Seedsware

EM Series

Panel Computer

With Embedded Linux OS

Software Development Manual

Seedsware Corporation
http://www.seedsware.co.jp/global/

17A4A5-00018E-7

Introduction

This document describes the development process of Linux applications for the EM Series of product, as well as application specifications.

This document describes working with the following models:

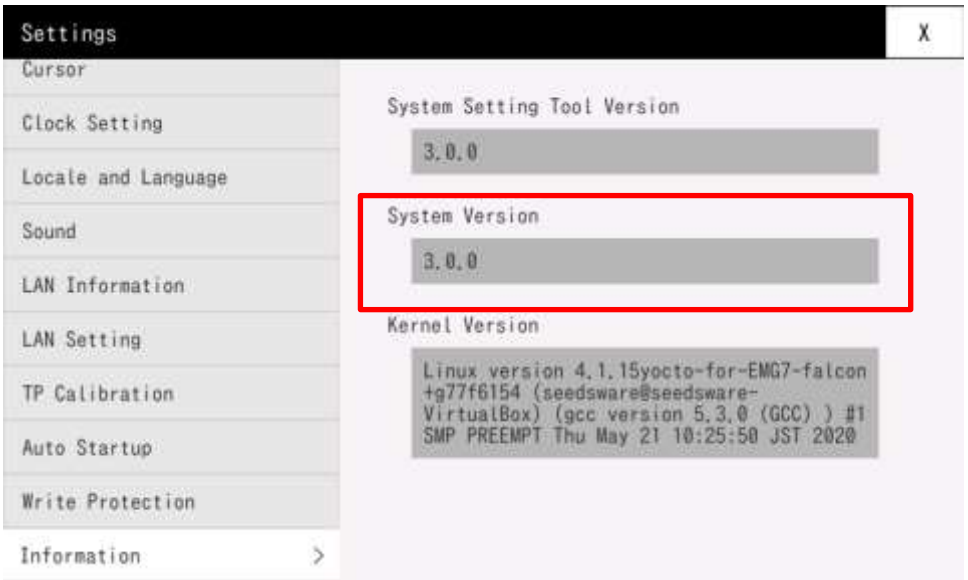
Model	Abbreviated model names	
EMG7-W207A8-00**-*07	EMG7-A8	EMG7-7W
EMG7-312A8-00**-*07		EMG7-12
EM8-W104A7-00**-*07	EM(G)8-A7	EM(G)8-4
EMG8-W104A7-00**-*07		
EM8-205A7-00**-*07		EM(G)8-5
EMG8-205A7-00**-*07		
EM8-W207A7-00**-*07		EM(G)8-7W
EMG8-W207A7-00**-*07		
EM8-W310A7-00**-*07		EM8-10W

The following system version products are targeted.

Target system version	3.0.0 ~

Check the system version of the EM series from the system setting tool.

For the system setting tool, refer to the attached "EM Series Tool Manual".



Copyright and Trademarks

Table of Contents

1 About the Development Environment

The development environment for this machine (hereafter referred to as “EM-Linux”) is provided as Oracle VM VirtualBox virtual machine (hereafter referred to as “virtual machine). The virtual machine is preconfigured with a cross compiler and other applications.

1.1 Configuring the Development Environment

1.1.1 Requirements

Equipment

Items	Description
PC (Windows® 10)	It is necessary to have the ability to operate a virtual machine with Oracle VM VirtualBox. (Reference performance) Oracle VM VirtualBox 6.1 CPU: Intel® Core™ i5 or above RAM: 4GB or more * Operations have been verified using this environment.
LAN cable	Used for data transmission. Cross cable or a HUB and straight cable

Data

Items	Description
EM-Linux development environment image (EM-Linux-App.vdi) * It is compressed using 7-Zip and stored with the file name "EM-Linux-App.7z".	Development environment virtual machine (Image of Oracle VM VirtualBox)
Linux SDK	Linux cross toolchain Rootfs image * Pre-installed with EM-Linux development environment image
USB device driver	Driver to use the USB device of the EM series as USB-Ether

Applications

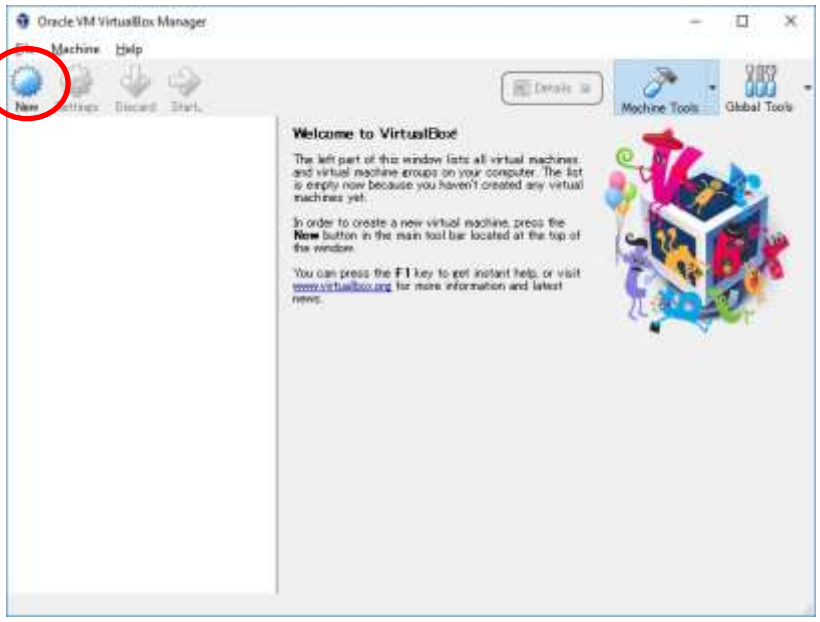
Items	Description
Oracle VM VirtualBox installer	Version: 6.1 or later
Terminal emulator	Used for console connections. Supports SSH connections. *This document uses Tera Term4.84 in its examples.
FTP client	Used for data transmission. Supports SFTP connections. *This document uses FileZilla 3.9.0.2 in its examples.

1.1.2 Registering the Development Environment

On your PC, install Oracle VM VirtualBox 6.1 or later.

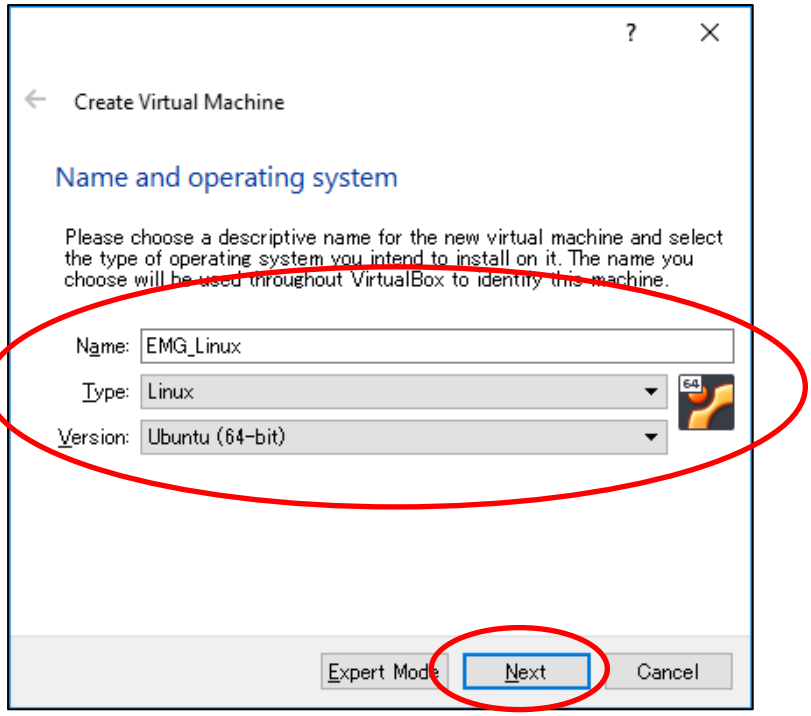
Launch the VirtualBox Manager.

Click [New].



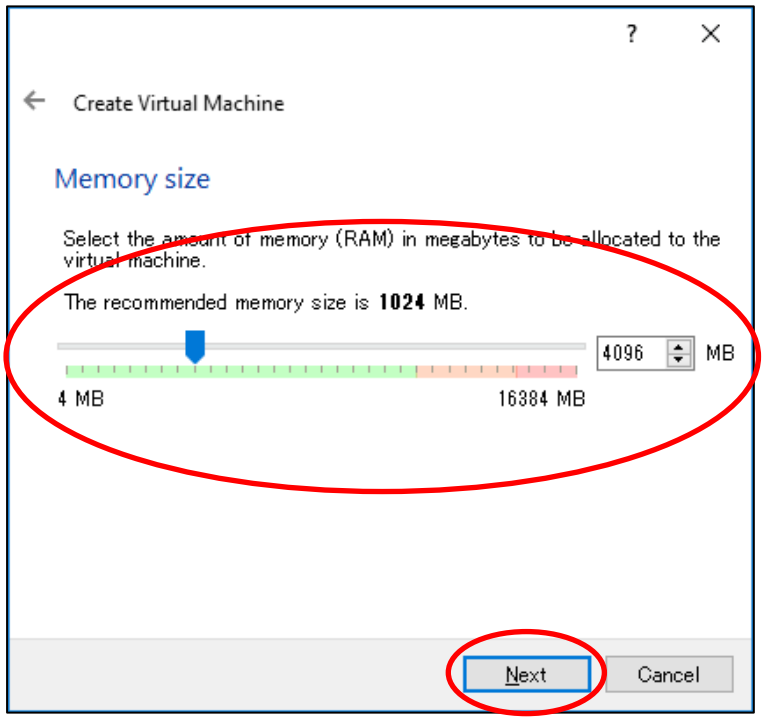
Specify the Name, Type, and Version as shown below, and click [Next].

* You can set the name as desired.



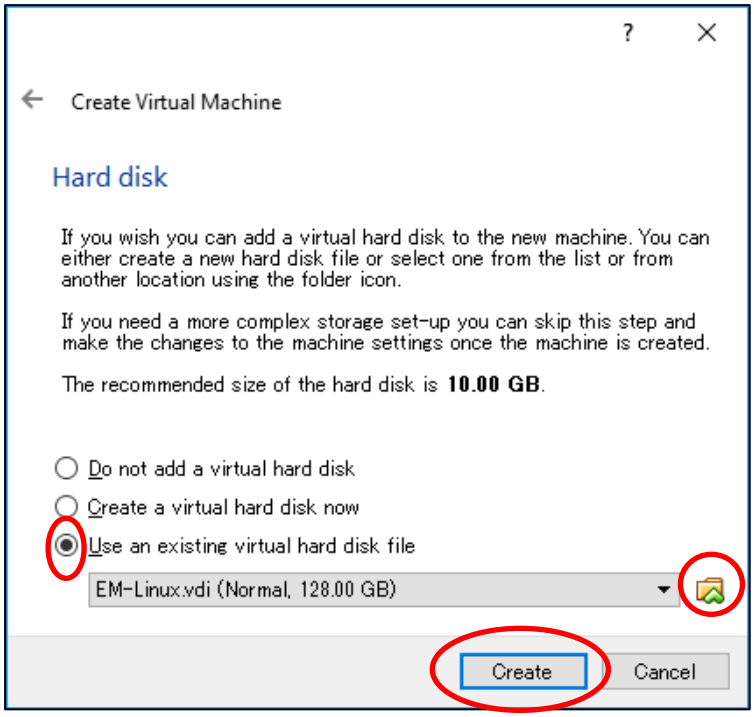
Specify the amount of memory for the virtual machine according to your PC, and click [Next].

* 2GB or more is recommended (4GB or more for the PC RAM memory)

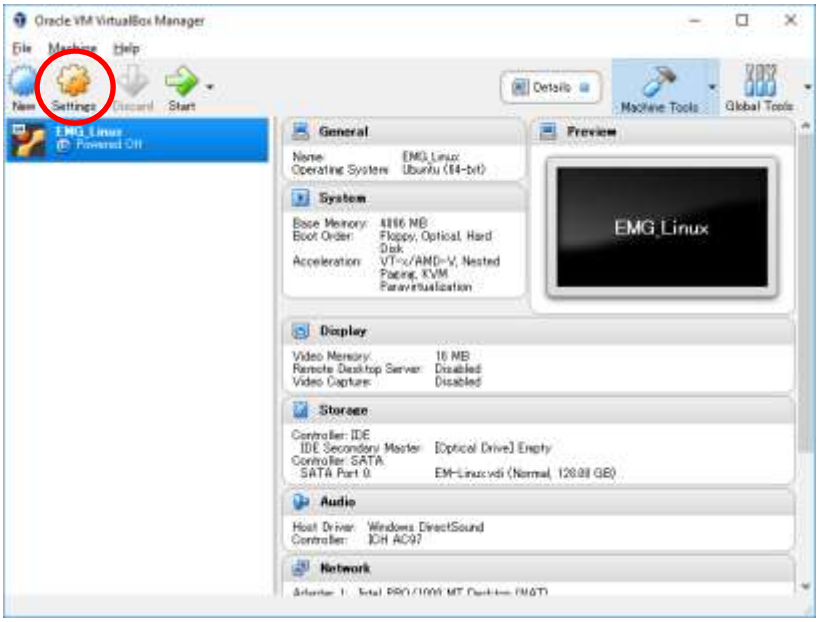


Select the [Use an existing virtual hard disk file] option, click ar after you select the data image "EM-Linux-em*.vdi" for the VirtualBox virtual machine, click [Create].

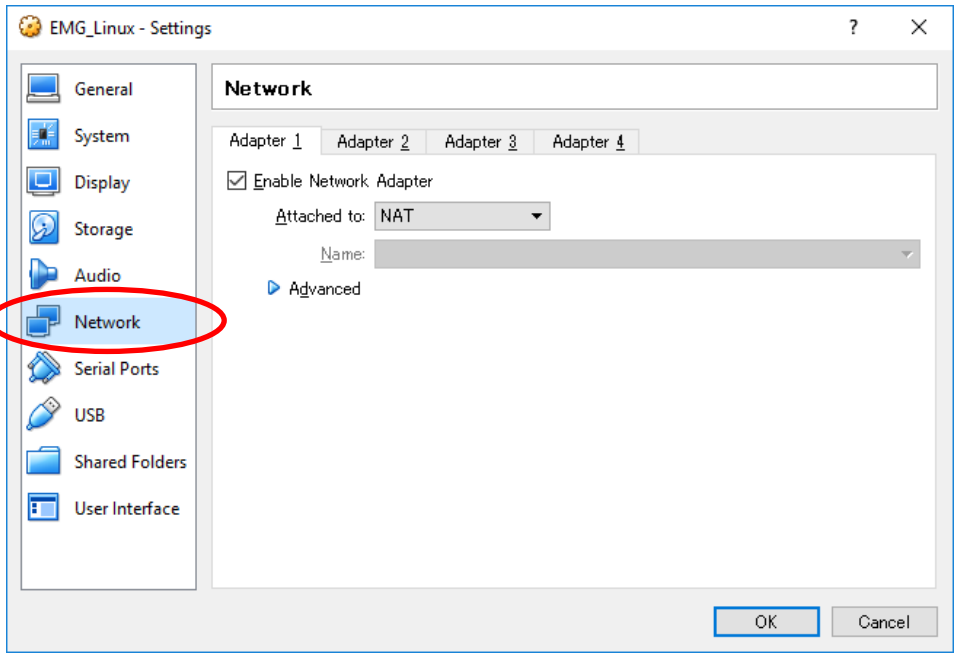
* The data image of VirtualBox virtual machine is stored with the file name of "EM-Linux-App.7z" in the "software"-"vmimage" folder in the DVD-ROM (VMimage). (7-Zip is used for compression. Please decompress and use)



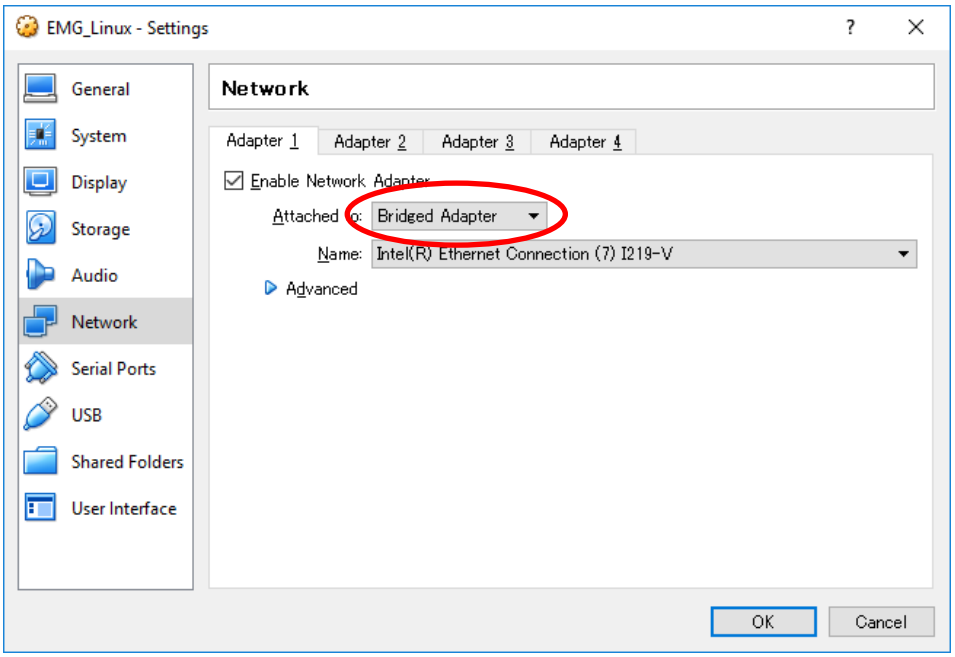
Click [Settings].



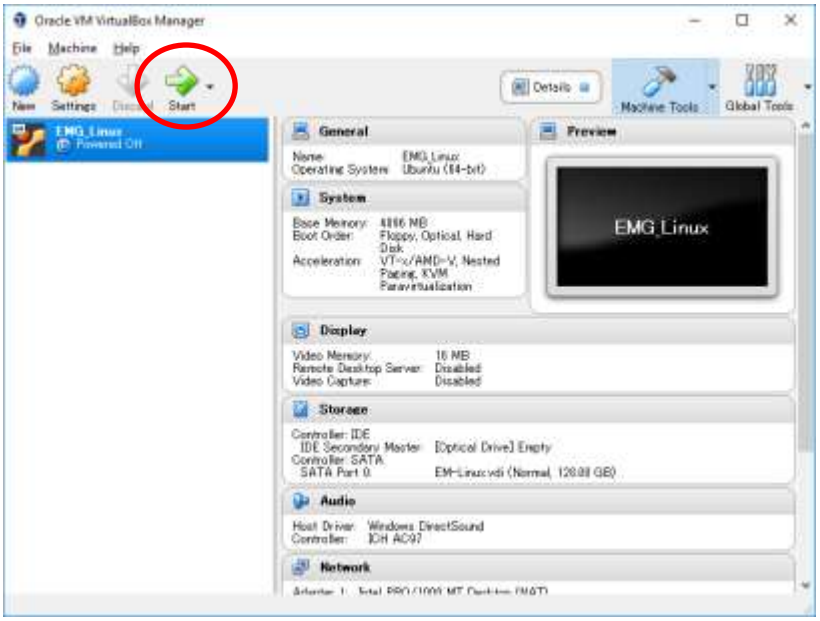
Click [Network].



Change the settings to match your environment.



Click [Start] to launch the virtual machine.



1.1.3 Development Environment Settings (Optional)

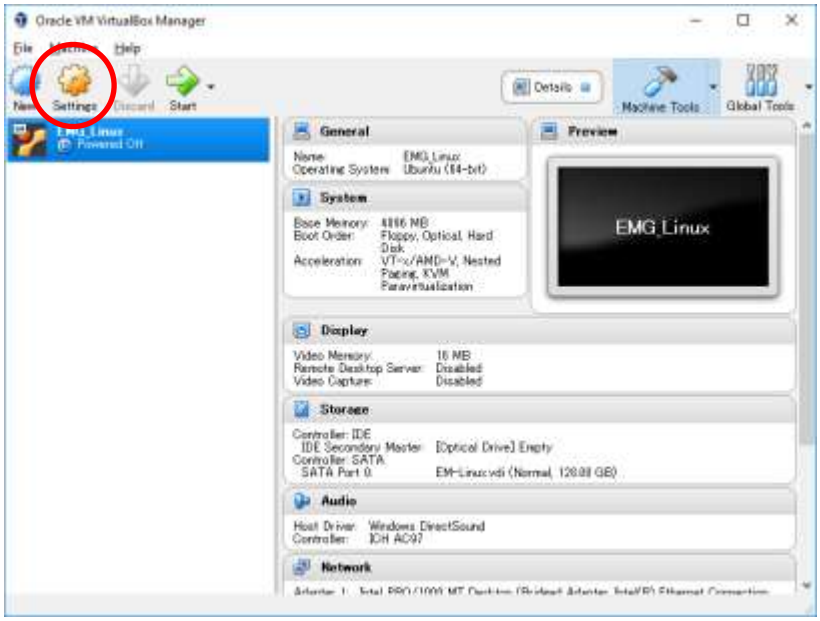
The following settings simplify development. Specify the settings as needed.

If the virtual machine is already launched, stop it before proceeding.

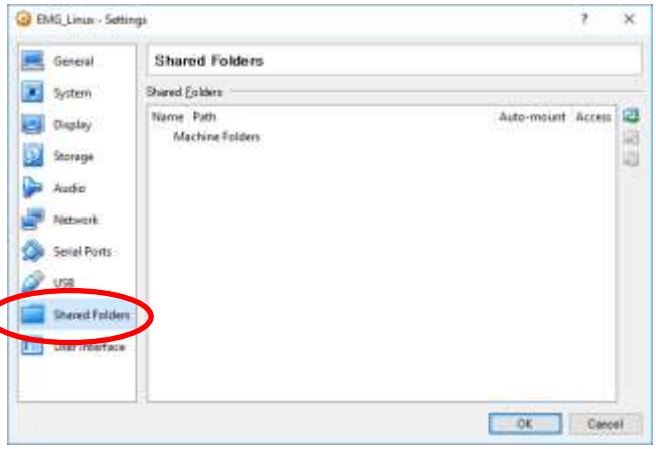
Sharing Folders

You can setup shared folders to move files between the host PC (Windows) and the virtual machine.

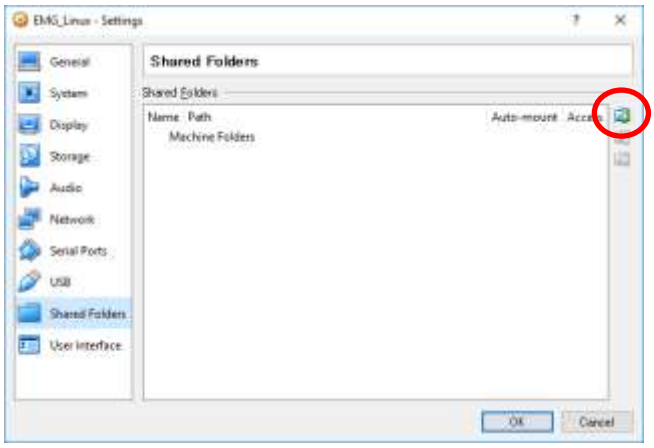
1) Click [Settings].



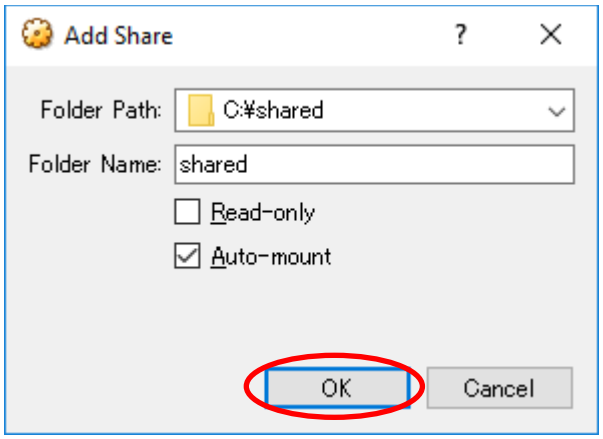
2) Click [Shared Folders].



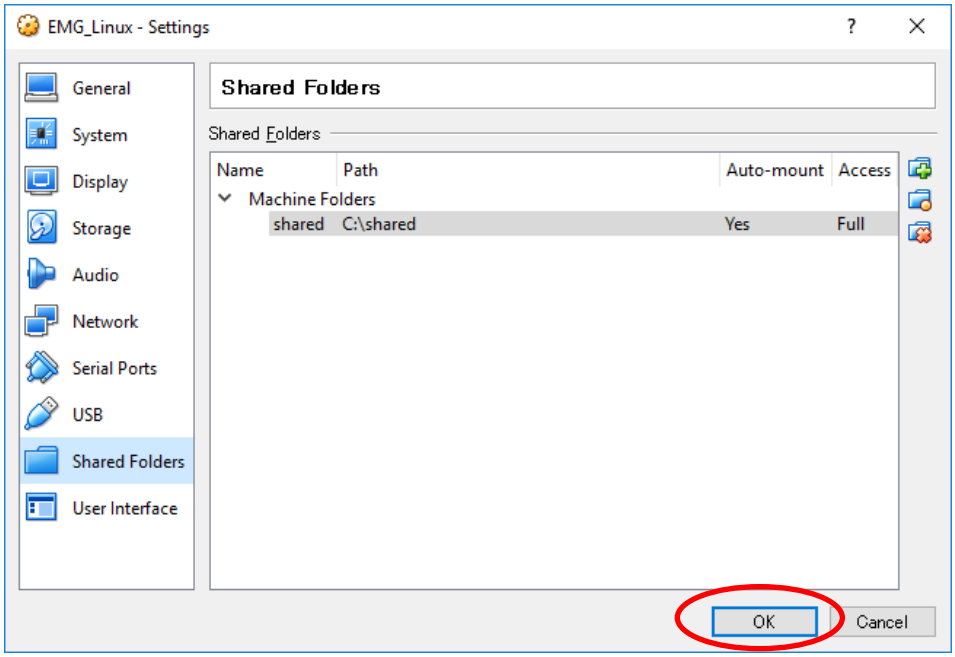
Click  .



Specify an arbitrary location for the folder path, folder name, and mount point, check the automatic mount, and click the OK button.



A shared folder is now set up. Click OK to close the window.



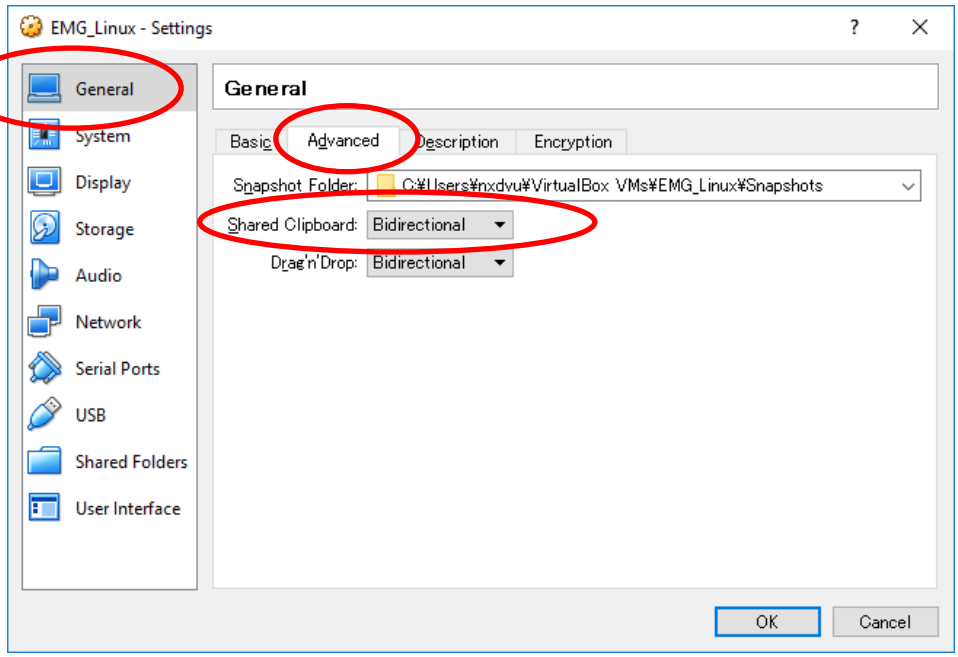
After a shared folder is set up, on the virtual machine copy a file to the shared folder and you can access the file in the shared folder from the specified host PC (Windows).

Sharing the Clipboard

You can share the clipboard to copy and paste between the host PC (Windows) and the virtual machine.

1) Click [General], then the [Advanced] tab.

From the [Shared Clipboard] list, select [Bidirectional].



1.1.4 Installing Linux SDK (Optional)

Included with the EM-Linux Development image (EM-Linux-em*.vdi). You can skip this step if you are using the EM-Linux development environment image.

1) Load the following files (SDK) from the DVD-ROM (Development Environment Kit) to the virtual machine.
   Files:
   For EMG7-A8
   - poky-glibc-x86_64-meta-toolchain-armv7a-neon-toolchain-2.1.2.sh
   - poky-glibc-x86_64-meta-toolchain-qt5-armv7a-neon-toolchain-2.1.2.sh
   - poky-glibc-x86_64-em-image-mx53-armv7a-neon-toolchain-2.1.2.sh

   For EM(G)8-A7
   - poky-glibc-x86_64-meta-toolchain-cortexa7hf-neon-toolchain-2.1.2.sh
   - poky-glibc-x86_64-meta-toolchain-qt5-cortexa7hf-neon-toolchain-2.1.2.sh
   - poky-glibc-x86_64-em-image-mx6ul-cortexa7hf-neon-toolchain-2.1.2.sh

Install the Linux SDK. Launch the installation operation from the terminal on the virtual machine.

```
$ cd <location of copied files in step 1>
---- Change File Attributes ----
$ chmod a+x poky-glibc-x86_64-meta-toolchain-*****.sh
$ chmod a+x poky-glibc-x86_64-meta-toolchain-qt5-*****.sh
$ chmod a+x poky-glibc-x86_64-em-image-*****.sh
---- Install Toolchain ----
$ ./poky-glibc-x86_64-meta-toolchain-*****.sh
Enter the target directory for SDK (default: /opt/poky/2.1.2): /opt/poky/2.1.2/ “Return”
You are about to install the SDK to "/opt/poky/2.1.2/em*". Proceed[Y/n]?y ← “y”
[sudo] password for ubuntu:    ← “Login user password”
Extracting SDK...done
Setting it up...done
SDK has been successfully set up and is ready to be used.
---- Installing QT5 SDK ----
$ ./ poky-glibc-x86_64-meta-toolchain-qt5-*****.sh
Enter the target directory for SDK (default: /opt/poky/2.1.2): /opt/poky/2.1.2/ “Return”
The directory "/opt/poky/2.1.2" already contains a SDK for this architecture.
If you continue, existing files will be overwritten! Proceed[y/N]?y ← “y”
[sudo] password for ubuntu:    ← “Login user password (display depends on circumstances)”
Extracting SDK...done
Setting it up...done
SDK has been successfully set up and is ready to be used.
---- Installing QT5 rootfs ----
$ ./poky-glibc-x86_64-em-image-*****.sh
Enter the target directory for SDK (default: /opt/poky/2.1.2): /opt/poky/2.1.2/ “Return”
```

1.2 Connection between PC and EM series

Configure the settings for connecting the PC and EM series with a network.

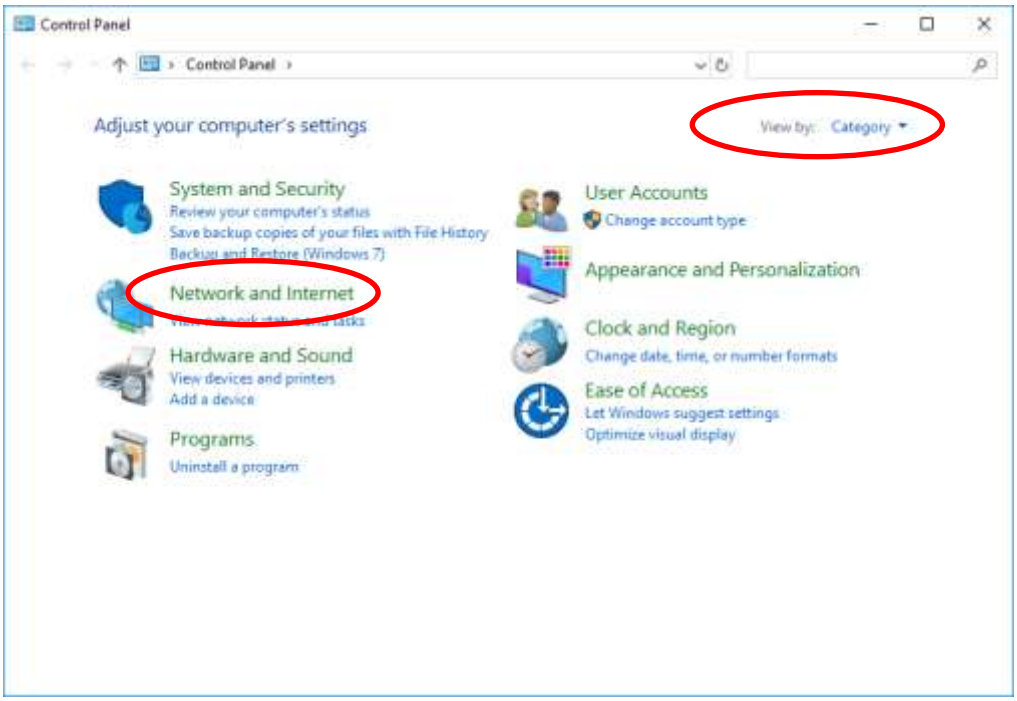
You can connect using the LAN port or USB device port.

1.2.1 Network settings on the LAN port

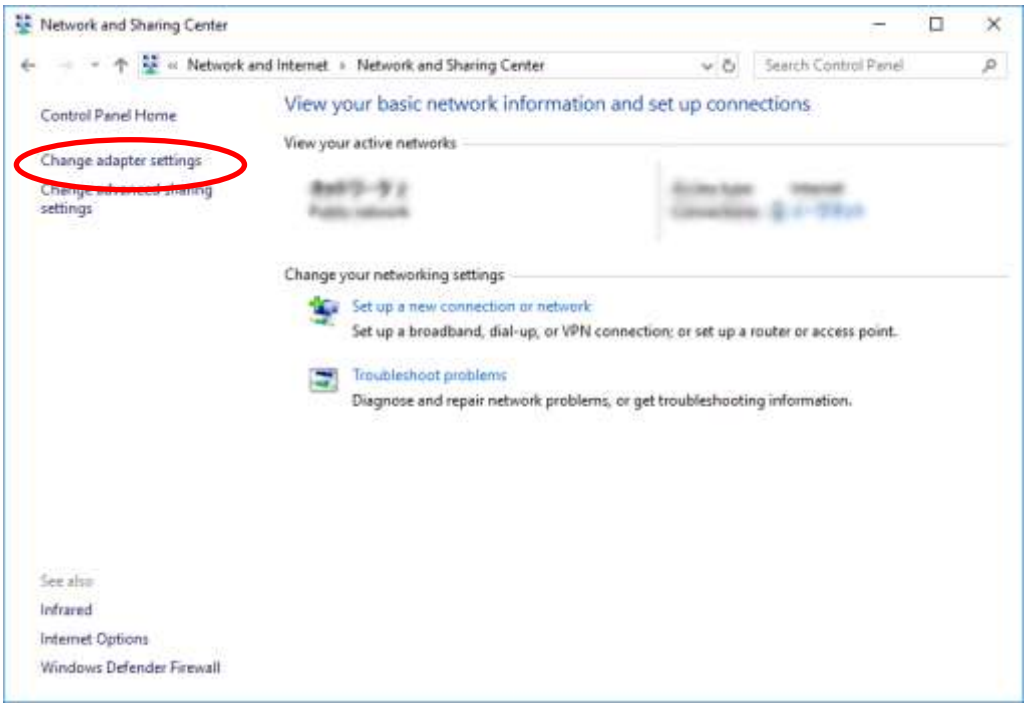
From Windows, change the network settings so that Windows can connect with EM-Linux.

1) Open the Control Panel and click [Network and Internet].

* If your screen looks different, in the top-right corner set [View by] to [Category].

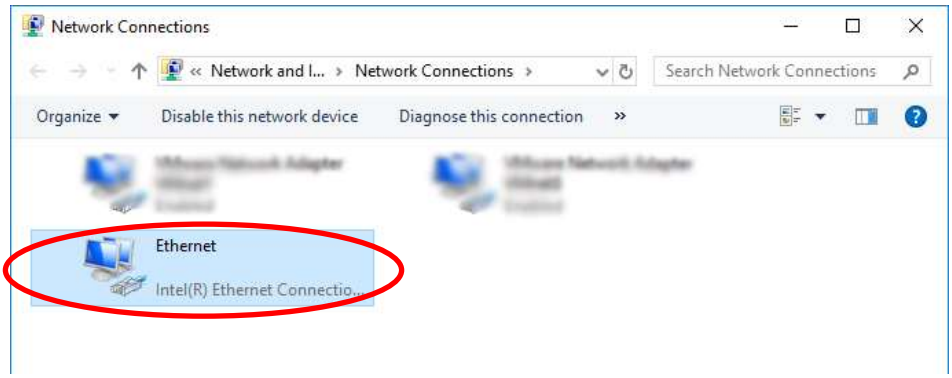


2) Click [Change adapter settings].

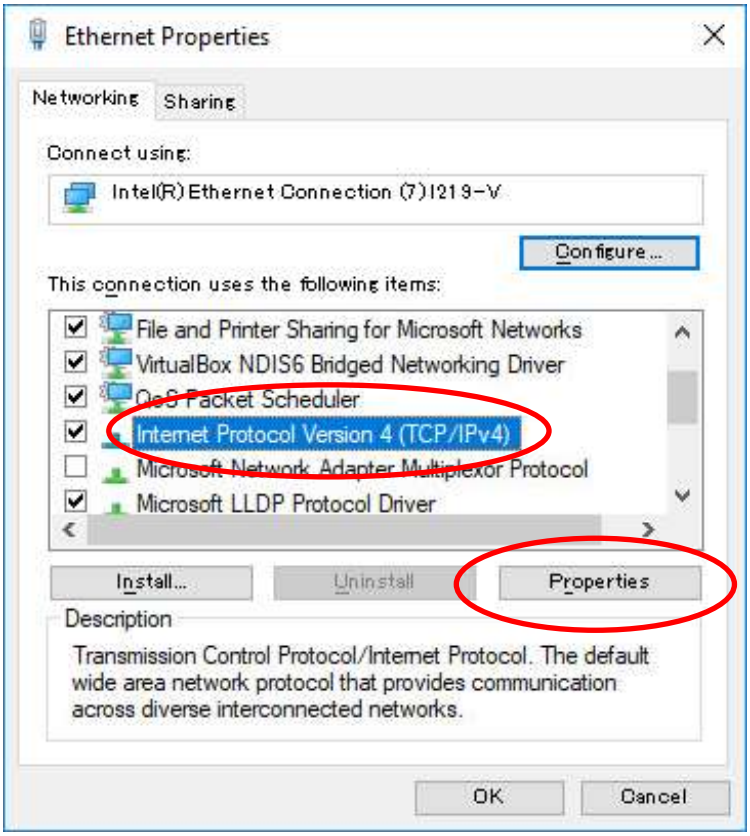


Right-click [Ethernet] and select [Properties].

* Depending on your environment, you may see a different name such as Local Area Connection. Select an adapter for the wired LAN port.



Select [Internet Protocol Version 4 (TCP/IPv4)], then click [Properties].



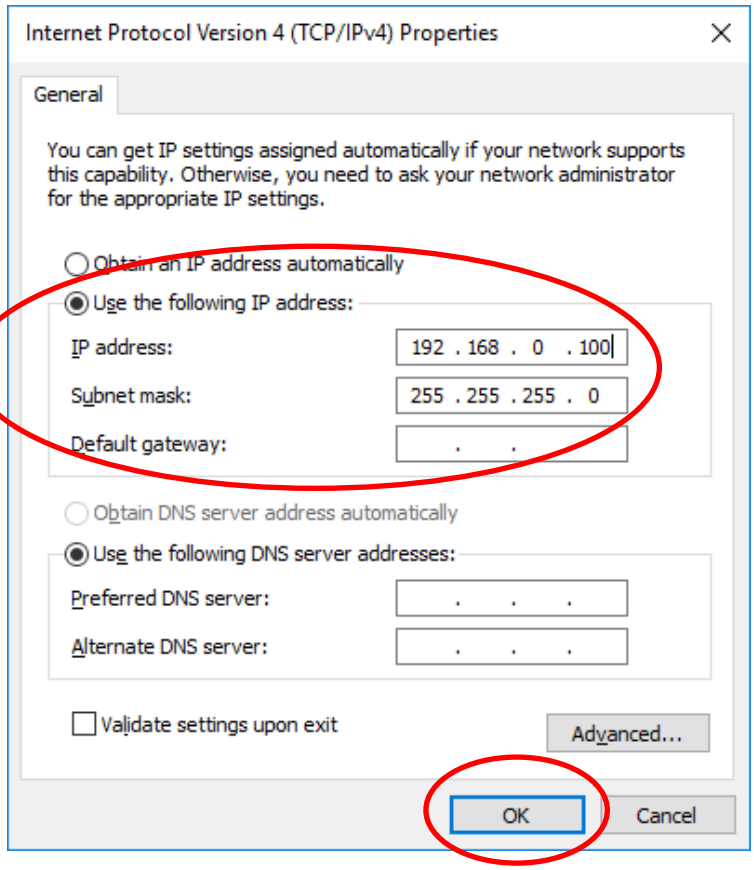
Set the IP address, sub-net mask, and default gateway, and then click OK.

This example uses the following settings:

IP address	192.168.0.100
Sub-net mask	255.255.255.0
Default gateway	not configured

* Change the settings, such as IP, to match your environment.

* "192.168.10.*" cannot be used because it is used for USB-Ether (usb0).



The Windows network settings on the host PC are now updated as defined.

1.2.2 Network settings on USB device port

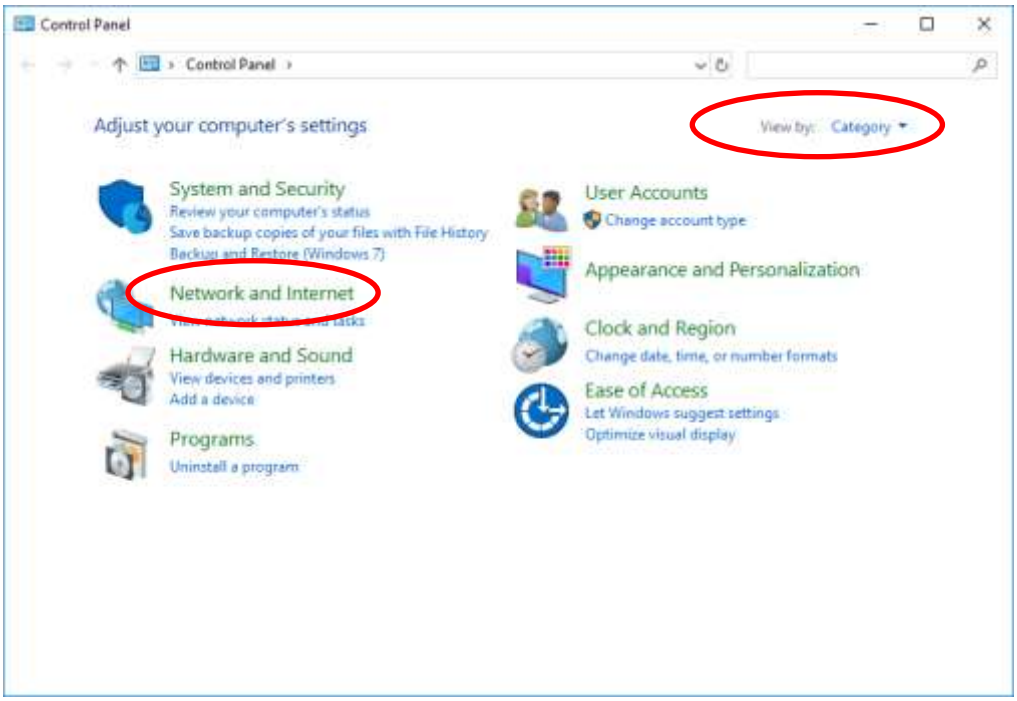
Since the USB device of the EM series main unit is used as USB-Ether, it is necessary to install the USB device driver and set the network on the PC.
With this setting, it is possible to connect the EM series main unit and the PC via LAN using the USB device port.

1) Install the USB device driver on your PC.

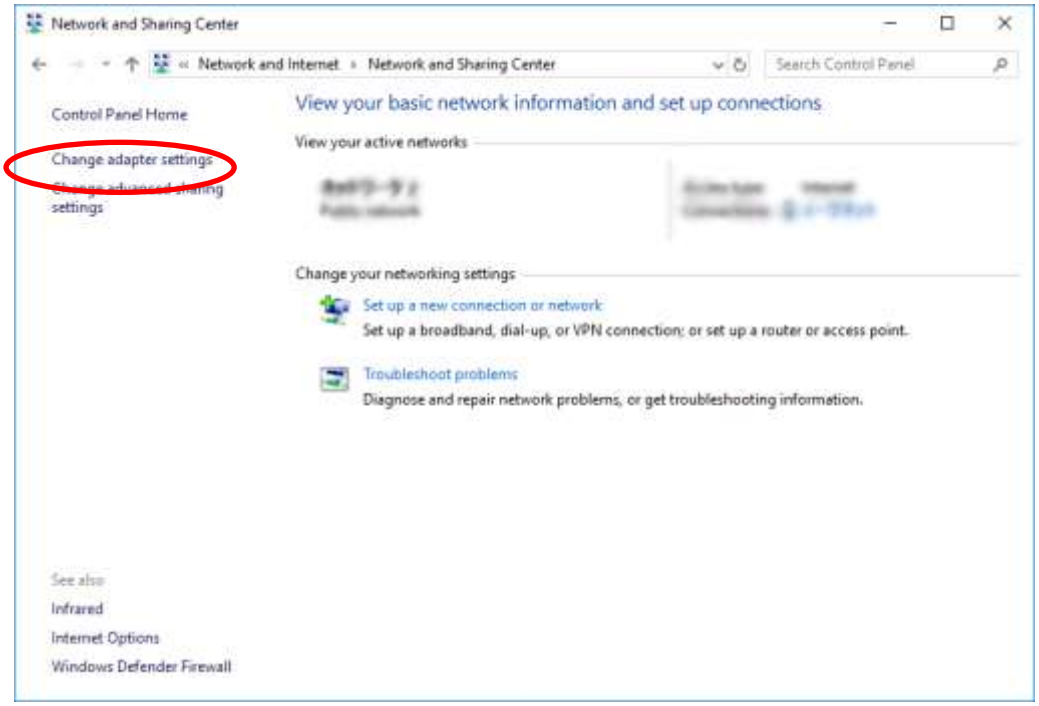
   Please execute "Install" from the right-click menu of "em_usbd.inf" in the "software"-"driver"-"em" folder in the sample kit disk.

2) Open the Control Panel and click [Network and Internet].

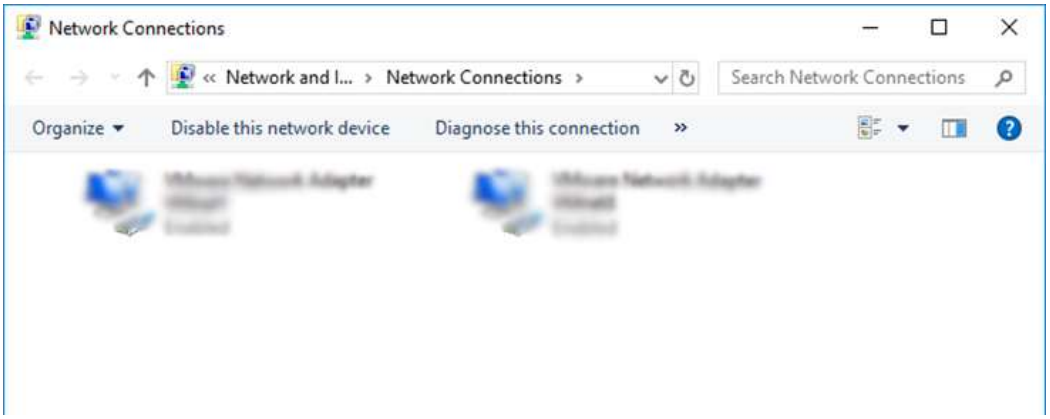
   * If your screen looks different, in the top-right corner set [View by] to [Category].



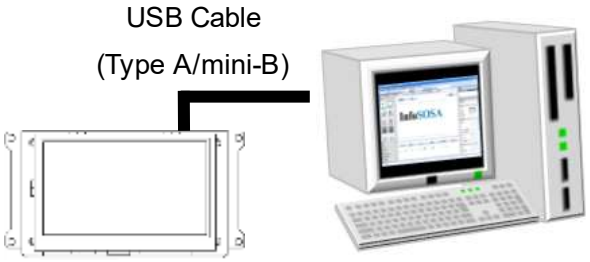
3) Click [Change adapter settings].



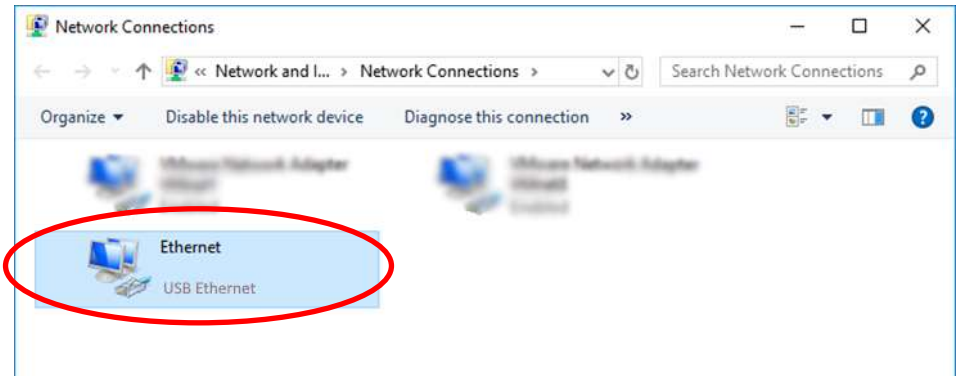
4) The [Change adapter settings] window opens.



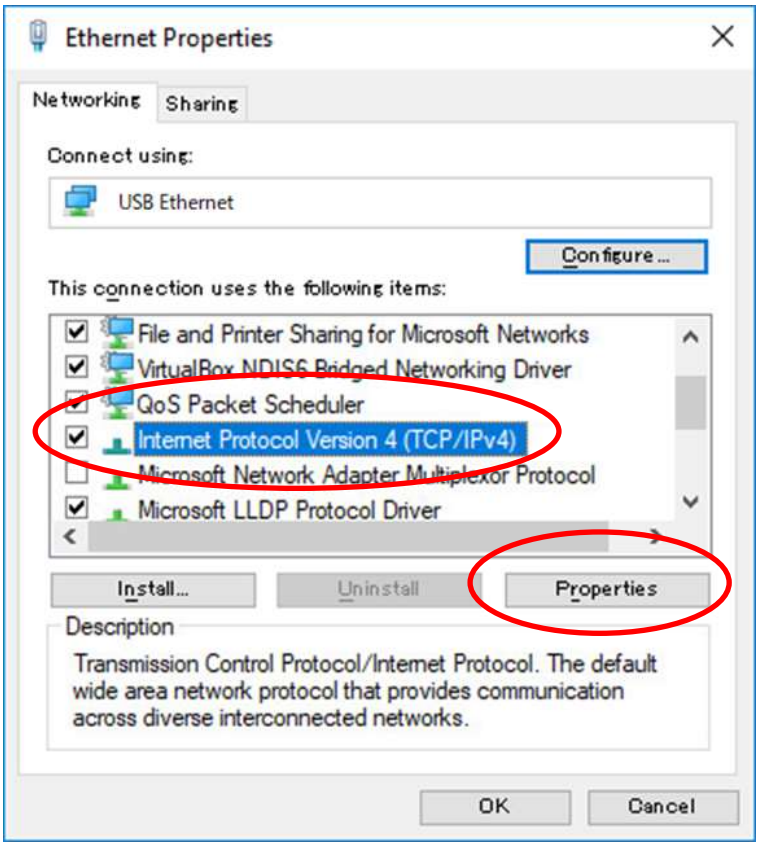
5) While the EM Series unit is on, connect to the PC with a USB cable (TypeA/mini-B).



6) Right-click the added adapter and click "Properties".

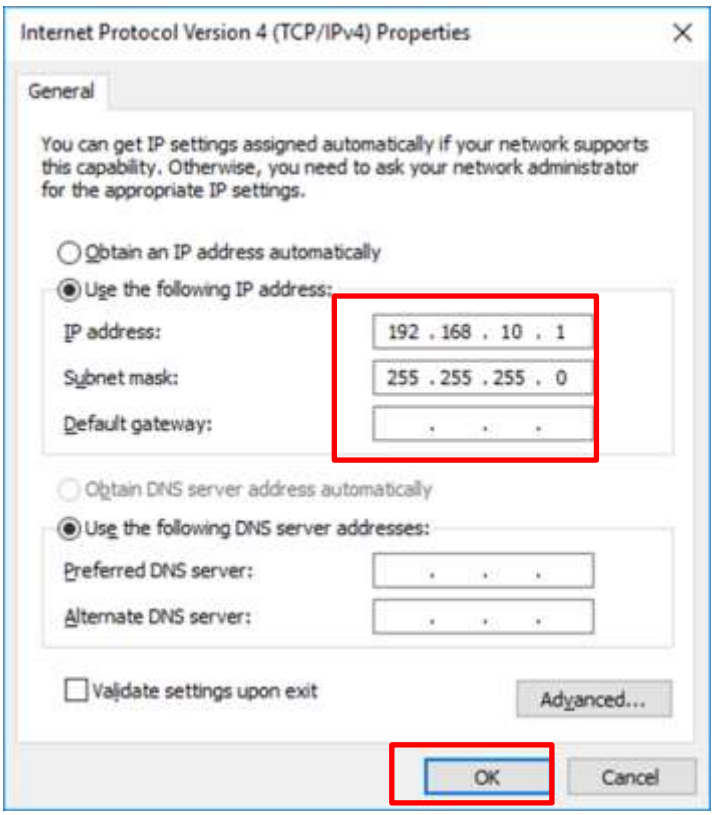


7) Select [Internet Protocol Version 4 (TCP/IPv4)], then click [Properties].



8) Set the IP address, subnet mask, and default gateway to the values below and click the OK button..

IP address	192.168.10.1
Sub-net mask	255.255.255.0
Default gateway	not configured



The Windows network settings on the host PC are now updated as defined.

1.2.3 How to connect the console

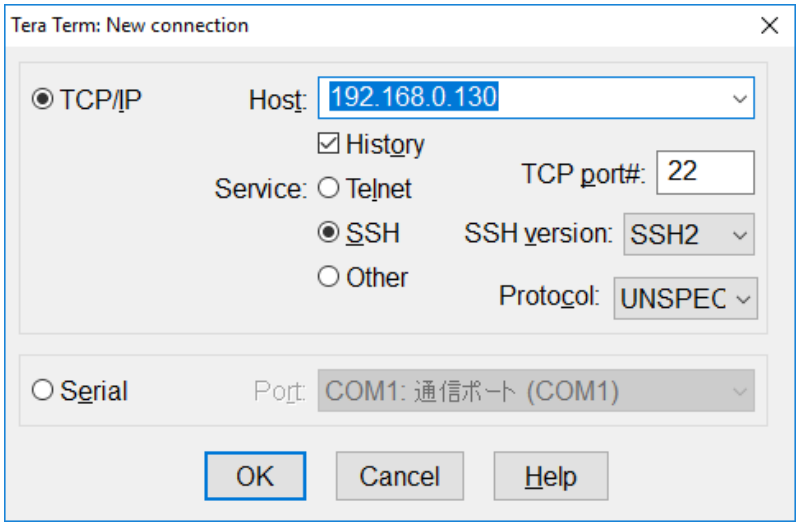
You can use a terminal emulator that has the SSH protocol to connect to EM-Linux.

In this example, we use Tera Term4.84.

1) Start the terminal emulator, and configure the connection settings.

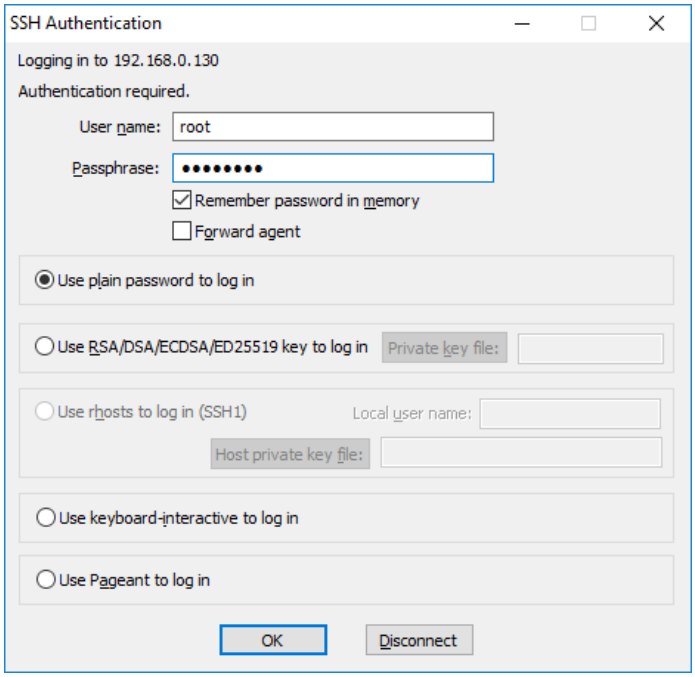
Set the following:

IP address	LAN cable：192.168.0.130 USB cable：192.168.10.130 * If the EM-Linux IP address has been updated, change this setting to match.
Port	22

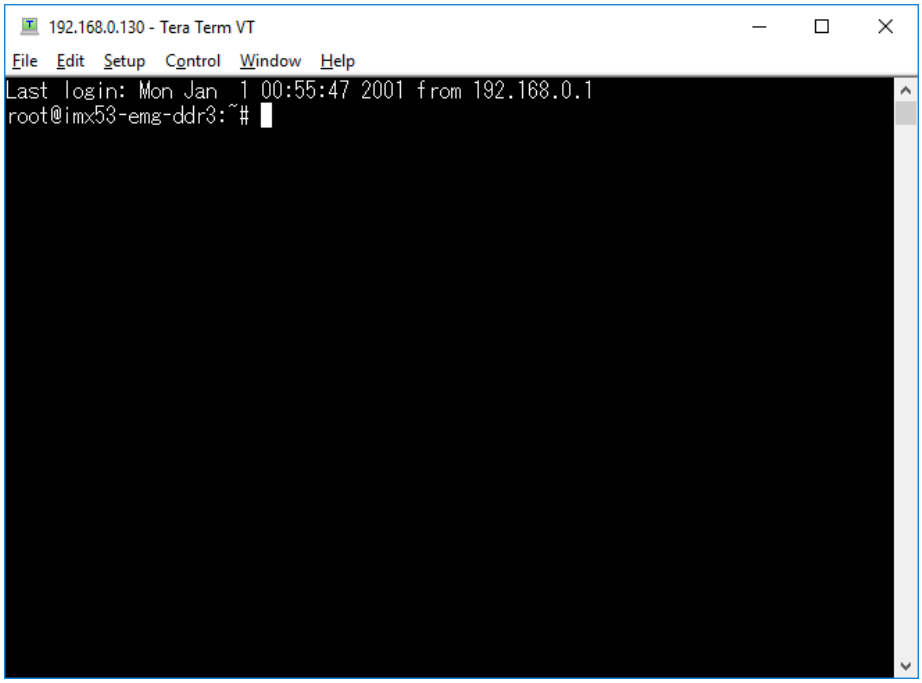


Log in as the following user:

User that can log in	Same as user account * See "1.4 EM-Linux User Account Settings"



A connection is established.



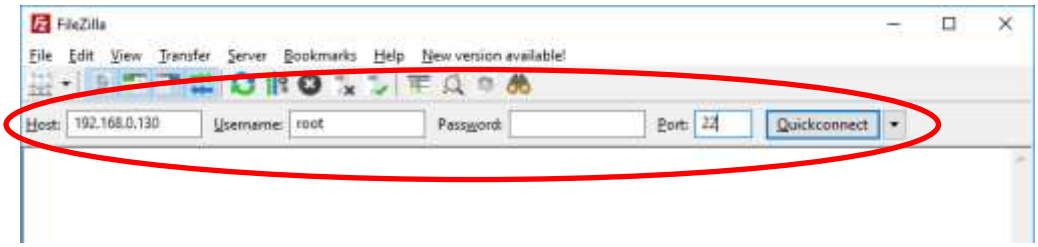
1.2.4 File transfer method (FTP)

Transfer to EM-Linux with FTP client.

1) Start a FTP client that can use SFTP connections, and configure the connection settings.
FileZilla 3.9.0.2. is used in this example.

Set the following:

Protocol	SFTP
IP address	LAN cable：192.168.0.130 USB cable：192.168.10.130 * If the IP address has changed, change this setting to match.
Port	22
Log In	Same as user account * See "1.4 EM-Linux User Account Settings"“

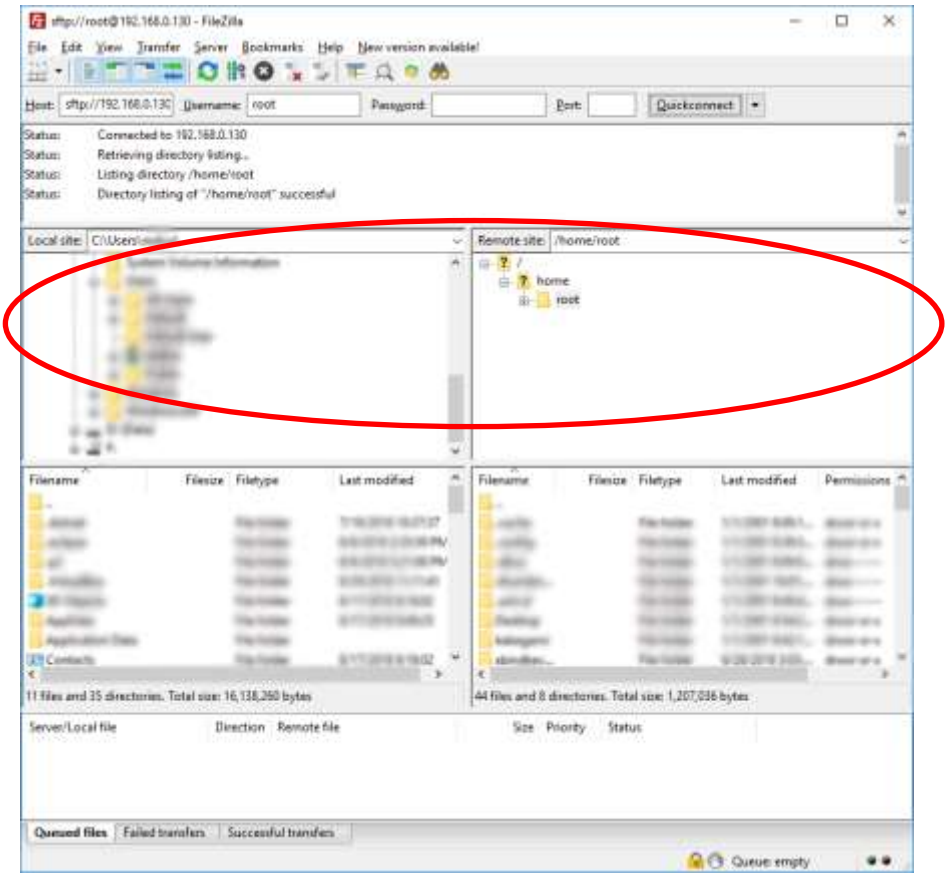


Click [Quickconnect].

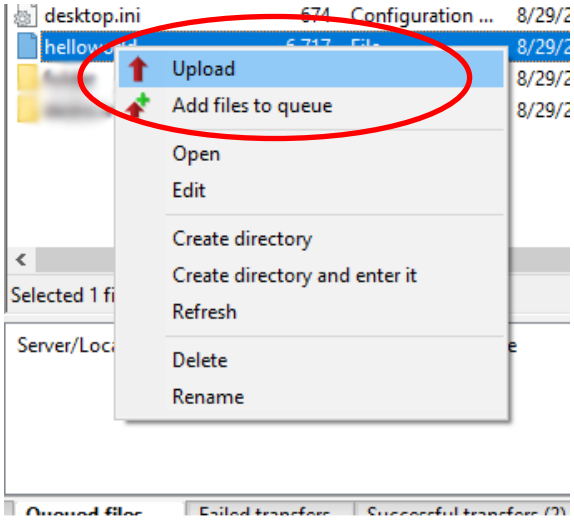
Set the local and remote sites.

This example uses the following settings:

Local site	Desktop
Remote site	/mnt/user/



Right-click the executable file, and from the shortcut menu click [Upload].



The target file has been copied to /mnt/user/.

1.2.5 File transfer method (Samba)

1) Access the following address with Explorer etc.

User folder1	LAN cable : \\192.168.0.130\em\user USB cable :\\192.168.10.130\em\user * If the IP address has changed, change this setting to match.
User folder2*	LAN cable : \\192.168.0.130\em\user2 USB cable :\\192.168.10.130\em\user2 * If the IP address has changed, change this setting to match.

*User folder 2 may not be available depending on the product.

2) Please log in as the following user.

Log In	Same as user account * See "1.4 EM-Linux User Account Settings"“

With the above operation, the user folder in EM-Linux can be accessed.
Copy the file with Explorer etc.

1.3 Specifications of Development Environment

1.3.1 Host Operating System

Operating System	Windows® 10
Oracle VM VirtualBox	Oracle VM VirtualBox 6.1 or later

1.3.2 Guest Operating System

Image Name	EM-Linux
Operating System	Linux Mint17.3 Cinnamon 64-bit
User account	User ID: em, password: em

Toolchain

Model	EMG7-A8	EM(G)8-A7
Toolchain	poky-glibc-x86_64-meta-toolchain-armv7a-neon-toolchain-2.1.2.sh poky-glibc-x86_64-meta-toolchain-qt5-armv7a-neon-toolchain-2.1.2.sh poky-glibc-x86_64-em-image-mx53-armv7a-neon-toolchain-2.1.2.sh	poky-glibc-x86_64-meta-toolchain-cortexa7hf-neon-toolchain-2.1.2.sh poky-glibc-x86_64-meta-toolchain-qt5-cortexa7hf-neon-toolchain-2.1.2.sh poky-glibc-x86_64-em-image-mx6ul-cortexa7hf-neon-toolchain-2.1.2.sh
Path	/opt/poky/2.1.2/	

1.4 EM-Linux User Account Settings

User	User ID	Password
root	root	not configured

On EM-Linux, by default the password is not configured.

If you are using Qt and performing either transfer or remote debug, from the EM-Linux console use the following command to set up a password.

For instructions on how to connect to the console, see "1.2 Connection between PC and EM series".

```
# passwd
Change the root password.
Enter the new password (minimum 5 characters)
Use uppercase, lowercase, and numbers.
New Password:  (Any password)
Enter New Password Again:  (Any password)
passwd:  The password has been updated.
```

2 Disabling Write Protection

To prevent system damage due to malfunctions and unexpected problems on EM-Linux, some folders are set as read-only (RO).

To write to a read-only folder, turn off the write protection temporarily.

> ⚠ **Note**
>
> **If you turn off write protection to write to a read-only folder, turn on write protection again as soon as writing is complete. If you leave write protection off, the system could get damaged by malfunctions or unexpected problems, resulting in abnormal operation.**

2.1 Setting the write-protected area in the user area

The user area (/mnt/user/, /mnt/user2/) can also be included in the write protection range.

2.1.1 Connecting the console

*For the connection method, refer to "1.2 Connection between PC and EM series".

2.1.2 Include the user area in the write-protected area

Operate the EM-Linux console.

To include the user area in the write-protected area, execute the following command.

User area 1 (mnt/user/)

```
# set_mode_of_user_area1 1
```

User area 2 (mnt/user2/)

```
# set_mode_of_user_area2 1
```

2.1.3 Do not include the user area in the write-protected area

Operate the EM-Linux console.

If the user area is not included in the write protection range, execute the following command.

User area 1 (mnt/user/)

```
# set_mode_of_user_area1 0
```

User area 2 (mnt/user2/)

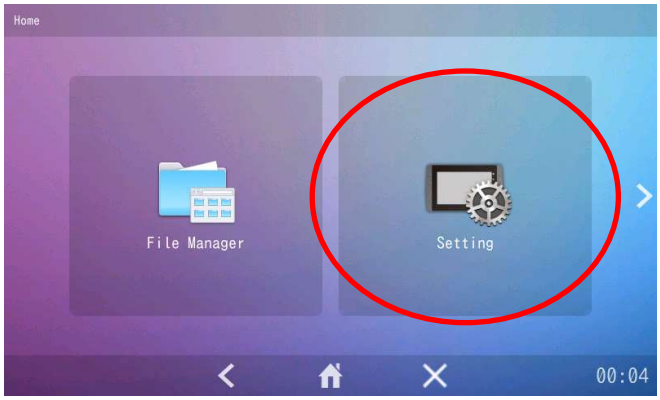
```
# set_mode_of_user_area2 0
```

2.2 Canceling temporary protection with System Setting Tool

2.2.1 Starting the System Setting Tool

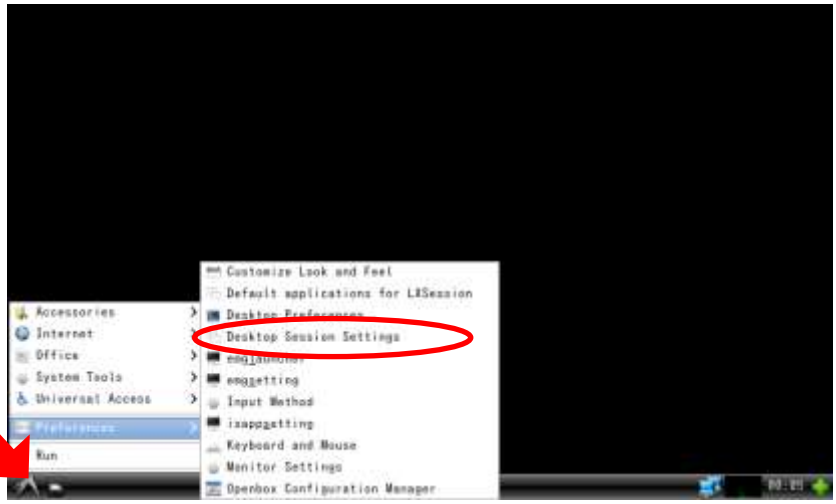
Method 1

Start the tool by opening the [EMG Launcher] application and clicking [Setting].



Method 2

Start the tool from [Start] menu > [Preferences] > [emgsetting].



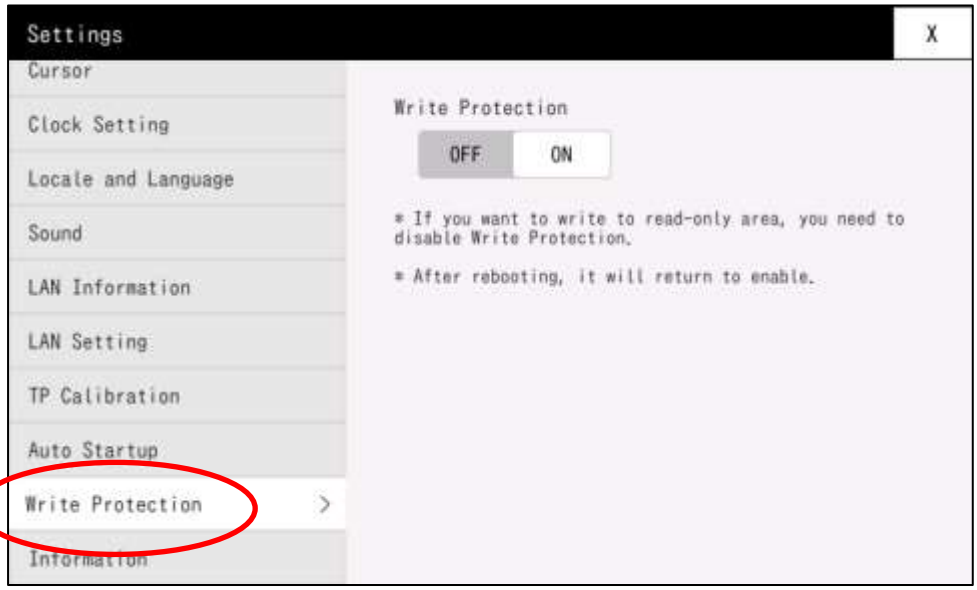
Method 3

Start the tool by running the following program:

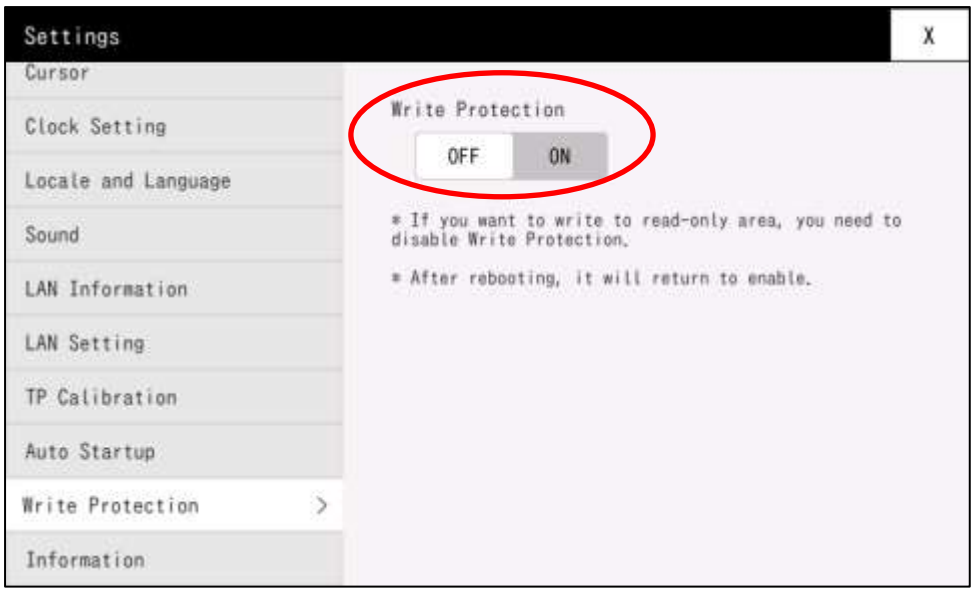
/usr/bin/emg_setting

2.2.2 Disabling Write Protection

1) From the menu, select [Write Protection].



Select OFF to disable write protection.

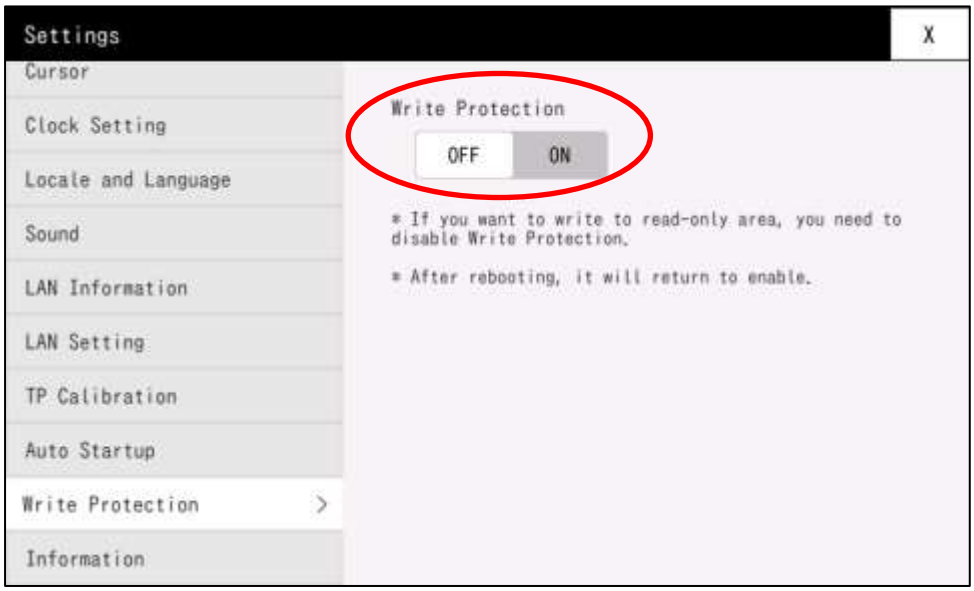


*If the user area is set as write-protected, the user's write-protection will also be disabled.

* Write protection is turned on again when you restart the system.

2.2.3 Enabling Write Protection

Touch [Write Protection] again to turn write protection on.



2.3 Set Up Using Commands

2.3.1 Connecting From Console

* For instructions on how to connect from the console, see "1.2 Connection between PC and EM series".

2.3.2 Disabling Write Protection

Run the EM-Linux console.

Execute the following command to disable write protection.

```
# wprotect_off
```

*If the user area is set as write-protected, the user's write-protection will also be disabled.

* Write protection is turned on again when you restart the system.

2.3.3 Enabling Write Protection

Run the EM-Linux console.

Execute the following commands to enable write protection.

```
# wprotect_on
```

*If the user area is set as write-protected, the user's write-protection will also be enable.

3 Application Development

This chapter describes the steps to develop an application that displays "Hello, world!" on the EM-Linux console.

3.1 Create Source Files

Create a source file on the virtual machine. Launch the virtual machine, see "1.1.2 Registering the Development Environment".

1) Start up a text editor to write the source code. Enter the following.

```
#include <iostream>

int main()
{
        std::cout << "Hello, world!" << std::endl;
}
```

2) Save the code as shown below.

Directory	/home/em/
File Name	helloworld.cpp

3.2 Build

Build the source files. Run on the virtual machine.

1) From the toolbar click  to start the [Terminal].

2) Enter commands as shown below.

Move to the source file directory

```
$ cd /home/em/
```

Set up build environment

* Set variables such as the PATH. You must perform this once every time you start the terminal.

For EMG7-A8

```
$ source /opt/poky/2.1.2/EM-A8/environment-setup-armv7a-neon-poky-linux-gnueabi
```

For EM(G)8-A7

```
$ source /opt/poky/2.1.2/EM-A7/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
```

Build helloworld.cpp:

```
$ $CXX -o helloworld helloworld.cpp
```

The following executable file is generated.

Executable file	helloworld

[Note]

By default, an executable file with debug symbols is generated. If you do not need debug symbols, change the setup script as follows.

* Executable files with debug symbols will be larger than usual.

Setup script

For EMG7-A8

```
/opt/poky/2.1.2/EM-A8/environment-setup-armv7a-neon-poky-linux-gnueabi
```

For EM(G)8-A7

```
/opt/poky/2.1.2/EM-A7/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
```

With debug symbol

```
export CFLAGS=" -O2 -pipe -g -feliminate-unused-debug-types  "
export CXXFLAGS=" -O2 -pipe -g -feliminate-unused-debug-types  "
```

No debug symbol

```
export CFLAGS=" -O2 -pipe -feliminate-unused-debug-types  "
export CXXFLAGS=" -O2 -pipe -feliminate-unused-debug-types  "
```

3.3 Transfer

3.3.1 Connecting the Console

* For instructions on how to connect from the console, see "1.2 Connection between PC and EM series".

3.3.2 Transferring the Executable file

Transfer the generated executable file to EM-Linux.

Perform the following steps with the console connected.

Step 1. Configure network settings

* For instructions on how to connect from the console, see "1.2 Connection between PC and EM series".

Step 2. Copying the File From Virtual Machine To Windows

Using a shared folder or other method, copy the generated executable file to Windows.

In this example the file is copied to the desktop.

* For instructions on configuring a shared folder, see "1.1.3 Development Environment Settings (Optional)".

Step 3. Transfer the executable file to EM-Linux

* For instructions on how to connect from the console, see "1.2 Connection between PC and EM series".

1) Start a FTP client that can use SFTP connections, and configure the connection settings.

FileZilla 3.9.0.2. is used in this example.

Move to the transferred directory.

```
# cd /mnt/user/
```

Display the file information for the directory.

```
# ls
```

The executable file (helloworld) has been copied to /home/root/.

3.4 Run

Run the executable file that was transferred.

Use the EM-Linux console.

1) Navigate to the directory where the executable file was transferred.

```
# cd /mnt/user/
```

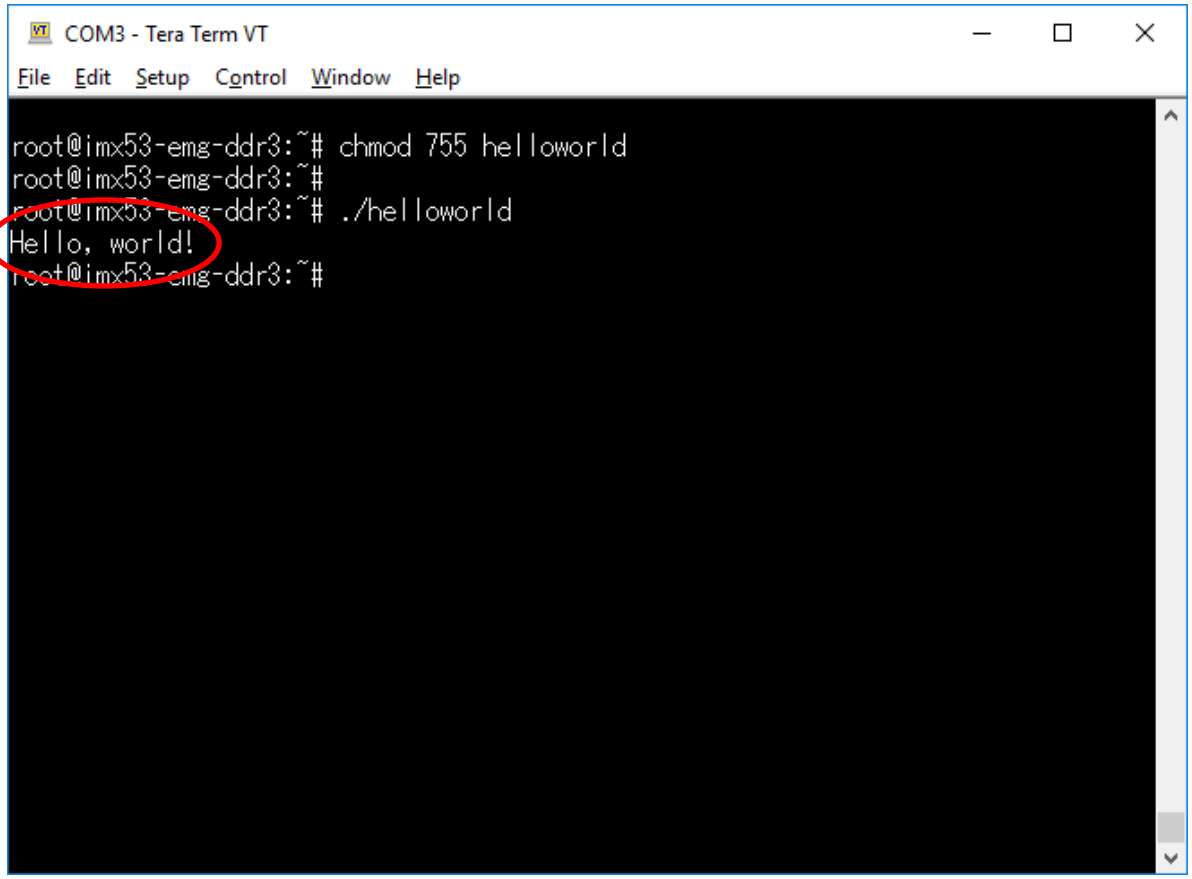
Set access rights for the executable file.

```
# chmod 755 helloworld
```

Enter the following command to execute.

```
# ./helloworld
```

The console displays " Hello, world!".



4 Qt Application Development

The development environment is installed with the QtCreator application development tool.

This tool provides another way for you to create applications.

This chapter explains how to use QtCreator to develop a Qt application that displays the text "Hello EMLinux" on the EM-Linux screen.

4.1 Starting

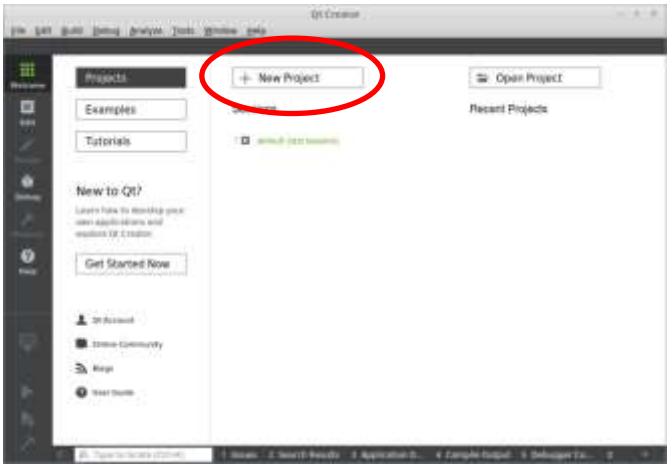
On the desktop, double click the "qtcreator-em" icon  to launch Qt Creator.

You can also launch Qt Creator with the following command.

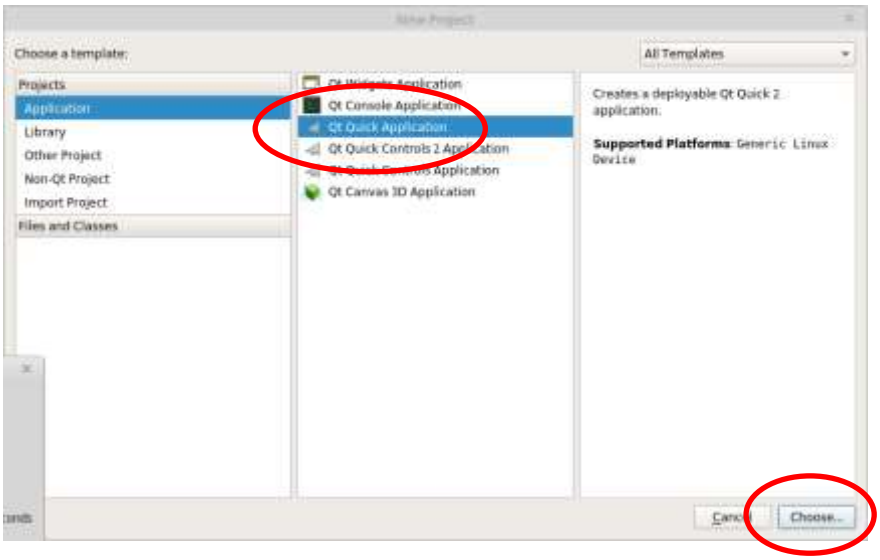
```
$ /opt/qtcreator-4.2.1/bin/qtcreator.sh
```

4.2 Creating a New Project

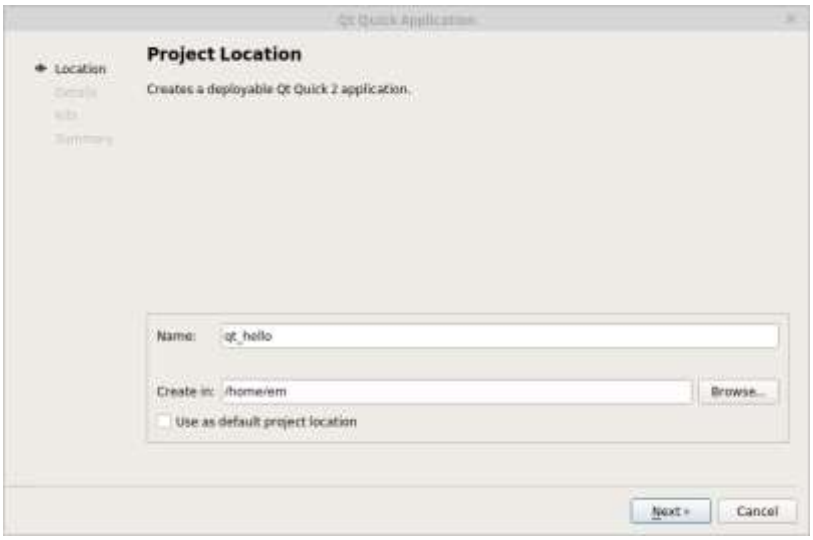
1) Click [New Project].



Select “Qt Quick Application”.

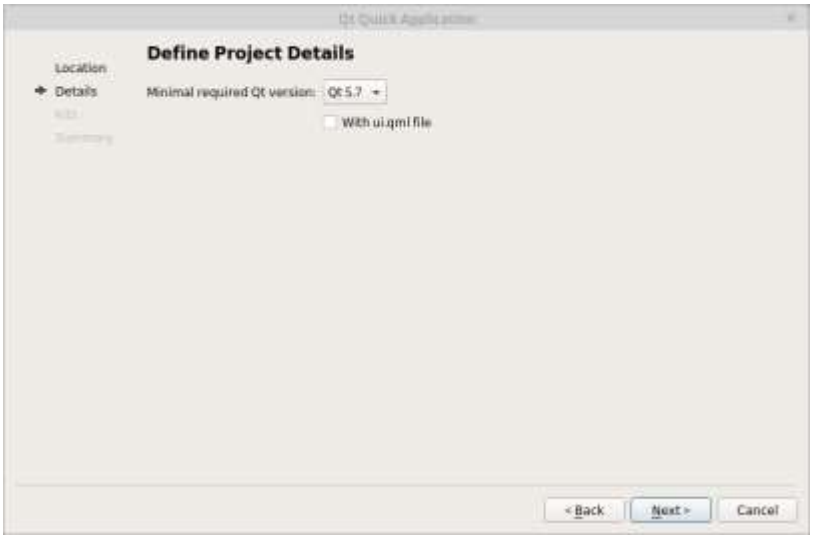


Enter the project name and path.



Enter the Qt version to use. Select Qt5.7.

Qt version	Qt 5.7



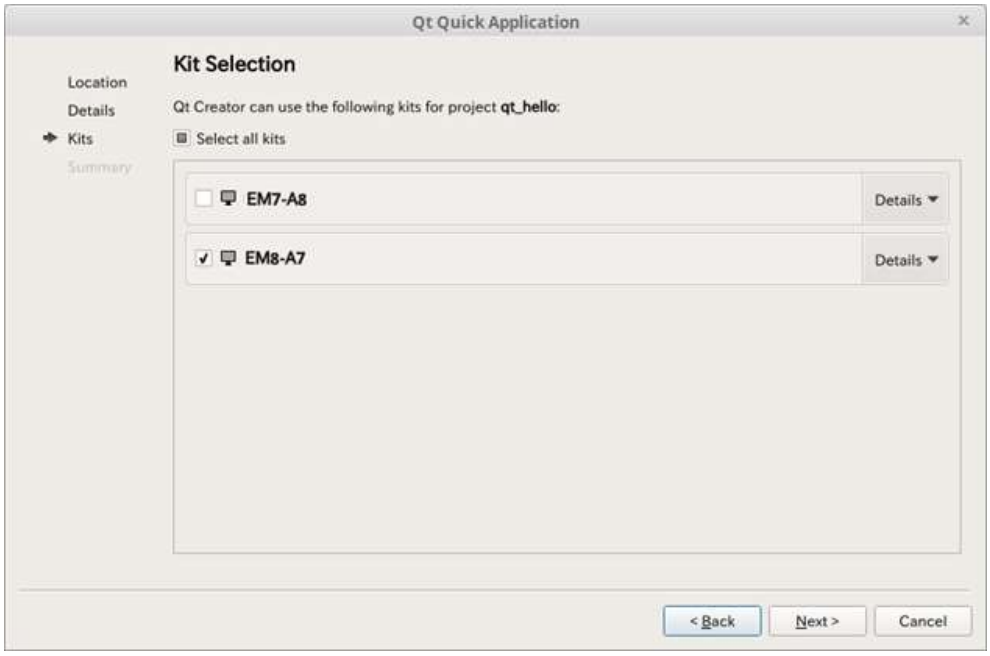
Select the EMG7-Linux kits.

EMG7-A8

EM7-A8

EM(G)8-A7

EM8-A7

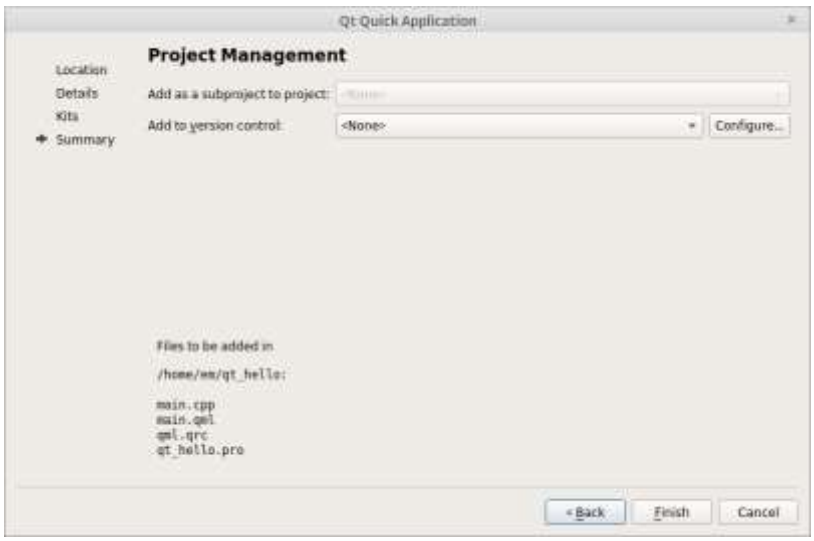


The following are details about the kit.

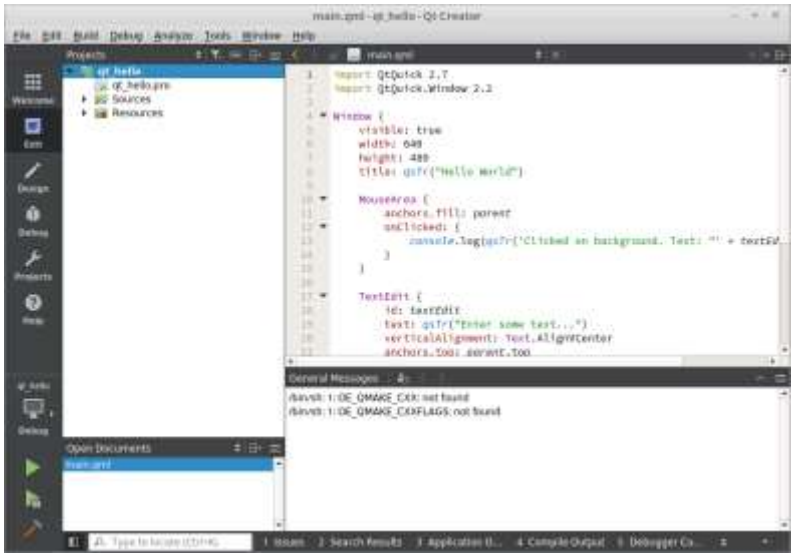
Kit name	EM7-A8
Qt version	Qt 5.7.1 (EM7-A8)
Qmake path	/opt/poky/2.1.2/EM-A8/sysroots/x86_64-pokysdk-linux/usr/bin/qt5/qmake
Compiler	GXX(EM7-A8)
Compiler path	/opt/poky/2.1.2/EM-A8/sysroots/x86_64-pokysdk-linux/ usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-g++
Debugger	GDB(EM7-A8)
Debugger path	/opt/poky/2.1.2/EM-A8/sysroots/x86_64-pokysdk-linux/ usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-gdb

Kit name	EM8-A7
Qt version	Qt 5.7.1 (EM8-A7)
Qmake path	/opt/poky/2.1.2/EM-A7/sysroots/x86_64-pokysdk-linux/usr/bin/qt5/qmake
Compiler	GXX(EM8-A7)
Compiler path	/opt/poky/2.1.2/EM-A7/sysroots/x86_64-pokysdk-linux/ usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-g++
Debugger	GDB(EM8-A7)
Debugger path	/opt/poky/2.1.2/EM-A7/sysroots/x86_64-pokysdk-linux/ usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-gdb

Do not use this screen to add the project to a version control system.



A new project is created.



[Note]

By default, an executable file with debug symbols is generated. If you do not need debug symbols, change the environment variables at build time as follows. Environment variables can be set from the project build settings.

* Executable files with debug symbols will be larger than usual.

With debug symbol

CFLAGS	-O2 -pipe -g -feliminate-unused-debug-types
CXXFLAGS	-O2 -pipe -g -feliminate-unused-debug-types

No debug symbol

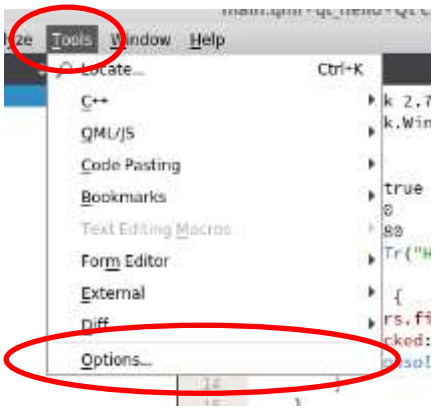
CFLAGS	-O2 -pipe -feliminate-unused-debug-types
CXXFLAGS	-O2 -pipe -feliminate-unused-debug-types

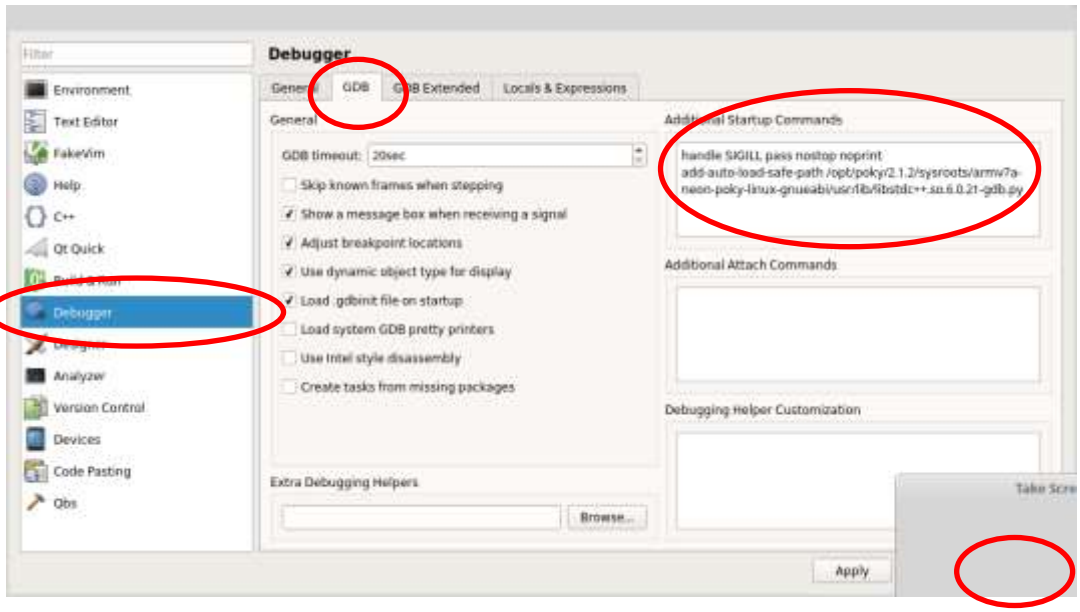
4.3 Configuring Remote Debug Function

This section describes the settings to make remote debugging possible.

Configure QtCreator as follows.

1) From the [Tools] menu, select [Options] > [Debugger] > [GDB] tab, to display the [Additional Startup Commands] field.

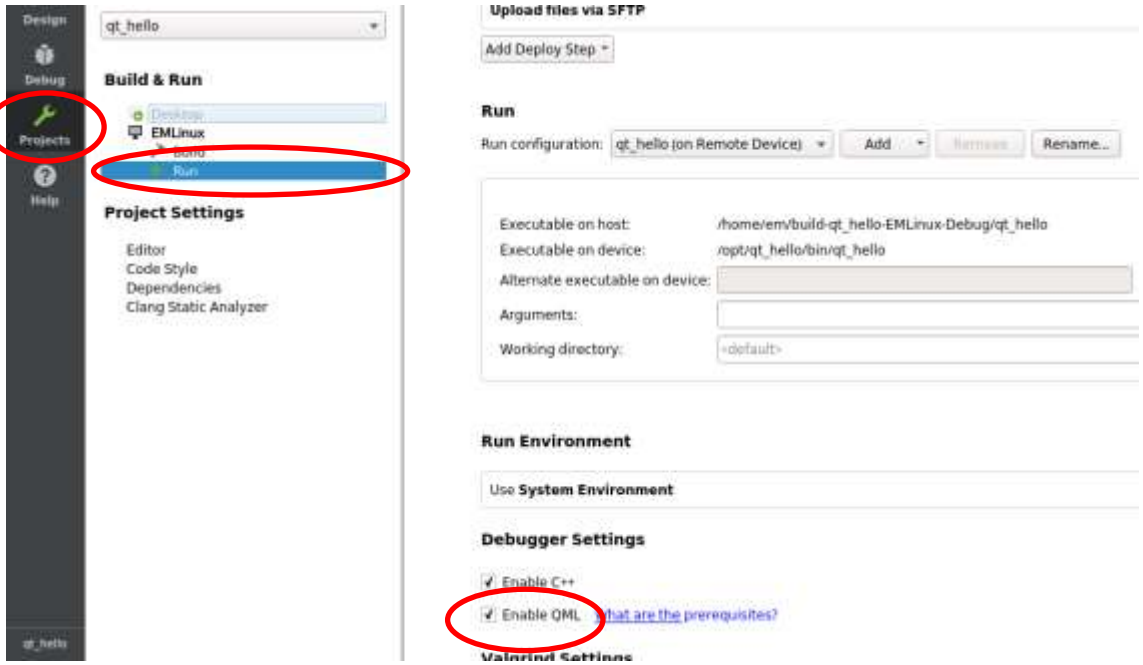




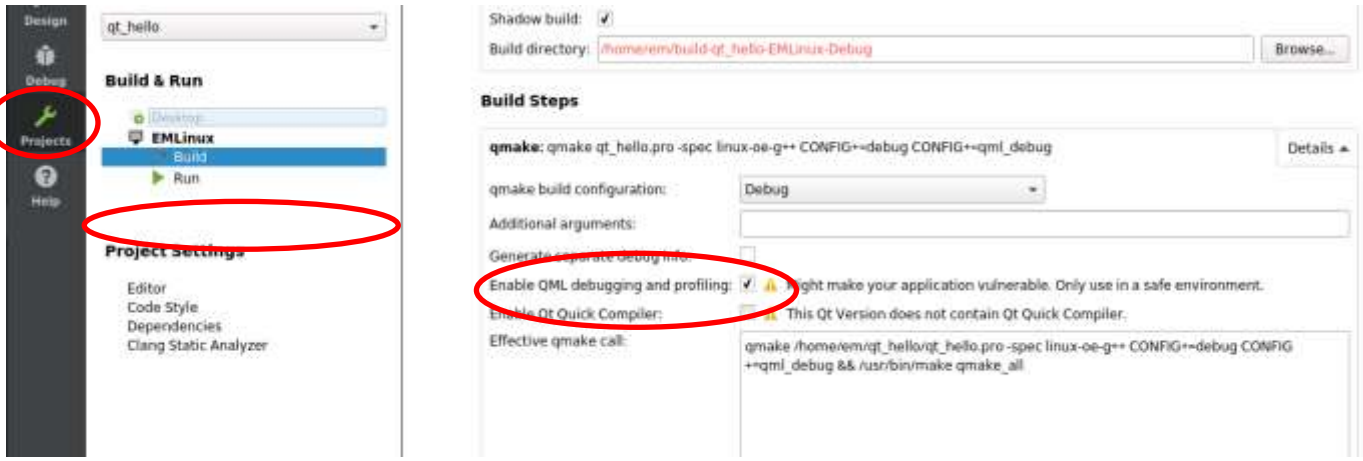
In the [Additional Startup Commands] field add the following two lines, and click OK.

Type	Additional Commands
EMG7-A8	handle SIGILL pass nostop noprint add-auto-load-safe-path /opt/poky/2.1.2/EM-A8/sysroots/armv7a-neon-poky-linux-gnueabi/usr/lib/libstdc++.so.6.0.21-gdb.py
EM(G)8-A7	handle SIGILL pass nostop noprint add-auto-load-safe-path /opt/poky/2.1.2/EM-A7/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/lib/libstdc++.so.6.0.21-gdb.py

From [Projects] > [Build & Run] > [Run] > [Debugger Settings], select the [Enable QML] check box.



From [Projects] > [Build & Run] > [Build] > [Build Steps], select the [Enable QML debugging and profiling] check box.



4.4 Project Settings

Set up the transfer path to EM-Linux so you can run the file from QtCreator.

This section describes the settings for transferring to "/home/root/".

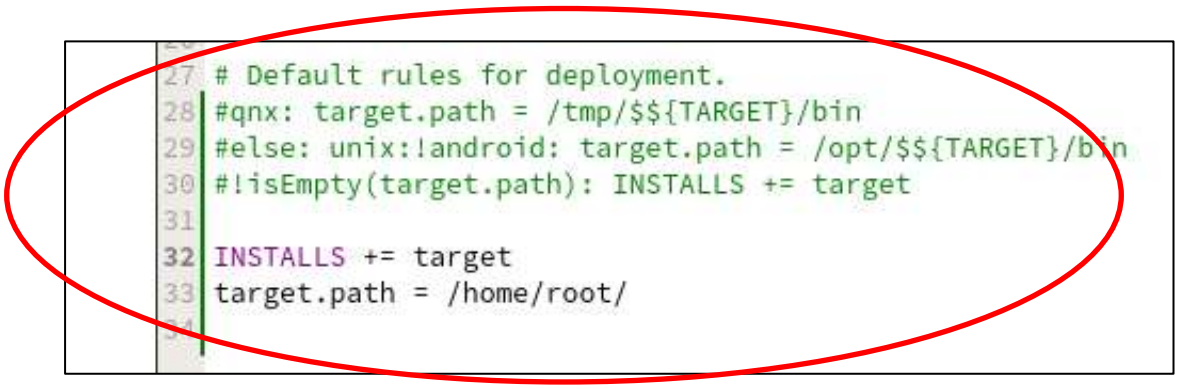
* To transfer a file to a read-only folder, you must turn off write protection. For instructions on disabling write protection, see "2 Disabling Write Protection".

1) Double click the project file (*.pro) to display it in QtCreator.

Add the following code to the project file (*.pro).

```
INSTALLS += target
target.path = /home/root/
QMAKE_CXXFLAGS += -DQT_NO_OPENGL_ES_3
```

Comment the default section as follows:

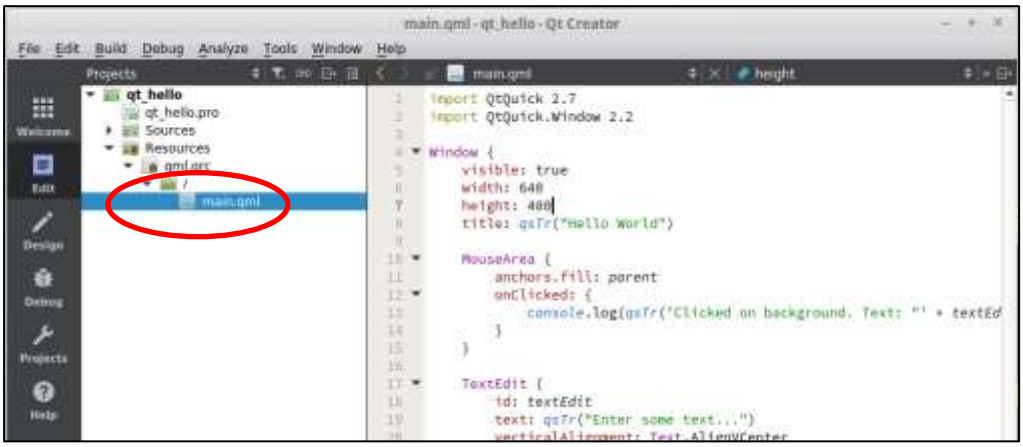


4.5 Editing the QML File

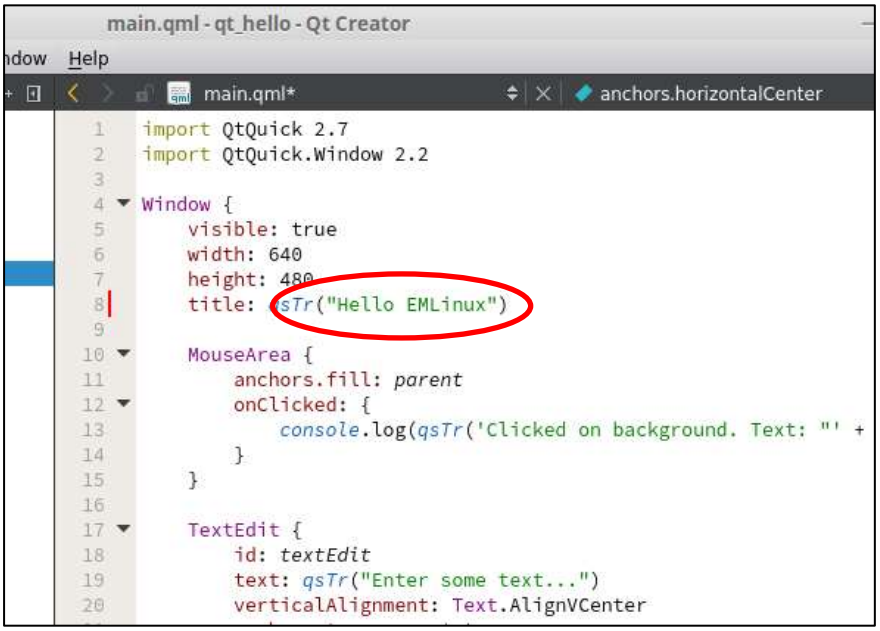
In a Qt Quick application, the screen configuration is described in a QML file.

This section describes how to change the text displayed on the screen.

1) Double click main.qml to display in Qt Creator.



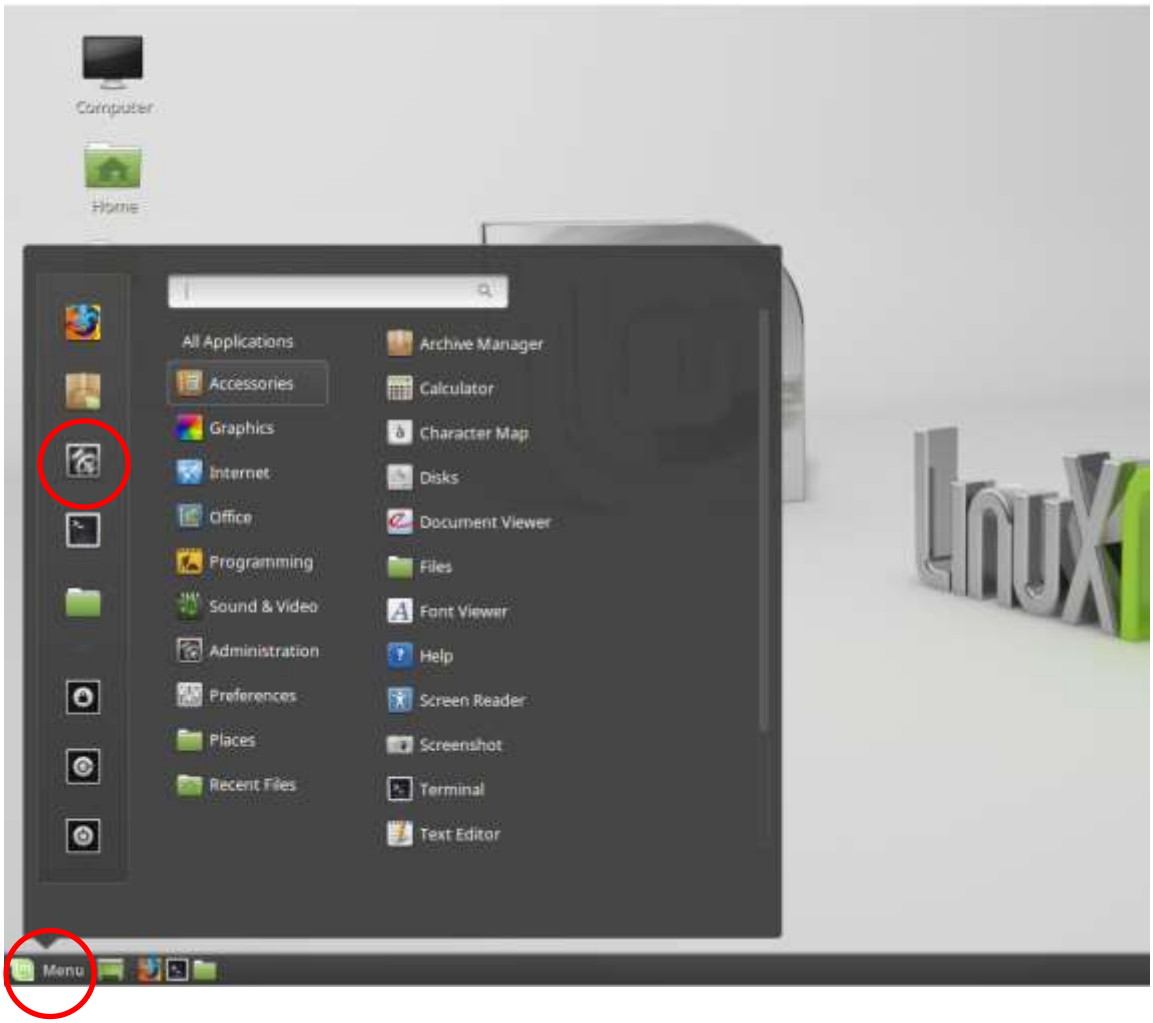
Change “Hello World” to “Hello EMLinux”.



4.6 Network Settings

Change the network settings of the virtual machine so it can connect to EM-Linux.

1) At the bottom-left corner of the desktop, select [Menu], then [System Settings].



Click [Networking].

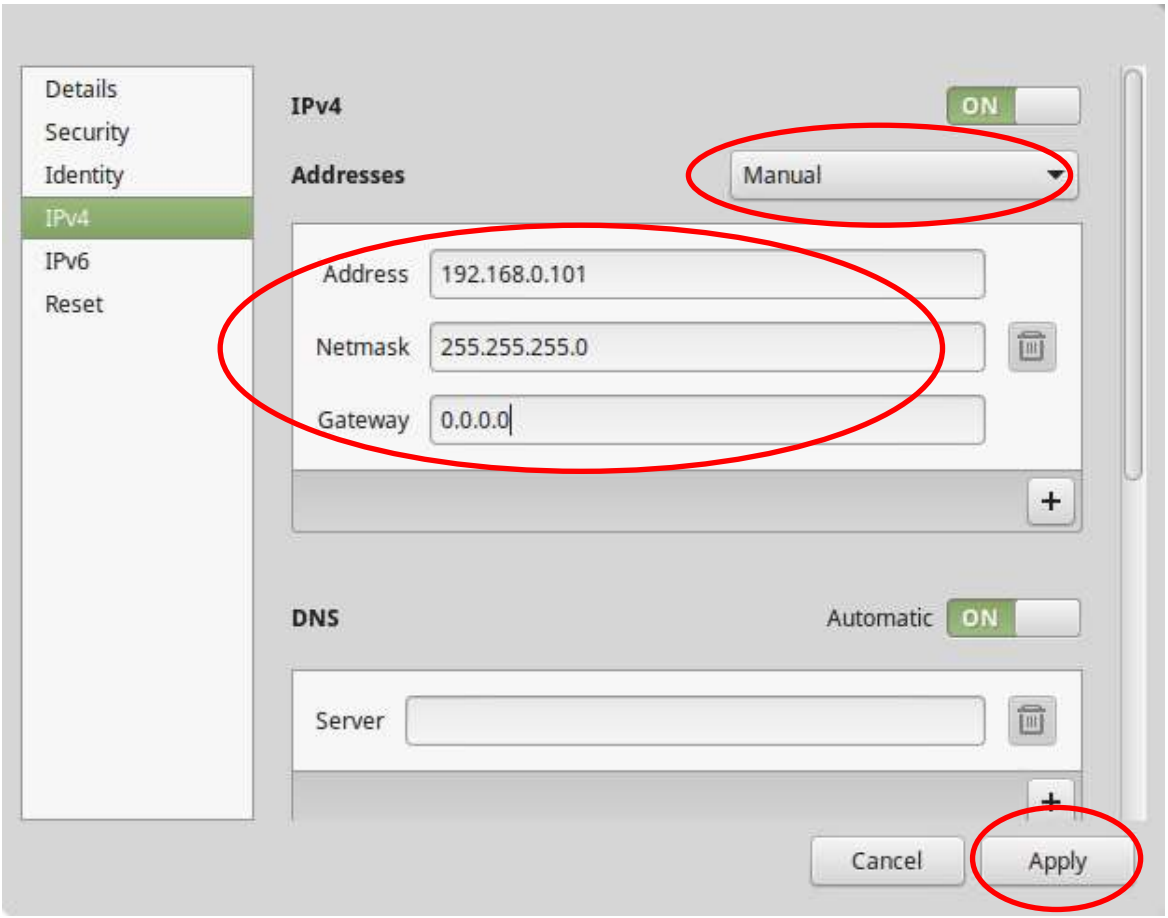
Click  .

Select IPv4, and specify the Address, Netmask, and Gateway settings.

This example uses the following settings:

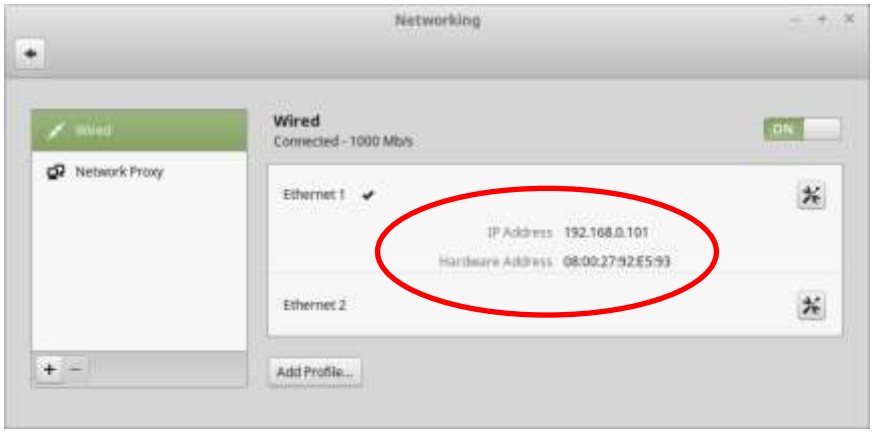
Addresses	Manual
Address	192.168.0.101
Netmask	255.255.255.0
Gateway	not configured

* Change the settings, such as IP, to match your environment.



Turn the ON/OFF switch off, then back on again.

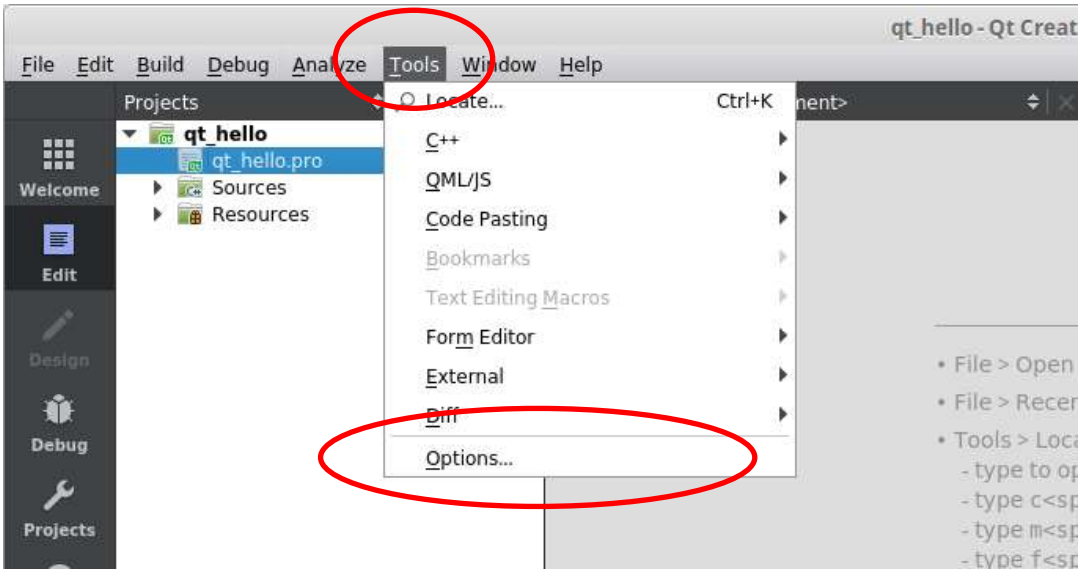
The network settings for the virtual machine are now updated.



4.7 Device Settings

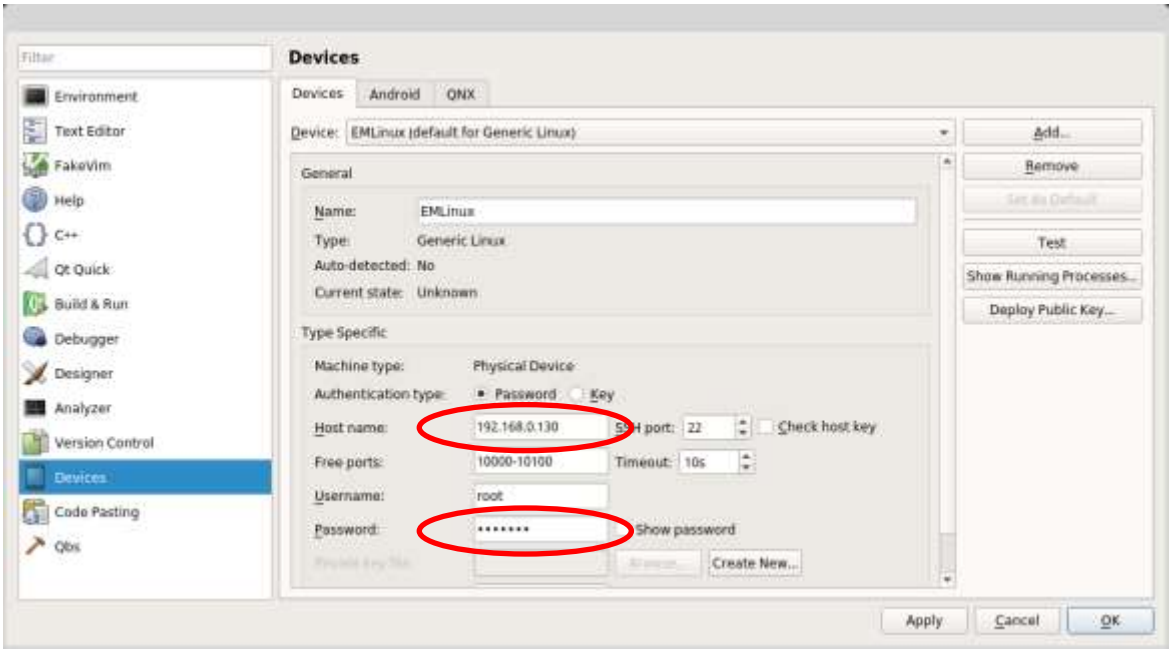
For QtCreator to connect to EM-Linux, specify the IP address and password for EM-Linux.

1) From the [Tools] menu, select [Options].



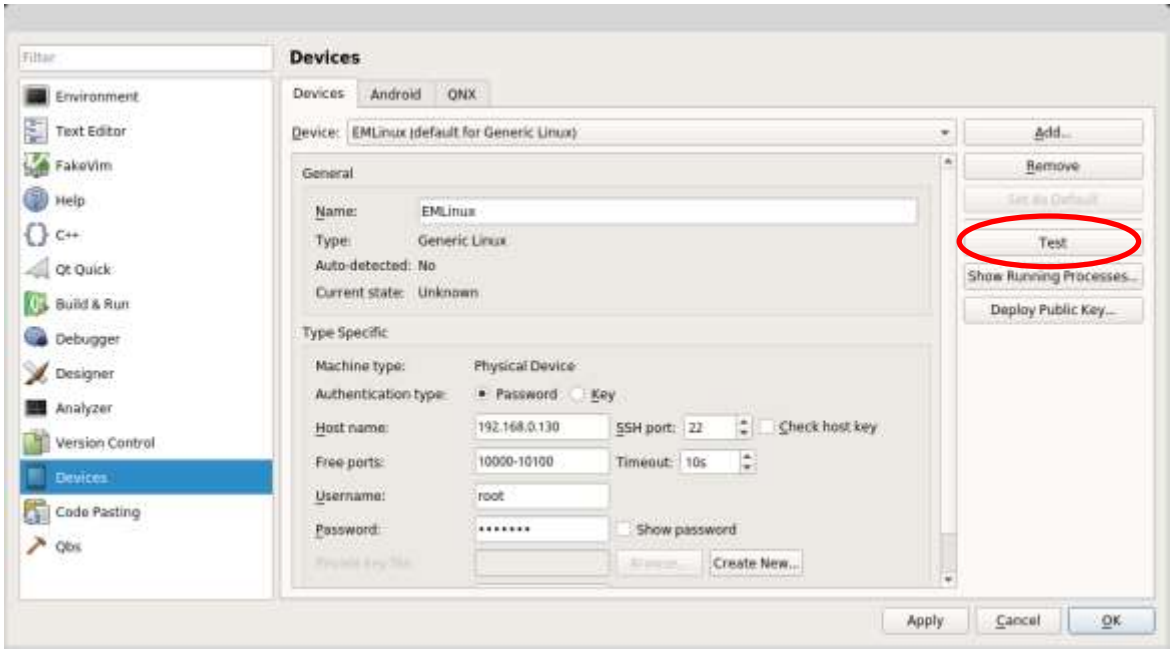
Enter the EM-Linux IP address and password.

If a password has not been specified, please see "1.4 EM-Linux User Account Settings".

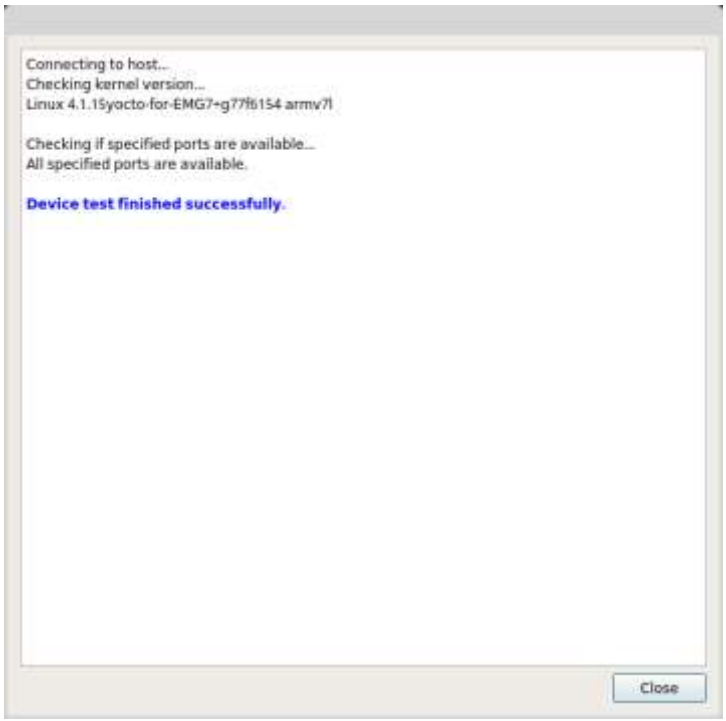


Device Name	EMLinux
Type	General Linux
Authentication method	Password
Host name	192.168.0.130 * If the IP address has changed, change this setting so it is on the same network.
SSH port	22
Empty port	10000 - 10100
Timeout	10 seconds
User name	root
Password	* Required. See "1.4 EM-Linux User Account Settings".

Click “Test” to run a connection test.



If the connection test is successful, the message "device test finished successfully." will appear.



If the connection test failed, check the following:

- Can you ping between the virtual machine and EM-Linux?

If not, check the network settings on the host PC (Windows), the virtual machine, and VirtualBox, use the following references.

Windows (host OS)	1.2 Connection between PC and EM series
Virtual machine	4.6 Network Settings
VirtualBox	1.1.2 Registering the Development Environment

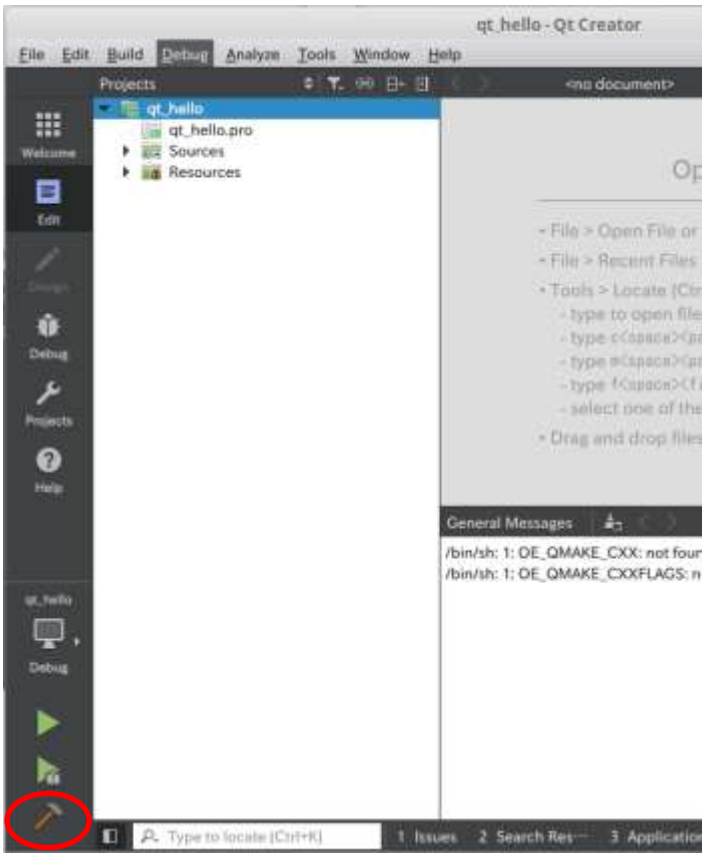
- Have you set up a password on EM-Linux?

If not, set up a password on EM-Linux. See "1.4 EM-Linux User Account Settings".

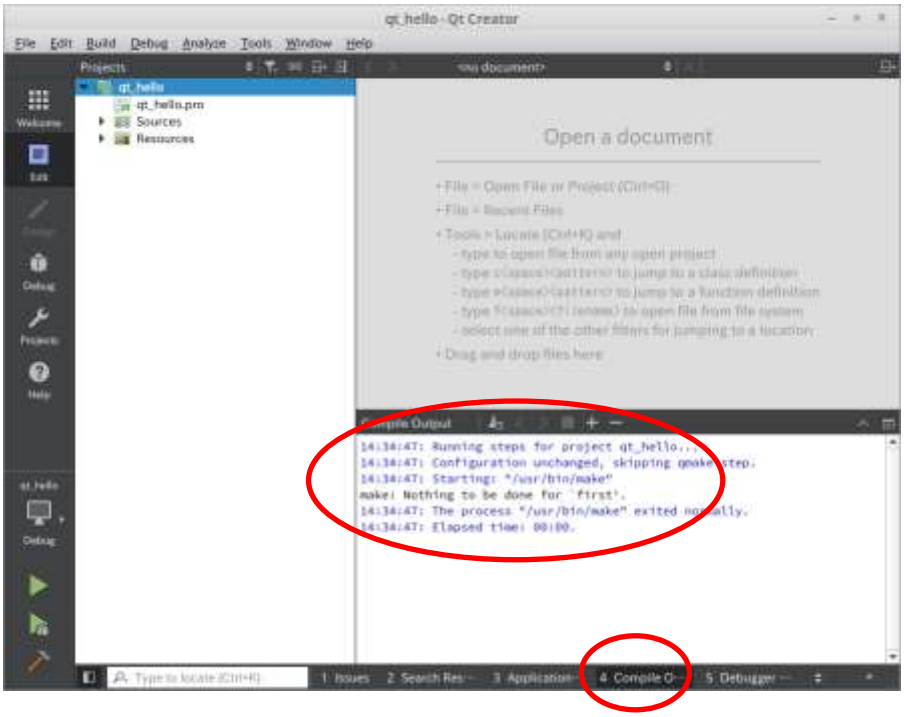
4.8 Build

Build the project that you created.

1) Click [Build].



The build completes successfully.

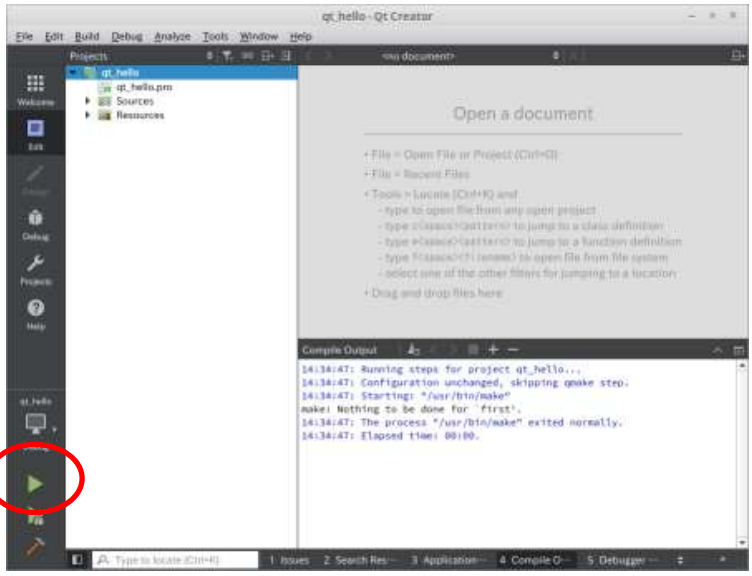


4.9 Run

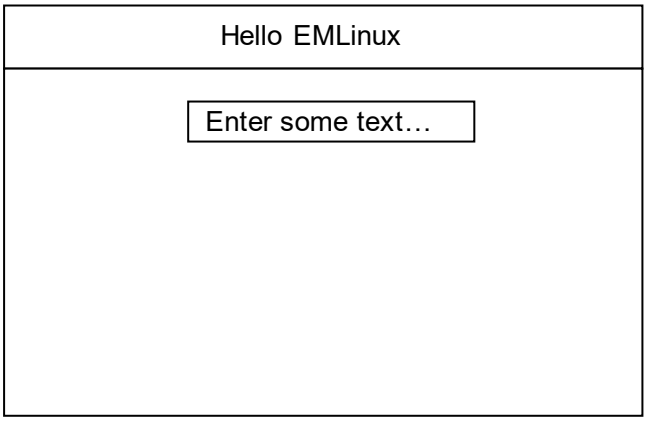
Run the file on EM-Linux.

On the [Run] command, the executable file is transferred and run on EM-Linux.

1) Click [Run].



The screen shown below is displayed.

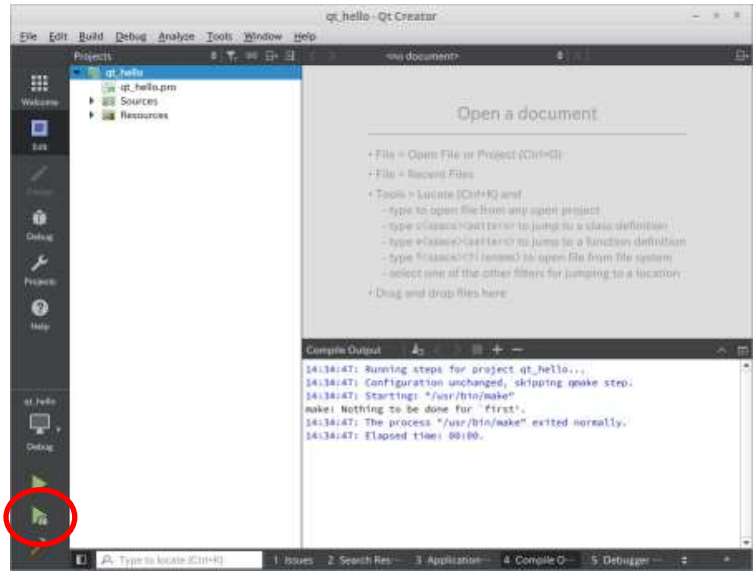


4.10 Remote Debug

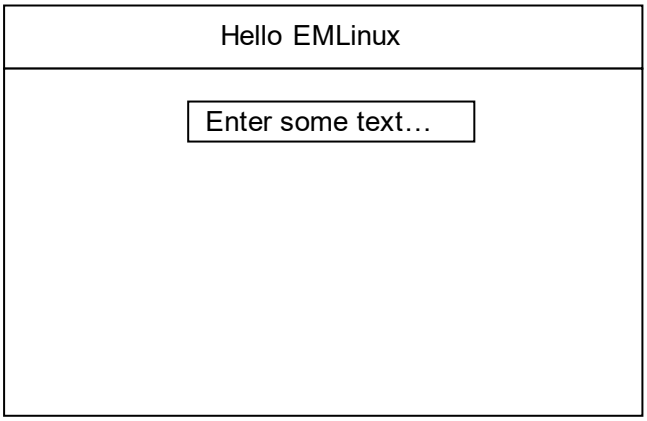
Run the remote debug operation on EM-Linux.

On the [Start Debug] command, the executable file is transferred to EM-Linux and debug is started.

1) Click [Start Debug].



The screen shown below is displayed.



5 Changing the Startup Screen

This chapter describes how to change the startup screen on the EM-Linux.

5.1 Create Source Files

Please prepare the image file in the following format.

Format	Uncompressed 24-bit bitmap
Size	Same as the actual resolution

5.2 Transfer

5.2.1 Connecting the Console

* For instructions on how to connect from the console, see "1.2 Connection between PC and EM series".

5.2.2 Transferring image files

Transfer the image file to EM-Linux.

* For instructions on how to connect from the console, see "1.2 Connection between PC and EM series".

After the transfer is completed, use the following command to write the image file to the boot screen area (/dev/mtd6) of the actual device. *Here, /mnt/user/abc.bmp is the path to the image file.

```
# psplash -w /mnt/user/abc.bmp /dev/mtd6
```

The startup screen is changed by the above procedure.
After writing to the dedicated area, the transferred image file can be deleted.

6 Auto Startup Setting

This chapter describes how to change the applications that launch automatically when the physical machine starts up.

By default, the task bar, desktop, and EMG Launcher launch at startup. To disable this setting, change the startup script.

6.1 Specifications

EM-Linux executes the following script at startup.

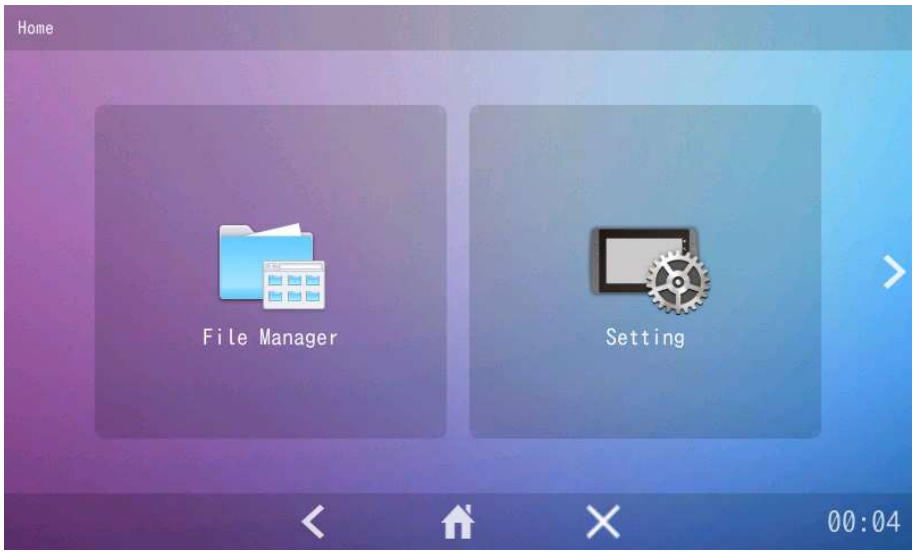
Items	Specification
Startup Script	/home/root/.config/lxsession/LXDE/autostart

By default, the following processes are registered.

Process	Contents
@lxpanel --profile LXDE	Task bar
@pcmanfm --desktop --profile LXDE	Desktop
/usr/bin/emsystem/autostart	EM-Linux Auto Start

With EM-Linux Auto Start, the items set in "Automatic startup" of the System Setting Tool are executed. The EMG launcher is started by default.

EMG Launcher



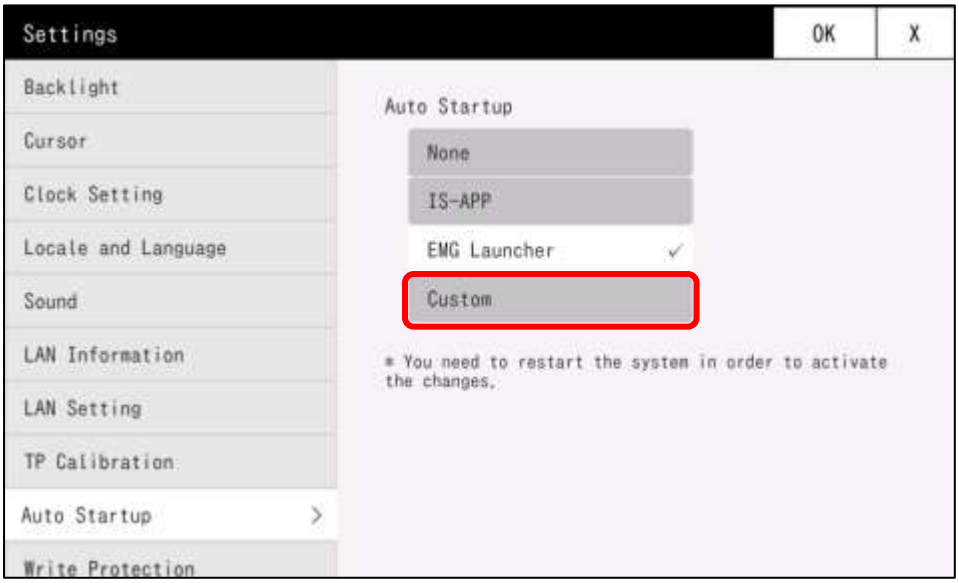
6.2 How to run an arbitrary program

Select "Custom" in "Auto Start" of System Setting Tool.

If you select Custom, "/mnt/user/startup.sh" will be executed.

Edit "/mnt/user/startup.sh" with vi editor (text editor).

Please refer to the separate "Tool Manual" for details on "Automatic startup" of the system setting tool.



6.3 Hiding the desktop and taskbar

If you want to hide the desktop and taskbar, please follow the steps below to correct them. You must remove the write protection before you can change it. For how to release, refer to “2 Disabling Write Protection”.

6.3.1 Connecting the Console

* For instructions on how to connect from the console, see "1.2 Connection between PC and EM series".

6.3.2 Modifying the Startup Script

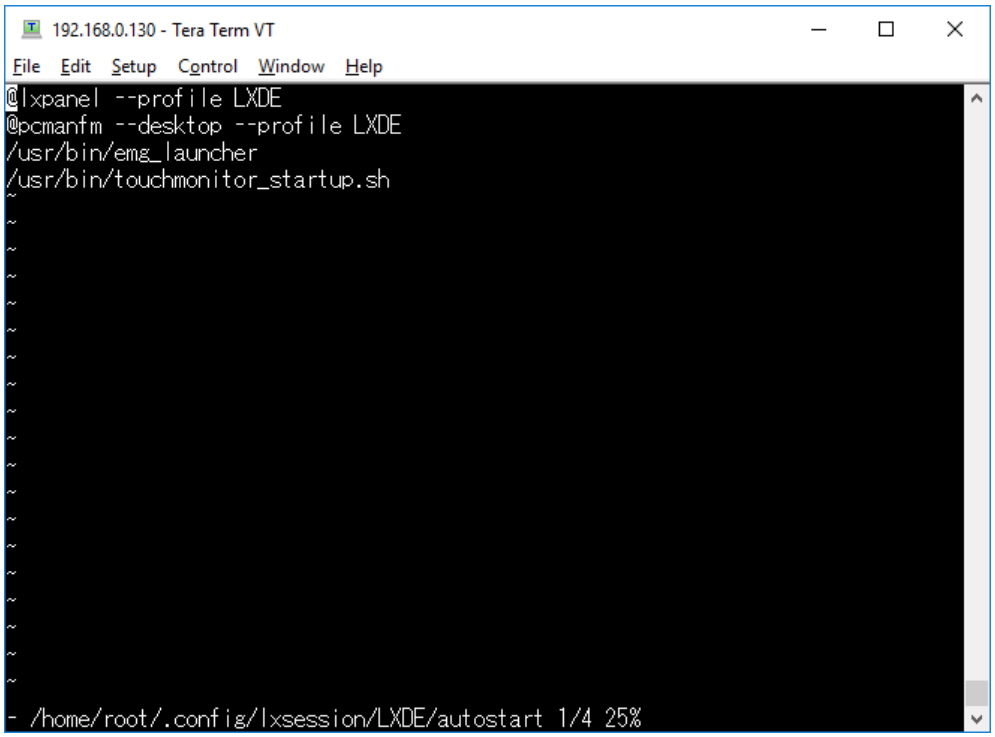
Modify the startup script.

Use the EM-Linux console.

1) Use the following command to open the startup script in the vi editor (text editor).

```
# vi /home/root/.config/lxsession/LXDE/autostart
```

Edit the startup script in the vi editor (text editor).



Change from command mode to edit mode. Press [i] on the keyboard.

Edit the startup script.

To disable a process, add "#" at the beginning of the line.

Example: To disable task bar

```
#@lxpanel --profile LXDE
@pcmanfm --desktop --profile LXDE
/usr/bin/emsystem/autostart
```

Example: To disable task bar and desktop

```
#@lxpanel --profile LXDE
#@pcmanfm --desktop --profile LXDE
/usr/bin/emsystem/autostart
```

Exit edit mode and return to command mode. Press [Esc] on the keyboard.

Save the changes and exit. Using the keyboard, type ":wq" and press [Enter].

Use the command below to restart the system.

```
# reboot
```

To display it again, delete "#".

```
@lxpanel --profile LXDE
@pcmanfm --desktop --profile LXDE
/usr/bin/emsystem/autostart
```

7 EM-Linux Specifications

This chapter describes EM-Linux specifications such as the file map.

7.1 Operating System Specifications

7.1.1 Boot loader

Item	Specification
Boot loader	u-boot 2015.04

24V version of EM(G)8-7W, EM8-10W only

Item	Specification
Boot loader	u-boot 2016.03

7.1.2 Linux kernel

Item	Specification
Kernel	Linux 4.1.15

7.1.3 Desktop environment

Item	Specification
Desktop environment	XWindow system (LXDE)

7.1.4 Start Menu

By default, the following items are registered in the Start menu.

Item		Specification
Accessories	Note	Leafpad
	Terminal	Terminal
	Image viewer	Display image
Internet	X11VNC server	VNC server
System Tools	LXTerminal	Terminal
	Task Manager	Task Display
	File Manager PCManFM	File manager
Universal Access	Florence Virtual Keyboard	Software keyboard
Preferences	Default applications for LXSession	Settings for Auto Startup and startup
	Openbox Configuration Manager	Settings for Windows display and behaviors
	emglauncher	Launcher application (emg_launcher)
	emgetting	EM Setting Tool
	isappsetting	IS-APP Setting Tool
	Keyboard and Mouse	Settings for keyboard and mouse
	Desktop Preferences	Settings for desktop display
	Desktop Session Settings	Auto Startup Setting
	Monitor Settings	Display settings
	Customize Look and Feel	Widget / Icon / Font settings
	Input Method	Input locale setting
Run		Run command

7.2 File map

Folder structure	Application	Location	FileSystem	Type	Size	R/W
/bin	RootFileSystem	/	/dev/mtd3	ubifs	340M	RO*1
/boot						
/home						
/lib						
/media						
/mnt						
/sbin						
/usr						
/var						
/www						
/etc						
/dev	devfs	/dev	udev	devtmpfs	(RAM)	RW
/proc	procfs	/proc	proc	proc	(RAM)	RW
/sys	sysfs	/sys	sysfs	sysfs	(RAM)	RW
/run	temporary file	/run	tmpfs	tmpfs	(RAM)	RW
/tmp	temporary file	/tmp	tmpfs	tmpfs	(RAM)	RW
/var/volatile	temporary file	/var/volatile	tmpfs	tmpfs	(RAM)	RW
/mnt/user	user area 1	/mnt/user	/dev/mtd4	ubifs	90MB	RW *2
/mnt/user2	user area 2	/mnt/user2	/dev/mtd5	ubifs	65MB	RW *2

*1 RootFileSystem is shipped as read-only (RO). If you want to write a file to the read-only area, you need to remove the write protection. For details, refer to “2 Disabling Write Protection”.

*2 The user area can also be changed to read-only (RO). For more details, please refer to “2.1 Setting the write-protected area in the user area”.

7.3 Root file system

Package/ Licence

For the packages installed in EM-Linux and their license terms, refer to licenses.tar.gz in the DVD-ROM (set of development environment).

7.4 Drivers

7.4.1 LAN Specifications

Port	Device	IP	Description
Wired LAN	eth0	dhcp/static	10/100BASE-TX * The initial IP address is 192.168.0.130 / 24

LAN setting file

File	Description
/etc/resolv.conf	DNS server settings
/etc/network/interfaces	Network Settings
/etc/network/lan0.conf	Eth0 settings

7.4.2 Touch Panel Specifications

Conforms to the Linux standard InputDriver.

Item	Description
Specification	Supported by LinuxInputSubsystem and tslib
Sampling frequency	50 /second
Device	/dev/input/touchscreen0
Calibration tool (Capacitance offset)	EMG7-A8 /usr/bin/Calibration EMG8-A7 /usr/bin/tpoffset
Calibration tool (Coordinates Calibration)	EM8-A7 /usr/bin/xinput_calibrator

7.4.3 Sound Specifications

Item	Description
Driver	ALSA Driver1.1.0
Specification	Conform to LinuxALSA driver specifications
Supported format	Uncompressed WAV file
Execution method	For EMG7-7W, EMG7-12
	aplay Example: aplay -D hw:0,0 sample.wav | arecord -D hw:0,0 -f S16_LE -r 44100 -c 2 -d 5 record.wav
	For EM(G)8-7W, EM8-10W
	alsaplayer Volume：0.0 - 1.0 (0.0=Mute, 0.35=35%, 1.0=100%) *Specify a volume for each execution. Example: alsaplayer --startvolume <*Volume*> --enqueue sample.wav

7.4.4 USB Specifications

USB host

Host driver: EHCI HCD (USB2.0)

Class driver	Description
USB HUB	USB hub
USB Mass Storage	USB drive
USB HID	USB mouse and keyboard

USB device

USB Gadget Drivers: USB-Ether

Gadget	Explanation
USB-Ether	When the USB port of this unit is connected to a PC, it is recognized as an Ethernet Adapter by the PC. *USB device driver must be installed on the PC. For details, refer to "1.2.2 Network settings on USB device port". *Please use a one-to-one connection between this unit and your PC. Device：usb0 IP Address：192.168.10.130 LAN configuration file：/etc/network/glan0.conf

7.4.5 LCD Specifications

Supports the Linux standard frame buffer.

Device file: /dev/fb0

Function		Description
open		Conforms to Linux frame buffer driver specifications
close		
ioctl	FBIOGET_FSCREENINFO	Gets fixed screen information. Return value 0: successful Other than 0: failed Input: none Output: fb_fix_screeninfo* type: retrieved fixed screen information
	FBIOGET_VSCREENINFO	Gets variable screen information. Return value 0: successful Other than 0: failed Input: none Output: fb_var_screeninfo* type: retrieved variable screen information
	FBIOPUT_VSCREENINFO	Sets the variable screen information. Return value 0: successful Other than 0: failed

		Input: fb_var_screeninfo* type: variable screen information to set
mmap		Conforms to Linux mmap specifications

Note 1) You can use mmap to map one screen worth of memory in the screen information to buffer memory.
However, because there is no exclusive control for system drawing, if drawing is run on the system-side, the drawing on the screen may not display as intended.
(The system side drawing take precedence.)
To prevent drawing from the system side, you could for example hide the task bar.

7.4.6 Backlight Specifications

File Path: /sys/class/backlight/backlight/

The following files control the backlight.

File	Read/Write	Description
bl_power	R/W	Backlight ON/OFF get / set 0: On 1: Off
brightness	R/W	get / set brightness 1 - 8 (8 levels) 0=OFF
bl_autooffenable	R/W	Backlight Auto Off get / set enable/disable 0: disable 1: enable
bl_autoofftime	R/W	Backlight Auto Off get / set time 1 - 65535 (seconds)

Note 1) After a touch input, Backlight Auto Off turns off the backlight automatically when there is no touch input for the amount of time specified as the Backlight Auto Off time.

Note 2) If the backlight is turned off by the Backlight Auto Off function, the backlight turns on again when there is a touch input, or an application executes bl_power with the value 0.

Note 3) If the backlight is turned on by something other than touch input (such as bl_power), the Backlight Auto Off does not function. Only after a touch input is the backlight turned off automatically when the amount of time specified as the Backlight Auto Off time has elapsed.

Note 4) When you set the parameters for Brightness, Backlight Auto Off (enabled/disabled), and Backlight Auto Off Time, they are saved automatically. As a result, the last defined parameters are used the next time you boot up.

7.4.7 SIO Specifications

Port assignments

EMG7-7W

Port	Device file	Interface	Description
SIO1	/dev/com1	RS232C	Link to /dev/ttymxc0

EMG7-12

Port	Device file	I/F	Description
SIO1	/dev/com1	RS232C	Link to /dev/ttymxc0
SIO2	/dev/com2	RS485	Link to /dev/ttymxc3

EM(G)8-4/ EM(G)8-5/ EM(G)8-7W/ EM8-10W

Port	Device file	I/F	Description
SIO1	/dev/com1	RS232C	Link to /dev/ttymxc4
SIO2	/dev/com2	RS422/RS485	Link to /dev/ttymxc2 I/F is switched by DIPSW

Common Specifications

Function	Description
General	Conforms to Linux serial driver specifications

RS422 Specifications

Set the communication mode to RS422 in the initialization process.

Turn RTS ON when sending data, and turn it OFF after sending.

The extended control code is defined in "seedsware_ext_ioctl.h" in "software" - "ioctrl_include" in the DVD-ROM (development environment set).

Function		Description
General		Conforms to Linux serial driver specifications
Ioctl	IOCTL_UART_MODE_RS422	Sets the communication mode to RS422. Return value 0: Success -1: Failure Input 0
	IOCTL_UART_ASSERT_DE	Turns RTS on. Return value 0:Success -1: Failure Input NULL

	IOCTL_UART_DEASSERT_DE	Turns off RTS. Return value 0:Success -1: Failure Input NULL

RS485 Specifications

Set the communication mode to RS485 in the initialization process.

The extended control code is defined in "seedsware_ext_ioctl.h" under "software" - "ioctrl_include" in the DVD-ROM (development environment set).

Function		Description
General		Conforms to Linux serial driver specifications
Ioctl	IOCTL_UART_MODE_RS485	Sets the communication mode to RS485. Return value 0: Success -1: Failure Input 0
	TIOCSRS485	Sets the details for RS485 operation. Return value 0: successful Other than 0: failed Input: rs485_config type pointer Output: none
	TIOCGRS485	Retrieves the details for RS485 operation. Return value 0: successful Other than 0: failed Input: none Output: rs485_config type pointer
flags	SER_RS485_ENABLED	Driver allows for RS485 operations
	SER_RS485_RTS_ON_SEND	Turns RTS on before transmission, and turns it off after transmission
	SER_RS485_RTS_AFTER_SEND	Turns RTS off before transmission, and turns it on after transmission
	SER_RS485_RX_DURING_TX	Enables incoming transmission while there is an outgoing transmission.

Note 1) If termination is required, enable by following the steps below.

For EMG7-12

Enable by writing 1 to: /sys/class/gpio/status_terminate/value.

Enable by command: echo 1 >/sys/class/gpio/status_terminate/value

For EM(G)8-4 / EM(G)8-5 / EM(G)8-7W / EM8-10W

Enable termination from the serial port settings. From the dipswitch, turn ON SW1.

7.4.8 RTC Specifications

Function	Specification
Configure date/time	2000/1/1 00:00:00 - 2037/12/31 23:59:59
Leap years	Supported from 2000 to 2037
Out-of-battery operations	Detect the battery level on driver initialization (= OS startup) If there is no power left, the device is reset to 1970/1/1 00:00:00

Device file: /dev/rtc0

This driver is for controlling an external RTC.

An external RTC can use batteries to operate even when the power is off.

Function	Specification
open	Conforms to Linux standard RTC driver specifications
close	
ioctl	

Note 1) You can set regular date/time settings through a standard Linux interface.
To the set date/time so it shows on an extermal RTC, execute the hwsync function.
If you do not execute hwsync, the setting becomes invalid when the power is turned off.

7.4.9 Status LED Specifications

EMG7-7W / EMG7-12

Applicable models
EMG7-7W
EMG7-12

The status LED on the front of the machine operates as follows:

Status	LED Display Color
Power OFF	Off
Boot loader in operation	Orange
OS startup	Orange
Normal	Green
Backlight off	Flashing green (at 0.5 second intervals)
Backlight problem	Flashing red (at 0.5 second intervals)

Additionally, get the LED status by reading the following files, and set the LED status by writing to the following files.

EMG7-7W

File	Read/Write	Instruction
/sys/class/leds/status_led_green/brightness	R/W	LED green 1: On 0: Off
/sys/class/gpio/status_led_red/brightness	R/W	LED red 1: On 0: Off

EMG7-12

File	Read/Write	Instruction
/sys/class/leds/status_led_green/brightness	R/W	LED green 1: On 0: Off
/sys/class/gpio/status_led_red/brightness	R/W	LED red 1: On 0: Off

Note 1) Although the system itself controls the LED, if you use the files above to set the LED, the system settings are also updated.

However, even if the application sets the LED status, if the backlight turns off or a problem occurs with the backlight, the LED will flash green or flash red accordingly.

7.4.10 SRAM Specifications

Applicable models
EMG7-7W
EMG7-12
EM(G)8-4
EM(G)8-5

EMG7-7W / EMG7-12 can access SRAM as a /dev/mem memory device.
EM(G)8-4 / EM(G)8-5 can access SRAM as a /dev/sram1 SPI device.

Device file: /dev/mem (EMG7-7W / EMG7-12)

Function	Specification
open	Conforms to Linux standard mem driver specifications
close	
mmap	
read	
write	
ioctl	

Device file: /dev/sram1 (EM(G)8-4 / EM(G)8-5)
Extended control code is defined in "seedsware_ext_ioctl.h" of "software"-"ioctrl_include" in DVD-ROM (complete development environment). Please include it before use.

Function	Specification
open	Conforms to Linux standard driver specifications
close	
read	
write	
mmap	Function call specifications are equivalent to Linux standard mmap. Because the interface with SRAM is SPI, the driver secures internal memory and supports quasi-mmap specifications.

ioctl	IOCTL_SRAM_GET_SIZE	Gets the size of SRAM. Return value 0: successful Other than 0: failed Input: none Output: int type variable pointer to the size of SRAM
	IOCTL_SRAM_FLUSH_M2D	Writes to SRAM the total internal memory data area retrieved by mmap. Return value 0: successful Other than 0: failed Input: none Output: none
	IOCTL_SRAM_FLUSH_M2D_PAR	Writes to SRAM the area specified by io_sram_t of the total internal memory data area retrieved by mmap. Return value 0: successful Other than 0: failled Input: io_sram_t type pointer (offset size) Output: none
	IOCTL_SRAM_FLUSH_D2M	Reads out all SRAM data to the internal memory data area retrieved by mmap. Return value 0: successful Other than 0: failed Input: none Output: none
	IOCTL_SRAM_FLUSH_D2M_PART	Reads out the specified SRAM area data to the internal memory data area retrieved by mmap. Return value 0: successful Other than 0: failed Input: io_sram_t type pointer (offset size) Output: none
	IOCTL_SRAM_SEEK	Specify the offset for read/write functions. Return value 0: successful Other than 0: failed Input: int type pointer, offset: 0 to SRAM max value - 1 Output: none

Note 1) The read and write functions use the SPI interface to directly read from and write to SRAM.
Before using the read or write function, set the offset using IOCTL_SRAM_SEEK.
Executing the read or write function does not change the offset.
If the read or write offset changes, specifying the offset with IOCTL_SRAM_SEE is required.

Note 2) The minimum and maximum sizes for offset and read/write functions are as follows:
Minimum size: 1. Maximum size: SRAM size - offset

Note 3) The size requested by mmap must be the size of SRAM.

Note 4) The mmap function only reserves internal memory.
Because internal memory is undefined, use IOCTL_SRAM_FLUSH_D2M to read SRAM data to internal memory.

(A) You can read/write from an application using the pointer to an area reserved with mmap.
If you use the pointer to read or write, no data is written to SRAM.
Use either IOCTL_SRAM_FLUSH_D2M or IOCTL_SRAM_FLUSH_D2M_PART to write internal memory data to SRAM.

(B) If either IOCTL_SRAM_FLUSH_D2M or IOCTL_SRAM_FLUSH_D2M_PART is not run, data from internal memory is flushed when the power is turned off. As a result, SRAM data remains as it was when either IOCTL_SRAM_FLUSH_D2M or IOCTL_SRAM_FLUSH_D2M_PART was last executed.

Note 5) When the battery runs out, the value in the SRAM will be indefinite data.

7.4.11 BUZZER Specifications

Device file: /dev/buzzer

Extended control code is defined in "seedsware_ext_ioctl.h" of "software"-"ioctrl_include" in DVD-ROM (complete development environment). Please include it before use.

Function		Specifications
open		Open / close device
close		
ioctl	IOCTL_PLAY	Sound the buzzer. Return value: Always 0 Input: none Output: none
	IOCTL_STOP	Stop the buzzer. Return value: Always 0 Input: none Output: none
	IOCTL_BEEP	Sound buzzer for the specified duration. Return value 0: successful Other than 0: failed Input: int type variable: buzzer duration (units of 100 ms) Output: none
	IOCTL_SET_BEEP_INTERVAL	Sets the buzzer frequency. Return value 0: successful Other than 0: failed Input: int type variable: frequency (1 - 30000 Hz) Output: none
	IOCTL_GET_BEEP_INTERVAL	Gets the defined buzzer frequency. Return value 0: successful

		Other than 0: failed Input: none Output: int type variable: the defined frequency (Hz)
	IOCTL_GET_BUZZER_STATE	Get the buzzer status. Return value 0: successful Other than 0: failed Input: none Output 0: buzzer off 1: buzzer on

Note 1) Buzzer sound commands are processed as they are received.
As a result, when the buzzer is emitting a sound triggered by IOCTL_BEEP, and IOCTL_BEEP is requested again with a different sound duration, the buzzer is applied with the duration of the second IOCTL_BEEP.

Note 2) If touch sound is enabled on the system, the buzzer may not sound for the intended duration of time if touch sound is triggered as the driver's buzzer and touch sounds are not exclusive processes.
To avoid this issue, disable touch sound on the system, and create a program with the driver at the application level to trigger the touch sound.

7.4.12 DIO Specifications

You can connect switches, LEDs, etc. to the DIO interface of the product.
Depending on the model, control may be performed using the DIO API or normal GPIO.

EM(G)8-4 / EM(G)8-5

Applicable models
EM(G)8-4
EM(G)8-5

For the target model, you can switch the operation mode and perform input/output operations through the DIO API.
Please refer to "7.4.12 DIO Specifications" for DIO API.

There are 3 operation modes:
The default operation mode is "GPIO SCAN mode".

Use the DIO API to change operation modes or modify input/output.
For information about the DIO API, see "7.6.1 DIO API".

DIO mode (with scan)

Using DOUT 4 pins as a SCAN line, and 6 pins as DIN, operate as 24 pin DIN.
Treat as normal DIN input and use the DIO API to get the DIN status of the 24 pins.
In GPIO SCAN mode, SCAN is executed on get.
You can also control 8 pin DOUT with the DIO API.

Number of DIN	Number of DOUT
24	8

DIO mode (without scan)

Operates as 12 pin DOUT and 6 pin DIN.
Use the DIO API to control the 12 pin DOUT and 6 pin DIN.

Number of DIN	Number of DOUT
6	12

Sheet Key Mode

Using DOUT 4 pins as a SCAN line, and 6 pin DIN as a RETURN line, operate as 24 pin sheet key.
4 lines are executed per process, with a processing interval of 100 ms.
The 24 pins are mapped to key codes and recognized as key inputs.
Key inputs are for key codes and ON/OFF status.
Key repeat is not supported.
You can also control 8 pin DOUT using the DIO API.

Number of DIN	Number of DOUT
-	8

EM(G)8-7W / EM8-10W

Applicable models
EM(G)8-7W
EM8-10W

Use the normal GPIO to control the 4 pin DOUT and 4 pin DIN.

Operates with positive logic.(On:1/Off:0)

It can be accessed by reading/writing /sys/class/gpio/gpio<*GPIO number*>/value.

4 pin DOUT

DOUT1	DOUT2	DOUT3	DOUT4
GPIO4_25(gpio121)	GPIO4_26(gpio122)	GPIO4_27(gpio123)	GPIO4_28(gpio124)

4 pin DIN

DIN1	DIN2	DIN3	DIN4
GPIO4_17(gpio113)	GPIO4_18(gpio114)	GPIO4_19(gpio115)	GPIO4_20(gpio116)

7.5 Applications

7.5.1 EMG Launcher

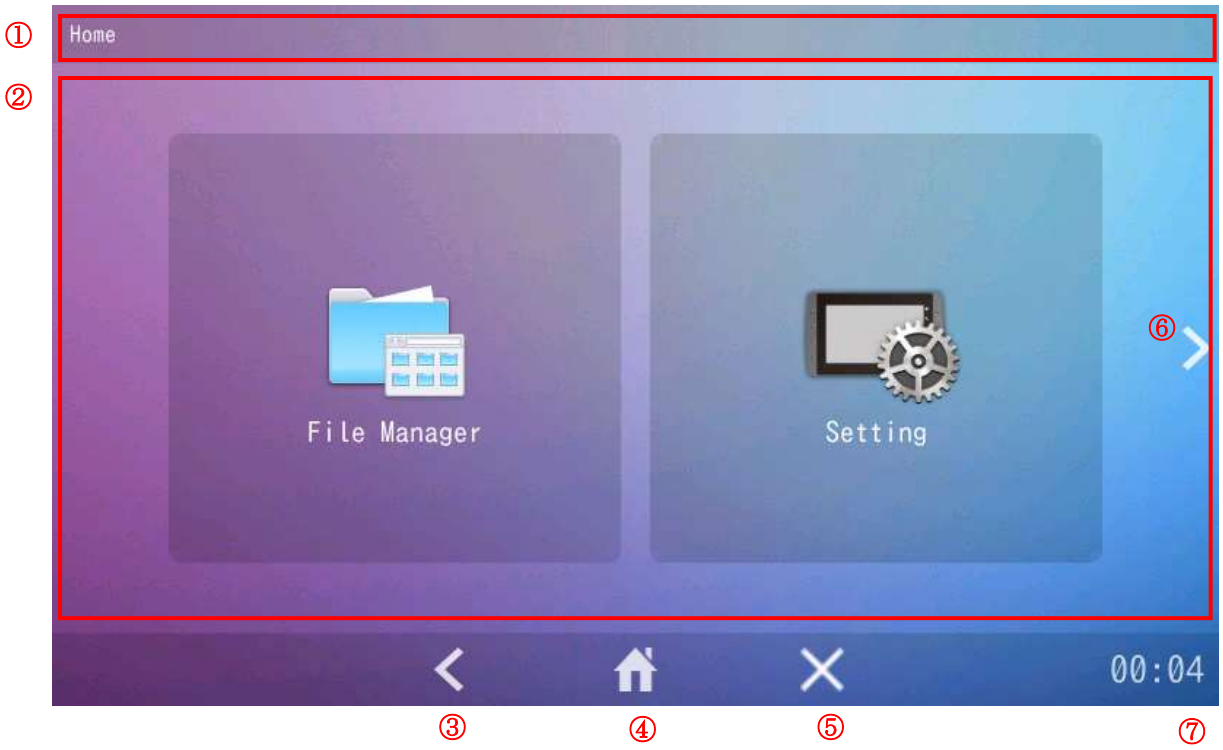
After turning on the power, EMG Launcher starts up automatically. From there launch the System Setting Tool, or register and start your own applications.

In the factory setting, there are 3 registered applications: File Manager, System Setting, and ISApp Setting.

Executable file

/usr/bin/emg_launcher

Basic Operation



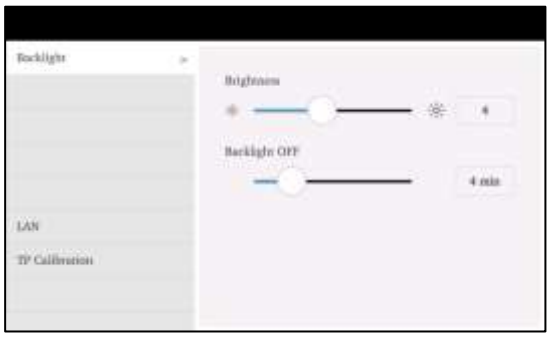
Number	Items	Contents
①	Titlebar	Displays the title of the current screen.
②	Menu Screen	Tap a button to start the corresponding application.
③	Return button	Go back to the previous screen using this button. Nothing happens if you tap this button on the home screen.
④	Home button	Tap this button to jump to the first page.
⑤	Exits button	Exits the EMG Launcher.
⑥	Next page	Displays the next page.
⑦	Clock	Displays the current time.

Application list

Icon	Item Name	Contents
	File Manager	Use the File Manager to display and operate the files stored in EMG7-Linux.
	Setting	Set the IP, clock, and other settings on the machine.
	ISAppSetting	Specify the ISApp launch method, communication settings, and other settings.

Launching Applications

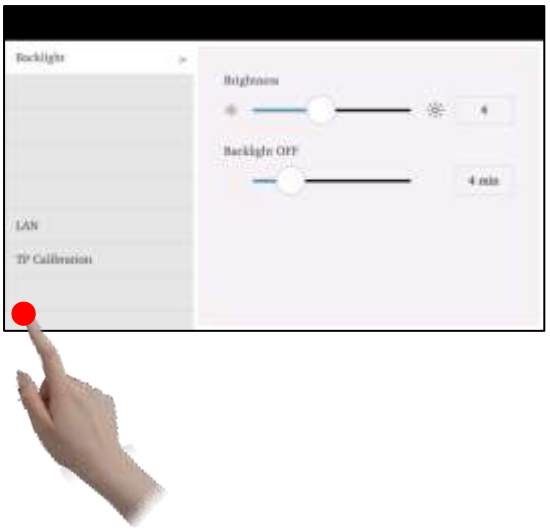
Tap an icon to start the registered application.



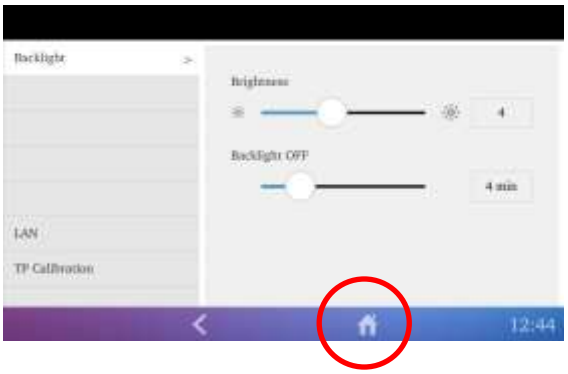
Exiting Applications

The following operation displays the taskbar.

Press and hold the bottom left corner for more than 2 seconds.



Touch  and the current application will stop.



Touch  to hide the taskbar.

Registering Applications

Register an application to the EMG Launcher by adding a desktop entry file in the following folder.

The change takes effect after restarting.

Folder Path	/etc/emg_launcher/applications/

The Desktop Entry file is a data file with information about the button to display on the Home menu. You can create the file using a text editor.

[Common Settings]

Value	Description
Name	Text that is displayed on the button as its application name. (Default)
Name[ja_JP]	Text that is displayed on the button as its application name when the locale is Japanese
Exec	Command that is executed when you tap the button. %f, %F, %u, %U, and other specifications are ignored.
Icon	Icon that is displayed on the button.
Type	Specify "Application".

* The first line must be [Desktop Entry].
* The file extension must be ".desktop".
* Save the file in "UTF-8" character encoding format.
* The file is ignored if any other settings are used.

Example:

```
[Desktop Entry]
Name=Setting
Name[ja_JP]=システム設定
Exec=/usr/bin/emg_setting
Icon=/etc/emg_launcher/icons/menu_tablet_settings.png
Type=Application
```

Deleting Applications

Delete an application from the EMG Launcher by deleting the associated desktop entry file in the following folder.

The change takes effect after restarting

Folder Path	/etc/emg_launcher/applications/

7.5.2 VNC server

To start the VNC server, from the Start menu select [Internet] > [x11VNC Server], or enter the command "/usr/bin/x11vnc".

Remotely connect from a computer that has a VNC client, to display and operate the server.

Executable file

/usr/bin/x11vnc

Options

Use the following command to view the run-time options.

/usr/bin/x11vnc --help

7.5.3 VNC client

Start the client with the command "usr/bin/vncviewer".

Use the VNC client to remotely connect to the computer where the VNC server is installed so you can display and operate the server.

Executable file

/usr/bin/vncviewer

Options

Use the following command to view the run-time options.

/usr/bin/vncviewer –help

7.6 API

7.6.1 DIO API

Applicable models
EM(G)8-4
EM(G)8-5

API for controlling the DIO interface.

Library file

libem_dio.so

Header file

em_dio.h

em_dio-c.h

em_dio-c++.h

em_types.h

The library and header files are located in the DVD-ROM (Development Environment Kit). Copy to the development environment.

Constants

Operation Mode

MODE_SHEETKEY	0	Sheet Key Mode
MODE_DIO_SCAN	1	DIO mode (with scan)
MODE_DIO_NON_SCAN	2	DIO mode (without scan)

Reference: Number of DIN and DOUT in each operation mode

	Number of DIN	Number of DOUT
DIO mode (with scan)	24	8
DIO mode (without scan)	6	12
Sheet Key Mode	-	8

DOUT Output

DOUT_OFF	0	Off
DOUT_ON	1	On

DIN Input

DIN_OFF	0	Off
DIN_ON	1	On

Error Codes

ERROR_CODE_SUCCESS	0x00000000	Successful
ERROR_CODE_INVALID_PARAMETER	0x20000001	Invalid parameter
ERROR_CODE_LIB_NOT_OPEN	0x20000002	Function call before library is open
ERROR_CODE_LIB_OPEN_FAILURE	0x20000011	Failed to open library
ERROR_CODE_LIB_CLOSE_FAILURE	0x20000012	Failed to close library
ERROR_CODE_INVALID_MODE	0x20000013	Operation mode error
ERROR_CODE_DRIVER_INTERNAL	0x20000021	Internal driver error

API (for C++)

Open library

Function	int OpenLib (int mode)
Argument	mode: Specify the operation mode [Operation Mode] MODE_SHEETKEY: Sheet Key Mode MODE_DIO_SCAN: DIO mode (with scan) MODE_DIO_NON_SCAN: DIO mode (without scan)
Return value	[Error Code] Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_INVALID_PARAMETER ERROR_CODE_LIB_OPEN_FAILURE ERROR_CODE_DRIVER_INTERNAL
Function	Open library and make it available.

Note 1) You must call OpenLib before working with the library.

Close library

Function	int CloseLib()
Argument	(None)
Return value	[Error Code] Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_CLOSE_FAILURE
Function	Closes the library.

Get operation mode

Function	int GetMode(int* mode)
Argument	mode: Specify the pointer to the memory of the get mode operation [Operation Mode] MODE_SHEETKEY: Sheet Key Mode MODE_DIO_SCAN: DIO mode (with scan) MODE_DIO_NON_SCAN: DIO mode (without scan)
Return value	[Error Code] Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_NOT_OPEN ERROR_CODE_DRIVER_INTERNAL
Function	Gets the current DIO operation mode.

DOUT Output

Function	int SetDout(int num, int set)
Argument	num: Specify the DOUT number [DOUT number] Sheet Key Mode: 1 - 8 DIO mode (with scan): 1 - 8 DIO mode (without scan): 1 - 12 set: Specify the DOUT output [DOUT output] (negative logic) DOUT_OFF: off DOUT_ON: on

Return value	[Error Code] Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_NOT_OPEN ERROR_CODE_INVALID_PARAMETER ERROR_CODE_DRIVER_INTERNAL
Function	Set the DOUT output.

Get DOUT status

Function	int GetDout(int num, int* val)
Argument	num: Specify the DOUT number [DOUT number] Sheet Key Mode: 1 - 8 DIO mode (with scan): 1 - 8 DIO mode (without scan): 1 - 12 val: Specify the pointer to the memory of the get DOUT status operation [DOUT output] (negative logic) DOUT_OFF: off DOUT_ON: on
Return value	Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_NOT_OPEN ERROR_CODE_INVALID_PARAMETER ERROR_CODE_DRIVER_INTERNAL
Function	Get the current DOUT status.

Get all DIN status

Function	int GetDinAll(DWORD* val)
Argument	val: Specify the pointer to the memory of the get DIN status operation DIO mode (with scan): Bit 31 - bit 24: reserved (fixed to the value 0). Bit 23 - bit 0: DIN24 - DIN1 DIO mode (without scan): Bit 31 - bit 6: reserved (fixed to the value 0). Bit 5 - bit 0: DIN6 - DIN1 [DIN Input] (negative logic) DIN_OFF: off DIN_ON: on

Return value	Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_NOT_OPEN ERROR_CODE_INVALID_MODE ERROR_CODE_DRIVER_INTERNAL
Function	Gets the current status all of the DIN.

Note 1) If the operation mode is Sheet Key mode, the return value is an error.

Get specific DIN status

Function	int GetDin(int num, int* val)
Argument	num: Specify the DIN number [DIN number] DIO mode (with scan): 1 - 24 DIO mode (without scan): 1 - 6 val: Specify the pointer to the memory of the get DIN status operation [DIN Input] (negative logic) DIN_OFF: off DIN_ON: on
Return value	Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_NOT_OPEN ERROR_CODE_INVALID_PARAMETER ERROR_CODE_INVALID_MODE ERROR_CODE_DRIVER_INTERNAL
Function	Gets the current DIN status of the specified pin number.

Note 1) If the operation mode is Sheet Key mode, the return value is an error.

Set key code map

Function	int SetKeyMap(int keyCodeMap, int num)
Argument	keyCodeMap: Specify the pointer to the key code array. num: Specify the number of key codes. [Number of key codes] 24 (fixed)

Return value	Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_NOT_OPEN ERROR_CODE_INVALID_PARAMETER ERROR_CODE_INVALID_MODE ERROR_CODE_DRIVER_INTERNAL
Function	Set up the key code map.

Note 1) If the operation mode is not Sheet Key mode, the return value is an error.

Key code array

int type keyCodeMap[24] Key code array

Relationship with key code positions

	RETURN LINE1	RETURN LINE2	RETURN LINE3	RETURN LINE4	RETURN LINE5	RETURN LINE6
SCAN LINE1	keyCodeMap[0]	keyCodeMap[1]	keyCodeMap[2]	keyCodeMap[3]	keyCodeMap[4]	keyCodeMap[5]
SCAN LINE2	keyCodeMap[6]	keyCodeMap[7]	keyCodeMap[8]	keyCodeMap[9]	keyCodeMap[10]	keyCodeMap[11]
SCAN LINE3	keyCodeMap[12]	keyCodeMap[13]	keyCodeMap[14]	keyCodeMap[15]	keyCodeMap[16]	keyCodeMap[17]
SCAN LINE4	keyCodeMap[18]	keyCodeMap[19]	keyCodeMap[20]	keyCodeMap[21]	keyCodeMap[22]	keyCodeMap[23]

DOUT output (OpenLib and CloseLib processes not required)

Function	int SetDoutDirect(int mode, int num, int set)
Argument	mode: Specify the operation mode [Operation Mode] MODE_SHEETKEY: Sheet Key Mode MODE_DIO_SCAN: DIO mode (with scan) MODE_DIO_NON_SCAN: DIO mode (without scan) num: Specify the DOUT number [DOUT number] Sheet Key Mode: 1 - 8 DIO mode (with scan): 1 - 8 DIO mode (without scan): 1 - 12 set: Specify DOUT output [DOUT output] (negative logic) DOUT_OFF: off DOUT_ON: on

Return value	Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_OPEN_FAILURE ERROR_CODE_INVALID_PARAMETER ERROR_CODE_DRIVER_INTERNAL
Function	Sets the DOUT output.

Note 1) You can call this function without OpenLib. Note that there is some processing load for each function call as the driver is loaded and then released within the function.

Get DOUT status (OpenLib and CloseLib processes not required)

Function	int GetDoutDirect(int mode, int num, int* val)
Argument	mode: Specify the operation mode [Operation Mode] MODE_SHEETKEY: Sheet Key Mode MODE_DIO_SCAN: DIO mode (with scan) MODE_DIO_NON_SCAN: DIO mode (without scan) num: Specify the DOUT number [DOUT number] Sheet Key Mode: 1 - 8 DIO mode (with scan): 1 - 8 DIO mode (without scan): 1 - 12 val: Specify the pointer to the memory of the get DOUT status operation [DOUT output] (negative logic) DOUT_OFF: off DOUT_ON: on
Return value	Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_OPEN_FAILURE ERROR_CODE_INVALID_PARAMETER ERROR_CODE_DRIVER_INTERNAL
Function	Gets the current DOUT status.

Note 1) You can call this function without OpenLib. Note that there is some processing load for each function call as the driver is loaded and then released within the function.

Get all DIN status (OpenLib and CloseLib processes not required)

Function	int GetDinAllDirect(int mode, DWORD* val)

Argument	mode: Specify the operation mode [Operation Mode] MODE_DIO_SCAN: DIO mode (with scan) MODE_DIO_NON_SCAN: DIO mode (without scan) val: Specify the pointer to the memory of the get DIN status operation DIO mode (with scan): Bit 31 - bit 24: reserved (fixed to the value 0). Bit 23 - bit 0: DIN24 - DIN1 DIO mode (without scan): Bit 31 - bit 6: reserved (fixed to the value 0). Bit 5 - bit 0: DIN6 - DIN1 [DIN Input] (negative logic) DIN_OFF: off DIN_ON: on
Return value	ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_OPEN_FAILURE ERROR_CODE_INVALID_PARAMETER ERROR_CODE_DRIVER_INTERNAL
Function	Get all the current DIN status in a block.

Note 1) If you specify Sheet Key mode for the operation mode, the return value is an error.

Note 2) You can call this function without OpenLib. Note that there is some processing load for each function call as the driver is loaded and released within the function.

Get individual DIN status (OpenLib and CloseLib processes not required)

Function	int GetDinDirect(int mode, int num, int* val)
Argument	mode: Specify the operation mode [Operation Mode] MODE_DIO_SCAN: DIO mode (with scan) MODE_DIO_NON_SCAN: DIO mode (without scan) num: Specify the DIN number [DIN number] DIO mode (with scan): 1 - 24 DIO mode (without scan): 1 - 6 val: Specify the pointer to the memory of the get DIN status operation [DIN Input] (negative logic) DIN_OFF: off DIN_ON: on

Return value	Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_OPEN_FAILURE ERROR_CODE_INVALID_PARAMETER ERROR_CODE_INVALID_MODE ERROR_CODE_DRIVER_INTERNAL
Function	Gets the current DIN status of the specified pin number.

Note 1) If you specify Sheet Key mode for the operation mode, the return value is an error.

Note 2) You can call this function without OpenLib. Note that there is some processing load for each function call as the driver is loaded and released within the function.

Get library version

Function	string GetLibVersion()
Argument	(None)
Return value	Version string [Library version] string ver: "1.0.0" (five single-byte characters)
Function	Gets the library version.

Note 1) You can call this function without OpenLib.

API (for C)

Open library (for C language)

Function	int EmDio_OpenLib(int mode)
Argument	mode: Specify the operation mode [Operation Mode] MODE_SHEETKEY: Sheet Key Mode MODE_DIO_SCAN: DIO mode (with scan) MODE_DIO_NON_SCAN: DIO mode (without scan)
Return value	[Error code] Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_INVALID_PARAMETER ERROR_CODE_LIB_OPEN_FAILURE ERROR_CODE_DRIVER_INTERNAL
Function	Open library and make it available.

Note 1) You must call EmDio_OpenLib before working with the library.

Close library (for C language)

Function	int EmDio_CloseLib()
Argument	(None)
Return value	[Error code] Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_CLOSE_FAILURE
Function	Close the library.

Get operation mode (for C language)

Function	int EmDio_GetMode(int* mode)
Argument	mode: Specify the pointer to the memory of the get mode operation [Operation Mode] MODE_SHEETKEY: Sheet Key Mode MODE_DIO_SCAN: DIO mode (with scan) MODE_DIO_NON_SCAN: DIO mode (without scan)
Return value	[Error code] Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_NOT_OPEN ERROR_CODE_DRIVER_INTERNAL
Function	Gets the current DIO operation mode.

DOUT output (for C language)

Function	int EmDio_SetDout(int num, int set)
Argument	num: Specify the DOUT number [DOUT number] Sheet Key Mode: 1 - 8 DIO mode (with scan): 1 - 8 DIO mode (without scan): 1 - 12 set: Specify the DOUT output [DOUT output] (negative logic) DOUT_OFF: off DOUT_ON: on

Return value	[Error code] Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_NOT_OPEN ERROR_CODE_INVALID_PARAMETER ERROR_CODE_DRIVER_INTERNAL
Function	Set the DOUT output.

Get DOUT status (for C language)

Function	int EmDio_GetDout(int num, int* val)
Argument	num: Specify the DOUT number [DOUT number] Sheet Key Mode: 1 - 8 DIO mode (with scan): 1 - 8 DIO mode (without scan): 1 - 12 val: Specify the pointer to the memory of the get DOUT status operation [DOUT output] (negative logic) DOUT_OFF: off DOUT_ON: on
Return value	Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_NOT_OPEN ERROR_CODE_INVALID_PARAMETER ERROR_CODE_DRIVER_INTERNAL
Function	Gets the current DIO status.

Get all DIN status (for C language)

Function	int EmDio_GetDinAll(DWORD* val)
Argument	val: Specify the pointer to the memory of the get DIN status operation DIO mode (with scan): Bit 31 - bit 24: reserved (fixed to the value 0). Bit 23 - bit 0: DIN24 - DIN1 DIO mode (without scan): bit 31 - bit 6: reserved (fixed to the value 0). Bit 5- bit 0: DIN6 - DIN1 [DIN Input] (negative logic) DIN_OFF: off DIN_ON: on

Return value	Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_NOT_OPEN ERROR_CODE_INVALID_MODE ERROR_CODE_DRIVER_INTERNAL
Function	Get all the current DIN status in a block.

Note 1) If the operation mode is Sheet Key mode, the return value is an error.

Get individual DIN status (for C language)

Function	int EmDio_GetDin(int num, int* val)
Argument	num: Specify the DIN number [DIN number] DIO mode (with scan): 1 - 24 DIO mode (without scan): 1 - 6 val: Specify the pointer to the memory of the get DIN status operation [DIN Input] (negative logic) DIN_OFF: off DIN_ON: on
Return value	Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_NOT_OPEN ERROR_CODE_INVALID_PARAMETER ERROR_CODE_INVALID_MODE ERROR_CODE_DRIVER_INTERNAL
Function	Gets the current DIN status of the specified pin number.

Note 1) If the operation mode is Sheet Key mode, the return value is an error.

Set key code map (for C language)

Function	int EmDio_SetKeyMap(int keyCodeMap, int num)
Argument	keyCodeMap: Specify the pointer to the key code array. num: Specify the number of key codes. [Number of key codes] 24 (fixed)

Return value	Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_NOT_OPEN ERROR_CODE_INVALID_PARAMETER ERROR_CODE_INVALID_MODE ERROR_CODE_DRIVER_INTERNAL
Function	Set up the key code map.

Note 1) If the operation mode is not Sheet Key mode, the return value is an error.

Key code array

int type keyCodeMap[24] Key code array

Relationship with key code positions

	RETURN LINE1	RETURN LINE2	RETURN LINE3	RETURN LINE4	RETURN LINE5	RETURN LINE6
SCAN LINE1	keyCodeMap[0]	keyCodeMap[1]	keyCodeMap[2]	keyCodeMap[3]	keyCodeMap[4]	keyCodeMap[5]
SCAN LINE2	keyCodeMap[6]	keyCodeMap[7]	keyCodeMap[8]	keyCodeMap[9]	keyCodeMap[10]	keyCodeMap[11]
SCAN LINE3	keyCodeMap[12]	keyCodeMap[13]	keyCodeMap[14]	keyCodeMap[15]	keyCodeMap[16]	keyCodeMap[17]
SCAN LINE4	keyCodeMap[18]	keyCodeMap[19]	keyCodeMap[20]	keyCodeMap[21]	keyCodeMap[22]	keyCodeMap[23]

DOUT output (EmDio_OpenLib and EmDio_CloseLib processes not required) (for C language)

Function	int EmDio_SetDoutDirect(int mode, int num, int set)
Argument	mode: Specify the operation mode [Operation Mode] MODE_SHEETKEY: Sheet Key Mode MODE_DIO_SCAN: DIO mode (with scan) MODE_DIO_NON_SCAN: DIO mode (without scan) num: Specify the DOUT number [DOUT number] Sheet Key Mode: 1 - 8 DIO mode (with scan): 1 - 8 DIO mode (without scan): 1 - 12 set: Specify the DOUT output [DOUT output] (negative logic) DOUT_OFF: off DOUT_ON: on
Return value	Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_OPEN_FAILURE ERROR_CODE_INVALID_PARAMETER ERROR_CODE_DRIVER_INTERNAL
Function	Set the DOUT output.

Note 1) You can call this function without EmDio_OpenLib. Note that there is some processing load for each function call as the driver is loaded and released within the function.

Get DOUT status (EmDio_OpenLib and EmDio_CloseLib processes not required) (for C language)

Function	int EmDio_GetDoutDirect(int mode, int num, int* val)
Argument	mode: Specify the operation mode [Operation Mode] MODE_SHEETKEY: Sheet Key Mode MODE_DIO_SCAN: DIO mode (with scan) MODE_DIO_NON_SCAN: DIO mode (without scan) num: Specify the DOUT number [DOUT number] Sheet Key Mode: 1 - 8 DIO mode (with scan): 1 - 8 DIO mode (without scan): 1 - 12 val: Specify the pointer to the memory of the get DOUT status operation [DOUT output] (negative logic) DOUT_OFF: off DOUT_ON: on
Return value	Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_OPEN_FAILURE ERROR_CODE_INVALID_PARAMETER ERROR_CODE_DRIVER_INTERNAL
Function	Retrieves the current DIO status.

Note 1) You can call this function without EmDio_OpenLib. Note that there is some processing load for each function call as the driver is loaded and released within the function.

Get all DIN status (EmDio_OpenLib and EmDio_CloseLib processes not required) (for C language)

Function	int EmDio_GetDinAllDirect(int mode, DWORD* val)
Argument	mode: Specify the operation mode [Operation Mode] MODE_DIO_SCAN: DIO mode (with scan) MODE_DIO_NON_SCAN: DIO mode (without scan) val: Specify the pointer to the memory of the get DIN status operation DIO mode (with scan): Bit 31 - bit 24: reserved (value is fixed as 0). Bit 23 - bit 0: DIN24 - DIN1 DIO mode (without scan): bit 31 - bit 6: reserved (value is fixed as 0). Bit 5- bit 0: DIN6 - DIN1 [DIN Input] (negative logic) DIN_OFF: off DIN_ON: on

Return value	ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_OPEN_FAILURE ERROR_CODE_INVALID_PARAMETER ERROR_CODE_DRIVER_INTERNAL
Function	Get all the current DIN status in a block.

Note 1) If you specify Sheet Key mode for the operation mode, the return value is an error.

Note 2) You can call this function without EmDio_OpenLib. Note that there is some processing load for each function call as the driver is loaded and released within the function.

Get individual DIN status (EmDio_OpenLib and EmDio_CloseLib processes not required) (for C language)

Function	int EmDio_GetDinDirect(int mode, int num, int* val)
Argument	mode: Specify the operation mode [Operation Mode] MODE_DIO_SCAN: DIO mode (with scan) MODE_DIO_NON_SCAN: DIO mode (without scan) num: Specify the DIN number [DIN number] DIO mode (with scan): 1 - 24 DIO mode (without scan): 1 - 6 val: Specify the pointer to the memory of the get DIN status operation [DIN Input] (negative logic) DIN_OFF: off DIN_ON: on
Return value	Successful: ERROR_CODE_SUCCESS Failed: ERROR_CODE_LIB_OPEN_FAILURE ERROR_CODE_INVALID_PARAMETER ERROR_CODE_INVALID_MODE ERROR_CODE_DRIVER_INTERNAL
Function	Gets the current DIN status of the specified pin number.

Note 1) If you specify Sheet Key mode for the operation mode, the return value is an error.

Note 2) You can call this function without EmDio_OpenLib. Note that there is some processing load for each function call as the driver is loaded and released within the function.

Get library version (for C language)

Function	char* EmDio_GetLibVersion()
Argument	(None)
Return value	Version string [Library version] char ver[6]: "1.0.0" (five single-byte characters)
Function	Gets the library version.

Note 1) You can call this function without EmDio_OpenLib.

Inquiries

If you have any questions about this document, feel free to contact us.

By E-mail

sales@uscoamerica.com

FAQ

www.seedsware.co.jp/global/support/faq/

7th Edition December, 2021

USCO America Inc.

136 W. Valletts Street, Elmhurst, IL 60126 USA

URL: http://uscoamerica.com/