マニュアル

Debian9 Linux ディストリビューション

『Algonomix6.0』 開発環境について

目　次

はじめに

第１章　概要

第２章　開発環境

付録

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、Debian9 Linux ディストリビューション（以降 **Algonomix6**）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

Algonomix6 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。Algonomix6 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、Algonomix6 開発環境の具体的な内容を説明する前に、Algonomix6 開発環境の概要について説明します。

１－１　Algonomix6 とは

「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、・・・・）
- procps（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

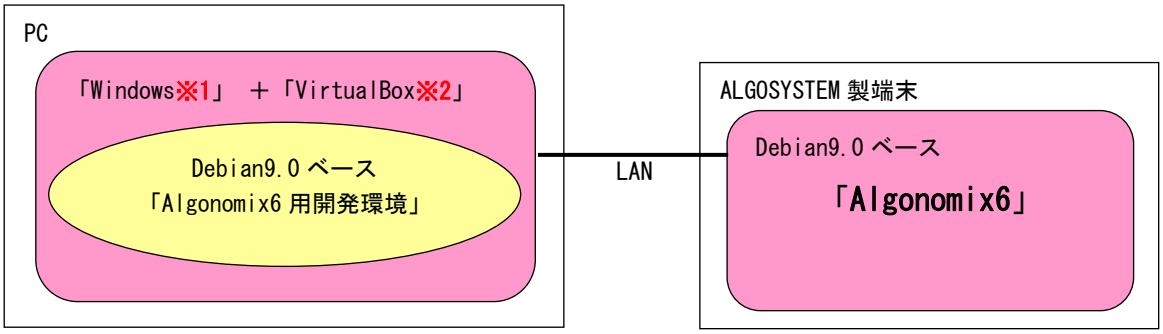
Linux カーネルといくつかの必要なソフトウェアパッケージをまとめて、OS として使えるようにしたものを Linux ディストリビューションといいます。

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。

Algonomix6 は、「Debian9.0 （コードネーム：Stretch)」という Linux ディストリビューションをベースにした、Linux ディストリビューションです。Algonomix6 は、Debian9.0 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 Algonomix6 用の開発環境をインストールすることで、Algonomix6 用のソフトウェアを開発することができます。

Algonomix6 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. Algonomix6 の開発環境

端末毎の Algonomix6 の詳細は、各端末毎の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させていただきます。

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。

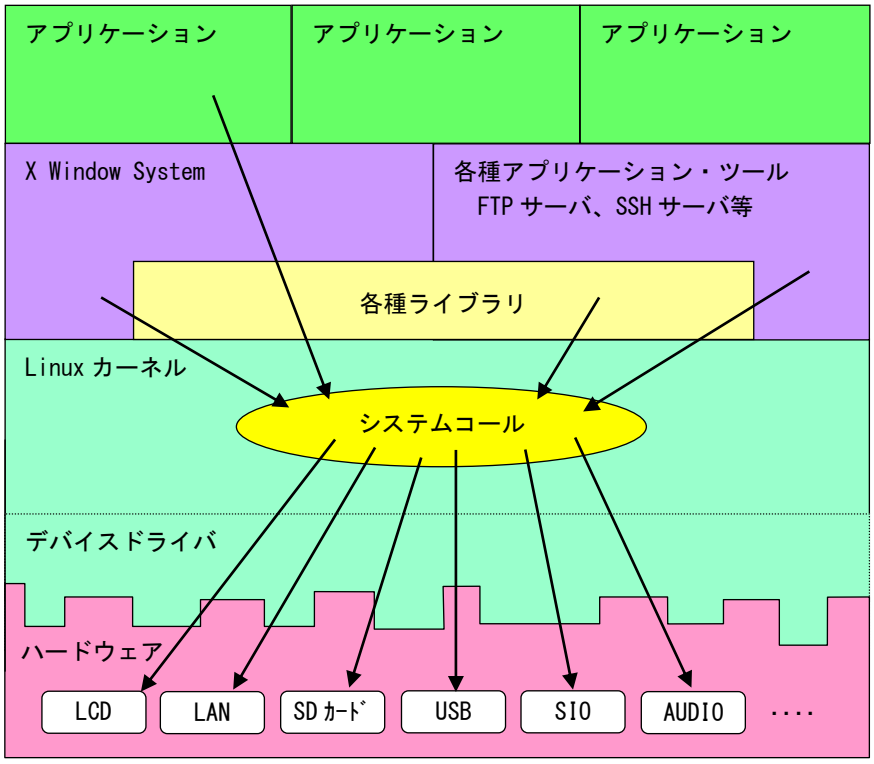


図 1-2-1. Linux ソフトウェア構成図

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、Algonomix6 の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となるのが、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）。

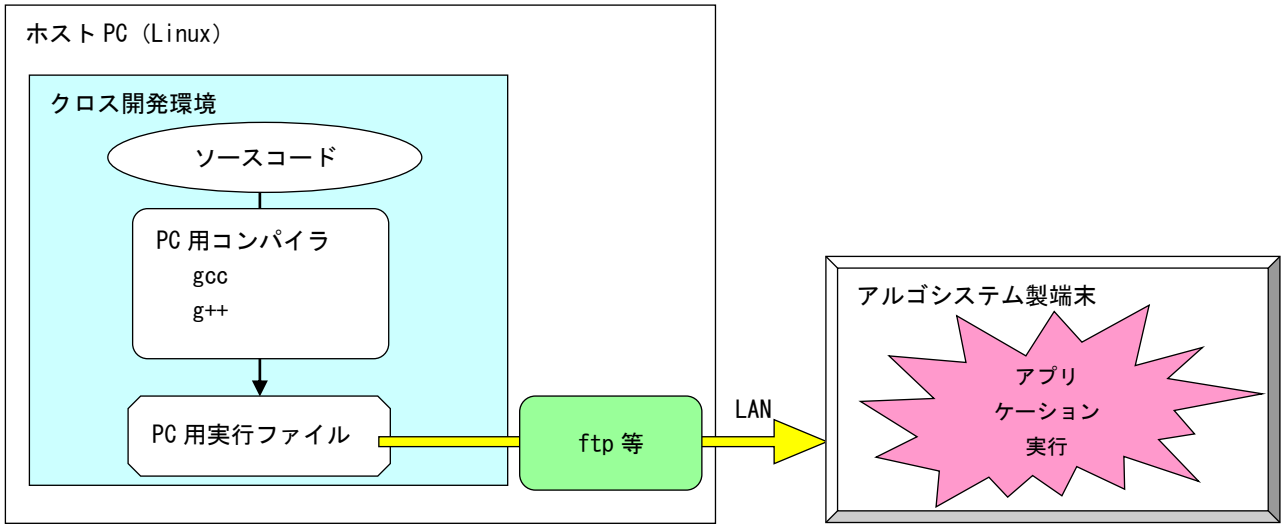


図 2-1-1. クロス開発方式イメージ図

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

Algonomix6 開発環境は、Debian9.0 ディストリビューションイメージに、Intel CPU 用と ARM CPU 用の２種類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 Algonomix6 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**Algonomix6 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となるものは下記の 3 点です。

１．Algonomix6 用開発環境 SD カード

Algonomix6 用アプリケーション開発環境一式です。
SD カードの内容を表 2-2-1-1 に示します。

※注：通常、開発環境 SD カードは弊社の端末ご購入時には添付しておりません。営業までお問い合わせください。（有償）

表 2-2-1-1. Algonomix6 用開発環境 SD カードの構成

SD カードのディレクトリ	内容
doc	Algonomix6 開発環境の取扱い説明書が格納されています。 ・Algonomix6 開発環境ユーザーズマニュアル.pdf 本書のことです。
development	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 Algonomix6 用パッケージが格納されています。 ・Algonomix6-tools-x64.tar.gz（Intel CPU 用） ・Alognomix6-tools-arm64.tar.gz（ARM CPU 用）
VirtualBox	VirtualBox 用の OS イメージファイルが格納されています。 ・Algonomix6_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 Algonomix6 開発環境 OS 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：Algonomix6 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。Algonomix6 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 上の環境では Algonomix6 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）

Algonomix6 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（Debian9）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。

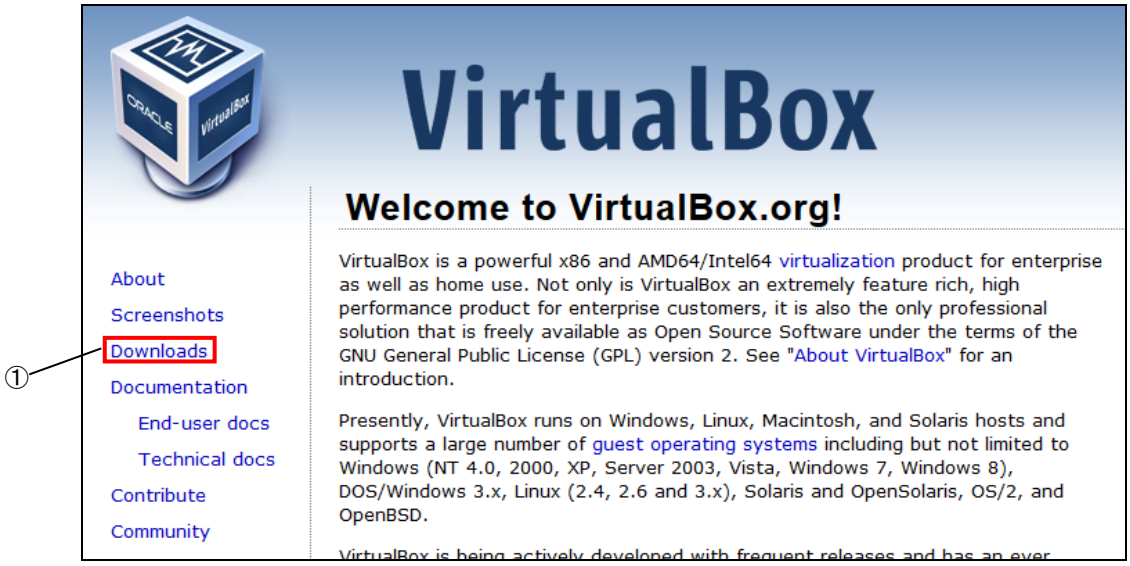


図 2-2-2-1.　公式 TOP ページ

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。

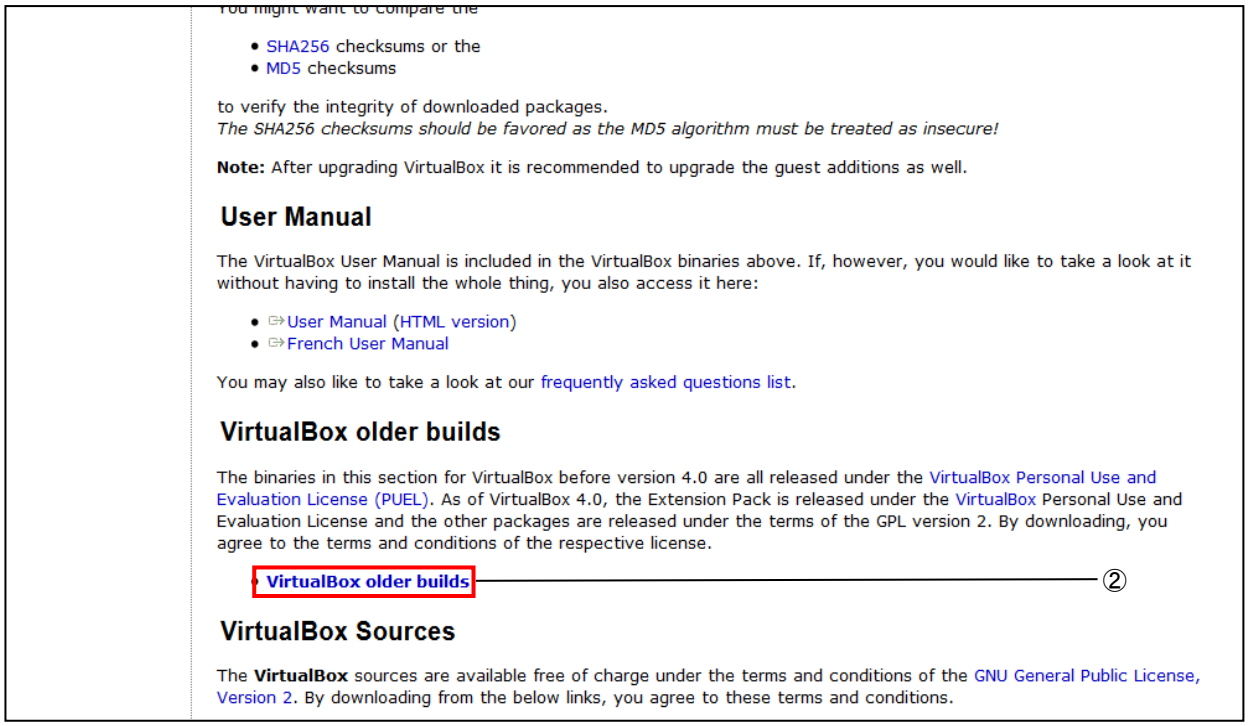


図 2-2-2-2.　VirtualBox ダウンロードページ

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。

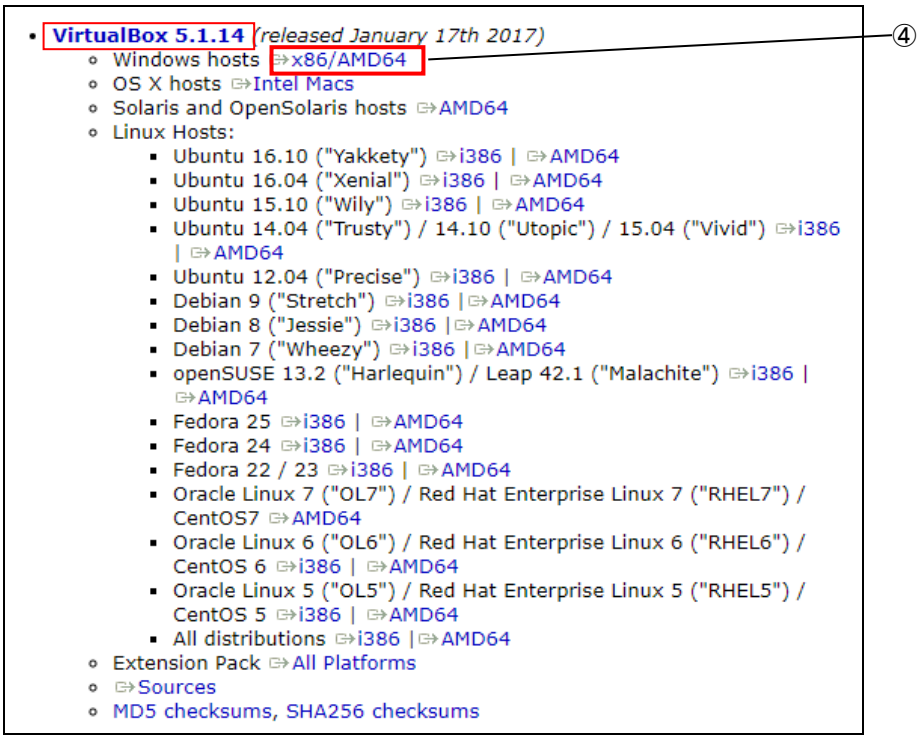


図 2-2-2-4. 古いバージョンの VirtualBox ダウンロードページ

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。

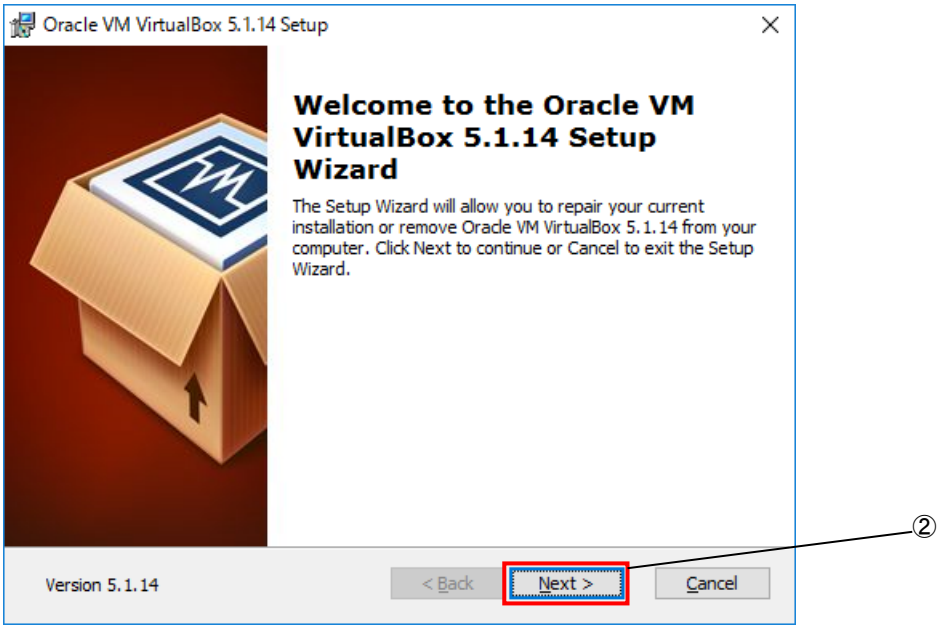


図 2-2-3-1.　ダウンローダー起動後の画面

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。

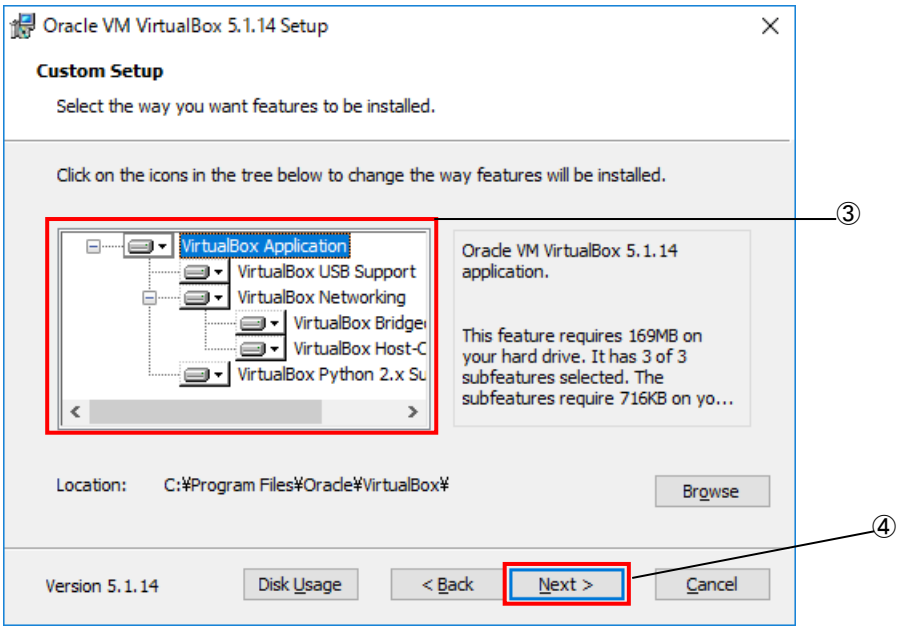


図 2-2-3-2.　カスタムセットアップ画面

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。

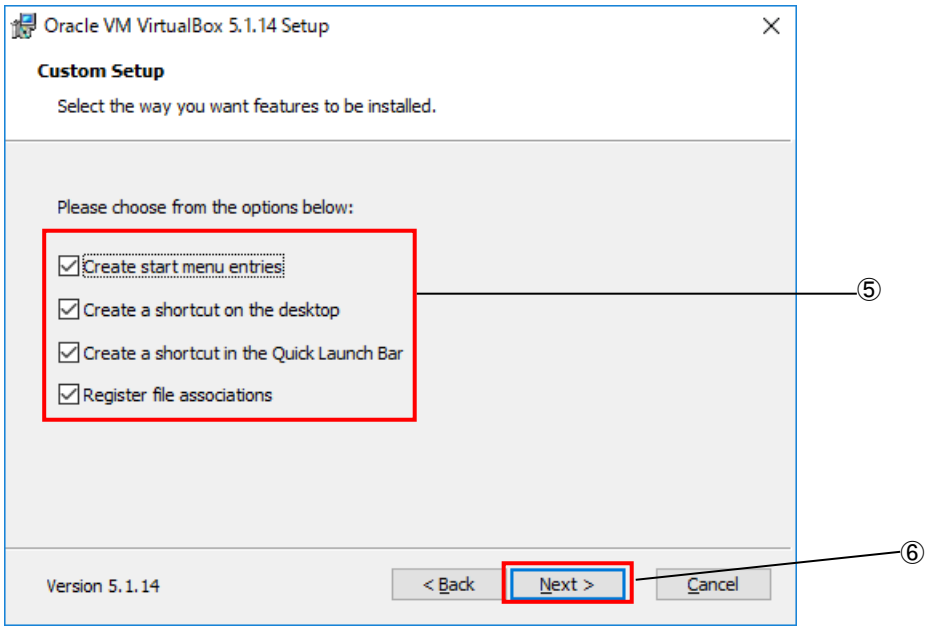


図 2-2-3-3.　インストールオプション設定画面

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。

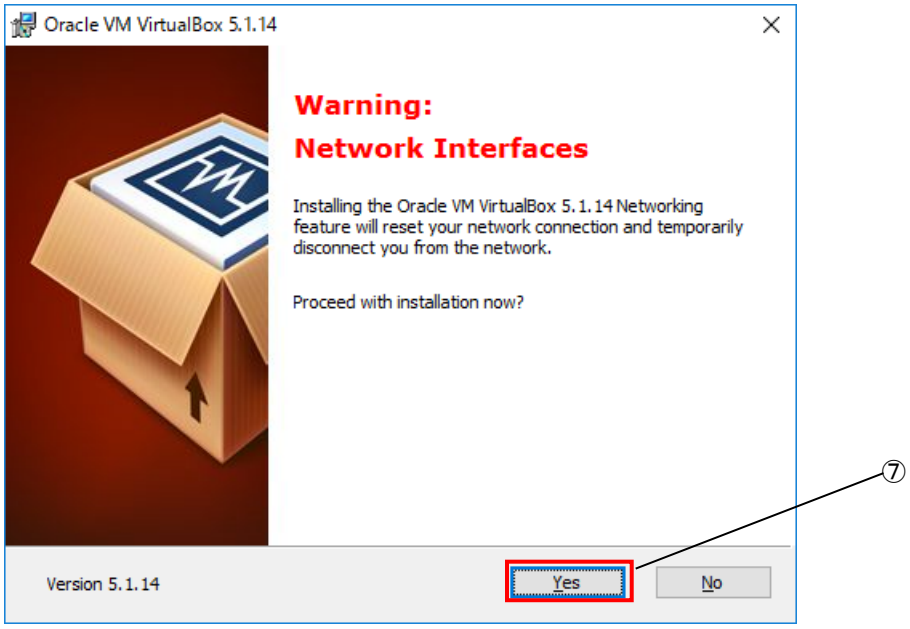


図 2-2-3-4.　ネットワークインストール確認画面

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。

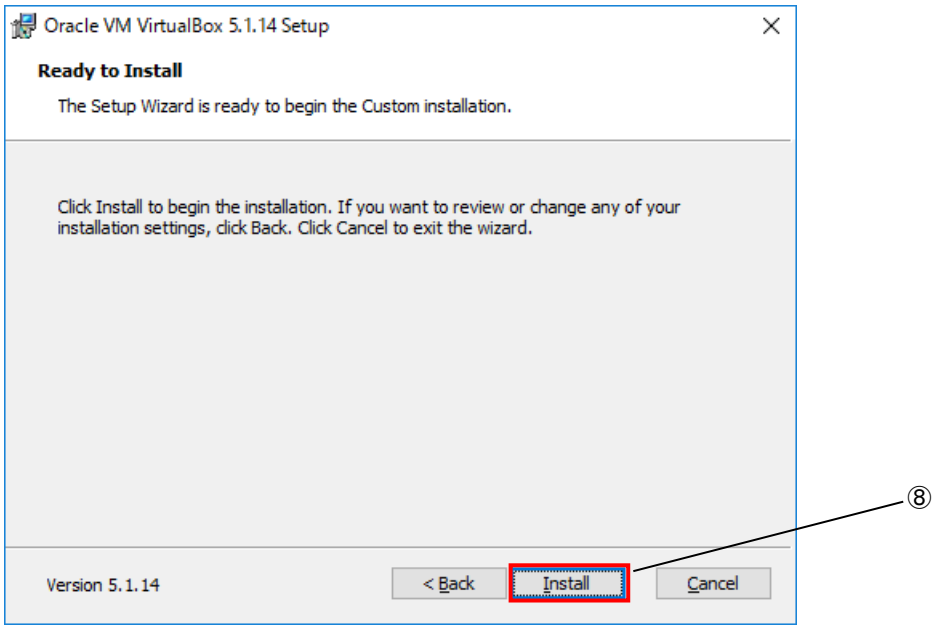


図 2-2-3-5.　インストール確認画面

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。

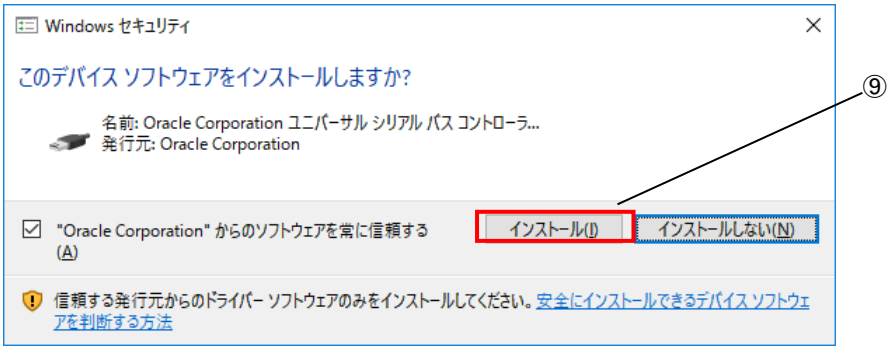


図 2-2-3-6.　ドライバのインストール警告画面

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。

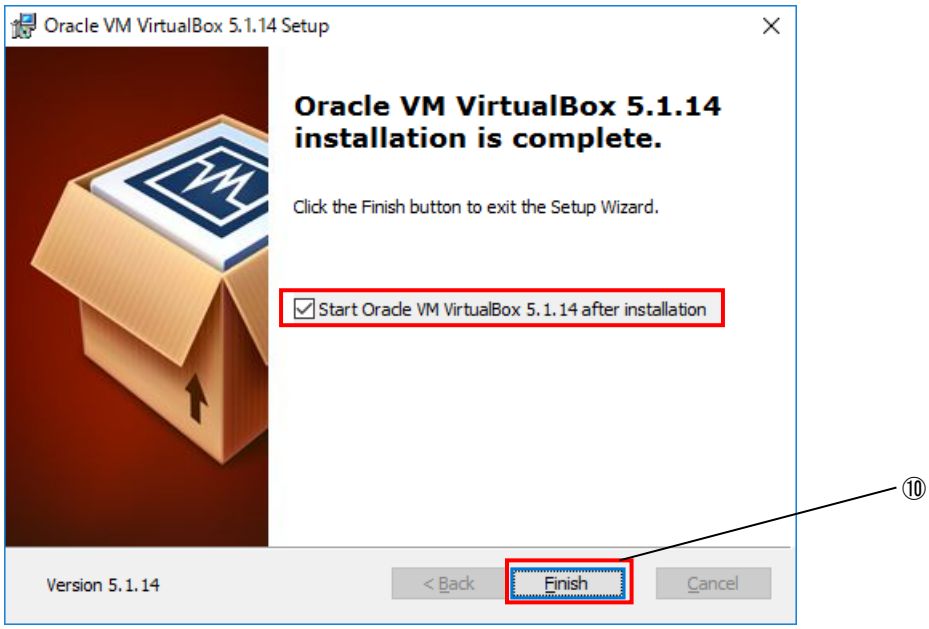


図 2-2-3-7. インストール完了画面

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。

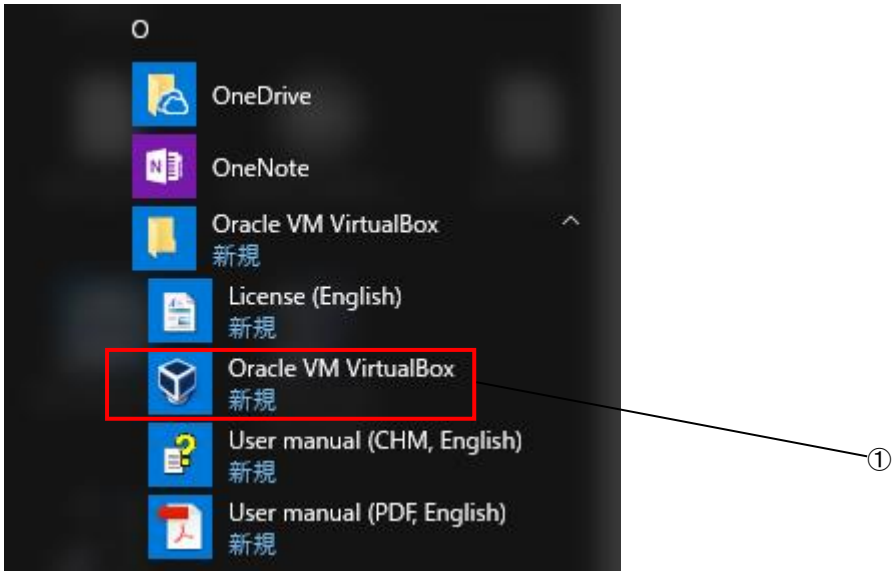


図 2-2-4-1.　VirtualBox 起動

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¥Algonomix6_Development64.7z をこのフォルダに展開します。

Algonomix6_Development64.7z を解凍することで Algonomix6_Development64.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。

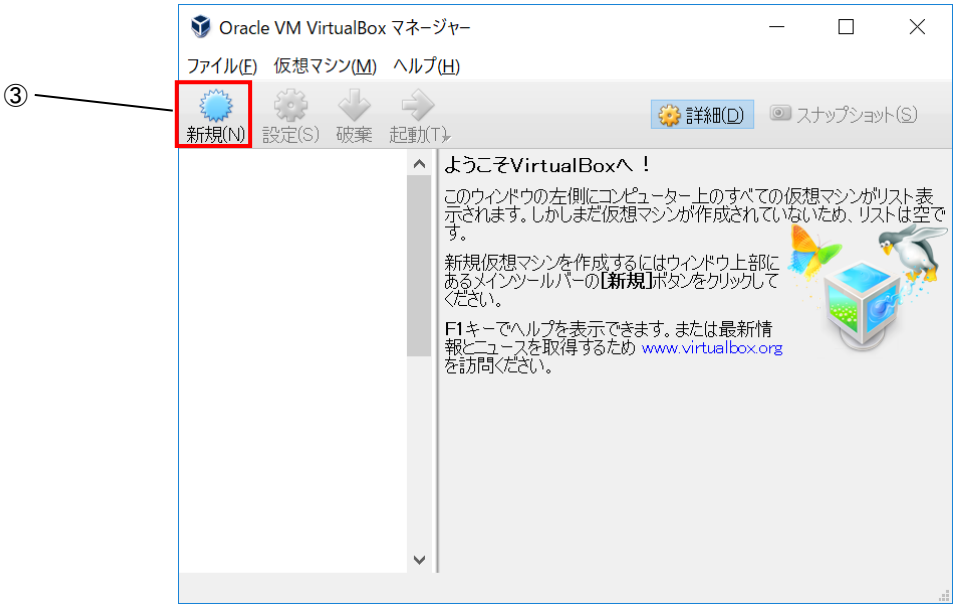


図 2-2-4-2.　VirtualBox メイン画面

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[Algonomix6_Development64]と入力します。
⑤　[OS タイプ(T)]では[Linux][Debian(64-bit)]を選択します。
⑥　[次へ(N)]をクリックします。

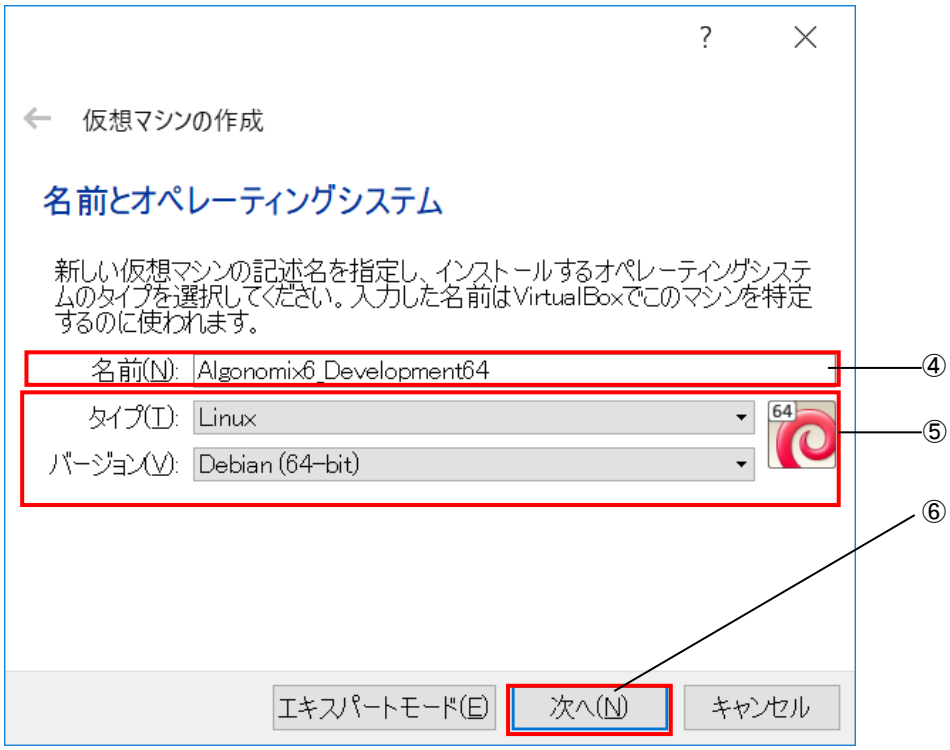


図 2-2-4-3. マシン名、OS タイプの設定

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。

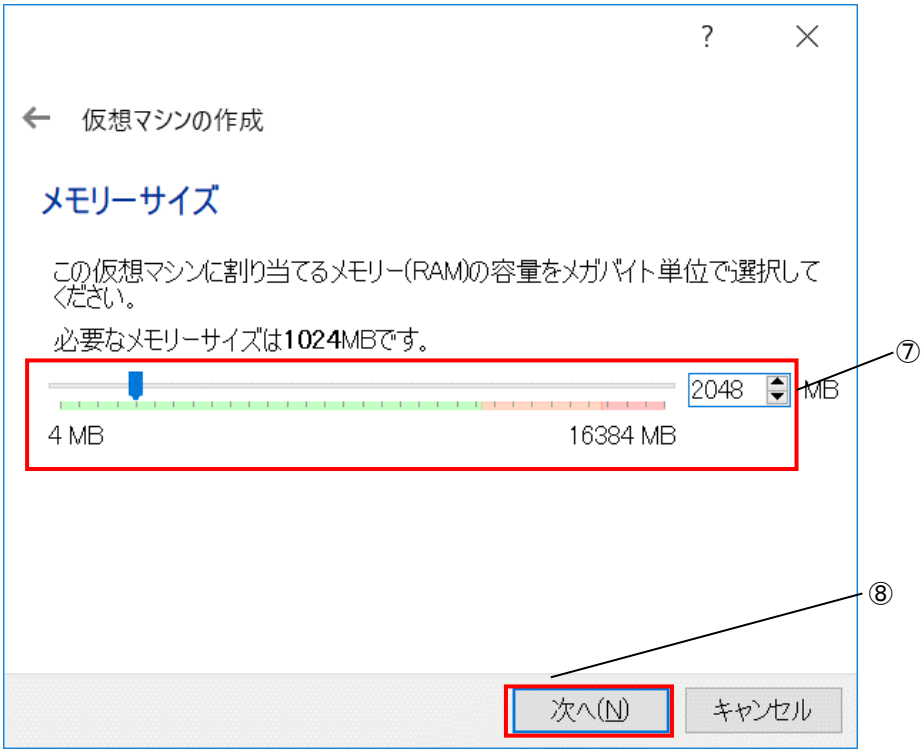


図 2-2-4-4.　メモリの設定

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 Algonomix6_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 Algonomix6_Development64.vdi がある場合、[作成]をクリックし、⑬へ進んでください。

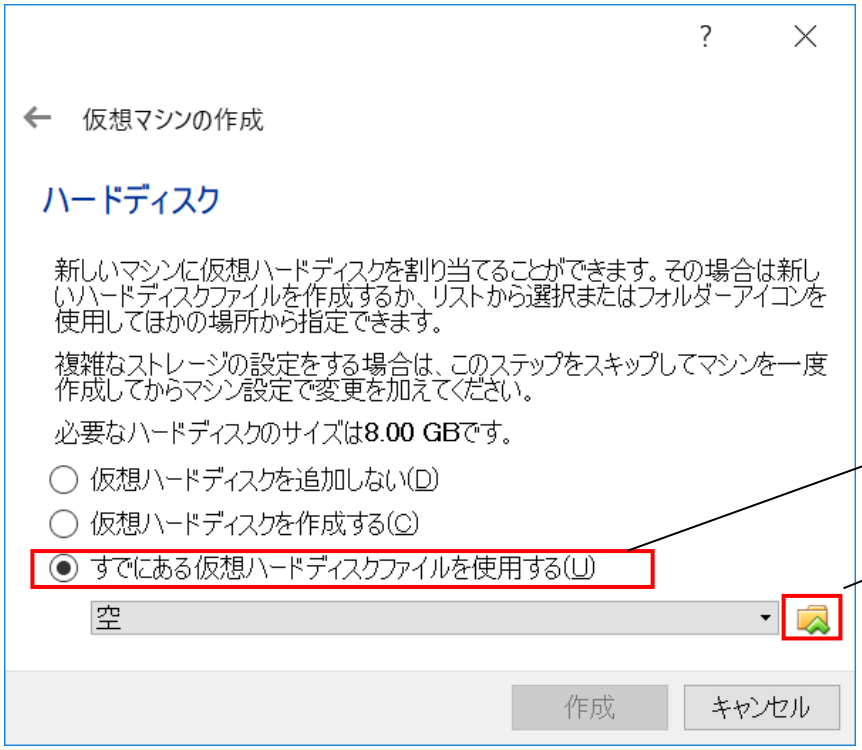


図 2-2-4-5.　仮想ハードディスクのイメージ選択画面

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。
②であらかじめ展開しておいた Algonomix6_Development64.vdi を選択します。

⑫　[開く(O)]を押してください。

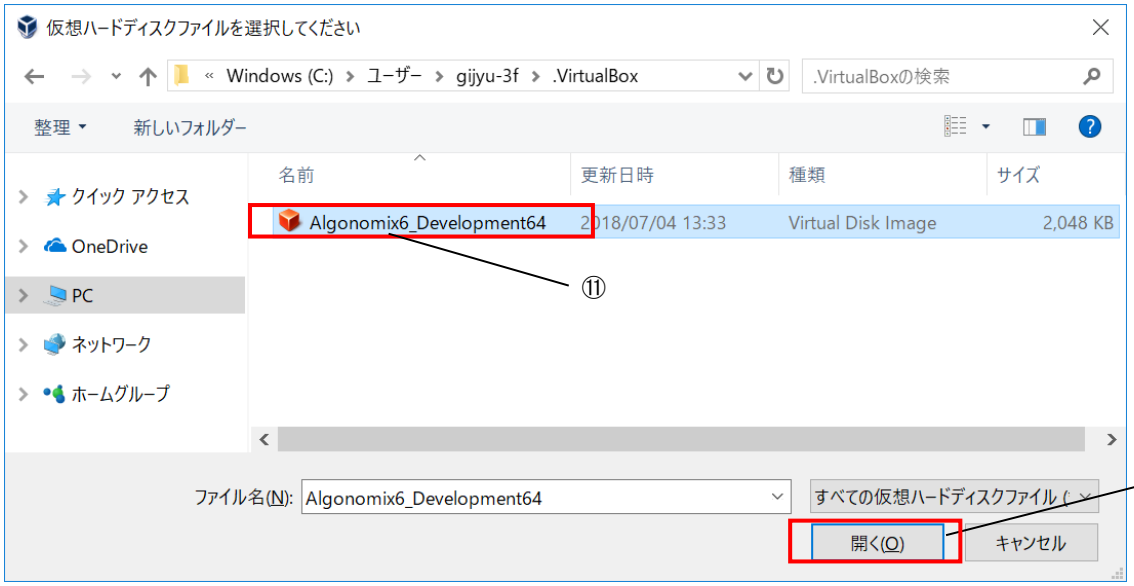


図 2-2-4-6.　仮想ディスク選択画面

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。

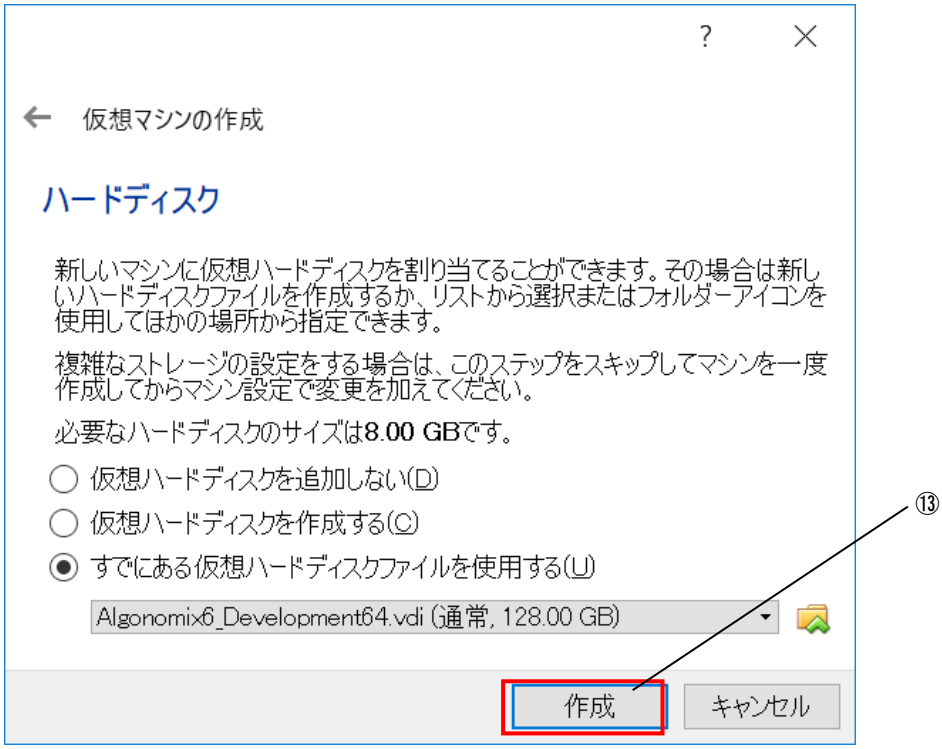


図 2-2-4-7.　仮想ハードディスクのイメージ選択画面

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。

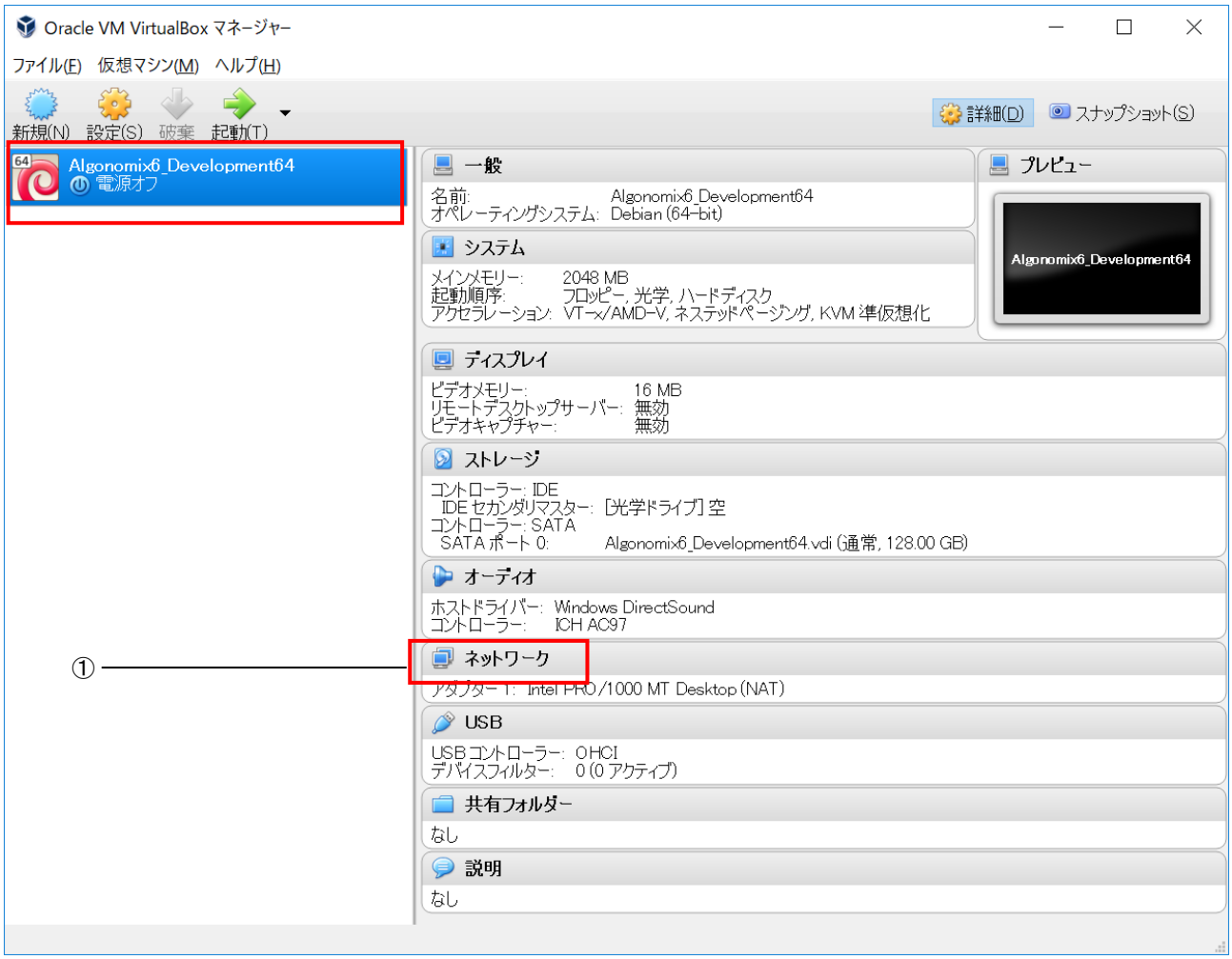


図 2-2-5-1. VirtualBox 初期画面

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。

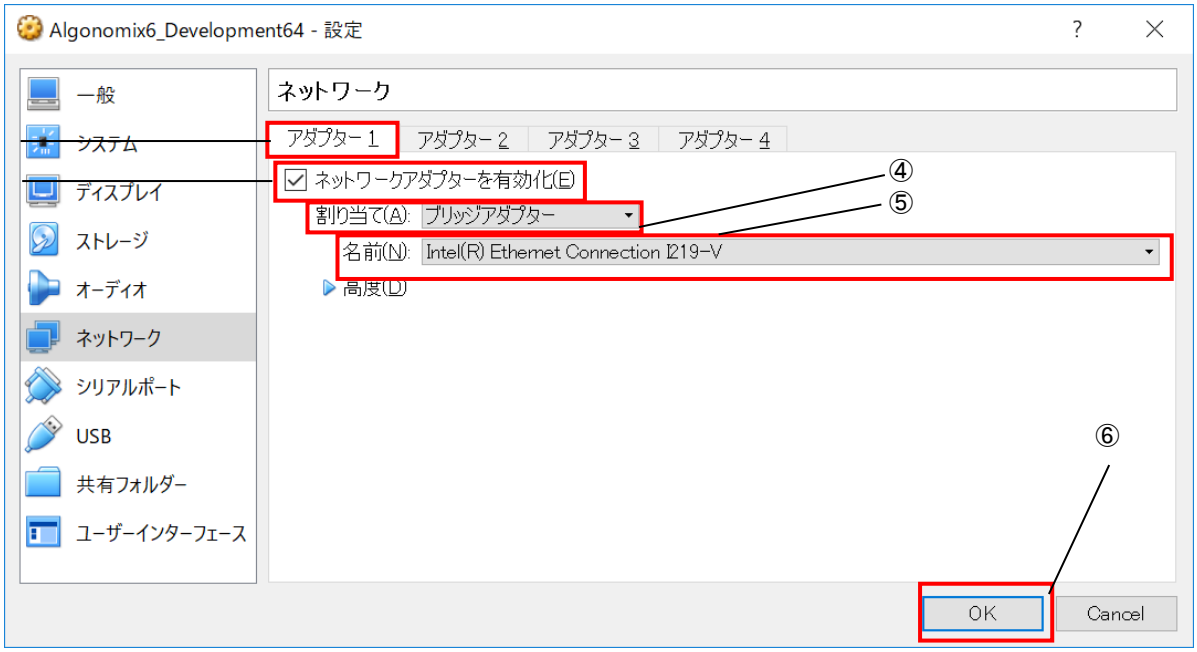


図 2-2-5-2. ネットワーク設定画面

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。

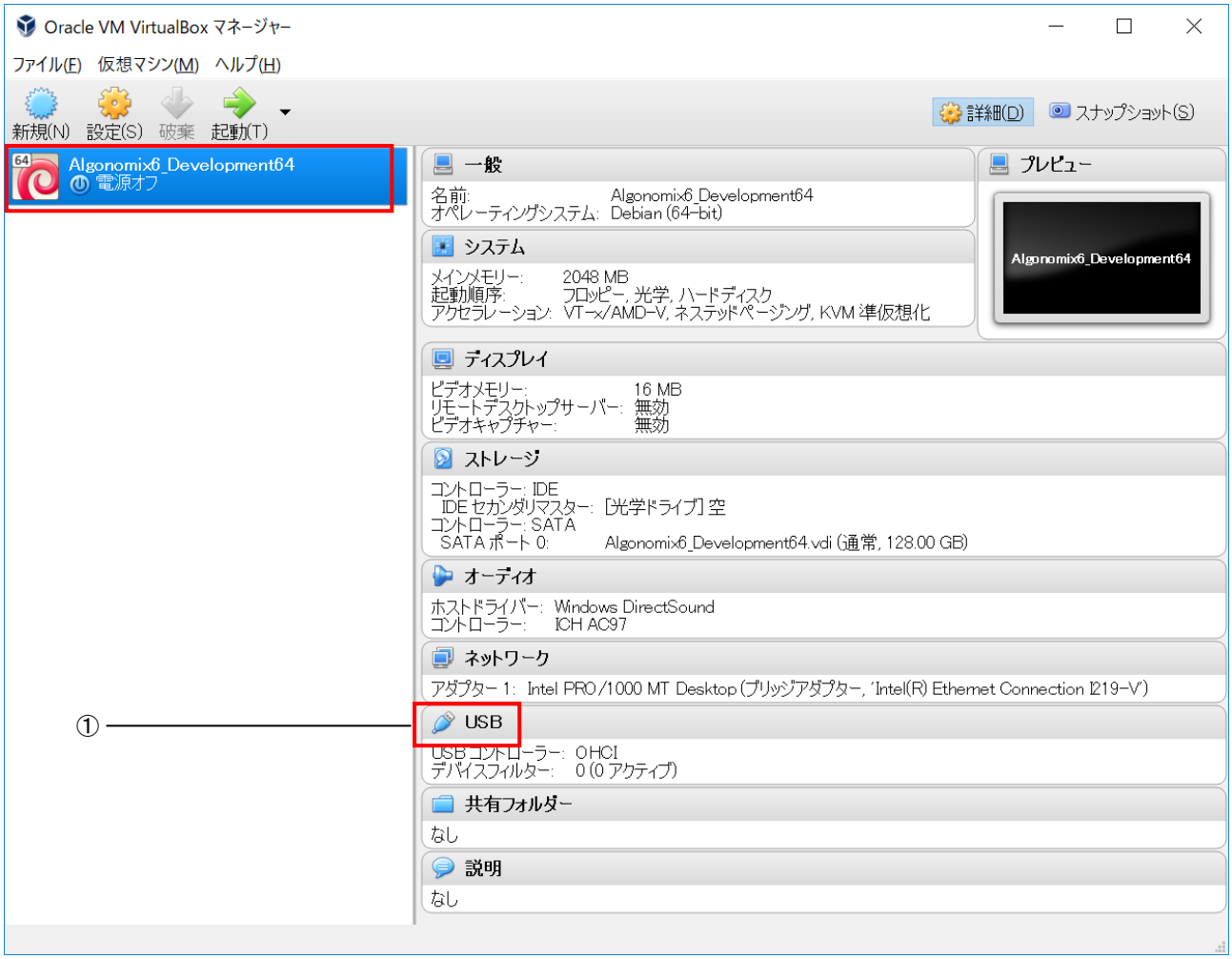


図 2-2-6-1.　VirtualBox 初期画面

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。

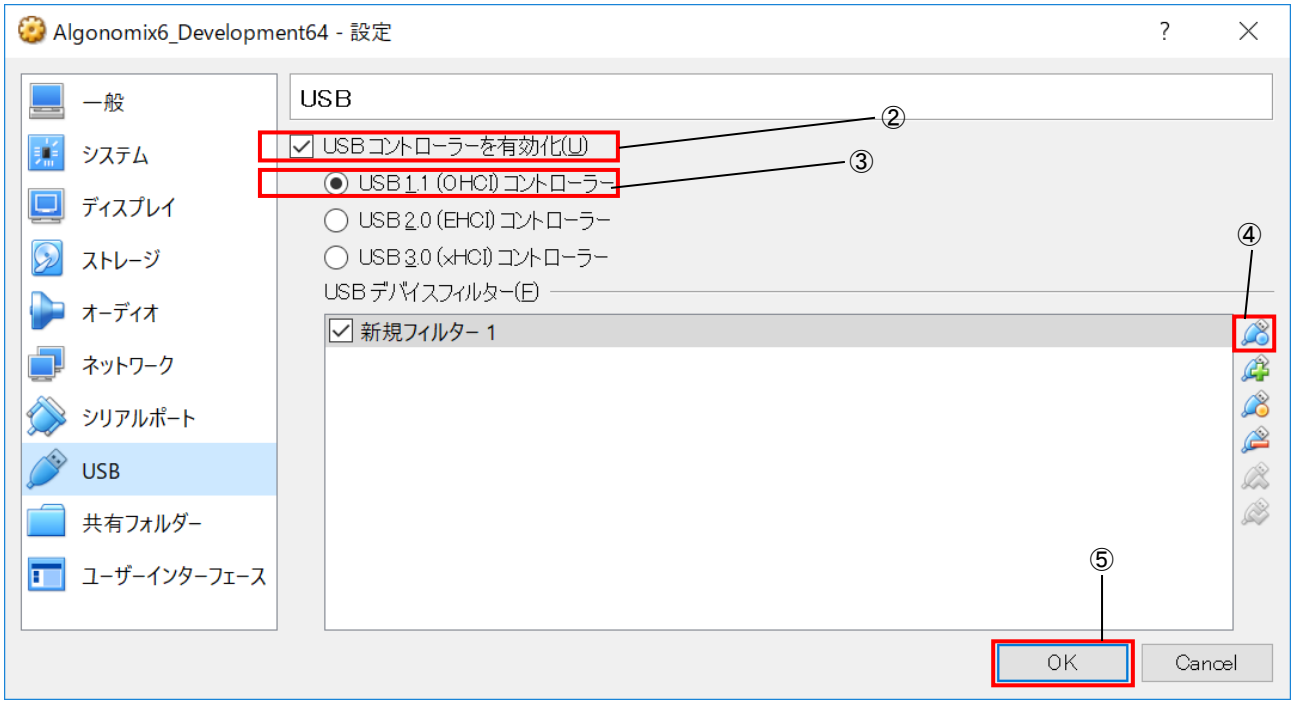


図 2-2-6-2. USB 設定画面

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。

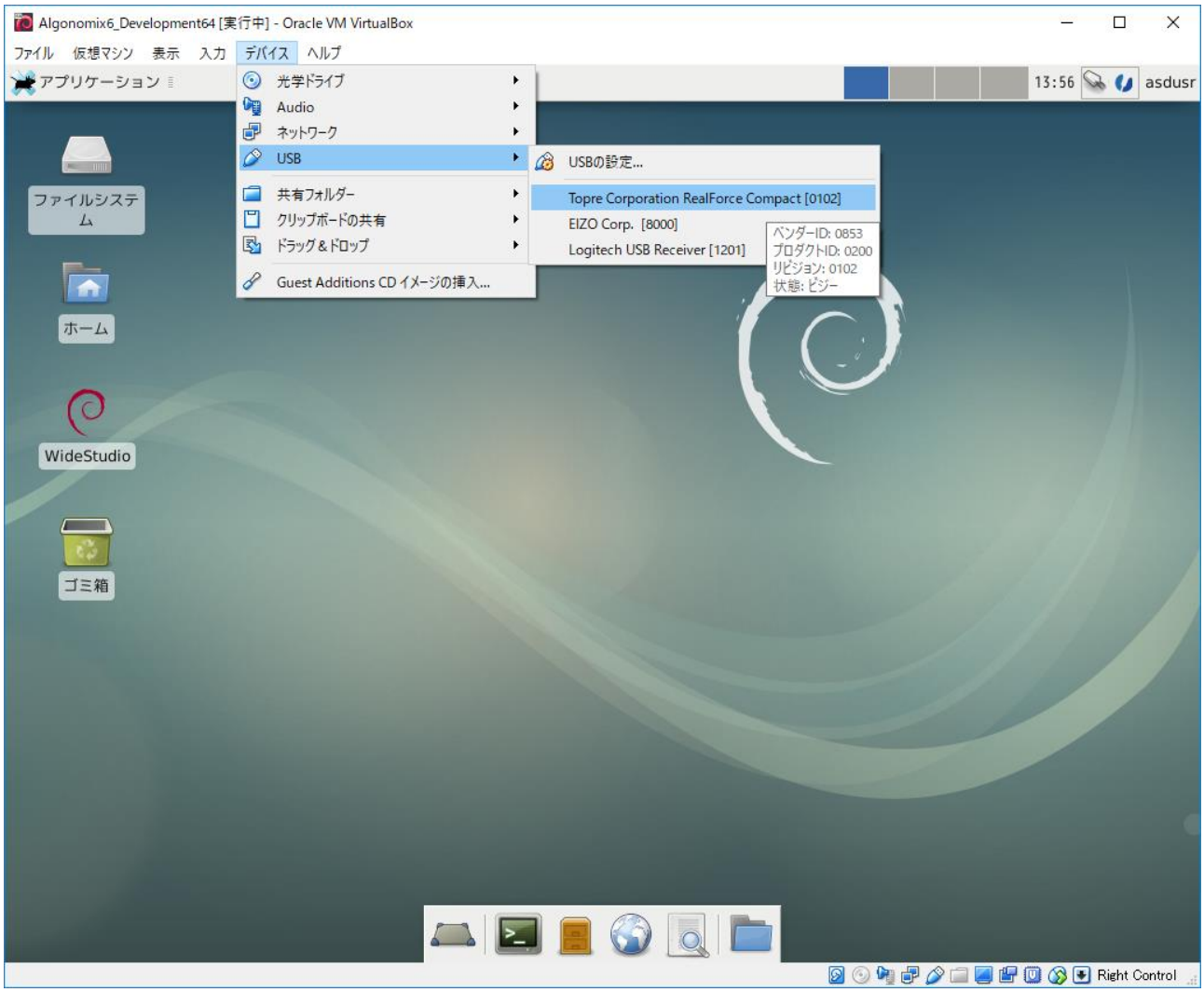


図 2-2-6-3. USB デバイス一覧

※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、Algonomix6_Development64.vdi は既に Algonomix6 開発環境がインストールされています。なお、Algonomix6_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

管理者権限のユーザ名とパスワードは下記のようになっています。
ユーザ名　　：　root
パスワード　：　rootroot

①　VirtualBox メイン画面左側の List の中から Algonomix6_Development64 を選択します。
②　[起動(T)]をクリックします。

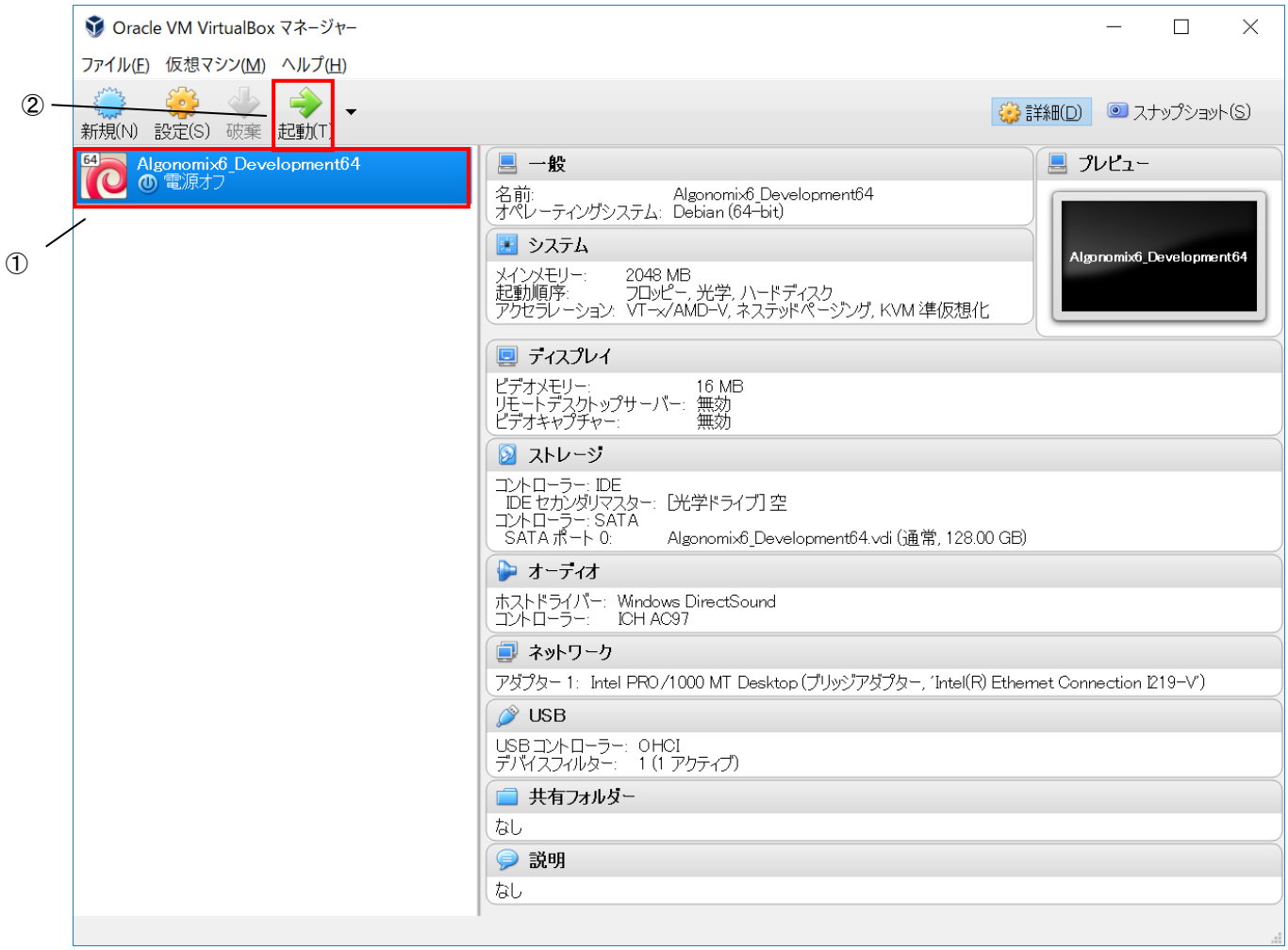


図 2-2-7-1.　仮想マシンの起動

図 2-2-7-2 のような仮想マシンのユーザ名、パスワード入力画面が起動します。

③　ユーザ名と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

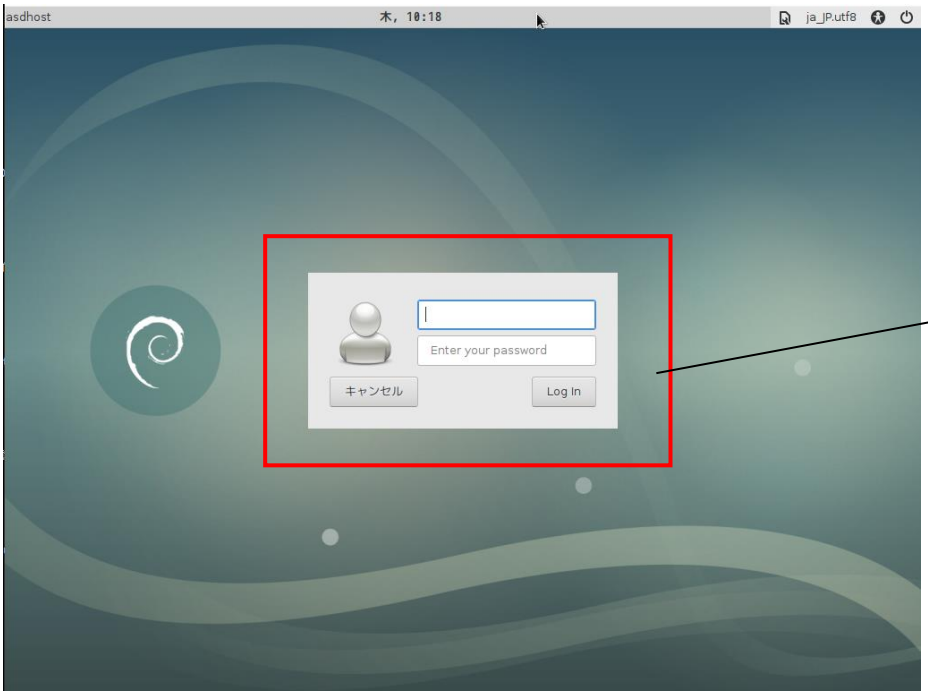


図 2-2-7-2. ユーザ名、パスワード入力画面

図 2-2-7-3 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。

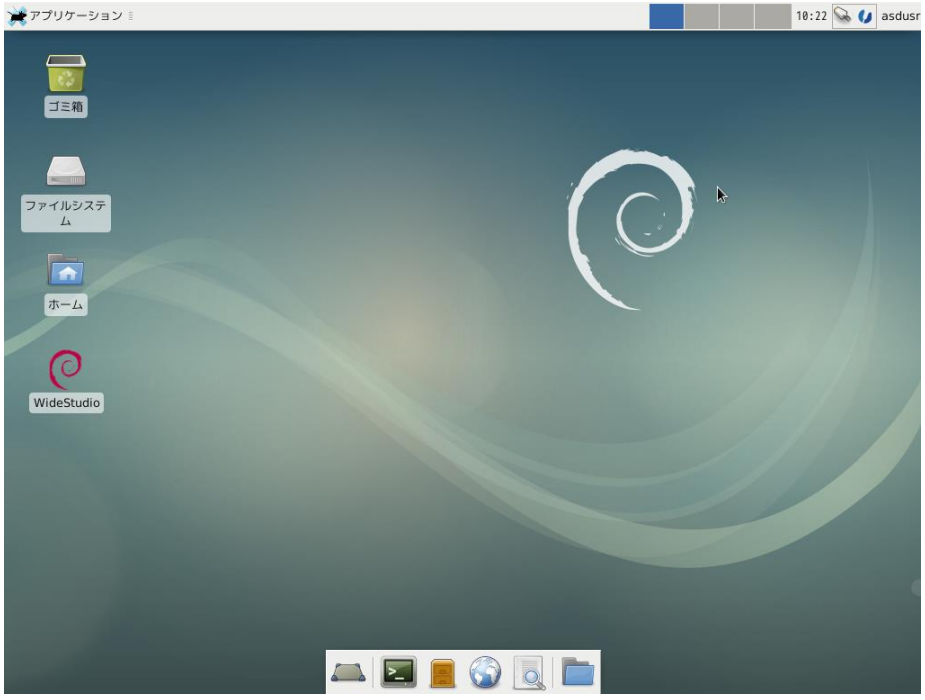


図 2-2-7-3. デスクトップ画面

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。

● 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 [アプリケーション]→[設定]→[ネットワーク接続]を選択します。

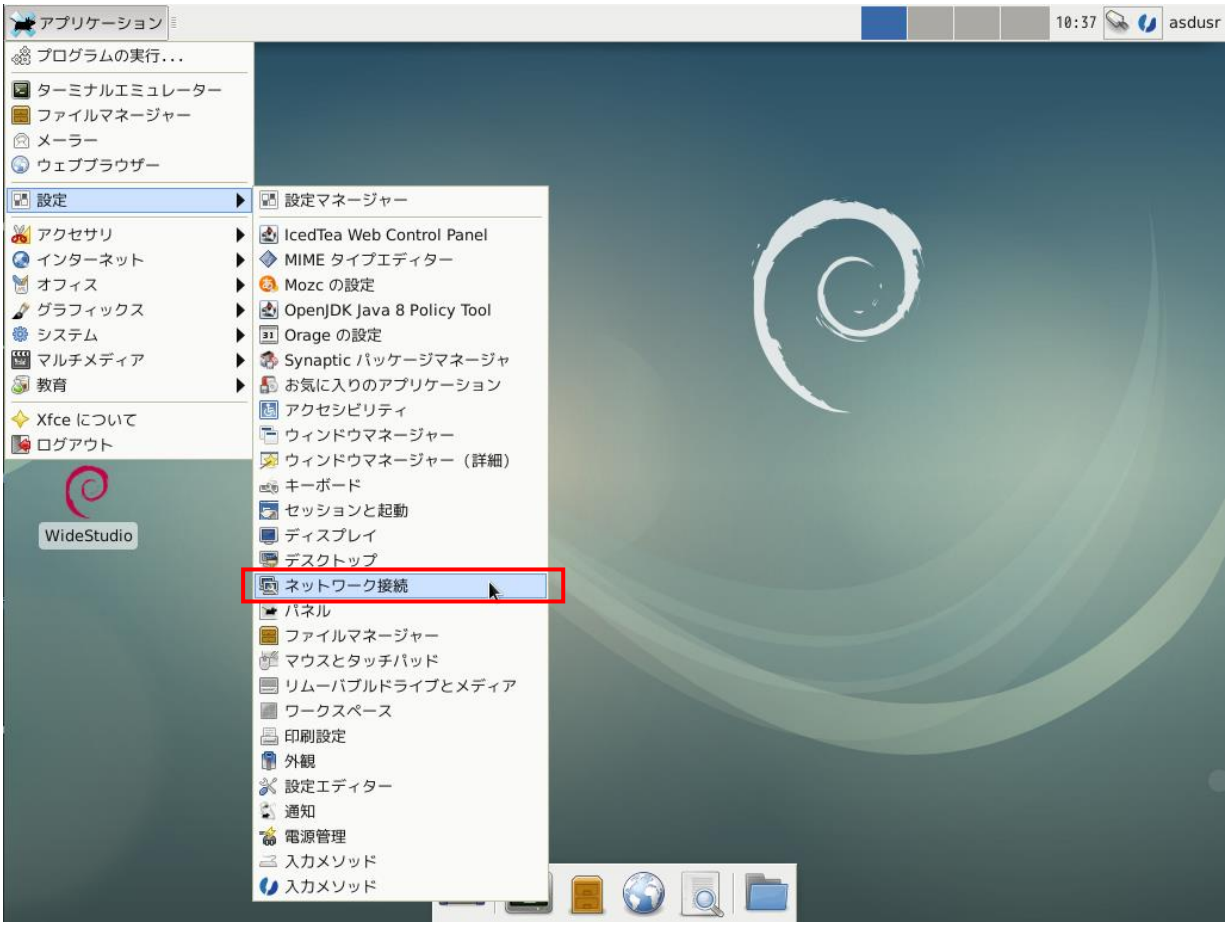


図 2-2-8-1. ネットワーク設定画面の起動

② 図 2-2-8-2 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[編集]ボタンをクリックします。

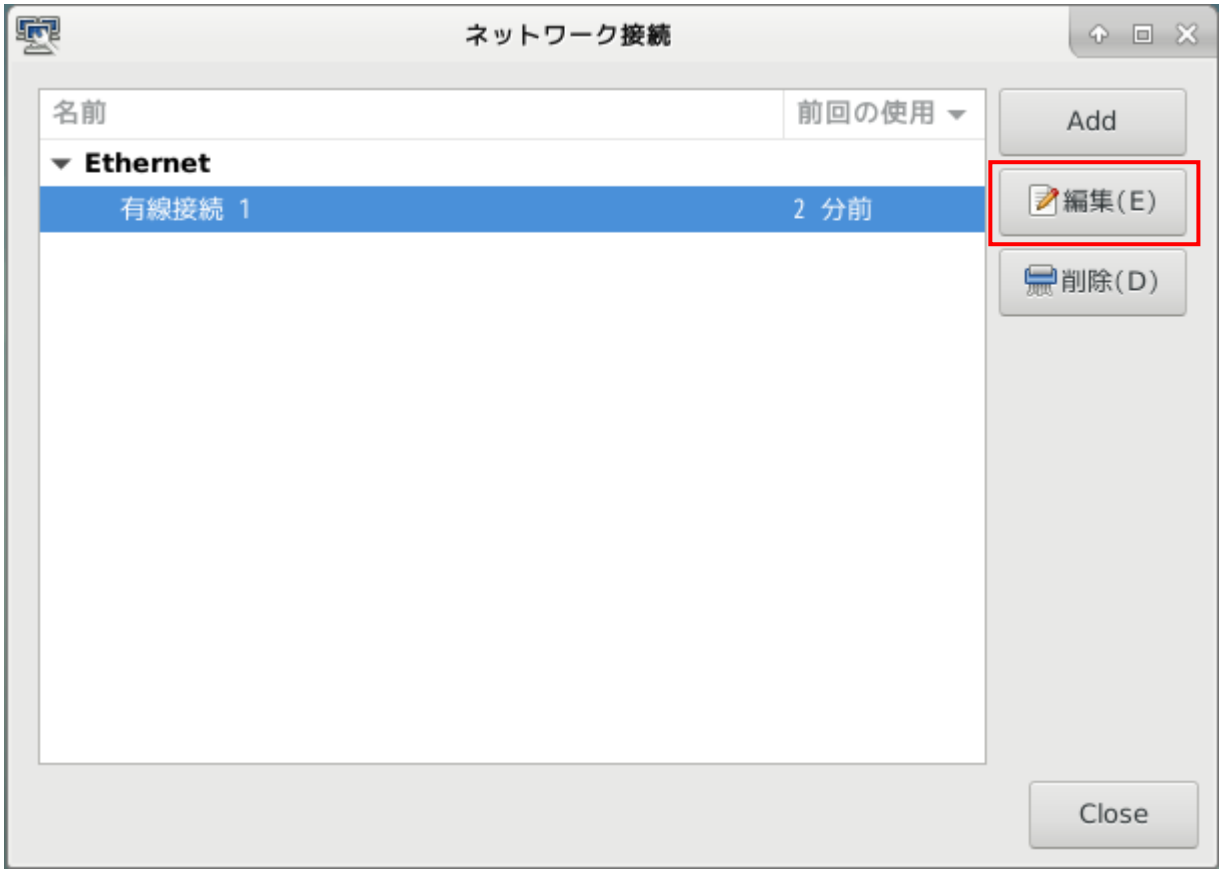


図 2-2-8-2. ネットワーク設定画面

③　図 2-2-8-3 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 2-2-8-3.　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、メンテナンスされいないため、数年前に非推奨なパッケージになりました。
Debian9.0 からは「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。
今日の Linux ディストリビューションでは、「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　Algonomix6 用開発環境のディレクトリ構成について

Algonomix6 用開発環境のディレクトリ構成について説明します。Algonomix6 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.　Algonomix6 用開発環境のディレクトリ構成

```
lib--+--x86_64-linux-gnu
     +--aarch64-linux-gnu
usr--+--lib--+--x86_64-linux-gnu
             +--aarch64-linux-gnu
     +--local--+--tools-x64----+--source--+--apl
                                          +--kernel
                                          +--pci
                                          +--widestudio
                               +--samples-+--sampleConsole
                                          +--sampleWideStudio
               +--tools-arm64--+--skales
                               +--source--+--apl
                                          +--kernel
                                          +--pci
                               +--toolchain
               +--ws
```

また、これらのディレクトリの内容を表 2-2-9-1、表 2-2-9-2、表 2-2-9-3 に示します。

表 2-2-9-1. ディレクトリの内容（tools-x64）

ディレクトリ名	内　容
/lib/x86_64-linux-gnu	Intel CPU 用ライブラリ
/usr/lib/x86_64-linux-gnu	Intel CPU 用 USR ライブラリ
/usr/local/tools-x64	Intel CPU Debian9.0 64bit 用の開発環境が格納されています。
/usr/local/tools-x64/source	Algonomix6 搭載 Intel CPU 端末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-x64/source/apl	Algonomix6 搭載 Intel CPU 端末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-x64/source/kernel	Algonomix6 搭載 Intel CPU 端末用のカーネルソースです。
/usr/local/tools-x64/source/widestudio	本開発環境にインストールしている WideStudio ソースおよび Algonomix6 用のライブラリです。
/usr/local/tools-x64/source/pci	Algonomix6 搭載 Intel CPU 端末用の PCI ドライバです。
/usr/local/tools-x64/samples	Algonomix6 搭載 Intel CPU 端末用用のサンプルプログラムです。

表 2-2-9-2. ディレクトリの内容（tools-arm64）

ディレクトリ名	内　容
/lib/aarch64-linux-gnu	SnapDragon CPU 用ライブラリ
/usr/lib/aarch64-linux-gnu	SnapDragon CPU 用 USR ライブラリ
/usr/local/tools-arm64	SnapDragon CPU Debian9.0 64bit 用の開発環境が格納されています。
/usr/local/tools-arm64/source	Algonomix6 搭載 SnapDragon CPU 端末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	Algonomix6 搭載 SnapDragon CPU 端末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/kernel	Algonomix6 搭載 SnapDragon CPU 端末用のカーネルソースです。
/usr/local/tools-arm64/source/pci	Algonomix6 搭載 SnapDragon CPU 端末用の PCI ドライバです。
/usr/local/tools-arm64/skales	Kernel をビルドする際に必要となるデバイスツリー生成プログラム
/usr/local/tools-arm64/toolchain	SnapDragon 用クロスコンパイル環境
/usr/local/tools-x64/samples	Algonomix6 用のサンプルプログラムです。

表 2-2-9-3. ディレクトリの内容（共通）

ディレクトリ名	内　容
/usr/local/ws	WideStudio 本体です。

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。

Algonomix6 用開発環境に組込まれている WideStudio/MWT は V3.98-7 をベースに Algonomix6 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

デスクトップ上の 

のアイコンをダブルクリックすることで WideStudio が起動します。

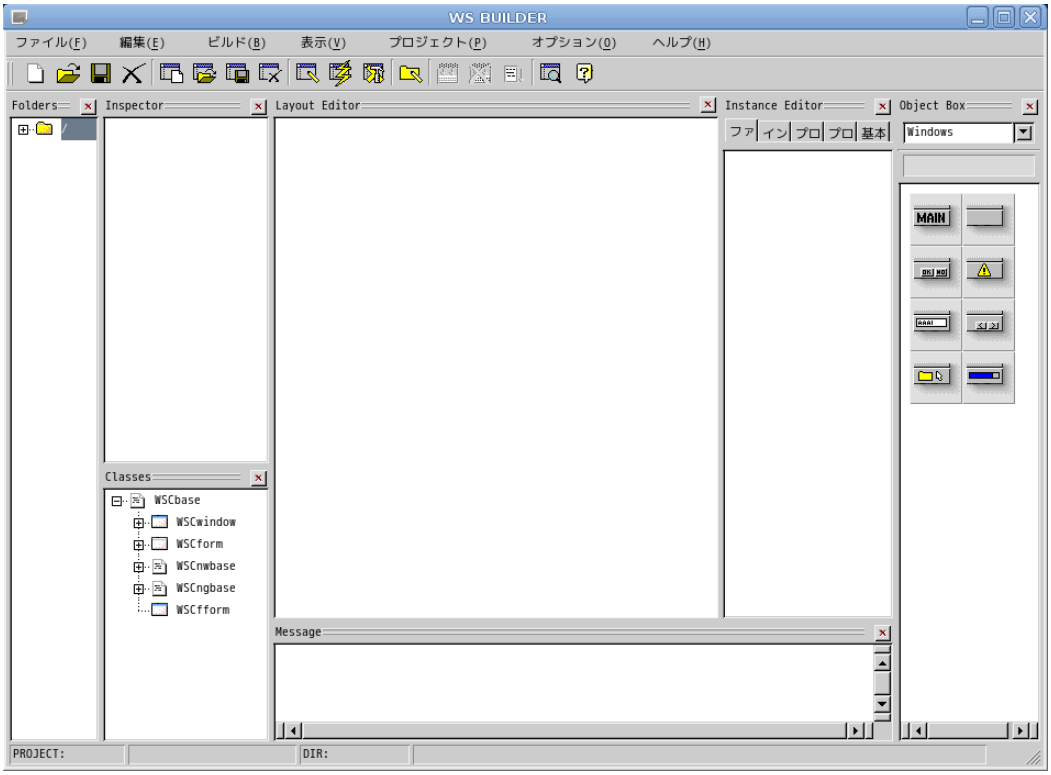


図 2-3-1-1. WideStudio 起動

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。

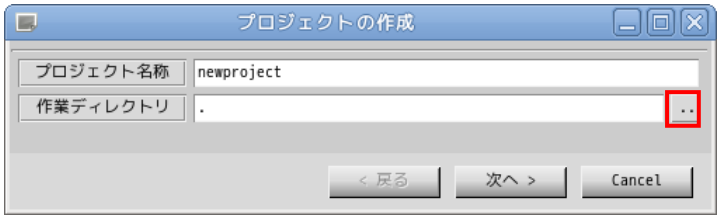


図 2-3-2-2.　プロジェクト作成画面

 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。

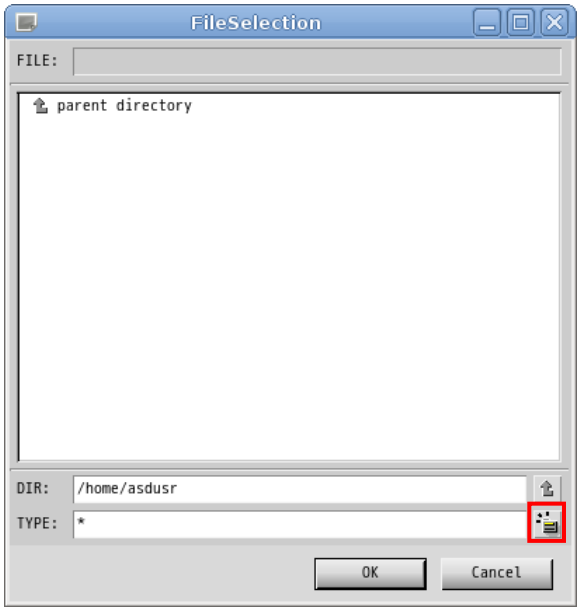


図 2-3-2-3.　ファイル選択画面

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：入力はマウスカーソ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。

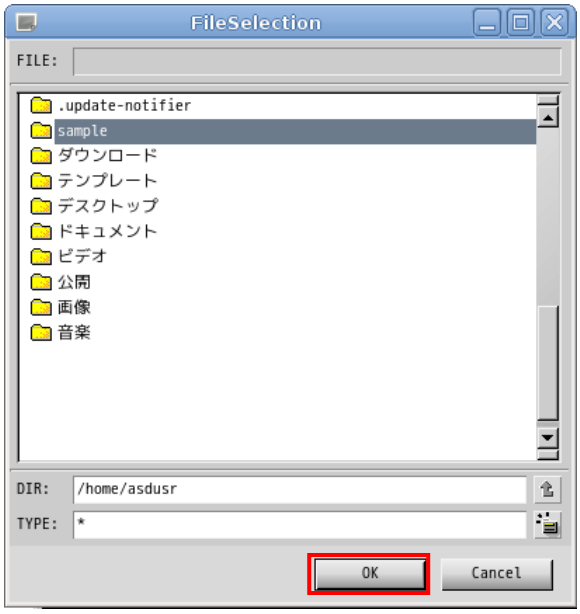


図 2-3-2-4. sample ディレクトリの作成

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。

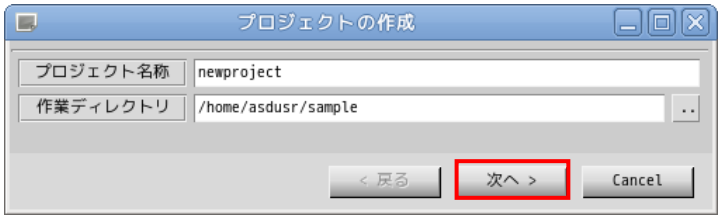


図 2-3-2-5. プロジェクト名と作業ディレクトリの設定完了

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。

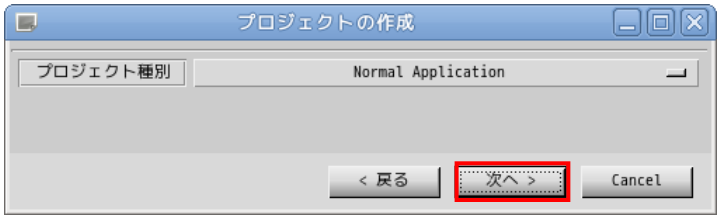


図 2-3-2-6. プロジェクト種別の選択

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。

Algonomix6 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。

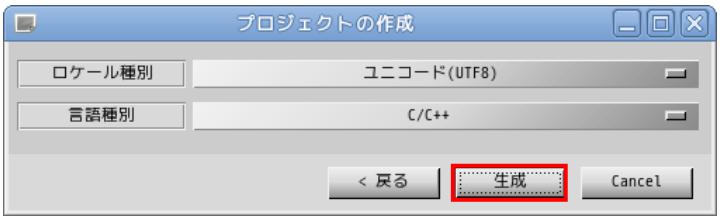


図 2-3-2-7.　ロケール種別と言語種別を選択

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。

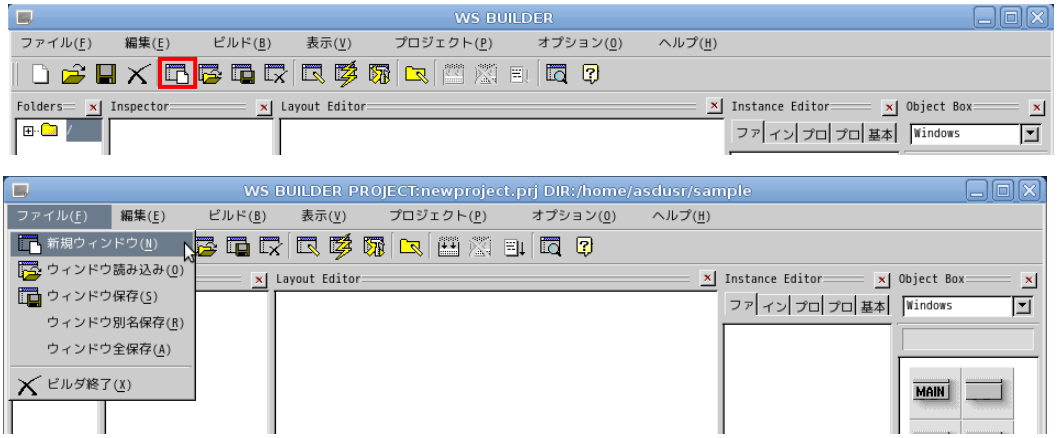


図 2-3-3-1.　新規ウィンドウ作成

 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。

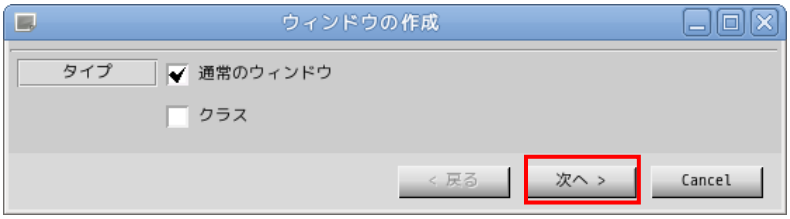


図 2-3-3-2.　アプリケーションウィンドウタイプ選択

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。

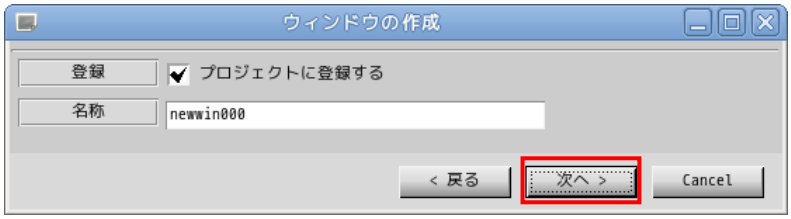


図 2-3-3-3. アプリケーションウィンドウ名称設定

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。

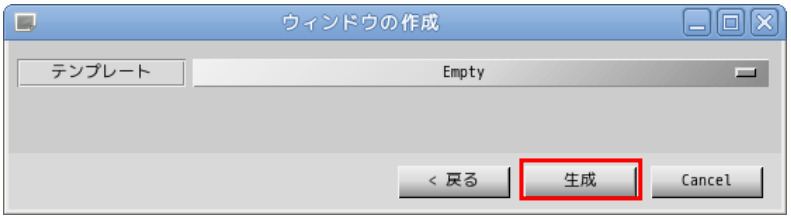


図 2-3-3-4. テンプレートの選択

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。

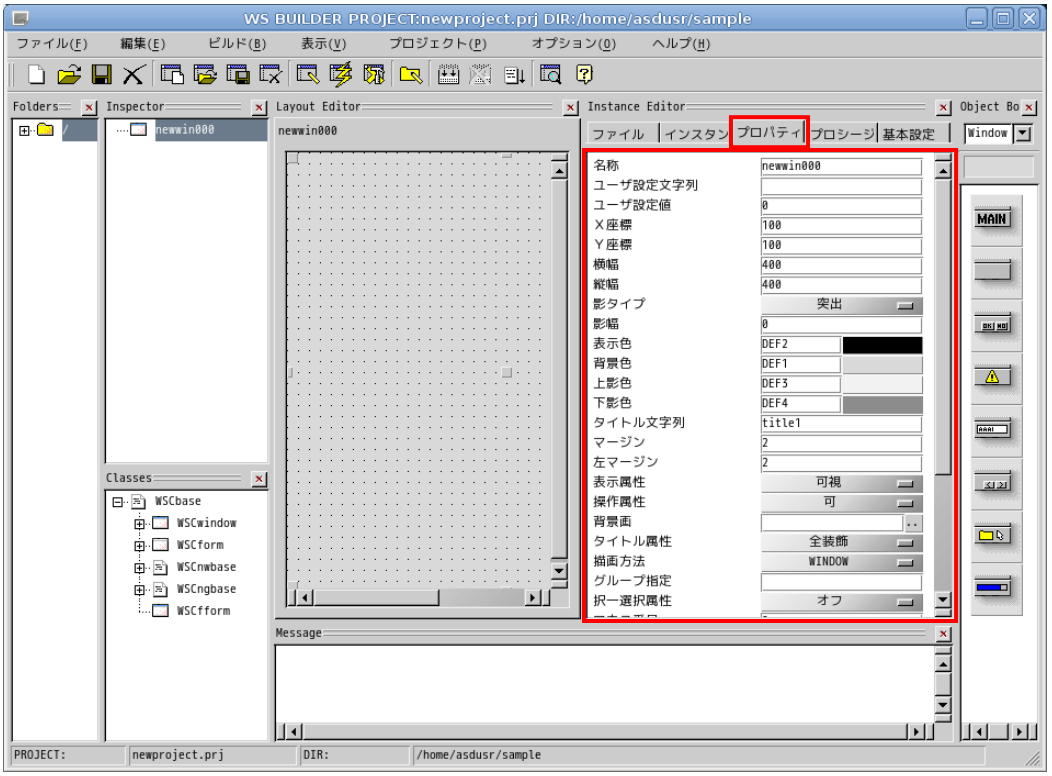


図 2-3-3-5. アプリケーションウィンドウの生成

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。

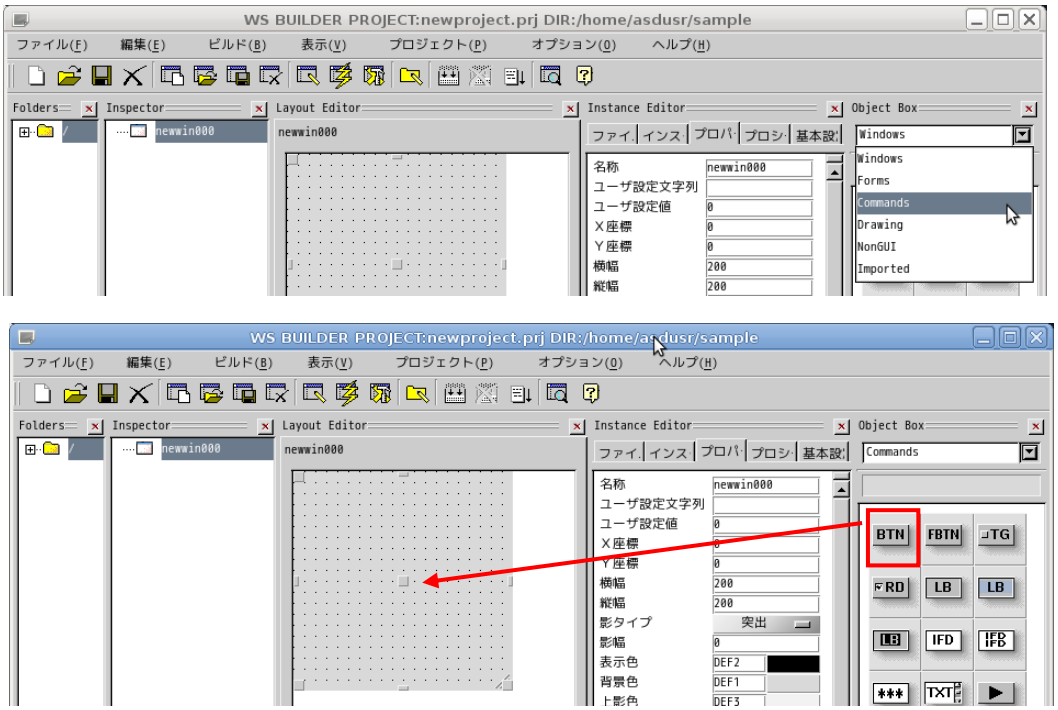


図 2-3-4-1. ボタンの配置

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。

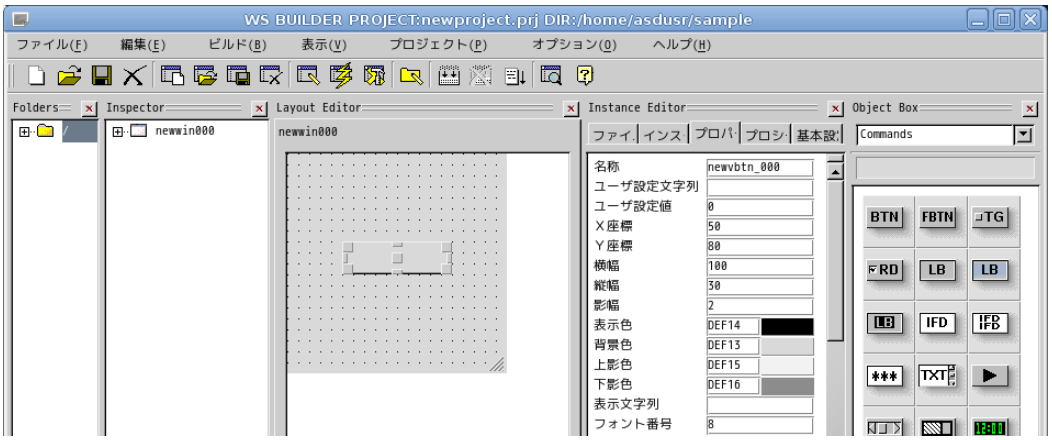


図 2-3-4-2. アプリケーションウィンドウに配置されたボタン

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：Algonomix6 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。

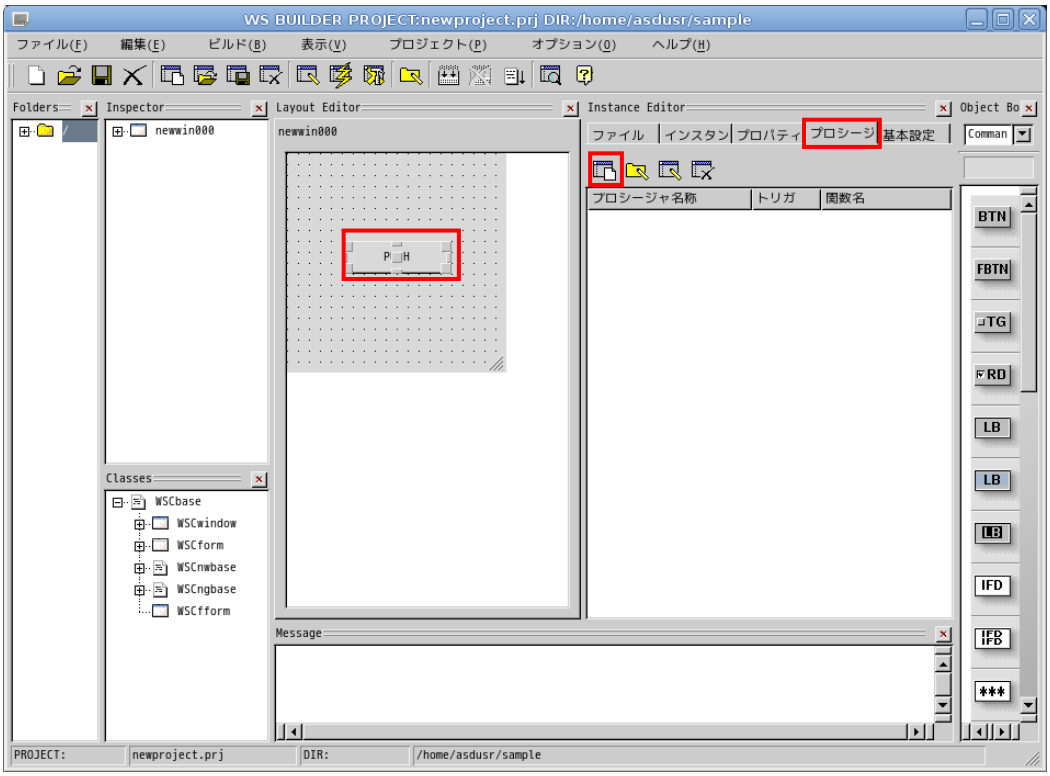


図 2-3-5-1. イベントプロシージャの作成

 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、離されたときに発生するイベントとして「ACTIVATE」を選択します。

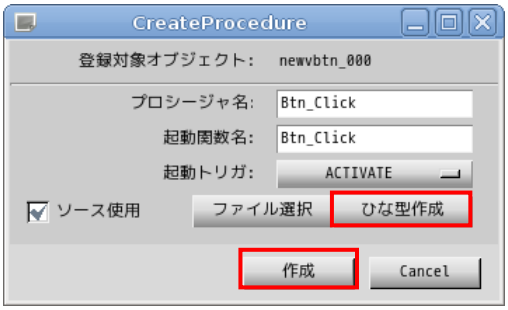


図 2-3-5-2. イベントプロシージャ作成ダイアログ

「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。

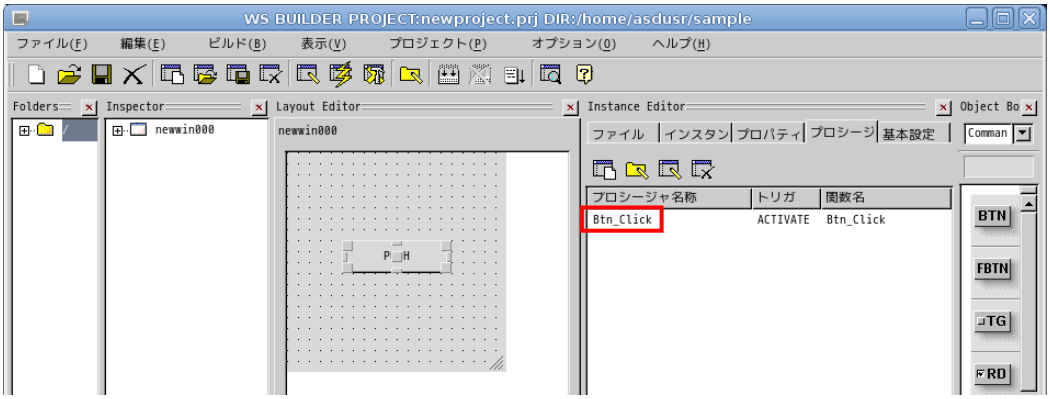


図 2-3-5-3. 作成されたイベントプロシージャ

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。

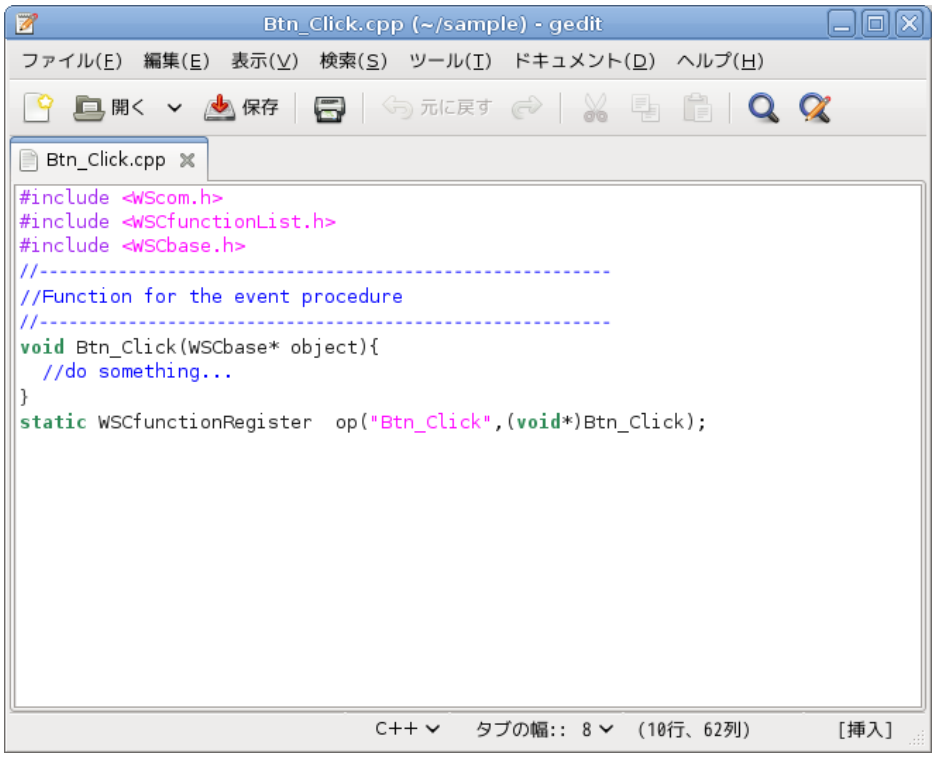


図 2-3-6-1. イベントプロシージャ初期ソースコード

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。

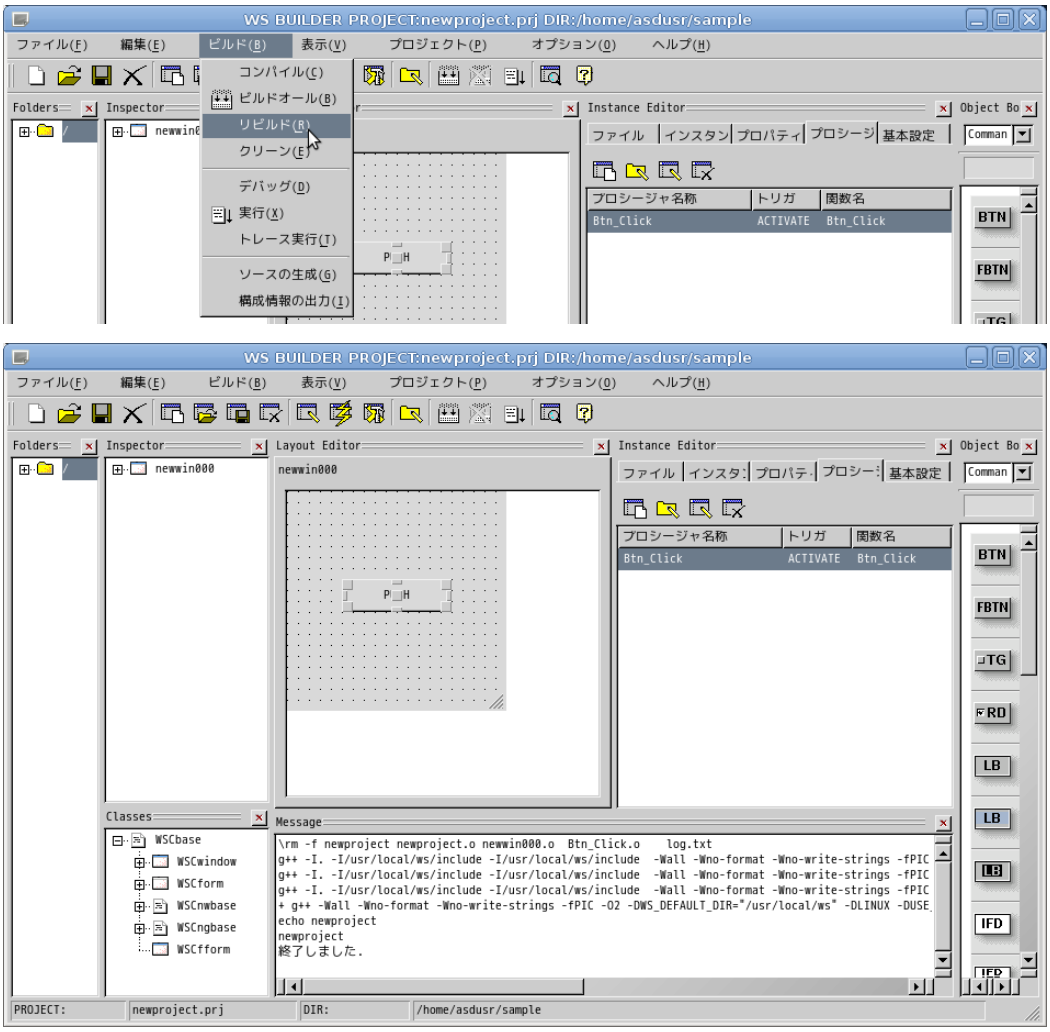


図 2-3-7-1. サンプルプロジェクトのコンパイル

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。

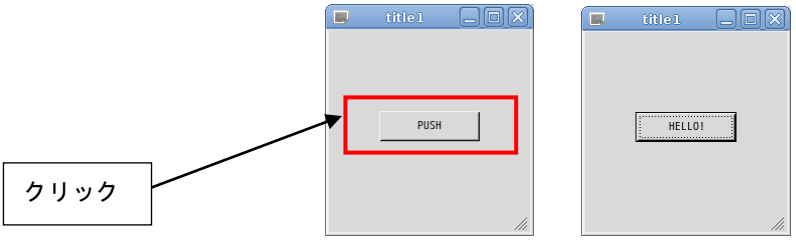


図 2-3-7-2. プログラム実行画面

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イ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。

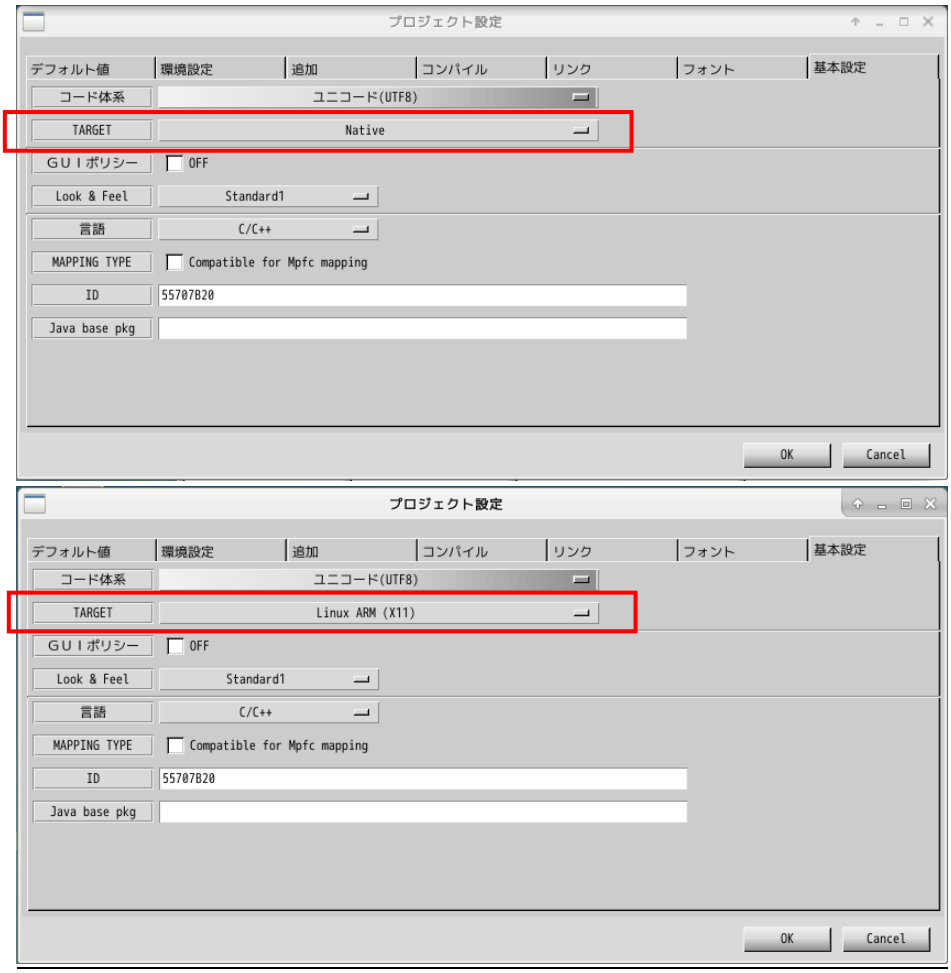


図 2-3-7-3. プロジェクト設定（基本設定）

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、Qualcomm 製 SnapDragon 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、ftp 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 ftp 転送

① LAN ケーブルをアルゴシステム製端末に接続します。
アルゴシステム製端末と開発環境を HUB 無しで接続する場合、クロスケーブルが必要です。

シリーズ側の OS である Algonomix6 のネットワークを設定するには、ASD Network Config を用います。各端末のマニュアルを参考にして設定してください。また、接続する Algonomix6 の IP アドレスを確認しておいてください。

※注：アルゴシステム製端末と開発環境の 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 のデスクトップ上の「アプリケーション」ツールバーから「インターネット」→「gFTP」を選択することで「gFTP」という FTP クライアントが起動されます。

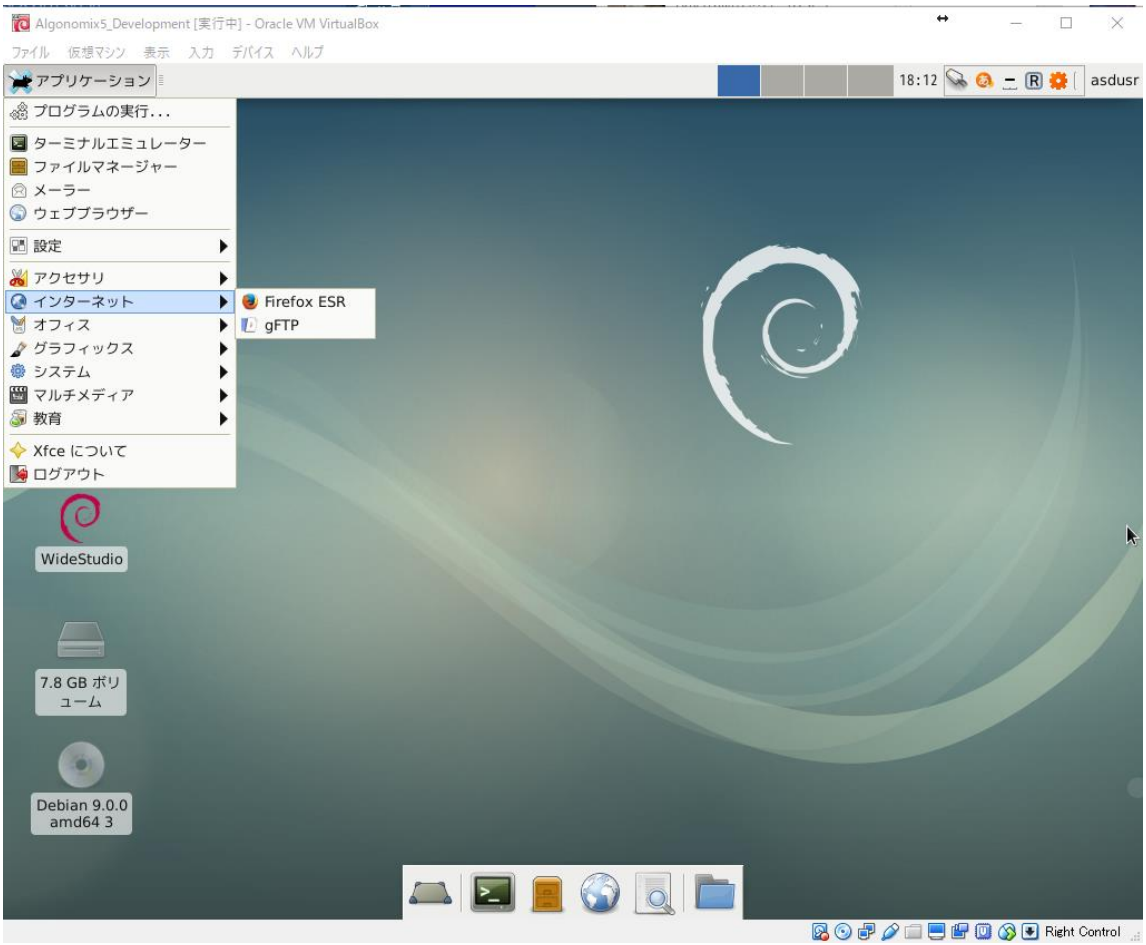


図 2-3-8-1. FTP クライアントの起動

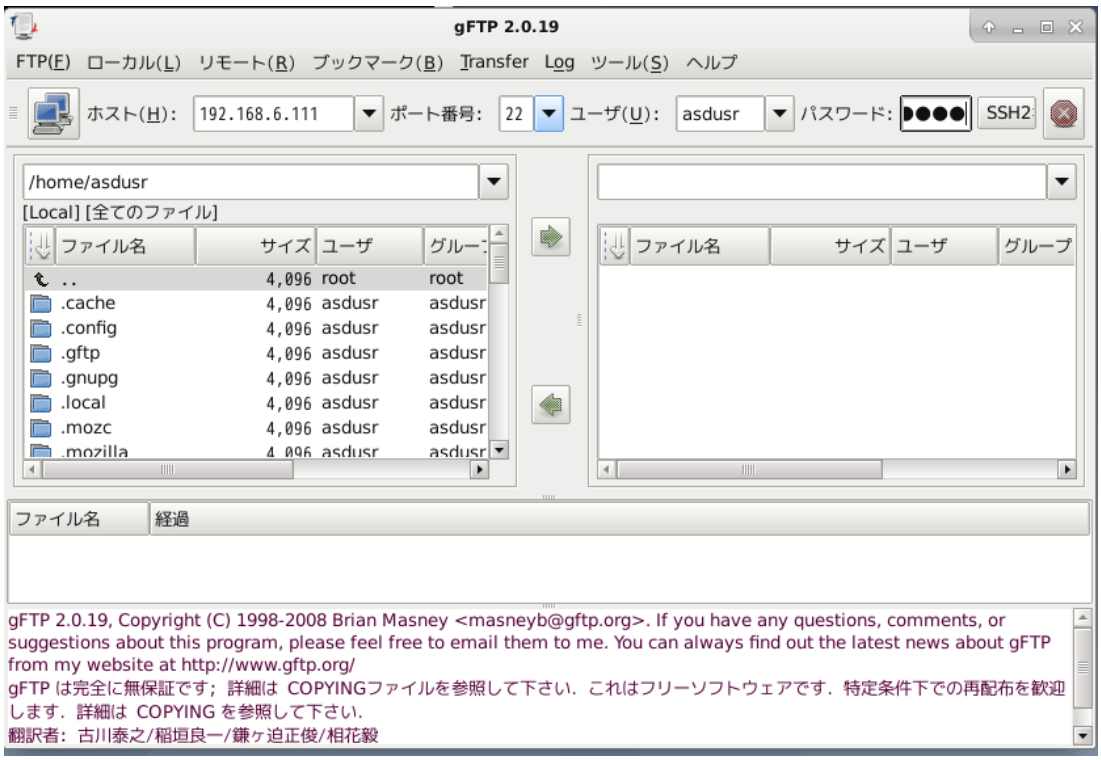


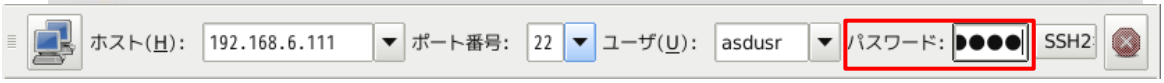
図 2-3-8-2. FTP クライアント画面

① 接続先であるアルゴシステム製端末の IP アドレスを指定します。

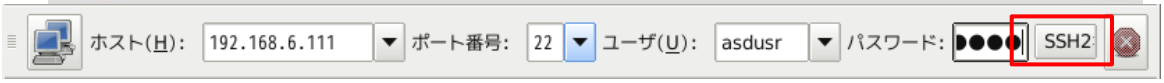
② ポート番号は「22」を設定します。

③ アルゴシステム製端末への FTP 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 アルゴシステム製端末への 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。



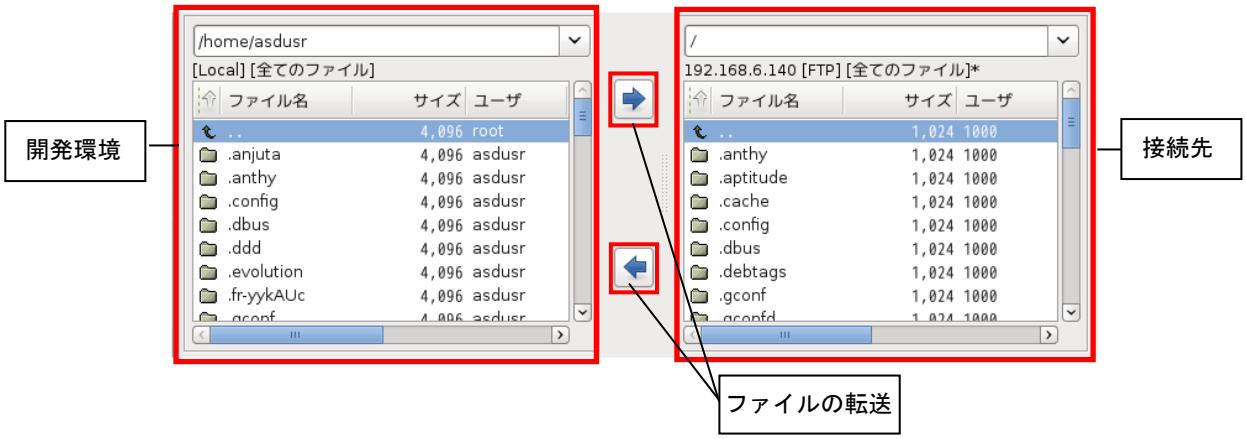
⑤ 通信タイプを指定します。「SSH2」としてください。



⑥ Algonomix7 上で起動している FTP サーバに対して、接続と切断を行います。

正常に接続されると、Algonomix6 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。Algonomix6 の FTP サーバ（vsftpd）のデフォルト設定ではセキュリティ上「/home/asdusr」ディレクトリ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 Algonomix6 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いてきます。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
Algonomix6 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/wsfonts 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：Algonomix6 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違う可能性があります。

●X Window System の場合
Algonomix6 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 1:太字  slant 0:無し 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。
　Algonomix6 に標準実装されているフォントは「東風ゴシック」のみです。
　WideStudio Ver3.98-7 時点ではボールド設定（Weight）とイタリック設定（slant）は使用することが出来ません。

　Algonomix6 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

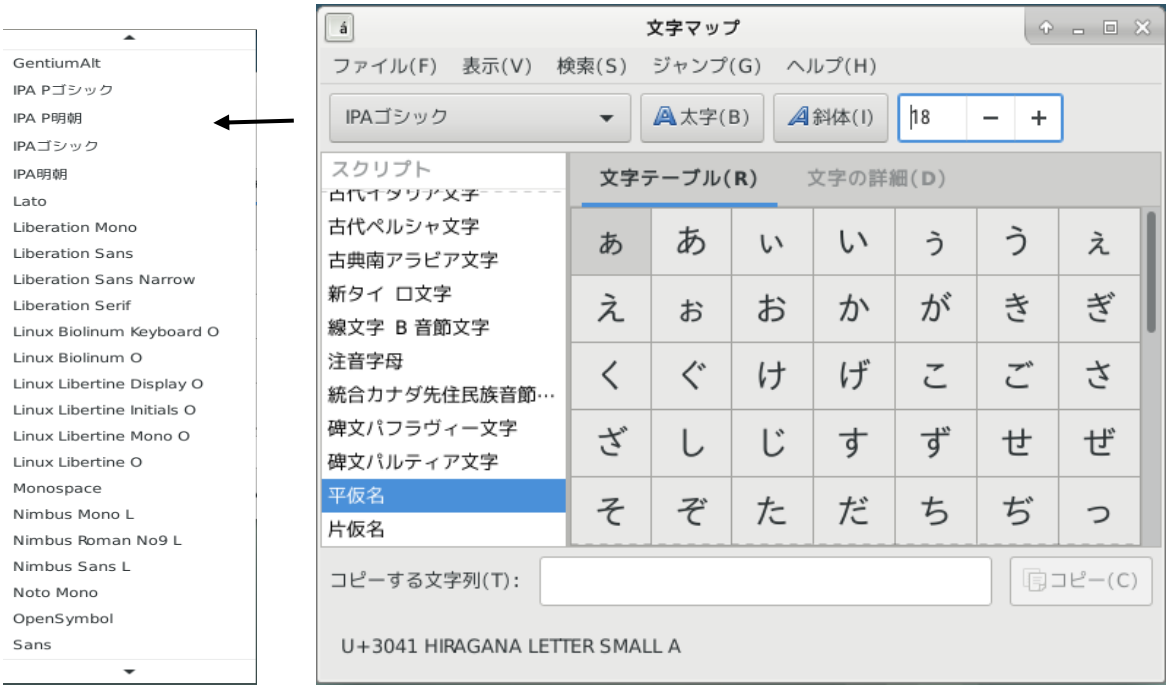


図 2-3-9-1. 文字マップツール

２－４　　Qt によるアプリケーション開発

２－４－１　Qt とは

Qt「キュート」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 スタートメニュー→「開発」→「Qt Creator (Community)」をクリックします。図 2-4-3-1 のような画面が起動します。
② 「＋新しいプロジェクト」をクリックします。図 2-4-4-1 のような画面が開きます。

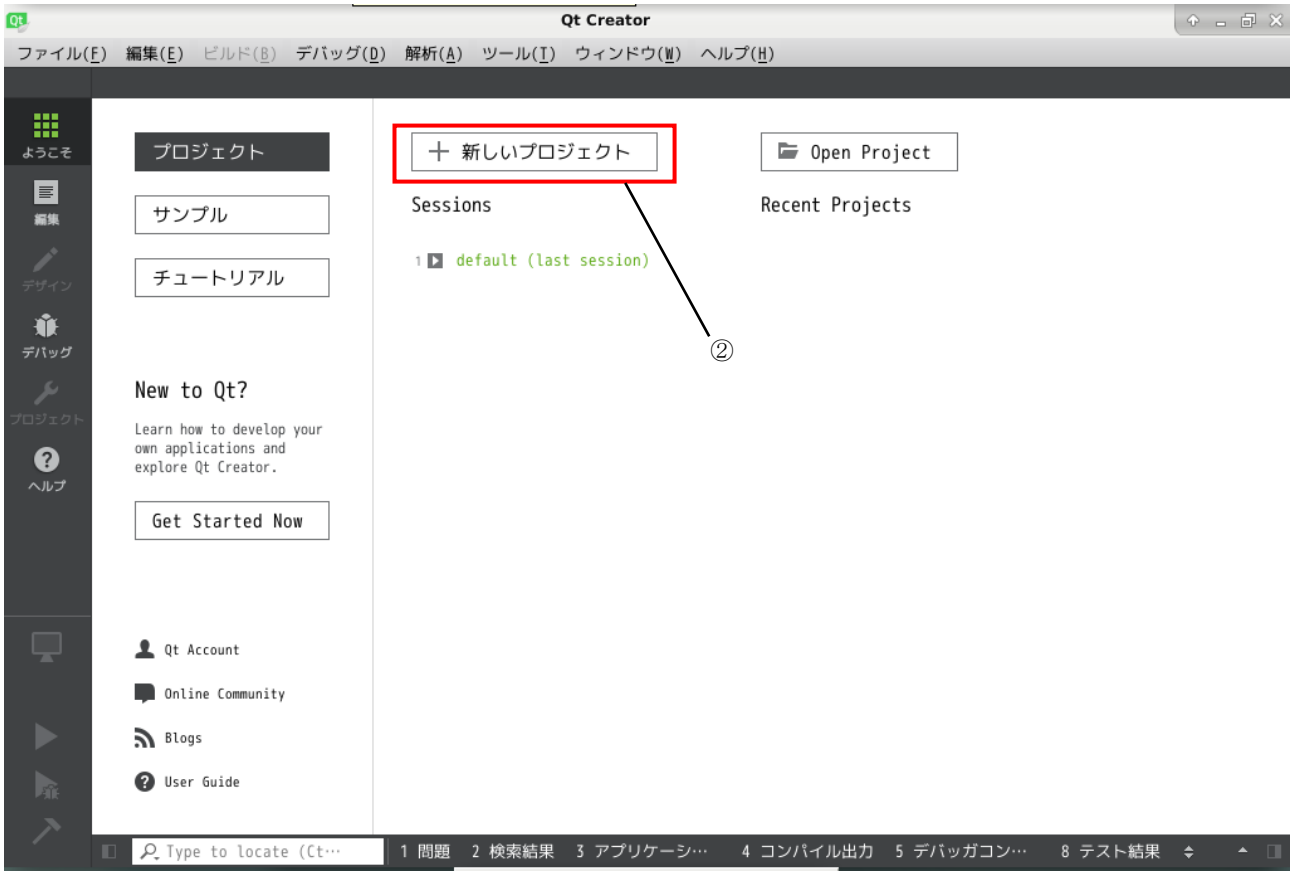


図 2-4-3-1. Qt Creator 起動画面

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。

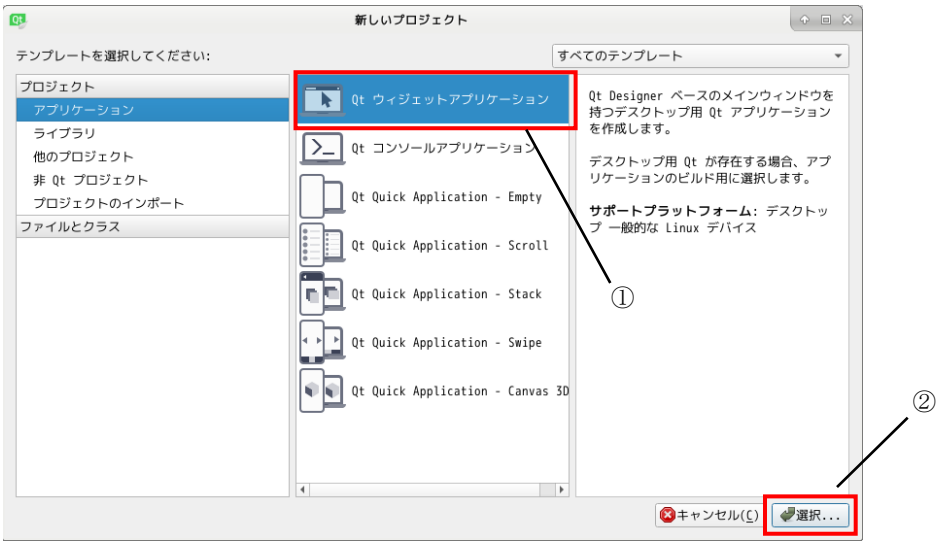


図 2-4-4-1. 新しいプロジェクト選択画面

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)	SnapDragon 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。

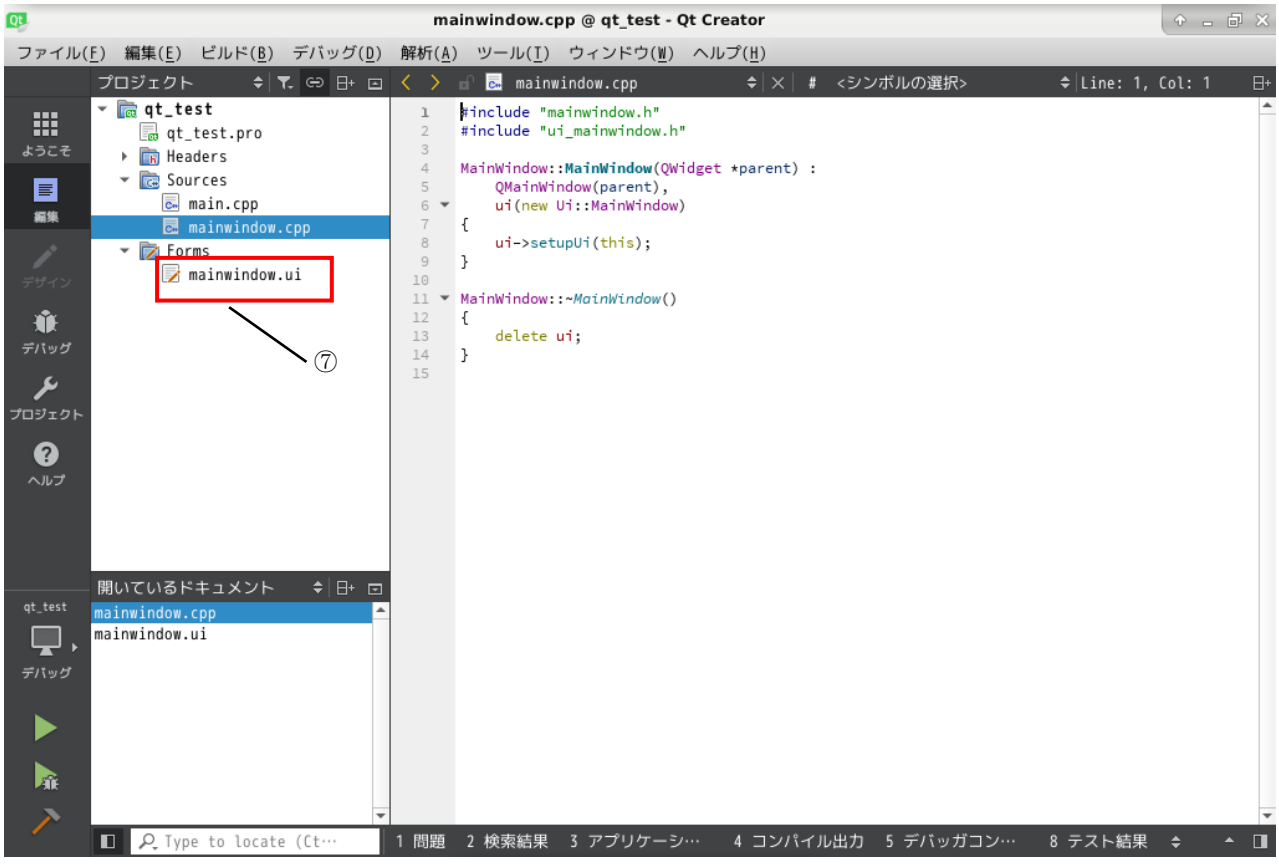


図 2-4-4-6.　コード編集画面

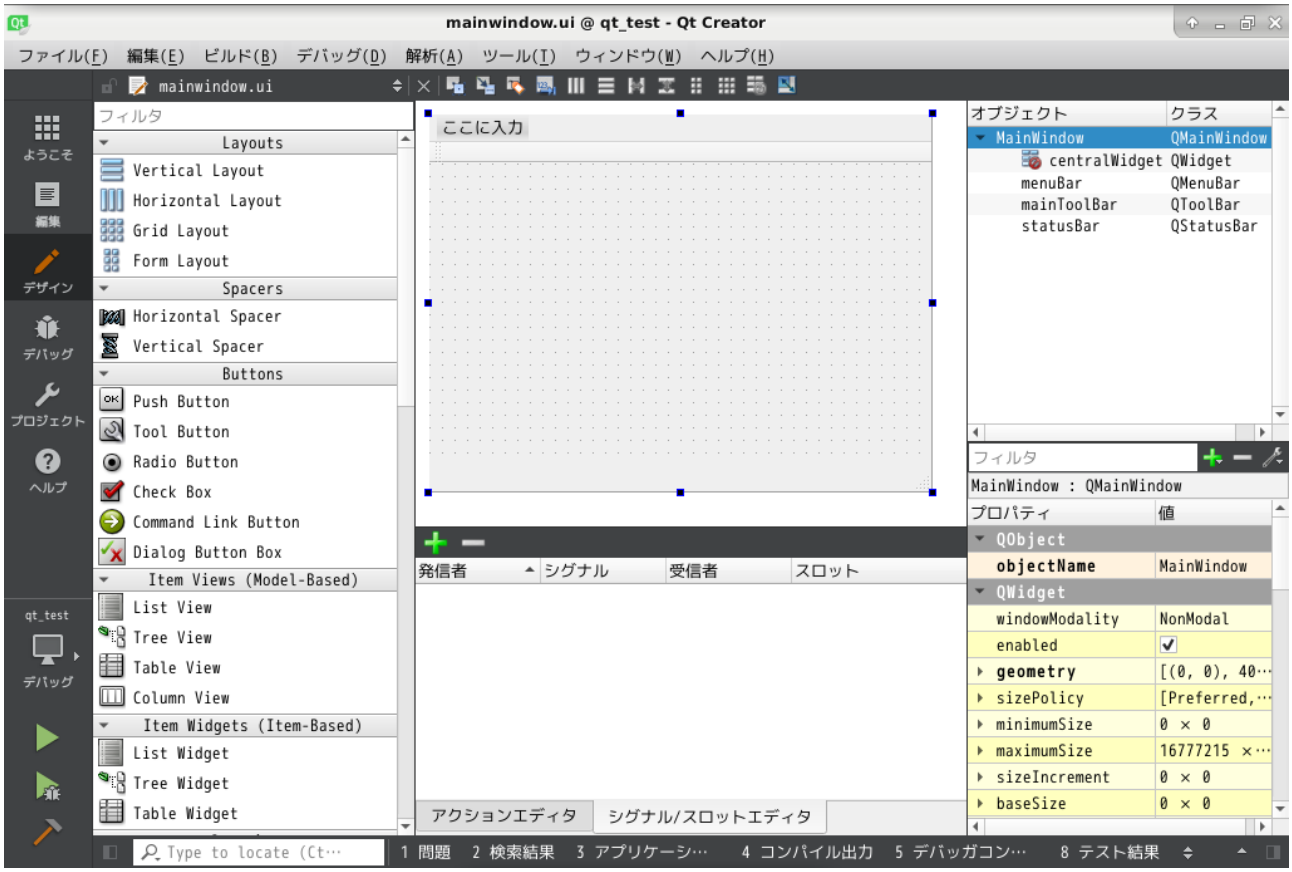


図 2-4-4-7. フォーム画面編集画面

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、ウインドウにドロップして配置します。

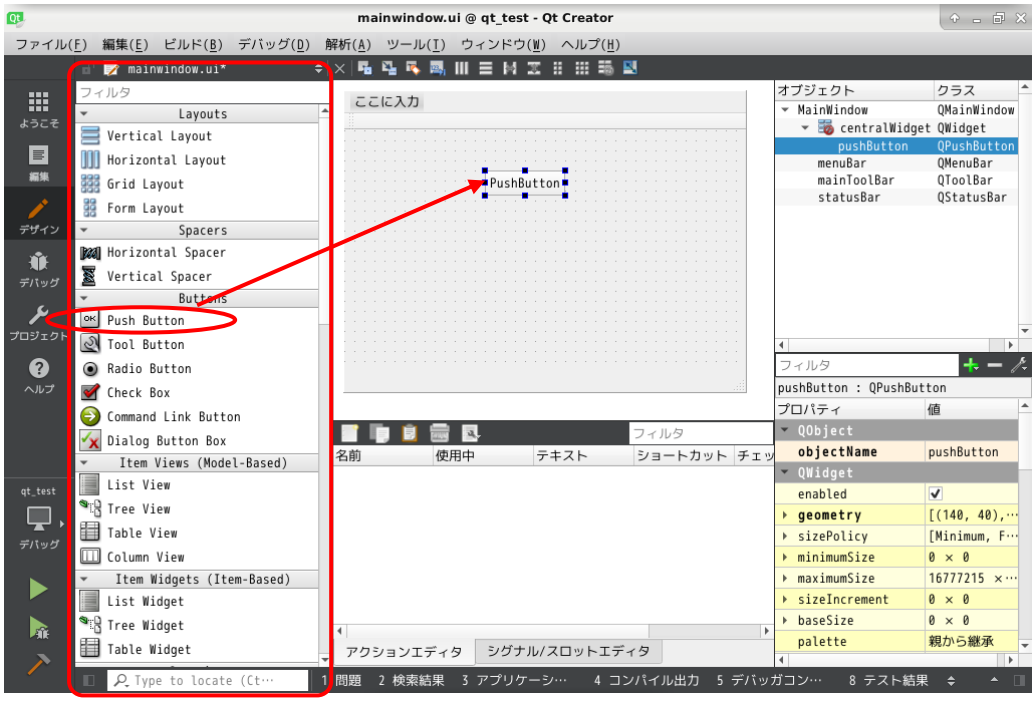


図 2-4-5-1. 部品の配置

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。

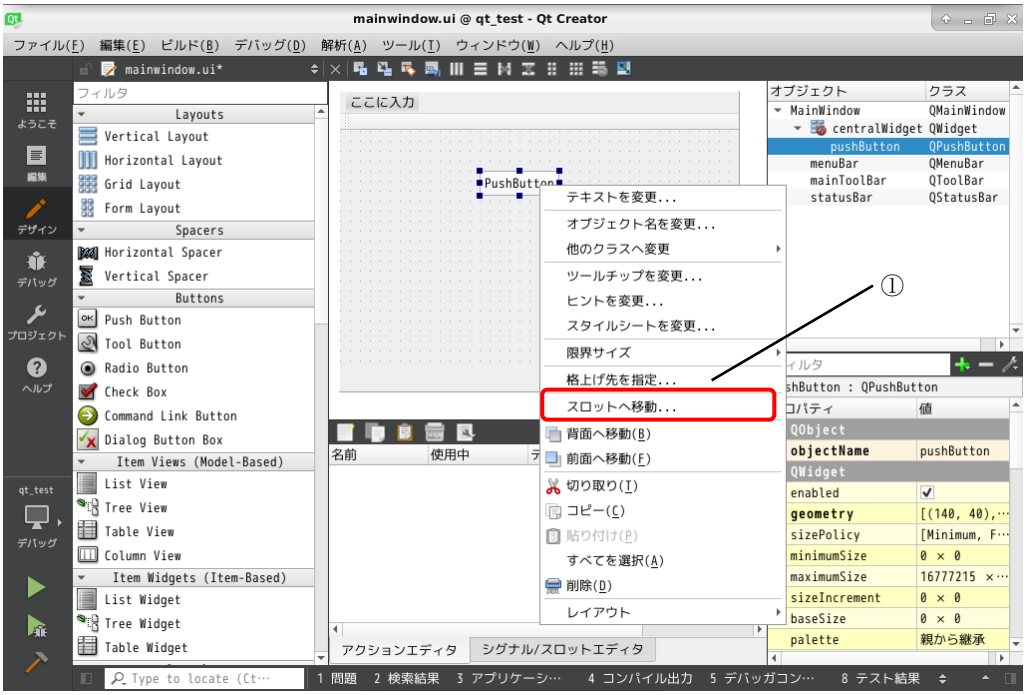


図 2-4-6-1. 部品右クリックメニュー

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

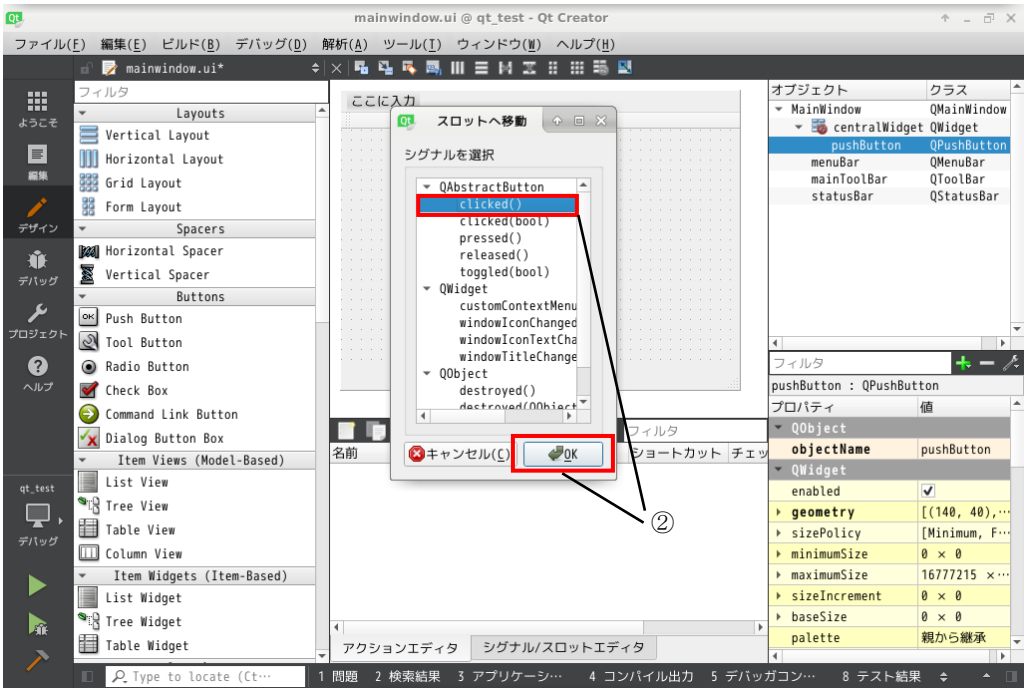
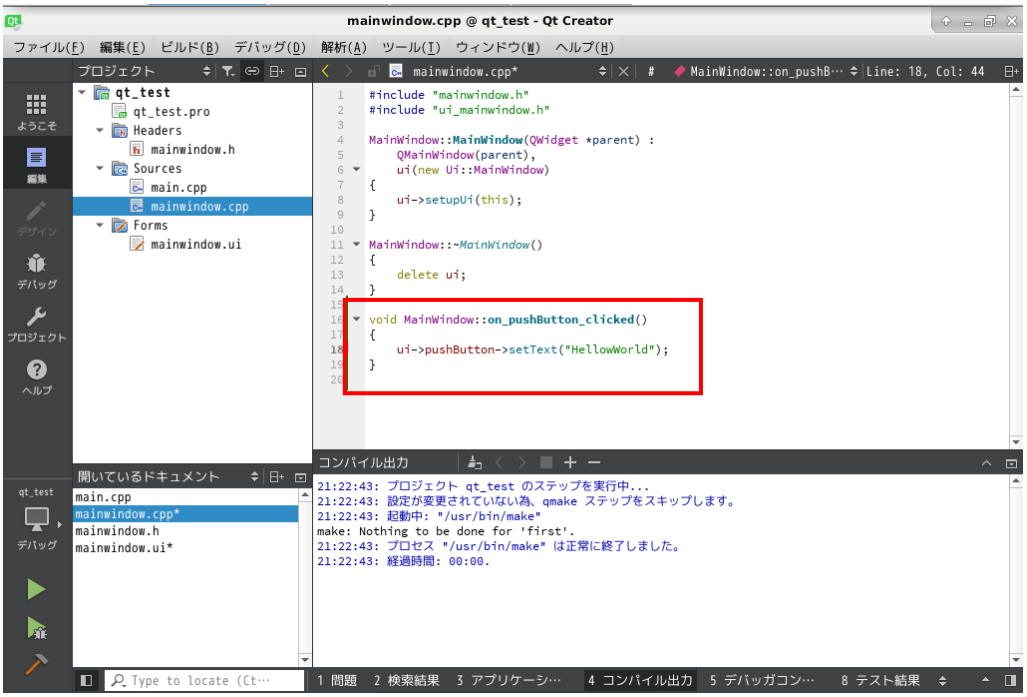


図 2-4-6-2. スロットの選択

③ mainwindows.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。関数の中身を図のように記述します。



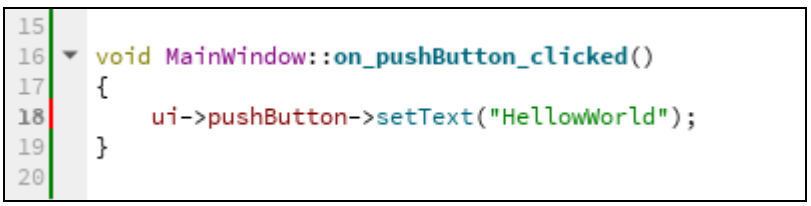


図 2-4-6-3. Click イベント処理追加

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 2-5-7-2 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。

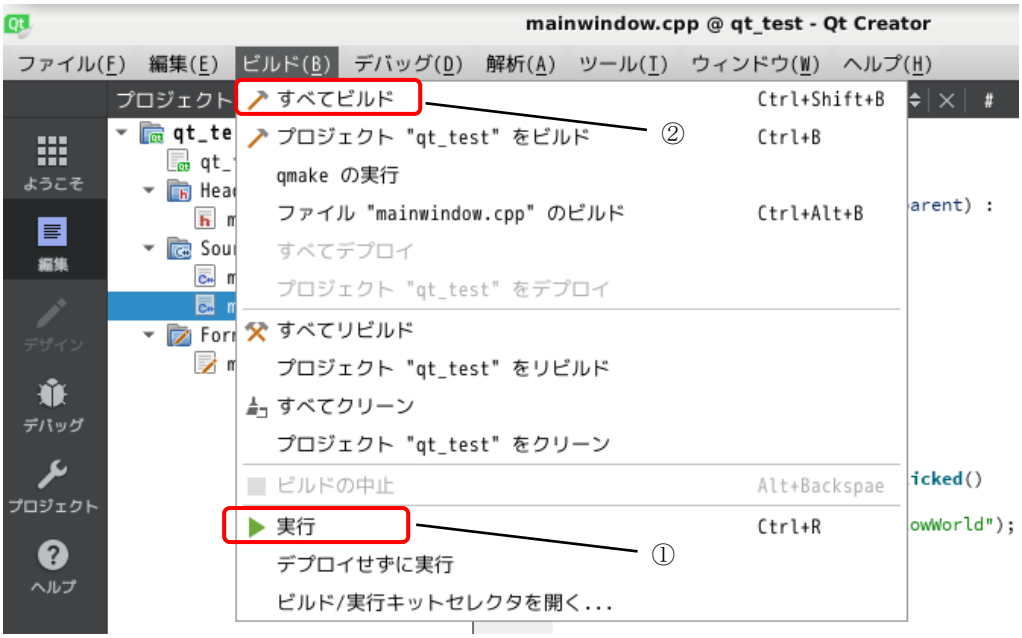


図 2-4-7-1. ビルドメニュー

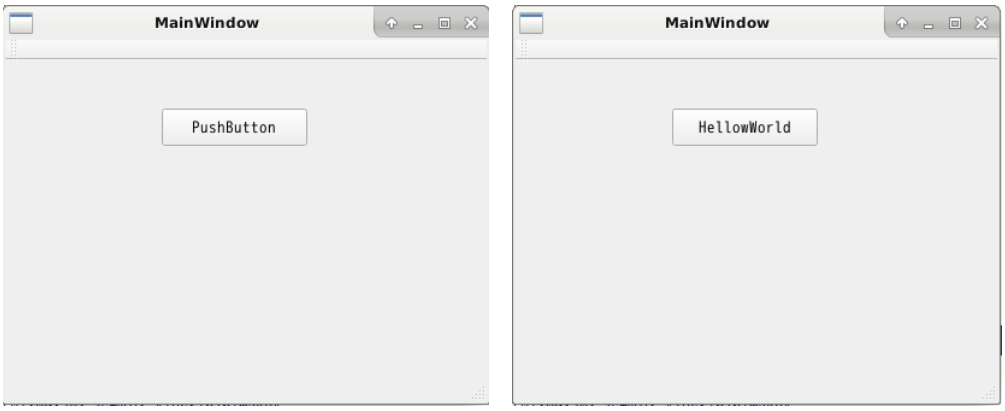


図 2-4-7-2. 実行画面

２－４－８　SnapDragon CPU 環境でビルド

① SnapDragon CPU 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。

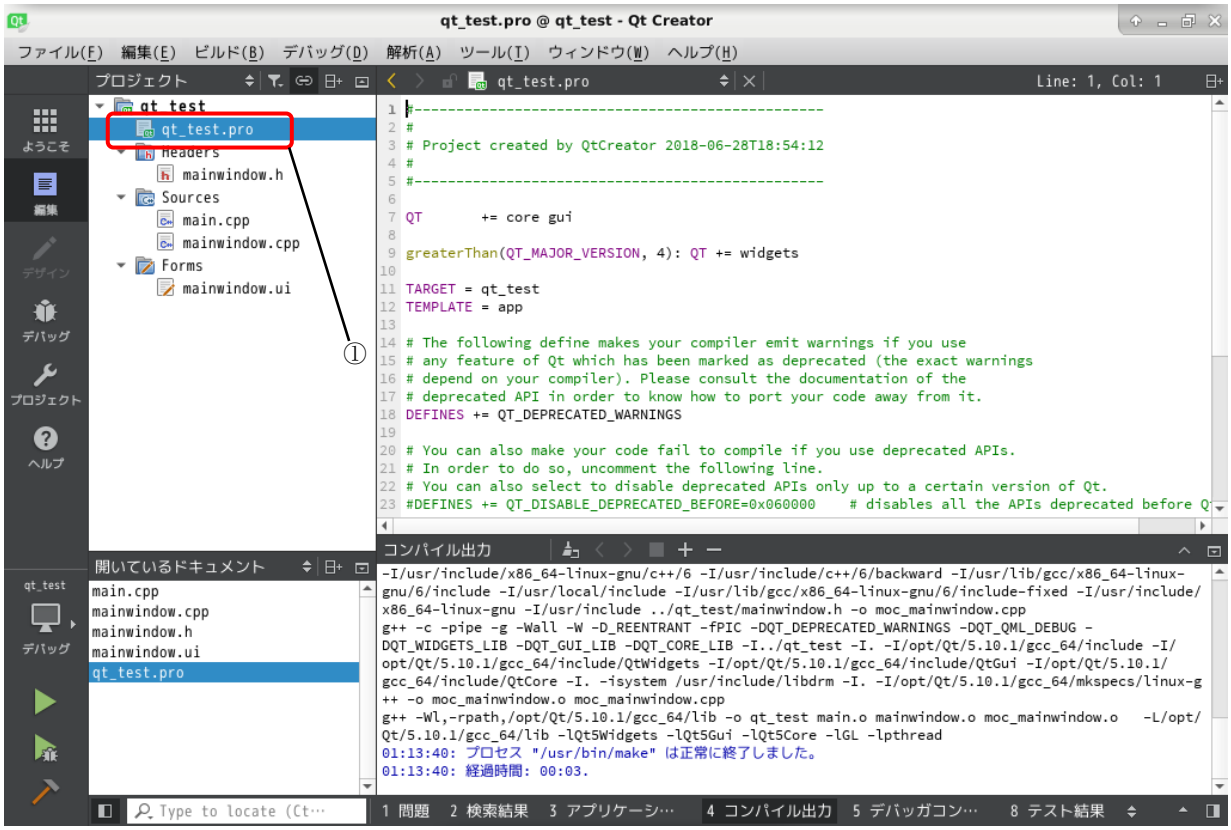


図 2-4-8-1.　プロジェクトファイル初期設定

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。

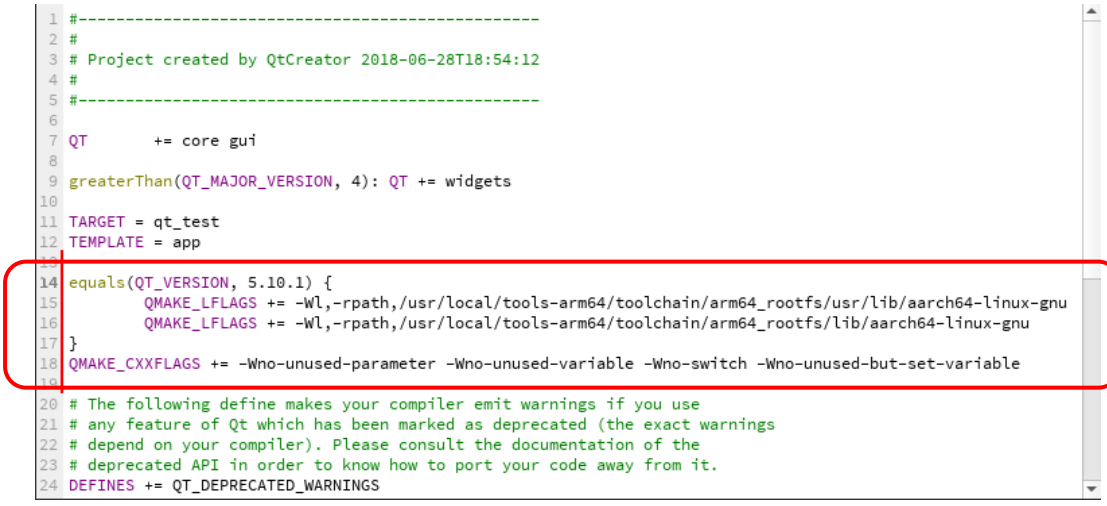


図 2-4-8-2.　プロジェクトファイルコンパイルオプション追加

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。

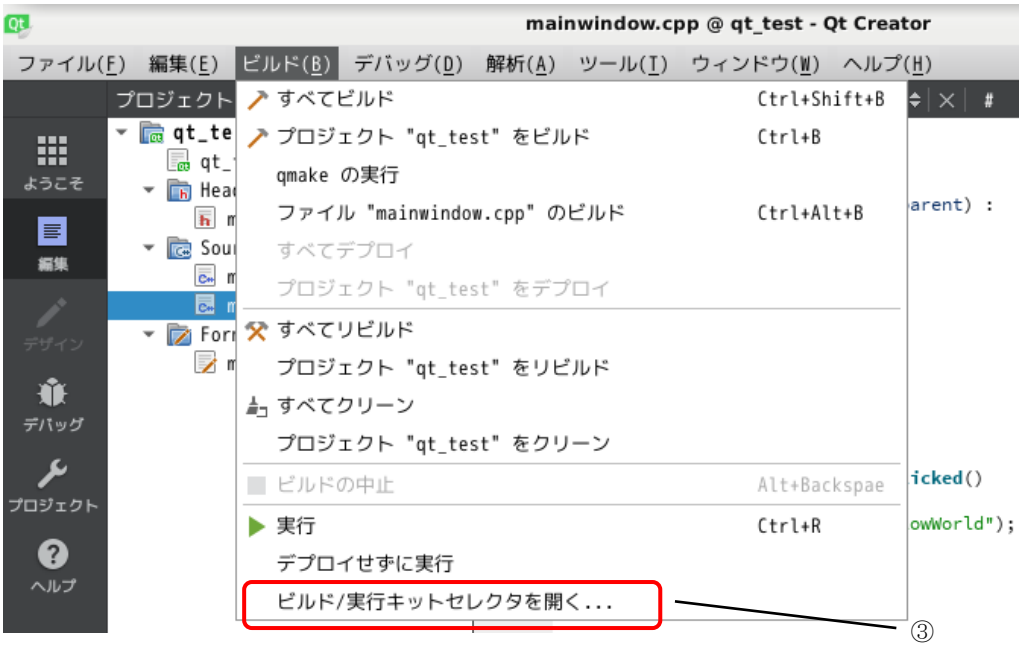


図 2-4-8-3.　ビルドメニュー

④ ビルドキットを選択します。

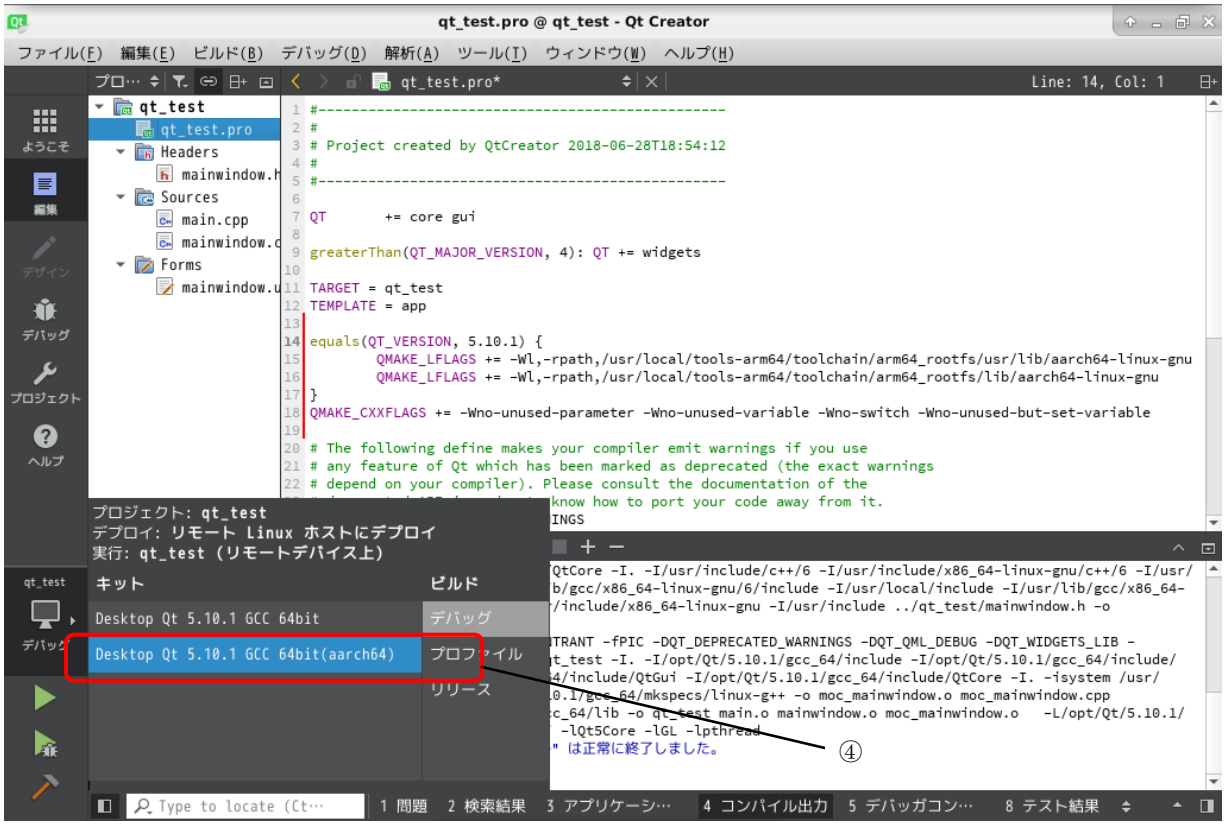


図 2-4-8-4.　ビルドキット選択

⑤「ビルド」→「すべてビルド」を実行します。

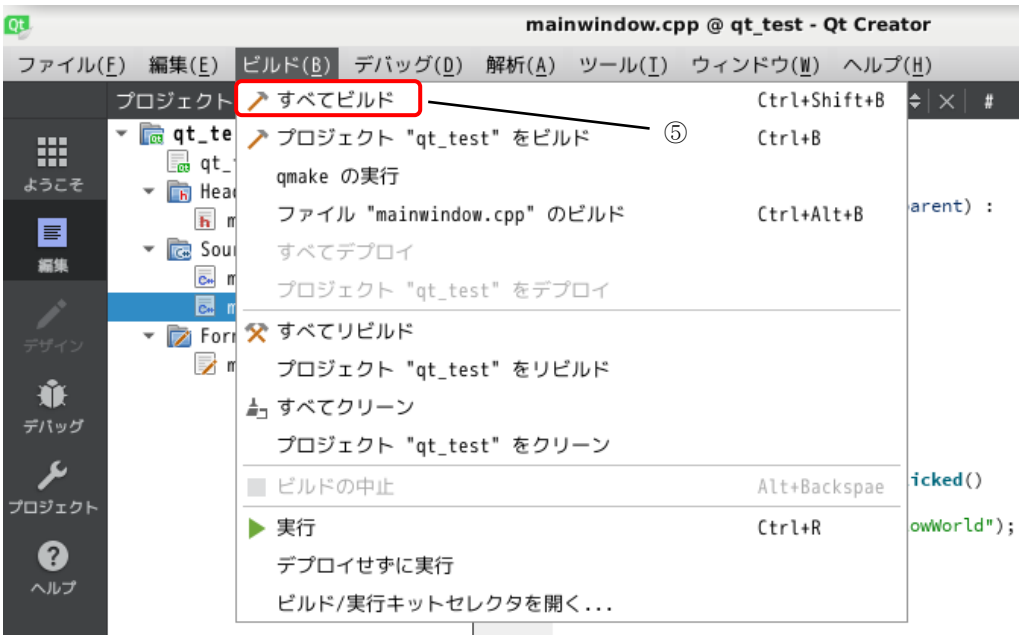


図 2-4-8-5. ビルドメニュー

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。

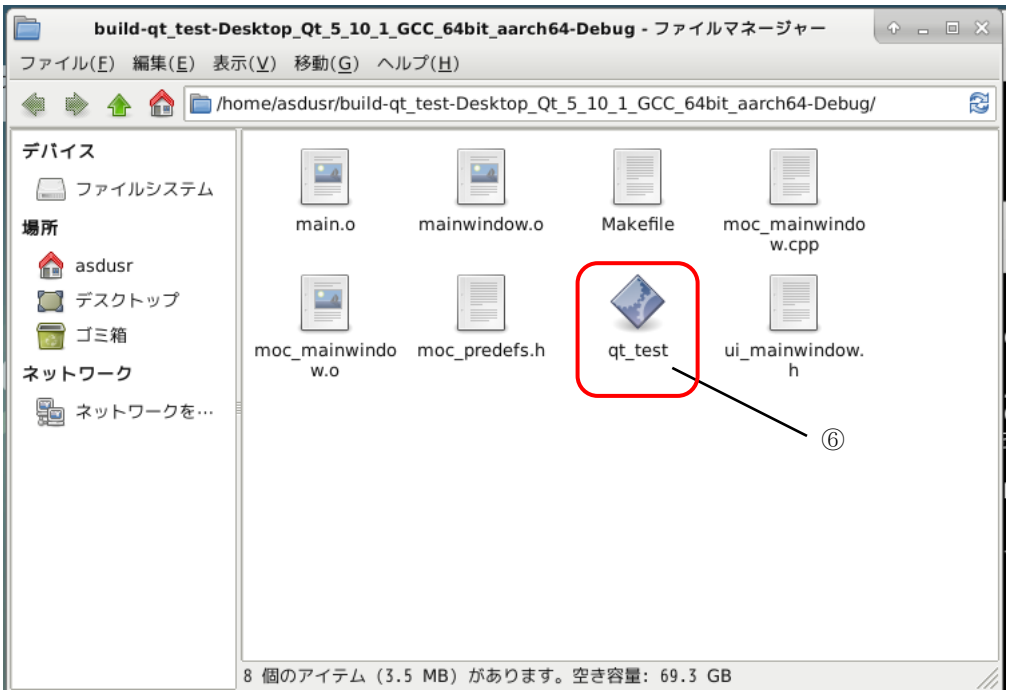


図 2-4-8-6. SnapDragon 用実行ファイル格納ディレクトリ

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。

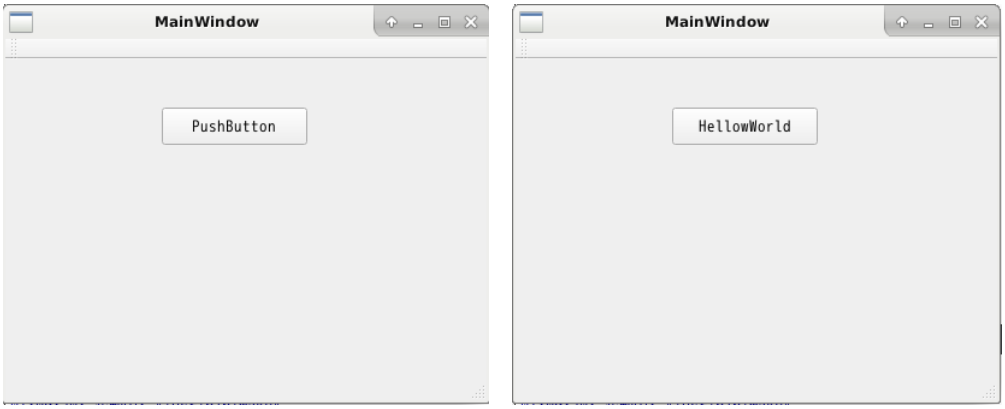


図 2-4-8-7. 実行画面

２－４－９　リモートデバッグ方法

SnapDragon CPU 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―ル」→「オプション」をクリックします。

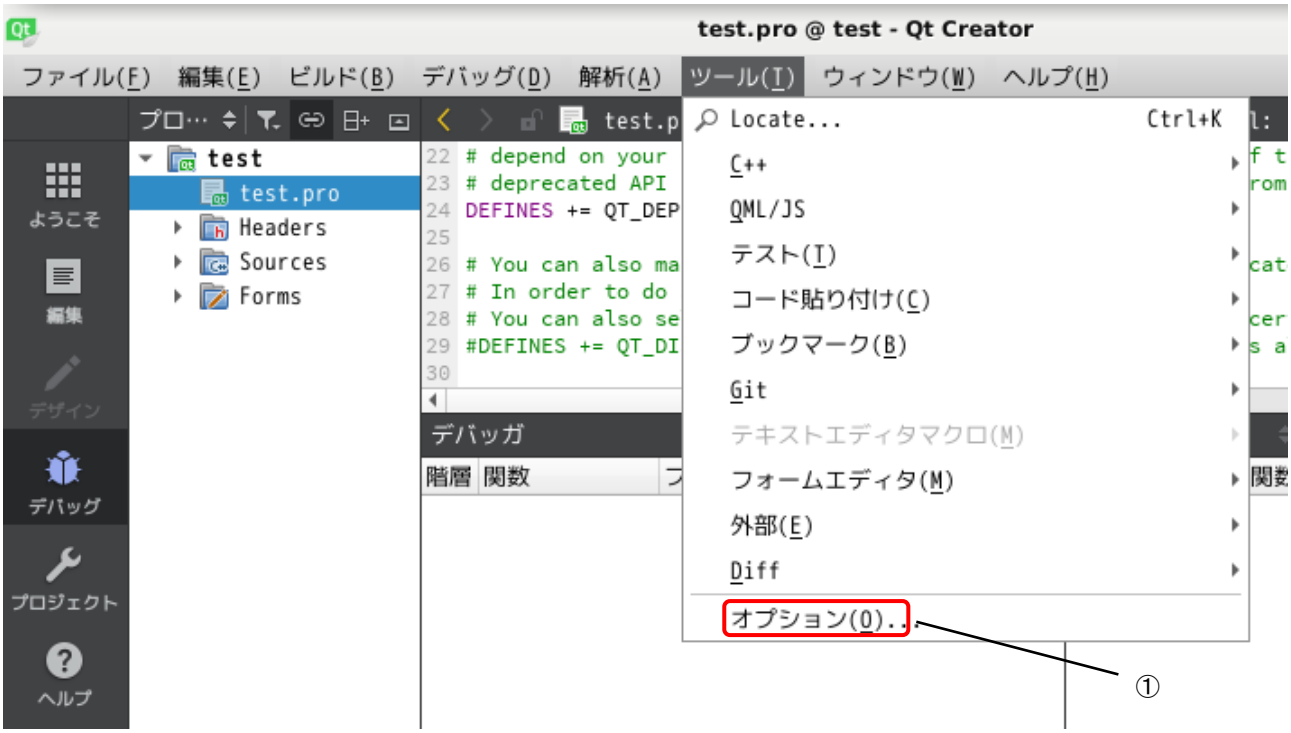


図 2-4-9-1. ツールメニュー

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、「asdusr」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。

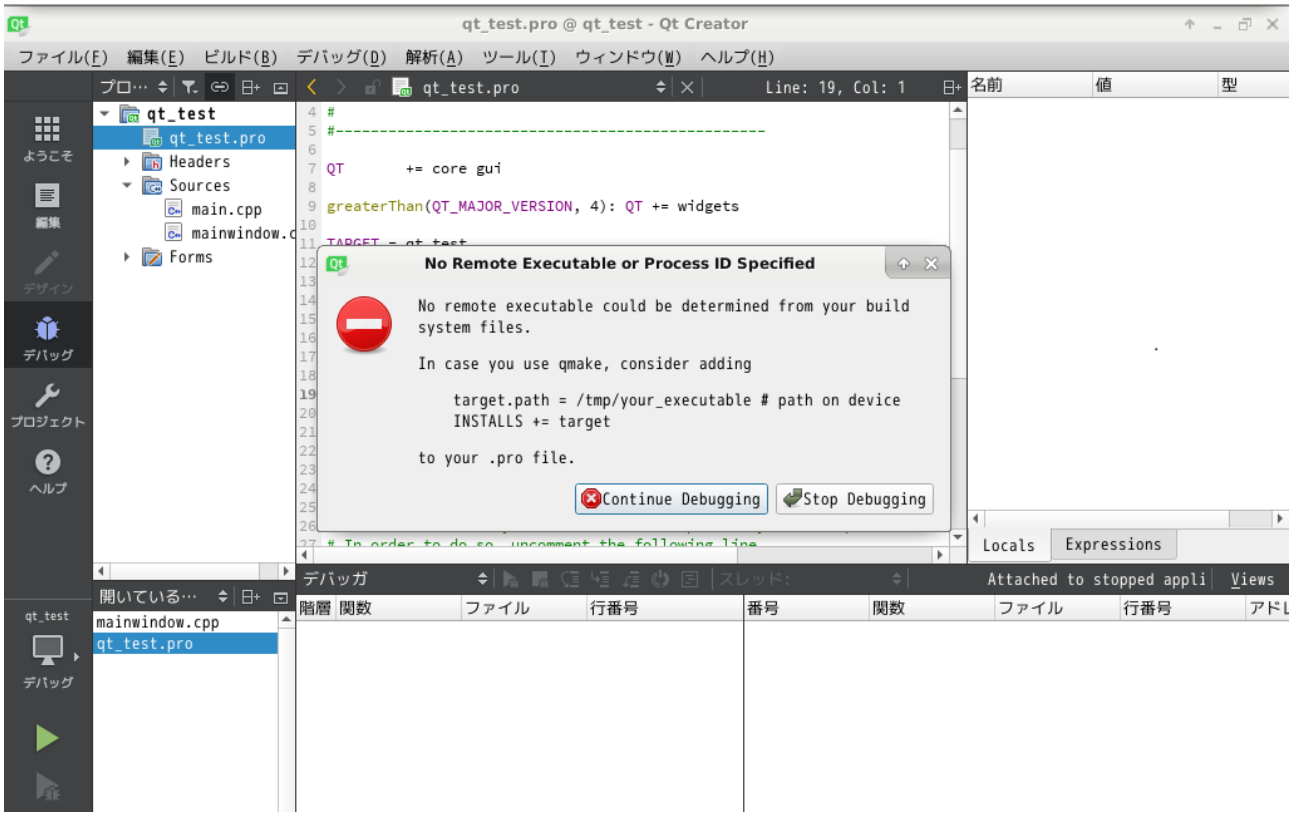


図 2-4-9-3. デバッグ実行時に出力されるメッセージ

② 図 2-4-9-4 の.pro ファイルを開きます。

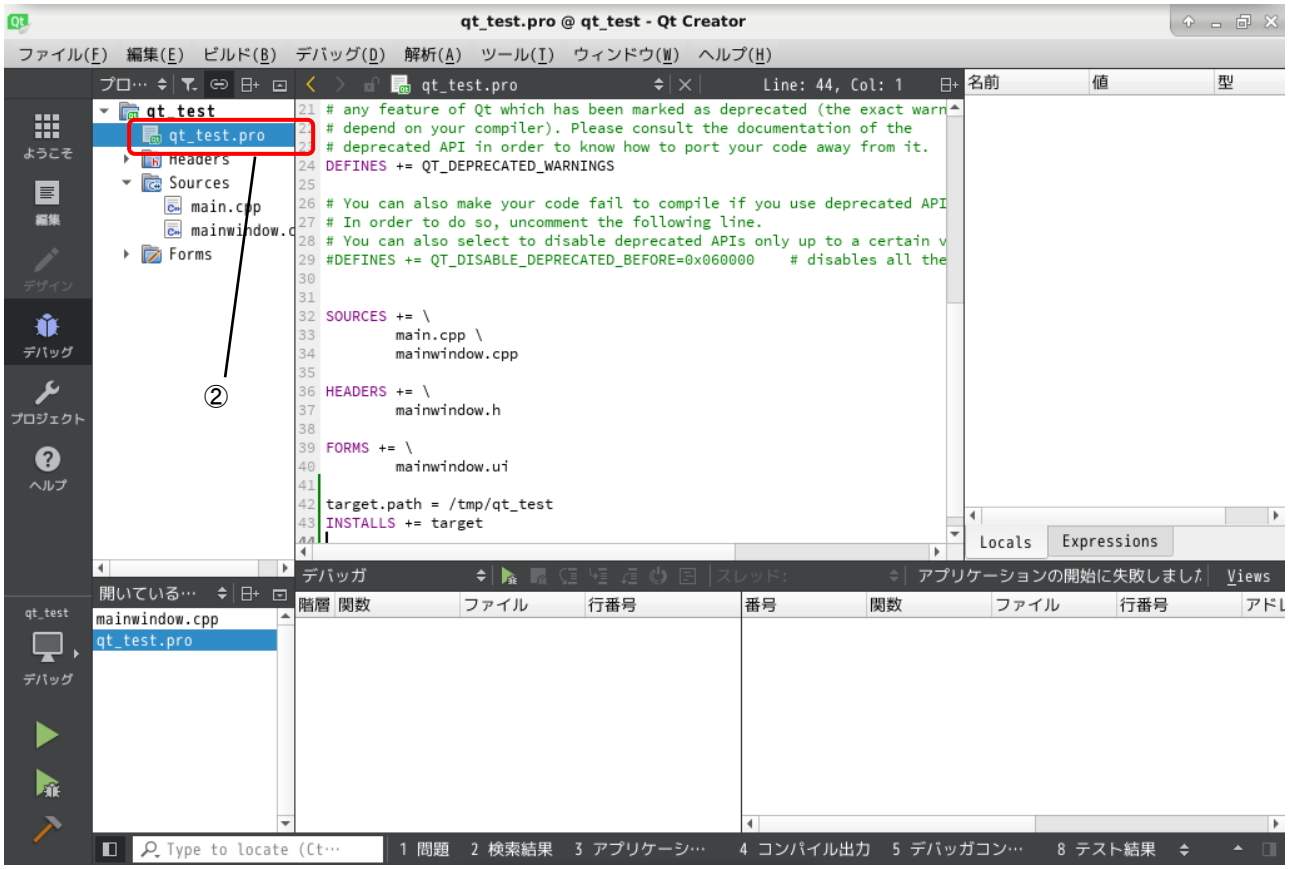


図 2-4-9-4. プロジェクトファイル

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。

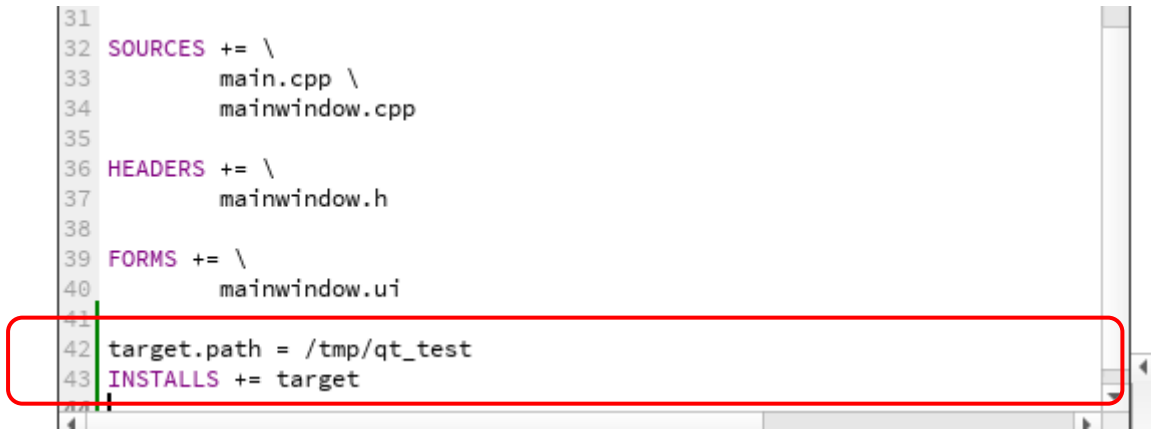


図 2-4-9-5. プロジェクトファイルコンパイルオプション追加

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。

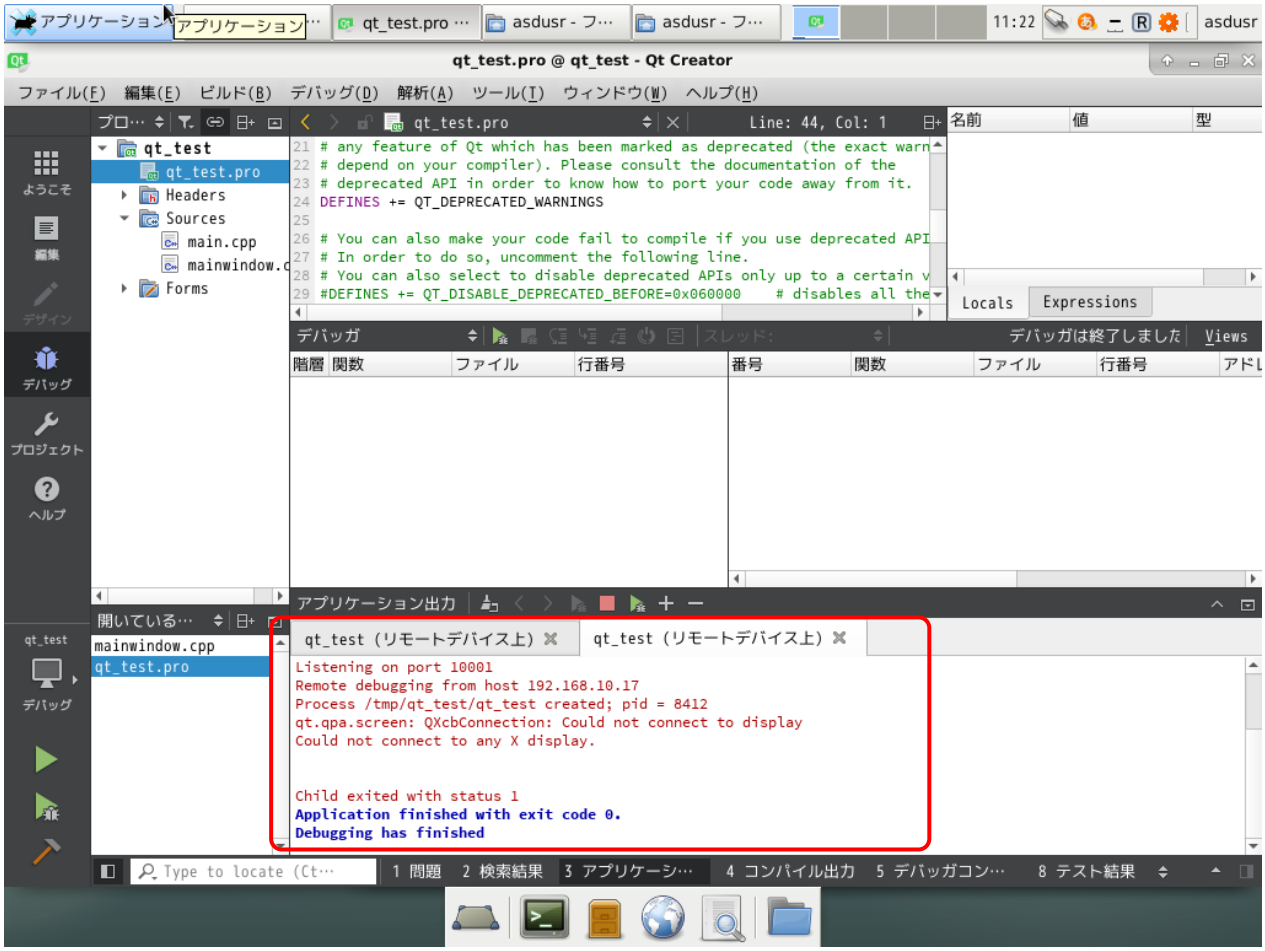


図 2-4-9-6. デバッグ実行時に出力されるメッセージ

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。

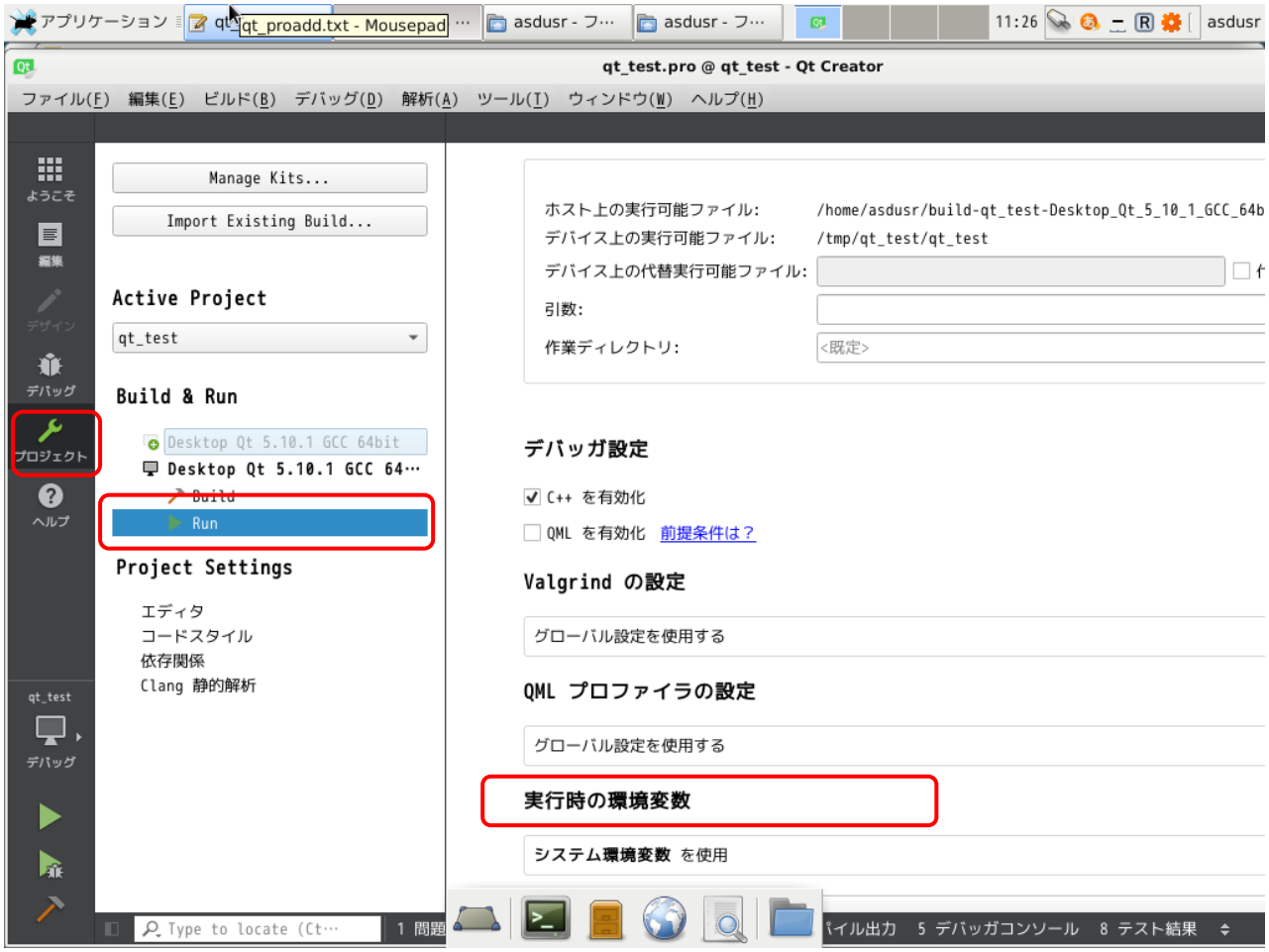


図 2-4-9-7. 実行時の環境変数設定

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。

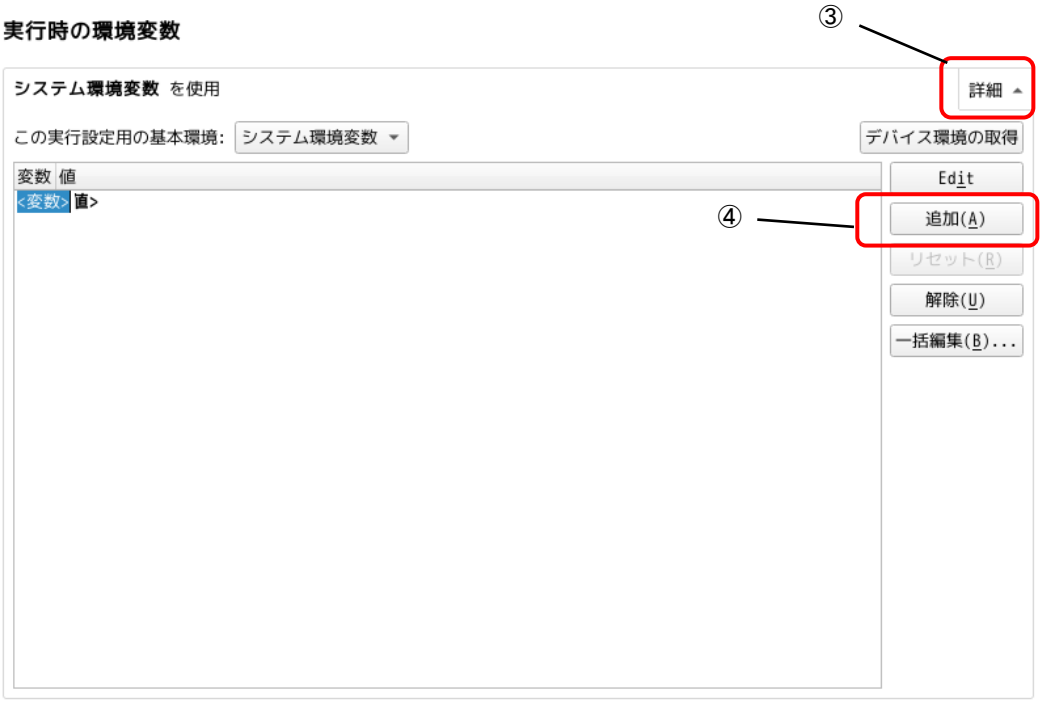


図 2-4-9-8. 実行時の環境変数設定（項目の追加）

⑤ 変数に「DISPLAY」、値に「:0」を設定します。

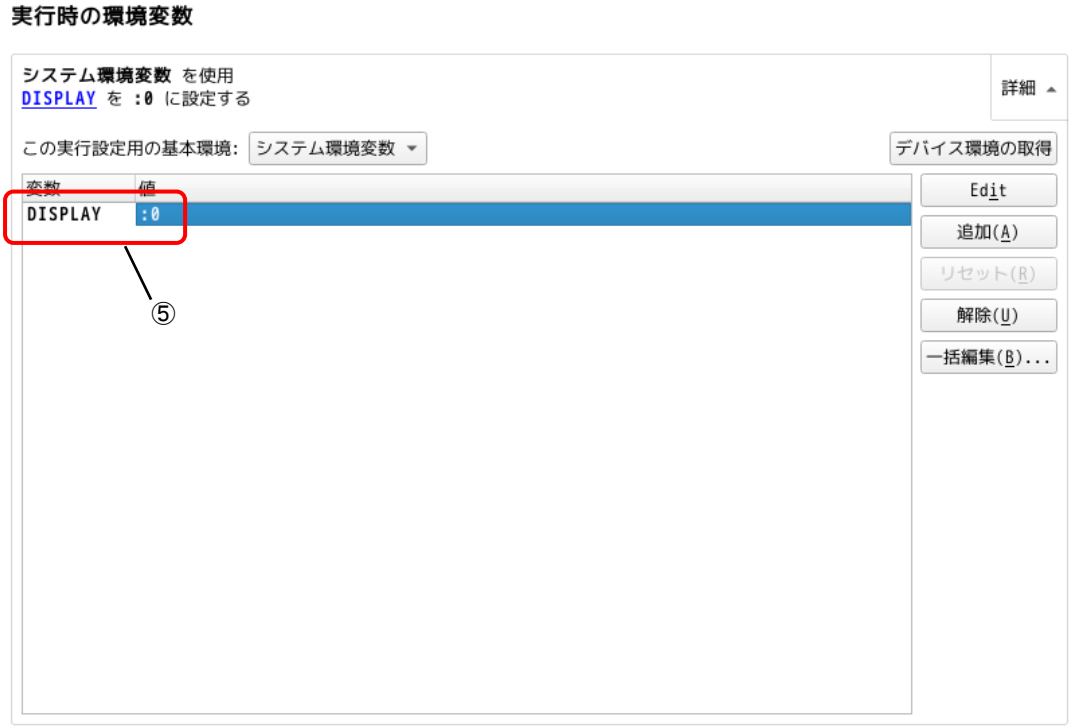


図 2-4-9-9. 実行時の環境変数設定（DISPLAY 環境変数設定）

■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。

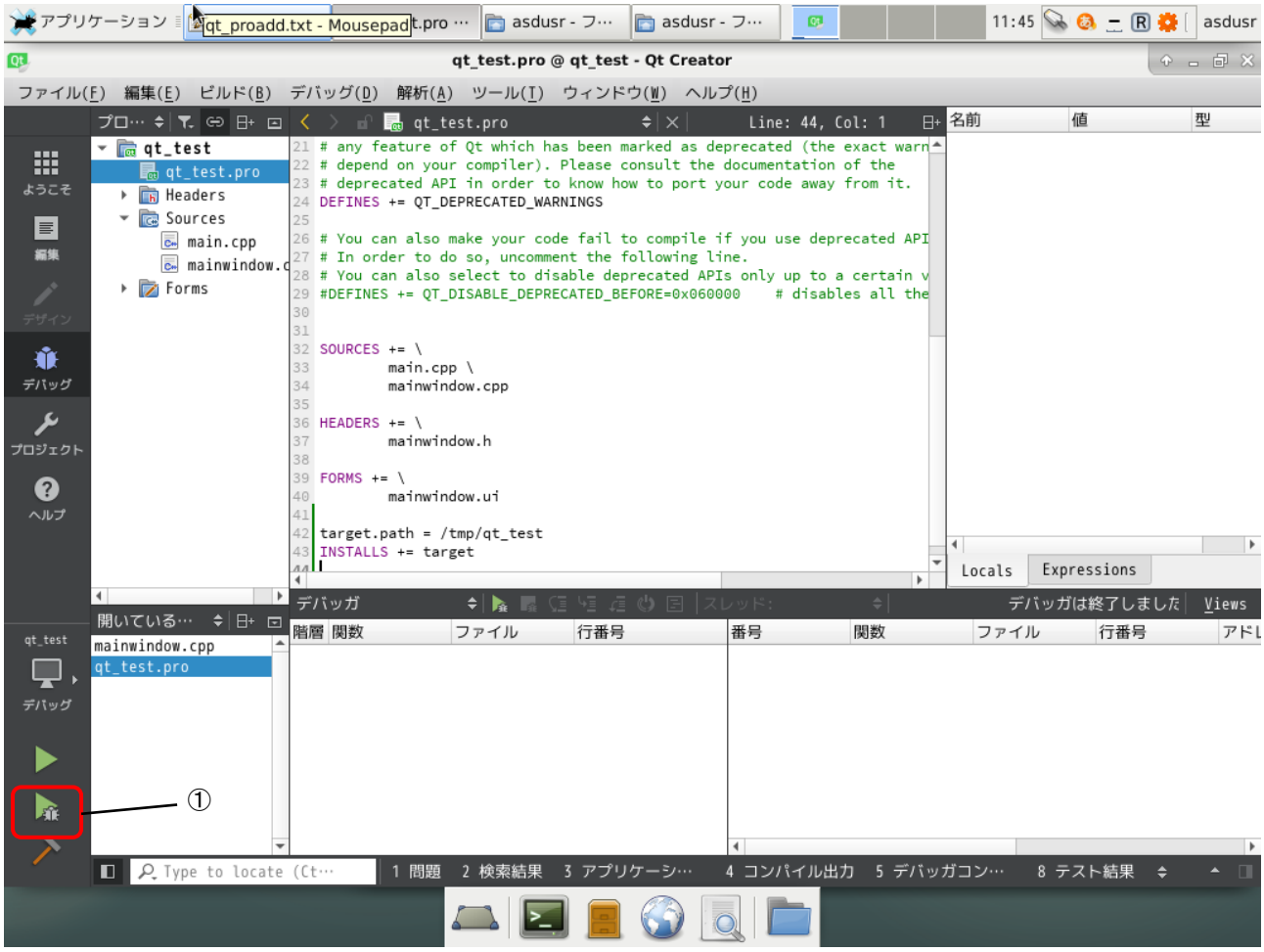


図 2-4-9-10. リモートデバッグ開始

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。

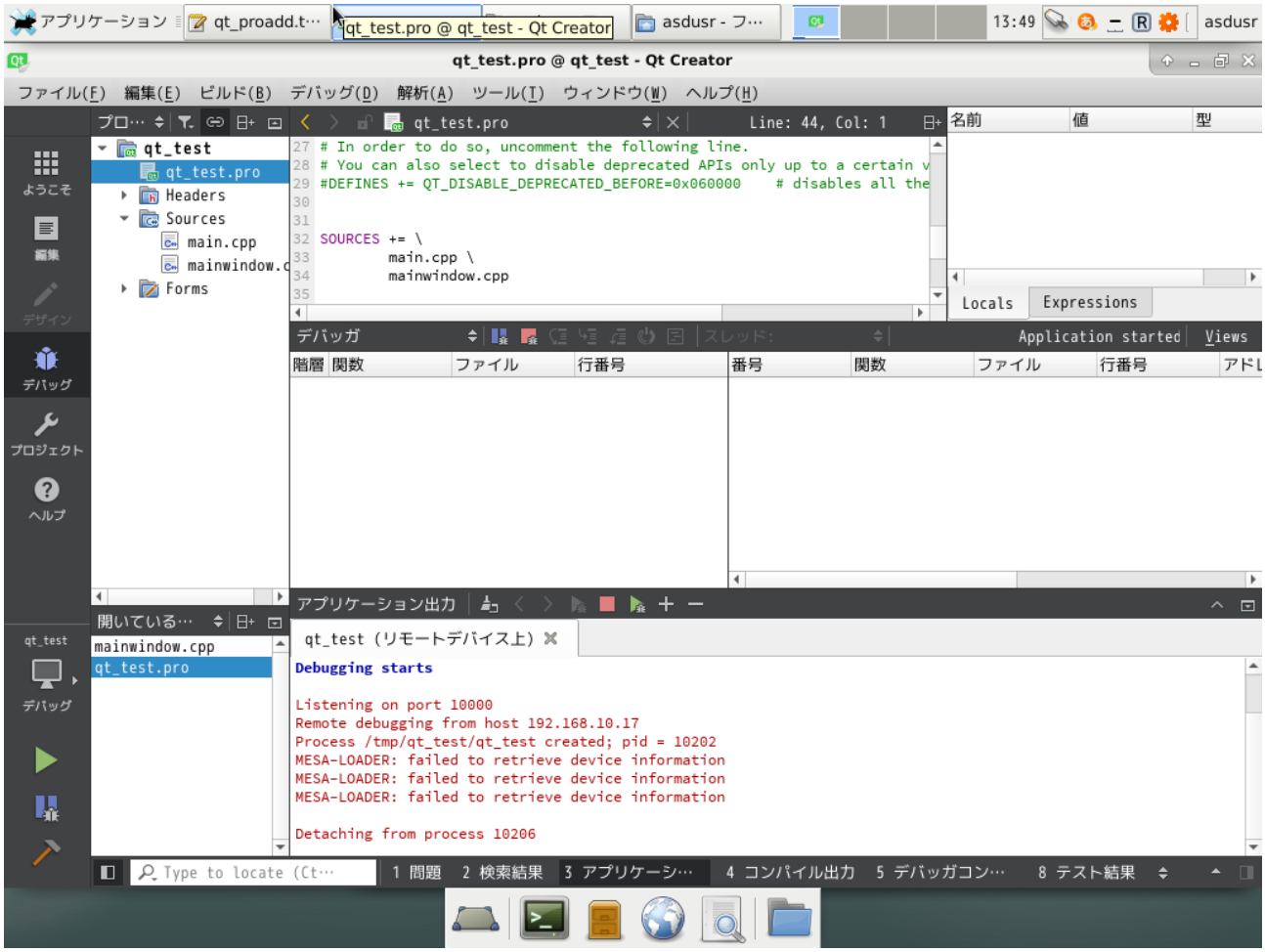


図 2-4-9-11. 正常に接続された場合

２－５　　MonoDevelop によるアプリケーション開発

Mono とは、マルチプラットフォームで動作する、.NET Framework 互換のフレームワークです。MonoDevelop とは、Mono 上で動作するアプリケーションを開発するための統合開発環境です。

本節では、MonoDevelop を用いたアプリケーション開発の例を示します。

２－５－１　作成するアプリケーション

本節では、MonoDevelop を用いたアプリケーションとして、図 2-5-1-1 に示す汎用入出力の監視、制御を行う GUI アプリケーション作成方法を示します。

このアプリケーションは、汎用入力、汎用出力の値を 100msec 周期で監視し、値が ON になれば対応するウィジェット(GUI 部品)を図 2-3-4-2 のように明るく変化させます。また、OUT のボタンが押されたとき、対応する汎用出力の ON/OFF を切り替えます。

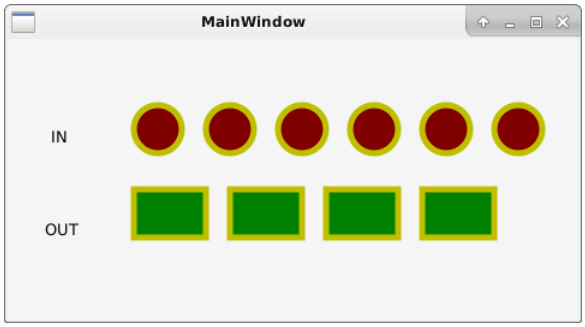


図 2-5-1-1. サンプルアプリケーション

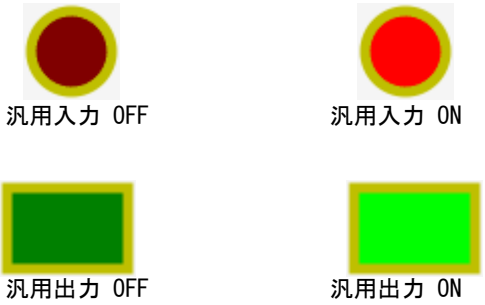


図 2-5-1-2. 汎用入出力

２－５－２　MonoDevelop の起動

ここでは、MonoDevelop の起動方法を説明します。

①　コンソールを起動し、以下のコマンドを実行することで、MonoDevelop が起動します。

```
$ flatpak run com.xamarin.MonoDevelop
```

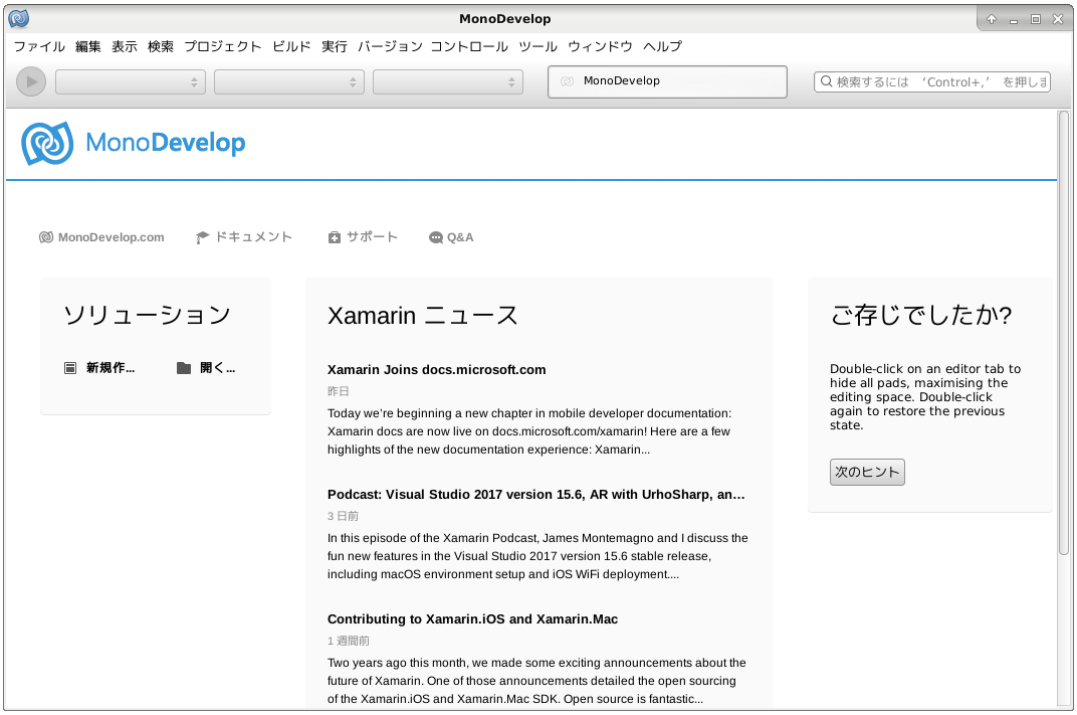


図 2-5-2-1. MonoDevelop 起動後の画面

２－５－３　ソリューションとプロジェクトの新規作成

ここでは、ソリューション、プロジェクトの作成方法について記します。

①　メニューの「ファイル」→「新しいソリューション」を選択します。

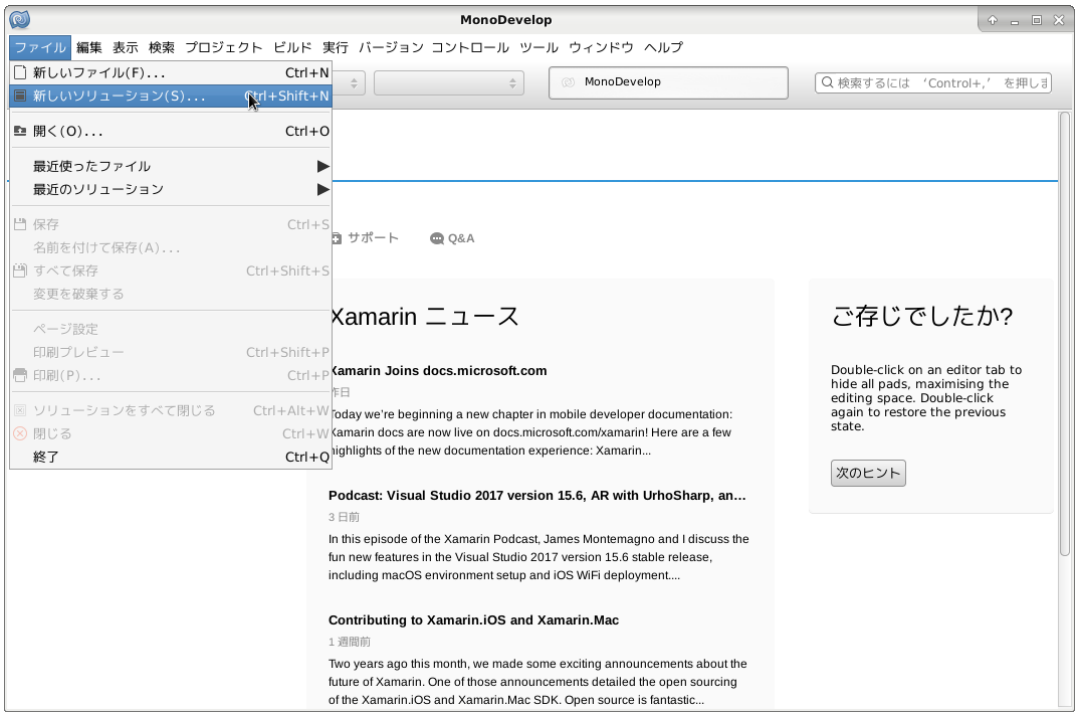


図 2-5-3-1.　新しいソリューションの作成

②　「新しいプロジェクト」ダイアログが立ち上がります。
「.NET」の「Gtk# 2.0 プロジェクト」を選択し、「次へ」ボタンをクリックします。

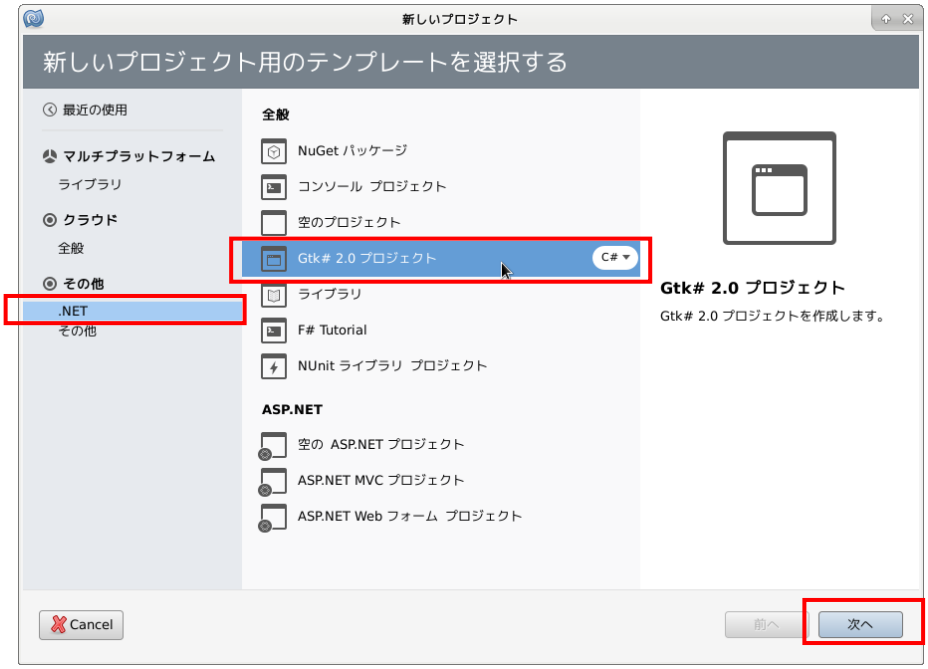


図 2-5-3-2.　「新しいプロジェクト」ダイアログ

③　プロジェクト名、ソリューション名、ソリューションの場所を入力する画面が表示されます。
ここでは、プロジェクト名、ソリューション名を「SampleGenIO」、ソリューションの場所を「/home/asdusr/Projects」とします。
入力後、「作成」ボタンをクリックします。

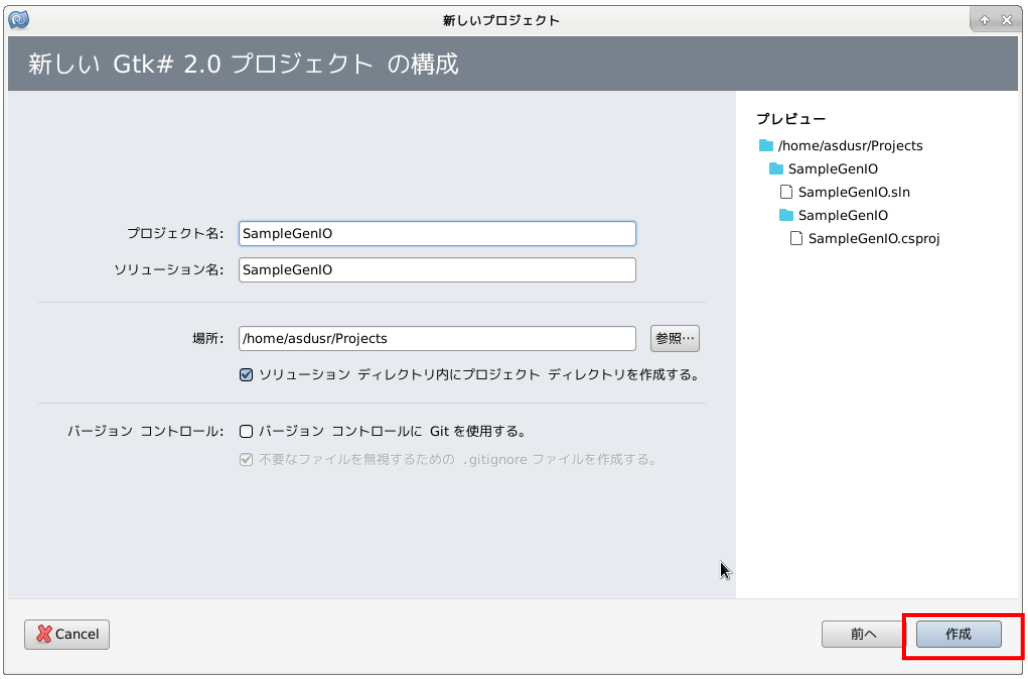


図 2-5-3-3.　プロジェクト名、ソリューション名、場所の入力

④　プロジェクト作成が完了すると、図 2-5-3-4 のような画面が表示されます。

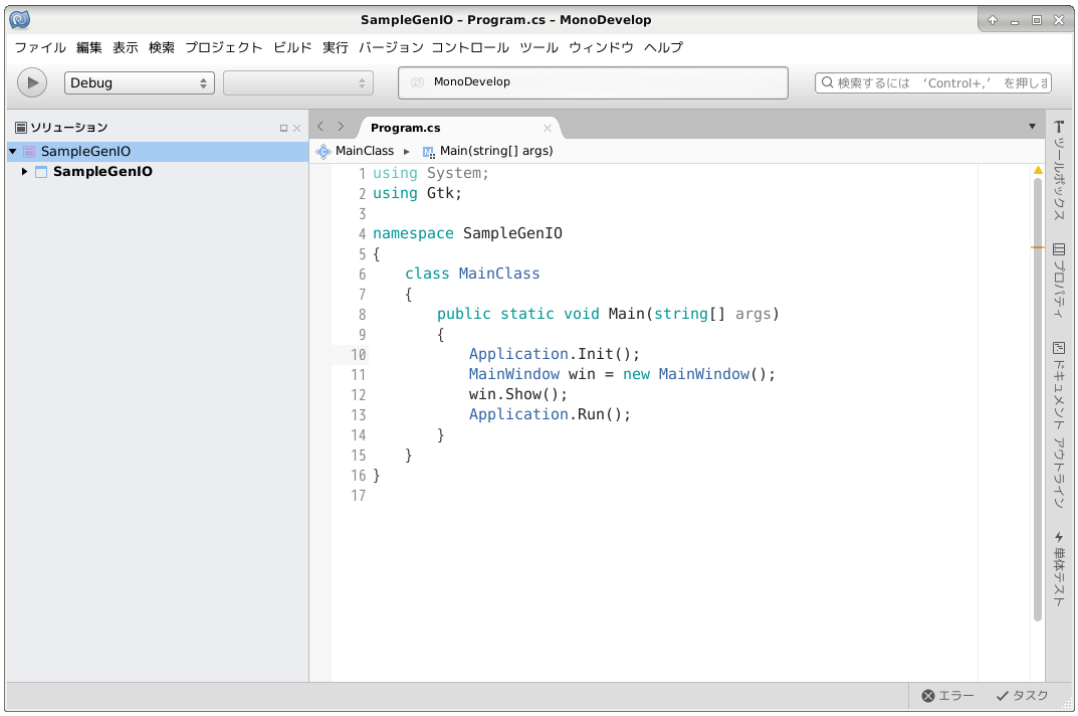


図 2-5-3-4.　プロジェクト作成後の画面

２－５－４　ウィンドウの作成

ここでは、デザイナーツールによる画面編集方法について記します。

① 画面左側の「ソリューション」から、「SampleGenIO」→「SampleGeIO」→「ユーザー インターフェイス」→「MainWindow」を選択し、ダブルクリックします。
図 2-5-4-2 に示すデザイナー画面が表示されます。

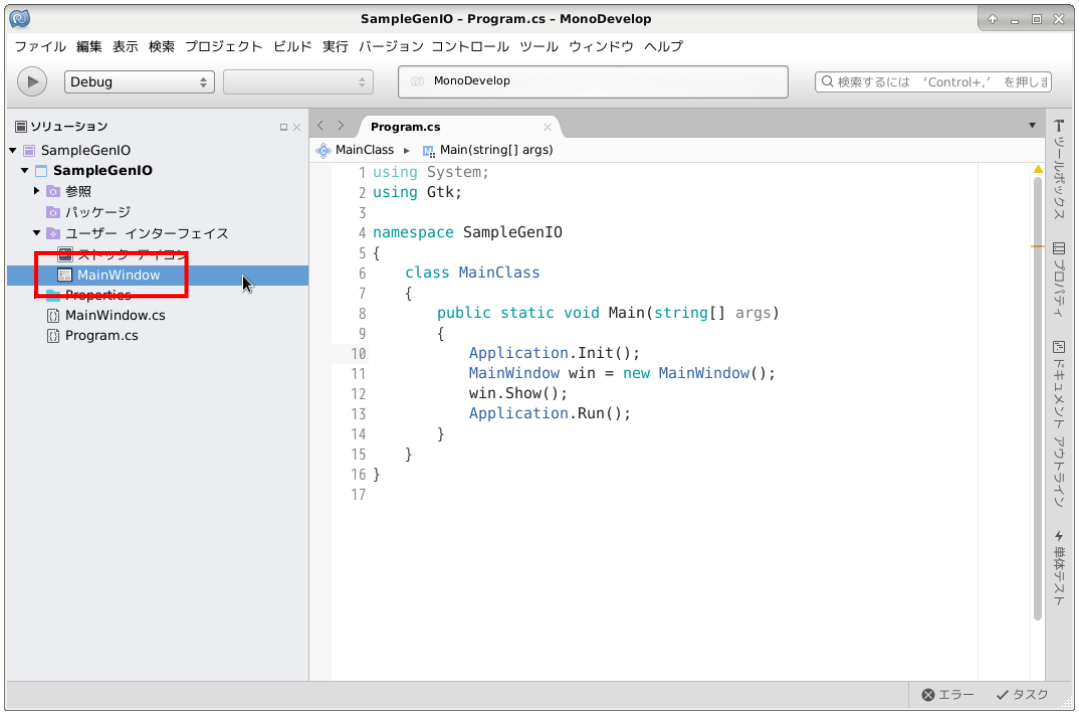


図 2-5-4-1. MainWindow 表示

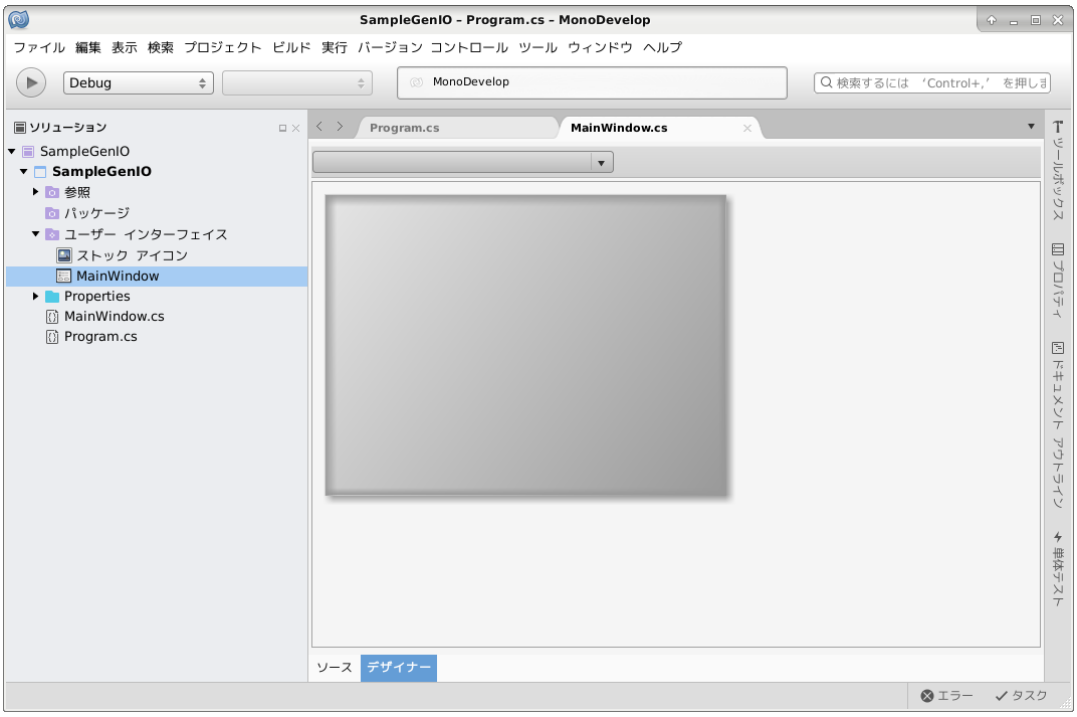


図 2-5-4-2. MainWindow デザイナー画面

② 「ツールボックス」をマウスオーバー、もしくはクリックすることでツールボックスが表示されます。ツールボックス内にある「Fixed」をデザイナー画面のメインウィンドウ（灰色の四角）上にドラッグ&ドロップします。

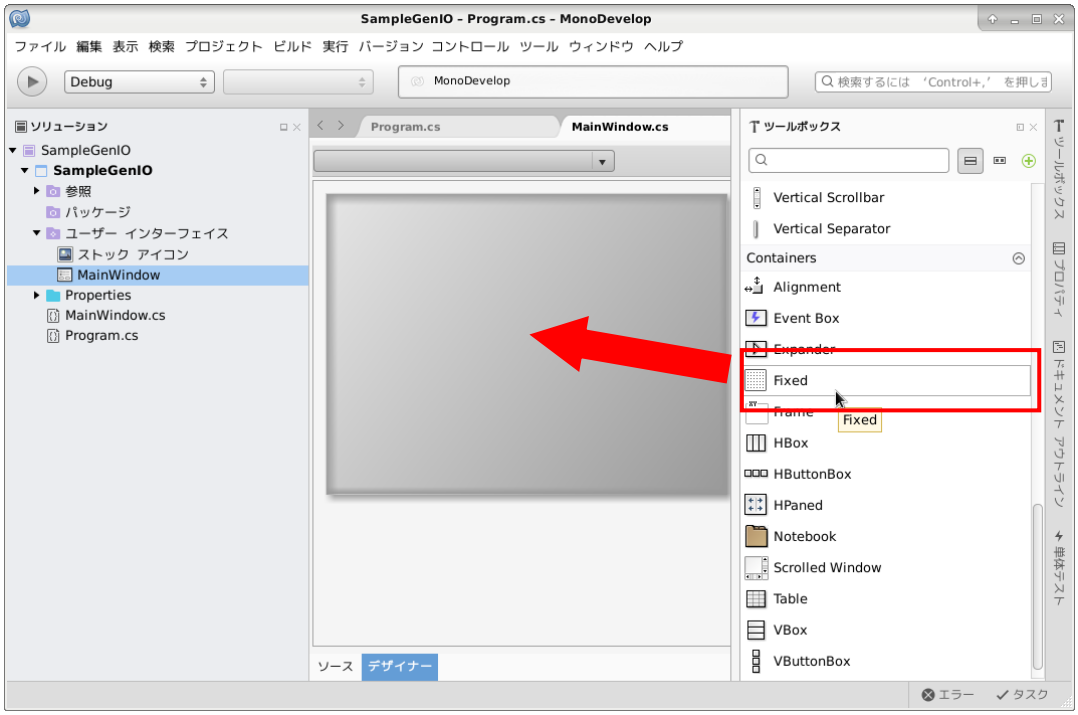


図 2-5-4-3. Fixed の貼り付け

③ Fixed をドラッグ&ドロップした後、メインウィンドウが白くなります。続いて、ツールボックス内にある「Label」をメインウィンドウ上にドラッグ&ドロップします。

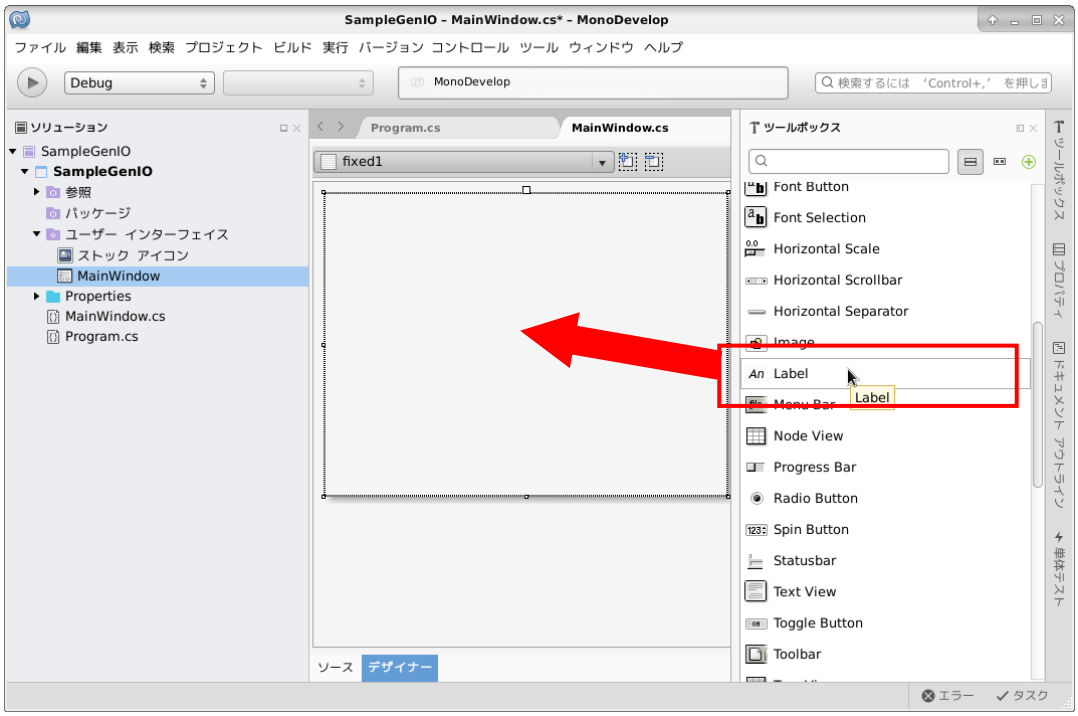


図 2-5-4-4. Label の貼り付け

④　Label をドラッグ&ドロップ後、メインウィンドウにラベルが表示されます。
ラベルを選択し、「プロパティ」をクリックすることで、プロパティ一覧が表示されます。プロパティ一覧の「Label Properties」以下にある「LabelProp」を「IN」に変更します。

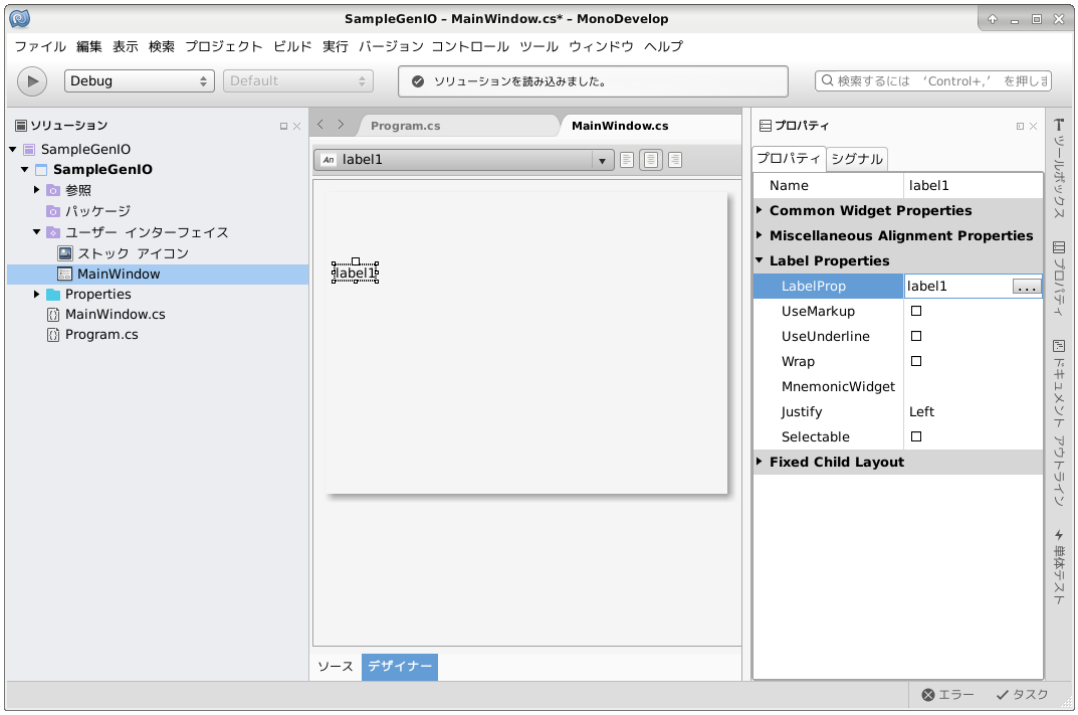


図 2-5-4-5. Label のプロパティ編集

⑤　同様に Label を追加し、「LabelProp」を「OUT」に変更します。

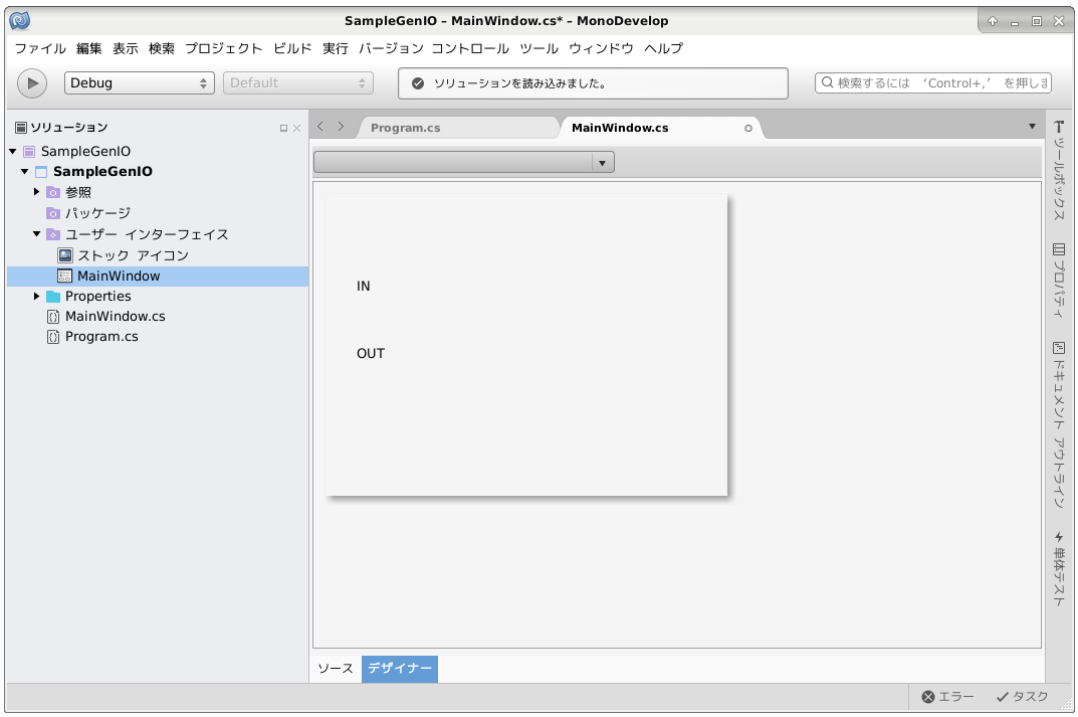


図 2-5-4-6. メインウィンドウ画面

⑥　必要に応じて、ラベル位置やウィンドウサイズを調整します。

２－５－５　カスタム描画ウィジェットの作成

ここでは、汎用入力状態を表すウィジェット、汎用出力状態を表すウィジェットの作成方法を記します。

これらのウィジェットを描画するために、cairo という名称のライブラリを使用します。cairo ライブラリの詳細は、インターネット等を参照してください。

①　画面左側の「ソリューション」にある「参照」を右クリックし、「参照の編集(E)...」を選択します。

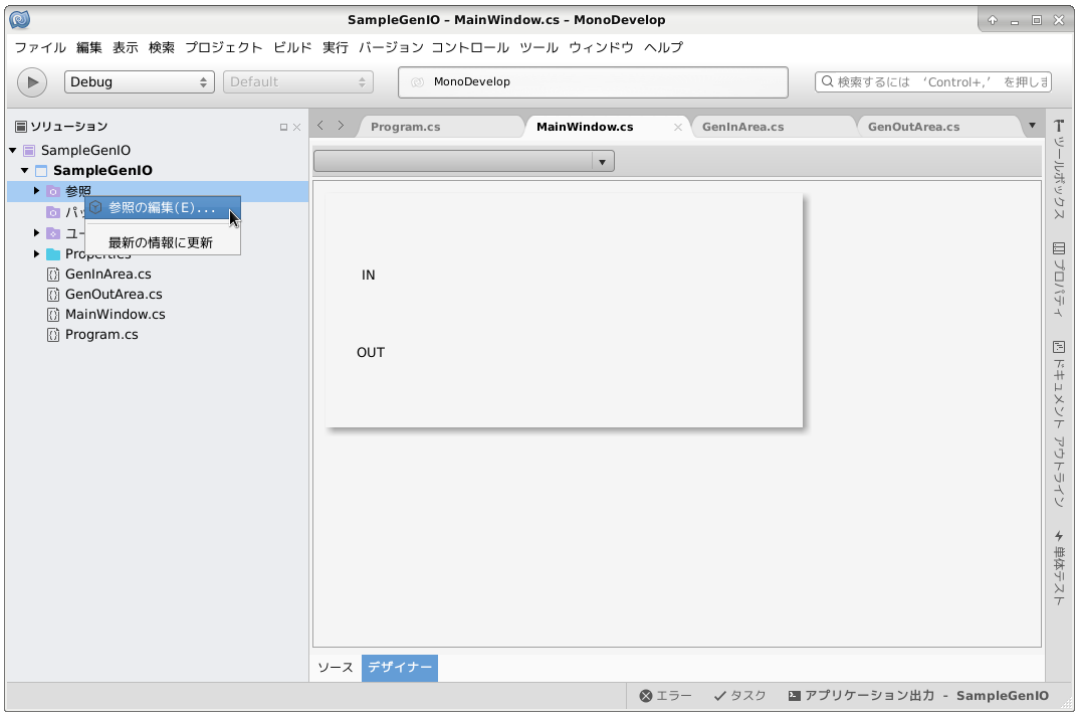


図 2-5-5-1. 参照の編集

②　「参照の編集」ダイアログが表示されます。「Mono.Cairo」の左にあるチェックボックスにチェックを入れ、「OK」ボタンをクリックします。

「参照の編集」ダイアログが閉じ、「Mono.Cairo」が参照に追加されます。

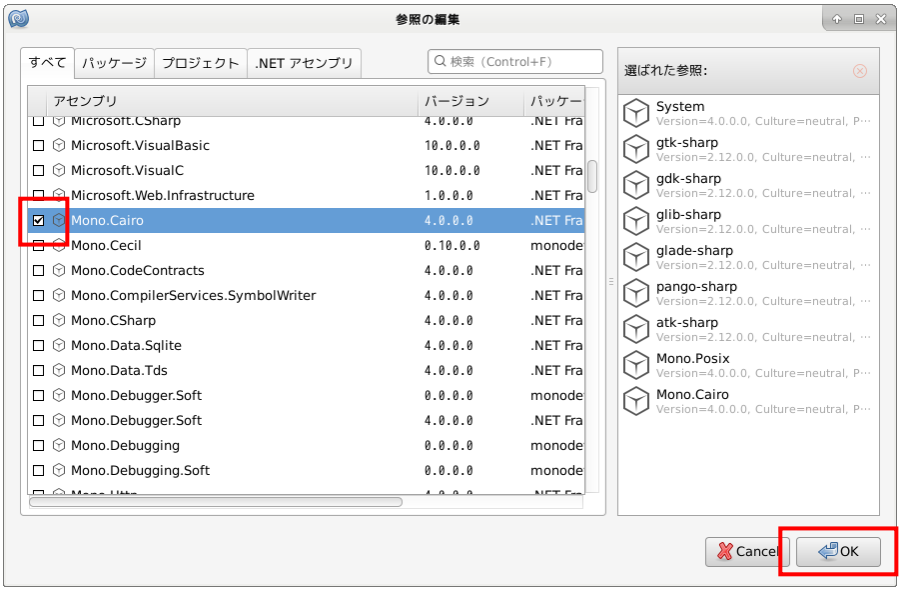


図 2-5-5-2. 「参照の編集」ダイアログ

③　画面左側の「ソリューション」にある「SampleGenIO」プロジェクトを右クリックし、「追加」→「新しいファイル」を選択します。

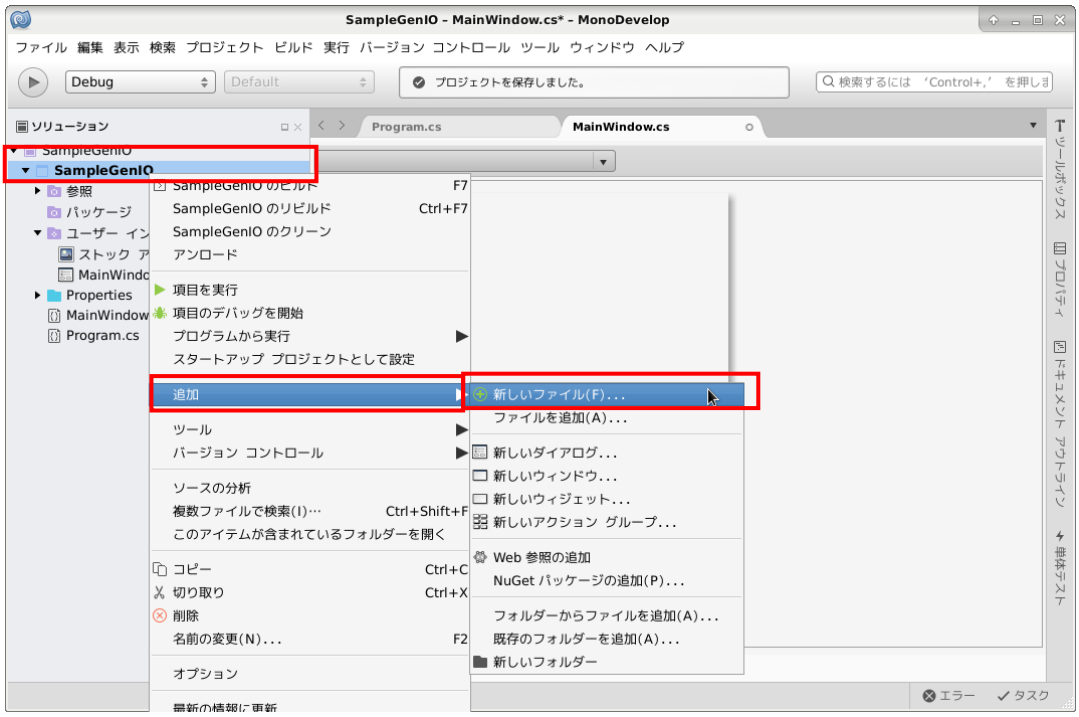


図 2-5-5-3. 新しいファイルの追加

④　「新しいファイル」ダイアログが表示されます。
「Gtk」の「カスタム描画ウィジェット」を選択します。
「名前」を「GenInArea」に変更し、「New」ボタンをクリックします。

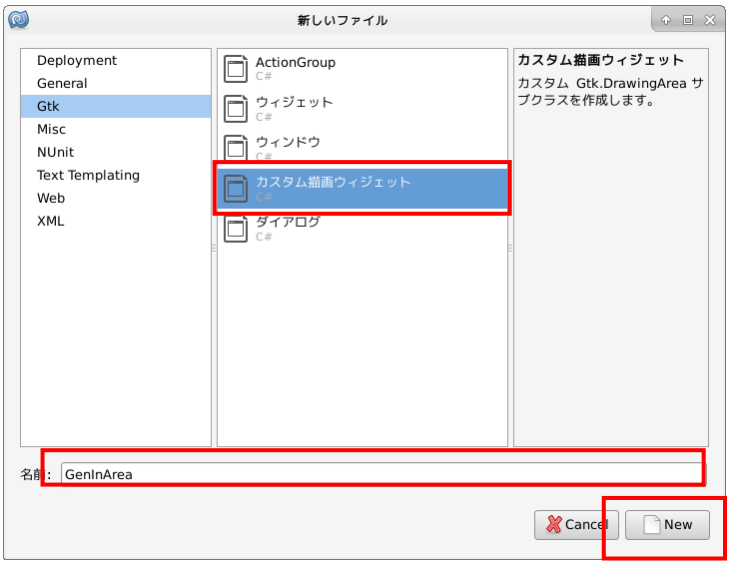


図 2-5-5-4. 「新しいファイル」ダイアログ

⑤　プロジェクトに「GenInArea.cs」が追加されます。「GenInArea.cs」をリスト 2-5-5-1 のように編集します。赤枠で囲った部分が追加部分です。
MonoDevelop 上で ASCII 以外の文字は入力できないので、ASCII 以外の文字を入力するときはエディタで編集してください。

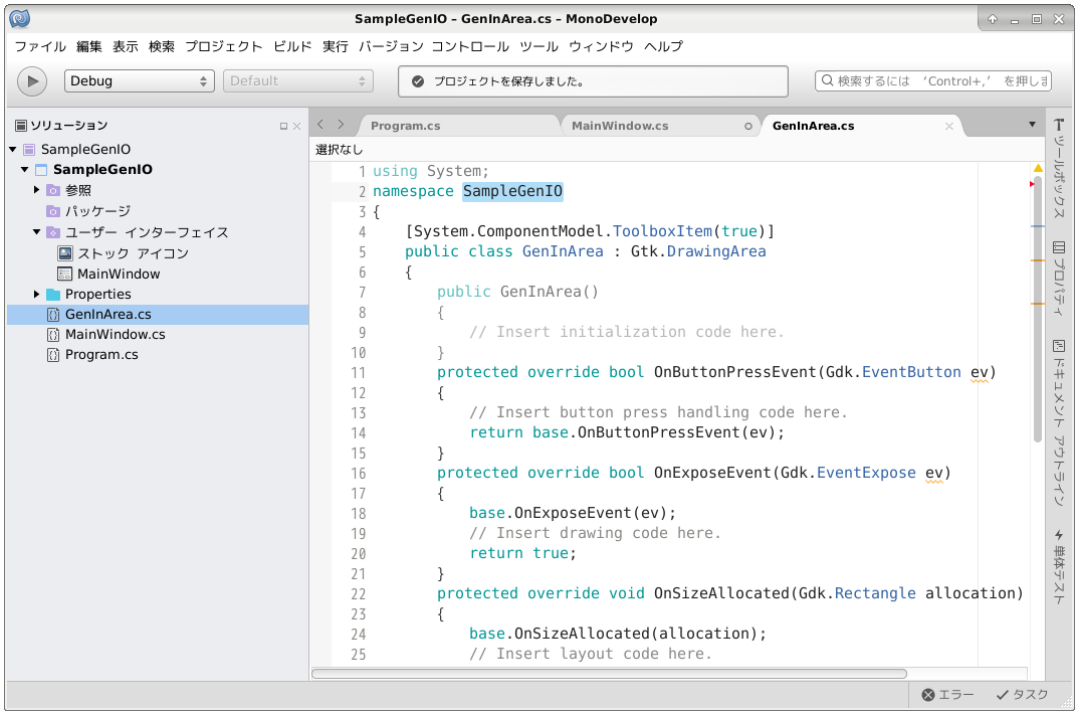


図 2-5-5-5.　「新しいファイル」ダイアログ

リスト 2-5-5-1.　GenInArea.cs

```
using System;
namespace SampleGenIO
{
    [System.ComponentModel.ToolboxItem(true)]
    public class GenInArea : Gtk.DrawingArea
    {
        private bool IsOn_;
        public bool IsOn
        {
            get
            {
                return this.IsOn_;
            }
            set
            {
                if (this.IsOn_ != value) {
                    this.IsOn_ = value;
                    // 図形を再描画
                    this.QueueDraw();
                }
            }
        }
```

```
        public GenInArea()
        {
            this.IsOn_ = false;
        }

        protected override bool OnButtonPressEvent(Gdk.EventButton ev)
        {
            // Insert button press handling code here.
            return base.OnButtonPressEvent(ev);
        }
        protected override bool OnExposeEvent(Gdk.EventExpose ev)
        {
            base.OnExposeEvent(ev);

            // cairo ライブラリを使用して、円を描画する
            Cairo.Context cr = Gdk.CairoHelper.Create(ev.Window);
            int centerX = this.Allocation.Width / 2;
            int centerY = this.Allocation.Height / 2;
            cr.LineWidth = 5.0;
            cr.SetSourceRGB(0.75, 0.75, 0.0);
            cr.Save();
            cr.Arc(centerX, centerY, this.Allocation.Width * 0.4, 0.0, 2 * System.Math.PI);
            if (this.IsOn) {
                cr.SetSourceRGB(1.0, 0.0, 0.0);
            } else {
                cr.SetSourceRGB(0.5, 0.0, 0.0);
            }
            cr.FillPreserve();
            cr.Restore();
            cr.Stroke();

            return true;
        }
        protected override void OnSizeAllocated(Gdk.Rectangle allocation)
        {
            base.OnSizeAllocated(allocation);
            // Insert layout code here.
        }
        protected override void OnSizeRequested(ref Gtk.Requisition requisition)
        {
            // Calculate desired size here.
            requisition.Height = 50;
            requisition.Width = 50;
        }
    }
}
```

⑥　『2-5-5 カスタム描画ウィジェットの作成』③～④と同様の手順で、「GenOutArea.cs」を追加し、リスト 2-5-5-2 のように編集します。

リスト 2-5-5-2.　GenOutArea.cs

```
using System;
namespace SampleGenIO
{
    [System.ComponentModel.ToolboxItem(true)]
    public class GenOutArea : Gtk.DrawingArea
    {
        private bool IsOn_;
        public bool IsOn
        {
            get
            {
                return this.IsOn_;
            }
            set
            {
                if (this.IsOn_ != value) {
                    this.IsOn_ = value;
                    // 図形を再描画
                    this.QueueDraw();
                }
            }
        }
        public GenOutArea()
        {
            this.IsOn_ = false;
            // マウスプレスイベントの有効化
            this.AddEvents((int)Gdk.EventMask.ButtonPressMask);
        }
        protected override bool OnButtonPressEvent(Gdk.EventButton ev)
        {
            // Insert button press handling code here.
            return base.OnButtonPressEvent(ev);
        }
        protected override bool OnExposeEvent(Gdk.EventExpose ev)
        {
            base.OnExposeEvent(ev);

            // cairo ライブラリを使用して、長方形を描画する
            Cairo.Context cr = Gdk.CairoHelper.Create(ev.Window);
            int centerX = this.Allocation.Width / 2;
            int centerY = this.Allocation.Height / 2;
            cr.LineWidth = 5.0;
            cr.SetSourceRGB(0.75, 0.75, 0.0);
            cr.Save();

            cr.Rectangle(5, 5, 60, 40);
            if (this.IsOn) {
```

```
                cr.SetSourceRGB(0.0, 1.0, 0.0);
            } else {
                cr.SetSourceRGB(0.0, 0.5, 0.0);
            }
            cr.FillPreserve();
            cr.Restore();
            cr.Stroke();

            return true;
        }
        protected override void OnSizeAllocated(Gdk.Rectangle allocation)
        {
            base.OnSizeAllocated(allocation);
            // Insert layout code here.
        }
        protected override void OnSizeRequested(ref Gtk.Requisition requisition)
        {
            // Calculate desired size here.
            requisition.Height = 50;
            requisition.Width = 70;
        }
    }
}
```

２－５－６　カスタム描画ウィジェットの追加

ここでは、作成したカスタム描画ウィジェットをメインウィンドウに追加し、実行する方法について説明します。

① 画面左側の「ソリューション」にある「MainWindow.cs」をダブルクリックします。その後、画面下側にある「ソース」をクリックし、ソースを表示します。

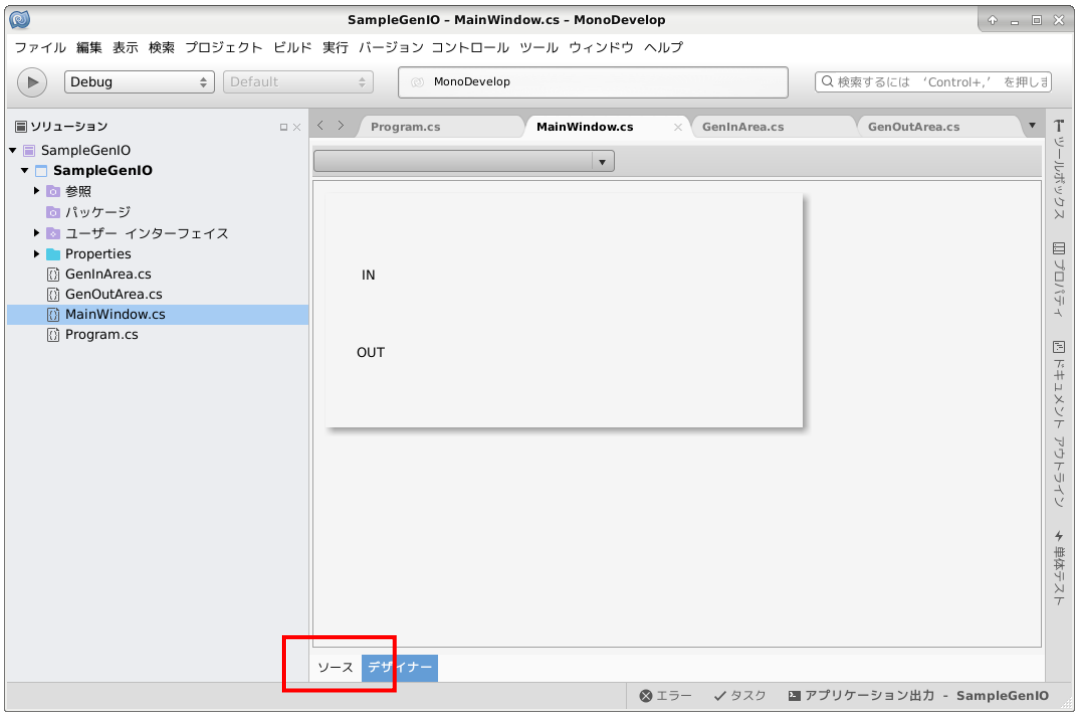


図 2-5-6-1. メインウィンドウの編集

② MainWindow.cs のソースが表示されるので、リスト 2-5-6-1 のように編集します。

リスト 2-5-6-1. MainWindow.cs

```
using System;
using Gtk;

public partial class MainWindow : Gtk.Window
{
    public MainWindow() : base(Gtk.WindowType.Toplevel)
    {
        Build();
        // ウィジェットの作成
        SampleGenIO.GenInArea in1 = new SampleGenIO.GenInArea();
        SampleGenIO.GenInArea in2 = new SampleGenIO.GenInArea();
        SampleGenIO.GenInArea in3 = new SampleGenIO.GenInArea();
        SampleGenIO.GenInArea in4 = new SampleGenIO.GenInArea();
        SampleGenIO.GenInArea in5 = new SampleGenIO.GenInArea();
        SampleGenIO.GenInArea in6 = new SampleGenIO.GenInArea();
        SampleGenIO.GenOutArea out1 = new SampleGenIO.GenOutArea();
        SampleGenIO.GenOutArea out2 = new SampleGenIO.GenOutArea();
        SampleGenIO.GenOutArea out3 = new SampleGenIO.GenOutArea();
        SampleGenIO.GenOutArea out4 = new SampleGenIO.GenOutArea();
```

```
        // ウィジェットの配置
        this.@fixed1.Put(in1, 100, 50);
        this.@fixed1.Put(in2, 160, 50);
        this.@fixed1.Put(in3, 220, 50);
        this.@fixed1.Put(in4, 280, 50);
        this.@fixed1.Put(in5, 340, 50);
        this.@fixed1.Put(in6, 400, 50);
        this.@fixed1.Put(out1, 100, 120);
        this.@fixed1.Put(out2, 180, 120);
        this.@fixed1.Put(out3, 260, 120);
        this.@fixed1.Put(out4, 340, 120);

        // ウィジェットの表示
        this.ShowAll();
    }

    protected void OnDeleteEvent(object sender, DeleteEventArgs a)
    {
        Application.Quit();
        a.RetVal = true;
    }
}
```

③　編集後、「実行」→「デバッグの開始」を選択することで、コンパイル後、デバッグが開始されます。

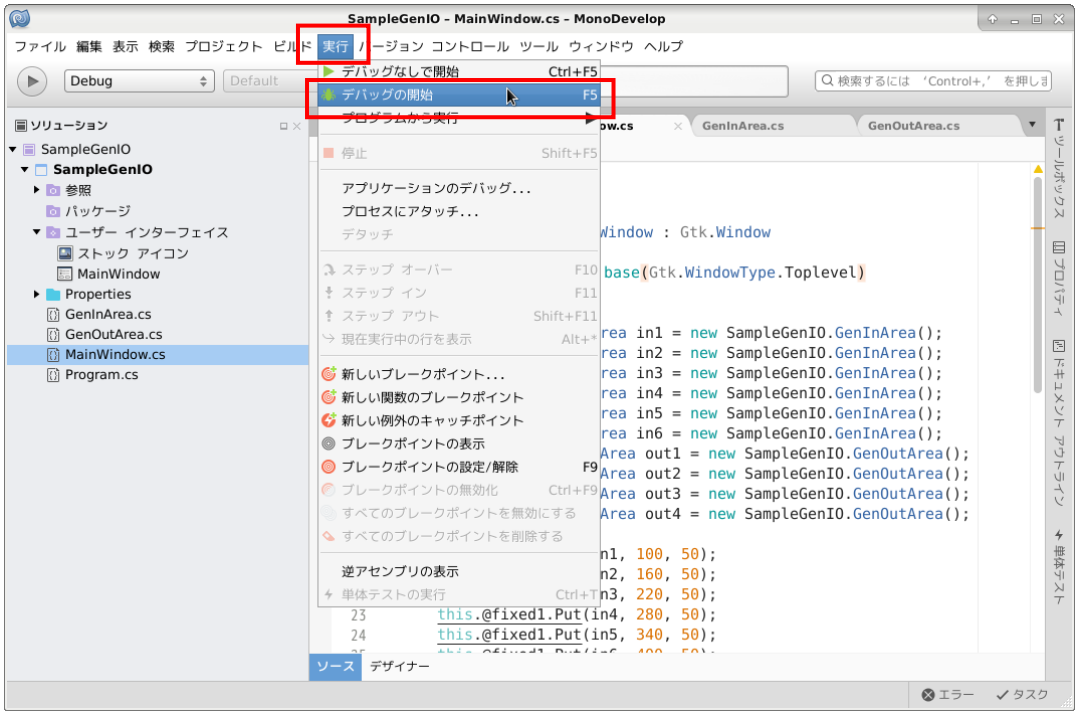


図 2-5-6-2.　デバッグの開始

④　コンパイルに成功すれば、アプリケーションが起動します。
右上の[X]をクリックすることで、アプリケーションが終了します。

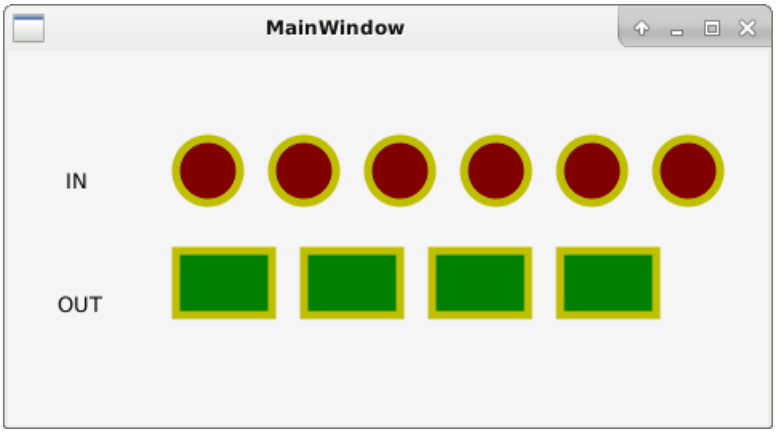


図 2-5-6-3. アプリケーション

２－５－７　汎用入出力処理の追加

ここでは、アプリケーションに汎用入出力の監視処理、汎用出力の ON/OFF 切り替え処理を追加する方法を説明します。

① 開発環境内にある「AsdLib.dll」をソリューションのあるディレクトリ以下にコピーします。
ここでは、「/home/asdusr/SampleGenIO/SampleGenIO/」以下にコピーするものとします。

② 画面左側の「ソリューション」にある「参照」を右クリックし、「参照の編集(E)...」を選択します。

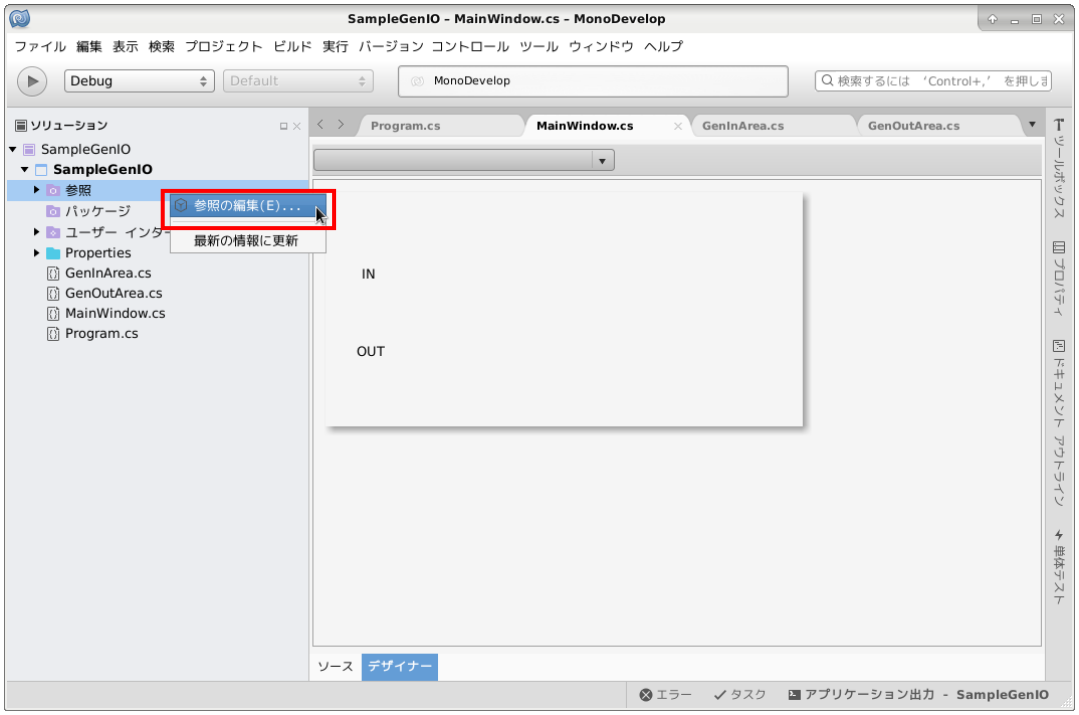


図 2-5-7-1. 参照の編集

③　「参照の編集」ダイアログが表示されます。
「.NET アセンブリ」タブを選択し、「参照...」ボタンをクリックします。

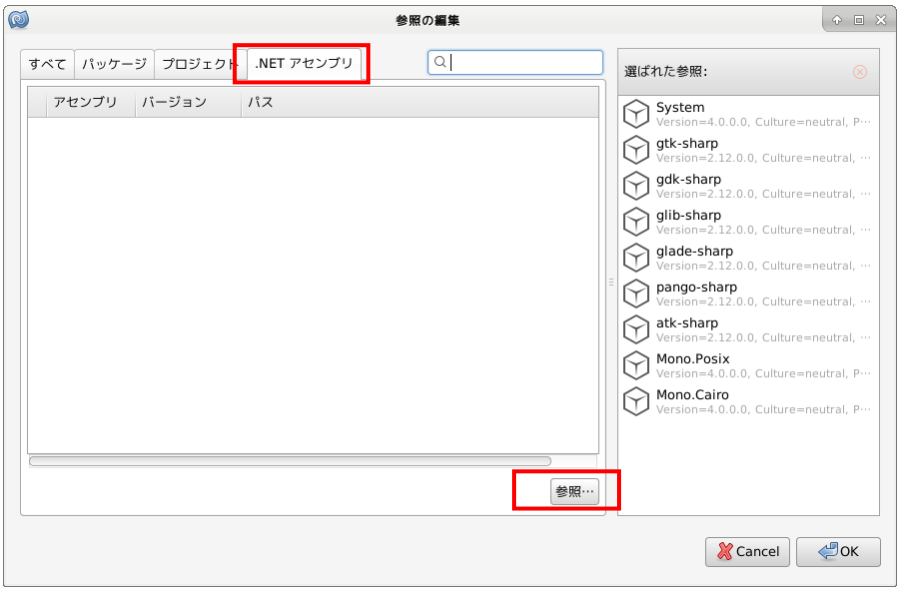


図 2-5-7-2.　「参照の編集」ダイアログ

④　「アセンブリの選択」ダイアログが表示されます。
「AsdLib.dll」を選択し、「Open」ボタンをクリックします。

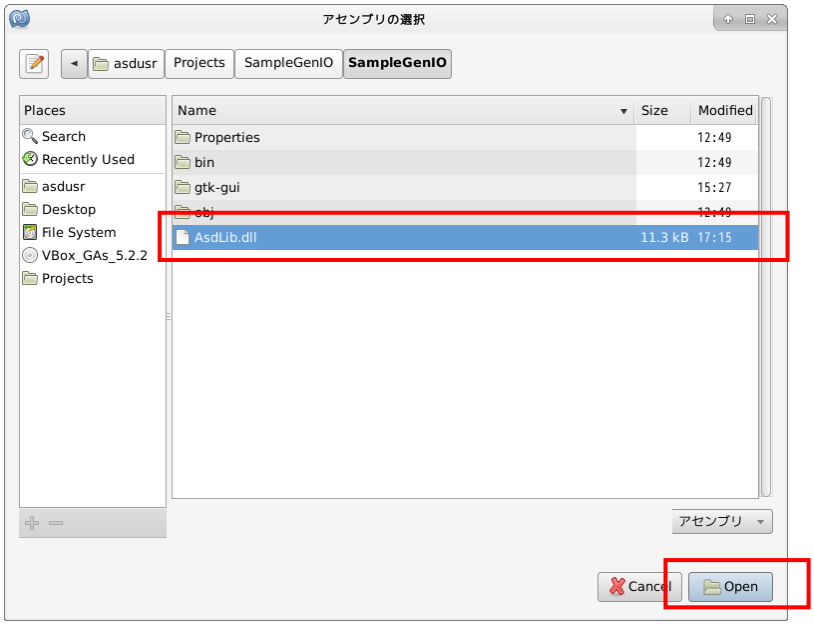


図 2-5-7-3.　「アセンブリの選択」ダイアログ

⑤　「参照の編集」ダイアログに「AsdLib.dll」が追加されるので、「OK」ボタンをクリックします。「参照の編集」ダイアログが閉じ、「AsdLib」が参照に追加されます。

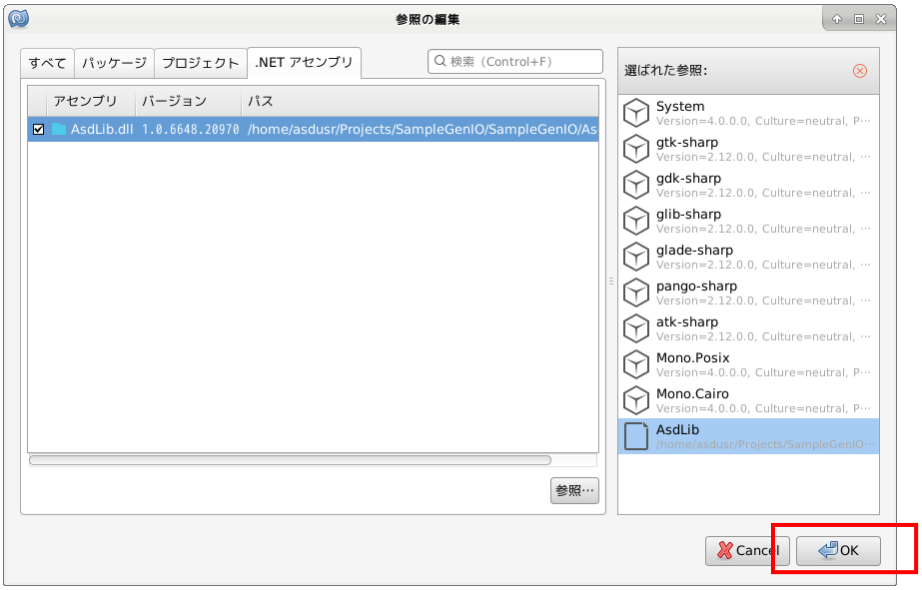


図 2-5-7-4.　「参照の編集」ダイアログ

⑥　画面左側の「ソリューション」にある「MainWindow.cs」をダブルクリックします。その後下部にある「ソース」をクリックし、ソースを表示します。

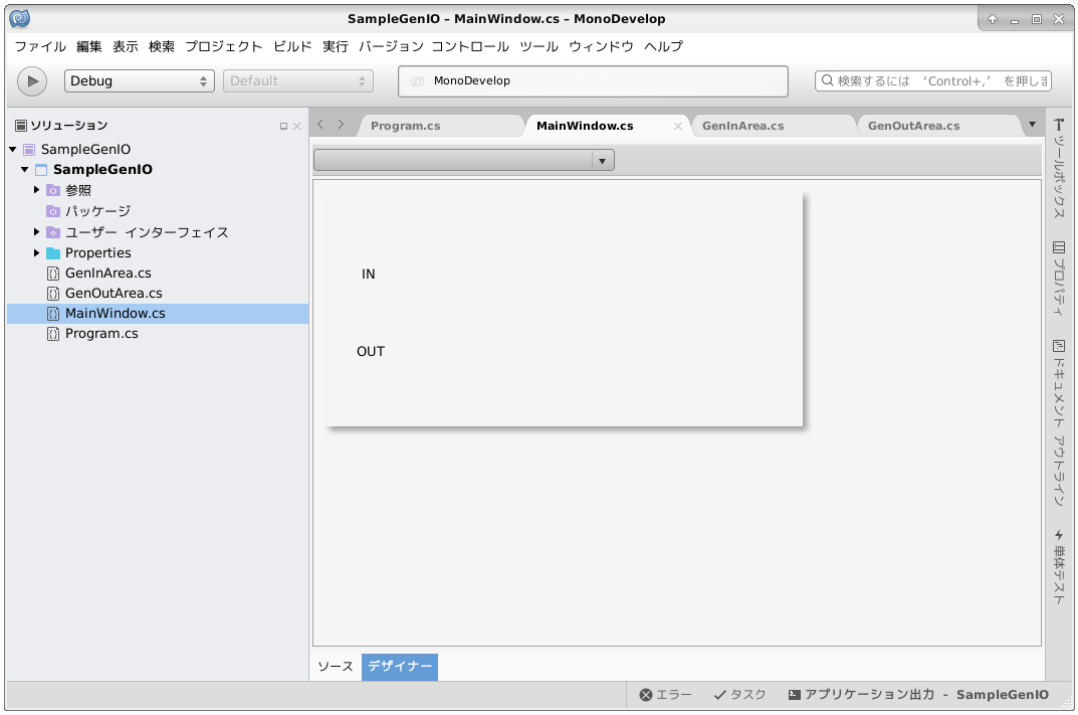


図 2-5-7-5.　メインウィンドウ編集画面

⑦　MainWindow.cs のソースが表示されるので、リスト 2-5-7-1 のように編集します。

リスト 2-5-7-1. MainWindow.cs

```
using System;
using Gtk;

public partial class MainWindow : Gtk.Window
{
    private System.Timers.Timer IntervalTimer;

    public MainWindow() : base(Gtk.WindowType.Toplevel)
    {
        Build();
        // ウィジェットの作成
        SampleGenIO.GenInArea in1 = new SampleGenIO.GenInArea();
        SampleGenIO.GenInArea in2 = new SampleGenIO.GenInArea();
        SampleGenIO.GenInArea in3 = new SampleGenIO.GenInArea();
        SampleGenIO.GenInArea in4 = new SampleGenIO.GenInArea();
        SampleGenIO.GenInArea in5 = new SampleGenIO.GenInArea();
        SampleGenIO.GenInArea in6 = new SampleGenIO.GenInArea();
        SampleGenIO.GenOutArea out1 = new SampleGenIO.GenOutArea();
        SampleGenIO.GenOutArea out2 = new SampleGenIO.GenOutArea();
        SampleGenIO.GenOutArea out3 = new SampleGenIO.GenOutArea();
        SampleGenIO.GenOutArea out4 = new SampleGenIO.GenOutArea();

        // ウィジェットの配置
        this.@fixed1.Put(in1, 100, 50);
        this.@fixed1.Put(in2, 160, 50);
        this.@fixed1.Put(in3, 220, 50);
        this.@fixed1.Put(in4, 280, 50);
        this.@fixed1.Put(in5, 340, 50);
        this.@fixed1.Put(in6, 400, 50);
        this.@fixed1.Put(out1, 100, 120);
        this.@fixed1.Put(out2, 180, 120);
        this.@fixed1.Put(out3, 260, 120);
        this.@fixed1.Put(out4, 340, 120);

        // ウィジェットの表示
        this.ShowAll();

        // 汎用出力ウィジェットにマウスプレスした時、汎用出力の ON/OFF を切り替えるようにす
る
        out1.ButtonPressEvent += (o, args) => {
            AsdCommon.GenIO.GenOut ^= 0x01;
        };
        out2.ButtonPressEvent += (o, args) => {
            AsdCommon.GenIO.GenOut ^= 0x02;
        };
        out3.ButtonPressEvent += (o, args) => {
            AsdCommon.GenIO.GenOut ^= 0x04;
        };
```

```
        out4.ButtonPressEvent += (o, args) => {
            AsdCommon.GenIO.GenOut ^= 0x08;
        };

        // 入力監視用タイマーを追加する
        this.IntervalTimer = new System.Timers.Timer(100);
        this.IntervalTimer.Elapsed += (sender, e) => {
            Gtk.Application.Invoke(delegate {
                // 汎用入力値の取得
                ushort genin = AsdCommon.GenIO.GenIn;
                // 汎用入力値に応じて汎用入力ウィジェットの表示を変える
                in1.IsOn = ((genin & 0x01) != 0);
                in2.IsOn = ((genin & 0x02) != 0);
                in3.IsOn = ((genin & 0x04) != 0);
                in4.IsOn = ((genin & 0x08) != 0);
                in5.IsOn = ((genin & 0x10) != 0);
                in6.IsOn = ((genin & 0x20) != 0);

                // 汎用出力値の取得
                ushort genout = AsdCommon.GenIO.GenOut;
                // 汎用出力値に応じて汎用出力ウィジェットの表示を変える
                out1.IsOn = ((genout & 0x01) != 0);
                out2.IsOn = ((genout & 0x02) != 0);
                out3.IsOn = ((genout & 0x04) != 0);
                out4.IsOn = ((genout & 0x08) != 0);
            });
        };
        // 入力監視用タイマの開始
        this.IntervalTimer.Start();
    }

    protected void OnDeleteEvent(object sender, DeleteEventArgs a)
    {
        Application.Quit();
        a.RetVal = true;
    }
}
```

⑧　メニューの「ビルド」→「すべてビルド」を選択し、プロジェクトをビルドします。

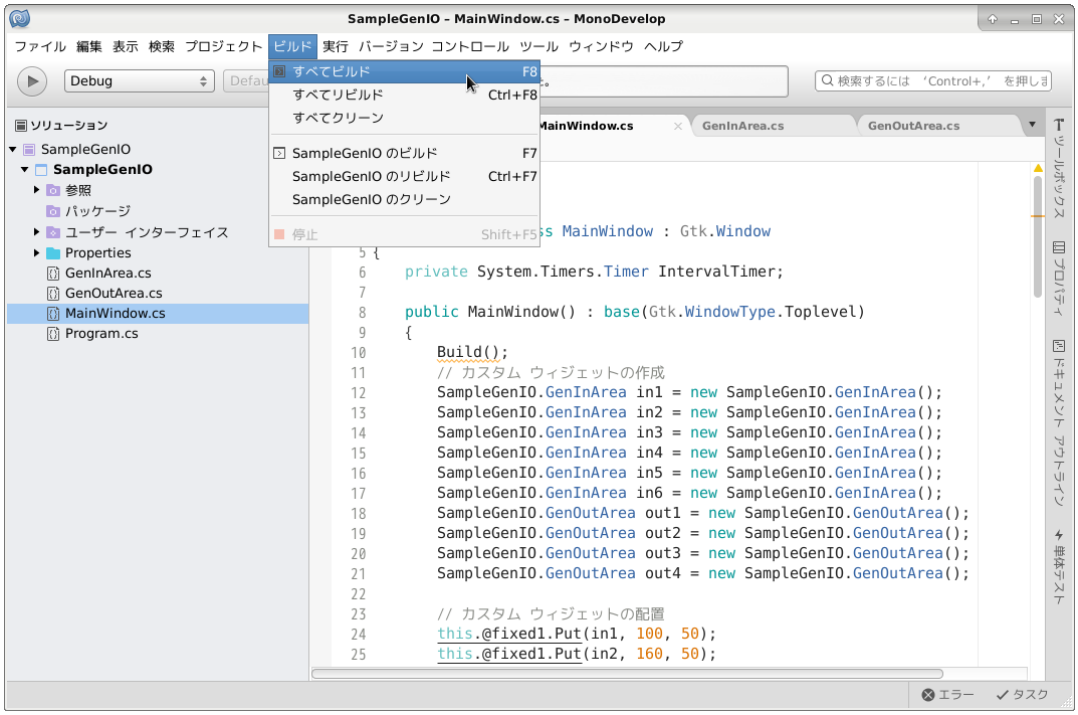


図 2-5-7-6.　メインウィンドウの編集

⑨　ビルドに成功すれば、「/home/asdusr/Projects/SampleGenIO/SampleGenIO/bin/Debug」以下に、「SampleGenIO.exe」というファイルが生成されます。
このファイルと、「AsdLib.dll」を汎用入出力のある端末上に転送します。
転送方法については、『2-3-8 ファイルの転送』を参照してください。

⑩　アプリケーションのコピー先で以下のコマンドを実行することで、アプリケーションが起動します。

```
$ mono SampleGenIO.exe
```

２－６　　GDB によるデバッグ方法の詳細

Algonomix6 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。Algonomix6 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。

Algonomix6 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 2-6-1 のように変更します。

リスト 2-6-1.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 2-6-1 のようなプロジェクト設定画面が表示されます。

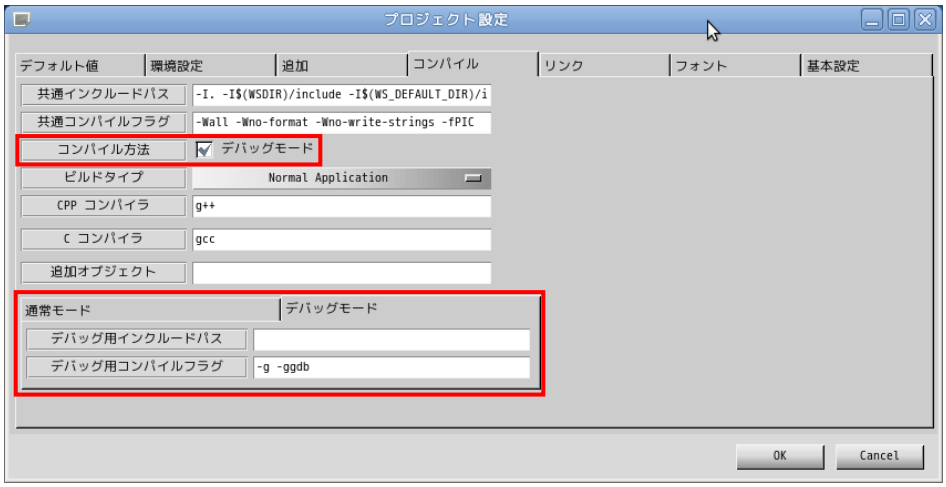


図 2-6-1. プロジェクト設定画面（コンパイルタブ）

「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

Algonomix6 で「gdbserver」を起動します。『3-3-8 ファイル転送』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、Algonomix6 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 externt permitted by law. Type “show copying”
and “show warrantry” 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。Algonomix6 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。比較的よく使うと思われるコマンドについて表 2-6-1 に示します。

表 2-6-1. 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	デバッガを終了します。

２－７　　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 Debian9.0 をインストールし、Algonomix6 の開発環境を構築する方法について説明します。

２－７－１　Debian9.0 のインストール

Debian9.0 をインストールするパソコンの推奨スペックを表 2-7-1-1 に示します。

表 2-7-1-1. Debian9.0 インストール必須パソコンスペック

CPU	Pentium4 1GHz 以上
メモリ	256MByte 以上　(1GByte 以上推奨)
HDD 空き領域	10GByte 以上

Algonomix6 では Debian9.0 を使用しています。バージョンが違うと不整合が起こる可能性がありますので、弊社で Algonomix6 を構築するときに使用した、Debian9.0 のインストールディスクを配布します。ネットワークアップデート等は利用しないようにしてください。

下記に示す Debian9.0 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 「Debian9.0 DVD1」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 2-7-1-1 のようなインストールメニュー画面が表示されます。
「Graphical Install」を選択して「Enter」キーを押します。

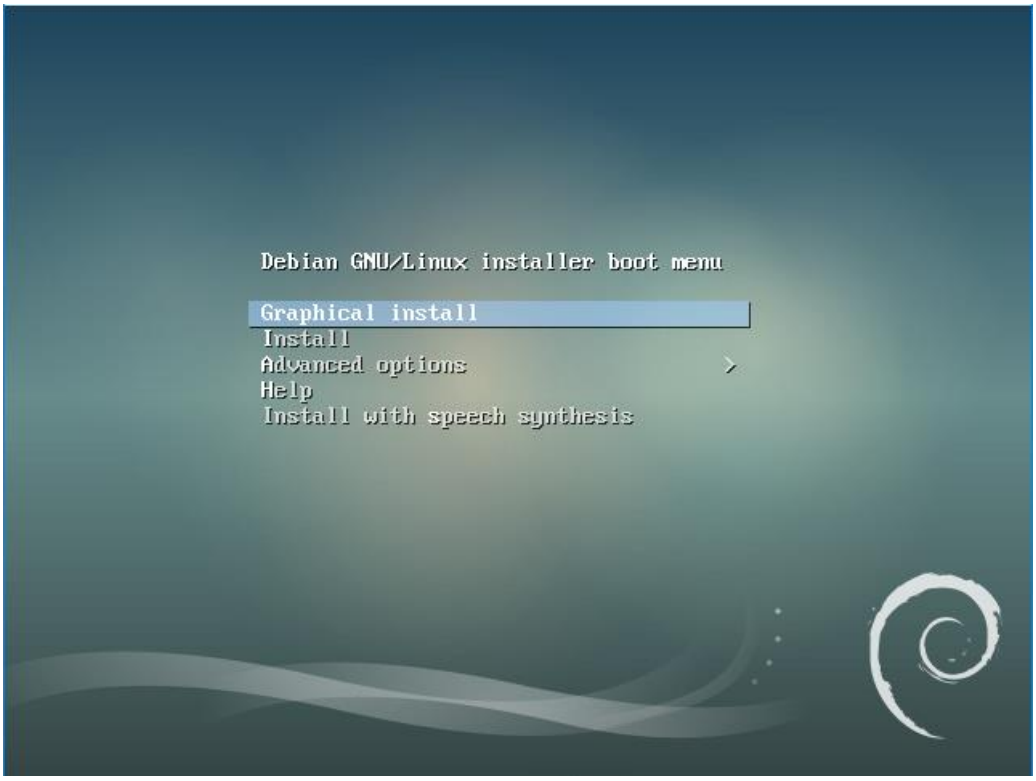


図 2-7-1-1. Debian9.0 インストールメニュー

③　図 2-7-1-2 のような言語選択画面が表示されます。ここでは実際にインストールする Debian の言語を選択します。カーソルキーで言語を選んで「Continue」ボタンをクリックします。

図 2-7-1-2. インストールする Debian の言語選択画面

④　図 2-7-1-3 のような場所の選択画面が表示されます。場所を選択して、[続ける]をクリックしてください。

図 2-7-1-3. 場所の選択画面

⑤　図 2-7-1-4 のようなキーボード設定画面が表示されます。ご使用のキーボードをにあったものを選択して「続ける」をクリックします。基本的なコンポーネントのインストールが始まります。

図 2-7-1-4. キーボード設定画面

⑥　基本的なコンポーネントのインストールが完了し、ネットワークの自動設定が行われた後、図 2-7-1-5 のようなホスト名入力画面が表示されます。ホスト名を入力して「続ける」をクリックしてください。ホスト名のデフォルト設定は「asdhost」としています。

図 2-7-1-5. ホスト名設定画面

⑦　図 2-7-1-6 のような、ドメイン名設定画面が表示されます。ドメイン名を設定される場合は、ここで設定してください。デフォルト設定ではドメイン名は設定しませんので、空白のまま、「続ける」をクリックしてください。

図 2-7-1-6.　キーボードレイアウト設定画面

⑧　図 2-7-1-7 のような管理者権限パスワード設定画面が表示されます。
管理者権限に移行するためのパスワードを入力後、[続ける]をクリックしてください。Algonomix6 のデフォルト設定では「rootroot」と設定されています。

図 2-7-1-7.　管理者権限パスワード設定画面

⑨　図 2-7-1-8 のようなユーザ名設定画面が表示されます。デフォルト設定では「asdusr」と入力後、「続ける」をクリックしてください。

図 2-7-1-8. ユーザ名設定画面

⑩　図 2-7-1-9 のようなユーザ名選択画面が表示されます。デフォルト設定では「asdusr」と入力済みなので、「続ける」をクリックしてください。

図 2-7-1-9. ユーザ名設定画面

⑪　図 2-7-1-10 のようなユーザパスワード設定画面が表示されます。ユーザのパスワードを設定して、「続ける」をクリックしてください。

図 2-7-1-10.　ユーザパスワード設定画面

⑫　図 2-7-1-11 のようなディスクのパーティショニング設定画面が表示されます。特に問題なければ、「ディスク全体を使う」を選択して、「続ける」をクリックしてください。

図 2-7-1-11. ディスクのパーティショニング設定画面

⑬　図 2-7-1-12 のような Debian をインストールするディスク選択画面が表示されます。インストールするディスクを選択して、「続ける」をクリックしてください。

図 2-7-1-12.　ディスク選択画面

⑭　図 2-7-1-13 のようなディスクパーティショニング設定画面が表示されます。特に問題なければ、「すべてのファイルを 1 つのパーティショニングに」を選択して、「続ける」をクリックしてください。

図 2-7-1-13.　ディスクパーティショニング設定画面

⑮　図 2-7-1-14 のようなディスクパーティショニング設定確認画面が表示されます。特に問題なければ、「パーティショニングの終了とディスクへの変更の書き込み」を選択して、「続ける」をクリックしてください。

図 2-7-1-14.　ディスクパーティショニング確認画面

⑯　図 2-7-1-15 のようなディスクパーティショニング最終確認画面が表示されます。特に問題なければ、「はい」を選択して、「続ける」をクリックしてください。ベースシステムのインストールが始まります。

図 2-7-1-15.　ディスクパーティショニング最終確認画面

⑰　インストールディスク（DVD3 枚）の登録を行います。Debian インストール後でも登録は可能です。登録する場合は、「はい」を選択して［続ける］をクリックし、画面の指示に従ってください。登録しない場合は「いいえ」を選択して「続ける」をクリックしてください。

図 2-7-1-16. インストールディスクの登録

⑱　図 2-7-1-17 のようなネットワークミラーの使用を確認する画面になります。自動的にアップデートされてしまう可能性があるので「いいえ」を選択して、「続ける」をクリックします。

図 2-7-1-17. ネットワークミラー使用確認

⑲　図 2-7-1-18 のようなパッケージ利用調査の参加を確認する画面になります。参加したくない場合は「いいえ」を選択して、「続ける」をクリックします。

図 2-7-1-18. ネットワークミラー使用確認

⑳　図 2-7-1-19 のようなインストールソフトウェア選択画面になります。図のとおりに設定して、「続ける」をクリックします。パッケージのインストールが開始されます。

図 2-7-1-19. インストールソフトウェア選択画面

㉑ 図 2-7-1-20 のような GRUB ブートローダインストール確認画面になります。「はい」を選択肢、「続ける」をクリックします。

図 2-7-1-20. GRUB ブートローダインストール確認画面

㉒ 図 2-7-1-21 のような GRUB ブートローダインストール先設定画面になります。インストール先を設定後、「続ける」をクリックします。

図 2-7-1-21. GRUB ブートローダインストール先設定画面

㉓ 正常にインストールが完了することで、図 2-7-1-22 のような画面が表示されます。DVD-ROM を取り出して「続ける」をクリックしてください。パソコンが再起動します。

図 2-7-1-22. インストール完了画面

㉔ 再起動されると、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。(自動的にログインするを選択していても、初回起動時はパスワードの入力が必要です)

２－７－２　開発環境構築前準備

Algonomix6 用の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　コンソールを起動します。

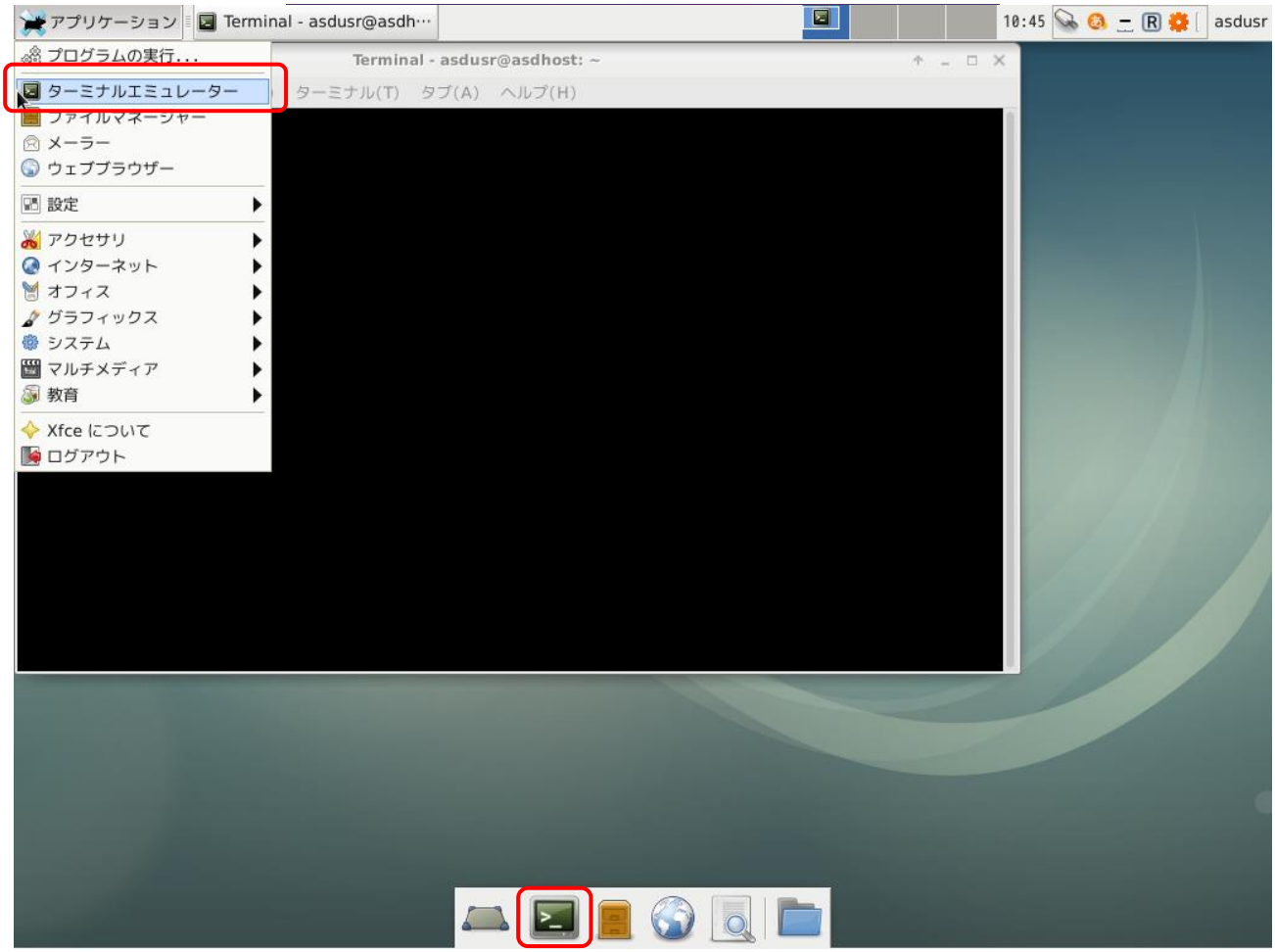


図 2-7-2-1. コンソールの起動

②　パッケージのイントールを行いますので、管理者権限に移行します。「su」コマンドを実行し、「2-7-1 ⑧」で設定した、管理者権限のパスワードを入力してください。

```
$ su
パスワード：rootroot          ←インストール時に設定した管理者権限パスワード
#
```

③　インストール時に DVD-ROM の登録をしなかった場合、以下のように登録します。画面の指示に従い、追加するディスクをドライブに入れて「Enter」キーを押します。Debian9.0 では DVD3 枚です。DVD1 はすでに登録済みなので、DVD2 と DVD3 を登録してください。

```
# apt-cdrom add
CD-ROM マウントポイント /media/cdrom/ を使用します
CD-ROM をアンマウントしています ...
ディスクを待っています ...
ディスクをドライブに入れて [Enter] キーを押してください
#
```

④　DVD を追加した後、apt パッケージの更新を行うため、以下のコマンドを実行します。

```
# apt-get update
# apt-get upgrade
#
```

⑤　「vim」パッケージをインストールします。これは、vi コマンドを実行したときのテキストの視認性を高めます。

```
# apt-get install vim
#
```

⑥　ネットワークサーバ名を変更します。Debian9.0 では、ネットワークデバイス名が「eth0」ではなく、マックアドレスを含んだものに変更されています。この場合、デバイス名の固定化が難しいため、下記のように Kernel の起動オプションを変更します。「/etc/devault/grub」ファイルを開き、「GRUB_CMDLINE_LINUX」に起動コマンドを追加します。その後、「update-grub」コマンドで grub をアップデートします。再起動すると、ネットワークデバイス名がいままでの「eth0」に戻ります。

```
# vi /etc/default/grub
    GRUB_CMDLINE_LINUX="net.ifnames=0 biosdevnames=0"
# update-grub
# reboot
#
```

⑦　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 2-7-2-1 に示します。「apt-get install XXX」で順次インストールしてください。

表 2-7-2-1. 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	4.1-9.1	コンパイルを制御するユーティリティ
gcc	4:6.3.0-4	GNU C コンパイラ
g++	4:6.3.0-4	GNU C++ コンパイラ
libncurses5-dev	6.0+20161126-1	developer's libraries for ncurses
git	1:2.11.0-3+deb9u1	速く、スケーラブルな分散型リビジョン管理システム
bc	1.06.95-9	GNU bc: 任意精度の計算言語
dc	1.06.95-9	GNU dc - 任意精度の逆ポーランド式計算器
device-tree-compiler	1.4.2-1	Device Tree Compiler for Flat Device Trees
libjpeg-dev	1:1.5.1-2	Development files for the JPEG library
libpng-dev	1.6.28-1	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	2:1.6.4-3	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	2:1.3.3-1	X11 用の雑多な拡張ライブラリ (開発用ヘッダ)
libxft-dev	2.3.2-1	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	5.24.1-3+deb9u1	Perl ライブラリ: 開発用ファイル
gucharmap	1:9.0.2-1	Unicode 文字ピッカおよびフォントブラウザ

```
# 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
#
```

２－７－３　Algonomix6 用開発環境インストール

Debian9.0 に Algonomix6 用の開発環境をインストールします。Algonomix6 用開発環境 SD カードに格納されている、「Algonomix6-tools-x64.tar.gz」と「Algonomix6-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 Algonomix6 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 Algonomix6 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。

```
$ su
パスワード：rootroot          ←インストール時に設定した管理者権限パスワード
# tar zxvf /media/asdusr/デバイス名/development/Algonomix6-tools-x64.tar.gz –C /
```

これで、Algonomix6 用の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-x64」というディレクトリが作成されます。

④ 引き続き ARM 用の開発環境を展開します。

```
# tar zxvf /media/asdusr/デバイス名/development/Algonomix6-tools-arm64.tar.gz –C /
```

これで、Algonomix6 用の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。

⑤ 正常に展開が終了されると「/usr/local/tools-arm64」というディレクトリが作成されます。

⑥ ARM64bitCPU 用の開発環境パッケージを展開します。

```
# cd /usr/local/tools-arm64/toolchain
# tar zxvf aarch64-linux-gnu-lib.tar.gz -C /
# tar zxvf aarch64-linux-gnu-usrlib.tar.gz -C /
# cd /usr/local/tools-arm64/toolchain
# cd gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/aarch64-linux-gnu/include
# tar zxvf /usr/local/tools-arm64/toolchain/aarch64-linux-gnu-inc.tar.gz
# cd gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/aarch64-linux-gnu
# tar zxvf /usr/local/tools-arm64/toolchain/aarch64-linux-gnu-usrinc.tar.gz
```

⑦ 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-x64/source/widestudio/ws-v3.98-7-custom/src
# make install
# ldconfig
```

⑧ WideStudio 実行用スクリプトをコピーします。

```
# cp /usr/local/tools-x64/source/widestudio/wsstart.sh /usr/local/bin
```

⑨　デスクトップ画面に WideStudio 実行用スクリプトのショートカットを登録します。デスクトップ上で右クリックして、「ランチャーの作成」をクリックしてください。図 2-7-3-1 のように、名前に「WideStudio」コマンドに「/usr/local/bin/wsstart.sh」を設定してください。アイコンは適当に選択してください。「作成」をクリックすると、デスクトップ上に WindowsStudio の起動ショートカットが作成されます。

図 2-7-3-1. WideStudio 起動ショートカット

⑩　ダブルクリックして WideStudio が起動されることを確認してください。

２－７－４　Windows 共有の設定

Windows PC から Algonomix6 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 パッケージのイントールを行いますので、管理者権限に移行します。「su」コマンドを実行し、管理者権限のパスワードを入力してください。デフォルト設定では「rootroot」となっています。

```
$ su
パスワード：rootroot          ←インストール時に設定した管理者権限パスワード
#
```

② 「/opt/share」ディレクトリを作成します。

```
# mkdir /opt/share
# chmod 777 /opt/share
```

③ 「samba」サーバパッケージをインストールします。

```
# apt-get install samba
#
```

④ 「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]の項目に 3 つの設定を行います。[x64_share] と[arm64_share]の共有フォルダを設定します。

```
# vi /etc/samba/smb.conf
[global]
   interfaces = 127.0.0.0/8 eth0            # <= 接続許可するインターフェースを指定
   bind interfaces only = yes               # <= interfaces 項目に指定したホストだけに接続を限定する

   map to guest = bad user                  # <= 未登録アカウントで接続された場合にゲストとして扱う
[opt_share]                                 # <= 共有名
   path           = /opt/share              # <= 共有ディレクトリ
   writable       = yes                     # <= 書込可
   guest ok       = yes                     # <= ゲストユーザー可
   guest only     = yes                     # <= 全てゲストとして扱う
   create mode    = 0777                    # <= ファイル作成はフル権限で
   directory mode = 0777                    # <= ディレクトリ作成はフル権限で
   share modes    = yes                     # <= 同一ファイルに同時アクセス時に警告
[x64_share]                                 # <= 共有名
   path           = /usr/local/tools-x64    # <= 共有ディレクトリ
   writable       = yes                     # <= 書込可
   guest ok       = yes                     # <= ゲストユーザー可
   guest only     = yes                     # <= 全てゲストとして扱う
   create mode    = 0777                    # <= ファイル作成はフル権限で
   directory mode = 0777                    # <= ディレクトリ作成はフル権限で
   share modes    = yes                     # <= 同一ファイルに同時アクセス時に警告
[arm_share]                                 # <= 共有名
   path           = /usr/local/tools-arm64  # <= 共有ディレクトリ
   writable       = yes                     # <= 書込可
   guest ok       = yes                     # <= ゲストユーザー可
   guest only     = yes                     # <= 全てゲストとして扱う
   create mode    = 0777                    # <= ファイル作成はフル権限で
   directory mode = 0777                    # <= ディレクトリ作成はフル権限で
   share modes    = yes                     # <= 同一ファイルに同時アクセス時に警告
#
```

⑤　「samba」サーバを再起動して、設定を有効化します。

```
# systemctl restart smbd
# systemctl restart nmbd
#
```

⑥　WindowsPC から Algonomix6 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 2-7-4-1 のような、ディレクトリが 3 つ見えていれば成功です。

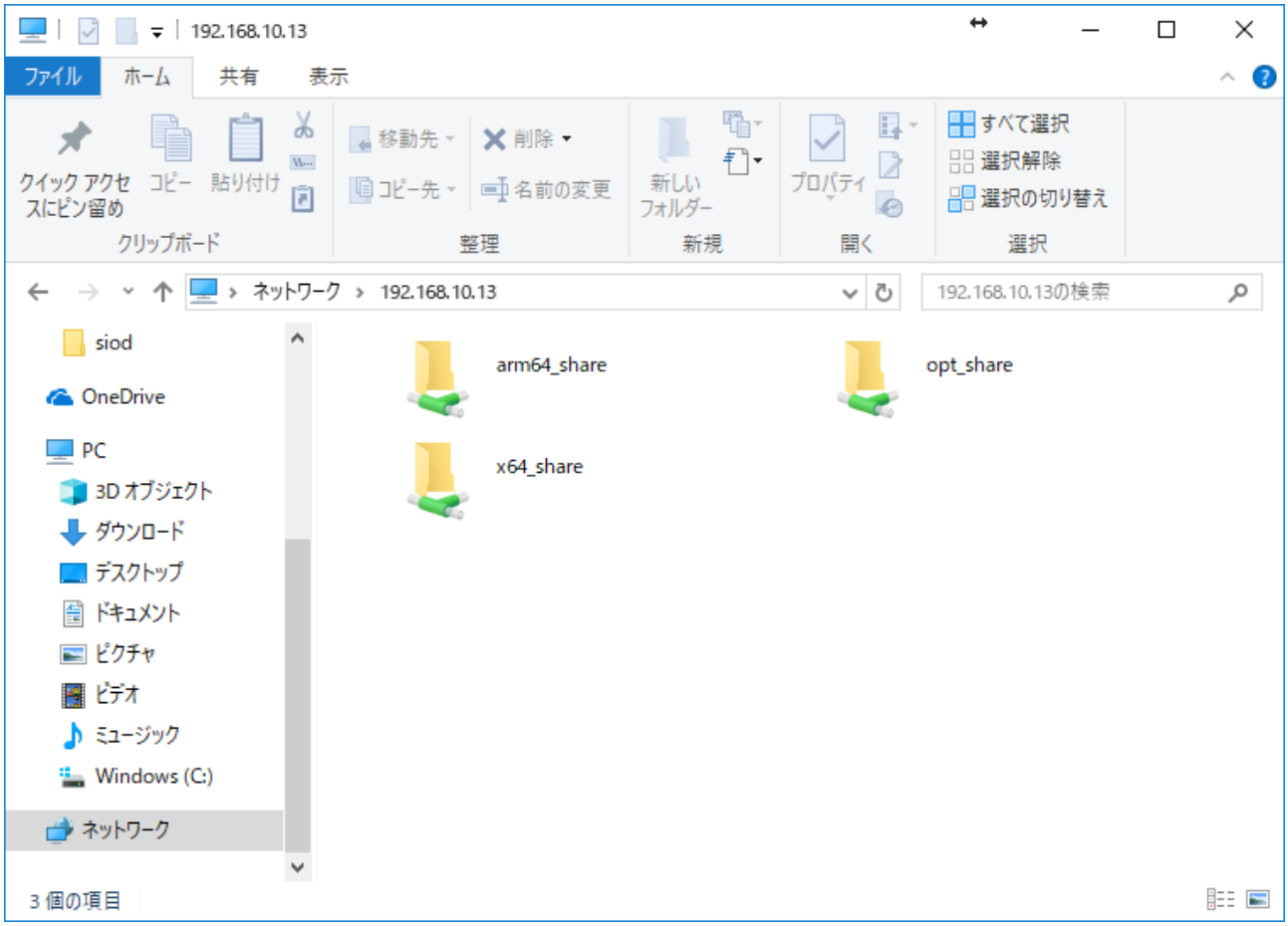


図 2-7-4-1. Windows 共有ディレクトリ

以上でパソコン上に Algonomix6 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

●「ふつうの 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付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。

７７Ｇ０５０００１Ａ　２０１７年　８月　初版
７７Ｇ０５０００１Ｈ　２０２０年　１月　第８版

ALGO 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