

UM3148

User manual

Getting started with X-LINUX-MEMS1 package for developing MEMS Applications on Linux

Introduction

The X-LINUX-MEMS1 is an STM32MPU OpenSTLinux software expansion package for the STM32MP157F-DK2 discovery kit, used to demonstrate sensor fusion (STM32_MotionFX_Library) running on the Cortex® M4, and sensor data logging application running on the Cortex® A7, of the STM32MP1.

The X-LINUX-MEMS1 software package is used with X-STM32MP-MSP01 multi-sensor board connected to a 40-pin GPIO header of the STM32MP1 DK2 board. The motion FX library running on the Cortex® M4 uses MEMS sensors data (ISM330DHCX: accelerometer and gyroscope. IIS2MDC: magnetometer) and sends data over RPMsg (remote processor messaging) to the Cortex® A7.

The firmware running on the Cortex® M4 is based on an X-CUBE-MEMS1 package.



Figure 1. X-LINUX software architecture diagram





1 X-LINUX-MEMS1 overview

The X-LINUX-MEMS1 software provides a user space application running on STM32MP157F-DK2 for the X-STM32MP-MSP01 expansion board with multiple motion sensors. The X-LINUX-MEMS1 software package is composed of 3 modules:

- 1. Data logger (C firmware running on Cortex® M4) containing firmware drivers for accelerometer, gyroscope and magnetometer.
- 2. Sensor fusion (STM32_MotionFX_Library running on Cortex® M4).
- 3. User application based on RPMsg (C application based on RPMsg running on Cortex® A7 and receiving data from Cortex® M4).

1.1 Features

The main features of the X-LINUX-MEMS1 are as follows:

- Firmware for Cortex®M4 for running Motion FX
- C console application based on RPMsg running on Cortex A7 for data logging
- Compatible with STM32 ODE
- Environment sensors driver included
- Free, user-friendly license terms

1.2 Software configuration

Apart from other settings there are two main settings for the software configuration:

- 1. CRC2 enable for Motion_FX running on Cortex® M4
- 2. RPMSG for communication between Cortex® M4 and Cortex® A7

Figure 2. IOC File Settings





1.3 Software package structure

The folder structure of the software package is below:



1.3.1 Cortex® M4 firmware

Cortex® M4 is firmware for MEMS data reading over I²C. The application is built for the below MEMS sensors:

- 1. ISM330DHCX: 3-axis accelerometer and 3-axis gyroscope.
- 2. IIS2MDC: 3-axis digital magnetometer.
- 3. IIS2DLPC: 3-axis accelerometer for industrial applications.

It uses the motion_FX sensor fusion library to show the sensor fusion data such as heading and quaternion on the console C – application

Figure 5. MotionFX Library for Cortex M4

| MotionFX_CM4F_wc16_ot.a |
|------------------------------|
| MotionFX_CM4F_wc32_ot.a |
| MotionFX_CM4F_wc32_ot_hard.a |



1.3.2 Cortex® A7 application

The RPMsg is a console C application based on the RPMsg framework used to receive the data from the Cortex® M4 at regular intervals. A high level application (QT, GTK or Python) can be made on top of this data, as shown below, and is sent at regular intervals.



Figure 6. Sensor data logging application



2 Hardware setup

57

The X-LINUX-MEMS1 software works with the X-STM32MP-MSP01 board, which is plugged on the 40-pin GPIO connector present on the STM32MP157F-DK2 platform.



Figure 7. X-STM32MP-MSP01 expansion board

Figure 8. X-STM32MP-MSP01 expansion board on STM32MP1 board





3 Software setup

This section describes the software setup, which is required for building, flashing, transferring, and running the MEMS application.

3.1 Recommended PC prerequisites

A Linux® PC running under Ubuntu® 20.04 is to be used. The developer can follow the link below: https://wiki.st.com/stm32mpu/wiki/PC prerequisites

3.2 Installing the SDK

This software package is built for the STM32 MPU ecosystem 4.1. This is required to build the application package. The package contains the binaries which can be transferred using the scp command or zmodem. In case customization is needed in the application, installing the SDK helps in building it. The developer can follow the link below:

STM32MPU SDK Installation.

Once the SDK is installed, export the tool chain using the below command to build any Cortex® A7 application:

source . /<path to SDK Installation Directory>/environment-setup-cortexa7t2hf-neon-vfpv4-ost l-linux-gnueabi

3.3 Downloading and running the package

This is required to build the X-LINUX-MEMS1 Yocto recipe for the distribution package and for creating the STM32MP1 SD card flashable images, which has the MEMS application (elf, rpmsg, Python) embedded. The developer can follow the given link to download the Distribution Package:

3.4 Connecting to the board and transferring files

This is required to transfer the built binaries (application, elf, Python) to the STM32MP157F-DK2 board from the development PC. The developer can transfer the binaries either by a:

- 1. a hotspot method
- 2. or setting up a Wi-Fi connection in the Board: How to setup a WLAN connection stm32mpu
- 3. or using any serial protocol (like zmodem from Tera Term)



4 Building the application

The Cortex® M4 firmware is built using STM32CubeIDE, while the rmpsg application is built using the tool chain provided in the STM32MP1 SDK.

4.1 Starter Packages

Download or clone the package (X-LINUX-MEMS1_V1.0.0.tar.xz) from www.st.com and extract it.

\$tar xvf X-LINUX-MEMS1_V1.0.0.tar.xz

You get the X-LINUX-MEMS1_V1.0.0 folder as shown in Figure 4. Folder content overview.

4.2 Cortex® M4 firmware

The firmware workspace is built using STM32CubeIDE:

Figure 9. X-LINUX-MEMS1 Cortex® M4 folder structure

X-LINUX-MEMS1_V1.0.0 > Application > Source > X-LINUX-MEMS1 >





Figure 10. STM32CubeIDE Cortex® M4 workspace

| Project Explorer X | E % 7 1 * | | R main c X | | | | |
|-----------------------------|----------------|------------------------------|--|--|---------------|---------|-------|
| X-LINUX-MEMS-FUSION1 (in X) | (-LINUX-MEMS1) | ^ | 1 /* USER CODE BEGIN Header */ | | | | |
| > 🗁 Common | | | 20/** | | | | |
| > 🗁 Drivers | | | 3 ********************* | ***************** | *********** | ****** | ***** |
| > 🗁 Middlewares | | | 4 * @tile : main.c | again bada | | | |
| > 😂 Utilities | | | 6 ************************************ | ************************************** | *********** | ****** | ***** |
| X-LINUX-MEMS-FUSION1_C | CA7 (in CA7) | | 7 * @attention | | | | |
| > 🗁 DeviceTree | | | 8 * | | | | |
| manifest.prop | | | 9 * Copyright (c) 2022 STMicro | oelectronics. | | | |
| V X-LINUX-MEMS-FUSION1_C | CM4 (in CM4) | | 10 * All rights reserved. | | | | |
| Binaries | | | 11 * | | | | |
| > 🔊 Includes | | | 12 * This software is licensed | under terms that ca | n be tound in | the LI | CENSI |
| > 🤮 Common | | | 14 * TE no LTCENSE file compres | with this software compon | it is provide | A AC TC | |
| v 🥴 Core | | | 15 * | with this soleware, | re is provide | u no 10 | • |
| > 😂 Inc | | | 16 ********************* | ****** | ********** | ****** | **** |
| V 😓 Src | | | 17 */ | | | | |
| > app_mems.c | | | 18⊕/* USER CODE END Header */ | | | | |
| > 🙆 com.c | | | 19 /* Includes | | | | |
| > demo serial.c | | | 20 #include "main.h" | | | | |
| > iks01a3 mems cor | ntrol_ex.c | | 22 #include "stm22mp1vy hal h" | | | | |
| > iks01a3 mems cor | ntrol.c | | 23 #include "virt wart.h" | | | | |
| > lock_resource.c | | | 24 #include "serial protocol.h" | | | | |
| > i main.c | | | 25 #include "com.h" | | | | |
| > i motion_fx_manage | er.c | | 26@/* Private includes | | | | |
| > serial protocol.c | | | 27 /* USER CODE BEGIN Includes *, | / | | | |
| > stm32mp15xx disc | co bus.c | | 28 20 /8 UCED CODE END Tabledas 8/ | | | | |
| > istm32mp15xx disc | co.c | | 29 /* USER CODE END INCLUDES **/ | | | | |
| stm32mp1xx hal msp.c | | 30 310 /* Private typedef | | | | | |
| > stm32mp1xx it.c | | | 32 /* USER CODE BEGIN PTD */ | | | | |
| > R syscalls.c | | | < | | | | |
| > sysmem.c | | | Problems × Plasks Console Prope | erties | | 7 : - | |
| > 🕞 Startup | | | 0 errors 14 warnings 0 others | | | | |
| > C Drivers | | | Description | Decourse | Dath | | Loc |
| > 🤒 Middlewares | | | Mamings (14 items) | nesource | rau | | LUC |
| > 🥴 OPENAMP | | | v warnings (14 items) | | | | |
| > Debug | | | | | | | |
| > BemoteProc | | | | | | | |
| STM32MP157FACX RAM | Ud | | | | | | |
| ICENSE.txt | | | | | | | |
| readme.html | | ~ | < | | | | > |

After building the application, X-LINUX-MEMS-FUSION1_CM4.elf is formed at X-LINUX-MEMS1\X-LINUX-MEMS\CM4\Debug.

4.3 Cortex® A7 application

The rpmsg can be built using the makefile provided via the make command. Before building using the make command, tool chain for STM32MP1SDK needs to be exported using source command mentioned in Section 3.2 Installing the SDK.

Makefile and the rpmsg source are present at: X-LINUX-MEMS1_V1.0.0\Application\Source\X-LINUX-MEMS1\Utilities\C\rpmsg.

4.4 Distribution package

This is required to build the Yocto recipes and create STM32MP1 images which has x-linux-mems1 application embedded in them.

Step 1. Create a directory for the distribution package and initialize repo in the current directory

```
PC $> repo init -u https://github.com/STMicroelectronics/oe-manifest.git -b refs/tag
s/openstlinux-5.15-yocto-kirkstone-mp1-v22.11.23
```

PC \$> repo sync

Step 2. Initializing the OpenEmbedded build environment for STM32MP1

PC \$> DISTRO=openstlinux-weston MACHINE=stm32mp1 source layers/meta-st/scripts/envse tup.sh

Step 3. Download the X-LINUX-MEMS1 application package

PC \$> cp -rf X-LINUX-MEMS1 V1.0.0/Recipe/meta-st-mems1/ ../layers/meta-st/

copy it into the layers/meta-st folder inside the build directory



Step 4. Add meta-st-mems1 layer

PC \$> bitbake-layers add-layer ../layers/meta-st/meta-st-mems1

Step 5. Update the configuration to add new components in the image

PC \$> echo 'IMAGE_INSTALL:append = "mems1"' >> ../layers/meta-st/meta-st-openstlinux
/conf/layer.conf

or

PC \$>vi ../layers/meta-st/meta-st-openstlinux/conf/layer.conf

IMAGE_INSTALL: append = "mems1"

Step 6. Build the image

PC \$> bitbake st-image-weston

New Images are formed in the tmp-glibc/deploy/images/stm32mp1/ directory

```
PC $> ls -l tmp-glibc/deploy/images/stm32mp1/FlashLayout_sdcard_stm32mp157f-dk2-trus ted.tsv
```

and FlashLayout_sdcard_stm32mp157f-dk2-trusted are created besides other images. Follow the "link" to flash the binary.

Step 7. Check if the file below is present on the discovery kit

\$ ls -l /lib/firmware

Elf file and rpmsg utility will be present.

4.5 Running the application

4.5.1 Using rpmsg utility

4.5.1.1 Starter package

The .elf and rpmsg can be built as explained in the above section. There are prebuilt binaries which can be pushed to the discovery kit at /lib/firmware.

Change the permission of rpmsg and run the application

\$chmod a+x rpmsg

\$./rpmsg

4.5.2 Using Python utility

Transfer X-LINUX-MEMS1_V1.0.0\Application\Source\X-LINUX-MEMS1\Utilities\sensor.py utility to /lib/firmware Run the below command in the terminal of the STM32MPU and Python GUI will be launched on the STM32MPU LCD Screen.

PC \$python3 sensor.py



Figure 11. Running Python GUI



Figure 12. Sensor.py Python GUI

| | X-STM32MP-MSP01 Sensors |
|--------------------------------|--------------------------------|
| Temperature (STTS22HH) | 31.37 deg C |
| Pressure (LPS22HH) | 988.97 Pa |
| Accelerometer (ISM330DHCX) | [27, -3, 1005] mg |
| Gyroscope (ISM330DHCX) | [70, -490, -210] deg/s |
| Magnetometer (IIS2MDC) | [-16641, -80385, 67071] mgauss |
| Quaternion (MotionFX) | [0.01, -0.76, 0.65] |
| Rotation Vector (MotionFX) | [98.61, 1.51, -0.13] |
| Gravity Vector (MotionFX) | [0.00, 0.03, -1.00] |
| Linear Acceleration (MotionFX) | [-0.00, -0.00, 0.01] m/s^2 |
| Heading (MotionFX) | 98.61 |
| Heading Error (MotionFX) | 3.14 |
| Elapsed time | 561 |
| ■ * | Thu Apr 28, 07:01 PM |

4.5.3 Distribution package

Once flashed using STM32CubeProgrammer, these files (.elf, rpmsg, sensor.py) can be found at the '/lib/firmware' location.



Revision history

Table 1. Document revision history

| Date | Revision | Changes |
|-------------|----------|-----------------------------|
| 05-May-2023 | 1 | Initial release. |
| 08-Jun-2023 | 2 | Watermark has been removed. |



Contents

| 1 | X-LI | NUX-MI | EMS1 overview | |
|-----|--------|----------|--|----|
| | 1.1 | Featur | res | |
| | 1.2 | Softwa | are configuration | |
| | 1.3 | Softwa | are package structure | |
| | | 1.3.1 | Cortex® M4 firmware | |
| | | 1.3.2 | Cortex® A7 application | 4 |
| 2 | Harc | lware s | etup | 5 |
| 3 | Soft | ware se | etup | 6 |
| | 3.1 | Recon | nmended PC prerequisites | 6 |
| | 3.2 | Installi | ing the SDK | 6 |
| | 3.3 | Downl | oading and running the package | 6 |
| | 3.4 | Conne | ecting to the board and transferring files | 6 |
| 4 | Buil | ding the | e application | |
| | 4.1 | Starte | r Packages | |
| | 4.2 | Cortex | ® M4 firmware | |
| | 4.3 | Cortex | ® A7 application | |
| | 4.4 | Distrib | oution package | |
| | 4.5 | Runnii | ng the application | |
| | | 4.5.1 | Using rpmsg utility. | |
| | | 4.5.2 | Using Python utility. | |
| | | 4.5.3 | Distribution package | 10 |
| Rev | vision | history | · | 11 |



List of tables

| Table 1. | Document revision history | 11 |
|----------|---------------------------|----|
|----------|---------------------------|----|



List of figures

| Figure 1. | X-LINUX software architecture diagram |
|------------|---|
| Figure 2. | IOC File Settings |
| Figure 3. | Release package structure Top Level |
| Figure 4. | Folder content overview |
| Figure 5. | MotionFX Library for Cortex M4 |
| Figure 6. | Sensor data logging application |
| Figure 7. | X-STM32MP-MSP01 expansion board |
| Figure 8. | X-STM32MP-MSP01 expansion board on STM32MP1 board 5 |
| Figure 9. | X-LINUX-MEMS1 Cortex® M4 folder structure |
| Figure 10. | STM32CubeIDE Cortex® M4 workspace |
| Figure 11. | Running Python GUI |
| Figure 12. | Sensor.py Python GUI |

IMPORTANT NOTICE - READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2023 STMicroelectronics – All rights reserved