

ユーザーズマニュアル

産業用組込み PC EC1G-01x/02x

開発環境

目次

産業用組込み PC EC1G-01x/02x

開発環境

はじめに

- 1) ……お願いと注意 …… 1
- 2) ……保証について …… 1

第1章 概要

- 1-1 EC1G-01x/02x 向け OS について …… 1-1
- 1-2 Linux の仕組み …… 1-2

第2章 開発環境

- 2-1 クロス開発環境 …… 2-1
- 2-2 開発環境インストール …… 2-2
 - 2-2-1 開発環境のインストールに必要なもの …… 2-2
 - 2-2-2 VirtualBox のダウンロード …… 2-4
 - 2-2-3 VirtualBox のインストール …… 2-6
 - 2-2-4 仮想マシンの作成 …… 2-11
 - 2-2-5 VirtualBox のネットワーク設定 …… 2-16
 - 2-2-6 USB 設定 …… 2-18
 - 2-2-7 仮想マシンの起動 …… 2-21
 - 2-2-8 開発環境の各種設定について …… 2-24
 - 2-2-9 EC1G-01x/02x 用開発環境のディレクトリ構成について …… 2-28
- 2-3 WideStudio/MWT によるアプリケーション開発 …… 2-29
 - 2-3-1 WideStudio の起動 …… 2-29
 - 2-3-2 プロジェクトの新規作成 …… 2-29

2-3-3	アプリケーションウィンドウの作成	2-32
2-3-4	部品の配置	2-34
2-3-5	イベントプロシージャの設定	2-36
2-3-6	イベントプロシージャの編集	2-37
2-3-7	コンパイル	2-38
2-3-8	ファイルの転送	2-40
2-3-9	WideStudio/MWT の開発例	2-44
2-4	Qt によるアプリケーション開発	2-47
2-4-1	Qt とは	2-47
2-4-2	Qt ライセンス	2-47
2-4-3	Qt Creator の起動	2-48
2-4-4	新規プロジェクト作成	2-49
2-4-5	部品の配置	2-53
2-4-6	ボタンイベント追加	2-53
2-4-7	インテル CPU 環境でビルド	2-55
2-4-8	ARM64 環境でビルド	2-56
2-4-9	リモートデバッグ方法	2-60
2-5	GDB によるデバッグ方法の詳細	2-68
2-6	VirtualBox を使用しない開発環境構築方法	2-73
2-6-1	Ubuntu 20.04 LTS のインストール	2-73
2-6-2	開発環境構築前準備	2-77
2-6-3	EC1G-01x/02x 用開発環境インストール	2-78
2-6-4	WideStudio インストール	2-79
2-6-5	Qt 開発環境インストール	2-81
2-6-6	Windows 共有の設定	2-86

付録

A-1	参考文献	2-1
-----	------	-----

はじめに

この度は、アルゴシステム製品をお買い上げいただきありがとうございます。

弊社製品を安全かつ正しく使用していただく為に、お使いになる前に本書をお読みいただき、十分に理解していただくようお願い申し上げます。

1) お願いと注意

本書では、産業用組込み PC EC1G-01x/02x（以下、EC1G-01x/02x）用開発環境に特化した部分について説明します。一般的な Linux についての詳細は省略させていただきます。Linux に関する資料および文献は、現在インターネット上や書籍など多数ございます。これらの書籍等と併せて本書をお読みください。

2) 保証について

EC1G-01x/02x 用開発環境の動作は出荷パッケージのバージョンでのみ動作確認しております。EC1G-01x/02x 用開発環境はお客様でソースの改変、ライブラリの追加と変更、プログラム設定の変更等を行うことができますが、これらの変更を行われた場合は動作保証することができません。

第 1 章 概要

本章では、EC1G-01x/02x 用開発環境の具体的な内容を説明する前に、EC1G-01x/02x 用開発環境の概要について説明します。

1-1 EC1G-01x/02x 向け OS について

「Linux」とは、Linux カーネルのみを指す言葉です。しかし、Linux カーネルのみでは、オペレーティングシステム（以下 OS）としての役割を果たすことができません。OS として使うには、Linux カーネルのほかに、以下のような各種ソフトウェアパッケージと併せて使用する必要があります。

- シェル (bash、ash、csh、tcsh、zsh、pdksh、……)
- util-linux (init、getty、login、reset、fdisk、……)
- procs (ps、pstree、top、……)
- GNU coreutils (ls、cat、mkdir、rmdir、cut、chmod、……)
- GNU grep、find、diff
- GNU libc
- 各種基本ライブラリ (ncurses、GDBM、zlib……)
- X Window System

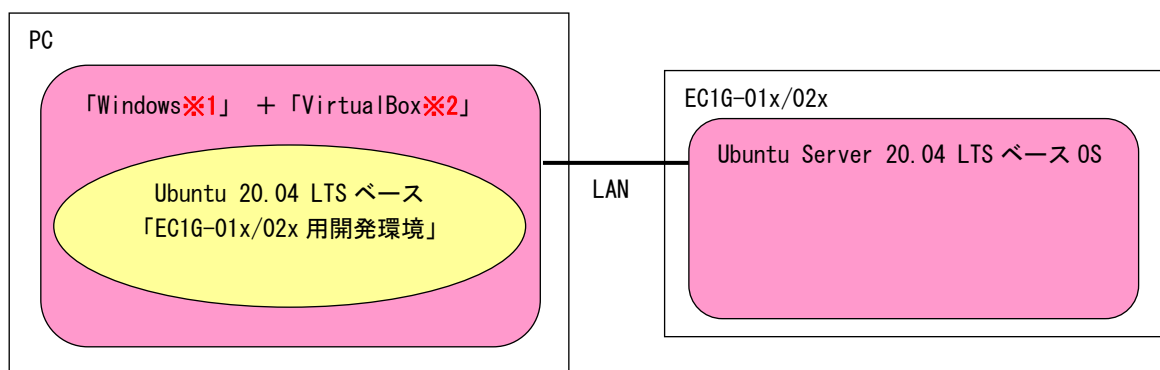
Linux カーネルといくつかの必要なソフトウェアパッケージをまとめて、OS として使えるようにしたものを Linux ディストリビューションといいます。

最初に述べましたとおり、「Linux」という言葉は、本来カーネルを指す言葉です。そのため、「カーネルとしての Linux」と「OS としての Linux」を厳密には区別する必要がありますが、本書では「Linux」とは「OS としての Linux」を指す言葉として使用します。

EC1G-01x/02x 向け OS は、「Ubuntu Server 20.04 LTS」という Linux ディストリビューションをベースに、独自の I/O ドライバを組み込んだものになります。

パソコン上に EC1G-01x/02x 用開発環境をインストールすることで、EC1G-01x/02x 用のソフトウェアを開発することができます。

EC1G-01x/02x 用開発環境イメージを図 1-1-1 に示します。



※注 1: Windows は米国 Microsoft Corporation の米国およびその他の国における登録商標です。

※注 2: VirtualBox は、米国 Oracle Corporation, Inc. の米国およびその他の国における商標または登録商標です。

図 1-1-1. EC1G-01x/02x の開発環境

EC1G-01x/02x 実行環境の詳細は、EC1G-01x/02x 実行環境ユーザーズマニュアルを参照してください。本マニュアルでは、開発環境についての説明を記述します。

1-2 Linux の仕組み

Linux のソフトウェア構成を図 1-2-1 に示します。

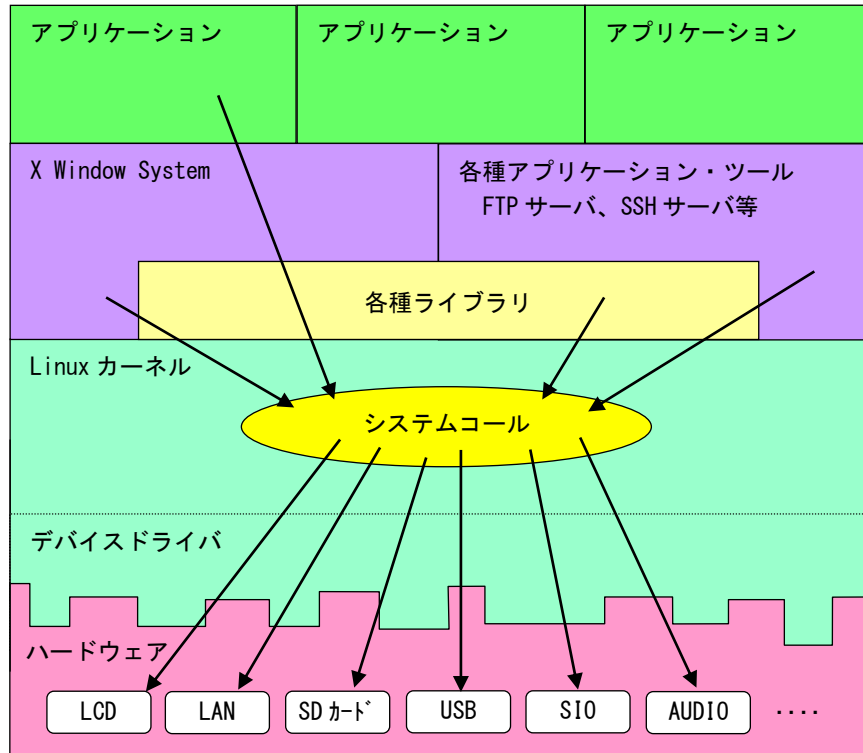


図 1-2-1. Linux ソフトウェア構成図

OS として重要な役割の一つに、ハードウェアアクセスの複雑さを隠し、統一されたプログラミングインターフェース（システムコールや API と呼ばれる）をアプリケーションに提供するというものがあります。Linux ではハードウェアを制御する為にドライバに関連付けられた「デバイスファイル」を読み書きすることで制御します。これは UNIX 系 OS の大きな特徴であり、ファイルを扱う感覚でハードウェアを制御することができます。Linux の代表的なシステムコールとして、open、close、read、write 等があります。これらのシステムコールは特別な呼び方をしてしているわけではなく、関数と同じように呼び出すことができます。

もう一つ OS の重要な役割として、CPU 時間、メモリ、ネットワーク等のリソースをプログラムやプロセス、スレッドに分配するというものもあります。これは Linux カーネルが処理しており、アプリケーション作成時に特に意識する必要はありません。図 1-2-1 にあるような X Window System や、SSH サーバや FTP サーバもプロセスの一つです。CPU 時間やメモリなどのリソースには限りがある為、複数のプロセスを同時に実行すると、それぞれのパフォーマンスは落ちます。そのため、必要最低限のプロセスで実行効率のよいプログラムを作成する必要があります。

第 2 章 開発環境

本章では、EC1G-01x/02x 用開発環境について説明します。

2-1 クロス開発環境

プログラムを開発する場合に必要なのが、ソースコードを記述するエディタ、ソースコードをコンパイルするコンパイラ、コンパイルされたプログラムを実行する為の実行環境です。

例えば、Microsoft 社の Windows 上で動作するアプリケーションを開発する場合、エディタでソースを書き、Visual Studio 等のコンパイラでコンパイルを行い、作成された exe ファイルを実行します。これで作成したアプリケーションが Windows 上で実行されます。

Linux の場合でも同じです。Linux マシン上で動作するエディタでソースを書き、gcc でコンパイル後に生成された実行ファイルを実行します。

両者とも、コンパイルと実行を同じパソコン環境上で行うことができます。このような開発方式をセルフ開発方式といいます。

当社端末では、クロス開発方式を採用しています。クロス開発方式とは、コンパイル環境と実行する環境が異なる方式です。ソースコードの記述やコンパイルはパソコン上でを行い、LAN 等で実行ファイルをターゲットに送って実行することになります。(図 2-1-1 参照)。

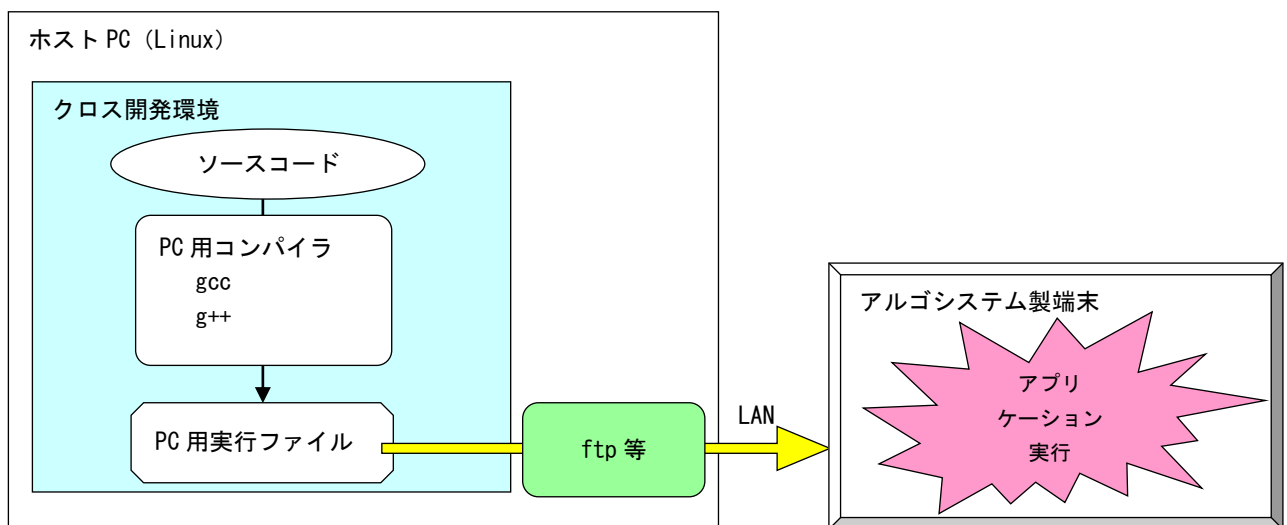


図 2-1-1. クロス開発方式イメージ図

このような開発環境を構築するには、表 2-1-1 に示すような開発環境ツールが必要となります。

表 2-1-1. クロス開発に必要なツール

ツール名	説明
GCC	GNU C コンパイラ
binutils	リンカ、アセンブラ等のソフトウェア開発ツール
GDB	デバッガ
glibc	GNU C ライブラリ

EC1G-01x/02x 用開発環境は、Ubuntu 20.04 LTS ディストリビューションイメージに、ARM CPU 用の開発環境を組み込んだものになります。

VirtualBox という仮想マシン上で EC1G-01x/02x 用開発環境を起動すれば、それぞれの CPU 搭載端末用のアプリケーションを開発することが可能です。

2-2 開発環境インストール

アプリケーション開発環境である EC1G-01x/02x 用開発環境を Oracle Corporation 製の VirtualBox 5.1.14 を用いて、Windows パソコン上で構築する手順について説明します。

2-2-1 開発環境のインストールに必要なもの

開発環境のインストールに必要なものは下記の 3 点です。

1. 開発環境用 DVD

EC1G-01x/02x 用アプリケーション開発環境一式です。
DVD の内容を表 2-2-1-1 に示します。

表 2-2-1-1. EC1G-01x/02x 用開発環境 DVD の構成

DVD のディレクトリ	内容
Doc	開発環境の取扱い説明書が格納されています。 ・ EC1G-01x/02x 用開発環境ユーザーズマニュアル. pdf 本書のことです。
Dev	VirtualBox を使用せずに直接、パソコン上に開発環境を構築する場合のパッケージが格納されています。 ・ append-tools-arm64. tar. gz (ARM CPU 用) ライブラリ、ドライバ、開発環境、サンプルのソースが含まれます。 ・ append-tools-imx8mq. tar. gz (ARM CPU 用) i. MX8MQ 用カーネル、ブートローダーのソースが含まれます。
VDI	VirtualBox 用の OS イメージファイルが格納されています。 ・ Ubuntu20.04_Development64. 7z インストール方法は『2-2-4 仮想マシンの作成』を参照してください。

2. VirtualBox Ver5.1.14

本書で使用している VirtualBox のバージョンは ver5.1.14 です。VirtualBox のバージョンによっては、本書の画面表示と異なる可能性があります。VirtualBox は Oracle Corporation 社の製品です。VirtualBox の使用にあたっては Oracle Corporation 社の使用許諾条件に従ってご使用ください。

3. 開発環境インストール用 Windows PC

Windows が動作しており、VirtualBox がインストール可能なパソコンが必要です。VirtualBox のインストールおよび EC1G-01x/02x 用開発環境用イメージを動作させる為に、最低限必要な環境として表 2-2-1-2 の PC スペックが必要になります。

表 2-2-1-2. 必須 PC スペック

CPU	X86 または X64 (1GHz 以上を推奨) Intel, AMD
メモリ	1024MByte 以上 (4GByte 以上推奨) ※注: VirtualBox はメモリ領域が 1024MByte 未満の場合、正常に動作しない場合があります。その為、必須環境を満たしているパソコンでも、ホスト OS 上で他のソフトウェアを同時に起動している場合、メモリ不足により VirtualBox が正常に動作しない事があります。その場合は他のソフトウェアを一度停止させメモリ領域を開放した上で、もう一度 VirtualBox を起動してください。
HDD 空き領域	50GByte 以上 (128GByte 以上推奨) ※注: EC1G-01x/02x 用開発環境では、ゲスト OS のイメージファイルは最大容量 128GByte の可変長の OS イメージとなっています。初期状態では OS イメージファイルは 17GByte ほどの大きさですが、開発をしていくにつれこのサイズは大きくなる為、ゲスト OS の空き領域を超えてしまう場合、正常に動作しなくなる場合があります。その場合はホスト OS の空き領域を増やすか、ゲスト OS 内の不要なファイルを消去し、空き領域を確保してください。
OS	Windows 7 以降
ファイルシステム	NTFS ※注: ファイルシステムが FAT32 の環境では単独でファイルサイズが 4GByte を超えるファイルは使用できません。EC1G-01x/02x 用開発環境のゲスト OS のイメージファイルは 8GByte を超えてしまう為、FAT32 上の EC1G-01x/02x 用開発環境は使用できません。
その他	USB ポート、LAN ポート (アルゴシステム製端末とパソコンの接続に最低でもいずれかひとつが必要になります)

EC1G-01x/02x 用開発環境をインストールするパソコンで動作している OS (Windows) と VirtualBox 上で動作する OS (Ubuntu 20.04 LTS) を区別する為、パソコン側の OS を「ホスト OS」、VirtualBox 上の OS を「ゲスト OS」と表記しています。

2-2-2 VirtualBox のダウンロード

VirtualBox のインストーラは <http://www.virtualbox.org/> よりダウンロードします。

- ① VirtualBox 公式ページの左メニューから [Downloads] を選択します。



図 2-2-2-1. 公式 TOP ページ

- ② 図 2-2-2-2 の使用するホスト OS の環境を選択する画面に変わります。
[VirtualBox older builds] を選択します。
なお、この画面は VirtualBox のバージョンアップに伴い、変化する可能性があります。

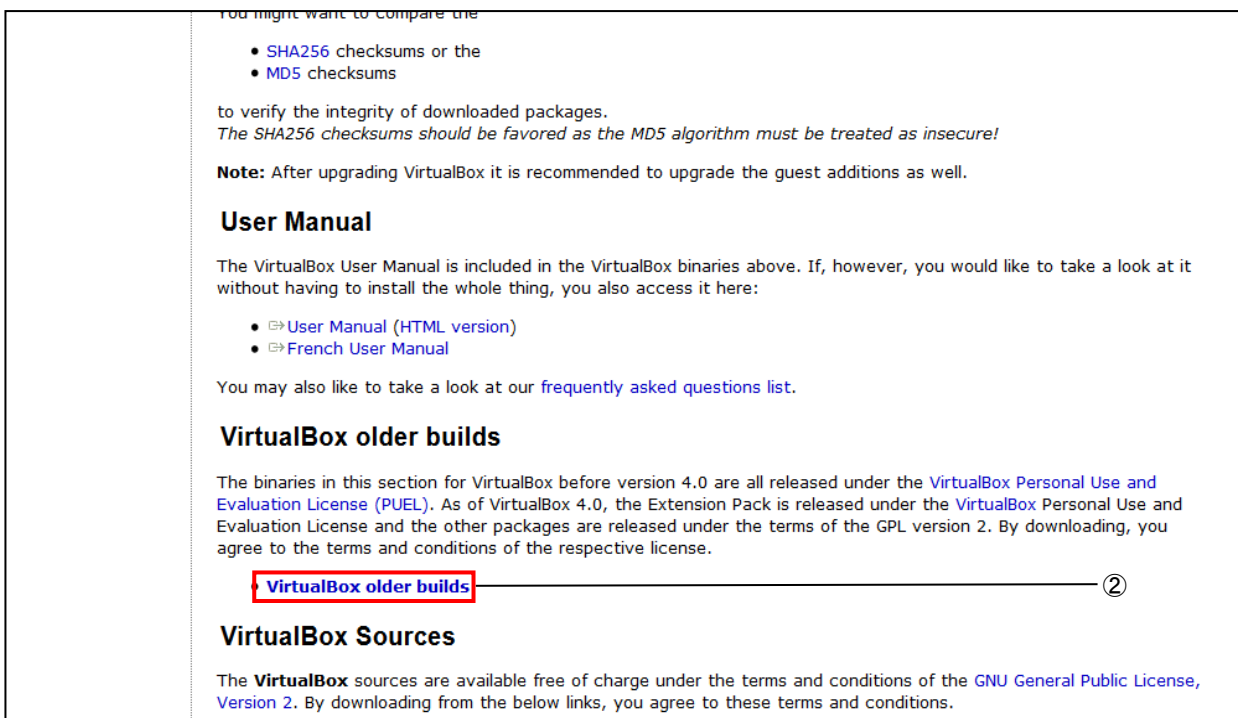


図 2-2-2-2. VirtualBox ダウンロードページ

- ③ 最新版が 5.1.14 でない場合、Virtual Box 5.1 をクリックしてください。



図 2-2-2-3. Virtual Box Old Builds ダウンロードページ

- ④ Virtual Box 5.1 系のリストが並んでいますので、『VirtualBox 5.1.14 for Windows hosts x86/AMD64』という項目をクリックしてください。



図 2-2-2-4. 古いバージョンの VirtualBox ダウンロードページ

- ⑤ ダウンロードが開始されます。

2-2-3 VirtualBox のインストール

- ① 『2-2-2 VirtualBox のダウンロード』でダウンロードした、「VirtualBox-5.1.14-112924-Win.exe」をダブルクリックします。
- ② 図 2-2-3-1 のような画面が表示されるので、[Next>]をクリックします。

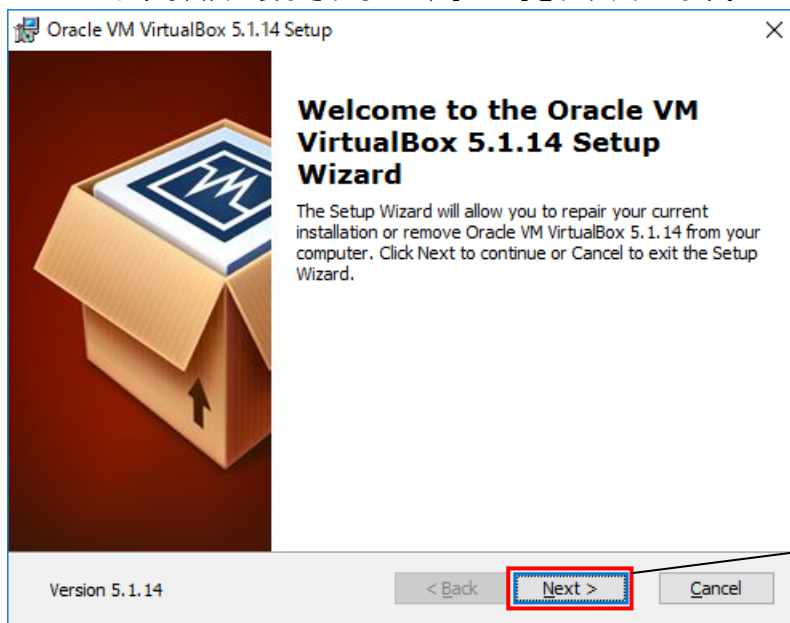


図 2-2-3-1. ダウンローダー起動後の画面

- ③ 図 2-2-3-2 のカスタムセットアップ画面に変わります。ここでは VirtualBox に追加するプラグイン機能を選択します。
- ・ VirtualBox USB Support : VirtualBox 上で USB 機能を使えるようにするプラグインです。
 - ・ VirtualBox Networking : VirtualBox 上でネットワークへ接続する為のプラグインです。
- 本製品を使用するに当たって、両方のプラグインをインストールする必要があります。デフォルトではインストールする設定になっています。
- ④ [Next>] をクリックします。

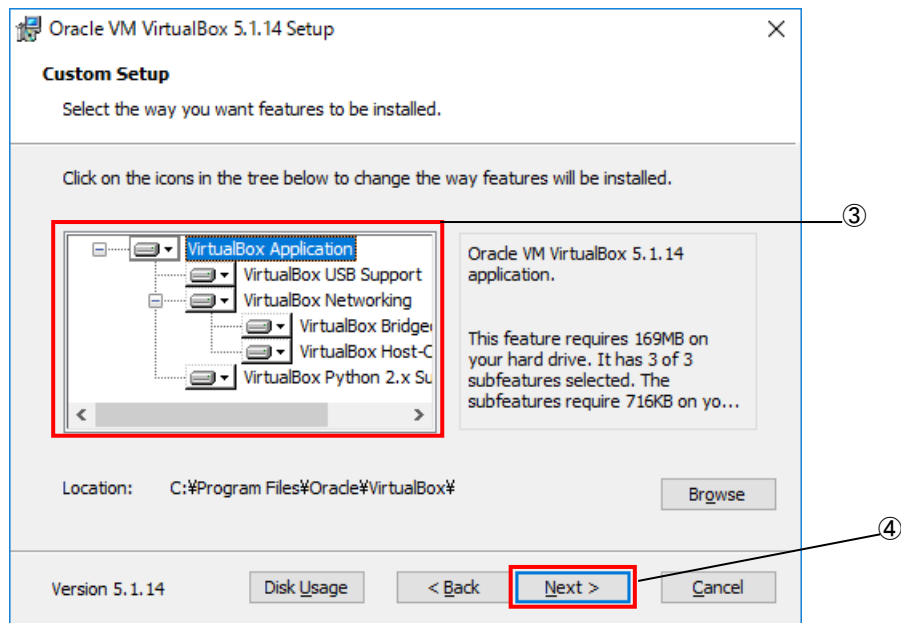


図 2-2-3-2. カスタムセットアップ画面

- ⑤ 図 2-2-3-3 のインストールオプション設定画面に切り替わります。インストールオプションを設定できます。[Next>] をクリックします。
- ・ Create start menu entries : スタートメニューに追加します。
 - ・ Create a shortcut on the desktop : ショートカットをデスクトップに配置します。
 - ・ Create a shortcut in the Quick Launch Bar : ショートカットをクイックランチャーに配置します。
 - ・ Register file associations : ファイルの関連付けを登録します。
- ⑥ 図 2-2-3-3 の [Next>] をクリックします。

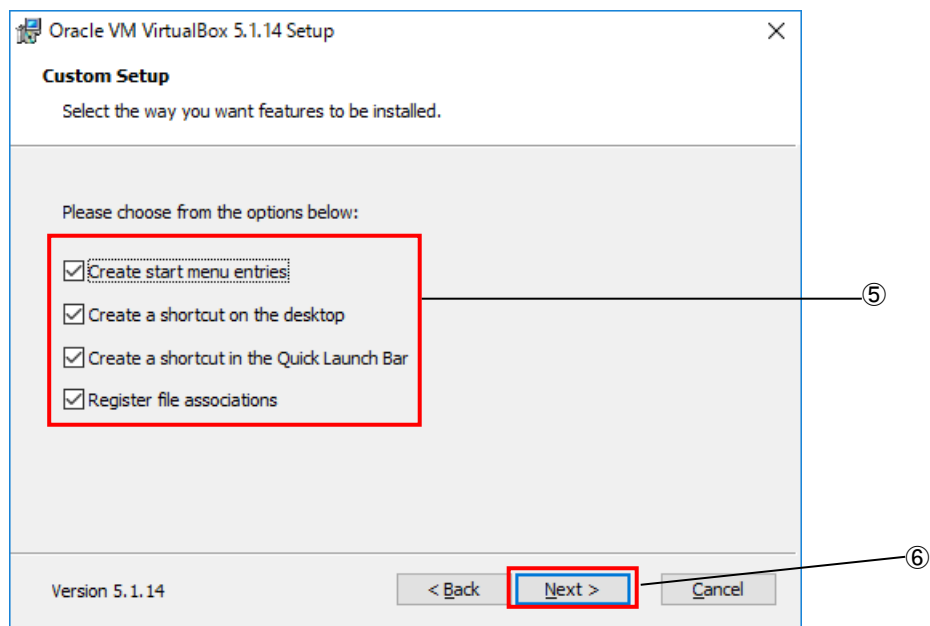


図 2-2-3-3. インストールオプション設定画面

- ⑦ 図 2-2-3-4 のネットワークインストール確認画面に切り替わりますので、[Yes] をクリックします。

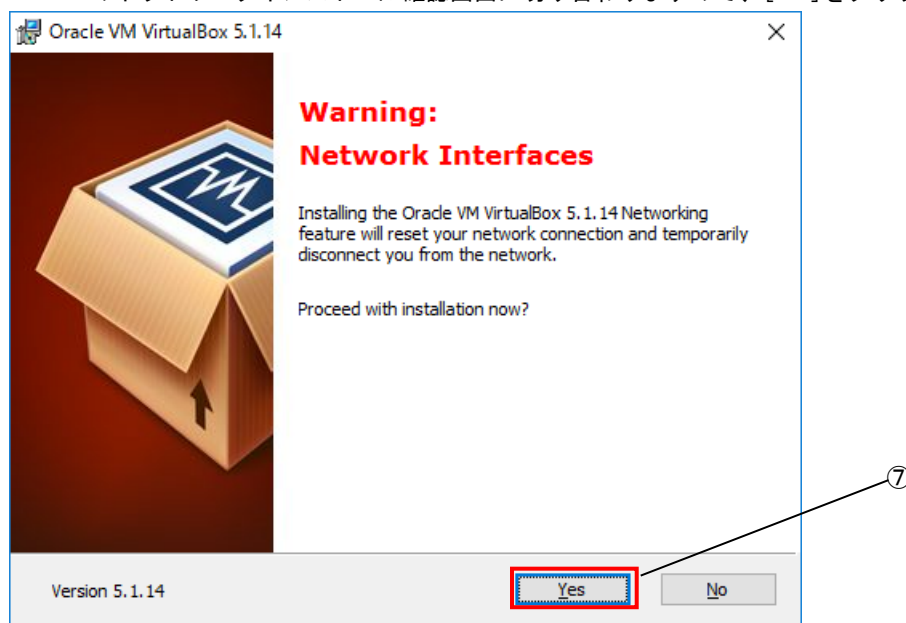


図 2-2-3-4. ネットワークインストール確認画面

- ⑧ 図 2-2-3-5 のインストール確認画面に切り替わりますので、[Install]をクリックします。

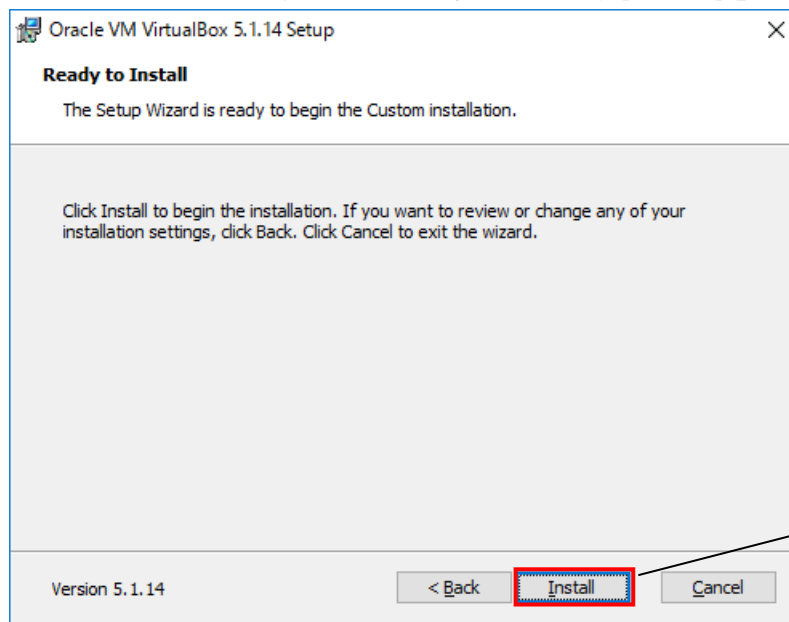


図 2-2-3-5. インストール確認画面

インストール開始時に、書き換え許可ウインドウが開く場合は、許可してください。

- ⑨ インストール中、図 2-2-3-6 のようなデバイスドライバのインストールの実行を確認する画面が開きますが、いずれも[インストール]をクリックします。

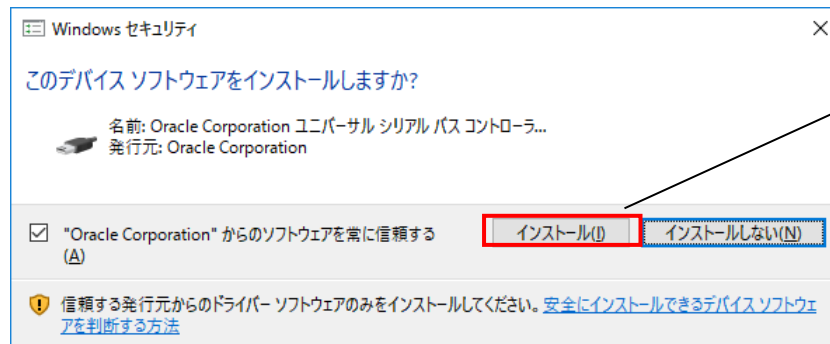


図 2-2-3-6. ドライバのインストール警告画面

- ⑩ インストールが完了すると、図 2-2-3-7 が表示されます。[Finish]をクリックします。

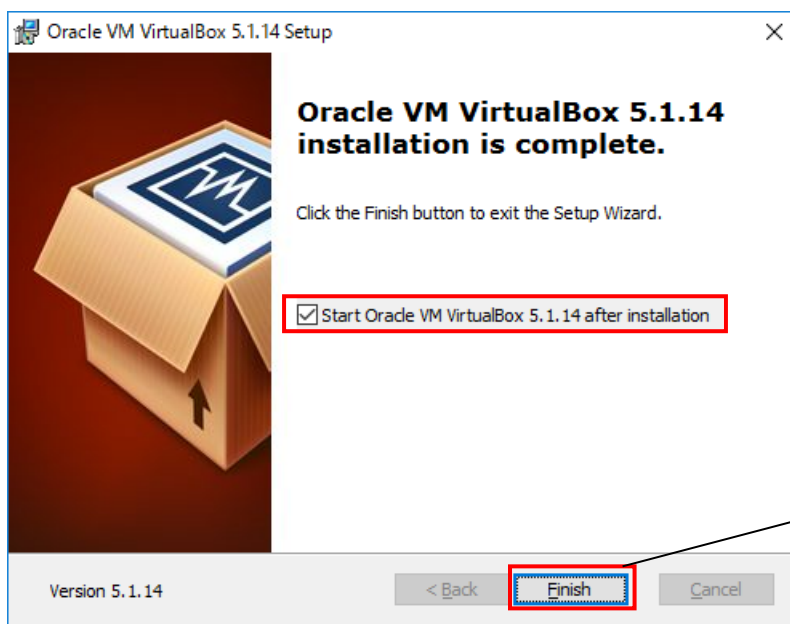


図 2-2-3-7. インストール完了画面

2-2-4 仮想マシンの作成

作成する仮想マシンの構成を指定します。

- ① [スタート]→[全てのプログラム]→[Oracle VM VirtualBox]→[Oracle VM VirtualBox]を選択します。

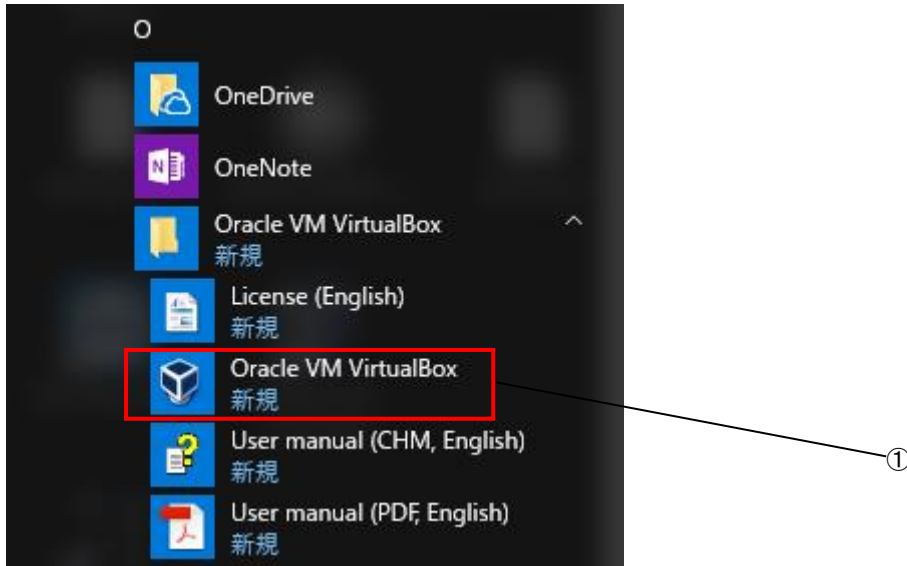


図 2-2-4-1. VirtualBox 起動

- ② VirtualBox の初回起動時に下記のような設定ファイルが作成されます。

C:\%User%\<ユーザ名>%VirtualBox

このフォルダは VirtualBox の OS イメージであるファイルを格納するフォルダです。開発環境 SD カードの<SD カード>%VirtualBox%Ubuntu20.04_Development64.7z をこのフォルダに展開します。Ubuntu20.04_Development.7z を解凍することで Ubuntu20.04_Development.vdi が作成されます。

- ③ VirtualBox が起動し、図 2-2-4-2 のような VirtualBox メイン画面が表示されるので、「新規(N)」をクリックします。

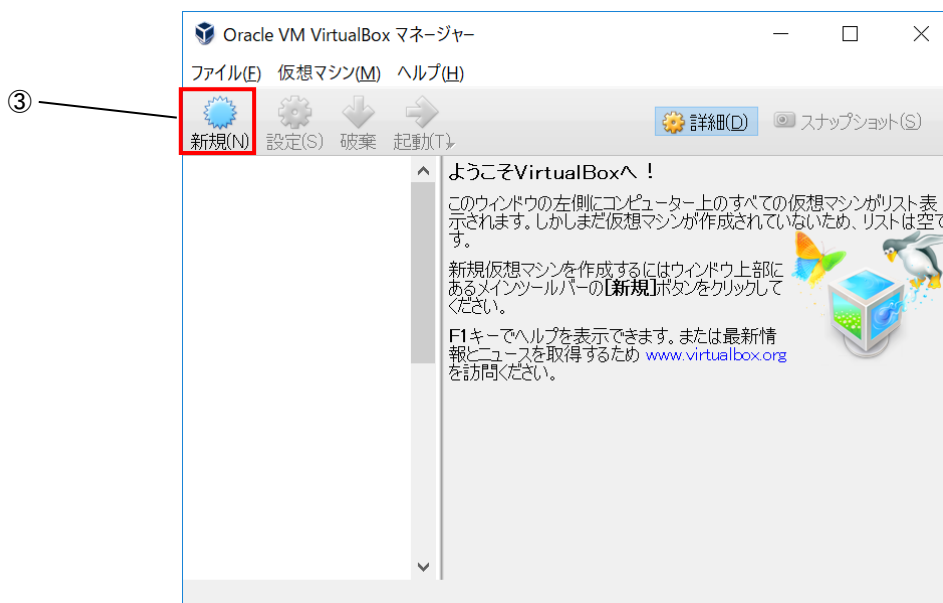


図 2-2-4-2. VirtualBox メイン画面

- ④ 図 2-2-4-3 のマシン名と OS 種類選択画面に変わるので、[名前 (N)] には [Ubuntu20.04_Development64] と入力します。
- ⑤ [OS タイプ (T)] では [Linux] [Ubuntu (64-bit)] を選択します。
- ⑥ [次へ (N)] をクリックします。

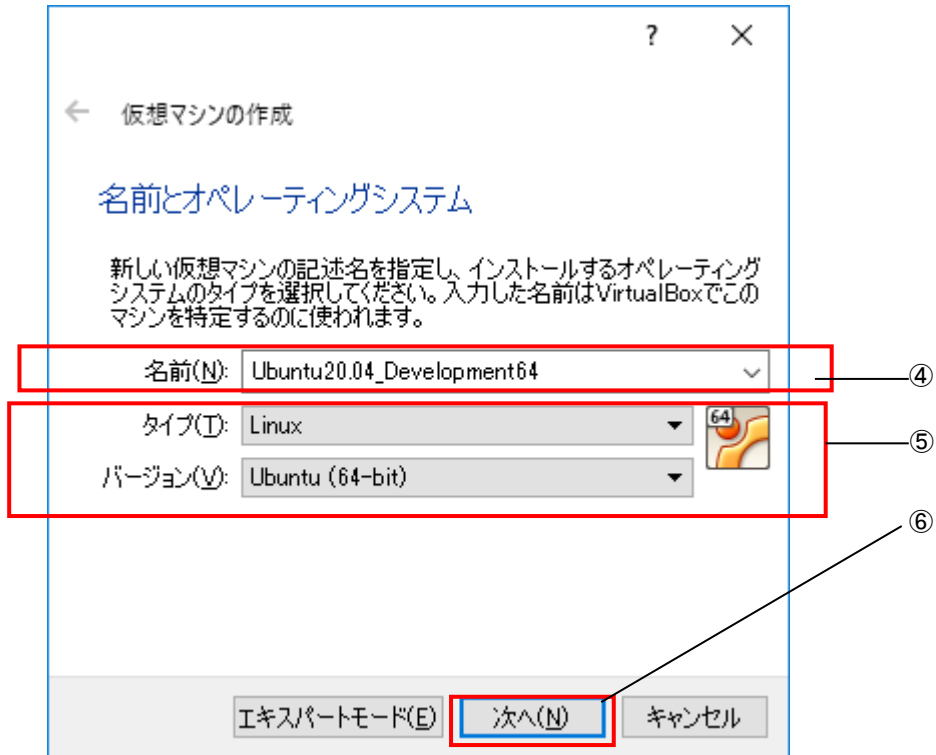


図 2-2-4-3. マシン名、OS タイプの設定

- ⑦ 図 2-2-4-4 のようなメモリ設定画面に変わります。仮想マシンのメモリサイズを入力します。
※注：ご使用のパソコンに応じてご指定ください。必須環境は 1024MByte 以上です。
- ⑧ [次へ(N)] をクリックします。

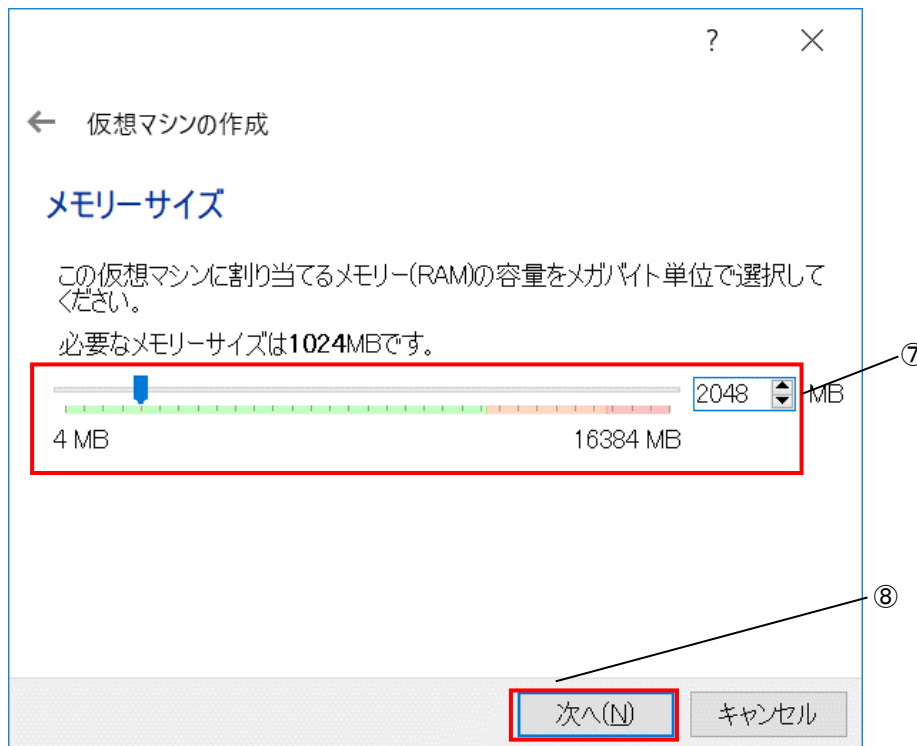



図 2-2-4-4. メモリの設定

- ⑨ 図 2-2-4-5 の仮想ハードディスクのイメージ選択画面へ変わるので、[すでにある仮想ハードドライブを使用する]をクリックしてください。
- ⑩ プルダウンメニューに Ubuntu20.04_Development64.vdi がない場合  をクリックして⑪に進みます。プルダウンメニューに Ubuntu20.04_Development64.vdi がある場合、[作成]をクリックし、⑬へ進んでください。

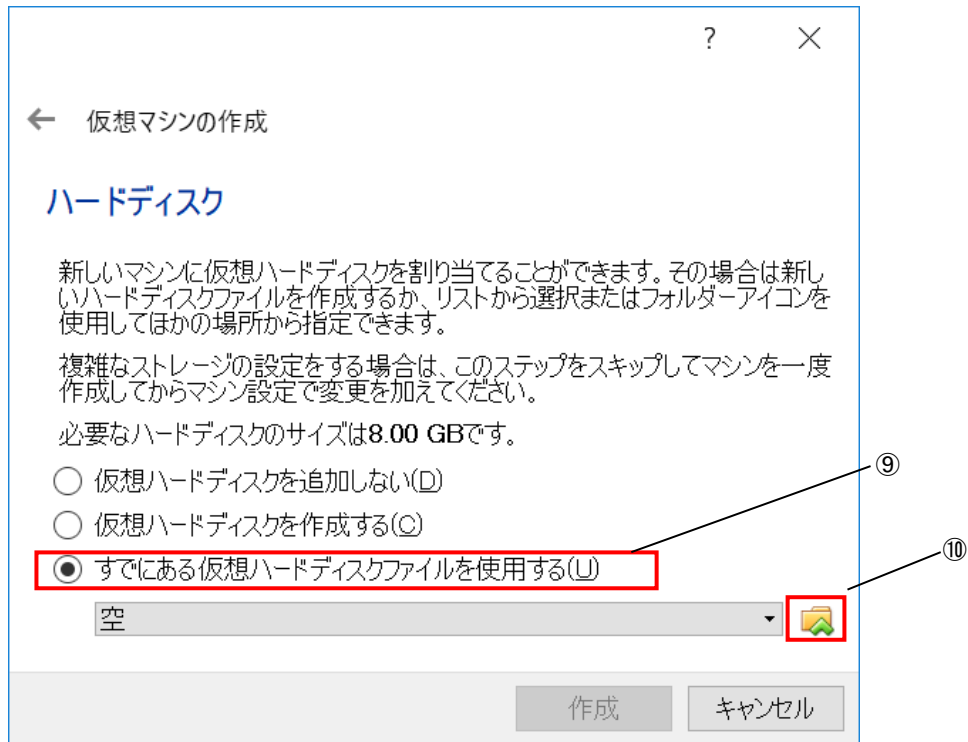


図 2-2-4-5. 仮想ハードディスクのイメージ選択画面

- ⑪ 図 2-2-4-6 のように仮想ディスク選択画面が開きます。
- ⑫ であらかじめ展開しておいた Ubuntu20.04_Development64.vdi を選択します。
- ⑬ [開く (O)] を押してください。

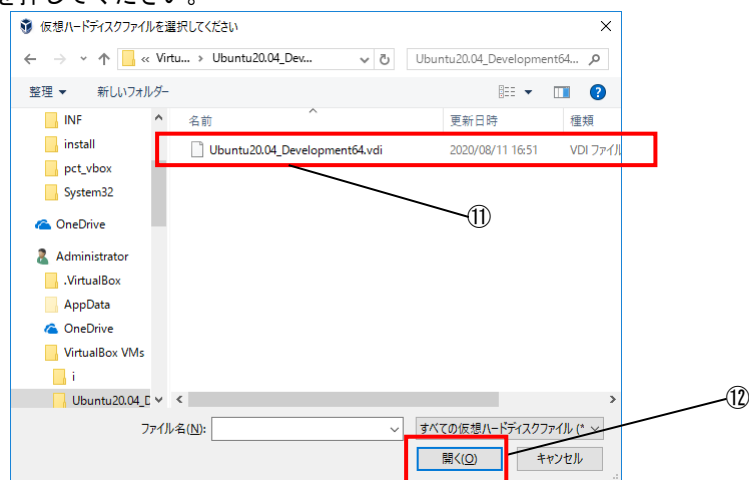


図 2-2-4-6. 仮想ディスク選択画面

- ⑬ 仮想ハードディスクのイメージ選択画面に戻ります。[作成]を押してください。

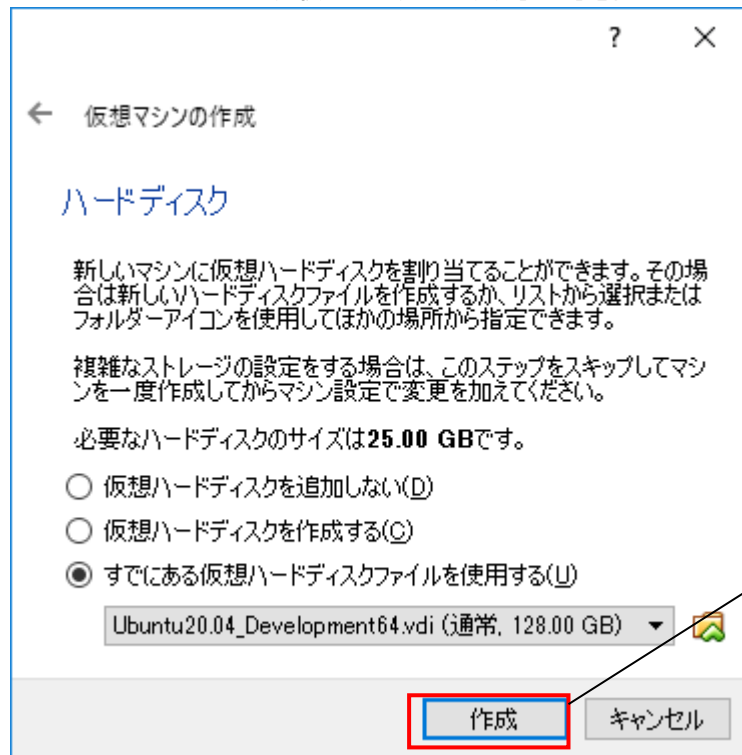


図 2-2-4-7. 仮想ハードディスクのイメージ選択画面

2-2-5 VirtualBox のネットワーク設定

正常に仮想マシンの作成が完了されると図 2-2-5-1 のような画面になります。ゲスト OS 上でネットワークを使用する為には、ゲスト OS を起動する前にネットワークアダプタの設定を行う必要があります。設定を行う仮想マシンが選択されていることを確認した上で、「ネットワーク」を選択します。

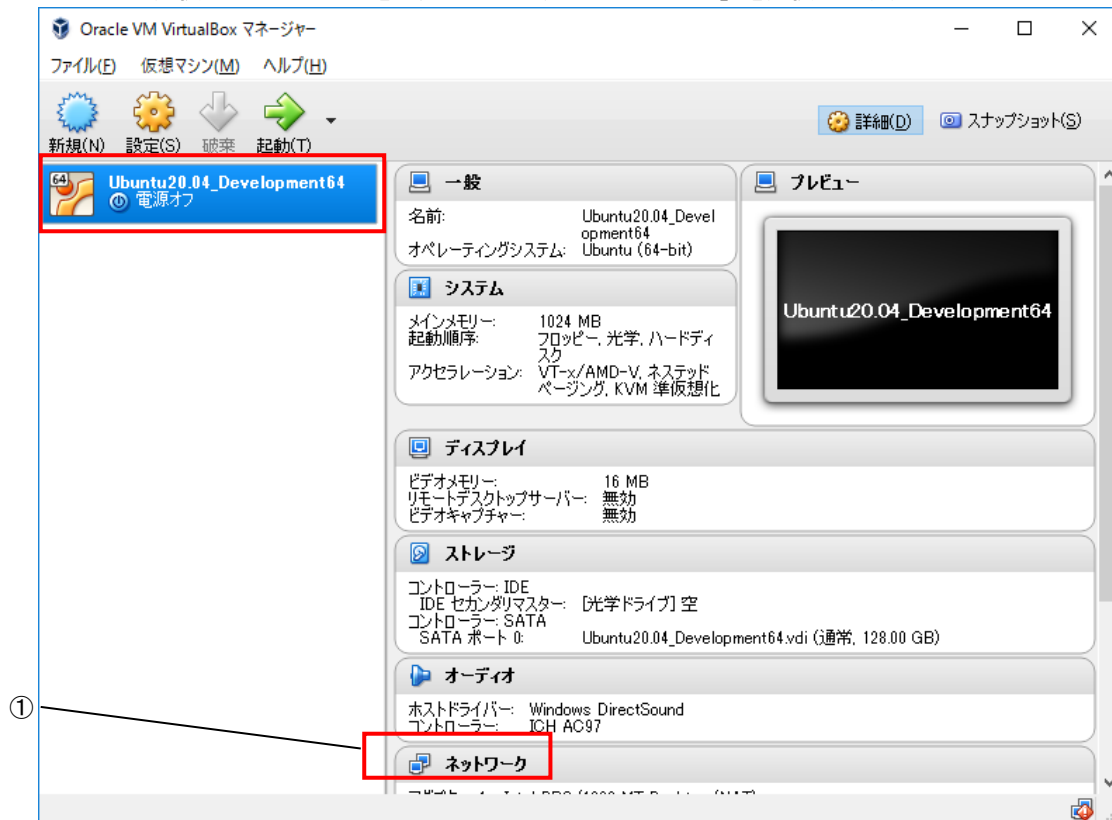


図 2-2-5-1. VirtualBox 初期画面

- ① 「ネットワーク」を選択することで、図 2-2-5-2 のような画面が開きます。

- ② 「アダプタ 1」を選択します。
- ③ 「ネットワークアダプタを有効化 (E)」にチェックを入れます。
- ④ 「割り当て (A)」を「ブリッジアダプター」にします。
- ⑤ 名前に、LAN チップの名称が記載されることを確認します。
- ⑥ 「OK」ボタンをクリックすることで、VirtualBox メイン画面に戻ります。

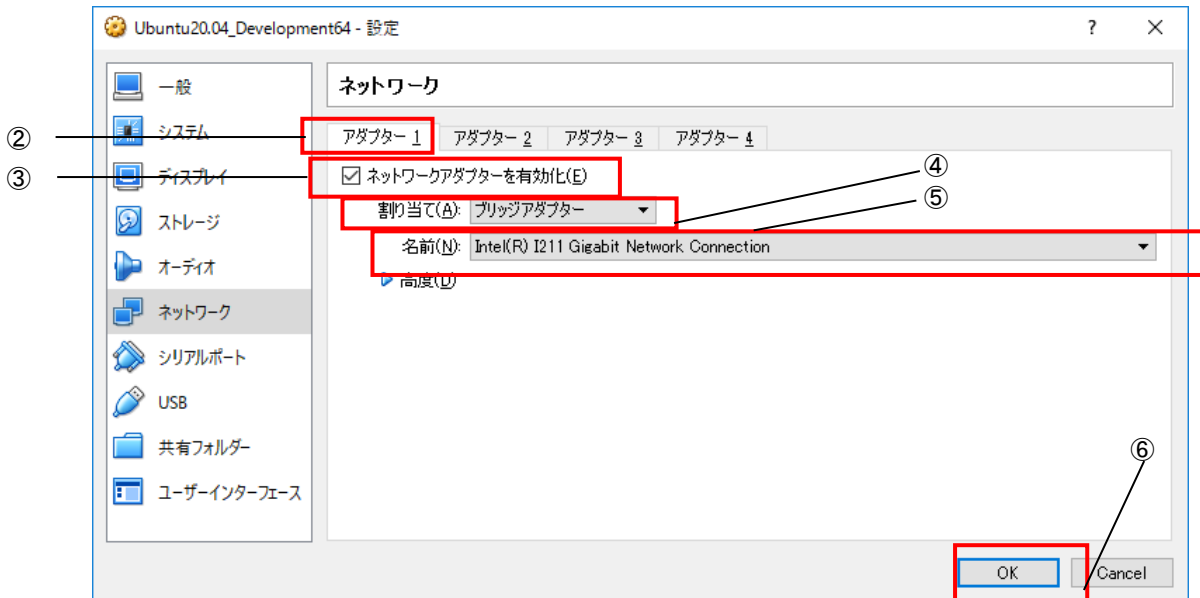


図 2-2-5-2. ネットワーク設定画面

2-2-6 USB 設定

正常に仮想マシンの作成が完了されると図 2-2-6-1 のような画面になります。ゲスト OS 上で USB 機器を使用する為には、ゲスト OS を起動する前に USB アダプタの設定を行う必要があります。設定を行う仮想マシンが選択されていることを確認した上で、「USB」を選択します。

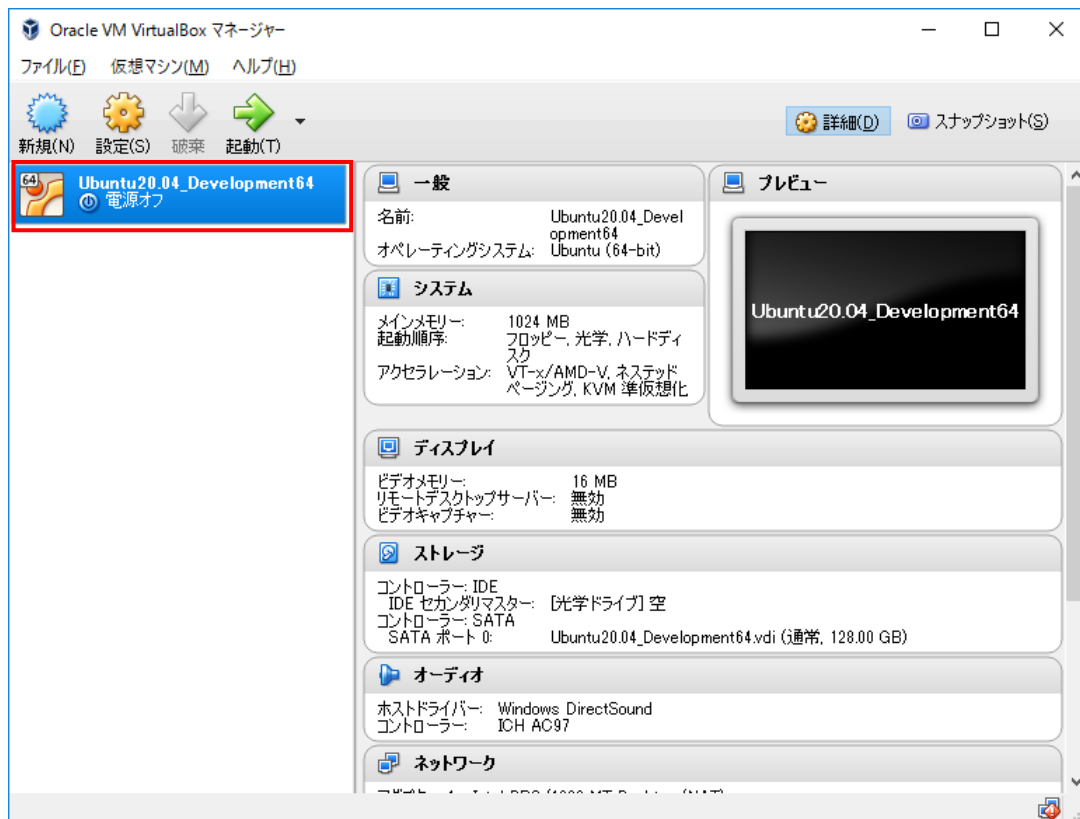


図 2-2-6-1. VirtualBox 初期画面

- ① 「USB」を選択することで、図 2-2-6-2 のような画面が開きます。

- ② 「USB コントローラを有効化(U)」にチェックを入れます。
- ③ 「USB2.0 (EHCI) コントローラを有効化(H)」にチェックを入れます。
- ④ 「空のフィルタを追加する」というボタンをクリックして空の新規フィルタを追加します。
- ⑤ 「OK」ボタンをクリックすることで、VirtualBox メイン画面に戻ります。

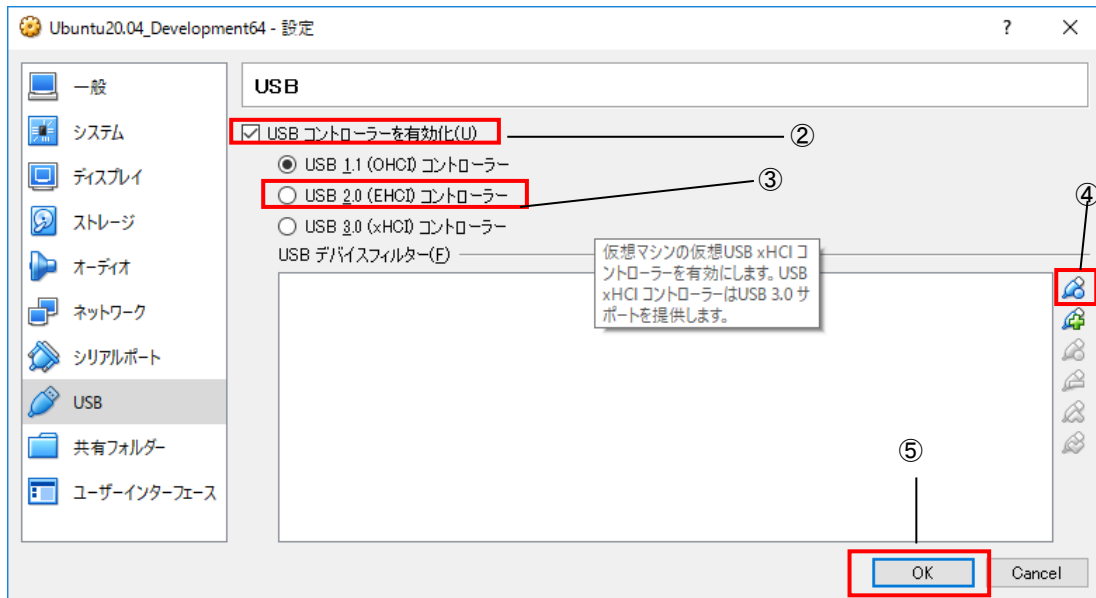


図 2-2-6-2. USB 設定画面

仮想マシン上で USB 機器を使用する為には、ゲスト OS を起動する前に、Windows 上で使用する USB 機器のドライバがインストールされており、正常に動作している必要があります。

ゲスト OS が起動している状態で新規の USB 機器を接続されると、その USB 機器は正常に動作しません。一度、ゲスト OS をシャットダウンし、USB 機器を挿入しなおして、Windows 用のドライバをインストールしてください。その後、ゲスト OS を起動することで、新規の USB 機器を使用することができます。

VirtualBox で認識している USB 機器は「デバイス(D)」→「USB」をクリックすることで一覧表示されます(図 2-2-6-3 参照)。チェックが付いているものはゲスト OS が認識しているものです。このチェックを ON/OFF することで、ゲスト OS から USB 機器を抜き差しすることができます。

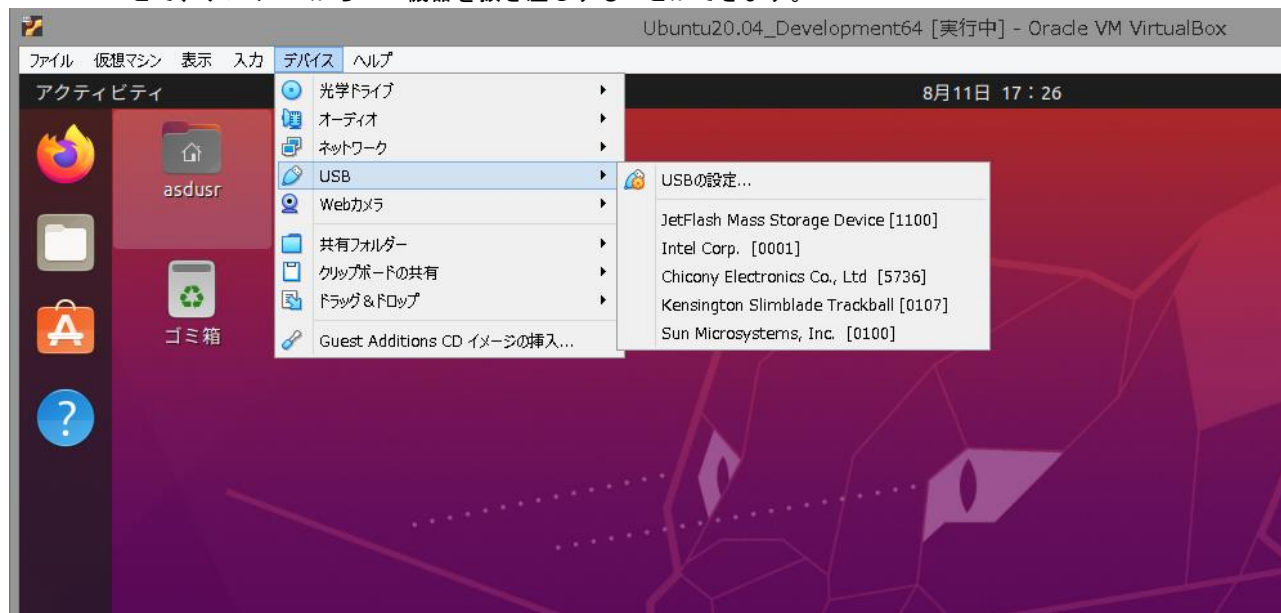


図 2-2-6-3. USB デバイス一覧

※注：USB とネットワーク設定以外の設定を変更する場合は、VirtualBox 付属のドキュメントを参照してください。

2-2-7 仮想マシンの起動

弊社配布の OS イメージ、Ubuntu20.04_Development64.vdi は既に EC1G-01x/02x 用開発環境がインストールされています。なお、Ubuntu20.04_Development64.vdi のユーザ名とパスワードは、初期状態では下記のように設定されています。ログイン時やシステム変更時にユーザ名、パスワードの入力を求められるので、下記のユーザ名とパスワードを入力します。

ユーザ名 : asdusr
パスワード : asdusr

- ① VirtualBox メイン画面左側の List から Ubuntu20.04_Development64 を選択します。
- ② [起動(T)] をクリックします。

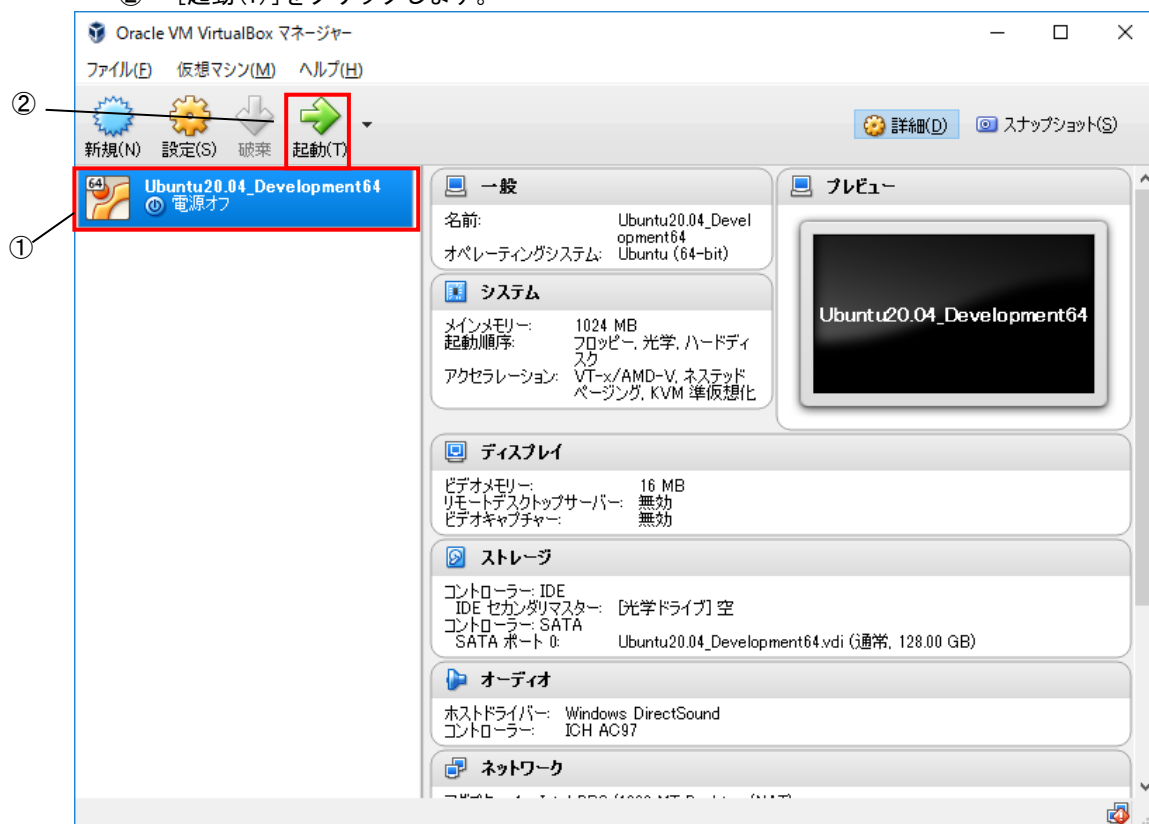


図 2-2-7-1. 仮想マシンの起動

- ③ 図 2-2-7-2 のようなユーザー一覧画面が表示されます。
「asusr」をクリックします。

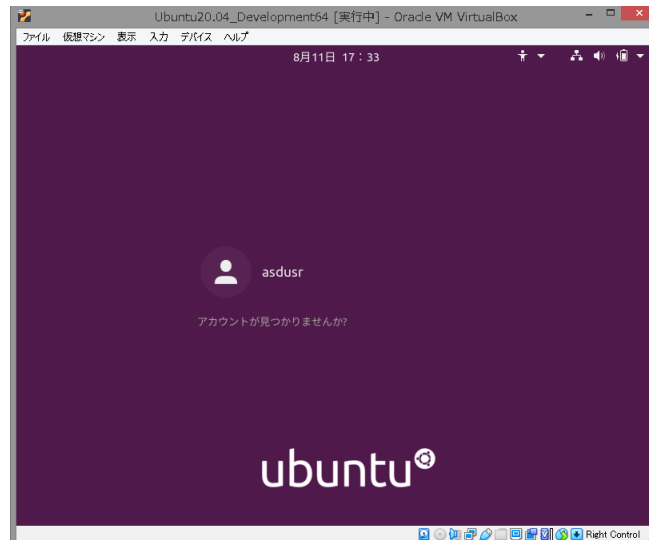


図 2-2-7-2. ユーザー一覧画面

- ④ パスワードの入力を求められるので、下記のパスワードを入力します。
パスワード : asusr

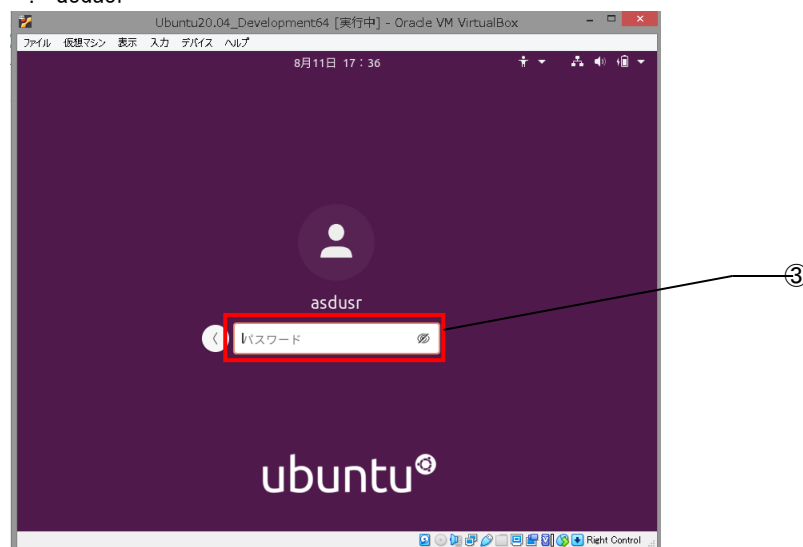


図 2-2-7-3. パスワード入力画面

図 2-2-7-4 のような仮想マシンのデスクトップ画面になり、ゲスト OS が使用できるようになります。

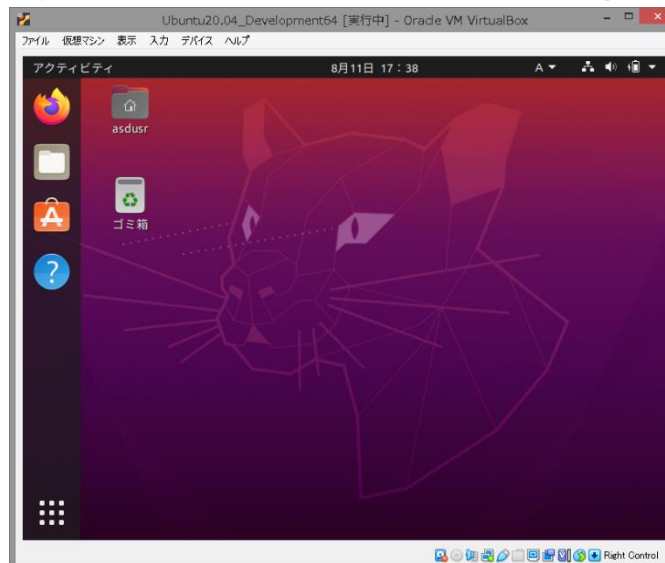


図 2-2-7-4. デスクトップ画面

2-2-8 開発環境の各種設定について

ここでは、開発環境を起動後に設定する項目について説明します。

● 画面解像度設定

ゲスト OS の画面解像度は、初期状態で 800x600 になっています。以下の手順で、画面の解像度を変更できます。

- ① デスクトップ上で右クリックします。コンテキストメニューが表示されるので、「ディスプレイの設定」を選択します。

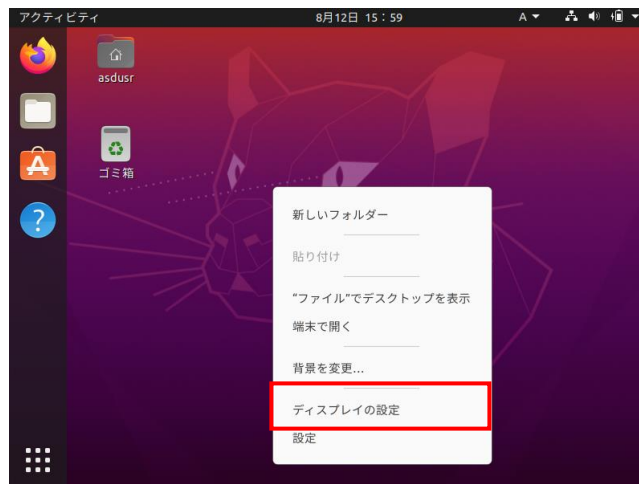


図 2-2-8-1. デスクトップ画面

- ② ディスプレイの設定画面が表示されます。解像度の欄にある▼をクリックし、使用しているモニタに合わせた解像度を選択します。



図 2-2-8-2. ディスプレイ設定画面

- ③ 「適用」ボタンをクリックして解像度を変更します。



図 2-2-8-3. 変更の適用

- ④ 「変更を保存」をクリックすることで、変更した解像度が保存されます。

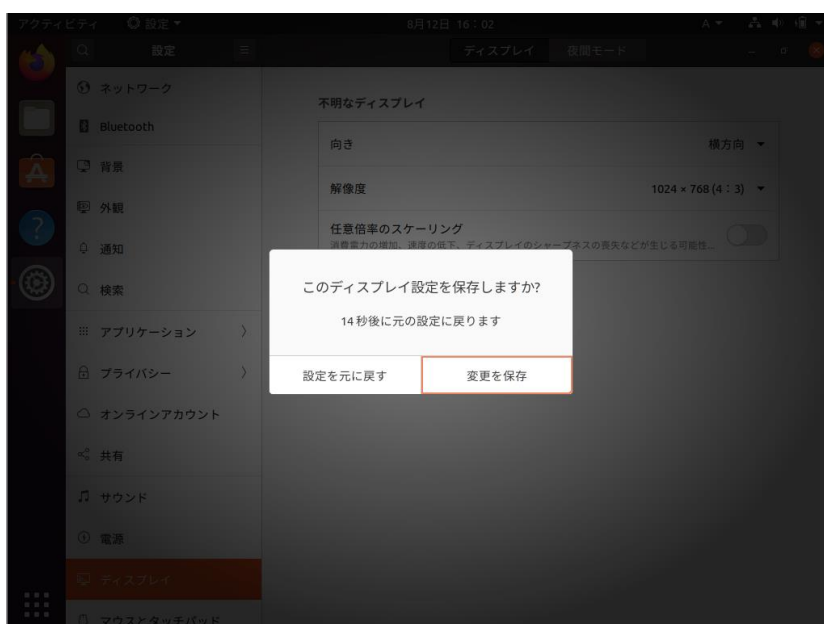


図 2-2-8-4. 変更の保存

- ネットワーク設定

ゲスト OS でネットワークを使用する際、もし接続できない場合や、IP 固定で使用したい場合は下記の方法で設定してください。

① 画面右上の「▼」をクリックし、「設定」を選択します。

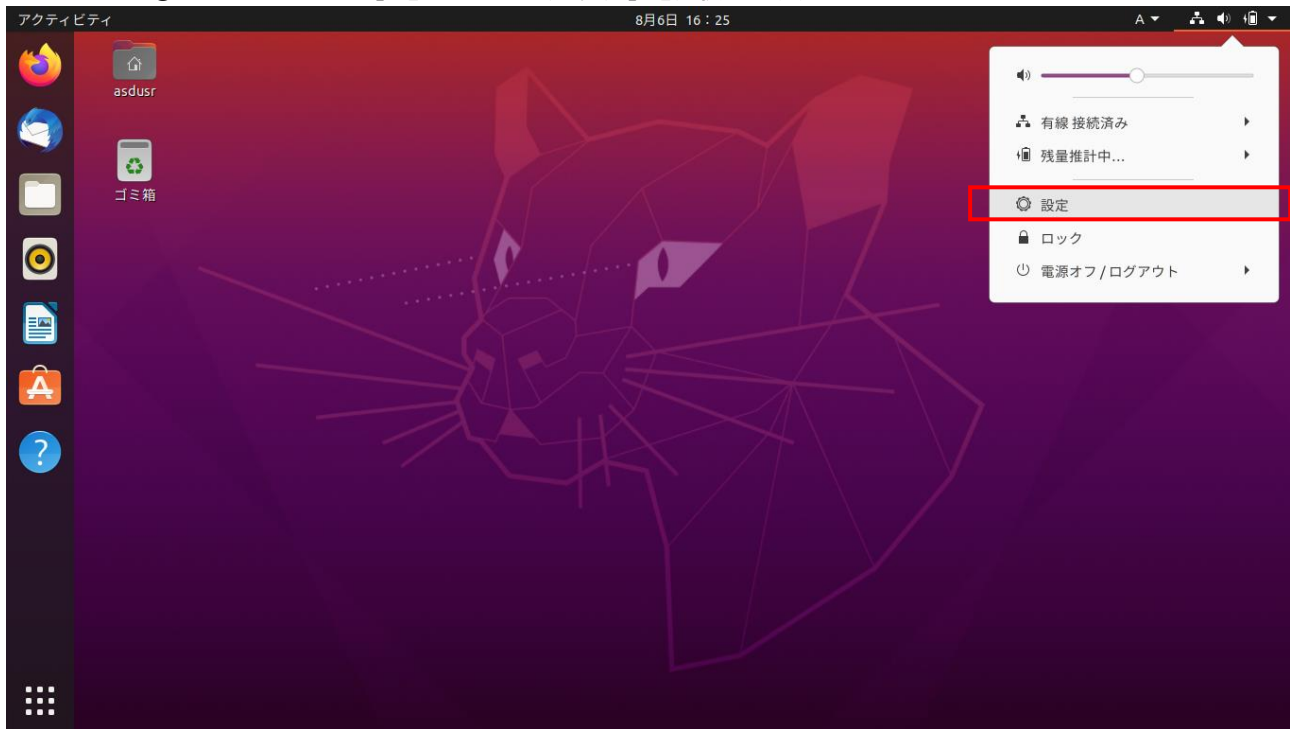



図 2-2-8-5. ネットワーク設定画面の起動

② 図 2-2-8-6 のようなネットワーク設定画面が起動します。認識している LAN の一覧が表示されています。設定したい接続を選択して  ボタンをクリックします。

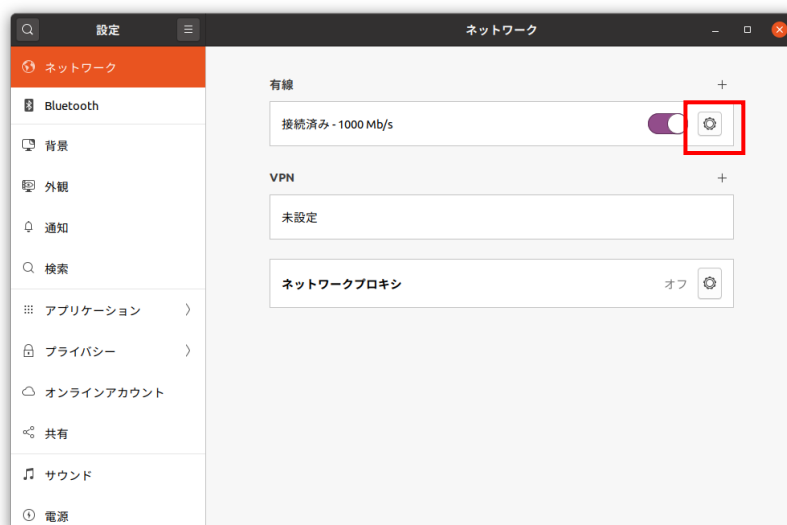


図 2-2-8-6. ネットワーク設定画面

- ③ 図 2-2-8-7 のような有線 LAN の設定画面が起動します。
接続しているネットワークの環境に合わせた設定を行ってください。



図 2-2-8-7. 有線 LAN 設定画面

- ④ 「OK」をクリックすることで、自動的に再接続されます。設定されている IP アドレスを確認する場合は、コンソールウィンドウを起動して下記のコマンドを実行してください。
- ・現在の設定を見る場合

```
$ ip a
```

IP アドレスの確認方法について、従来の Linux ディストリビューションでは、IP アドレスを確認するために、「ifconfig」というコマンドを使用されていました。しかし、「ifconfig」を含む「net-tools」というパッケージは、メンテナンスされていないため、数年前に非推奨なパッケージになりました。今日の Linux ディストリビューションでは、「net-tools」パッケージはインストールされておらず、「ifconfig」コマンドも使えません。その代わりに、「iproute2」（「ip」コマンドの正式名）が正式採用されています。

「ifconfig」コマンドに対応される「ip」コマンドは以下になります。

	ifconfig	iproute2
各種インターフェースの設定と表示	ifconfig	ip a (ip addr show)
インターフェースの起動	ifconfig eth0 up	ip link set eth0 up
インターフェースの停止	ifconfig eth0 down	ip link set eth0 down

「ip」コマンドは、「ifconfig」よりも、豊富な機能を搭載しています。詳細は、インターネット等で確認してください。

2-2-9 EC1G-01x/02x 用開発環境のディレクトリ構成について

EC1G-01x/02x 用開発環境のディレクトリ構成について説明します。EC1G-01x/02x 用開発環境用開発環境のディレクトリ構成をリスト 2-2-9-1 に示します。

リスト 2-2-9-1. EC1G-01x/02x 用開発環境のディレクトリ構成

```

usr---+---lib-----+---aarch64-linux-gnu
  +---local+---+---tools-arm64+---+---samples
    +---source+---+---apl
      +---kernel
      +---qt5
      +---widestudio
    +---toolchain
  +---tools-imx8mq+---+---source+---+---kernel
    +---u-boot
    +---toolchain
opt+---+---Qt
  
```

また、これらのディレクトリの内容を表 2-2-9-1 に示します。

表 2-2-9-1. ディレクトリの内容 (tools-arm64)



ディレクトリ名	内 容
/usr/lib/aarch64-linux-gnu	ARM CPU 用 USR ライブラリ
/usr/local/tools-arm64	EC1G-01x/02x 用開発環境が格納されています。
/usr/local/tools-arm64/samples	EC1G-01x/02x 用サンプルが格納されています。
/usr/local/tools-arm64/source	EC1G-01x/02x 用のデバイスドライバやアプリケーションソースが格納されています。
/usr/local/tools-arm64/source/apl	EC1G-01x/02x 用のアプリケーションです。 ASD コンフィグツールのソースが格納されています。
/usr/local/tools-arm64/source/qt5	EC1G-01x/02x 用 qt5 フレームワークソースです。
/usr/local/tools-arm64/source/widestudio	EC1G-01x/02x 用 WideStudio ソースです。
/usr/local/tools-arm64/toolchain	ARM 用クロスコンパイル環境
/usr/local/tools-imx8mq	i.MX8MQ 用カーネル、ブートローダーのソースです。
/usr/local/tools-imx8mq/source/kernel	i.MX8MQ 用カーネルのソースです。
/usr/local/tools-imx8mq/source/u-boot	i.MX8MQ 用ブートローダーのソースです。
/usr/local/tools-imx8mq/toolchain	i.MX8MQ 用クロスコンパイル環境
/opt/Qt	Qt 開発環境

2-3 WideStudio/MWT によるアプリケーション開発

WideStudio/MWT はデスクトップアプリケーションを迅速に作成することのできる統合開発環境です。詳細は WideStudio ホームページ (<http://www.widestudio.org/index.html>) を参照してください。

EC1G-01x/02x 用開発環境に組込まれている WideStudio/MWT は V3.98-7 をベースに EC1G-01x/02x 用の環境設定を加えてコンパイルしたものです。ここで、WideStudio/MWT で簡単なプログラムをコンパイルして実際に動作させます。

2-3-1 WideStudio の起動

画面左下の  のアイコンをクリックし、アプリケーション一覧の  をクリックすることで WideStudio が起動します。

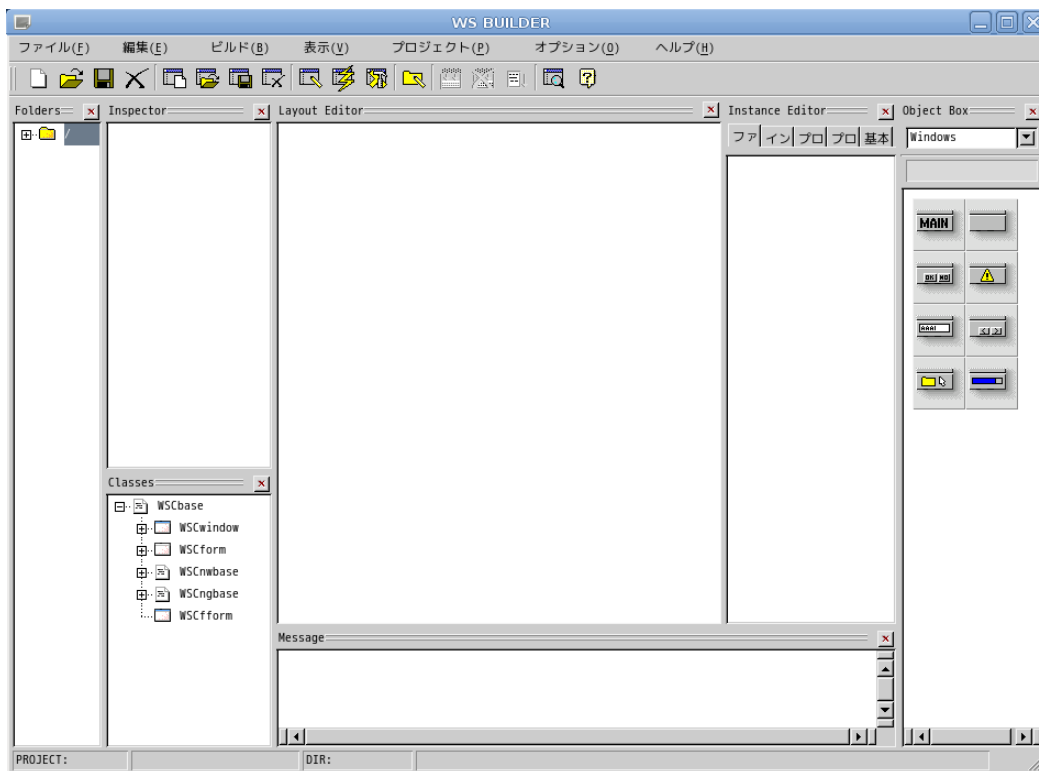



図 2-3-1-1. WideStudio 起動

2-3-2 プロジェクトの新規作成

アプリケーションを作成するためのプロジェクトを以下の手順で作成します。



図 2-3-2-1. 新規プロジェクト作成

 をクリックするか、メニューの「プロジェクト(P)」→「新規プロジェクト(N)」をクリックすることで、
図 2-3-2-2 のような画面が表示されます。ここでプロジェクト名称を記入してください。本項ではデフォルト
の newproject とします。

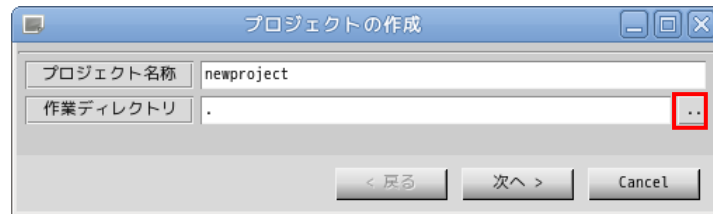



図 2-3-2-2. プロジェクト作成画面

 をクリックすることで、図 2-3-2-3 のようなファイル選択ダイアログが表示されます。

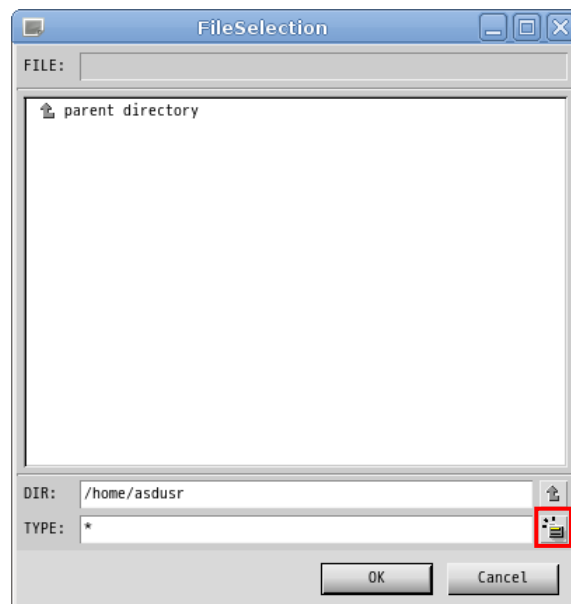



図 2-3-2-3. ファイル選択画面

DIR に「/home/asdusr」を指定します。sample というディレクトリを作成するために、 をクリックし、「sample」と入力し「OK」ボタンをクリックします。

※注：入力はマウスカースールを入力ダイアログ上に持っていった上で行ってください。

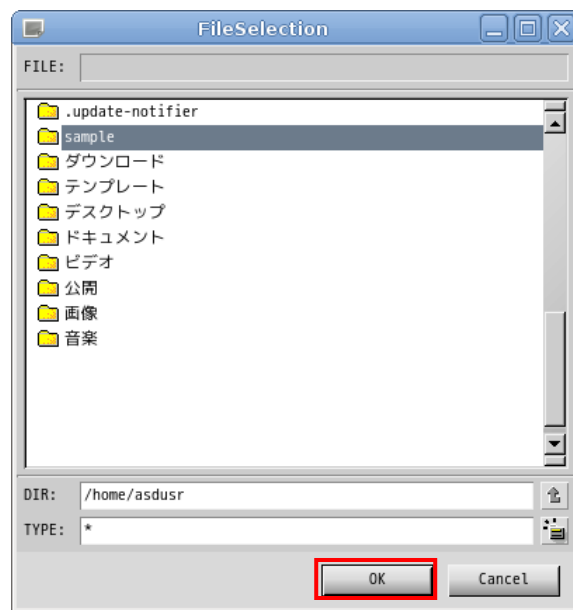


図 2-3-2-4. sample ディレクトリの作成

sample ディレクトリをダブルクリックし、DIR が「/home/asdusr/sample」と指定されたことを確認して「OK」ボタンをクリックします。

プロジェクト名と作業ディレクトリの指定が完了しましたので「次へ」ボタンをクリックします。

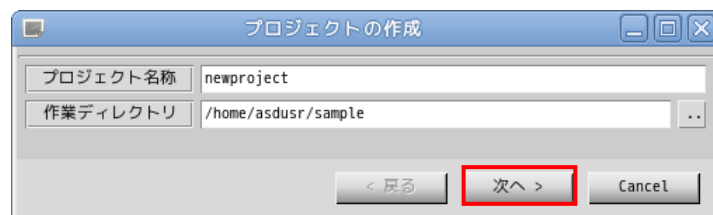


図 2-3-2-5. プロジェクト名と作業ディレクトリの設定完了

プロジェクトの種別を選択します。通常のアプリケーションを作成するので、そのまま「次へ」をクリックします。

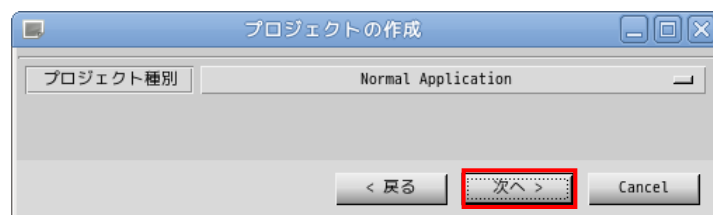


図 2-3-2-6. プロジェクト種別の選択

アプリケーションで使用するロケール種別（文字コード）、言語種別（プログラミング言語）を選択します。本項では、ロケール種別に「UTF8」を、言語種別に「C/C++」を選択します。

EC1G-01x/02x で動作確認したロケール種別は、ユニコード (UTF8) と日本語 (EUC) と日本語 (SJIS) のみです。また、言語種別は C/C++ のみサポートしています。

※注：ロケール種別については注意が必要です。例えば、シリアル通信を通して、SJIS の文字列が送られてきてそれを表示する場合、ここでのロケール種別は SJIS にします。

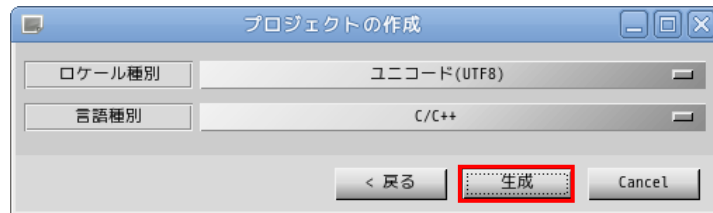


図 2-3-2-7. ロケール種別と言語種別を選択

以上でプロジェクトの作成は完了です。

2-3-3 アプリケーションウィンドウの作成

前項でプロジェクトの作成が完了しましたので、次にアプリケーションウィンドウを作成します。

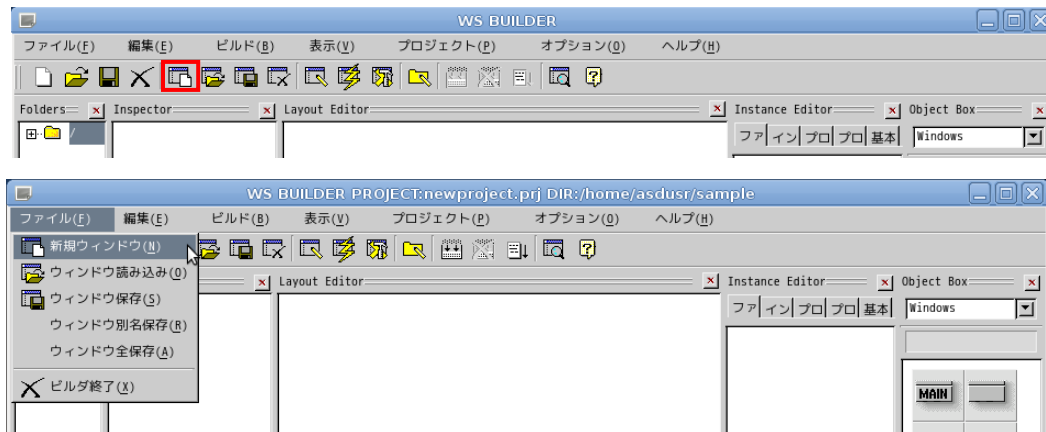



図 2-3-3-1. 新規ウィンドウ作成

 をクリックするか、メニューの「ファイル (F)」→「新規ウィンドウ (N)」をクリックすることで、図 2-3-3-2 の画面が表示されます。「通常のウィンドウ」を選択し、「次へ >」ボタンをクリックします。

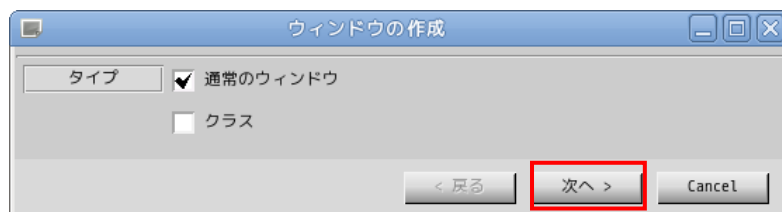


図 2-3-3-2. アプリケーションウィンドウタイプ選択

アプリケーションウィンドウの名称とプロジェクトに登録するかを選択します。名称は変数として使われるので空白のない英数字のみ有効です。ここではデフォルト設定のままとします。設定を変更せずに「次へ >」をクリックします。

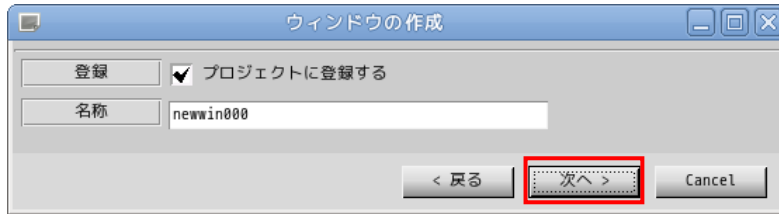


図 2-3-3-3. アプリケーションウィンドウ名称設定

テンプレートの選択を行います。あらかじめ用意されたテンプレートを選択することで、標準的なメニューやツールバーを持つアプリケーションウィンドウを作成することができます。今回はメニューを持たないウィンドウを作成するので、「なし」を選択して「生成」ボタンをクリックします。

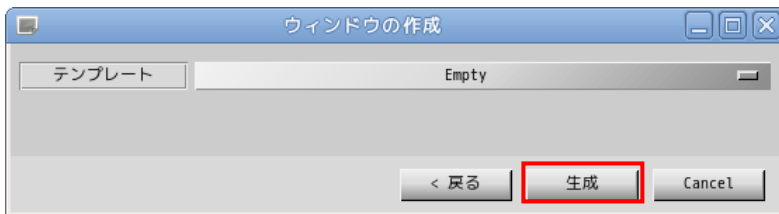


図 2-3-3-4. テンプレートの選択

これで、アプリケーションウィンドウが作成できました。作成したアプリケーションウィンドウをクリックし、「Instance Editor」の「プロパティ」タブをクリックすることで、アプリケーションウィンドウのプロパティが設定できるようになります。表 2-3-3-1 を参考にプロパティを変更してください。

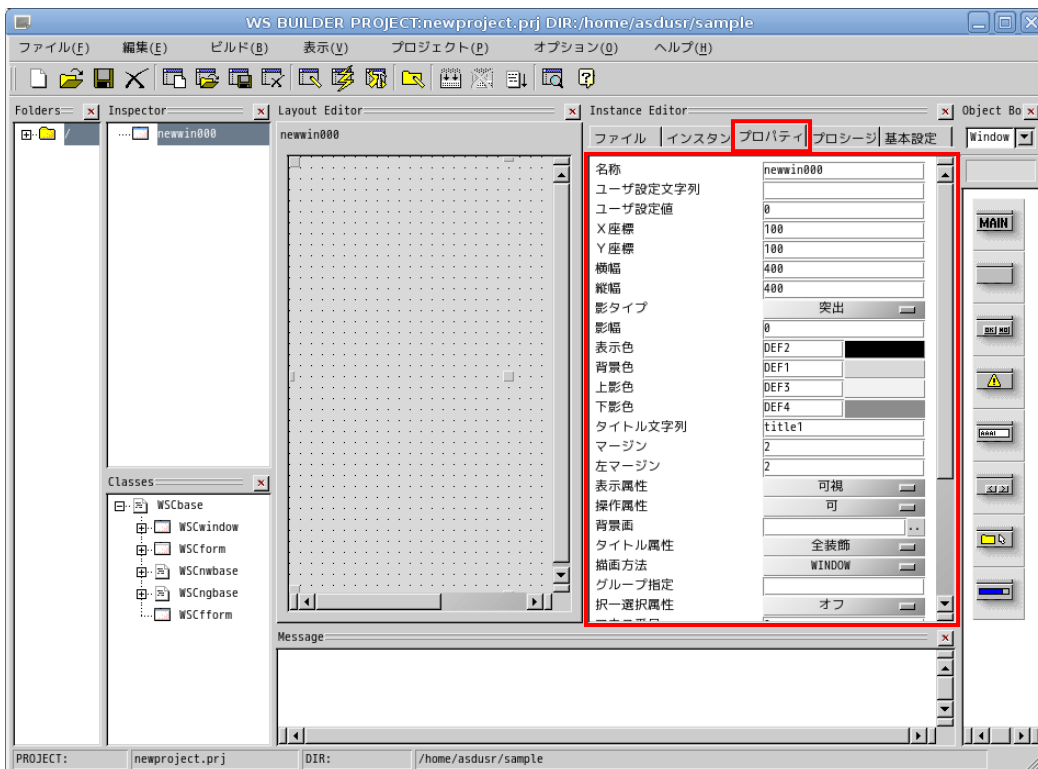


図 2-3-3-5. アプリケーションウィンドウの生成

表 2-3-3-1. アプリケーションウィンドウのプロパティ変更

プロパティ名	説明	設定値
X 座標	ウィンドウの初期起動 X 位置	0
Y 座標	ウィンドウの初期起動 Y 位置	0
横幅	ウィンドウの横幅	200
縦幅	ウィンドウの縦幅	200

以上で、アプリケーションウィンドウの作成は完了です。

2-3-4 部品の配置

アプリケーションウィンドウの上に部品を配置します。

ボタンを配置しますので、「Object Box」の「Commands」を選択し、ボタンオブジェクトをアプリケーションウィンドウにドラッグ&ドロップで配置します。

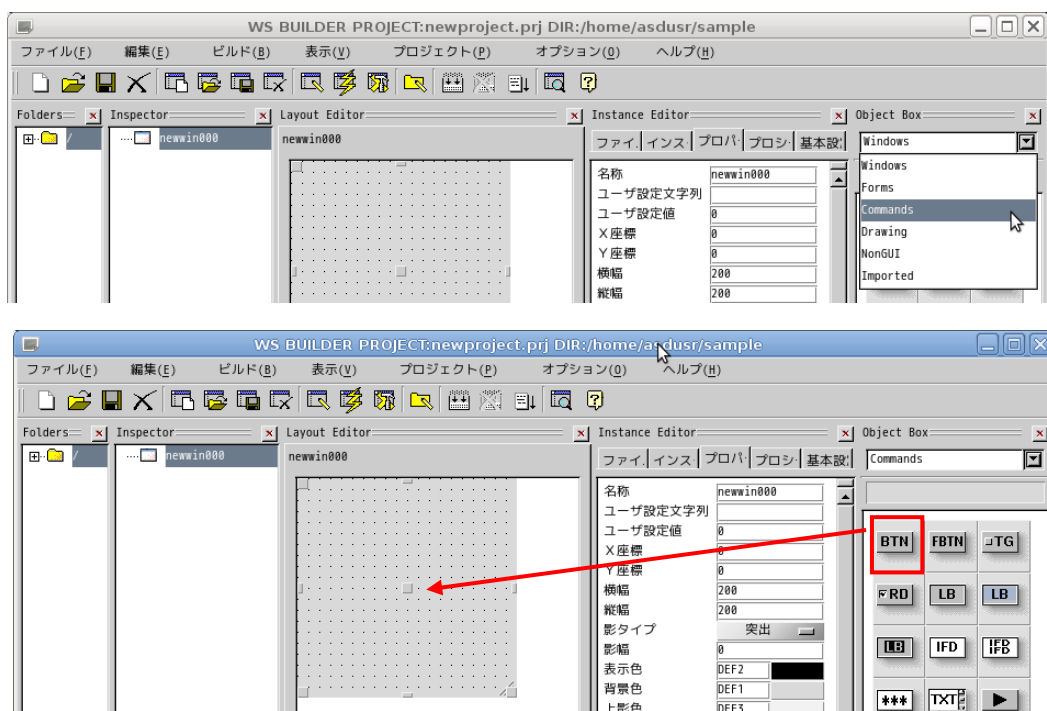


図 2-3-4-1. ボタンの配置

図 2-2-4-2 のようにボタンが配置されます。他の部品も同じようにオブジェクトボックスから目的のアイコンを選び出し、ドラッグ&ドロップすることでウィンドウ上に配置できます。

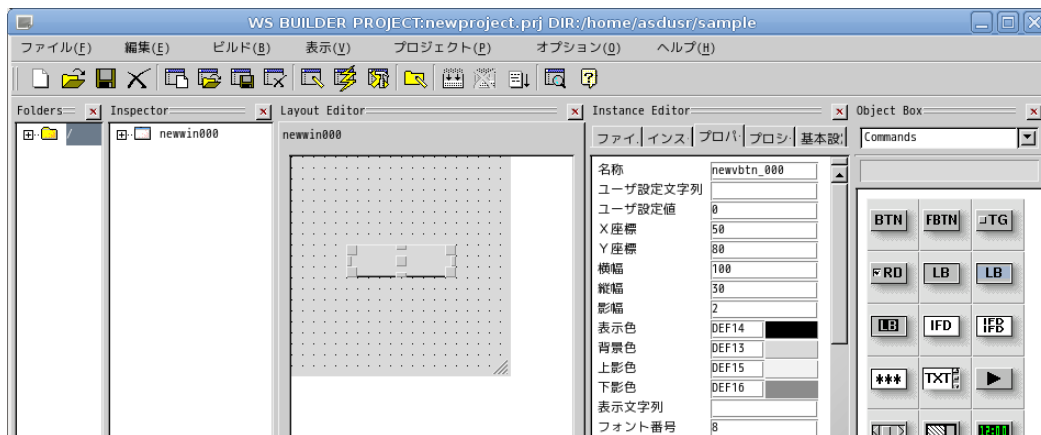


図 2-3-4-2. アプリケーションウィンドウに配置されたボタン

ボタンのプロパティを変更します。アプリケーションウィンドウ上に配置されたボタンをクリックし、「Instance Editor」の「プロパティ」タブをクリックすることで、ボタンのプロパティを設定できるようになります。表 2-3-4-1 を参考に値を変更してください。

表 2-3-4-1. ボタンのプロパティ変更

プロパティ名	説明	設定値
表示文字列	ボタンに表示される文字列の設定	PUSH

※注：ここでは、例としてウィンドウにボタンオブジェクトを貼り付けただけのテストサンプルを作成しています。他の部品の詳細については、ヘルプや WideStudio の書籍を参照してください。

※注：EC1G-01x/02x 用開発環境に組込まれていないライブラリを使用している部品についてはコンパイルできません。組込まれているライブラリと組込まれていないライブラリについては表 2-3-4-2 を参照してください。

表 2-3-4-2. WideStudio コンフィグ時に組込まれるライブラリ

ライブラリ名	状態	内容
OpenGL	No	3D グラフィックスのためのプログラムインターフェイス
JpegLib	HAS	JPEG ファイルを圧縮・伸張するためのライブラリ
PngLib	HAS	PNG ファイルを圧縮・伸張するためのライブラリ
XpmLib	HAS	XPM ファイルを圧縮・伸張するためのライブラリ
ODBC library	No	Open DataBase Connectivity の仕様に基づいたデータベースにアクセスするためのライブラリ
PostgreSQL development library	No	PostgreSQL と呼ばれる、オープンソース系データベースにアクセスするためのライブラリ
MySQL development library	No	MySQL と呼ばれる、オープンソース系データベースにアクセスするためのライブラリ
Python	HAS	Python と呼ばれる、オブジェクト指向スクリプト言語を扱うためのライブラリ
Python header	No	Python を扱うためのヘッダファイル
Ruby	No	Ruby と呼ばれる、オブジェクト指向スクリプト言語を扱うためのライブラリ
Perl	HAS	Perl と呼ばれる、インタプリタ形式のプログラム言語を扱うためのライブラリ

※注：No となっているライブラリは出荷時状態では組込まれていません。

2-3-5 イベントプロシージャの設定

ボタンのクリックという動作で実行されるプログラムを記述するには、ボタンオブジェクトにプロシージャと呼ばれるプログラムを設定します。

アプリケーションウィンドウ上のボタンをクリックし、「Instance Editor」の「プロシージャ」タブを選択することで、図 2-3-5-1 のような画面が起動します。

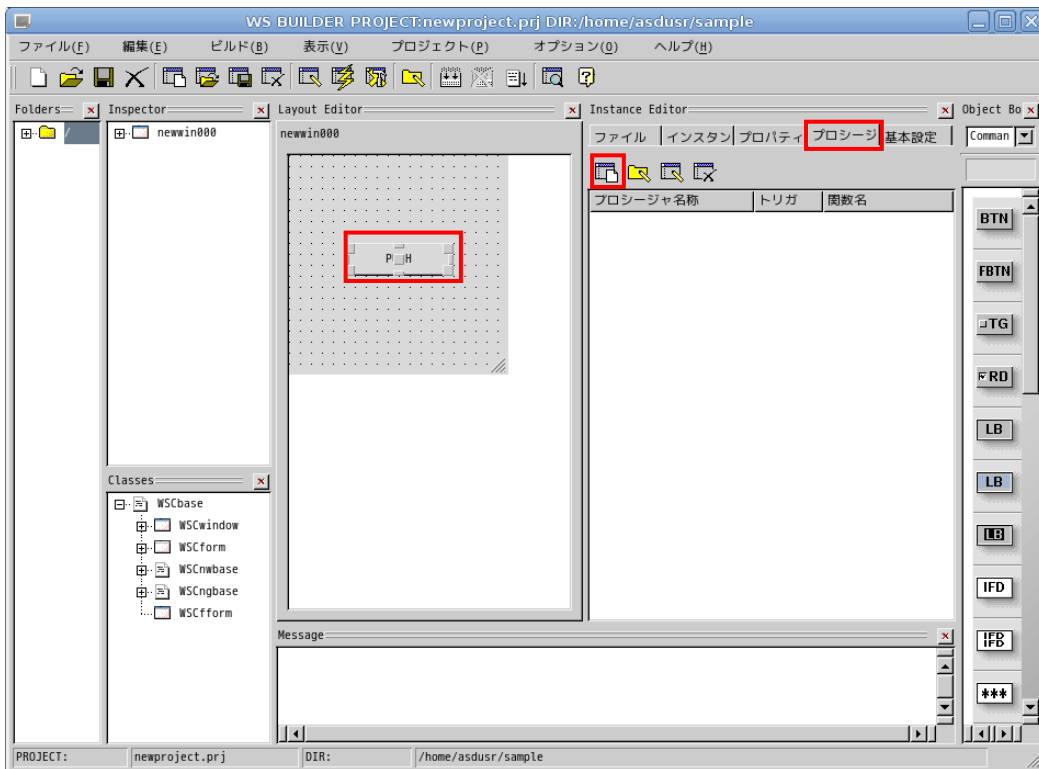



図 2-3-5-1. イベントプロシージャの作成

 をクリックすることで、図 2-3-5-2 のようなイベントプロシージャ作成ダイアログが表示されます。

プロシージャ名は、イベントプロシージャを識別するための名前です。今回は、「Btn_Click」と入力します。起動関数名は、イベント発生時に起動される C/C++ の関数名です。この関数に処理を記述します。今回はプロシージャ名と同じ「Btn_Click」と入力します。

起動トリガは、イベントの発生条件を選択します。今回は、ボタンを押して、離れたときに発生するイベントとして「ACTIVATE」を選択します。

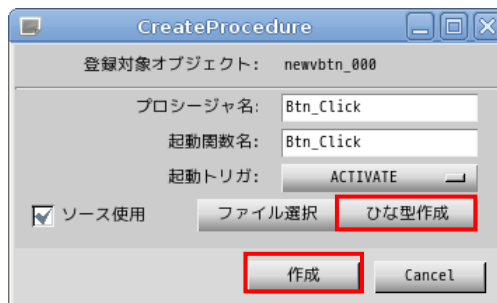


図 2-3-5-2. イベントプロシージャ作成ダイアログ

「ひな型作成」ボタンをクリックすることで、確認ダイアログが表示されるので「OK」ボタンをクリックします。これで、イベントプロシージャのソースコードが自動的に生成されます。「作成」ボタンをクリックします。この操作でイベントプロシージャを作成します。

図 2-3-5-3 のように「Instance Editor」の「プロシージャ」画面に、作成したイベントプロシージャが表示されます。プロシージャ名をダブルクリックすることで、エディタを起動し、処理を記述することができます。

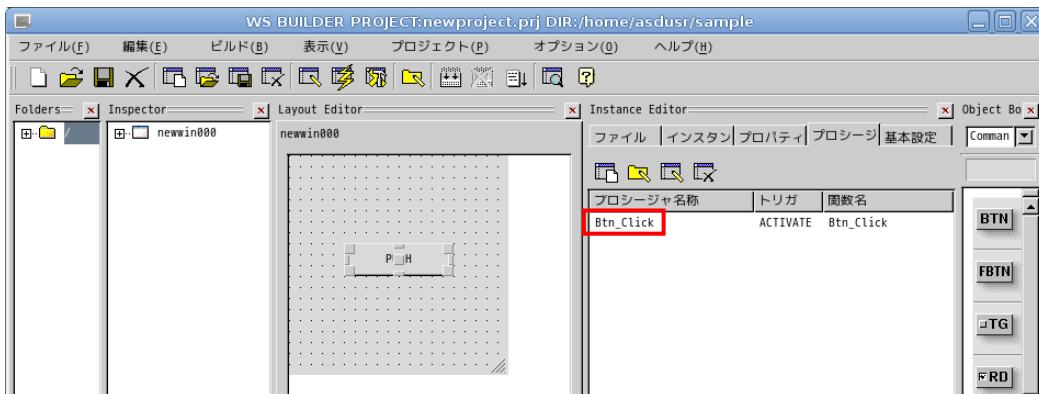


図 2-3-5-3. 作成されたイベントプロシージャ

2-3-6 イベントプロシージャの編集

ボタンをクリックしたときに、ボタンの表示文字列を変更するコードを記述します。イベントプロシージャの初期状態は図 2-3-6-1 のようになっています。

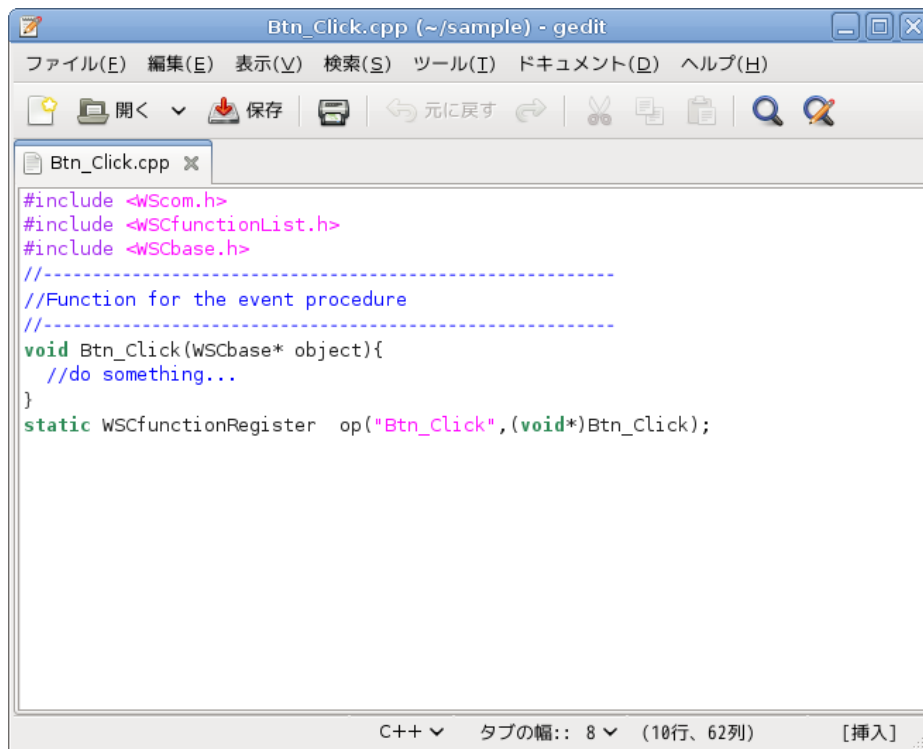


図 2-3-6-1. イベントプロシージャ初期ソースコード

リスト 2-3-6-1 のようにコードを変更します。

リスト 2-3-6-1. 表示文字列を変更するコード

```
#include <WSCom.h>
#include <WSCfunctionList.h>
#include <WSCbase.h>
//-----
//Function for the event procedure
//-----
void Btn_Click(WSCbase* object) {
    object->setProperty (WSNlabelString, "HELLO!");    //表示文字列変更
}
static WSCfunctionRegister op("Btn_Click", (void*)Btn_Click);
```

これで、ボタンをクリックしたら、「PUSH」が「HELLO!」となるプログラムができます。保存してエディタを終了します。以上でコーディングは完了です。

2-3-7 コンパイル

サンプルプログラムのビルドを行います。メニューの「ビルド(B)→リビルド(R)」をクリックしてください。プログラムのコンパイルが始まります。

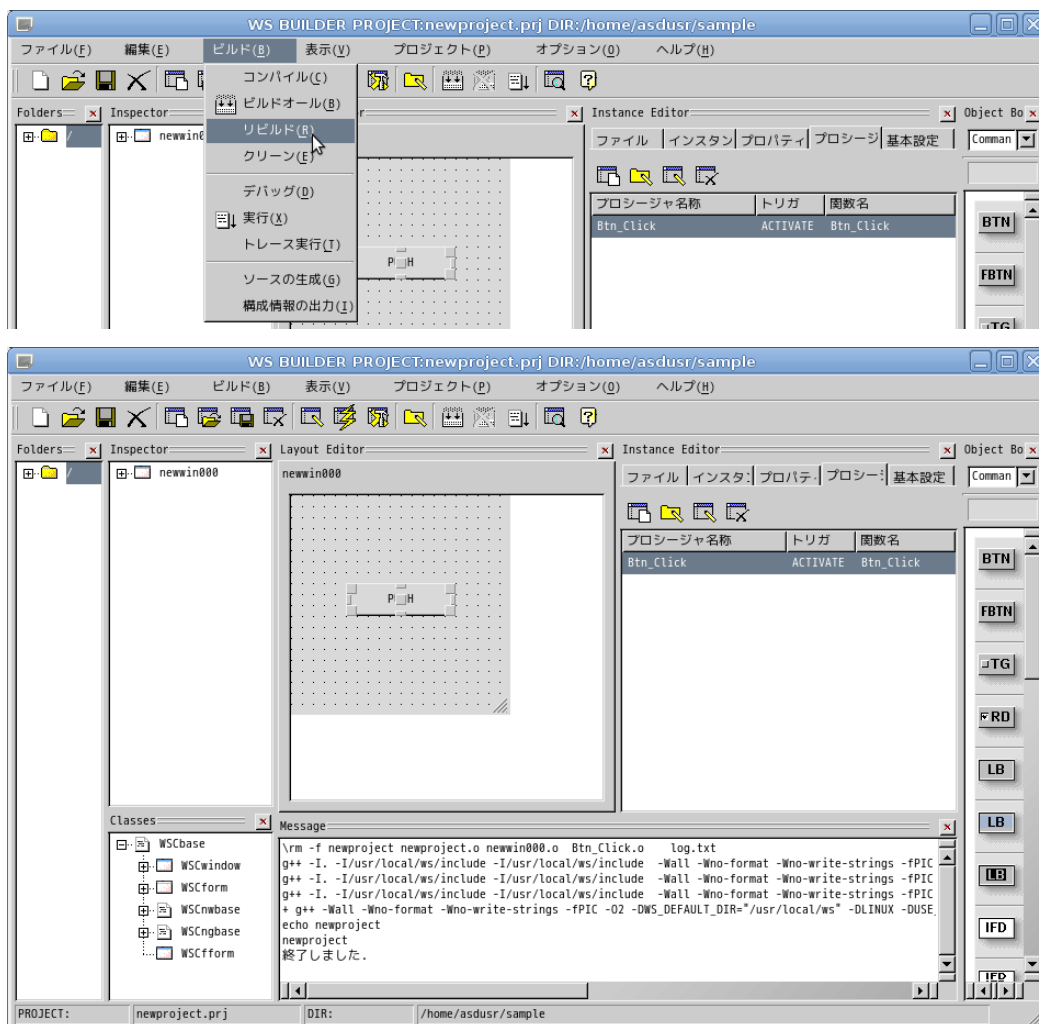


図 2-3-7-1. サンプルプロジェクトのコンパイル

コンパイル完了後、メニューの「ビルド(B)→実行(X)」をクリックしてください。サンプルプログラムが開発環境上で実行されます。「PUSH」ボタンをクリックして、「HELLO!」と表示されることを確認してください。

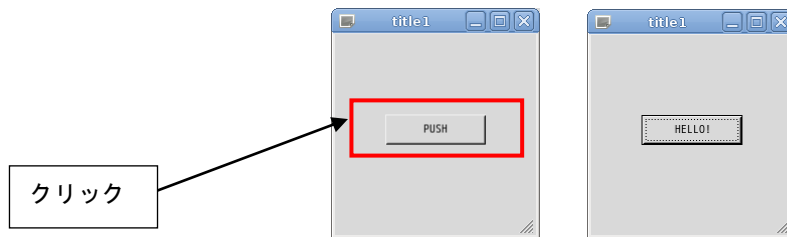


図 2-3-7-2. プログラム実行画面

動作確認後、プログラムを終了させてください。メニューの「ビルド(B)→実行中止(X)」で終了できます。「終了」または「X」をクリックした場合でも、メニューの「ビルド(B)→実行中止(X)」をクリックしてください。

「プロジェクト(P) → プロジェクト設定(E)」をクリックすると、プロジェクト設定ウィンドウが表示されます。「基本設定」タブをクリックすると、図 2-3-7-3 のような画面が表示されます。

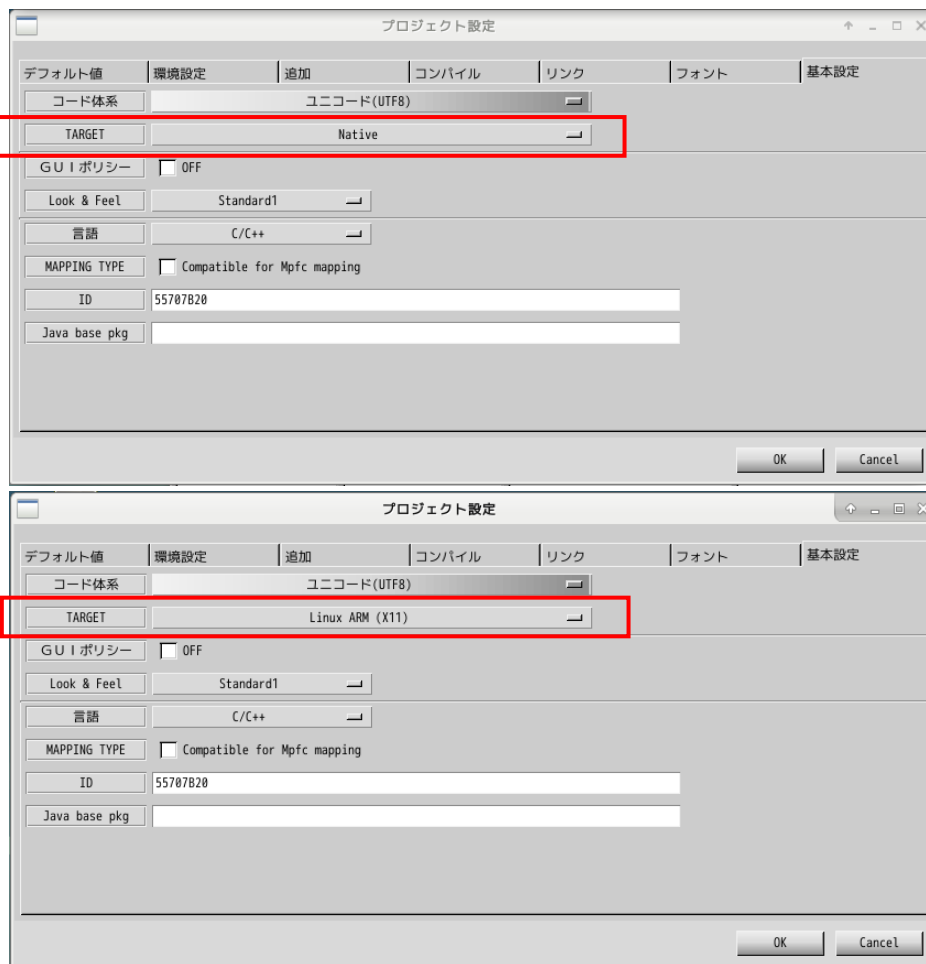


図 2-3-7-3. プロジェクト設定 (基本設定)

TARGET を「Native」と設定することでインテル CPU 用のプログラムを生成することができます。
TARGET を「Linux ARM (X11)」と設定することで、NXP 製 i.MX8M 用のプログラムを生成することができます。

2-3-8 ファイルの転送

開発環境にて作成した実行プログラムをアルゴシステム製端末に移して実行します。

実行プログラムだけでなく、設定ファイルや画像データ等、アルゴシステム製端末にデータを転送するには USB メモリでの転送と、LAN 経由での転送の 2 種類の方法があります。

●USB メモリでの転送

- ① 開発環境パソコンに USB メモリを挿入します。正常に認識されるとデスクトップ上にハードディスクのアイコンが表示され、内容が表示されます。
- ② 転送するデータ（サンプルプログラム）を USB メモリにコピーします。
- ③ ハードディスクアイコンを右クリックし「アンマウント」をクリックします。正常に USB メモリがアンマウントされれば、アイコンが消えますので、USB メモリを抜いてください。
- ④ 転送するデータが保存された USB メモリをアルゴシステム製端末に挿入します。
- ⑤ USB メモリが自動でマウントされます。ホームディレクトリ上で、以下のコマンドを実行します。
***の部分は USB メモリのデバイス名です。

```
# cp /media/asdusr/**/newproject  
# ./newproject
```

- ⑥ ⑤の最後のコマンドで、作成したサンプルプログラムが実行されます。
「PUSH」ボタンを押下したら、「HELLO!」となることを確認してください。
- ⑦ USB メモリを抜くときは、以下のコマンドを入力します。

```
# sync  
# umount /media/asdusr/**/
```

これで USB メモリがアンマウントされます。

※注：アンマウントせずに USB メモリを抜くとファイルが破損する可能性があります。

●LAN 経由でファイル転送

LAN 経由でファイル転送する前に、実行環境上で SSH による認証を有効にする必要があります。
以下、実行環境上で SSH 認証を有効化する手順を記します。

- ① SSH の設定ファイルをエディタで開きます。

```
$ sudo vi /etc/ssh/ssh_config
```

- ② 以下の行を編集します。

```
PasswordAuthentication no
```



```
PasswordAuthentication yes
```


- ③ sshd を再起動します。

```
$ sudo systemctl restart sshd.service
```

以下、LAN 経由でファイル転送する手順を記します。

- ① LAN ケーブルを実行環境に接続します。
実行環境と開発環境を HUB 無しで接続する場合、クロスケーブルが必要です。
接続する端末の IP アドレスを確認しておいてください。

※注：実行環境と開発環境の IP アドレスは、同一のネットワークアドレスと異なるホストアドレスを指定する必要があります。

- ② ゲスト OS のデスクトップ左下の  アイコンクリックで表示されるアプリケーション一覧から、「gFTP」を選択することで「gFTP」というファイル転送クライアントが起動されます。

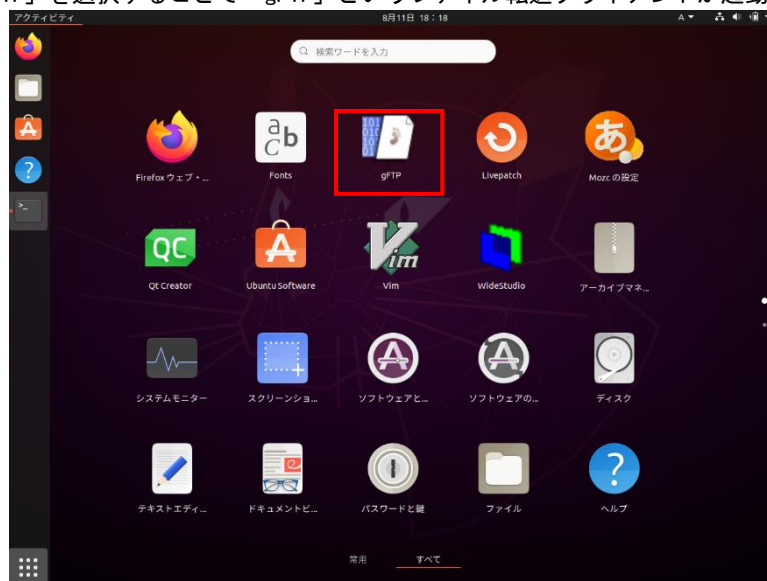


図 2-3-8-1. ファイル転送クライアントの起動

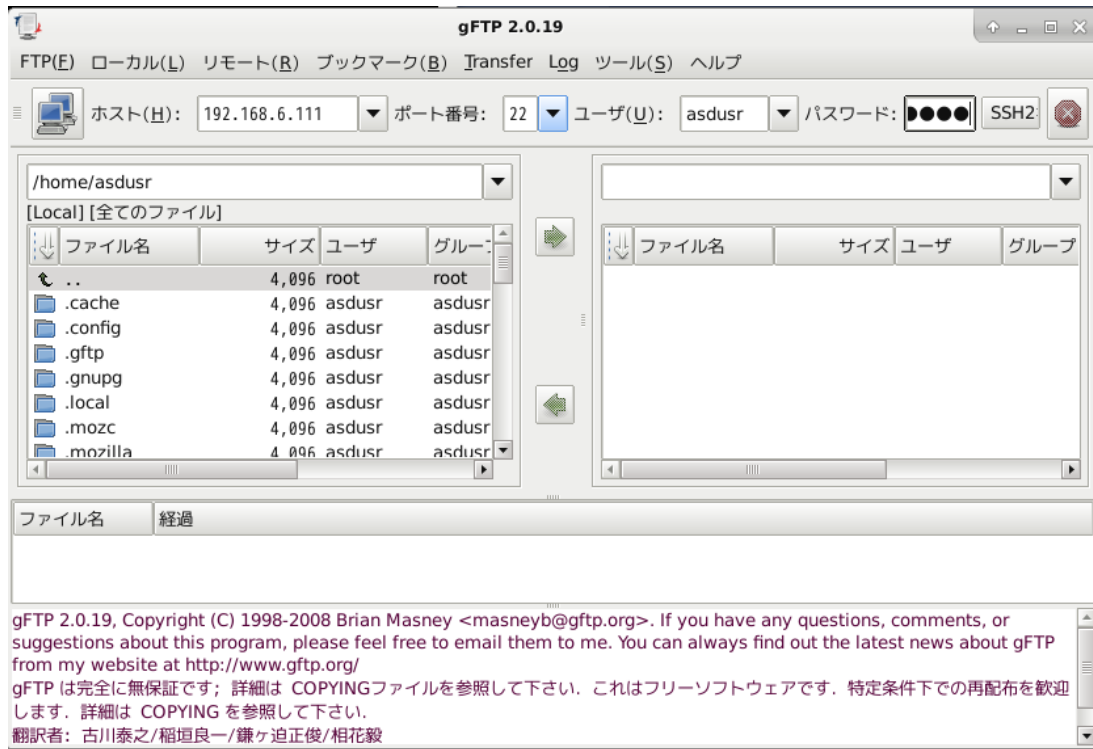
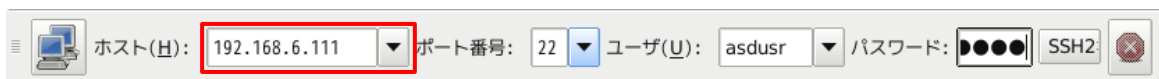
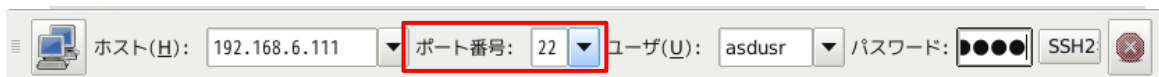


図 2-3-8-2. ファイル転送クライアント画面

- ① 接続先である EC1G-01x/02x の IP アドレスを指定します。



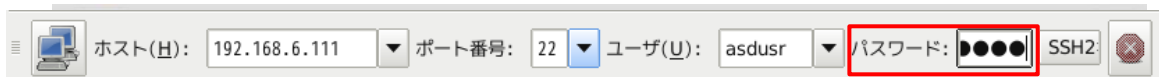
- ② ポート番号は「22」を設定します。



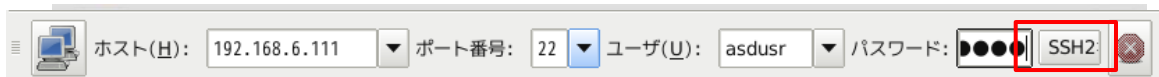
- ③ EC1G-01x/02x の接続ユーザ名を指定します。デフォルトでは「asdusr」となっています。



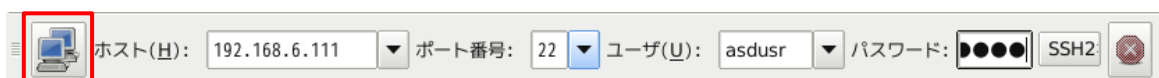
- ④ EC1G-01x/02x への FTP 接続パスワードを指定します。デフォルトでは「asdusr」となっています。



- ⑤ 通信タイプを指定します。「SSH2」としてください。



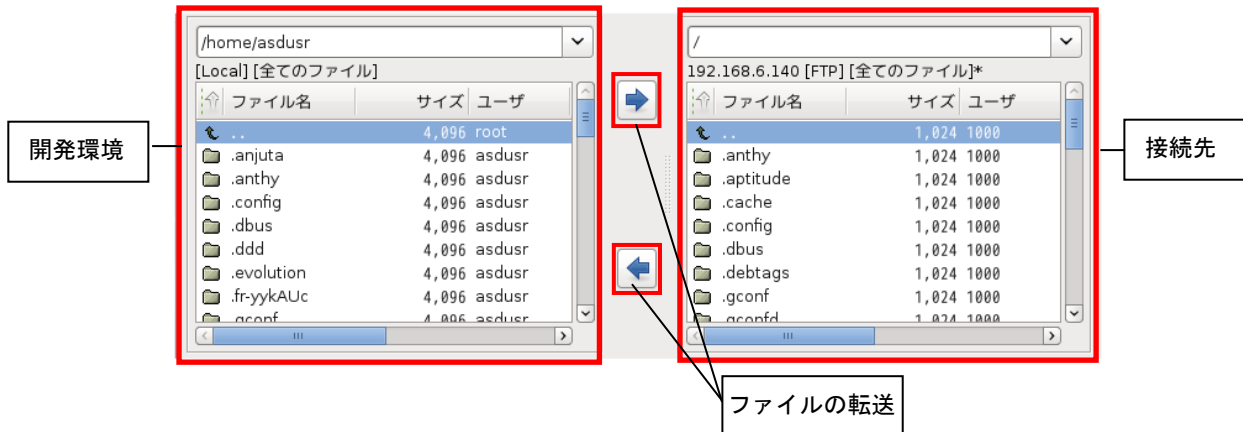
- ⑥ EC1G-01x/02x に対して、接続と切断を行います。



正常に接続されると、EC1G-01x/02x の「/home/asdusr」ディレクトリがルートディレクトリとしてリスト表示されます。EC1G-01x/02x のデフォルト設定ではセキュリティ上「/home/asdusr」ディレクト

りより上のディレクトリには移動できないように設定されています。

移動したいファイルを選択して、矢印キーをクリックすることで、開発環境側と実行環境側でファイルの転送が可能です。



以上でアプリケーション開発の説明は終了です。

2-3-9 WideStudio/MWT の開発例

本項では、弊社が行った開発事例を列挙します。アプリケーション開発の参考資料としてご使用ください。

● X Window System 上で全画面表示させる方法

X Window System 上でアプリケーションを実行する場合、起動された GUI アプリケーションには必ずタイトルバーがつけられます。これは、X Window System 上で Window Manager が動作しており、Window Manager がタイトルバーをつけ、複数のアプリケーションを管理しているためです。組込み用途で使用する場合、メイン画面はタイトルバーをつけずに全画面表示させる場合があります。表 2-3-9-1 のプロパティを変更することで、タイトルバーのついていないアプリケーションを開発することができます。この場合、起動したプログラムが常に前に表示されるので、アプリケーションウィンドウの表示、非表示を切り替えて使用する必要があります。

表 2-3-9-1. アプリケーションウィンドウのプロパティ変更

プロパティ名	説明	設定値
タイトル属性	ウィンドウのタイトルバー属性の設定	WM 管理外
終了	ウィンドウを非表示にしたとき終了する/しないの設定	オフ

サンプルとして作成した、HELLO プログラムを修正して試してみます。アプリケーションウィンドウのプロパティを表 2-3-9-1 に書かれているように変更してください。Btn_Click のイベントプロシージャをリスト 2-3-9-1 のように記述してください。

このプログラムをコンパイルして実行すると、タイトルバーが付いていない画面が起動されます。ボタンを押下することで、メイン画面が非表示になり、xeyes というプログラムが起動されます。xeyes が終了されると、再度メイン画面が表示されます。

リスト 2-3-9-1. 別アプリケーションの起動サンプル

```
#include "stdlib.h"
#include <WScom.h>
#include <WSCfunctionList.h>
#include <WSCbase.h>
#include "newwin000.h"
//-----
//Function for the event procedure
//-----
void Btn_Click(WSCbase* object) {
    newwin000->setVisible(False);           //アプリケーションウィンドウの非表示
    system("xeyes");                        //xeyes というプログラムを起動する
    newwin000->setVisible(True);           //アプリケーションウィンドウの表示
}
static WSCfunctionRegister op("Btn_Click", (void*)Btn_Click);
```

●WideStudio のフォント設定について

WideStudio には次の 4 系統のフォント設定があります。

■X11 系の設定

FONT0: サイズ フォント名 Weight(0: 無し 1: bold) slant(0: 無し 1: イタリック)

[例] FONT0: 10 * 0 0

FONT1: 12 * 0 0

EC1G-01x/02x の/etc/xunicoderc にて詳細なフォントを定義

■T-Engine 系の設定

FONT0: サイズ フォント ID

[例] FONT0: 10 60c6

FONT1: 12 60c6

■DirectFB 系 Linux フレームバッファ系、T-Engine フレームバッファ系

FONT0: font0

[例] FONT0: font0

FONT1: font1

/etc/wsfnts にてフォントファイルを定義

■Windows 系の設定

Windows フォントパラメータをカンマ区切りで列挙

WideStudio のフォント設定は、プロジェクト設定のフォント設定タブで設定します。また、「prj ファイル」に記録されているフォント設定の値を直接変更することでフォント設定を行うこともできます。

※注：実行環境上のフォントと開発環境上のフォントは設定が変更されている可能性があるため、開発時に見えているフォントと実際に動作させたときのフォントが異なる可能性があります。

●X Window System の場合

実行環境上で WideStudio アプリケーションを動作させる際のフォント設定方法を以下に示します。

リスト 2-3-9-2. X11 用のフォント設定例 (prj ファイル)

```
#FONT0
12 東風ゴシック 0 0 /* サイズ 12 フォント名 東風ゴシック Weight 無し slant 無し */
#FONT1
14 東風ゴシック 0 0 /* サイズ 14 フォント名 東風ゴシック Weight 無し slant 無し */
#FONT2
16 東風ゴシック 0 0 /* サイズ 16 フォント名 東風ゴシック Weight 無し slant 無し */
#FONT3
18 東風ゴシック 0 0 /* サイズ 18 フォント名 東風ゴシック Weight 無し slant 無し */
#FONT4
24 東風ゴシック 0 0 /* サイズ 24 フォント名 東風ゴシック Weight 無し slant 無し */
#FONT5
30 東風ゴシック 0 0 /* サイズ 30 フォント名 東風ゴシック Weight 無し slant 無し */
#FONT6
34 東風ゴシック 0 0 /* サイズ 34 フォント名 東風ゴシック Weight 無し slant 無し */
#FONT7
36 東風ゴシック 0 0 /* サイズ 36 フォント名 東風ゴシック Weight 無し slant 無し */
```

prj ファイル

#FONT0

12 * 0 0

サイズ 12 フォント名 * Weight 0: 無し slant 0: 無し
1: 太字 1: 斜体

WideStudio Ver3.98-4 から XFT ライブラリを利用し、アンチエイリアスの効いたフォント表示を行うことが

できます。

EC1G-01x/02x 用開発環境では、文字マップというプログラムを使用することで TrueType フォントを確認することができます。起動方法と画面を図 2-3-9-1 に示します。



フォントリスト

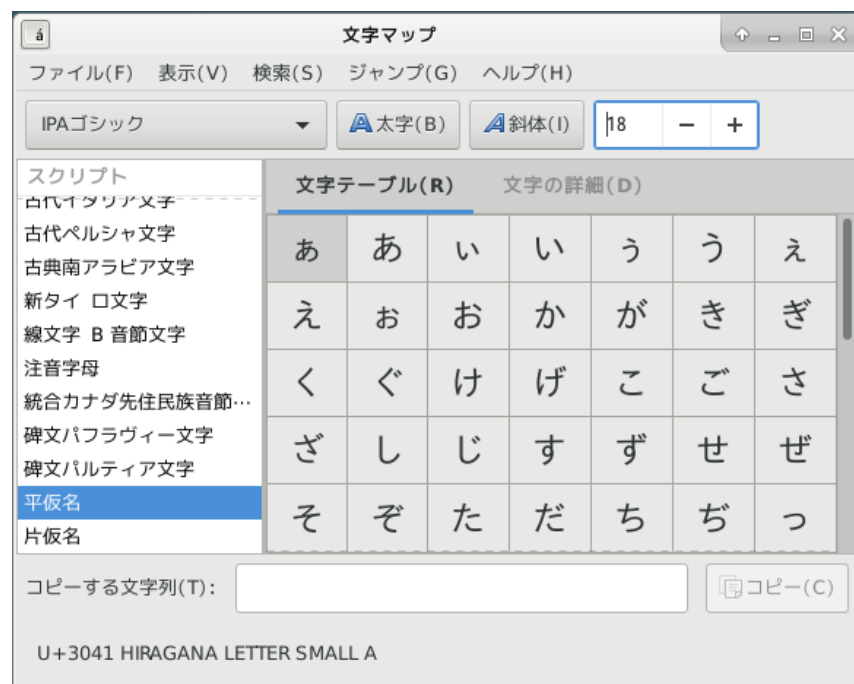


図 2-3-9-1. 文字マップツール

2-4 Qt によるアプリケーション開発

2-4-1 Qt とは

Qt「キョウト」は、クロスプラットフォームアプリケーションフレームワークで、同一のソースコードから複数のプラットフォームで動作するアプリケーションを開発することができます。

Qt は C++ で開発されています。

2-4-2 Qt ライセンス

Qt には 3 つのライセンスが用意されています。

◆商用ライセンス

オープンソースのライセンス義務なしに、自身の条件でソフトウェアを作成し、配布するための完全な権利を与えます。商用ライセンスでは、公式の Qt サポートにアクセスし、Qt 社との戦略的関係を密にして、開発目標が確実に達成されるようにします。

◆GPL ライセンス

GPL は GNU General Public License の略称で、プログラムの複製物を所持しているものに、以下の 4 つを許諾するライセンスです。

- ①プログラムの実行
- ②プログラムの動作を調べ、それを改変すること
- ③複製物の再頒布
- ④プログラムを改良し、改良を公衆にリリースする

GPL は二次的著作物についても、上記 4 点の権利が保護され、GPL ライセンスでなければなりません。

◆LGPL ライセンス

LGPL は GNU Lesser General Public License の略称で、GPL ライセンスよりも、④の「プログラムを改良し、改良を公衆にリリース」権利がゆるくなったライセンスになります。

開発したプログラムに組み込んで（静的リンク）使用した場合は、開発したプログラムは LGPL ライセンスとなり、④の権利が発生します。

LGPL ライセンスのライブラリを外に置いておいて、実行時に流用する（動的リンク）場合は、開発したプログラムは、LGPL ライセンスから外すことが可能です。

Qt モジュール毎に、適用されているライセンスが異なります。本開発環境では、Qt5.10.1 を採用しており、下記の条件が該当します。

Qt 5.7 以降は、Qt WebEngine の一部と Qt Serial Port を除いた LGPL v2.1 が適用されているすべての既存モジュールが LGPL v3 に変更されます。また、Qt 5.7 以降の Qt Charts、Qt Data Visualization、Qt Virtual Keyboard、Qt Purchasing、Qt Quick 2D Renderer の各モジュールは GPL v3 で提供されます。

※注：GPL v3 で提供されるモジュールについては、プロジェクトに組み込むとソースコードの公開義務が生じますのでご注意ください。

2-4-3 Qt Creator の起動

- ① アプリケーション一覧→「Qt Creator」をクリックします。図 2-4-3-1 のような画面が開きます。
- ② 「+新しいプロジェクト」をクリックします。

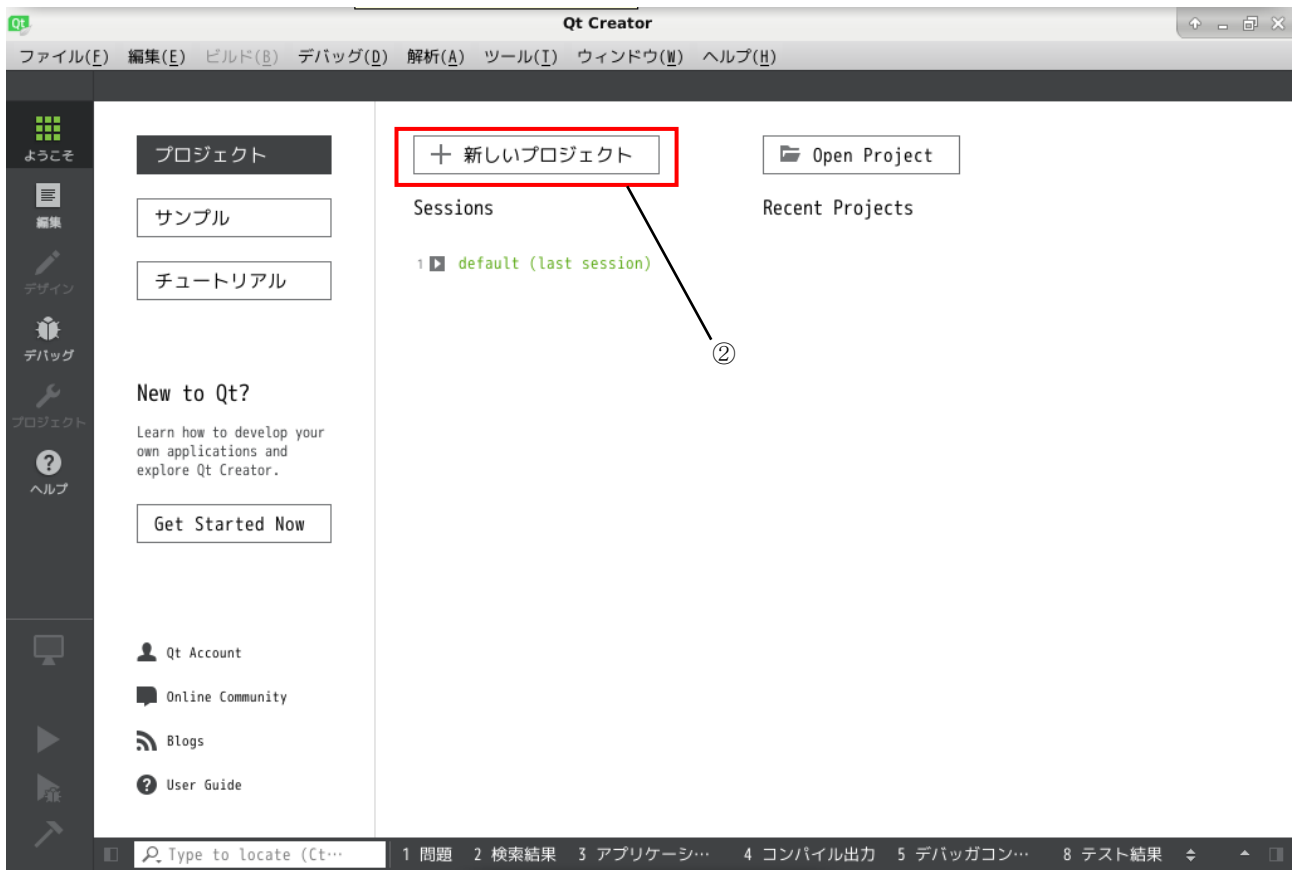


図 2-4-3-1. Qt Creator 起動画面

2-4-4 新規プロジェクト作成

- ① 今回は、フォーム画面付きのアプリケーションを作成しますので、「Qt ウィジェットアプリケーション」を選択します。
- ② 「選択…」ボタンをクリックします。

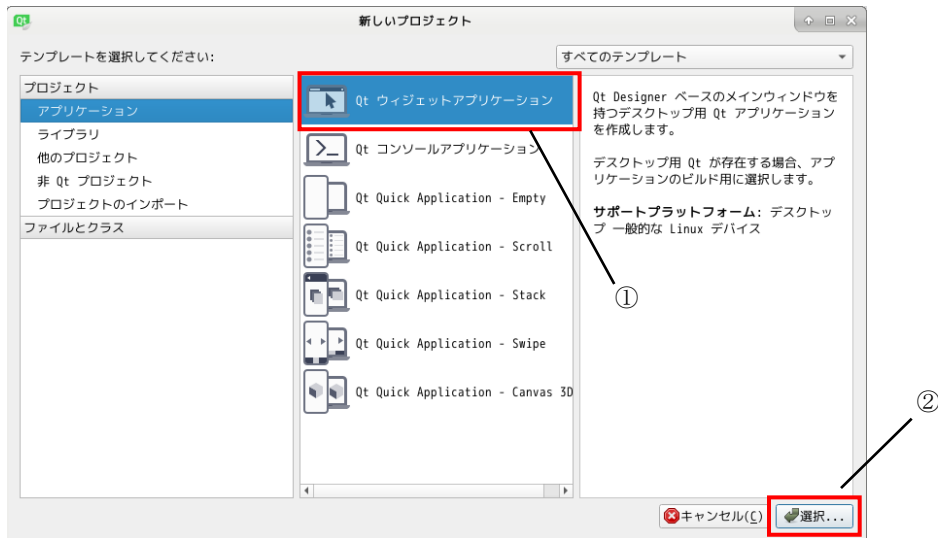


図 2-4-4-1. 新しいプロジェクト選択画面

- ③ プロジェクト名とパスを指定します。ここではプロジェクト名を「qt_test」、パスは「/home/asdusr」としています。「次へ」をクリックします。



図 2-4-4-2. プロジェクト名とパスの指定

④ 使用するキットの選択を行います。

Desktop Qt 5.10.1 GCC 64bit	インテル CPU 用ターゲット
Desktop Qt 5.10.1 GCC 64bit (aarch64)	ARM 64bit CPU 用ターゲット

コンパイルしたいキットにチェックを入れます。今回は両方のターゲット用のプログラムを作成するため両方にチェックを入れます。



図 2-4-4-3. キットの選択

⑤ クラス情報を設定します。特に変更がなければそのまま「次へ」をクリックします。



図 2-4-4-4. クラス情報設定

- ⑥ バージョン管理システムに追加するかを設定します。今回は、使用しないため、無しで「完了」をクリックします。



図 2-4-4-5. プロジェクト管理設定

- ⑦ プロジェクトが生成されると、図 2-4-4-6 のような、コード編集画面になります。UI ファイルをダブルクリックすると図 2-4-4-7 のようなフォーム画面編集画面になります。

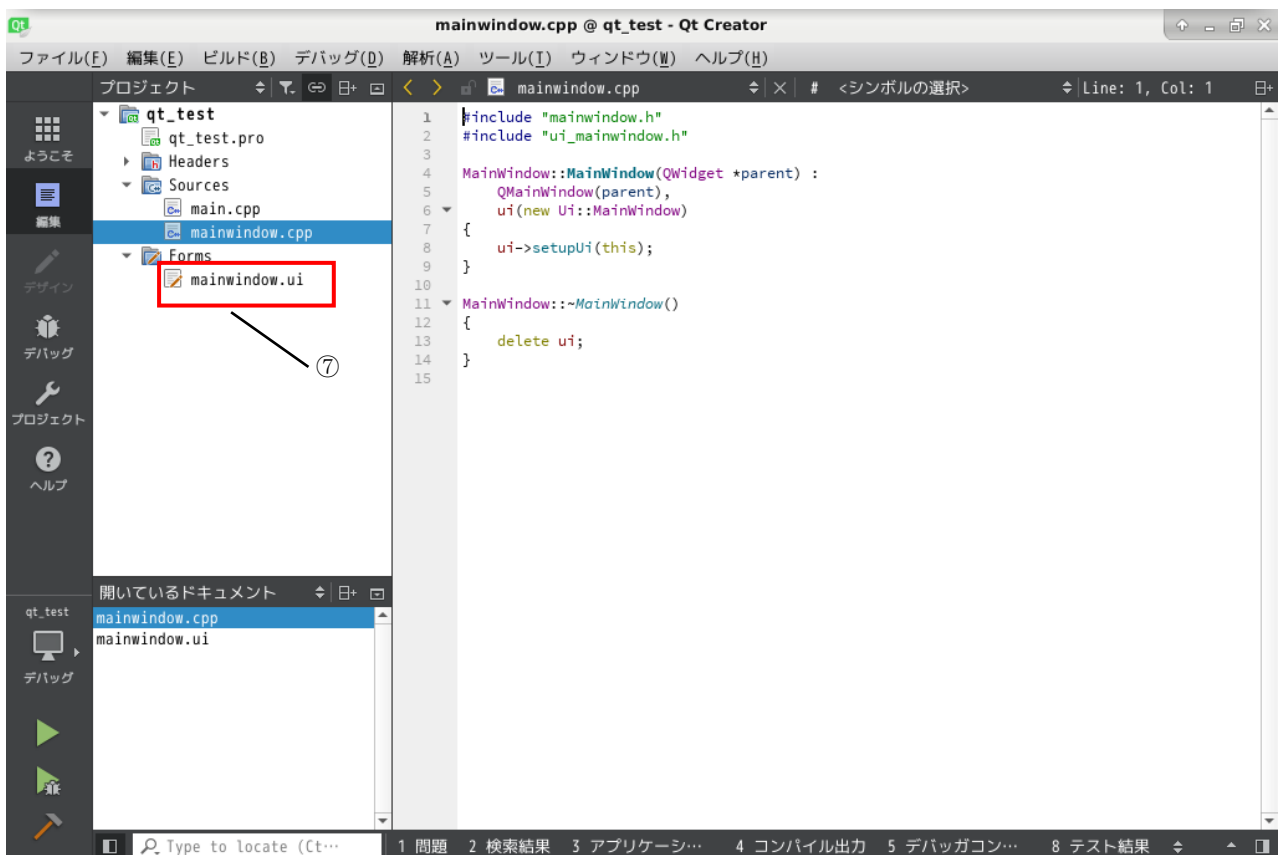


図 2-4-4-6. コード編集画面

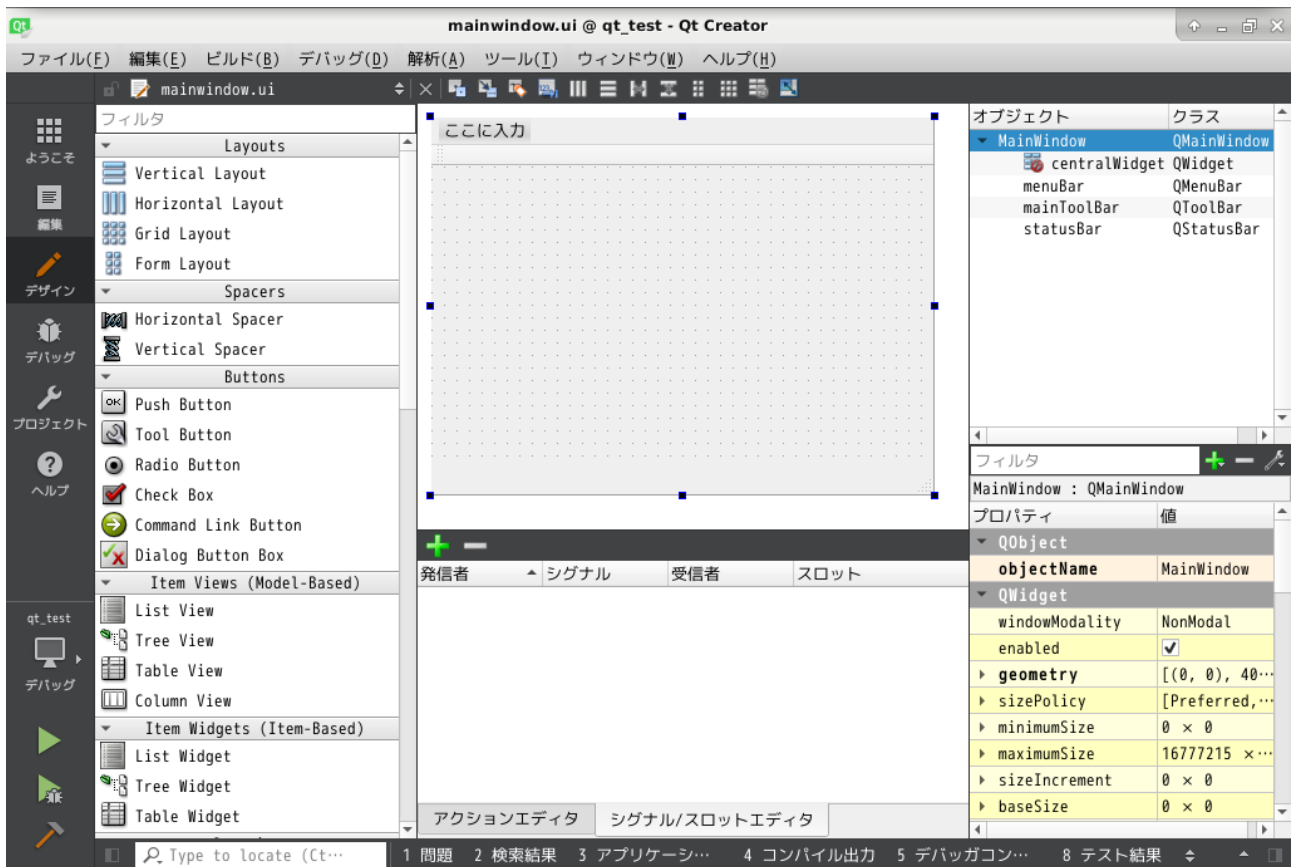


図 2-4-4-7. フォーム画面編集画面

2-4-5 部品の配置

① 図 2-4-5-1 のツールから、「Push Button」をドラッグして、ウィンドウにドロップして配置します。

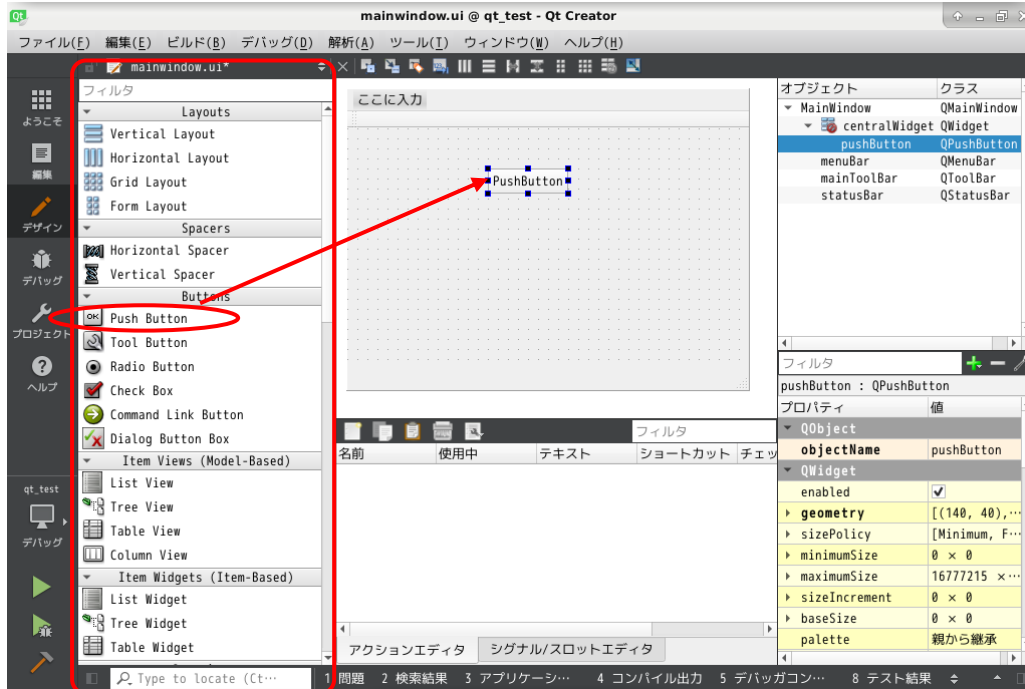


図 2-4-5-1. 部品の配置

2-4-6 ボタンイベント追加

① 前項で配置したボタンをクリックしたときのイベント処理を追加します。追加したボタンを右クリックして、「スロットへ移動…」をクリックします。

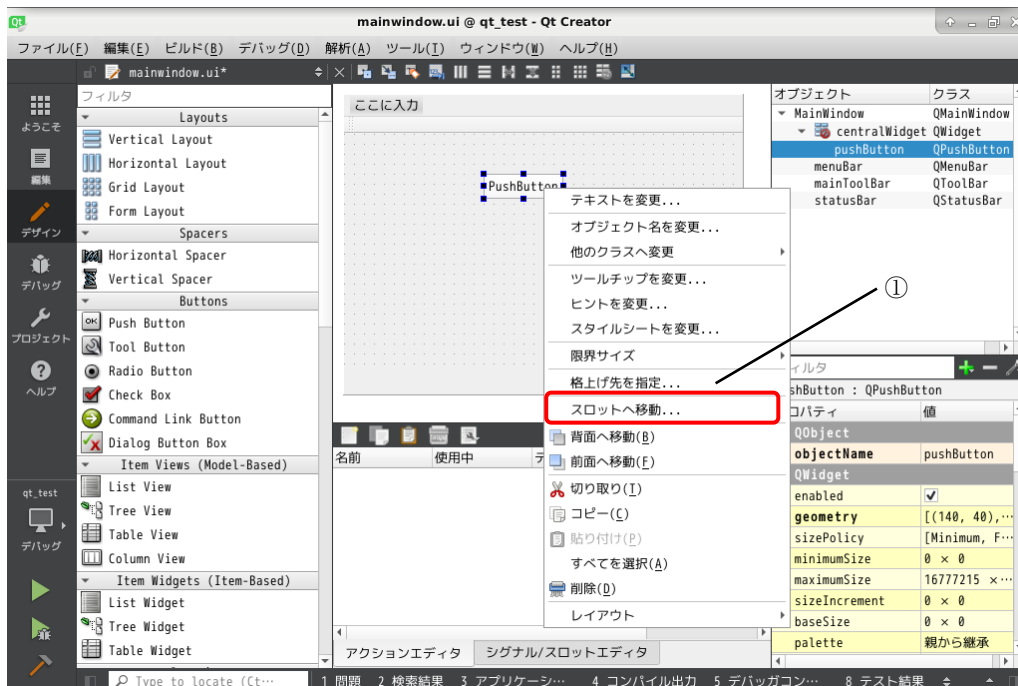


図 2-4-6-1. 部品右クリックメニュー

- ② スロットへ移動するための選択画面が表示されます。ここでは、Clicked() を選択して、「OK」ボタンをクリックします。

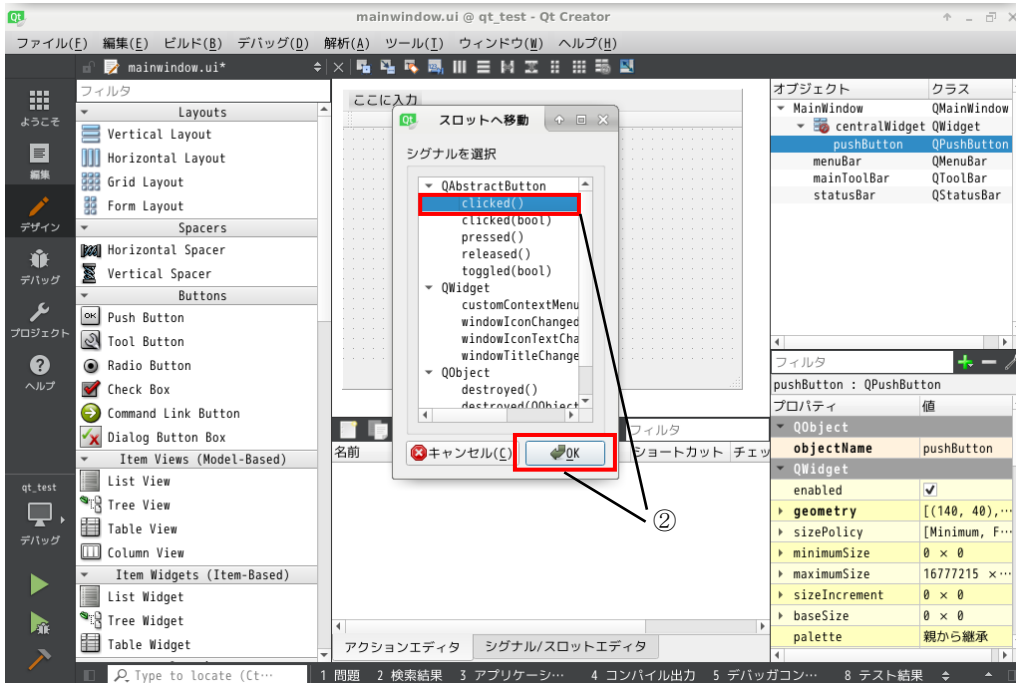
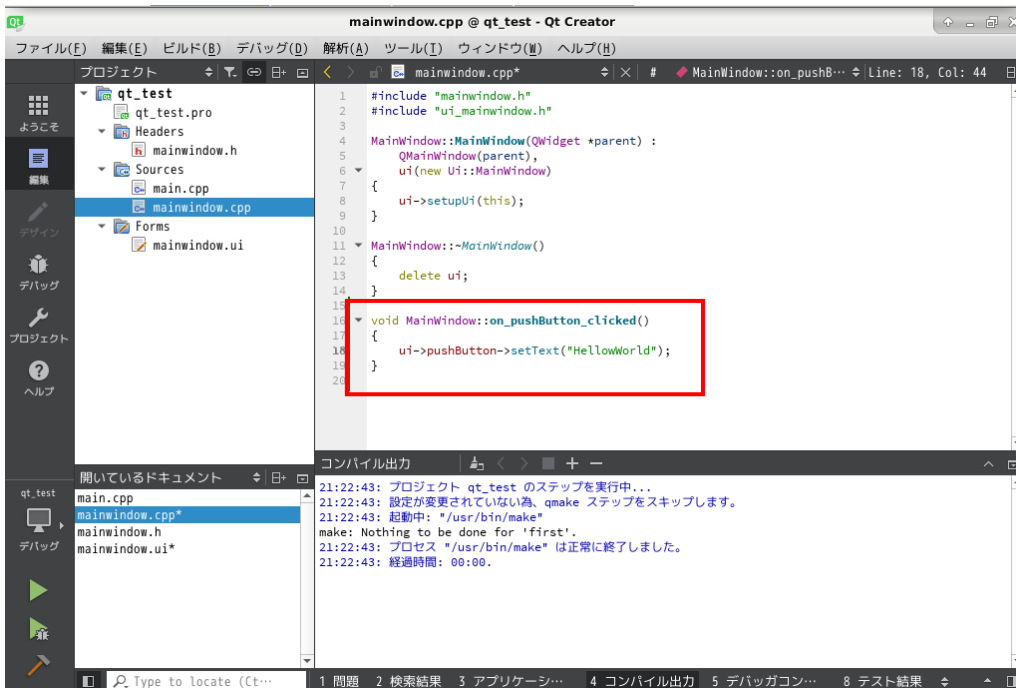


図 2-4-6-2. スロットの選択

- ③ mainwindow.cpp に図 2-4-6-3 のような、on_pushButton_clicked という関数が生成されます。関数の自身を図のように記述します。



```

15
16 void MainWindow::on_pushButton_clicked()
17 {
18     ui->pushButton->setText("HelloWorld");
19 }
20

```

図 2-4-6-3. Click イベント処理追加

2-4-7 インテル CPU 環境でビルド

- ① 「ビルド」 → 「すべてビルド」 を実行します。
- ② その後、「実行」をクリックすると、図 2-4-7-2 のアプリケーションが起動されます。ボタンをクリックすると、ボタンのテキストが変わります。

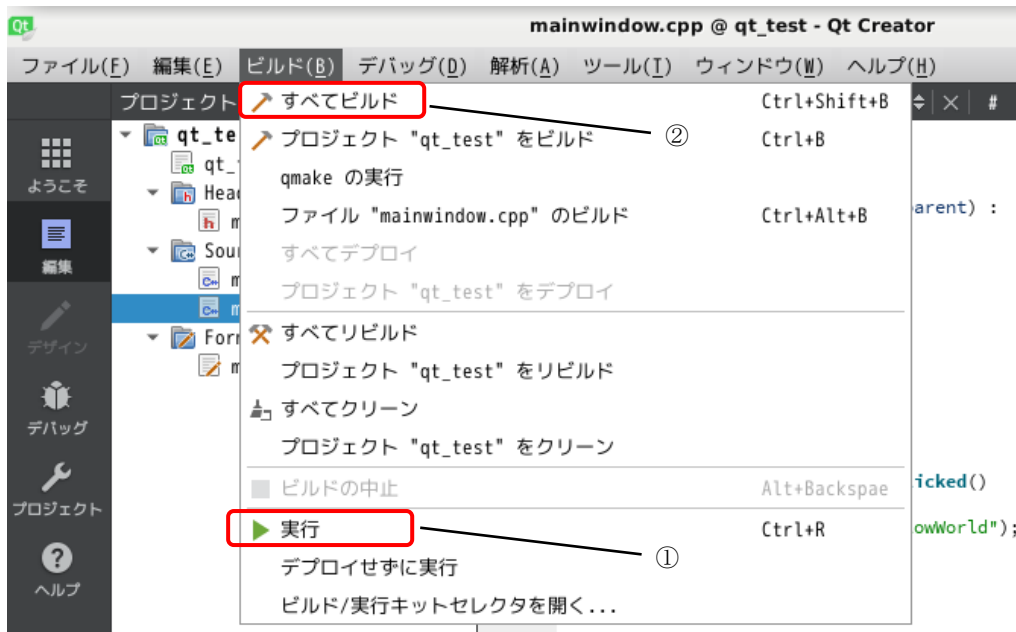


図 2-4-7-1. ビルドメニュー

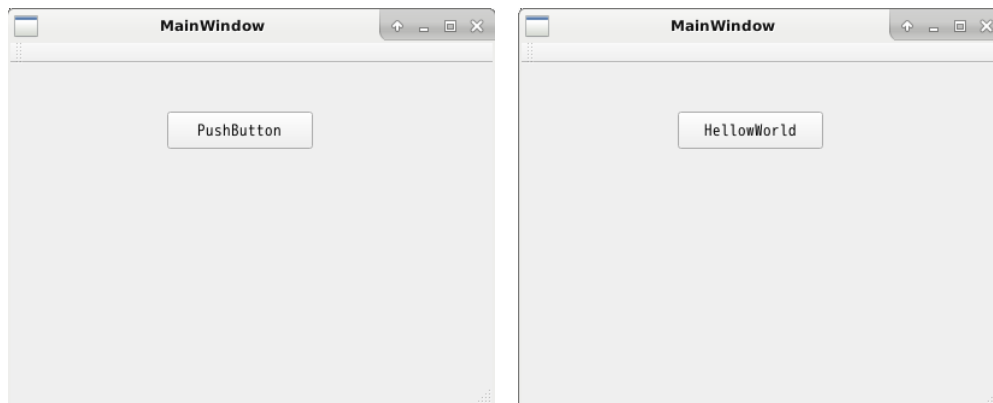


図 2-4-7-2. 実行画面

2-4-8 ARM64 環境でビルド

- ① ARM64 環境でコンパイルするために、キットを aarch64 用に変更する必要があります。まず、図 2-4-8-1 の .pro ファイルを開きます。

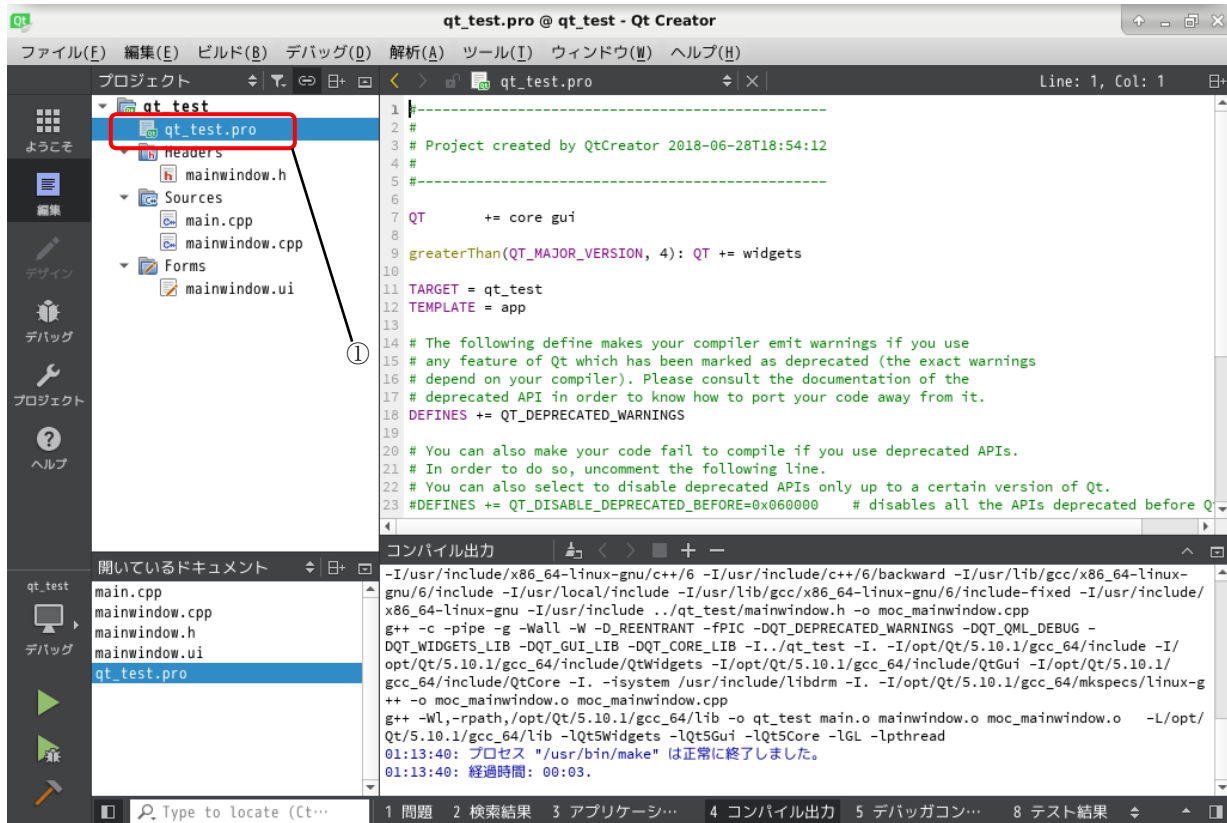


図 2-4-8-1. プロジェクトファイル初期設定

② 図 2-4-8-2 のように、リスト 2-4-8-1 の内容を追加します。

```

1 #-----
2 #
3 # Project created by QtCreator 2018-06-28T18:54:12
4 #
5 #-----
6
7 QT      += core gui
8
9 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
10
11 TARGET = qt_test
12 TEMPLATE = app
13
14 equals(QT_VERSION, 5.10.1) {
15     QMAKE_LFLAGS += -Wl,-rpath,/usr/local/tools-arm64/toolchain/arm64_rootfs/usr/lib/aarch64-linux-gnu
16     QMAKE_LFLAGS += -Wl,-rpath,/usr/local/tools-arm64/toolchain/arm64_rootfs/lib/aarch64-linux-gnu
17 }
18 QMAKE_CXXFLAGS += -Wno-unused-parameter -Wno-unused-variable -Wno-switch -Wno-unused-but-set-variable
19
20 # The following define makes your compiler emit warnings if you use
21 # any feature of Qt which has been marked as deprecated (the exact warnings
22 # depend on your compiler). Please consult the documentation of the
23 # deprecated API in order to know how to port your code away from it.
24 DEFINES += QT_DEPRECATED_WARNINGS

```

図 2-4-8-2. プロジェクトファイルコンパイルオプション追加

リスト 2-4-8-1. 追加内容

```

equals(QT_VERSION, 5.10.1) {
    QMAKE_LFLAGS += -Wl,-rpath,/usr/local/tools-arm64/toolchain/arm64_rootfs/usr/lib/aarch64-linux-gnu
    QMAKE_LFLAGS += -Wl,-rpath,/usr/local/tools-arm64/toolchain/arm64_rootfs/lib/aarch64-linux-gnu
}
QMAKE_CXXFLAGS += -Wno-unused-parameter -Wno-unused-variable -Wno-switch -Wno-unused-but-set-variable

```

③ 「ビルド」 → 「ビルド/実行キットセレクタを開く...」をクリックします。

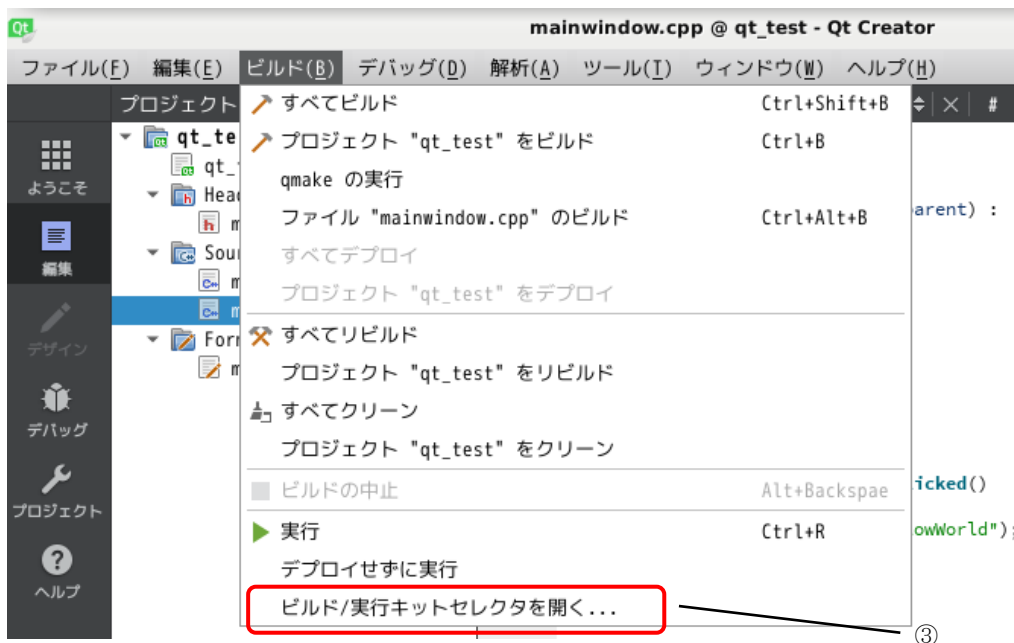


図 2-4-8-3. ビルドメニュー

⑥ コンパイルしたプログラムは、図 2-4-8-6 のようなディレクトリに格納されます。

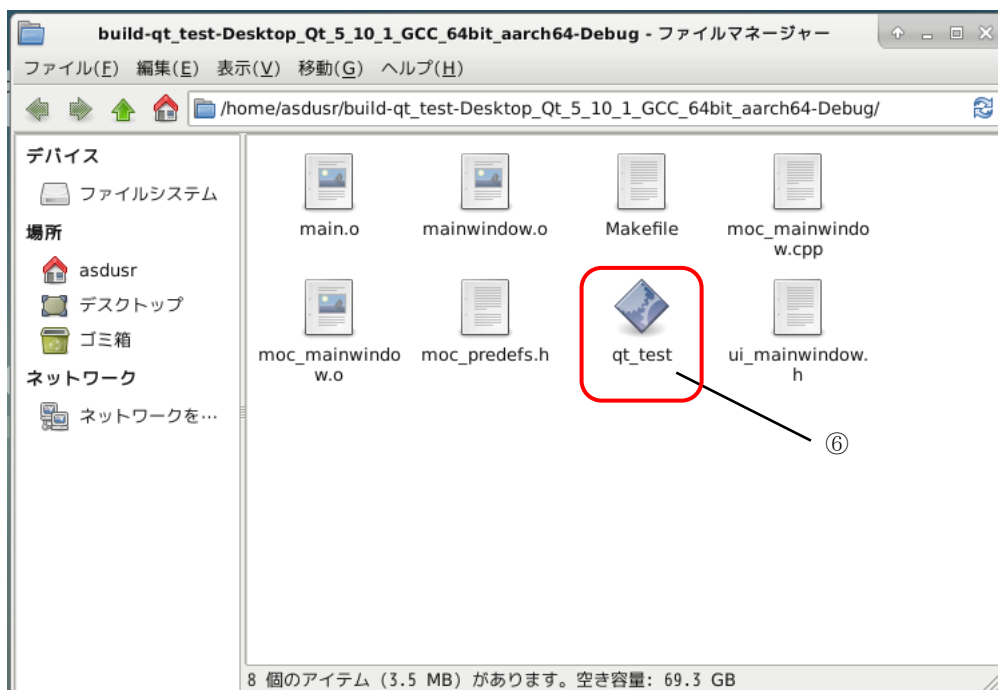


図 2-4-8-6. ARM 用実行ファイル格納ディレクトリ

⑦ このプログラムをターゲットにコピーして実行すると、下記のような画面が表示されます。

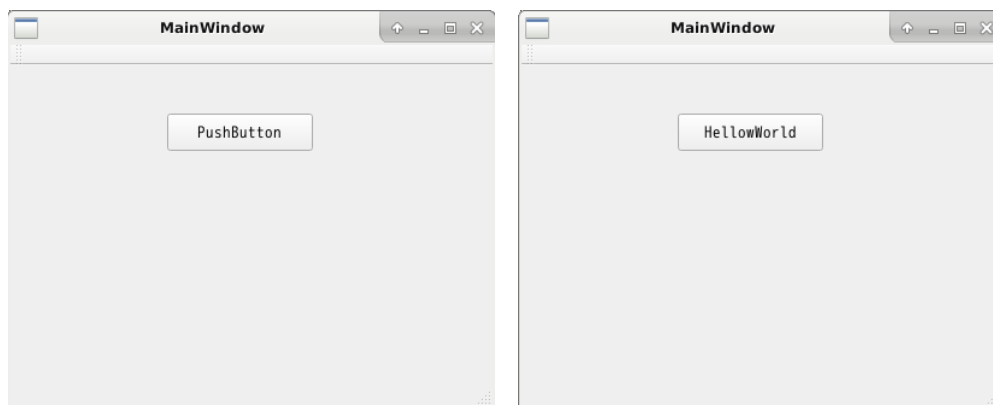


図 2-4-8-7. 実行画面

2-4-9 リモートデバッグ方法

ARM64 環境へ LAN 接続し、リモートデバッグを行う方法を示します。

■ ターゲットデバイス設定

- ① 「ツール」 → 「オプション」 をクリックします。

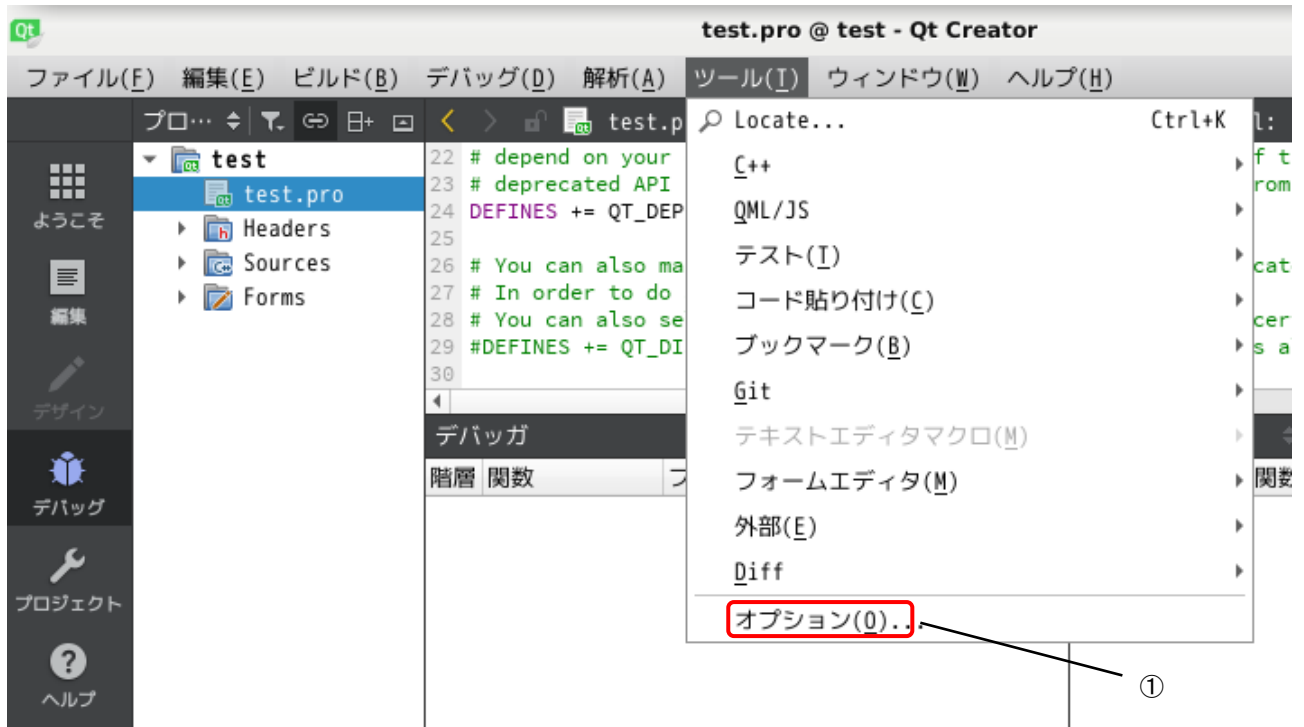


図 2-4-9-1. ツールメニュー

- ② 「デバイス」を開き、ホスト名をターゲットの IP アドレスに設定します。ユーザ名とパスワードは変更していないのであれば、「asusr」です。他の設定はそのままにしてください。
③ 「適用」または「OK」をクリックします。



図 2-4-9-2. オプション「デバイス」設定

■プロジェクトファイル デバッグ設定

- ① デバッグ実行したとき、下記のようなメッセージが表示された場合、プロジェクトファイルに設定を追加する必要があります。

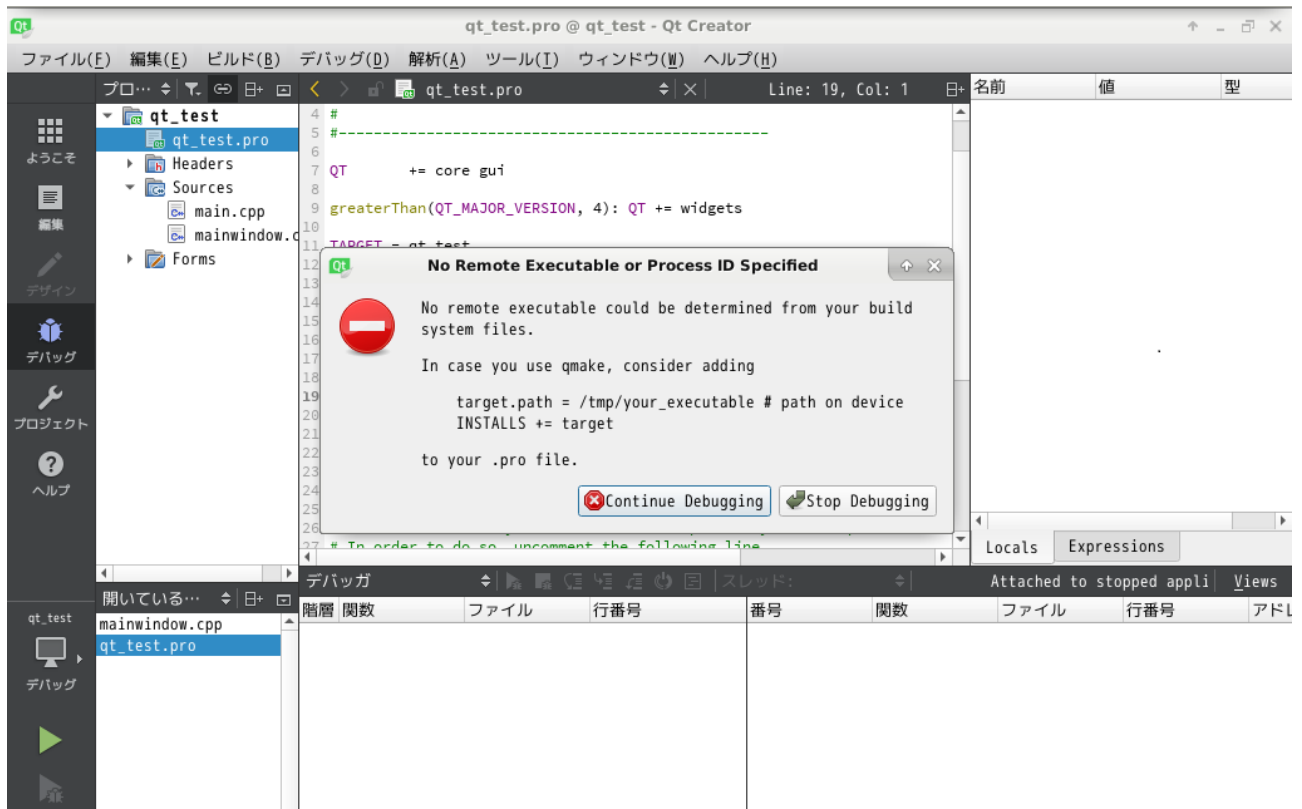


図 2-4-9-3. デバッグ実行時に出力されるメッセージ

② 図 2-4-9-4 の .pro ファイルを開きます。

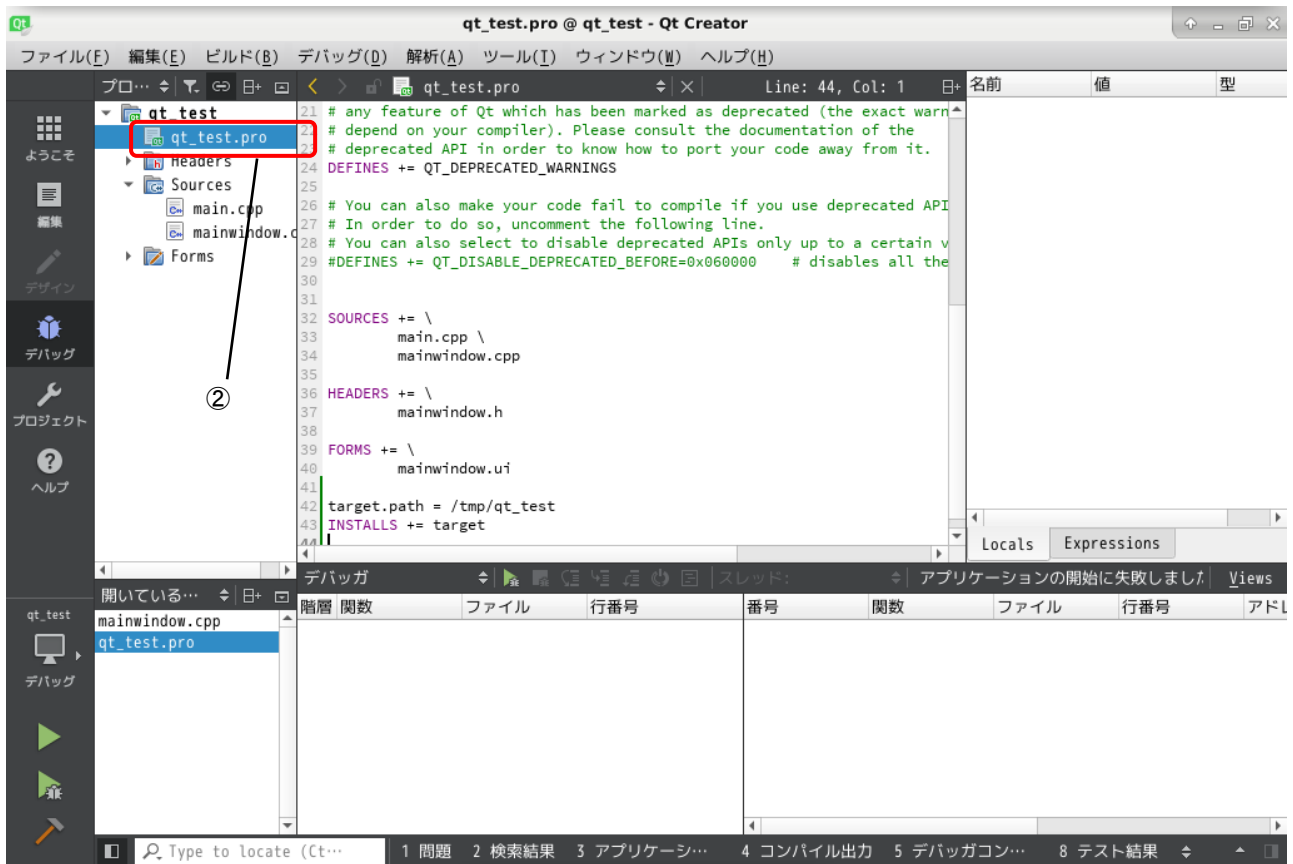


図 2-4-9-4. プロジェクトファイル

③ 図 2-4-9-5 のように、リスト 2-4-9-1 の内容を追加します。

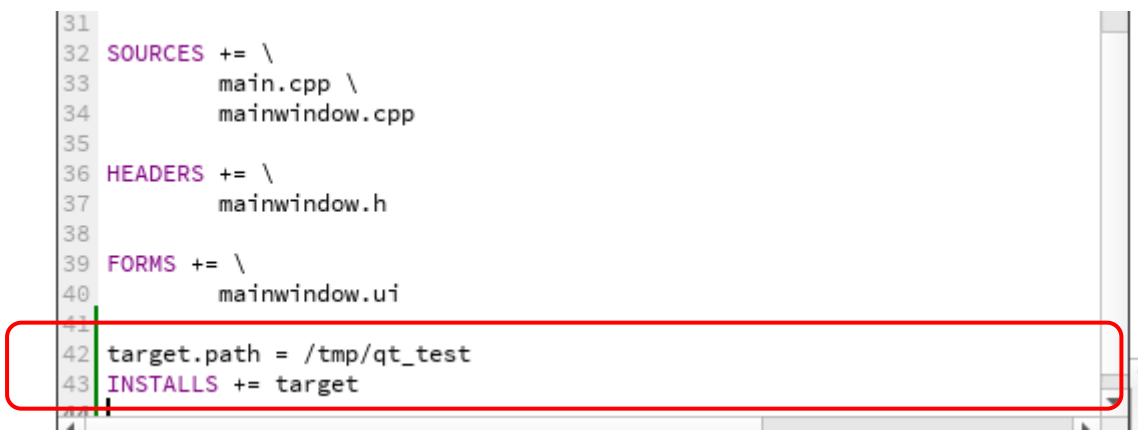


図 2-4-9-5. プロジェクトファイルコンパイルオプション追加

リスト 2-4-9-1. 追加内容

```
target.path = /tmp/<実行ディレクトリ>
INSTALLS += target
```

■ 画面付きアプリケーションデバッグ設定

- ① デバッグ実行したとき、下記のようなメッセージが表示された場合、DISPLAY 環境変数を追加する必要があります。

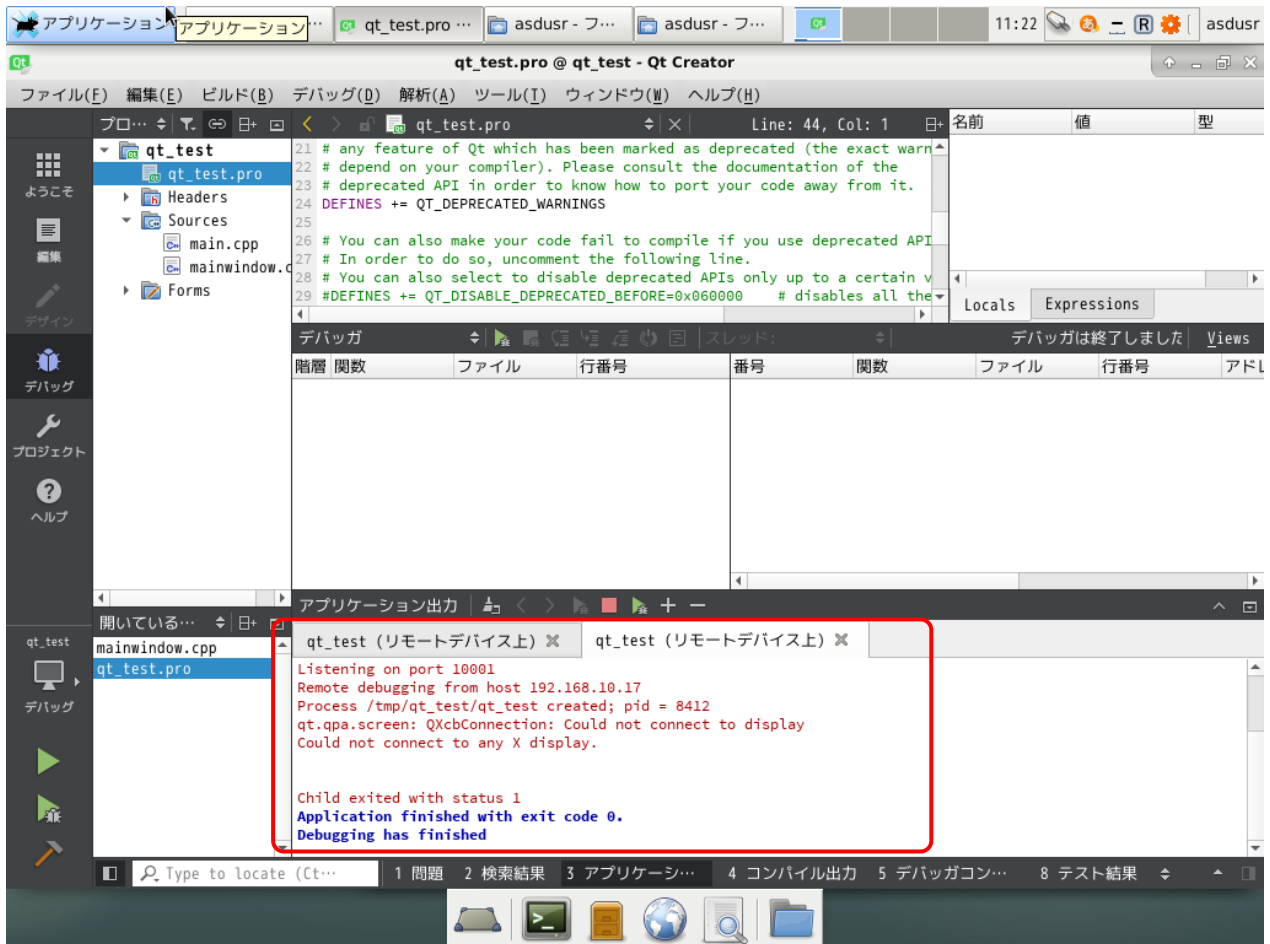


図 2-4-9-6. デバッグ実行時に出力されるメッセージ

- ② 「プロジェクト」 → 「RUN」 設定を表示します。「実行時の環境変数」の項目に DISPLAY 設定を追加します。



図 2-4-9-7. 実行時の環境変数設定

- ③ 「実行時の環境変数」の「詳細 ▼」をクリックします。
- ④ 「追加」をクリックします。

実行時の環境変数



図 2-4-9-8. 実行時の環境変数設定 (項目の追加)

- ⑤ 変数に「DISPLAY」、値に「:0」を設定します。

実行時の環境変数



図 2-4-9-9. 実行時の環境変数設定 (DISPLAY 環境変数設定)

■ リモートデバッグ開始

① デバッグ開始ボタンをクリックします。

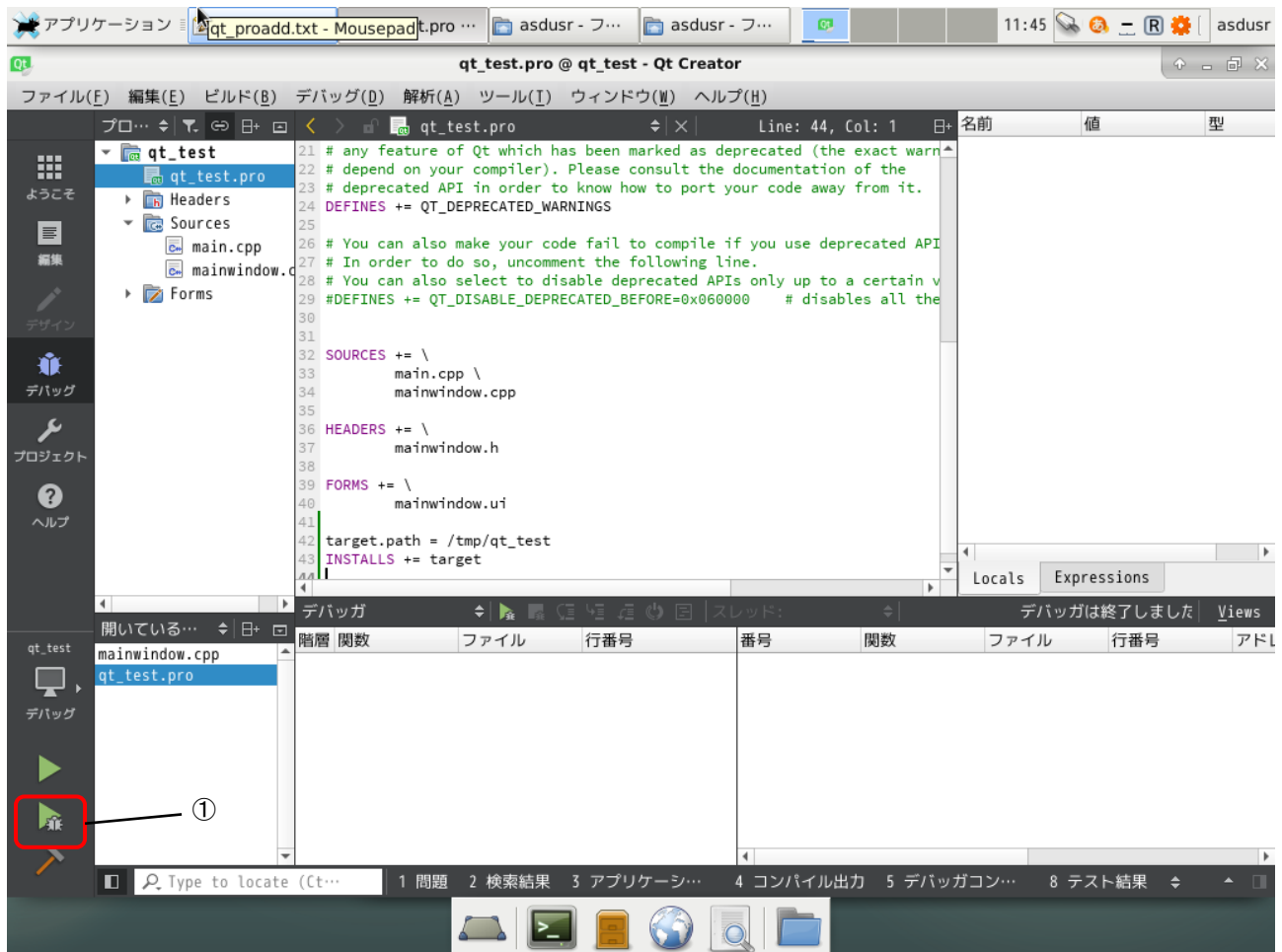


図 2-4-9-10. リモートデバッグ開始

② 正常に接続されると、下図のようなメッセージが表示され、実行環境側に画面が表示されます。

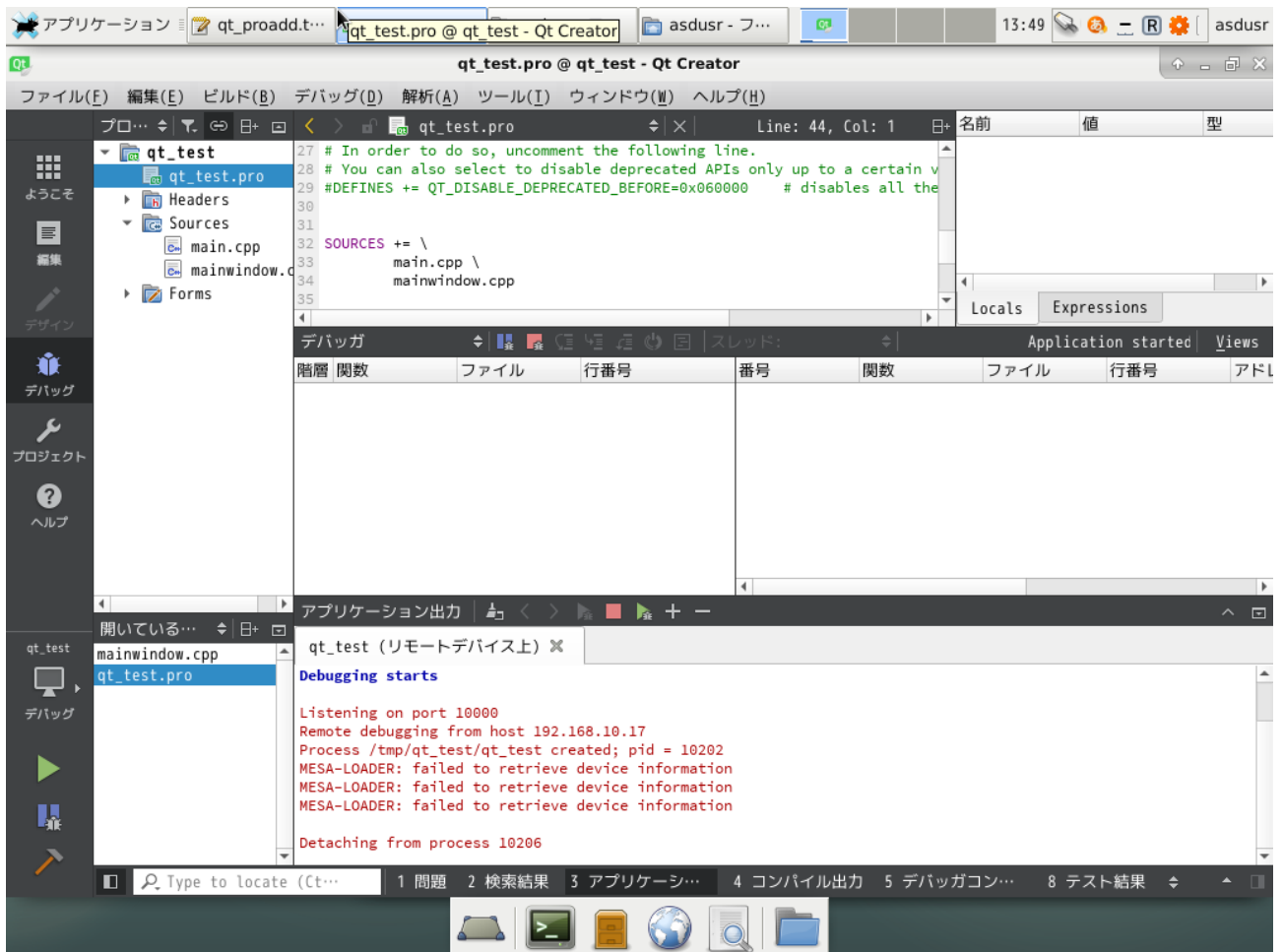


図 2-4-9-11. 正常に接続された場合

2-5 GDB によるデバッグ方法の詳細

実行環境上で実行されるプログラムのデバッグには GDB や GDB サーバを使用します。実行環境上では GDB サーバを実行させ、開発環境側では GDB を実行します。開発環境側で GDB 用のコマンドを実行することによりブレークやステップ実行などを行うことができます。

本項では簡単な使い方について説明します。詳細な使用方法については、GDB のマニュアルや解説書などを参照ください。

実行環境では、ネットワークポートを使用したデバッグ方法を標準としています。

以下より『2-3 WideStudio/MWT によるアプリケーション開発』でサンプルプログラムとして作成したプログラムを使用して、GDB にてデバッグする方法を示します。


- ① デバッグモードでのコンパイル
デバッグ過程がよくわかるように、Hello! と表示するボタンのプロシージャ関数のコードをリスト 2-5-1 のように変更します。

リスト 2-5-1. 表示文字列を変更するコード (デバッグ確認用)

```
#include <WScom.h>
#include <WSCfunctionList.h>
#include <WSCbase.h>
//-----
//Function for the event procedure
//-----
void Btn_Click(WSCbase* object) {
    int i, j, k;

    k=0;
    for (i=0; i<10; i++) {
        for (j=0; j<10; j++) {
            k++;
        }
    }
    object->setProperty (WSNlabelString, k);           //表示文字列変更 (k の値を表示)
}
static WSCfunctionRegister op("Btn_Click", (void*)Btn_Click);
```

GDB を使用する場合、コンパイルオプションとして「-g」や「-ggdb」を付加してコンパイルする必要があります。

WideStudio の場合は、をクリックする、またはメニューの「プロジェクト(P)」→「プロジェクト設定(E)」をクリックします。「コンパイル」タブをクリックすることで、図 2-5-1 のようなプロジェクト設定画面が表示されます。

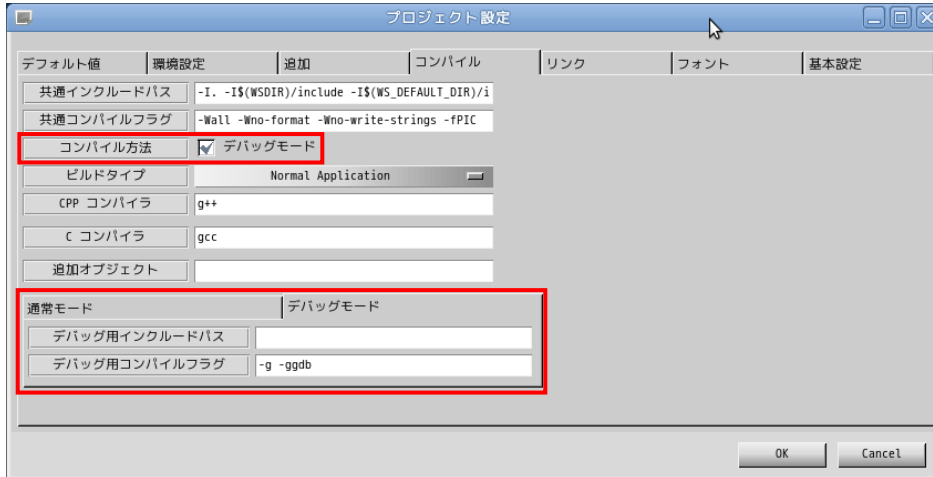


図 2-5-1. プロジェクト設定画面（コンパイルタブ）

「コンパイル方法」項目の、「デバッグモード」のチェックボックスにチェックを入れた場合は、デバッグモードのコンパイルフラグが使用されるようになります。デバッグモードのタブに「-g -ggdb」デバッグ用コンパイルフラグが設定されていますので確認してください。

「OK」ボタンをクリックし、プロジェクト全体をコンパイルします。通常モードで作成される実行プログラム名に「d」が付いた実行プログラムが作成されます。（「newproject」→「newprojectd」）

② GDB サーバの起動

実行環境で「gdbserver」を起動します。『2-3-8 ファイルの転送』の「LAN 経由で ftp 転送」の項目を参考に、①でコンパイルした「newprojectd」をアルゴシステム製端末に転送します。以下のコマンドを実行して「gdbserver」を起動します。

```
# chmod 755 newprojectd
# gdbserver host1:2345 ./newprojectd
Process ./newprojectd created; pid = 21507
Listening on port 2345
```

③ GDB の起動

開発環境のデスクトップ上の「アプリケーション」ツールバーから「アクセサリ」→「端末」をクリックし、別のコンソールを起動します。デバッグするプログラムのソースディレクトリへ移動します。

```
# cd /home/asdusr/sample
```

GDB を起動し、実行環境の GDB サーバと接続します。

```
# gdb ./newprojectd
GNU gdb (GDB) 7.0.1-debian
Copyright (C) 2009 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "i484-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/asdusr/sample/newprojectd... done.
(gdb) target remote 192.168.0.1:2345
Remote debugging using 192.168.0.1:2345
[New Thread 3009]
0x295568c0 in ?? () from /lib/ld-linux.so.2
(gdb)
```

ここまでで、GDB の起動は完了しました。

④ デバッグ

ブレークポイントを指定します。ここではボタンのプロシージャ関数でブレークするようにします。

```
(gdb) b Btn_Click(WSCbase*)
Breakpoint 1 at 0x401904: file Btn_Click.cpp, line 10.
(gdb)
```

コンティニュー実行します。実行環境上にサンプルプログラムの画面が表示されます。

```
(gdb) c
Continuing.
```

「PUSH」 ボタンをクリックすることで、以下のメッセージがでてブレークされます。

```
Breakpoint 1, Btn_Click (object=0x452d08) at Btn_Click.cpp:10
10          k=0;
Current language: auto
The current source language is "auto; currently c++".
(gdb)
```

監視する変数を指定します。

```
(gdb) display k
1: k = 152474568
(gdb) display i
2: i = -1076372104
(gdb) display j
3: j = -1207467296
(gdb)
```

ステップ実行します。

```
(gdb) n
11          for (i=0; i<10; i++) {
3: j = -1207467296
2: i = -1076372104
1: k = 0
(gdb) n
12          for (j=0; j<10; j++) {
3: j = -1207467296
2: i = 0
1: k = 0
```

```
(gdb) n
13                               k++;
3: j = 0
2: i = 0
1: k = 0
(gdb) n
12                               for (j=0; j<10; j++) {
3: j = 0
2: i = 0
1: k = 1
(gdb)
```

リストを表示します。入力された数値を中心に 10 行表示されます。

```
(gdb) list 13
8      int i, j, k;
9
10     k=0;
11     for (i=0; i<10; i++) {
12         for (j=0; j<10; j++) {
13             k++;
14         }
15     }
16     object->setProperty (WSNLabelString, k);
17 }
(gdb)
```

16 行目にブレークを張り、コンティニュー実行します。

```
(gdb) b 16
Breakpoint 2 at 0x40193a: file Btn_Click.cpp, line 16.
(gdb) c
Continuing.

Breakpoint 2, Btn_Click (object=0x452d08) at Btn_Click.cpp:16
16     object->setProperty (WSNLabelString, k);
3: j = 10
2: i = 10
1: k = 100
(gdb)
```

この時点で、ボタンの表示が変更される直前まで実行しました。さらにコンティニュー実行を行うとボタンに 100 と表示されます。サンプルプログラムを終了し、GDB を終了します。

```
(gdb) c
Continuing.

Program exited normally.
(gdb) q
#
```

以上で GDB によるデバッグ方法について簡単に説明しました。ここで紹介したコマンドの他にもいろいろなコマンドが用意されています。比較的好く使うと思われるコマンドについて表 2-5-1 に示します。

表 2-5-1. GDB コマンド (抜粋)

コマンド (省略)	書式	説明	
実行	continue (c)	c	停止中のプログラムを再開します。
	next (n)	n	実行する行が関数の場合、関数の中へ入らずに次の行まで実行します。
	step (s)	s	実行する行が関数の場合、関数の中に入って実行します。 ※注: 実行する関数が WideStudio の関数の場合はライブラリがデバッグ対応でないためステップ実行することができません。
中断	break (b)	b <関数名> b <行番号> b <ファイル名>:<行番号>	ブレークポイントを設定します。
変数	display (disp)	disp <変数名>	監視する変数を設定します。 プログラムが停止するたびに値が表示されます。
	print (p)	p <変数名>	変数の値をモニタします。
	set	set <変数名>=<値>	変数の値を変更します。
その他	list (l)	l <行番号> l <関数名>	指定した箇所のソースを 10 行表示します。
	delete (d)	d <ブレークポイント> d <監視中の変数>	指定した番号のブレークポイントや監視中の変数を削除できます。
	info (i)	i breakpoints i local i func	デバッグ中の情報を表示します。他のサブコマンドについては「help info」を参照してください。 breakpoints : 現在、張っているブレークポイントの表示 local : ローカル変数の表示 func : 関数の表示
	quit (q)	q	デバッグを終了します。

2-6 VirtualBox を使用しない開発環境構築方法

本項では、パソコンに直接 Ubuntu 20.04 をインストールし、EC1G-01x/02x 用開発環境を構築する方法について説明します。

2-6-1 Ubuntu 20.04 LTS のインストール

Ubuntu 20.04 LTS をインストールするパソコンの推奨スペックを表 2-6-1-1 に示します。

表 2-6-1-1. Ubuntu 20.04 LTS インストール推奨パソコンスペック

CPU	2GHz デュアルコアプロセッサ以上
メモリ	4GByte 以上
HDD 空き領域	128GByte 以上

EC1G-01x/02x 用開発環境では「Ubuntu 20.04 LTS 日本語 Remix」を使用しています。バージョンが違っていると不整合が起こる可能性がありますので、弊社で EC1G-01x/02x 用開発環境を構築するときに使用した、「Ubuntu 20.04 LTS 日本語 Remix」のインストールディスクを配布します。

下記に示す Ubuntu 20.04 LTS のインストール方法はパソコンによって若干変化する可能性があります。基本的には画面の指示通りにインストールしてください。

- ① 「Ubuntu 20.04 LTS 日本語 Remix」の DVD-ROM をパソコンに入れ、DVD-ROM から起動します。DVD-ROM から起動するには、インストールするパソコンの BIOS 設定を行う必要があります。
- ② 正常に DVD-ROM から起動されると、図 2-6-1-1 のようなインストールメニュー画面が表示されます。「Ubuntu をインストール」を選択します。



図 2-6-1-1. Ubuntu インストールメニュー

- ③ 図 2-6-1-2 のようなキーボードレイアウト選択画面が表示されます。使用するキーボードのレイアウトを選んで「Continue」ボタンをクリックします。

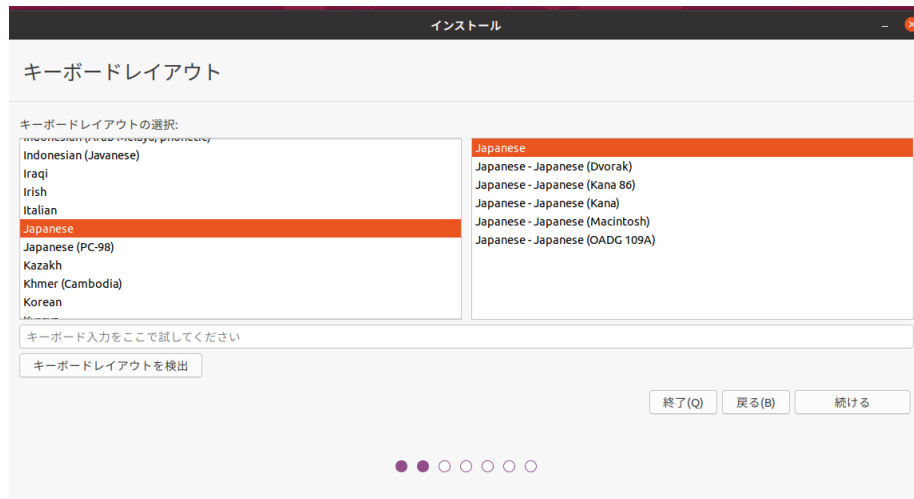


図 2-6-1-2. キーボードレイアウト選択画面

- ④ 図 2-6-1-3 のような場所のインストール設定画面が表示されます。[続ける]をクリックしてください。



図 2-6-1-3. 場所の選択画面

- ⑤ 図 2-6-1-4 のようなインストールの種類を設定する画面が表示されます。特に問題なければ、「ディスクを削除して Ubuntu をインストール」を選択し、「インストール」ボタンをクリックします。



図 2-6-1-4. インストールの種類

- ⑥ 図 2-6-1-5 のようなタイムゾーン選択画面が表示されます。タイムゾーンを選択して「続ける」をクリックしてください。

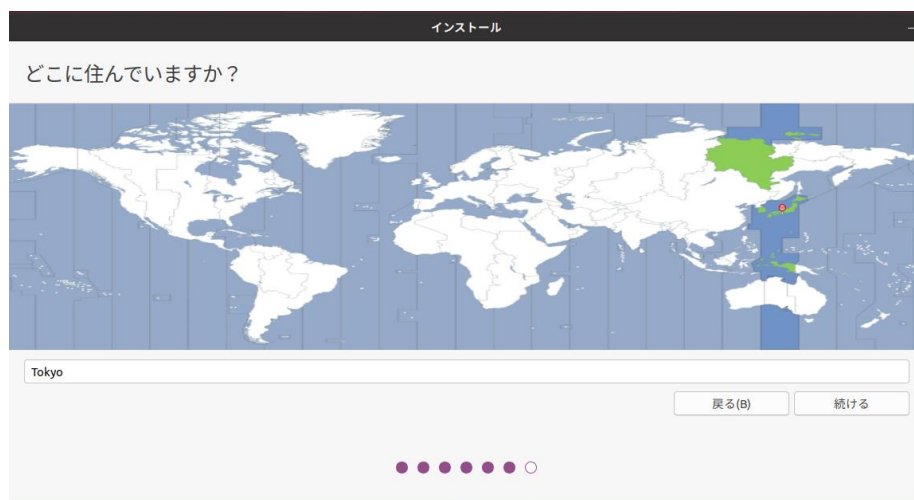


図 2-6-1-5. タイムゾーン設定画面

- ⑦ 図 2-6-1-6 のような、ユーザー情報入力画面が表示されます。EC1G-01x/02x 用開発環境はデフォルトで、以下のように設定しています。

あなたの名前 : asdusr
コンピューターの名前 : asdhost
ユーザー名 : asdusr
パスワード : asdusr

入力後、「続ける」をクリックすることで、インストールが始まります。



The screenshot shows a window titled 'インストール' (Install) with the subtitle 'あなたの情報を入力してください' (Please enter your information). The form contains the following fields and options:

- 'あなたの名前:' (Your name): Input field containing 'asdusr' with a green checkmark.
- 'コンピューターの名前:' (Computer name): Input field containing 'asdhost' with a green checkmark. Below it is the text '他のコンピューターと通信するときに使用する名前です。' (Name used when communicating with other computers).
- 'ユーザー名:' (Username): Input field containing 'asdusr' with a green checkmark.
- 'パスワード:' (Password): Input field with masked characters and a red '弱いパスワード' (Weak password) warning.
- 'パスワードの確認:' (Confirm password): Input field with masked characters and a green checkmark.
- Radio buttons for login options:
 - 自動的にログインする (Log in automatically)
 - ログイン時にパスワードを要求する (Require password at login)

At the bottom right, there are two buttons: '戻る(B)' (Back) and '続ける' (Next). At the bottom center, there are six purple dots, with the first one filled.

図 2-6-1-6. ユーザー情報設定画面

- ⑧ インストール完了後、PC が再起動します。再起動後、ユーザ名入力画面が表示されますので、インストール時に設定したユーザ名とパスワードを入力してください。

2-6-2 開発環境構築前準備

EC1G-01x/02x 用開発環境を構築する為の前準備方法を以下に示します。

- ① アプリケーション一覧から「端末」を選択し、コンソールを起動します。

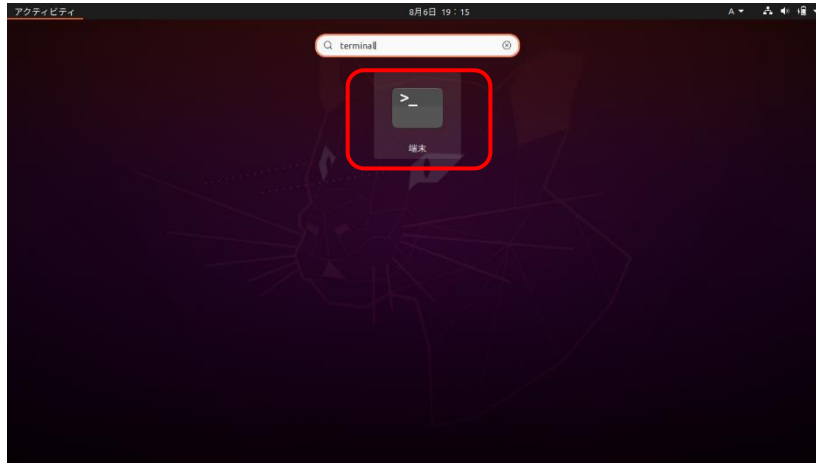


図 2-6-2-1. コンソールの起動

- ② パッケージのインストール前に、apt パッケージを更新するため、以下のコマンドを実行します。

```
$ sudo apt update
$ sudo apt upgrade
```

- ③ 開発環境に必要なパッケージをインストールします。インストールするパッケージを表 2-6-2-1 に示します。「sudo apt install XXX」で順次インストールしてください。

表 2-6-2-1. 開発環境構築追加パッケージ

パッケージ名	バージョン	内容
make	4.2.1	コンパイルを制御するユーティリティ
gcc	9.3.0-10ubuntu2	GNU C コンパイラ
g++	9.3.0-10ubuntu2	GNU C++ コンパイラ
libncurses5	6.2-0ubuntu2	shared library for terminal handling
libncurses5-dev	6.2-0ubuntu2	developer's libraries for ncurses
libpython2.7-dev	2.7.18~rc1-2	
git	2.25.1	速く、スケーラブルな分散型リビジョン管理システム
bc	1.07.1	GNU bc: 任意精度の計算言語
dc	1.07.1	GNU dc - 任意精度の逆ポーランド式計算器
fakeroot	1.24	tool for simulating superuser privileges
device-tree-compiler	1.5.0	Device Tree Compiler for Flat Device Trees
libjpeg-dev	8c-2ubuntu8	Development files for the JPEG library
libpng-dev	1.6.37-2	PNG ライブラリ - 開発用 (バージョン 1.6)
libxpm-dev	1:3.5.12-1	X11 pixmap ライブラリ (開発用ヘッダ)
libx11-dev	2:1.6.9-2ubuntu1	X11 クライアントサイドライブラリ (開発用ヘッダ)
libxt-dev	1:1.1.5-1	X11 toolkit intrinsics library (development headers)
libxext-dev	2:1.3.4-0ubuntu1	X11 用の雑多な拡張ライブラリ (開発用ヘッダ)
libxft-dev	2.3.3-0ubuntu1	X 用の FreeType ベースのフォント描画ライブラリ (開発用ファイル)
libperl-dev	5.30.0-9build1	Perl ライブラリ: 開発用ファイル
gucharmap	1:13.0.1-1	Unicode 文字ピッカおよびフォントブラウザ
vim	8.1.2269	enhanced vi editor

gFTP	2.0.19	X/GTK+ and console FTP client
------	--------	-------------------------------

```
# apt-get install make gcc g++ libncurses5-dev git bc dc device-tree-compiler
libjpeg-dev libpng-dev libxpm-dev libx11-dev libxt-dev libxext-dev libxft-dev
libperl-dev gucharmap
#
```

2-6-3 EC1G-01x/02x 用開発環境インストール

Ubuntu 20.04 LTS に EC1G-01x/02x 用開発環境をインストールします。EC1G-01x/02x 用開発環境 SD カードに格納されている、「Ubuntu20.04-tools-arm64.tar.gz」を展開することでインストールされます。

インストールされるパッケージの内容については『2-2-9 EC1G-01x/02x 用開発環境のディレクトリ構成について』を参照してください。

- ① EC1G-01x/02x 用開発環境 SD カードをパソコンにセットします。しばらくすると、デスクトップ上にメモリドライブのアイコンが表示され、内容が表示されます。
- ② コンソールを起動し、下記のコマンドを実行します。

```
$ sudo tar zxvf /media/asdusr/デバイス名/development/Ubuntu20.04-tools-arm64.tar.gz -C /
パスワード: asdusr ←インストール時に設定したユーザーパスワード
$ sudo tar zxvf /media/asdusr/デバイス名/development/Ubuntu20.04-tools-imx8mq.tar.gz -C /
```

これで、EC1G-01x/02x 用開発環境一式が展開されます。パソコンスペックにもよりますが、展開終了までに数十分程度かかります。

- ③ 正常に展開が終了されると「/usr/local/tools-arm64」、「/usr/local/tools-imx8mq」というディレクトリが作成されます。

2-6-4 WideStudio インストール

ここでは、Ubuntu 20.04 LTS に WideStudio 開発環境をインストールする手順を記します。WideStudio をインストールする前に、EC1G-01x/02x 用開発環境をインストールする必要があります。EC1G-01x/02x 用開発環境のインストール手順は、『2-6-3 EC1G-01x/02x 用開発環境インストール』を参照してください。

- ① WideStudio をインストールします。下記のようにコマンドを実行してください。

```
$ cd /usr/local/tools-arm64/source/widestudio/ws-v3.98-7-custom/src
$ sudo make install
$ sudo ldconfig
```

- ② WideStudio 実行用スクリプトをコピーします。

```
$ sudo cp /usr/local/tools-arm64/source/widestudio/wsstart.sh /usr/local/bin
```

- ③ アプリケーション一覧に WideStudio 実行用スクリプトのショートカットを登録します。
/home/asdusr/.local/share/applications、もしくは/usr/share/applications ディレクトリ以下に、widestudio.desktop という名称で、リスト 2-6-4-1 のようなファイルを作製してください。

リスト 2-6-4-1. widestudio.desktop

```
[Desktop Entry]
Version=3.98-7
Name=WideStudio
GenericName=WideStudio
Comment=Multiplatform GUI SDK
Type=Application
Exec=/usr/bin/wsstart.sh
Icon=/usr/share/pixmaps/widestudio.ico
Keywords=Development;SDK;
Categories=Development;SDK;
```

- ④ アイコンファイルをコピーします。以下のコマンドを実行してください。

```
$ sudo cp /usr/local/tools-arm/widestudio/ws-v3.98-7/src/wsagent/main.ico
/usr/share/pixmaps/widestudio.ico
```

- ⑤ 「Alt」 + 「F2」 で「コマンドを実行」ダイアログが表示されます。「restart」と入力し、Enter キーを押してください。

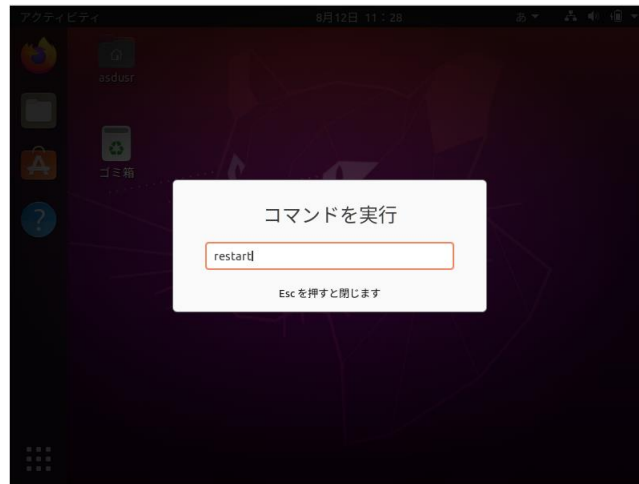


図 2-6-4-1. コマンドを実行

- ⑥ アプリケーション一覧に WideStudio ランチャーが追加されます。クリックして WideStudio が起動されることを確認してください。

2-6-5 Qt 開発環境インストール

ここでは、Ubuntu20.04 LTS に Qt 開発環境をインストールする手順を記します。インストールする Qt フレームワークのバージョンは 5.10 となります。Qt 開発環境をインストールする前に、EC1G-01x/02x 用開発環境をインストールする必要があります。EC1G-01x/02x 用開発環境のインストール手順は、『2-6-3 EC1G-01x/02x 用開発環境インストール』を参照してください。

- ① 開発環境用 Qt フレームワークをインストールします。以下のコマンドを実行することで、開発環境上に Qt フレームワークがインストールされます。

```
$ sudo tar zxvf qt-5-10.1-dev.tar.gz -C /
```

- ② Qt 用 IDE (Qt developer) をインストールします。以下のコマンドを実行してください。図 2-6-5-1 のような画面が表示されます。

```
$ ./qt-creator-opensource-linux-x86_64-4.12.4.run
```

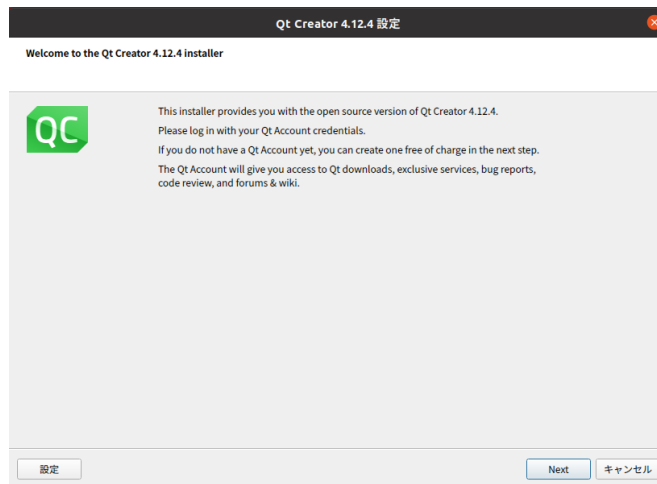


図 2-6-5-1. Qt Creator インストーラ

- ③ 「Next」をクリックすることで、ライセンス登録画面が表示されます。ネットワークに接続していない場合、この画面はスキップされます。

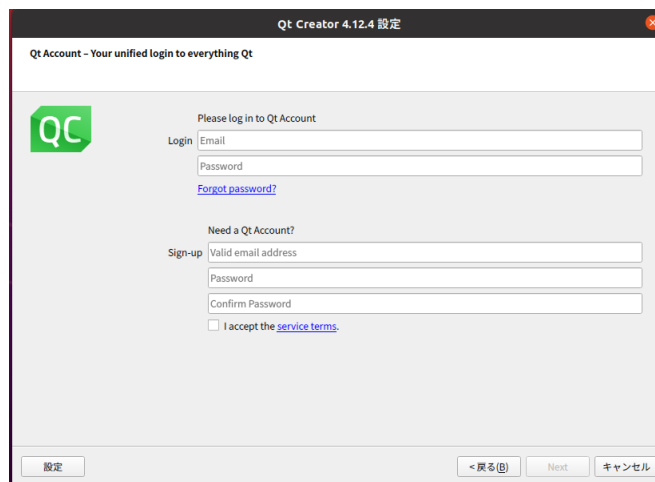


図 2-6-5-2. Qt Creator ライセンス登録

- ④ Qt Creator のインストールフォルダ—選択画面が表示されます。ここでは、「/Opt/Qt/Tools/QtCreator」を指定します。



図 2-6-5-3. Qt Creator インストールフォルダ—

- ⑤ インストールするコンポーネントの選択画面が表示されます。「次へ」をクリックします。

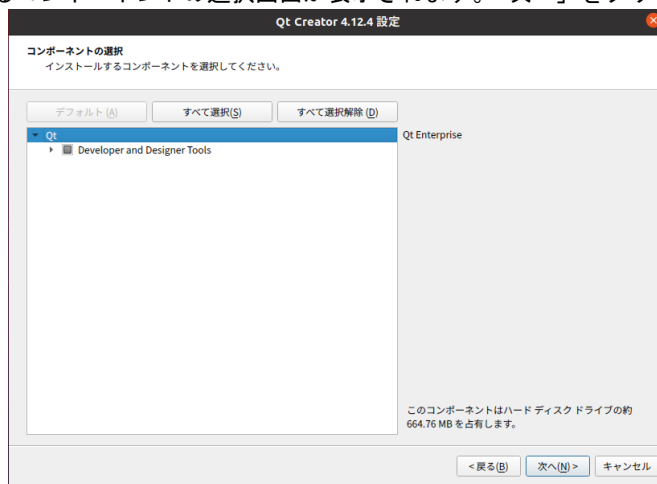


図 2-6-5-4. Qt Creator コンポーネント選択

- ⑥ 使用許諾契約画面が表示されます。「I have read and agree to the terms contained in the license agreements.」を選択し、「次へ」をクリックします。

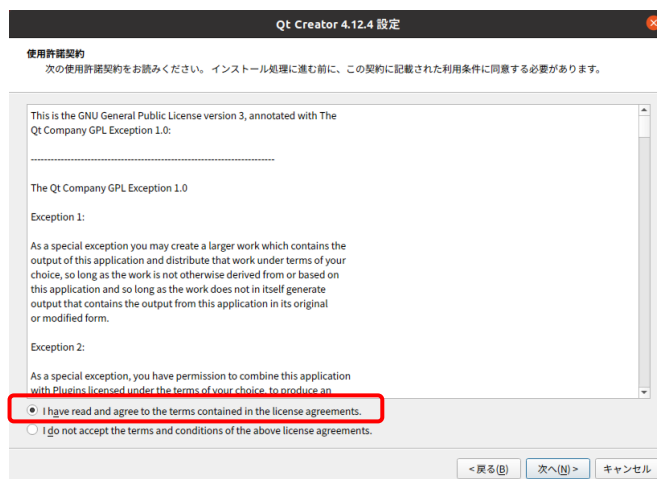


図 2-6-5-5. Qt Creator 使用許諾契約

- ⑦ 「インストール」をクリックすることで、インストールが開始されます。



図 2-6-5-6. Qt Creator インストール

- ⑧ インストール完了後、Qt Creator を起動して設定を行います。「ツール」→「オプション」を選択します。

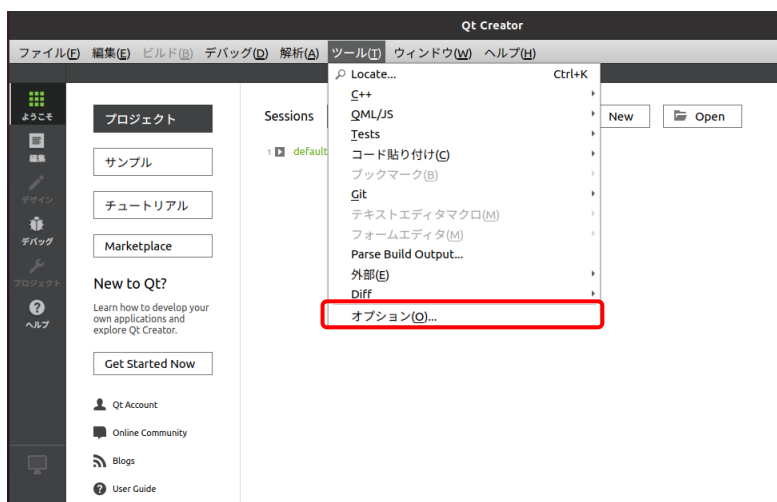


図 2-6-5-7. Qt Creator 起動画面

- ⑨ デバイスを追加します。追加方法は、「2-4-9 リモートデバッグ方法」を参照してください。
 ⑩ コンパイラを設定します。「kits」の「コンパイラ」タブの「追加」→「GCC」→「C」および、「追加」→「GCC」→「C++」を選択し、以下のように設定します。

表 2-6-5-1. ARM 用 GCC

名前	aarch64-linux-gnu-gcc
コンパイラのパス	/usr/local/tools-arm64-toolchain/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-gcc
ABI	arm-linux-generic-elf-64bit

表 2-6-5-2. ARM 用 G++

名前	aarch64-linux-gnu-g++
コンパイラのパス	/usr/local/tools-arm64-toolchain/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-g++
ABI	arm-linux-generic-elf-64bit

- ⑪ デバッガを設定します。「kits」の「デバッガ」タブの「追加」を選択し、以下のように設定します。

表 2-6-5-3. ARM 用 GDB

名前	aarch64-linux-gnu-gdb
コンパイラのパス	/usr/local/tools-arm64-toolchain/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-gdb
ABI	arm-linux-generic-elf-64bit

- ⑫ qmake を設定します。左ペインから「Kits」を選択し、「Qt バージョン」の「追加」をクリックし、以下のよう設定します。

表 2-6-5-4. x86_64 用 Qt バージョン

バージョン名	Qt 5.10.1 (gcc_64)
qmake のパス	/opt/Qt/5.10.1/gcc_64/bin/qmake

表 2-6-5-5. ARM 用 G++

名前	Qt 5.10.1 (qt5)
qmake のパス	/usr/local/tools-arm64-toolchain/arm64-rootfs/opt/qt5/bin/qmake

- ⑬ キットを設定します。「kits」の「キット」タブの「追加」を選択し、以下のよう設定します。

表 2-6-5-6. ARM 用キット

名前	Desktop Qt 5.10.1 GCC 64bit (aarch64)	
Device Type	Generic Linux Device	
Device	⑨で追加したデバイス	
Sysroot	/usr/local/tools-arm64/toolchain/arm64-rootfs	
Compiler	C	aarch64-linux-gnu-gcc
	C++	aarch64-linux-gnu-g++
Debugger	aarch64-linux-gnu-gdb	
Qt version	Qt 5.10.1 (qt5)	

- ⑭ 以上で Qt Creator の設定は完了です。『2-4 Qt によるアプリケーション開発』を参考にアプリケーションを作成し、実行環境上で動作することを確認してください。

2-6-6 Windows 共有の設定

Windows PC から EC1G-01x/02x 用開発環境のファイルを共有フォルダとしてアクセスできるように設定します。

- ① 「samba」サーバパッケージをインストールします。

```
$ sudo apt install samba
```

- ② 「samba」サーバパッケージのコンフィグファイルを設定し、Windows PC から共有できるようにします。
[global]への項目の追加、および[asdusr] と[tools-arm64]の共有フォルダを設定します。

```
$ sudo vi /etc/samba/smb.conf
[global]
    ntlm auth = yes                # <= Windows の NTLM 認証を有効にする
[asdusr]
    path          = /home/asdusr    # <= 共有名
                                # <= 共有ディレクトリ
    browseable    = yes            # <= ディレクトリを表示する
    writable     = yes            # <= 書込可
[tools-arm64]
    path          = /usr/local/tools-arm64 # <= 共有名
                                # <= 共有ディレクトリ
    browseable    = yes            # <= ディレクトリを表示する
    writable     = yes            # <= 書込可
```

- ③ samba にユーザーを追加します。

```
$ sudo pdbedit -a asdusr
```

パスワード等を対話的に問われるので、適切に入力してください。

- ④ 「samba」サーバを再起動して、設定を有効化します。

```
$ sudo systemctl restart smbd
$ sudo systemctl restart nmbd
```

- ⑤ WindowsPC から EC1G-01x/02x 用開発環境 PC へエクスプローラを使って接続してみてください。図 2-6-6-1 のように、ディレクトリが 2 つ見えていれば成功です。



図 2-6-6-1. Windows 共有ディレクトリ

以上でパソコン上に EC1G-01x/02x 用開発環境が構築されました。使用方法については、本書の開発環境の章を参照してください。

付録

A-1 参考文献

- 「ふつ々のLinuxプログラミング Linuxの仕組みから学べるGCCプログラミングの王道」
 - 著者 青木 峰郎
 - 発行所 ソフトバンク パブリッシング
 - 発行年 2005年
- 「How Linux Works Linuxの仕組み」
 - 著者 Brian Ward
 - 訳 吉川 典秀
 - 発行所 毎日コミュニケーションズ
 - 発行年 2006年
- 「Linux デバイスドライバ 第3版」
 - 著者 Jonathan Corbet
Alessandro Rubini
Greg Kroah-hartman
 - 訳 山崎 康宏
山崎 邦子
長原 宏治
長原 陽子
 - 発行所 オライリー・ジャパン
 - 発行年 2005年

このマニュアルについて

- (1)本書の内容の一部又は全部を当社からの事前の承諾を得ることなく、無断で複写、複製、掲載することは固くお断りします。
- (2)本書の内容に関しては、製品改良のためお断りなく、仕様などを変更することがありますのでご了承下さい。
- (3)本書の内容に関しては万全を期しておりますが、万一ご不審な点や誤りなどお気づきのことがございましたらお手数ですが巻末記載の弊社までご連絡下さい。その際、巻末記載の書籍番号も併せてお知らせ下さい。

77EC60004A

2021年 1月 初版

 株式会社アルゴシステム

本社
〒587-0021 大阪府堺市美原区小平尾656番地

TEL (072) 362-5067
FAX (072) 362-4856

ホームページ <http://www.algosystem.co.jp>