acontis technologies GmbH

SOFTWARE

LxWin

Windows real-time Hypervisor: run real-time Linux alongside Windows

Product Manual
Edition: 2021-02-10

Compact Table of Contents

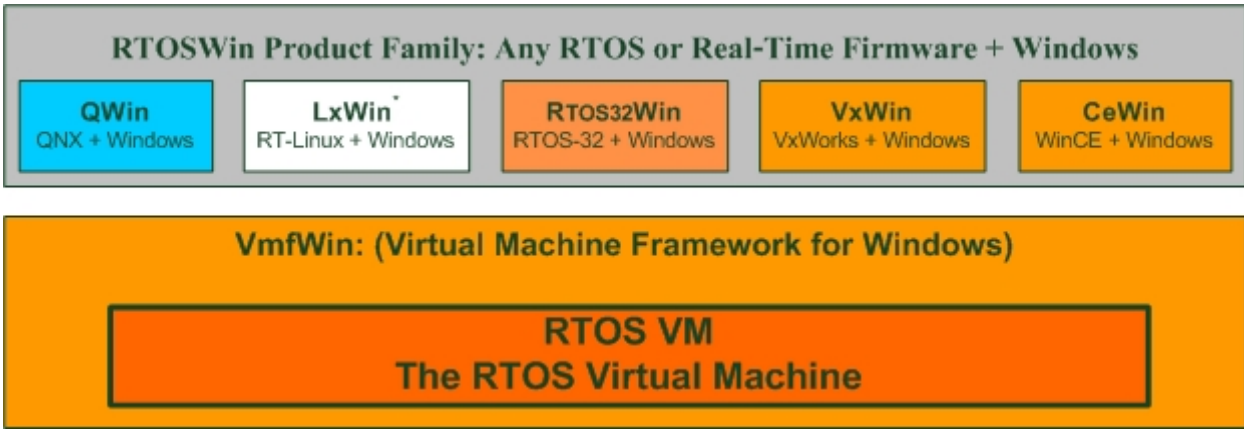
Table of Contents

1 Introduction

1.1 The ACONTIS RTOSWin Product Family

The ACONTIS RTOSWin family is a family of Windows virtualization solutions for multiple real-time operating systems and Microsoft Windows.

The key component of all these solutions is the RTOS Virtual Machine. The real-time operating systems are executed on top of the RTOS VM. How you can see in the following diagram, it's possible to use the real-time firmware LxWin on the top of the RTOS-VM.



acontis technologies GmbH is providing a special Linux Board Support Package (BSP). Using this BSP together with the RTOS VM runtime Linux can be executed together with Windows.

More details about the virtual machine can be found in the RTOS VM User Manual.

- Operating Modes
- Realtime Device Management (how to control hardware)
- RTOS VM configuration
- Booting the RTOS
- Communication Services: The RTOS Library

1.2 The ACONTIS RTOS Virtual Machine

The ACONTIS RTOS-VM provides a light-weight real-time virtualization platform for Windows.

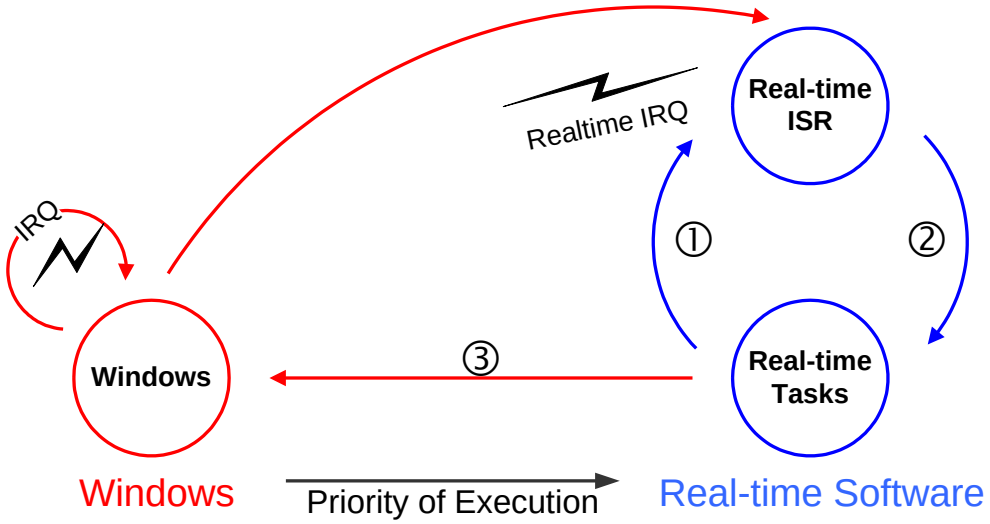
On top of this platform either real-time firmware, custom or off-the-shelf real-time operating systems can be executed.

When using multicore CPUs one can choose between two general operation modes. A more detailed description about possible operation modes can also be found in the RTOS VM User Manual.

1.2.1 Shared Mode Operation

Windows shall run on all CPU cores and only one CPU core shall additionally run the real-time software. If the Windows application needs a lot of CPU power (e.g. for image processing) this will be the appropriate operation mode even on multi-core CPUs. In shared mode operation Windows (on this core) will usually only get CPU time when the real-time software is idle.

The following diagram illustrates the flow of control:



Operating states of the RTOS-VM in shared mode

① Exception-handling or a higher priority interrupt becomes outstanding.

② Interrupt Service Routine optionally starts a new task and then finishes.

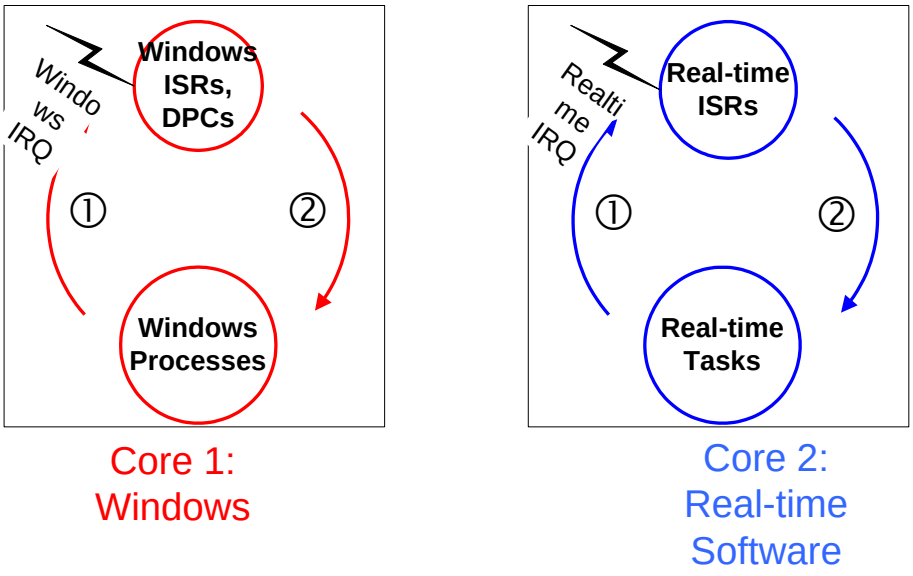
③ From the idle-state, Linux transfers control to Windows operating system.

Note: When running the RTOS-VM in shared mode on multiprocessor/multicore systems this state diagram is only applicable for one CPU core in the system (by default on the first core). All other CPU cores will run Windows only.

1.2.2 Exclusive Mode Operation

Windows and the real-time software shall run fully independently on different CPU cores. Using this mode will lead to much shorter interrupt and task latencies as there is no need to switch from Windows to the real-time software.

The following diagram, illustrates the flow of control on a dual core system:



Operating states of the RTOS-VM in exclusive mode

① Exception-handling or a higher priority interrupt becomes outstanding.

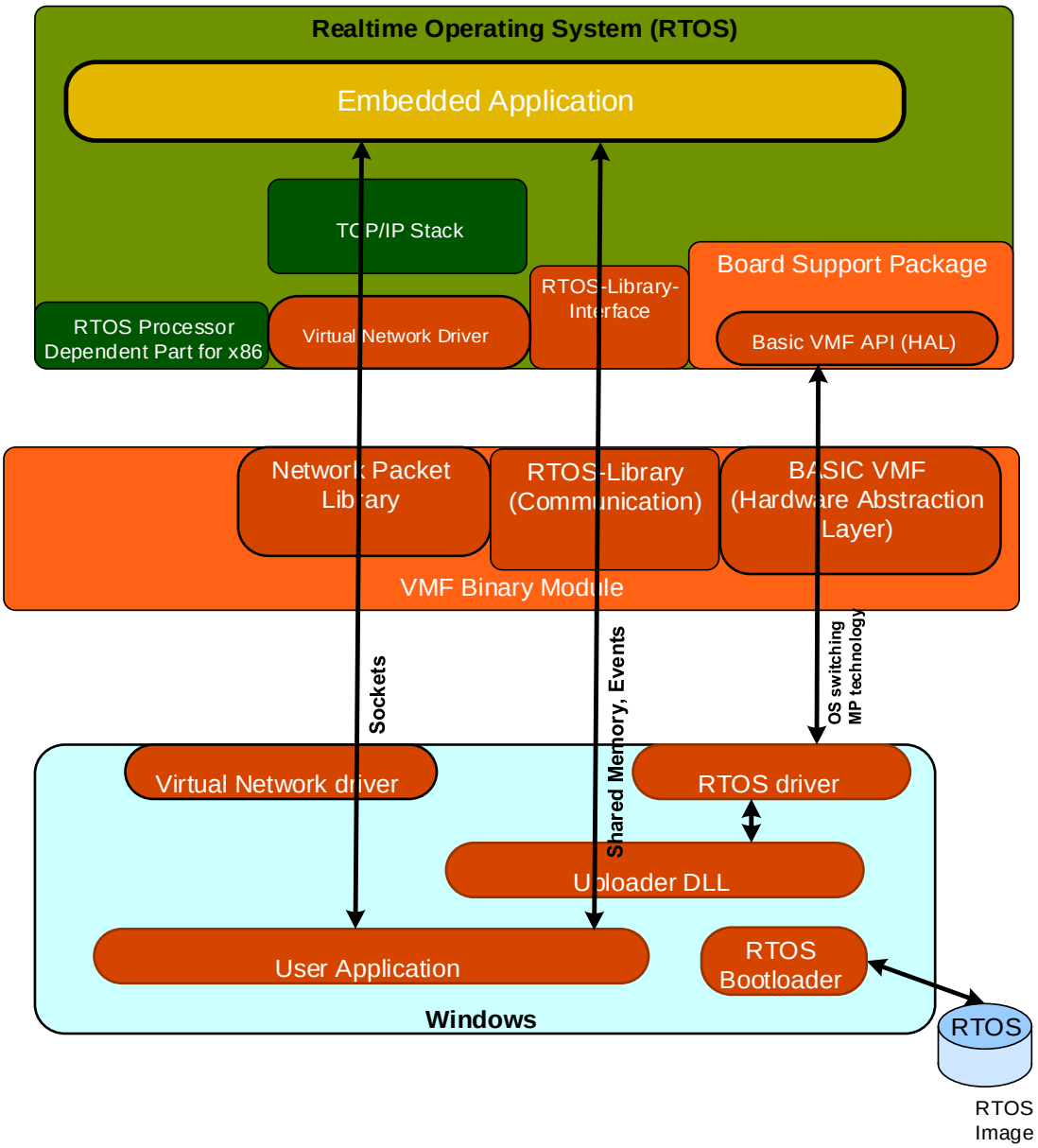
② Interrupt Service Routine optionally starts a new task and then finishes.

Note: When running the RTOS-VM in exclusive mode Windows will never be interrupted. Application and interrupt processing run concurrently and independently on both CPU cores. There is no need in the real-time software to enter the idle state.

1.3 LxWin Architecture Overview

LxWin is split into two main components:

a) The ACONTIS RTOS Virtual Machine runtime (VMF runtime)
b) The Linux BSP. This component is also provided by acontis technologies.



1.4 Components

1.4.1 Development components

To develop software for LxWin the following components are provided.

a) LxWin BSP for Linux (Yocto based)
   Important:
   The LxWin BSP is not part of the LxWin setup program. Please contact Customer Support for the latest LxWin BSP adequate to your desired Linux version.

b) LxWin development components (SDK + Documentation)
   The following files are shipped with LxWin for development support:
   - LxWin User Manual (this document)
   - RTOS VM User Manual (basic technology description)
   - RTOS Library for Windows/RTOS communication
   - SDK with Example applications

1.4.2 Runtime components

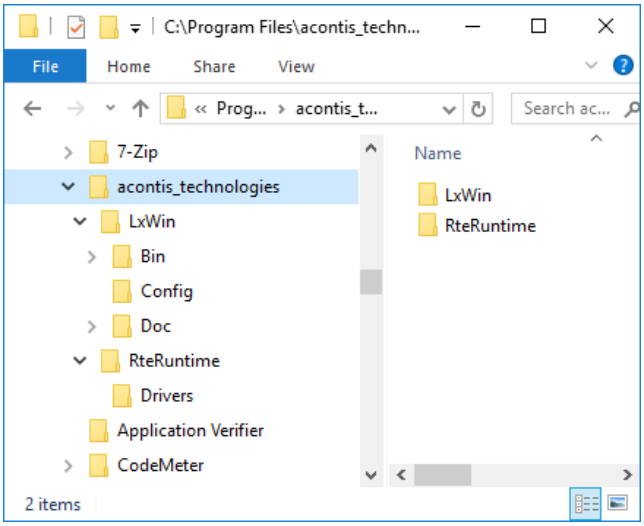
The runtime components are split into the following main parts:

a) LxWin runtime components (ACONTIS RTOS Virtual Machine)
b) Configuration files.
   For more information about the Configuration files see chapter 2.1.
c) Linux LxWin runtime images (this is the image built by Linux + LxWin BSP)

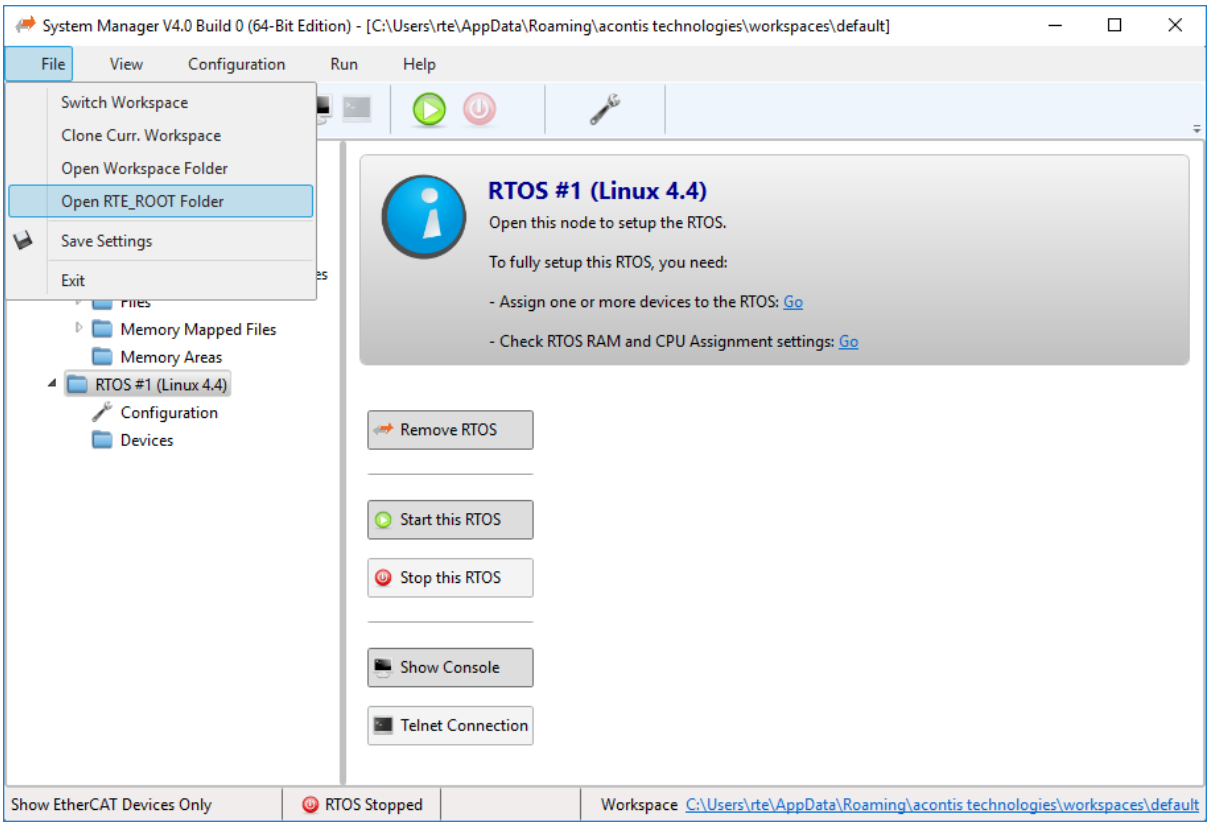
2 LxWin configuration settings

2.1 System Manager

The LxWin System Manager is used to create a LxWin runtime environment. This runtime environment is stored beneath a specific workspace and, among other files, includes all configuration settings needed.
If a new workspace is created, all configuration settings are copied from the LxWin installation directory (…\LxWin\Config) into the workspace configuration directory.

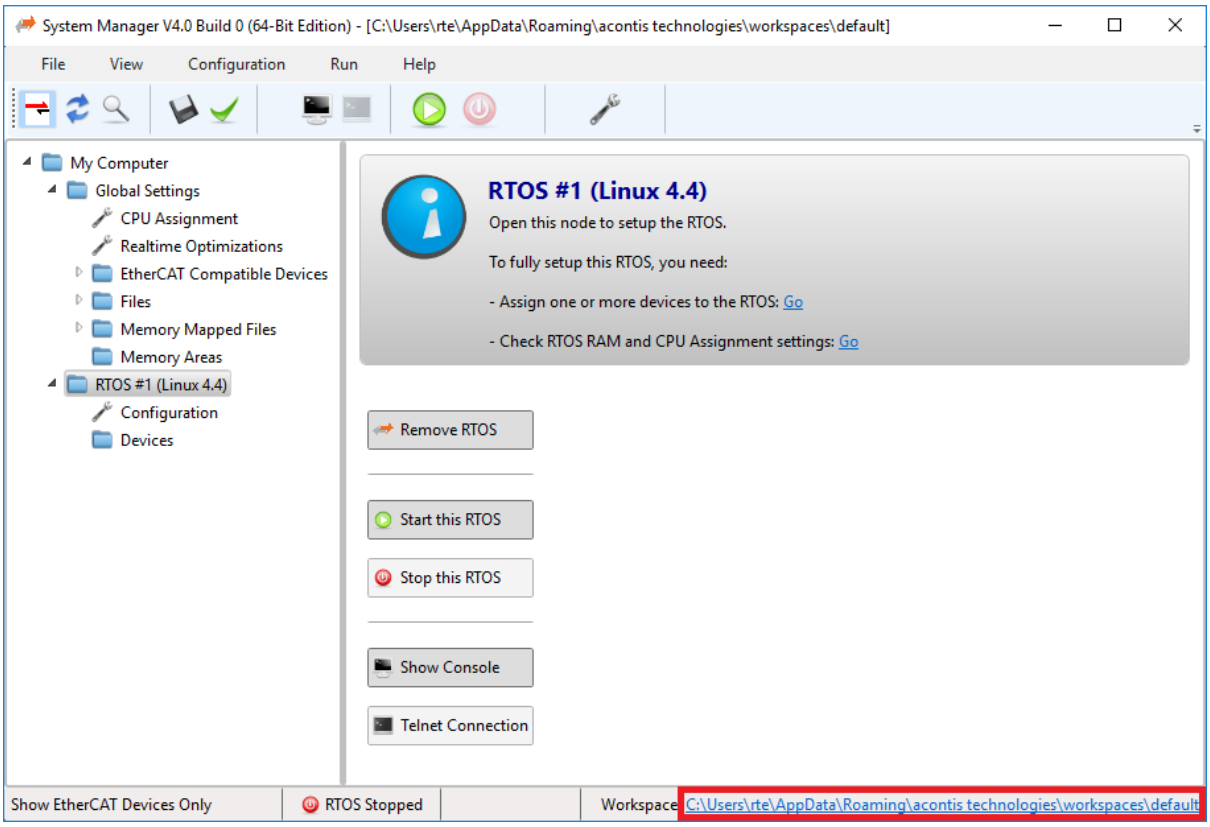


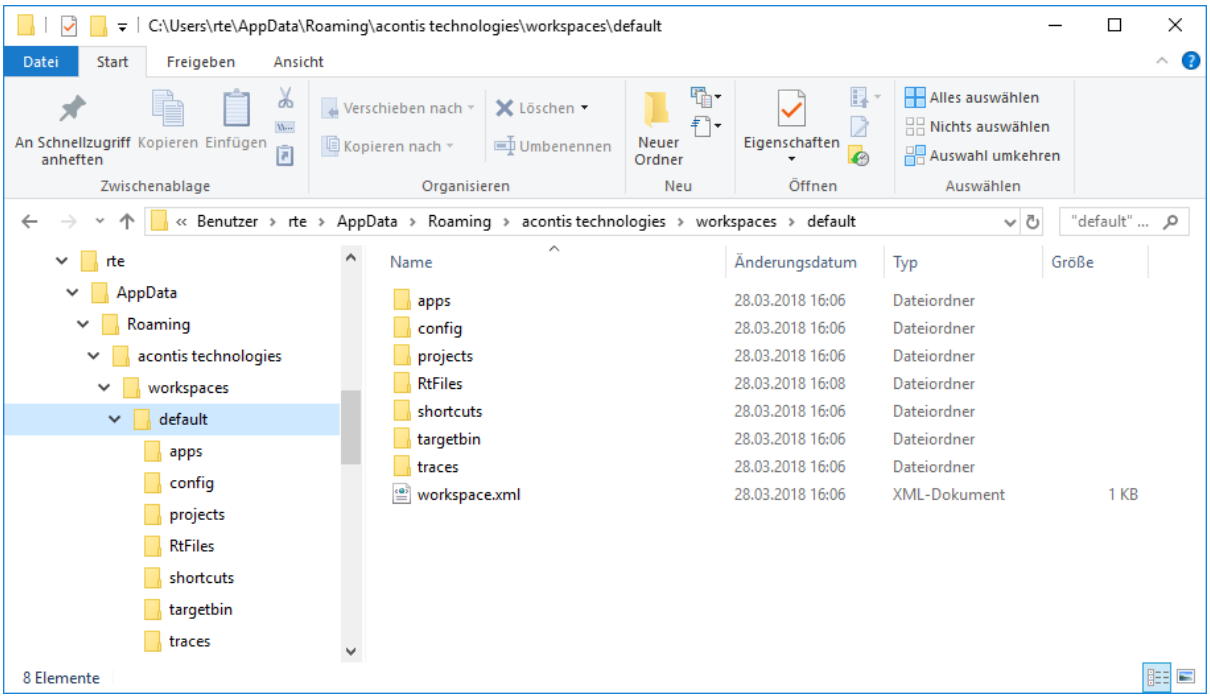
You may open the RTE_ROOT folder (…\RteRuntime) from within the System Manager



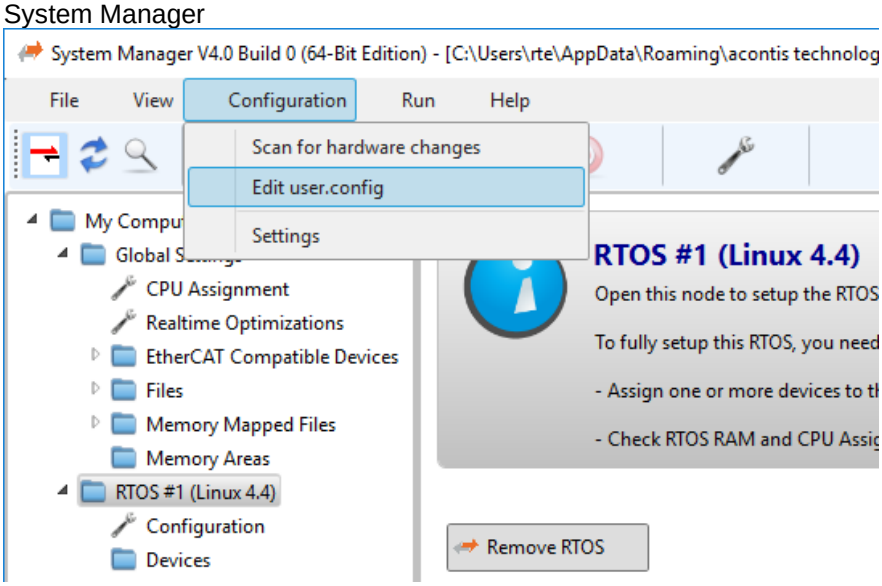
2.2 LxWin configuration files

If a new workspace is created, all configuration settings are stored in the config directory inside the workspace. You can simply open the Windows Explorer from within the System Manager and directly switch into the workspace directory.





The root configuration file is the startup.config file which includes other configuration files.
The main configuration is stored in the systemmgr.config file which is automatically adjusted by the System Manager. Please do not adjust this file, it will be overwritten by the System Manager. All user settings have to be stored in the systemmgr_user.config file which you can open from within the System Manager



2.3 Syntax

The configuration file is an ASCII file that can be modified with a simple editor. Its syntax is similar to that of a Windows registry file.

2.3.1 General rules

- The first entry in a configuration file must be *RtosConfig*.
- Because comments are introduced by a semicolon (;), all characters following a semicolon will be ignored.
- No single line in the configuration file may exceed 256 characters.

2.3.2 Keys, Entries, and Values

The configuration settings are stored using keys. Specific settings are stored in entries. Every entry has both a name and a value. Every entry is subordinated to a specific key.

Example:

```
[Key]
      "EntryName"=EntryValue
```

There are different types of values:

- Single hexadecimal value (dword)

  "EntryName"=dword:1F

- Strings

  "EntryName"="This is a string"

- Multiple Strings

  "EntryName"=multi_sz:"First string","Second string","Last string"

- Multiple hexadecimal values

  "EntryName"=hex:XX,YY,…,ZZ

2.3.3 Include statements

Configuration commands and parameters can be split into multiple files. You may use an *include* statement to incorporate other configuration files into *startup.config*.

Examples:

```
#include "myOs.config"
#include "myApp.config"
```

Important notes:

If no additional path information is given, included files must be located in the same directory as *startup.config*.

Since nested includes are not supported, *include* statements may only be used in the main configuration file.

Common configuration files:

Filename	Meaning
startup.config	Main configuration file – contains includes to other configuration file(s) and maybe some common parameters
systemmgr.config	General configuration file – contains information for the Uploader Utility
systemmgr_user.config	User configuration file – contains user configuration

2.4 LxWin specific configuration parameters

All network configuration parameters are stored in systemmgr.config file under the [***Rtos\Vnet\0***] key. The user should not change systemmgr.config file, but can move required setting to the systemmgr_user.config file:

Entry Name	Type	Description
MacAddress	String	Virtual Network Adapter MAC address.
IpAddress	String	IP address for the LxWin virtual network adapter
PollingPeriodMs	dword	The polling period can be adjusted using this parameter. A value of 0 enables the interrupt mode.

Under the [***Rtos\Autostart\1***] key you can define optional autostart parameters:

Entry Name	Type	Description
Executable	string	Name and path of the executable file
Parameter	string	Parameters of the executable

The [***Rtos\Linux***] key contains Linux specific settings. The followings entries are optional and provides possibility to disable some LxWin components:

Entry Name	Type	Description
novnet	string	If “yes” or “y”, VNET network will be not initialized.
nossh	string	If “yes” or “y”, OpenBSD Secure Shell server will not be started
notelnet	string	If “yes” or “y”, Telnet server will not be started
nortosfs	string	If “yes” or “y”, real time file system will be not initialized and no mount for /mnt/rtfiles will be created.
nortosservice	string	If “yes” or “y”, RTOS service will not be started
cmd_line	string	Command line string passed to the Kernel. Default value is nopti

The [***Rtos\TimeSync***] key contains time synchronization settings. Linux has here some own specific parameters. Please note, Rtos/TimeSync and Windows/TimeSync can have other parameters as well, valid for Linux and all other operating systems. Please refer to “RtosVM User Manual” to get details.

Entry Name	Type	Description
TimezoneLinux	string	Contains timezone for Linux, as in “TZ” Linux environment variable, ex. “Europe/Berlin”. By default, Linux starts with UTC timezone. All existing time zones could be found in /usr/share/zoneinfo directory. if this parameter is set and TaskEnabled parameter is not 0, boot script creates a link to the specified time zone binary file from /etc/localtime. Note: this timezone should be exactly the same as in Windows by meaning, but Windows and Linux timezones have different string representation. Example: Linux: Europe/Berlin Windows: (UTC+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna
TaskEnabled	dword	0 Task will not be started 1 Task will be started 2 Task will be started, init time and timezone once and then finish. omitted = 0 = Task will be started. It differs from omitted value from other OSes defined in “RtosVM User Manual”

3 Tutorials

3.1 Installing LxWin

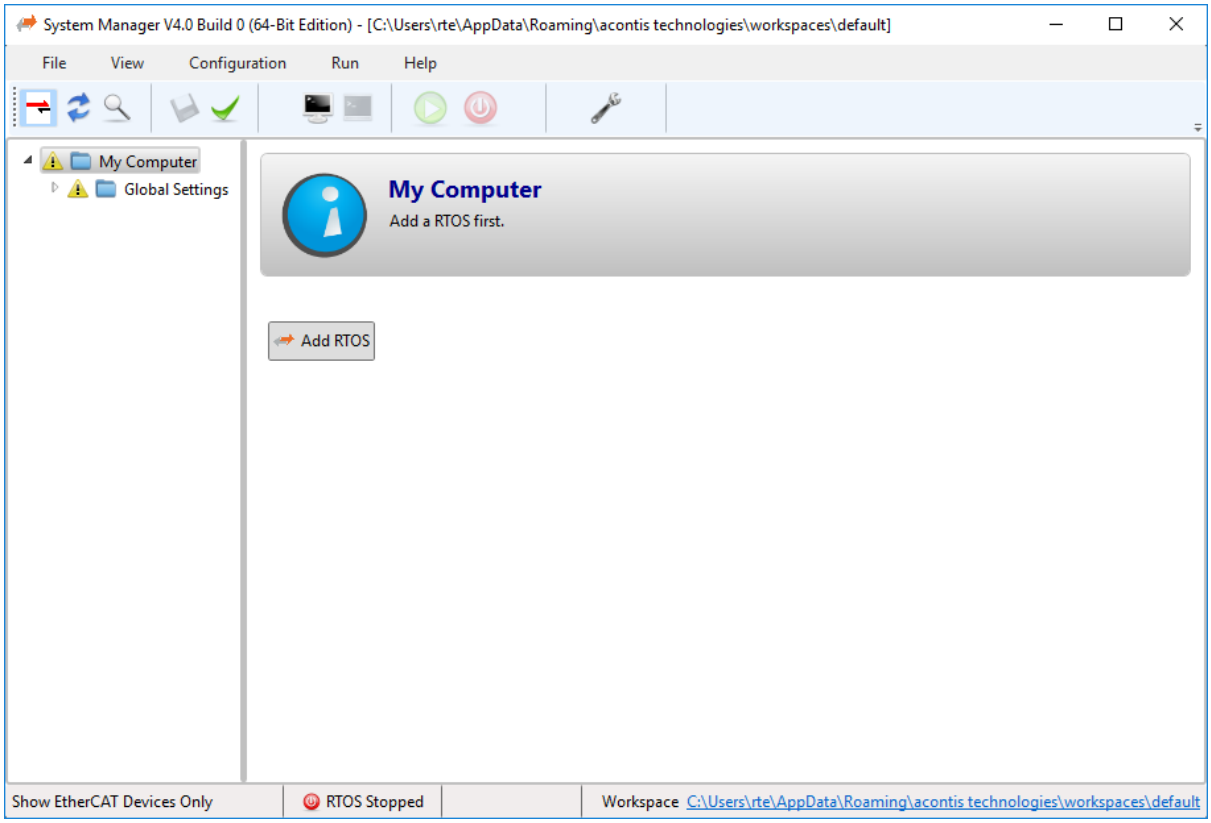
This tutorial shows how to install LxWin.

- Start LxWin.exe
- Agree with license and proceed to install
- Installer verifies if required windows updates installed.
- Restart the PC if needed.

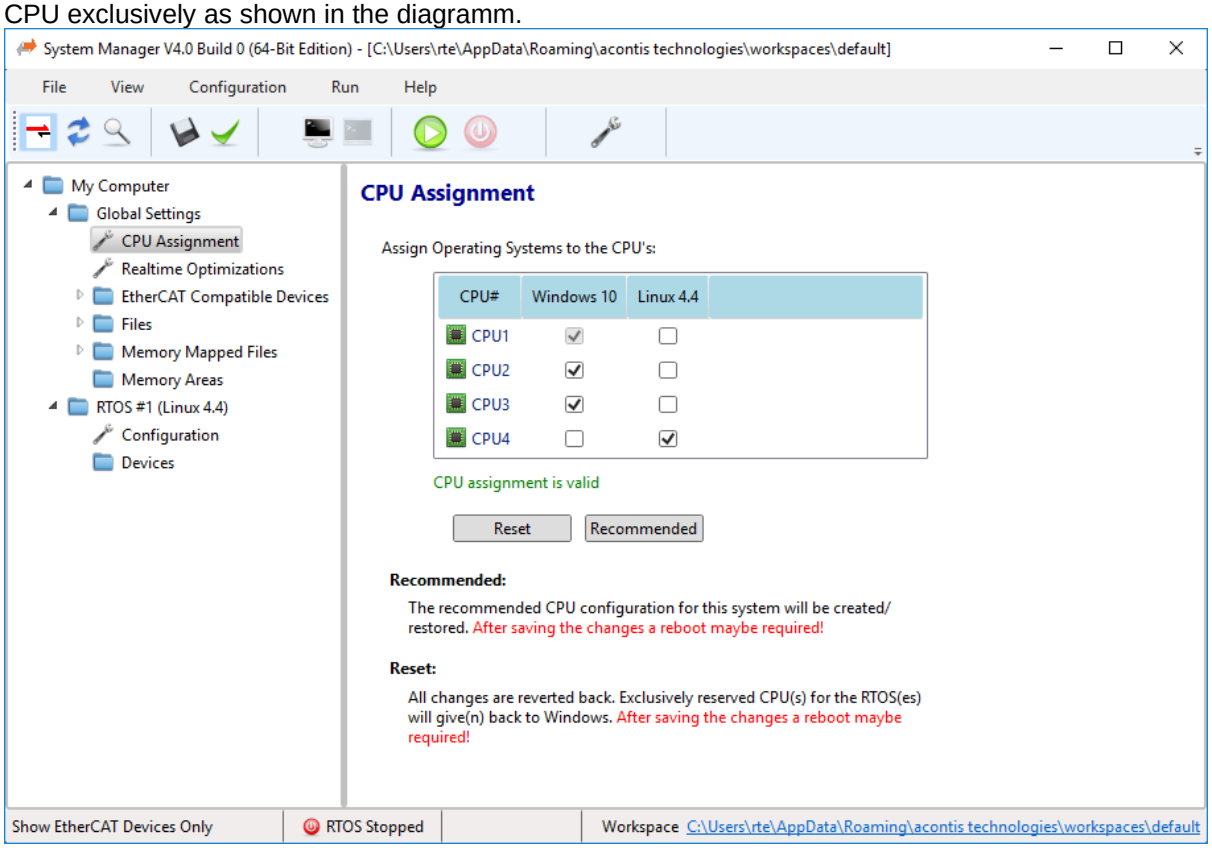
3.2 Running the shipped Linux image

This tutorial shows how to run a shipped Linux image. It assumes that LxWin is installed on the PC.

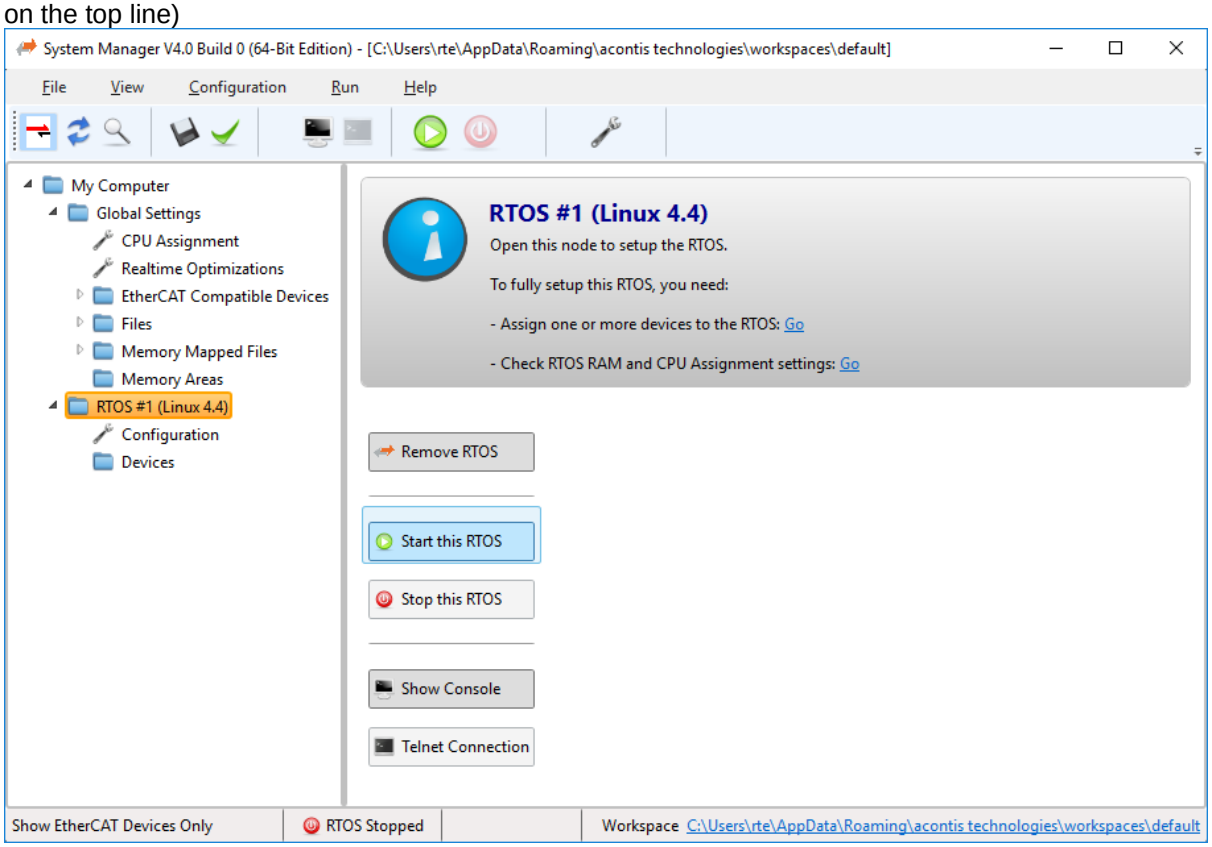
- Start the System Manager
    - If it's the first launch of the System Manager opens a dialog to enter a workspace directory.
- Select "My Computer" node on the tree view and add an RTOS to the configuration by the Button on the right.



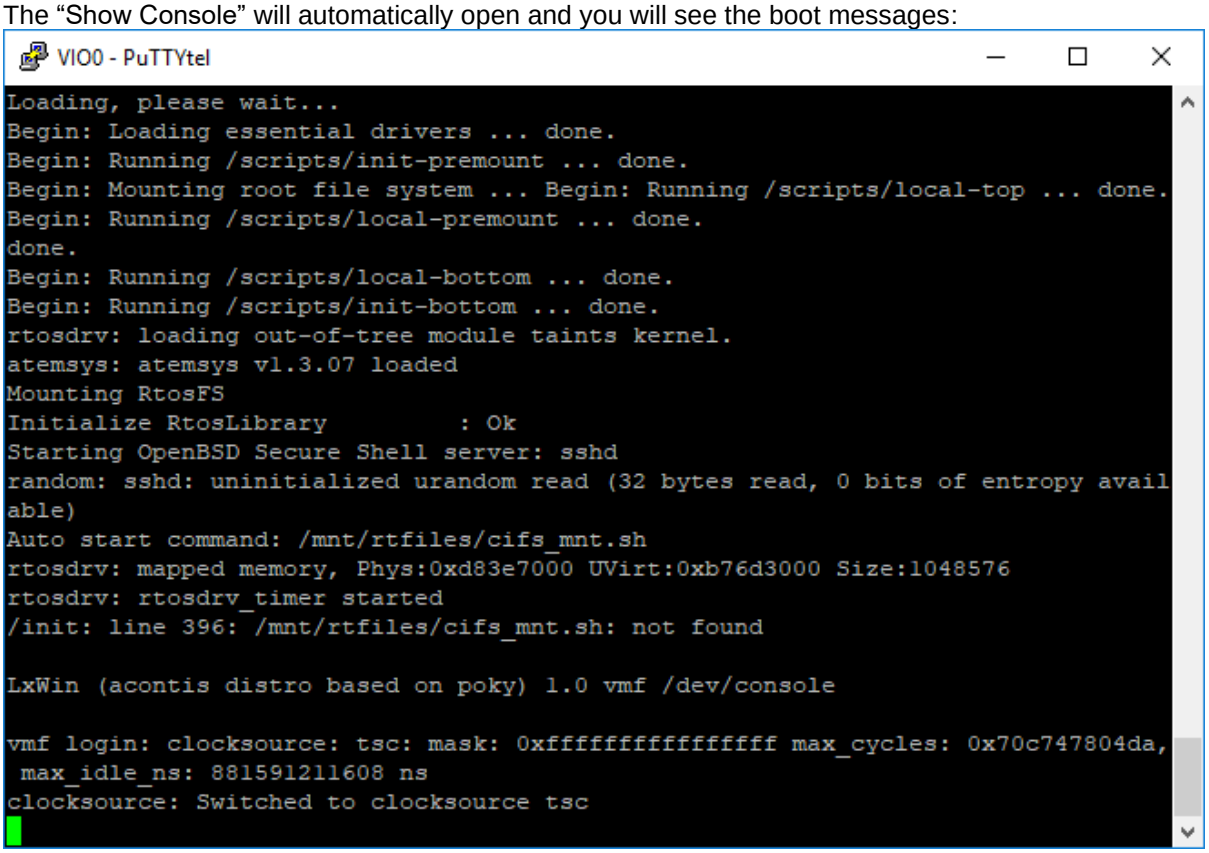
- Verify “CPU assignment” are set correctly. Note. For Windows x64 you should give Linux one CPU exclusively as shown in the diagramm.



- Select the RTOS #1 section and push the “Start this RTOS” button (or the green start button on the top line)



- The “Show Console” will automatically open and you will see the boot messages:



- You can log in into the Linux OS shell. Username and password are both **root**
- Enter “RealtimeDemo” to start the RealtimeDemo-Application
- To stop the RealtimeDemo-Application use [ctrl]+[c]

3.3 How to work with Eclipse

Eclipse is one of the most popular Linux development tools.

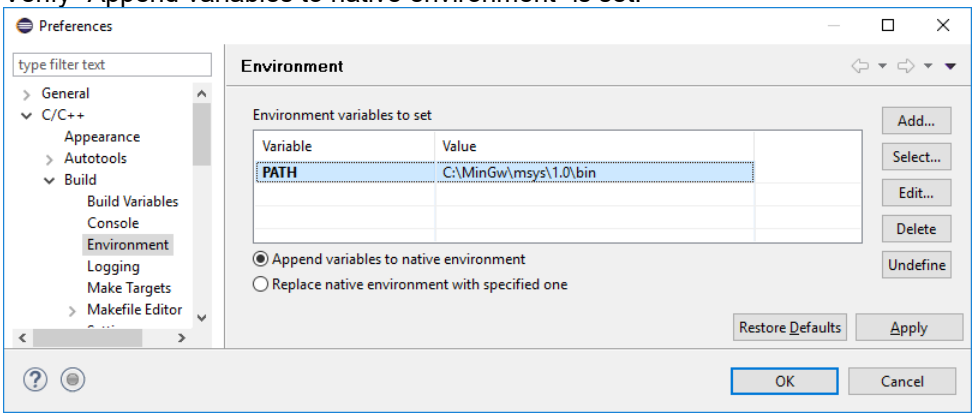
A ready-to-run solution including the appropriate toolchain for LxWin can be downloaded from acontis website. Please download http://software.acontis.com/LxWin/eclipse.zip and http://software.acontis.com/LxWin/MinGW.zip.

For x64 development download toolchain http://software.acontis.com/LxWin/MinGw64.zip

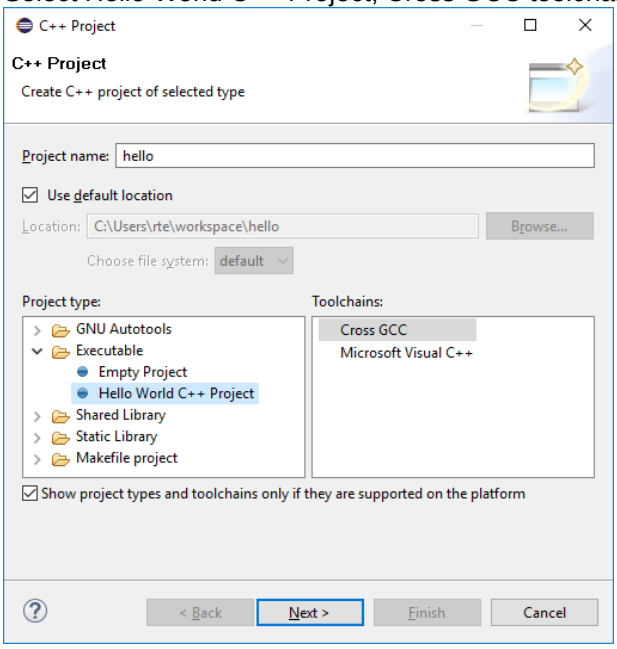
- Extract both zip files into C:\ (if you want extract into another directory, ensure there are **no blanks**!)
- Create a shortcut into the Windows Start menu to C:\eclipse\eclipse.exe

3.3.1 Create a new project

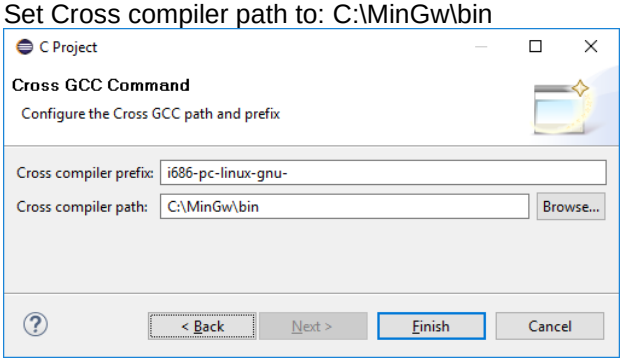
- Start Eclipse
- Add MSYS path to the PATH variable:
  - Windows - Preferences
  - Open C/C++ - Build – Environment.
  - Click Add button,
    set Name to: PATH
    set Value as C:\MinGw\msys\1.0\bin
    for x64 toolchain set Value as C:\MinGw64\bin
  - Verify “Append variables to native environment” is set:



- Create a new project:
  - File – New C++ Project
  - Select Hello World C++ Project, Cross GCC toolchain



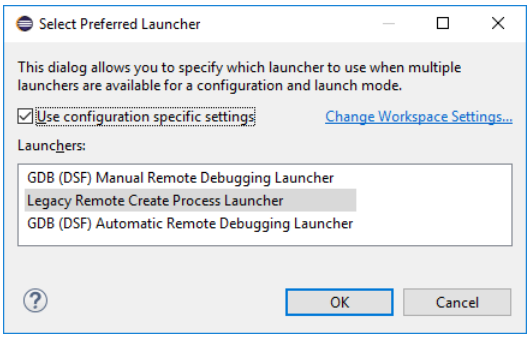
  - Press Next until you reach the Cross GCC Command configuration dialog
    Set Prefix to: i686-pc-linux-gnu-
    Set Cross compiler path to: C:\MinGw\bin



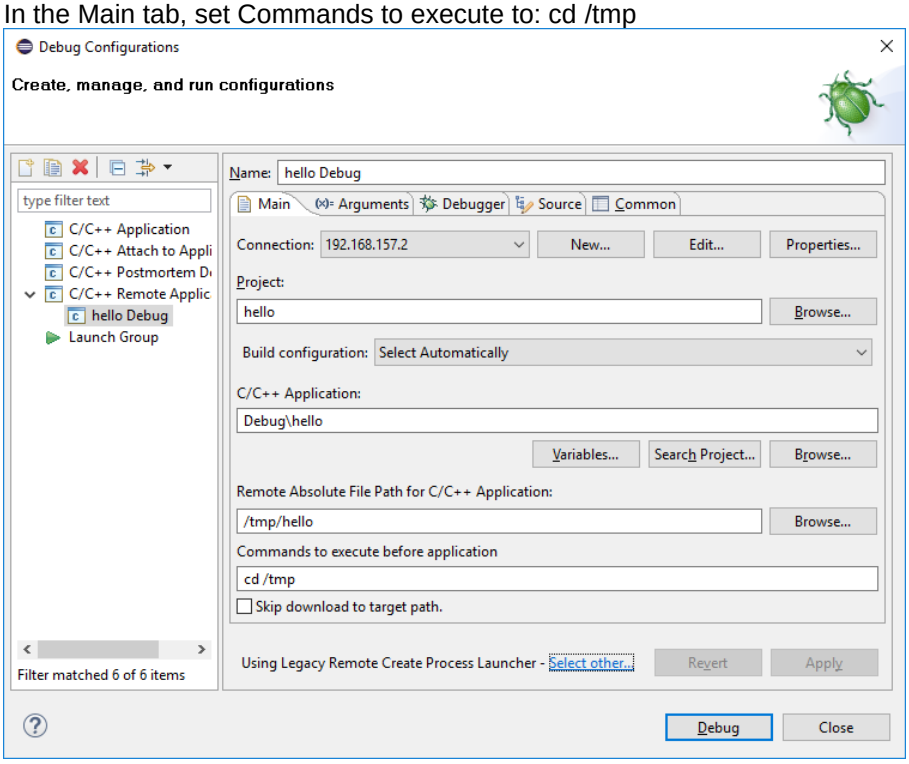
  - For x64 toolchain set Prefix to: x86_64-pc-linux-gnu-
    Set Cross compiler path to: C:\MinGw64\bin
  - Ignore the Eclipse problems that “g++” and “gcc“ not found in PATH.
  - Now you can build the project

3.3.2 Create a new debug configuration

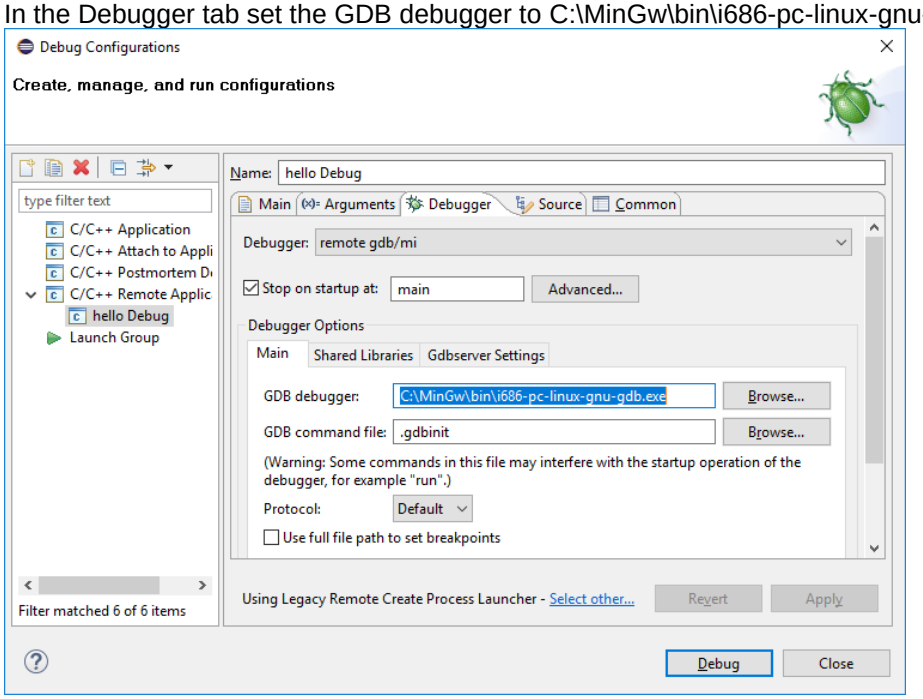
- Run – Debug Configurations, then select C/C++ Remote Application and press the ‘New” button
- In the Main tab at the bottom in the Debug Configurations dialog, Press Select other… and then choose the “Legacy Remote Create Process Launcher”



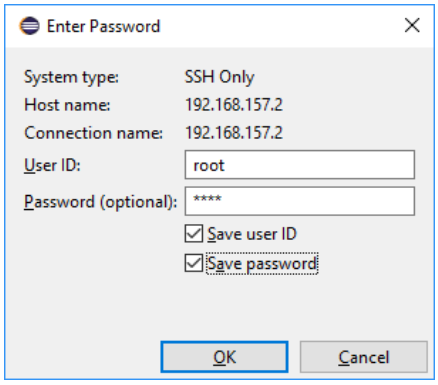
- In the Main tab, press the New… button to create a new Connection, then select SSH Only, Host name = 192.168.157.2 (the IP address of the LxWin target)
- In the Main tab, set the Remote Absolute File Path to /tmp/%ExecutableName% (in this example it is /tmp/hello)
- In the Main tab, set Commands to execute to: cd /tmp



- In the Debugger tab set the GDB debugger to C:\MinGw\bin\i686-pc-linux-gnu-gdb.exe



- For x64 toolchain set the GDB debugger to C:\MinGw64\bin\x86_64-pc-linux-gnu-gdb.exe
- Start the LxWin System Manager, add the Linux RTOS and start LxWin. Assure you can successfully connect to it via TCP/IP (e.g. run ping 192.168.157.2 at the Windows command line)
- Enter User ID: root and Password: root and press OK and acknowledge the next few questions



- Now you can debug the project

3.4 How to work with Visual Studio

You may use the third party VisualGDB solution for development and debugging of LxWin applications using Microsoft Visual Studio.

An evaluation license can be obtained here: http://visualgdb.com/download

After installing VisualGDB restart Visual Studio to get the latest VisualGDB package updates.

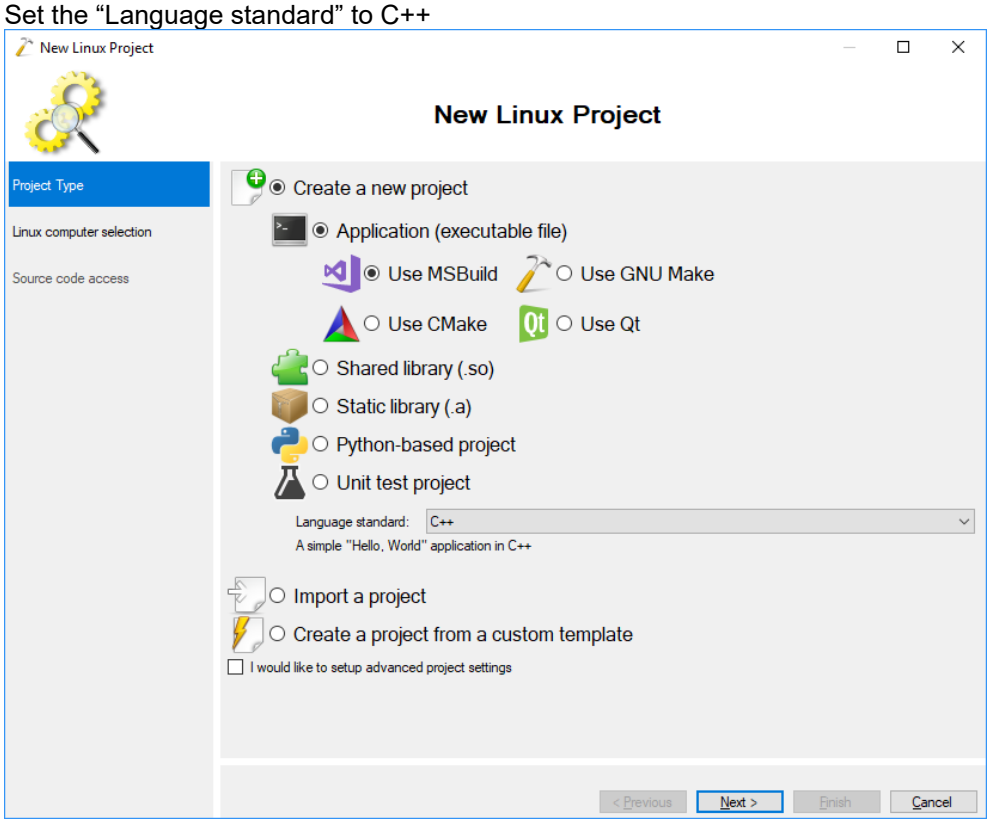
The toolchain to be used can be downloaded from http://software.acontis.com/LxWin/MinGW.zip

For x64 development download toolchain http://software.acontis.com/LxWin/MinGw64.zip

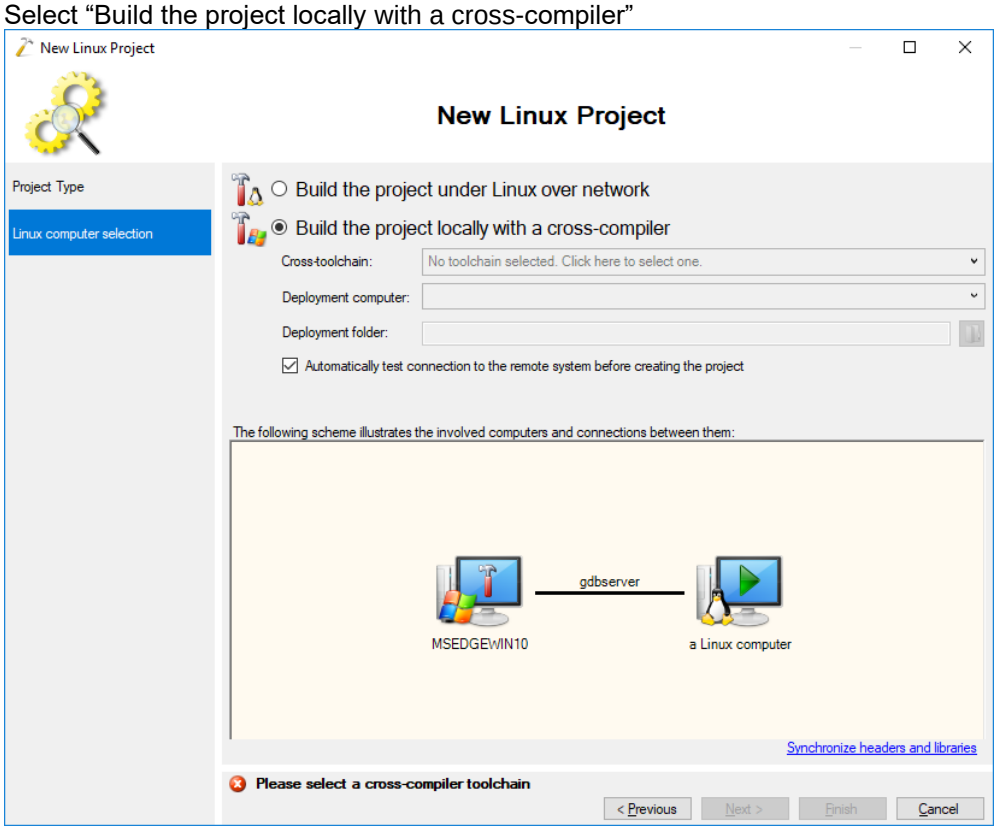
- Extract the zip file into C:\ (If you want extract into another directory, ensure there are **no blanks**!)

3.4.1 Create a new project

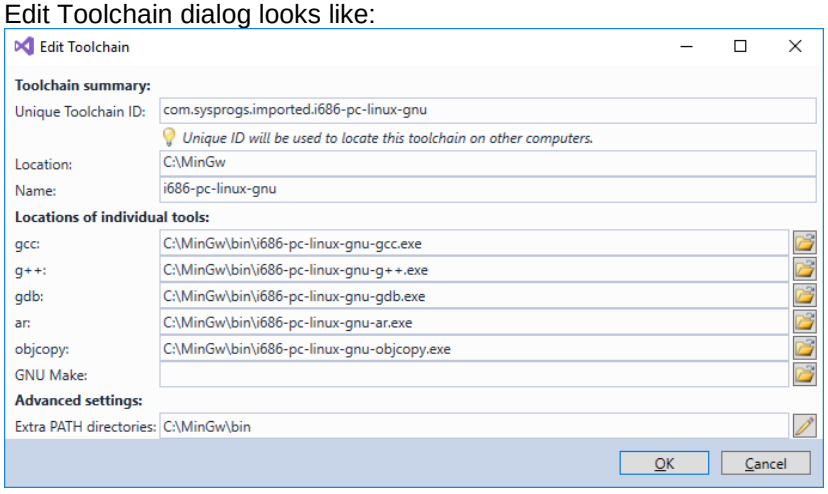
- Start the LxWin System Manager, add the Linux RTOS and start Linux. Assure you can successfully connect to it via TCP/IP (e.g. run ping 192.168.157.2 at the Windows command line)
- Start Visual Studio
- Create a new VisualGDB project by using the “Linux Project Wizard”
- Set up the project as Application and use MSBuild
- Set the “Language standard” to C++



- Select “Build the project locally with a cross-compiler”

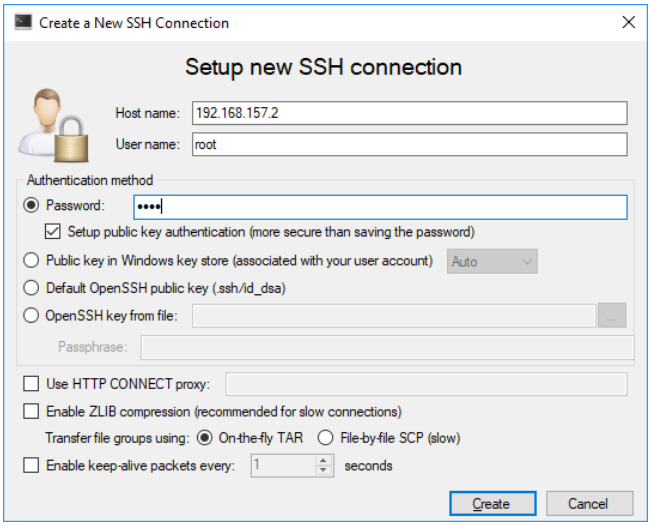


- In the Cross-toolchain field select “Locate a cross-toolchain by finding its gdb.exe” and select C:\MinGw\bin\i686-pc-linux-gnu-gdb.exe
- For x64 toolchain select C:\MinGw64\bin\x86_64-pc-linux-gnu-gdb.exe
- Edit Toolchain dialog looks like:

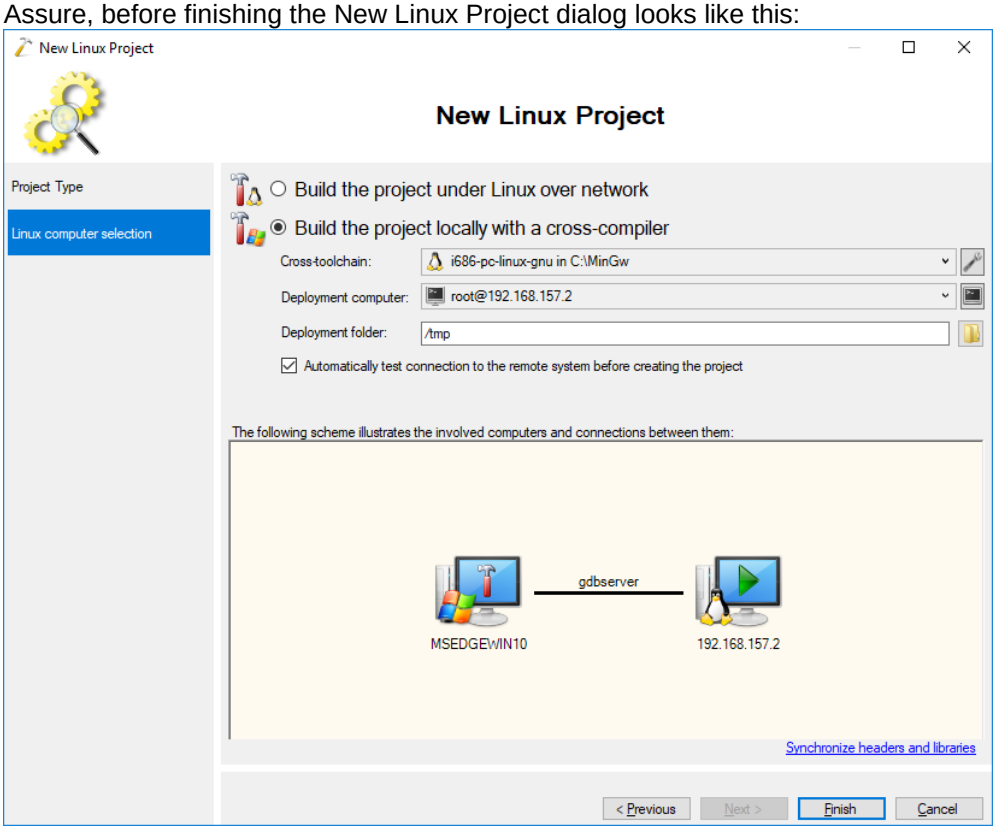


- In the “New Linux Project”-View, click the drop-down-field “Deployment computer” to create a new SSH connection

- Assure LxWin is started before you create the SSH connection!
  As host name use the IP address of the Linux target (192.168.157.2). User name and password are both “root”.



Assure, before finishing the New Linux Project dialog looks like this:



Then press Finish. Accept the “Mismatching environment detected” message with the OK Button.

- Now you can debug the project

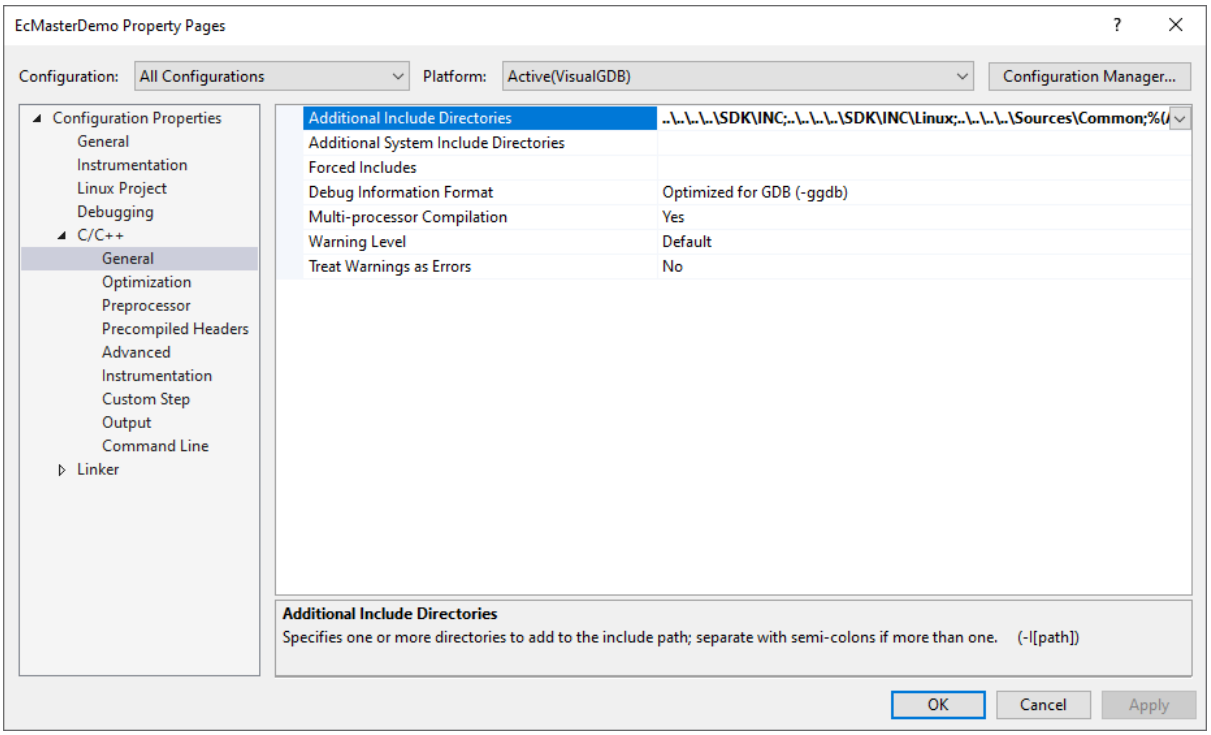
3.4.2 Project settings for the acontis EtherCAT master stack

The following settings show how to set include directories, libraries and debug settings appropriate for the demo application shipped together with the acontis EC-Master EtherCAT master stack.

It is assumed a VisualGDB project is created below the EcMasterDemo application shipped with EC-Master.

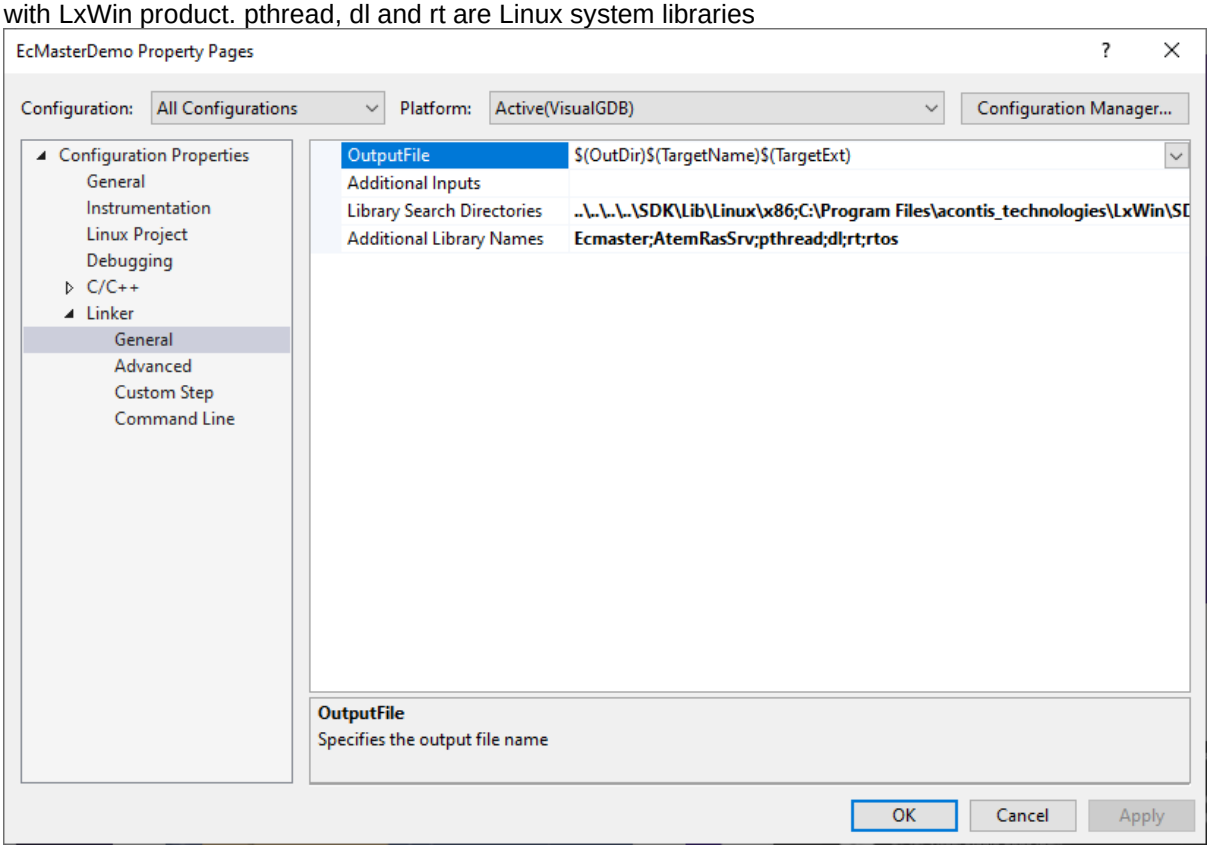
Set Additional Include Directories:

..\..\..\..\SDK\INC
..\..\..\..\SDK\INC\Linux
..\..\..\..\Sources\Common

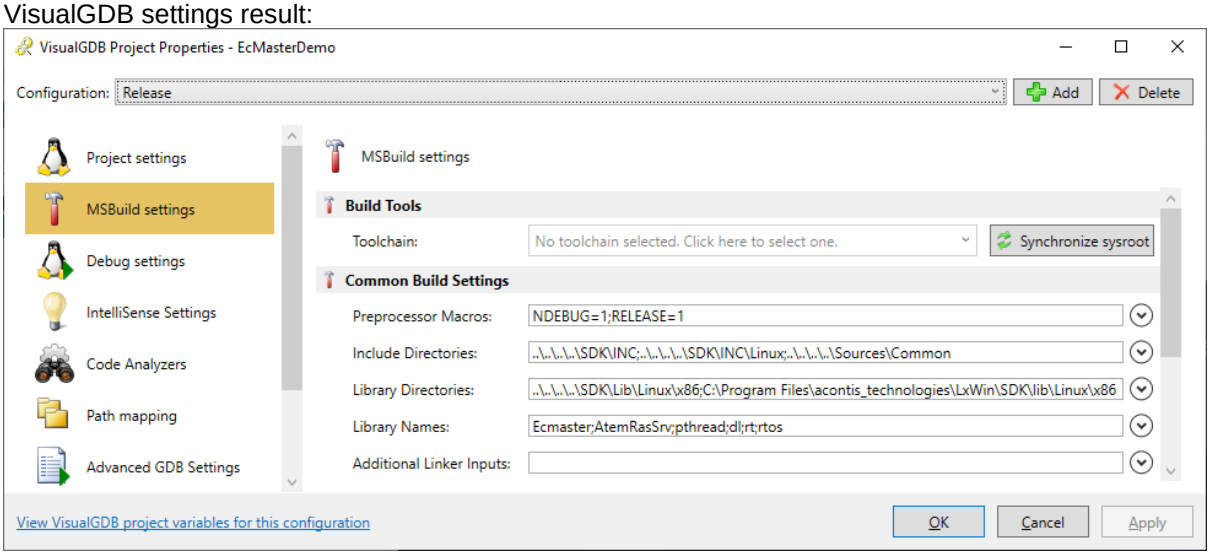


Library Search Directories:
..\..\..\..\SDK\Lib\Linux\x86
C:\Program Files\acontis_technologies\LxWin\SDK\lib\Linux\x86

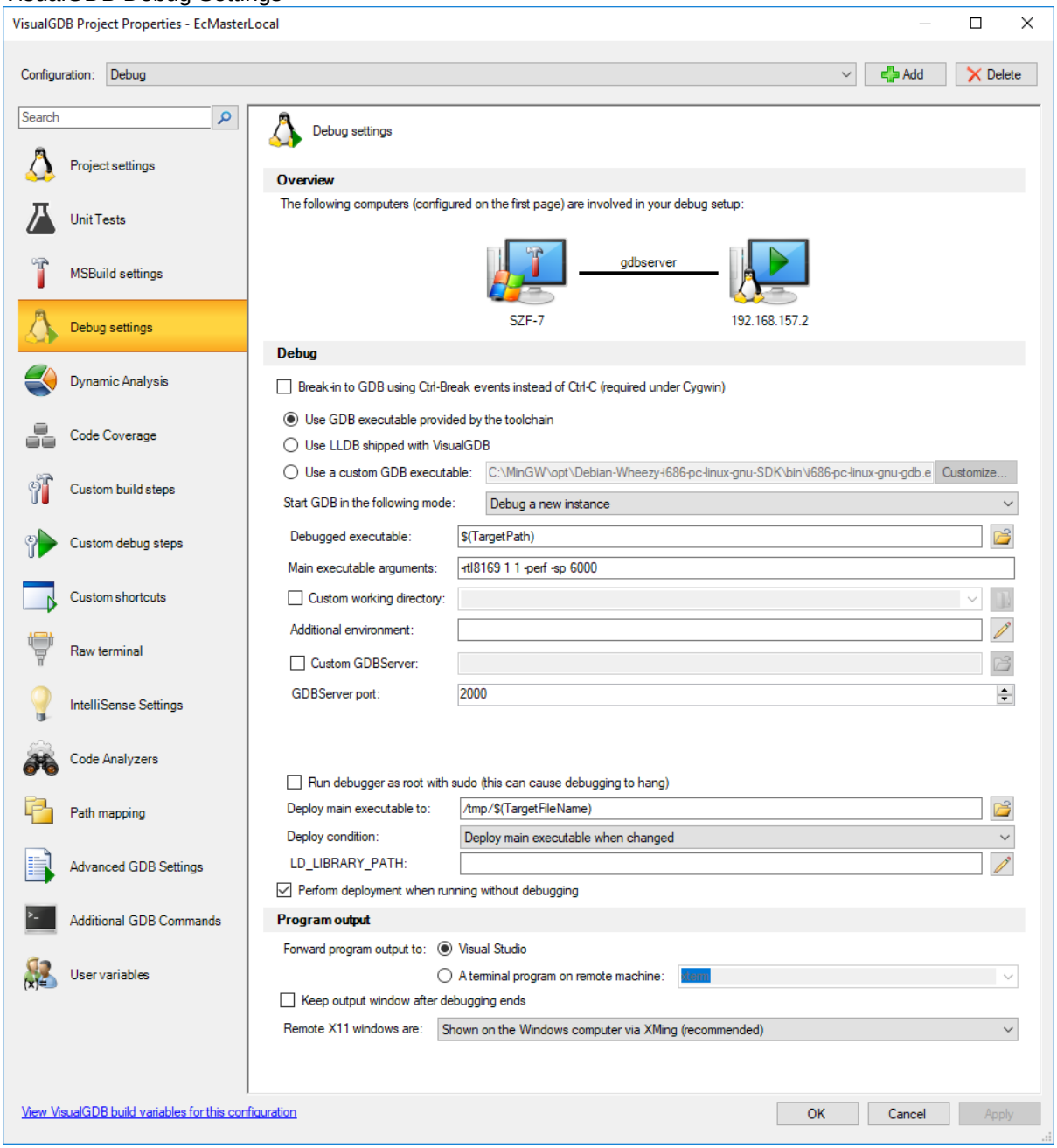
Additional Library Names: EcMaster;AtemRasSrv;pthread;dl;rt;rtos
Note: EcMaster and AtemRasSrv libraries delivered with EtherCAT master stack. Librtos.so delivered with LxWin product. pthread, dl and rt are Linux system libraries



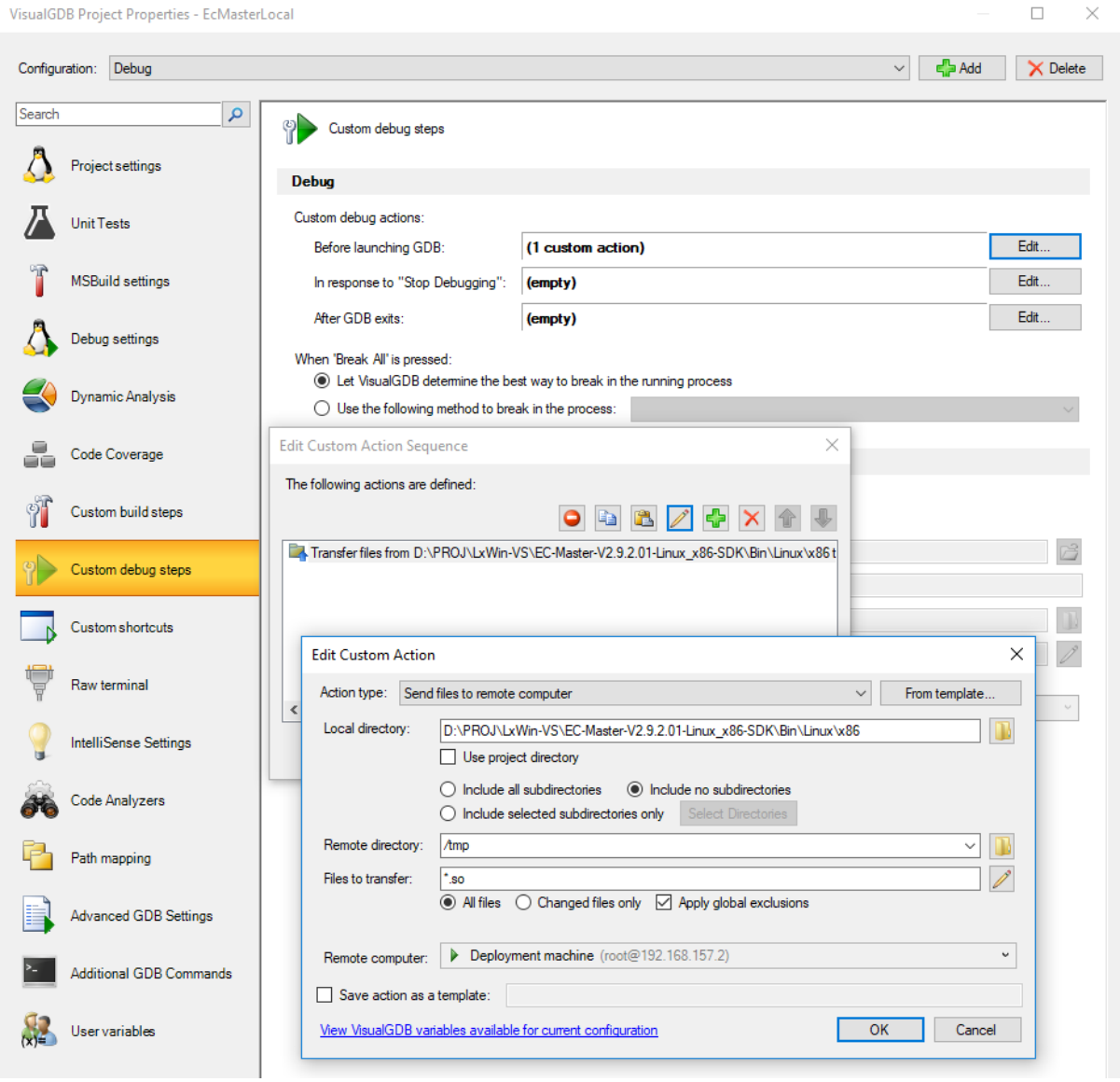
VisualGDB settings result:



VisualGDB Debug Settings



VisualGDB Custom debug step: copy shared object files to target:



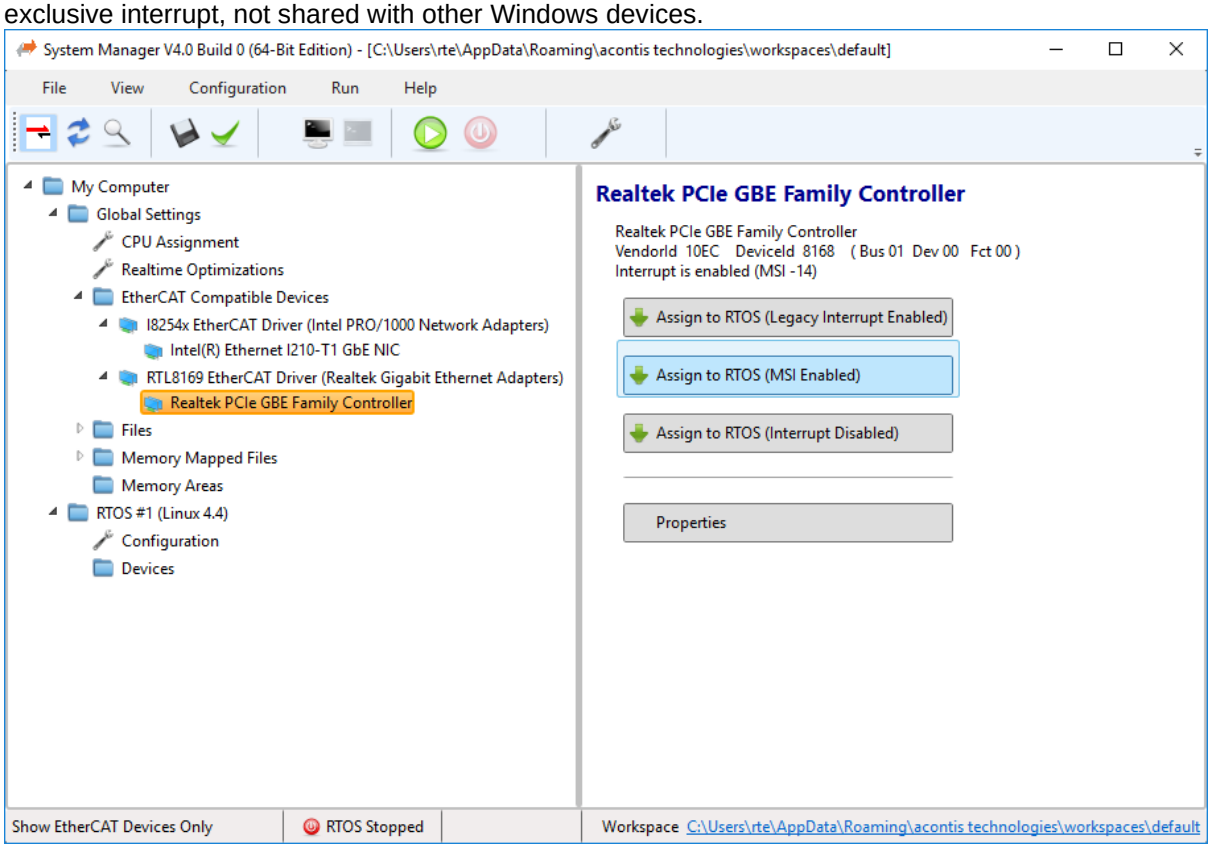
Detailed description of EcMasterDemo could be found in “EC-Master_ClassB.pdf” document.

When you debug the demo, you can stop debugging in Visual Studio or using Ctrl+C in console window.

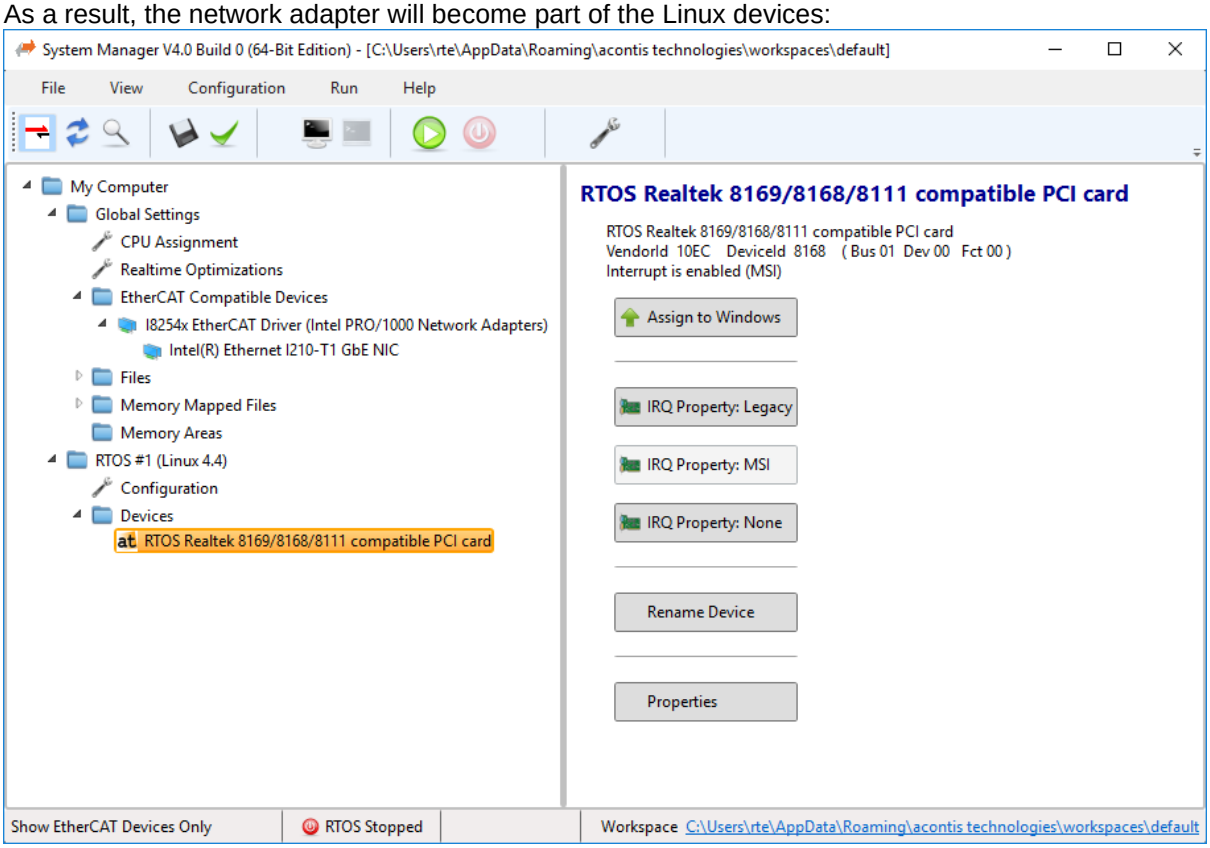
3.5 Add a physical Ethernet adapter to Linux

This tutorial shows how to add a physical Ethernet adapter to Linux. This adapter then can be used from within Linux to directly connect to an Ethernet network (e.g. to connect with GigE cameras, to connect with the cloud etc.).

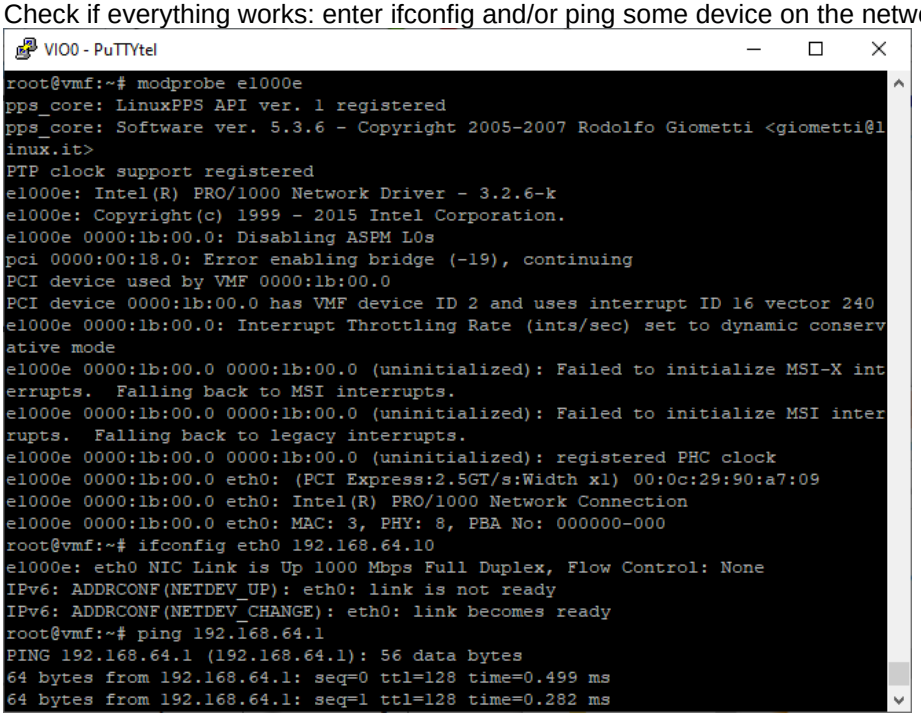
- Start the System Manager and press the top left button (toggle view mode). This will show all physical devices that may be accessed by Linux.
  Alternatively you can unselect View – Show EtherCAT Devices only
- Then select the Ethernet controller you want to use for Linux and for PCI Express devices press the “Assign to RTOS (MSI Enabled) button. In case of a legacy PCI devices you need to press the “Assign to RTOS (Legacy Interrupt Enabled)” button. Legacy PCI devices need to have an exclusive interrupt, not shared with other Windows devices.



- As a result, the network adapter will become part of the Linux devices:



- Start LxWin and login into the Linux shell
- Add the appropriate driver (these two examples are for Intel or Realtek Gigabit drivers):
  - modprobe e1000e
  - modprobe igb
  - modprobe r8169
- Add the adapter to the IP stack (assign an appropriate IP address) using the ifconfig command: ifconfig eth0 192.168.64.10
- Check if everything works: enter ifconfig and/or ping some device on the network



3.6 How to start a user application automatically

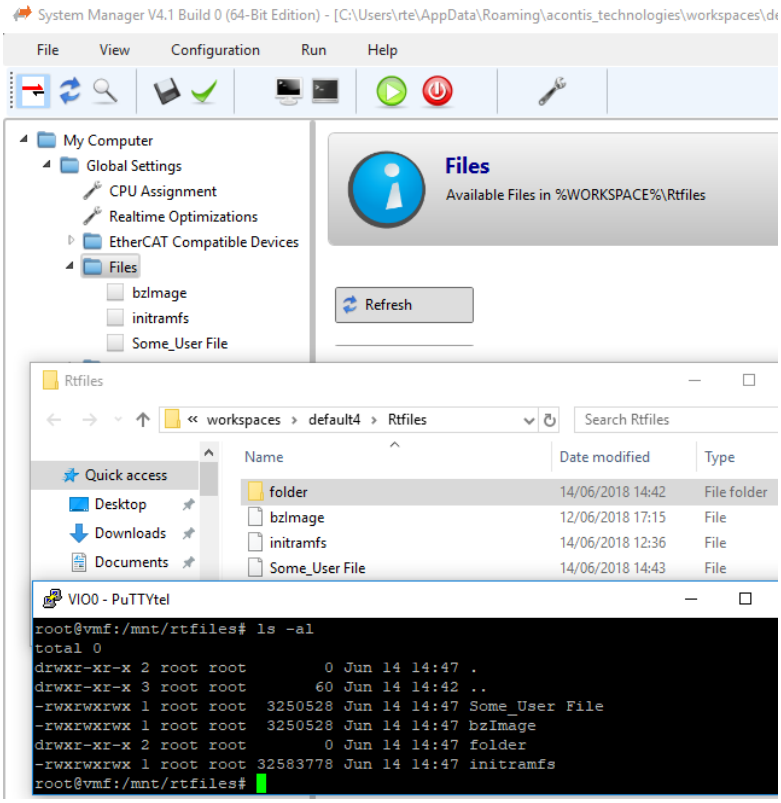
LxWin provides possibility to start an application or a shell script automatically. The usage of this parameter you can find in “5.5 Access Windows file system using Samba” chapter.

3.7 File exchange between Windows and Linux

LxWin is shipped with a predefined file system prebuild inside lxwinboot.bin. It is extracted into the RAM during Linux bootup. Any changes in file system are not stored when Linux is stopped and restarted.

3.7.1 Linux access to the hard disk

Linux has access to the files located at <workspace>\Rtfiles. It is automatically mounted to the path /mnt/rtfiles. So, it is possible to exchange files between Windows and Linux by using this directory.



3.7.2 SSH/SFTP access to the Linux filesystem

Since LxWin supports SSH/SFTP access it is possible to change the RAM based file system at runtime. This is useful to get results of Linux programs.

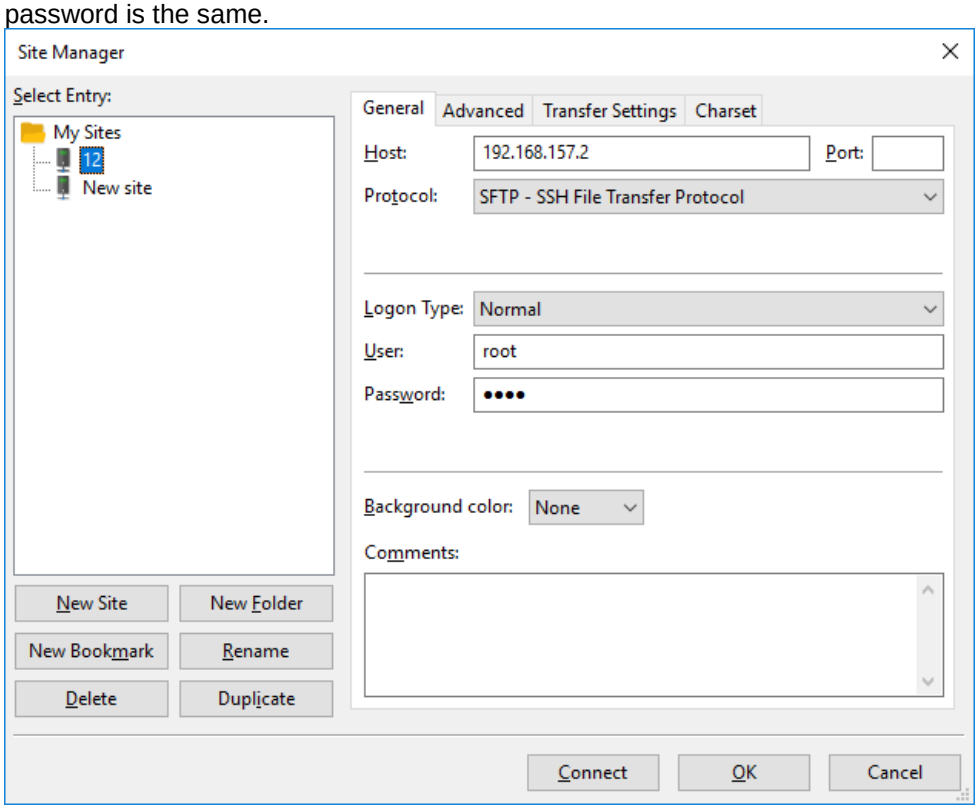
For doing so, you need an SSH client to establish a connection to LxWin.

The screenshot below shows how to use the FileZilla client, you may of course use other SSH/SFTP clients as well:

1. Start LxWin and the FileZilla client.

2. In FileZilla menu File -> Site Manager click “New Site” button.

3. Host is “192.168.157.2”. Protocol is SFTP. Logon type is Normal. User is “root” password is the same.



4. Click connect. Now you can easy access the LxWin file system via the FileZilla Client.

4 Example Applications

4.1 General

To make your initial experiences in working with LxWin and Linux go smoothly, a number of example application programs have been provided with the product release. Some are intended as exercises to help familiarizing you with various system features; others are useful tools. Much of the code can serve as model software that can save you time in developing your own applications.

The example applications are located at the product release CD-ROM at *CD:\SDK\Examples*. Per default, the LxWin setup will copy the sample applications to the directory: *C:\Program Files\acontis_technologies\LxWin\SDK\Examples*

4.2 Building a Windows application

The Microsoft Windows example applications are located in “C:\Program Files\acontis_technologies\LxWin\SDK\Examples\Windows”.

You can create and debug these applications with Microsoft Visual Studio 2008 or later.
A short description how to build the examples can be found in this directory (file HowToBuildTheExamplesWindows.txt).

4.3 Building a Real-time Linux application

The Linux example applications are located in “C:\Program Files\acontis_technologies\LxWin\SDK\Examples\Linux”.

You can create and debug these applications with Microsoft Visual Studio or eclipse. Please refer to 3.3 How to work with Eclipse or to 3.4 How to work with Visual Studio.

A short description how to build the examples can be found in this directory (file HowToBuildTheExamplesLinux.txt).

4.4 List of examples

These examples are based on the RTOS-Library. For more information about this library see the RTOS VM User Manual.

Windows	RTOS (Linux)	Description
CSharpDemo	-	The C# demo shows you how to use RtosLib functions from a C# application.
EventDemo	EventDemo	Use the Shared event demo to experiment with Shared Events between two OS.
-	FileDemo	The File demo shows you how a Linux application can remotely access the Windows file system.
InterlockDemo	InterlockDemo	The interlock demo shows you how to use the InterlockedCompareExchange function to synchronize shared memory access works between two OS.
MsgQueueDemo	MsgQueueDemo	Use the Message queue demo to see how a simple communication channel can be established between two OS using a shared memory based message queue to transfer data.
PipeDemo	PipeDemo	Use the Pipe demo to see how a simple communication channel can be established between two OS using a shared memory-based pipe to transfer data.
ShmDemo	ShmDemo	Use the Shared memory demo to see how a simple communication channel can be established between two OS using Shared Memory to transfer data.
SocketDemo	SocketDemo	Use the Socket demo to see how a simple communication channel can be established between two OS using shared memory-based socket to transfer data.
-	TimerDemo	Use the Timer demo to see how a periodic task with real time priority can be started
UploadDemo	-	Use the Upload demo to see how a simple application can start and stop a RTOS.

5 User's Guide

5.1 Multithreading, Synchronization, Timer

Linux/LxWin supports the POSIX API which provide a great amount of functions for multithreading and multiprocessing support. The Internet provides sufficient information as well as example source code about how to use POSIX under Linux.

Threading: POSIX threads (e.g. pthread_create)
Synchronization: Mutex variables (e.g. pthread_mutex_init)
Synchronization: Condition variables (e.g. pthread_cond_init)
Timer: POSIX timer (e.g. timer_create)

There are no POSIX events similar to Win32 events. Behavior of events can be achieved by using POSIX functions. You may find several solutions in the Internet like this http://www.it.uu.se/katalog/larme597/win32eoposix or this https://github.com/neosmart/pevents.

5.2 More information

5.2.1 General

- https://www.bogotobogo.com/Linux/linux_process_and_signals.php
- https://www.bogotobogo.com/Linux/linux_shell_programming_tutorial.php
- https://www.bogotobogo.com/Linux/linux_drivers_1.php
- https://igm.univ-mlv.fr/~yahya/progsys/linux.pdf

5.2.2 Tutorials

- https://www.tldp.org/HOWTO/RTLinux-HOWTO-4.html
- https://rt.wiki.kernel.org/index.php/HOWTO:_Build_an_RT-application
- https://wiki.linuxfoundation.org/realtime/documentation/howto/applications/application_base
- https://wiki.linuxfoundation.org/realtime/documentation/howto/applications/preemptrt_setup
- http://www.ee.nmt.edu/~rison/ee352_spr12/Getting_Started_with_RTLinux.pdf

5.3 Real-time application programming

LxWin is shipped with a pre-configured and validated hard real-time capable Linux image.

The basic technology behind it is the PREEMPT_RT patch for Linux.

Valuable information about how to create real-time capable Linux applications can be found at the Linux Foundation WIKI:
https://wiki.linuxfoundation.org/realtime/documentation/howto/applications/start

5.4 RTOS Library

The RTOS library provides higher level communication services for synchronizing Windows with Linux or to exchange data between the operating systems. The RTOS library is based on VMF-functions which provide the basic communication functionality.

A description of the RTOS Library can be found in the document RtosVM User Manual chapter 10.1.

5.5 Access Windows file system using Samba

It is possible to access the file system of a Windows PC using Samba. Here is a description of how to start samba automatically via a start-up shell-script.

1. Create a new Windows share in Windows or use existing one.

2. Modify the <workspace>\config\systemmgr_user.config file by adding these two lines:

```
[Rtos\Autostart\1]
    "Executable"="/mnt/rtfiles/cifs_mnt.sh"
```

3. Create a new file named “cifs_mnt.sh” at <workspace>\RtFiles with the following content:

```
HOST_IP="192.168.157.1"
SHARE_NAME="SomeName"
USER="UserName"
PASSWORD="UserPassword"

mkdir -p /mnt/${SHARE_NAME}
mount -t cifs //${HOST_IP}/${SHARE_NAME} /mnt/${SHARE_NAME} -v -o
username=${USER},password=${PASSWORD},domain=${HOST_IP},vers=2.1
echo "cifs mounted"
```

4. Modify the SHARE_NAME, USER and PASSWORD parameters appropriate to your windows share from step 1.

5. Change line endings of cifs_mnt.sh file. Windows programs create CR/LF line ending, by default. Shell pre-processor cannot recognize this so you have to convert the line endings to only LF. Common Editors have functions to do that.

6. Start LxWin and verify the mount results with the “mount” command.

Note: Windows XP does not support Samba 2.1. After removal of “`,vers=2.1`” from the cifs_mnt.sh file it works fine.

5.6 Synchronize Linux Time with Windows Host

LxWin contains a special time synchronisation daemon, that works in background and synchronizes Linux time with Windows host time. This daemon is by default disabled, but it is started automatically, when time synchronisation feature in enabled in configuration.

To enable this feature please change the following config sections:

```
[Windows\TimeSync]
  "TimeSyncMaster"="Windows"
  "TaskEnabled"=dword:1
[Rtos\TimeSync]
  "TimeSyncMaster"="Windows"
  "TaskEnabled"=dword:1
  "TimezoneLinux"="Europe/Berlin"
```

Now start LxWin and take a look at a new log message, that timesync daemon has been started.

If you want to have different timezone in Linux and Windows, you should set TZ variable in Autostart script.

All existing time zones could be found in /etc/share/timezone directory. Default LxWin image does not contain all available timezones, because of size. If need to use other timezones, please change yocto\recipes-core\images\lxwin-image-initramfs.bb file.

5.7 How to create your own LxWin Image (BSP)

LxWin is shipped with 2 source packages located at \SDK directory:

- \SDK\sources_x86.tar.xz – sources for 32-bit Linux version
- \SDK\sources_x64.tar.xz – sources for 64-bit Linux version

Each packet contains sources and closed source binaries required to build the image.

5.7.1 Build image

Ubuntu 18.04 or Ubuntu 20.04 is required to build the LxWin image.

You should copy sources_x86.tar.xz or sources_x64.tar.xz file to your Linux machine and decompress it with the following command:

```
$ tar -xf sources_x64.tar.xz
```

After decompression you will have 2 script files:

`prepare.sh` – prepare Linux machine by installing all required packages. Sudo rights will be required.

`build.sh` – builds the image. The resulting binary file will be stored in delivery\Bin\lxwinboot.bin.

5.7.2 How to change the default image

LxWin uses yocto to build images. Yocto is a flexible and well documented system. If the shipped default LxWin image does not satisfy your requirements, you can change it.

Here is the description of some files which are first candidates for changes:

Source/yocto/recipes-core/images/lxwin-image-initramfs.bb
BitBake file which describes all packages included to the default image.

Source/yocto/recipes-core/images/lxwin-image-initramfs-small.bb
BitBake file for a minimal image. Can be used for systems with low memory footprint or as an example of which packages could be removed.

Both of these Bitbake files change the “PACKAGE_INSTALL” variable. This variable shows which packages will be installed on the system.

Source/yocto/recipes-kernel/linux/linux-yocto-rt/defconfig and
Source/yocto/recipes-kernel/linux/linux-yocto-rt/x86-64/defconfig
are Kernel configuration files for 32- and 64-bit versions.

Here are several use cases:

1. Reduce the size of the image.
   - You can remove packages from the PACKAGE_INSTALL variable. For example, GDB server is not required in the production image.
   - You can install only required kernel modules instead of all modules. Package “kernel-modules” installs all modules. Inside the lxwin-image-initramfs-small.bb file you can see how to install specific modules.

2. Add specific closed source package.

    - Package Source/yocto/recipes-core/initrdscripts contains a list of files added to the image. You can see several binary files are located inside initrdscripts/files/usr/bin directory.

3. Change the Kernel configuration.

    - defconfig is the Kernel configuration file.

4. Change default user credentials:

    - By default, the user “root” with the password “root” is used. It is defined in the Source/yocto/conf/layer.conf file.

5.8 Linux kernel real time configuration

LxWin is shipped with the kernel, optimized for real time usage. Customers can configure the kernel if necessary, for their specific needs.

The Linux configuration is stored in the file .config. It can be changed with the “make menuconfig” command.

Here you can find the most important settings which are required to assure real-time behaviour.

CONFIG_VMF – Enables VMF paravirtualization code. Without this setting LxWin will be not able to start. Must be set.

CONFIG_PREEMPT_RT_FULL – Enables real time. Should be set.

CONFIG_HIGH_RES_TIMERS – Enables high resolution timers. Without these timers, applications will be not able to sleep less than a single OS tick. Its tick is controlled with the CONFIG_HZ setting.

Power management is controlled by Windows and must be disabled in Linux: CONFIG_PM, CONFIG_POWER_SUPPLY, CONFIG_ACPI, CONFIG_SUSPEND, CONFIG_LOCKUP_DETECTOR.

CPU frequency changes, idle states and overheating control will have bad effects on the real-time behaviour. Thus, the following settings should be not set: CONFIG_CPU_FREQ, CONFIG_CPU_IDLE, CONFIG_X86_MCE.

The following devices are not supported by LxWin and should be disabled:

- USB devices: CONFIG_USB_SUPPORT
- Parallel and serial ports: CONFIG_PARPORT, CONFIG_SERIAL_8250
- Disk devices: CONFIG_CDROM_PKTCDVD, CONFIG_SCSI,
- Graphics devices: CONFIG_AGP, CONFIG_FB
- HPET should be controlled by VMF: CONFIG_HPET

5.9 Debugging - Using Eclipse or Visual Studio

5.9.1 Two development arrangements

To develop software for LxWin, the user may choose either of two basic host/target configurations:

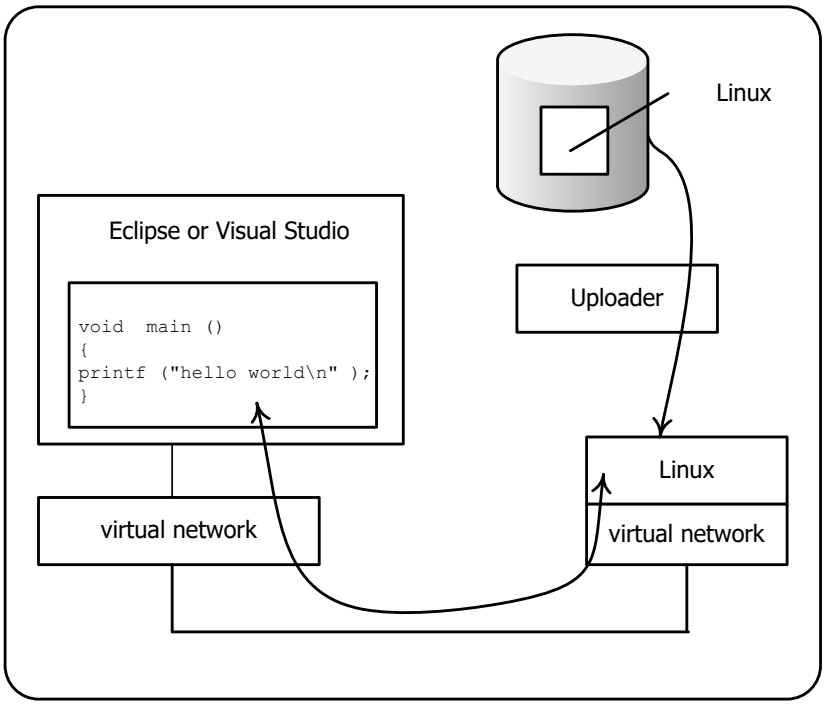
The one-system method: Develop a Linux application under Windows and then run it under LxWin on the same computer...

-or-

The two-system method: Develop a Linux application on one computer (Windows or Linux) and run it on another computer, on which LxWin is installed. If you choose the two-system method, you can use a single- or a double-network technique.

5.9.2 The one-system method

In a combined host/target system the connection between Windows and the target is made using the LxWin Virtual Network.



In this configuration, the development software runs under Windows and the run-time target's components run under Linux, which was installed on the same PC. The development process is described in Tutorials section 3.3.2 for Eclipse and section 3.4.1 for Visual Studio.

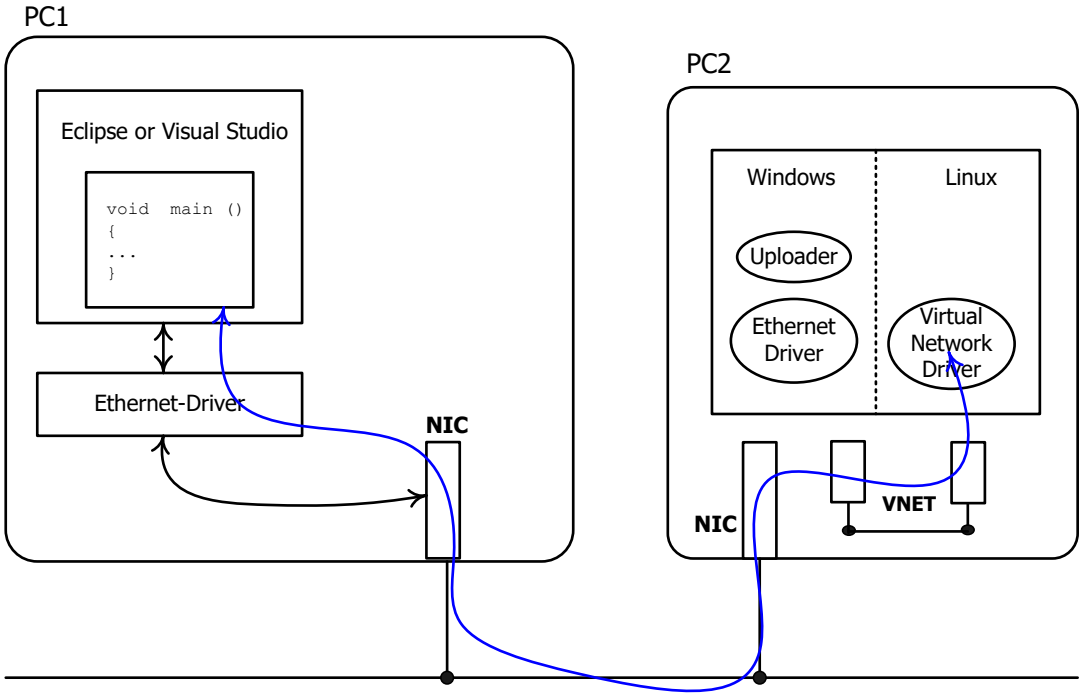
5.9.3 The two-system method

When host and target systems are physically separated, the connection between them can be made using either of two techniques.

Two systems – using the virtual network and bridging

For connecting the development host system with the LxWin target, network bridging in the LxWin target computer can be used.

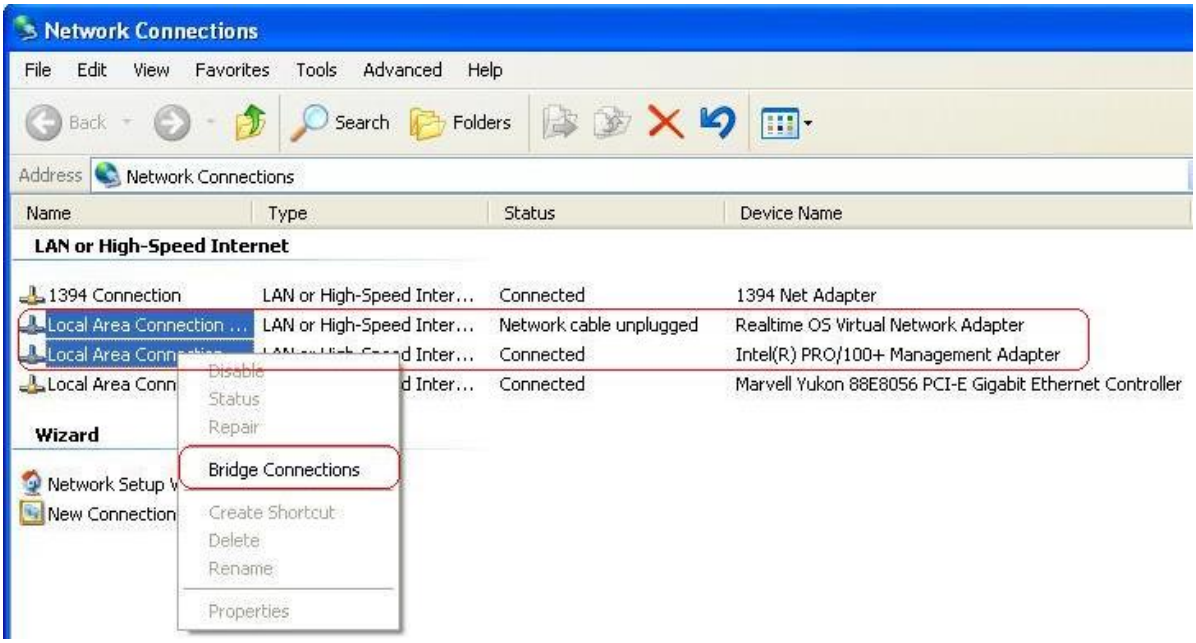
The following figure shows how PC1 (the development host system) is connected with PC2 (the LxWin target system). Both PCs are connected via Windows using TCP/IP networking. Only Linux has direct access to the virtual network.



Without network bridging only Windows on PC2 have access to Linux.

With network bridging, the Linux operating system becomes also visible for Windows on PC1 and all other PCs in the same subnet of PC1 and PC2 (blue line).

On PC2 open the network property sheet and select both, the Windows network adapter and the virtual network adapter and choose “Bridge Connections”.



If multiple remote target systems are used with active network bridging the MAC address of the virtual network adapter has to be adjusted to avoid serious networking problems.

By default the MAC address of the virtual network adapter is set to AA:BB:CC:DD:F0:02.

Normally this address is hidden outside the virtual network. But when bridging is active, this MAC address will be seen in the bridged network as well. If more than one bridged system is active this would lead to duplicate MAC addresses in the network and finally lead to unpredictable behavior (unreachable systems).

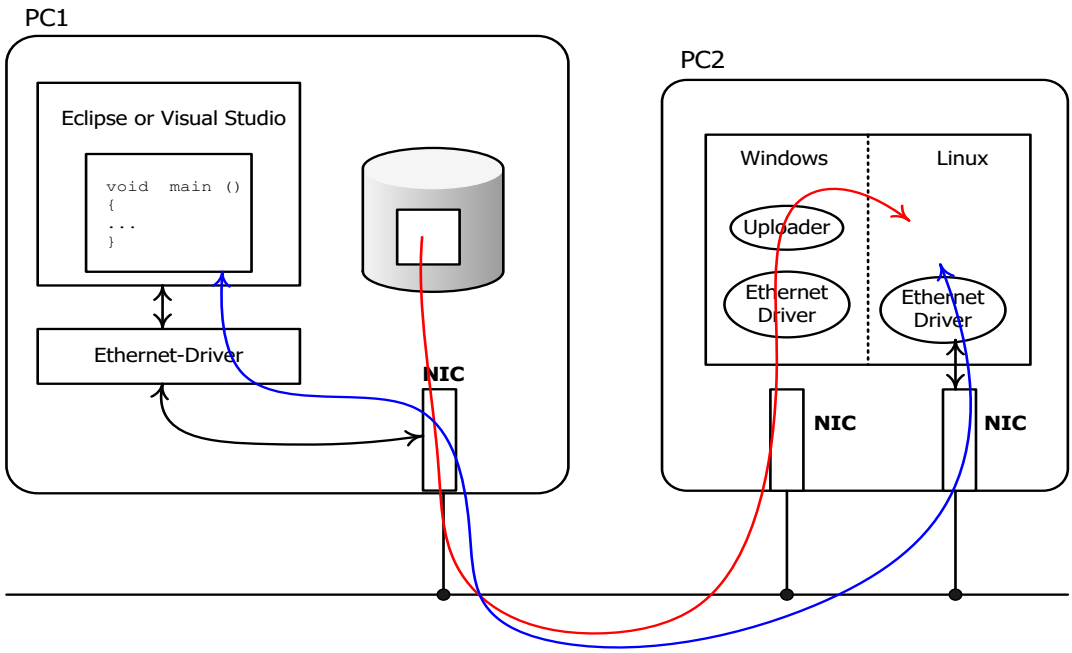
To avoid such situations the MAC addresses of all bridged LxWin target systems have to be set to unique values. The MAC address is determined in the systemmgr.config file. Please follow chanpter 2.4 for details.

A simple solution would be to change the 5th byte from F0 to a unique value beginning with 00 up to FF, for example:
First LxWin target system: `"MacAddress"="AA:BB:CC:DD:00:02"`
Second LxWin target system: `"MacAddress"="AA:BB:CC:DD:01:02"`
Third LxWin target system: `"MacAddress"="AA:BB:CC:DD:02:02"`

Default LxWin IP address is 192.168.157.2. If PC1 has IP address not in 192.168.157.xxx range, it will be not possible to connect to PC1 and LxWin. You should change LxWin IP address using "IpAddress" setting under [***Rtos\Vnet\0***] key in your config file.

Two systems – three network adapters

Using this technique (as illustrated below), the target system has two network adapters, one for Windows alone and one for Linux:



Development components are installed on the host computer – PC1, while run-time components are installed on the target computer – PC2. The development process is as follows:

- Step 1: Build a new Linux application on PC1.
- Step 2: On PC2, assign NIC to LxWin in interrupt mode and start LxWin.
- Step 3: When Linux is started, you should enable NIC in Linux. Please refer chapter 3.5 for details.
- Step 4: Configure Eclipse or Visual studio on PC1 to debug the application on PC2.

5.10 RtosService utility

RtosService.exe is needed when using the RtosLib.lib.

If you take a look at the default General.config file, you will see that RtosService.exe is automatically launched.

RtosService.exe starts the communication between Windows and Linux. It also starts the time synchronisation between Windows and Linux.

5.11 Accessing PCI cards

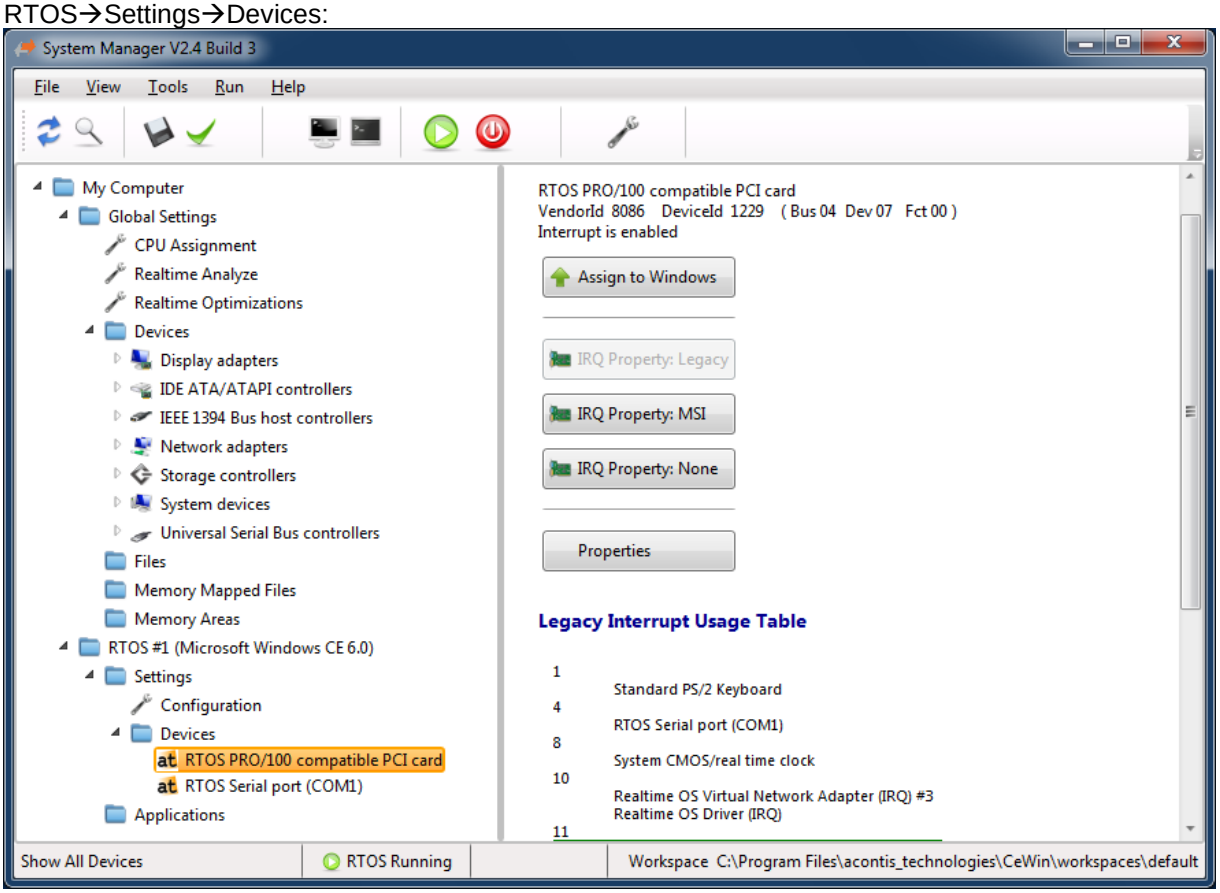
When PCI cards shall be controlled by Linux the PC will have to be partitioned first.

By default, all hardware belongs to Windows.

To being able to access hardware from within Linux it will have to be separated from Windows. This is described in detail in the RTOS VM User Manual.

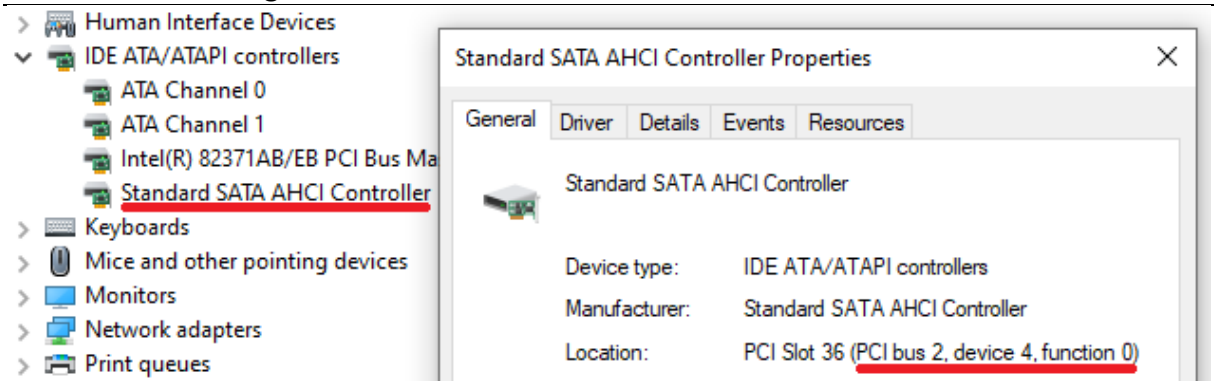
After separation is done, the usual methods within Linux can be used to access PCI cards. See the Linux manuals for more information.

Within the System Manager all devices assigned to a RTOS will appear below RTOS→Settings→Devices:

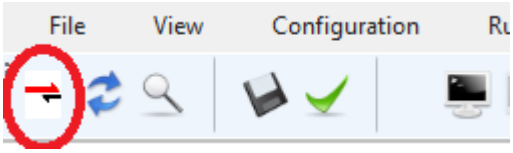


5.12 How to use SATA devices in LxWin

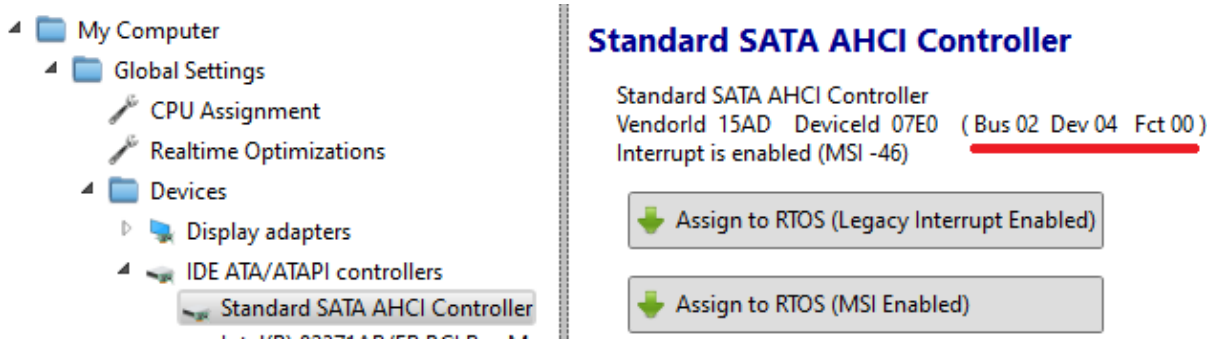
You should know which SATA device you should assign to LxWin. In device manager find your SATA controller and identify its location. You will need bus, device, function numbers.



In System Manager uncheck “Show EtherCAT devices only” to enable all PCI devices:



In the device tree from System Manager select required SATA device and verify that it has the same bus, device, function.



Assign device to RTOS and start LxWin.

In console Window load ahci kernel module with command: `modprobe ahci`. In Log you will see several SDA devices like:

```
sd 1:0:0:0: Attached scsi generic sg0 type 0
 sda: sda1 sda2 sda3 sda4 sda5
sd 1:0:0:0: [sda] Attached SCSI disk
```

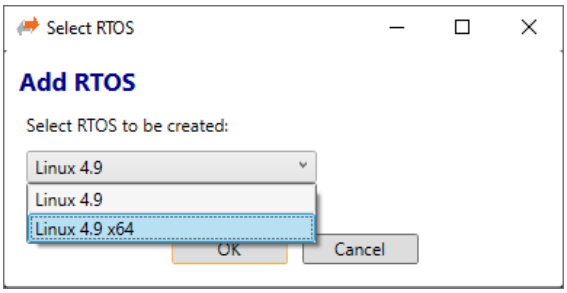
You can mount your device into some location. With commands like:

- for Ext4 file system: `mount -t ext4 /dev/sda5 /mnt/5`
- for NTFS file system: `mount -t ntfs /dev/sda4 /mnt/4`

LxWin do not provide any tools for the disk formatting. Please do it on native or VM Linux before assigning to LxWin.

5.13 64-bit Kernel

When creating a new System Manager workspace, you can select between a 32 and 64-bit Kernel:



The current x64 Kernel version has several limitations:

- It is not possible to allocate more than 2,9 GB memory.
- SMP is not supported.
- Shared Mode Operation is not supported.

5.14 LxWin GPL compliance

LxWin uses open-source packets with GPL licenses. LxWin is compliant to GPL V2, because the Linux kernel and the BusyBox (user interface shell including several commands) are based on GPL V2.

Compliance to the FLOSS (Free/Libre Open Source Software) licenses are achieved in LxWin by the following means:

1. The related source code is provided.
2. License text for the used software is provided.
3. Compilation scripts and modifications to the source code are provided.

LxWin delivers source code and compilation scripts inside the following files:

- \SDK\sources_x86.tar.xz – sources for 32-bit Linux version
- \SDK\sources_x64.tar.xz – sources for 64-bit Linux version

All licenses texts are located at:

- \SDK\licenses_x64.tar.xz – licenses for 32-bit Linux version
- \SDK\licenses_x64.tar.xz – licenses for 64-bit Linux version

Customer software which is added into the Linux kernel of LxWin also will have to be GPL compliant according to the above rules.

Important: Customer software and applications running in user mode has not to be compliant to GPL and thus can stay closed source and can be provided under any kind of software license the customer chooses.

5.15 LxWin real-time guard band

A typical real-time application has to need to run a deterministic constant real-time cycle (e.g. for motion control or PLC operation). To generate such periodic behavior, in Linux the function clock_nanosleep is used. This function uses high-resolution timers inside the Kernel.

LxWin provides optimized version of the function which has several benefits:

- Performance. It is faster, because no high-resolution timers are processed.
- Reduced number of interruptions. The functions guarantees that no timer interrupts will be generated within a guard band around the generated timer event and thus can help to overcome real-time issues.

Optimized nanosleep is implemented as ioctl for RTOS driver.

```
struct RTOS_NANOSLEEP_PARAMS
{
  struct timespec request;
  __u64 pre_guard;
  __u64 post_guard;
  __u64 out_tsc;
};
```

Parameter

request

[in] requested absolute time requested time until than the calling thread should sleep for. Refer to Linux documentation on clock_nanosleep function for details about this parameter.

pre_guard

[in] amount of time in ns when no other wake-ups could happens before requested time

post_guard

[in] amount of time in ns when no other wake-ups could happens after requested time

out_tsc

[out] could be ignored and used for performance measurements only.

Return

0 on successfully sleeping for the requested interval.

EINTR - The sleep was interrupted by a signal handler

EINVAL - The value in the tv_nsec field was not in the range 0 to 999999999 or tv_sec was negative. or pre_guard or post_guard is more than 1 ms.

Limitations:

- It is possibly to freeze whole system if you set too big pre_guard and post_guard values and you call the function frequently. You should give the OS some time for the OS specific tasks.

- The function is CPU specific. You should set appropriate affinity for your thread.

- You cannot call the function several times parallel on the same CPU. Only one caller per CPU is possible.

5.16 Security

The default LxWin image covers the needs of most customers.

In a typical use case, the Linux part of a LxWin based application does not have any connection to the Internet. In case the Linux part needs access to the outside network (and possibly is connected to the Internet also), for example by assigning a PCI network card to Linux, security measures may be required.

Here you will find some recommendations to increase security of the Linux part in such cases:

- The **Telnet server** should be disabled by setting the “notelnet” parameter to “yes”. You can build you own kernel image without the Telnet server.

- **OpenBSD Secure Shell server** may be disabled by setting the "nossh" parameter to "yes". You can build you own kernel image without the SSH server.

- The default root password should be changed. It is defined at Source/yocto/conf/layer.conf file.

- Default certificates for SSH should be changed by changing the files at Source/yocto/recipes-core/initrdscripts/files/etc/ssh folder.

- For 64-bit Kernel the “nopti” parameter should be removed. You should set “cmd_line” parameter to “auto”

6 Version History

A general version history containing information about new features, migration hints and improvements can be found in the release notes file “ReleaseHistory.txt”.