第3版

画像処理ユニット　NVP-Ax430シリーズ

ソフトウェア開発キット

NVP-Ax430SDK

チュートリアル

maxell
マクセルフロンティア株式会社

はじめに

このたびは、NVP-Ax430シリーズの画像処理ユニット(NVP-Ax430CL/430ACL/435CL/435FCL)および ソフトウェア開発キット NVP-Ax430SDKをお買い上げいただきまして、誠にありがとうございます。

本マニュアルはNVP-Ax430シリーズの画像処理ユニットを使用したアプリケーション作成のためのツールのダウンロード、インストール、操作方法を記載しています。また、別紙「ユーザーズマニュアル」はNVP-Ax430SDKで提供する、仕様、操作方法、使用上の注意事項などについて、別紙「コマンドリファレンス」は、NVP-Ax430SDKで提供するAPIについての仕様、注意事項などについて詳細を記載しています。正しくご使用いただくために、各章の熟読をお勧めいたします。

別紙「ハードウェアマニュアル」にNVP-Ax430シリーズのハードウェア仕様、設置方法、注意事項などを記載しています。取扱い、設置方法を誤ると重大事故の可能性がありますので、ご一読をお願い致します。

ご注意

- 本アプリケーションの操作を行う前に、本マニュアルの記載内容をよく読み、書かれている指示や注意を十分理解してください。誤った操作によりシステムの故障の原因となる場合がありますので十分ご注意ください。

- お客様の誤った操作に起因する、事故発生や損害につきましては、弊社は責任を負いかねますのでご了承ください。

- 弊社提供のハードウェアおよびソフトウェアを無断で改造しないでください。この場合の品質および安全につきましては、弊社は責任を負いかねますのでご了承ください。

- NVP-Ax430シリーズに搭載されているLinuxは、32ビットのLinuxシステムであり、管理できる日付と時刻の範囲は、UTC（協定世界時）時刻で1970年1月1日0時00分00秒～2038年1月19日3時14分7秒までとなります。それ以外の日付と時刻の管理が必要なアプリケーションには対応できませんのでご了承ください。

- NVP-Ax430シリーズに搭載されているLinuxは、リアルタイムOSではございません。お客様のアプリケーションで使用されるユースケースにて性能評価を実施頂き、実製品への適用可能性を十分検討頂くようお願い致します。

- 本マニュアルの内容について予告なく変更する場合があります。

略語および略称の説明

略語/略称	英語名	日本語名
NVP	NVP-Ax430CL/ACL , -Ax435CL/FCL	NVP-Ax430CL/ACL , -Ax435CL/FCL の略称
NVP-Linux	Linux installed on the NVP unit	NVPユニットに搭載されているLinux
PC	Personal Computer	パーソナルコンピュータ
SSH	Secure Shell	セキュアな通信を行うためのプロトコル

目次

1. NVP-Ax430SDKでの開発環境設定

1.1 SDKのインストールと環境設定

NVP-Ax430SDKインストールマニュアルに従い以下のインストールと環境設定を行ってください。

① Windowsドライバのインストール

② NVP-Ax430SDKのインストール

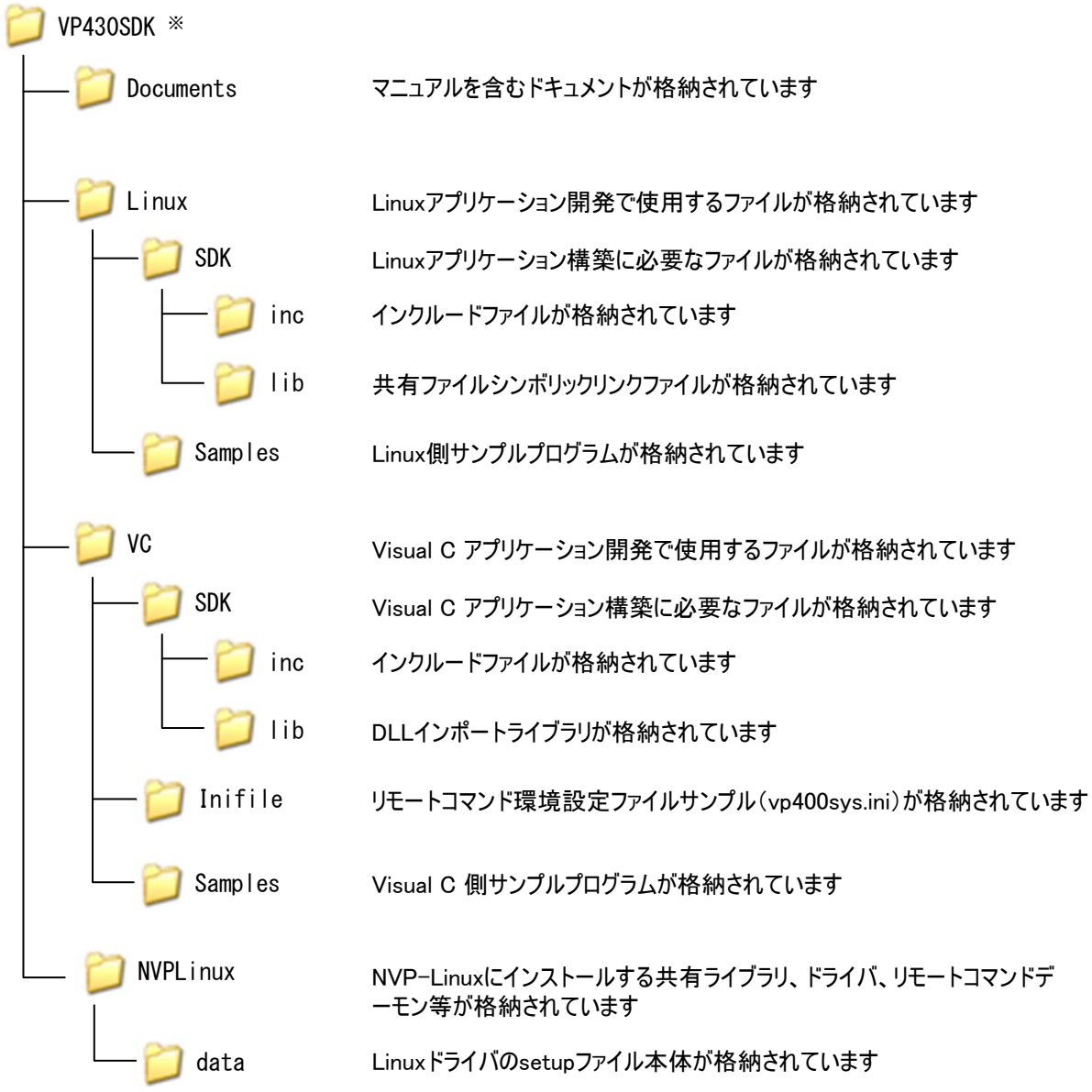
③ 環境設定

④ Linuxドライバのインストール

1.2 インストール内容とサンプルプログラム

NVP-Ax430SDKのインストールにより以下に示すフォルダ・ファイルがSDKインストールドライブに作成されます。サンプルプログラムは、Linux側は「Linux¥Samples」フォルダ、PC側は「VC¥Samples」フォルダに格納されています。

SDKインストールドライブ



※インストールフォルダが、「VP430SDK」の場合

図1-1 SDKフォルダ構成

2. Linuxアプリケーション開発環境の準備

NVP-Linuxで動作するアプリケーションを開発する場合の開発環境について説明します。Linuxアプリケーションの開発が必要ない場合は、本章で説明するコンパイラ、make、CMakeの開発環境は必要ありませんが、EclipseやTeraTermについては必要に応じてインストールしてください。

また、本章で説明するコンパイラ、make、Cmake、EclipseやTeraTermのライセンスは、各ツールのダウンロード先で確認の上、ご使用ください。

2.1 開発環境のダウンロード

開発環境をインターネットからダウンロードし、使用しているパソコンにインストールを行います。
以降の開発環境ダウンロード格納先は「D:¥SDK430_SOFT」としておりますが、任意で読み替えてください。
（事前に「D:¥SDK430_SOFT」フォルダを作成してください。）

※本マニュアルで記載しているURLやファイル名、URL表示情報は2019年2月のものです。
表示情報が変更されている場合、適宜読み替えをお願い致します。

2.2 コンパイラのインストール

下記、URLより”gcc クロスコンパイラ・ツールチェイン gcc version 5.5(gcc-linaro-5.5.0-2017.10)”をダウンロードし、解凍してください。

表2-1 gcc version 5.5のダウンロード

URL	https://releases.linaro.org/components/toolchain/binaries/5.5-2017.10/arm-linux-gnueabihf/
ファイル名	gcc-linaro-5.5.0-2017.10-i686-mingw32_arm-linux-gnueabihf.tar.xz

※本マニュアルはv5.5.0 にて以下の説明を行います。
※本マニュアルでは”c:¥gcc_toolchain”フォルダを作成し、”gcc-linaro-5.5”に解凍しています。
※gcc最新版は上記URLからダウンロードできます。

2.2.1 コンパイラのインストール手順

①「C:¥gcc_toolchain」フォルダを作成し、「D:¥SDK430_SOFT」に表2-1に示すURLから表2-1に示すファイルをダウンロードする

②「C:¥gcc_toolchain」フォルダを作成し、「D:¥SDK430_SOFT」に保存されている「gcc-linaro-5.5.0-2017.10-i686-mingw32_arm-linux-gnueabihf.tar.xz」を解凍し、さらに「C:¥gcc_toolchain」フォルダに解凍したtarファイルを解凍します。「C:¥gcc_toolchain」に解凍後のフォルダ名を「gcc-linaro-5.5」に変更してください(フォルダ名を短縮します)。

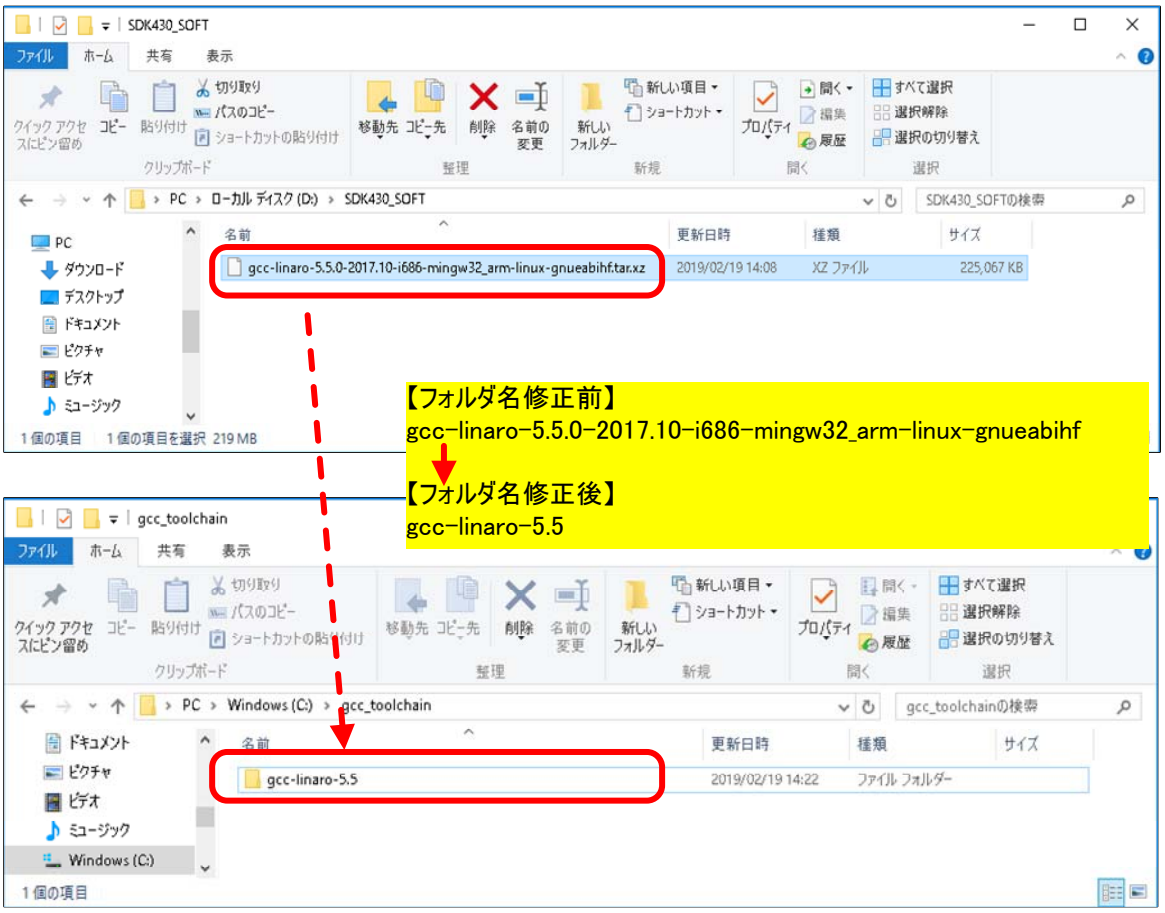
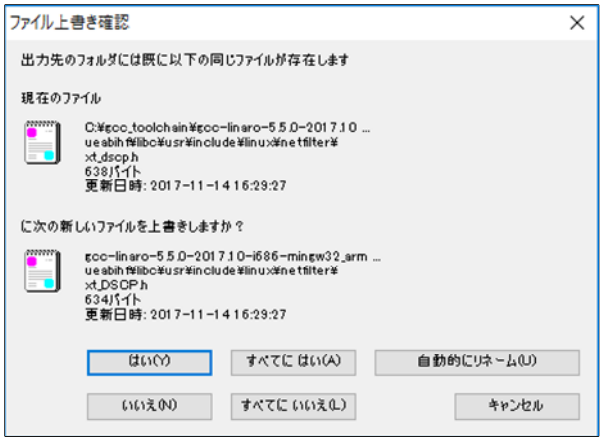


図2-1 ダウンロードファイルの解凍とフォルダ名変更

Note

解凍の途中で「ファイルが重複している」旨のメッセージが出力されます。



解凍の途中で「ファイルが重複している」旨のメッセージが出力され、「上書きする」、「上書きしない」、「自動的にリネーム」(別名で保存)のどれかを選択する選択画面が表示されます。「自動的にリネーム」または「別名で保存」が選択可能であれば、全て「自動的にリネーム」または「別名で保存」を選択してください。無い場合は、「上書きしない」で処理を進めてください。

重複の原因は、Linuxでは、「xt_DSCP.h」と「xt_dscp.h」のように英大文字と英小文字のファイル名は別ファイルとして扱われるのに対し、Windowsでは、同一ファイルとして扱われるためです。
以下に、重複するファイルを示します。
アプリケーションで下表のファイルを使用する場合は、解凍時「別名で保存」しインクルード指定でファイル名称を変更するなどの対応をしてください。

【重複ファイル】

No.	ディレクトリ(解凍ディレクトリからの相対)	重複ファイル名	
1	¥arm-linux-gnueabihf¥libc¥usr¥include¥linux¥netfilter¥	xt_dscp.h	xt_DSCP.h
2		xt_cnnmark.h	xt_CNNMARK.h
3		xt_rateest.h	xt_RATEEST.h
4		xt_tcpmss.h	xt_TCPMSS.h
5		xt_mark.h	xt_MARK.h
6	¥arm-linux-gnueabihf¥libc¥usr¥include¥linux¥netfilter_ipv6¥	ip6t_hl.h	ip6t_HL.h
7	¥arm-linux-gnueabihf¥libc¥usr¥include¥linux¥netfilter_ipv4¥	ipt_ttl.h	ipt_TTL.h
8		ipt_ecn.h	ipt_ECN.h

2.3 Eclipse(統合開発環境)のインストール

下記、URLより”Eclipse 4.5 Mars-32bit/Full Edition/C/C++”をダウンロードし、解凍してください。

表2-2 Eclipse 4.5 Marsのダウンロード

URL	http://mergedoc.osdn.jp/
ファイル名	pleiades-e4.5-cpp-32bit-jre_20160312.zip

※本マニュアルはv4.5 にて以下の説明を行います。
※本マニュアルでは”C:¥eclipse”フォルダを作成してファイルをコピーし、解凍しています。
※Eclipse最新版は上記URLからダウンロードできます。

2.3.1 Eclipse(統合開発環境)のインストール手順

①表2-2に示すURLを開きます。

②「Eclipse 4.5 Mars」を選択して「Pleiades All in One 日本語ディストリビューション (zip) ダウンロード」画面を表示します。

③「32bit / Full Edition / C/C++」 の[Download]」ボタンをクリックし、「名前を付けて保存(A)」を選択して「D:¥SDK430_SOFT」にEclipseをダウンロードします。

④「C:¥eclipse」を作成し、③でダウンロードしたEclipseを解凍します。

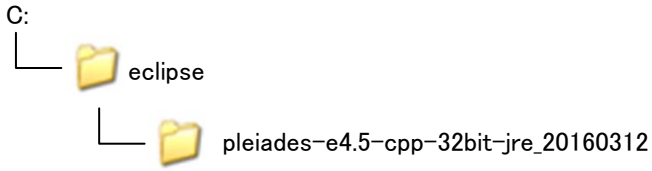


図2-2 解凍後フォルダ

2.4 makeのインストール

下記、URLより”make for Windows”をダウンロードし、解凍してください。

表2-3 make 3.81のダウンロード

URL	http://gnuwin32.sourceforge.net/packages/make.htm
ファイル名	make-3.81-bin.zip, make-3.81-dep.zip

※本マニュアルはv3.81 にて以下の説明を行います。
※本マニュアルでは”C:¥AP¥make”フォルダを作成してファイルをコピーし、解凍しています。
※make最新版は上記URLからダウンロードできます。

2.4.1 makeのインストール手順

①表2-3に示すURLを開きます。

②「Make for Windows」画面で「Download」カテゴリー→「Binaries」の「Zip」をクリックします。ダイアログが表示されますので、「名前を付けて保存(A)」を選択してmakeのバイナリデータをダウンロードします。

③「Make for Windows」画面で「Requirements」カテゴリー→「dependencies zip file」をクリックします。ダイアログが表示されますので、「名前を付けて保存(A)」を選択してmake用のDLLファイルをダウンロードします。

④ダウンロードした「make-3.81-bin.zip」と「make-3.81-dep.zip」を「D:¥SDK430_SOFT」フォルダに解凍します。

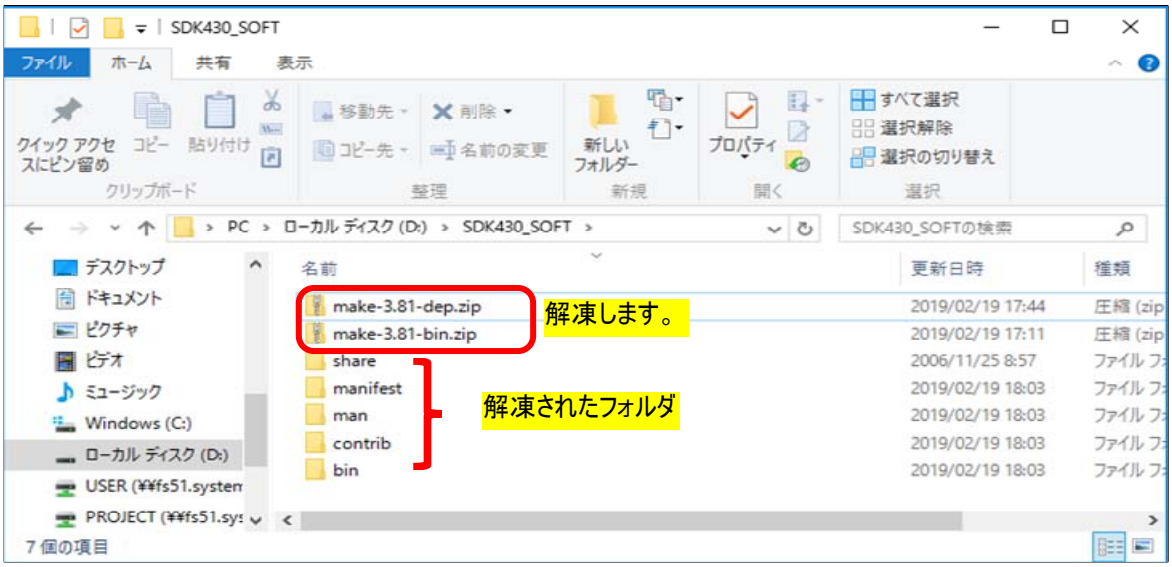


図2-3 圧縮ファイルの解凍

⑤C:¥AP¥makeフォルダを作成し、make実行ファイルとDLLファイルを格納します。

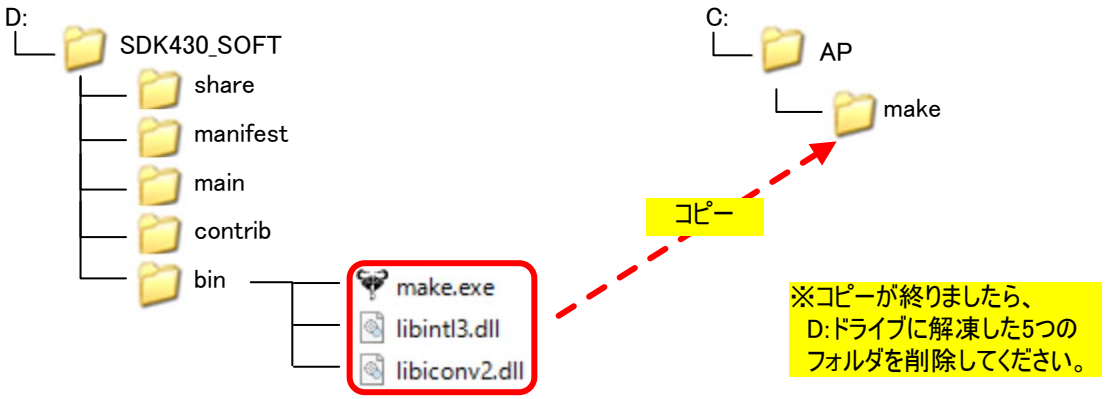


図2-4 makeファイルの格納

2.5 CMakeのインストール

下記、URLより”CMake”をダウンロードし、インストールしてください。

表2-4 cmakeのダウンロード

URL	https://cmake.org/download/
ファイル名	cmake-3.13.4-win32-x86.msi

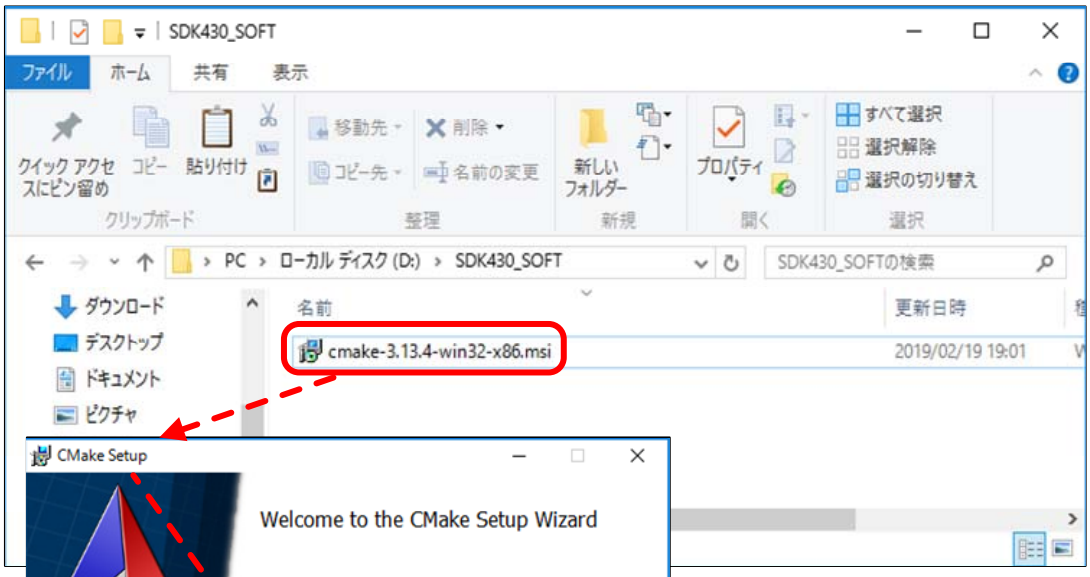
※本マニュアルはv3.13.4 にて以下の説明を行います。
※CMake最新版は上記URLからダウンロードできます。

2.5.1 CMakeのインストール手順

①表2-4に示すURLを開きます。

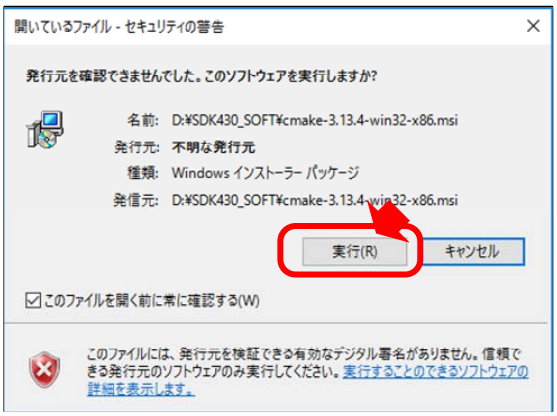
②「Cmake」画面から「Latest Release(3.13.4)」カテゴリー→「cmake-3.13.4-win32-x86.msi」をクリックし、「名前を付けて保存(A)」を選択して「D:¥SDK430_SOFT」にcmakeをダウンロードします。

③「D:¥SDK430_SOFT」に保存した「cmake-3.13.4-win32-x86.msi」を実行します。



Note

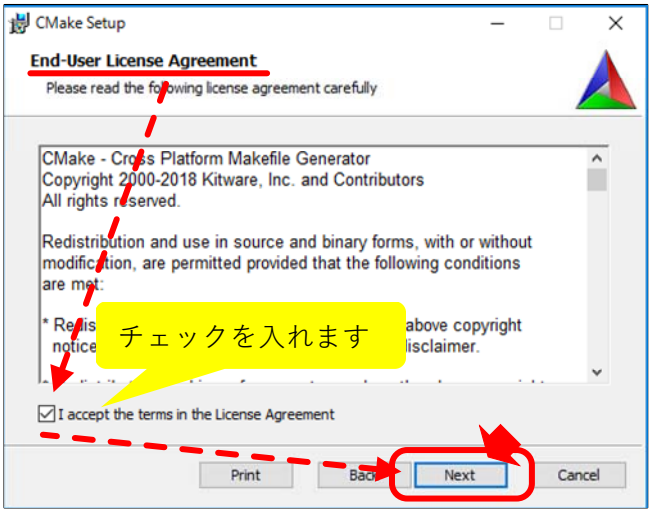
セキュリティ警告画面が表示された場合



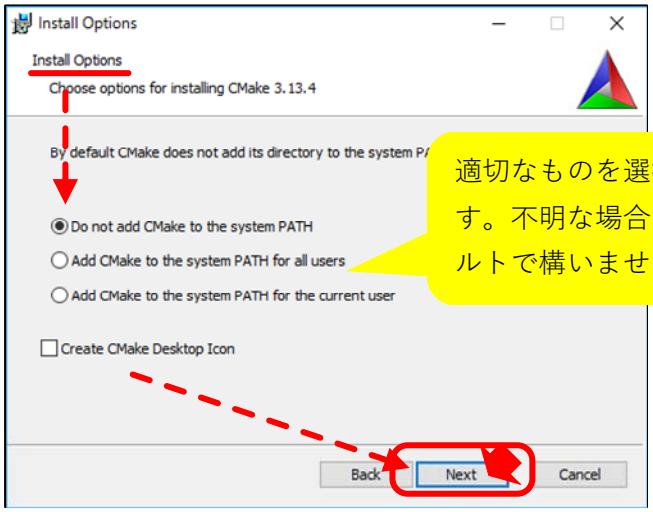
「発行元を確認できませんでした。」のアラートウィンドウが表示された場合、[実行(R)]ボタンを押してインストールを継続してください。

図2-5 CMakeのインストール(1/2)

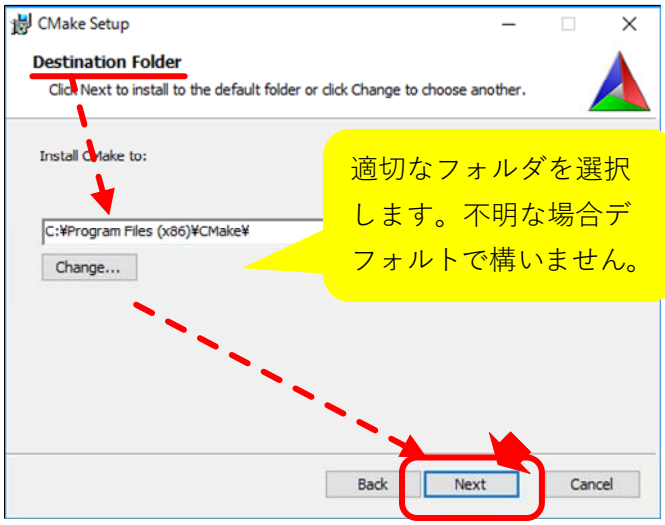
Step1 ：使用許諾への承認



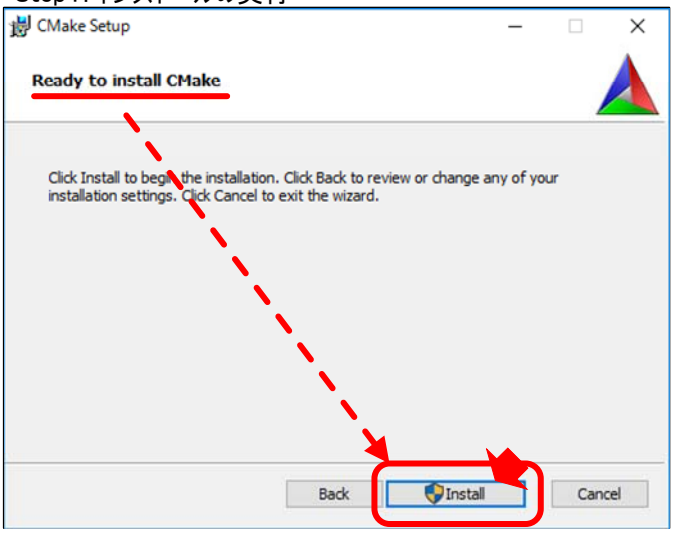
Step2：インストールオプションの選択



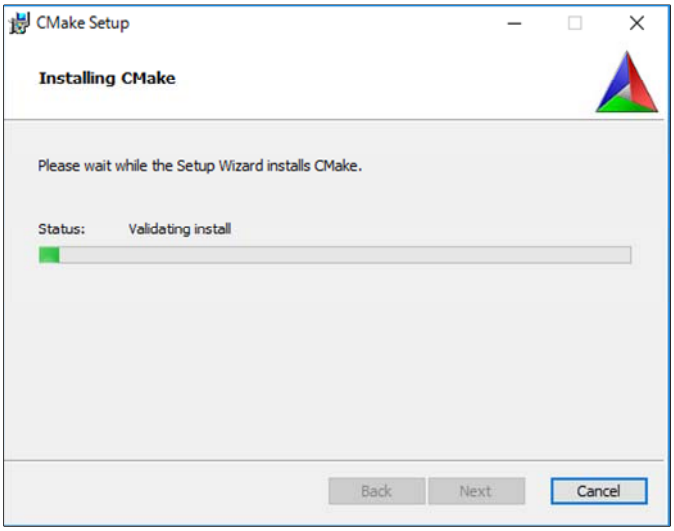
Step3：インストール先の選択



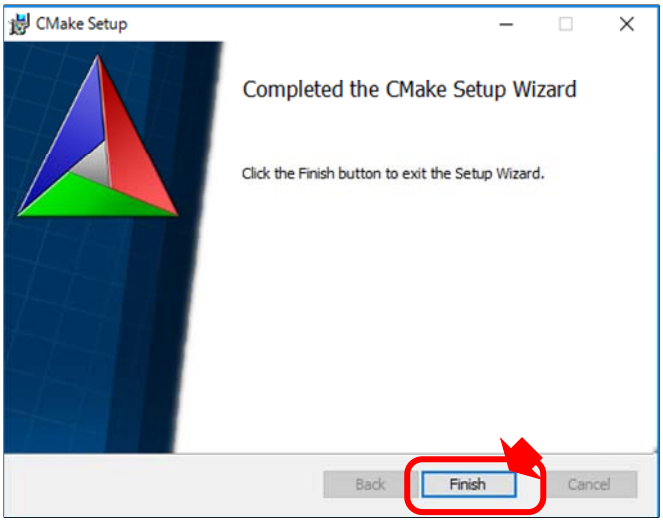
Step4：インストールの実行



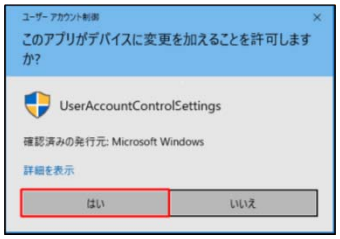
Step5：インストール



Step6：インストール終了



Note



Step4とStep5の途中で、ユーザアカウントの制御画面となった場合、[はい]ボタンを押してインストール作業を継続してください。

図2-6　CMakeのインストール(2/E)

2.6 TeraTermのインストール

下記、URLより”TeraTerm”をダウンロードし、インストールしてください。

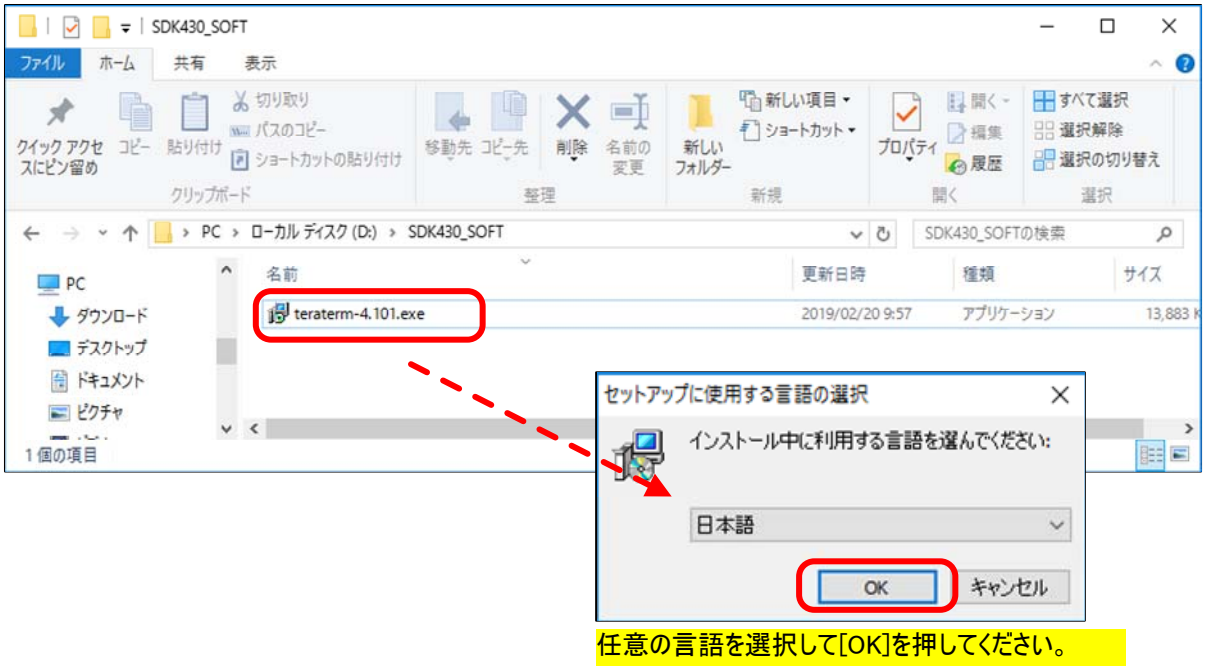
表2-5 TeraTermのダウンロード

URL	https://ja.osdn.net/projects/ttssh2/releases/
ファイル名	teraterm-4.101.exe

※本マニュアルはv4.101 にて以下の説明を行います。
※TeraTerm最新版は上記URLからダウンロードできます。

2.6.1 TeraTermのインストール手順

①表2-5に示すURLを開きます。
②「Tera Term」画面の「ダウンロードパッケージ一覧」カテゴリ→「teraterm-4.101.exe」をクリックし、「名前を付けて保存(A)」を選択して「D:¥SDK430_SOFT」にTeraTermをダウンロードします。
③「D:¥SDK430_SOFT」に保存した「teraterm-4.101.exe」を実行します。

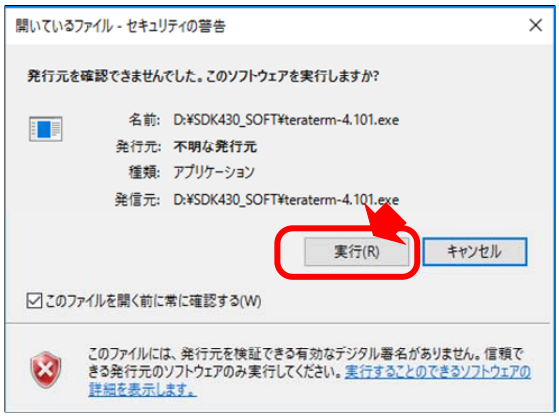


任意の言語を選択して[OK]を押してください。

図2-7　teraterm-4.101.exeの実行

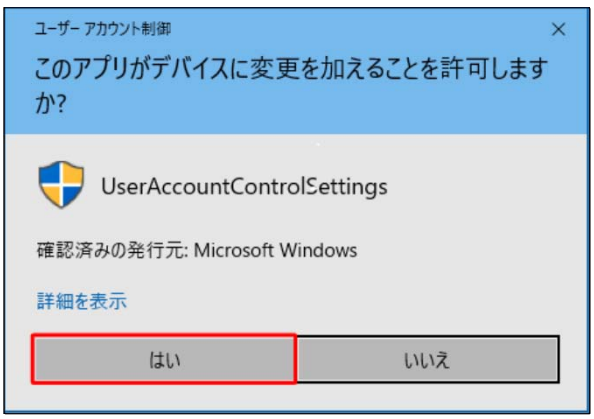
Note

セキュリティ警告画面が表示された場合



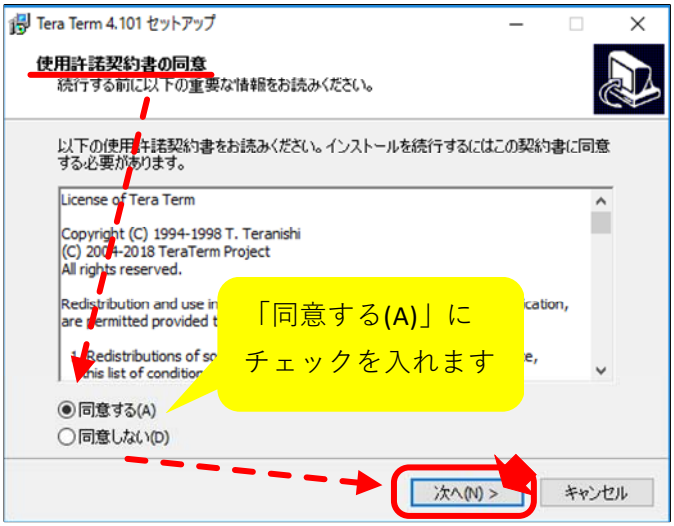
「発行元を確認できませんでした。」のアラートウィンドウが表示された場合、[実行(R)]ボタンを押してインストールを継続してください。

ユーザアカウント制御画面が表示された場合

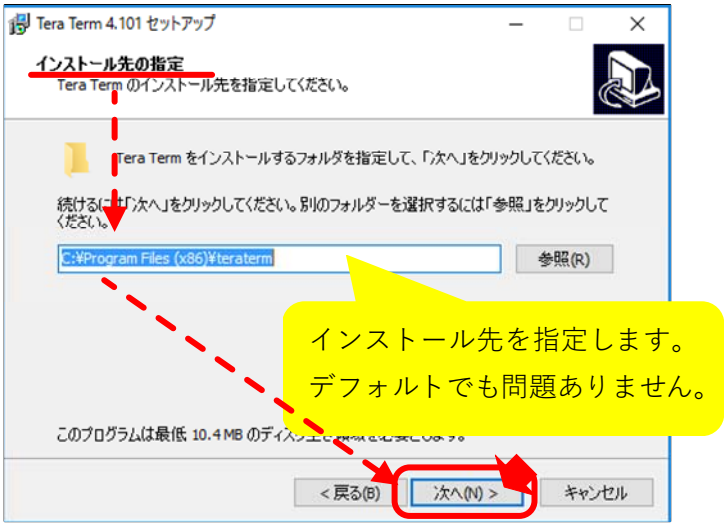


「ユーザアカウント制御」のウィンドウが表示された場合、[はい]ボタンを押してインストールを継続してください。

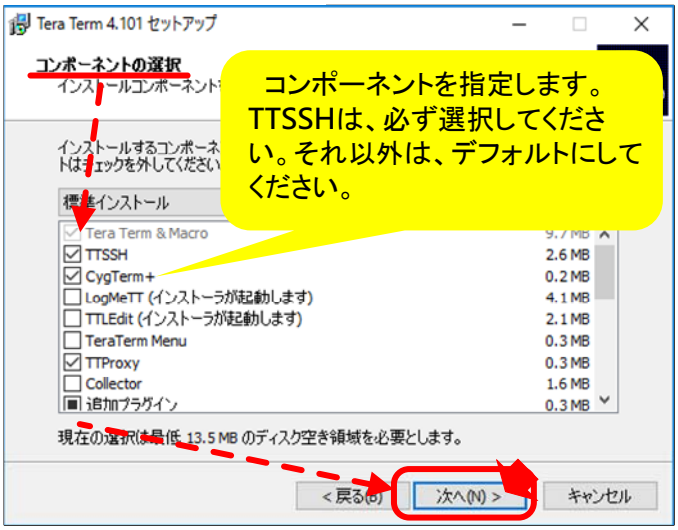
Step1:使用許諾の同意



Step2:インストール先の指定



Step3:コンポーネントの選択



Step4:言語の選択

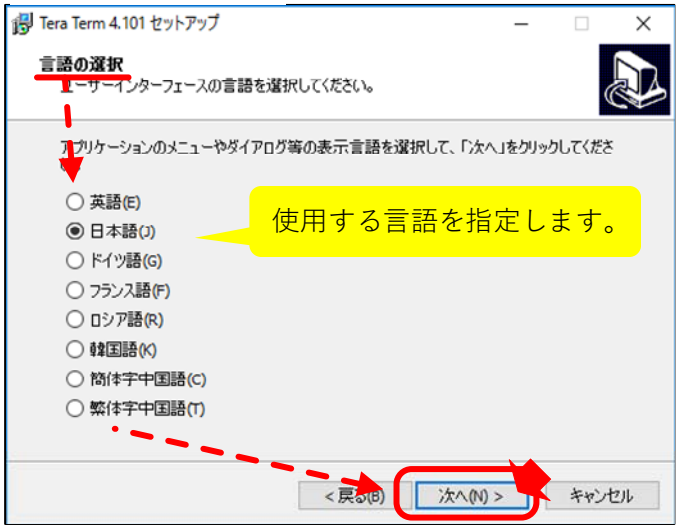
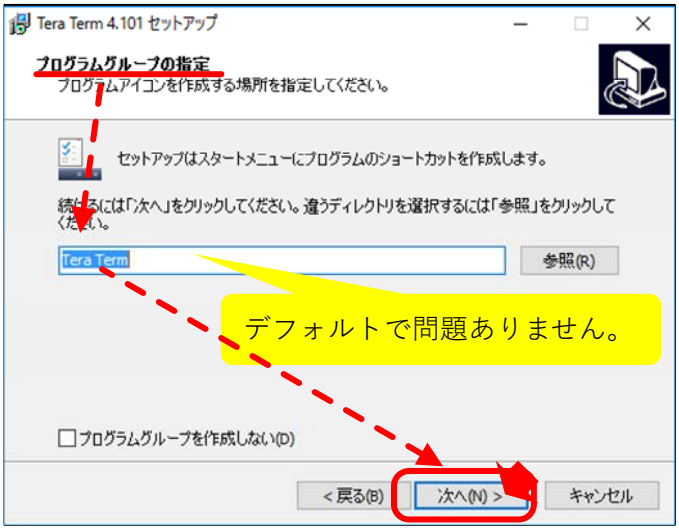
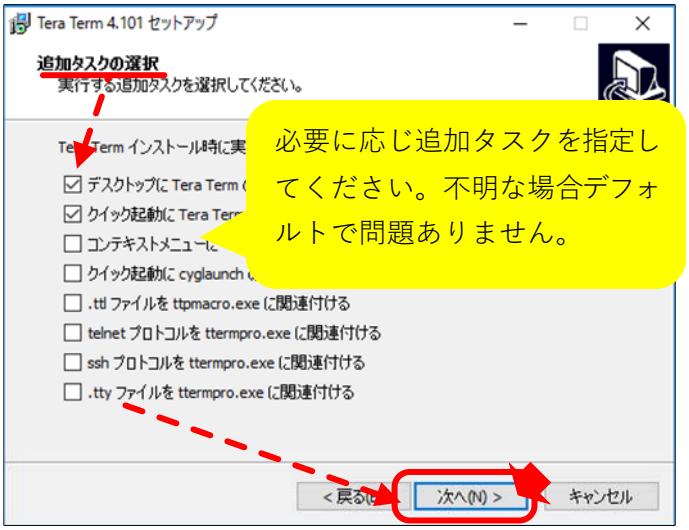


図2-8 TeraTermのインストール(1/2)

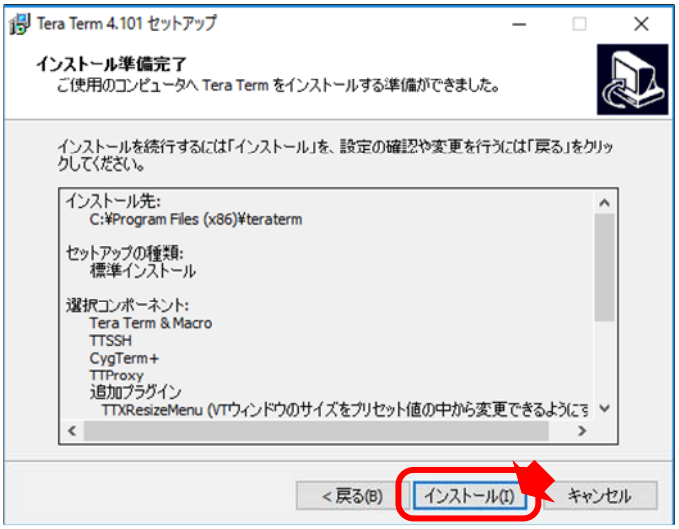
Step5:プログラムグループの指定



Step6:追加タスクの選択



Step7:インストール



Step8:インストールの完了

図2-9　TeraTermのインストール(2/E)

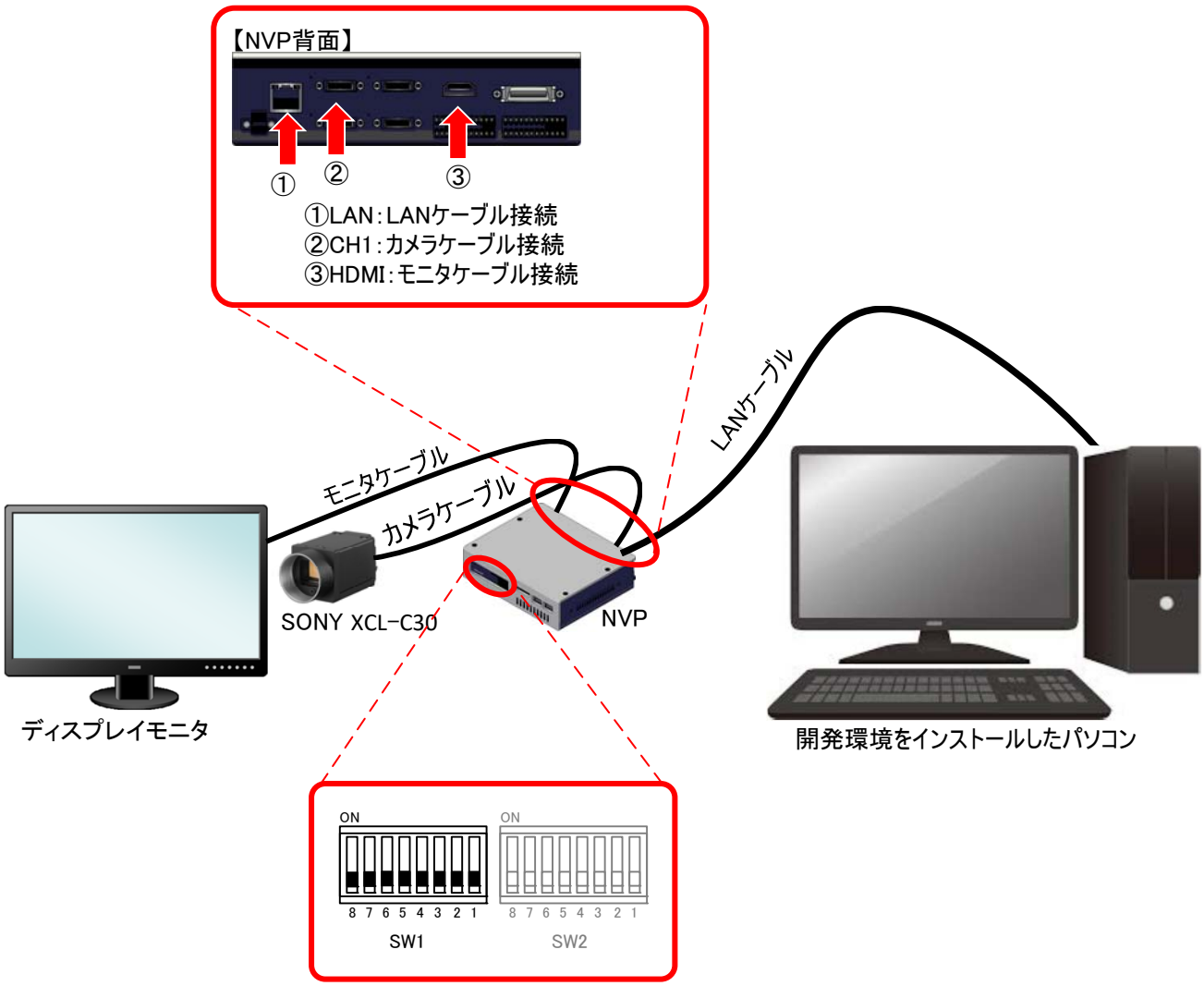
3. チュートリアル

NVP-Ax430SDKのLinuxアプリケーション及びPC側リモートコマンドアプリケーションのサンプルプログラムを例に開発手順を説明します。なお、本説明で使用するツール(コンパイラ、make、CMake、Eclipse、TeraTerm)の操作に関しての詳細は、個々のツールの説明書を参照してください。

3.1 ハードウェアの準備

開発環境として、以下機器接続を行ってください。

・開発環境をインストールしたパソコン
・NVP本体(電源ケーブル含む)
・カメラ「SONY XCL-C30」(カメラケーブル含む)
・ディスプレイモニタ(モニタケーブル含む)
・LANケーブル(クロス/ストレートどちらでも可能)



※NVPのDIPSWの設定についてはユーザーズマニュアルを参照してください。

※NVPの初期IPアドレスは「192.168.0.205」になっております。PC側のIPアドレスは「192.168.0.50」等のNVPと接続可能なアドレスに事前に設定してください。

図3-1 機器構成と接続

3.2 NVP-Linuxで動作するサンプルアプリケーション

カメラの入力映像を取り込み、２値化を行いディスプレイモニタに表示するまでを行うNVP-Linuxで動作するアプリケーションの構築とデバッグの説明を行います。

3.2.1 NVP-Linuxアプリケーション動作時のDIPSW(SW1)の設定

NVP-Linuxアプリケーションを動作させる場合、DIPSW（SW1）をNo1～No8の全てのスイッチをOFFにしてNVPを起動してください。

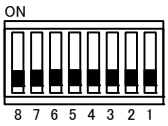


図3-2 DIPSW(SW1)設定

3.2.2 NVP-Linuxサンプルプロジェクトの構成

「NVP-Linuxで動作するサンプルアプリケーション」で使用するプロジェクトはSDKにサンプルとして提供しており、「図1-1 SDKフォルダ構成」の「Linux¥Samples」以下のファイルとなります。サンプルプロジェクトの構成を図3-3に示します。

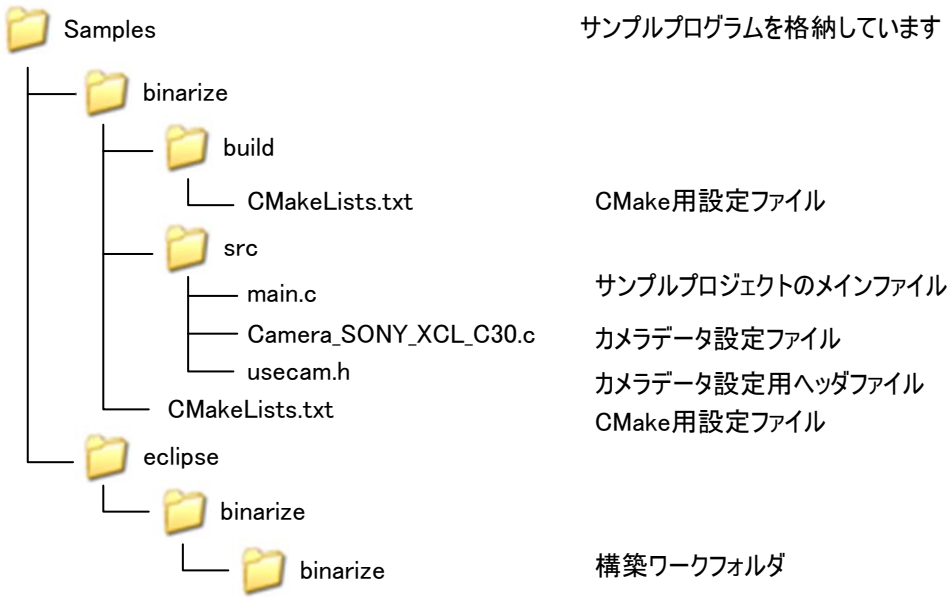


図3-3 Linux側サンプルフォルダ構成

3.3 Linuxサンプルアプリケーションの構築及びデバッグ

Linuxアプリケーション開発の一連の作業の流れを図3-4に示します。

Step1： ビルド環境/プロジェクトを作成 ・・・ 「3.3.1 CMakeによるEclipseプロジェクトの作成」参照

↓

Step2： アプリケーションのビルド ・・・ 「3.3.2 Eclipseでのアプリケーションのビルド」参照

↓

Step3： デバッグ環境の設定 ・・・ 「3.3.3 Eclipseでのデバッグ環境の設定」参照

↓

Step4： デバッグ ・・・ 「3.3.4 Eclipseでのデバッグ」参照

図3-4 Linuxアプリケーション開発手順

※チュートリアルのフォルダ構成は「指定フォルダ」を「C:¥VP430SDK」として図1-1、図3-3構成で説明します。
必要に応じ、SDKインストール時の設定に従ったフォルダ構成に読み替えてください。

3.3.1 CMakeによるEclipseプロジェクトの作成

①CMakeを起動し、”ソースコードがあるフォルダ”と”実行形式ファイルを出力するフォルダ”を指定します。

Where is the source code: C:/VP430SDK/Linux/Samples/binarize
Where to build the binaries: C:/VP430SDK/Linux/Samples/eclipse/binarize/binarize

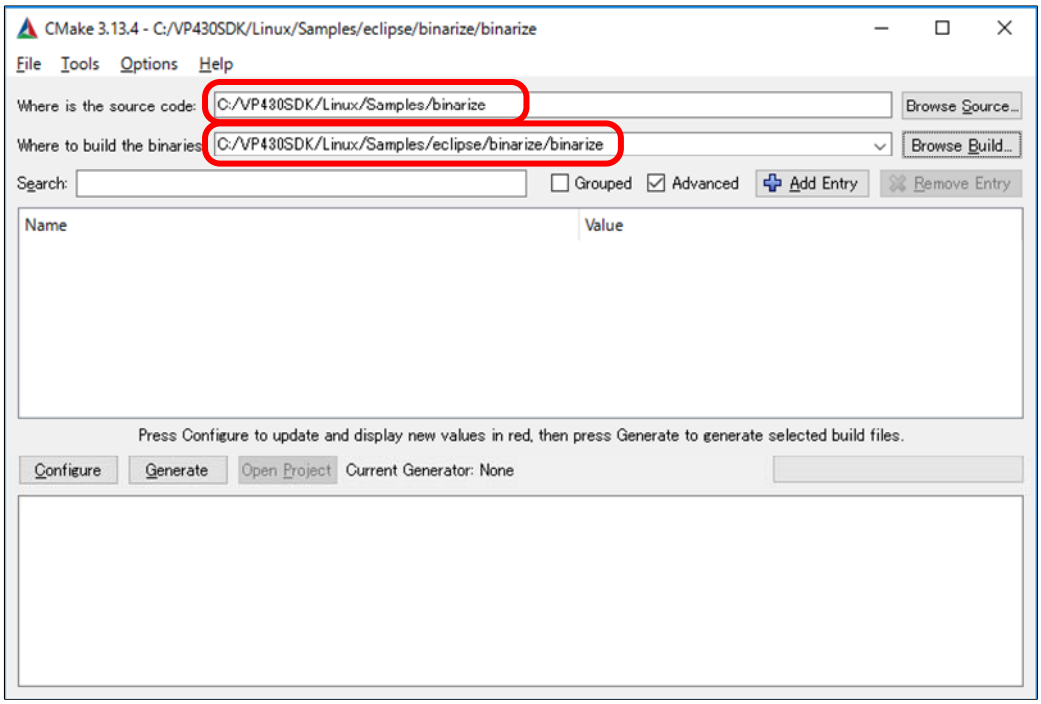
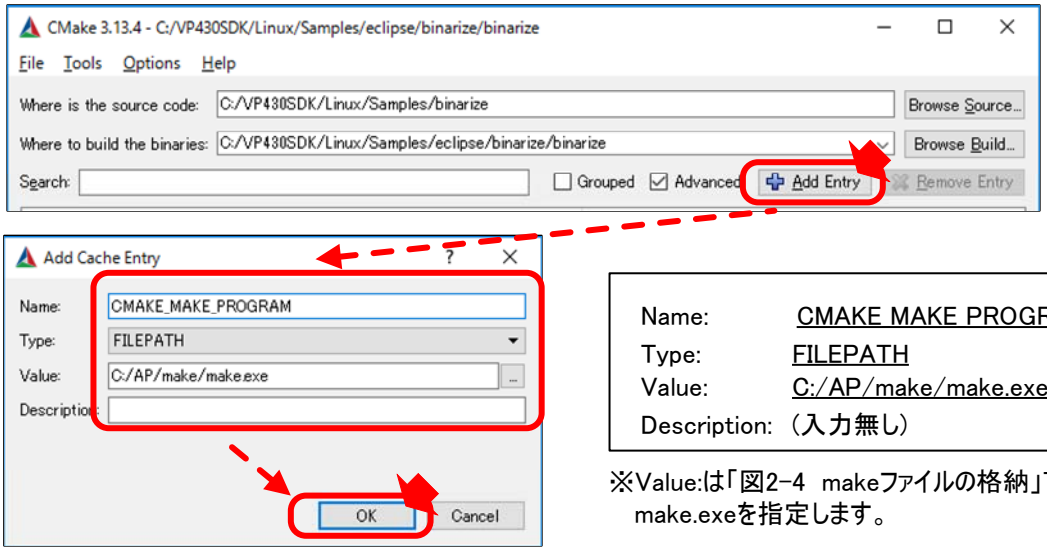


図3-5 フォルダ指定

Note

Advancedにチェックを入れる事により、以降の作業で詳細な情報が表示されるようになります。(任意)

②[+Add Entry]ボタンを押して、make.exe情報を追加します。



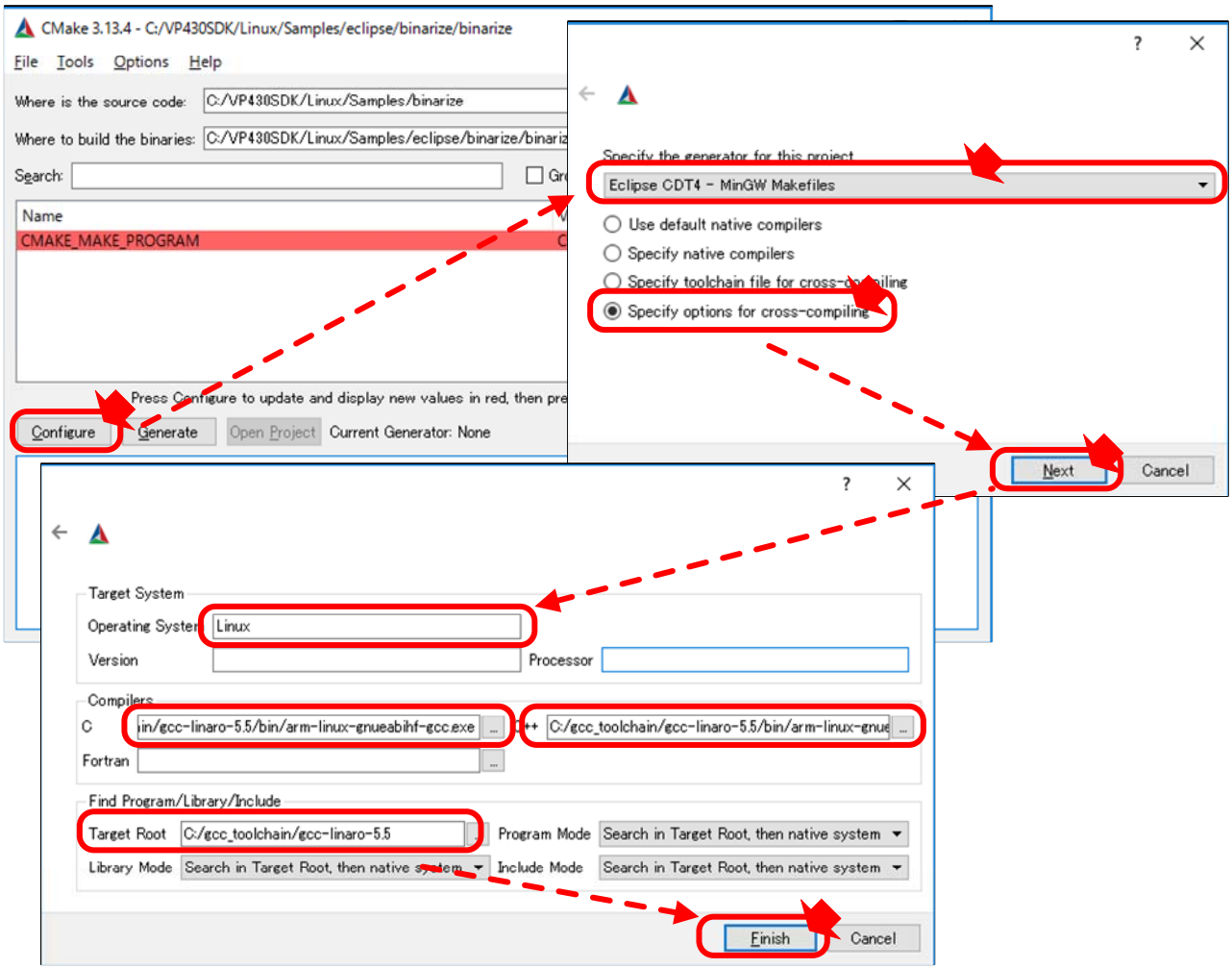
Name:	CMAKE_MAKE_PROGRAM
Type:	FILEPATH
Value:	C:/AP/make/make.exe
Description:	（入力無し）

※Value:は「図2-4　makeファイルの格納」で格納した make.exeを指定します。

図3-6 Add Cache Entryの設定

③[Configure]ボタンを押してコンパイラ情報等を設定しコンフィギュレーションを実行します。

Specify the generator for this project	Eclipse CDT4 - MinGW Makefiles　を選択
	Specify options for cross-compiling　を選択



Target System - Operating System		Linux
Compilers - C		C:/gcc_toolchain/gcc-linaro-5.5/bin/arm-linux-gnueabihf-gcc.exe
	C++	C:/gcc_toolchain/gcc-linaro-5.5/bin/arm-linux-gnueabihf-g++.exe
Find Program/Library/Include - Target Root		C:/gcc_toolchain/gcc-linaro-5.5

（他の設定はデフォルトのままでOKです）

図3-7 Configureの設定

④Eclipseのバージョン情報とビルドタイプを選択し、Generateを行います。

Name	Value
CMAKE_BUILD_TYPE	Debug を選択
CMAKE_ECLIPSE_VERSION	4.5 (Mars) を選択

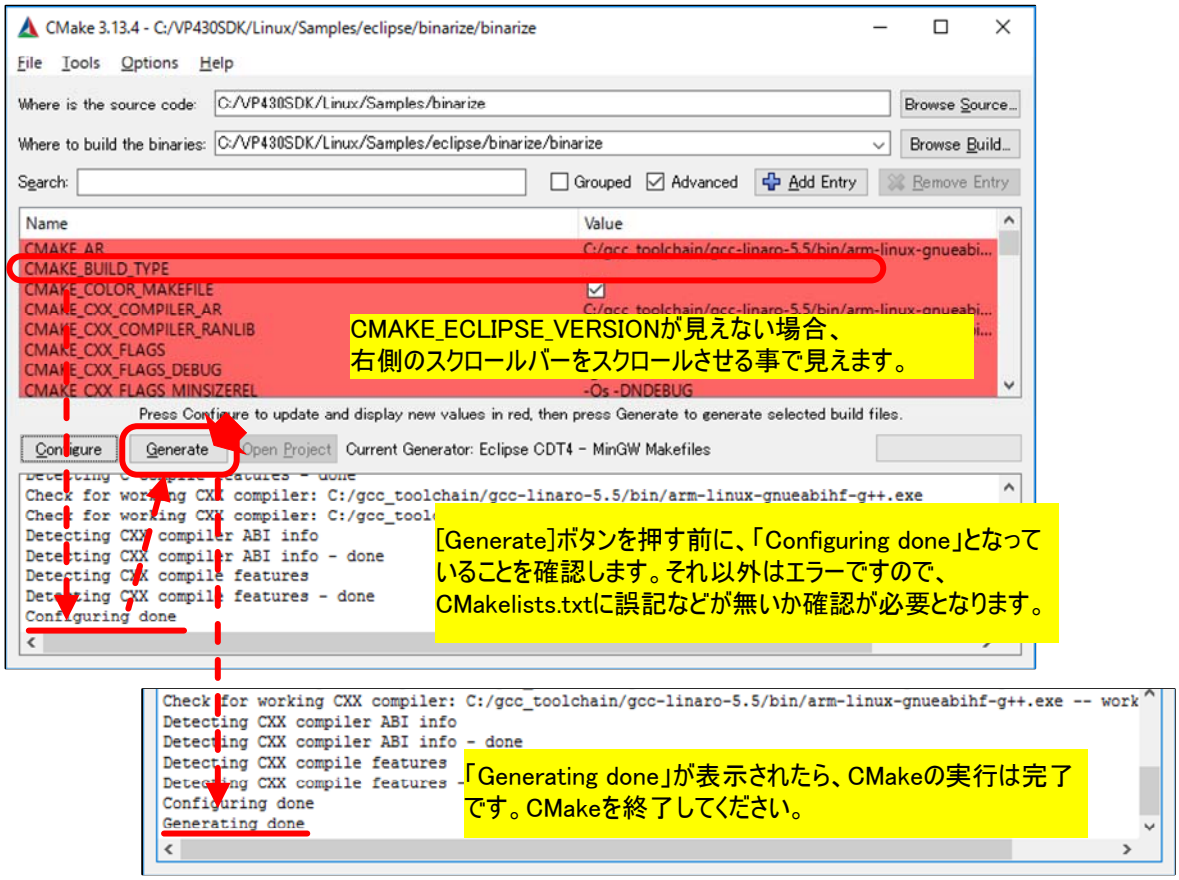


図3-8 Generateの設定、およびGenerate

Note

構築環境にインクルードパスやソースファイル、定数定義を追加したい場合

構築環境にユーザプログラムやインクルードパスやソースファイル、定数定義を追加する場合、CMake用設定ファイルに、それらの情報を追記します。

【CMake用設定ファイル】
sample¥Linux¥Samples¥binarize¥build¥CMakeLists.txt

```
##############################################################################
#    NVP-Ax430SDK サンプルプログラム
#

                （中略）

   # include path
   include_directories(
       ../src
       ../../../SDK/inc
   )

   add_executable( binarize
       ../src/main.c
       ../src/Camera_SONY_XCL_C30.c
   )

   find_library(dll_ipxcmds NAMES libipxcmds.so PATHS ../../../SDK/lib)

   target_link_libraries( binarize
       ${dll_ipxcmds}
       pthread
       rt
   )
```

インクルードファイルの有るパスを追記します。

ユーザプログラムのソースファイルを追記します。

また、プリプロセッサに定数定義を追加する場合は、

```
   target_compile_options( binarize PUBLIC
       -DXXXXXXXX
   )
```

XXXXXXXX は、定義名です

を上記CMakeList.txtに定義してください。

Note

gccのコンパイルオプションを追加、変更したい場合

gccのコンパイルオプションを追加、変更する場合、CMake用設定ファイルの「_PLATFORM_C_FLAGS」「_PLATFORM_C_FLAGS_DEBUG」、「_PLATFORM_C_FLAGS_RELEASE」の定義部分を追加、変更してください。

【CMake用設定ファイル】
sample¥Linux¥Samples¥binarize¥CMakeLists.txt

```
                （中略）

# default compiler options
if (${CMAKE_SYSTEM_NAME} MATCHES "Linux")
    # PC/ARM Linux
    set (_PLATFORM_C_FLAGS "-Wall -fsigned-char -fno-builtin-printf
-fPIC -march=armv7-a -mfpu=vfpv4 -mtune=cortex-a15
-lm -pthread -finput-charset=cp932 -fexec-charset=cp932")

    set (_PLATFORM_C_FLAGS_DEBUG "-g -O0")

    set (_PLATFORM_C_FLAGS_RELEASE "-O3")

    set(CMAKE_EXE_LINKER_FLAGS "${CMAKE_EXE_LINKER_FLAGS} -Wl,-Map -
Wl,MAP.map ")
endif()
```

デバッグ、リリース版共通オプション

デバッグ版専用オプション

リリース版専用オプション

3.3.2 Eclipseでのアプリケーションのビルド

アプリケーションのコンパイル、リンク（ビルド）はEclipseにより行います。

①Eclipseを起動し、「ワークスペースランチャー」ウィンドウが表示されるので以下フォルダを選択し、Workbenchを開きます。

ワークスペース(W): C:¥VP430SDK¥Linux¥Samples¥Eclipse¥binarize

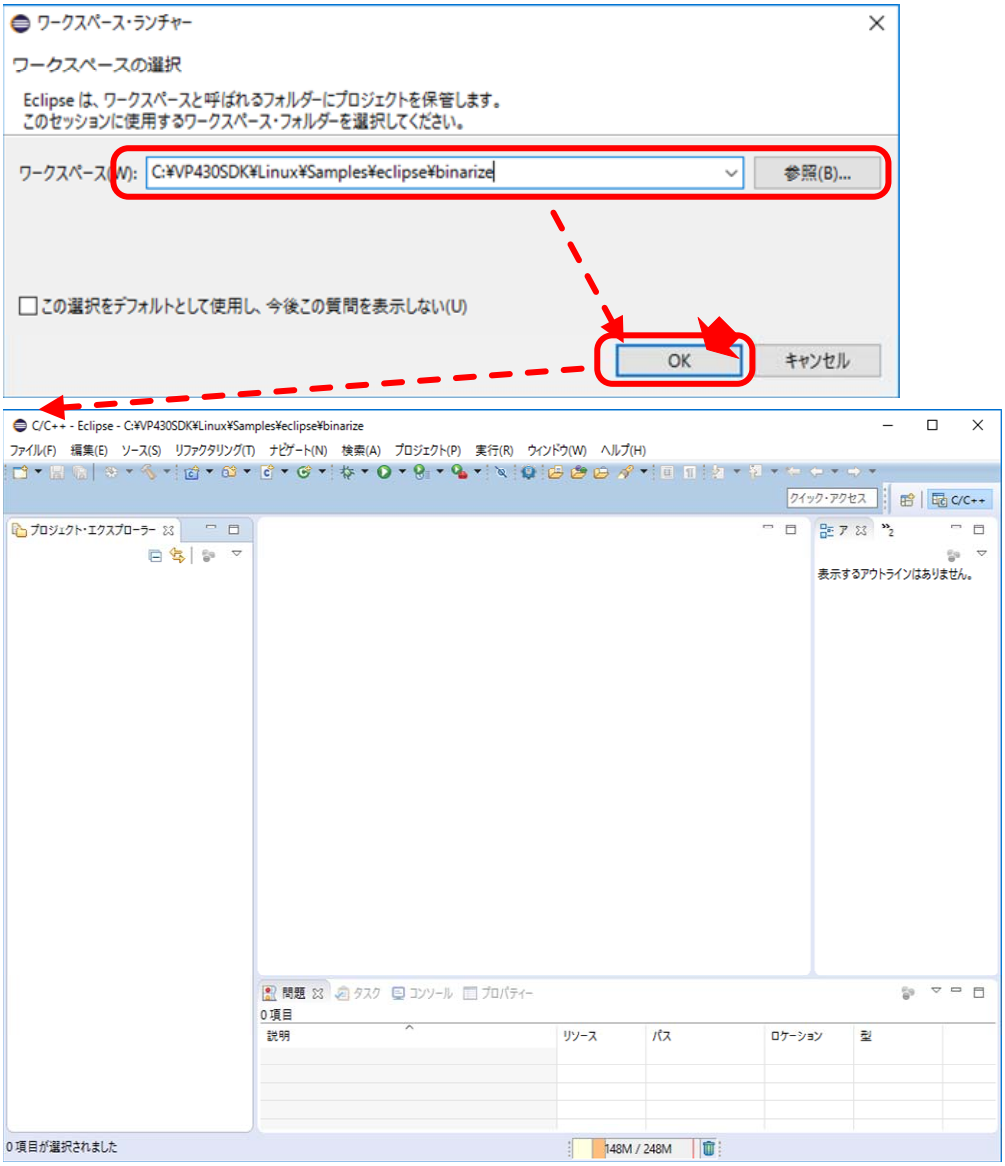


図3-9 Eclipse Workbenchのオープン

Note

Eclipseの設定によっては、ワークペース入力後「ようこそ」画面が表示されます。その場合、右上の「ワークベンチ」アイコンを選択して次に進んでください。

②CMakeで作成したプロジェクトをインポートします。

「ファイル(F)」メニュー→「インポート(I)...」を選択し、「インポート」ウィンドウを表示します。

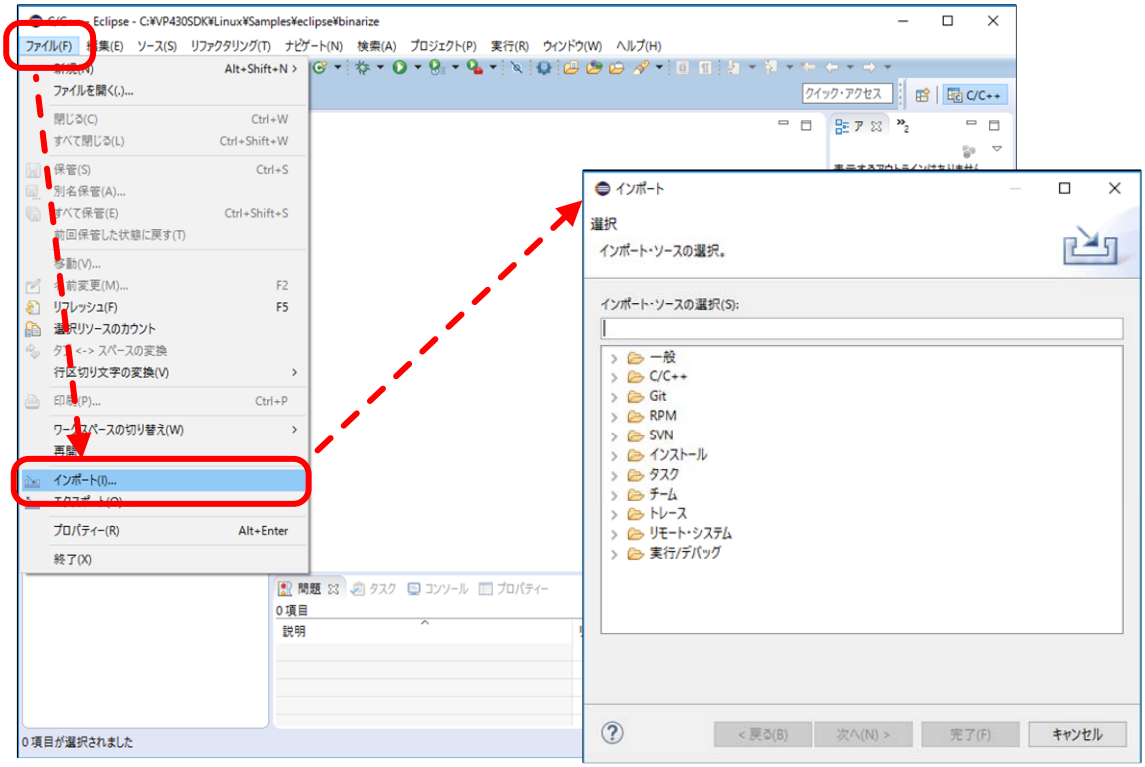


図3-10 インポートウィンドウの表示

「一般」→「既存プロジェクトをワークスペースへ」を選択し、[次へ(N)>]ボタンを押し、Eclipseプロジェクトを選択します。

ルート・ディレクトリーの選択(T): C:¥VP430SDK¥Linux¥Samples¥eclipse¥binarize¥binarize

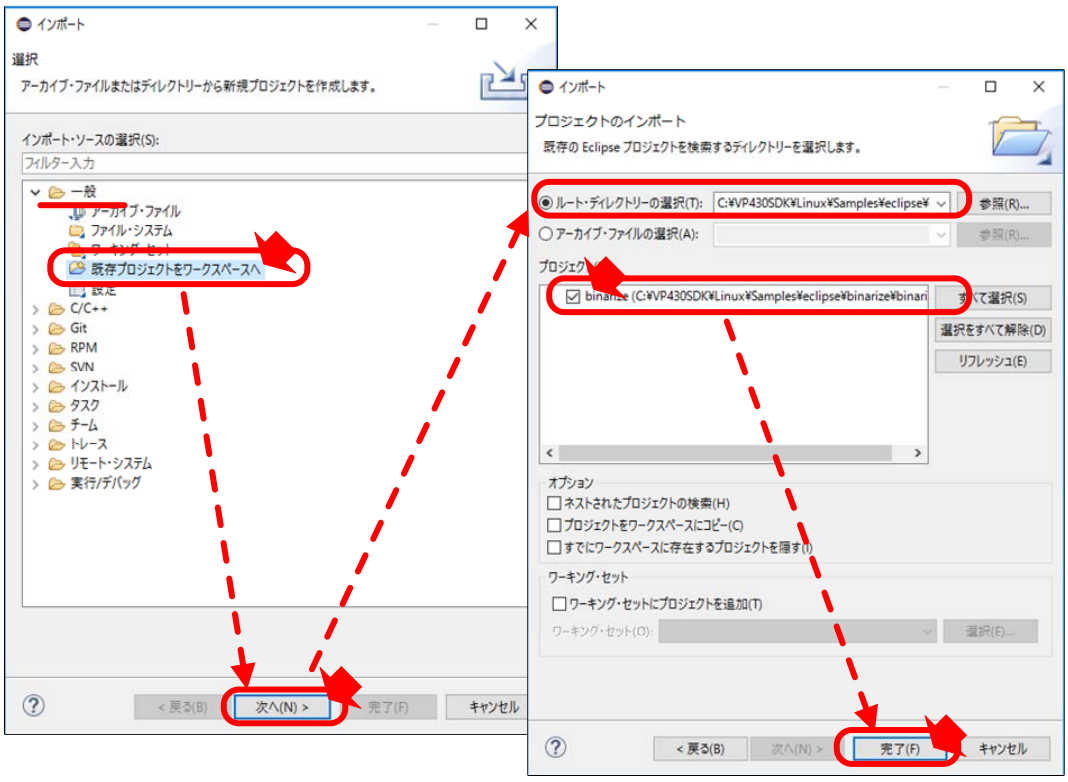


図3-11 Eclipseプロジェクトの選択

③サンプルプロジェクトのビルド

Workbenchの左側「プロジェクト・エクスプローラー」ビューの「binarize」
→[Source directory]→[src]を展開表示し、main.cファイルなどが登録されていることを確認し、
「プロジェクト(P)」メニュー→「すべてビルド(A) Ctrl+B」を選択して構築を実行します。

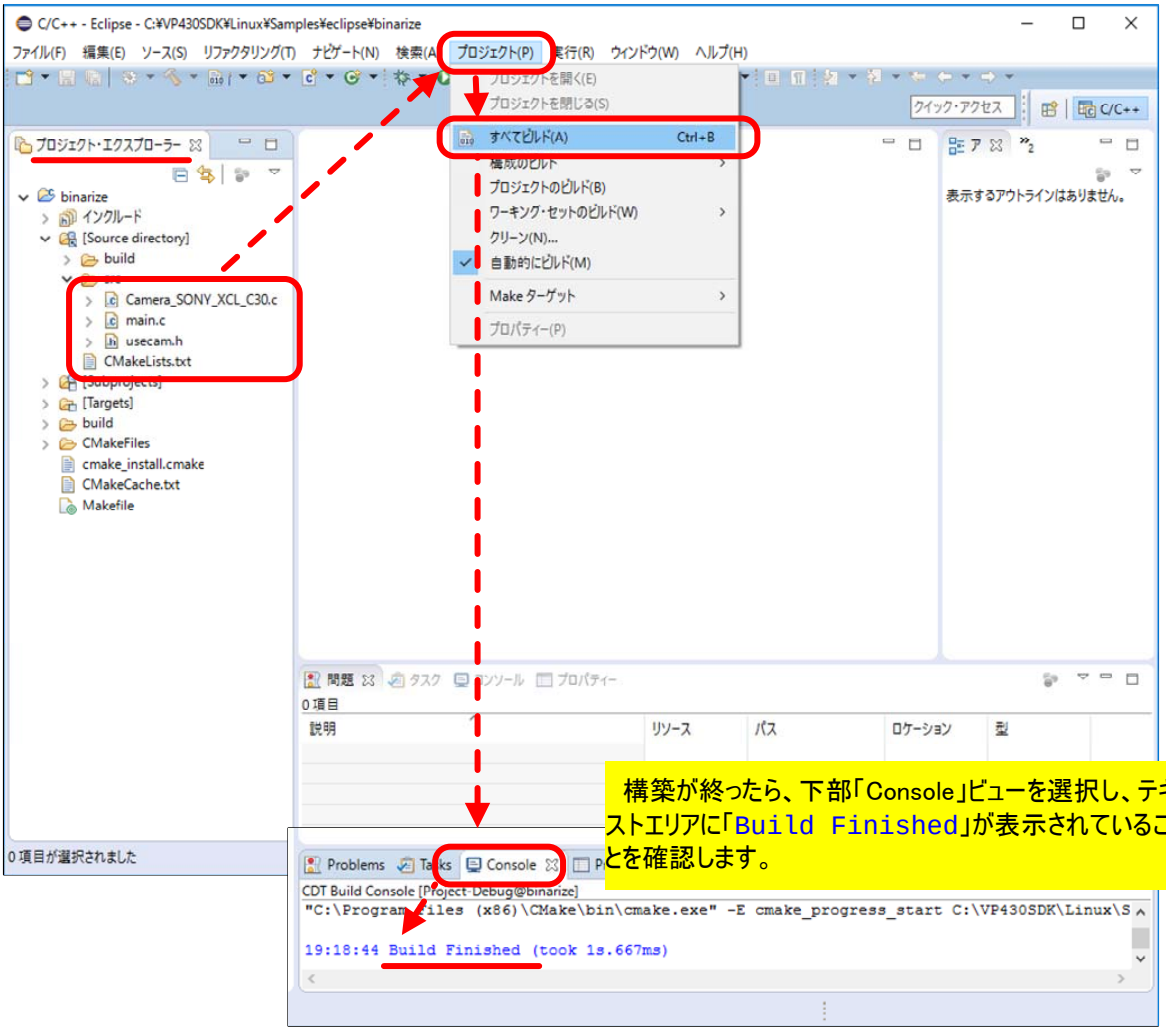


図3-12 サンプルプロジェクトの構築

Note

ソースコードをEclipseのエディタで表示、編集する場合、日本語文字にSHIFT-JISなどを使用している場合、コメントなどの日本語部が文字化けを起こします。正しく表示させるためには、Eclipseエディタの環境を事前に設定する必要があります。環境設定方法は「3.4 Eclipseの環境設定」を参照してください。

④サンプルプロジェクトの実行ファイル

①～③の操作でサンプルプロジェクト「binarize」をビルドした場合、

C:¥VP430SDK¥Linux¥Samples¥eclipse¥binarize¥binarize¥build¥

ディレクトリに実行ファイル名「binarize」として生成されます。

3.3.3 Eclipseでのデバッグ環境の設定

プロジェクトのビルド後、アプリケーションをデバッグする場合、デバッグの環境を設定する必要があります。

① NVP、およびモニタディスプレイの電源をONの後、プロジェクト「binarize」選択後、「実行(R)」メニュー→「デバッグの構成(B)...」を選択し、「デバッグ構成」ウィンドウを表示します。

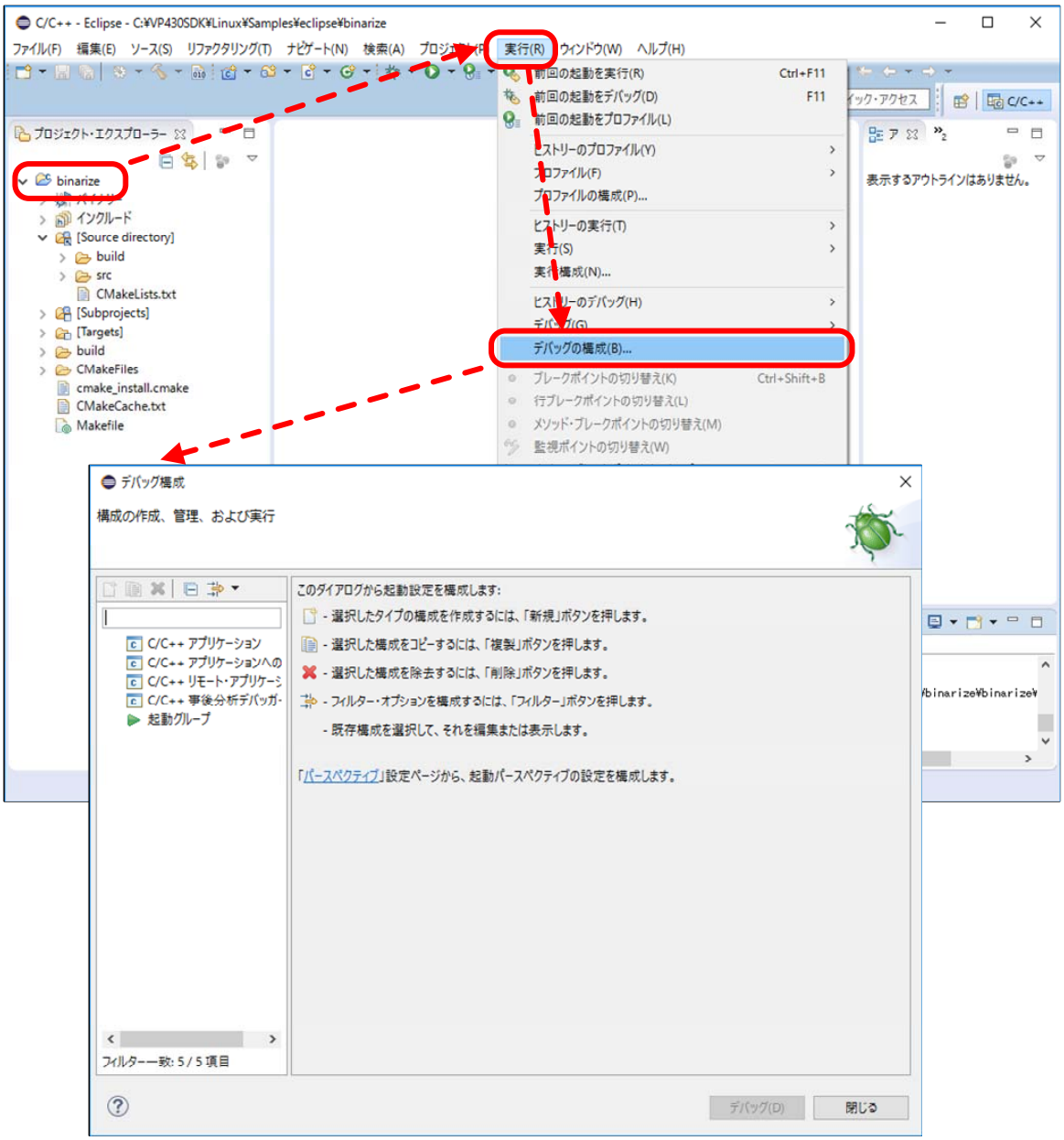


図3-13「デバッグ構成」ウィンドウの表示

② 「C/C++ リモート・アプリケーション」の上にマウスオーバし、右クリック→「新規(W)」を選択し、「メイン」ビューを表示します。

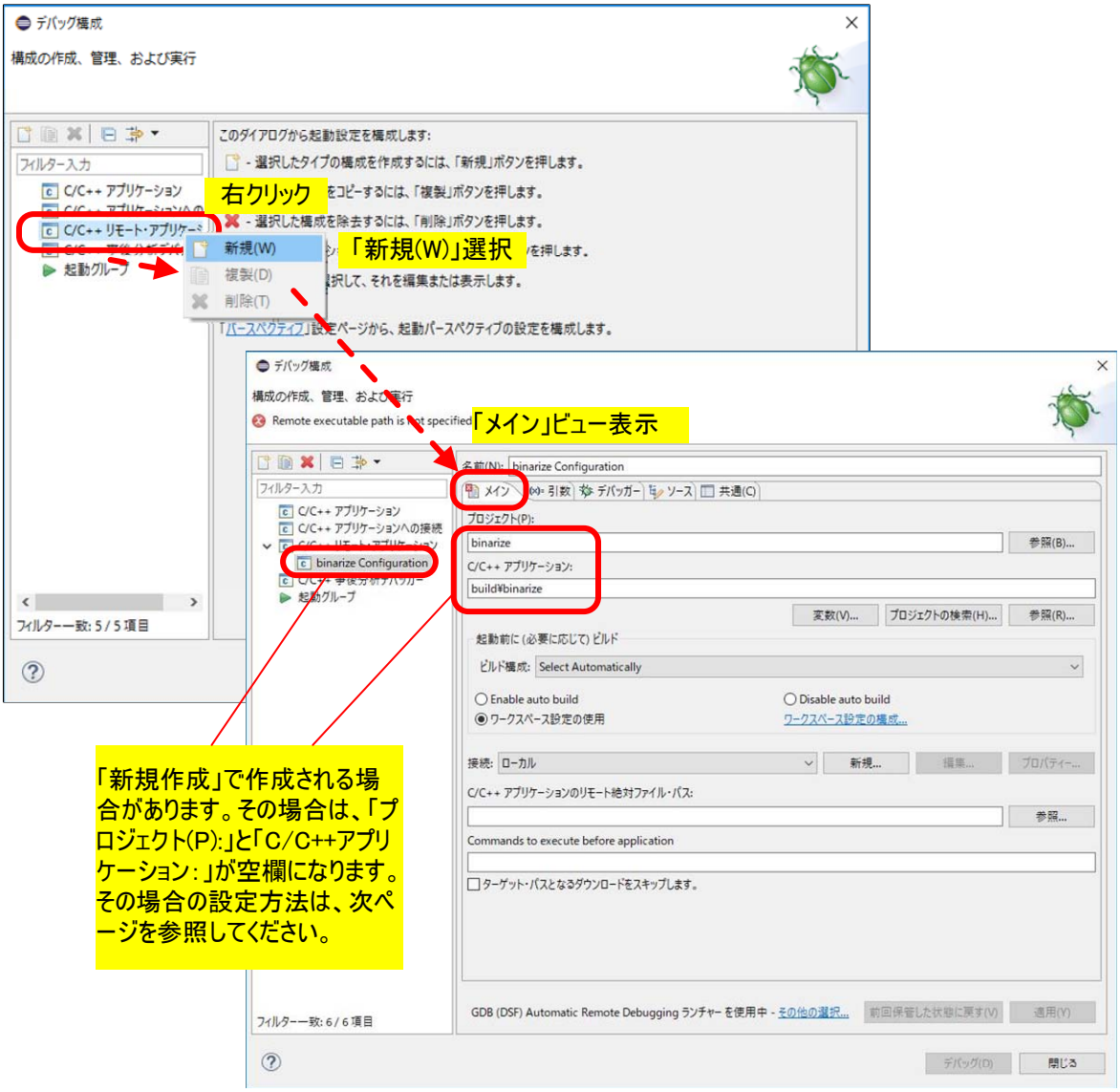
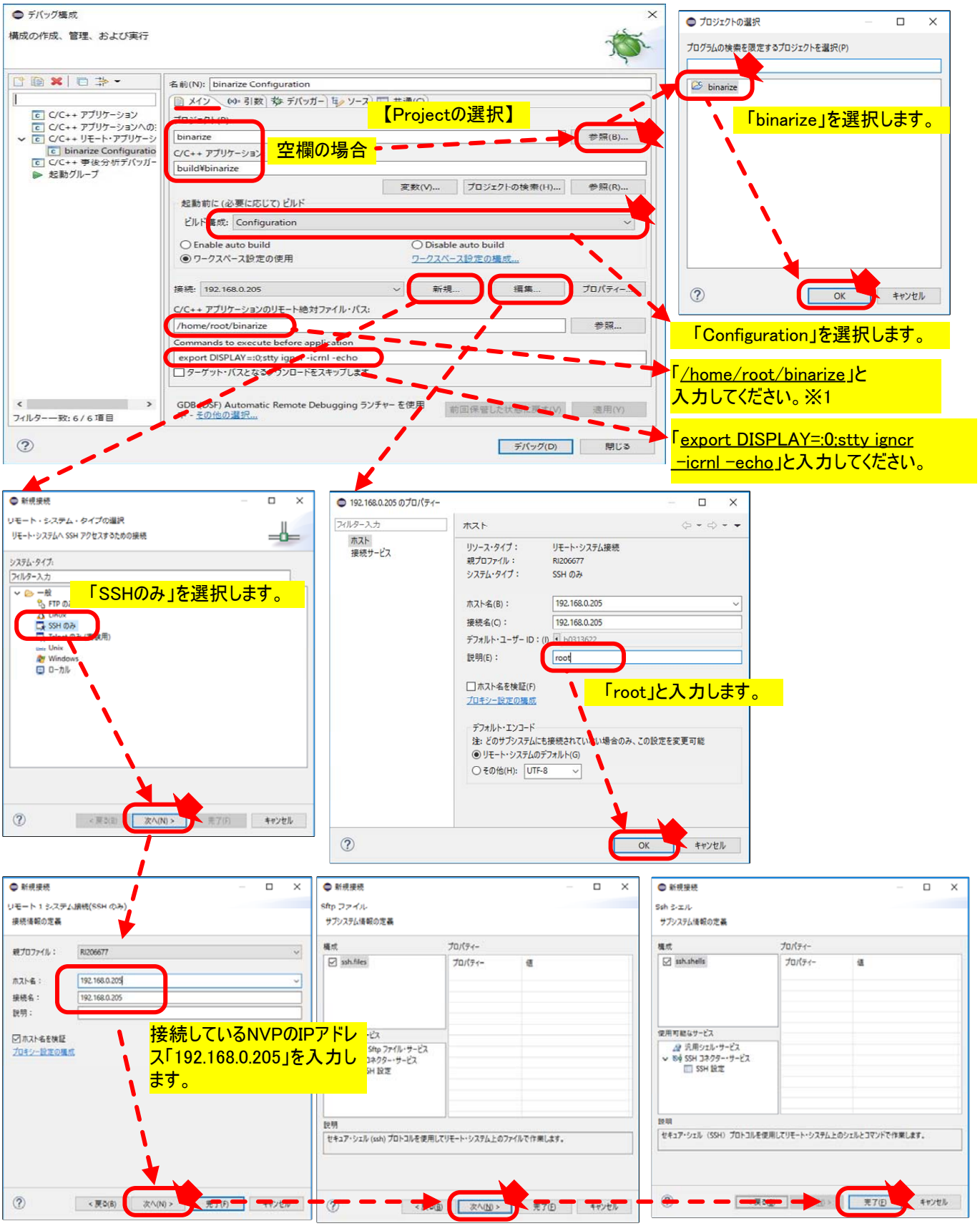


図3-14「メイン」ビューの表示

③「メイン」ビューの情報設定



※1「C/C++アプリケーションのリモート絶対ファイル・パス:」は、デバッグする実行ファイルのNVP-Linuxのパス(実行ファイル名含む)を指定するものです。「/home/root/binarize」と入力していますが、「/home/root/」がディレクトリ、「binarize」は、実行ファイル名になります。また、任意フォルダ、実行ファイル名の入力が可能です。なお、指定するディレクトリは事前に作成されている必要があります。

図3-15「メイン」ビューの情報設定

④「デバッガー」ビューの情報設定

「デバッガー」ビューの「メイン」タブを選択し、以下を設定します。

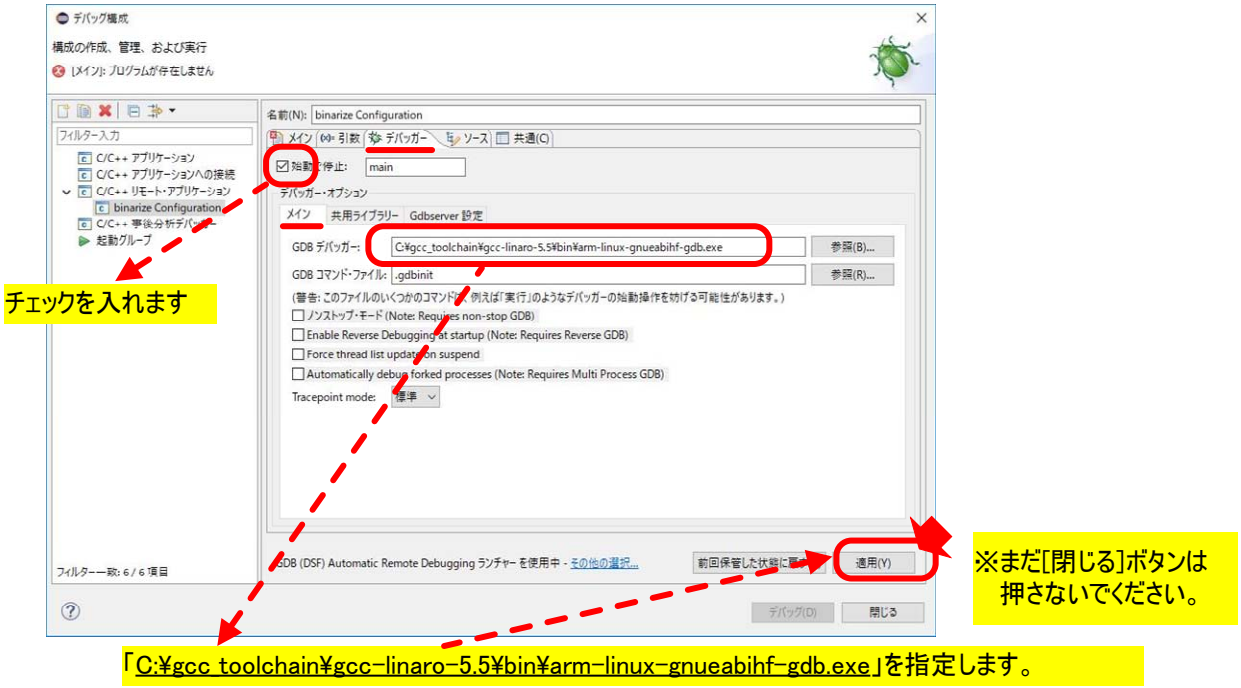


図3-16「デバッガー(「メイン」タブ)」ビューの情報設定

「デバッガー」ビューの「共有ライブラリ」タブを選択し、「共用ライブラリ・シンボルを自動的にロード」のチェックを外します。

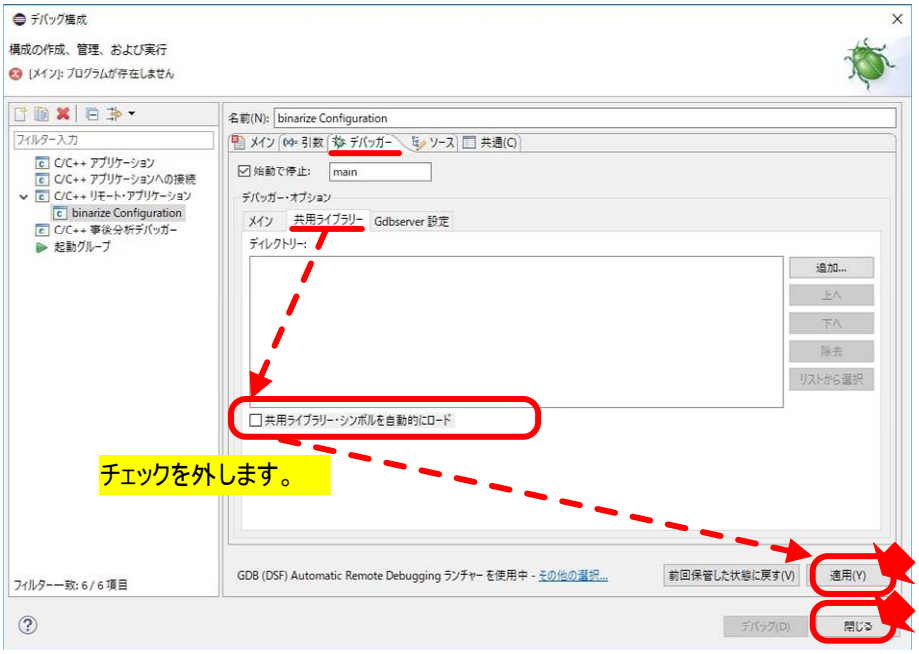


図3-17「デバッガー(「共有ライブラリー」タブ)」ビューの情報設定

3.3.4 Eclipseでのデバッグ

①デバッグは、「実行(R)」メニュー→「デバッグの構成(B)...」を選択し「デバッグ構成」ウィンドウを表示し、
「デバッグ構成」ウィンドウの左フレームビュー「C/C++リモート・アプリケーション」を展開表示し
「binarize Configuration」を選択し、ウィンドウ右下の[デバッグ(C)]ボタンを選択します。

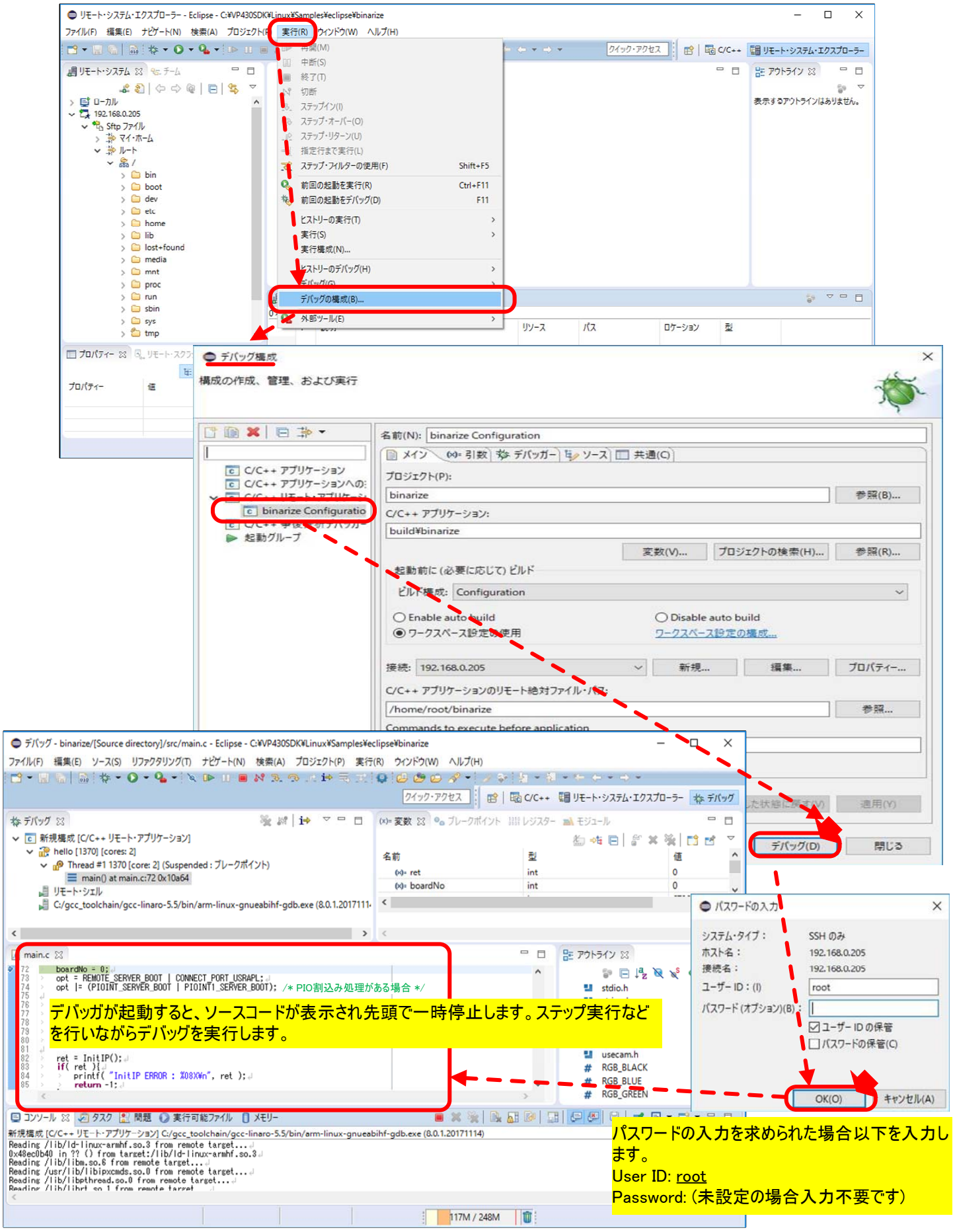
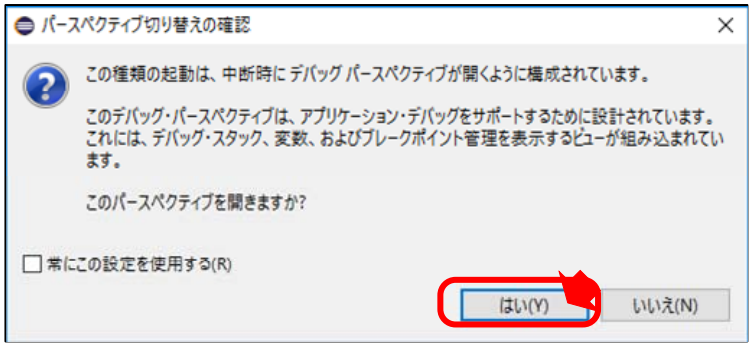


図3-18 デバッグ開始

Note

パースペクティブ変更の確認メッセージが表示された場合は「はい」ボタンをクリックします。



デバッグは、実行再開、ステップイン、ステップオーバー、ステップアウトなどの機能を使用してデバッグしてください。詳細はEclipseのヘルプなどを参照してください。

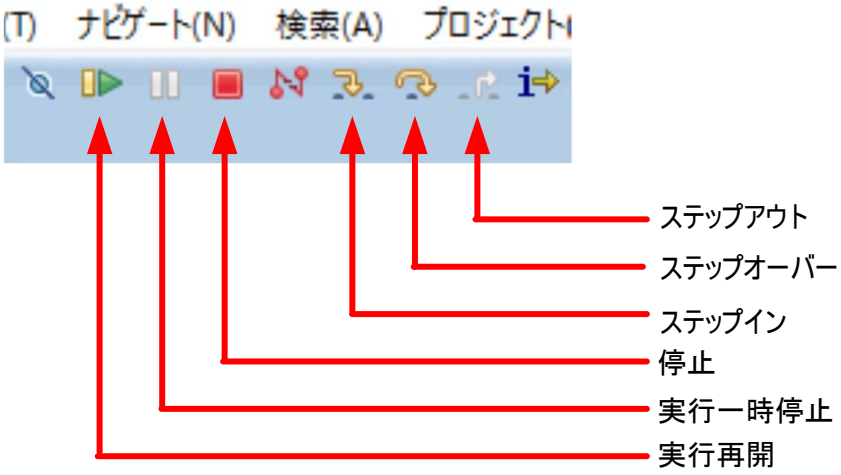


図3-19 デバッグアイコン

②デバッグを終了する場合、停止ボタンで実行を停止し、対象のリモートアプリケーションで右クリックを
行い、ドロップダウンリストメニューから「終了および除去(V)」を選択してください。

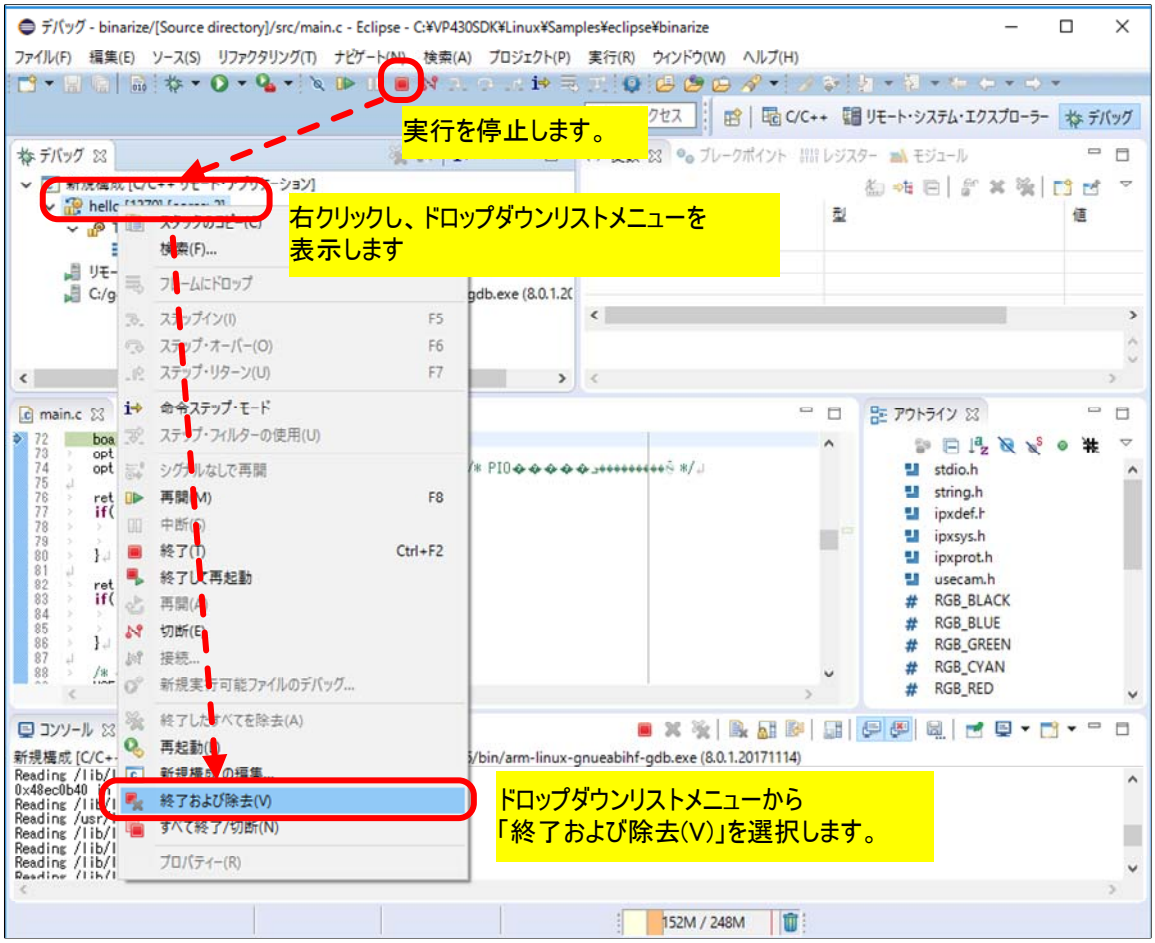


図3-20 デバッグの終了

3.4 Eclipseの環境設定

日本語版Eclipseを使用する場合の環境設定例を示します。設定は必須ではありません。必要に応じて設定してください。

3.4.1 ワークスペースの設定

日本語版Eclipseは、デフォルトで文字コードがUTF-8に設定されています。日本語文字コードとしてSHIFT-JIS等を使用する場合、テキストファイルのエンコードを「MS932」に設定してください。

また、外部編集したソースファイルをeclipseが変更を認識できるようにする場合は、「ネイティブフックまたはポーリングを使用してリフレッシュ」をチェックしてください。

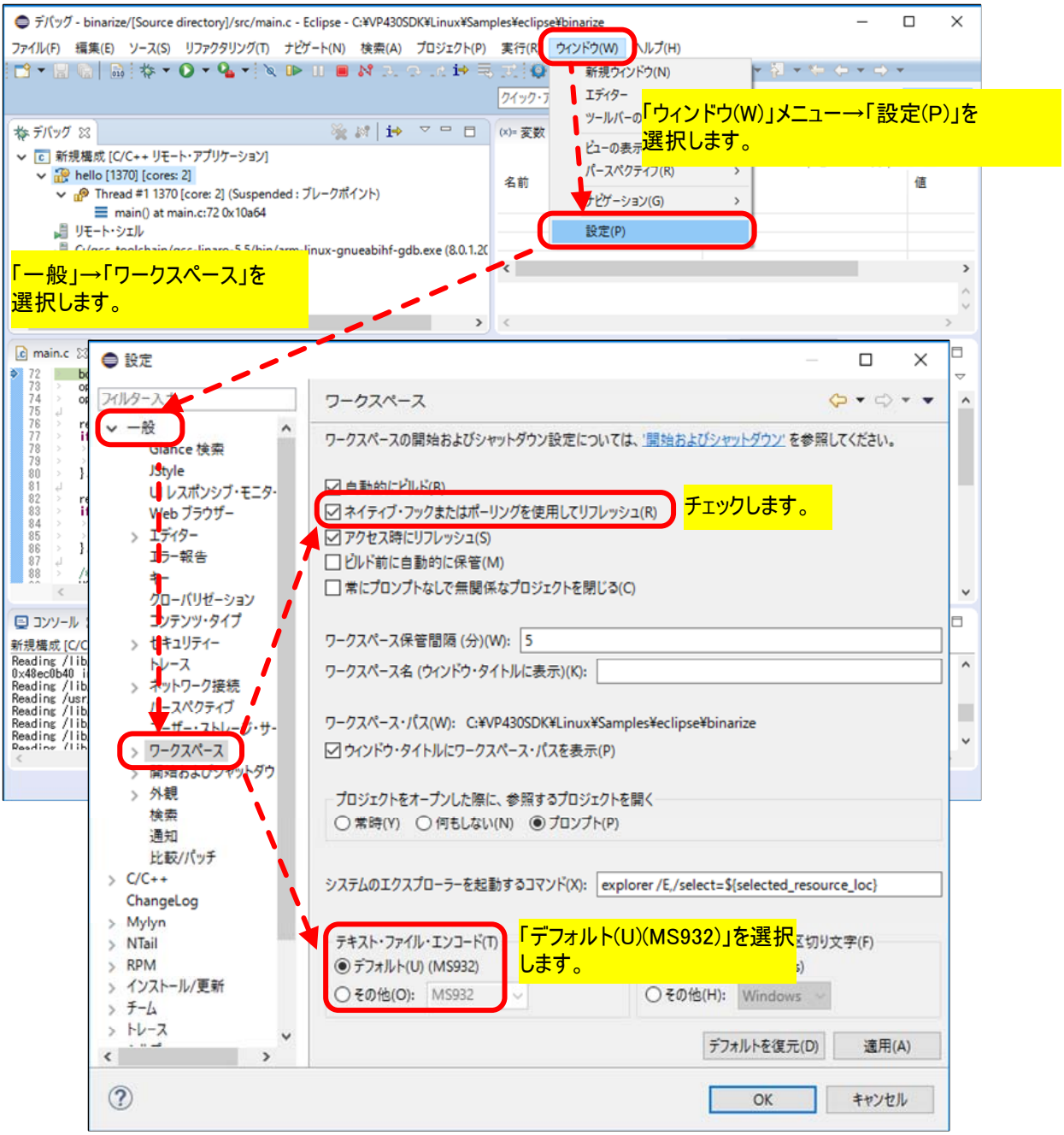


図3-21 ワークスペースの設定

3.4.2 エディタフォントの設定

Eclipseのエディタフォントを変更する場合は、以下を設定してください。

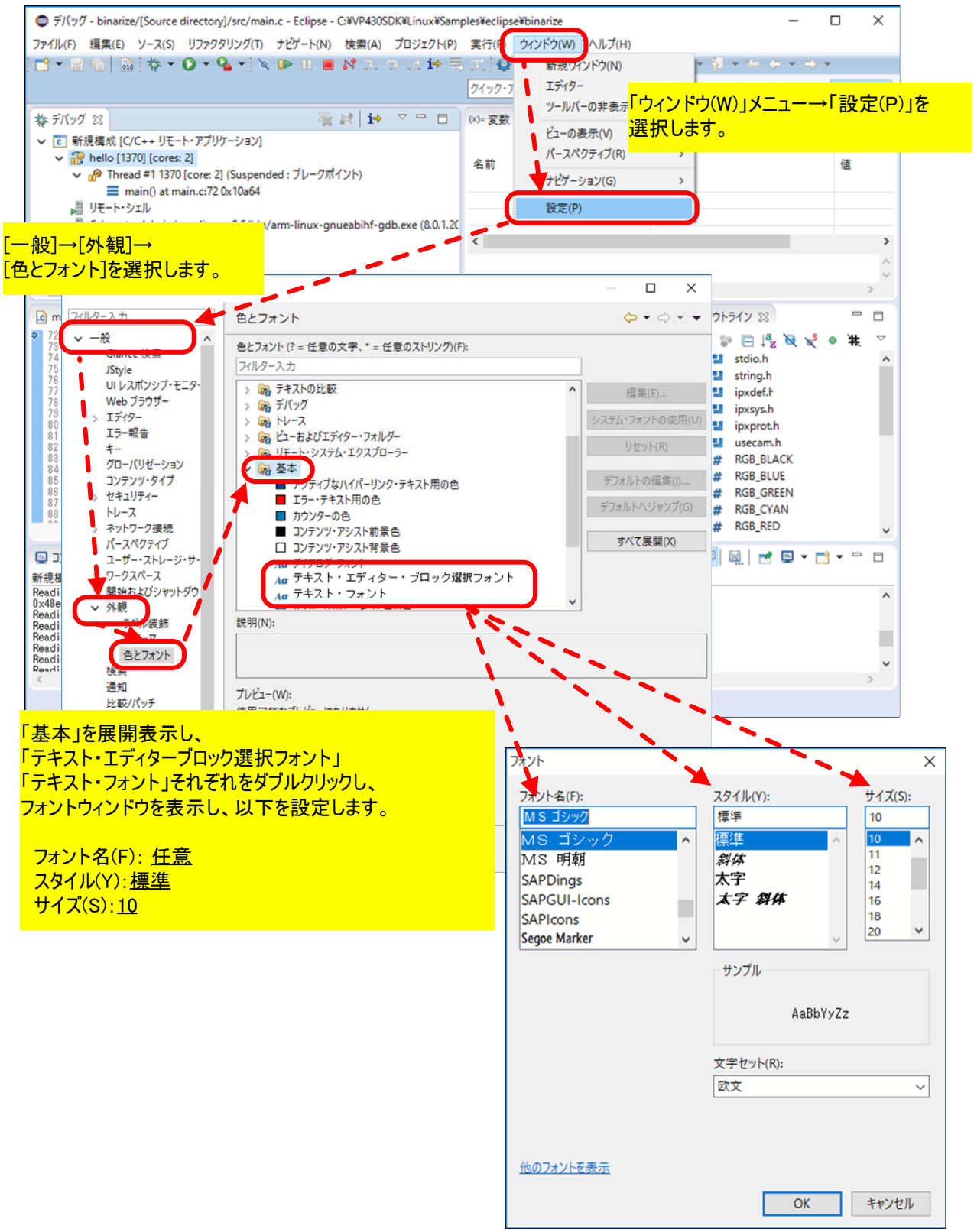


図3-22 エディタフォントの設定

3.4.3 スペルチェックの無効化

Eclipseのエディタのスペルチェックを無効化する場合は、以下を設定してください。

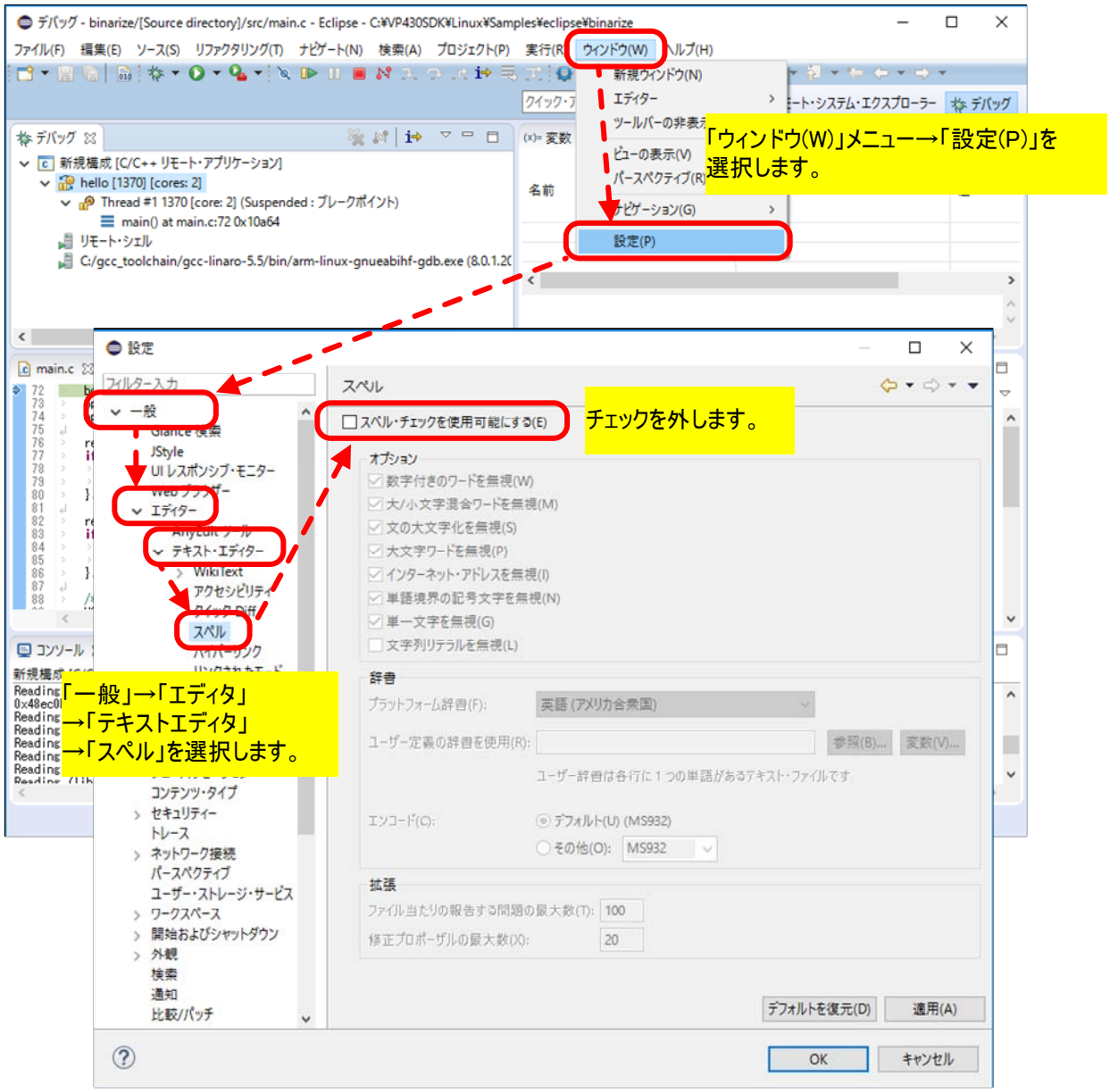


図3-23 スペルチェックの設定

3.5 リモート・システム・エクスプローラ

Eclipseでは、リモート・システム・エクスプローラー（RSE）を使用し、NVP-Linux側システムのファイルをWindowsのエクスプローラのようにアクセスすることができます。設定方法について説明します。

①RSE(Remote System Explorer)を設定します。

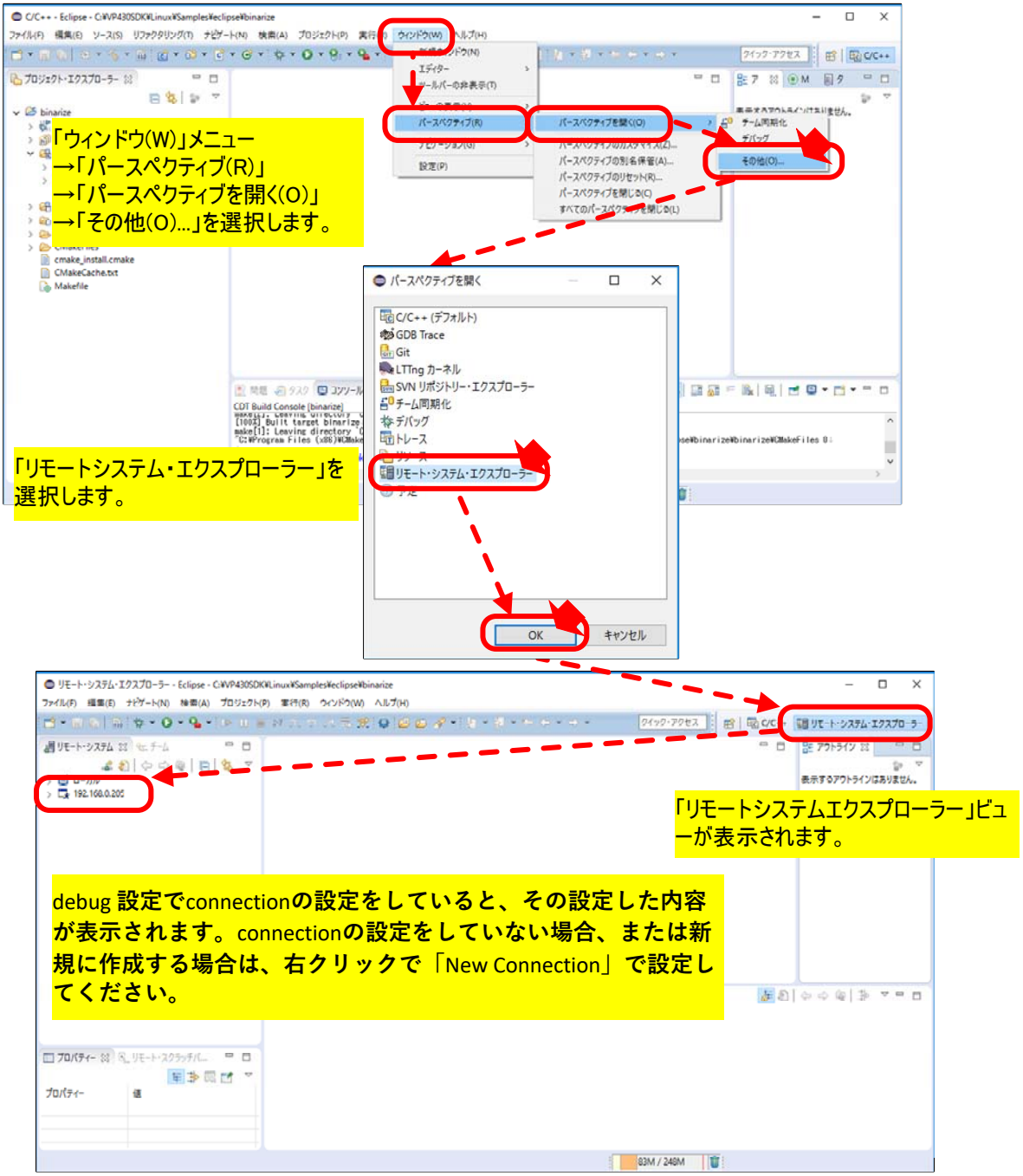


図3-24 RSE(Remote System Explorer)の設定

②「Remote Systems」ビューで、NVP-Linuxのフォルダ情報が表示される事を確認します。

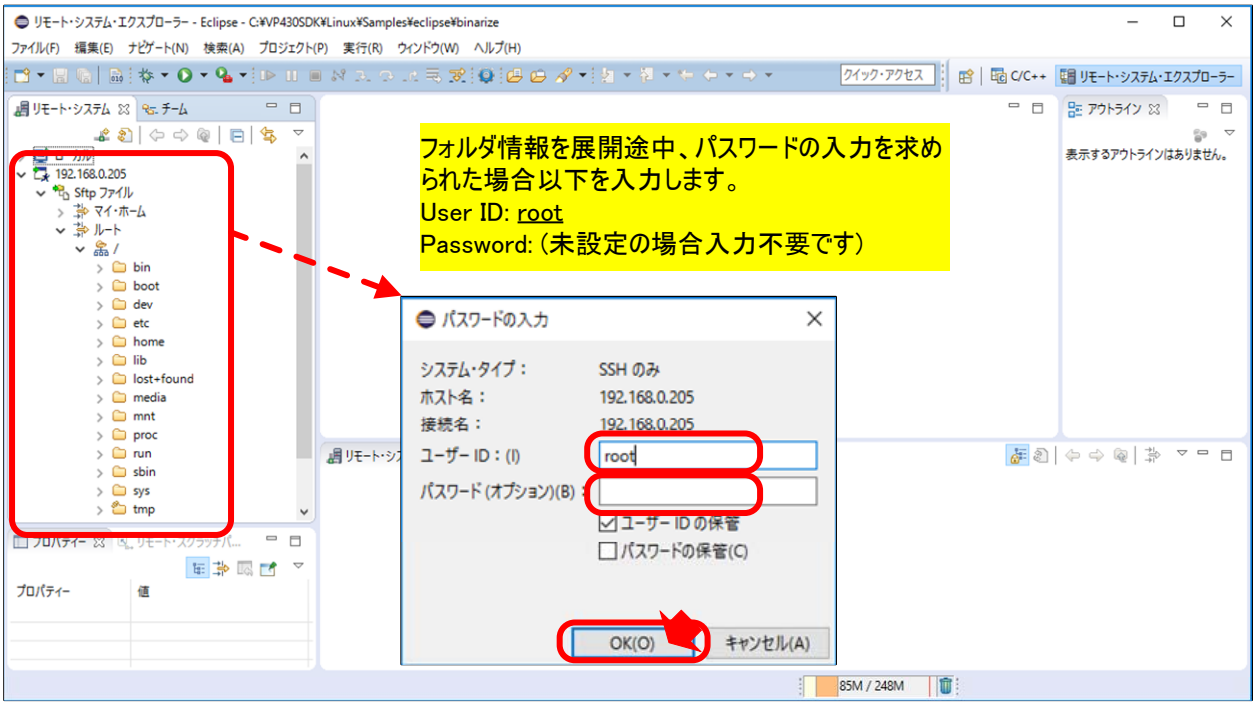


図3-25 フォルダ情報の確認

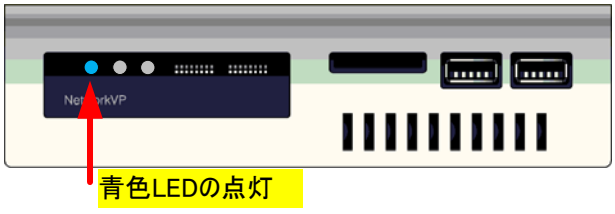
リモート・システム・エクスプローラでWindows上のファイルをNVPーLinuxファイルシステムに転送した場合、ファイル属性が設定されません。転送後、「Remote Systems」ビューで転送したファイルを右クリックし、「プロパティ(R)」から属性を変更するか、または、Linuxシェルから「chmod」コマンドで任意の属性を設定してください。

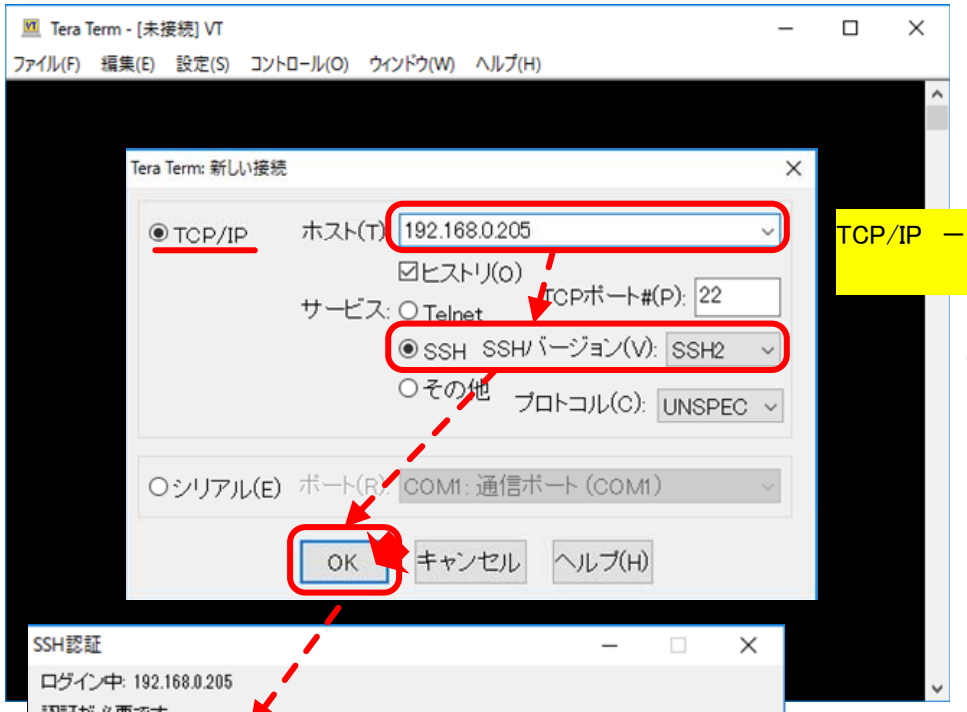
3.6 TeraTermでのSSHサーバーへの接続

NVPのLinuxはSSHサーバーが動作しています。TeraTermのTCP/IP通信機能を使用し、Linuxにログインすることでシェルの操作だけでなく、ファイルの送受信等も行うことが可能です。それらの方法等を説明します。

3.6.1 NVP-Linux SSHサーバへのログイン

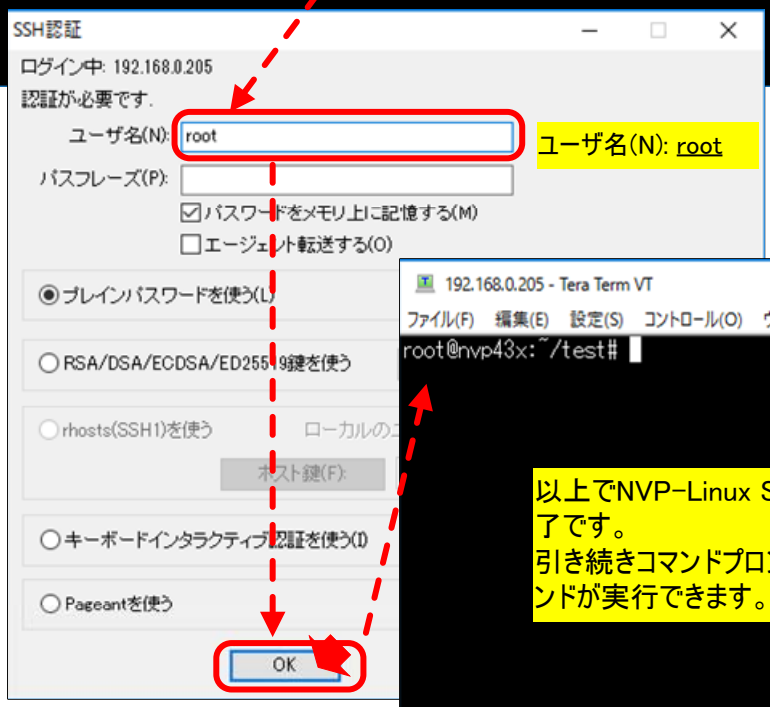
NVPに電源を入れ、筐体前面の青色LEDが点灯後、TeraTermを起動してください。



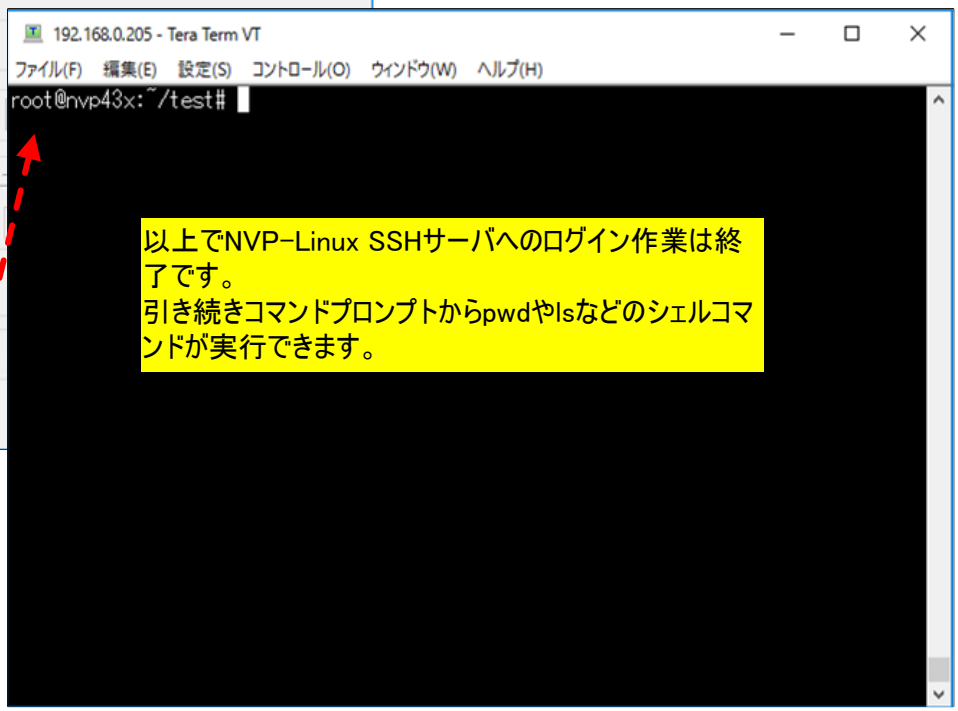


TCP/IP ー ホスト(T) 192.168.0.205
サービス SSHを選択

※ NVPのIPアドレス指定
※ SSHバージョンは任意



ユーザ名(N): root



以上でNVP-Linux SSHサーバへのログイン作業は終了です。
引き続きコマンドプロンプトからpwdやlsなどのシェルコマンドが実行できます。

図3-26 NVP-Linux SSHサーバへのログイン

TeraTermの端末の設定

「設定(S)」メニュー→「端末(T)...」を選び、「Tera Term: 端末の設定」ウィンドウを表示します。

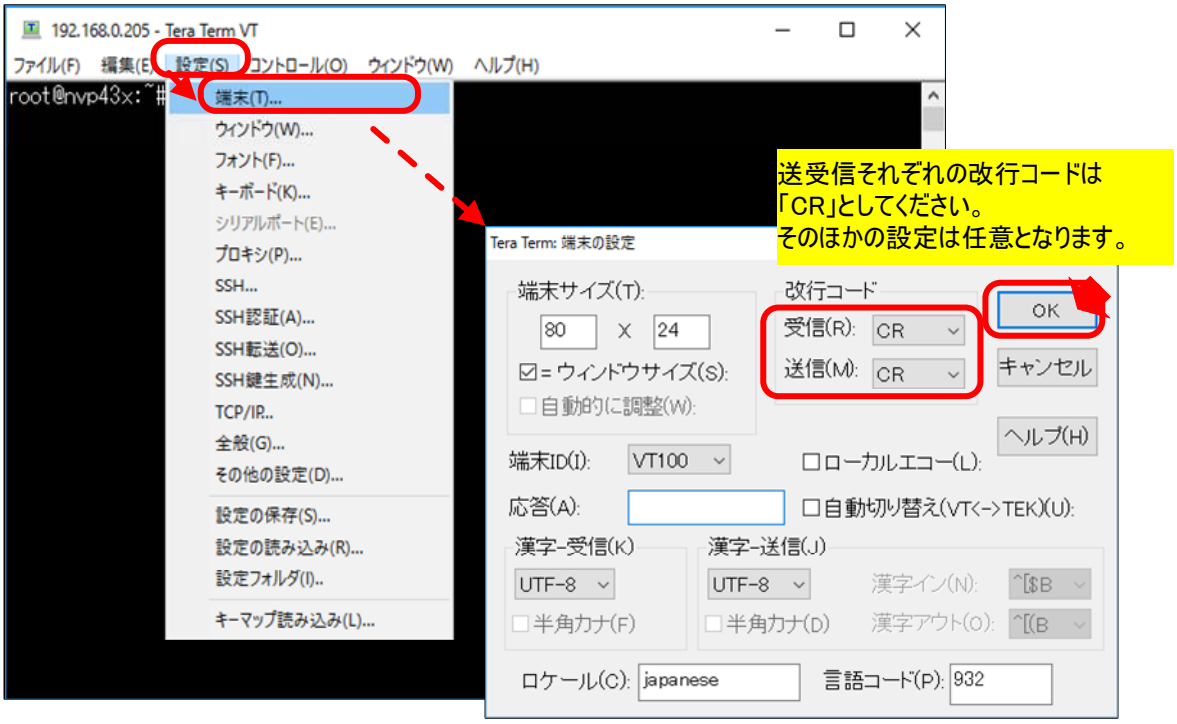


図3-27 端末の設定

3.6.2 Window上のファイルのNVP−Linuxファイルシステムへの転送

TeraTermの画面に転送したいファイルをドラッグ＆ドロップを行います。

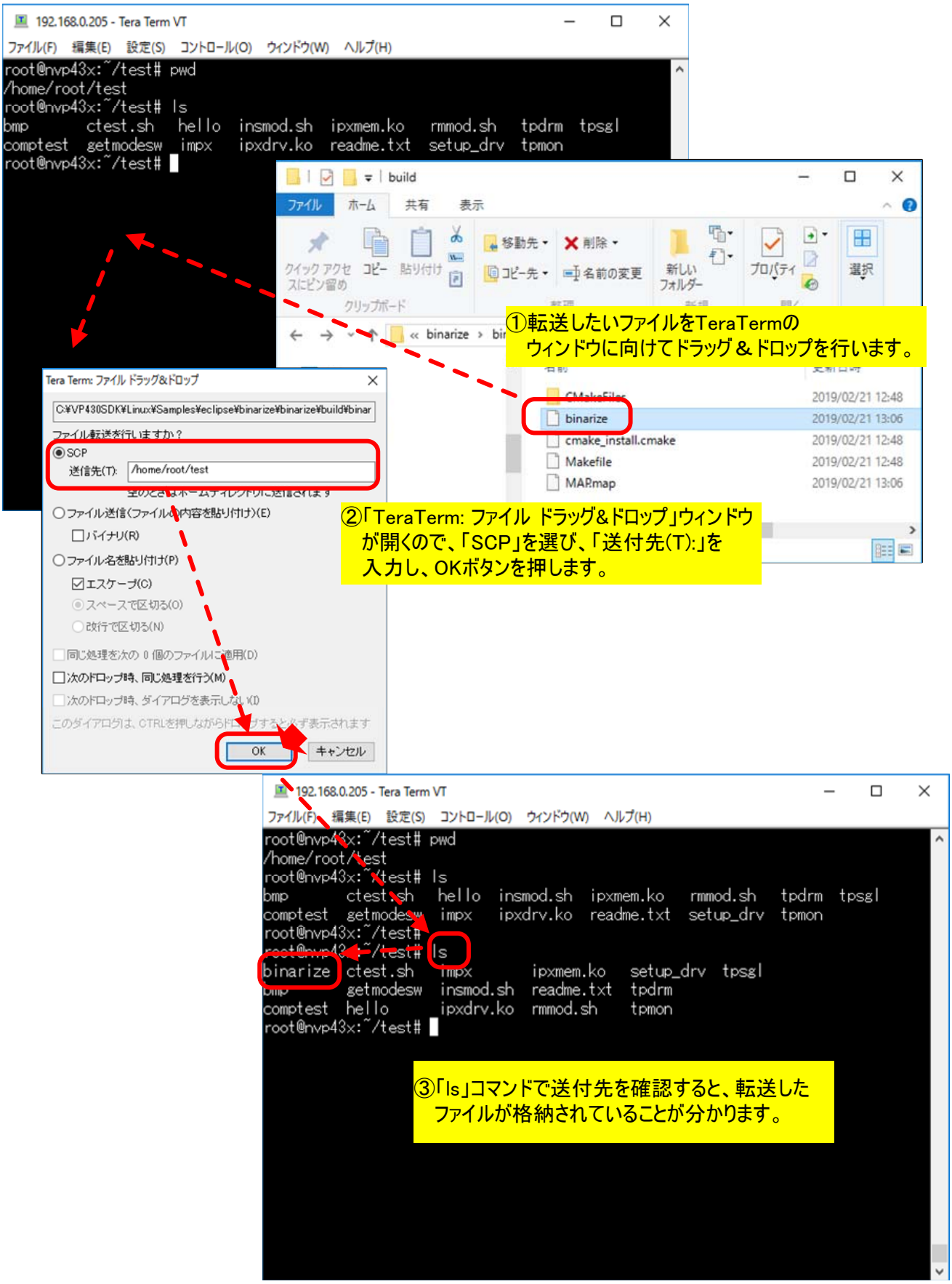


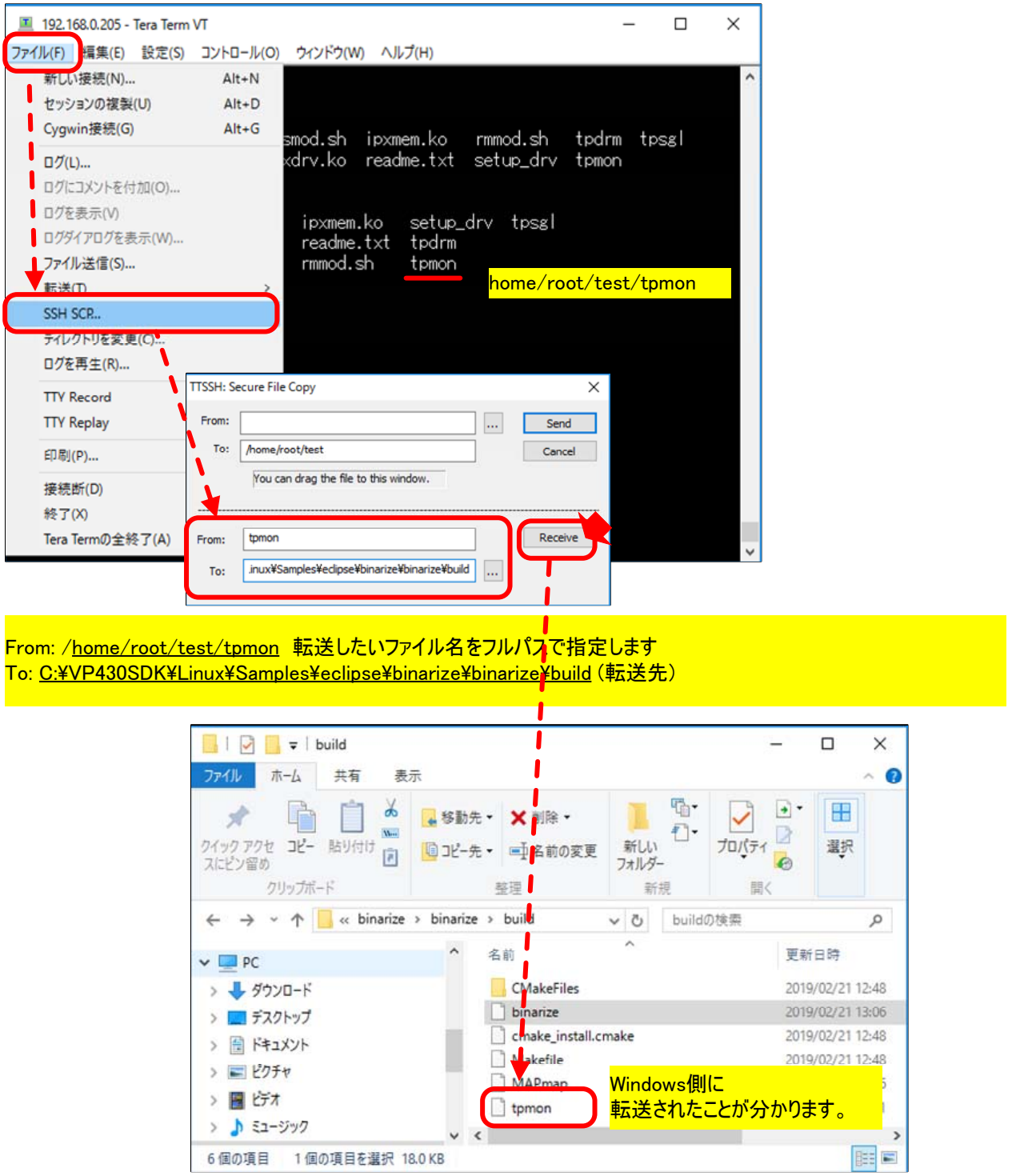
図3-28 Windows上のファイルのNVP−Linuxファイルシステムへの転送

Windows上のファイルをNVP－Linuxファイルシステムに転送した場合、ファイル属性が設定されません。転送後、「chmod」コマンドで任意の属性を設定してください。

3.6.3 NVP-Linux上のファイルのWindowsファイルシステムへの転送

Linux側の「/home/root/test/tpmon」ファイルをWindows側の
「C:¥VP430SDK¥Linux¥Samples¥eclipse¥binarize¥binalize¥build」ディレクトリに転送する例を以下に示します。

TeraTermの「ファイル(F)」メニュー→「SSH_SCR...」を選択します。



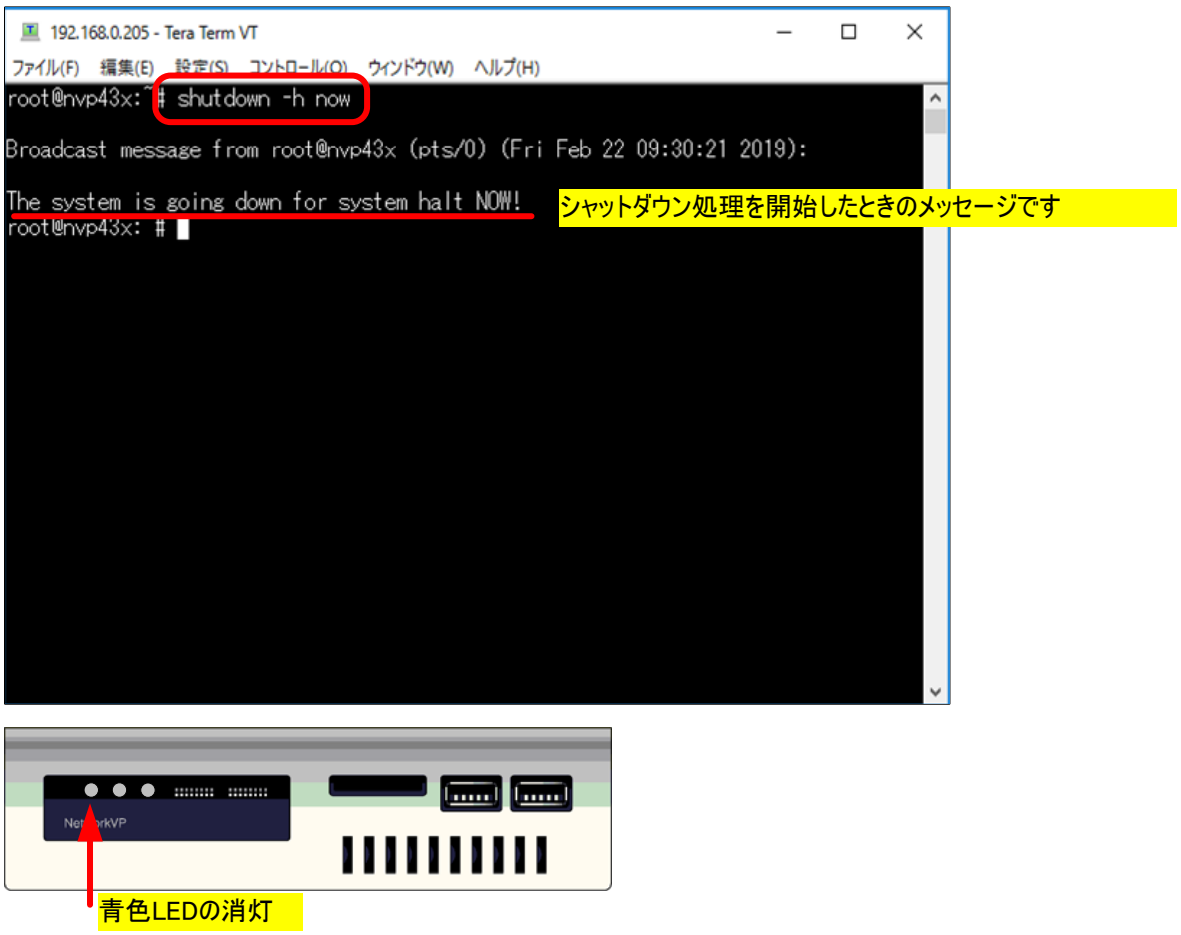
From: /home/root/test/tpmon　転送したいファイル名をフルパスで指定します
To: C:¥VP430SDK¥Linux¥Samples¥eclipse¥binarize¥binarize¥build（転送先）

図3-29 NVP-Linux上のファイルのWindowsファイルシステムへの転送

※TeraTermのSCP転送を使ってNVP-Linux上のファイルをWindowsへ転送した場合、TeraTermがファイルのアクセス権を解放しない場合があります。この場合TeraTermを終了することでファイルのアクセス権が解放されます。

3.6.4 NVP-Linuxのシャットダウン

Linuxのコンソールから「shutdown -h now」と入力します。



筐体前面の青色LEDが消灯の後、NVPの電源をOFFにすることができます。

図3-30 NVP-Linuxのシャットダウン

3.7 TeraTermでのシリアルターミナル

TeraTermのシリアル通信機能を使ったNVPとの接続を説明します。
なお、事前にNVPとパソコンをシリアルケーブルで接続してください。

図3-31 シリアルポートの接続

3.7.1 TeraTermの設定

TeraTermを立ち上げ、シリアル接続の選択を行います。

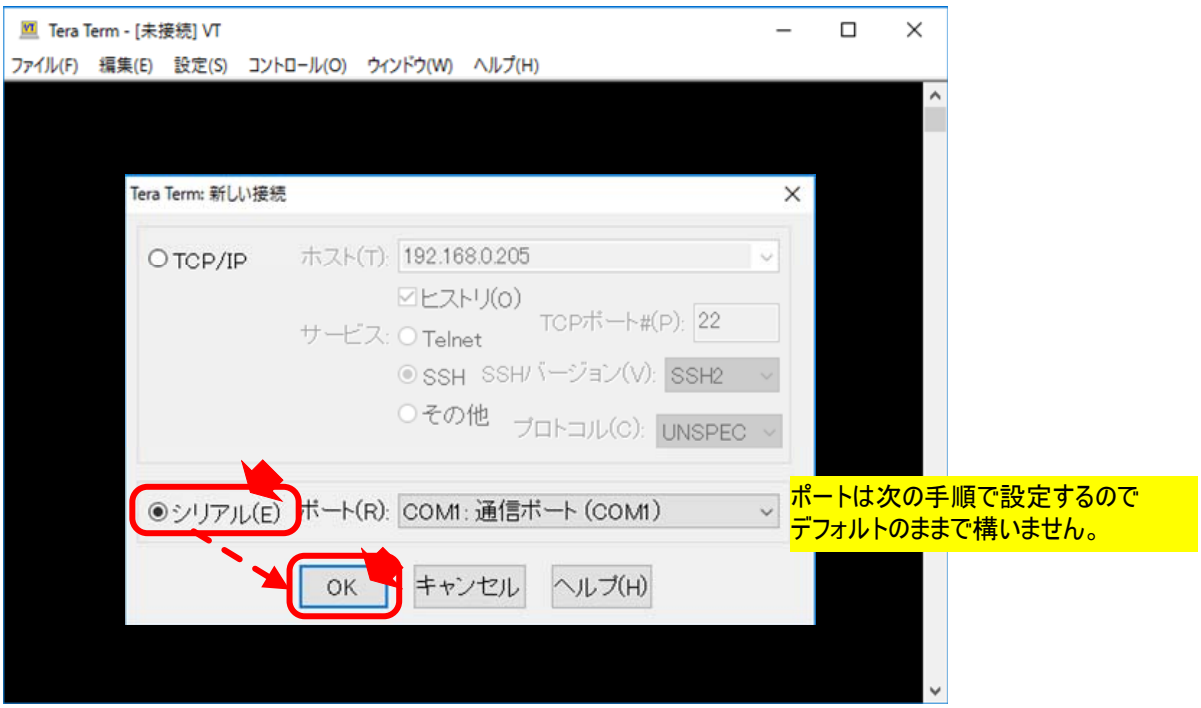
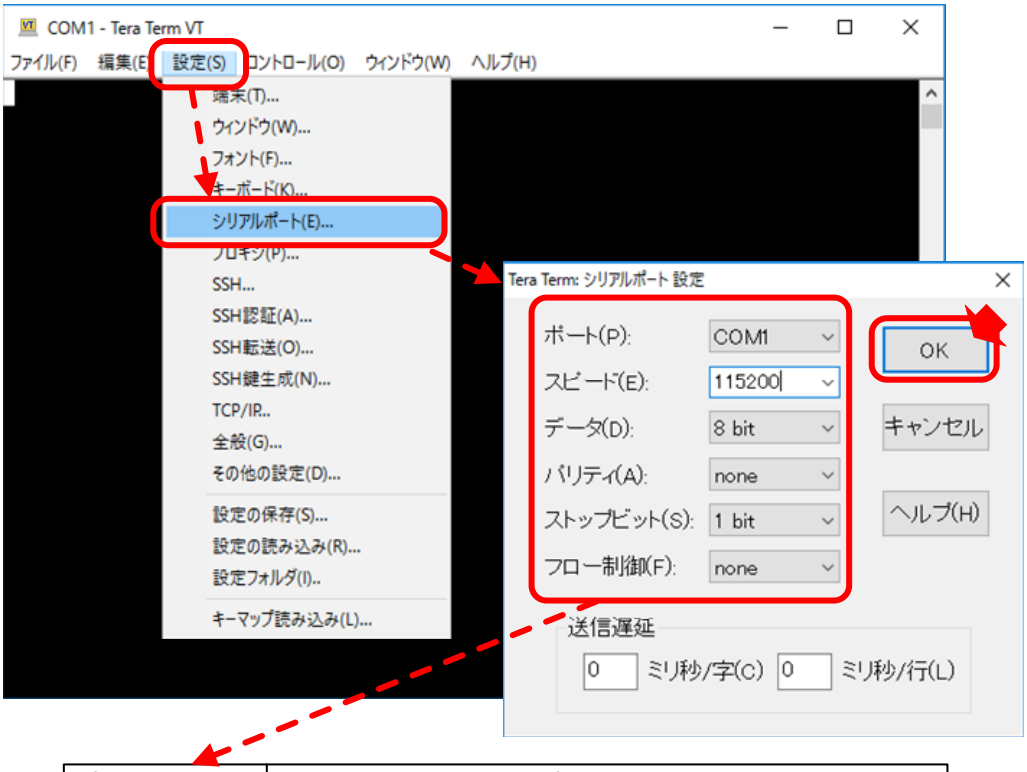


図3-32 接続方法の選択

「設定(S)」メニュー→「シリアルポート(E)...」を選択して「Tera Term: シリアルポート設定」ウィンドウを表示します。



ポート(P):	NVPと接続しているCOMポートを選択してください
スピード(E):	115200
データ(D):	8 bit
パリティ(A):	None
ストップビット(S):	1 bit
フロー制御(F):	none

図3-33 シリアルポートの設定

3.7.2 ログインシェルへのログイン

NVPの電源をONにすると、TeraTermの画面に起動メッセージが表示された後、ログイン入力画面となります。ログインIDに「root」と入力してNVP-Linuxのログインシェルにログインを行ってください。

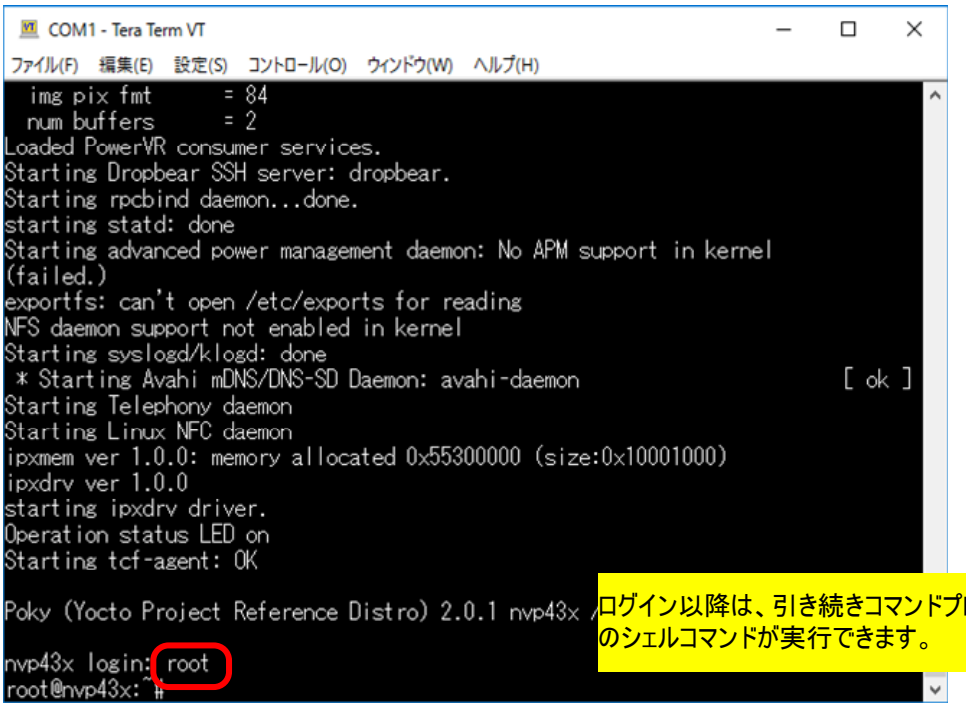


図3-34 Linuxシリアルシェルサーバーへのログイン

Note

Linuxサーバ起動後に表示される次のメッセージは、エラーメッセージではありません。

```
nvp43x login: root
root@nvp43x:~# random: nonblocking pool is initialized
```

「random: nonblocking pool is initialized」

このメッセージは、/dev/urandom が内部的に使用するプール初期化完了を示します。

3.7.3 NVP-Linuxのシャットダウン

Linuxのコンソールから「shutdown -h now」と入力します。

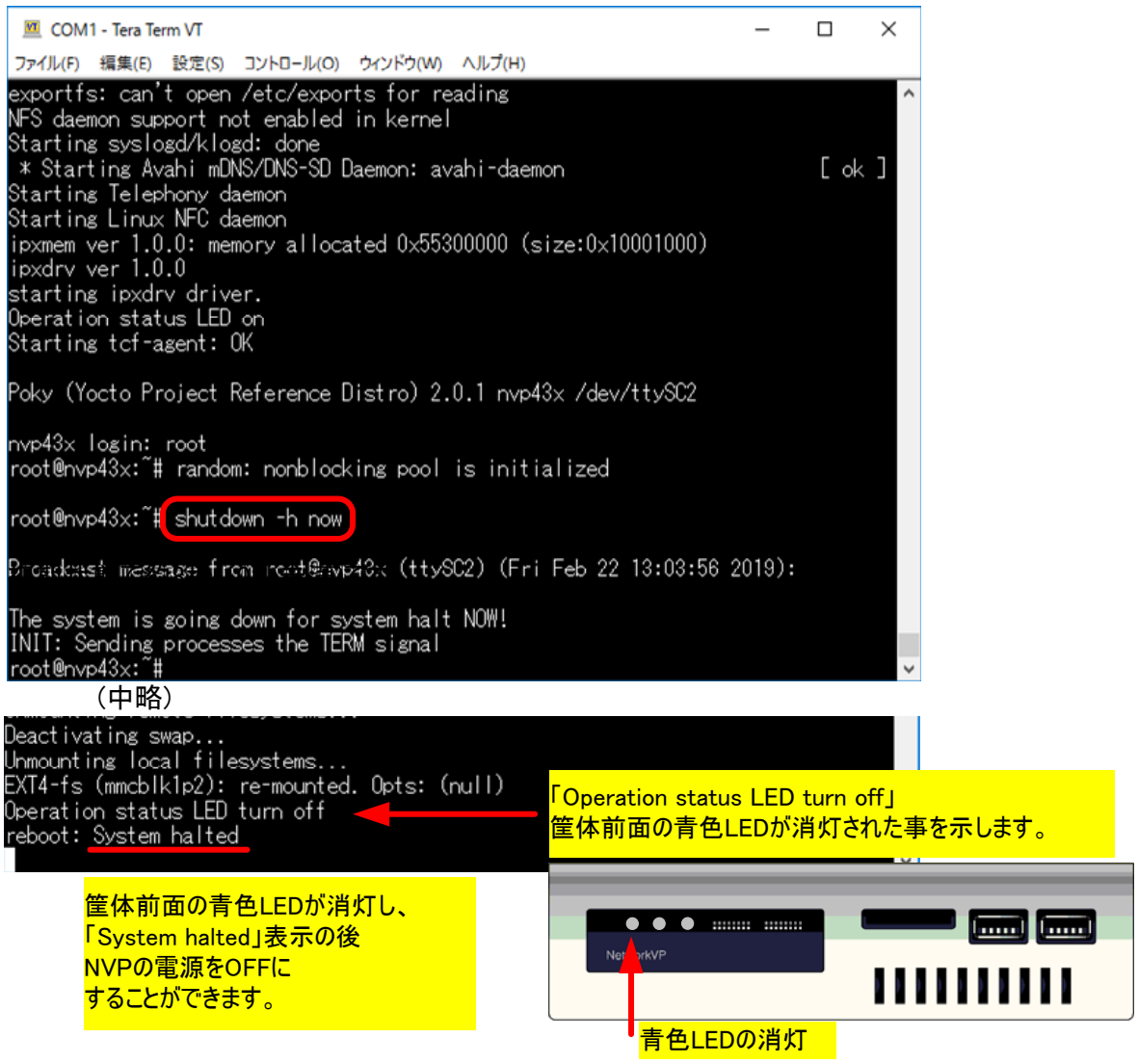


図3-35 NVP-Linuxのシャットダウン

3.8 PCリモートコマンドで動作するサンプルアプリケーション

Windowsアプリケーションを作成する場合、Microsoft Visual Studio 2013 Professional Visual C++(Visual C)用のリモートコマンドライブラリを使用します。NVP-Ax430SDKで提供しているリモートコマンドのインクルードファイルパスの設定とライブラリのリンク設定が必要です。本説明ではPC側リモートコマンド動作のサンプルアプリケーション構築(デバイスID付き)とデバッグ開始までの説明を行います。

本説明で使用するVisual C実行環境は下記の通りです。

Microsoft Visual Studio Professional 2013 (Visual C++ 2013)
Update 5 + MBCS(マルチバイト文字対応)MFCライブラリ

なお、Visual C の操作方法に関しての詳細は、Visual C の説明書を参照してください。

3.8.1 PCリモートコマンド動作時のDIPSW(SW1)の設定

PC側リモートコマンド動作を行う場合、DIPSW（SW1）のNo1のスイッチをON、No2～No8のスイッチをOFFにしてNVPを起動してください。

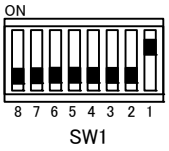


図3-36 DIPSW(SW1)設定(PCリモートコマンド動作時)

3.8.2 PCリモートコマンド動作用サンプルプロジェクトの構成

本説明で使用するプロジェクトはSDKにサンプルとして提供しており、「図1-1 SDKフォルダ構成」の「VC¥Samples」および「VC¥Inifile」以下のファイルとなります。サンプルプロジェクトの構成を図3-37に示します。

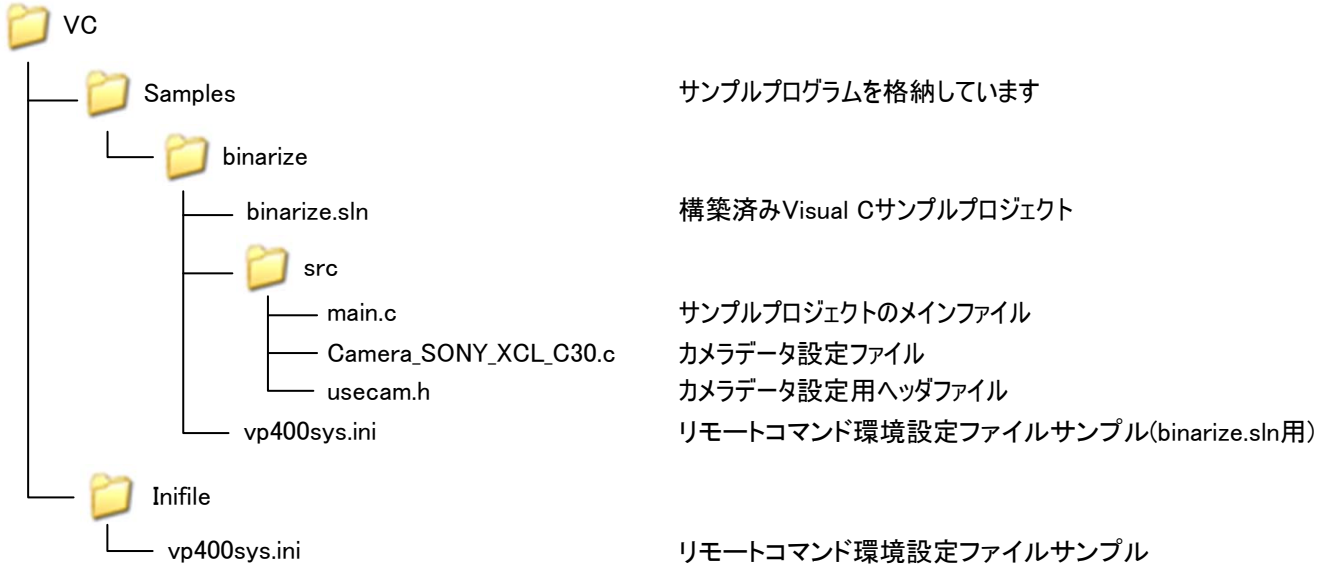


図3-37 PC版サンプルアプリケーションフォルダ構成

本説明では「図3-37 PC版サンプルアプリケーションフォルダ構成」に示すソースファイルを使用したVisual Cプロジェクトの構築手順を説明するため、VCフォルダ以下に新規に「Tutorial」フォルダを用意します。
本手順が不要な場合は、構築済みの「binarize.sln」プロジェクトを使用してください。

3.9 PC版アプリケーションのコンパイルとリンク

PC版リモートコマンドAPIはリモートコマンドで次の2種類のインタフェースを用意しています。
(1)ユニットを識別するためのユニット識別子(デバイスID)無しのAPI
(2)ユニットを識別するためのユニット識別子(デバイスID)付きのAPI
本サンプルでは「(2)ユニットを識別するためのユニット識別子(デバイスID)付きのAPI」を使用したサンプルアプリケーションの構築を行います。Visual Cでプログラムをコンパイル、リンクする場合、下記の要領でインクルードファイルのディレクトリパスの設定とライブラリのプロジェクトへの追加を行ってください。

※チュートリアルのフォルダ構成は「C:¥VP430SDK」を起点として説明します。
必要に応じ、SDKインストール時の設定に従ったフォルダ構成に読み替えてください。

3.9.1 Visual Cによる新規プロジェクトの作成

①チュートリアル用のプロジェクトフォルダとして、「VC」以下に「Tutorial」フォルダを新たに作成してください。

図3-38 プロジェクトフォルダの新規作成

②Visual Cを起動し「ファイル(F)」から「新規作成(N)」→「プロジェクト(P)...」を選択します。

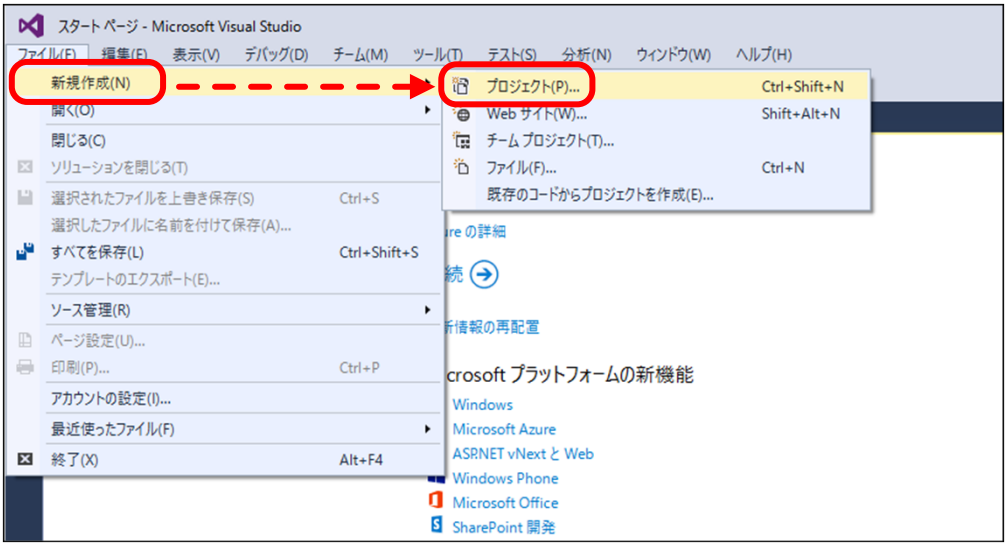


図3-39 プロジェクトの新規作成

③”①”で作成した「Tutorial」フォルダを選択し「フォルダーの選択」をクリックしてください。

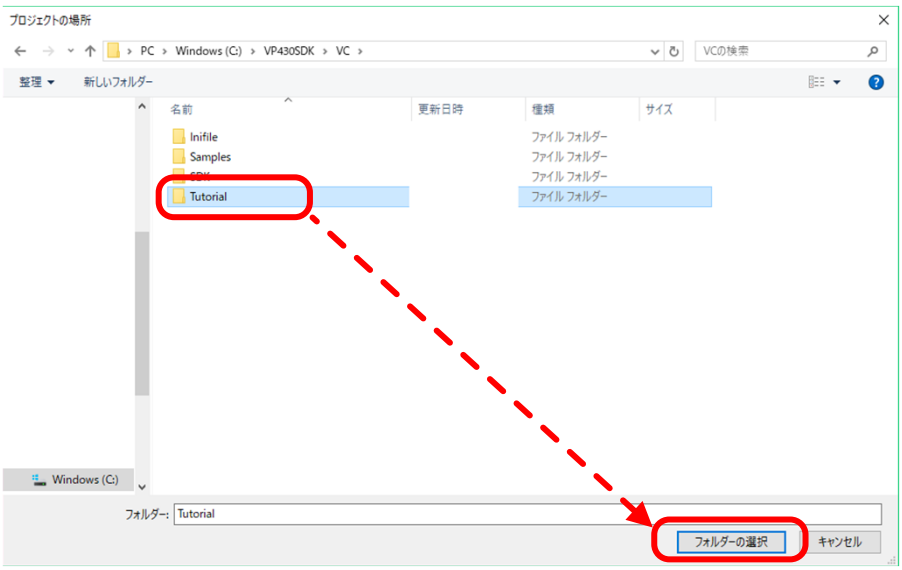


図3-40 プロジェクトフォルダの選択

④「インストール済み」→「テンプレート」→「Visual C++」→「Win32」から「Win32 コンソールアプリケーション」を選択し、
「名前(N):」にプロジェクト名を入力してください。(※本説明では「binarize」とします。)
「ソリューションのディレクトリを作成(D)」のチェックを外して「OK」をクリックしてください。

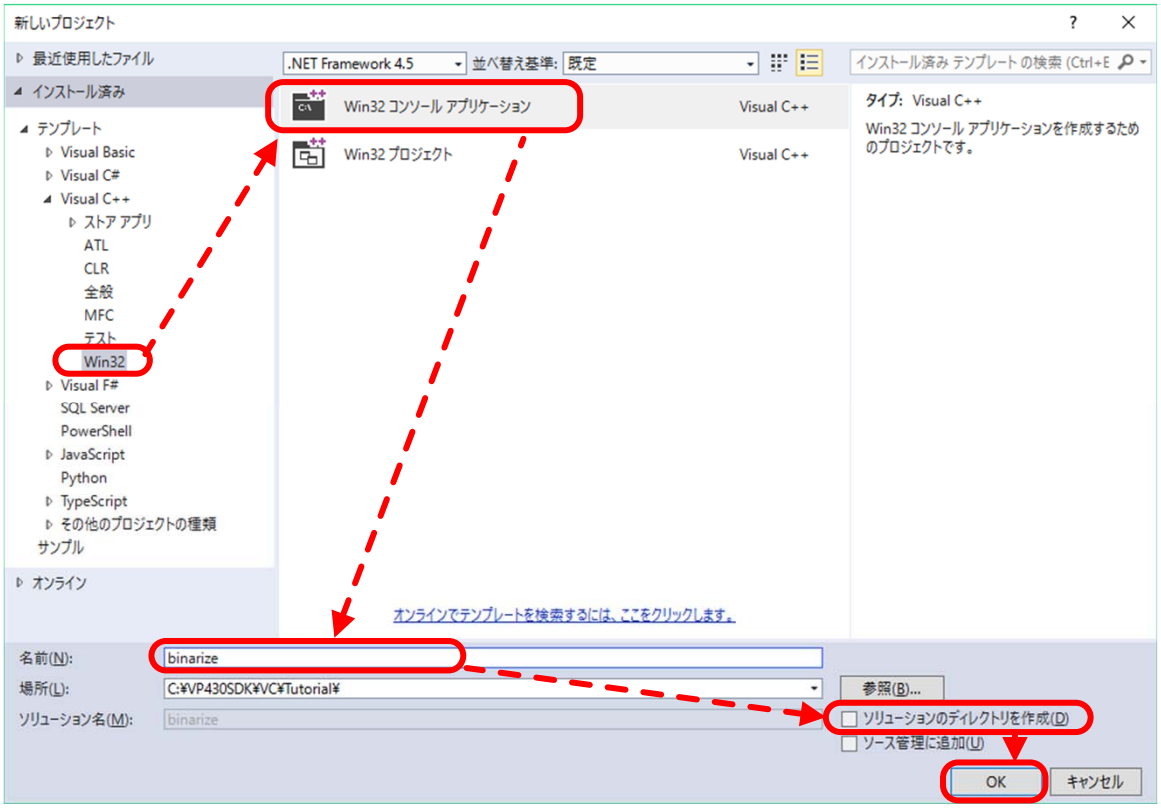


図3-41 プロジェクトの作成(1/3)

⑤「次へ >」をクリックしてください。

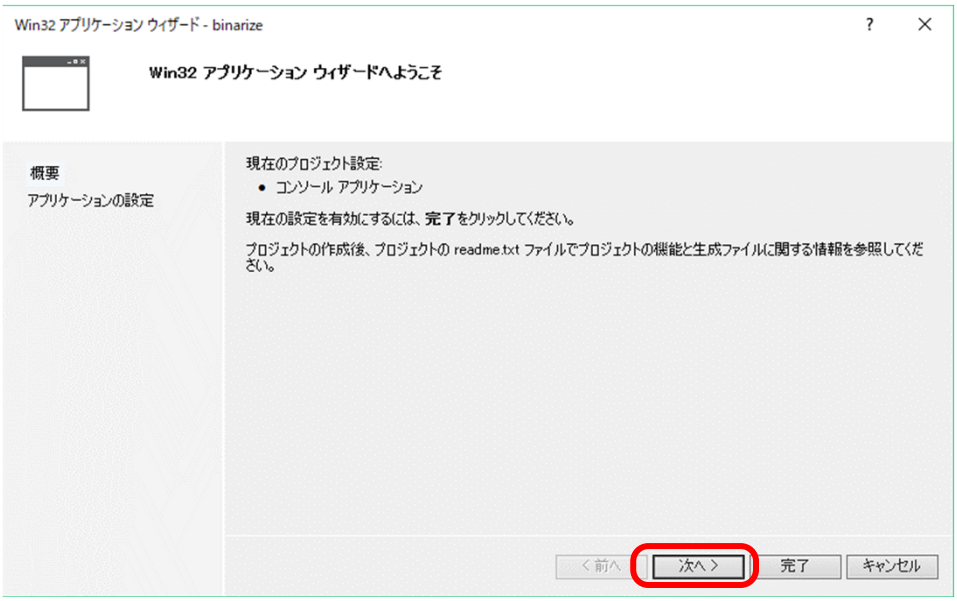


図3-42 プロジェクトの作成(2/3)

⑥「コンソールアプリケーション(O)」と「空のプロジェクト(E)」を選択し、
「Security Development Lifecycle(SDL)チェック(C)」のチェックを外し「完了」をクリックしてください。

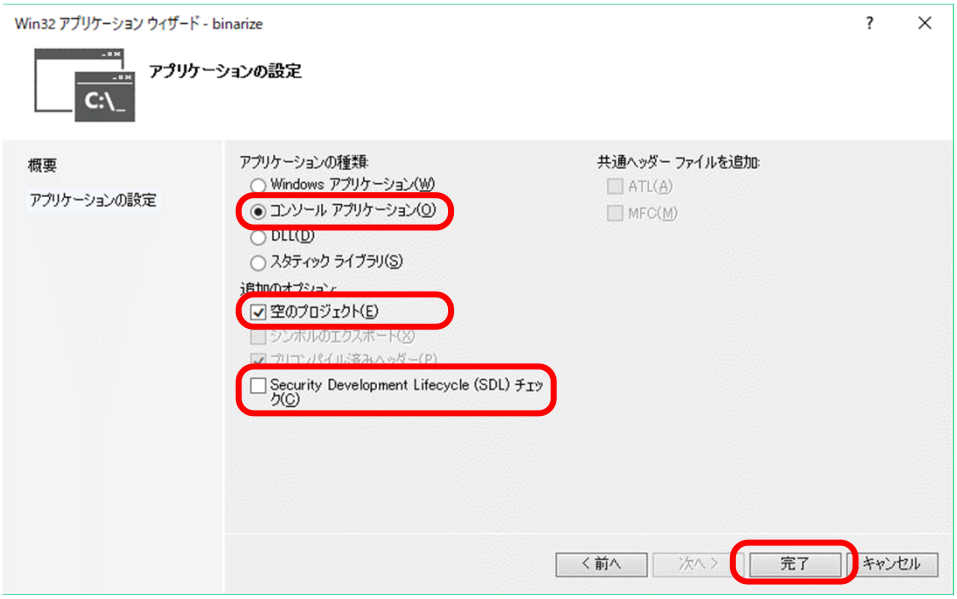


図3-43 プロジェクトの作成(3/3)

Tutorialフォルダの中に”④”で入力した名前のフォルダ(binarizeフォルダ)が生成され、
Visual Cのソリューションエクスプローラーに作成したプロジェクトが表示されたら本作業は終了です。

3.9.2 プロジェクトへのサンプルソースコード登録

①「VC¥Tutorial¥binarize」フォルダに「VC¥Samples¥binarize¥src」をフォルダごとコピーしてください。

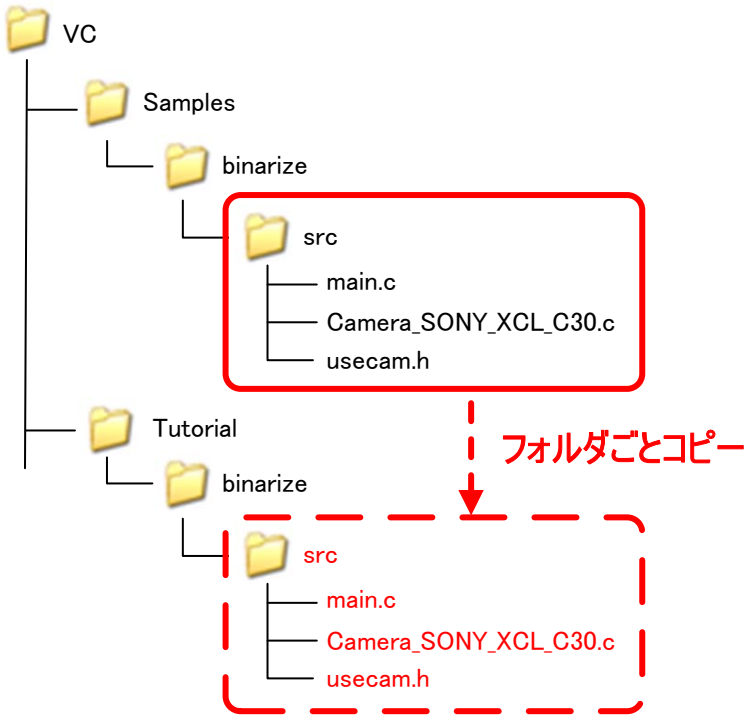


図3-44 サンプルソースコード準備

②「VC¥Tutorial¥binarize」フォルダに「VC¥Inifile」フォルダの「vp430sys.ini」をコピーしてください。

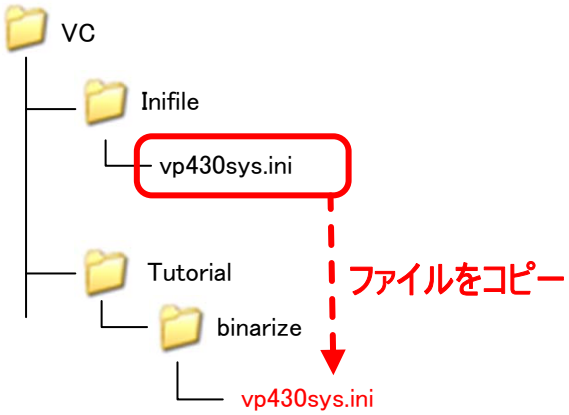


図3-45 リモートコマンド環境設定ファイル準備

③Visual Cのソリューションエクスプローラーから「ソースファイル」を右クリックし「追加(D)」→「既存の項目(G)...」をクリックしてください。

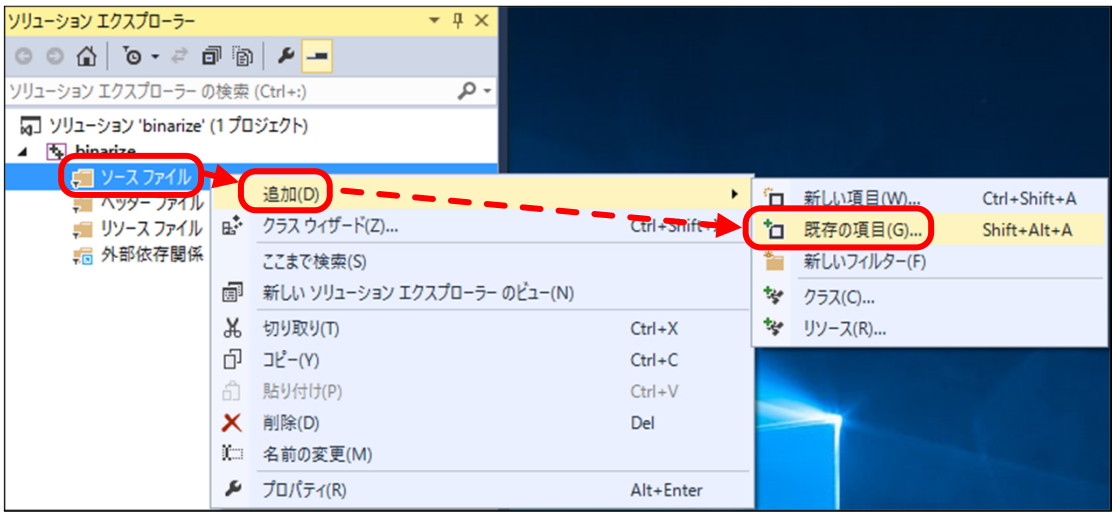


図3-46 ソースコードの登録(1/3)

④「VC¥Tutorial¥binarize¥src」フォルダ内のファイルをすべて選択し「追加(A)」をクリックしてください。

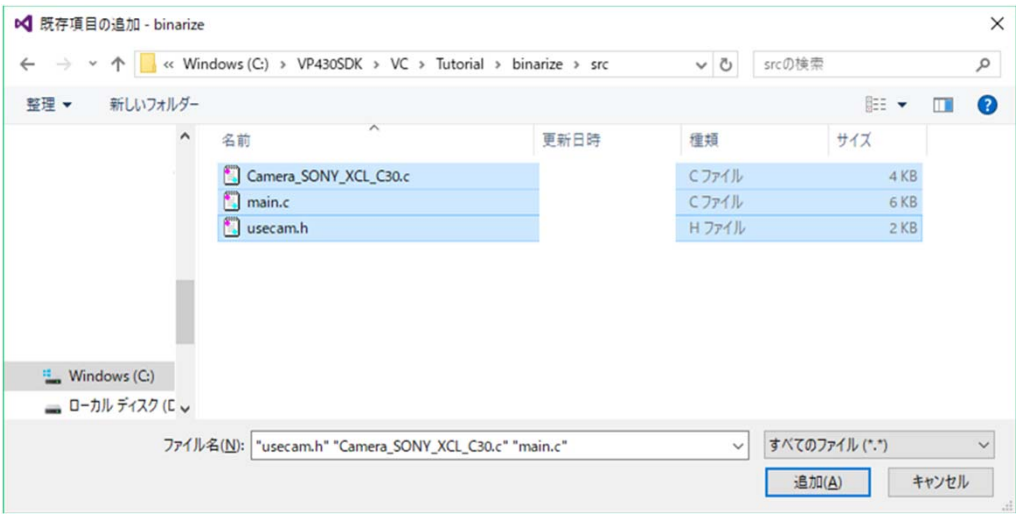


図3-47 ソースコードの登録(2/3)

Visual Cのソリューションエクスプローラーにソースコードが登録されたら本作業は終了です。

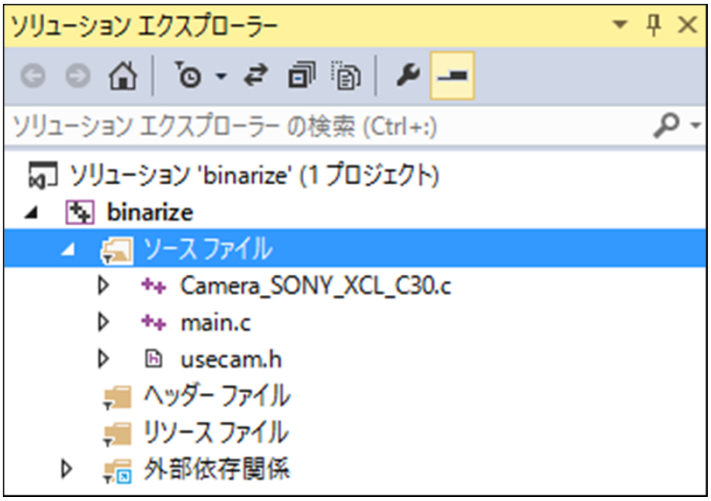


図3-48 ソースコードの登録(3/3)

3.9.3 文字セットの設定

①ソリューションエクスプローラーから「binarize」プロジェクトを右クリックし「プロパティ(R)」をクリックしてください。

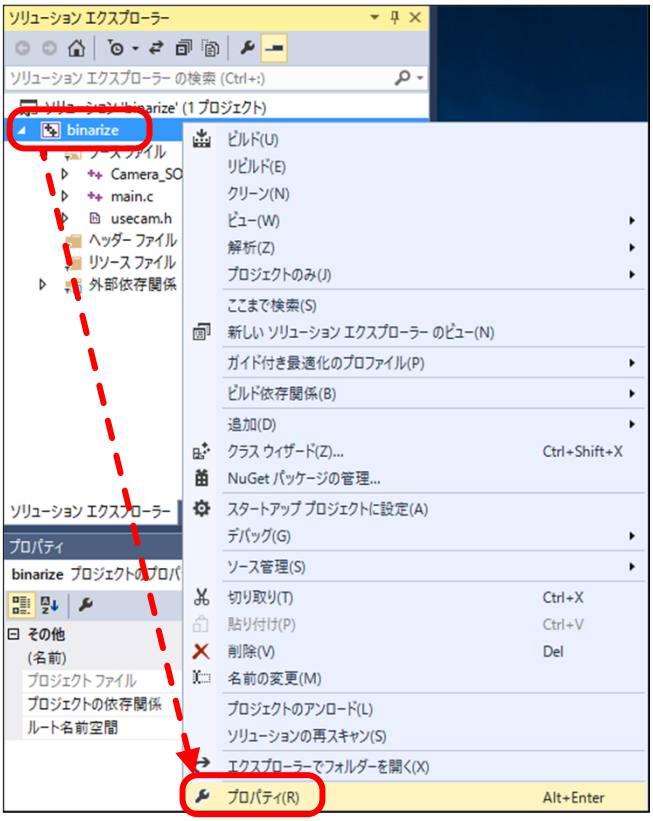


図3-49 文字セットの設定(1/2)

②「構成プロパティ」→「全般」→「文字セット」の項目から「<編集....>」をクリックし、
「**マルチバイト文字セットを使用する**」を選択し「適用(A)」をクリックしてください。

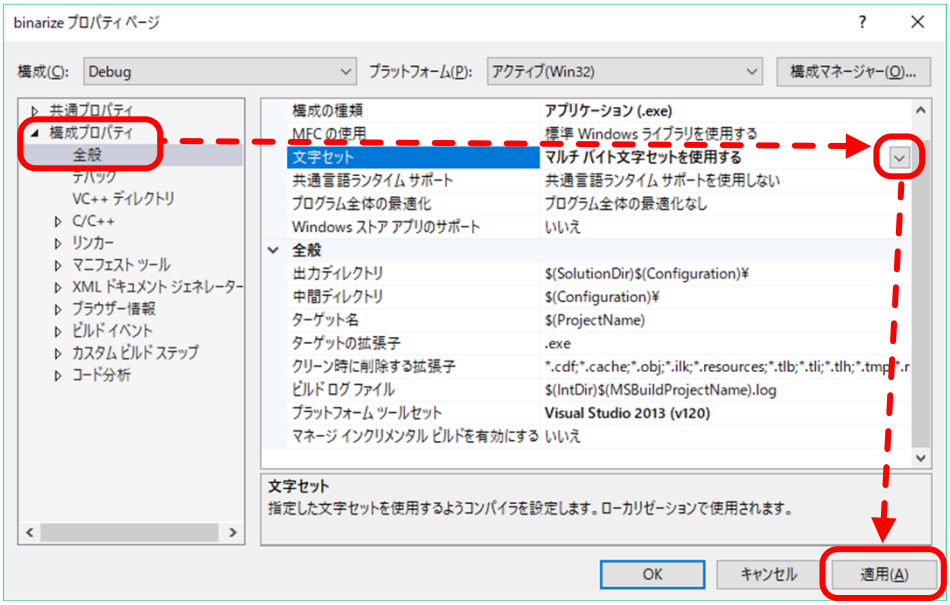


図3-50 文字セットの設定(2/2)

3.9.4 インクルードパスの設定

①「構成プロパティ」→「C/C++」→「全般」→「追加のインクルードディレクトリ」の項目から「<編集...>」をクリックし、フォルダのアイコンから「**C:¥VP430SDK¥VC¥SDK¥inc**」を選択し「OK」の後に「適用(A)」をクリックしてください。

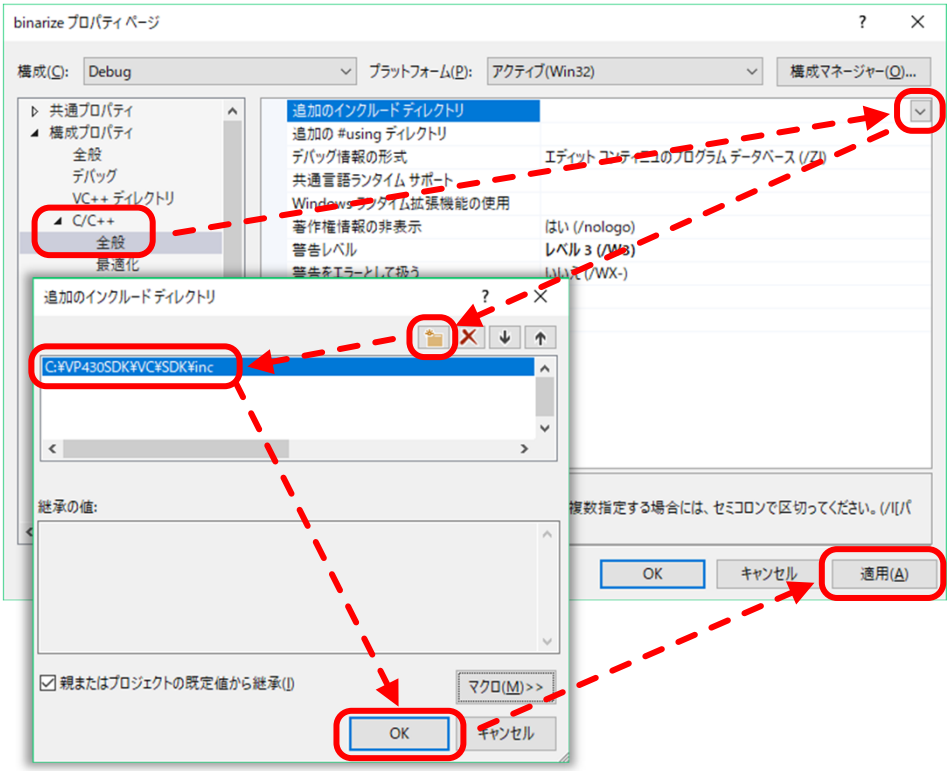


図3-51 インクルードディレクトリの設定

3.9.5 プリプロセッサの設定

①「構成プロパティ」→「C/C++」→「プリプロセッサ」→「プリプロセッサの定義」の項目から「<編集...>」をクリックし、「**MULTI_BOARD_CONFIG**」を追記し「OK」の後に「適用(A)」をクリックしてください。
※必要に応じて「**_CRT_SECURE_NO_DEPRECATE**」を追記してください。これはコンパイル時のC4996警告を抑止するためのマクロ定義です。またはアプリケーションをC4996準拠で作成してください。

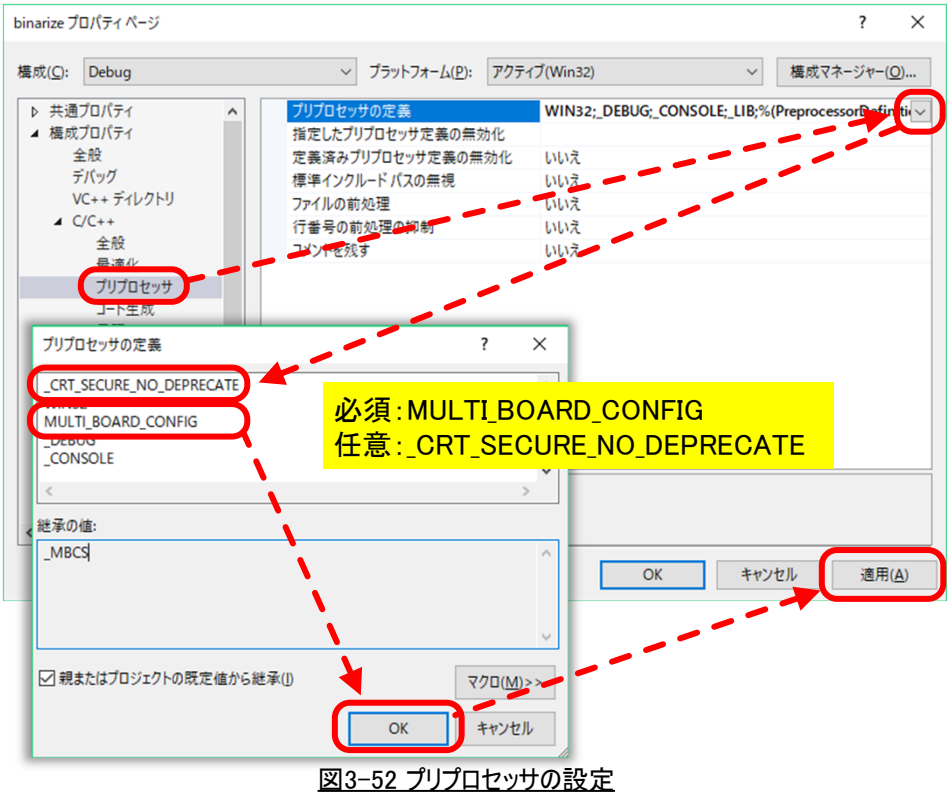


図3-52 プリプロセッサの設定

3.9.6 ライブラリパスの設定とライブラリの登録

①「構成プロパティ」→「リンカー」→「全般」→「追加のライブラリディレクトリ」の項目から「<編集...>」をクリックし、フォルダのアイコンから「**C:¥VP430SDK¥VC¥SDK¥lib**」を選択し「OK」の後に「適用(A)」をクリックしてください。

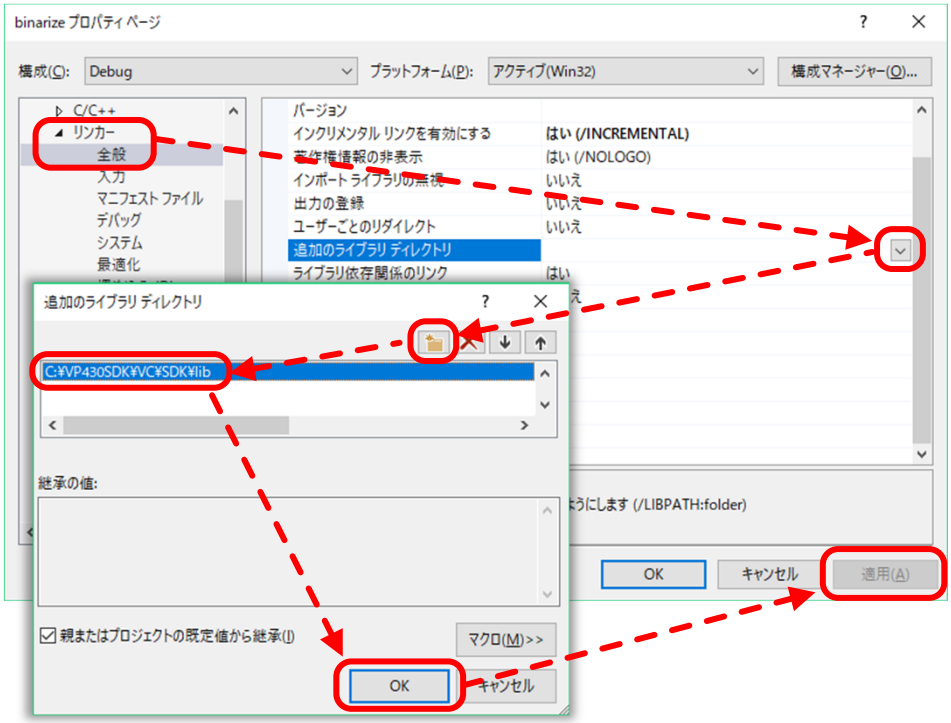


図3-53 ライブラリパスの設定

②「構成プロパティ」→「リンカー」→「入力」→「追加の依存ファイル」の項目から「<編集...>」をクリックし、「**vp400mul.lib**」を追記し「OK」の後に「適用(A)」をクリックしてください。

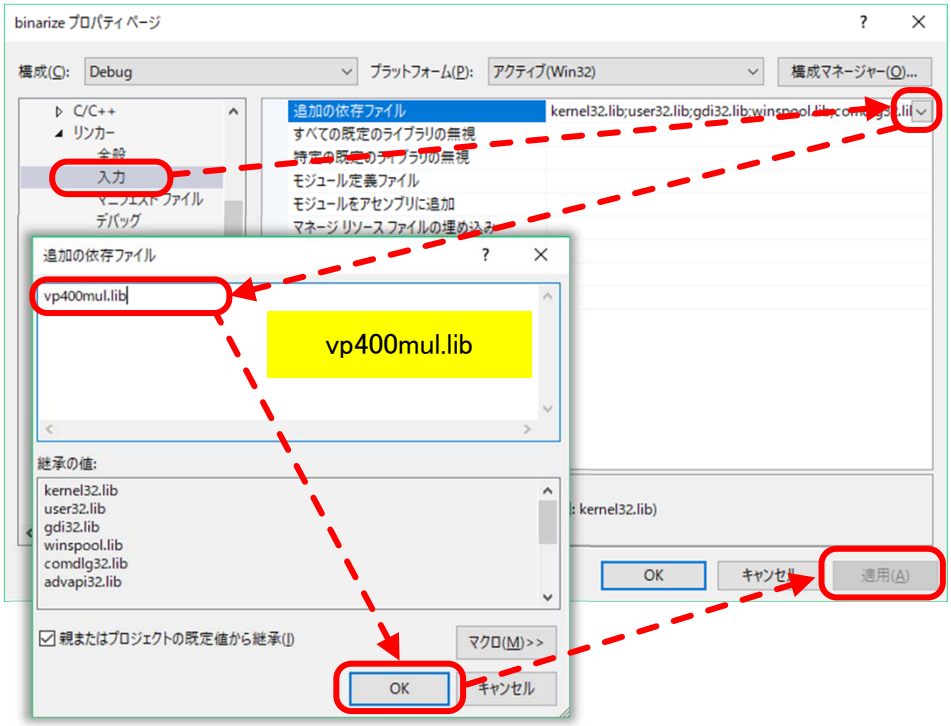


図3-54 ライブラリの登録

③「OK」をクリックしウィンドウを閉じてください。

3.9.7 リモートコマンドのデバッグ

①「ビルド(B)」→「ソリューションのビルド(B)」を選択し、ビルドを開始してください。
ビルドが完了すると出力ウィンドウに結果が表示されます。

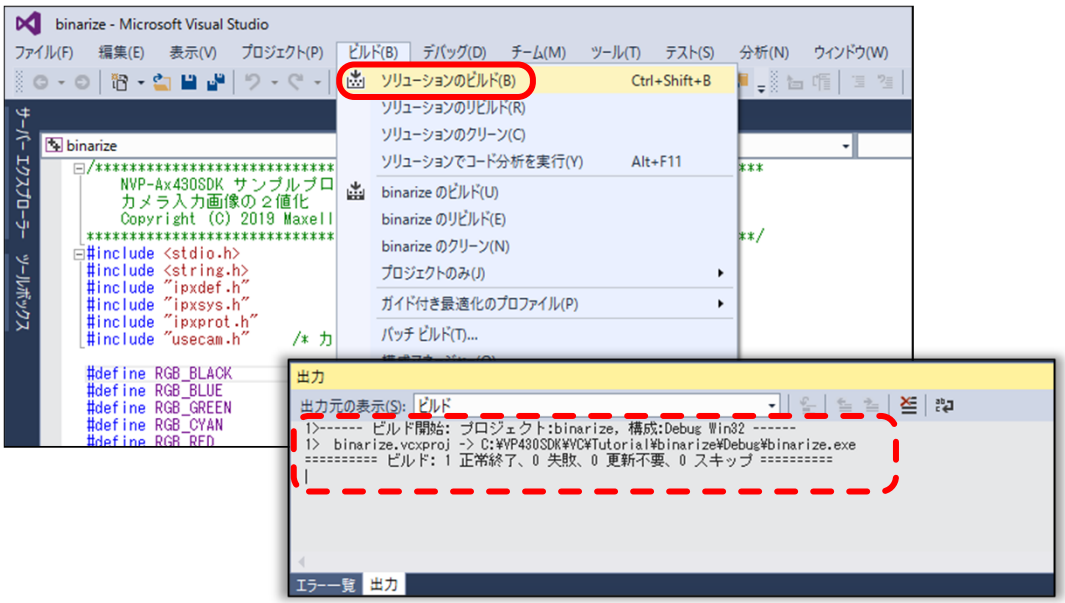


図3-55 ソリューションのビルド

②ソリューションエクスプローラーから「main.c」を開き、任意の行にブレークポイントを設置し、
「デバッグ(D)」→「デバッグ開始(S)」からデバッグ実行を開始してください。

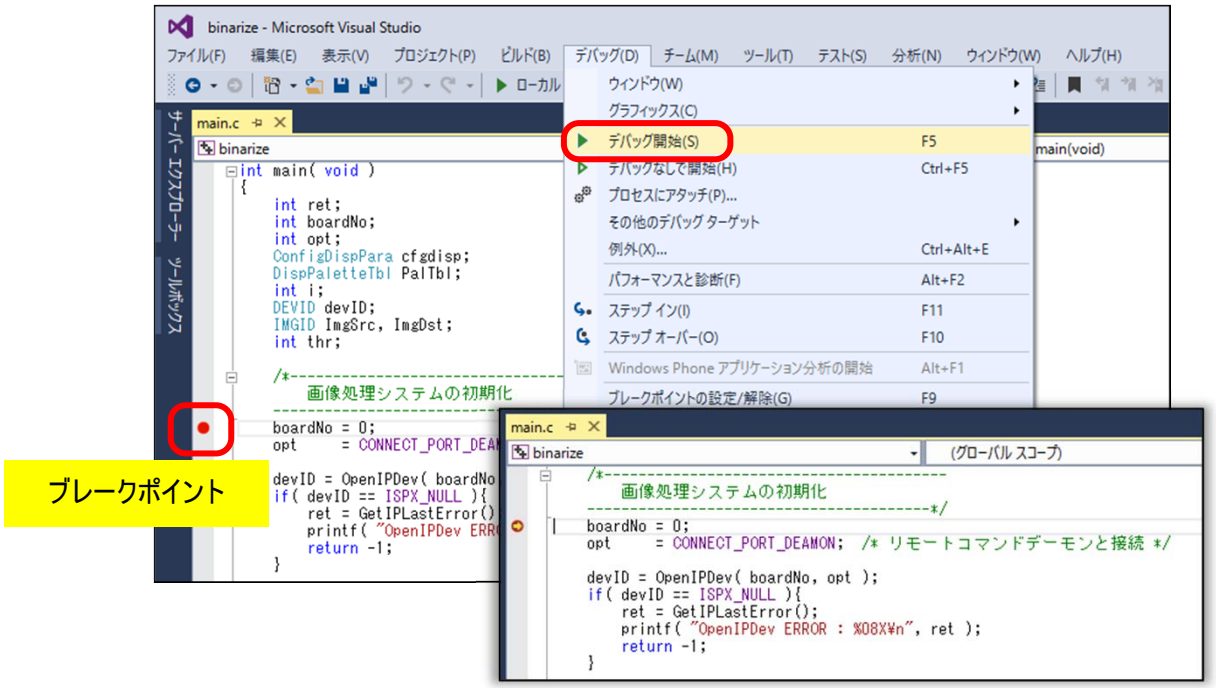


図3-56 デバッグ実行

画像処理ユニット NVP-Ax430シリーズ　ソフトウェア開発キット

NVP-Ax430SDK

チュートリアル（第３版）

開発元

マクセルフロンティア株式会社

営業部　〒244-0801 神奈川県横浜市戸塚区品濃町549-2三宅ビル

技術サポート窓口　URL www.frontier.maxell.co.jp
mail : vp-support@maxell.co.jp