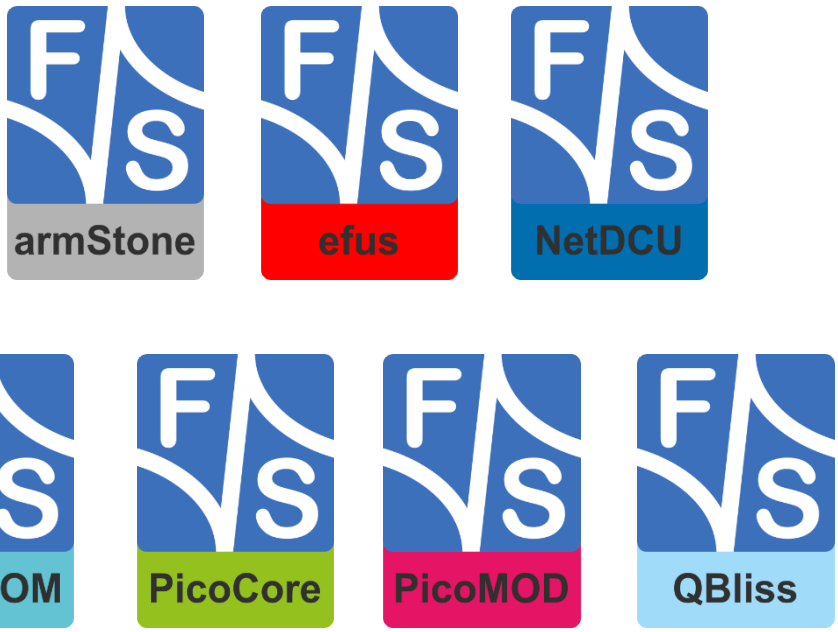
F&S Introduction to CRANK

Setup an Application on an embedded Device

Version 1.1
(2022-07-13)



Untere Waldplätze 23
D-70569 Stuttgart
Germany

Phone: +49(0)711-123722-0
Fax: +49(0)711-123722-99

About This Document

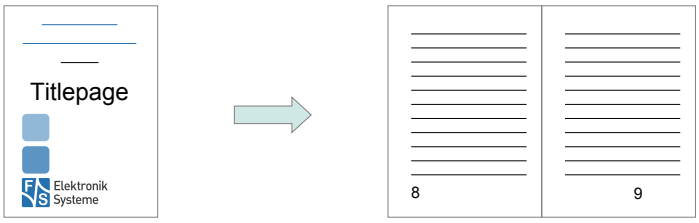
This document describes how to debugging remote device using QT Creator Automatic Remote Debugging Launcher under Linux. The software is configured for architectures fsimx6/fsimx6sx/fsimx6ul/fsimx7ulp/fsimx8mm from F&S under Linux/Buildroot.

Remark

The version number on the title page of this document is the version of the document. It is not related to the version number of any software release. The latest version of this document can always be found at http://www.fs-net.de.

How to Print This Document

This document is designed to be printed double-sided (front and back) on A4 paper. If you want to read it with a PDF reader program, you should use a two-page layout where the title page is an extra single page. The settings are correct if the page numbers are at the outside of the pages, even pages on the left and odd pages on the right side. If it is reversed, then the title page is handled wrongly and is part of the first double-page instead of a single page.



Typographical Conventions

We use different fonts and highlighting to emphasize the context of special terms:

`File names`

Menu entries

```
Board input/output
```

`Program code`

```
PC input/output
```

```
Listings
```

```
Generic input/output
```

`Variables`

History

Date	V	Platform	A,M,R	Chapter	Description	Au
2021-10-01	1.0	*	A	ALL	Create Documentation	PJ
2021-11-19	1.1	*	A, M	4-6	Added Debugging Chapter + Buildroot/Yocto differences	TG

V	Version
A,M,R	Added, Modified, Removed
Au	Author

Table of Contents

1 Introduction

F&S offers a whole variety of Systems on Module (SOM) and Single Board Computers (SBC). There are different board families that are named NetDCU, PicoMOD, PicoCore, PicoCOM, armStone, QBliss, and efus.

Crank Software is so much more than just a GUI design and development software company. Storyboard, our cross-platform embedded GUI development framework, in addition to our embedded engineering services team are the extended arms of UX-led companies around the globe. This document describes how to debug an application on a remote device using the automatic remote debugging launcher plug-in.

This document describes how to setup your embedded device and execute an example project from CRANK software with Crank Storyboard. More information can be found on the website of CRANK and also on their YouTube channel.

https://www.cranksoftware.com/

https://support.cranksoftware.com/hc/en-us/articles/360058858132-VIDEO-Preparing-your-embedded-GUI-for-Linux-deployment

1.1 Install Crank on your OS

To install CRANK software on your operating system please go to website https://www.cranksoftware.com/

and request the crank software installer. Afterwards follow the steps of the installer and execute the Crank Storyboard.

1.2 Tested Boards and Releases

Board:	Release:
PicoCoreMX8MM-V3	Buildroot (fsimx8mm-B2021.06)
PicoCoreMX8MM-V3	Yocto (fsimx8mm-Y2021.04)

2 Remote Connection

First of all the SSH connection on the remote system will be setup.

2.1 Setup SSH connection on SBC/SOM

You have to setup your F&S Board. Therefore you can have a look into F&S *LinuxOnFSBoards_eng.pdf*. After that boot your F&S Board.

To work with SSH the board should have a valid date. This is necessary to create certificates for SSH. To setup a date you can use the following command:

```
date "2020-05-04 10:13"
```

Afterwards we have to enable the network interface. You can also set the network on command in UBoot to enable network interface at each boot. For further information please take a look in *FSiMX_FirstSteps_eng.pdf*.

Dynamically:

```
udhcpc -i eth0
```

Static (e.g. ip address is 10.0.0.84):

```
ifconfig eth0 10.0.0.84 up
```

The Root-Filesystem is read-only mounted, but we have to modify something in the filesystem so we need it read-writeable.

```
mount -o remount,rw /
```

Open *sshd_*config file

```
vi /etc/ssh/sshd_config
```

and edit the following lines:

```
…
PermitRootLogin yes
…
```

Optional you can also allow to login without password, but we recommend you to not do this because it's a security risk. If you want to do it anyway add the following line to *sshd_config* file.

```
…
PermitEmptyPasswords yes
…
```

After that you have to start the SSH server. (only necessary if you are using Buildroot)

```
/etc/init.d/S50sshd start
```

Now the SSH server is running on our SBC/SOM. Let's test it, therefore we are going back to our VM. To connect via ssh we open a Terminal and send the following command:

```
ssh root@10.0.0.84
```

```
[jakob@localhost ~]$ ssh root@10.0.0.84
The authenticity of host '10.0.0.84 (10.0.0.84)' can't be established.
ECDSA key fingerprint is b1:b1:aa:83:12:d1:f1:21:7b:e3:6a:61:89:6e:31:ea.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.0.84' (ECDSA) to the list of known hosts.
```

Figure 1: SSH connection terminal

Now the SSH connection is successfully established and tested.

3 Working with Crank Storyboard

3.1 Import an example application

After everything is setup, you can create an import an example project. From *File* select I*mport*…. Select *Storyboard Development* and *Storyboard Sample* and click Next >.

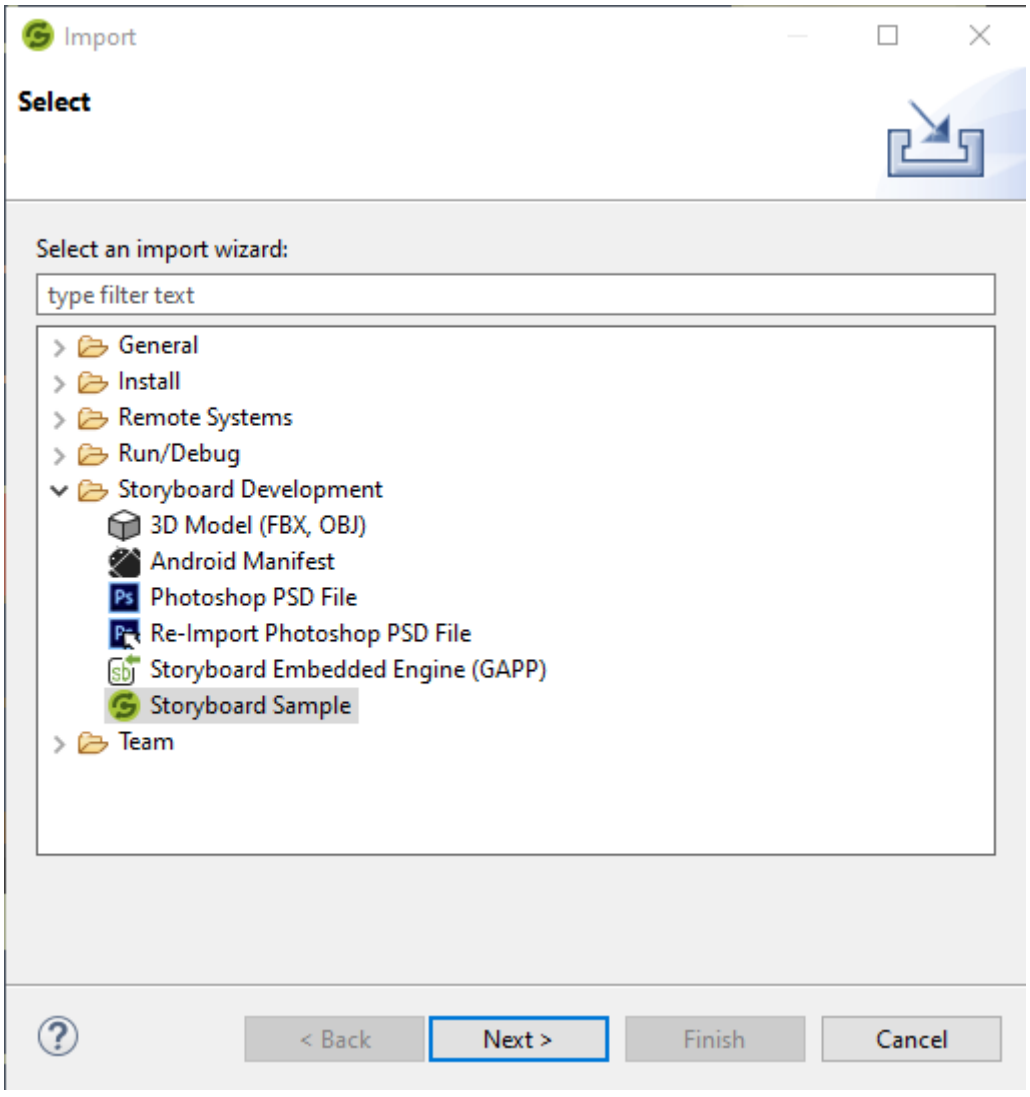


Figure 2: Select Storyboard Sample

Choose a Storyboard Sample and the name of the project. In this case we are using *AddressBook* Press *Finish*.

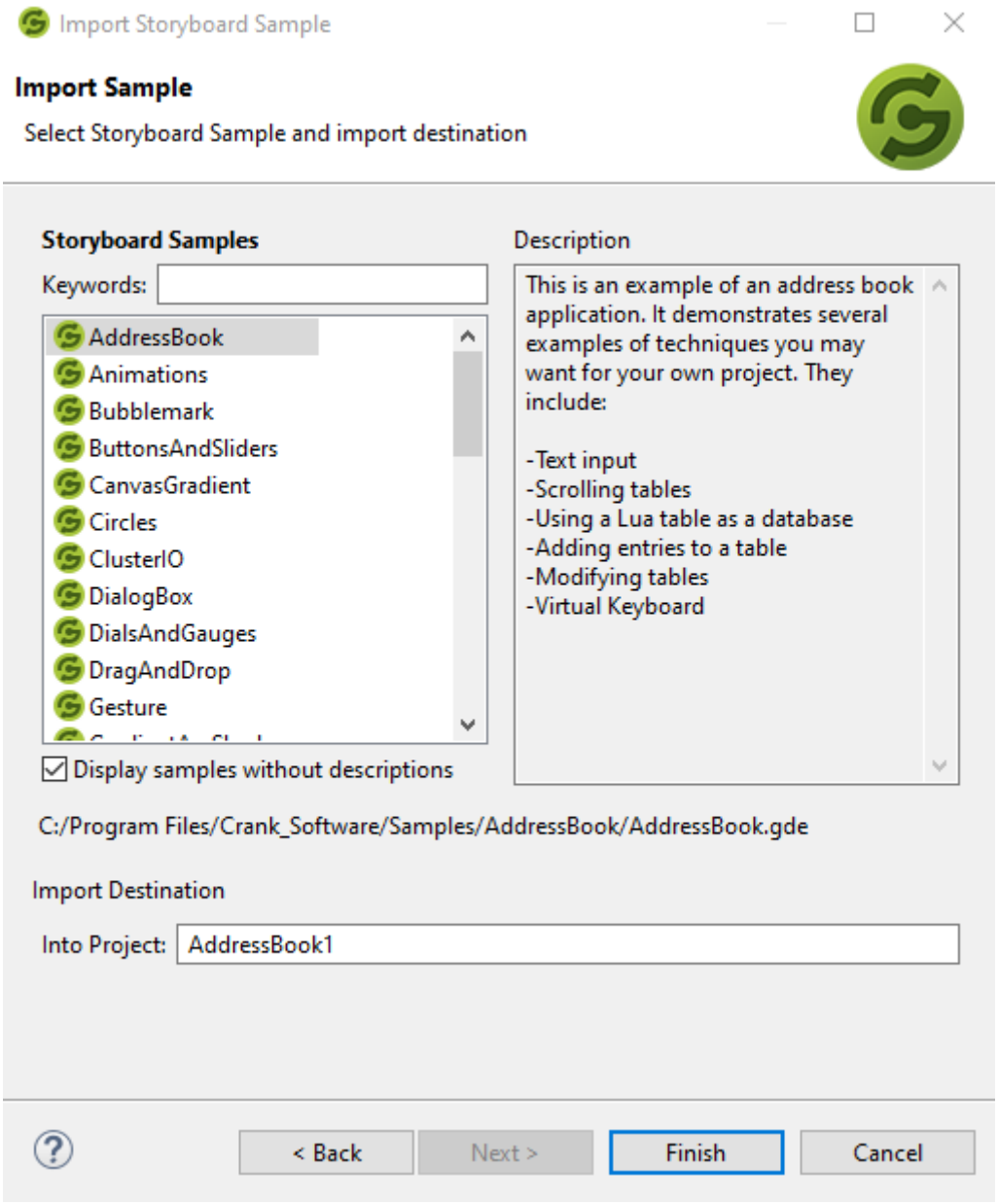


Figure 3: Choose Storyboard sample

Now the project is successfully imported.

3.2 Setup SSH connection on crank software

Open *Storyboard Application Export Configurations*  in the top bar of Crank Storyboard. Now s*elect Storyboard Embedded Engine (GAPP).Select SCP Transfer under the Transfer tab.* Setup the fields with the correct input, see example picture below (change the paths to /home/root/… if you are using Yocto).

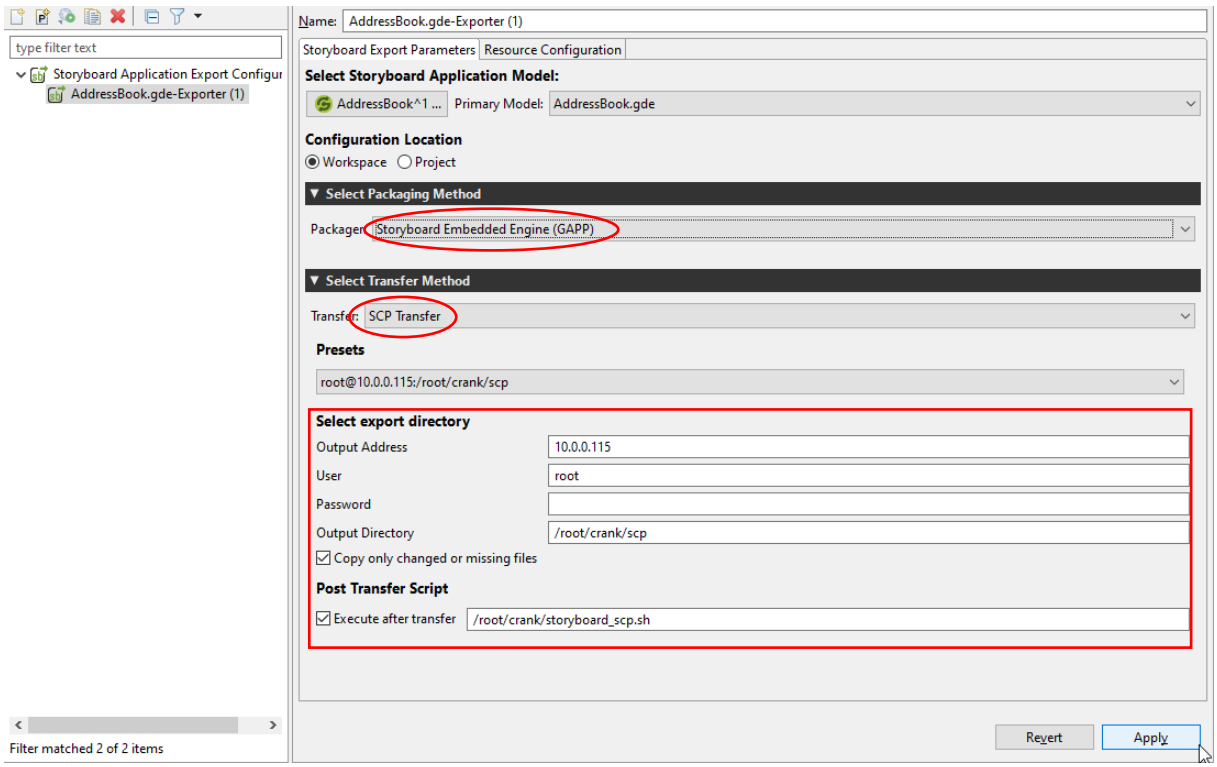


Figure 4: Setup Storyboard Application Export Configurations

After pressing *Apply* you can *close* it.

4 Prepare embedded Device

4.1 Copy Storyboard Engine to your embedded device

Before we can execute the corresponding example we have to copy the storyboard engine and toolchain to the embedded device. The Storyboard toolchain can be found in your install directory of Crank Storyboard framework. Within this folder you can find another folder called *Storyboard_Engine*.

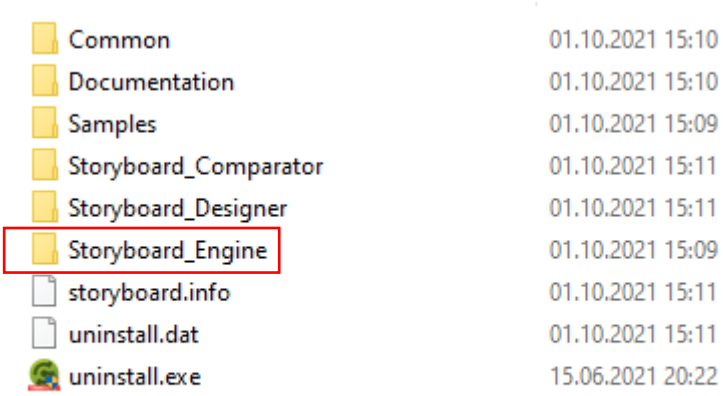


Figure 5: Folder structure of Crank Storyboard

In this folder and subfolder there are all toolchains which are supported. In our case we have an imx8 platform so we have to copy the folder *linux-imx8yocto-armle-opengles_2.0-wayland-obj* to our embedded device.

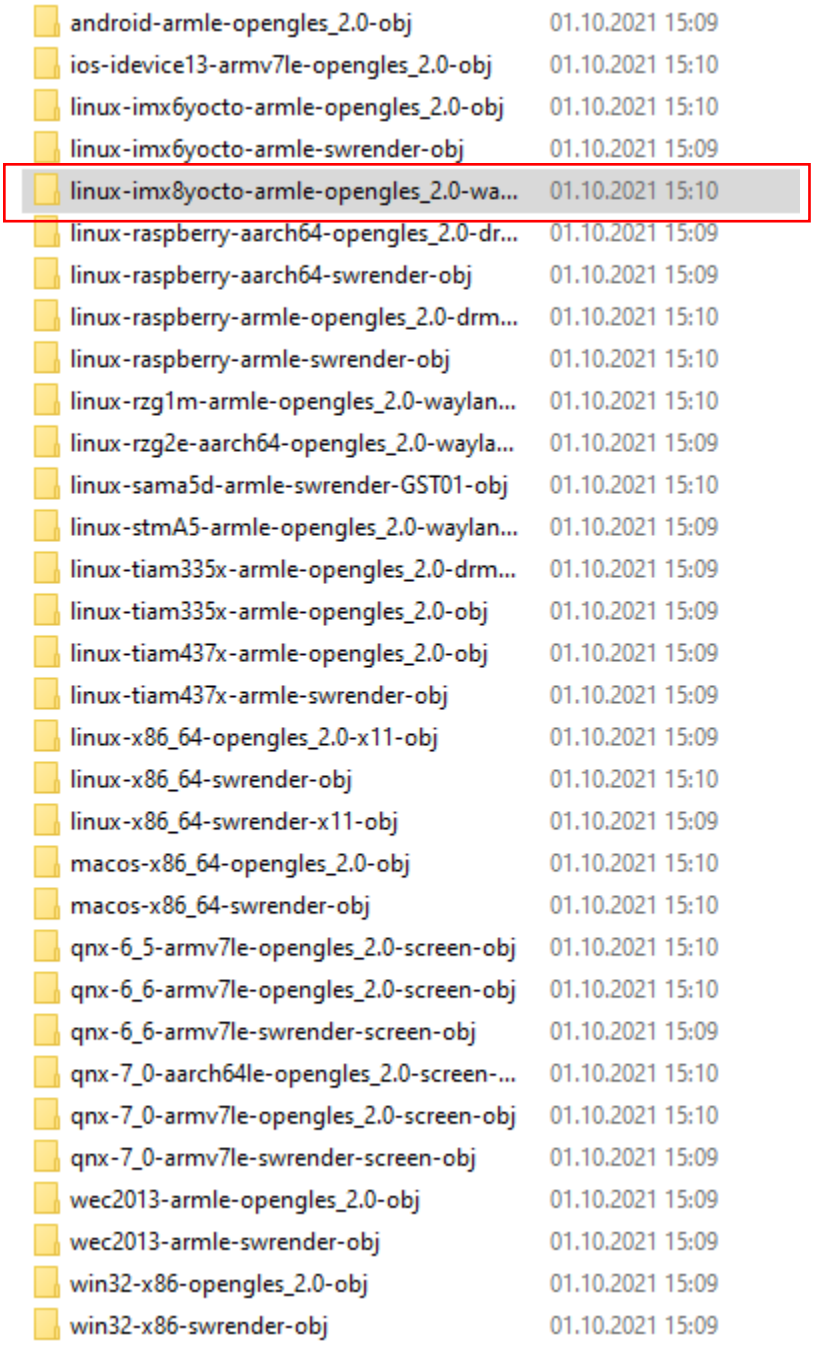


Figure 6: Structure of Storyboard_Engine

There are several ways how to copy the folder to our embedded device. Please refer to LinuxOnFSBoards documentation. Copy this folder to the directory /root/ (/home/root on yocto) in your rootfs of your embedded device.

4.2 Create an script to start the application

Go to your embedded device and create a script called *storyboard_scp.sh* under */root/crank/*. (or /home/root/crank if you are using Yocto)The script must have the following entries:

```
#!/bin/sh


killall sbengine


echo "Starting Storyboard from SCP..."


echo 0 >> /sys/class/graphics/fbcon/cursor_blink
echo -e '\033[9;9]' >> /dev/tty1
echo 0 >> /sys/class/graphics/fb0/blank


export SB_DEMO=1
export SB_VIDEO=0
#Buildroot
#export SBROOT=/root/linux-imx8yocto-armle-opengles_2.0-wayland-
obj
#Yocto
#export SBROOT=/home/root/linux-imx8yocto-armle-opengles_2.0-
wayland-obj
export SB_ENGINE=$SBROOT/bin/sbengine
export SB_PLUGINS=$SBROOT/plugins
export LD_LIBRARY_PATH=$SBROOT/lib
#Buildroot
#export APP_PATH=/root/crank/scp/AddressBook.gapp
#Yocto
#export APP_PATH=/home/root/crank/scp/AddressBook.gapp


OPTIONS="-orender_mgr,fullscreen"
#debug
#OPTIONS="-vv -orender_mgr,fullscreen"


$SB_ENGINE $OPTIONS $APP_PATH
```

Simply uncomment the one you need. If you are using Buildroot, then you need to uncomment the one under the #Buildroot mark. If you are using Yocto uncomment the ones under the #Yocto mark.

(Note: only uncomment the one you need else it may cause your application to work improperly)

You will also need to give your script the needed permissions. You can do that by using the following command.

```
chmod u+x storyboard_scp.sh
```

After that, you need to go into your Storyboard engine directory and give your sbengine the same rights. Use the following command in the bin directory of the engine you transferred earlier.

```
chmod u+x sbengine
```

(Note: if you are using Yocto you need to set a link to a file. You can do that by going into your /usr/lib directory and typing the following command :)

```
ln -s libwayland-egl.so.1.0.0 libwayland-egl.so
```

5 Run demo application

Open *Storyboard Application Export Configurations*  in the top bar of Crank Storyboard. Now check your setup which you have done in 3.2. Setup the fields with the correct input, see example picture below. (Especially keep an eye on the paths and make sure that they are correct)

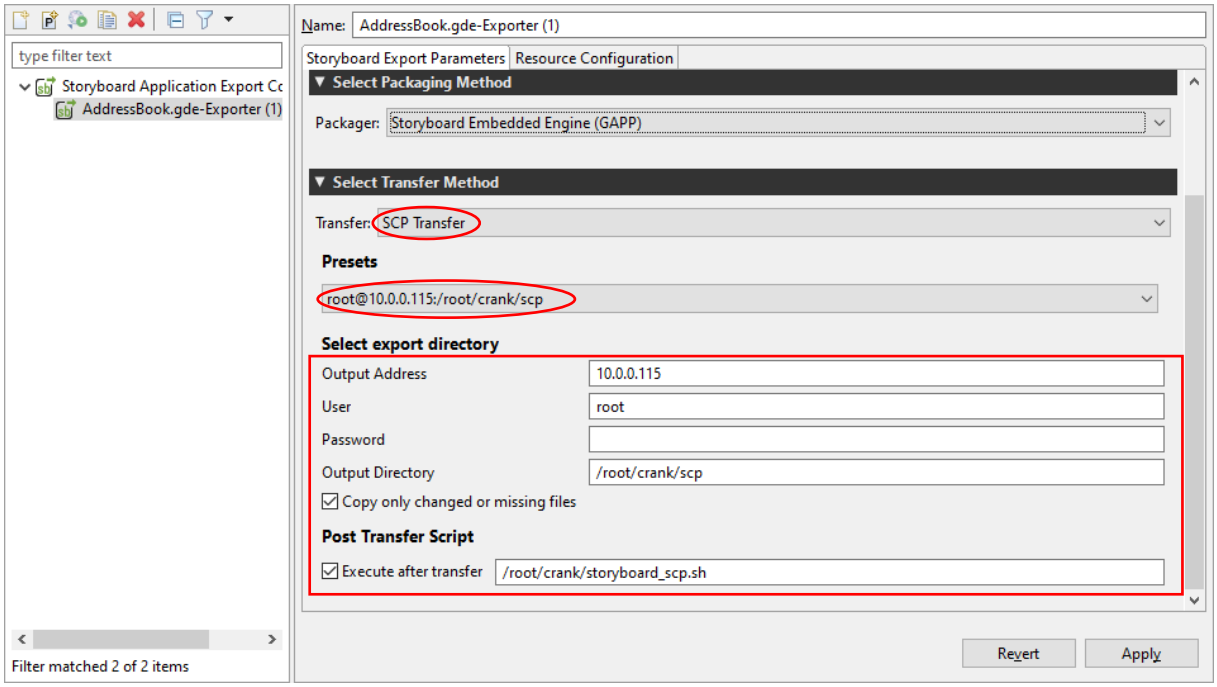


Figure 7: Setup Storyboard Application Export Configurations

After pressing *Apply*, *run* the demo application should be transferred and directly executed on the embedded device.

6 Debugging your Application

The Lua debugger is configured such that it can only be used with the simulator runtimes on the host platforms that support Storyboard Designer. For assistance in configuring the debugger for embedded targets, contact Crank Software support (support@cranksoftware.com).

Debugging your application on your host system is possible through the Storyboard Application. Simply head over to the Storyboard Simulator Configurations.

Figure 8: Storyboard Simulator Configurations

Next you want to make a new Configuration and use the win32 Storyboard Engine. Then you will need to go to ‘lua’ in ‘Plugin Options’ and enable the Lua Debugger. Now you are ready to debug your application.

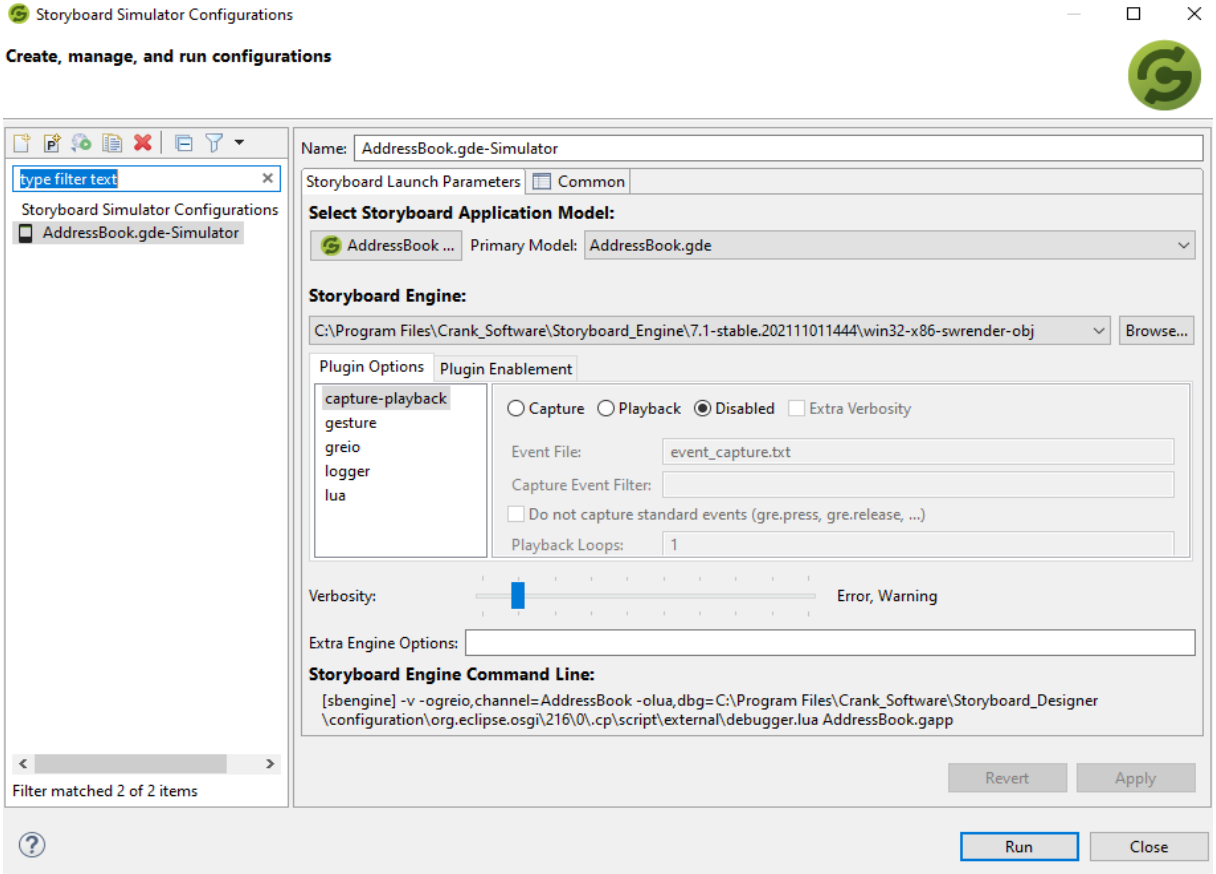


Figure 9: Storyboard Simulator Configuration

7 Appendix

List of Figures

Important Notice

The information in this publication has been carefully checked and is believed to be entirely accurate at the time of publication. F&S Elektronik Systeme assumes no responsibility, however, for possible errors or omissions, or for any consequences resulting from the use of the information contained in this documentation.

F&S Elektronik Systeme reserves the right to make changes in its products or product specifications or product documentation with the intent to improve function or design at any time and without notice and is not required to update this documentation to reflect such changes.

F&S Elektronik Systeme makes no warranty or guarantee regarding the suitability of its products for any particular purpose, nor does F&S Elektronik Systeme assume any liability arising out of the documentation or use of any product and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

Products are not designed, intended, or authorised for use as components in systems intended for applications intended to support or sustain life, or for any other application in which the failure of the product from F&S Elektronik Systeme could create a situation where personal injury or death may occur. Should the Buyer purchase or use a F&S Elektronik Systeme product for any such unintended or unauthorised application, the Buyer shall indemnify and hold F&S Elektronik Systeme and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim of personal injury or death that may be associated with such unintended or unauthorised use, even if such claim alleges that F&S Elektronik Systeme was negligent regarding the design or manufacture of said product.