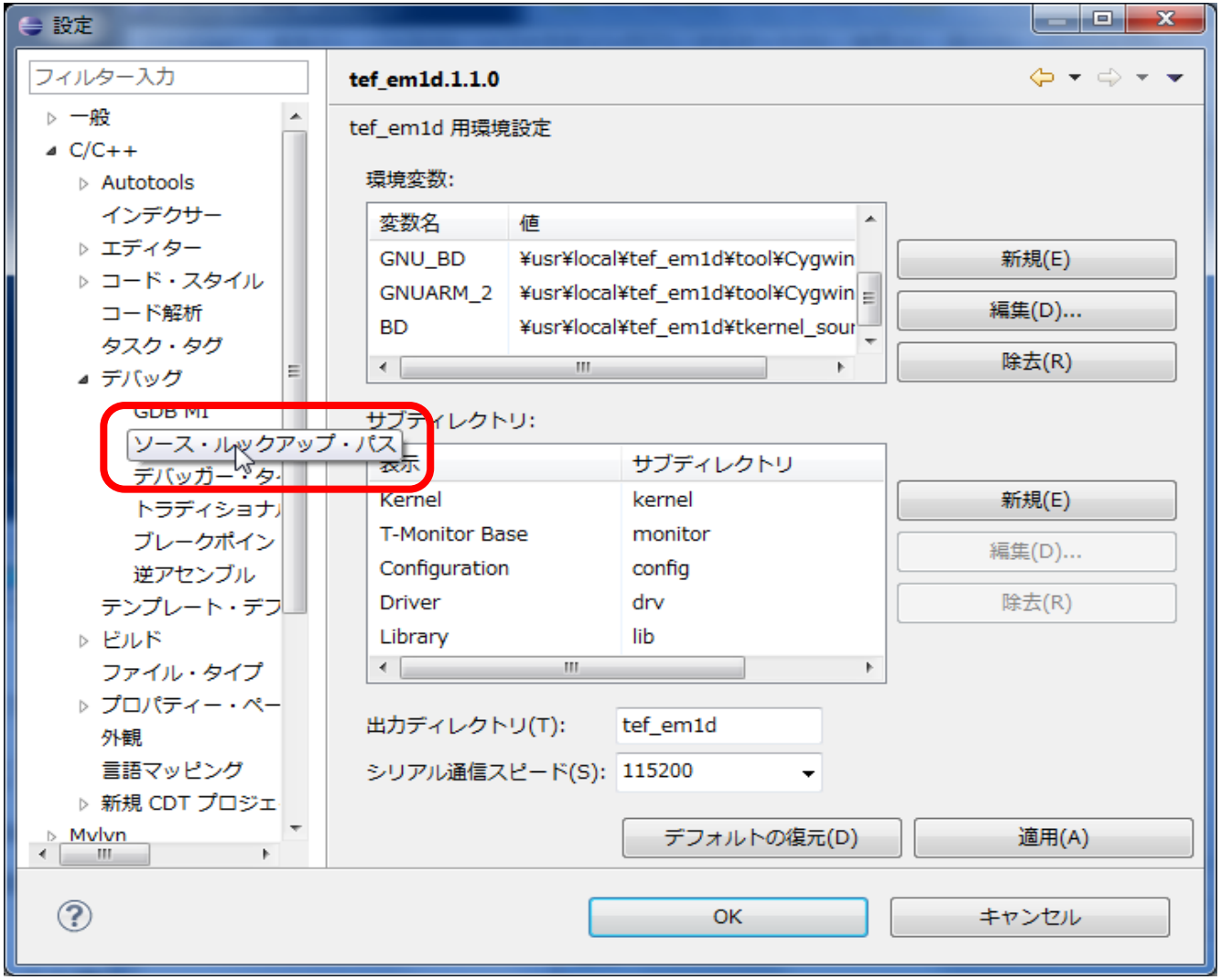


GNU_BD	/usr/local/tef_em1d/tool/Cygwin-i686
GNUARM_2	/usr/local/tef_em1d/tool/Cygwin-i686/arm_2-unknown-tkernel
BD	/usr/local/tef_em1d/tkernel_source

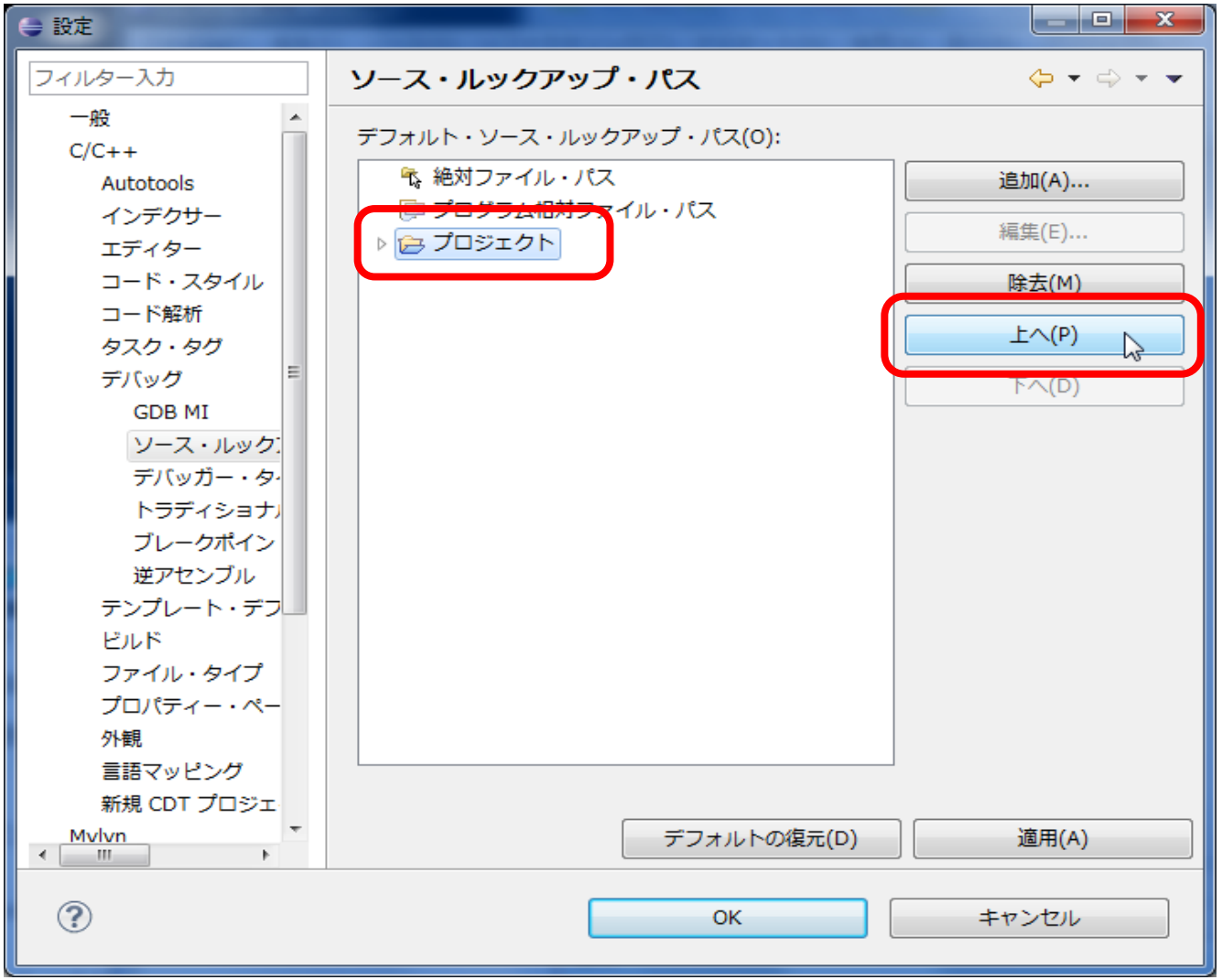
Eclipseの起動（12/15）



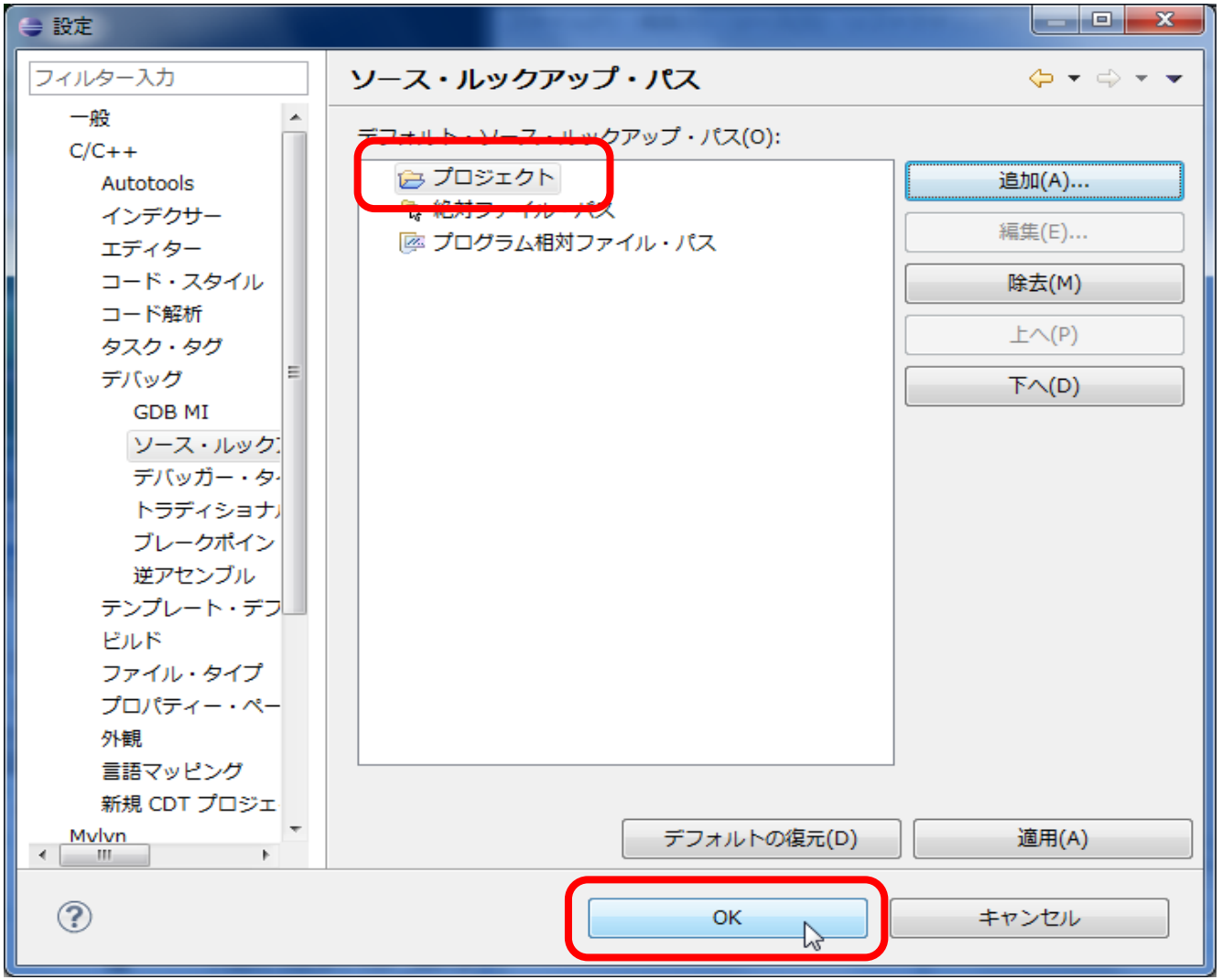
Eclipseの起動（13/15）



Eclipseの起動（14/15）

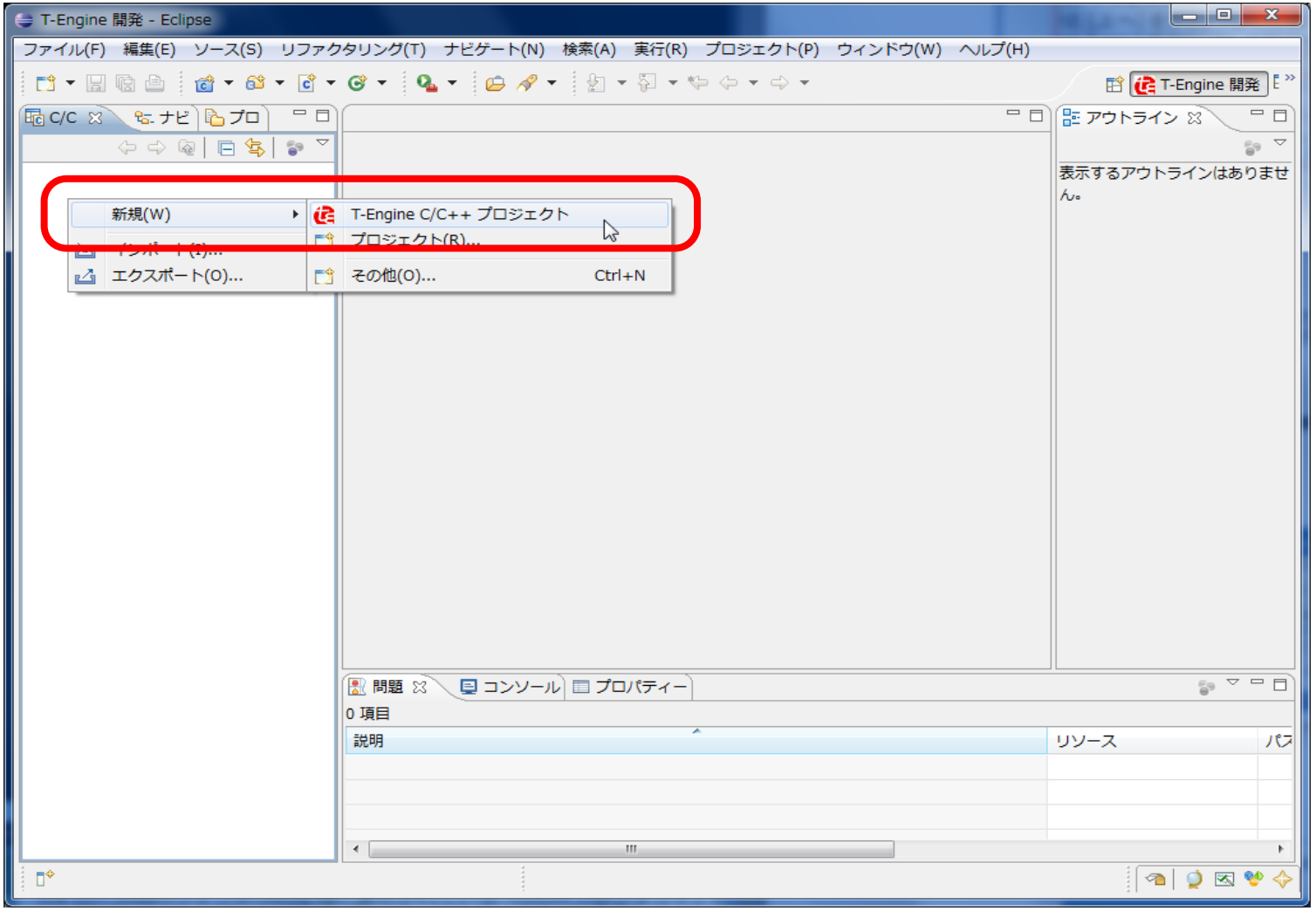


Eclipseの起動（15/15）

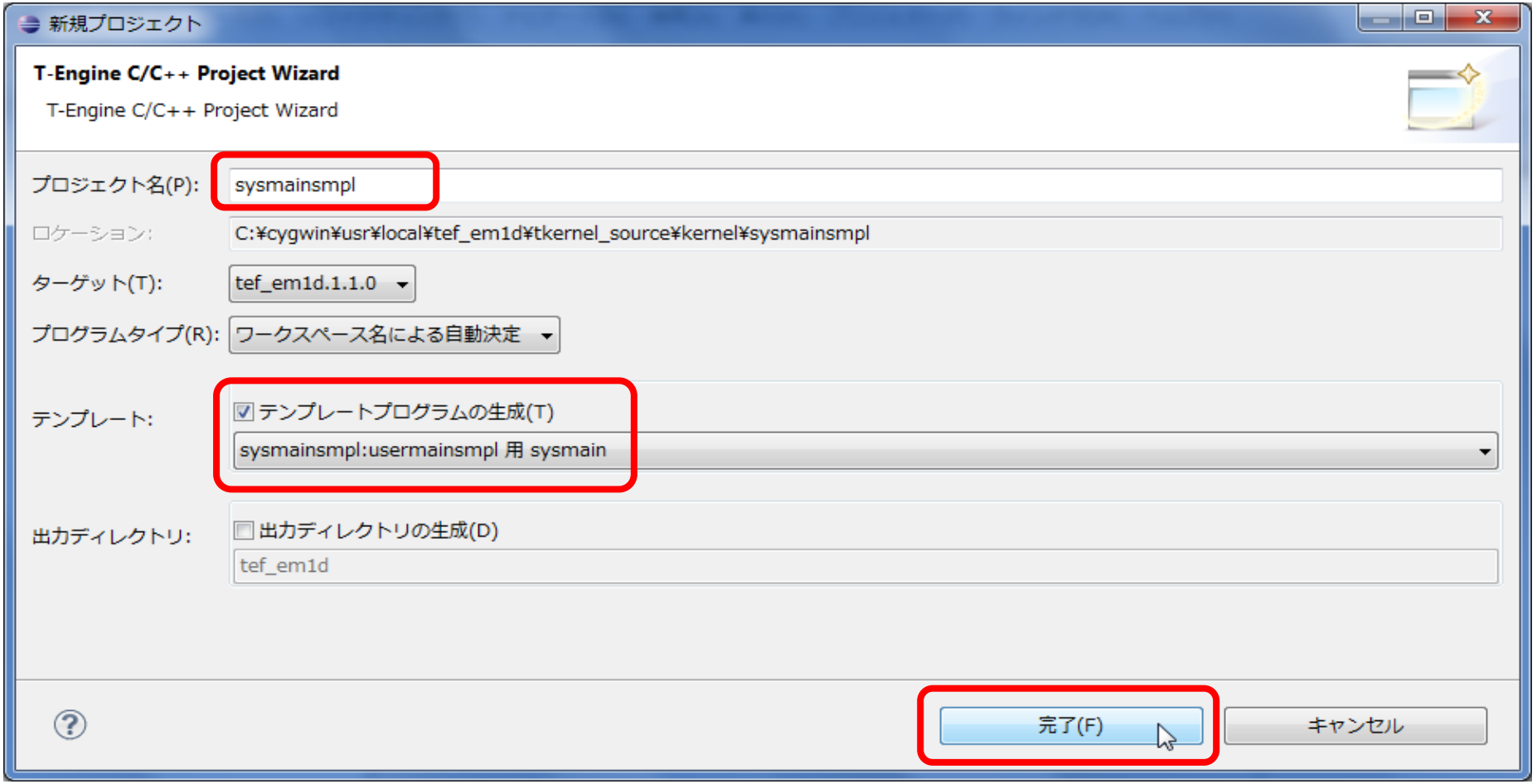


プロジェクトの設定

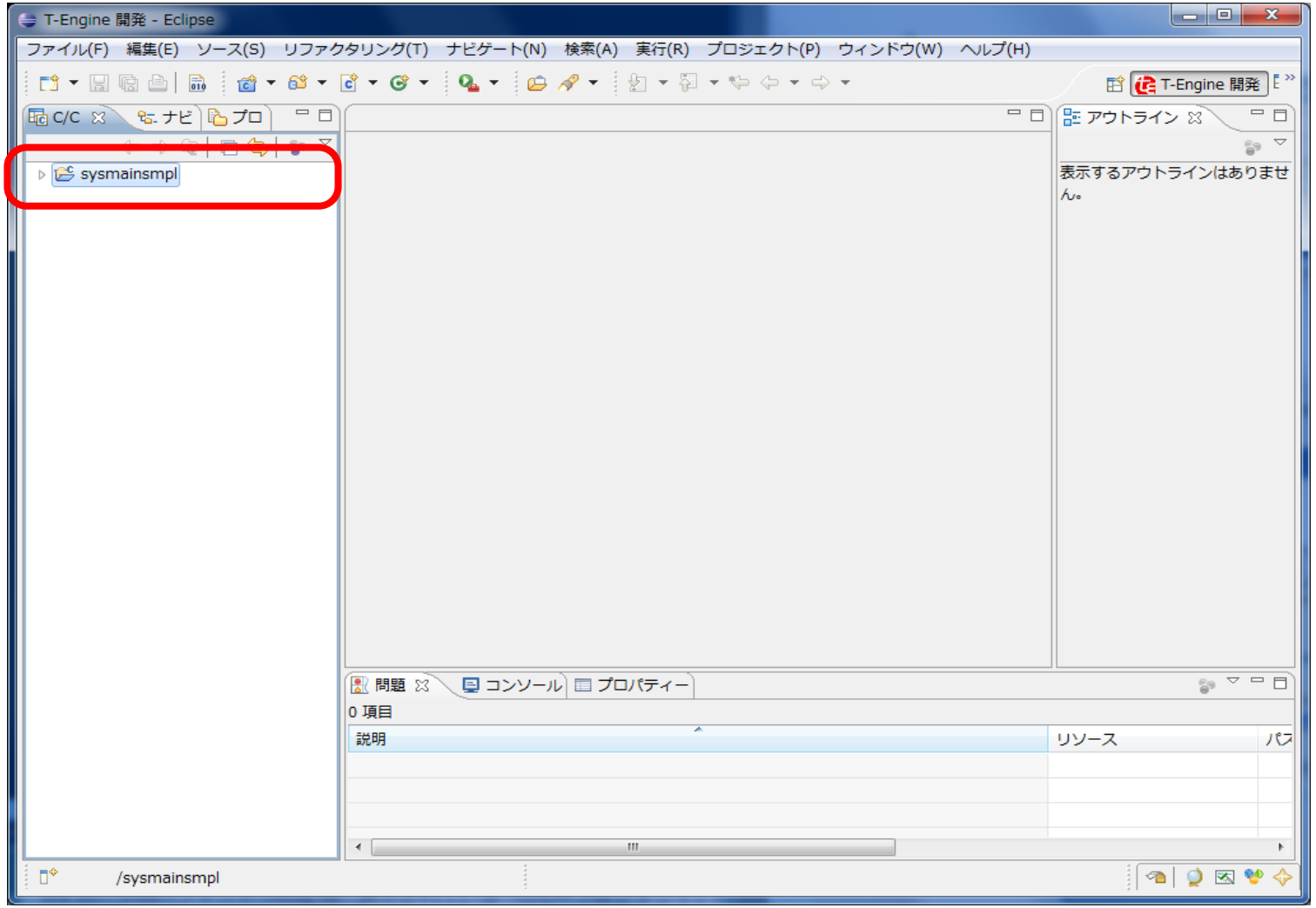
プロジェクトの設定 (1/9)



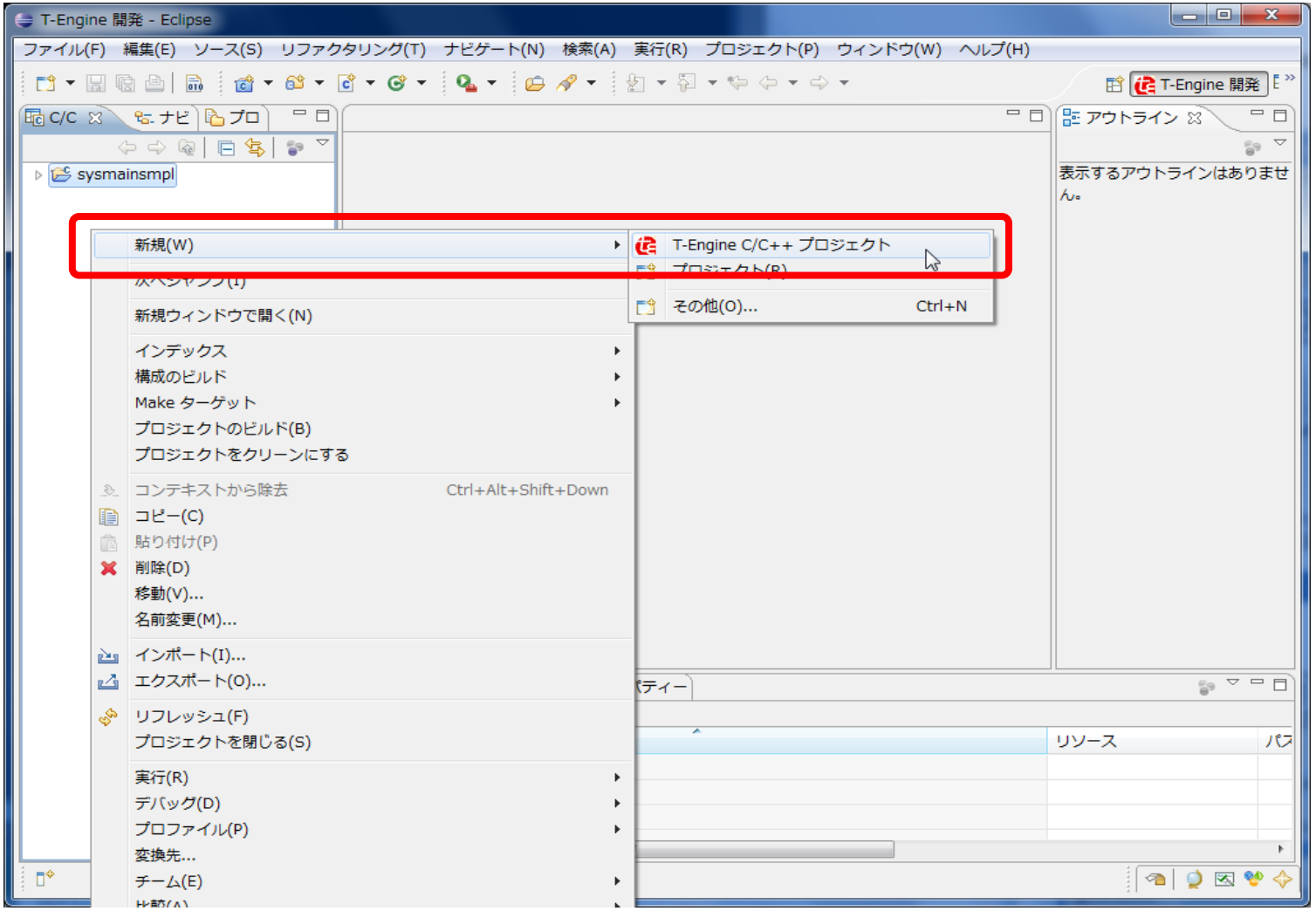
プロジェクトの設定 (2/9)



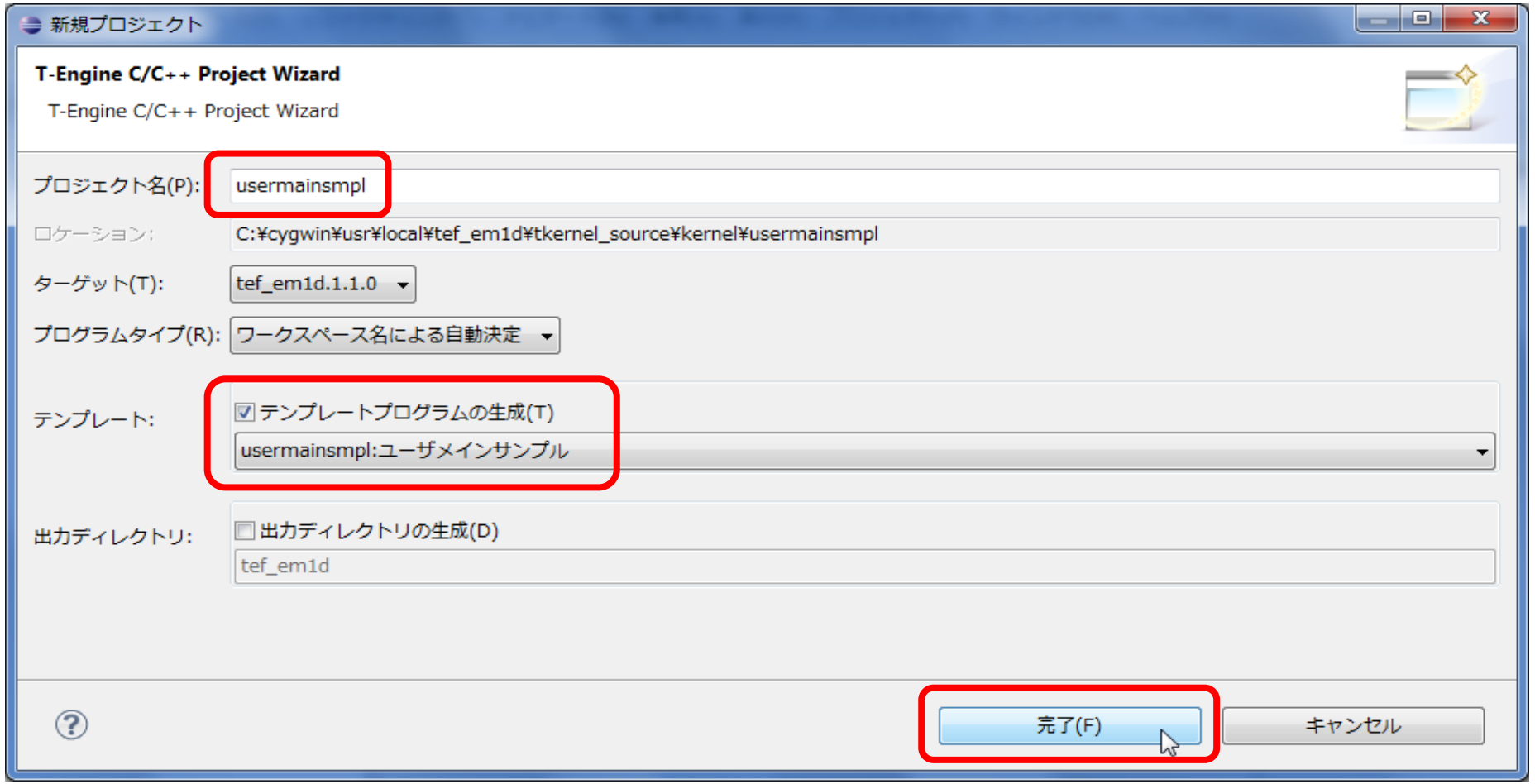
プロジェクトの設定 (3/9)



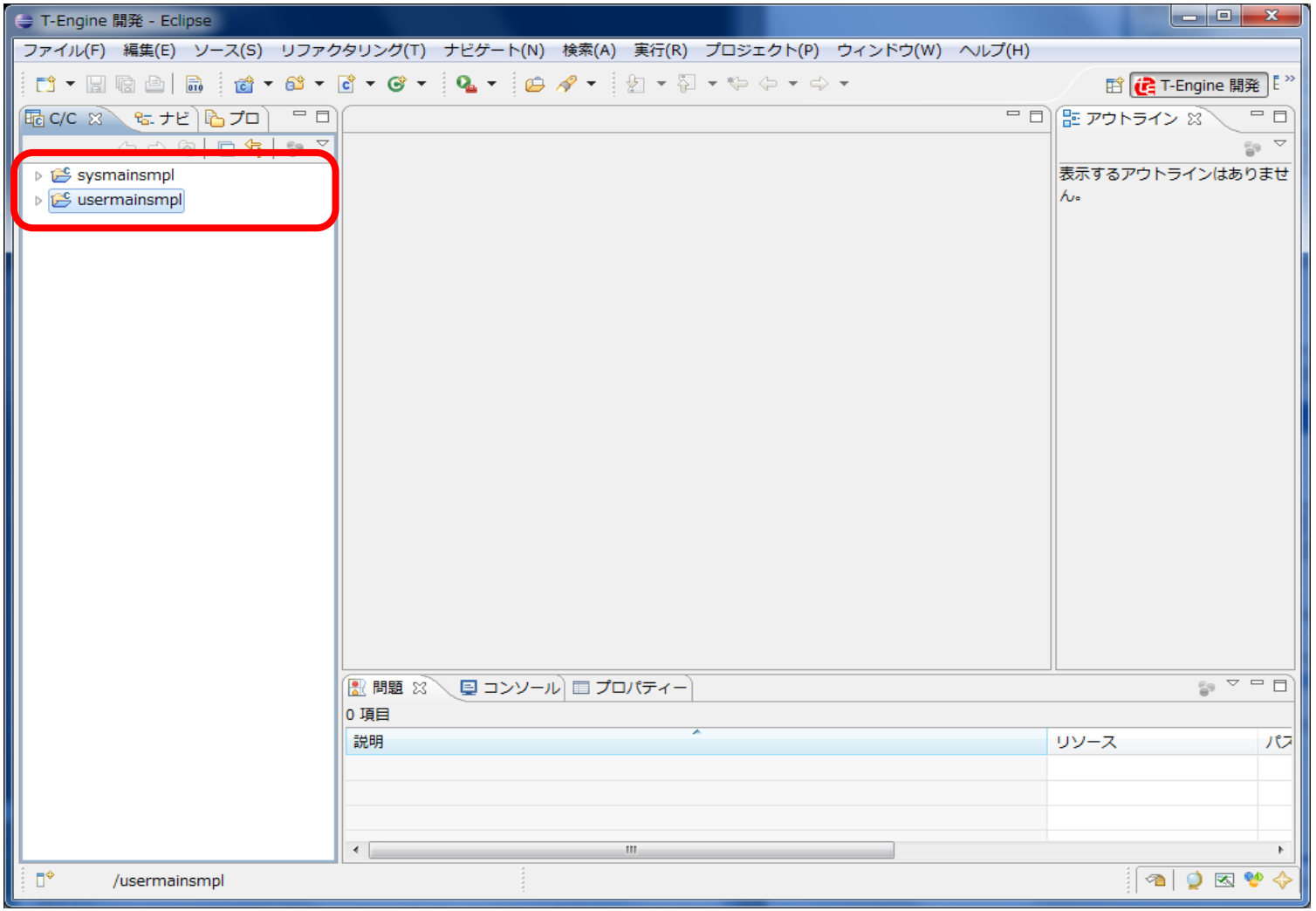
プロジェクトの設定 (4/9)



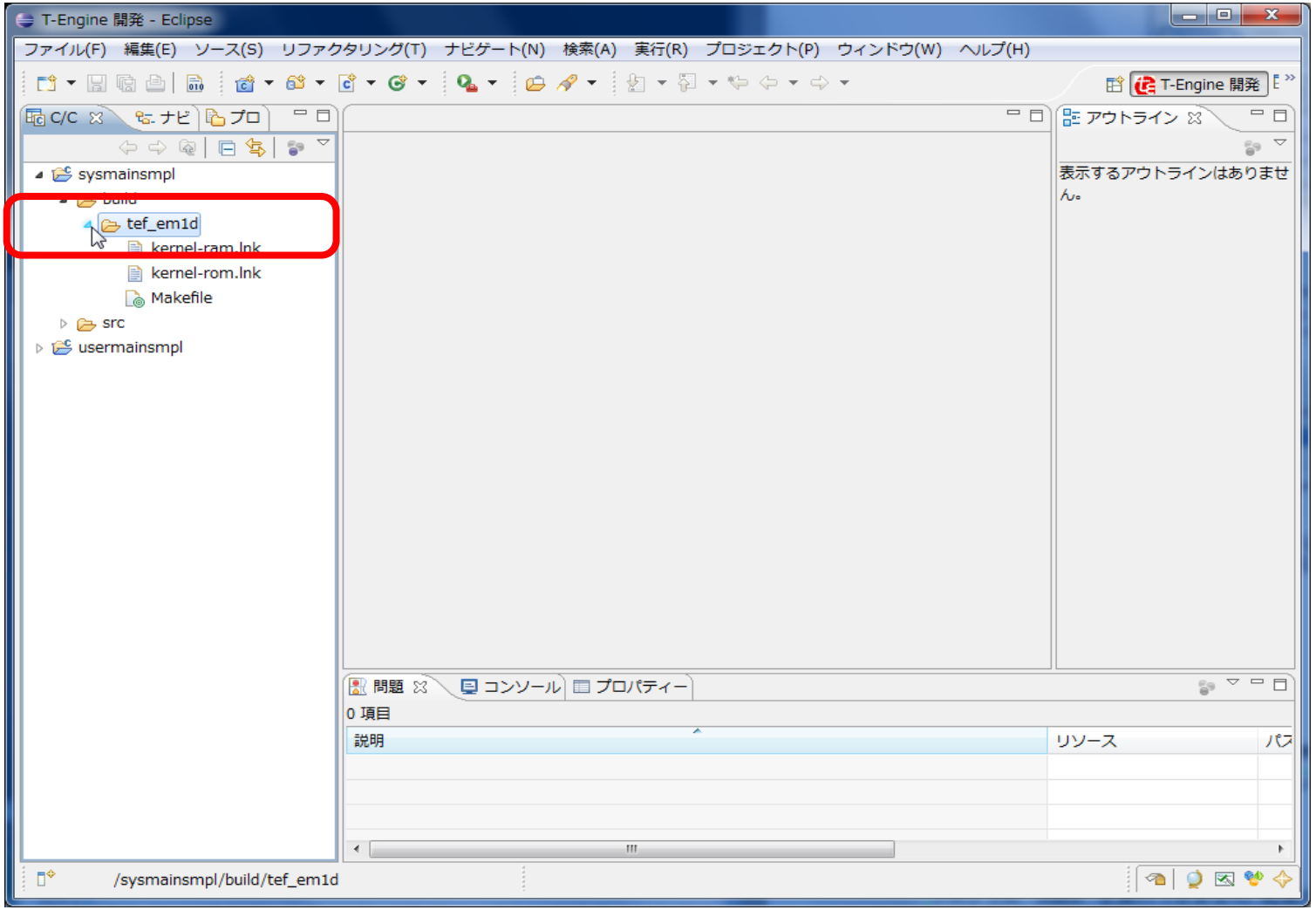
プロジェクトの設定 (5/9)



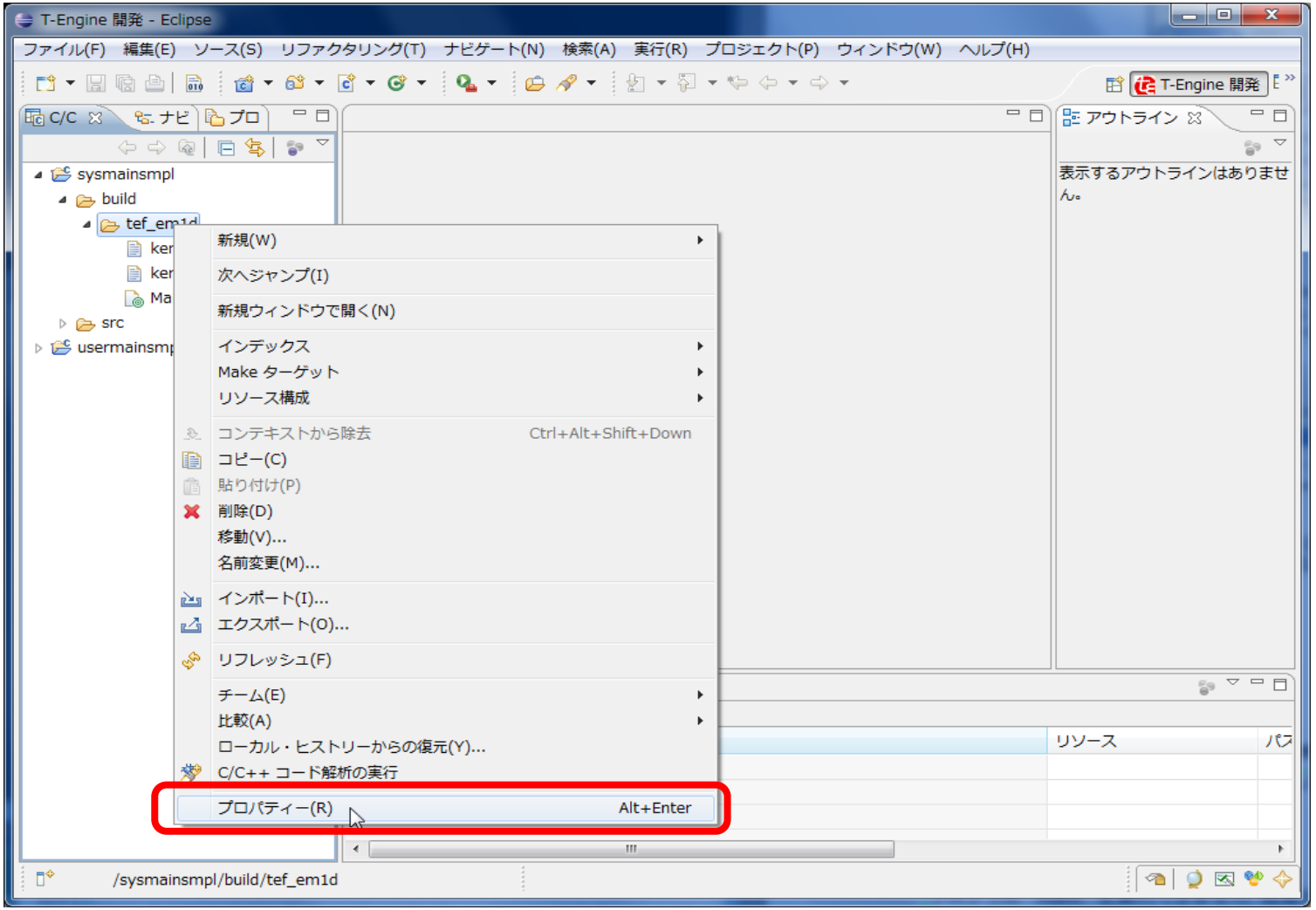
プロジェクトの設定 (6/9)



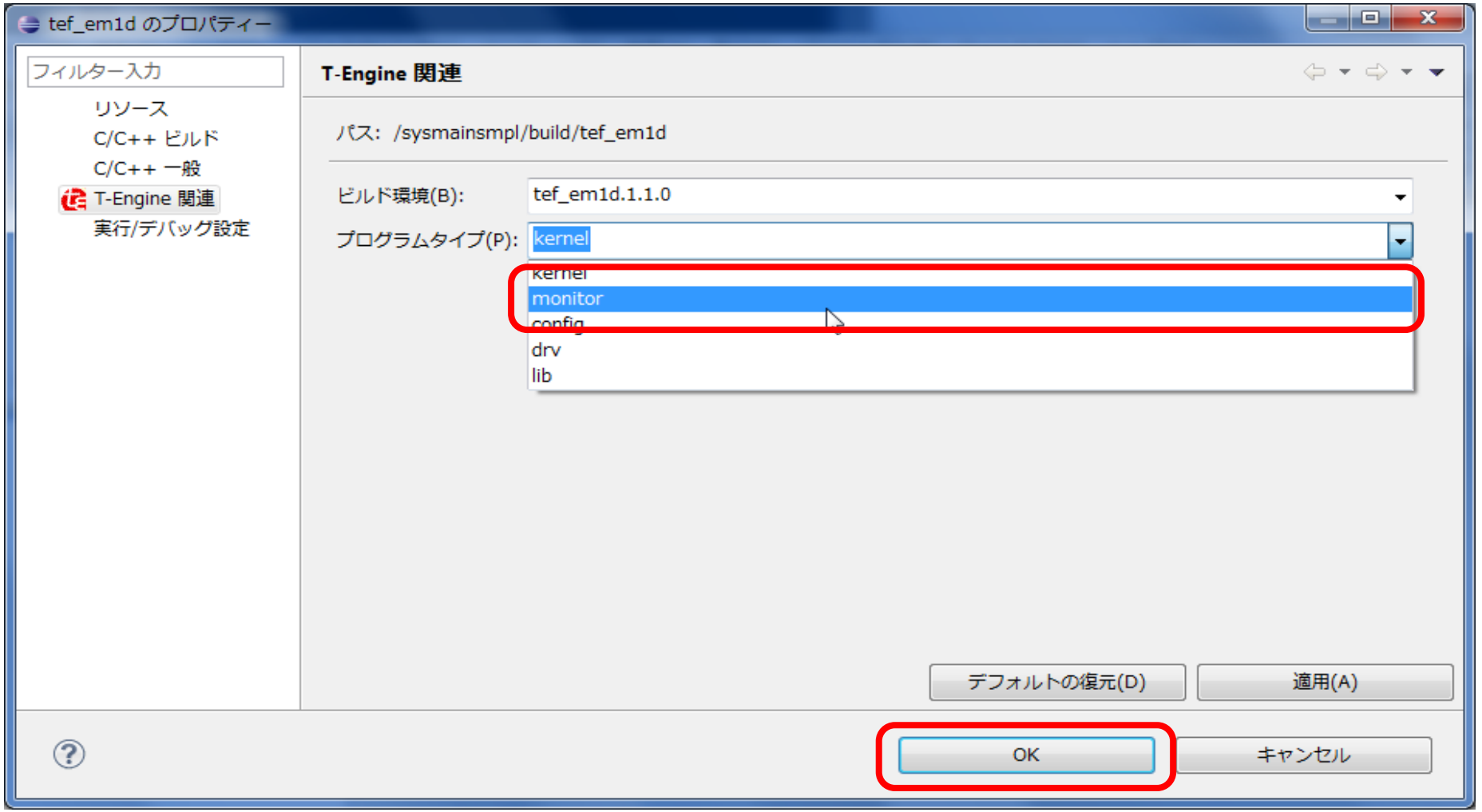
プロジェクトの設定 (7/9)



プロジェクトの設定 (8/9)

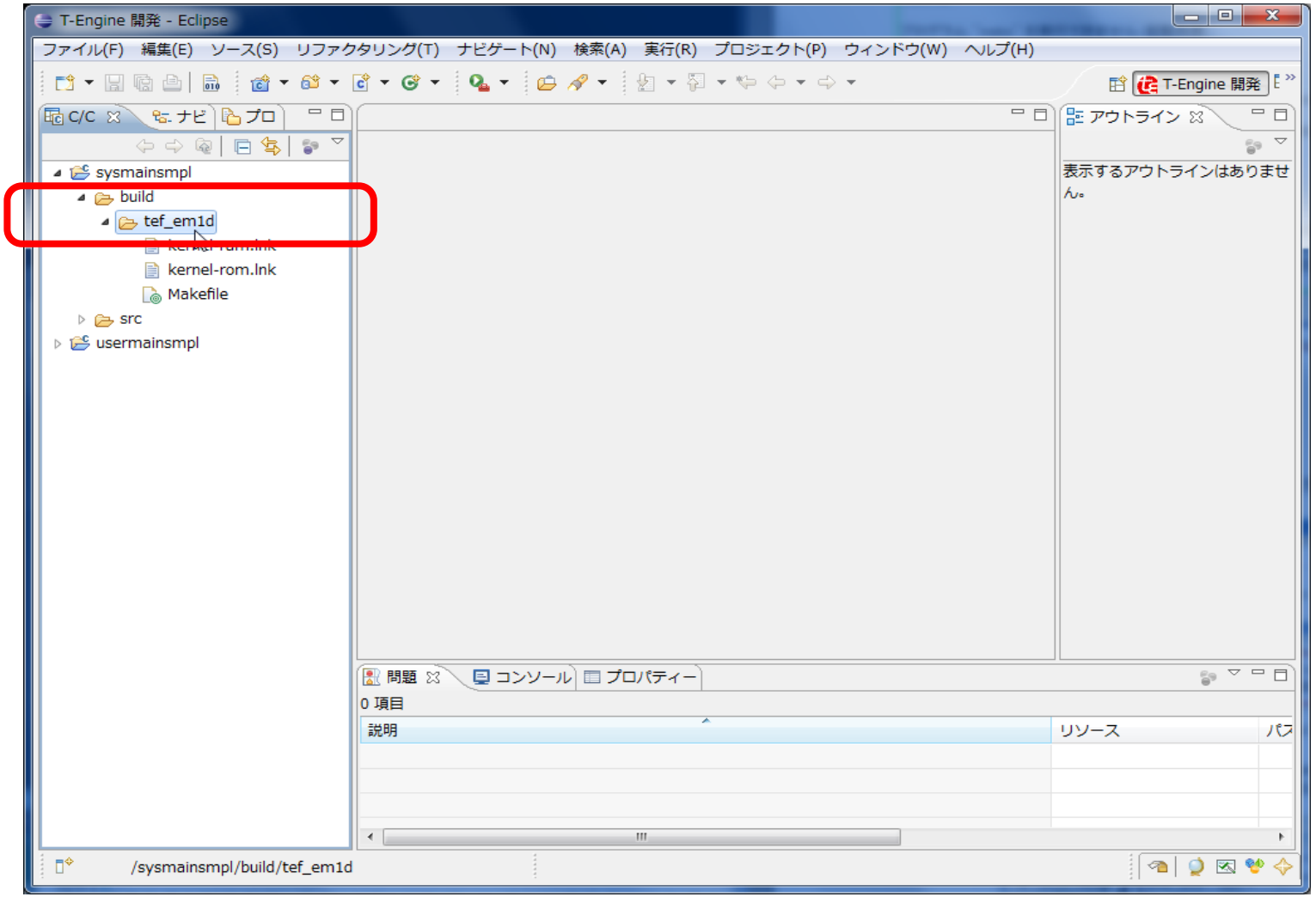


プロジェクトの設定 (9/9)

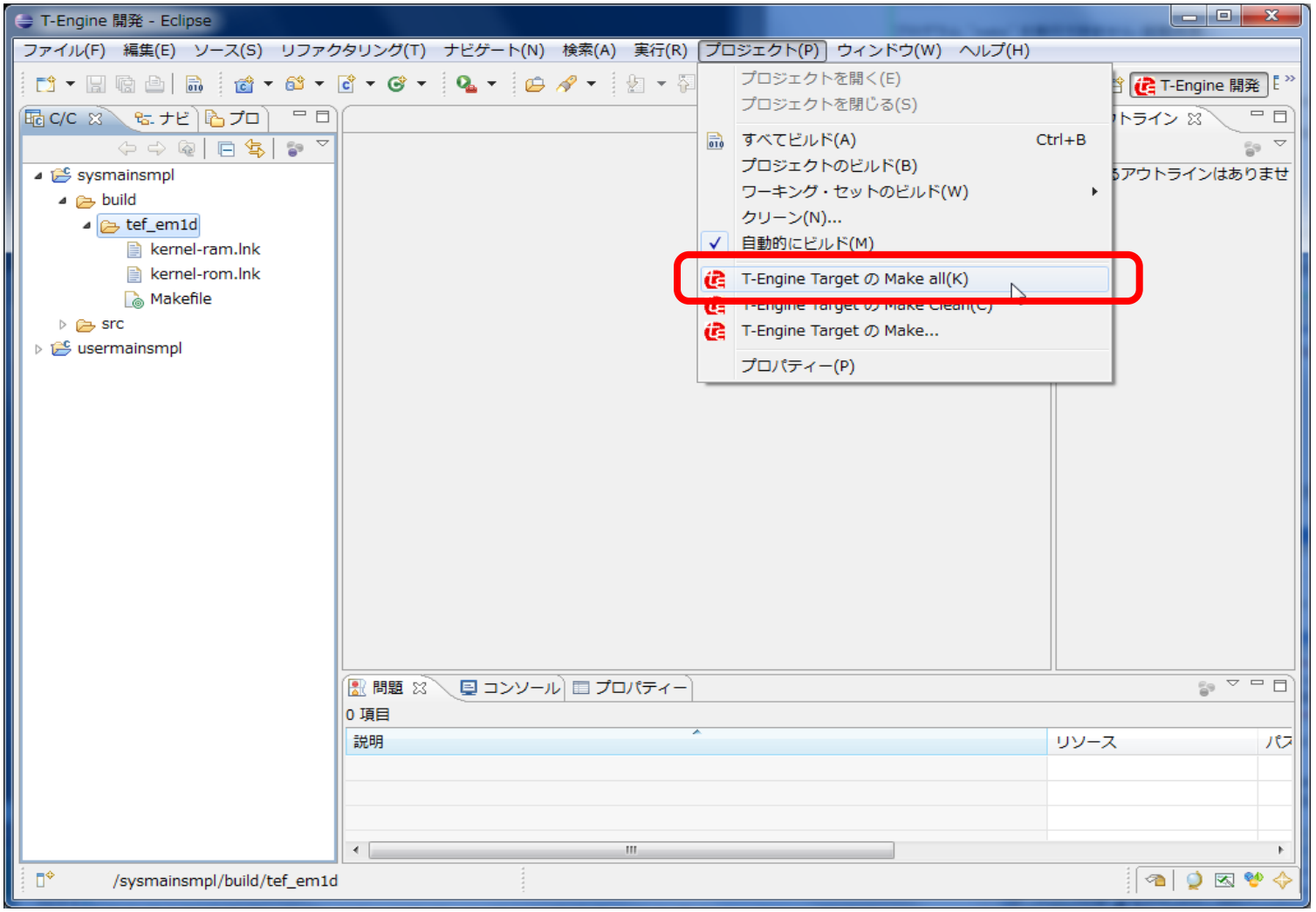


ビルド

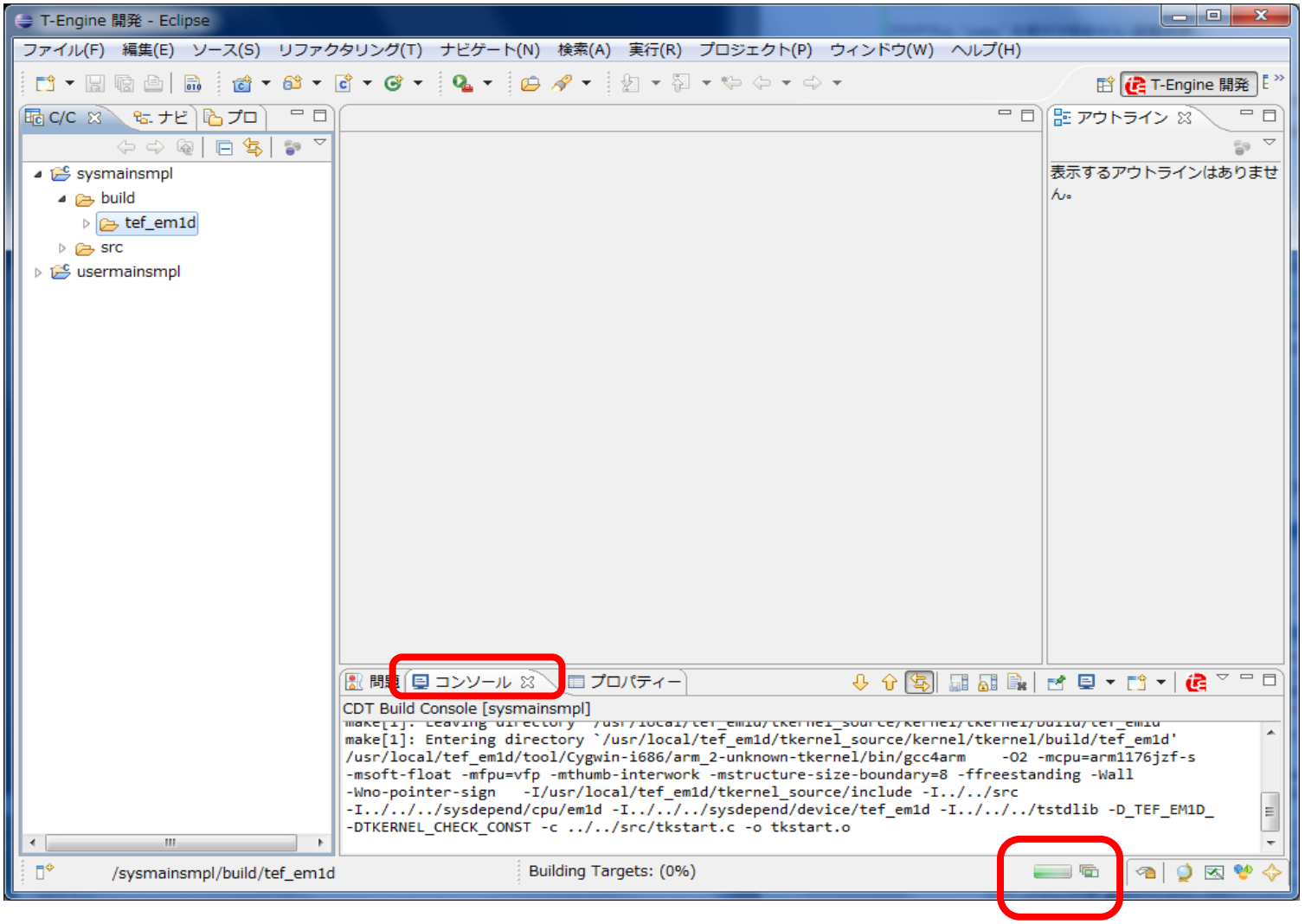
ビルド (1/4)


T-Engine 開発 - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
T-Engine 開発
C/C
ナビ
プロ
sysmainsmpl
build
tef_em1d
kernel-rom.lnk
Makefile
src
usermainsmpl
アウトライン
表示するアウトラインはありません。
問題
コンソール
プロパティー
0 項目
説明
リソース
/sysmainsmpl/build/tef_em1d

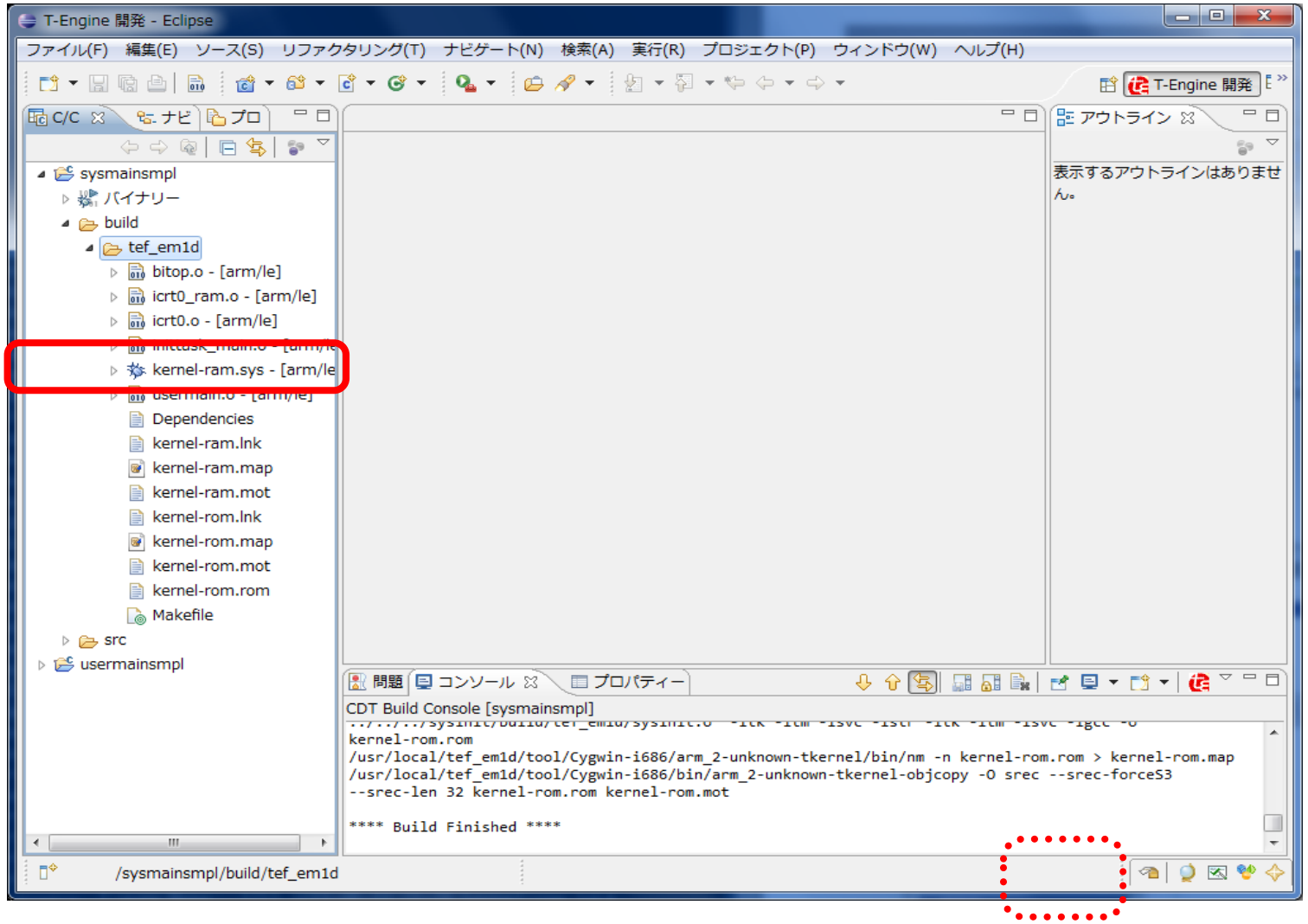
ビルド (2/4)


T-Engine 開発 - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
プロジェクトを開く(E)
プロジェクトを閉じる(S)
すべてビルド(A) Ctrl+B
プロジェクトのビルド(B)
ワーキング・セットのビルド(W)
クリーン(N)...
自動的にビルド(M)
T-Engine Target の Make all(K)
T-Engine Target の Make...
プロパティー(P)
T-Engine 開発
C/C
ナビ
プロ
sysmainsmpl
build
tef_em1d
kernel-ram.lnk
kernel-rom.lnk
Makefile
src
usermainsmpl
問題
コンソール
プロパティー
0 項目
説明
リソース
/sysmainsmpl/build/tef_em1d

ビルド (3/4)


T-Engine 開発 - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
T-Engine 開発
C/C ナビ プロ
sysmainsmpl
build
tef_em1d
src
usermainsmpl
アウトライン
表示するアウトラインはありません。
問題 コンソール プロパティー
CDT Build Console [sysmainsmpl]
make[1]: Entering directory `/usr/local/tef_em1d/tkernel_source/kernel/tkernel/build/tef_em1d'
/usr/local/tef_em1d/tool/Cygwin-i686/arm_2-unknown-tkernel/bin/gcc4arm -O2 -mcpu=arm1176jzf-s
-msoft-float -mfpu=vfp -mthumb-interwork -mstructure-size-boundary=8 -ffreestanding -Wall
-Wno-pointer-sign -I/usr/local/tef_em1d/tkernel_source/include -I../../src
-I../../../sysdepend/cpu/em1d -I../../../sysdepend/device/tef_em1d -I../../../tstdlib -D_TEF_EM1D_
-DTKERNEL_CHECK_CONST -c ../../src/tkstart.c -o tkstart.o
/sysmainsmpl/build/tef_em1d
Building Targets: (0%)

ビルド (4/4)

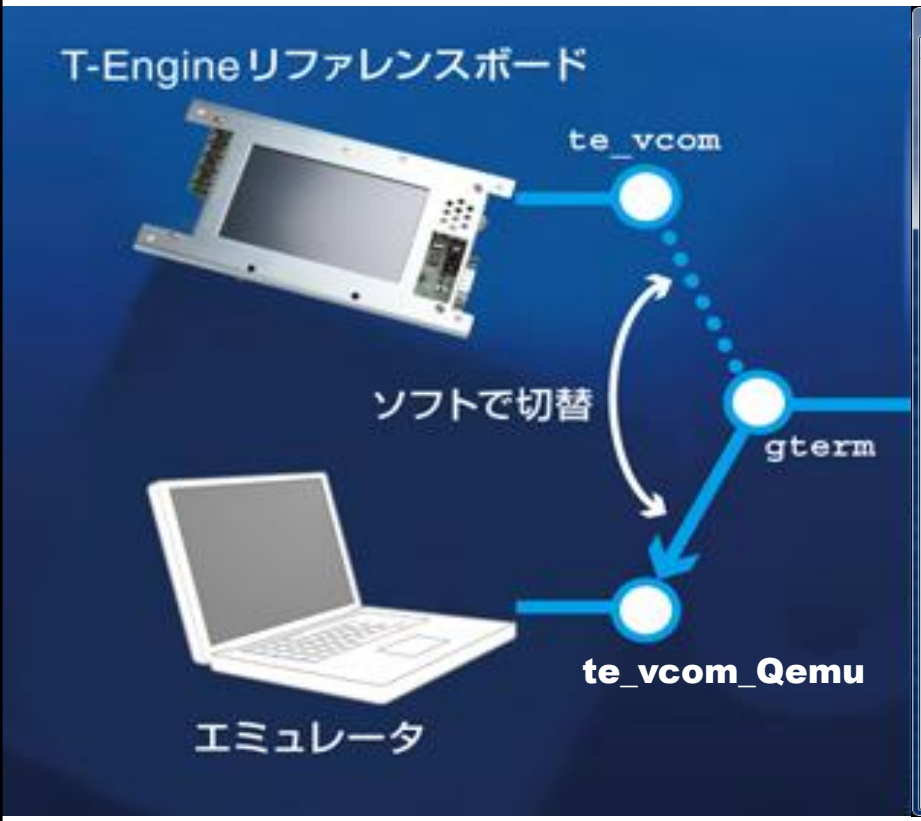

T-Engine 開発 - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
T-Engine 開発
C/C ナビ プロ
sysmainsmpl
バイナリー
build
tef_em1d
bitop.o - [arm/le]
icrt0_ram.o - [arm/le]
icrt0.o - [arm/le]
kernel-ram.sys - [arm/le
Dependencies
kernel-ram.lnk
kernel-ram.map
kernel-ram.mot
kernel-rom.lnk
kernel-rom.map
kernel-rom.mot
kernel-rom.rom
Makefile
src
usermainsmpl
アウトライン
表示するアウトラインはありません。
問題 コンソール プロパティー
CDT Build Console [sysmainsmpl]
kernel-rom.rom
/usr/local/tef_em1d/tool/Cygwin-i686/arm_2-unknown-tkernel/bin/nm -n kernel-rom.rom > kernel-rom.map
/usr/local/tef_em1d/tool/Cygwin-i686/bin/arm_2-unknown-tkernel-objcopy -O srec --srec-forceS3
--srec-len 32 kernel-rom.rom kernel-rom.mot
**** Build Finished ****
/sysmainsmpl/build/tef_em1d

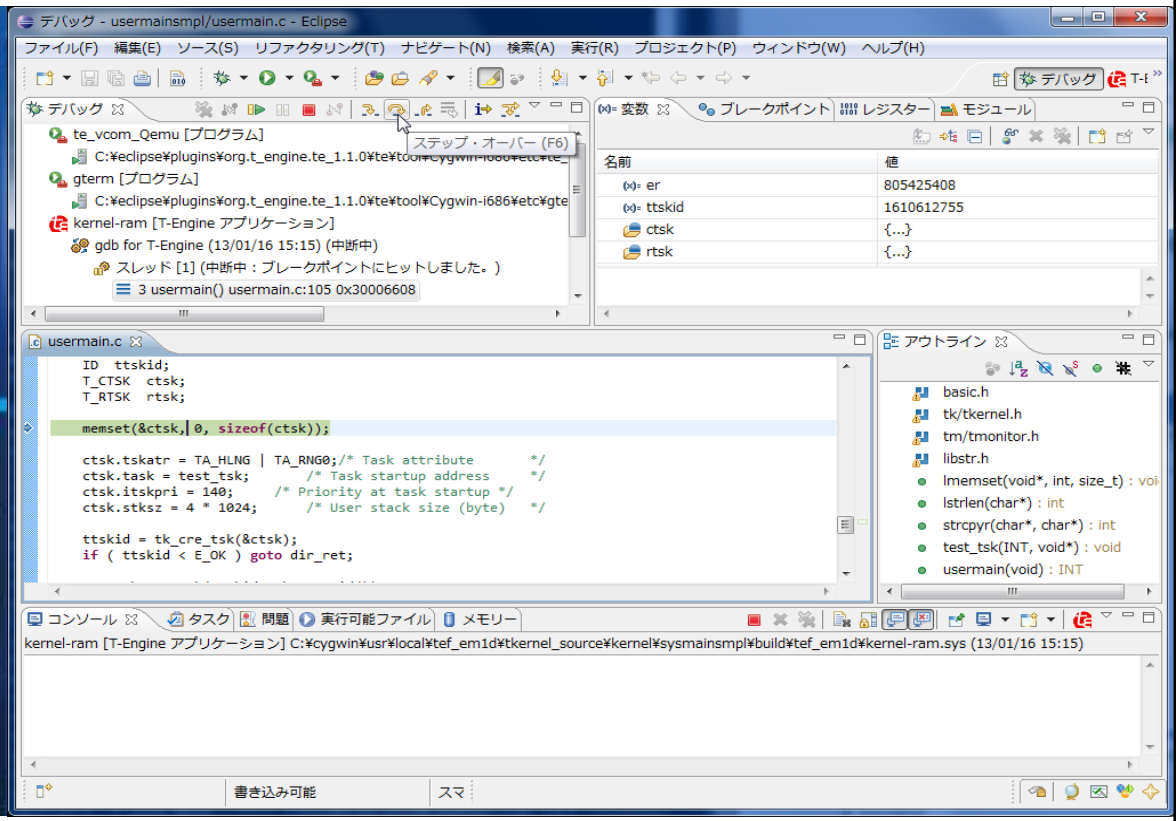
エミュレータと接続して実行

エミュレータと接続して実行

- EclipseにエミュレータとのEclipseにエミュレータとの接続用の設定を追加
  - デフォルトの設定はT-Engineリファレンスボード用
    → エミュレータに接続するように te_vcom を複製
- デバッグ → デバッグの構成 → デバッグ

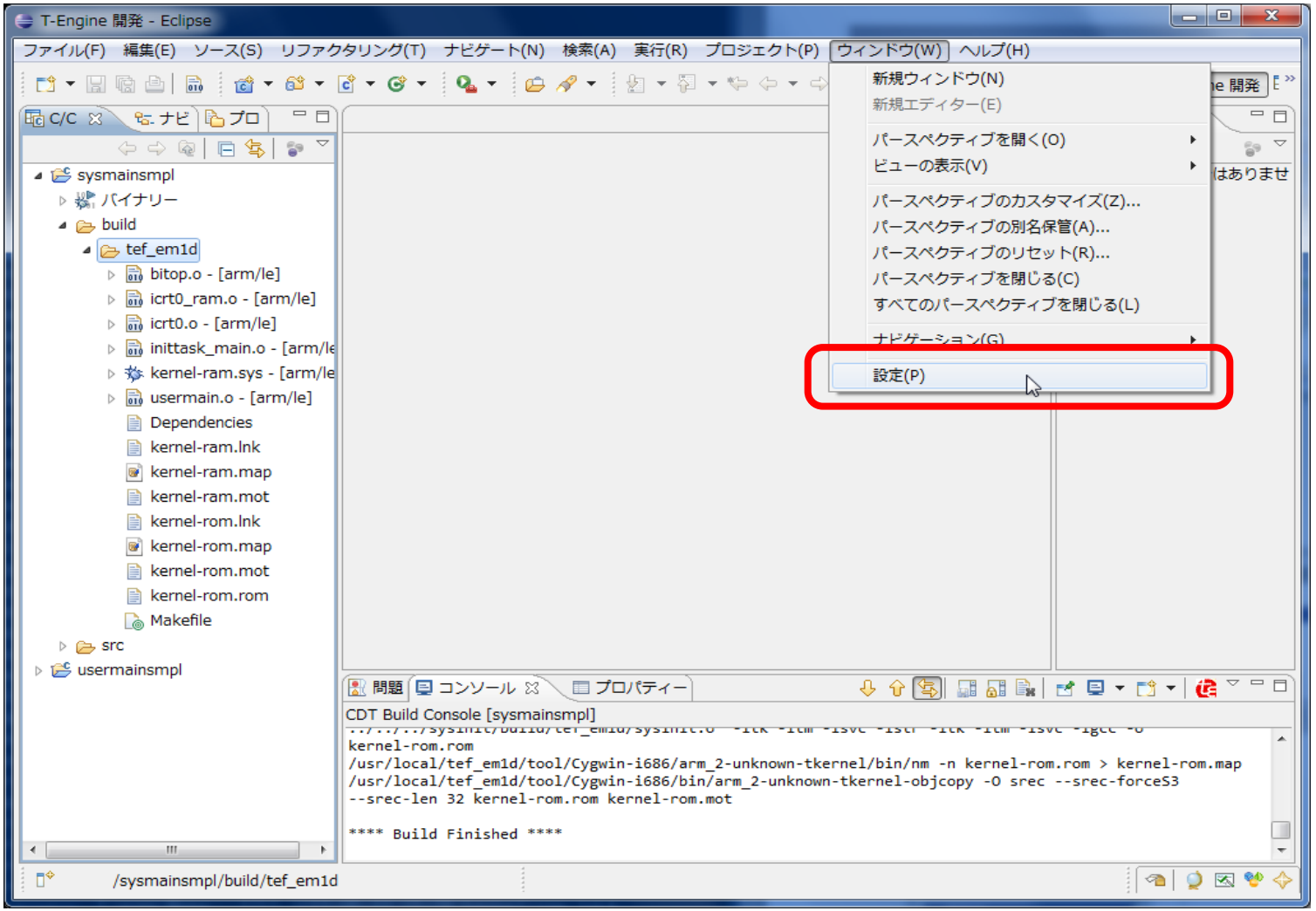
エミュレータとの接続イメージ



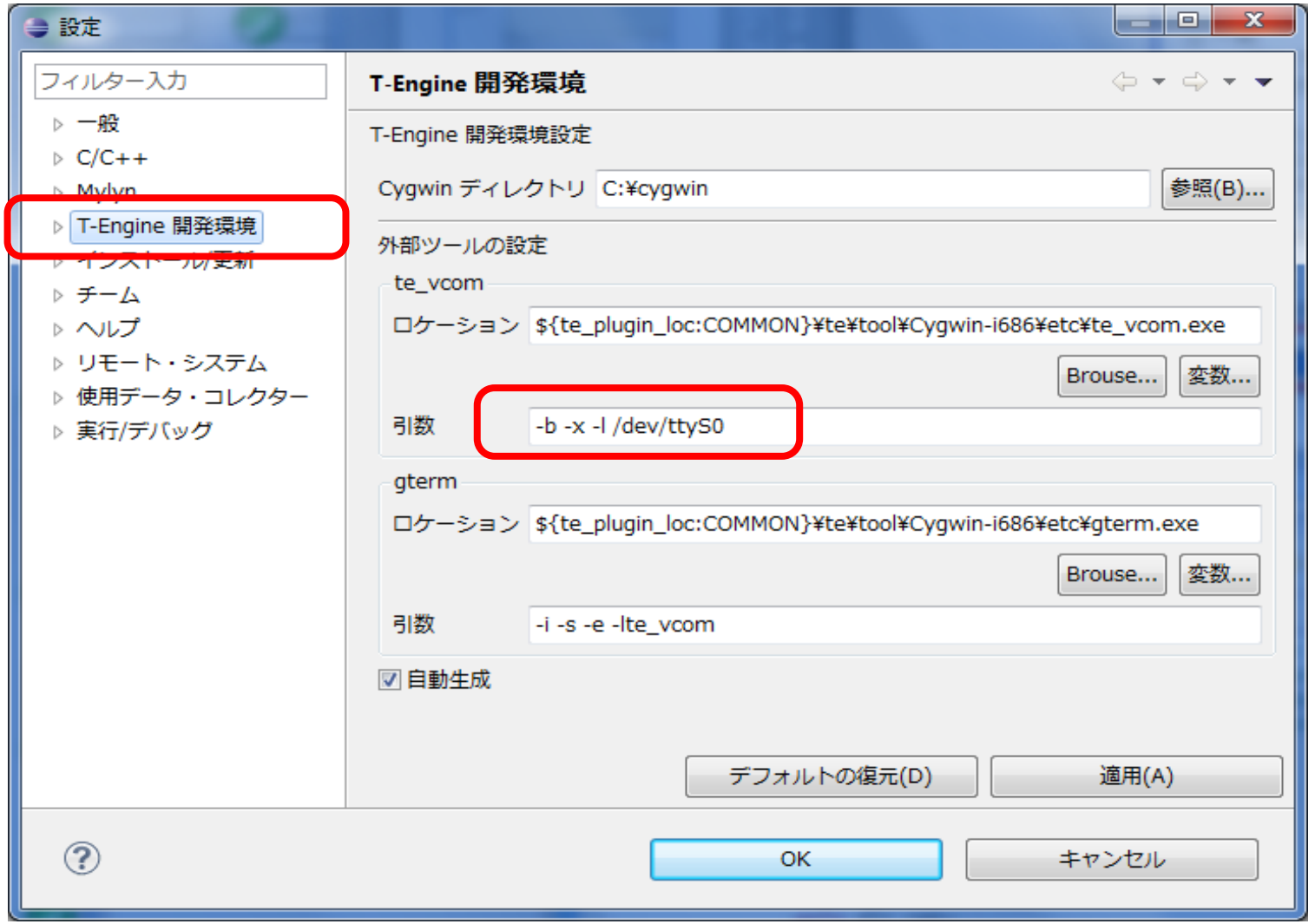


- te_vcom　仮想通信ポート
- gterm　通信ソフト (ターミナルソフト)

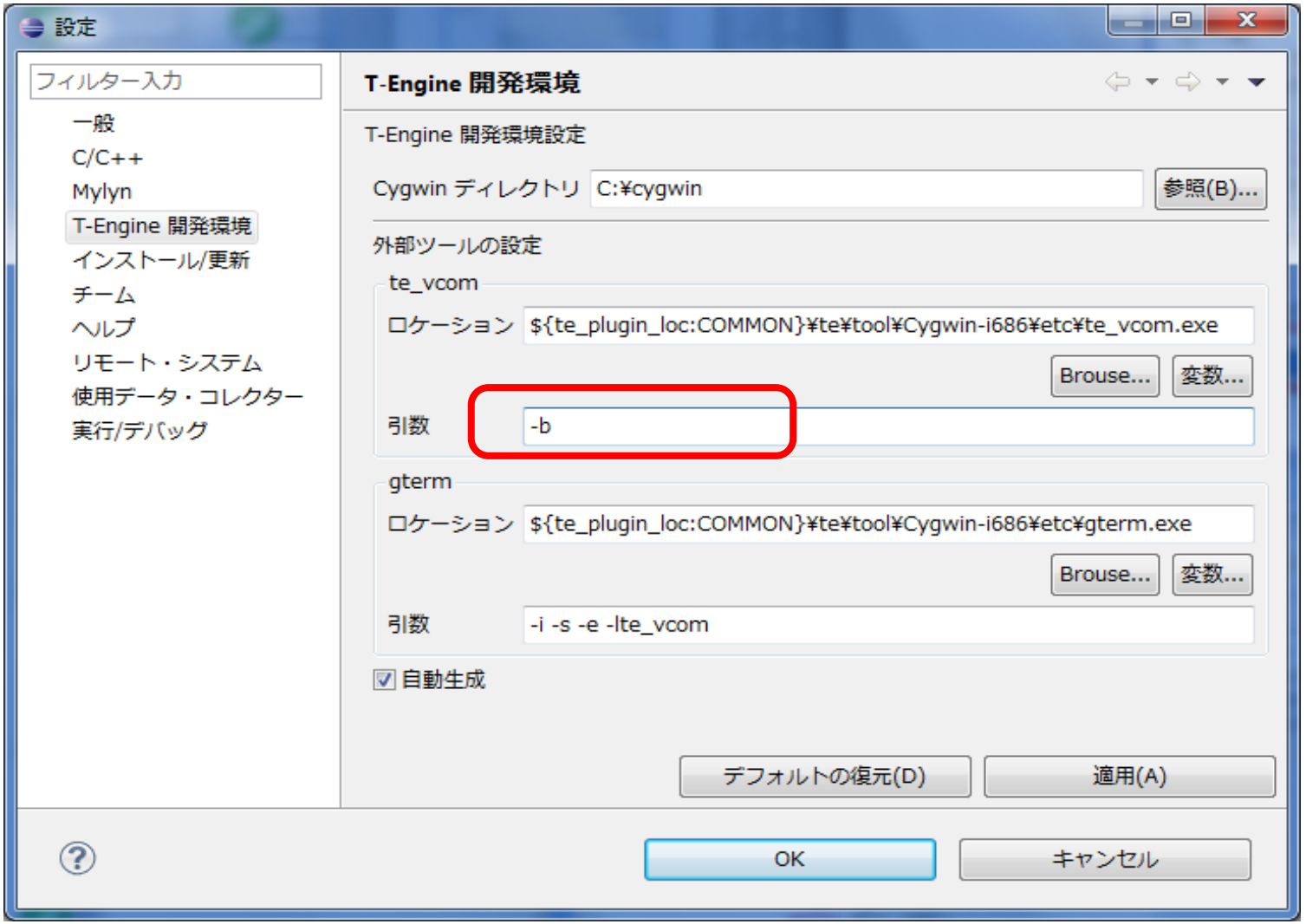
エミュレータとの接続 (1/13)



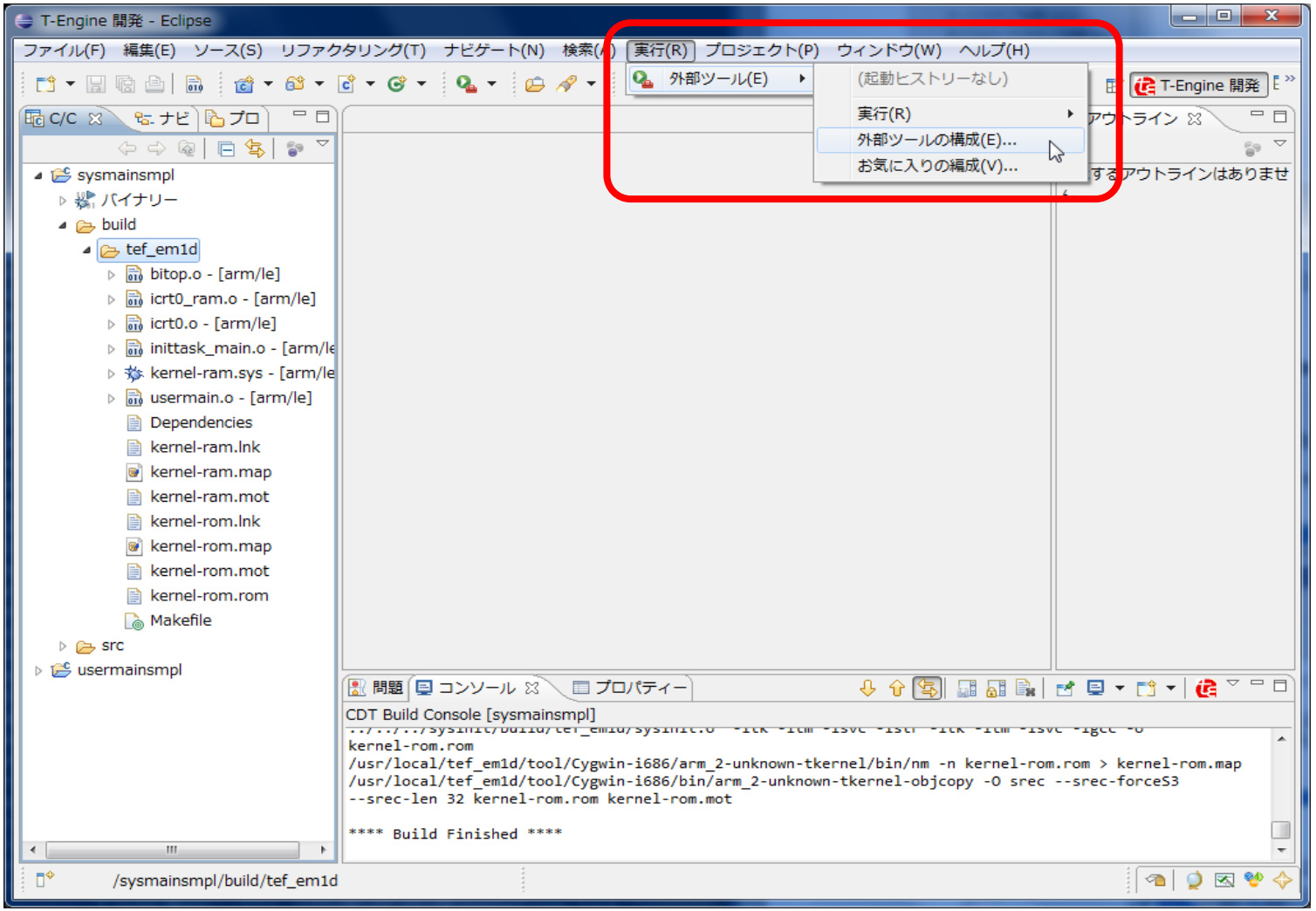
エミュレータとの接続 (2/13)



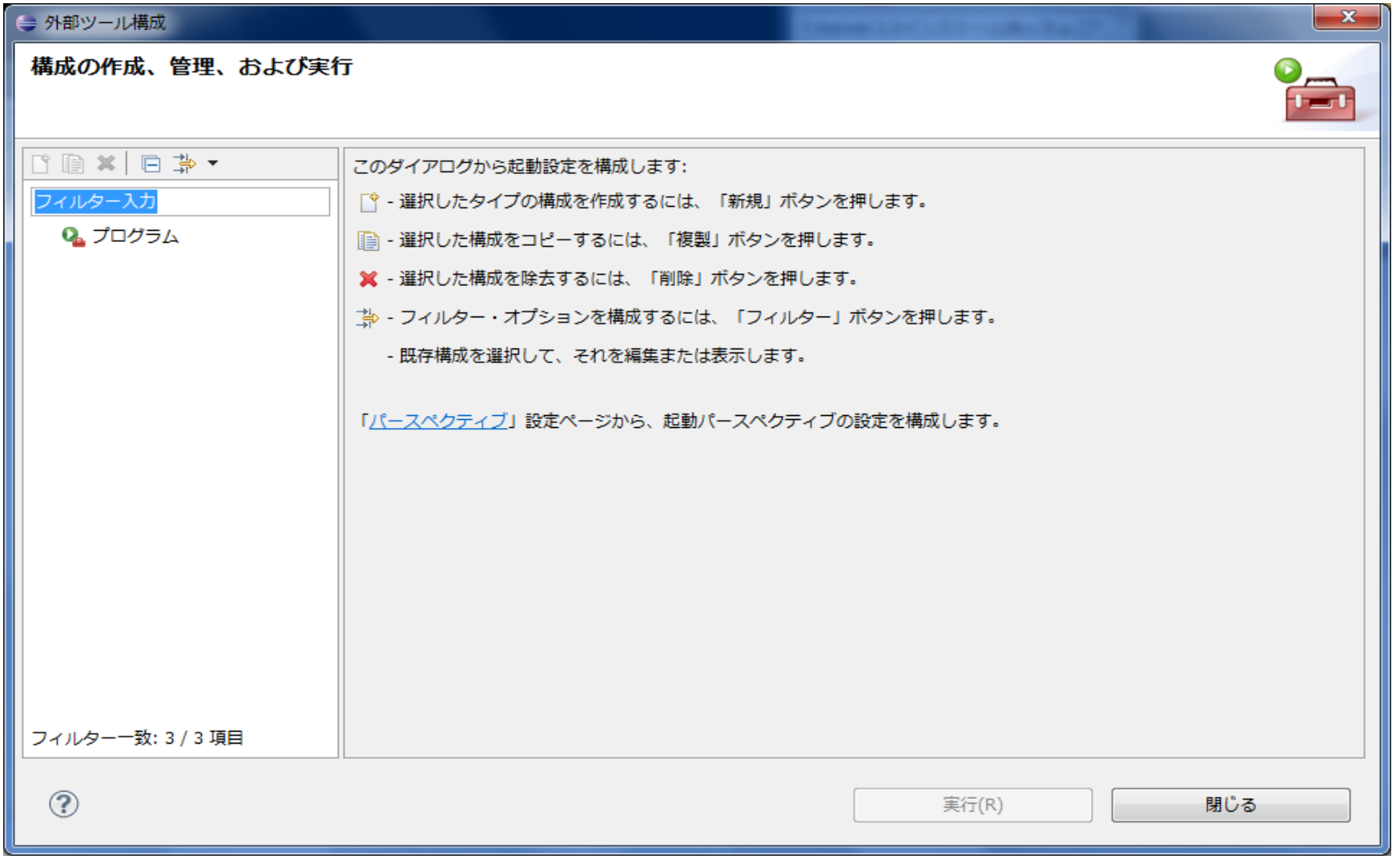
エミュレータとの接続 (3/13)



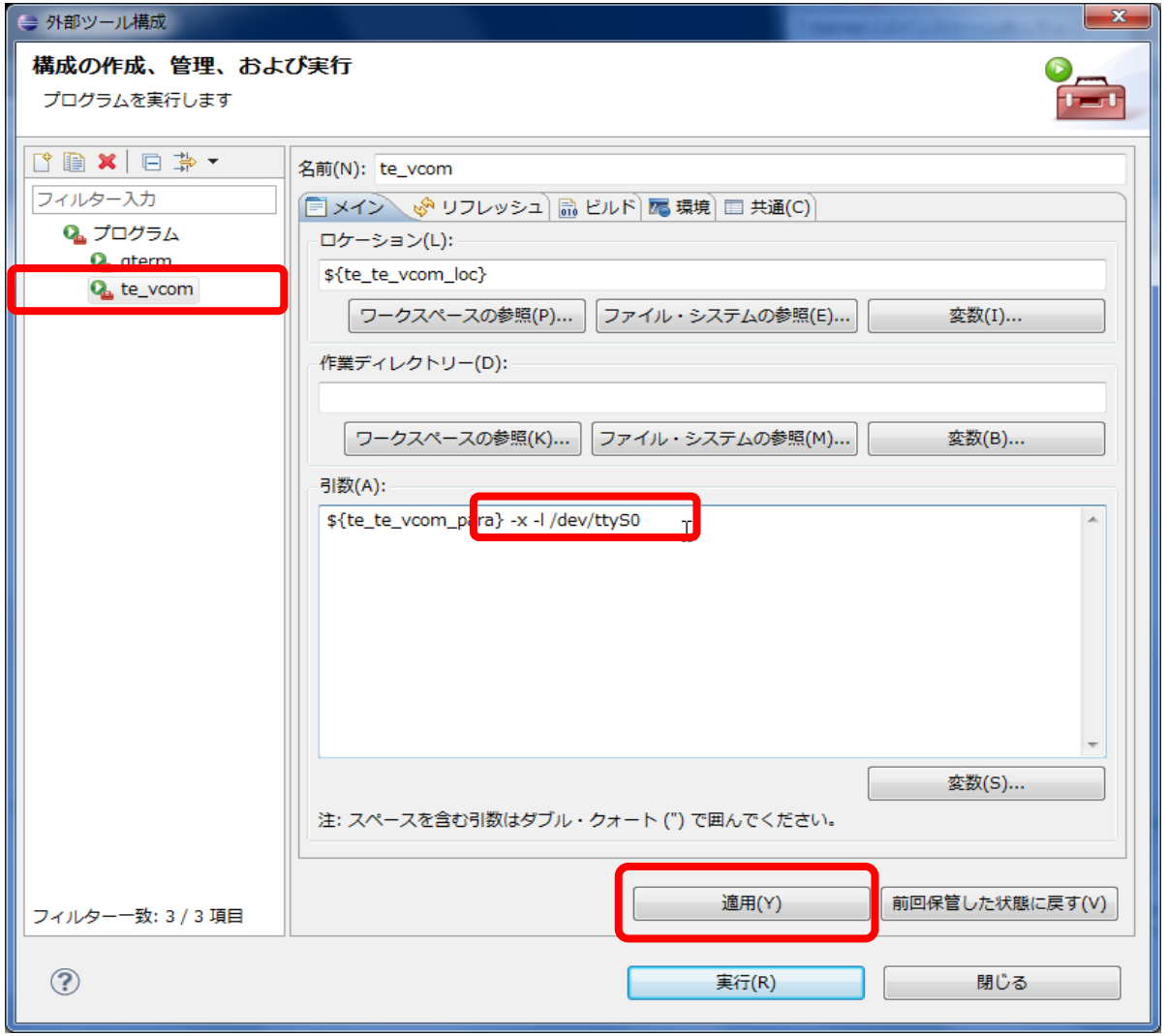
エミュレータとの接続 (4/13)



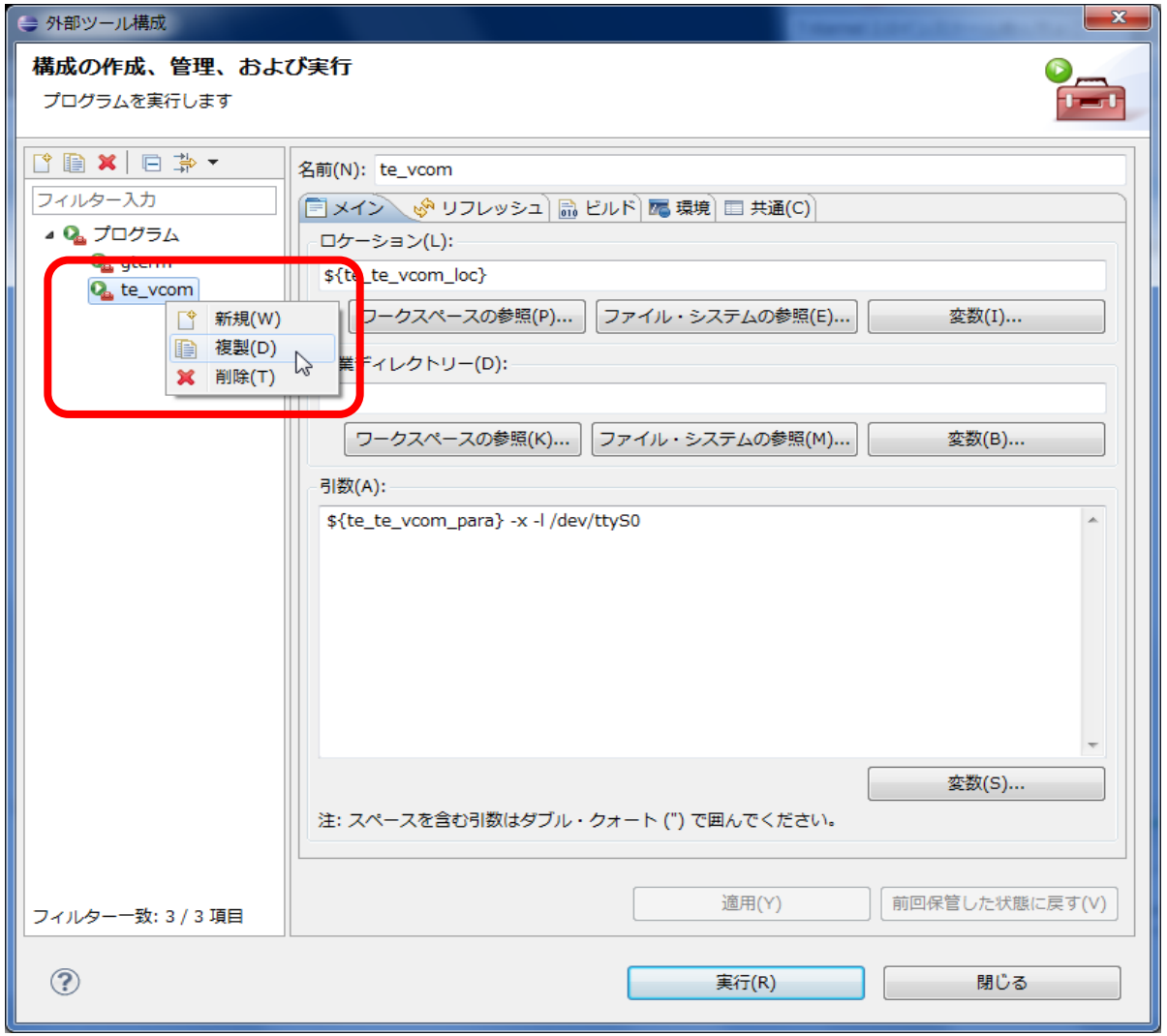
エミュレータとの接続 (5/13)



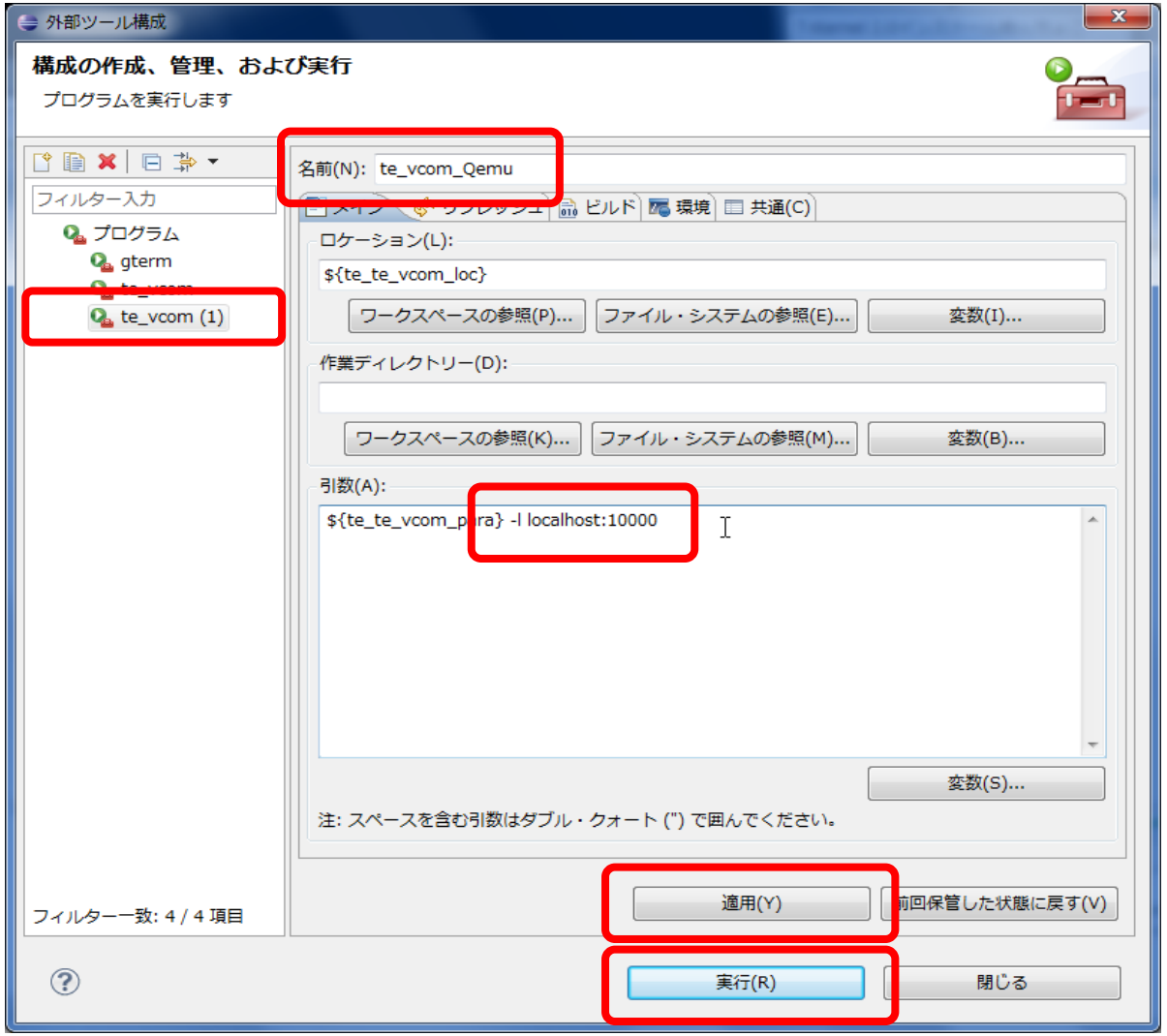
エミュレータとの接続 (6/13)



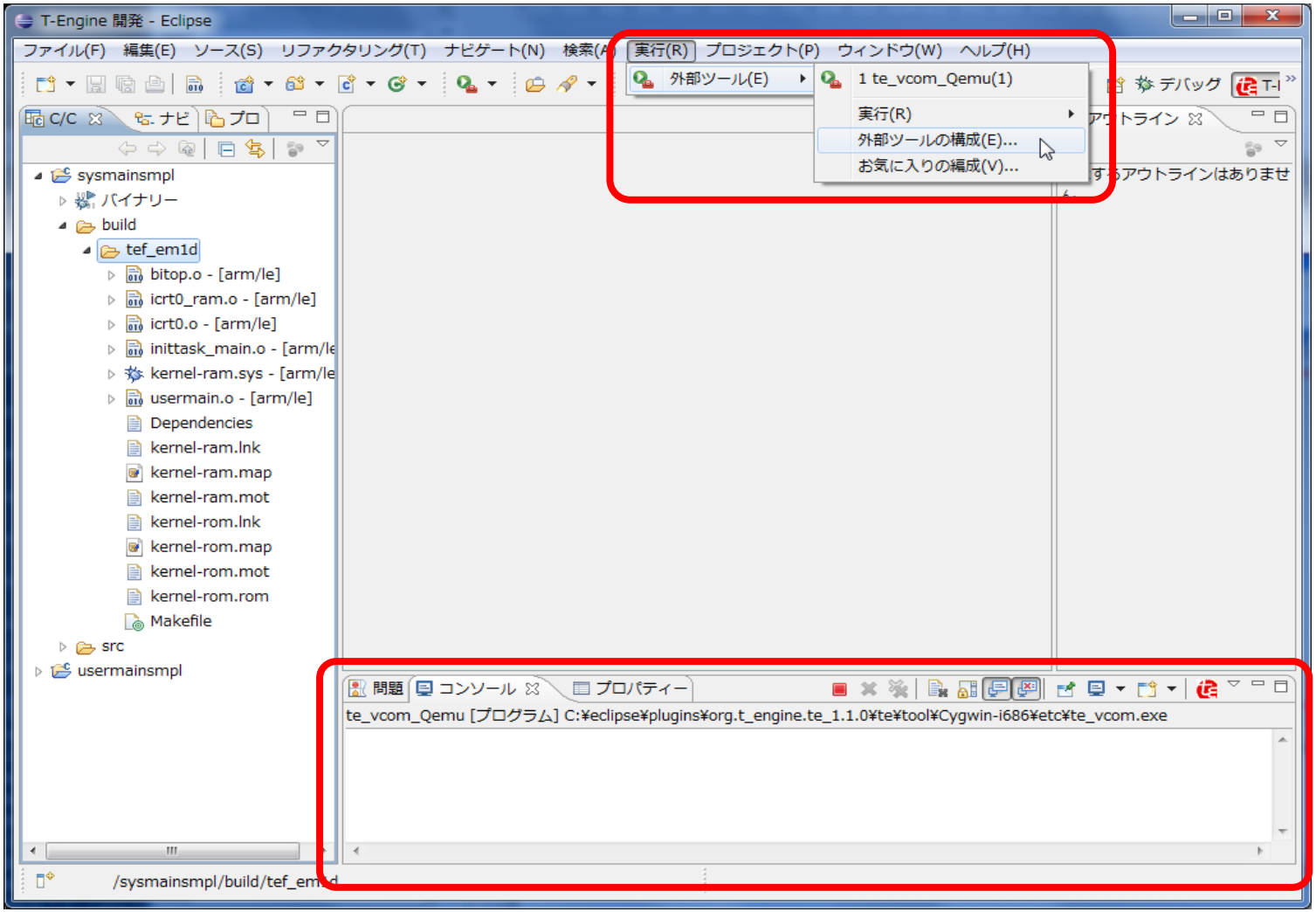
エミュレータとの接続 (7/13)



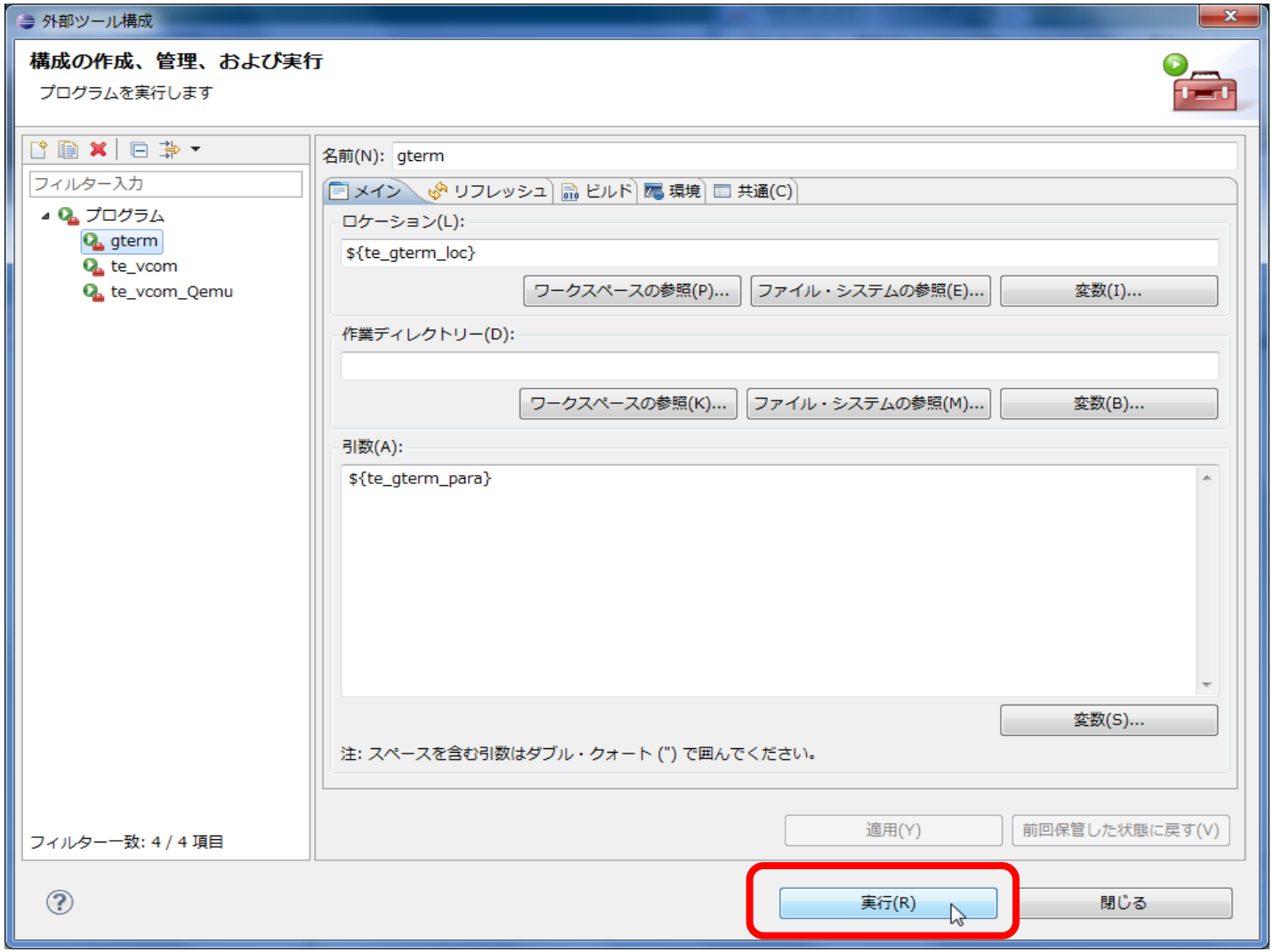
エミュレータとの接続 (8/13)



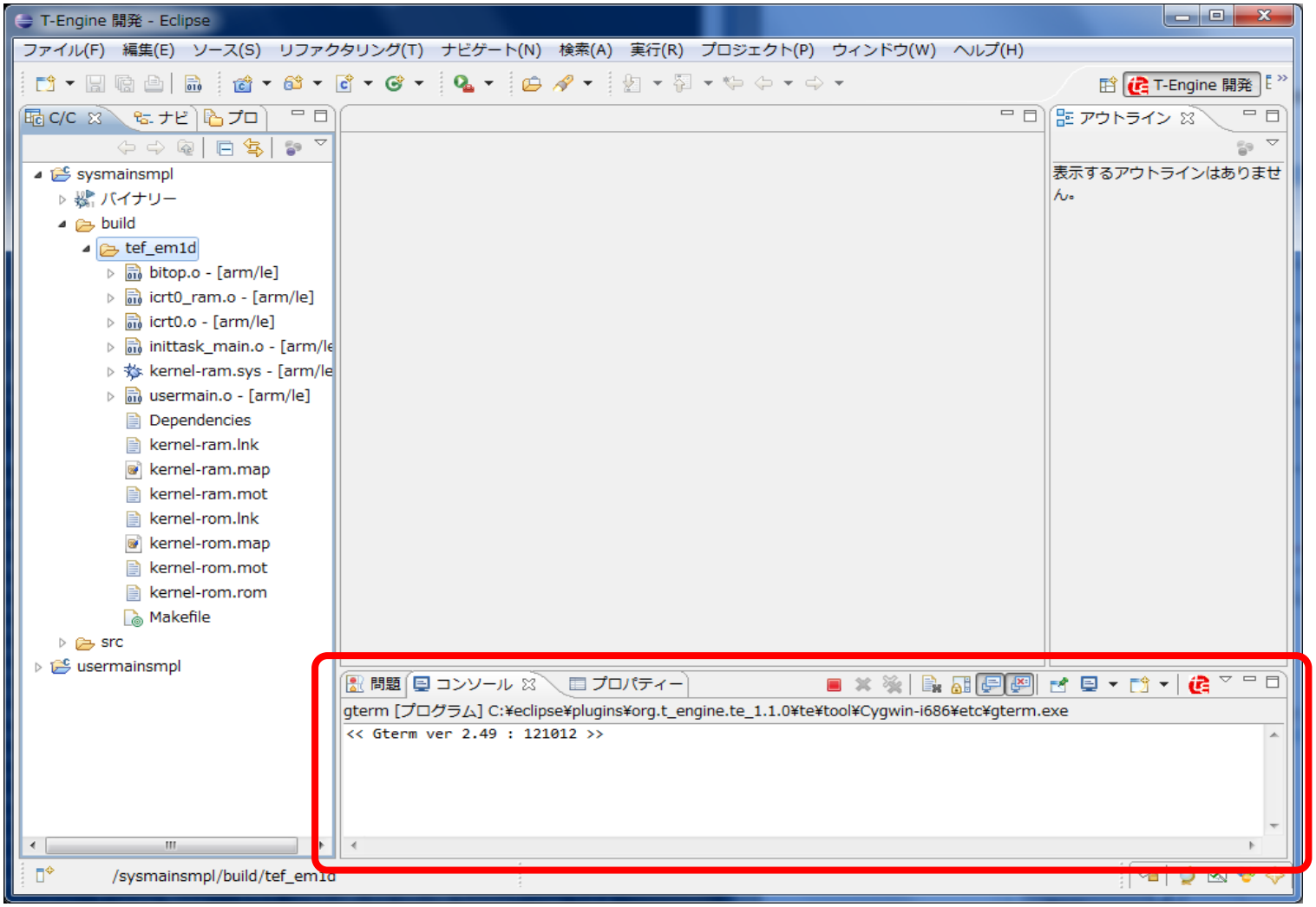
エミュレータとの接続 (9/13)



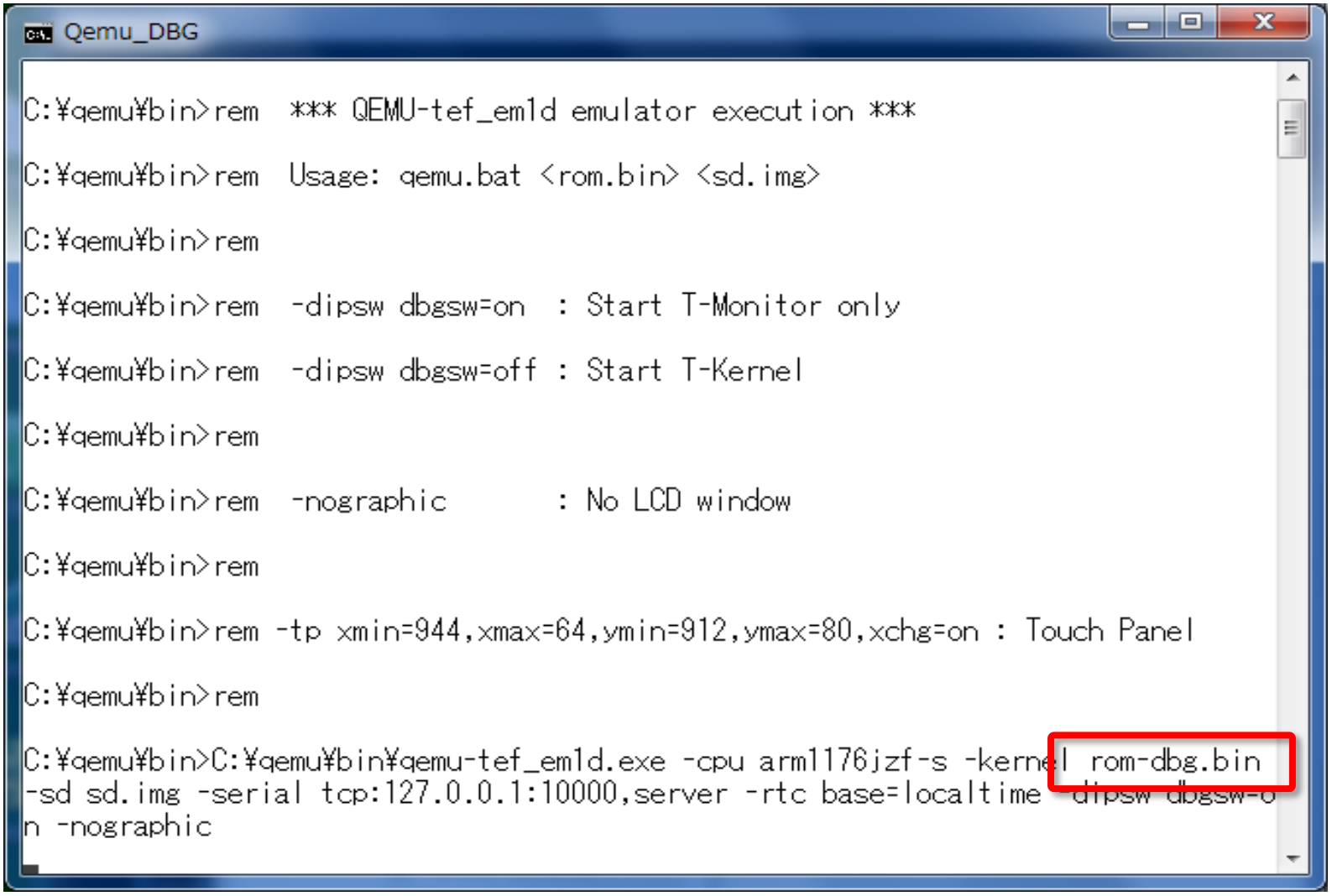
エミュレータとの接続 (10/13)



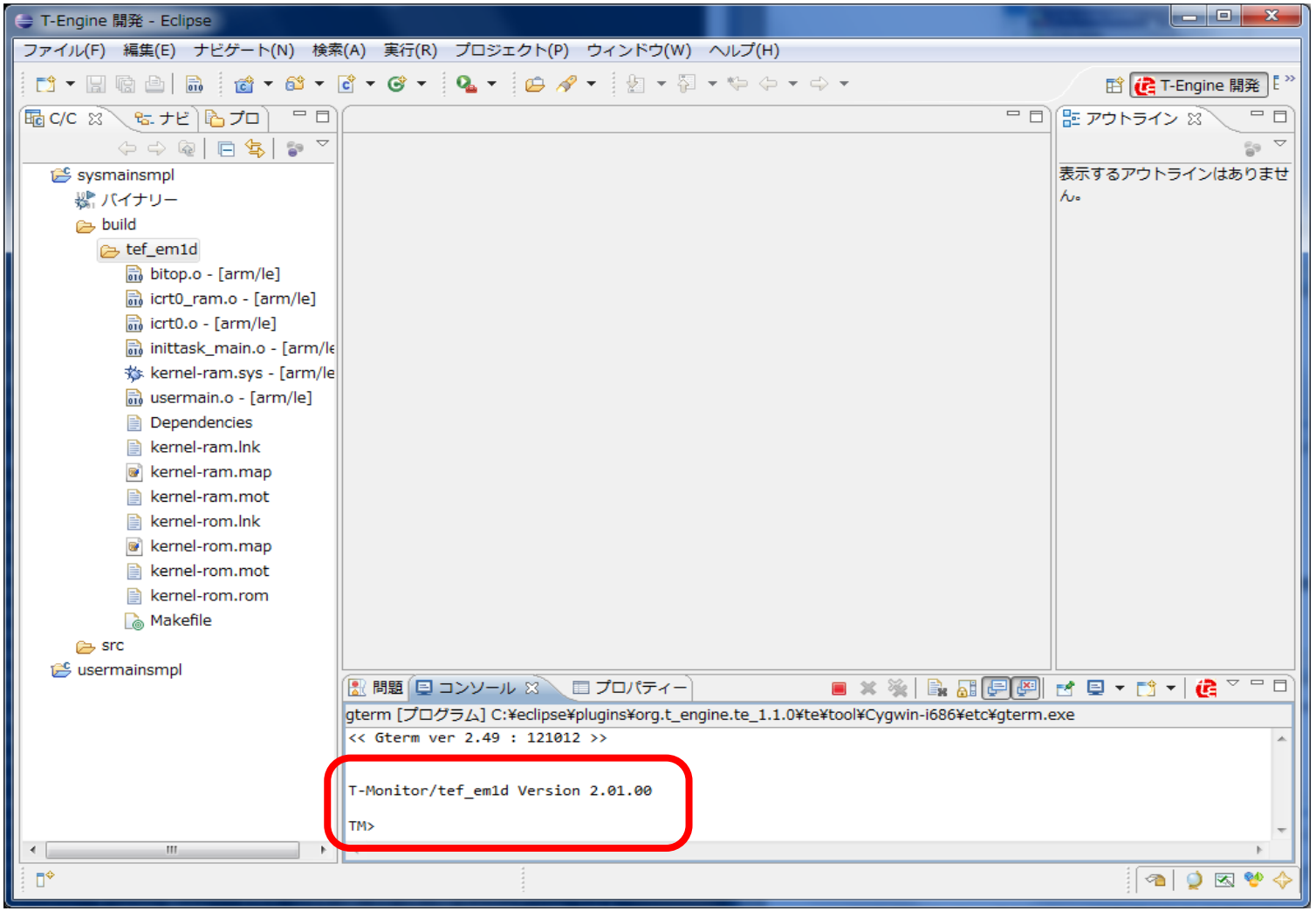
エミュレータとの接続 (11/13)



エミュレータとの接続 (12/13)



エミュレータとの接続 (13/13)

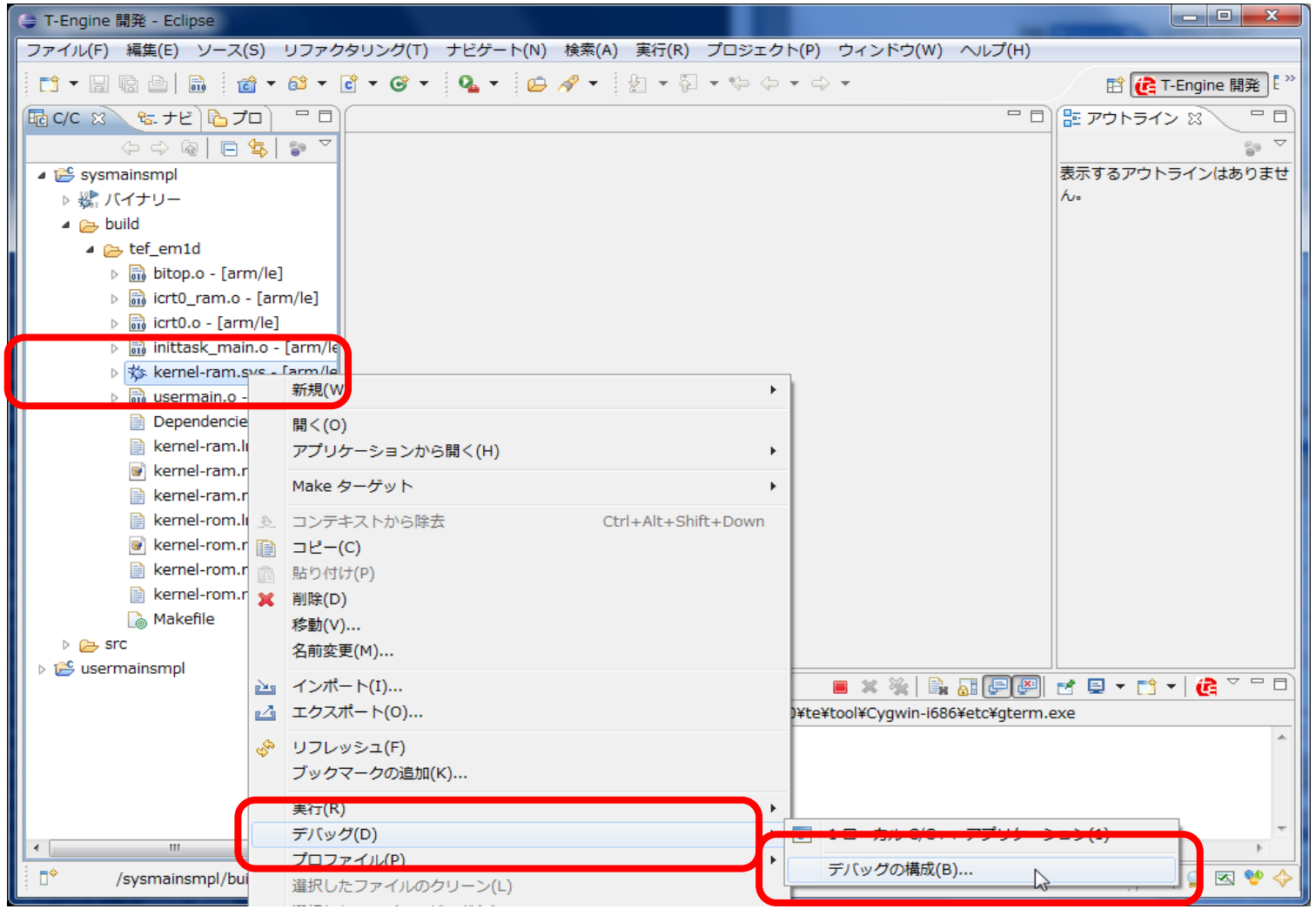


エミュレータと接続して実行

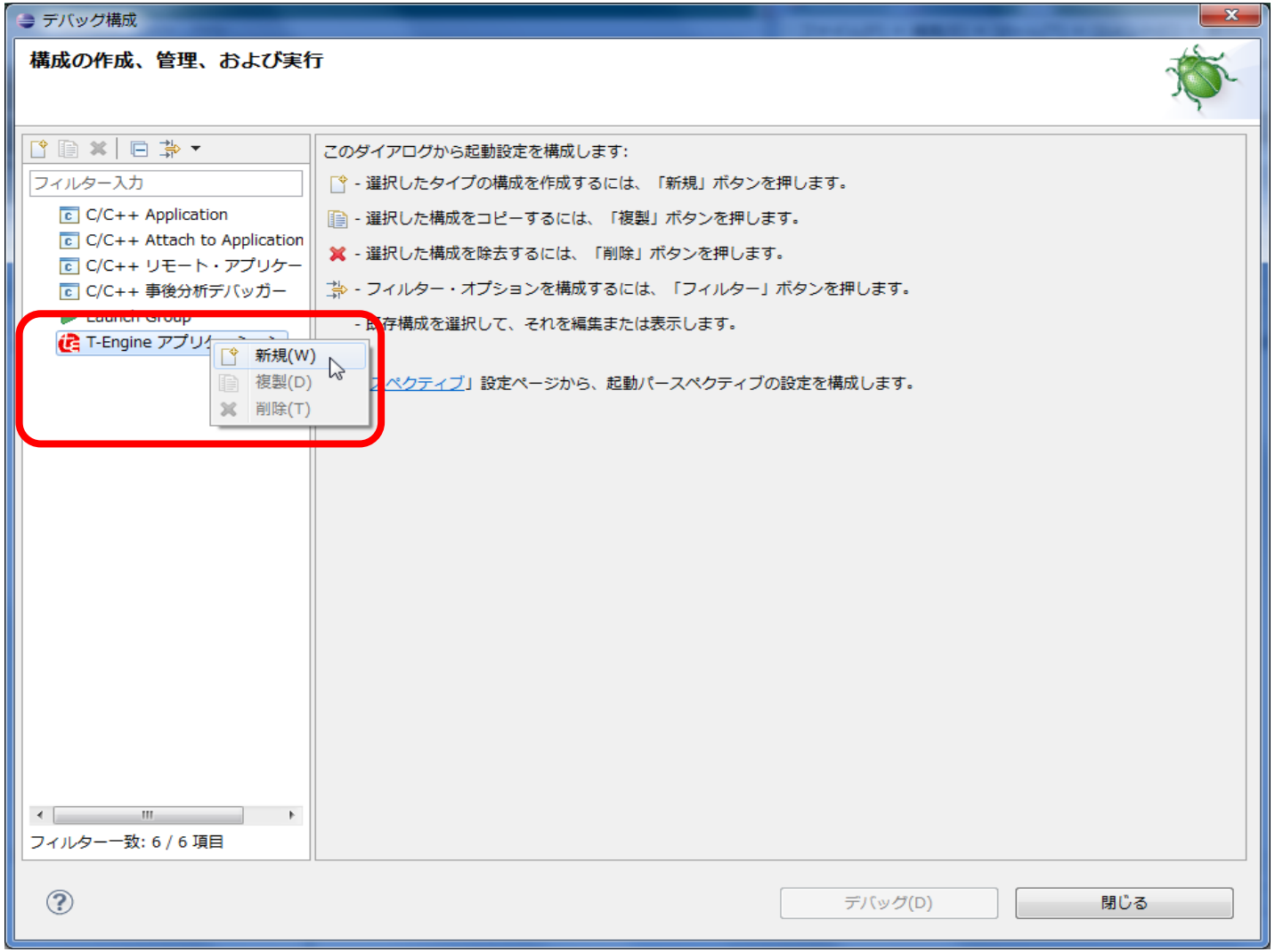
- EclipseにエミュレータとのEclipseにエミュレータとの接続用の設定を追加
  - デフォルトの設定はT-Engineリファレンスボード用
    → エミュレータに接続するように te_vcom を複製

- デバッグ → デバッグの構成 → デバッグ

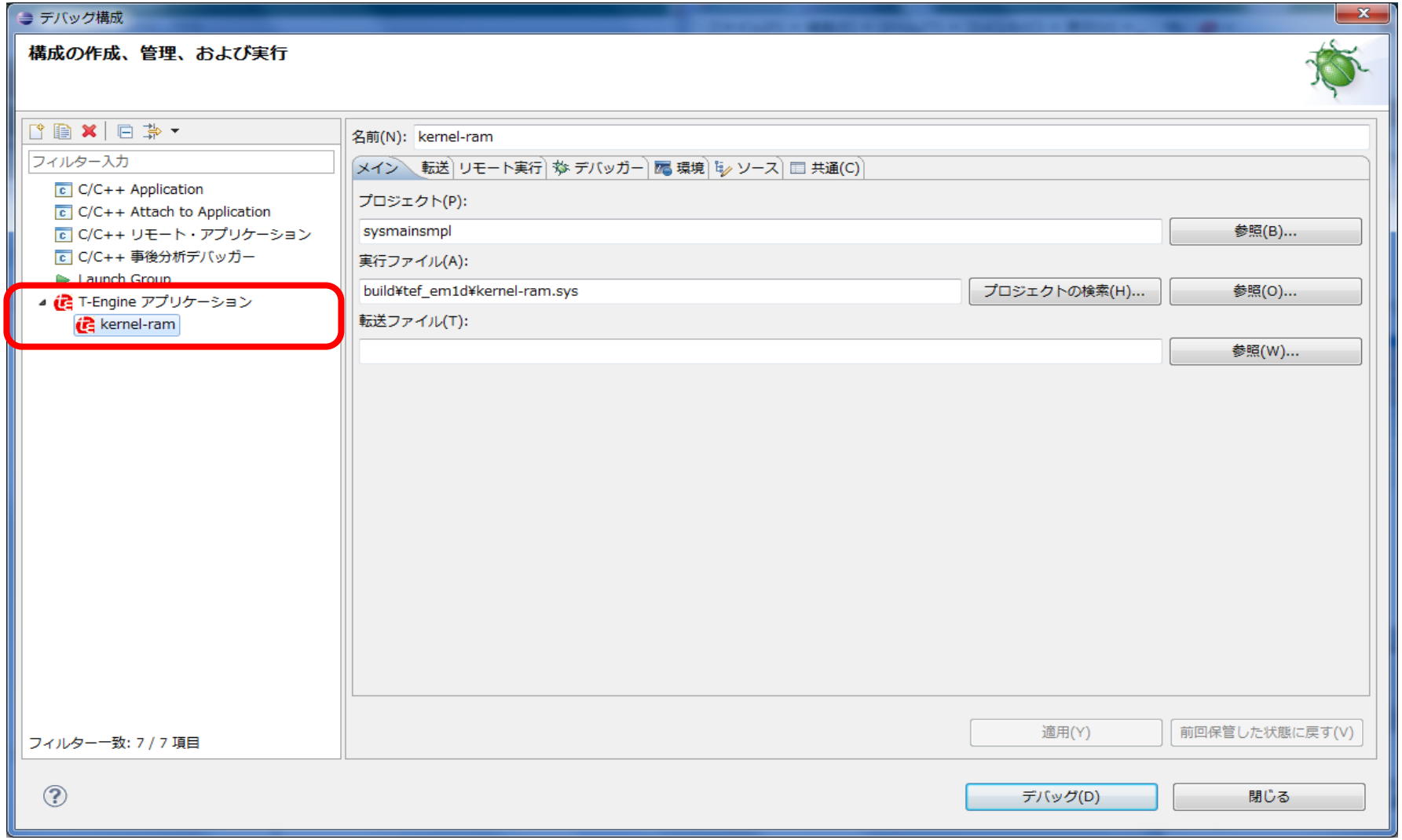
実行 (1/9)


T-Engine 開発 - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
T-Engine 開発
C/C
ナビ
プロ
アウトライン
表示するアウトラインはありません。
sysmainsmpl
バイナリー
build
tef_em1d
bitop.o - [arm/le]
icrt0_ram.o - [arm/le]
icrt0.o - [arm/le]
inittask_main.o - [arm/le]
kernel-ram.sys - [arm/le]
usermain.o -
Dependencie
kernel-ram.l
kernel-ram.r
kernel-ram.r
kernel-rom.l
kernel-rom.r
kernel-rom.r
kernel-rom.r
Makefile
src
usermainsmpl
新規(W
開く(O)
アプリケーションから開く(H)
Make ターゲット
コンテキストから除去 Ctrl+Alt+Shift+Down
コピー(C)
貼り付け(P)
削除(D)
移動(V)...
名前変更(M)...
インポート(I)...
エクスポート(O)...
リフレッシュ(F)
ブックマークの追加(K)...
実行(R)
デバッグ(D)
プロファイル(P)
選択したファイルのクリーン(L)
¥te¥tool¥Cygwin-i686¥etc¥gterm.exe
デバッグの構成(B)...
/sysmainsmpl/bui

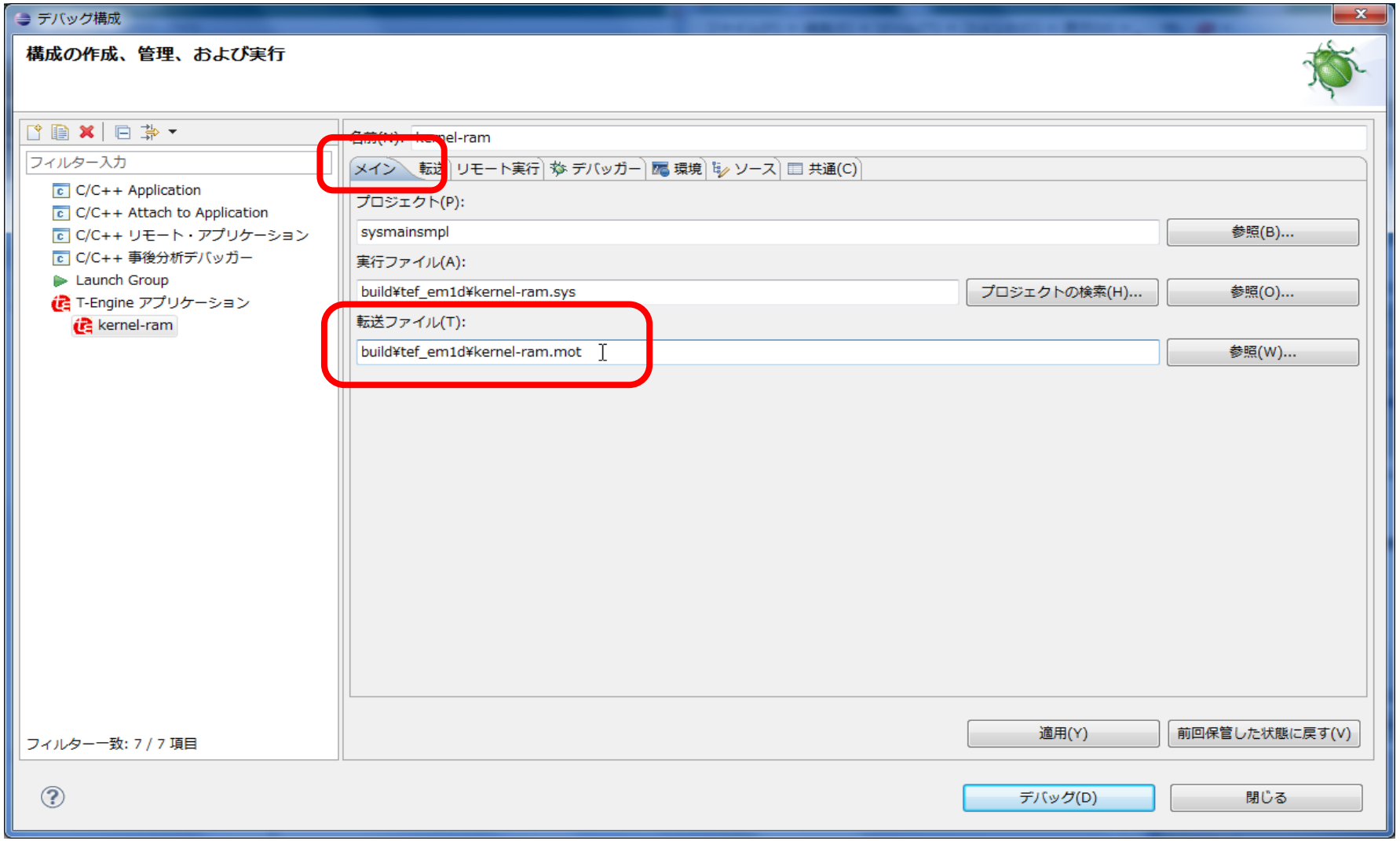
実行 (2/9)


デバッグ構成
構成の作成、管理、および実行
フィルター入力
C/C++ Application
C/C++ Attach to Application
C/C++ リモート・アプリケー
C/C++ 事後分析デバッガー
T-Engine アプリ
新規(W)
複製(D)
削除(T)
このダイアログから起動設定を構成します:
- 選択したタイプの構成を作成するには、「新規」ボタンを押します。
- 選択した構成をコピーするには、「複製」ボタンを押します。
- 選択した構成を除去するには、「削除」ボタンを押します。
- フィルター・オプションを構成するには、「フィルター」ボタンを押します。
- 既存構成を選択して、それを編集または表示します。
「パースペクティブ」設定ページから、起動パースペクティブの設定を構成します。
フィルター一致: 6 / 6 項目
デバッグ(D)
閉じる

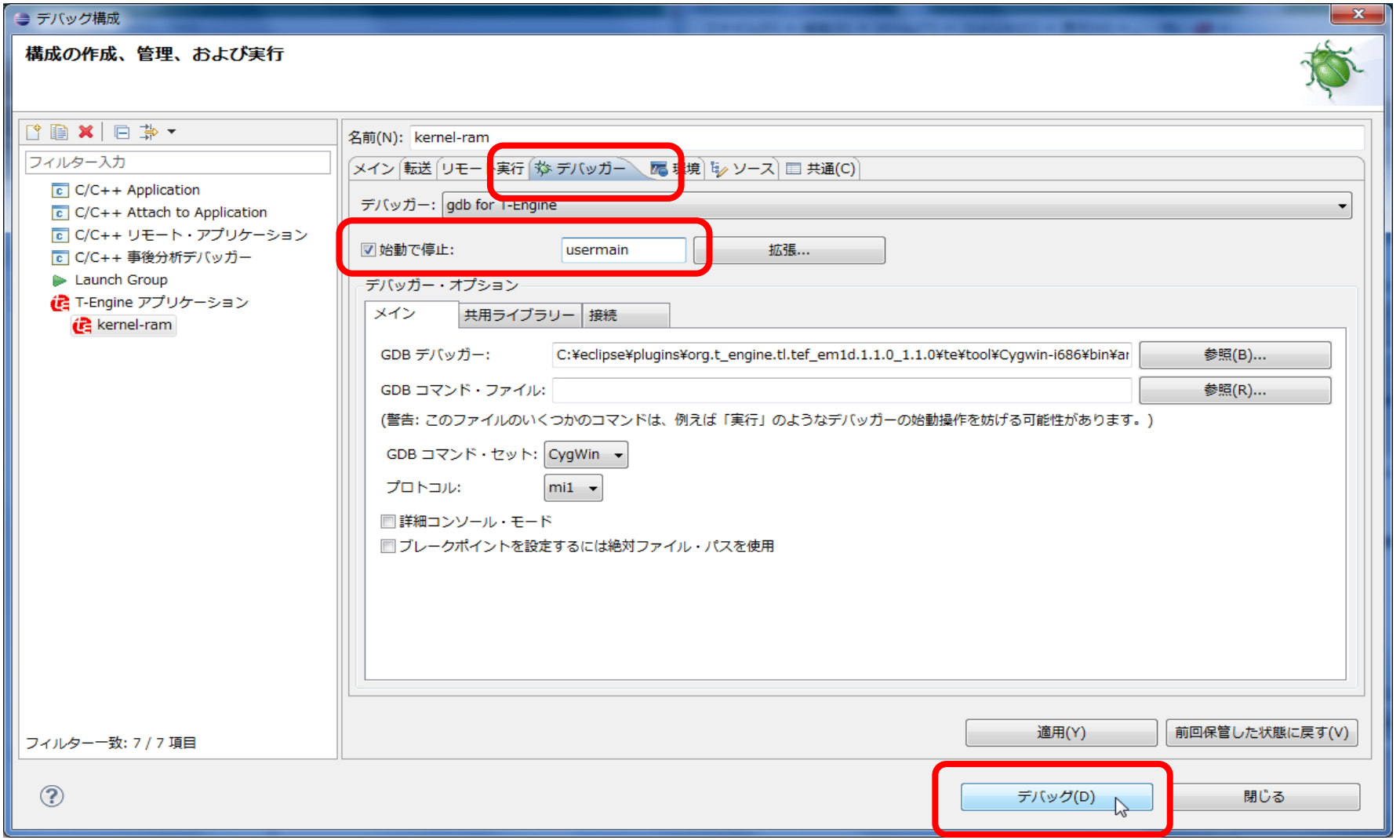
実行 (3/9)


デバッグ構成
構成の作成、管理、および実行
フィルター入力
C/C++ Application
C/C++ Attach to Application
C/C++ リモート・アプリケーション
C/C++ 事後分析デバッガー
Launch Group
T-Engine アプリケーション
kernel-ram
フィルター一致: 7 / 7 項目
名前(N): kernel-ram
メイン
転送
リモート実行
デバッガー
環境
ソース
共通(C)
プロジェクト(P):
sysmainsmpl
参照(B)...
実行ファイル(A):
build¥tef_em1d¥kernel-ram.sys
プロジェクトの検索(H)...
参照(O)...
転送ファイル(T):
参照(W)...
適用(Y)
前回保管した状態に戻す(V)
デバッグ(D)
閉じる

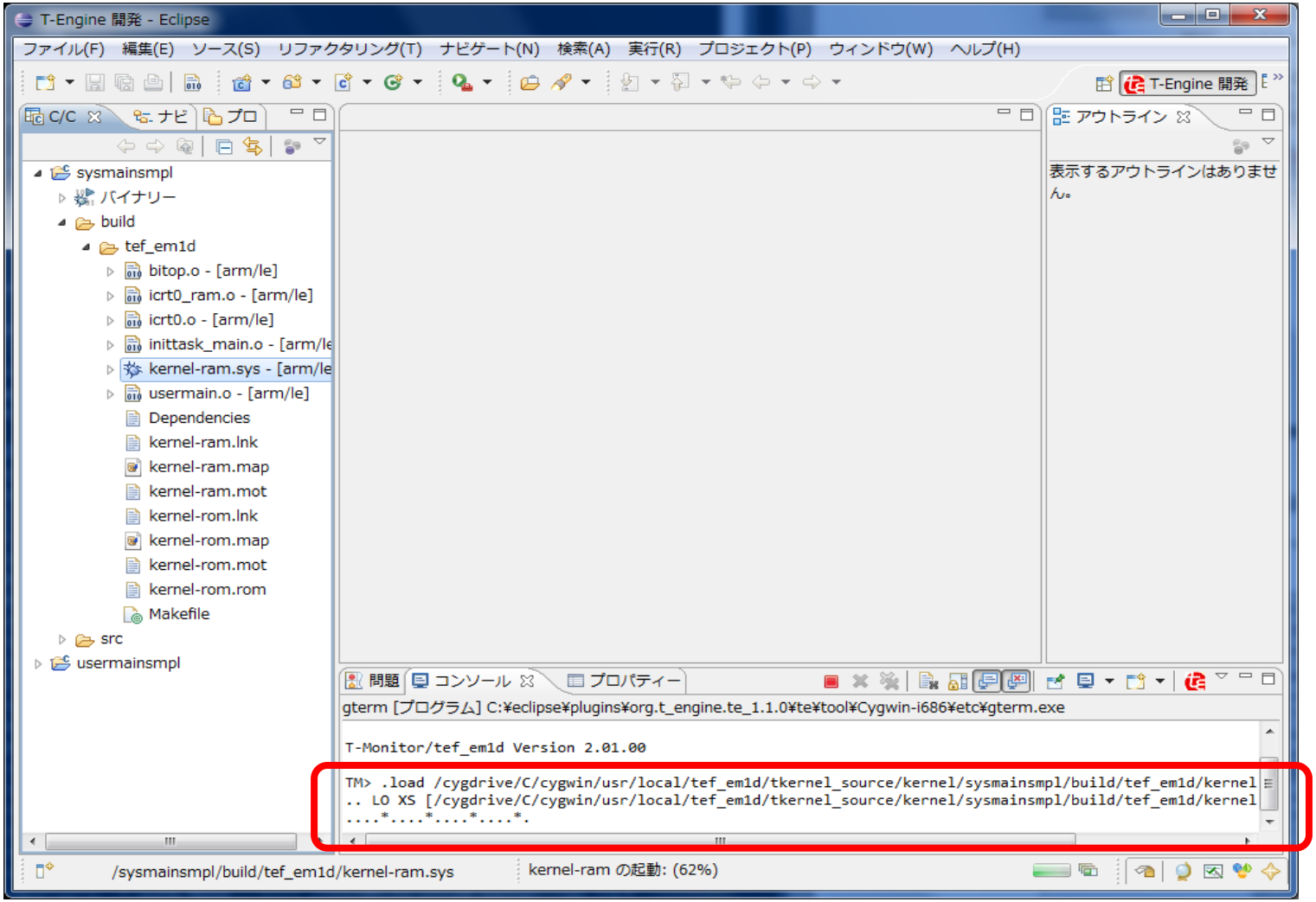
実行 (4/9)


デバッグ構成
構成の作成、管理、および実行
フィルター入力
C/C++ Application
C/C++ Attach to Application
C/C++ リモート・アプリケーション
C/C++ 事後分析デバッガー
Launch Group
T-Engine アプリケーション
kernel-ram
フィルター一致: 7 / 7 項目
el-ram
メイン
転送
リモート実行
デバッガー
環境
ソース
共通(C)
プロジェクト(P):
sysmainsmpl
参照(B)...
実行ファイル(A):
build¥tef_em1d¥kernel-ram.sys
プロジェクトの検索(H)...
参照(O)...
転送ファイル(T):
build¥tef_em1d¥kernel-ram.mot
参照(W)...
適用(Y)
前回保管した状態に戻す(V)
デバッグ(D)
閉じる

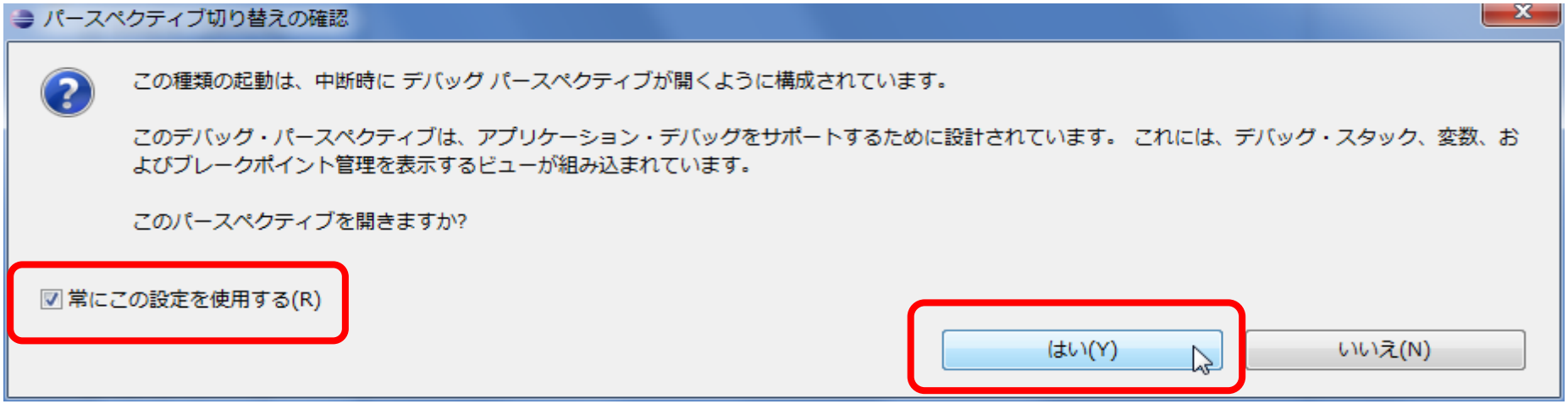
実行 (5/9)


デバッグ構成
構成の作成、管理、および実行
フィルター入力
C/C++ Application
C/C++ Attach to Application
C/C++ リモート・アプリケーション
C/C++ 事後分析デバッガー
Launch Group
T-Engine アプリケーション
kernel-ram
フィルター一致: 7 / 7 項目
名前(N): kernel-ram
メイン
転送
リモー
実行
デバッガー
環境
ソース
共通(C)
デバッガー: gdb for T-Engine
始動で停止:
usermain
拡張...
デバッガー・オプション
メイン
共用ライブラリー
接続
GDB デバッガー:
C:¥eclipse¥plugins¥org.t_engine.tl.tef_em1d.1.1.0_1.1.0¥te¥tool¥Cygwin-i686¥bin¥ar
参照(B)...
GDB コマンド・ファイル:
参照(R)...
(警告: このファイルのいくつかのコマンドは、例えば「実行」のようなデバッガーの始動操作を妨げる可能性があります。)
GDB コマンド・セット:
CygWin
プロトコル:
mi1
詳細コンソール・モード
ブレークポイントを設定するには絶対ファイル・パスを使用
適用(Y)
前回保管した状態に戻す(V)
デバッグ(D)
閉じる

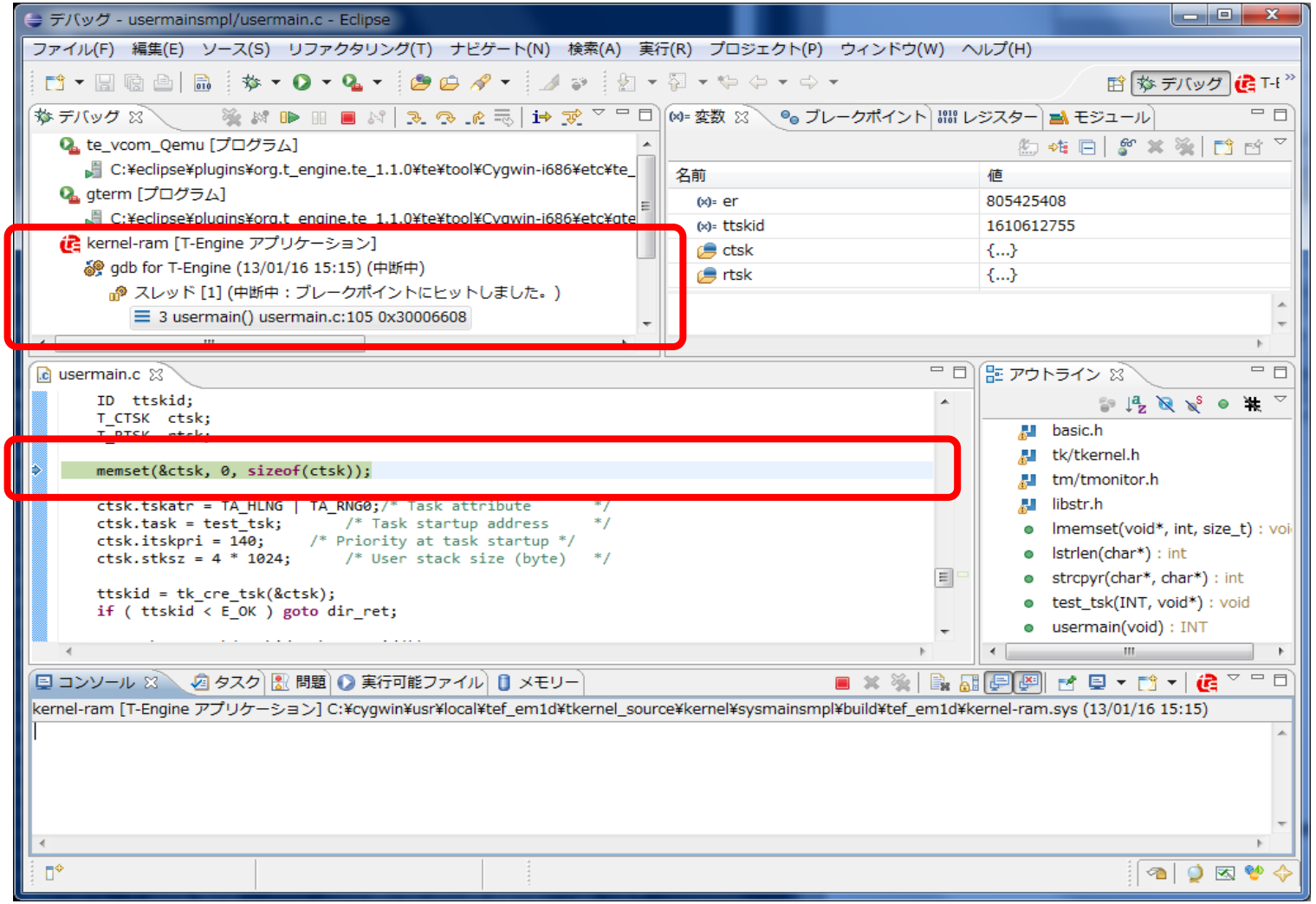
実行 (6/9)


T-Engine 開発 - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
T-Engine 開発
C/C ナビ プロ
sysmainsmpl
バイナリー
build
tef_em1d
bitop.o - [arm/le]
icrt0_ram.o - [arm/le]
icrt0.o - [arm/le]
inittask_main.o - [arm/le]
kernel-ram.sys - [arm/le]
usermain.o - [arm/le]
Dependencies
kernel-ram.lnk
kernel-ram.map
kernel-ram.mot
kernel-rom.lnk
kernel-rom.map
kernel-rom.mot
kernel-rom.rom
Makefile
src
usermainsmpl
アウトライン
表示するアウトラインはありません。
問題 コンソール プロパティー
gterm [プログラム] C:¥eclipse¥plugins¥org.t_engine.te_1.1.0¥te¥tool¥Cygwin-i686¥etc¥gterm.exe
T-Monitor/tef_em1d Version 2.01.00
TM> .load /cygdrive/C/cygwin/usr/local/tef_em1d/tkernel_source/kernel/sysmainsmpl/build/tef_em1d/kernel
.. LO XS [/cygdrive/C/cygwin/usr/local/tef_em1d/tkernel_source/kernel/sysmainsmpl/build/tef_em1d/kernel
....*....*....*....*.
/sysmainsmpl/build/tef_em1d/kernel-ram.sys
kernel-ram の起動: (62%)

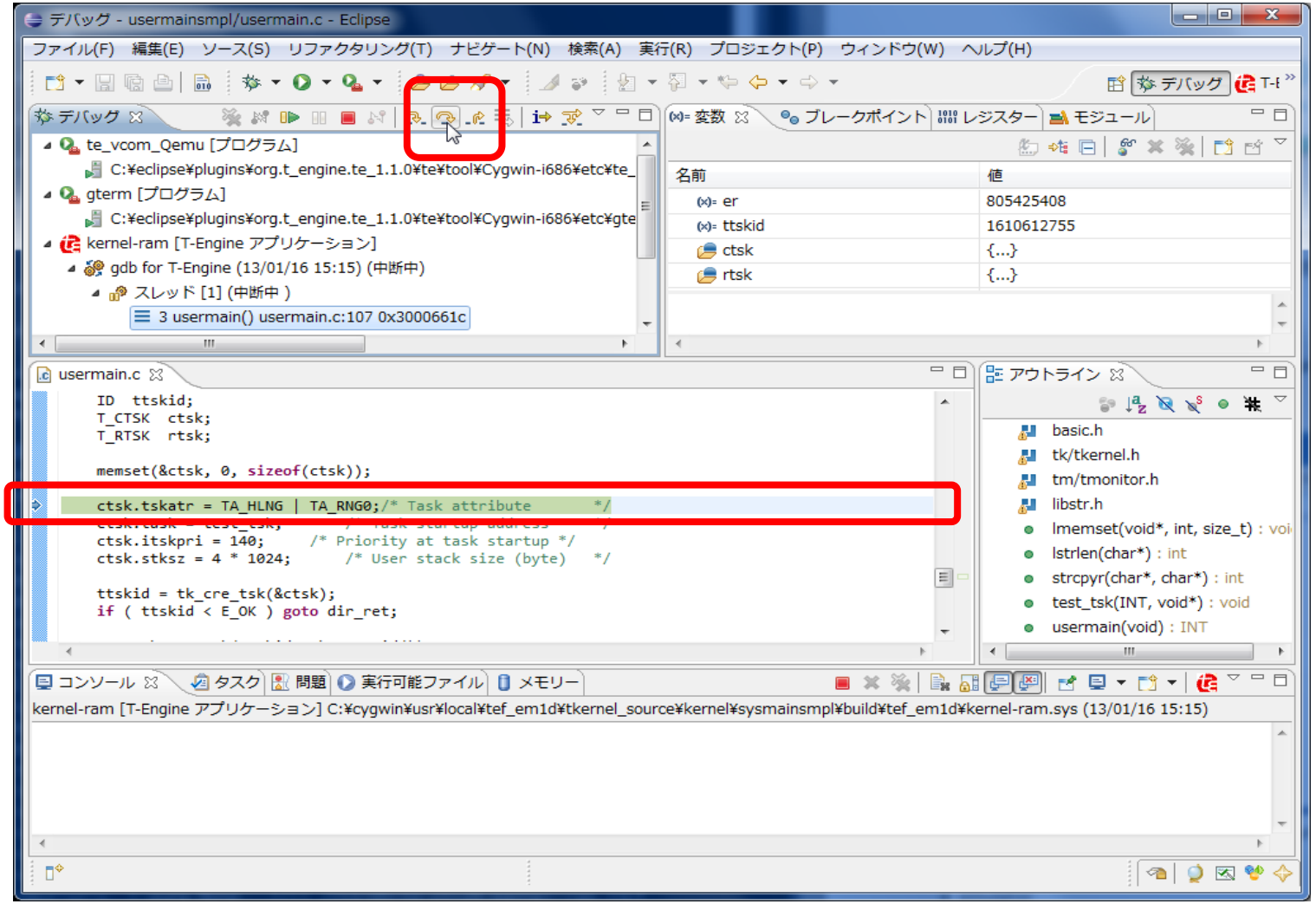
実行 (7/9)


パースペクティブ切り替えの確認
この種類の起動は、中断時に デバッグ パースペクティブが開くように構成されています。
このデバッグ・パースペクティブは、アプリケーション・デバッグをサポートするために設計されています。 これには、デバッグ・スタック、変数、およびブレークポイント管理を表示するビューが組み込まれています。
このパースペクティブを開きますか?
常にこの設定を使用する(R)
はい(Y)
いいえ(N)

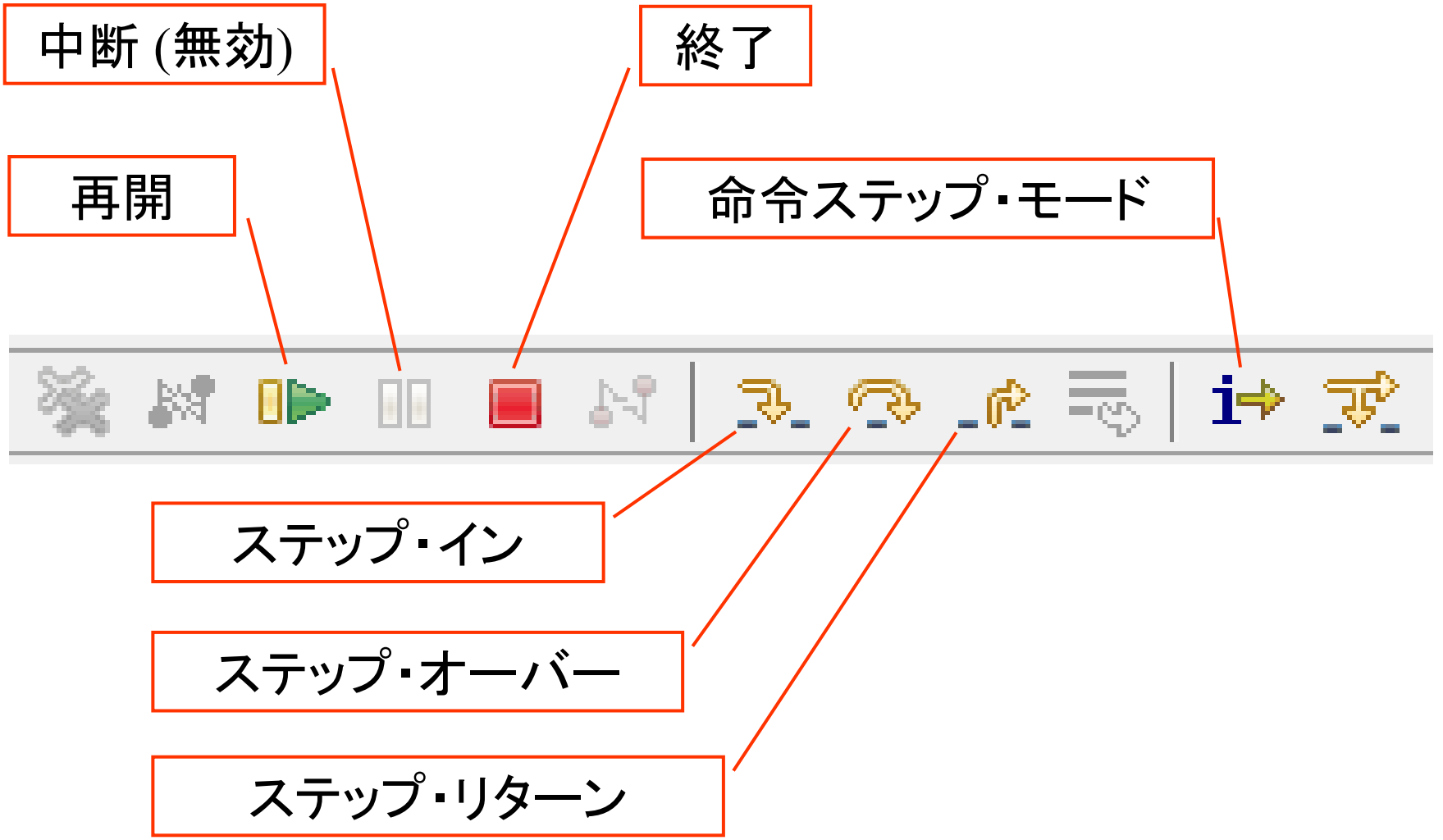
実行 (8/9)


デバッグ - usermainsmpl/usermain.c - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
デバッグ
te_vcom_Qemu [プログラム]
C:¥eclipse¥plugins¥org.t_engine.te_1.1.0¥te¥tool¥Cygwin-i686¥etc¥te_
gterm [プログラム]
kernel-ram [T-Engine アプリケーション]
gdb for T-Engine (13/01/16 15:15) (中断中)
スレッド [1] (中断中：ブレークポイントにヒットしました。)
3 usermain() usermain.c:105 0x30006608
変数
ブレークポイント
レジスター
モジュール
名前
値
er
805425408
ttskid
1610612755
ctsk
{...}
rtsk
{...}
usermain.c
ID ttskid;
T_CTSK ctsk;
memset(&ctsk, 0, sizeof(ctsk));
ctsk.tskatr = TA_HLNG | TA_RNG0;/* Task attribute */
ctsk.task = test_tsk; /* Task startup address */
ctsk.itskpri = 140; /* Priority at task startup */
ctsk.stksz = 4 * 1024; /* User stack size (byte) */
ttskid = tk_cre_tsk(&ctsk);
if (ttskid < E_OK) goto dir_ret;
アウトライン
basic.h
tk/tkernel.h
tm/tmonitor.h
libstr.h
lmemset(void*, int, size_t) : voi
lstrlen(char*) : int
strcpyr(char*, char*) : int
test_tsk(INT, void*) : void
usermain(void) : INT
コンソール
タスク
問題
実行可能ファイル
メモリー
kernel-ram [T-Engine アプリケーション] C:¥cygwin¥usr¥local¥tef_em1d¥tkernel_source¥kernel¥sysmainsmpl¥build¥tef_em1d¥kernel-ram.sys (13/01/16 15:15)

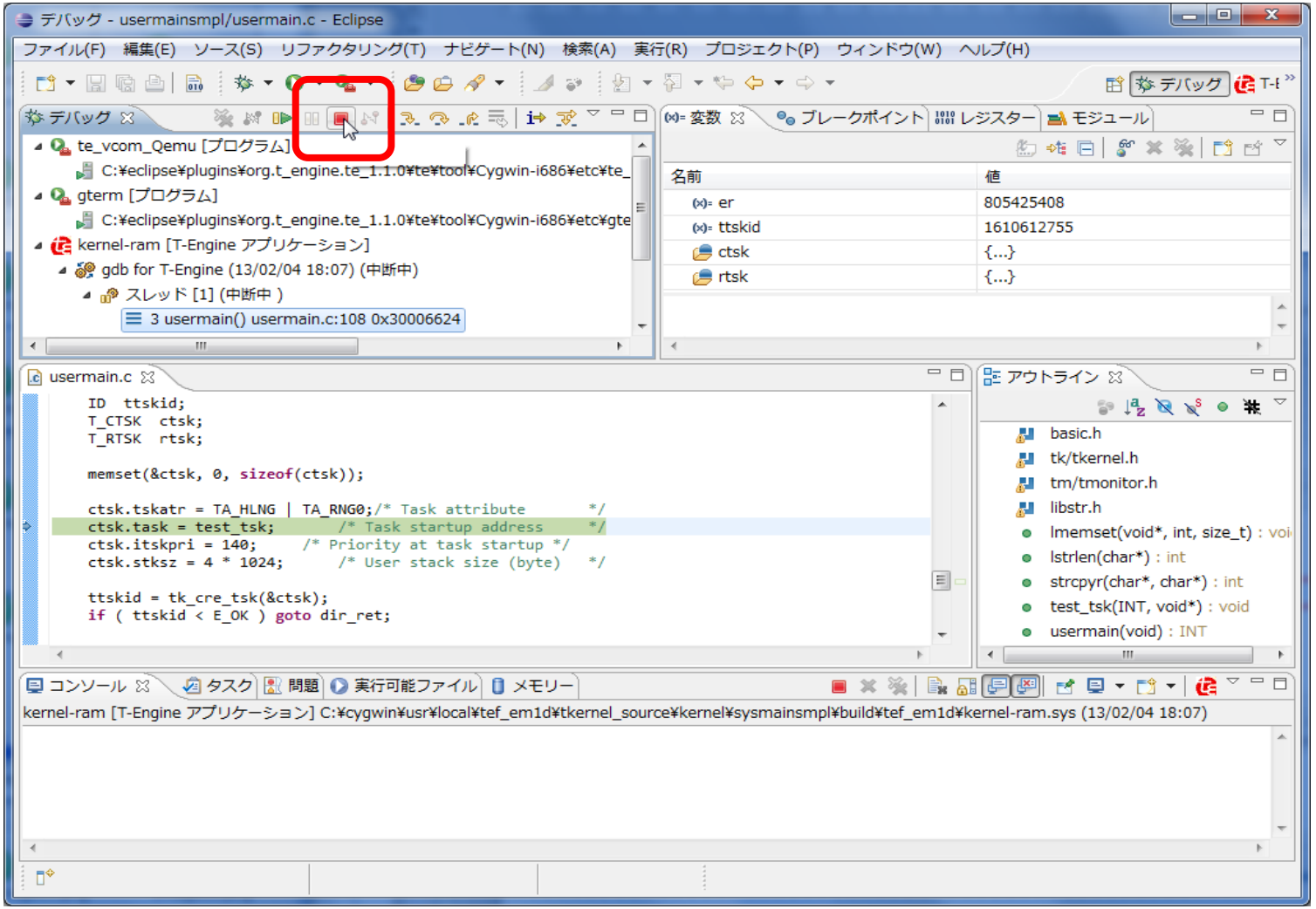
実行 (9/9)


デバッグ - usermainsmpl/usermain.c - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
デバッグ
te_vcom_Qemu [プログラム]
C:¥eclipse¥plugins¥org.t_engine.te_1.1.0¥te¥tool¥Cygwin-i686¥etc¥te_
gterm [プログラム]
C:¥eclipse¥plugins¥org.t_engine.te_1.1.0¥te¥tool¥Cygwin-i686¥etc¥gte
kernel-ram [T-Engine アプリケーション]
gdb for T-Engine (13/01/16 15:15) (中断中)
スレッド [1] (中断中)
3 usermain() usermain.c:107 0x3000661c
変数
ブレークポイント
レジスター
モジュール
名前
値
er
805425408
ttskid
1610612755
ctsk
{...}
rtsk
{...}
usermain.c
ID ttskid;
T_CTSK ctsk;
T_RTSK rtsk;
memset(&ctsk, 0, sizeof(ctsk));
ctsk.tskatr = TA_HLNG | TA_RNG0;/* Task attribute */
ctsk.itskpri = 140; /* Priority at task startup */
ctsk.stksz = 4 * 1024; /* User stack size (byte) */
ttskid = tk_cre_tsk(&ctsk);
if (ttskid < E_OK) goto dir_ret;
アウトライン
basic.h
tk/tkernel.h
tm/tmonitor.h
libstr.h
lmemset(void*, int, size_t) : voi
lstrlen(char*) : int
strcpyr(char*, char*) : int
test_tsk(INT, void*) : void
usermain(void) : INT
コンソール
タスク
問題
実行可能ファイル
メモリー
kernel-ram [T-Engine アプリケーション] C:¥cygwin¥usr¥local¥tef_em1d¥tkernel_source¥kernel¥sysmainsmpl¥build¥tef_em1d¥kernel-ram.sys (13/01/16 15:15)

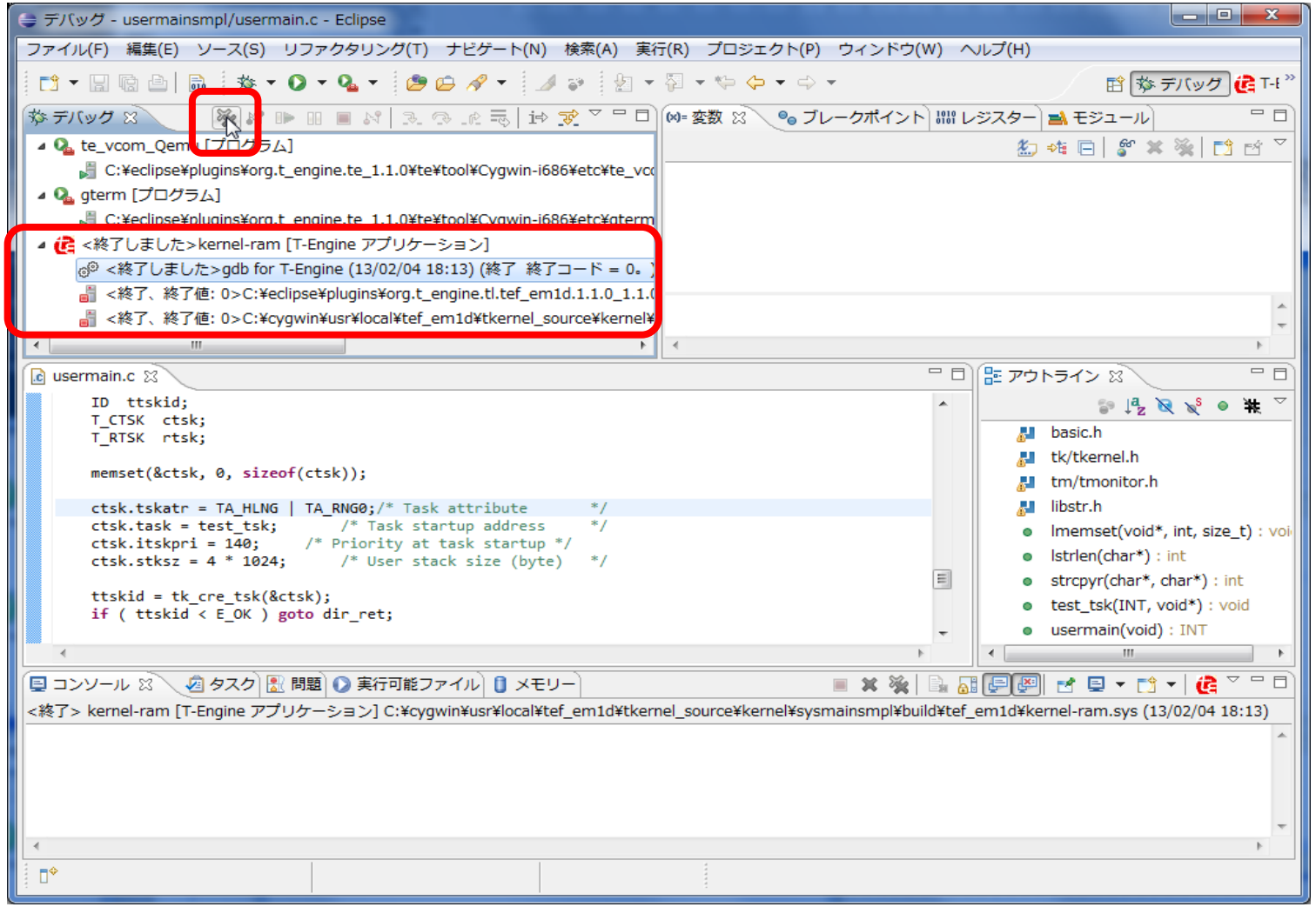
デバッグ操作



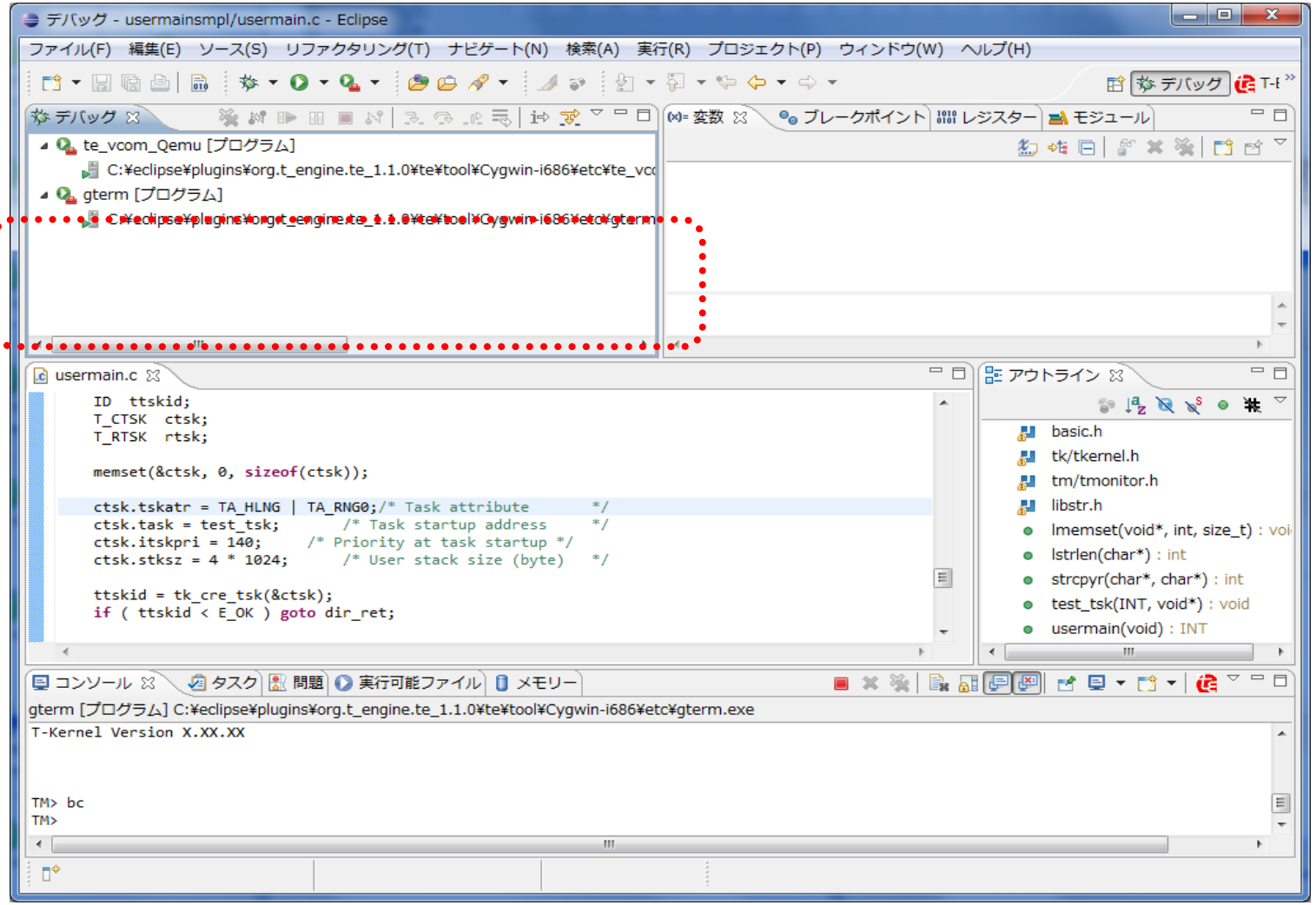
アプリケーションの終了（1/3）



アプリケーションの終了 (2/3)



アプリケーションの終了（3/3）



アプリケーションの追加

usermain.cに
アプリケーション
タスクを追加

タスクの生成と起動

- tk_cre_tsk
- tk_sta_tsk

- Eclipseでデバッグ

usermain.cにタスクを追加

- タスクの基本形

```
void  taskA( INT stacd, VP exinf )
{
        /* 処理 */
        tk_ext_tsk();
}
```

追加するタスク

- 無限ループと待ちも追加

```
void  taskA( INT stacd, VP exinf )
{
        while(1){
            tk_dly_tsk( 100 );
        }
        tk_ext_tsk();
}
```

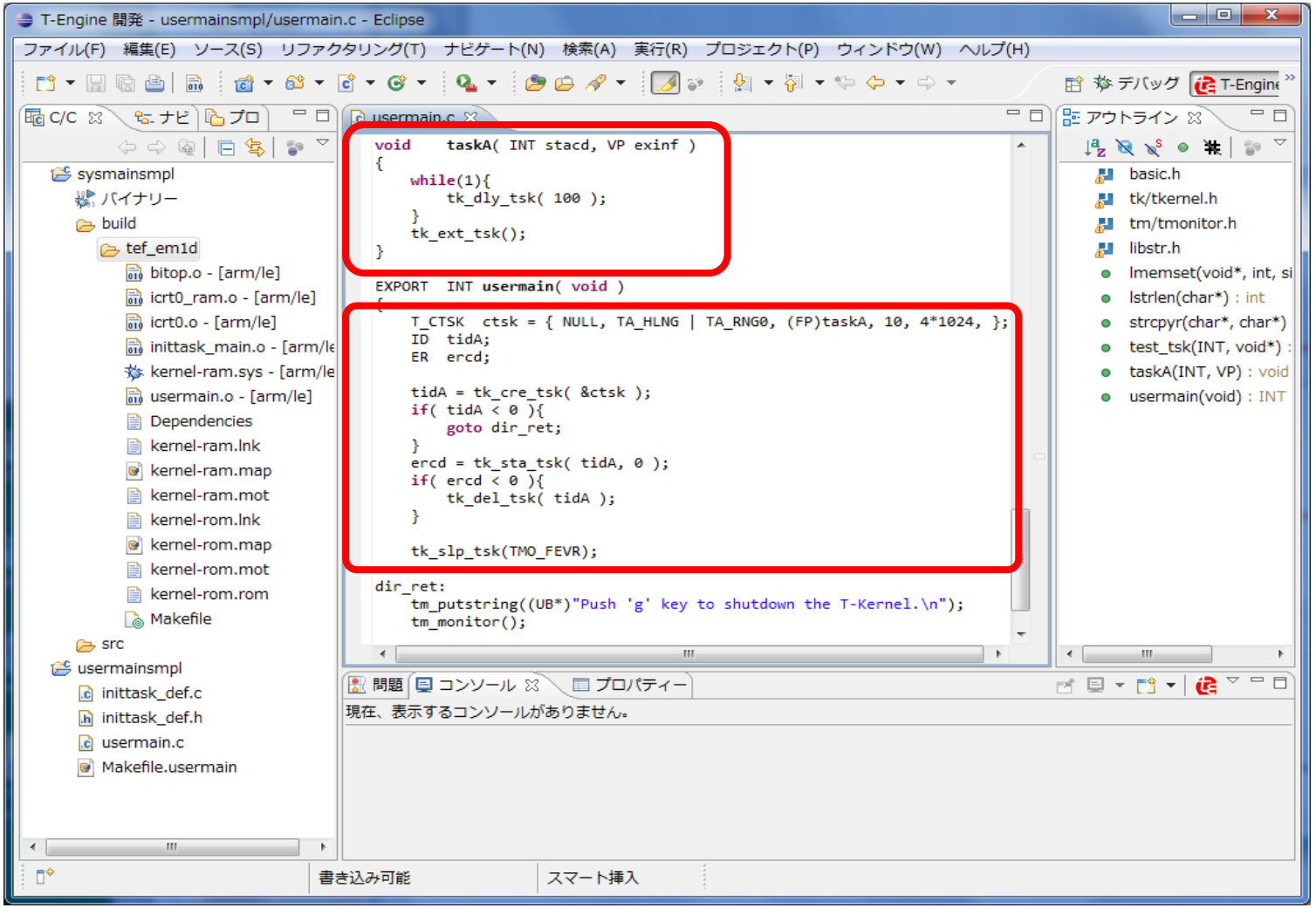
usermain() でタスクを生成・起動

```
EXPORT INT    usermain( void )
{
        T_CTSK ctsk = { NULL, TA_HLNG | TA_RNG0,  (FP)taskA,  10,  4*1024,  };
        ID     tidA;
        ER     ercd;

        tidA = tk_cre_tsk( &ctsk );
        if( tidA < 0 ){
                goto dir_ret;
        }
        ercd = tk_sta_tsk( tidA, 0 );
        if( ercd < 0 ){
                tk_del_tsk( tidA );
        }

        tk_slp_tsk(TMO_FEVR);

        tm_putstring((UB*)"Push 'g' key to shutdown the T-Kernel.¥n");
        tm_monitor();

        return 0;
}
```

以下を設定しておけばタスクとしては起動する。

- tskatr　タスク属性
- func　タスクにする関数の先頭アドレス (関数へのポインタ)
- itskpri　初期タスク優先度
- stksz　スタックサイズ

taskAを追加


T-Engine 開発 - usermainsmpl/usermain.c - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
C/C ナビ プロ
sysmainsmpl
バイナリー
build
tef_em1d
bitop.o - [arm/le]
icrt0_ram.o - [arm/le]
icrt0.o - [arm/le]
inittask_main.o - [arm/le
kernel-ram.sys - [arm/le
usermain.o - [arm/le]
Dependencies
kernel-ram.lnk
kernel-ram.map
kernel-ram.mot
kernel-rom.lnk
kernel-rom.map
kernel-rom.mot
kernel-rom.rom
Makefile
src
usermainsmpl
inittask_def.c
inittask_def.h
usermain.c
Makefile.usermain
usermain.c
void taskA(INT stacd, VP exinf)
{
while(1){
tk_dly_tsk(100);
}
tk_ext_tsk();
}
EXPORT INT usermain(void)
{
T_CTSK ctsk = { NULL, TA_HLNG | TA_RNG0, (FP)taskA, 10, 4*1024, };
ID tidA;
ER ercd;
tidA = tk_cre_tsk(&ctsk);
if(tidA < 0){
goto dir_ret;
}
ercd = tk_sta_tsk(tidA, 0);
if(ercd < 0){
tk_del_tsk(tidA);
}
tk_slp_tsk(TMO_FEVR);
dir_ret:
tm_putstring((UB*)"Push 'g' key to shutdown the T-Kernel.\n");
tm_monitor();
アウトライン
basic.h
tk/tkernel.h
tm/tmonitor.h
libstr.h
lmemset(void*, int, si
lstrlen(char*) : int
strcpyr(char*, char*)
test_tsk(INT, void*) :
taskA(INT, VP) : void
usermain(void) : INT
問題 コンソール プロパティー
現在、表示するコンソールがありません。
書き込み可能
スマート挿入

ビルドして実行

- ビルド
  - tef_em1d を選択
  - [プロジェクト] → [T-Engine Target の Make all]
- デバッグ (実行)
  - kernel-ram.sys を選択
  - [デバッグ] → [デバッグの構成] → [デバッグ]
- 動作確認
  - taskA にブレークポイントを設定
  - 実行
- 終了

タスクをもう一つ追加

- taskA と同じ内容の taskBを
  taskA() と usermain() の間に追加

```
void  taskB( INT stacd, VP exinf )
{
        while(1){
            tk_dly_tsk( 200 );
        }
        tk_ext_tsk();
}
```

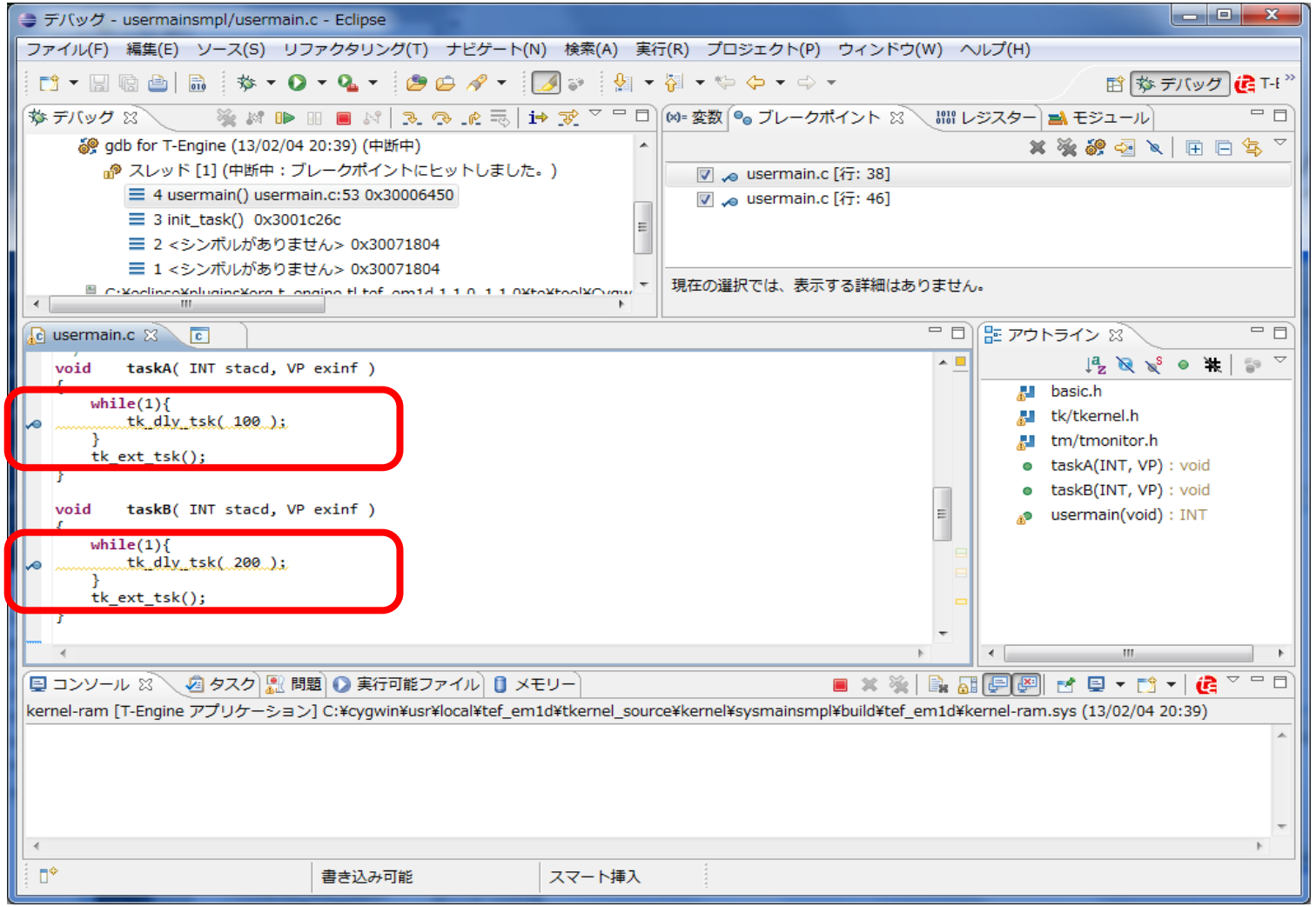
taskBも生成・起動

```
EXPORT INT    usermain( void )
{
        T_CTSK ctsk = { NULL, TA_HLNG | TA_RNG0, NULL, 10, 4*1024, };
        ID     tidA;
        ID     tidB;
        ER     ercd;

        ctsk.task = (FP)taskA;
        tidA = tk_cre_tsk( &ctsk );
        ercd = tk_sta_tsk( tidA, 0 );

        ctsk.task = (FP)taskB;
        tidB = tk_cre_tsk( &ctsk );
        ercd = tk_sta_tsk( tidB, 0 );

        tk_slp_tsk(TMO_FEVR);

        tm_putstring((UB*)"Push 'g' key to shutdown the T-Kernel.¥n");
        tm_monitor();

        return 0;
}
```

ビルドして実行

- ビルド
  - tef_em1d を選択
  - [プロジェクト] → [T-Engine Target の Make all]
- デバッグ (実行)
  - kernel-ram.sys を選択
  - [デバッグ] → [デバッグの構成] → [デバッグ]
- 動作確認
  - taskA と taskB にブレークポイントを設定
  - 実行
- 終了

taskA と taskB にブレークポイントを設定



Tips

- ウィンドウ → ビューの表示 → 逆アセンブル
- 変数 → フォーマット → 自然 / 10進数 / 16進数
  - 16進数のエラーコード一覧があると便利 → Appendix. A
- プログラムが起動しない
  - シグナル発生
    - EclipseとQemuの再起動では復活しない
  - The program is not being run.

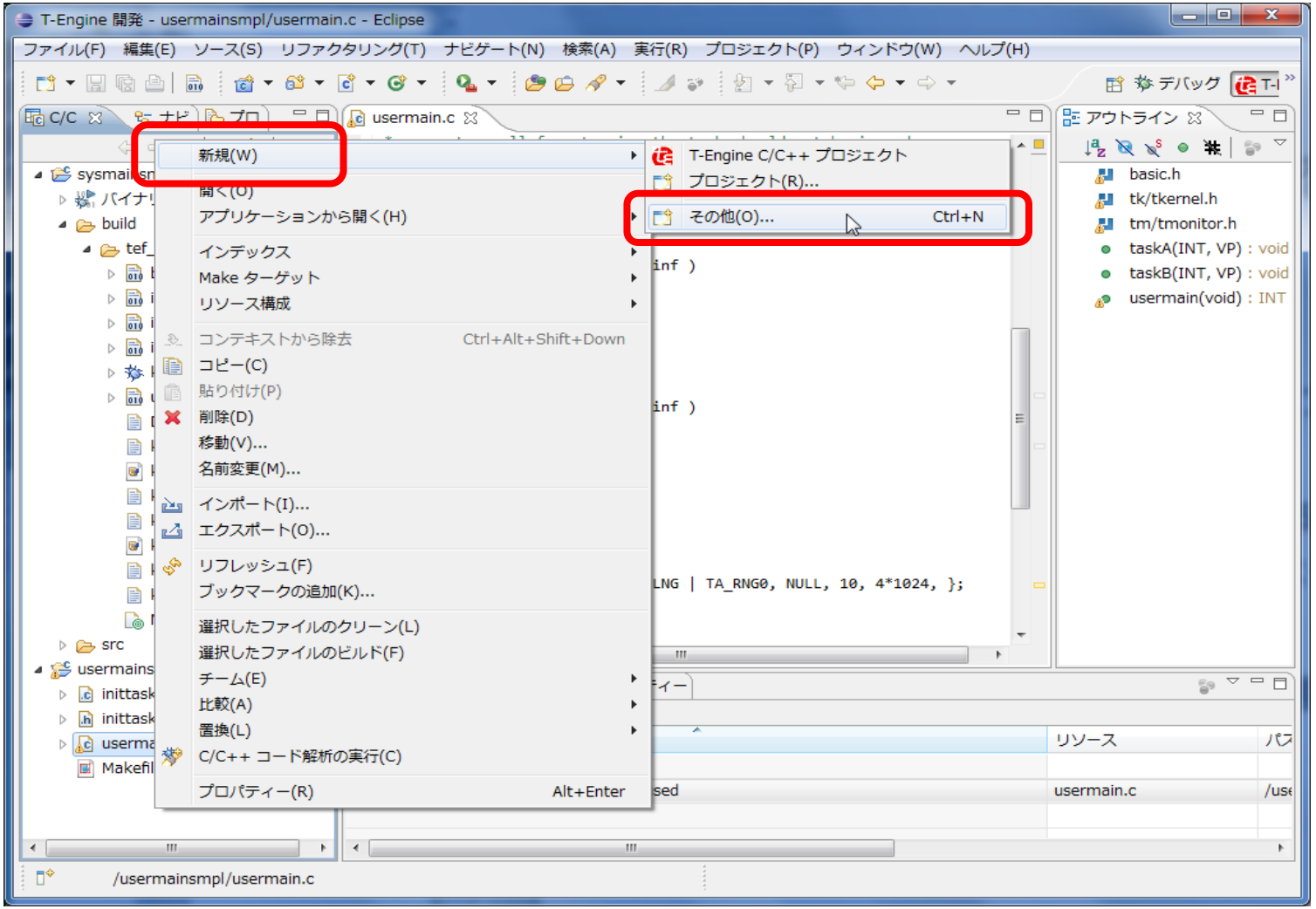
  - .te_vcom ファイルが無い → Appendix. B
  - プロセスが残っている → 削除 or ログオフ or PCのリセット

アプリケーションを
別ファイルで追加

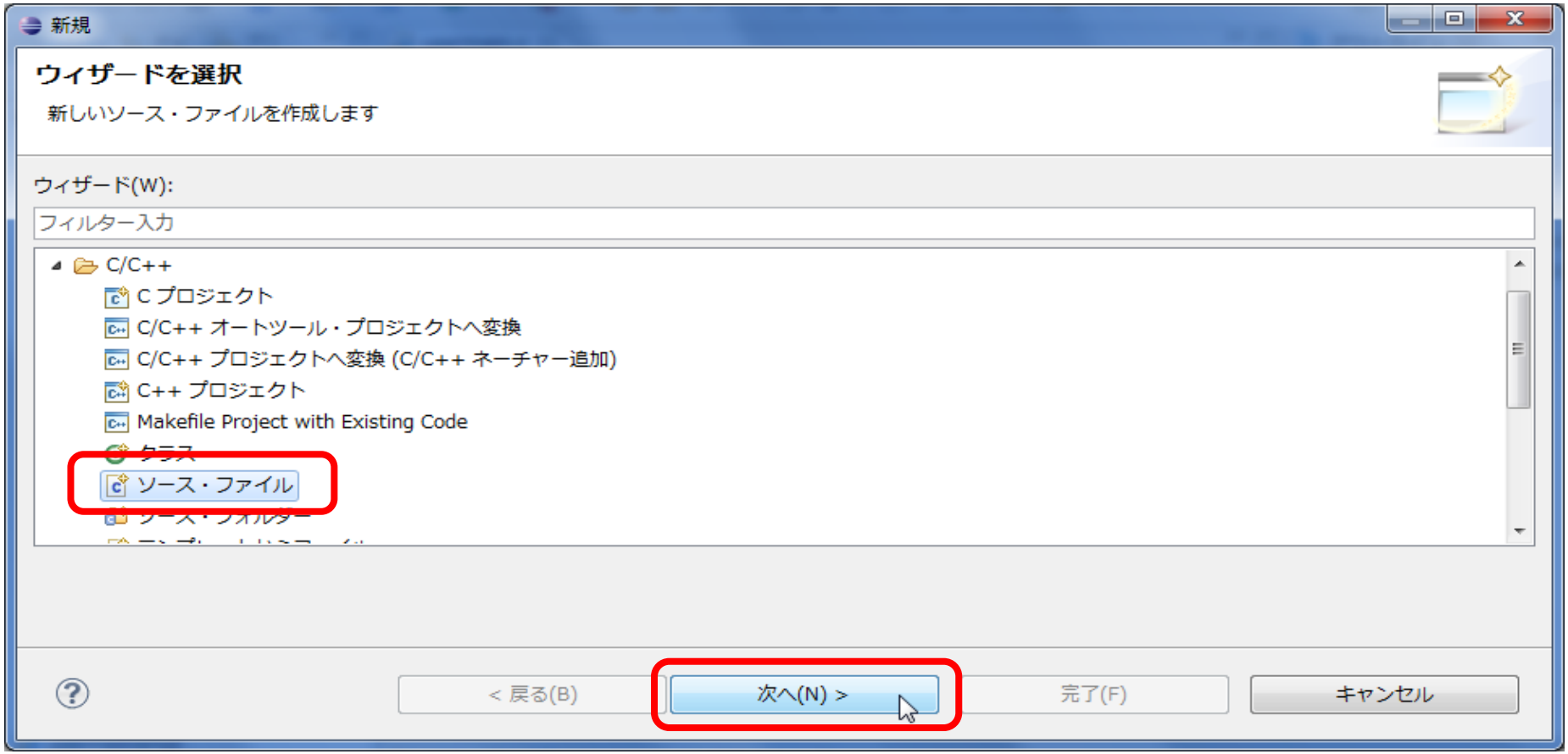
ソースファイルを追加

- Eclipseのメニューから追加
- usermain.c から taskAとtaskBを移動
  - ヘッダファイルのインクルードはそのままコピー
- usermain.c の編集
  - taskAとtaskBのプロトタイプ宣言を追加
  - taskAとtaskBを削除
  - 未使用の関数も削除

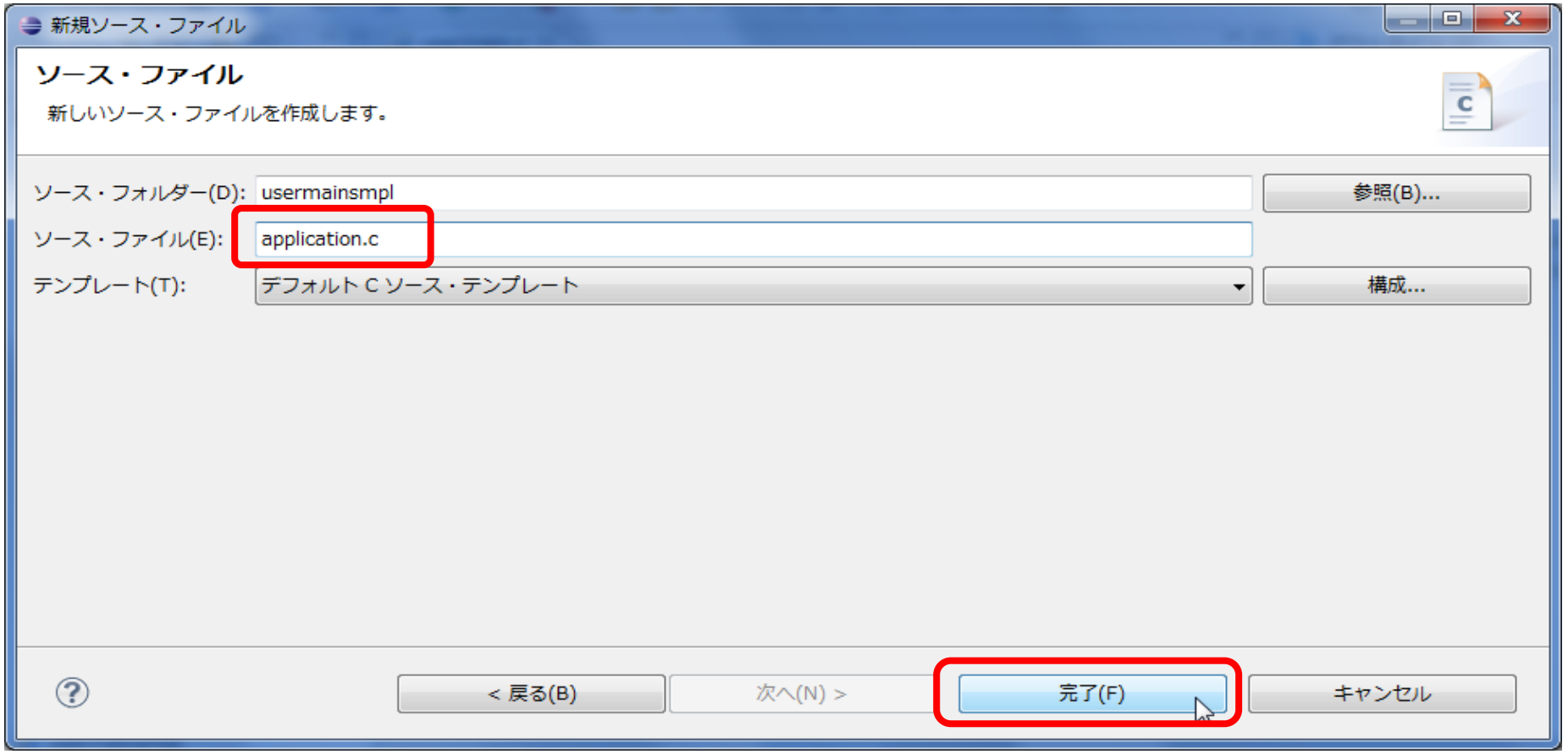
新しいソースファイルの追加 (1/5)



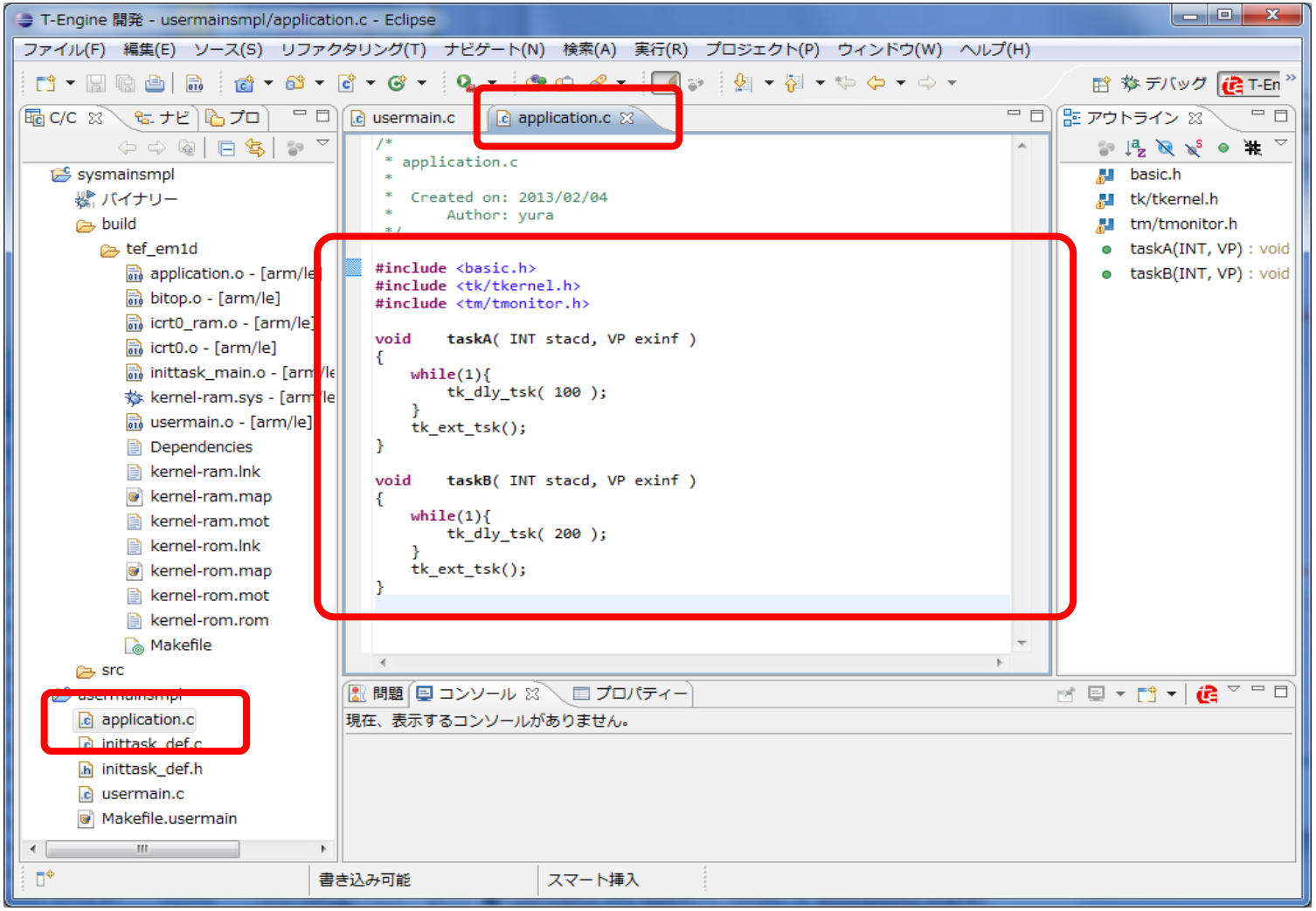
新しいソースファイルの追加 (2/5)



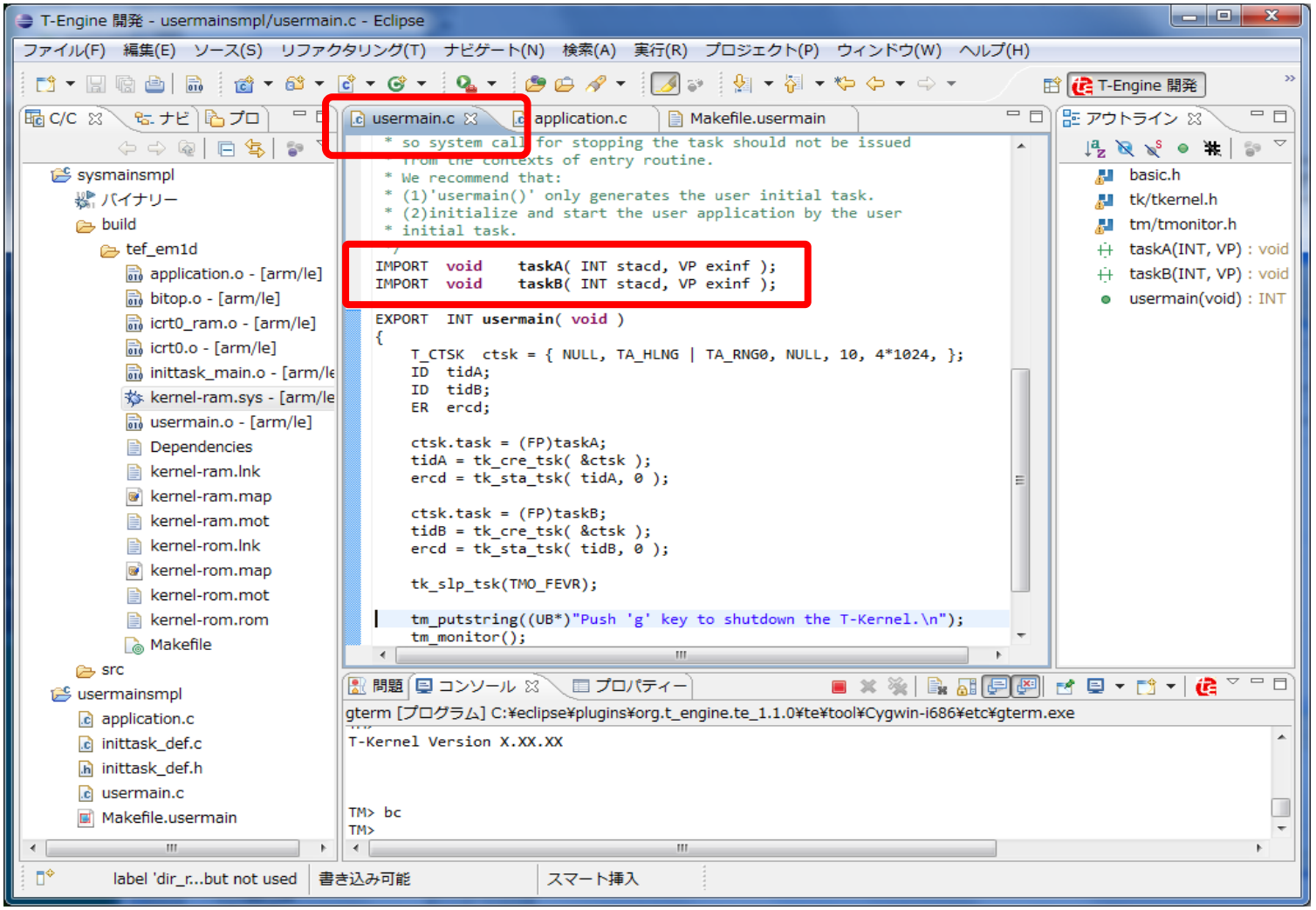
新しいソースファイルの追加 (3/5)



新しいソースファイルの追加 (4/5)



新しいソースファイルの追加 (4/5)

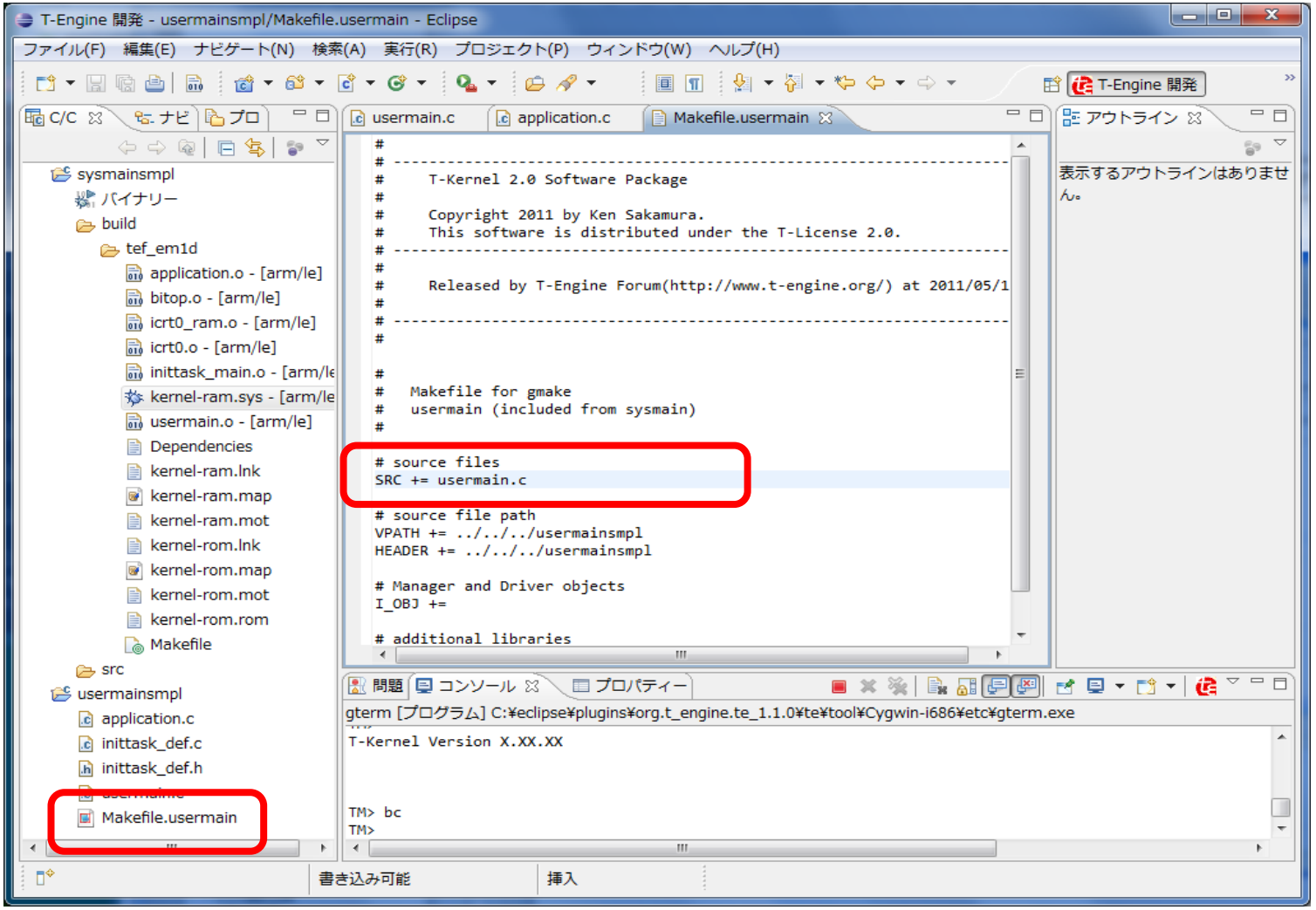


Makefile.usermainを編集

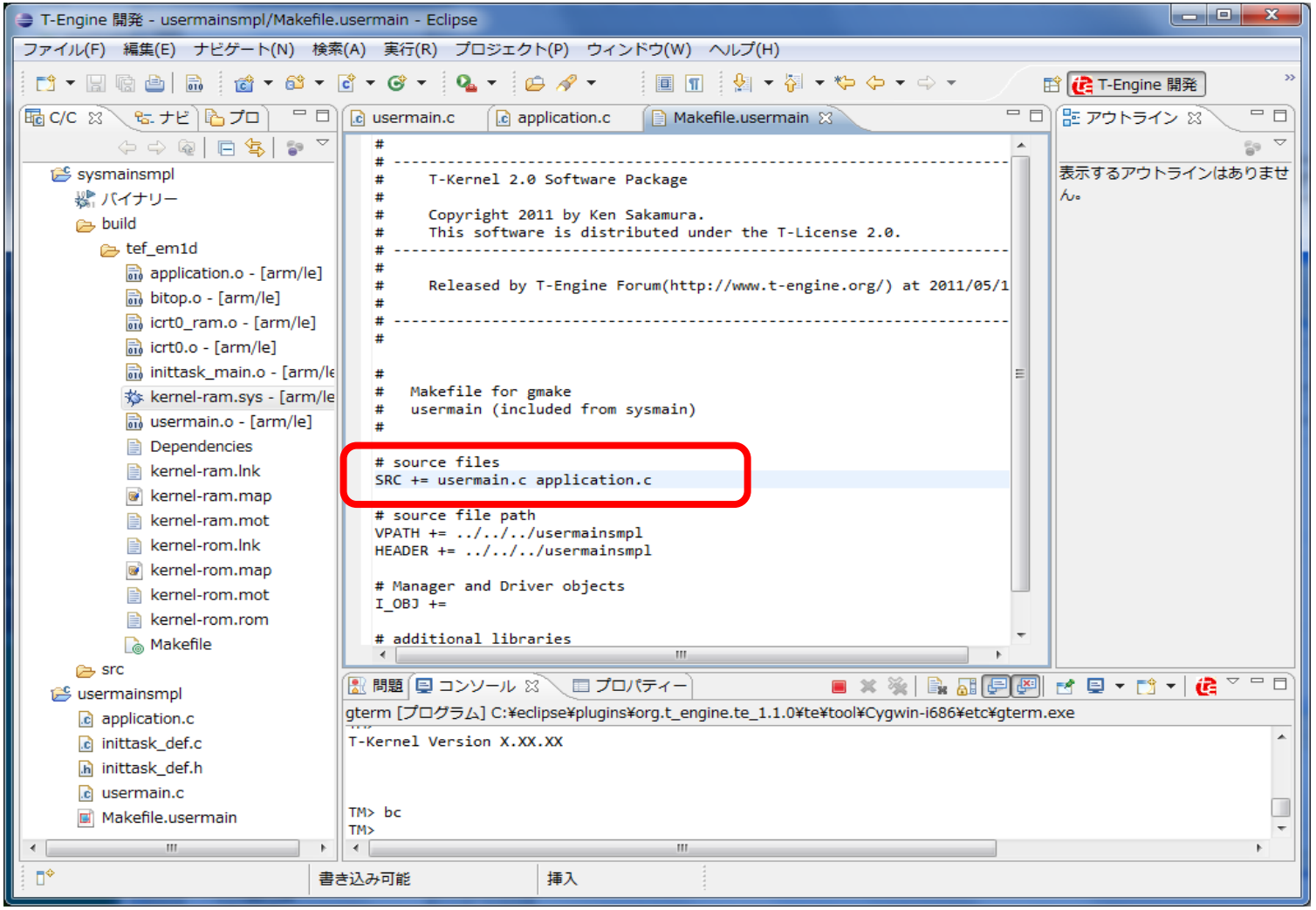
- SRCに application.c を追加

SRC += usermain.c application.c

SRCにapplication.cを追加 (1/2)



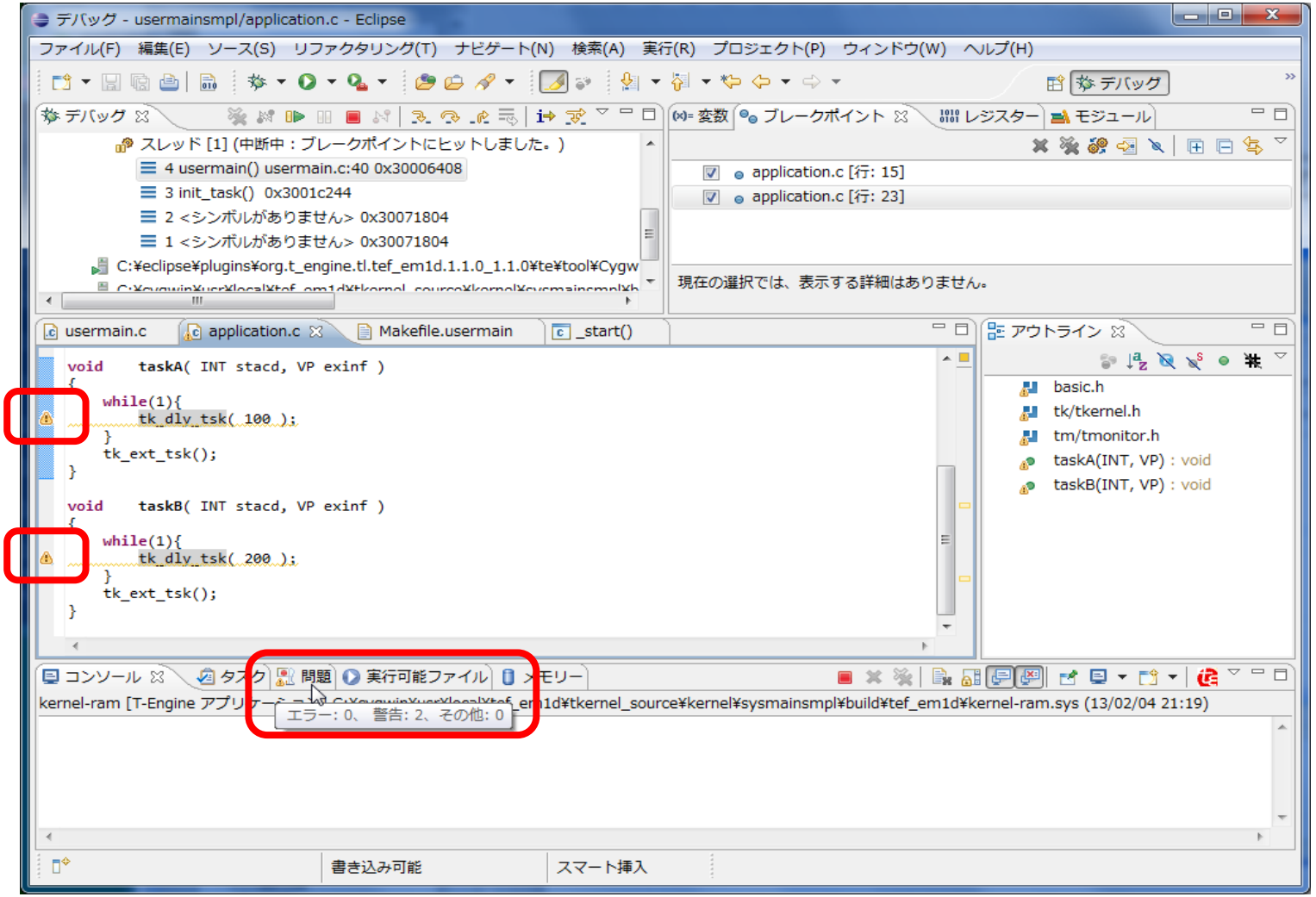
SRCにapplication.cを追加 (2/2)



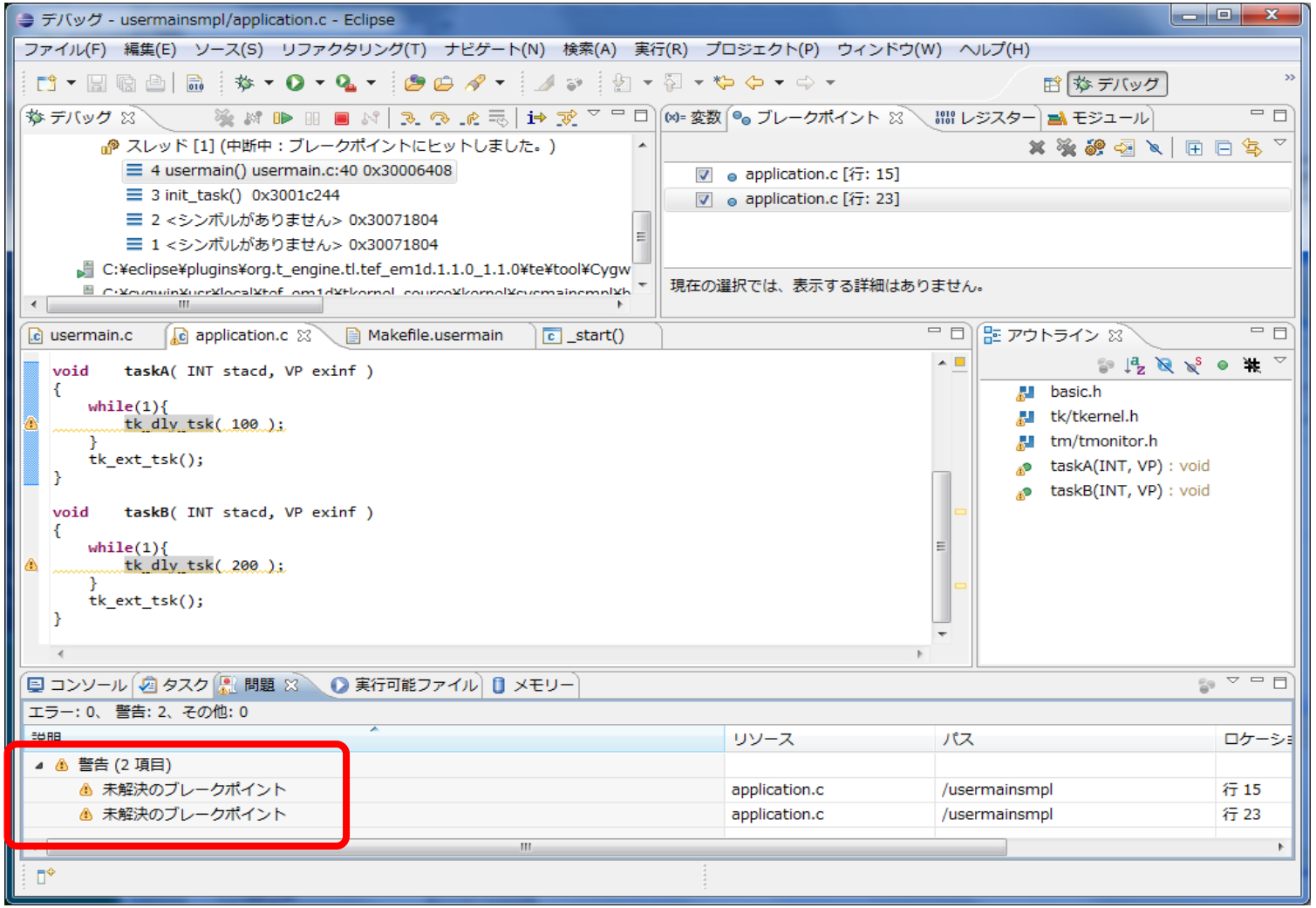
ビルドして実行

- ビルド
  - tef_em1d を選択
  - [プロジェクト] → [T-Engine Target の Make all]
- デバッグ (実行)
  - kernel-ram.sys を選択
  - [デバッグ] → [デバッグの構成] → [デバッグ]
- 動作確認
  - taskA と taskB にブレークポイントを設定
  - 実行
- 終了

警告？


デバッグ - usermainsmpl/application.c - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
デバッグ
スレッド [1] (中断中：ブレークポイントにヒットしました。)
4 usermain() usermain.c:40 0x30006408
3 init_task() 0x3001c244
2 <シンボルがありません> 0x30071804
1 <シンボルがありません> 0x30071804
C:¥eclipse¥plugins¥org.t_engine.tl.tef_em1d.1.1.0_1.1.0¥te¥tool¥Cygw
変数 ブレークポイント レジスター モジュール
application.c [行: 15]
application.c [行: 23]
現在の選択では、表示する詳細はありません。
usermain.c application.c Makefile.usermain _start()
void taskA(INT stacd, VP exinf)
{
while(1){
tk_dly_tsk(100);
}
tk_ext_tsk();
}
void taskB(INT stacd, VP exinf)
{
while(1){
tk_dly_tsk(200);
}
tk_ext_tsk();
}
アウトライン
basic.h
tk/tkernel.h
tm/tmonitor.h
taskA(INT, VP) : void
taskB(INT, VP) : void
コンソール タスク 問題 実行可能ファイル メモリー
エラー: 0、 警告: 2、その他: 0
kernel-ram [T-Engine アプリケーション] C:¥cygwin¥usr¥local¥tef_em1d¥tkernel_source¥kernel¥sysmainsmpl¥build¥tef_em1d¥kernel-ram.sys (13/02/04 21:19)
書き込み可能
スマート挿入

警告の内容を確認


デバッグ - usermainsmpl/application.c - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
デバッグ
スレッド [1] (中断中：ブレークポイントにヒットしました。)
4 usermain() usermain.c:40 0x30006408
3 init_task() 0x3001c244
2 <シンボルがありません> 0x30071804
1 <シンボルがありません> 0x30071804
C:¥eclipse¥plugins¥org.t_engine.tl.tef_em1d.1.1.0_1.1.0¥te¥tool¥Cygw
変数
ブレークポイント
レジスター
モジュール
application.c [行: 15]
application.c [行: 23]
現在の選択では、表示する詳細はありません。
usermain.c
application.c
Makefile.usermain
_start()
void taskA(INT stacd, VP exinf)
{
while(1){
tk_dly_tsk(100);
}
tk_ext_tsk();
}
void taskB(INT stacd, VP exinf)
{
while(1){
tk_dly_tsk(200);
}
tk_ext_tsk();
}
アウトライン
basic.h
tk/tkernel.h
tm/tmonitor.h
taskA(INT, VP) : void
taskB(INT, VP) : void
コンソール
タスク
問題
実行可能ファイル
メモリー
エラー: 0、 警告: 2、その他: 0
説明
リソース
パス
ロケーション
警告 (2 項目)
未解決のブレークポイント
application.c
/usermainsmpl
行 15
未解決のブレークポイント
application.c
/usermainsmpl
行 23

ブレークポイントが設定できない？

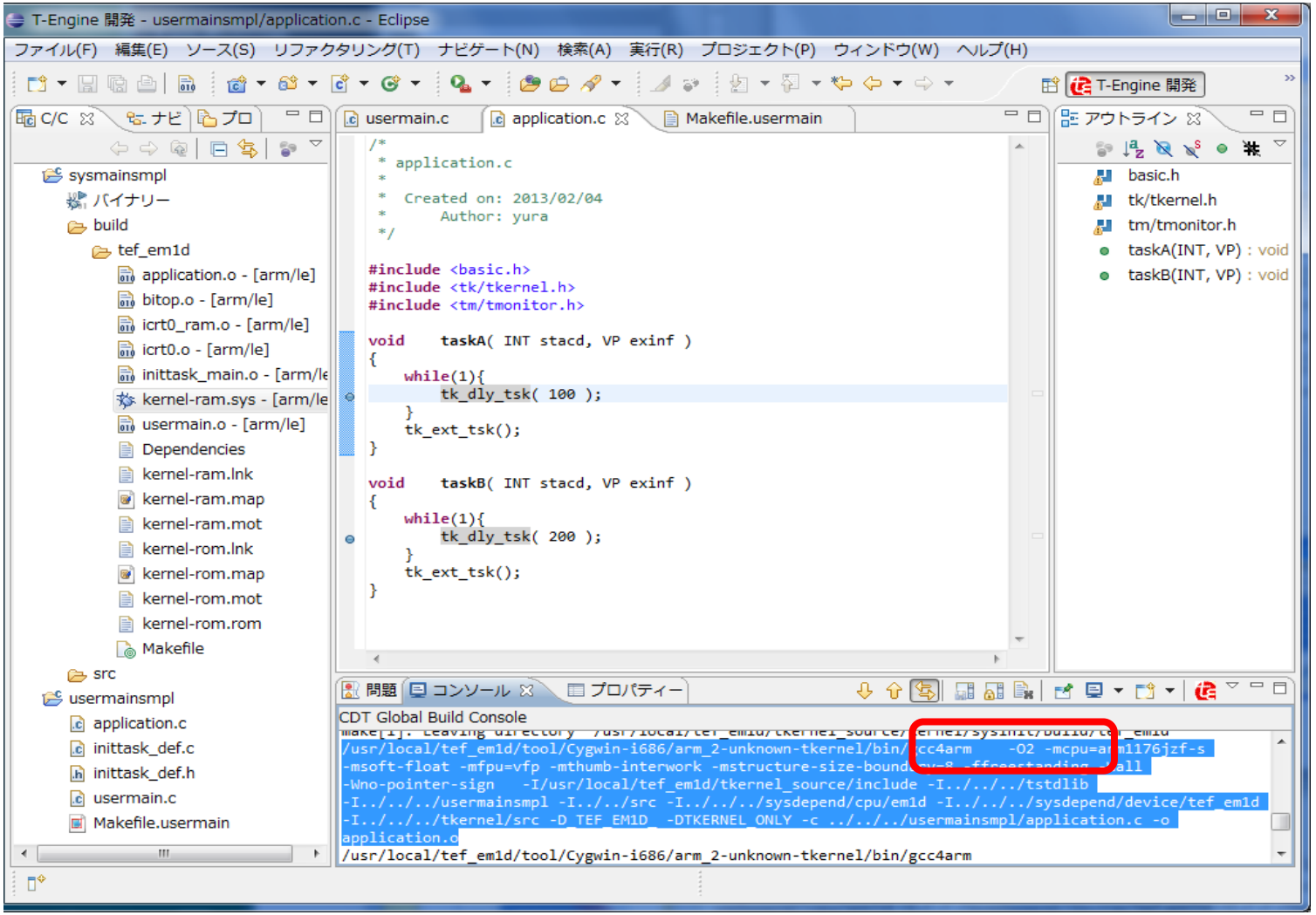
- taskAとtaskBのブレークポイントが未解決？

- application.oにデバッグ情報がないから。

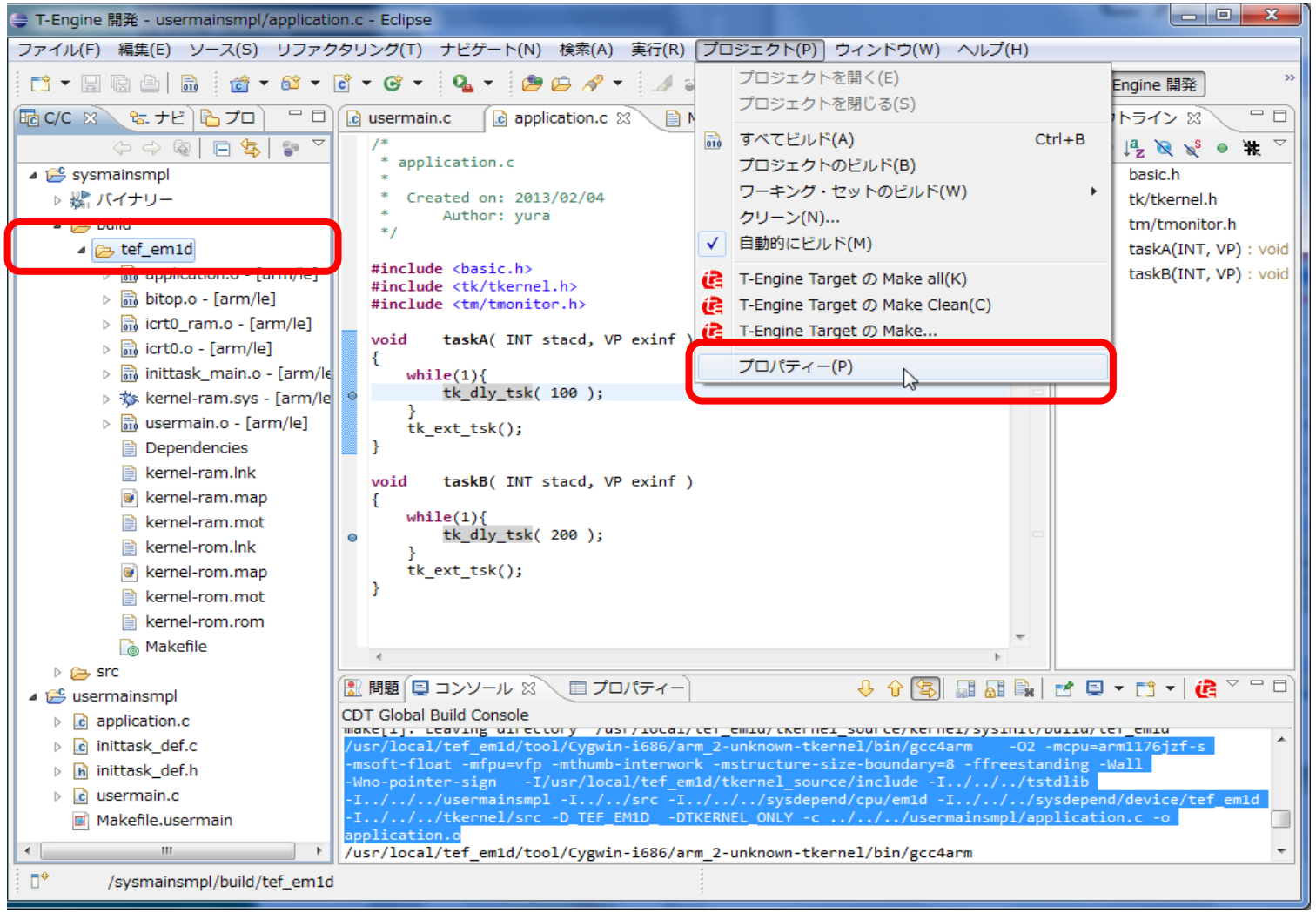
- application.c のコンパイルオプションを確認
  - -02 あり（最適化あり）
  - -g なし（デバッグ情報なし）

- ビルドの設定を変更
  - make mode=debug

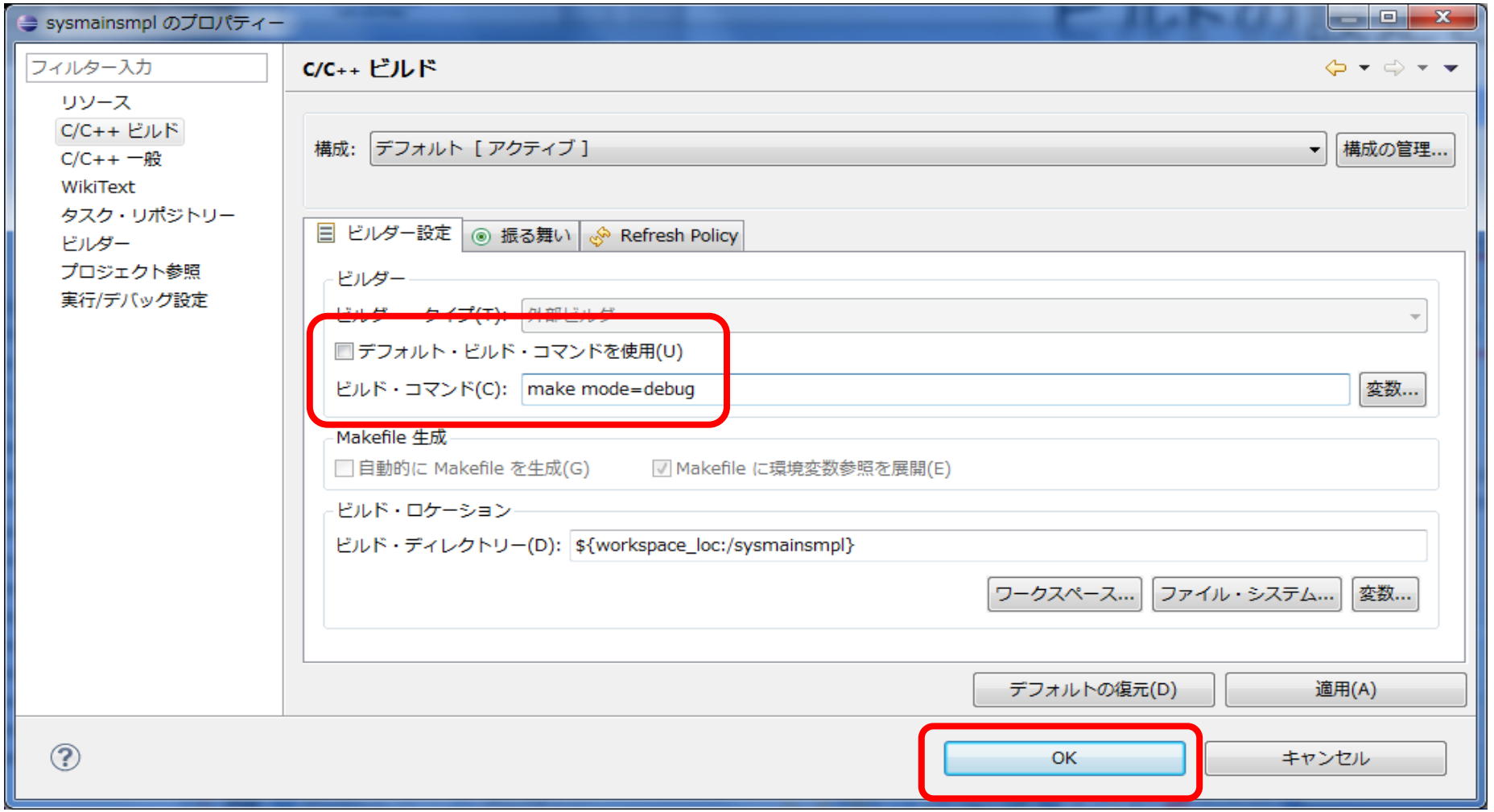
状況を確認してみる


T-Engine 開発 - usermainsmpl/application.c - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
T-Engine 開発
C/C ナビ プロ
sysmainsmpl
バイナリー
build
tef_em1d
application.o - [arm/le]
bitop.o - [arm/le]
icrt0_ram.o - [arm/le]
icrt0.o - [arm/le]
inittask_main.o - [arm/le]
kernel-ram.sys - [arm/le]
usermain.o - [arm/le]
Dependencies
kernel-ram.lnk
kernel-ram.map
kernel-ram.mot
kernel-rom.lnk
kernel-rom.map
kernel-rom.mot
kernel-rom.rom
Makefile
src
usermainsmpl
application.c
inittask_def.c
inittask_def.h
usermain.c
Makefile.usermain
usermain.c
application.c
Makefile.usermain
/*
* application.c
*
* Created on: 2013/02/04
* Author: yura
*/
#include <basic.h>
#include <tk/tkernel.h>
#include <tm/tmonitor.h>
void taskA(INT stacd, VP exinf)
{
while(1){
tk_dly_tsk(100);
}
tk_ext_tsk();
}
void taskB(INT stacd, VP exinf)
{
while(1){
tk_dly_tsk(200);
}
tk_ext_tsk();
}
アウトライン
basic.h
tk/tkernel.h
tm/tmonitor.h
taskA(INT, VP) : void
taskB(INT, VP) : void
問題 コンソール プロパティー
CDT Global Build Console
/usr/local/tef_em1d/tool/Cygwin-i686/arm_2-unknown-tkernel/bin/gcc4arm -O2 -mcpu=arm1176jzf-s
-msoft-float -mfpu=vfp -mthumb-interwork -mstructure-size-boundary=8 -ffreestanding -Wall
-Wno-pointer-sign -I/usr/local/tef_em1d/tkernel_source/include -I../../../tstdlib
-I../../../usermainsmpl -I../../src -I../../../sysdepend/cpu/em1d -I../../../sysdepend/device/tef_em1d
-I../../../tkernel/src -D TEF_EM1D_ -DTKERNEL_ONLY -c ../../../usermainsmpl/application.c -o
application.o
/usr/local/tef_em1d/tool/Cygwin-i686/arm_2-unknown-tkernel/bin/gcc4arm

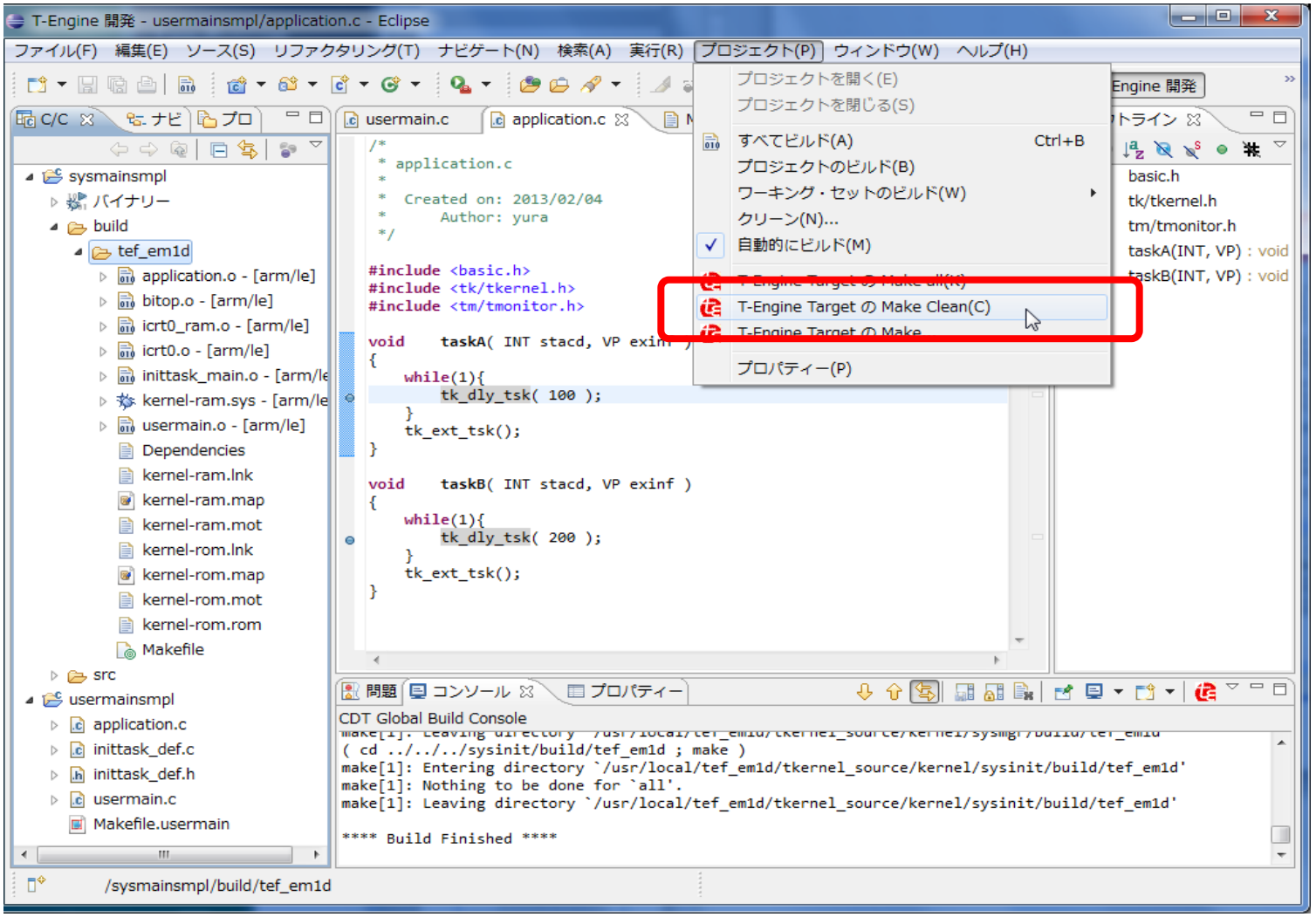
ビルドの設定を変更 (1/2)



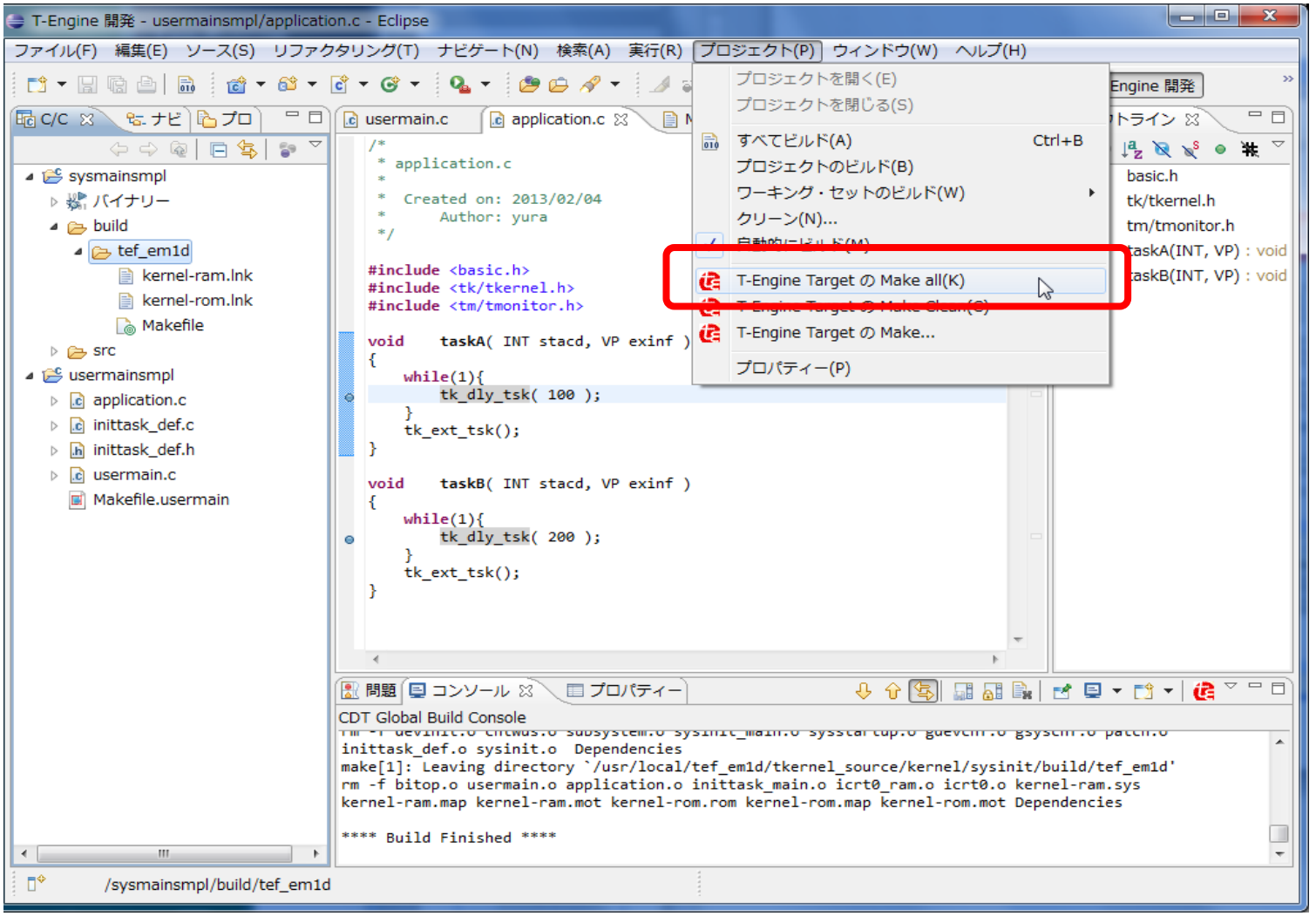
ビルドの設定を変更 (2/2)



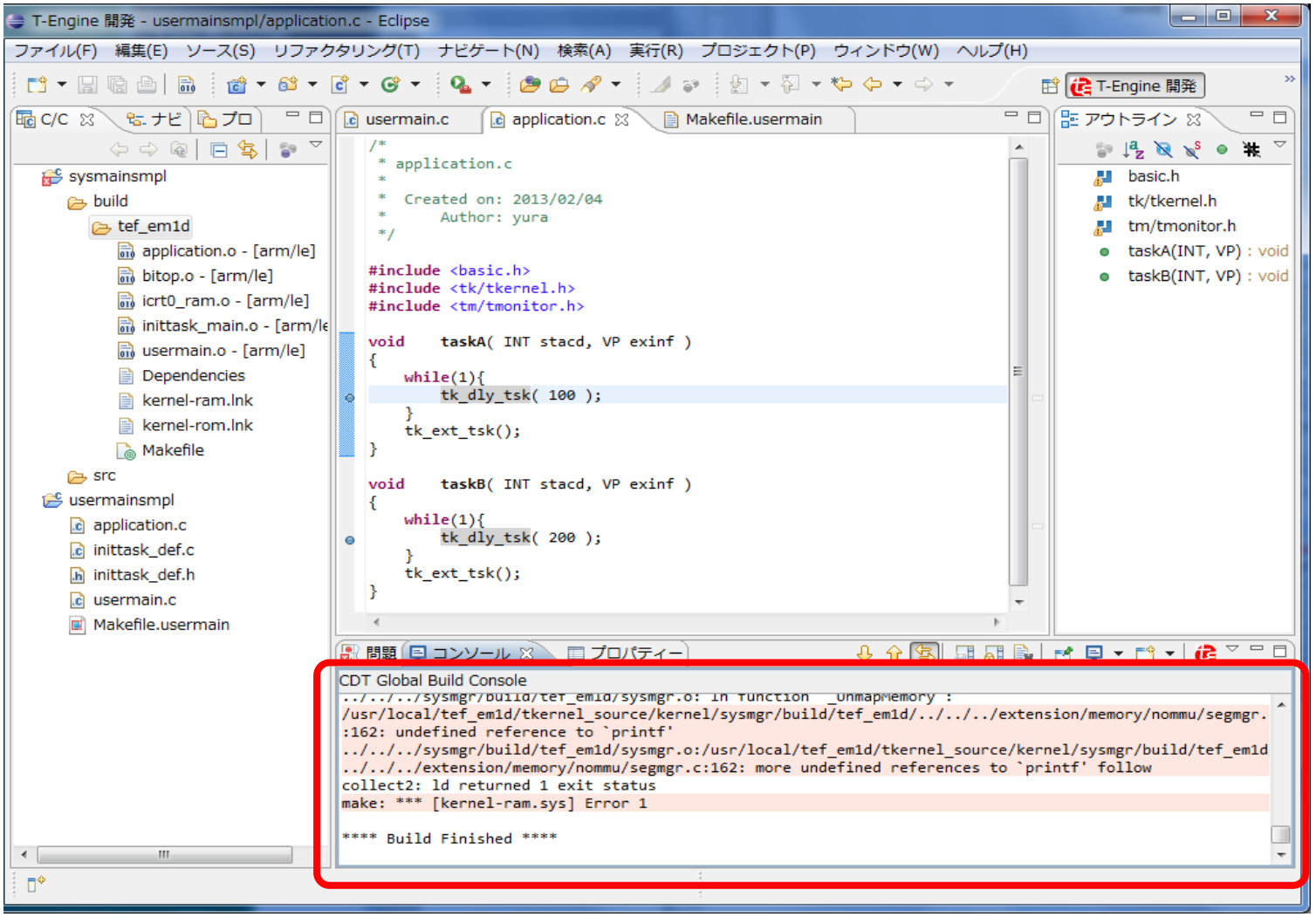
一旦 make clean してから



再度ビルドを実行


T-Engine 開発 - usermainsmpl/application.c - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
プロジェクトを開く(E)
プロジェクトを閉じる(S)
すべてビルド(A) Ctrl+B
プロジェクトのビルド(B)
ワーキング・セットのビルド(W)
クリーン(N)...
T-Engine Target の Make all(K)
T-Engine Target の Make...
プロパティー(P)
usermain.c
application.c
sysmainsmpl
バイナリー
build
tef_em1d
kernel-ram.lnk
kernel-rom.lnk
Makefile
src
usermainsmpl
application.c
inittask_def.c
inittask_def.h
usermain.c
Makefile.usermain
/*
* application.c
*
* Created on: 2013/02/04
* Author: yura
*/
#include <basic.h>
#include <tk/tkernel.h>
#include <tm/tmonitor.h>
void taskA(INT stacd, VP exinf)
{
while(1){
tk_dly_tsk(100);
}
tk_ext_tsk();
}
void taskB(INT stacd, VP exinf)
{
while(1){
tk_dly_tsk(200);
}
tk_ext_tsk();
}
basic.h
tk/tkernel.h
tm/tmonitor.h
問題
コンソール
プロパティー
CDT Global Build Console
inittask_def.o sysinit.o Dependencies
make[1]: Leaving directory `/usr/local/tef_em1d/tkernel_source/kernel/sysinit/build/tef_em1d'
rm -f bitop.o usermain.o application.o inittask_main.o icrt0_ram.o icrt0.o kernel-ram.sys
kernel-ram.map kernel-ram.mot kernel-rom.rom kernel-rom.map kernel-rom.mot Dependencies
**** Build Finished ****
/sysmainsmpl/build/tef_em1d

コンパイルエラー発生？


T-Engine 開発 - usermainsmpl/application.c - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
T-Engine 開発
C/C ナビ プロ
sysmainsmpl
build
tef_em1d
application.o - [arm/le]
bitop.o - [arm/le]
icrt0_ram.o - [arm/le]
inittask_main.o - [arm/le]
usermain.o - [arm/le]
Dependencies
kernel-ram.lnk
kernel-rom.lnk
Makefile
src
usermainsmpl
application.c
inittask_def.c
inittask_def.h
usermain.c
Makefile.usermain
usermain.c
application.c
Makefile.usermain
/*
* application.c
*
* Created on: 2013/02/04
* Author: yura
*/
#include <basic.h>
#include <tk/tkernel.h>
#include <tm/tmonitor.h>
void taskA(INT stacd, VP exinf)
{
while(1){
tk_dly_tsk(100);
}
tk_ext_tsk();
}
void taskB(INT stacd, VP exinf)
{
while(1){
tk_dly_tsk(200);
}
tk_ext_tsk();
}
アウトライン
basic.h
tk/tkernel.h
tm/tmonitor.h
taskA(INT, VP) : void
taskB(INT, VP) : void
問題 コンソール プロパティー
CDT Global Build Console
../../../sysmgr/build/tef_em1d/sysmgr.o: In function `_UnmapMemory':
/usr/local/tef_em1d/tkernel_source/kernel/sysmgr/build/tef_em1d/../../../extension/memory/nommu/segmgr.
:162: undefined reference to `printf'
../../../sysmgr/build/tef_em1d/sysmgr.o:/usr/local/tef_em1d/tkernel_source/kernel/sysmgr/build/tef_em1d
../../../extension/memory/nommu/segmgr.c:162: more undefined references to `printf' follow
collect2: ld returned 1 exit status
make: *** [kernel-ram.sys] Error 1
**** Build Finished ****

何がコンパイルエラーか？

- コンパイルエラー
  - undefined reference to `printf'

- T-Kernelが使用しているログ表示
  → 使っていないのでこの機能は削除

printf 関連を削除

- tkernel_source/include/sys/debug.h を変更

```
        :
/*
 * Example
 *      DEBUG_PRINT(("error = %d¥n", ercd));
 *
 *      DO_DEBUG( if ( ercd < ER_OK ) DEBUG_PRINT(("error = %d¥n", ercd)); )
 */
//#ifdef DEBUG
#if    0

#ifndef DEBUG_MODULE
#define DEBUG_MODULE  ""      /* Normally define like "(module name)" */
#endif

#define DEBUG_PRINT(arg)                                                ¥
(                                                                       ¥
        printf("%s#%d%s:", __FILE__, __LINE__, DEBUG_MODULE),           ¥
        printf arg                                                      ¥
)
        :
```

別ファイルの追加完了

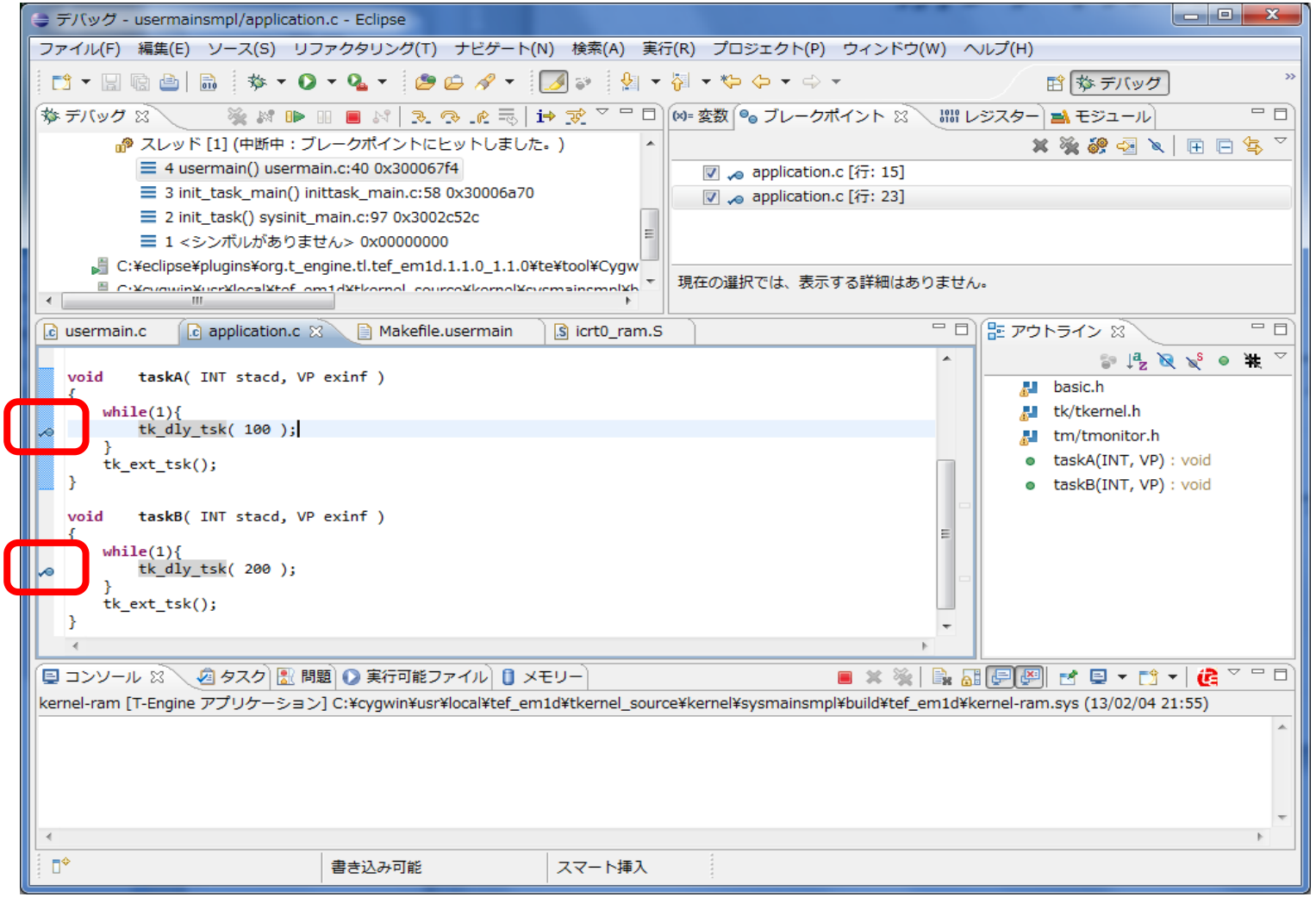
- 一旦cleanしてから、再度ビルドを実行

- application.c の追加完了

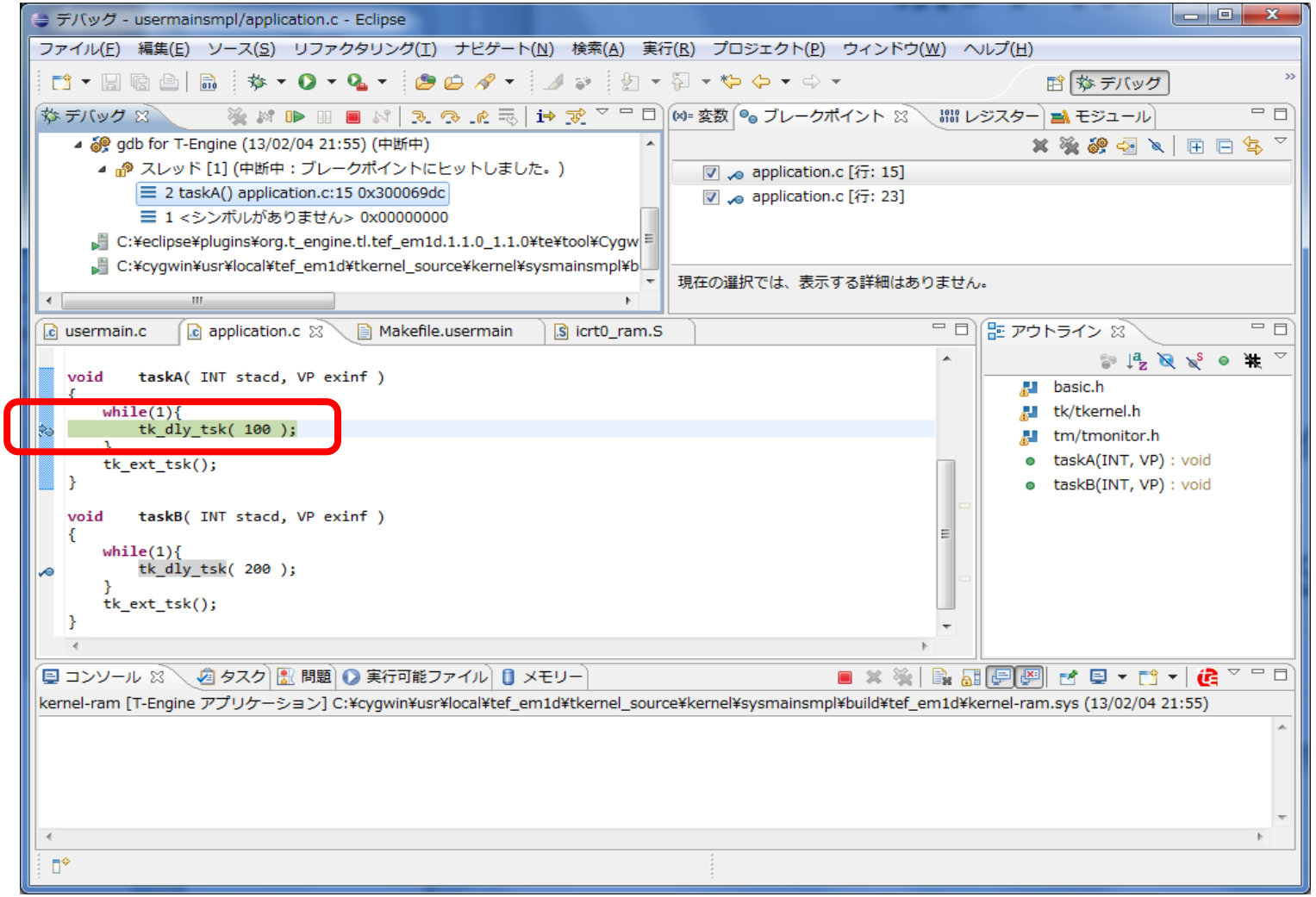
- taskA、taskBにブレークポイントが設定できる

- taskA、taskBでブレークすることを確認

ブレークポイントが解決された！



ブレークポイントにヒットすることを確認



演習問題

- 初期タスクの優先度を変更して動作を確認
- tk_chg_pri(TSK_SELF, 1);
  - taskA, taskBよりも高い優先度
- tk_chg_pri(TSK_SELF, 10);
  - taskA, taskBと同じ優先度

応用問題

- 強制的なブレークができない。
  - 本デバッグ環境はGDBを使ってソフトウェア的に実現
  - ヒットするブレークポイントを設定せずに実行すると、中断ボタン()をクリックしても停止しない。

- エミュレータを再起動するしかない。
  - Eclipseでロードしたアプリケーションも終了する。

- 適当なブレーク方法はないか？

回答例

```
EXPORT INT    usermain( void )
{
        T_CTSK ctsk = { NULL, TA_HLNG | TA_RNG0, NULL, 10, 4*1024, };
        ID     tidA;
        ID     tidB;
        ER     ercd;

               :

        while(1){
               tk_slp_tsk( 10 * 1000 );
        }
        return 0;
}
```

万能ではない

- ブレークポイントを設定し忘れたら意味がない。
- 優先度の高いタスクが実行し続けていたら、
  初期タスクのブレークポイントはヒットしない。
  - 初期タスクの優先度 = 138

- 任意のタイミングでブレークさせるには？
  - Interface 2012年5月号

→ ２日目へ

2日目

デバイスドライバを使う

デバイスドライバを使う

- 液晶画面(LCD)を表示
- 簡易ブレーク機能の追加

特定のPDイベントでブレークさせる。

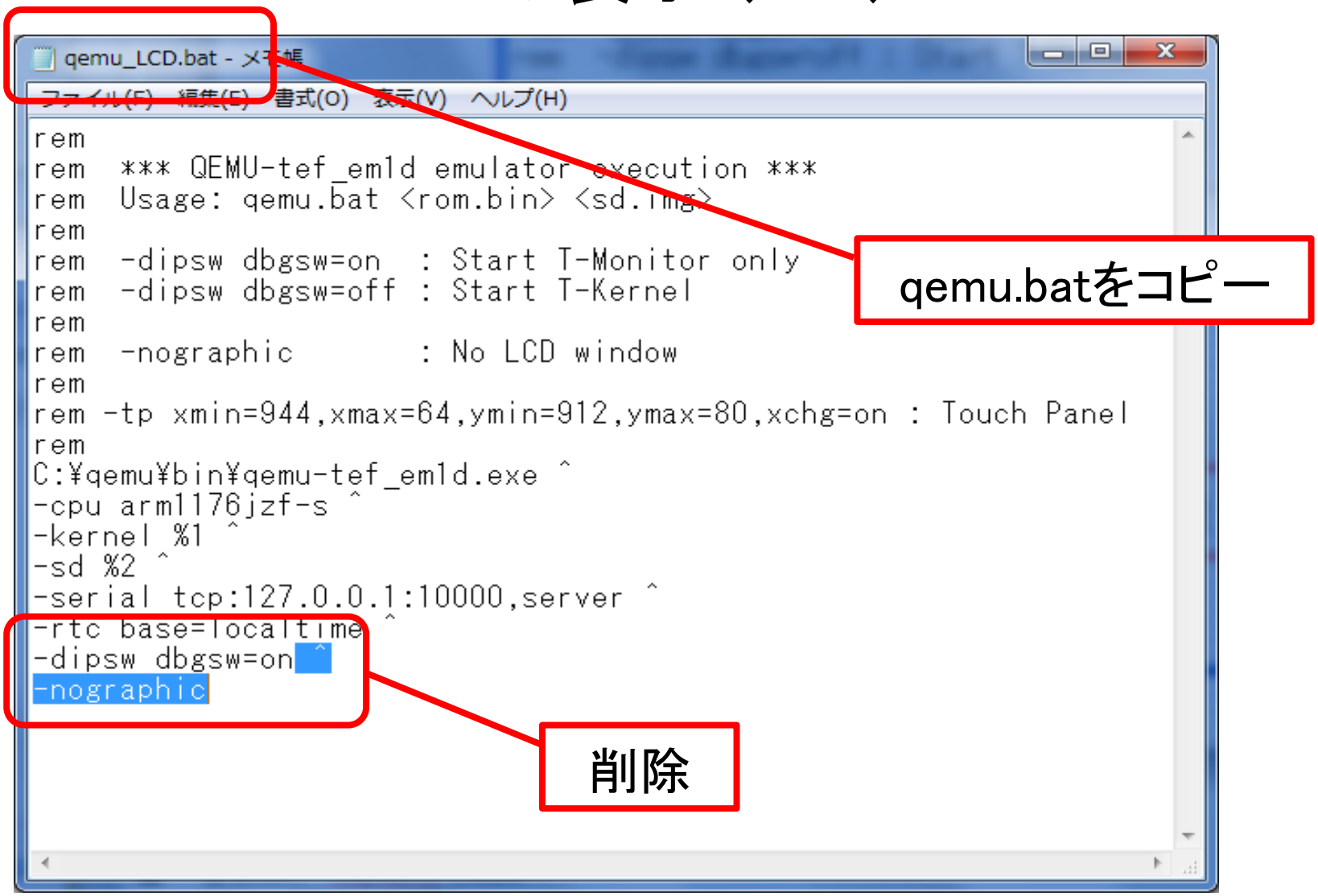
液晶画面(LCD)を表示

～デバイスドライバを使ってみる～

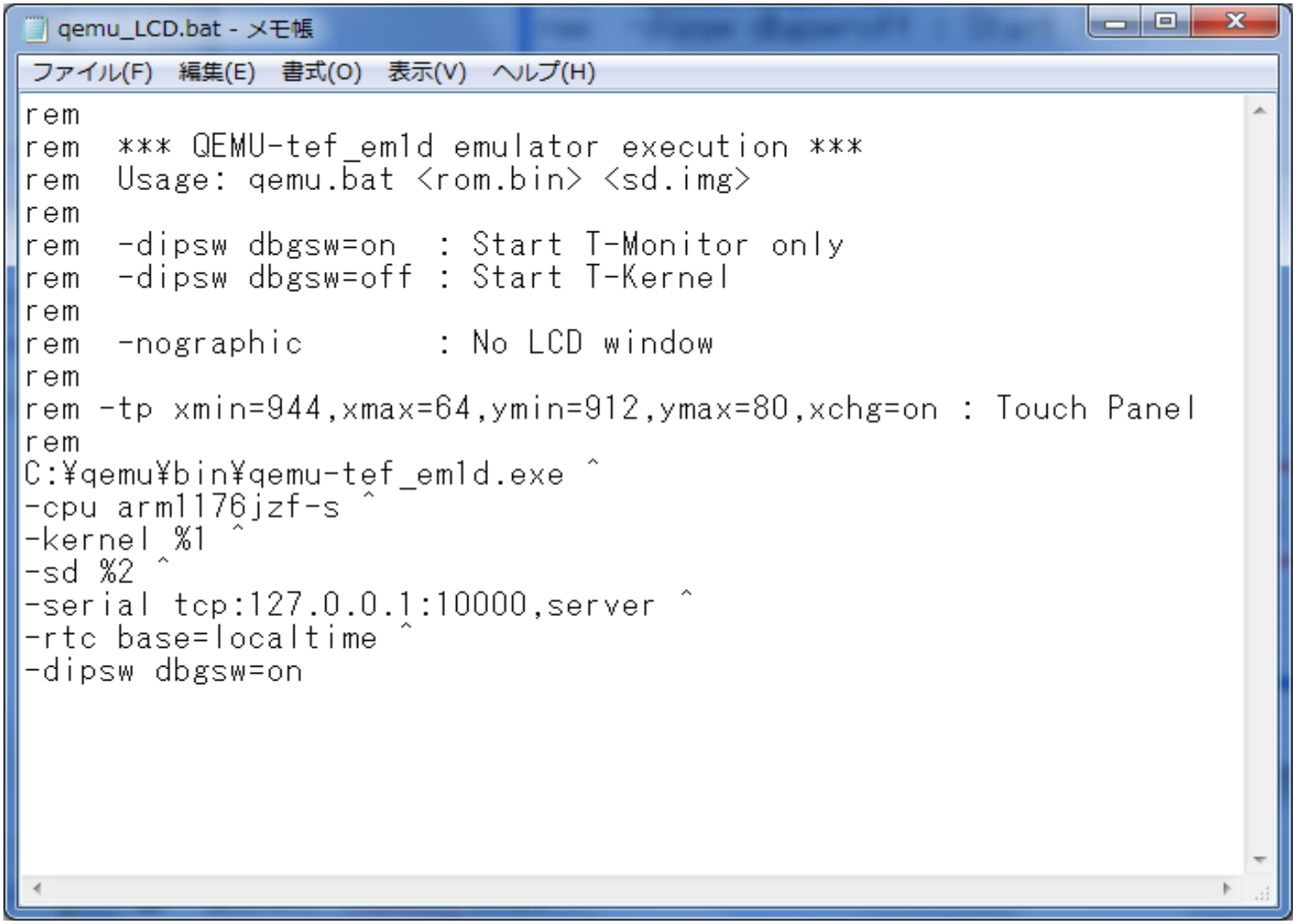
液晶画面(LCD)の表示

- Qemuの起動オプションを変更
  - デフォルト設定は、LCDなし
    → LCDを表示するように qemu.bat を変更
- 実行
- Eclipseと接続
- デバイスドライバを追加
- T-Kernel本体のビルド
- 液晶画面への描画

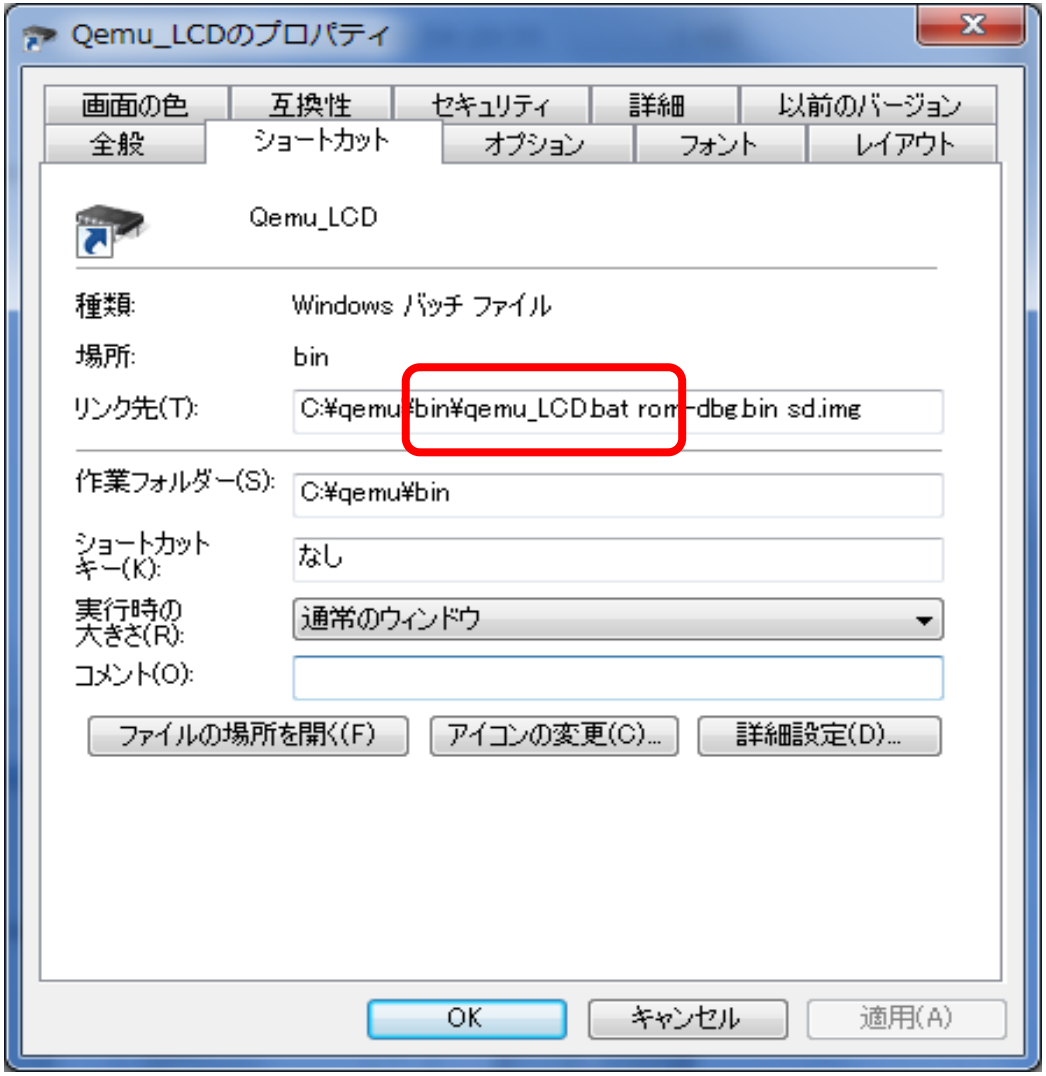
LCDの表示 (1/8)


qemu_LCD.bat - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
rem
rem *** QEMU-tef_em1d emulator execution ***
rem Usage: qemu.bat <rom.bin> <sd.img>
rem
rem -dipsw dbgsw=on : Start T-Monitor only
rem -dipsw dbgsw=off : Start T-Kernel
rem
rem -nographic   : No LCD window
rem
rem -tp xmin=944,xmax=64,ymin=912,ymax=80,xchg=on : Touch Panel
rem
C:¥qemu¥bin¥qemu-tef_em1d.exe ^
-cpu arm1176jzf-s ^
-kernel %1 ^
-sd %2 ^
-serial tcp:127.0.0.1:10000,server ^
-rtc base=localtime ^
-dipsw dbgsw=on ^
-nographic
qemu.batをコピー
削除

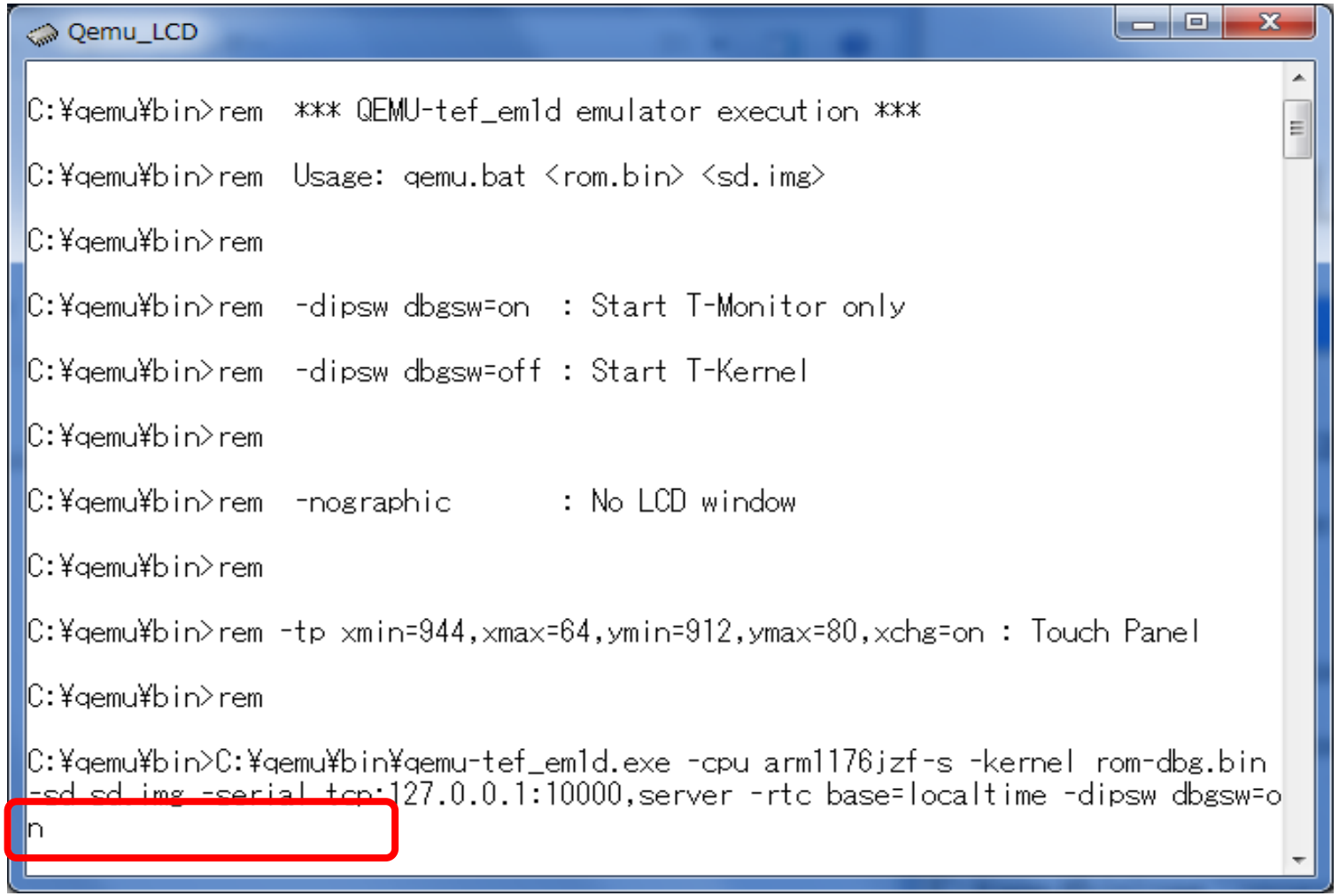
LCDの表示 (2/8)


qemu_LCD.bat - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
rem
rem *** QEMU-tef_em1d emulator execution ***
rem Usage: qemu.bat <rom.bin> <sd.img>
rem
rem -dipsw dbgsw=on : Start T-Monitor only
rem -dipsw dbgsw=off : Start T-Kernel
rem
rem -nographic   : No LCD window
rem
rem -tp xmin=944,xmax=64,ymin=912,ymax=80,xchg=on : Touch Panel
rem
C:¥qemu¥bin¥qemu-tef_em1d.exe ^
-cpu arm1176jzf-s ^
-kernel %1 ^
-sd %2 ^
-serial tcp:127.0.0.1:10000,server ^
-rtc base=localtime ^
-dipsw dbgsw=on

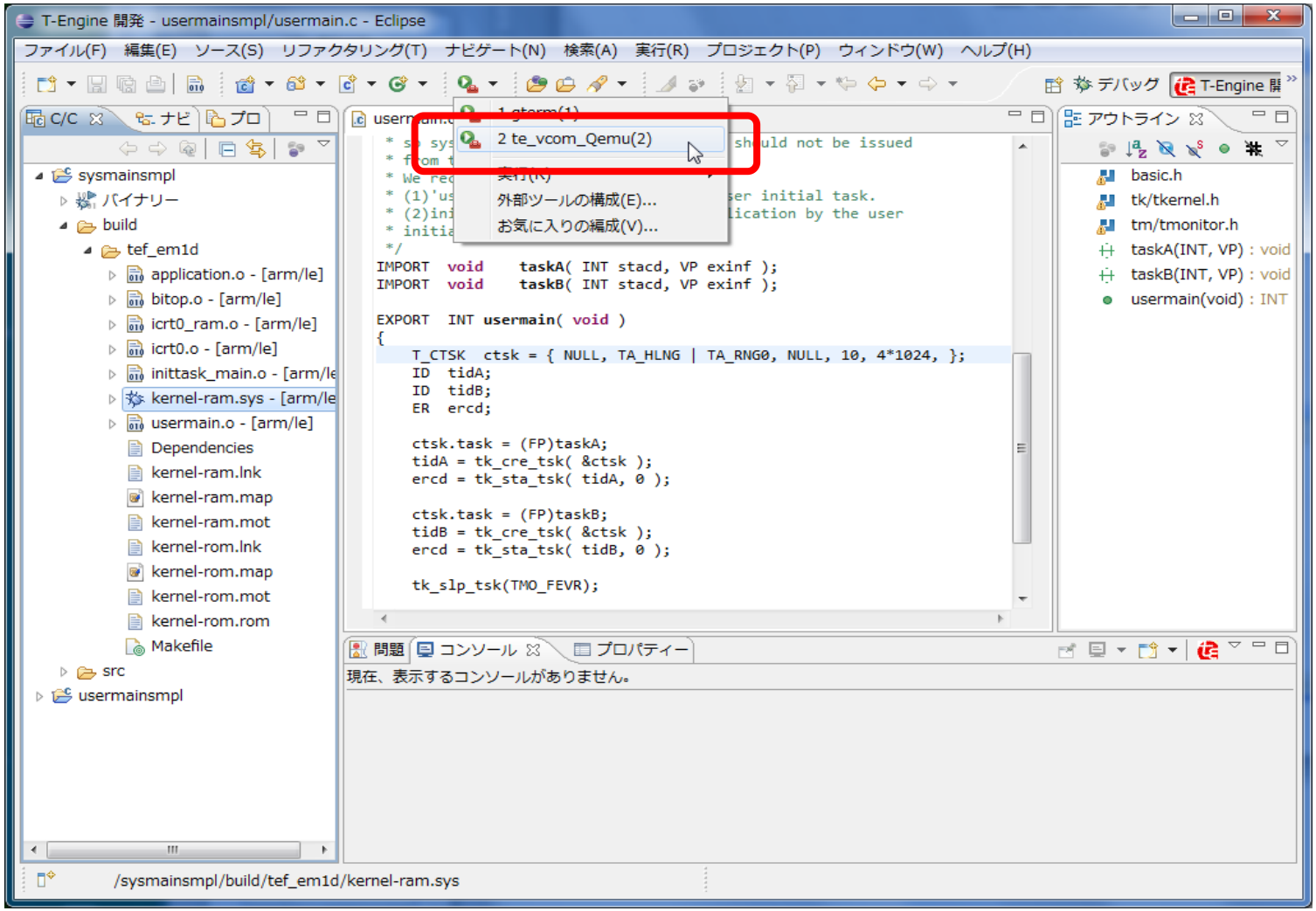
LCDの表示 (3/8)


Qemu_LCDのプロパティ
画面の色
互換性
セキュリティ
詳細
以前のバージョン
全般
ショートカット
オプション
フォント
レイアウト
Qemu_LCD
種類:
Windows バッチ ファイル
場所:
bin
リンク先(T):
C:¥qemu¥bin¥qemu_LCD.bat rom-dbg.bin sd.img
作業フォルダー(S):
C:¥qemu¥bin
ショートカット キー(K):
なし
実行時の大きさ(R):
通常のウィンドウ
コメント(O):
ファイルの場所を開く(F)
アイコンの変更(C)...
詳細設定(D)...
OK
キャンセル
適用(A)

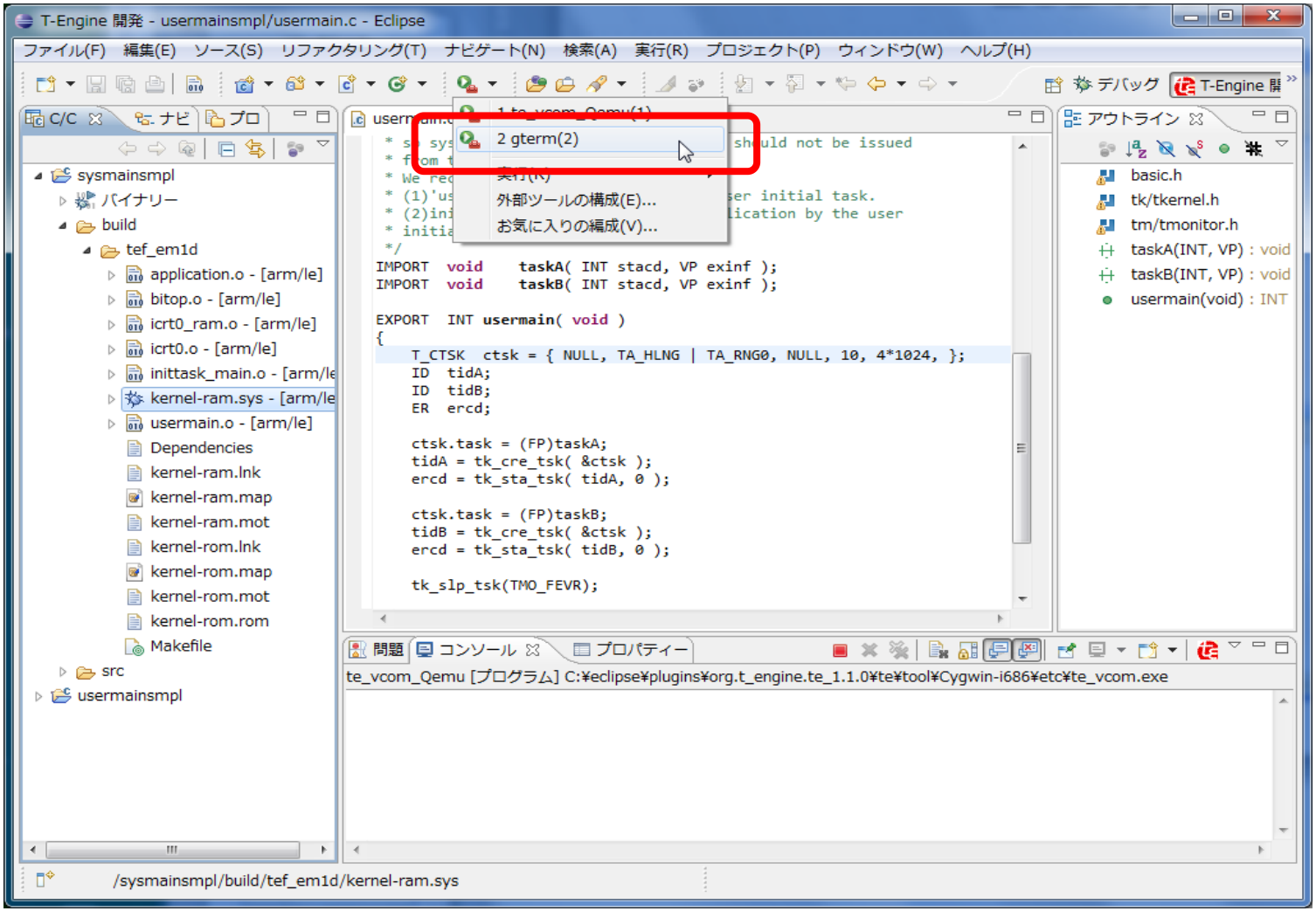
LCDの表示 (4/8)


Qemu_LCD
C:¥qemu¥bin>rem *** QEMU-tef_em1d emulator execution ***
C:¥qemu¥bin>rem Usage: qemu.bat <rom.bin> <sd.img>
C:¥qemu¥bin>rem
C:¥qemu¥bin>rem -dipsw dbgsw=on : Start T-Monitor only
C:¥qemu¥bin>rem -dipsw dbgsw=off : Start T-Kernel
C:¥qemu¥bin>rem
C:¥qemu¥bin>rem -nographic   : No LCD window
C:¥qemu¥bin>rem
C:¥qemu¥bin>rem -tp xmin=944,xmax=64,ymin=912,ymax=80,xchg=on : Touch Panel
C:¥qemu¥bin>rem
C:¥qemu¥bin>C:¥qemu¥bin¥qemu-tef_em1d.exe -cpu arm1176jzf-s -kernel rom-dbg.bin
-sd sd.img -serial tcp:127.0.0.1:10000,server -rtc base=localtime -dipsw dbgsw=o
n

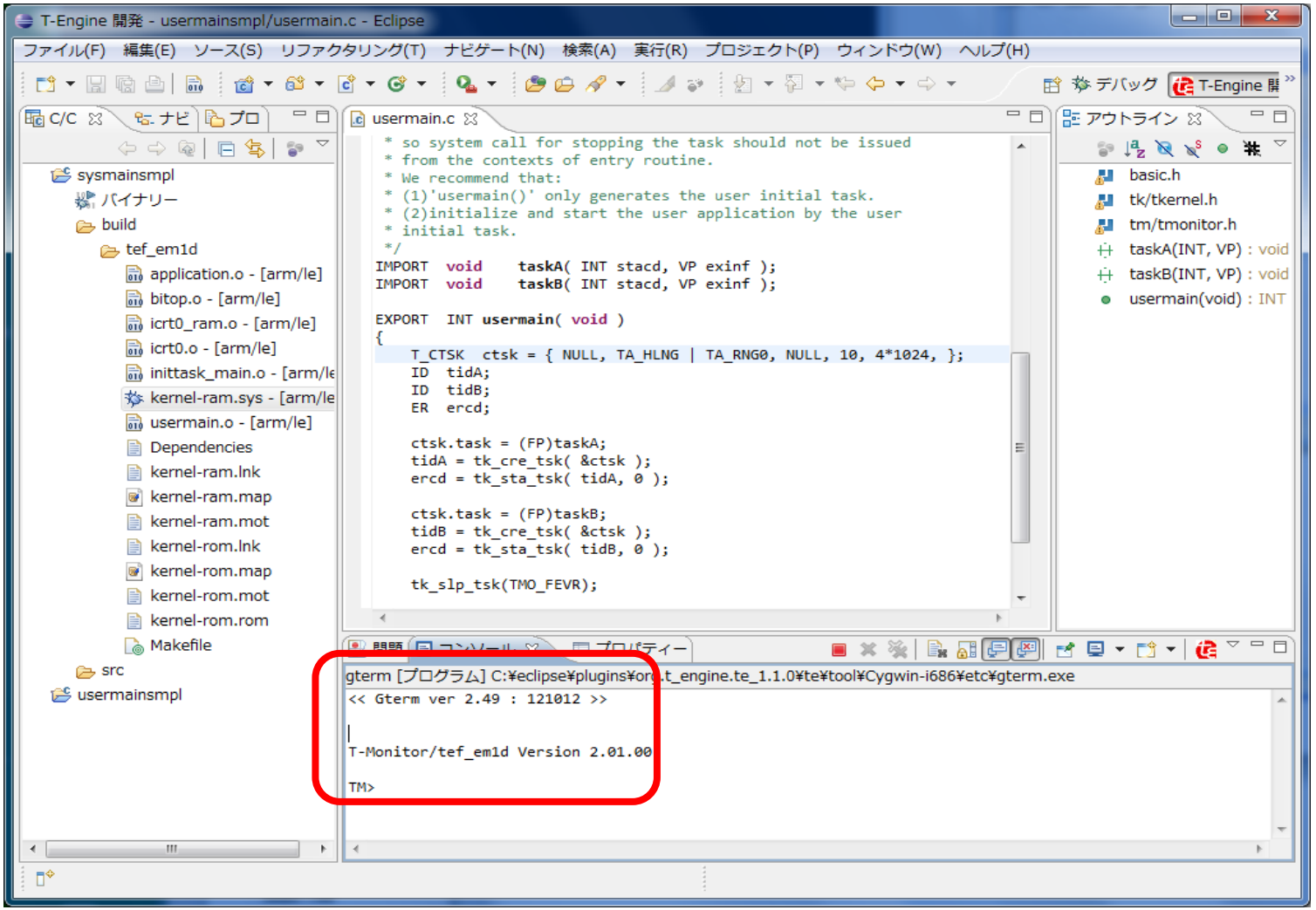
LCDの表示 (5/8)


T-Engine 開発 - usermainsmpl/usermain.c - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
デバッグ
T-Engine
2 te_vcom_Qemu(2)
外部ツールの構成(E)...
お気に入りの編成(V)...
C/C
ナビ
プロ
sysmainsmpl
バイナリー
build
tef_em1d
application.o - [arm/le]
bitop.o - [arm/le]
icrt0_ram.o - [arm/le]
icrt0.o - [arm/le]
inittask_main.o - [arm/le]
kernel-ram.sys - [arm/le]
usermain.o - [arm/le]
Dependencies
kernel-ram.lnk
kernel-ram.map
kernel-ram.mot
kernel-rom.lnk
kernel-rom.map
kernel-rom.mot
kernel-rom.rom
Makefile
src
usermainsmpl
IMPORT void taskA(INT stacd, VP exinf);
IMPORT void taskB(INT stacd, VP exinf);
EXPORT INT usermain(void)
{
T_CTSK ctsk = { NULL, TA_HLNG | TA_RNG0, NULL, 10, 4*1024, };
ID tidA;
ID tidB;
ER ercd;
ctsk.task = (FP)taskA;
tidA = tk_cre_tsk(&ctsk);
ercd = tk_sta_tsk(tidA, 0);
ctsk.task = (FP)taskB;
tidB = tk_cre_tsk(&ctsk);
ercd = tk_sta_tsk(tidB, 0);
tk_slp_tsk(TMO_FEVR);
アウトライン
basic.h
tk/tkernel.h
tm/tmonitor.h
taskA(INT, VP) : void
taskB(INT, VP) : void
usermain(void) : INT
問題
コンソール
プロパティー
現在、表示するコンソールがありません。
/sysmainsmpl/build/tef_em1d/kernel-ram.sys

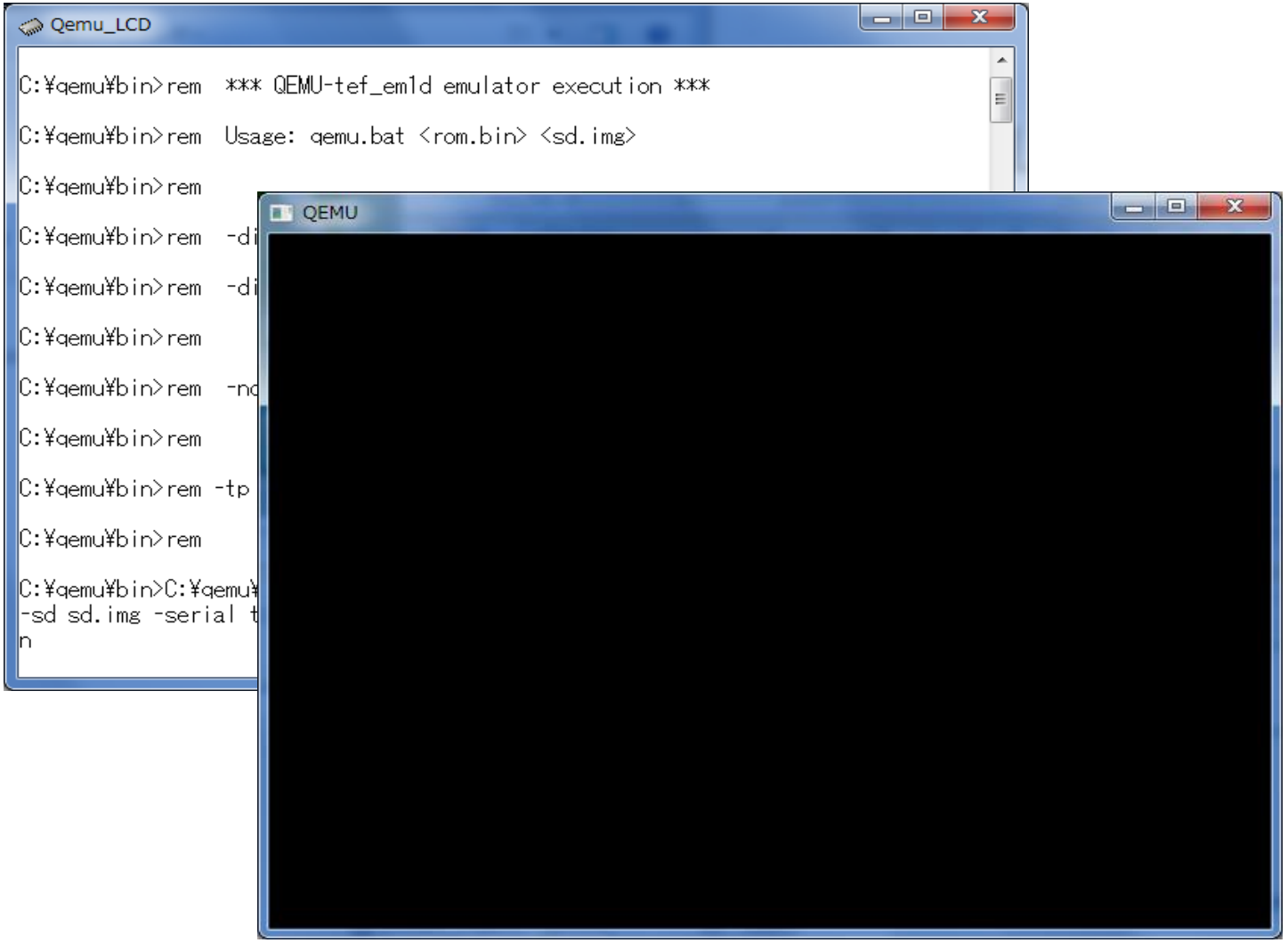
LCDの表示 (6/8)


T-Engine 開発 - usermainsmpl/usermain.c - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
デバッグ
T-Engine
2 gterm(2)
外部ツールの構成(E)...
お気に入りの編成(V)...
C/C
ナビ
プロ
sysmainsmpl
バイナリー
build
tef_em1d
application.o - [arm/le]
bitop.o - [arm/le]
icrt0_ram.o - [arm/le]
icrt0.o - [arm/le]
inittask_main.o - [arm/le]
kernel-ram.sys - [arm/le]
usermain.o - [arm/le]
Dependencies
kernel-ram.lnk
kernel-ram.map
kernel-ram.mot
kernel-rom.lnk
kernel-rom.map
kernel-rom.mot
kernel-rom.rom
Makefile
src
usermainsmpl
IMPORT void taskA(INT stacd, VP exinf);
IMPORT void taskB(INT stacd, VP exinf);
EXPORT INT usermain(void)
{
T_CTSK ctsk = { NULL, TA_HLNG | TA_RNG0, NULL, 10, 4*1024, };
ID tidA;
ID tidB;
ER ercd;
ctsk.task = (FP)taskA;
tidA = tk_cre_tsk(&ctsk);
ercd = tk_sta_tsk(tidA, 0);
ctsk.task = (FP)taskB;
tidB = tk_cre_tsk(&ctsk);
ercd = tk_sta_tsk(tidB, 0);
tk_slp_tsk(TMO_FEVR);
アウトライン
basic.h
tk/tkernel.h
tm/tmonitor.h
taskA(INT, VP) : void
taskB(INT, VP) : void
usermain(void) : INT
問題
コンソール
プロパティー
te_vcom_Qemu [プログラム] C:¥eclipse¥plugins¥org.t_engine.te_1.1.0¥te¥tool¥Cygwin-i686¥etc¥te_vcom.exe
/sysmainsmpl/build/tef_em1d/kernel-ram.sys

LCDの表示 (7/8)


T-Engine 開発 - usermainsmpl/usermain.c - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
デバッグ
T-Engine
C/C
ナビ
プロ
sysmainsmpl
バイナリー
build
tef_em1d
application.o - [arm/le]
bitop.o - [arm/le]
icrt0_ram.o - [arm/le]
icrt0.o - [arm/le]
inittask_main.o - [arm/le]
kernel-ram.sys - [arm/le]
usermain.o - [arm/le]
Dependencies
kernel-ram.lnk
kernel-ram.map
kernel-ram.mot
kernel-rom.lnk
kernel-rom.map
kernel-rom.mot
kernel-rom.rom
Makefile
src
usermainsmpl
usermain.c
* so system call for stopping the task should not be issued
* from the contexts of entry routine.
* We recommend that:
* (1)'usermain()' only generates the user initial task.
* (2)initialize and start the user application by the user
* initial task.
*/
IMPORT void taskA(INT stacd, VP exinf);
IMPORT void taskB(INT stacd, VP exinf);
EXPORT INT usermain(void)
{
T_CTSK ctsk = { NULL, TA_HLNG | TA_RNG0, NULL, 10, 4*1024, };
ID tidA;
ID tidB;
ER ercd;
ctsk.task = (FP)taskA;
tidA = tk_cre_tsk(&ctsk);
ercd = tk_sta_tsk(tidA, 0);
ctsk.task = (FP)taskB;
tidB = tk_cre_tsk(&ctsk);
ercd = tk_sta_tsk(tidB, 0);
tk_slp_tsk(TMO_FEVR);
アウトライン
basic.h
tk/tkernel.h
tm/tmonitor.h
taskA(INT, VP) : void
taskB(INT, VP) : void
usermain(void) : INT
問題
コンソール
プロパティー
gterm [プログラム] C:¥eclipse¥plugins¥org.t_engine.te_1.1.0¥te¥tool¥Cygwin-i686¥etc¥gterm.exe
<< Gterm ver 2.49 : 121012 >>
T-Monitor/tef_em1d Version 2.01.00
TM>

LCDの表示 (8/8)


Qemu_LCD
C:¥qemu¥bin>rem *** QEMU-tef_em1d emulator execution ***
C:¥qemu¥bin>rem Usage: qemu.bat <rom.bin> <sd.img>
C:¥qemu¥bin>rem
C:¥qemu¥bin>rem -di
C:¥qemu¥bin>rem -di
C:¥qemu¥bin>rem
C:¥qemu¥bin>rem -nc
C:¥qemu¥bin>rem
C:¥qemu¥bin>rem -tp
C:¥qemu¥bin>rem
C:¥qemu¥bin>C:¥qemu¥
-sd sd.img -serial t
n
QEMU

液晶画面(LCD)の表示

- Qemuの起動オプションを変更
  - デフォルト設定は、LCDなし
    → LCDを表示するように qemu.bat を変更
- 実行
- Eclipseと接続
- デバイスドライバを追加
- T-Kernel本体のビルド
- 液晶画面への描画

デバイスドライバの構築

- ReadMe.txt
  - 4.6 デバイスドライバの構築
- srcpkg/doc/ja/driver.txt
  - 2. デバイスドライバの構築方法
    - 2.1 ドライバI/Fライブラリのビルド
    - 2.2 コンソールドライバのビルド
    - :
    - **2.5 スクリーンドライバのビルド**
    - **2.6 KB/PDドライバのビルド**
    - :
  - 2.8 デバイスドライバを含むT-Kernel本体のビルド

デバイスドライバのビルド手順 (1/2)

```
cd ${BD}/lib/libdrvif/build/tef_em1d
make                    ← ドライバI/Fライブラリをビルド
make install
cd ${BD}/drv/tef_em1d/console/build
make source
make                    ← コンソールドライバをビルド
make install
cd ${BD}/drv/tef_em1d/clk/build
make source
make                    ← 時計ドライバをビルド
make install
```

デバイスドライバのビルド手順 (2/2)

```
cd ${BD}/drv/tef_em1d/sysdsk/build
make                    ← システムディスクドライバをビルド
make install
cd ${BD}/drv/tef_em1d/screen/build
make                    ← スクリーンドライバをビルド
make install
cd ${BD}/drv/tef_em1d/kbpd/build
make                    ← KB/PDドライバをビルド
make install
cd ${BD}/drv/tef_em1d/lowkbpd/build
make                    ← KB/PD実IOドライバをビルド
make install
```

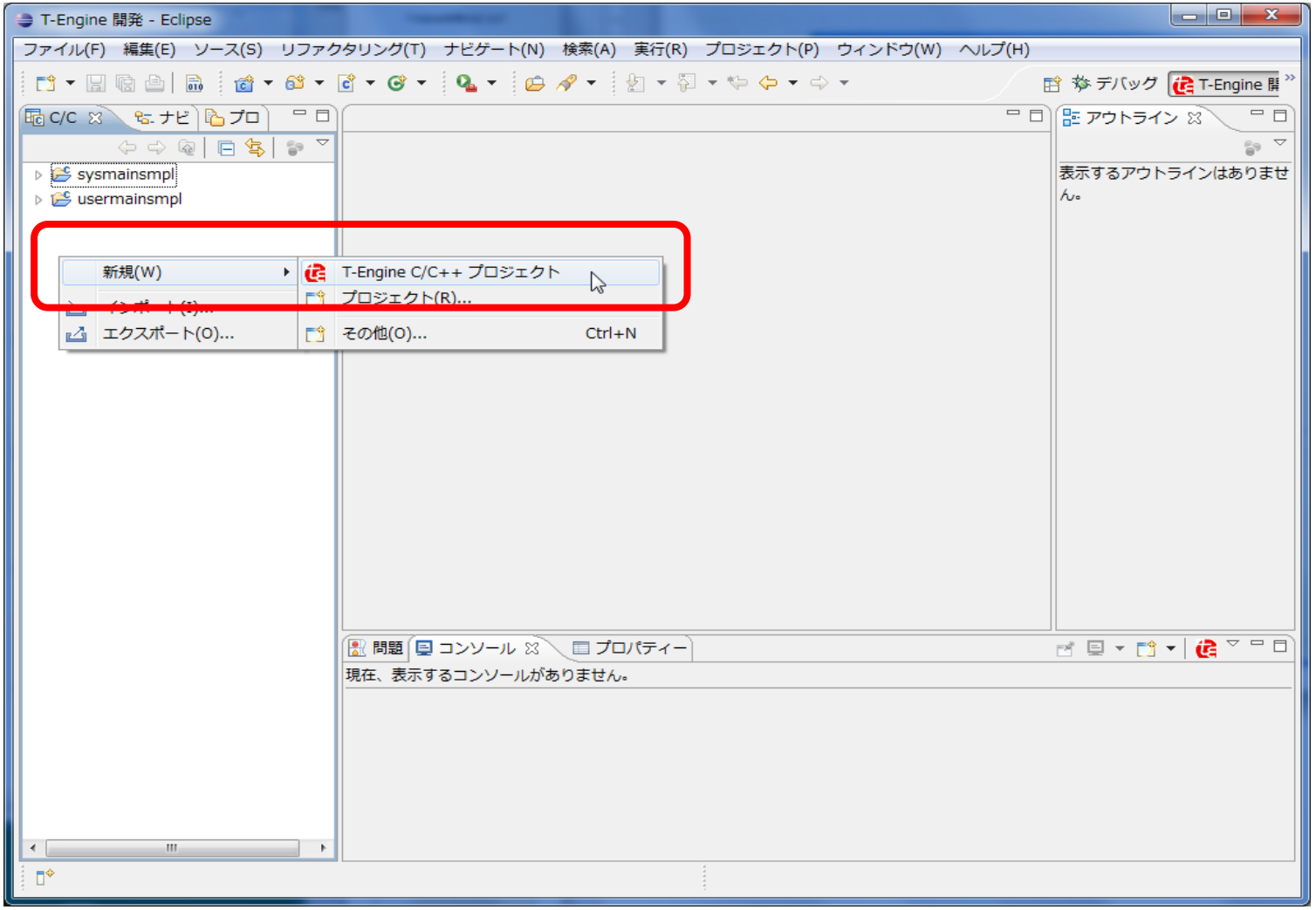
液晶画面(LCD)の表示

- Qemuの起動オプションを変更
  - デフォルトの設定はLCDなし
    → LCDを表示するように qemu.bat を変更
- 実行
- Eclipseと接続
- デバイスドライバを追加
- T-Kernel本体のビルド (Eclipseで実行)
- 液晶画面への描画

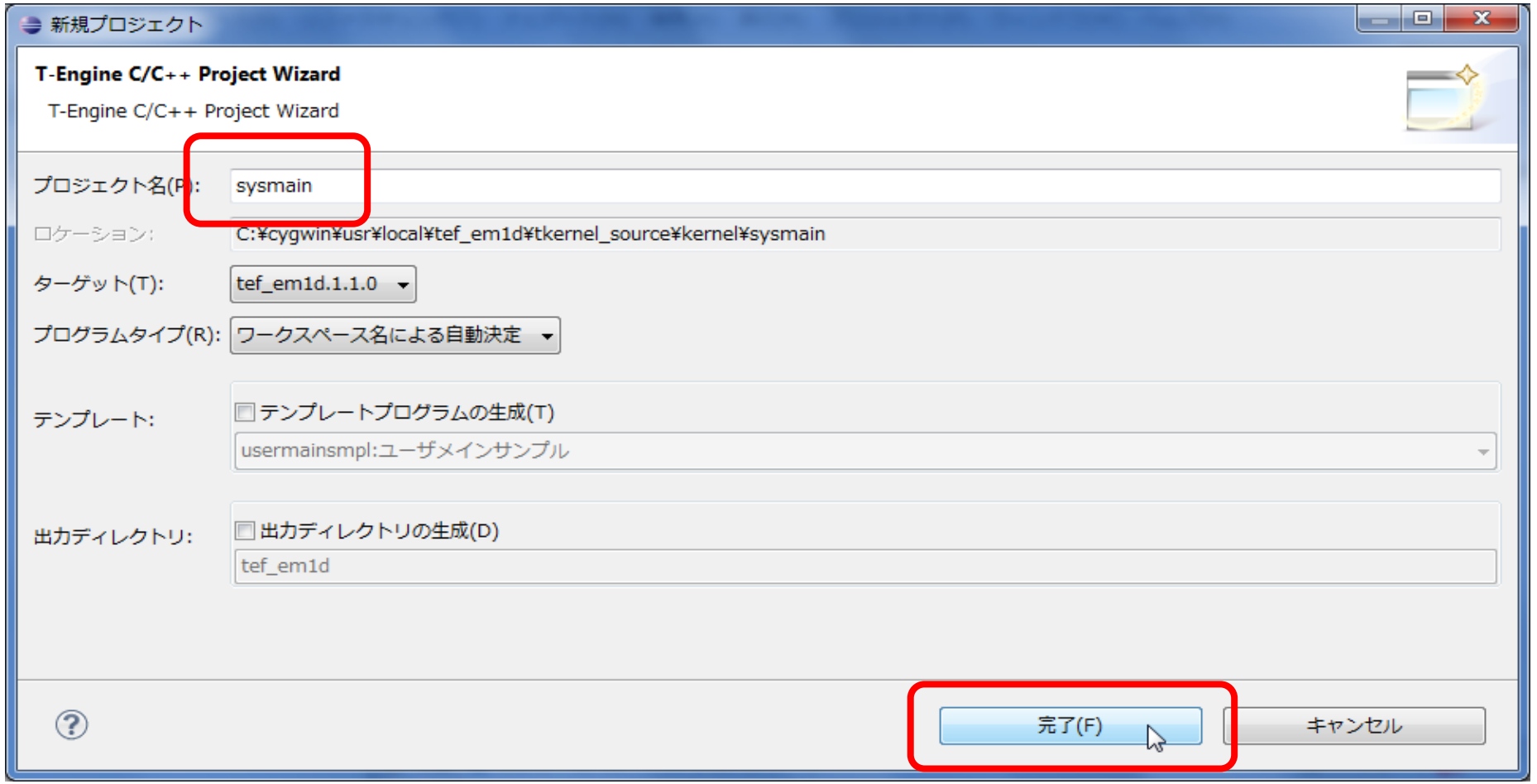
T-Kernel本体のビルド

- プロジェクトの追加
  - ワークスペース
    - C:¥cygwin¥usr¥local¥tef_em1d¥tkernel_source¥kernel
  - プロジェクト
    - sysmain , usermain_drv
- Makefileの編集
- build_drv でビルド
- Qemuと接続
- デバッグの構成

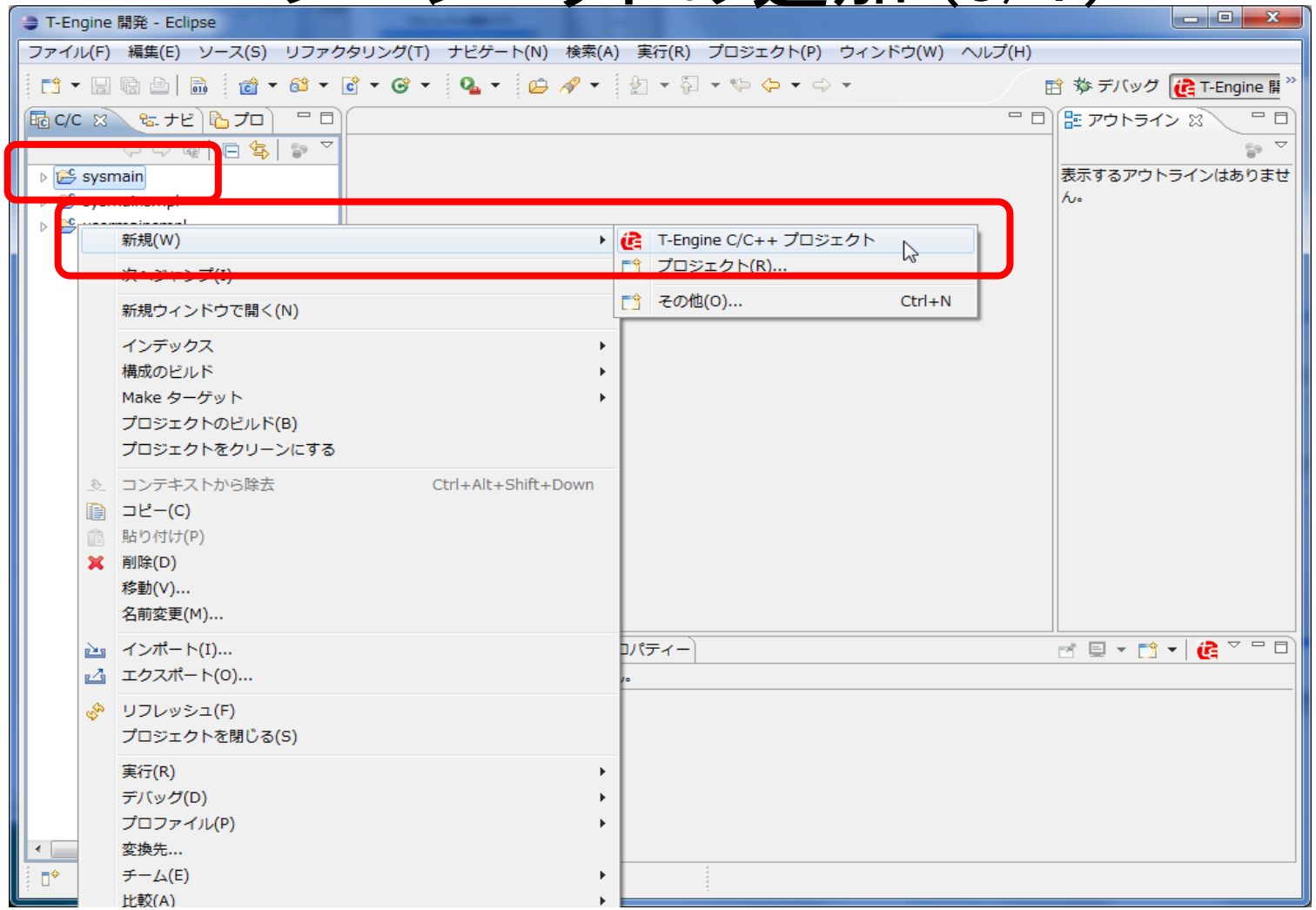
プロジェクトの追加 (1/7)



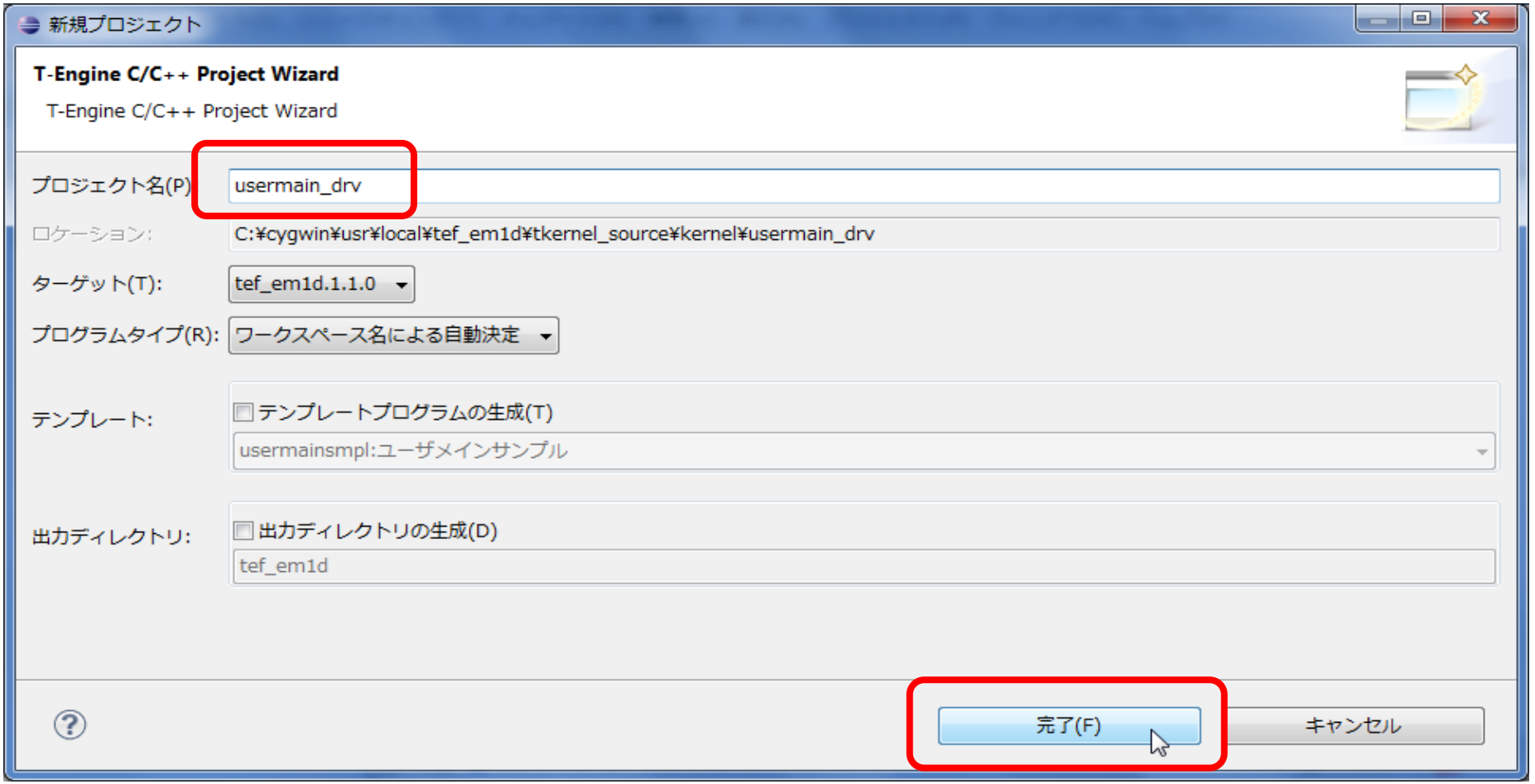
プロジェクトの追加 (2/7)



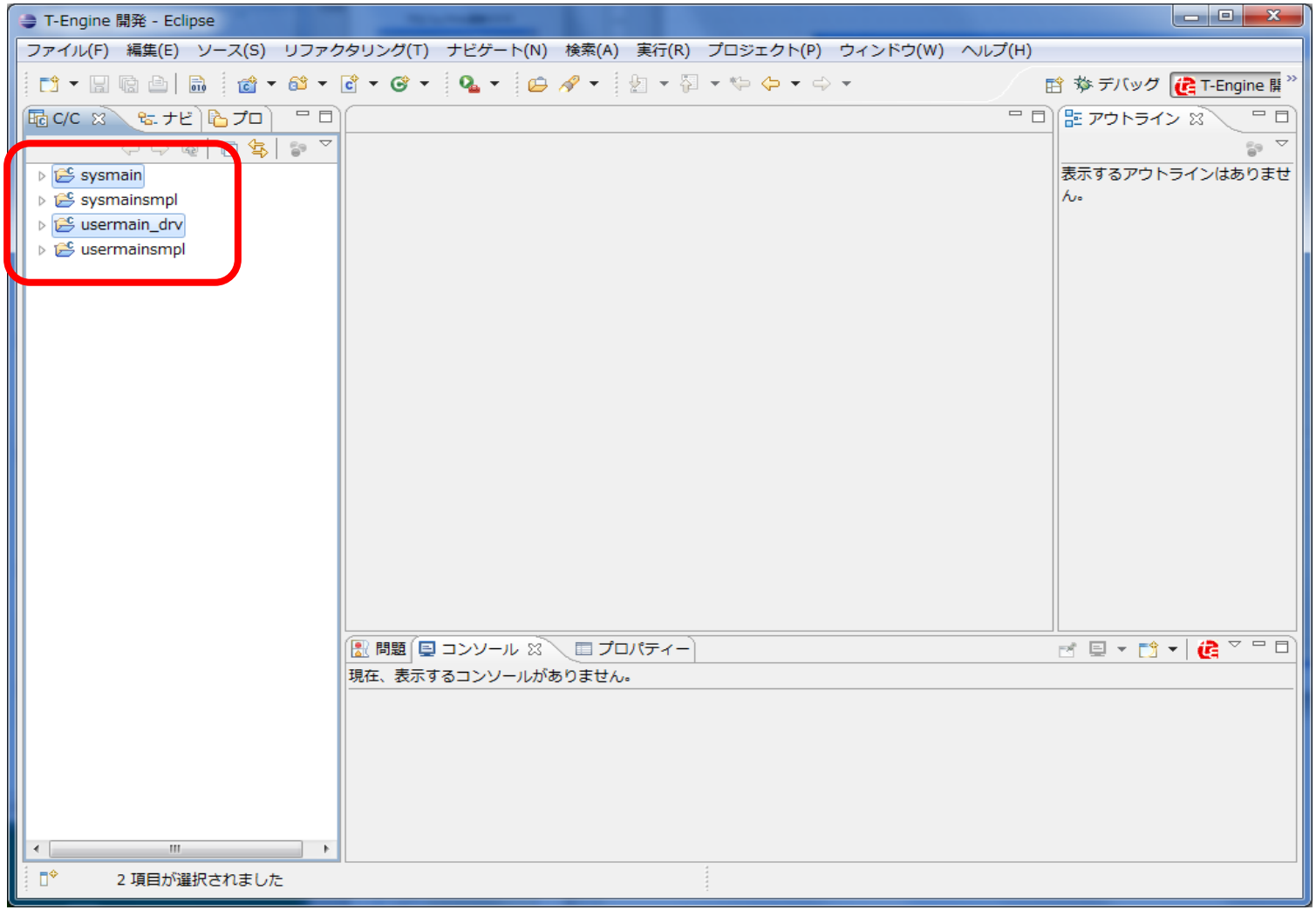
プロジェクトの追加 (3/7)



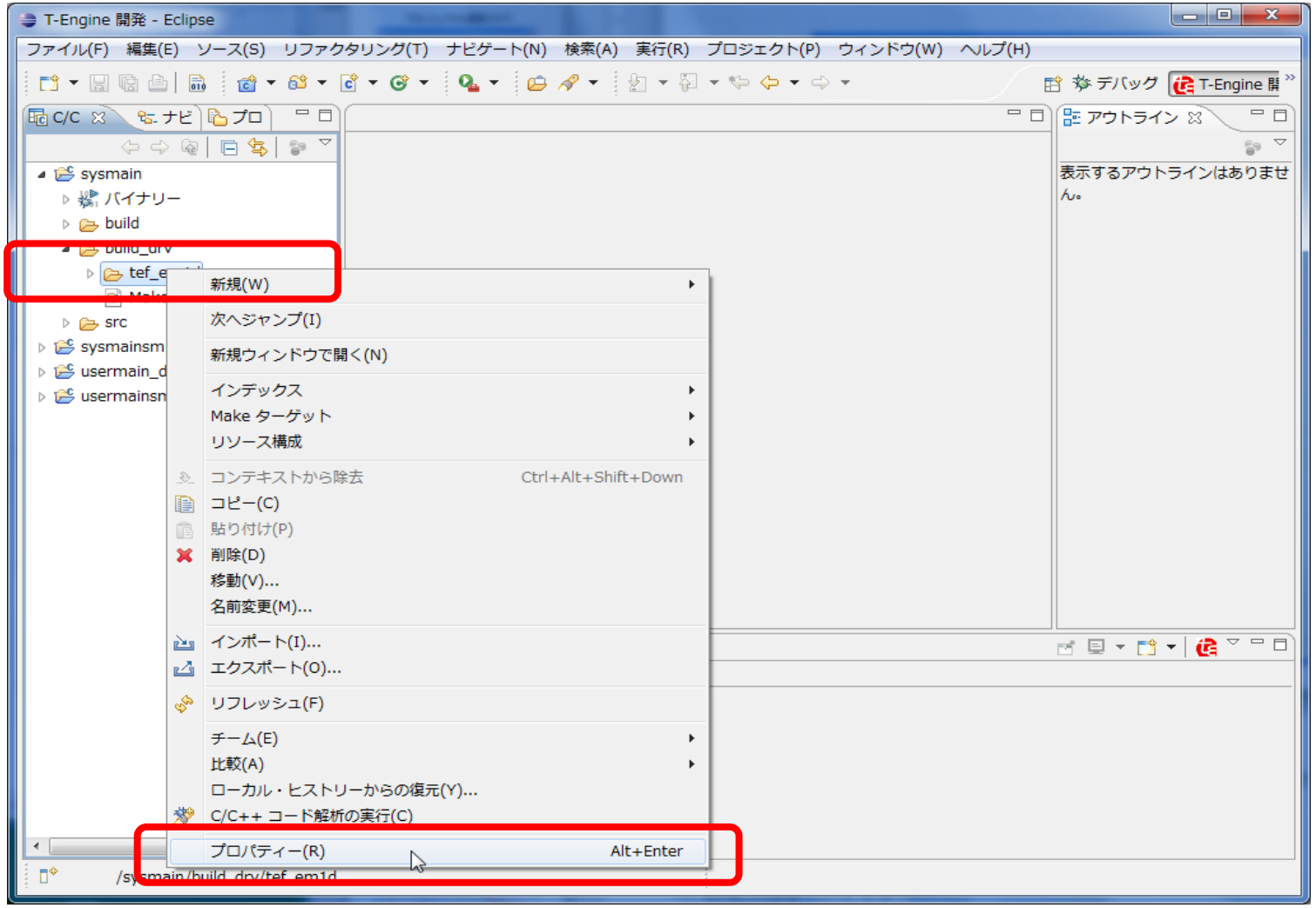
プロジェクトの追加 (4/7)



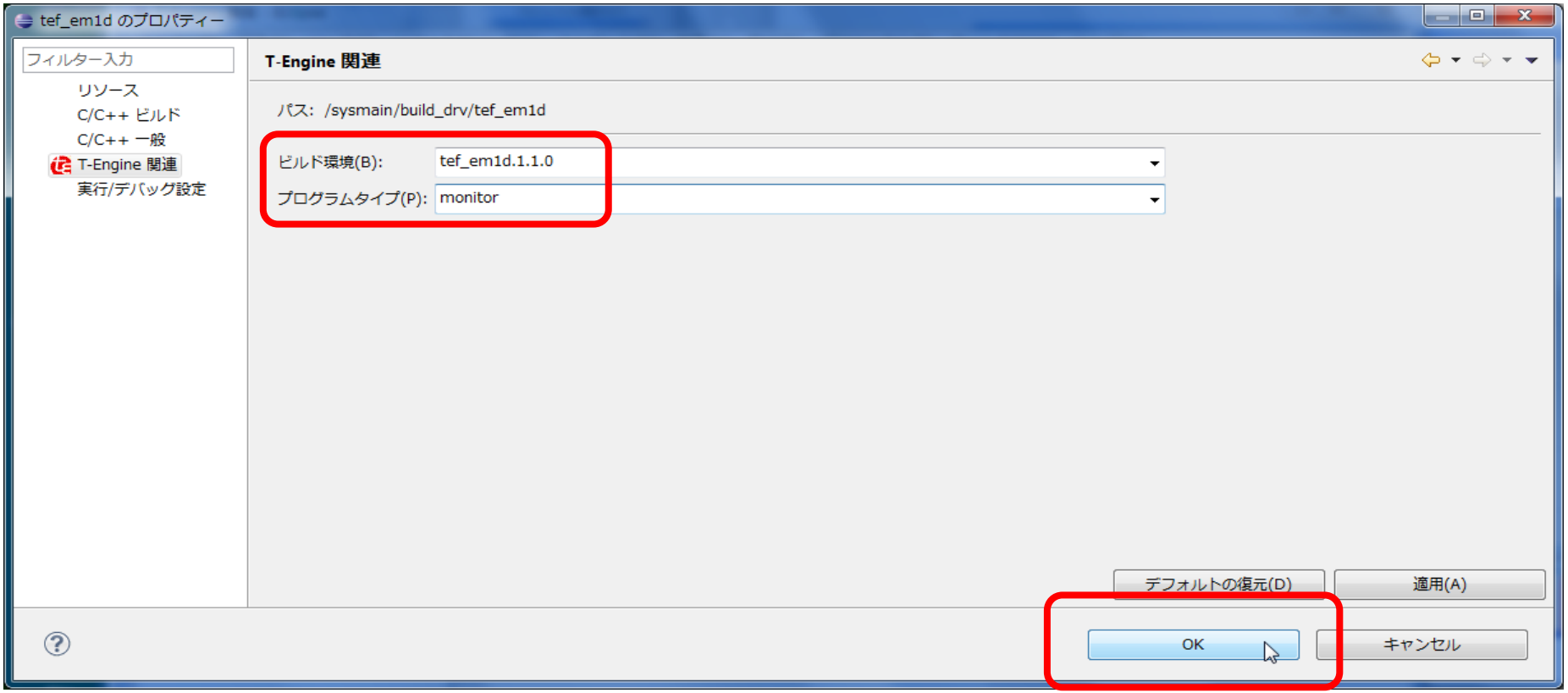
プロジェクトの追加 (5/7)



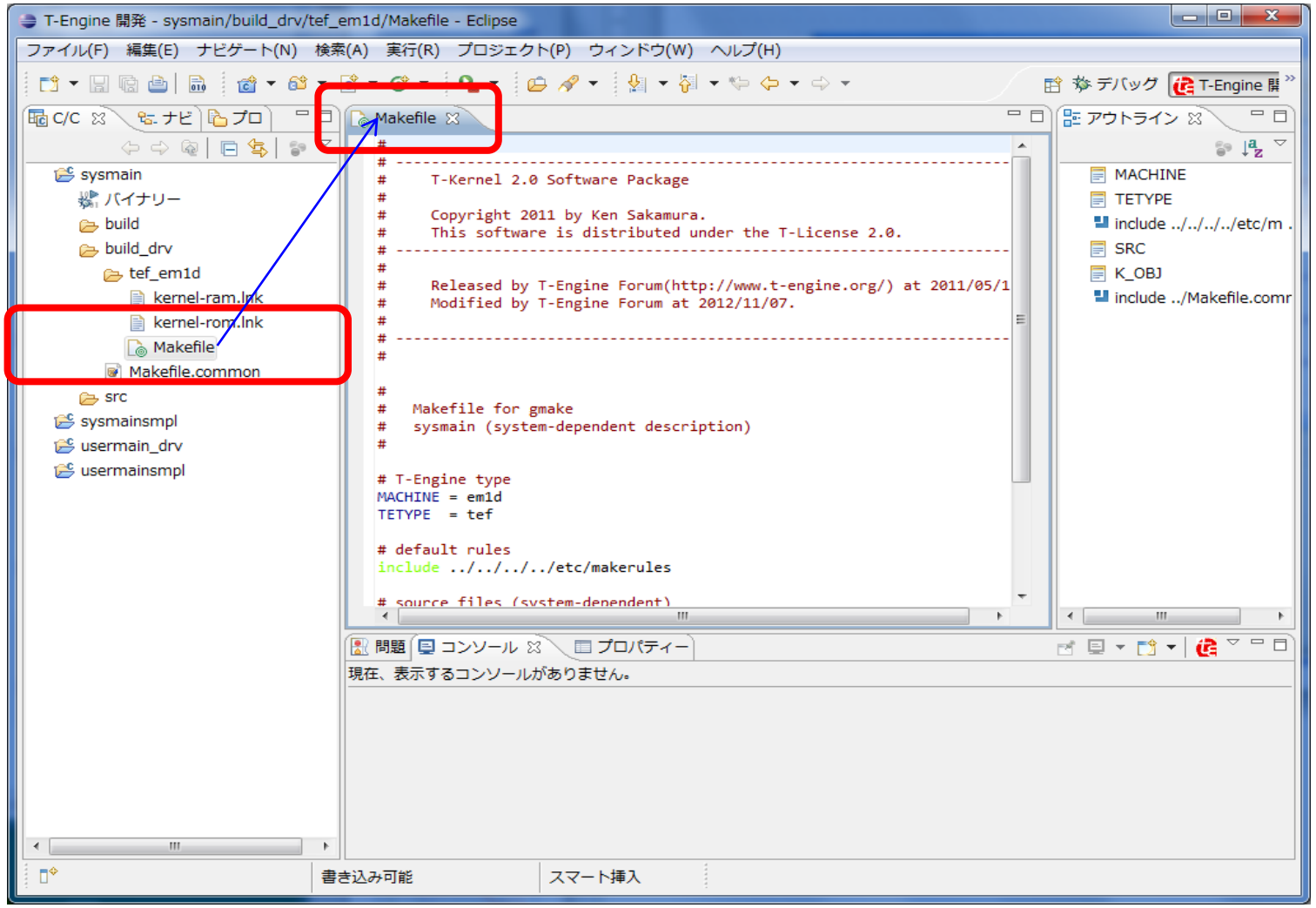
プロジェクトの追加 (6/7)



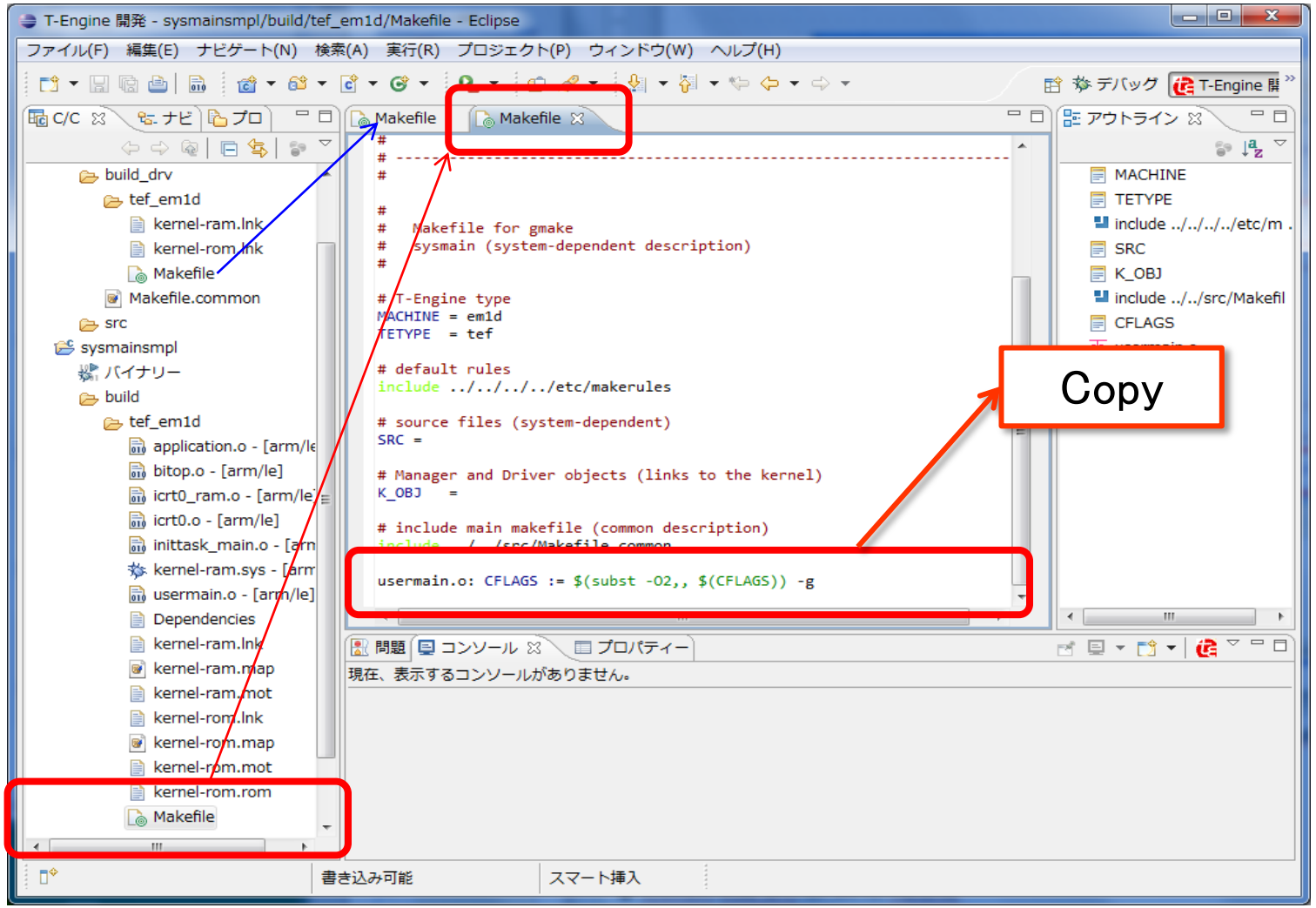
プロジェクトの追加 (7/7)



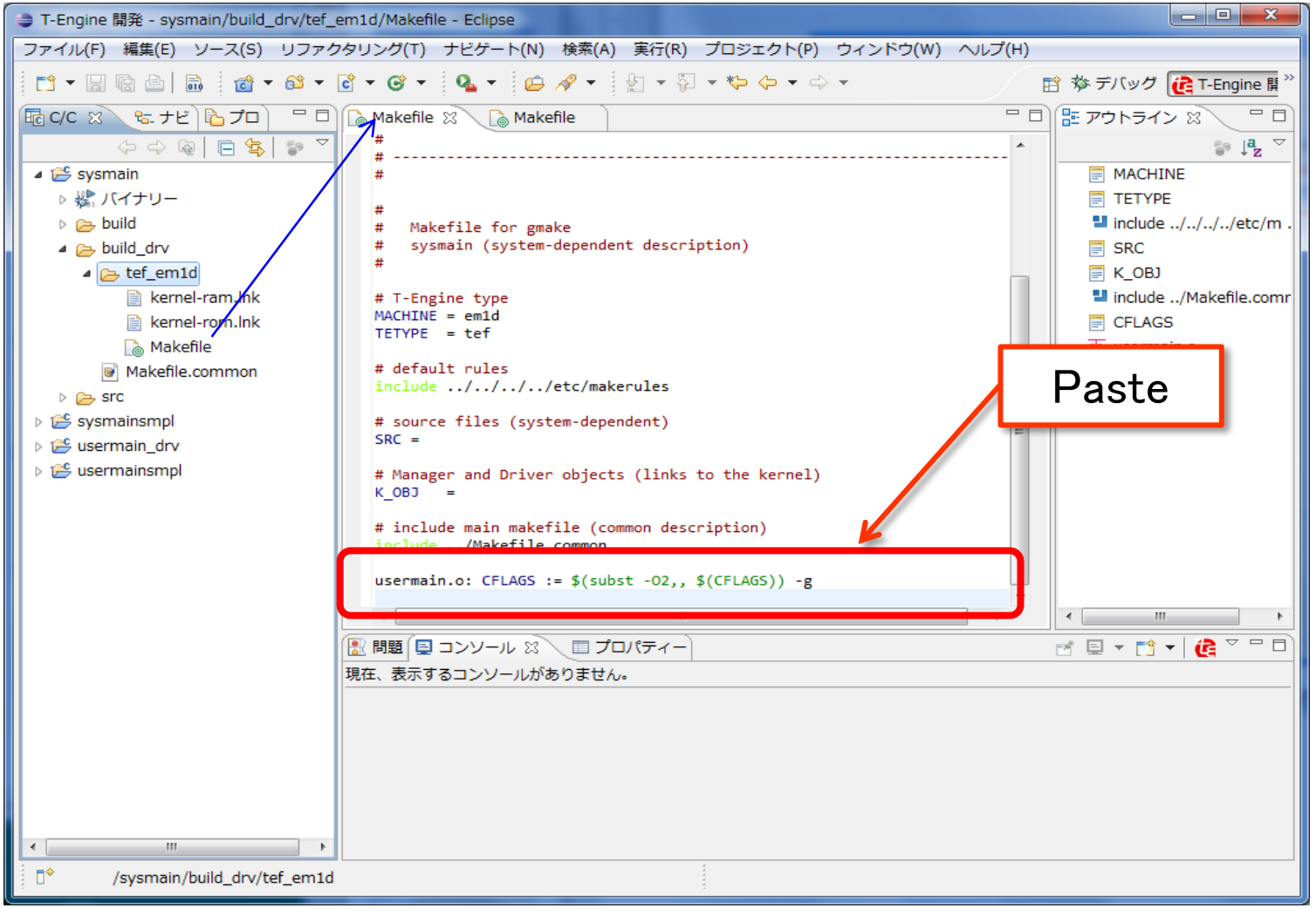
Makefileの編集 (1/3)



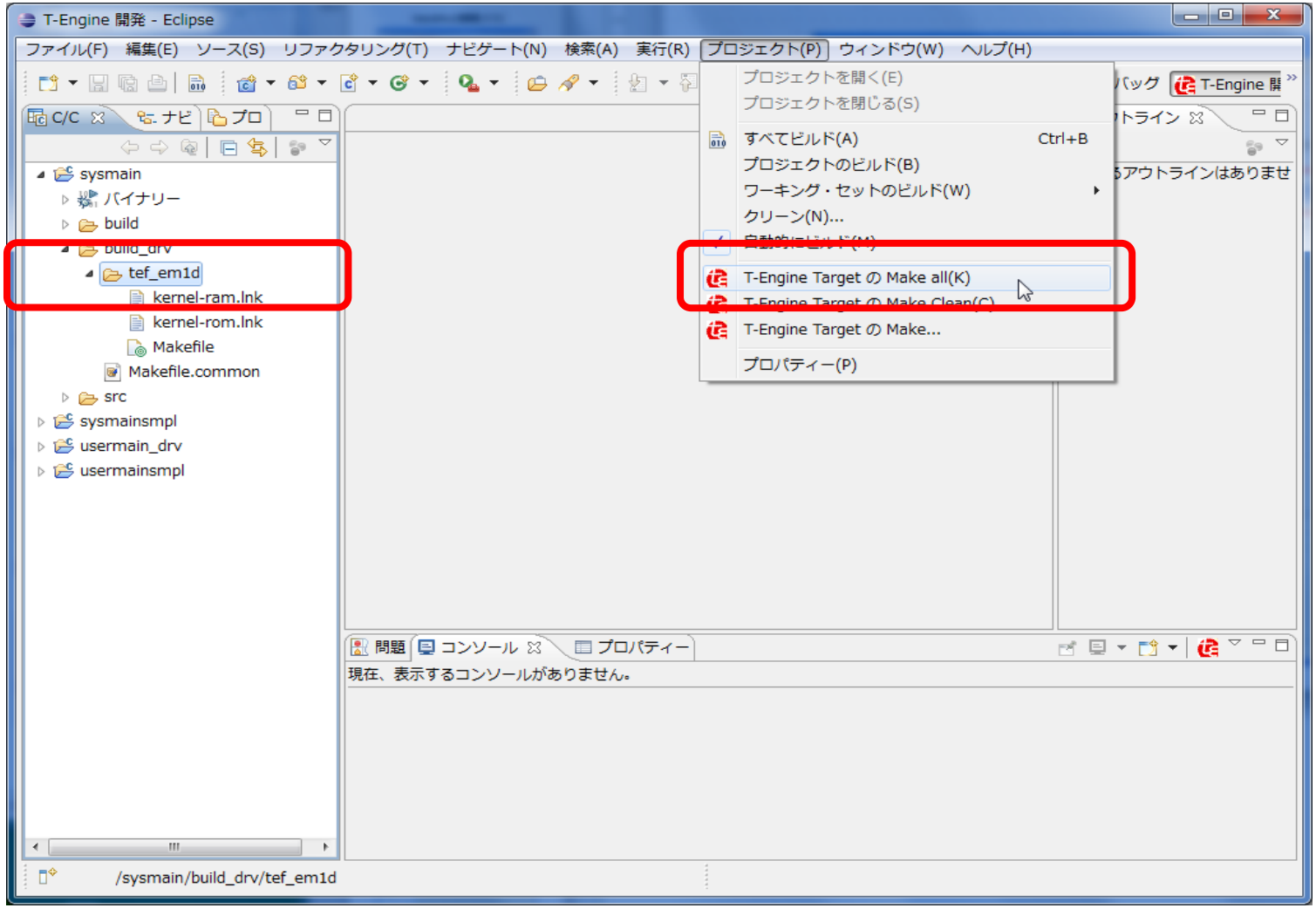
Makefileの編集 (2/3)



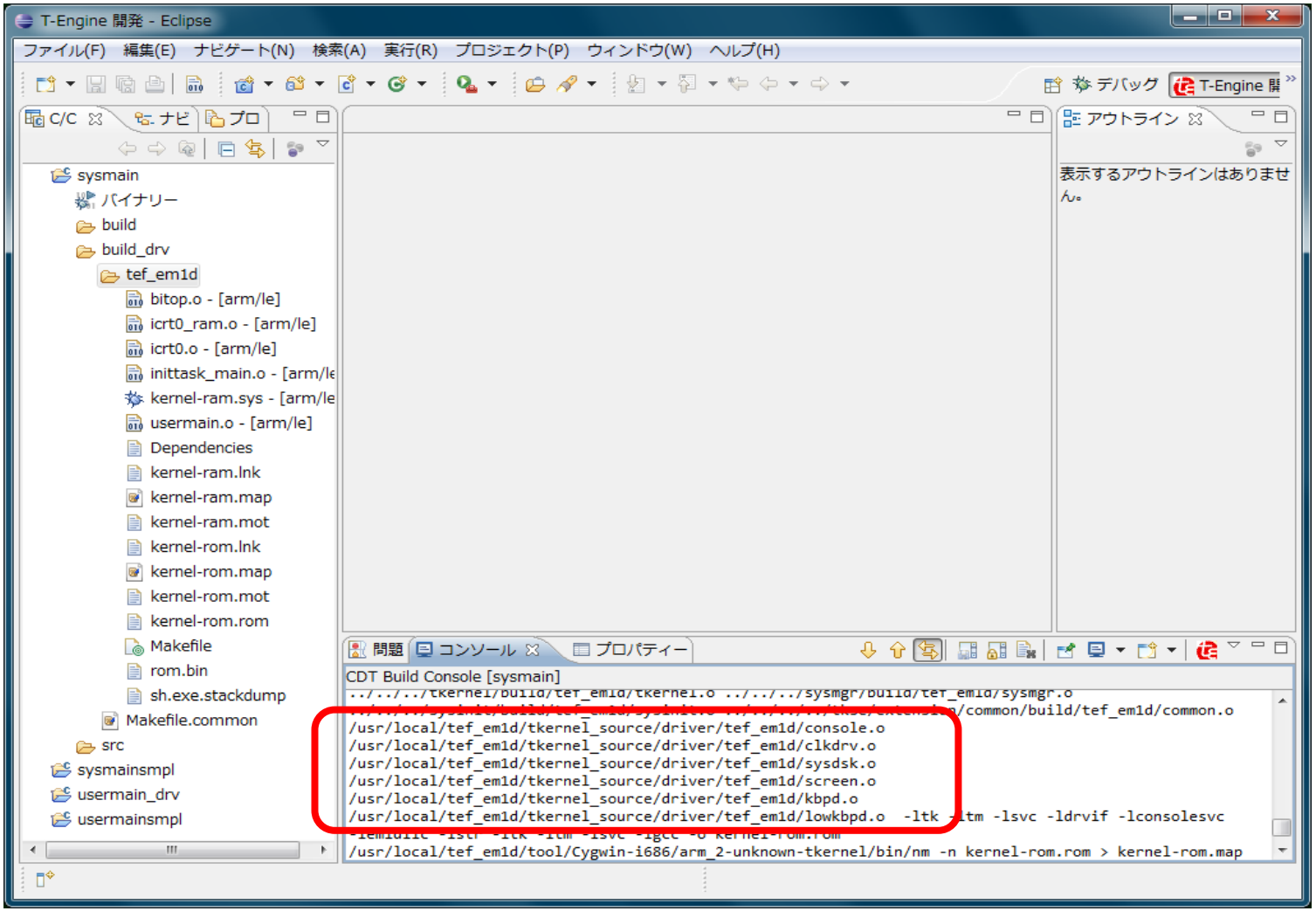
Makefileの編集 (3/3)



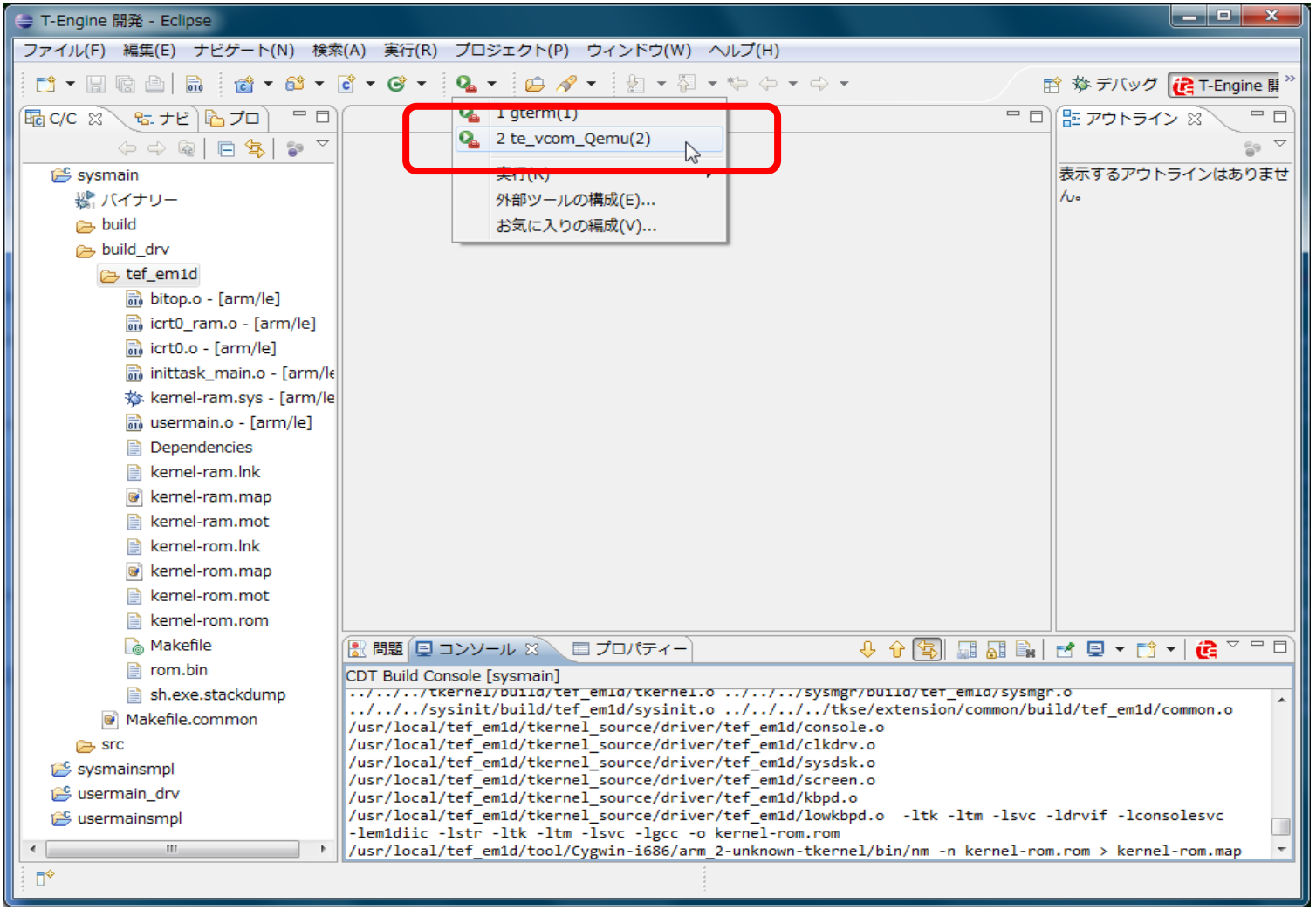
build_drvでビルド（1/2）



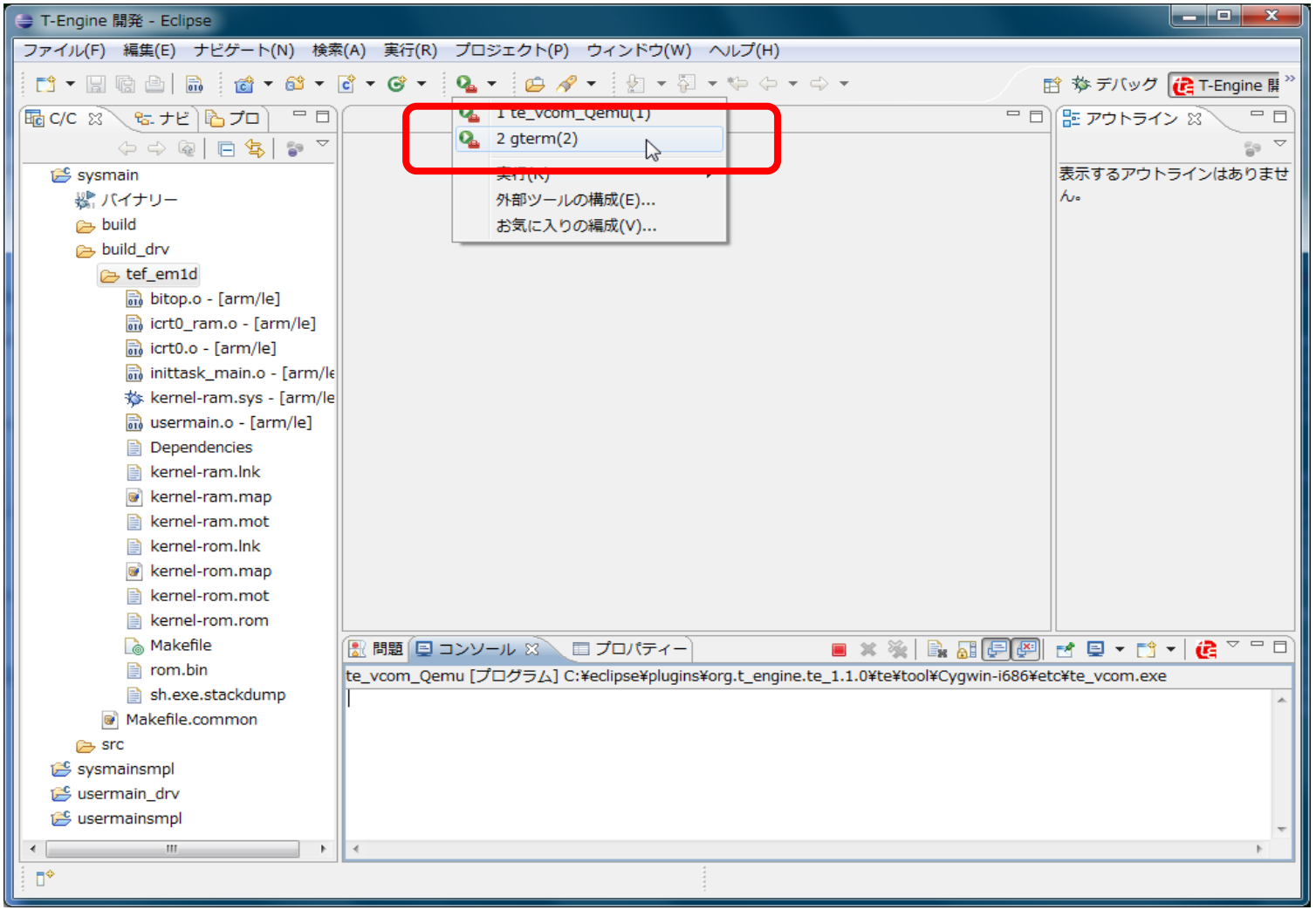
build_drvでビルド (2/2)



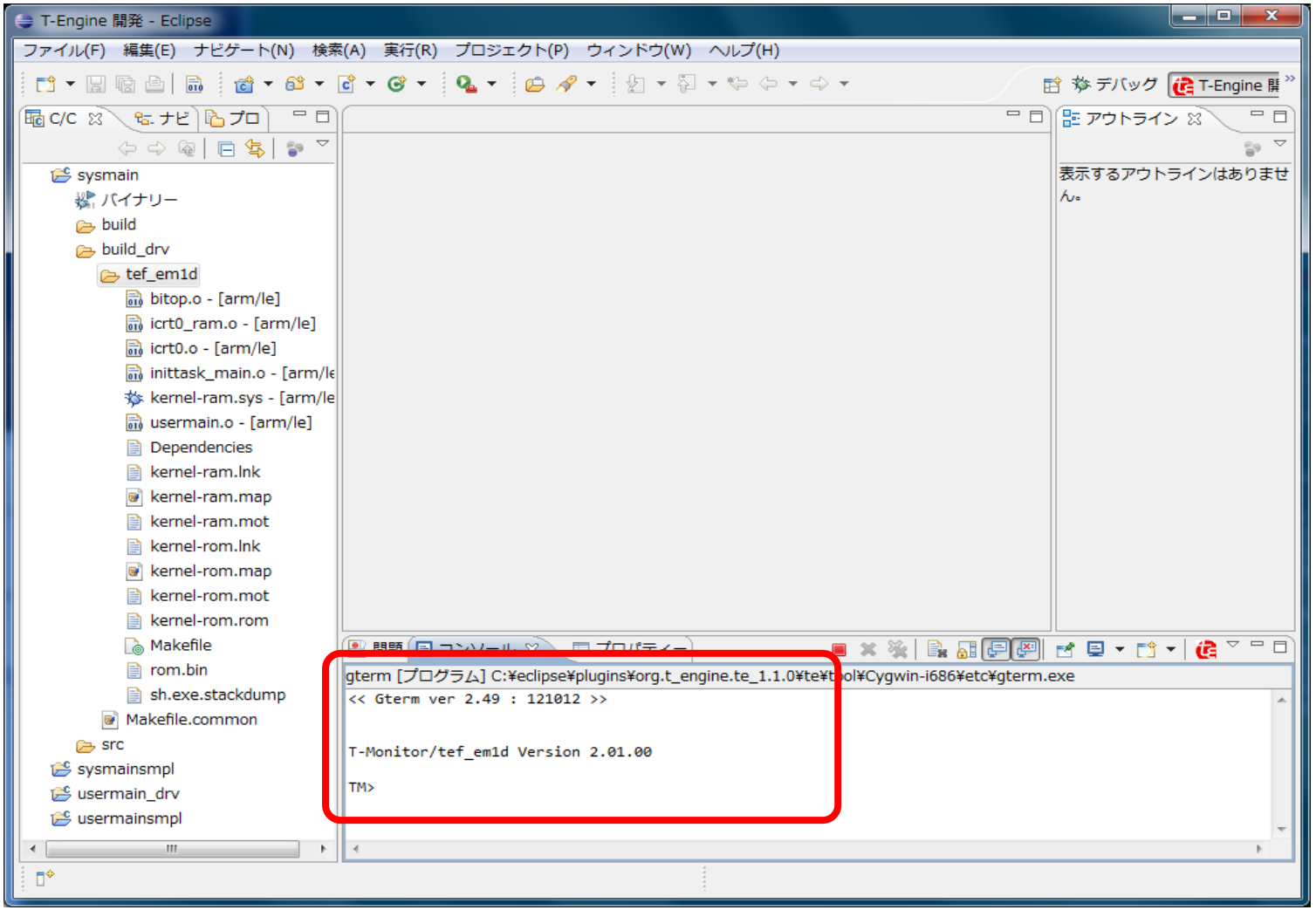
Qemuと接続 (1/3)


T-Engine 開発 - Eclipse
ファイル(F) 編集(E) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
デバッグ
T-Engine 開
C/C
ナビ
プロ
1 gterm(1)
2 te_vcom_Qemu(2)
実行(R)
外部ツールの構成(E)...
お気に入りの編成(V)...
アウトライン
表示するアウトラインはありません。
sysmain
バイナリー
build
build_drv
tef_em1d
bitop.o - [arm/le]
icrt0_ram.o - [arm/le]
icrt0.o - [arm/le]
inittask_main.o - [arm/le
kernel-ram.sys - [arm/le
usermain.o - [arm/le]
Dependencies
kernel-ram.lnk
kernel-ram.map
kernel-ram.mot
kernel-rom.lnk
kernel-rom.map
kernel-rom.mot
kernel-rom.rom
Makefile
rom.bin
sh.exe.stackdump
Makefile.common
src
sysmainsmpl
usermain_drv
usermainsmpl
問題
コンソール
プロパティー
CDT Build Console [sysmain]
../../../tkernel/build/tef_em1d/tkernel.o ../../../sysmgr/build/tef_em1d/sysmgr.o
../../../sysinit/build/tef_em1d/sysinit.o ../../../../tkse/extension/common/build/tef_em1d/common.o
/usr/local/tef_em1d/tkernel_source/driver/tef_em1d/console.o
/usr/local/tef_em1d/tkernel_source/driver/tef_em1d/clkdrv.o
/usr/local/tef_em1d/tkernel_source/driver/tef_em1d/sysdsk.o
/usr/local/tef_em1d/tkernel_source/driver/tef_em1d/screen.o
/usr/local/tef_em1d/tkernel_source/driver/tef_em1d/kbpd.o
/usr/local/tef_em1d/tkernel_source/driver/tef_em1d/lowkbpd.o -ltk -ltm -lsvc -ldrvif -lconsolesvc
-lem1diic -lstr -ltk -ltm -lsvc -lgcc -o kernel-rom.rom
/usr/local/tef_em1d/tool/Cygwin-i686/arm_2-unknown-tkernel/bin/nm -n kernel-rom.rom > kernel-rom.map

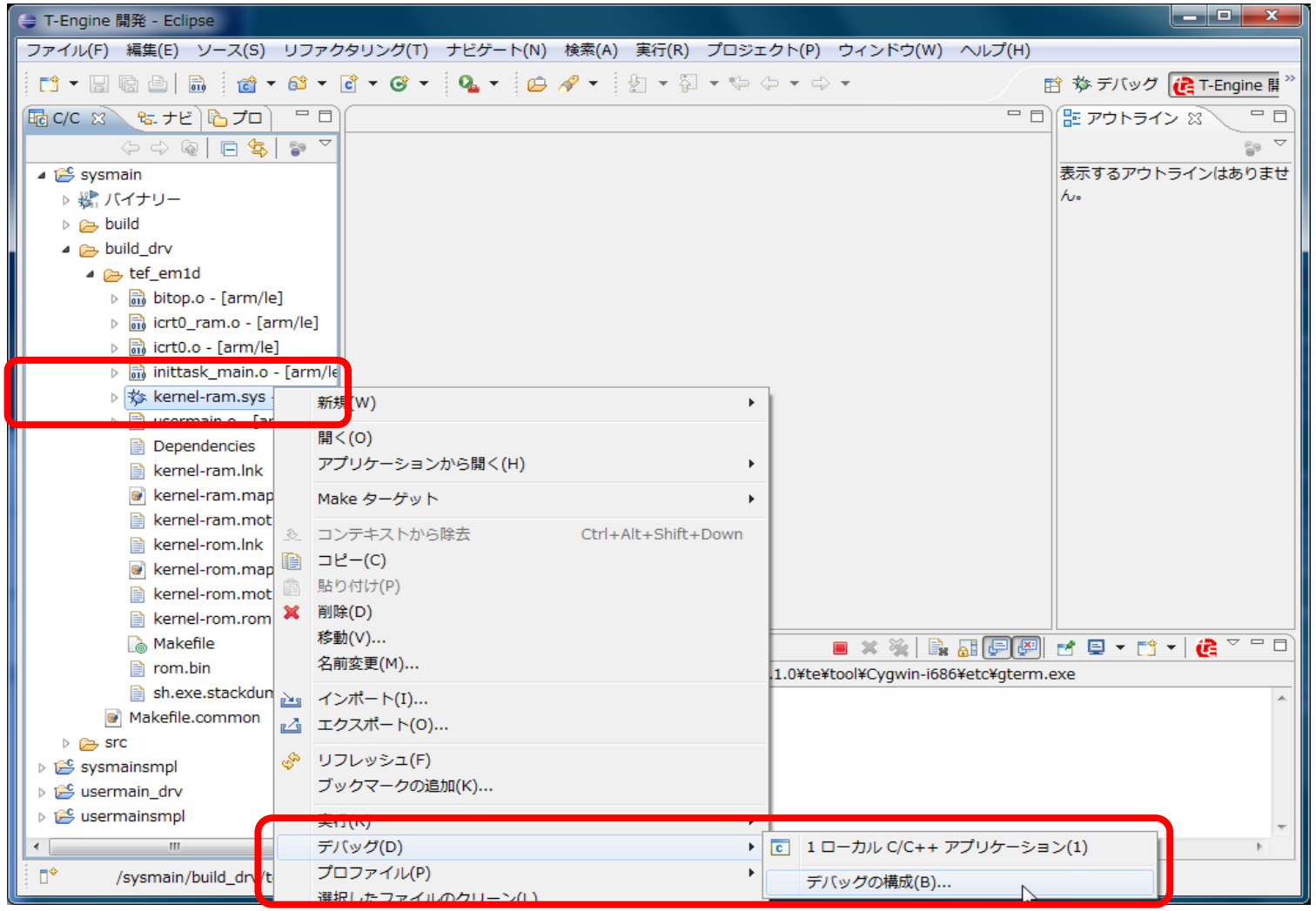
Qemuと接続 (2/3)


T-Engine 開発 - Eclipse
ファイル(F) 編集(E) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
デバッグ
T-Engine 開
C/C
ナビ
プロ
1 te_vcom_Qemu(1)
2 gterm(2)
実行(R)
外部ツールの構成(E)...
お気に入りの編成(V)...
アウトライン
表示するアウトラインはありません。
sysmain
バイナリー
build
build_drv
tef_em1d
bitop.o - [arm/le]
icrt0_ram.o - [arm/le]
icrt0.o - [arm/le]
inittask_main.o - [arm/le
kernel-ram.sys - [arm/le
usermain.o - [arm/le]
Dependencies
kernel-ram.lnk
kernel-ram.map
kernel-ram.mot
kernel-rom.lnk
kernel-rom.map
kernel-rom.mot
kernel-rom.rom
Makefile
rom.bin
sh.exe.stackdump
Makefile.common
src
sysmainsmpl
usermain_drv
usermainsmpl
問題
コンソール
プロパティー
te_vcom_Qemu [プログラム] C:¥eclipse¥plugins¥org.t_engine.te_1.1.0¥te¥tool¥Cygwin-i686¥etc¥te_vcom.exe

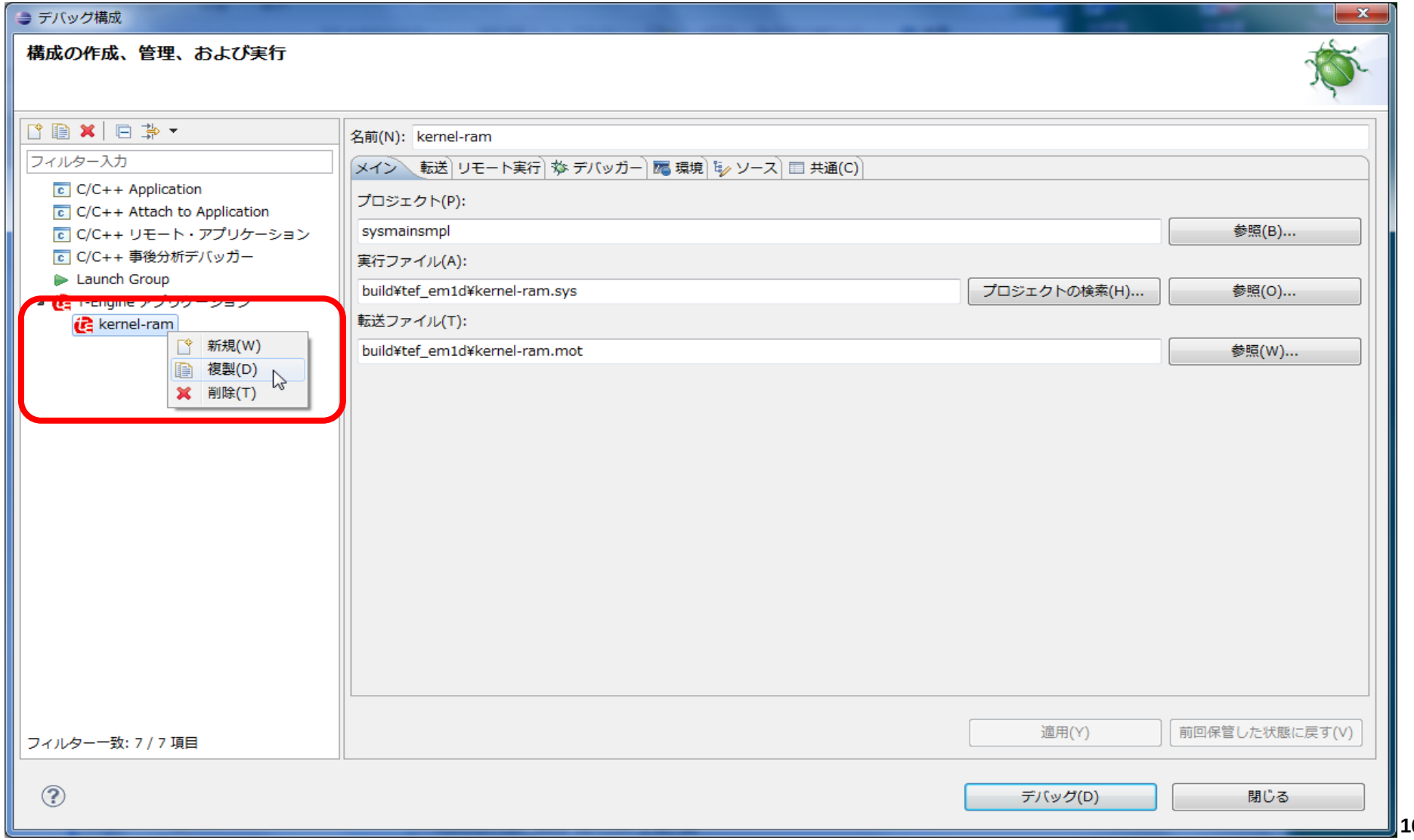
Qemuと接続 (3/3)


T-Engine 開発 - Eclipse
ファイル(F) 編集(E) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
デバッグ
T-Engine
C/C
ナビ
プロ
アウトライン
表示するアウトラインはありません。
sysmain
バイナリー
build
build_drv
tef_em1d
bitop.o - [arm/le]
icrt0_ram.o - [arm/le]
icrt0.o - [arm/le]
inittask_main.o - [arm/le
kernel-ram.sys - [arm/le
usermain.o - [arm/le]
Dependencies
kernel-ram.lnk
kernel-ram.map
kernel-ram.mot
kernel-rom.lnk
kernel-rom.map
kernel-rom.mot
kernel-rom.rom
Makefile
rom.bin
sh.exe.stackdump
Makefile.common
src
sysmainsmpl
usermain_drv
usermainsmpl
問題
コンソール
プロパティー
gterm [プログラム] C:¥eclipse¥plugins¥org.t_engine.te_1.1.0¥te¥tool¥Cygwin-i686¥etc¥gterm.exe
<< Gterm ver 2.49 : 121012 >>
T-Monitor/tef_em1d Version 2.01.00
TM>

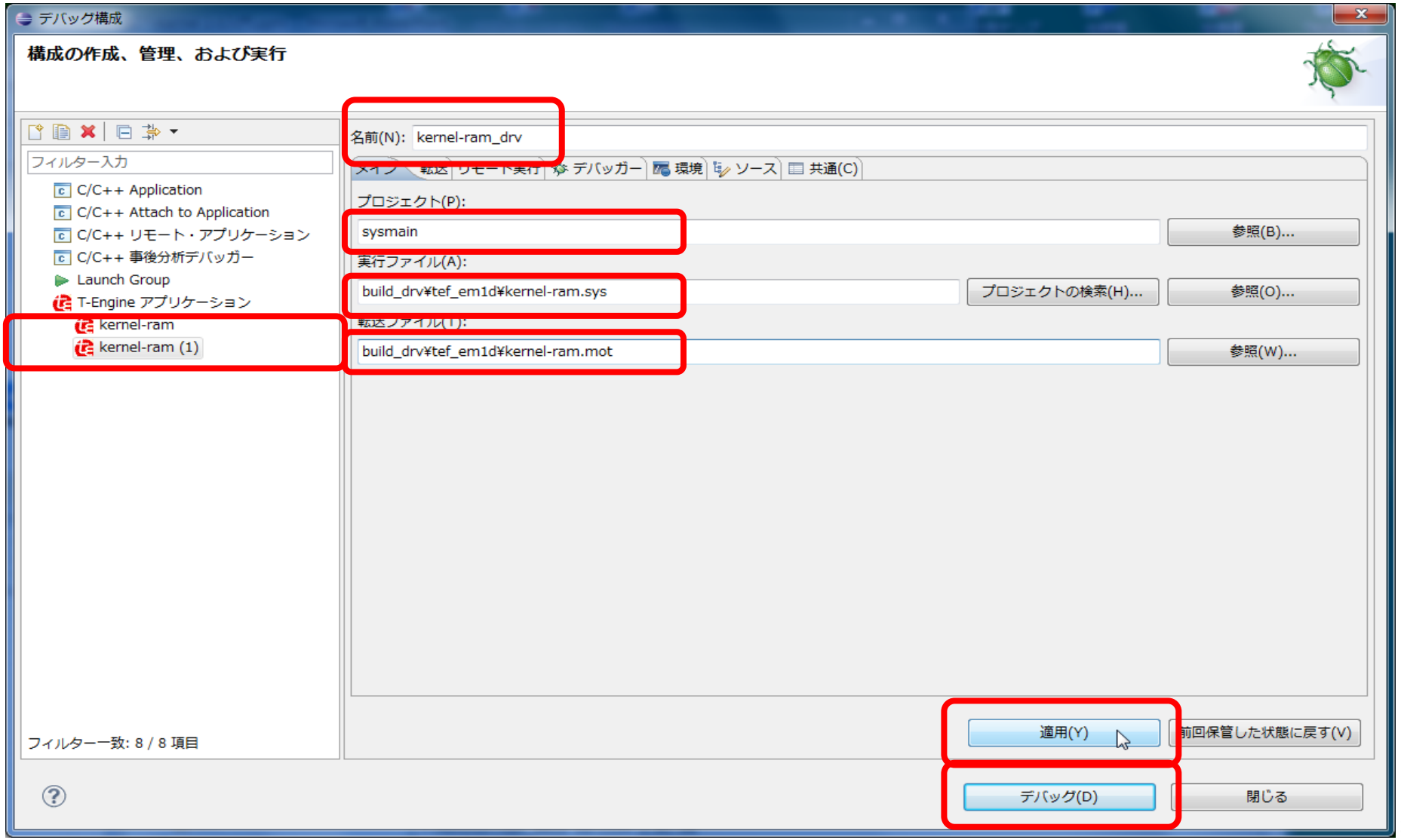
デバッグの構成 (1/4)


T-Engine 開発 - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
デバッグ
T-Engine 開
C/C
ナビ
プロ
アウトライン
表示するアウトラインはありません。
sysmain
バイナリー
build
build_drv
tef_em1d
bitop.o - [arm/le]
icrt0_ram.o - [arm/le]
icrt0.o - [arm/le]
inittask_main.o - [arm/le]
kernel-ram.sys
Dependencies
kernel-ram.lnk
kernel-ram.map
kernel-ram.mot
kernel-rom.lnk
kernel-rom.map
kernel-rom.mot
kernel-rom.rom
Makefile
rom.bin
sh.exe.stackdun
Makefile.common
src
sysmainsmpl
usermain_drv
usermainsmpl
/sysmain/build_drv/t
新規(W)
開く(O)
アプリケーションから開く(H)
Make ターゲット
コンテキストから除去 Ctrl+Alt+Shift+Down
コピー(C)
貼り付け(P)
削除(D)
移動(V)...
名前変更(M)...
インポート(I)...
エクスポート(O)...
リフレッシュ(F)
ブックマークの追加(K)...
デバッグ(D)
プロファイル(P)
1 ローカル C/C++ アプリケーション(1)
デバッグの構成(B)...
.1.0¥te¥tool¥Cygwin-i686¥etc¥gterm.exe

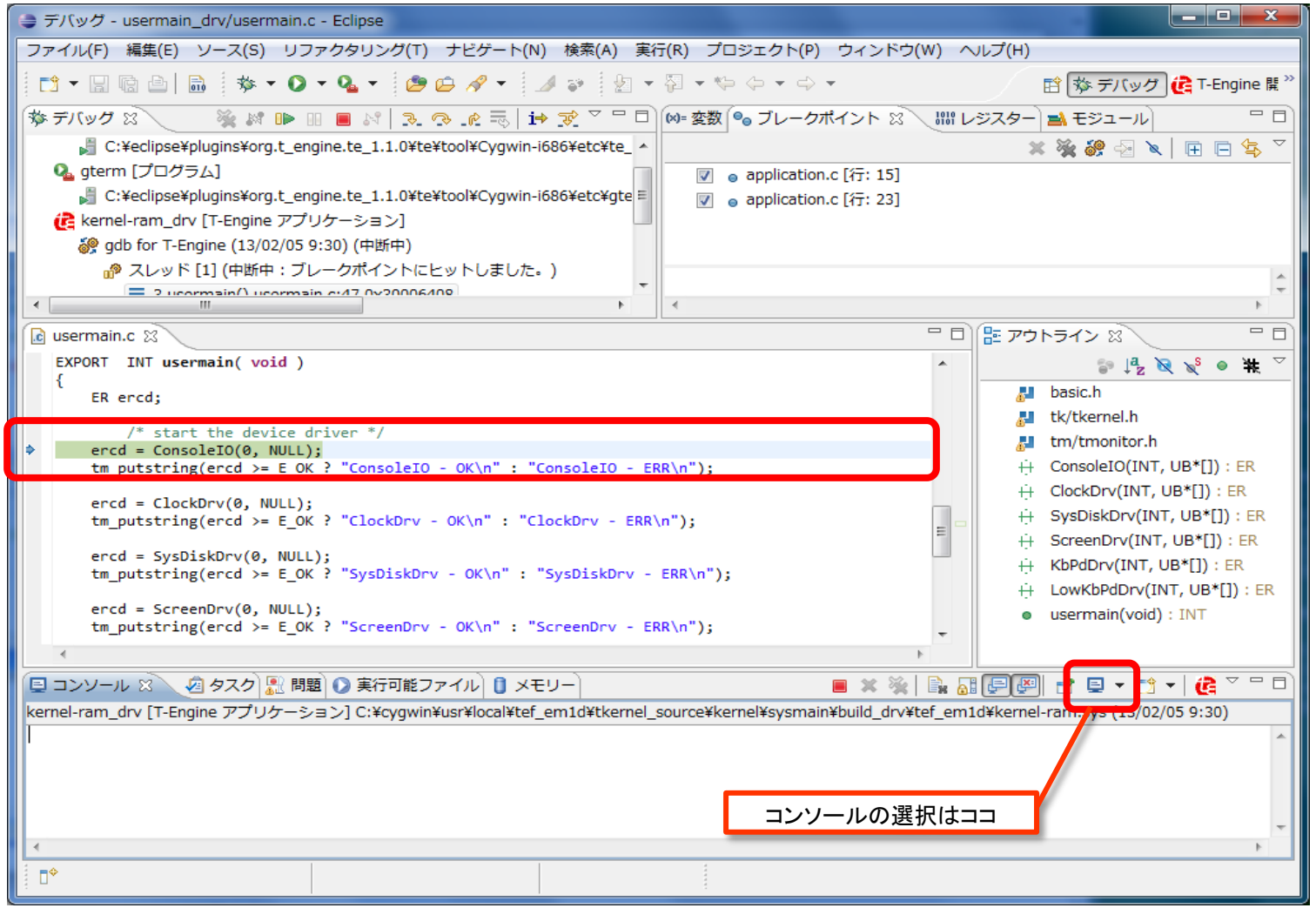
デバッグの構成 (2/4)


デバッグ構成
構成の作成、管理、および実行
フィルター入力
C/C++ Application
C/C++ Attach to Application
C/C++ リモート・アプリケーション
C/C++ 事後分析デバッガー
Launch Group
kernel-ram
新規(W)
複製(D)
削除(T)
フィルター一致: 7 / 7 項目
名前(N): kernel-ram
メイン
転送
リモート実行
デバッガー
環境
ソース
共通(C)
プロジェクト(P):
sysmainsmpl
参照(B)...
実行ファイル(A):
build¥tef_em1d¥kernel-ram.sys
プロジェクトの検索(H)...
参照(O)...
転送ファイル(T):
build¥tef_em1d¥kernel-ram.mot
参照(W)...
適用(Y)
前回保管した状態に戻す(V)
デバッグ(D)
閉じる

デバッグの構成 (3/4)


デバッグ構成
構成の作成、管理、および実行
名前(N): kernel-ram_drv
フィルター入力
C/C++ Application
C/C++ Attach to Application
C/C++ リモート・アプリケーション
C/C++ 事後分析デバッガー
Launch Group
T-Engine アプリケーション
kernel-ram
kernel-ram (1)
メイン
転送
リモート実行
デバッガー
環境
ソース
共通(C)
プロジェクト(P):
sysmain
参照(B)...
実行ファイル(A):
build_drv¥tef_em1d¥kernel-ram.sys
プロジェクトの検索(H)...
参照(O)...
転送ファイル(T):
build_drv¥tef_em1d¥kernel-ram.mot
参照(W)...
フィルター一致: 8 / 8 項目
適用(Y)
前回保管した状態に戻す(V)
デバッグ(D)
閉じる

デバッグの構成 (4/4)


デバッグ - usermain_drv/usermain.c - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
デバッグ
T-Engine
C:¥eclipse¥plugins¥org.t_engine.te_1.1.0¥te¥tool¥Cygwin-i686¥etc¥te_
gterm [プログラム]
C:¥eclipse¥plugins¥org.t_engine.te_1.1.0¥te¥tool¥Cygwin-i686¥etc¥gte
kernel-ram_drv [T-Engine アプリケーション]
gdb for T-Engine (13/02/05 9:30) (中断中)
スレッド [1] (中断中：ブレークポイントにヒットしました。)
変数
ブレークポイント
レジスター
モジュール
application.c [行: 15]
application.c [行: 23]
usermain.c
EXPORT INT usermain(void)
{
ER ercd;
/* start the device driver */
ercd = ConsoleIO(0, NULL);
tm_putstring(ercd >= E_OK ? "ConsoleIO - OK\n" : "ConsoleIO - ERR\n");
ercd = ClockDrv(0, NULL);
tm_putstring(ercd >= E_OK ? "ClockDrv - OK\n" : "ClockDrv - ERR\n");
ercd = SysDiskDrv(0, NULL);
tm_putstring(ercd >= E_OK ? "SysDiskDrv - OK\n" : "SysDiskDrv - ERR\n");
ercd = ScreenDrv(0, NULL);
tm_putstring(ercd >= E_OK ? "ScreenDrv - OK\n" : "ScreenDrv - ERR\n");
アウトライン
basic.h
tk/tkernel.h
tm/tmonitor.h
ConsoleIO(INT, UB*[]) : ER
ClockDrv(INT, UB*[]) : ER
SysDiskDrv(INT, UB*[]) : ER
ScreenDrv(INT, UB*[]) : ER
KbPdDrv(INT, UB*[]) : ER
LowKbPdDrv(INT, UB*[]) : ER
usermain(void) : INT
コンソール
タスク
問題
実行可能ファイル
メモリー
kernel-ram_drv [T-Engine アプリケーション] C:¥cygwin¥usr¥local¥tef_em1d¥tkernel_source¥kernel¥sysmain¥build_drv¥tef_em1d¥kernel-ram...
コンソールの選択はココ

液晶画面(LCD)の表示

- Qemuの起動オプションを変更
  - デフォルトの設定はLCDなし
    → LCDを表示するように qemu.bat を変更
- 実行
- Eclipseと接続
- デバイスドライバを追加
- T-Kernel本体のビルド
- 液晶画面への描画

液晶画面への描画

- スクリーン(ディスプレイ)ドライバの操作
  - 描画関数の追加
  - デバイス仕様の確認
- 描画サンプルによる描画
  - 点 (point)
  - 矩形 (rectangle)
  - 線分 (line)

描画関数の追加

- 以下のファィルを usermain_drv にコピー
  - usermain.c, Makefile.usermain
  - display.h, display.c
- ビルド
  - make mode=debug に変更
  - sysmain/build_drv/tef_em1d を選択
  - [プロジェクト] → [T-Engine Target の Make all]
- ロード
  - sysmain/build_drv/tef_em1d/kernel-ram.sys を選択
  - メニュー → [デバッグ] → [デバッグの構成]
  - [kernel-ram_drv] → [デバッグ]

usermain()とMakefile.usermainへの追加

■ usermain_drv/usermain.c

```
EXPORT   INT      usermain( void )
{
            :
        tm_putstring(ercd >= E_OK ? "LowKbPdDrv - OK¥n" : "LowKbPdDrv - ERR¥n");

        /*------------------------------------------------------*/
        initDisplay();
        drawTest();
        tk_slp_tsk(TMO_FEVR);
        /*------------------------------------------------------*/

        tm_putstring((UB*)"Push any key to shutdown the T-Kernel.¥n");
            :
```

■ usermain_drv/Makefile.usermain

```
    :
# source files
SRC += usermain.c
SRC += display.c
```

LCDのオープン

```
EXPORT  void    initDisplay(void)
{
        DEV_SPEC  devspec;
        BMP       devbmp;
        FP        updfn;
        W         sz;
        ER        ercd;

        scrid = tk_opn_dev( "SCREEN", TD_UPDATE );

        ercd = tk_srea_dev( scrid, DN_SCRSPEC, &devspec, sizeof(DEV_SPEC), &sz );

        ercd = tk_srea_dev( scrid, DN_SCRUPDFN, &updfn, sizeof(FP), &sz );

        ercd = tk_srea_dev( scrid, DN_SCRBMP, &devbmp, sizeof(BMP), &sz );

        screenWidth  = devspec.hpixels;
        screenHeight = devspec.vpixels;
        bmp = (UH*)devbmp.baseaddr[0];
}
```

devspec の情報を確認

```
H       attr;       /* デバイス属性 */
H       planes;     /* プレーン数 */
H       pixbits;    /* ピクセルビット数(境界/有効) */
H       hpixels;    /* 横のピクセル数 */
H       vpixels;    /* 縦のピクセル数 */
H       hres;       /* 横の解像度 */
H       vres;       /* 縦の解像度 */
H       color[4];   /* カラー情報 */
H       resv[6];
```

スクリーンの概要

- カラー方式　RGB
- プレーン数　1
- 画面サイズ　800 × 480
- ピクセル値の形式　RRRRRGGGGGGBBBBB

- イメージ領域　devbmp.baseaddr[0]
- 更新関数　なし

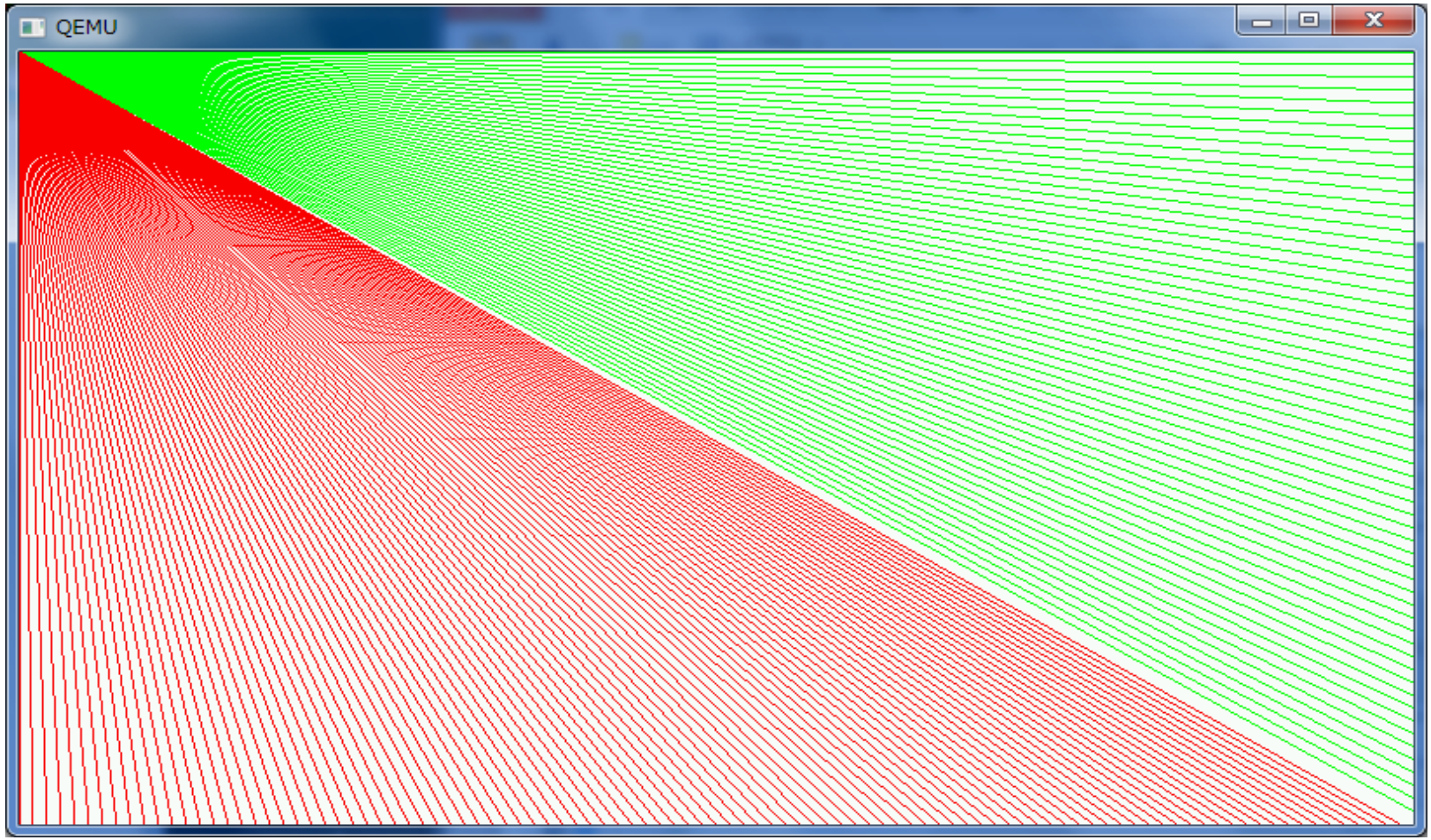
液晶画面への描画

- スクリーン(ディスプレイ)ドライバの操作
  - デバイス仕様の確認
- 描画サンプルによる描画
  - 点 (point)
  - 矩形 (rectangle)
  - 線分 (line)

描画関数

- 点の描画
  - void　drawPnt(H x, H y, INT col);
- 矩形の描画
  - void　drawRect(H x1, H y1, H x2, H y2, INT col, BOOL fill);
- 線分の描画
  - void　drawLine(H x1, H y1, H x2, H y2, INT col);
- 描画テスト
  - void　drawTest(void);

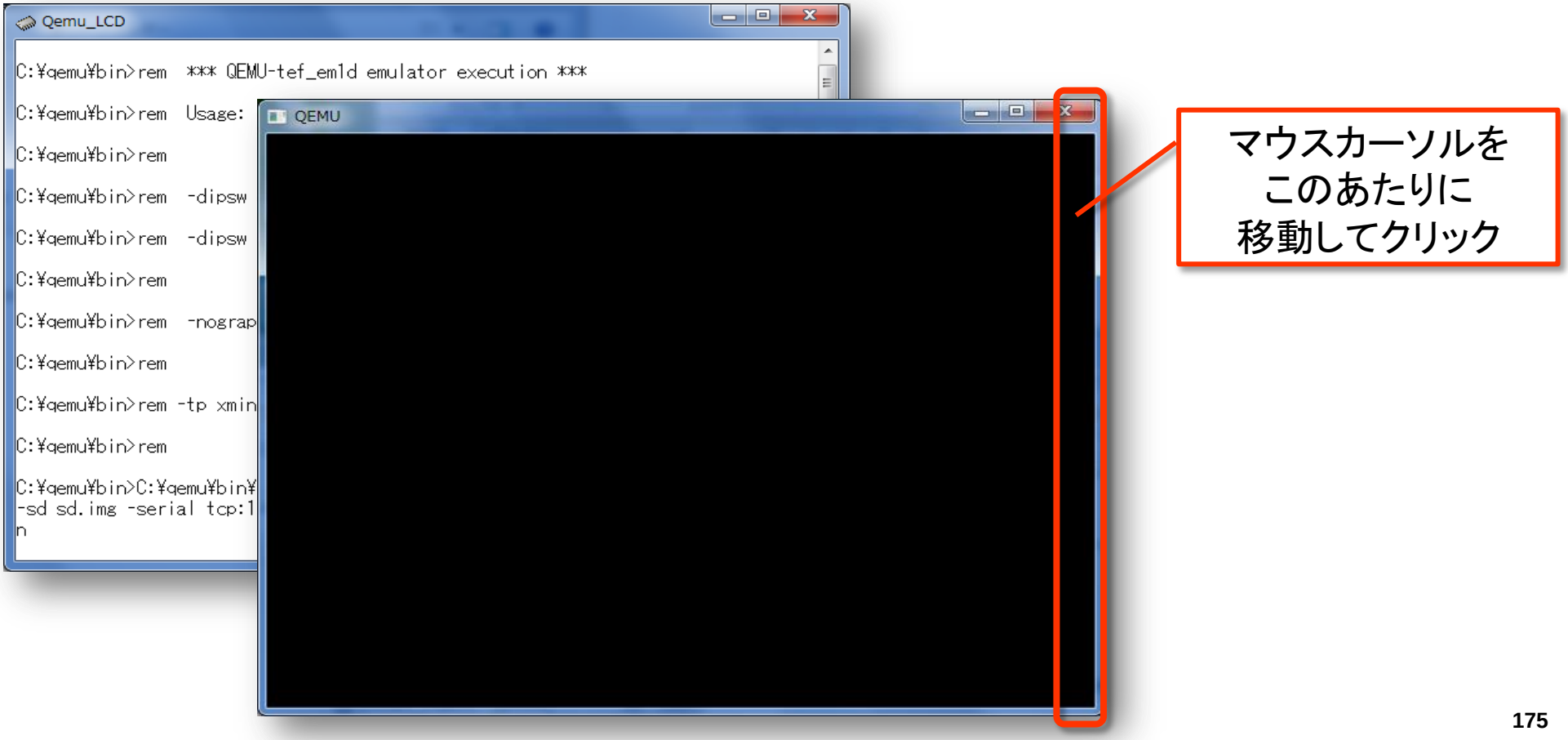
描画テストの実行結果



簡易ブレーク機能の追加

KBPDのイベントでブレーク

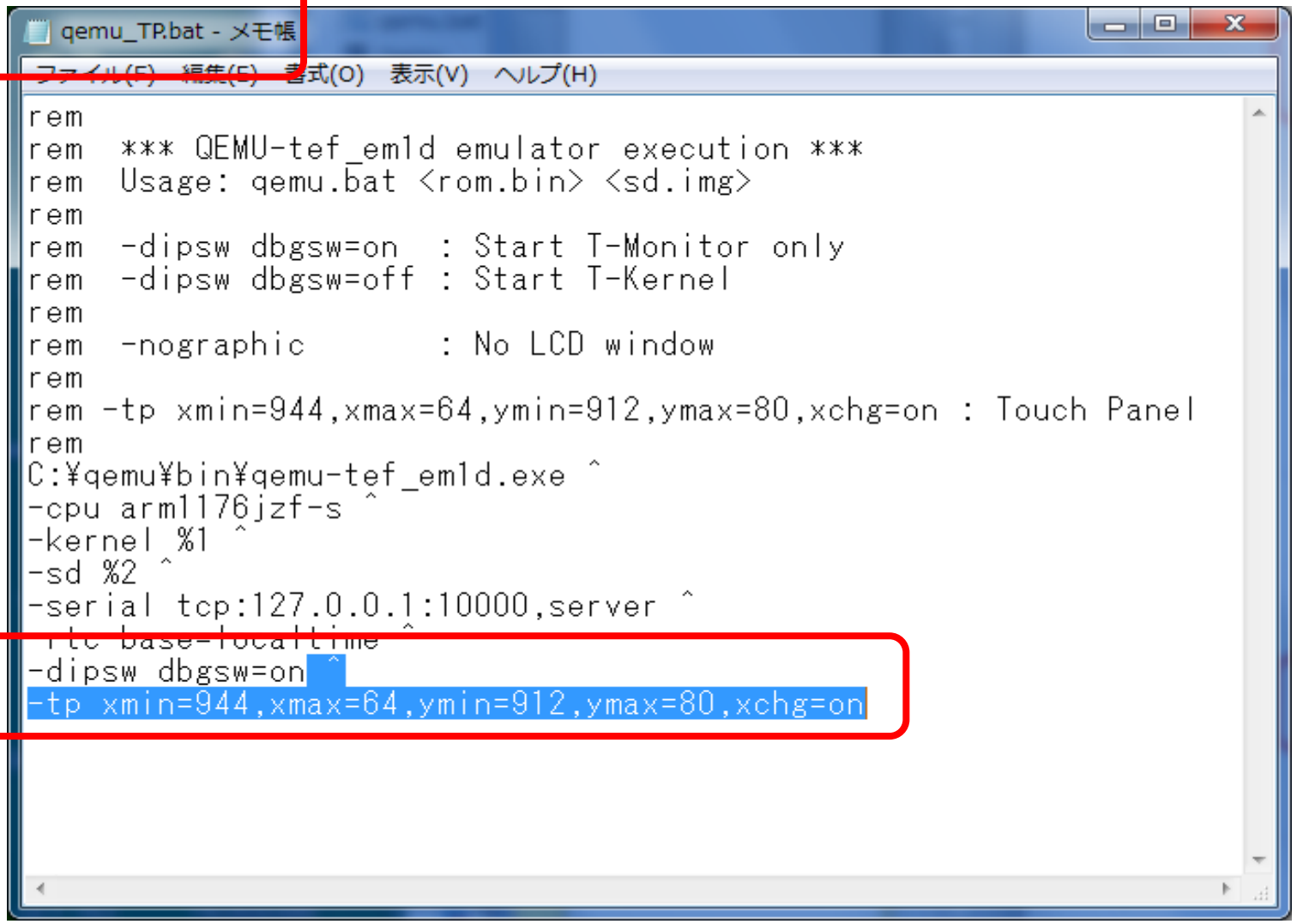
- KBPDドライバからのイベントでブレークを発生



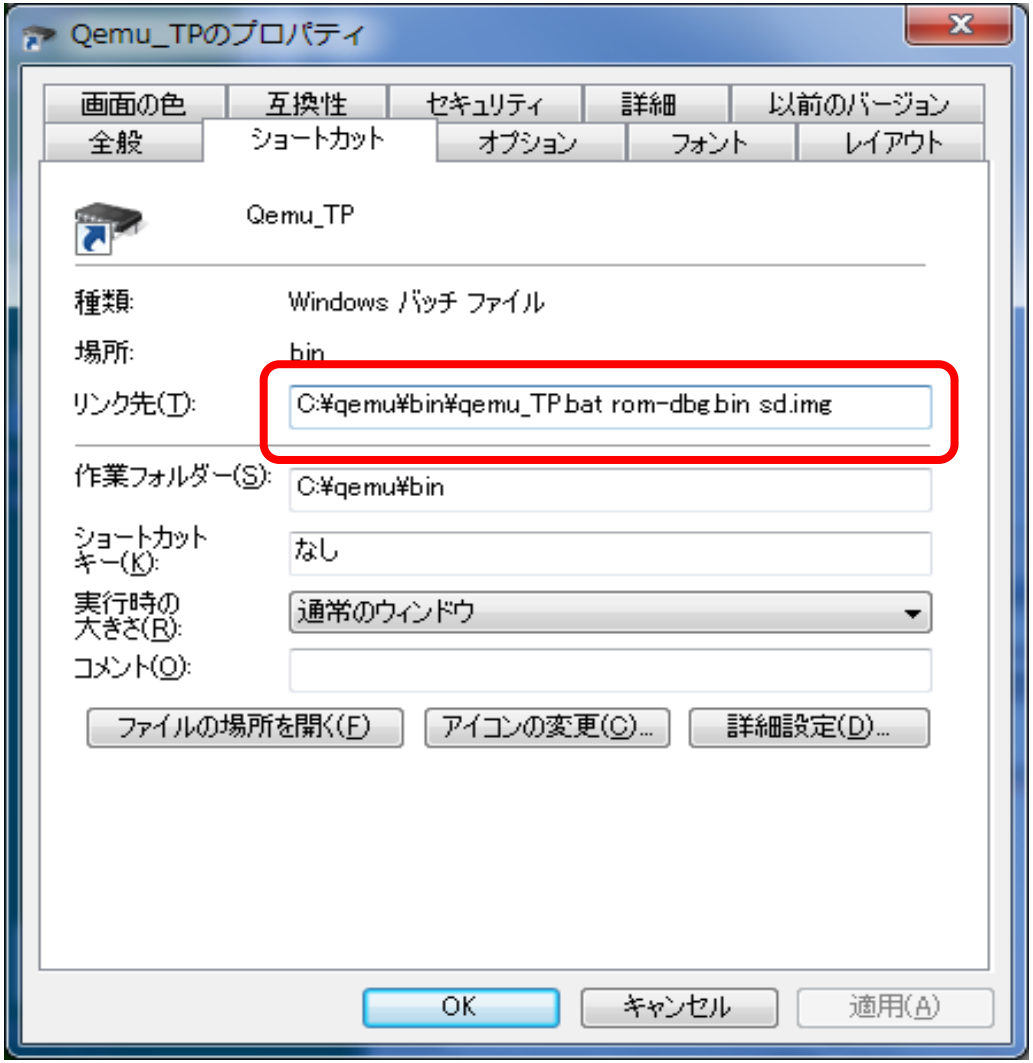
タッチパネル機能の追加

- Qemuの設定を変更
- タッチパネル用タスクの追加

Qemuの設定を変更 (1/2)



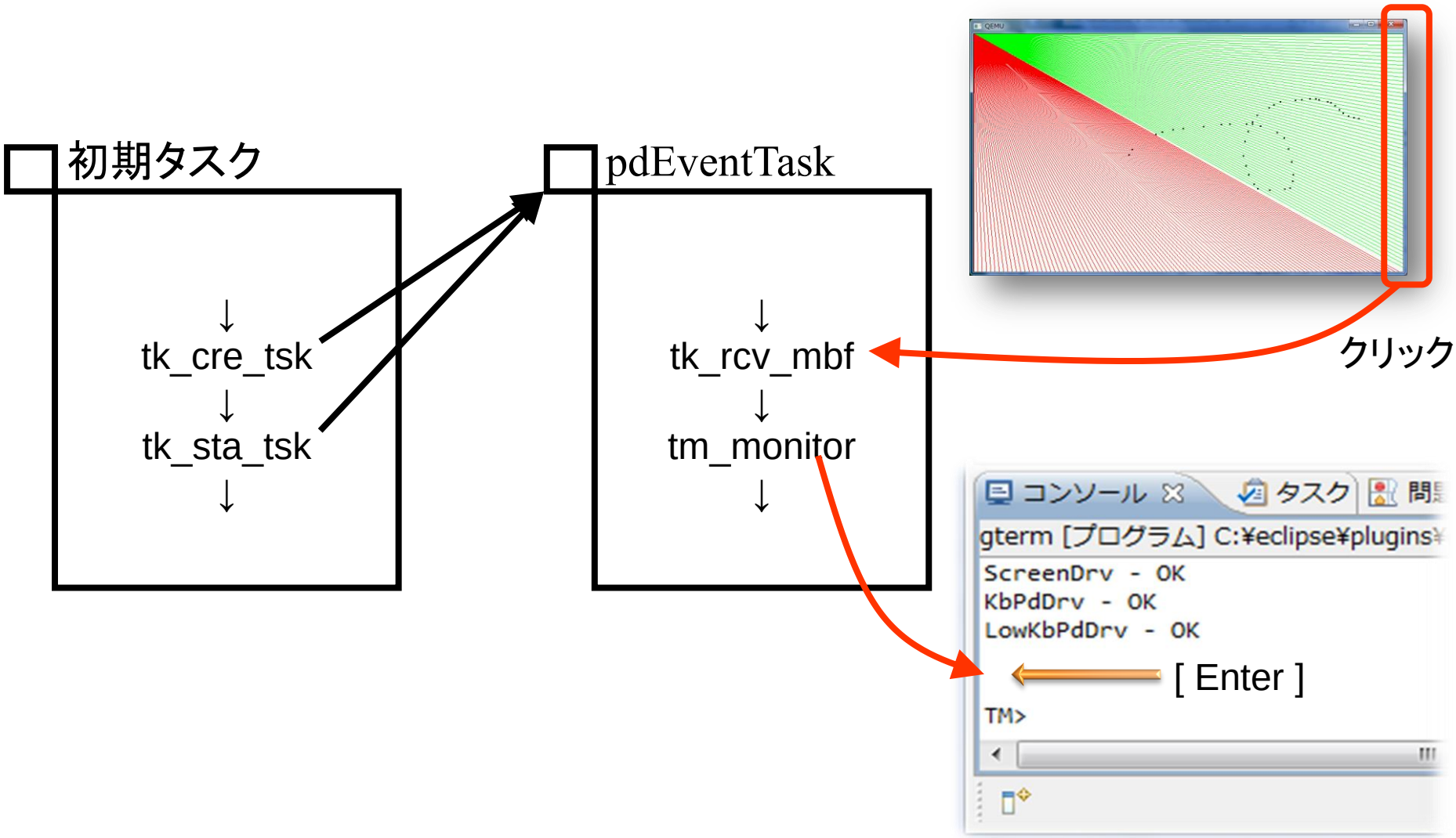
Qemuの設定を変更 (2/2)



タッチパネル用タスクの追加

- 以下のファイルを usermain_drv にコピー
  - usermain.c ， Makefile.usermain
  - **inputDevice**.c ， **inputDevice**.h
- ビルド
  - sysmain/build_drv/tef_em1d を選択
  - [プロジェクト] → [T-Engine Target の Make all]
- ロード
  - sysmain/build_drv/tef_em1d/kernel-ram.sys を選択
  - メニュー → [デバッグ] → [デバッグの構成]
  - [kernel-ram_drv] → [デバッグ]

基本構造


初期タスク
↓
tk_cre_tsk
↓
tk_sta_tsk
↓
pdEventTask
↓
tk_rcv_mbf
↓
tm_monitor
↓
クリック
コンソール
タスク
gterm [プログラム] C:¥eclipse¥plugins¥
ScreenDrv - OK
KbPdDrv - OK
LowKbPdDrv - OK
[Enter]
TM>

ブレーク後の操作

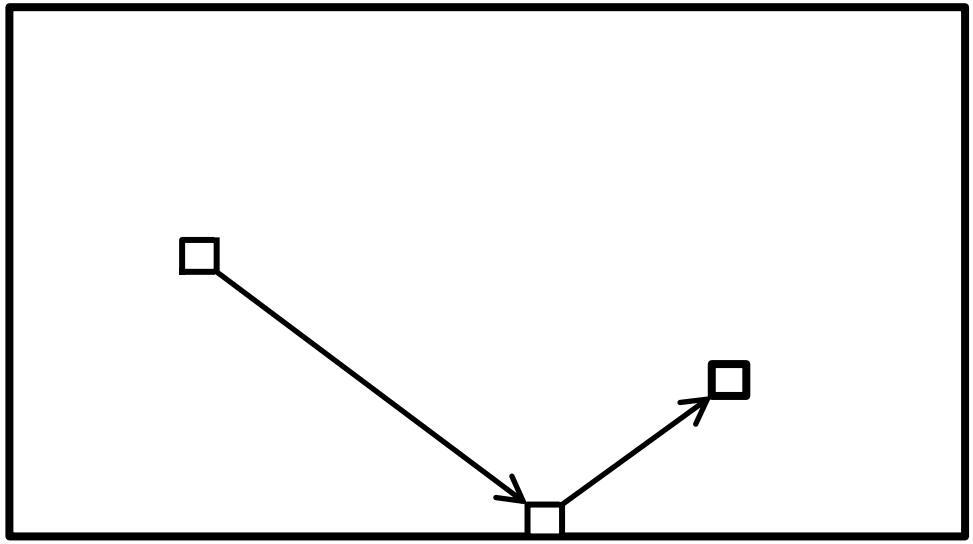
- ブレークポイントを設定
  - 通常の操作でブレークポイントを設定可能
- 再開()

  - 注意
    - ここで追加した機能は正確には「T-Monitorの起動」であり、JTAGデバッガなどによるブレークとは異なる。

タスクによる描画

タスクによる描画

- １つの矩形を四辺で反転させながら移動



- 複数の矩形を四辺で反転させながら移動
- セマフォによる排他制御

タスクの生成(1/2)

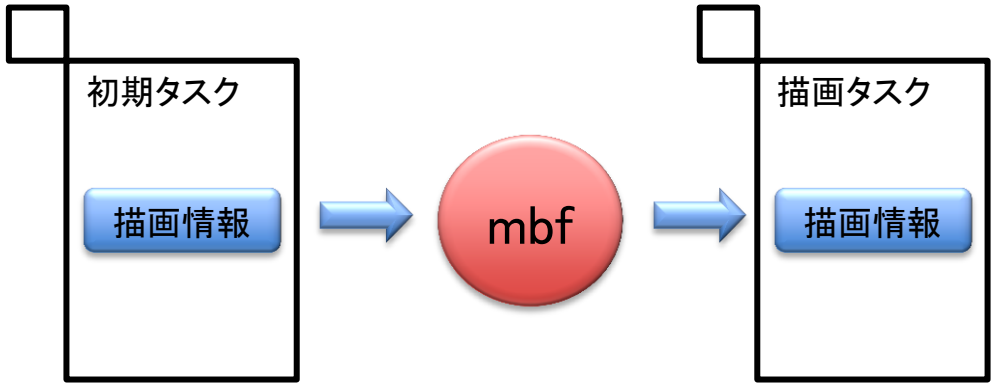
■ usermain_drv/drawTask.c

```
EXPORT ID     launchTask( PRI pri, H ix, H iy, H dx, H dy, INT col, RELTIM dt )
{
        T_CTSK ctsk = { NULL, TA_HLNG|TA_RNG0, (FP)drawTask, 1, 4*1024, };
        DDATA  ddata;
        ID     tid;
        ER     ercd;

        ctsk.itskpri = pri;
        tid = tk_cre_tsk( &ctsk );
        if( tid < 0 ){
                return( (ER)tid );
        }

        ercd = tk_sta_tsk( tid, 0 );
        if( ercd < 0 ){
                return( ercd );
        }
```

H	ix	矩形の初期座標(X)
H	iy	矩形の初期座標(Y)
H	dx	矩形の移動量(X)
H	dy	矩形の移動量(Y)
H	col	矩形の色
H	dt	描画間隔 (msec)

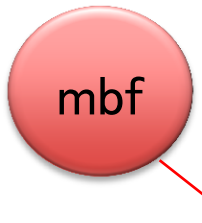
タスクの生成 (2/2)

```
        ddata.ix  = ix;
        ddata.iy  = iy;
        ddata.dx  = dx;
        ddata.dy  = dy;
        ddata.col = col;
        ddata.dt  = dt;

        ercd = tk_snd_mbf( mbfid, &ddata, sizeof(DDATA), 100 );

        return( tid );
}
```



メッセージバッファの生成

```
ID        mbfid;                /* Message Buffer for initialize */

EXPORT    ER        initDrawTaskEnv(void)
{
          T_CMBF    cmbf = { 0, TA_TFIFO, 1024, sizeof(DDATA), };

          mbfid = tk_cre_mbf( &cmbf );

          return( (mbfid < 0)? (ER)mbfid : E_OK );
}
```



exinf	0	未使用
mbfatr	TA_TFIFO	送信待ちタスクのキューイングはFIFO
bufsz	1024	メッセージバッファのサイズ
maxsz	sizeof(DDATA)	メッセージの最大長
dsname[8]		未使用

描画タスク(1/2)

```
void    drawTask( INT stacd, VP *exinf )
{
        DDATA   ddata;              /* Initial parameter */
        H       x;
        H       y;
        ER      ercd;

        ercd = tk_rcv_mbf( mbfid, &ddata, TMO_FEVR );
        if( ercd < 0 ){
                tk_exd_tsk();
        }

        x = ddata.ix;
        y = ddata.iy;
        while(1){

                :

        }
        tk_exd_tsk();
}
```

次ページへ

描画タスク(2/2)

```
while(1){
        T_MARK( x, y, ddata.col );
        tk_dly_tsk( ddata.dt );
        T_MARK( x, y, COL_BLACK );   /* erase */
        x += ddata.dx;
        y += ddata.dy;

        if( x < 0 ){                                          } 左端
                x *= -1;                                      }
                ddata.dx *= -1;                               }
        }else if( x > (screenWidth - RECTSZ) ){               } 右端
                x = (screenWidth - RECTSZ)*2 - x;             }
                ddata.dx *= -1;                               }
        }                                                     } 上端
        if( y < 0 ){                                          }
                y *= -1;                                      }
                ddata.dy *= -1;                               } 下端
        }else if( y > (screenHeight - RECTSZ) ){              }
                y = (screenHeight - RECTSZ)*2 - y;            }
                ddata.dy *= -1;                               }
        }
}
```

usermain()とMakefile.usermain

■ usermain_drv/usermain.c

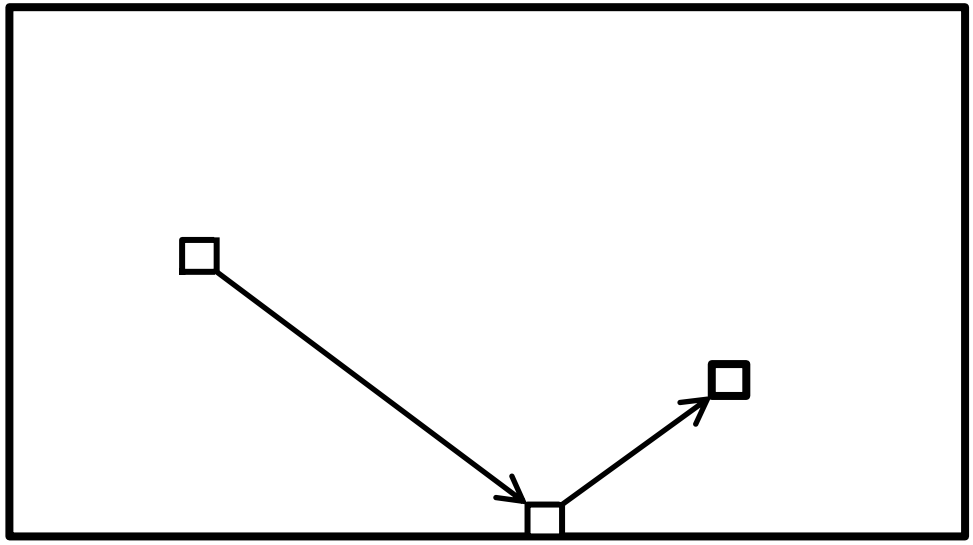
```
EXPORT   INT      usermain( void )
{
            :
        /*----------------------------------------------------*/
        initDisplay();
        initInputDevice();
        initDrawTaskEnv();
        launchTask( 5, 100, 200,  1,  6, COL_RED,   10 );
        tk_slp_tsk(TMO_FEVR);
        /*----------------------------------------------------*/
            :
```

■ usermain_drv/Makefile.usermain

```
    :
# source files
SRC += usermain.c
SRC += display.c inputDevice.c drawTask.c
```

タスクによる描画

- １つの矩形を四辺で反転させながら移動



- 複数の矩形を四辺で反転させながら移動
- セマフォによる排他制御

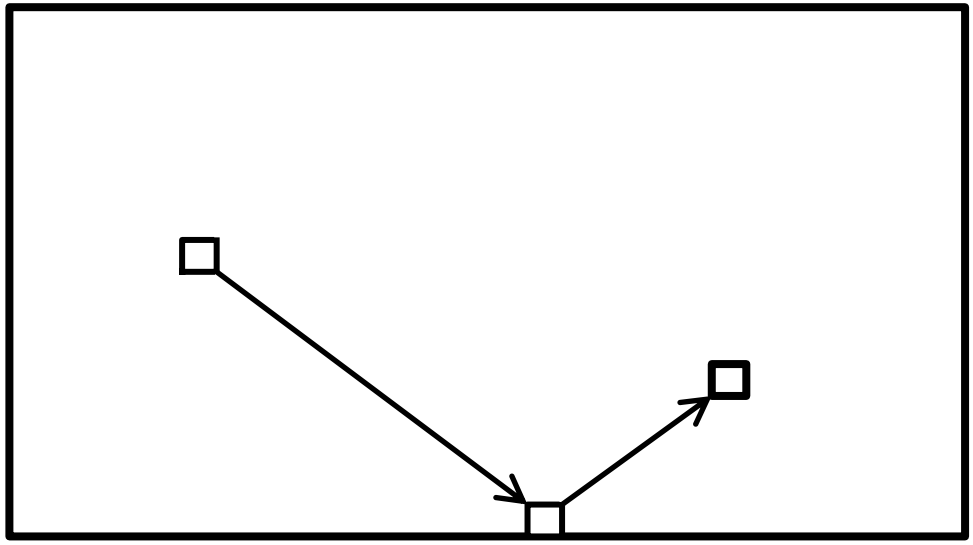
usermain()

■ usermain_drv/usermain.c

```
EXPORT   INT      usermain( void )
{
            :
        tm_putstring(ercd >= E_OK ? "LowKbPdDrv - OK¥n" : "LowKbPdDrv - ERR¥n");

        /*-------------------------------------------------------*/
        initDisplay();
        initInputDevice();
        initDrawTaskEnv();
        launchTask( 5, 100, 200,  1,  6, COL_RED,    10 );
        launchTask( 5, 100, 200, -3,  4, COL_GREEN,  10 );
        launchTask( 5, 100, 200,  7, -5, COL_BLUE,   10 );
        launchTask( 5, 100, 200, -5, -3, COL_WHITE,  10 );
        tk_slp_tsk(TMO_FEVR);
        /*-------------------------------------------------------*/

        tm_putstring((UB*)"Push any key to shutdown the T-Kernel.¥n");
            :
```

タスクによる描画

- １つの矩形を四辺で反転させながら移動



- 複数の矩形を四辺で反転させながら移動
- セマフォによる排他制御

セマフォの生成を追加

■ usermain_drv/drawTask.c

```
ID      mbfid;              /* Message Buffer for initialize */
ID      semid;              /* Semaphore ID for exclusive control test */

EXPORT  ER      initDrawTaskEnv(void)
{
        T_CMBF  cmbf = { NULL, TA_TFIFO, 1024, sizeof(DDATA), };
        T_CSEM  csem = { NULL, TA_TFIFO|TA_FIRST, 1, 1, };

        semid = tk_cre_sem( &csem );
        if( semid < 0 ){
                return( (ER)semid );
        }

        mbfid = tk_cre_mbf( &cmbf );
        return( (mbfid < 0)? (ER)mbfid : E_OK );
}
```

描画タスク(1/2)

```
#define EX_AREA         ((screenWidth/2)-RECTSZ)
#define RECTSZ                      (10)
#define T_MARK(x,y,c)  drawRect((x),(y),(x)+RECTSZ,(y)+RECTSZ,(c),RECT_FILL)
void        drawTask( INT stacd, VP *exinf )
{
            DDATA       ddata;                  /* Initial parameter */
            H           x;
            H           y;
            BOOL        isExArea;
            ER          ercd;

            ercd = tk_rcv_mbf( mbfid, &ddata, TMO_FEVR );
            if( ercd < 0 ){
                        tk_exd_tsk();
            }

            x = ddata.ix;
            y = ddata.iy;
            isExArea = FALSE;
            while(1){

                        :


            }
            tk_exd_tsk();
}
```

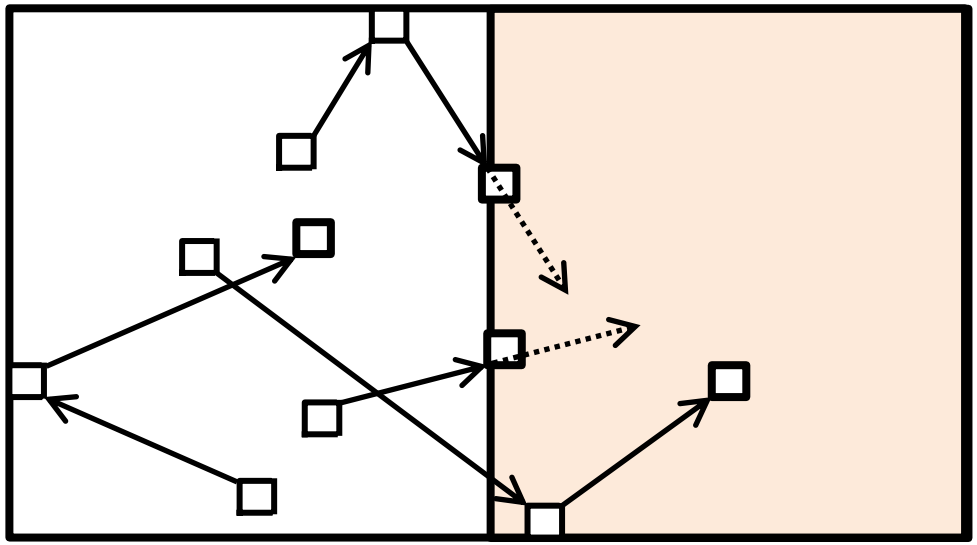
➡ 次ページへ

描画タスク(2/2)

```
while(1){
        /* 略 */
        if( x < 0 ){
                x *= -1;
                ddata.dx *= -1;
        }else if( x > (screenWidth - RECTSZ) ){
                x = (screenWidth - RECTSZ)*2 - x;
                ddata.dx *= -1;
        }else if( (isExArea != TRUE)&&(x >= EX_AREA) ){                    /* Enter */
                T_MARK(EX_AREA,y+(ddata.dy*(EX_AREA-x))/ddata.dx,ddata.col);
                tk_wai_sem( semid, 1, TMO_FEVR );
                T_MARK(EX_AREA,y+(ddata.dy*(EX_AREA-x))/ddata.dx,COL_BLACK);
                isExArea = TRUE;
        }else if( (isExArea == TRUE)&&(x < EX_AREA) ){                     /* Exit */
                tk_sig_sem( semid, 1 );
                isExArea = FALSE;
        }
        if( y < 0 ){
                y *= -1;
                ddata.dy *= -1;
        }else if( y > (screenHeight - RECTSZ) ){
                y = (screenHeight - RECTSZ)*2 - y;
                ddata.dy *= -1;
        }
}
```

セマフォによる排他制御

- 右半分の領域に入れる矩形は1つのみ



- 他のタスクは先行する矩形が出るまで待たされる。
- セマフォはTA_TFIFO属性
  → 先に待ち状態になったタスクが優先的に獲得

演習問題

- 右半分の領域に矩形が2つ入れるようにするには？

T-Kernelの中に入る

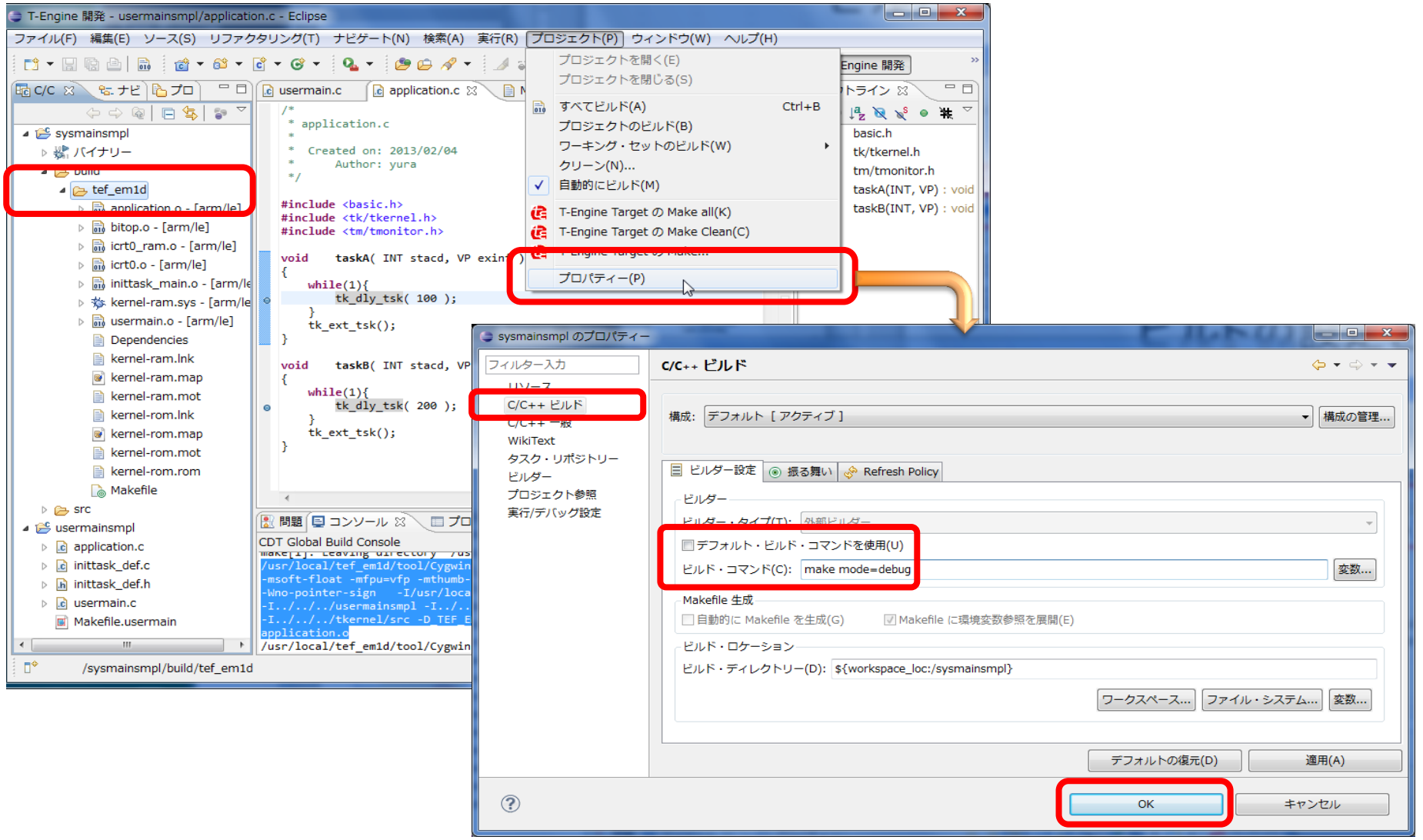
T-Kernelの中に入る

- パラメータなどのエラーの原因解明
  - エラーの原因をピンポイントで確認　→　効率的にデバッグ
    - 例1) オブジェクト生成時の属性指定間違い
    - 例2) メモリサイズの指定間違い (指定可能範囲外など)
- チューニング
  - カーネルの動作を考慮した効率的なアプリケーションの設計
  - カーネル自体の実装の見直し
  - 対象となるシステムでは使用しない機能の削除など

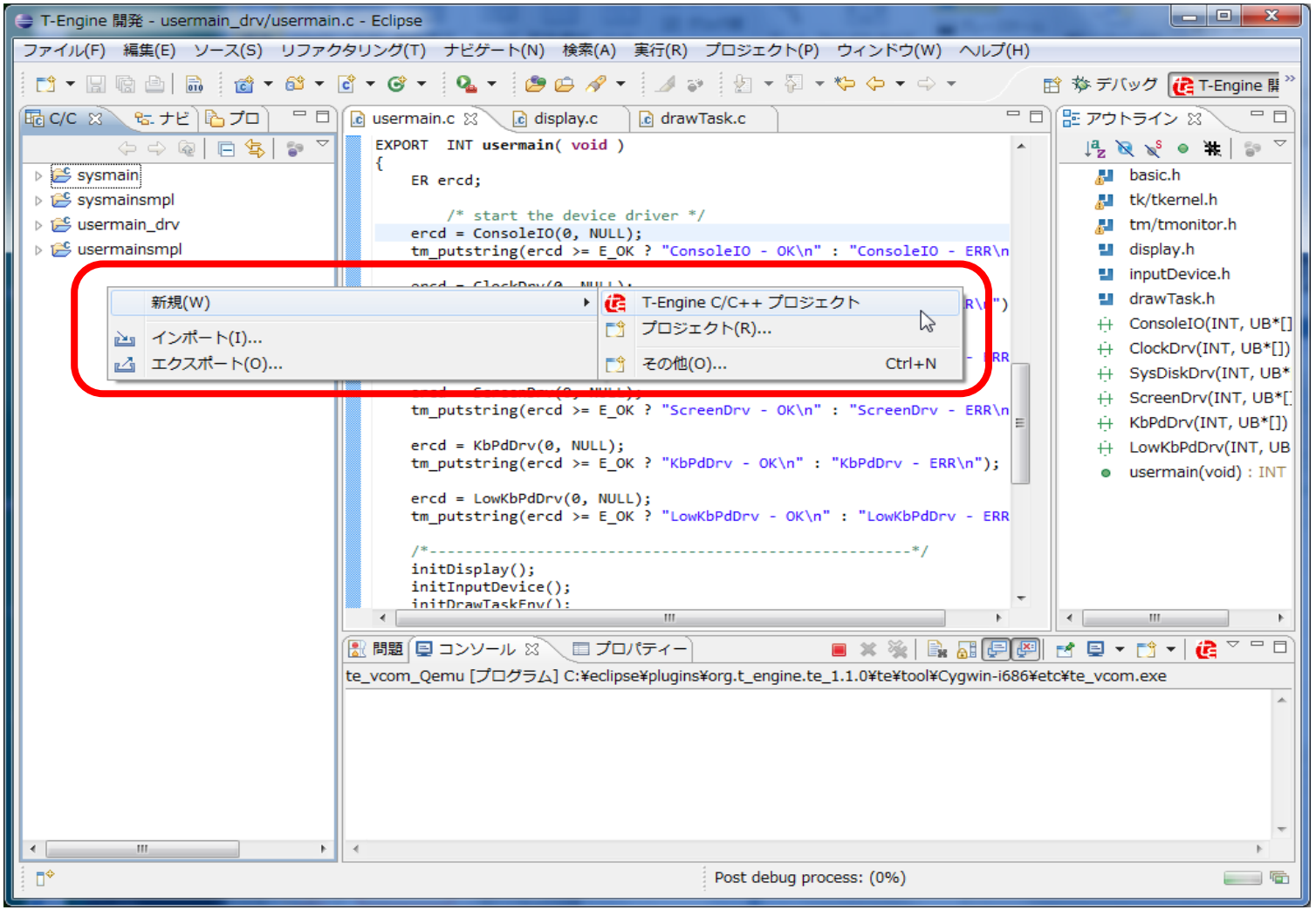
T-Kernelのソースを表示する準備

- デバッグモードの設定（設定済）
  - make mode=debug
    - デバッグオプション -g が追加される。
- プロジェクトの追加
  - tkernel_source/kernel/tkernel
- ブレークポイントの設定
- ロードして実行
  - ソース・ルックアップ・パスの設定

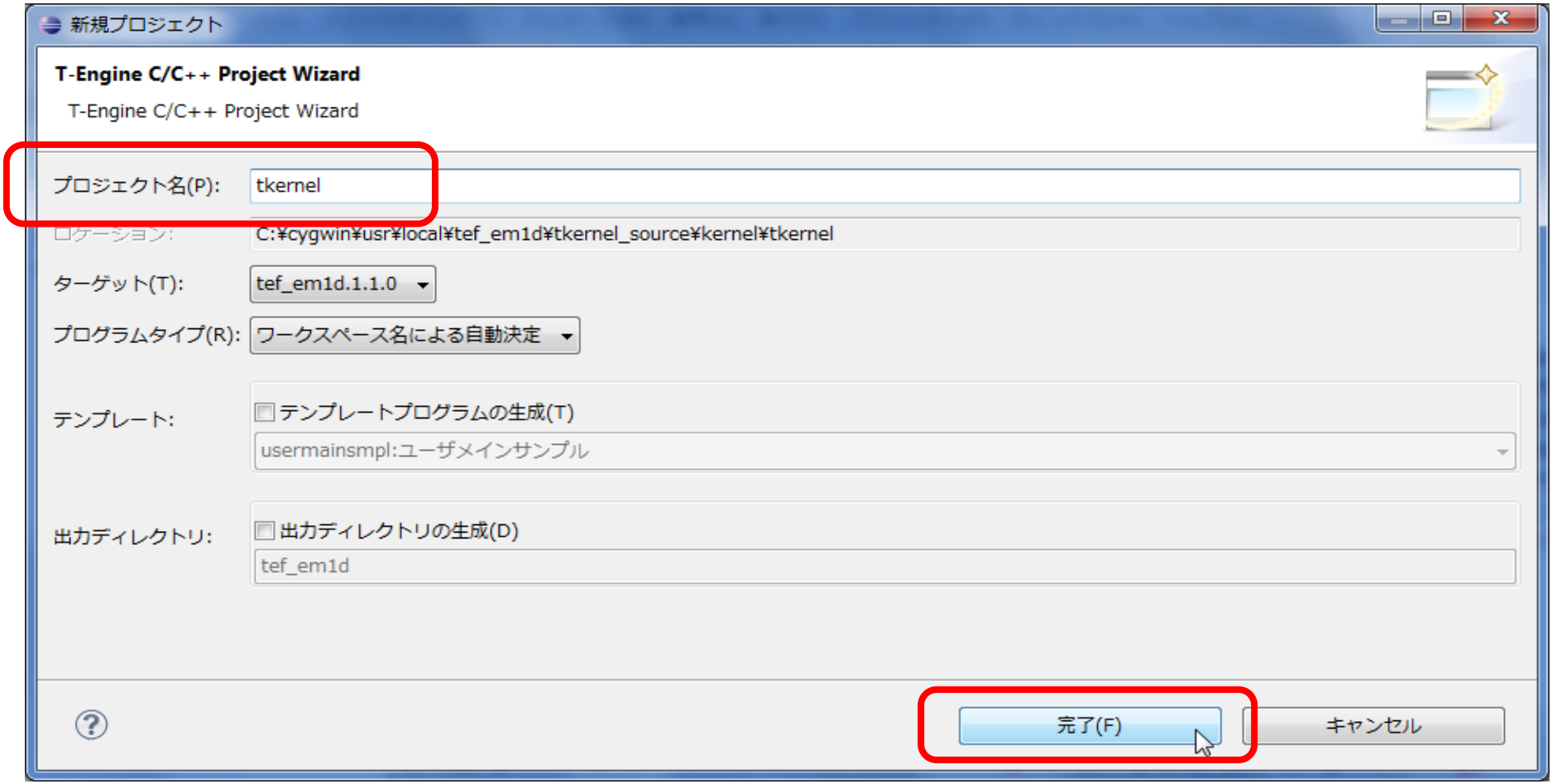
デバッグモードの設定


T-Engine 開発 - usermainsmpl/application.c - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) 実行(R) プロジェクト(P) ウィンドウ(W) ヘルプ(H)
プロジェクトを開く(E)
プロジェクトを閉じる(S)
すべてビルド(A) Ctrl+B
プロジェクトのビルド(B)
ワーキング・セットのビルド(W)
クリーン(N)...
自動的にビルド(M)
T-Engine Target の Make all(K)
T-Engine Target の Make Clean(C)
プロパティー(P)
sysmainsmpl
usermainsmpl
tef_em1d
CDT Global Build Console
sysmainsmpl のプロパティー
C/C++ ビルド
構成: デフォルト [アクティブ]
構成の管理...
ビルダー設定
振る舞い
Refresh Policy
デフォルト・ビルド・コマンドを使用(U)
ビルド・コマンド(C): make mode=debug
変数...
Makefile 生成
ビルド・ロケーション
ビルド・ディレクトリー(D): ${workspace_loc:/sysmainsmpl}
ワークスペース...
ファイル・システム...
デフォルトの復元(D)
適用(A)
OK
キャンセル

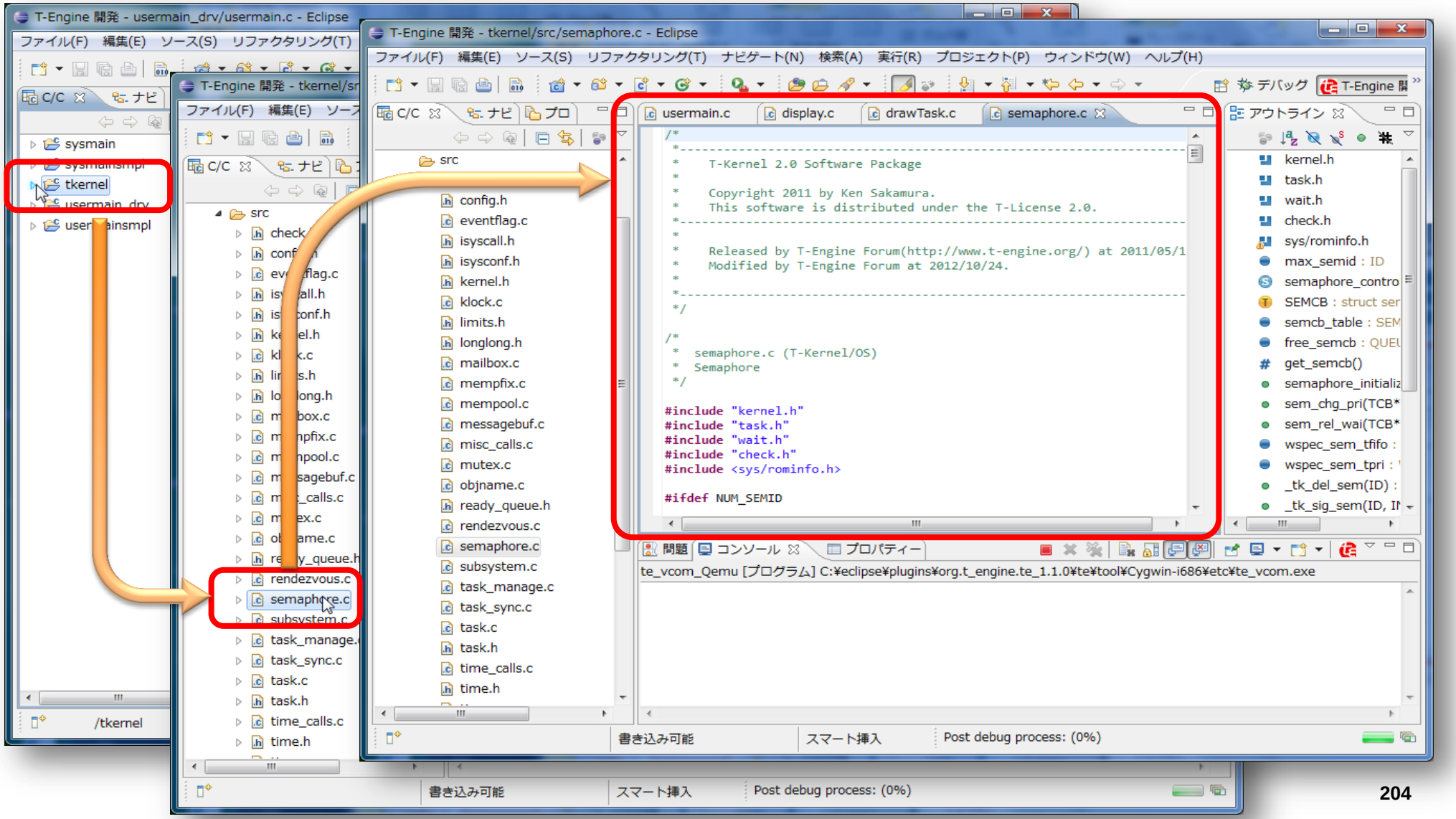
プロジェクトの追加 (1/2)



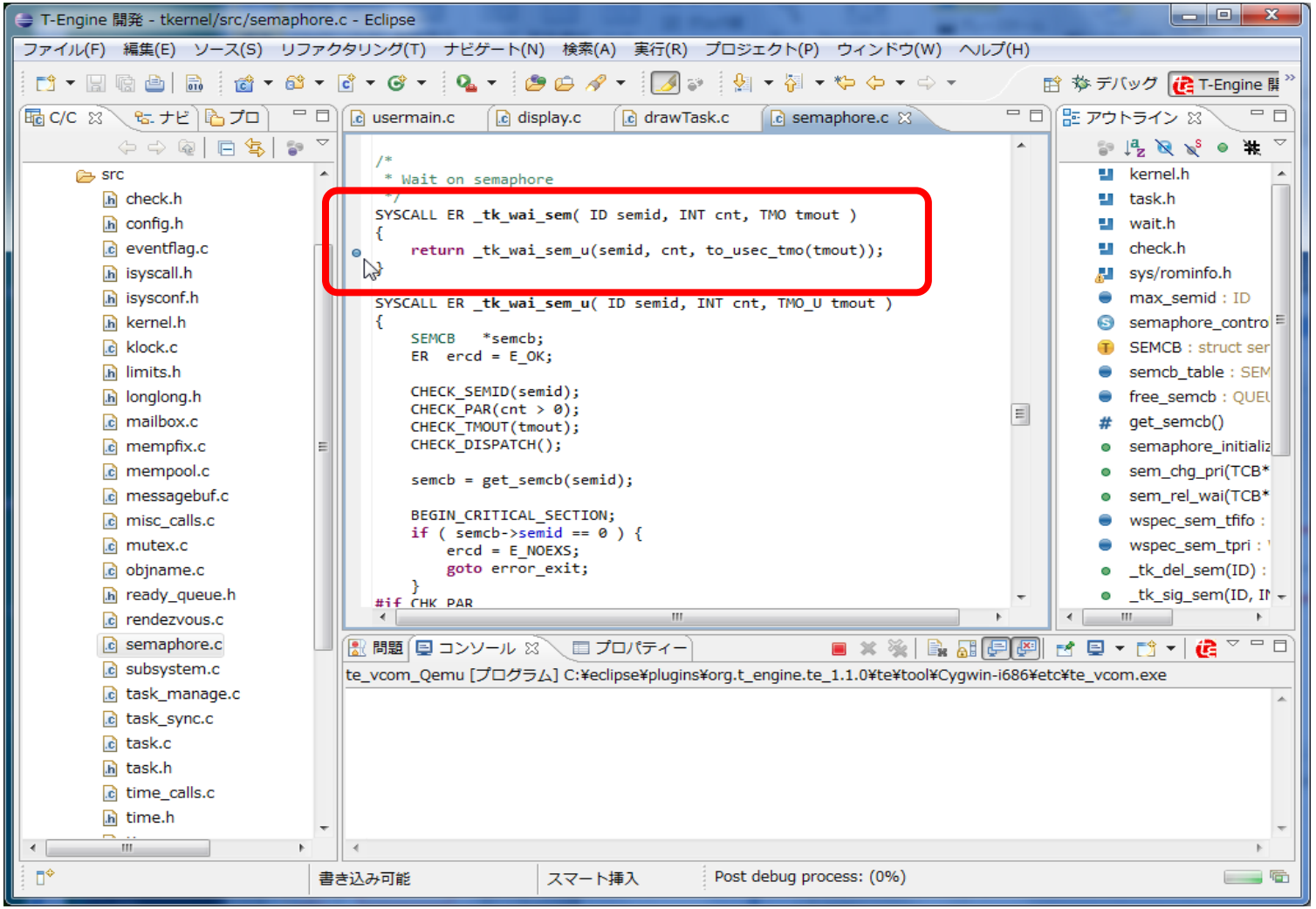
プロジェクトの追加 (2/2)



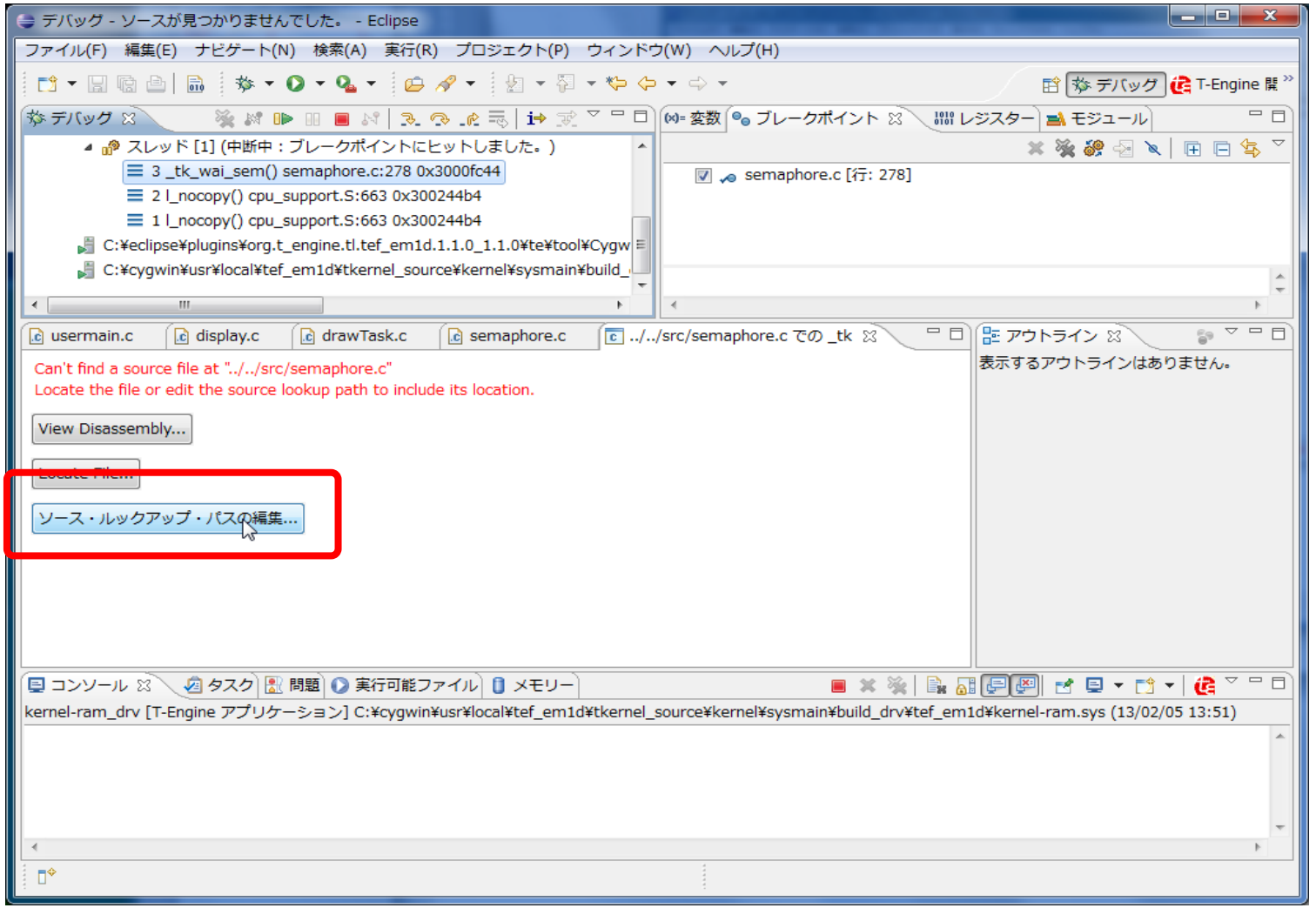
ブレークポイントの設定 (1/2)



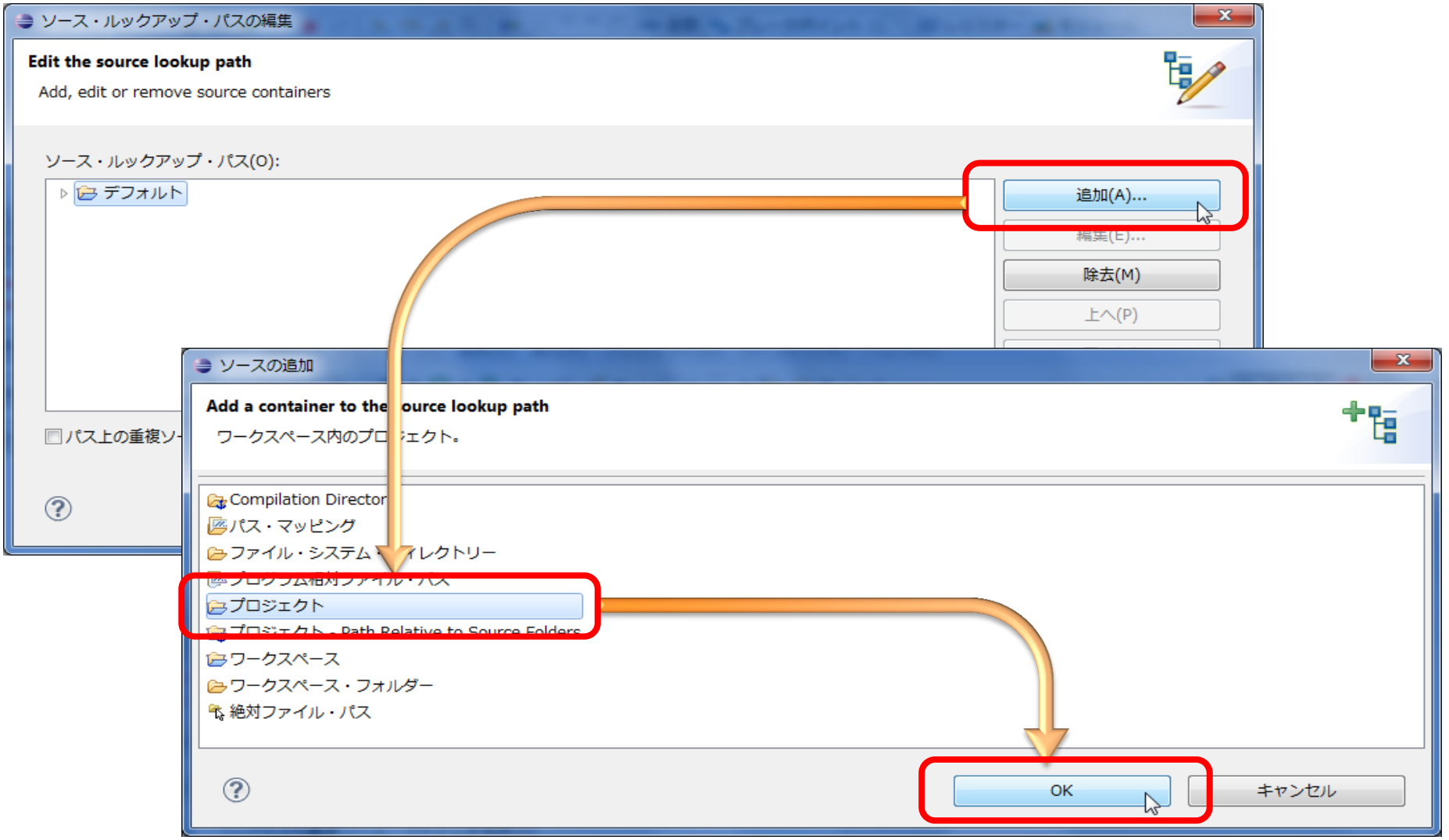
ブレークポイントの設定 (2/2)



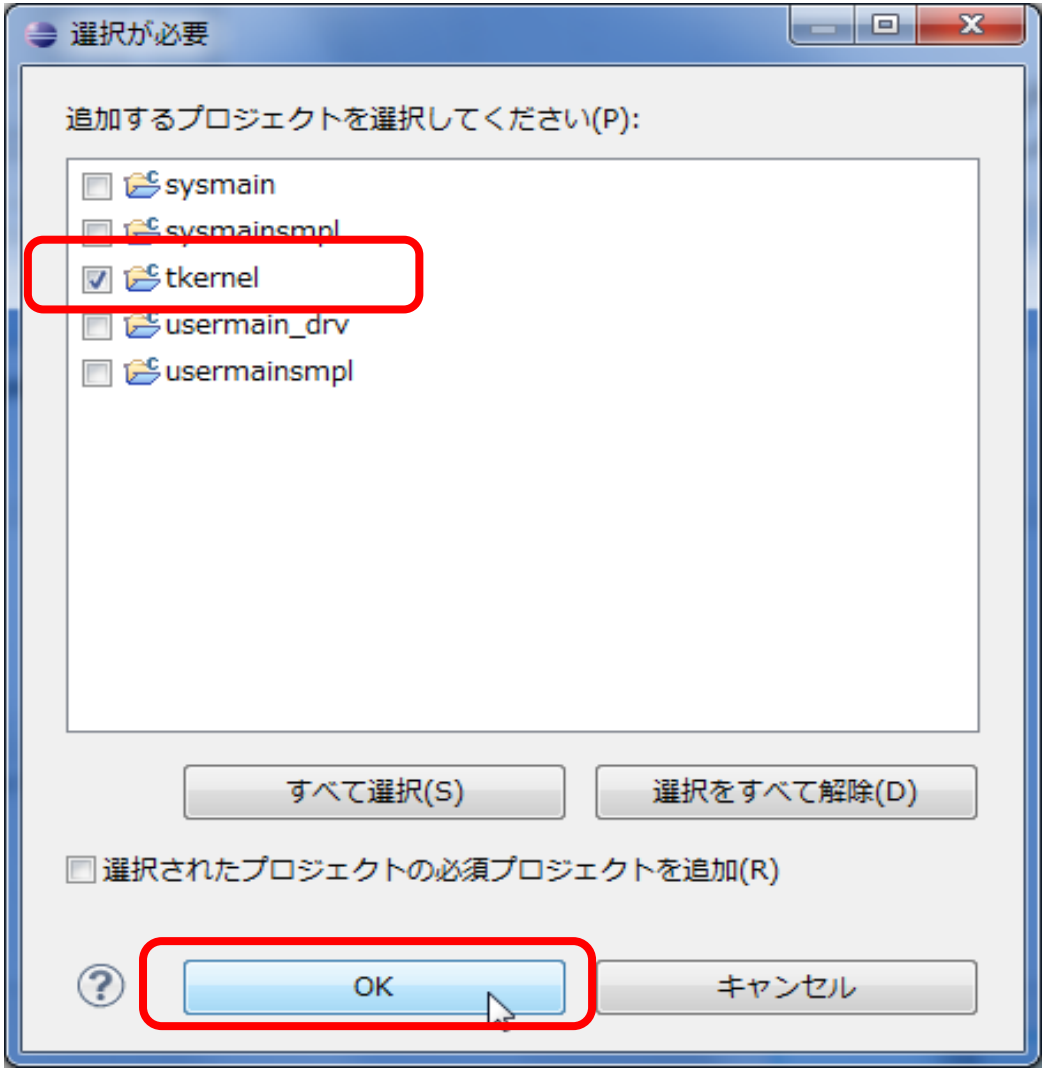
ソース・ルックアップ・パスの設定 (1/5)



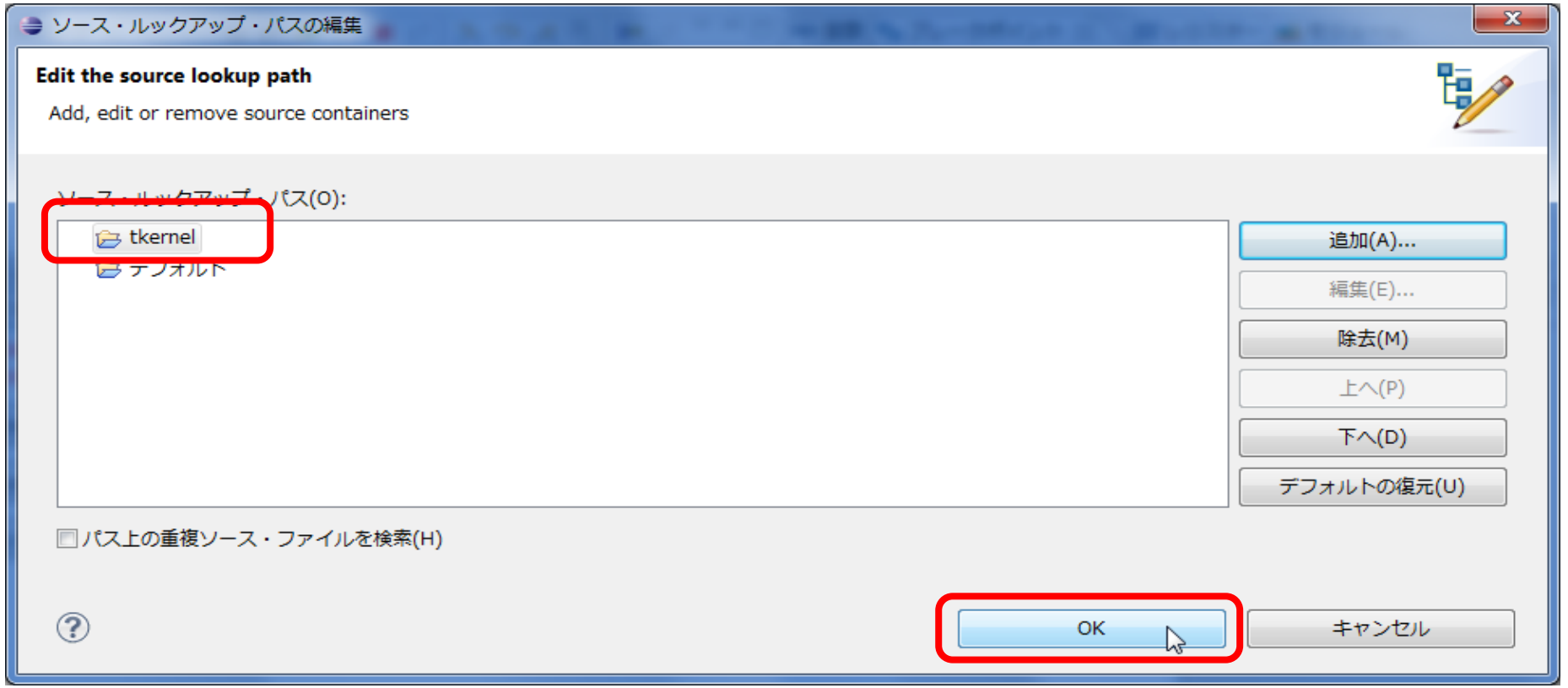
ソース・ルックアップ・パスの設定 (2/5)



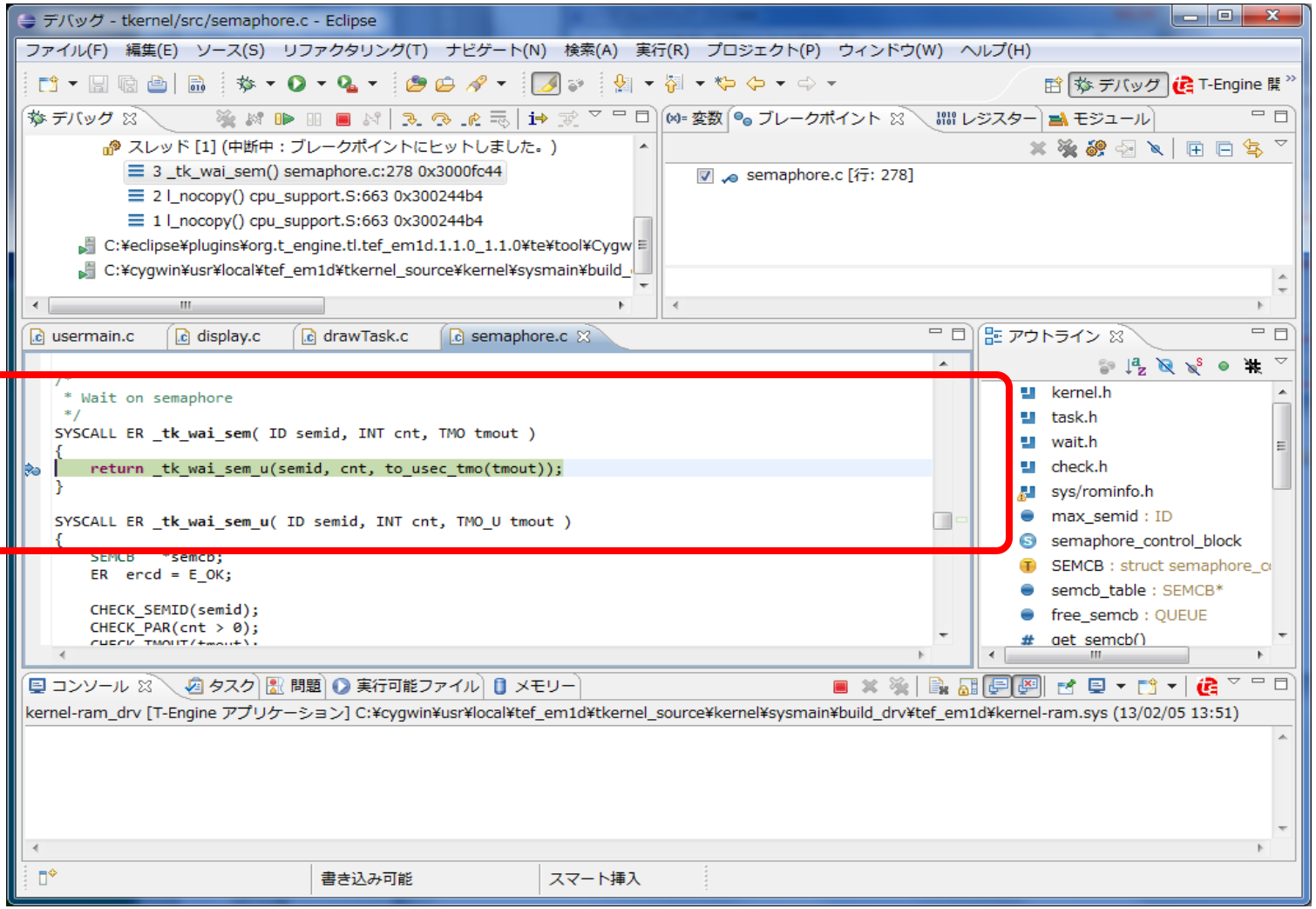
ソース・ルックアップ・パスの設定 (3/5)



ソース・ルックアップ・パスの設定 (3/4)



ソース・ルックアップ・パスの設定 (3/3)



まとめ

まとめ (1/2)

- T-Kernel 2.0 Software Package
  - 入手方法
  - パッケージ内容
- Installing T-Kernel 2.0
  - 開発環境のインストール
  - エミュレータとの接続
- アプリケーションの追加
  - usermain.cにアプリケーションタスクを追加
  - アプリケーションを別ファイルで追加

まとめ (2/2)

- デバイスドライバの利用方法
  - 液晶画面(LCD)を表示
    - ドライバ版のビルド
    - LCDの表示
    - 簡易ブレーク機能の追加
    - タスクによる矩形表示
    - セマフォによる排他制御
- T-Kernelの中に入る
  - ソース・ルックアップ・パスの追加

質疑応答

More Information

http://www.t-engine.org/

Appendix

Appendix

Appendix. A

16進数でのエラーコード一覧

16進数でのエラーコード一覧

E_OK	0x00000000	(0)	正常終了
E_SYS	0xfffb0000	(-5)	システムエラー
E_NOCOP	0xfffa0000	(-6)	コプロセッサ使用不可
E_NOSPT	0xfff70000	(-9)	未サポート機能
E_RSFN	0xfff60000	(-10)	予約機能コード番号
E_RSATR	0xfff50000	(-11)	予約属性
E_PAR	0xffef0000	(-17)	パラメータエラー
E_ID	0xffee0000	(-18)	不正ID番号
E_CTX	0xffe70000	(-25)	コンテキストエラー
E_MACV	0xffcc0000	(-25)	メモリアクセス不能,メモリアクセス権違反
E_OACV	0xffe50000	(-27)	オブジェクトアクセス権違反
E_ILUSE	0xffe40000	(-28)	システムコール不正使用
E_NOMEM	0xffdf0000	(-33)	メモリ不足
E_LIMIT	0xffde0000	(-34)	システムの制限を超過
E_OBJ	0xffd70000	(-41)	オブジェクトの状態が不正
E_NOEXS	0xffd60000	(-42)	オブジェクトが存在していない
E_QOVR	0xffd50000	(-43)	キューイングまたはネストのオーバーフロー
E_RLWAI	0xffcf0000	(-49)	待ち状態強制解除
E_TMOUT	0xffce0000	(-50)	ポーリング失敗またはタイムアウト
E_DLT	0xffcd0000	(-51)	待ちオブジェクトが削除された
E_DISWAI	0xffcc0000	(-52)	待ち禁止による待ち解除
E_IO	0xffc70000	(-57)	入出力エラー
E_NOMDA	0xffc60000	(-58)	メディアがない
E_BUSY	0xffbf0000	(-65)	ビジー状態
E_ABORT	0xffbe0000	(-66)	中止した
E_RONLY	0xffbd0000	(-67)	書込み禁止

Appendix. B
Troubleshooting

The program is not being run.（1/5）

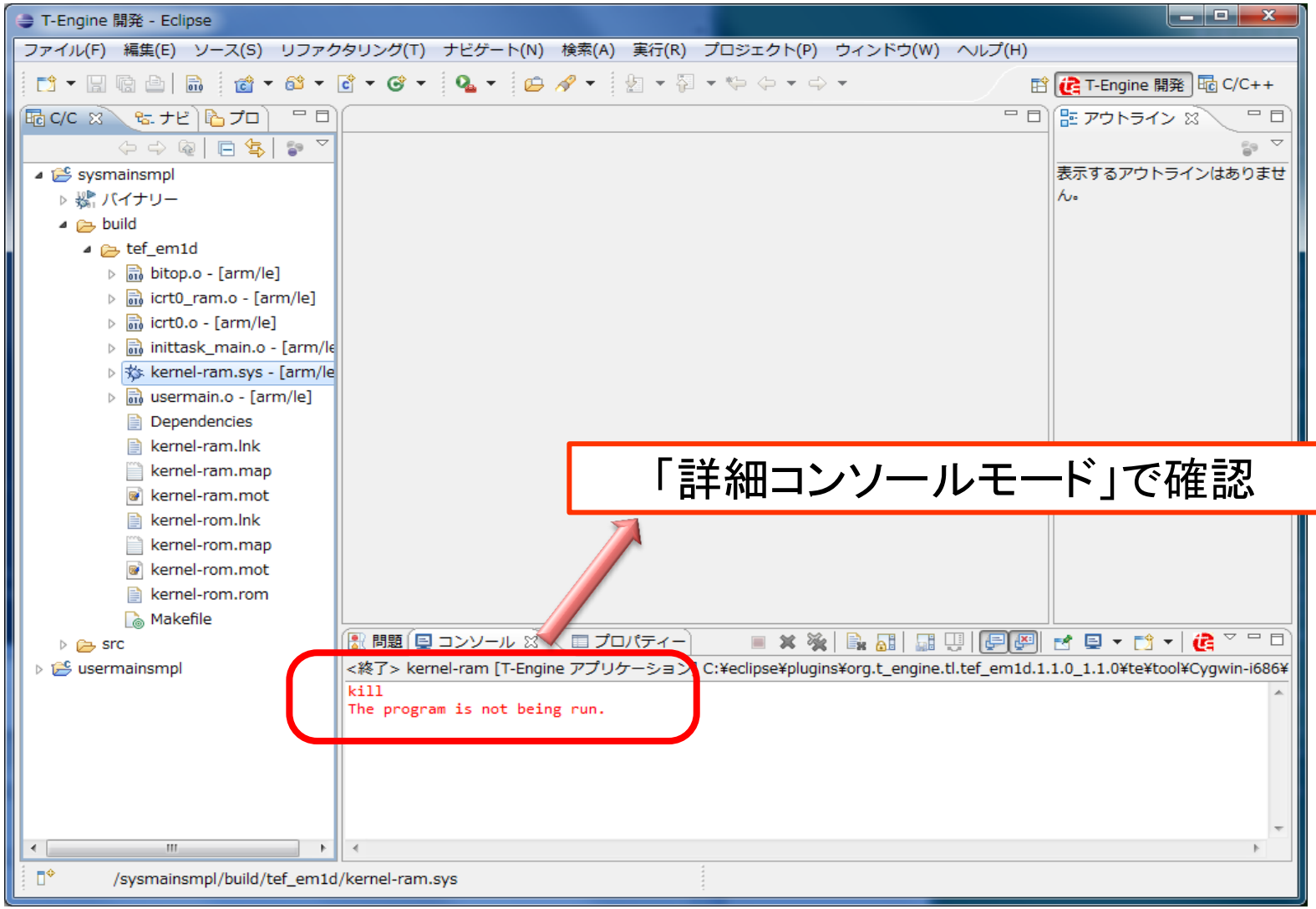
- te_vcomの実行時に以下が表示されて、te_vcomが起動しない。
- または、プログラムロード後に以下が表示されて、デバッガモードに移行しない。

Kill

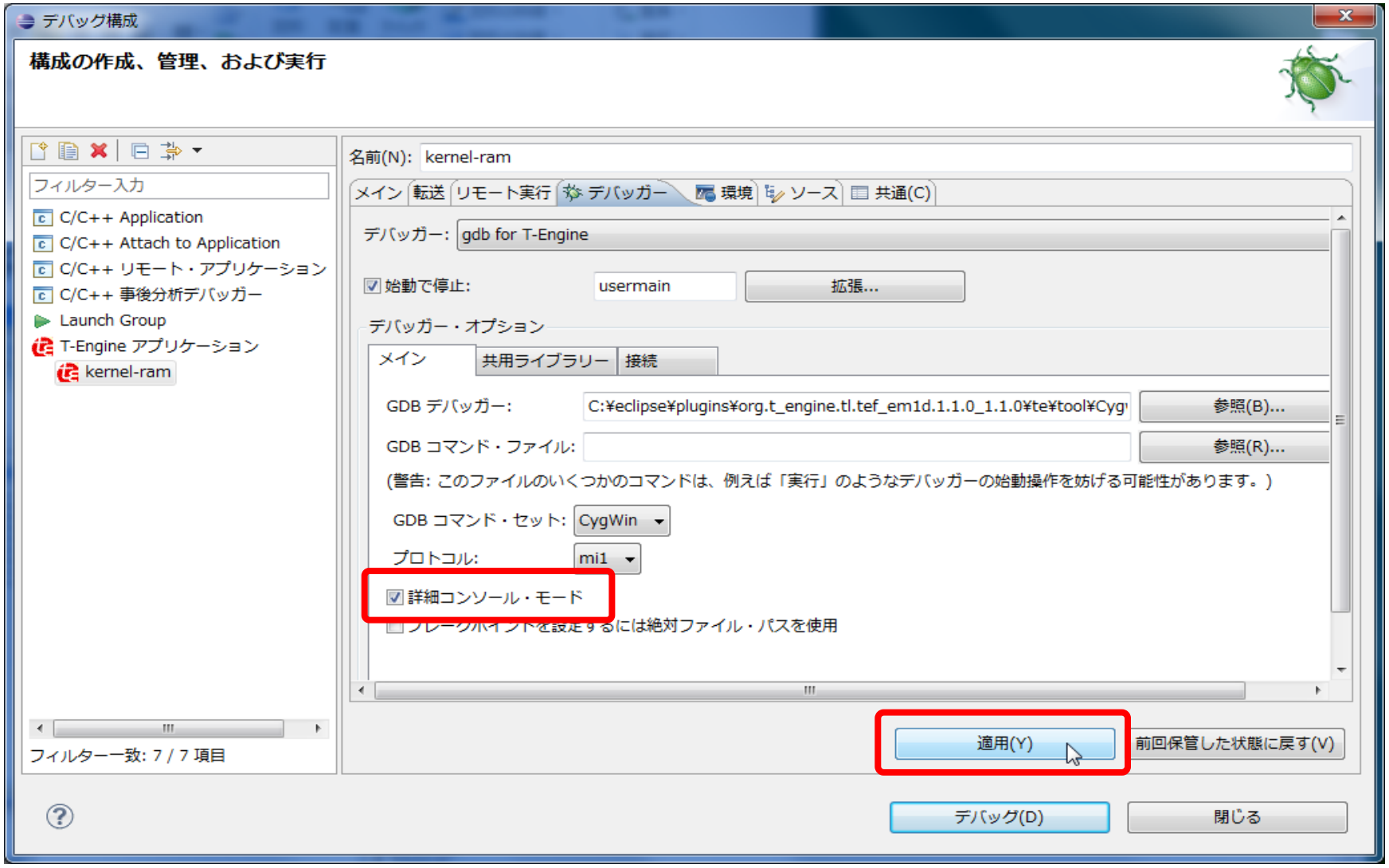
The program is not being run.

- 詳細コンソールモードを有効にして確認してみる。

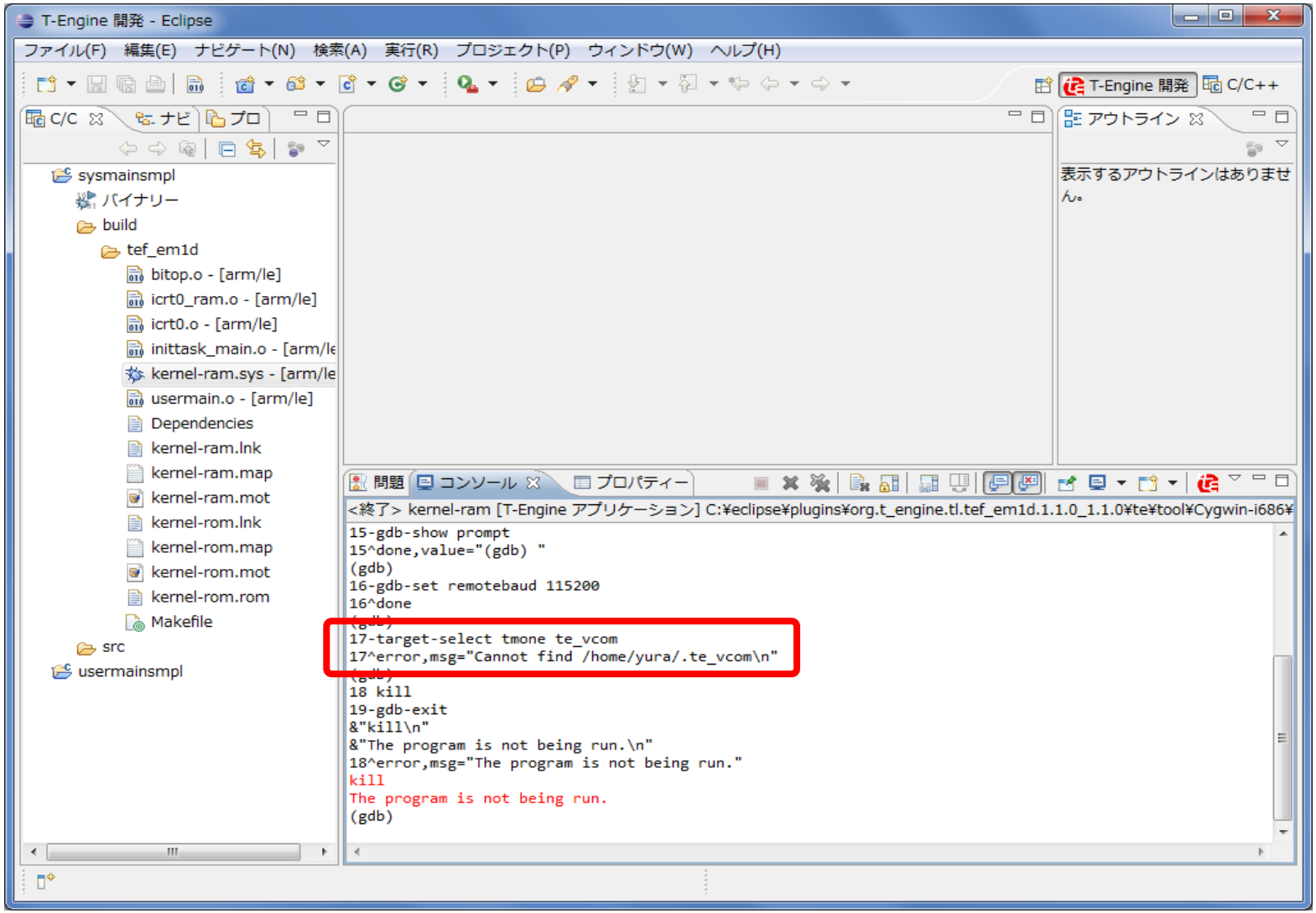
The program is not being run.（2/5）



The program is not being run.（3/5）



The program is not being run.（4/5）



The program is not being run.（5/5）

- /home/[ユーザ名]/.te_vcom がない !?

- 以下の内容の設定ファイルを用意する。

  /dev/tty0

  /dev/tty1

- 念のためQemu_DBGを再起動してから
  ロードすると成功する…はず。
  - Qemuを再起動しないと動作しないこともある。

それでも、アプリケーションが起動しない？

- 相変わらず…

  Kill

  The program is not being run.

- 不要なプロセスが邪魔をしている(ことがある)。
  → タスクマネージャで不要プロセスを削除
  1. Eclipseとエミュレータを一旦終了
  2. zombie processを削除
  3. Eclipseとエミュレータを再度起動

プロセスが残っている

- Eclipseの動作がおかしい場合、以下のプロセスが残っていないか確認 (残っていたら削除)
  - gterm.exe
  - te_vcom.exe
  - arm_2-unknown-tmonitor-gdb.exe

- 確認、削除手順
  1. Eclipseとエミュレータを終了
  2. タスクマネージャを起動 (CTRL+ALT+DEL)
  3. プロセスタブを選択
  4. 上記のプロセスが残っていたら[プロセスの終了]で削除

Appendix. C

リアルタイムOS講座
「RTOS入門編」

リアルタイムOS講座「RTOS入門編」

- T-Kernel 2.0を利用したRTOS入門講座
  - T-Kernel 2.0で動作するサンプルコードを多数掲載
  - APS ACADEMYとの連携企画
- T-Engineフォーラム版

  → http://www.t-engine.org/ja/sympo/rtos
  - より詳細な解説や追加のサンプルがある。
- APS ACADEMY版

  → http://www.aps-web.jp/academy/rtos/01/a.html
  - ARMパートナー事例集 (広報誌)
  - 英語版、中国語(簡体)版がある。

用語集

用語集 (1/3)

- Cygwin
  - Windowsで動作するUnixライクな環境の一種
  - T-Kernelの開発環境として使用している。
- Eclipse
  - オープンな統合開発環境(IDE)
- GDB (The GNU Project Debugger)
  - オープンなデバッガ
  - T-Kernel 2.0の開発環境の一部として使用されている。
- Makefile.common
  - T-Kernelのメイク(ビルド)実行時に読み込まれるMakefileの一種
- Qemu
  - オープンソースのPCエミュレータ
- T-Engineリファレンスボード
  - T-Kernel 2.0が動作する標準のターゲットボード

用語集 (2/3)

- T-Monitor
  - T-Kernelを動作させるためにターゲットボードに組み込まれる基本モニタ
  - 主にハードウェアの初期化やシステムの起動、例外、割込みのハンドリング、基本的なデバッグ機能を提供するプログラム
  - ハードウェアの電源投入（システムリセット）時にまずT-Monitor が起動される。
  - T-Monitorが必要なハードウェアの初期化を行った後、T-Kernel を起動する。
- gterm
  - Eclipseから起動するターミナルソフト (通信ソフト)
- icrt0.S, icrt0_ram.S
  - T-Kernelを含めたロードモジュールの初期化処理部のソースコード
- kernel-ram.map
  - ロードモジュール(kernel-ram.sys)に対応するmapファイル
- kernel-ram.sys
  - T-Kernelのロードモジュール(ELF)
- te_vcom
  - Eclipseとターゲットボード(エミュレータ)を接続するための仮想通信ポート

用語集 (3/3)

- ucode　→ http://www.uidcenter.org/ja/learning-about-ucode
  - 現実世界のさまざまな「モノ」や「場所」などを識別するための固有識別番号
  - T-Kernel 2.0の管理番号として使用されている。
- usermain関数
  - T-Kernelの初期タスクから呼び出される関数
- usermain.c
  - usermain関数が含まれているソースファイル
- ソース・ルックアップ・パス
  - Eclipseがソースコードを表示するために参照するパスの一覧
- デバッグモード
  - デバッグ用のT-Kernelをビルドするモード
  - makeコマンドを以下のパラメータ付きで入力することで指定する。
    make mode=debug
- パースペクティブ
  - Eclipseのワークベンチ内のビューのレイアウト
  - ビュー、メニューバー、ツールバーなどを開発目的に合わせて定義したもの

【実習】T-Kernel 2.0入門テキスト「エミュレータを使った実習と解説」

著者　T-Engine Forum