Setting up and using Xilinx KRIA KV260

南山大学

2023-8-24

Vincent Conus - Source available at GitLab

Contents

1 Introduction

1.1 Motivation

This guide will present how to setup and use Xilinx's KRIA board, in particular for running ROS on a host Ubuntu system, as well as for deploying micro-ROS as a firmware on the MCU part of this board's chip.

The use of this device in particular is interesting because of the presence of a CPU comprising both a general purpose ARM core, capable of running a Linux distribution, as well as another ARM core, real-time enabled, capable to run a RTOS.

1.2 Build instructions for this report

The base file for this report is actually this README.org file itself. However, upon local build, this file is regularly exported as a `.tex` file that can be built normally. On a moderately recent Ubuntu-base distribution, the following packages seemed to be required to build the report:

```
sudo apt-get install texlive-base texlive-latex-recommended texlive-lang-japanese
```

Then, the actual build can be made with a simple:

```
pdflatex README.tex
```

No fancy Lua or theme at the moment !

1.2.1 Automatic build with CI/CD pipeline

If you don't want to build the report yourself, a CI pipeline is used to make it on GitLab.

You can check the steps in the .gitlab-ci.yml file. This build uses a base Ubuntu image and basically takes the same steps as presented above for a local build.

A PDF artifact can be downloaded.

1.2.2 Headers and LaTeX settings for export

A large amount of headers and parameters are needed in order to have this "README" document being exportable as a LaTeX document formatted the way I wanted it to be.

The detail can be seen in the raw `.org` version of this README.

2 Boot firmware

The goal for the Linux side of the deployment is to have the latest LTS version of Ubuntu up and running. In order to be able to boot such a newer version of Linux, the boot image of the board must first be updated.

The procedure is available in the official documentation, but I will present it step by step here.

2.1 Getting the new firmware

A 2022 version of the board firmware is required in order to run the latest version of Ubuntu properly.

The image can be downloaded at the atlassian page on the topic, or even directly with the following command:

```
wget https://www.xilinx.com/member/forms/download/\
     design-license-xef.html?filename=BOOT-k26-starter-kit-20230516185703.bin
```

2.2 Reaching the board recovery tool

Now the firmware `.bin` image is available, it is possible to update it using the boards recovery tool. Here are the steps that must be taken in order to reach this tool and update the board:

- Connect the board to your machine via a Ethernet cable. This will obviously cut you internet access, so you should be set for that.
- Select the wired network as your connection (must be "forced", since it doesn't have internet access).
- Set a fixed IP address for your machine, in the `192.168.0.1/24` range, except the specific `192.168.0.111`, which will be used by the board.
- Using a web browser on your host machine, access `http://192.168.0.111`. Thou shall now see the interface, as visible on the figure 1 below.

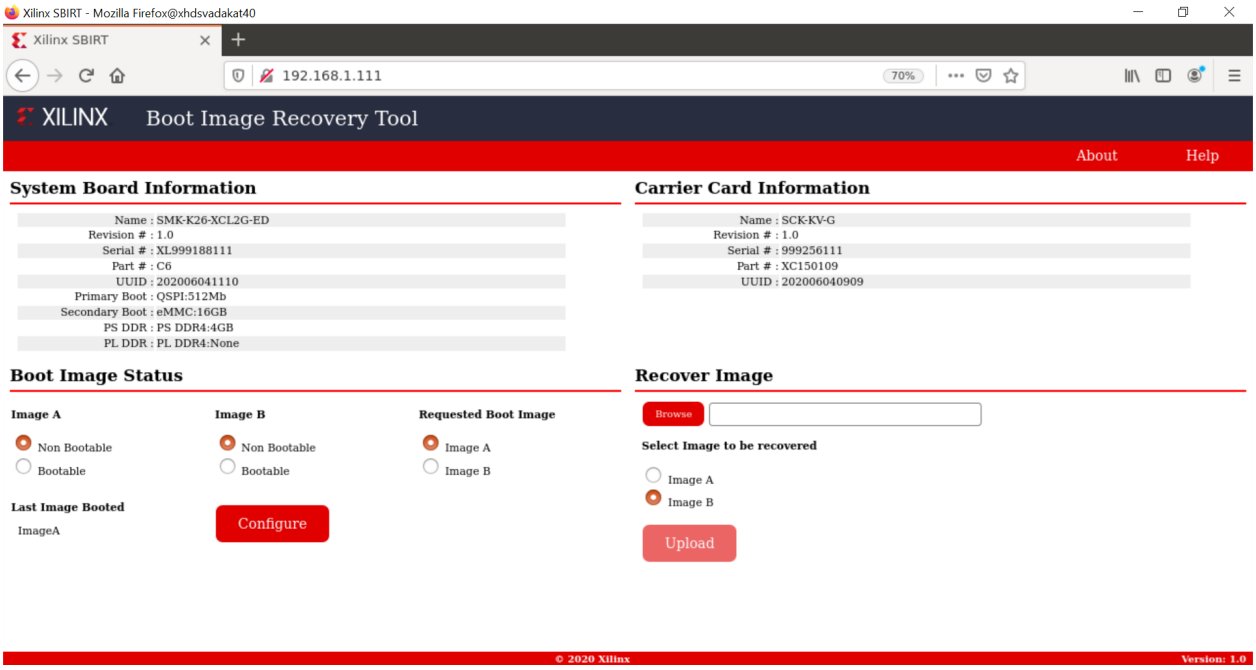

Figure 1: The recovery tool for the board, access from Firefox. We can see board information at the center, and the tools to upload the firmware at the bottom of the page.

2.3 Updating the boot firmware

From this "recovery" page, it is possible to upload the `.bin` file downloaded previously onto the board using the "Recover Image" section at the bottom right of the page.

The board can be re-booted afterwards.

3 Installing Linux

Withe the boot firmware being up-to-date, we can proceed to install a Linux distribution on our Kria board. The step needed to archive a full installation of Ubuntu 22.04 will be presented in this section.

3.1 Preparing and booting a Ubuntu 22.04 media

An official Ubuntu image exists and is provided by Xilinx, allowing the OS installation to be quick and straightforward. Ubuntu is a common and easy to use distribution. Furthermore, it allows to install ROS2 as a package, which is most convenient and will be done later in this guide.

Once the image has been downloaded at Canonical's page we can flash it onto the SD card, with the following instructions.

> DANGER: The next part involve the `dd` command writing on disks!!! As always with the dd command, thou have to be VERY careful on what arguments thou give. Selecting the wrong disk will result on the destruction of thy data !! If you are unsure of what to do, seek assistance !

With the image available on thy machine and a SD card visible as `/dev/sda` Once the SD card is flashed and put back in the board, the micro-USB cable can be connected from the PC to the board. It is then possible to connect to the board in serial with an appropriate tool, for example `picocom`, as in the following example (the serial port that "appeared" was the `/dev/ttyUSB1` in this case, and the 115200 bitrate is the default value for the board):

```
sudo picocom /dev/ttyUSB1 -b 115200
```

Once logged in, it is typically easier and more convenient to connect the board using SSH. When the board is connected to the network, it is possible to know it's IP address with the `ip` command; then it is possible to connect to the board with ssh, as follow (example, with the first command to be run on the board and the second one on the host PC, both without the first placeholder hostnames):

```
kria# ip addr

host# ssh ubuntu@192.168.4.11
```

3.2 Network and admin setups

This section presents a variety of extra convenience configurations that can be used when setting-up the Kria board.

3.2.1 Proxy and DNS

An issue that can occur when connecting the board to the internet is the conflicting situation with the university proxy. Indeed, as the network at Nanzan University requires to go through a proxy, some DNS errors appeared.

Firstly, it is possible to set a DNS IP address in `/etc/resolv.conf` by editing it and adding your favorite DNS, for example `nameserver 1.1.1.1` next to the other `nameserver` entry. The resolver can then be restarted.

```
sudo nano /etc/resolv.conf

sudo systemctl restart systemd-resolved
```

Secondly, it might become needed to setup the proxy for the school.

This can be done as follow, by exporting a https base proxy configuration containing you AXIA credentials (this is specific to Nanzan University IT system), then by consolidating the configuration for other types of connections in the `bashrc`:

```
export https_proxy="http://<AXIA_username>:\
       <AXIA_psw>@proxy.ic.nanzan-u.ac.jp:8080"

```

```
echo "export http_proxy=\""$https_proxy"\"" >> ~/.bashrc \
     echo "export https_proxy=\""$https_proxy"\"" >> ~/.bashrc \
     echo "export ftp_proxy=\""$https_proxy"\"" >> ~/.bashrc \
     echo "export no_proxy=\"localhost, 127.0.0.1,::1\"" \
     >> ~/.bashrc
```

Eventually the board can be rebooted in order for the setup to get applied cleanly.

3.2.2 root password

> WARNING: Depending on your use-case, the setup presented in this subsection can be a critical security breach as it remove the need for a root password to access the admin functions of the board's Linux. When in doubt, do not apply this configuration!!

If you board does not hold important data and is available to you only, for test or development, it might be convenient for the sudo tool to not ask for the password all the time. This change can be done by editing the sudoers file, and adding the parameter `NOPASSWD` at the sudo line:

```
sudo visudo

%sudo   ALL=(ALL:ALL) NOPASSWD: ALL
```

Again, this is merely a convenience setup for devices staying at you desk. If the board is meant to be used in any kind of production setup, a password should be set for making administration tasks.

With all of these settings, you should be able to update the software of your board without any issues:

```
sudo apt-get update
sudo apt-get dist-upgrade
sudo reboot now
```

3.2.3 Static IP address

A static IP can be set by writing the following configuration into your `netplan` configuration file.

The name of the files might vary:

```
sudo nano /etc/netplan/50-cloud-init.yaml
```

You can then set the wanted IP as follow. Note that a custom DNS was also set in that case.

```
network:
  renderer: NetworkManager
  version: 2
  ethernets:
    eth0:
      addresses:
        - 192.168.11.103/24
      routes:
        - to: default
          via: 192.168.11.1
      nameservers:
        addresses:
          - 8.8.8.8
          - 1.1.1.1
```

Finally, the change in settings can be applied as follow:

```
sudo netplan apply
```

3.2.4 Purging snap

As the desktop-specific software are not used at all in the case of our project, there are some packages that can be purges in order for the system to become more lightweight.

In particular, the main issue with Ubuntu systems is the forced integration of Snap packages. Here are the command to use in order to remove all of that. These steps take a lot of time and need to be executed in that specific order[1], but the system fan runs sensibly slower without all of this stuff:

```
sudo systemctl disable snapd.service
sudo systemctl disable snapd.socket
sudo systemctl disable snapd.seeded.service

sudo snap list #show installed package, remove then all:
sudo snap remove --purge firefox
sudo snap remove --purge gnome-3-38-2004
sudo snap remove --purge gnome-42-2204
sudo snap remove --purge gtk-common-themes
sudo snap remove --purge snapd-desktop-integration
sudo snap remove --purge snap-store
sudo snap remove --purge bare
sudo snap remove --purge core20
sudo snap remove --purge core22
sudo snap remove --purge snapd
sudo snap list # check that everything is uninstalled

sudo rm -rf /var/cache/snapd/
sudo rm -rf ~/snap
sudo apt autoremove --purge snapd

systemctl list-units | grep snapd
```

3.2.5 Other unused heavy packages

Some other pieces of software can safely be removed since the desktop is not to be used:

```
sudo apt-get autoremove --purge yaru-theme-icon \
fonts-noto-cjk yaru-theme-gtk vim-runtime \
ubuntu-wallpapers-jammy humanity-icon-theme

sudo apt-get autoclean
sudo reboot now
```

4 Enabling `remoteproc`

One of the advantage of this Kria board, as cited previously, is the presence of multiple types of core (APU, MCU, FPGA) on the same chip.

[1] The `snap` package depends on each other. Thus dependencies cannot be remove before the package(s) that depends on them.

The part in focus in this guide is the usage of both the APU, running a Linux distribution and ROS2; and the MCU, running FreeRTOS and micro-ROS.

The communication between both side is meant to be done using shared memory, but some extra setup is required in order to be running the real-time firmware, in particular for deploying micro-ROS on it.

As a first step in that direction, this section of the report will present how to setup and use as an example firmware that utilizes the `remoteproc` device in Linux in order to access shared memory and communicate with the real-time firmware using the RPMsg system.

5 Loading a real-time firmware

6 Building micro-ROS as a static library

7 Building a real-time firmware

7.1 Setting up Vitis IDE

8 Adding micro-ROS to a firmware project

9 Loading a real-time firmware

10 Running a ROS2 node

10.1 On the host Linux

10.2 In a container

11 micro-ROS agent

A DTO patch

This file is available in this repositroy: system.patch

```
diff -u --label /ssh\:kria\:/home/ubuntu/system_original.dts --label /ssh\:kria\:/home/ubuntu/system.dts /tmp/tramp.KJbEbz.dts /tmp/tramp.zHBG7v.dts
--- /ssh:kria:/home/ubuntu/system_original.dts
+++ /ssh:kria:/home/ubuntu/system.dts
@@ -138,7 +138,7 @@

         firmware {

-                zynqmp-firmware {
+                zynqmp_firmware: zynqmp-firmware {
                         compatible = "xlnx,zynqmp-firmware";
                         #power-domain-cells = <0x01>;
                         method = "smc";
@@ -719,7 +719,7 @@
                         phandle = <0x44>;
                 };

-                interrupt-controller@f9010000 {
+                gic: interrupt-controller@f9010000 {
                         compatible = "arm,gic-400";
                         #interrupt-cells = <0x03>;
                         reg = <0x00 0xf9010000 0x00 0x10000 0x00 0xf9020000 0x00 0x20000 0x00 0xf9040000 0x00 0x20000 0x00 0xf9060000 0x00 0x20000>;
@@ -1536,7 +1536,7 @@
                         pinctrl-names = "default";
                         u-boot,dm-pre-reloc;
                         compatible = "xlnx,zynqmp-uart\0cdns,uart-r1p12";
-                        status = "okay";
+                        status = "disabled";
                         interrupt-parent = <0x04>;
                         interrupts = <0x00 0x16 0x04>;
                         reg = <0x00 0xff010000 0x00 0x1000>;
@@ -1909,6 +1909,84 @@
                 pwms = <0x1b 0x02 0x9c40 0x00>;
         };

+       reserved-memory {
+           #address-cells = <2>;
+           #size-cells = <2>;
+           ranges;
+           rpu0vdev0vring0: rpu0vdev0vring0@3ed40000 {
+               no-map;
+               reg = <0x0 0x3ed40000 0x0 0x4000>;
+           };
+           rpu0vdev0vring1: rpu0vdev0vring1@3ed44000 {
+               no-map;
+               reg = <0x0 0x3ed44000 0x0 0x4000>;
+           };
+           rpu0vdev0buffer: rpu0vdev0buffer@3ed48000 {
+               no-map;
+               reg = <0x0 0x3ed48000 0x0 0x100000>;
```

```
+            };
+            rproc_0_reserved: rproc_0_reserved@3ed00000 {
+                no-map;
+                reg = <0x0 0x3ed00000 0x0 0x40000>;
+            };
+        };
+        tcm_0a: tcm_0a@ffe00000 {
+            no-map;
+            reg = <0x0 0xffe00000 0x0 0x10000>;
+            status = "okay";
+            compatible = "mmio-sram";
+            power-domain = <&zynqmp_firmware 15>;
+        };
+        tcm_0b: tcm_0b@ffe20000 {
+            no-map;
+            reg = <0x0 0xffe20000 0x0 0x10000>;
+            status = "okay";
+            compatible = "mmio-sram";
+            power-domain = <&zynqmp_firmware 16>;
+        };
+        rf5ss@ff9a0000 {
+            compatible = "xlnx,zynqmp-r5-remoteproc";
+            xlnx,cluster-mode = <1>;
+            ranges;
+            reg = <0x0 0xFF9A0000 0x0 0x10000>;
+            #address-cells = <0x2>;
+            #size-cells = <0x2>;
+            r5f_0 {
+                compatible = "xilinx,r5f";
+                #address-cells = <2>;
+                #size-cells = <2>;
+                ranges;
+                sram = <&tcm_0a &tcm_0b>;
+                memory-region = <&rproc_0_reserved>, <&rpu0vdev0buffer>,
 ↪ <&rpu0vdev0vring0>, <&rpu0vdev0vring1>;
+                power-domain = <&zynqmp_firmware 7>;
+                mboxes = <&ipi_mailbox_rpu0 0>, <&ipi_mailbox_rpu0 1>;
+                mbox-names = "tx", "rx";
+            };
+        };
+        zynqmp_ipi1 {
+            compatible = "xlnx,zynqmp-ipi-mailbox";
+            interrupt-parent = <&gic>;
+            interrupts = <0 29 4>;
+            xlnx,ipi-id = <7>;
+            #address-cells = <1>;
+            #size-cells = <1>;
+            ranges;
+            /* APU<->RPU0 IPI mailbox controller */
+            ipi_mailbox_rpu0: mailbox@ff990600 {
+                reg = <0xff990600 0x20>,
+                      <0xff990620 0x20>,
+                      <0xff9900c0 0x20>,
+                      <0xff9900e0 0x20>;
```

```
+                reg-names = "local_request_region",
+                        "local_response_region",
+                        "remote_request_region",
+                        "remote_response_region";
+                #mbox-cells = <1>;
+                xlnx,ipi-id = <1>;
+            };
+        };
+
+
          __symbols__ {
                  cpu0 = "/cpus/cpu@0";
                  cpu1 = "/cpus/cpu@1";

Diff finished.  Wed May 24 10:15:49 2023
```

B Custom toolchain CMake settings

This file is available in this repositroy: custom r5f toolchain.cmake

```
set(CMAKE_SYSTEM_NAME Generic)
set(CMAKE_CROSSCOMPILING 1)
set(CMAKE_TRY_COMPILE_TARGET_TYPE STATIC_LIBRARY)
set(CMAKE_INSTALL_LIBDIR /usr/)
set(PLATFORM_NAME "LwIP")


set(ARCH_CPU_FLAGS "-mcpu=cortex-r5 -mthumb -mfpu=vfpv3-d16 -mfloat-abi=hard -DARMR5 -O0
↪  -Wall -fdata-sections -ffunction-sections -fno-tree-loop-distribute-patterns
↪  -Wno-unused-parameter -Wno-unused-value -Wno-unused-variable -Wno-unused-function
↪  -Wno-unused-but-set-variable" CACHE STRING "" FORCE)
set(ARCH_OPT_FLAGS "")

set(CMAKE_C_COMPILER arm-none-eabi-gcc)
set(CMAKE_CXX_COMPILER arm-none-eabi-g++)

set(CMAKE_C_FLAGS_INIT "-std=c11 ${ARCH_CPU_FLAGS} ${ARCH_OPT_FLAGS}
↪  -DCLOCK_MONOTONIC=0" CACHE STRING "" FORCE)
set(CMAKE_CXX_FLAGS_INIT "-std=c++14 ${ARCH_CPU_FLAGS} ${ARCH_OPT_FLAGS}
↪  -DCLOCK_MONOTONIC=0" CACHE STRING "" FORCE)



set(__BIG_ENDIAN__ 0)
```

C Custom Colcon meta settings

This file is available in this repositroy: custom r5f colcon.meta

```
{
    "names": {
        "tracetools": {
            "cmake-args": [
                "-DTRACETOOLS_DISABLED=ON",
                "-DTRACETOOLS_STATUS_CHECKING_TOOL=OFF"
            ]
        },
        "rosidl_typesupport": {
            "cmake-args": [
               "-DROSIDL_TYPESUPPORT_SINGLE_TYPESUPPORT=ON"
            ]
        },
        "rcl": {
            "cmake-args": [
                "-DBUILD_TESTING=OFF",
                "-DRCL_COMMAND_LINE_ENABLED=OFF",
                "-DRCL_LOGGING_ENABLED=OFF"
            ]
        },
        "rcutils": {
            "cmake-args": [
                "-DENABLE_TESTING=OFF",
                "-DRCUTILS_NO_FILESYSTEM=ON",
                "-DRCUTILS_NO_THREAD_SUPPORT=ON",
                "-DRCUTILS_NO_64_ATOMIC=ON",
                "-DRCUTILS_AVOID_DYNAMIC_ALLOCATION=ON"
            ]
        },
        "microxrcedds_client": {
            "cmake-args": [
                "-DUCLIENT_PIC=OFF",
                "-DUCLIENT_PROFILE_UDP=OFF",
                "-DUCLIENT_PROFILE_TCP=OFF",
                "-DUCLIENT_PROFILE_DISCOVERY=OFF",
                "-DUCLIENT_PROFILE_SERIAL=OFF",
                "-UCLIENT_PROFILE_STREAM_FRAMING=ON",
                "-DUCLIENT_PROFILE_CUSTOM_TRANSPORT=ON"
            ]
        },
        "rmw_microxrcedds": {
            "cmake-args": [
                "-DRMW_UXRCE_MAX_NODES=1",
                "-DRMW_UXRCE_MAX_PUBLISHERS=5",
                "-DRMW_UXRCE_MAX_SUBSCRIPTIONS=5",
                "-DRMW_UXRCE_MAX_SERVICES=1",
                "-DRMW_UXRCE_MAX_CLIENTS=1",
                "-DRMW_UXRCE_MAX_HISTORY=4",
                "-DRMW_UXRCE_TRANSPORT=custom"
            ]
        }
```

```
52      }
53  }
```

D Firmware time functions

D.1 main

This file is available in this repositroy: clock.c

```c
#include "microros.h"


int _gettimeofday( struct timeval *tv, void *tzvp )
{
        XTime t = 0;
    XTime_GetTime(&t);  //get uptime in nanoseconds
    tv->tv_sec = t / 1000000000;  // convert to seconds
    tv->tv_usec = ( t % 1000000000 ) / 1000;  // get remaining microseconds
    return 0;  // return non-zero for error
} // end _gettimeofday()


void UTILS_NanosecondsToTimespec( int64_t llSource,
                                  struct timespec * const pxDestination )
{
    long lCarrySec = 0;

    /* Convert to timespec. */
    pxDestination->tv_sec = ( time_t ) ( llSource / NANOSECONDS_PER_SECOND );
    pxDestination->tv_nsec = ( long ) ( llSource % NANOSECONDS_PER_SECOND );

    /* Subtract from tv_sec if tv_nsec < 0. */
    if( pxDestination->tv_nsec < 0L )
    {
        /* Compute the number of seconds to carry. */
        lCarrySec = ( pxDestination->tv_nsec / ( long ) NANOSECONDS_PER_SECOND ) + 1L;

        pxDestination->tv_sec -= ( time_t ) ( lCarrySec );
        pxDestination->tv_nsec += lCarrySec * ( long ) NANOSECONDS_PER_SECOND;
    }
}

int clock_gettime( clockid_t clock_id,
                   struct timespec * tp )
{
        TimeOut_t xCurrentTime = { 0 };

    /* Intermediate variable used to convert TimeOut_t to struct timespec.
     * Also used to detect overflow issues. It must be unsigned because the
     * behavior of signed integer overflow is undefined. */
    uint64_t ullTickCount = 0ULL;

    /* Silence warnings about unused parameters. */
    ( void ) clock_id;

    /* Get the current tick count and overflow count. vTaskSetTimeOutState()
     * is used to get these values because they are both static in tasks.c. */
    vTaskSetTimeOutState( &xCurrentTime );
```

```
50
51      /* Adjust the tick count for the number of times a TickType_t has overflowed.
52       * portMAX_DELAY should be the maximum value of a TickType_t. */
53      ullTickCount = ( uint64_t ) ( xCurrentTime.xOverflowCount ) << ( sizeof( TickType_t
    ↪   ) * 8 );
54
55      /* Add the current tick count. */
56      ullTickCount += xCurrentTime.xTimeOnEntering;
57
58      /* Convert ullTickCount to timespec. */
59      UTILS_NanosecondsToTimespec( ( int64_t ) ullTickCount * NANOSECONDS_PER_TICK, tp );
60
61      return 0;
62  }
```

D.2 header file

```
1   /**< Microseconds per second. */
2   #define MICROSECONDS_PER_SECOND    ( 1000000LL )
3   /**< Nanoseconds per second. */
4   #define NANOSECONDS_PER_SECOND     ( 1000000000LL )
5   /**< Nanoseconds per FreeRTOS tick. */
6   #define NANOSECONDS_PER_TICK       ( NANOSECONDS_PER_SECOND / configTICK_RATE_HZ )
```

E Firmware memory allocation functions

E.1 main

This file is available in this repositroy: allocators.c

```
#include "allocators.h"

//int absoluteUsedMemory = 0;
//int usedMemory = 0;

void * __freertos_allocate(size_t size, void * state){
  (void) state;
  LPRINTF("-- Alloc %d (prev: %d B)\r\n",size, xPortGetFreeHeapSize());
//  absoluteUsedMemory += size;
//  usedMemory += size;

  LPRINTF("Return for the allocate function w parameter size = %d\r\n", size);

  return pvPortMalloc(size);
}

void __freertos_deallocate(void * pointer, void * state){
  (void) state;
  LPRINTF("-- Free 0x%x (prev: %d B)\r\n", pointer, xPortGetFreeHeapSize());
  if (NULL != pointer)
  {
//          LPRINTF("Pointer is not null.\r\n");
//          usedMemory -= getBlockSize(pointer);
//          LPRINTF("usedMemory var updated: %d\r\n", usedMemory);
          vPortFree(pointer);
  }
  else
  {
          LPERROR("Trying to deallocate a null pointed. Doing nothing.\r\n");
  }
}

void * __freertos_reallocate(void * pointer, size_t size, void * state){
  (void) state;
  LPRINTF("-- Realloc 0x%x -> %d (prev: %d B)\r\n", pointer, size,
  ↪  xPortGetFreeHeapSize());
//  absoluteUsedMemory += size;
//  usedMemory += size;
  if (NULL == pointer)
  {
    return __freertos_allocate(size, state);
  }
  else
  {
//    usedMemory -= getBlockSize(pointer);
//
//    return pvPortRealloc(pointer,size);

          __freertos_deallocate(pointer, state);
```

```
49            return __freertos_allocate(size, state);
50
51    }
52  }
53
54  void * __freertos_zero_allocate(size_t number_of_elements, size_t size_of_element, void
    ↪  * state){
55    (void) state;
56    LPRINTF("-- Calloc %d x %d = %d -> (prev: %d
    ↪  B)\r\n",number_of_elements,size_of_element, number_of_elements*size_of_element,
    ↪  xPortGetFreeHeapSize());
57  //  absoluteUsedMemory += number_of_elements*size_of_element;
58  //  usedMemory += number_of_elements*size_of_element;
59
60    return pvPortCalloc(number_of_elements,size_of_element);
61  }
```

E.2 header file

```
1  #ifndef _ALLOCATORS_H_
2  #define _ALLOCATORS_H_
3
4  #include "microros.h"
5
6  extern int absoluteUsedMemory;
7  extern int usedMemory;
8
9
10 void * __freertos_allocate(size_t size, void * state);
11 void __freertos_deallocate(void * pointer, void * state);
12 void * __freertos_reallocate(void * pointer, size_t size, void * state);
13 void * __freertos_zero_allocate(size_t number_of_elements,
14 size_t size_of_element, void * state);
15
16 #endif // _ALLOCATORS_H_
```