acontis technologies GmbH

SOFTWARE

Hypervisor-RTOS-32-Remote-Debug-Guide

acontis Real-time Hypervisor and RTOS-32 Remote Debug

Version 1.0

Edition: November 3, 2022

Table of Contents

1 Step by Step Host and Target Configuration with System Manager and Shell Scripts

1.1 Introduction

It is recommended to carry out the steps in *the order described* so that you get the remote debugging running. Steps, which are marked with `<optional>`, can be skipped.

Important: The steps are grouped in parts **A-E** over host and target. Each part **depends** on the previous part!

On the development PC our classic EC-Win(RTOS-32) is used. On the hypervisor the steps must be done by the shell and/or supplied shell scripts.

1.2 Default values:

This tutorial assumes the following default values. Adjust it accordinly to your needs on the places of occurence.

- `$Target_HV_IP$ = 172.17.10.53` - IP address of hypervisor network adapter **and** of the bridge.
- `$Target_IP$ = 172.17.10.5` - Remote IP address of the RTOS part.
- `\\172.17.10.53\debug` - Default network share (adjust share name `'debug'` at `/etc/samba/smb.conf` if needed)
- `enp2s0` default name of the hypervisor network adapter.

1.3 Prerequisites

- EC-Win(RTOS-32) 7.1 or newer
- acontis Hypervisor package
- Development PC
- Target PC/IPC
- Visual Studio (2005-2019) with C/C++ workload

On Development PC: Visual Studio + EC-Win(RTOS-32)

On Target PC/IPC: installed acontis Hypervisor package

2 Part A - Development Host Configuration

- Start the System Manager

Hint: If it's the first launch of the System Manager a dialog to enter a workspace directory opens.

- Select **My Computer** node on the tree view and add a RTOS to the configuration.

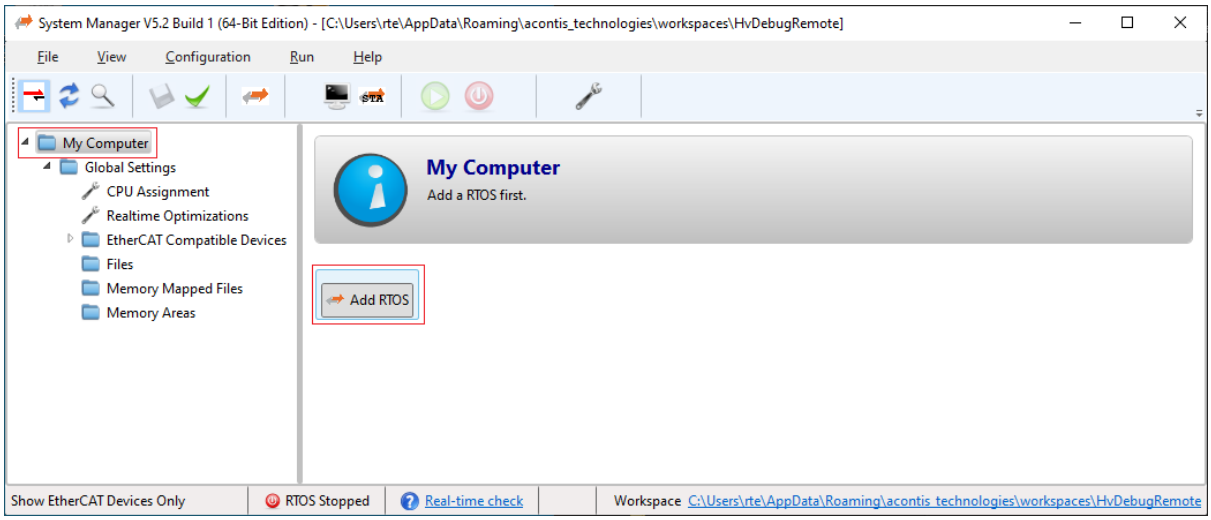

Fig. 2.1: Select Rtos in SysMgr.

- Select the `Application` node of the `RTOS #1` section and push the `Create New Application Project (Debug Only)` button.

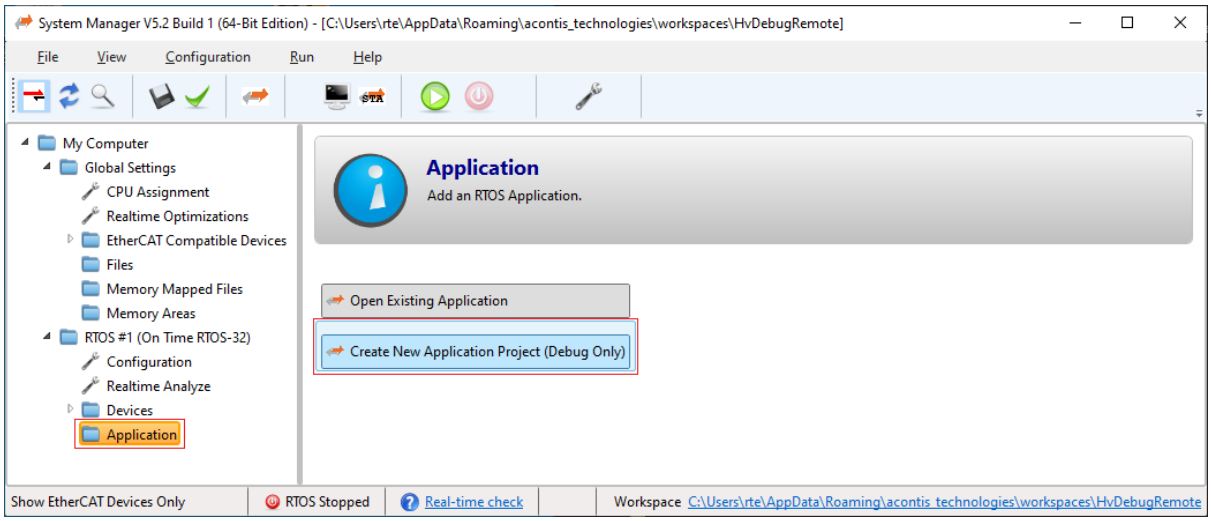

Fig. 2.2: Select source application in SysMgr.

- Select for ex. RealtimeDemo and push `OK` button.

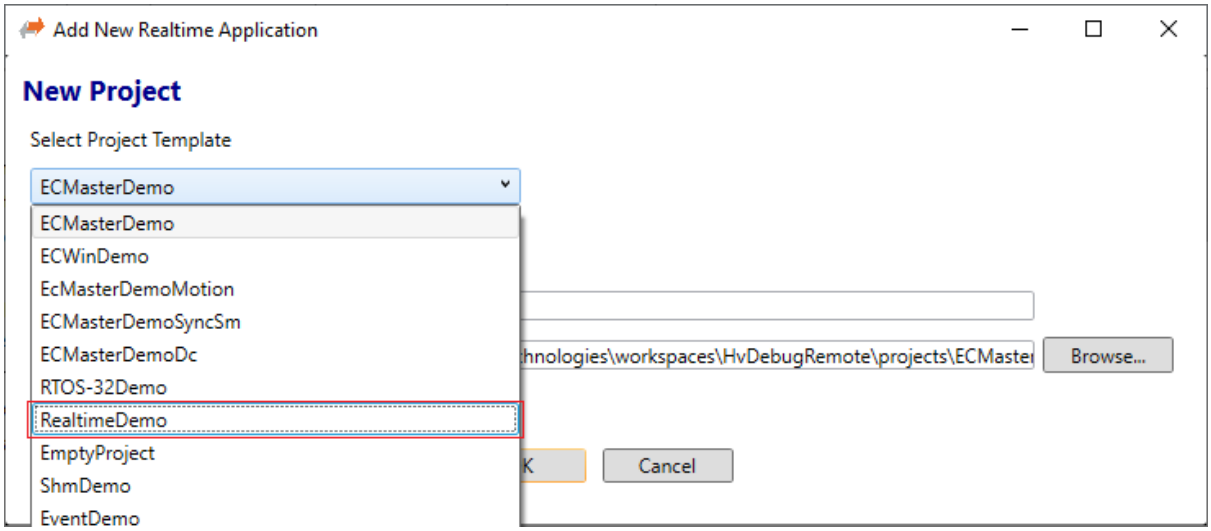

Fig. 2.3: Select for ex. RealtimeDemo as source application in SysMgr.

- Scroll to `<Development|Debugging>` section and push the `Settings` button.

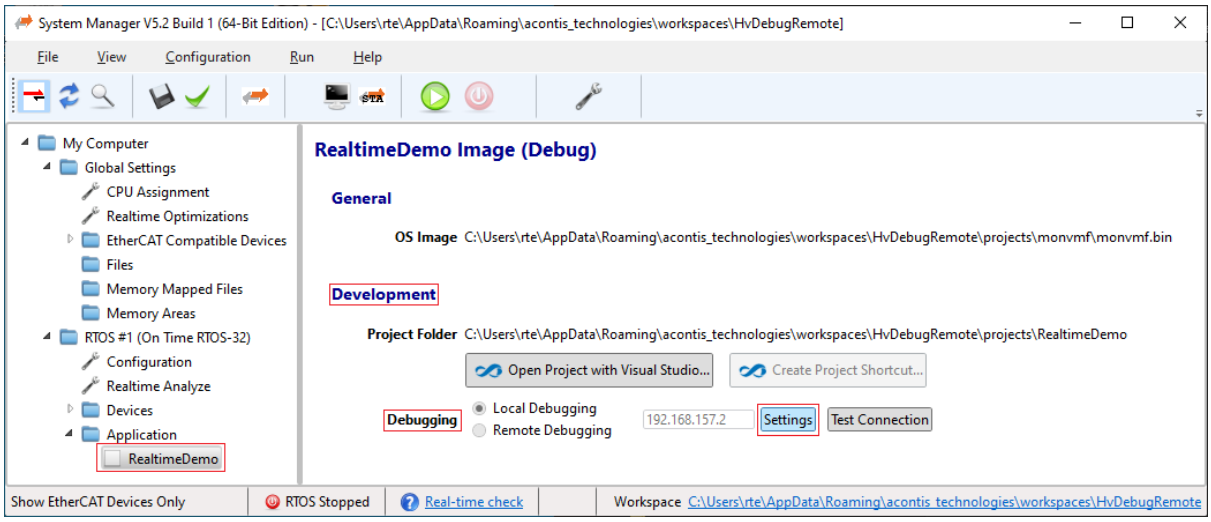

Fig. 2.4: Select debugging settings.

- `Edit DEBUG Connection Settings` dialog will pop up.

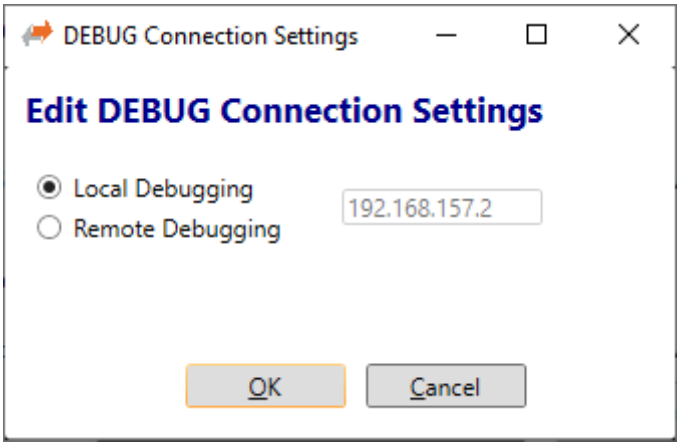

Fig. 2.5: Debugging settings dialog.

- Select `Remote Debugging` and enter the IP address of the target. Please note, this IP address must **not be** in use by other computers in your network and it has to fit into your company's IP environment. You may have to ask your System Administrator to get a free IP address for that purpose. Both, host and target computer *must be* in the **same** network. In following steps this IP address will be called `$Target-IP$`

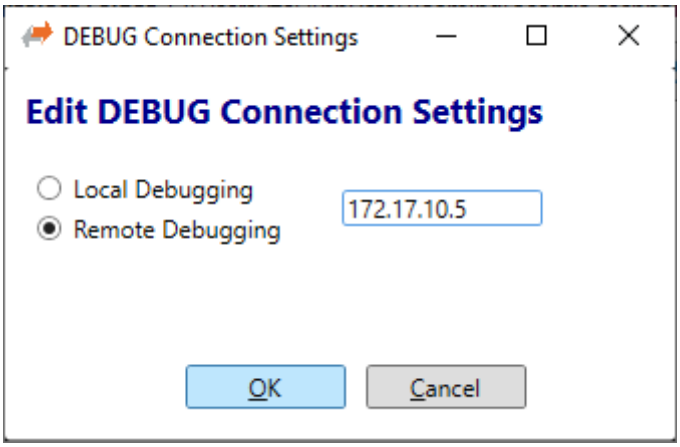

Fig. 2.6: Debugging settings dialog. Remote IP entered.

- Push the `OK` button.
- Check if remote debugging and changed `$Target-IP$` has been saved.

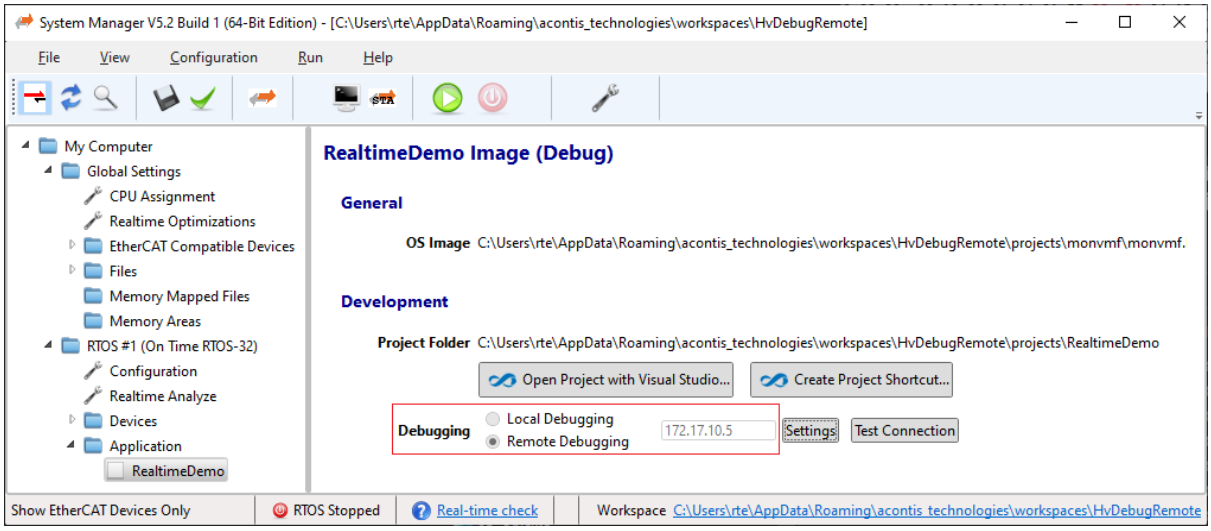

Fig. 2.7: Debugging settings changed to remote IP.

- Push `Open Project with Visual Studio` button.

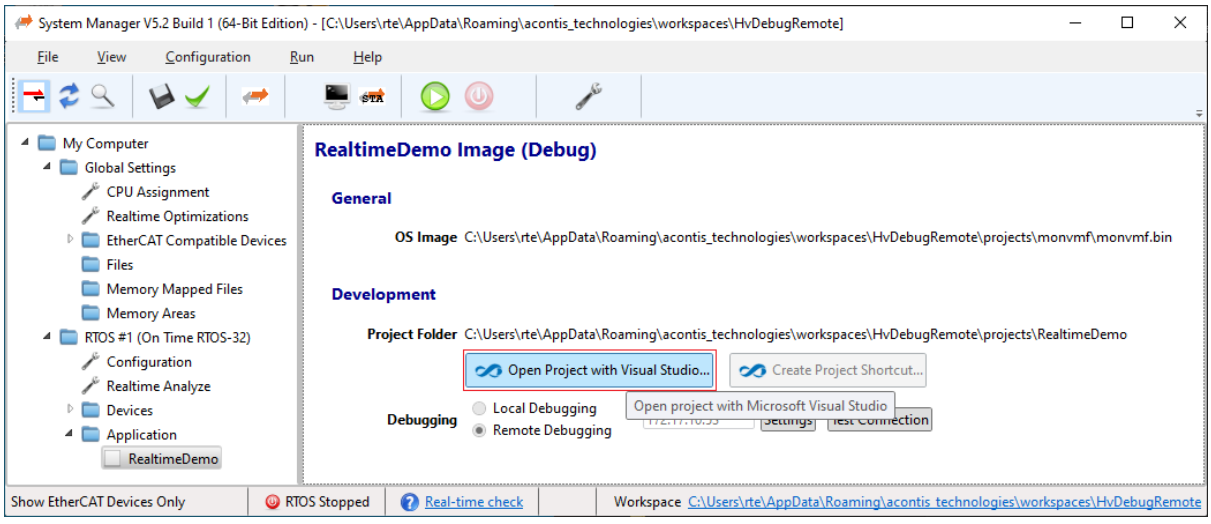

Fig. 2.8: Push button to open Visual Studio and automatically create the source project.
Project will be created and all required settings are set.

- Adjust `$Target-IP$` in generated *Visual Studio Loader* project.

  Select the file `Vmftarget.cfg` in the *Visual Studio Loader* project and edit the setting `COMPort Ethernet $Target-IP$`.

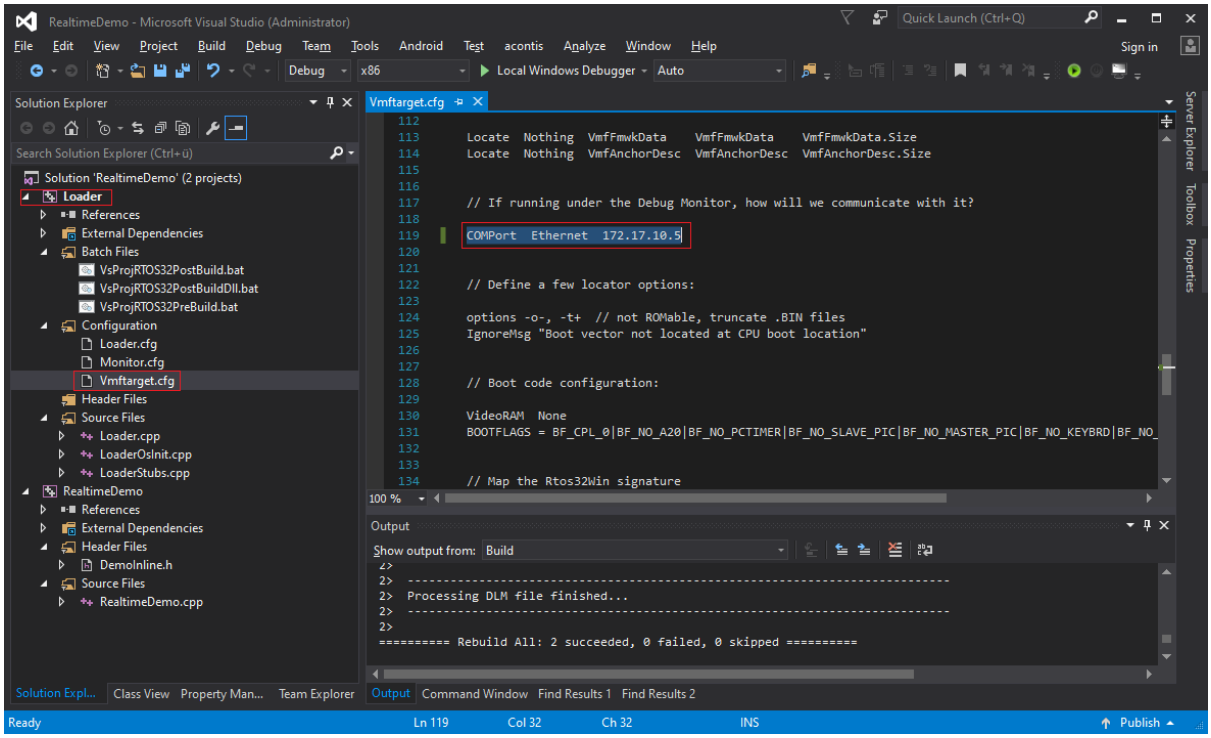

Fig. 2.9: Edit *Visual Studio Loader* project file `Vmftarget.cfg`.

- Press `F7` to compile the project, if first time compiled. Otherwise a **rebuild all** (`ALT + F7`) is required!

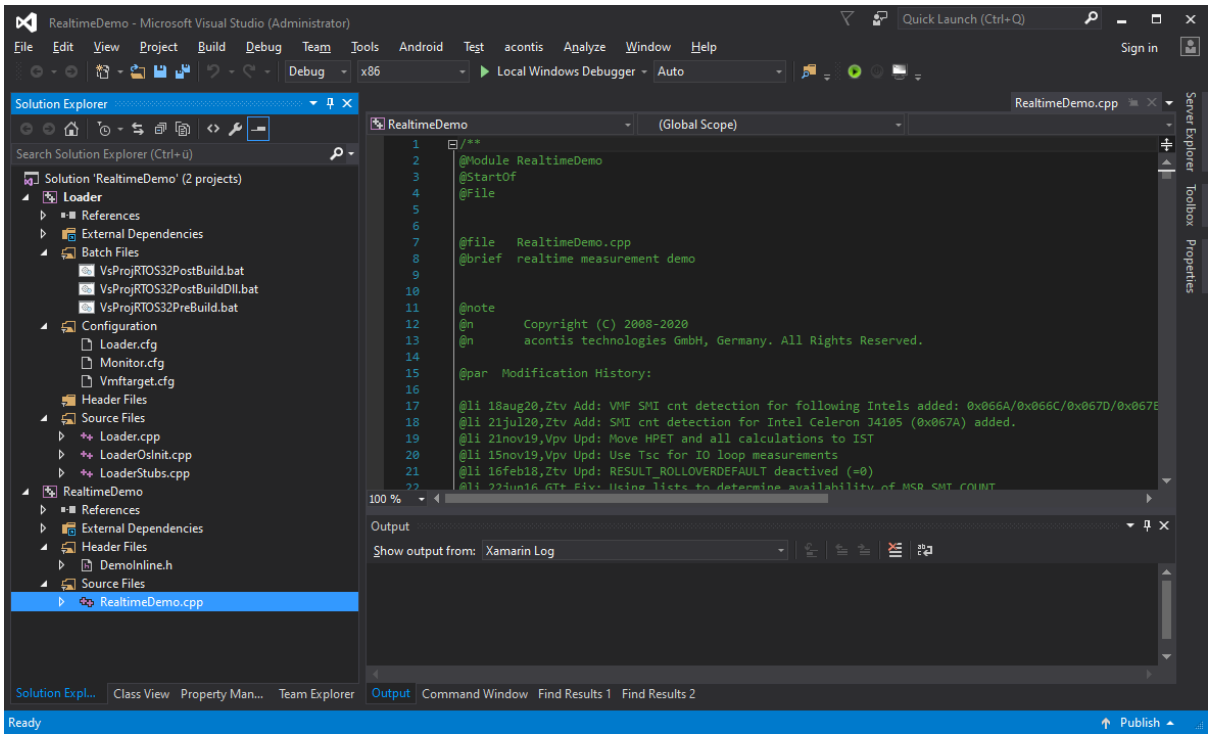

Fig. 2.10: Visual Studio contains the generated source project.

3 Part B - Hypervisor Target Configuration

The following shell scripts are available on the hypervisor to setup and establish a remote debug connection:

- `/hv/rtos-32/realtimedemo-debug.sh` starts debug monitor that awaits requests from a remote debugger on a Windows machine with installed *EC-Win(RTOS-32)*, *Visual Studio*, and *RTE Visual Studio Plugin*.
- `/hv/hvctl/brvnetset.sh` *creates* a virtual network bridge on Linux host to forward debugger `TCP/IP/UDP` packets from `LAN1` to `RTOS-32 VM`. It is required to start this script if you need to perform *remote* debugging of a `RTOS-32` app from another machine.
- `/hv/hvctl/brvnetclr.sh` *deletes* bridge, after the `RTOS-32 VM` has been stopped.
- `/hv/hvctl/brvnetconfig.sh` contains the configuration values for the bridge. This file must be edit to adjust at least the `$IP$` address.

Hint: The files are supplied in *default* state. At least `/hv/hvctl/brvnetconfig.sh` needs adjustments.

Hint: See chapter “Bridge virtual and physical network” in the Hypervisor manual for details how to configure the bridge.

---

4 Part C - \RtFiles\debug\. Subdirectory

The debug monitor and the compiled debug files of the RTOS-32 project needs to be available on the target system.

4.1 Recommended

It is recommended to create a network share for the rtfiles\debug directory on the target.

Note: **Every time** a DLM project is changed and rebuilt on the host it is **required** to copy it into the target again! Therefore it's recommended to use the network share as *output* directory of the *Visual Studio* project.

As our hypervisor is running on XUBUNTU, a SAMBA server is needed to accomplish this.

The hypervisor contains an installed SAMBA server, which must be properly configured for this specific task.

Hint: The used **default** user in this tutorial is rtv. If an **other** user is desired, change the specific parts accordingly.

Go to the SAMBA configuration file:

```
$ sudo gedit /etc/samba/smb.conf
```

Add the following section to the end of the smb.conf file **[if not yet available]** and save:

```
[debug]
  comment = Remote debug
  path = /hv/rtos-32/rtfiles/debug
  browseable = yes
  valid users = rtv
  guest ok = yes
  read only = no
```

Allow the user rtv to access the SAMBA share:

```
$ sudo smbpasswd -a rtv
```

Restart the SAMBA service:

```
$ sudo systemctl restart smbd.service nmbd.service
```

To check, if the share is active, try to access the SAMBA share from the Windows explorer.

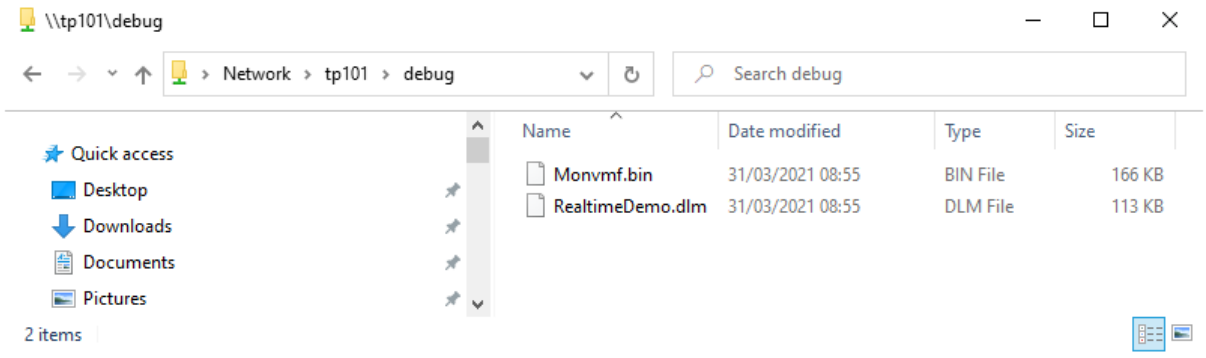

Fig. 4.1: Access to share from Windows file explorer.

4.2 Copy Debug Monitor

Copy the whole *Debug Monitor* directory `<host>\%WORKSPACE%\projects\monvmf\*.*` to `\\172.17.10.53\debug\*.*` and **override** the existing files. Leave the directory structure **intact**!

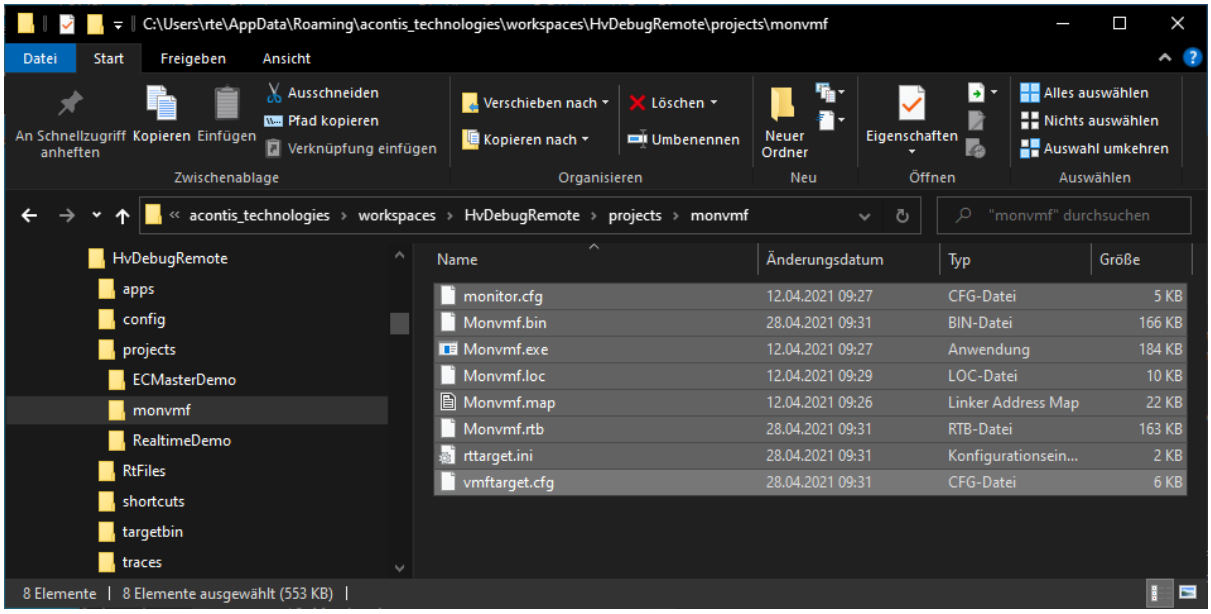

Fig. 4.2: Copy *Debug Monitor* directory.

Hint: Some of the files will be updated after rebuild of VS project.

4.3 Visual Studio Project Settings

The following screenshot shows how to set the output directory to a target system with IP address `172.17.10.53` and the network share name `debug`. This setting is required for **both** projects within the `Visual Studio` solution (*Loader* as well as the *DLM/DLL* project).

Caution: Adjust the `IP address` accordingly to your needs.

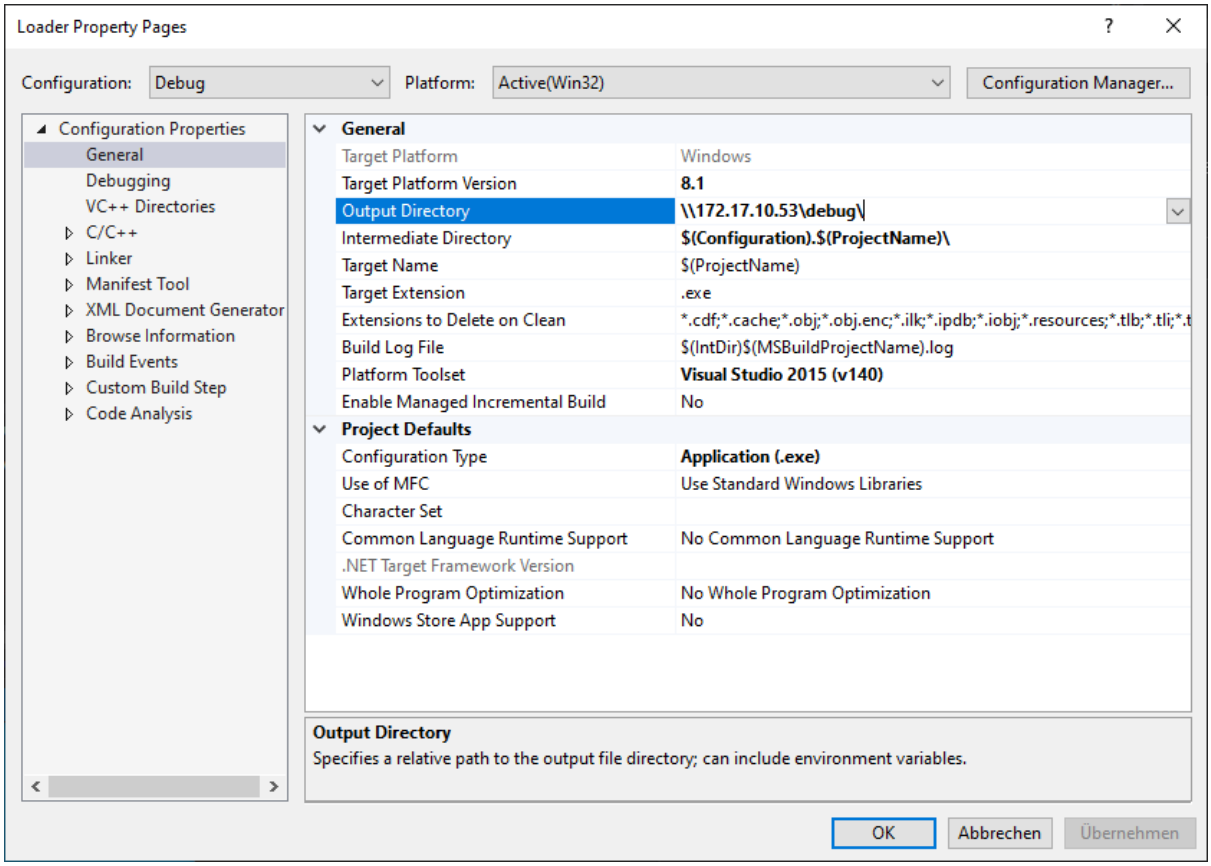

Fig. 4.3: Visual Studio Loader Project Settings.

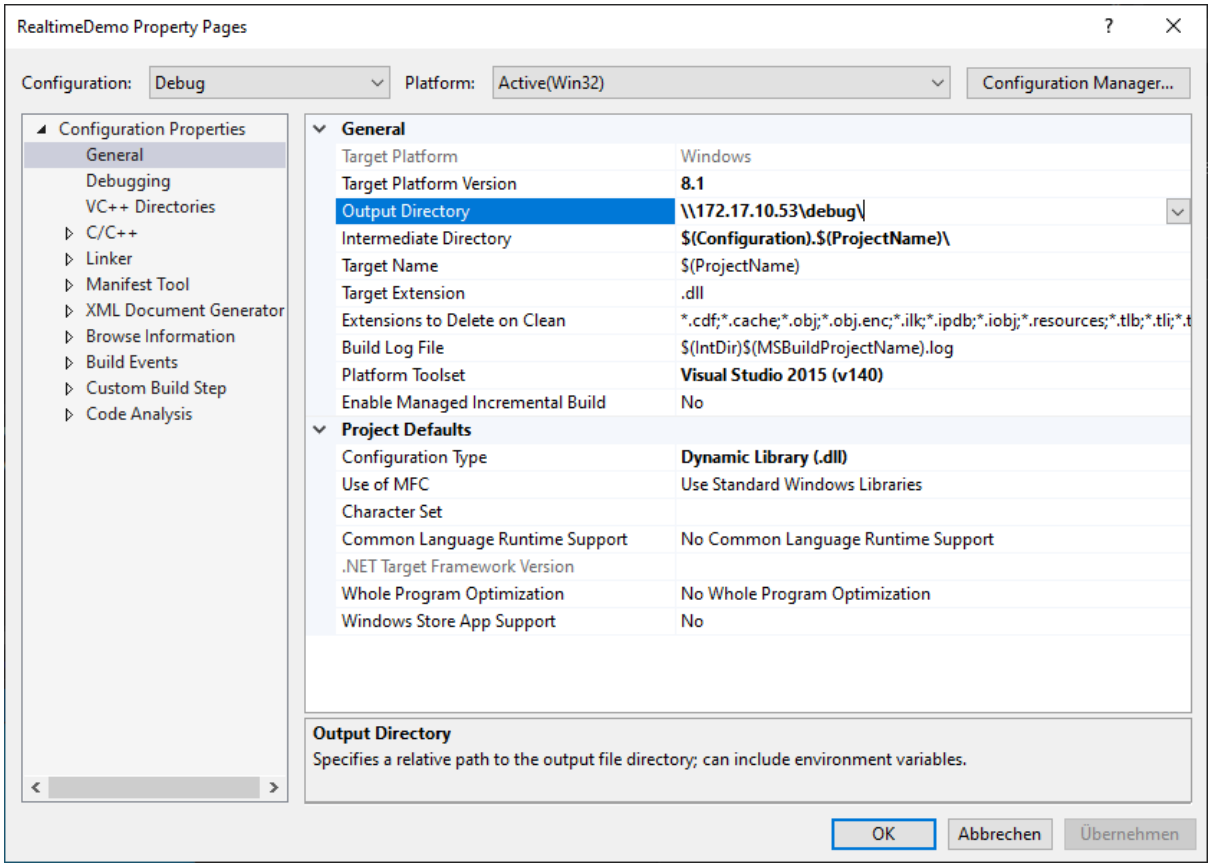

Fig. 4.4: Visual Studio RealtimeDemo Project Settings.

5 Part D - Hypervisor Target Configuration

Start *RTOS-32 Debug Monitor*:

```
$ cd /hv/rtos-32
$ sudo ./realtimedemo-debug.sh
```

The `realtimedemo-debug.sh` will automatically starts the output console `dbgcon.sh`.

The output of the *RTOS-32 Debug Monitor* should be similar to that:

```
RTTarget-32 Debug Monitor 6.23 Evaluation Version
Copyright (c) 1996,2021 On Time Informatik GmbH

Monitor Header at: 04037A3C, Current CPL: 0
Ethernet: RTOS32Win VNet, MAC: AA-BB-CC-DD-EE-03, IP: 172.17.10.5
```

If the aforementioned information isn't shown, start the output console `dbgcon.sh` manually to get the *RTOS-32 Debug Monitor* output.

```
$ cd /hv/rtos-32
$ ./dbgcon.sh
```

Open new shell (right click on desktop and select *'Open Terminal here'* **or** press `CRTL` + `ALT` + `T`)

Create bridged network connection:

```
$ cd /hv/hvctl
$ sudo ./brvnetset.sh
```

Output of successful bridge creation:

```
rtv@rtv-TEST:/hv/hvctl$ sudo ./brvnetset.sh
[sudo] password for rtv:
load vnet bridge configuration
add enp2s0 and vnet0 to bridge vnetbr
bridge name bridge id           STP enabled interfaces
vnetbr      8000.0060c8000000   no          enp2s0
                                            vnet0
ifconfig vnetbr 172.17.10.53 netmask 255.255.0.0
ip route add default via 172.17.5.2
RTNETLINK answers: No such process
```

Important: Remove bridged network connection after debugging!

```
$ cd /hv/hvctl
$ sudo ./brvnetclr.sh
```

Hint: See chapter “Bridge virtual and physical network” in the Hypervisor manual for details how to configure the bridge.

6 Part E - Development Host Configuration

- Scroll to `<Development|Debugging>` section and push the `Test Connection` button.

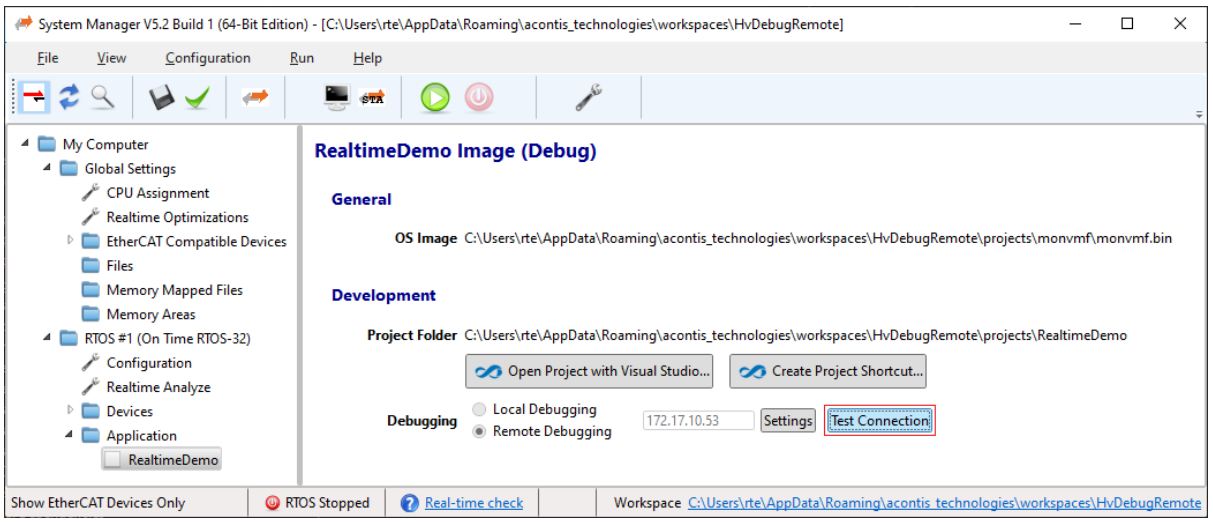

Fig. 6.1: Push `Test Connection` button.

- Successful

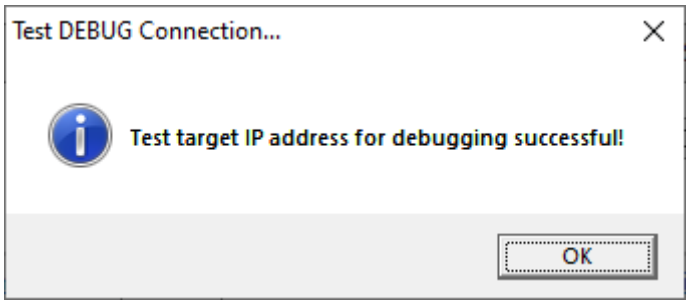

Fig. 6.2: Test Connection successful.

- Failed

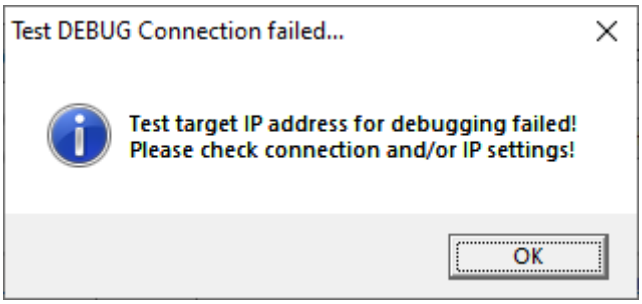

Fig. 6.3: Test Connection **NOT** successful.

- Pressing `F5` in Visual Studio will start the debugging.

7 Important hints

Important: After changing the `$Target-IP$` on host side, repeating parts `C - E` are **always** necessary! Furthermore, a running instance of Visual Studio with the current project **must** be restarted and then the project **must** be recompiled with `rebuild all` command.

Caution: Do **not** re-launch the debug monitor in an on-going debug session!