

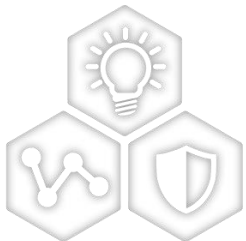
SAM9X60 ARM926EJ-S

Microprocessor & Embedded Linux

LNx1001-Buildroot v1.00



A Leading Provider of Smart, Connected and Secure Embedded Control Solutions



SMART | CONNECTED | SECURE

Kevin Lu
CAE Taiwan
Microchip Technology Inc.
September 11, 2025

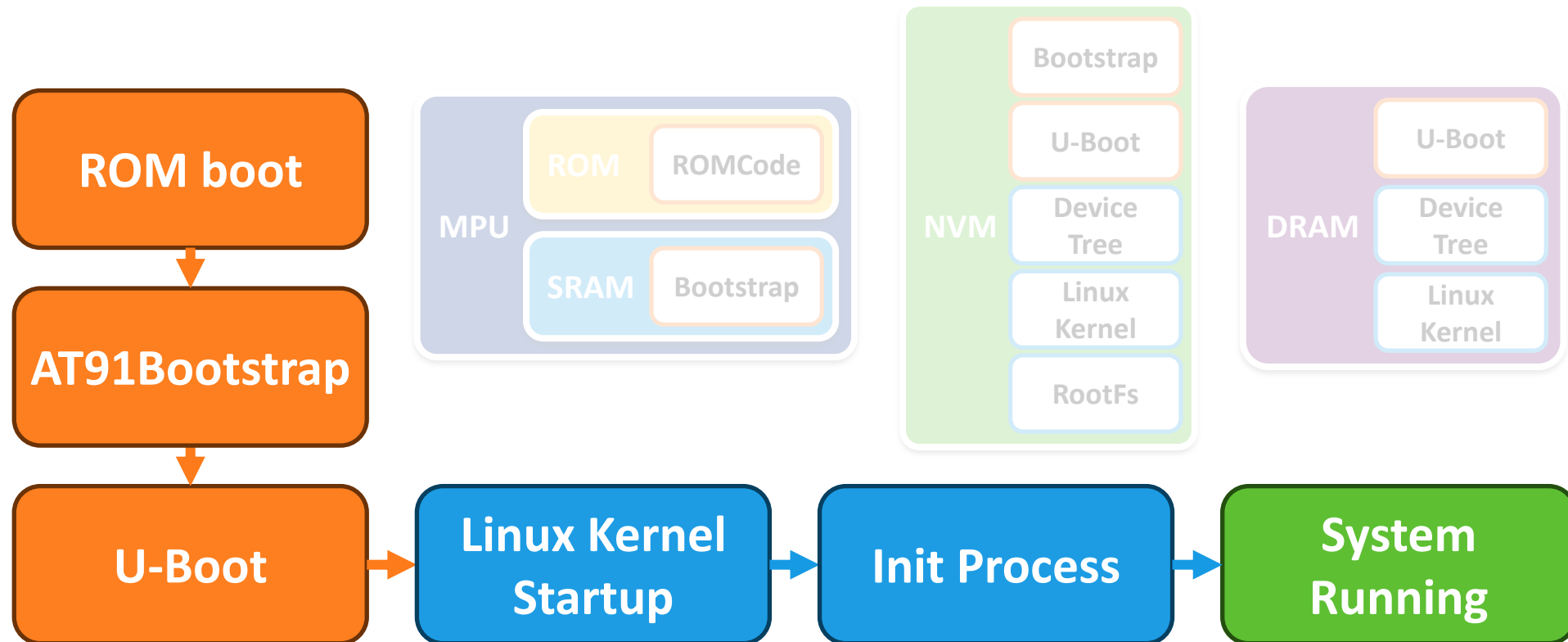
Course Outline

- Embedded Linux Startup Process
- Components of Embedded Linux System for MPU
 - AT91Bootstrap
 - U-Boot
 - Device-Tree
 - Linux Kernel
 - Root Filesystem
- Build System and Buildroot
 - Lab1 – Prepare Buildroot and First Build
- Modify the Configuration of Buildroot for Hobby Kit
 - Lab2 – Modify the Configuration of Buildroot and Rebuild
- Customize External Tree Project for Buildroot
 - Lab3 - Modify br2-external-tree and Rebuild the Buildroot
- Use MPIO to Test Peripherals
 - Lab4 – Use MPIO to Control Peripherals

Embedded Linux Startup Process

Embedded Linux Startup Process

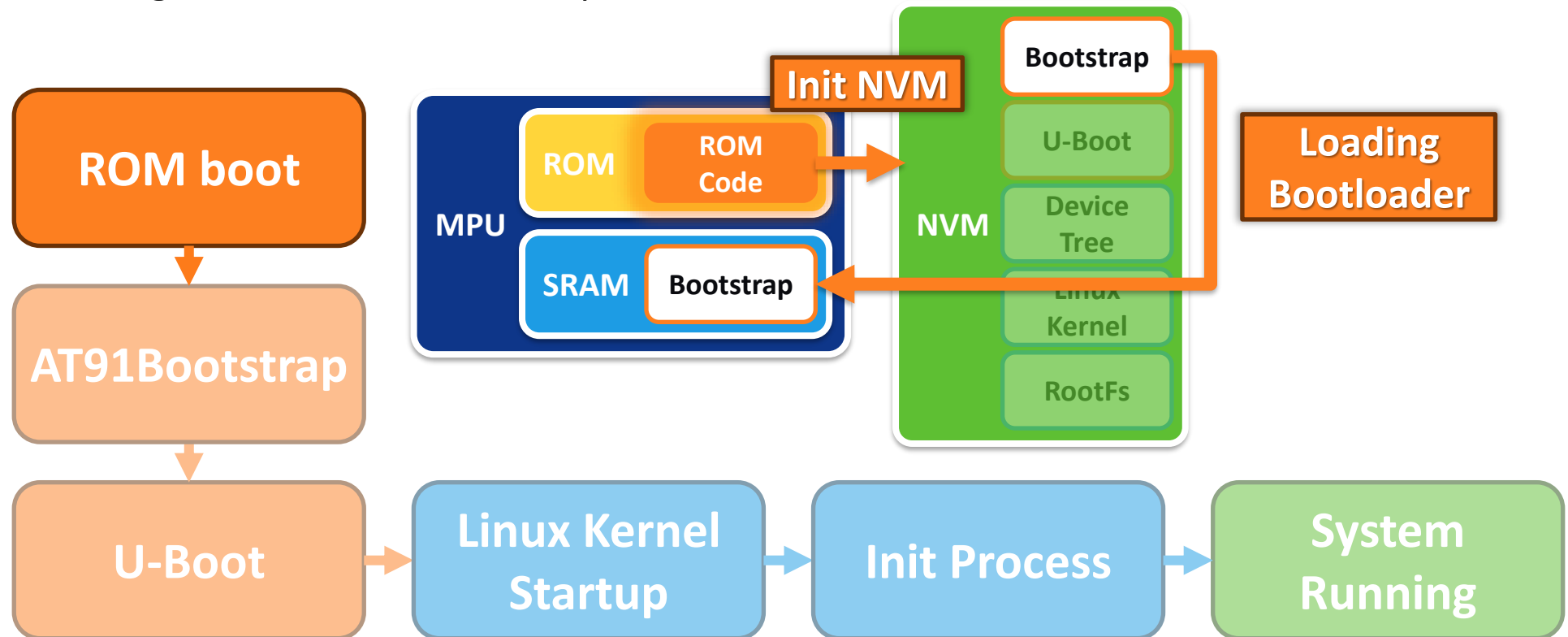
- Embedded Linux is a powerful operating system. When running on an MPU, the boot process is a complex and orderly process.



Embedded Linux Startup Process

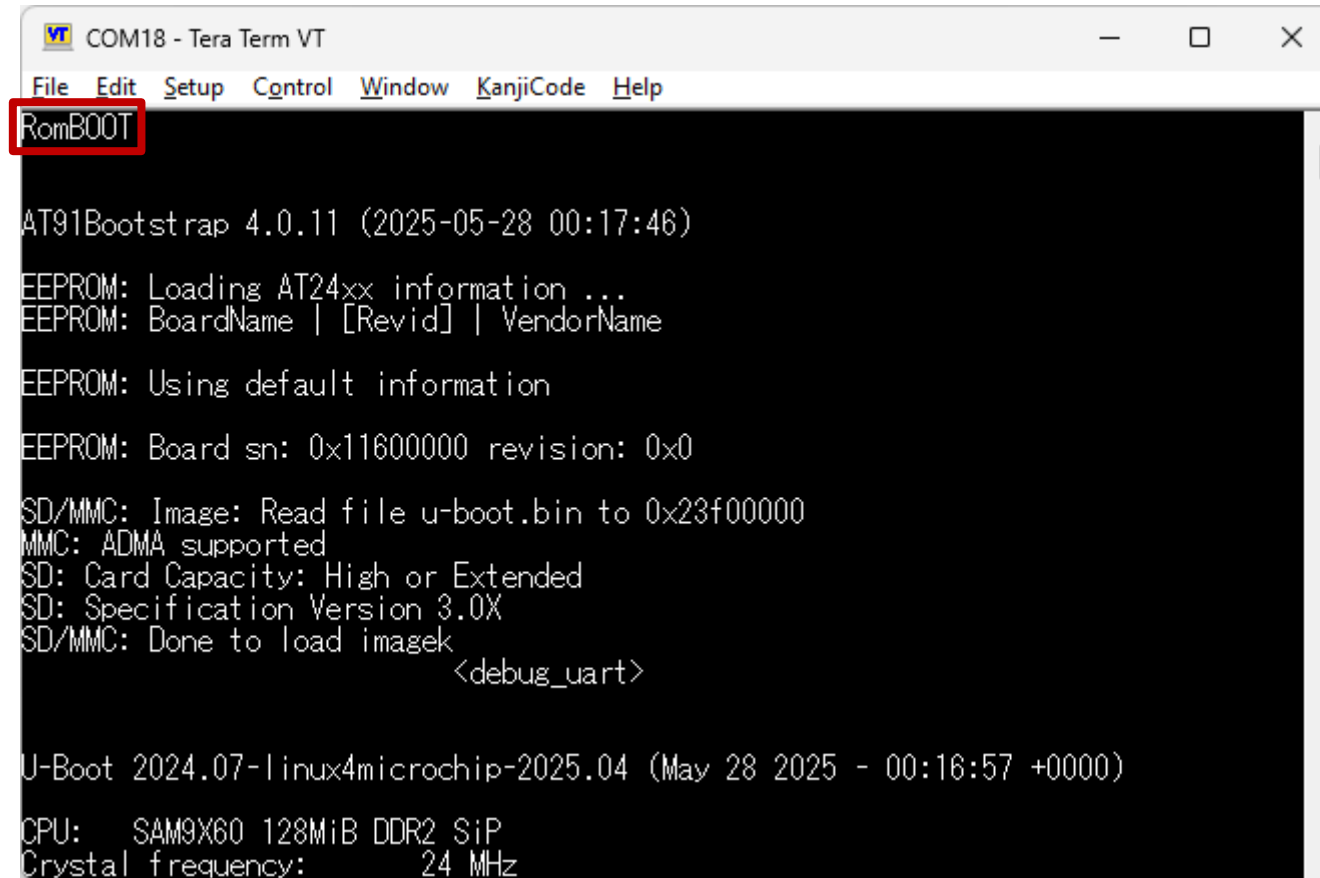
1. Power-On and Self Test (ROM boot)

- When the system powers on, the ROM-Code starts executing.
- Check hardware components.
- The ROM-Code loads the Bootloader from an external storage device. (e.g., SD card, NAND Flash).
- Transferring control to the Bootstrap.



Embedded Linux Startup Process

Power-On and Self Test (ROM boot)



```
COM18 - Tera Term VT
File Edit Setup Control Window KanjiCode Help
RomBOOT

AT91Bootstrap 4.0.11 (2025-05-28 00:17:46)

EEPROM: Loading AT24xx information ...
EEPROM: BoardName | [Revid] | VendorName

EEPROM: Using default information

EEPROM: Board sn: 0x11600000 revision: 0x0

SD/MMC: Image: Read file u-boot.bin to 0x23f00000
MMC: ADMA supported
SD: Card Capacity: High or Extended
SD: Specification Version 3.0X
SD/MMC: Done to load imagek
                                <debug_uart>

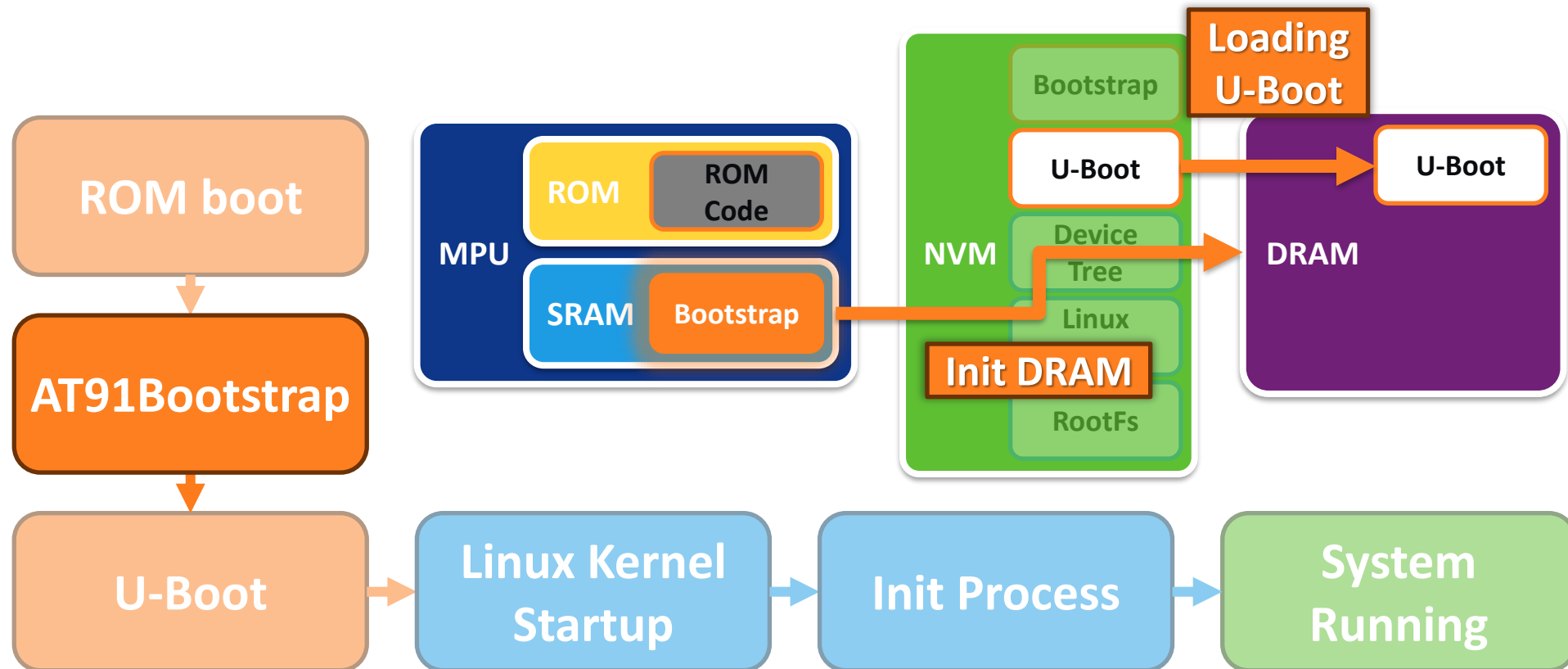
U-Boot 2024.07-linux4microchip-2025.04 (May 28 2025 - 00:16:57 +0000)

CPU:   SAM9X60 128MiB DDR2 SiP
Crystal frequency:    24 MHz
```

Embedded Linux Startup Process

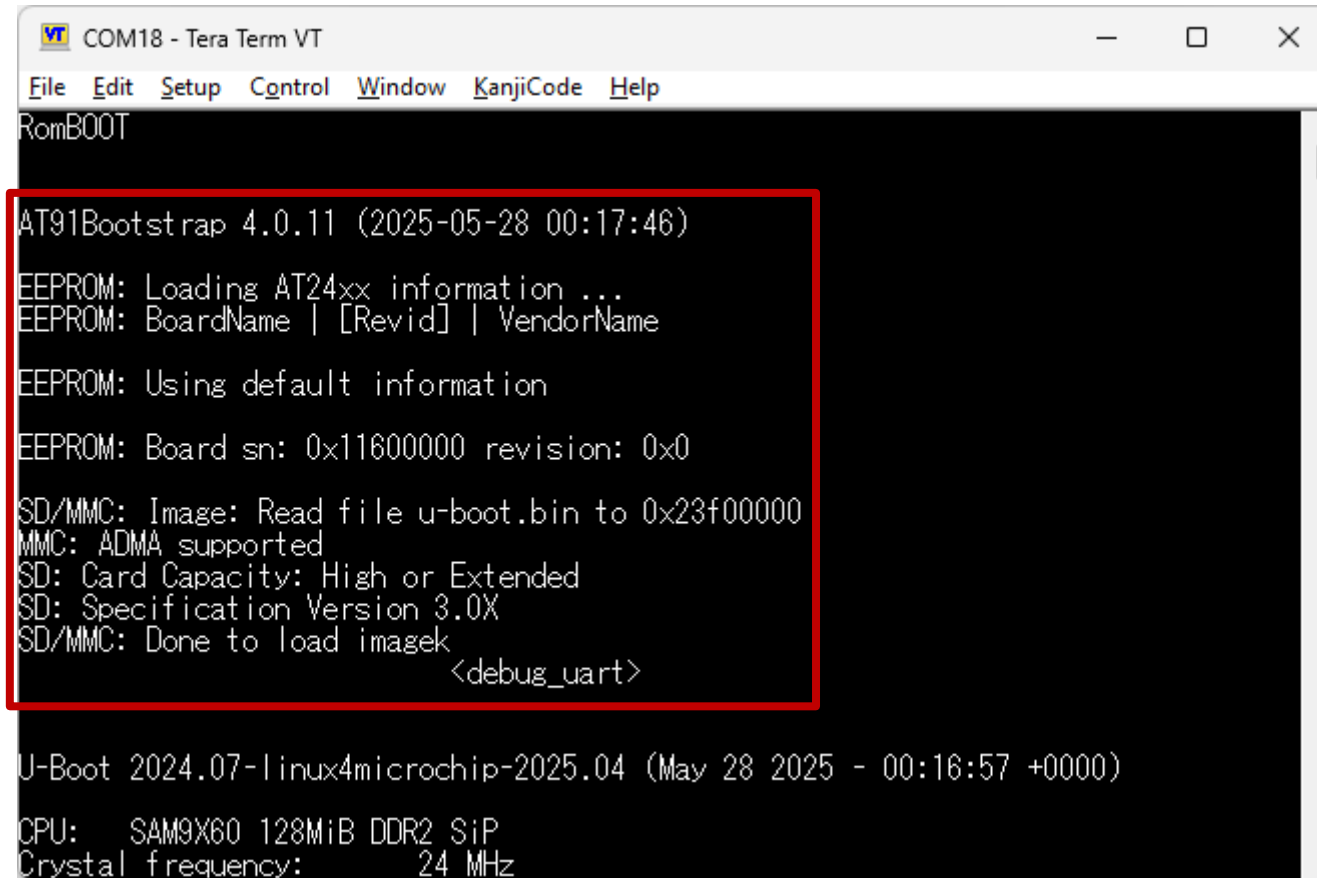
2. Second-stage Bootloader Startup (bootloader for Microchip SoC, AT91Bootstrap)

- Initializing the CPU, Memory, Clock, and other basic hardware.
- The Bootstrap loads the U-Boot from an external storage device to DRAM.
- Transferring control to the U-Boot.



Embedded Linux Startup Process

Second-stage Bootloader Startup (AT91Bootstrap)

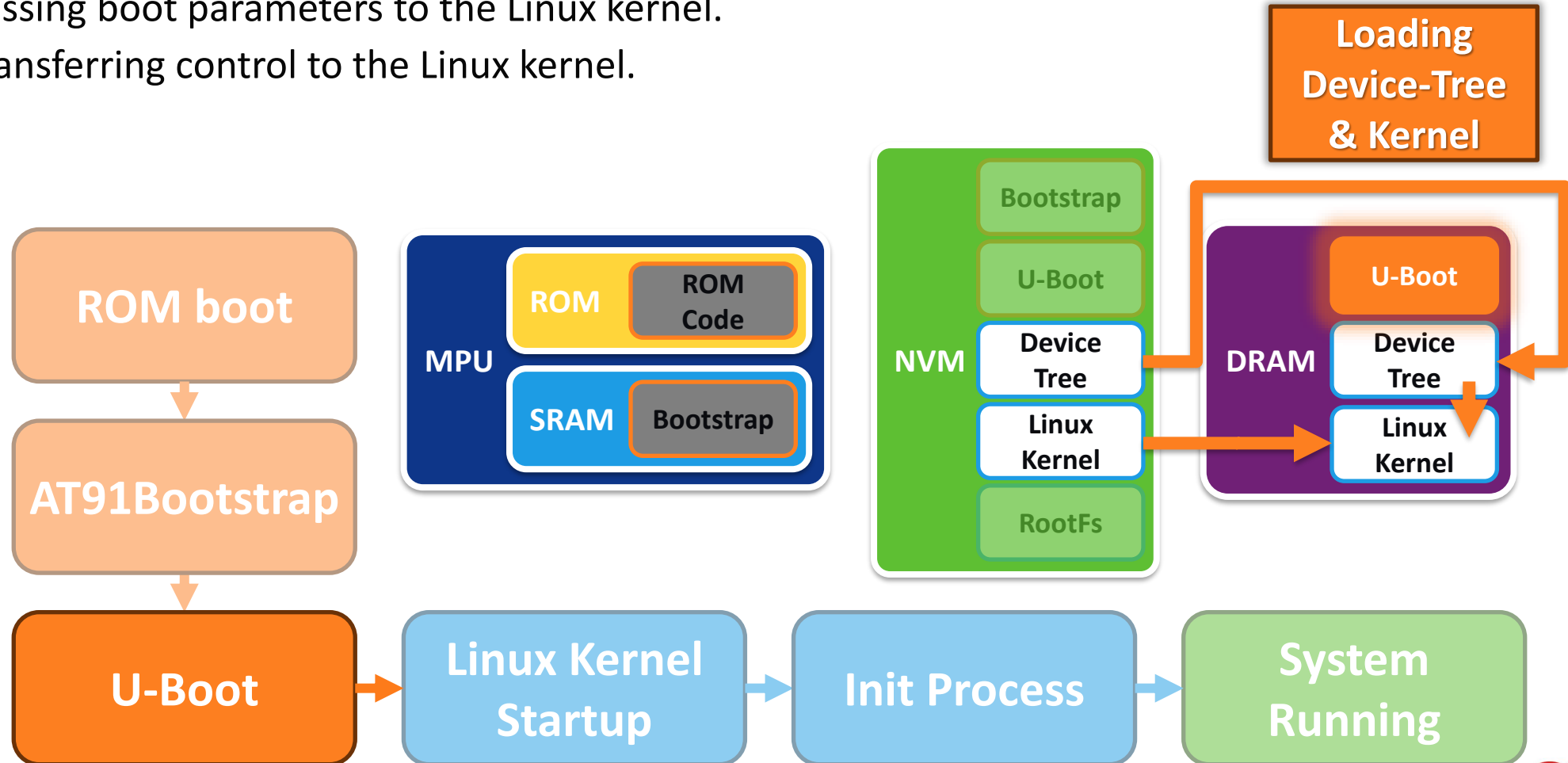


```
COM18 - Tera Term VT
File Edit Setup Control Window KanjiCode Help
RomBOOT
AT91Bootstrap 4.0.11 (2025-05-28 00:17:46)
EEPROM: Loading AT24xx information ...
EEPROM: BoardName | [Revid] | VendorName
EEPROM: Using default information
EEPROM: Board sn: 0x11600000 revision: 0x0
SD/MMC: Image: Read file u-boot.bin to 0x23f00000
MMC: ADMA supported
SD: Card Capacity: High or Extended
SD: Specification Version 3.0X
SD/MMC: Done to load imagek
<debug_uart>
U-Boot 2024.07-linux4microchip-2025.04 (May 28 2025 - 00:16:57 +0000)
CPU: SAM9X60 128MiB DDR2 SiP
Crystal frequency: 24 MHz
```

Embedded Linux Startup Process

3. Third-stage Bootloader Startup (U-Boot)

- Loading the Device-Tree and Linux kernel image into memory.
- Passing boot parameters to the Linux kernel.
- Transferring control to the Linux kernel.



Embedded Linux Startup Process

Third-stage Bootloader Startup (U-Boot)

```
COM18 - Tera Term VT
File Edit Setup Control Window KanjiCode Help
U-Boot 2024.07-linux4microchip-2025.04 (May 28 2025 - 00:16:57 +0000)
CPU: SAM9X60 128MiB DDR2 SiP
Crystal frequency: 24 MHz
CPU clock : 600 MHz
Master clock : 200 MHz

DRAM: 128 MiB
Core: 152 devices, 25 uclasses, devicetree: separate
NAND: atmel-nand-controller nand-controller: NAND scan failed: -19
Failed to probe nand driver (err = -19)
0 MiB
MMC: sdhci-host@80000000: 0, sdhci-host@90000000: 1
Loading Environment from FAT... OK
In: serial
Out: serial
Err: serial
PDA TM5000 detected
Net:
Error: ethernet@f802c000 No valid MAC address found.
No ethernet found.

Hit any key to stop autoboot: 0
5631560 bytes read in 250 ms (21.5 MiB/s)
```

```
## Loading kernel from FIT Image at 21000000 ...
Using 'kernel_dtb' configuration
Trying 'kernel' kernel subimage
Description: Linux4SAM Linux kernel
Type: Kernel Image
Compression: uncompressed
Data Start: 0x210000dc
Data Size: 5585416 Bytes = 5.3 MiB
Architecture: ARM
OS: Linux
Load Address: 0x22000000
Entry Point: 0x22000000
Hash algo: crc32
Hash value: df013e1d
Hash algo: sha1
Hash value: 757c00eaa62ea827bd25b200f001438df5590203
Verifying Hash Integrity ... crc32+ sha1+ OK

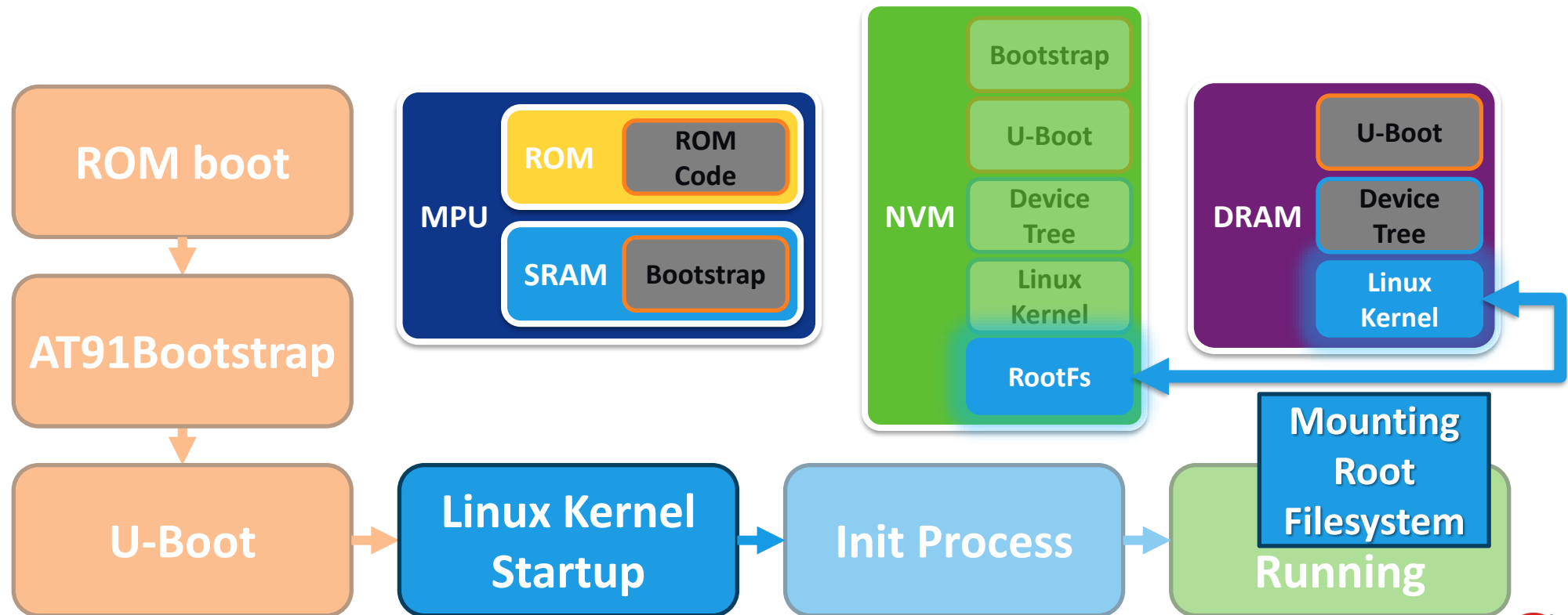
## Loading fdt from FIT Image at 21000000 ...
Using 'kernel_dtb' configuration
Trying 'base_fdt' fdt subimage
Description: SAM9X60-Curiosity Flattened Device Tree Blob
Type: Flat Device Tree
Compression: uncompressed
Data Start: 0x21553c24
Data Size: 37072 Bytes = 36.2 KiB
Architecture: ARM
Load Address: 0x23000000
Hash algo: crc32
Hash value: a2a04ced
Hash algo: sha1
Hash value: e415c99905145611ee0e5dd9c4b14ad4b189b0de
Verifying Hash Integrity ... crc32+ sha1+ OK
Loading fdt from 0x21553c24 to 0x23000000
Booting using the fdt blob at 0x23000000
Working FDT set to 23000000
Loading Kernel Image to 22000000
Loading Device Tree to 27ef0000, end 27efc0cf ... OK
Working FDT set to 27ef0000

Starting kernel ...
```

Embedded Linux Startup Process

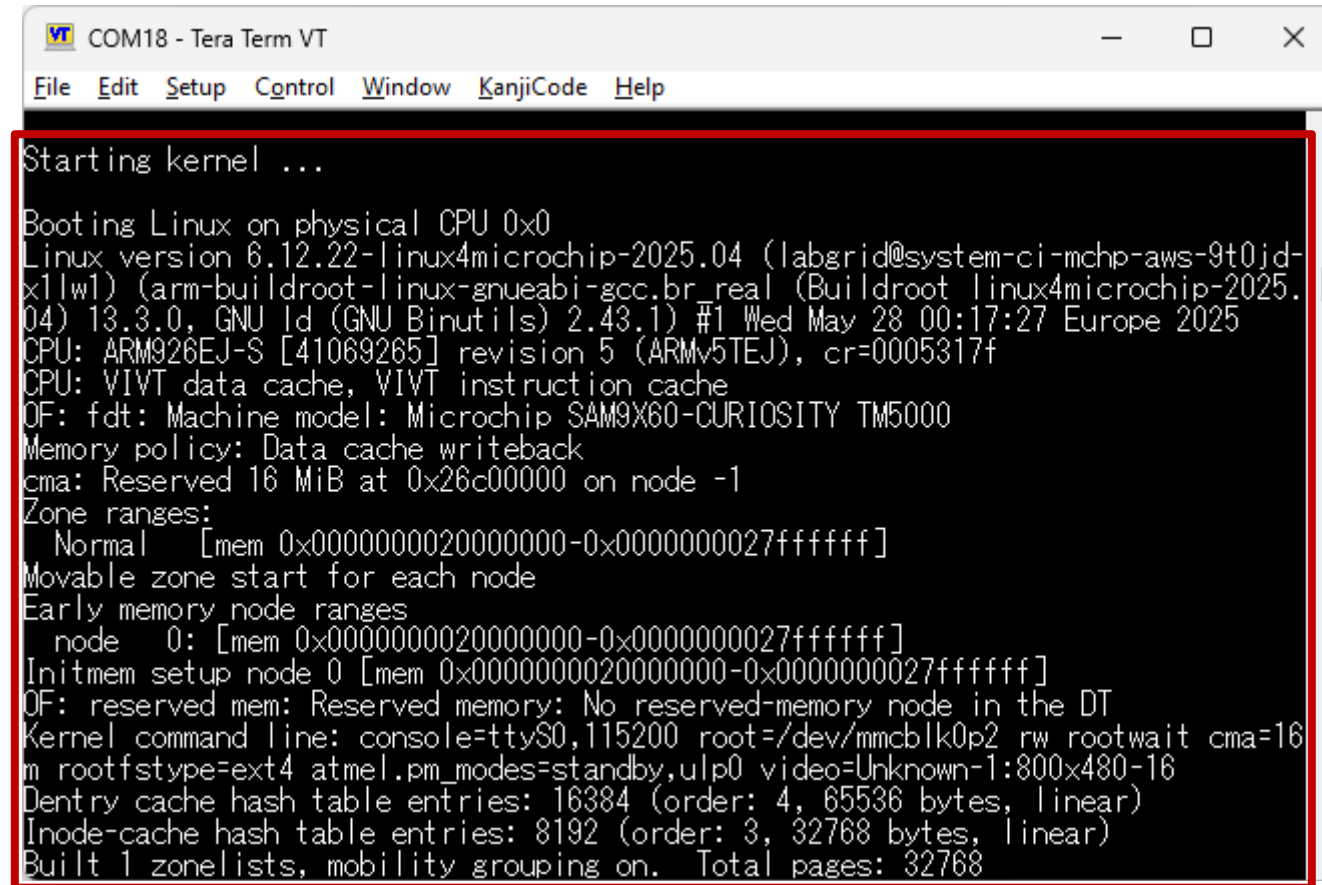
4. Linux Kernel Startup

- Initializing Kernel functionalities.
(e.g., memory management, process management, and device drivers).
- Mounting Root Filesystem.
- Starting init Process.



Embedded Linux Startup Process

Linux Kernel Startup

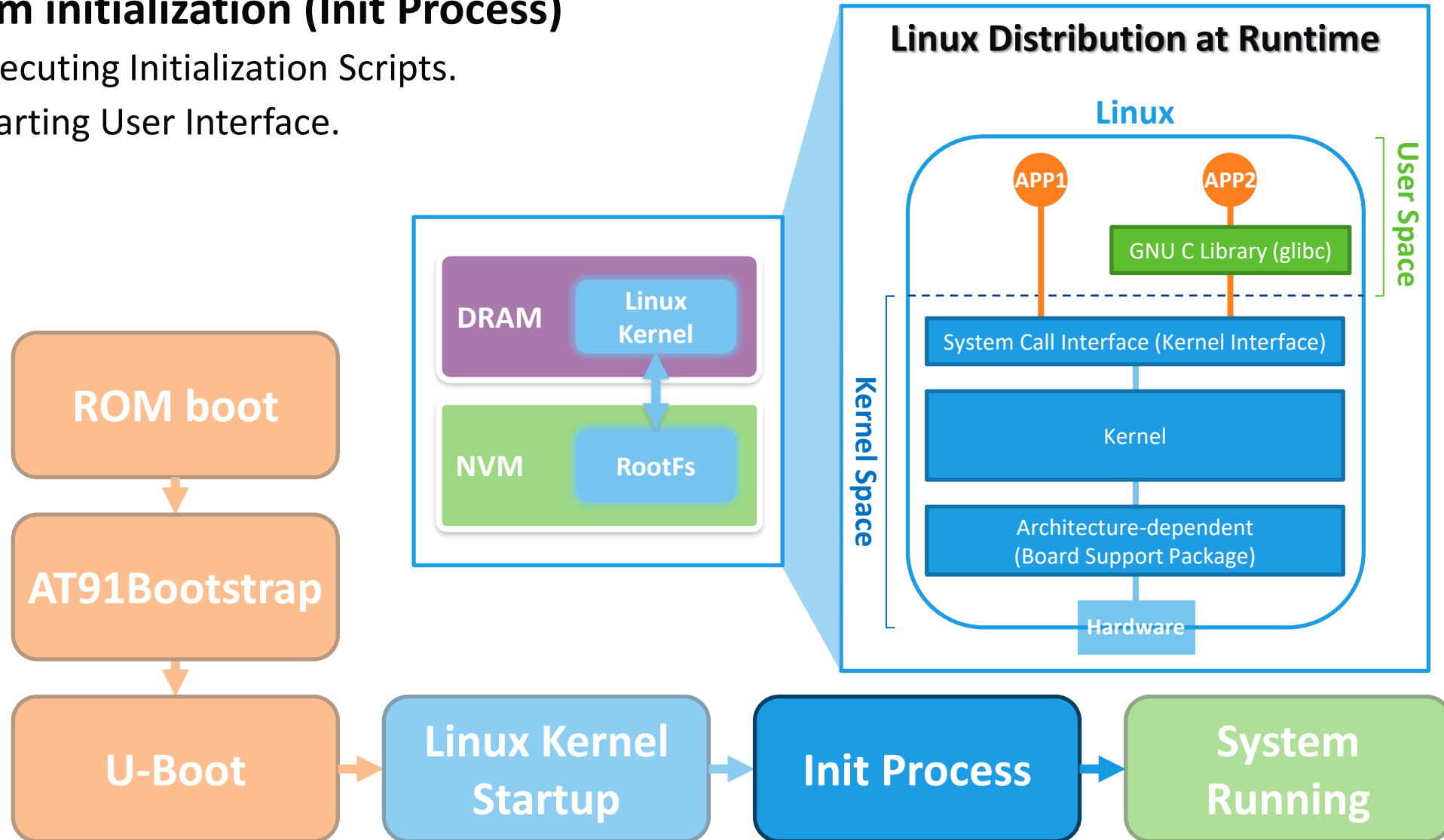
A screenshot of a terminal window titled "COM18 - Tera Term VT". The window has a menu bar with "File", "Edit", "Setup", "Control", "Window", "KanjiCode", and "Help". The terminal output shows the Linux kernel startup process. The text is as follows:

```
Starting kernel ...  
Booting Linux on physical CPU 0x0  
Linux version 6.12.22-linux4microchip-2025.04 (labgrid@system-ci-mchp-aws-9t0jd-  
x1lw1) (arm-buildroot-linux-gnueabi-gcc.br_real (Buildroot linux4microchip-2025.  
04) 13.3.0, GNU ld (GNU Binutils) 2.43.1) #1 Wed May 28 00:17:27 Europe 2025  
CPU: ARM926EJ-S [41069265] revision 5 (ARMv5TEJ), cr=0005317f  
CPU: VIVT data cache, VIVT instruction cache  
OF: fdt: Machine model: Microchip SAM9X60-CURIOSITY TM5000  
Memory policy: Data cache writeback  
cma: Reserved 16 MiB at 0x26c00000 on node -1  
Zone ranges:  
  Normal [mem 0x0000000020000000-0x0000000027ffffff]  
Movable zone start for each node  
Early memory node ranges  
  node 0: [mem 0x0000000020000000-0x0000000027ffffff]  
Initmem setup node 0 [mem 0x0000000020000000-0x0000000027ffffff]  
OF: reserved mem: Reserved memory: No reserved-memory node in the DT  
Kernel command line: console=ttyS0,115200 root=/dev/mmcblk0p2 rw rootwait cma=16  
m rootfstype=ext4 atmel.pm_modes=standby,ulp0 video=Unknown-1:800x480-16  
Dentry cache hash table entries: 16384 (order: 4, 65536 bytes, linear)  
Inode-cache hash table entries: 8192 (order: 3, 32768 bytes, linear)  
Built 1 zonelists, mobility grouping on. Total pages: 32768
```

Embedded Linux Startup Process

5. System initialization (Init Process)

- Executing Initialization Scripts.
- Starting User Interface.



Embedded Linux Startup Process

6. System Enters Running State

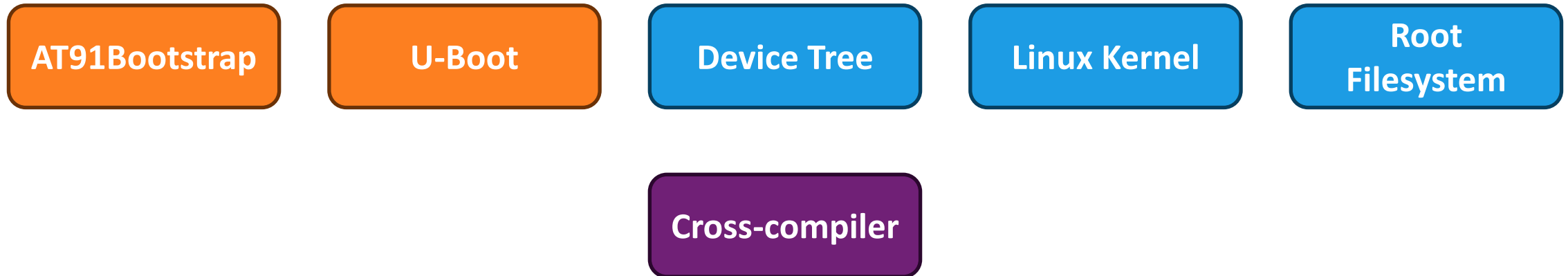
- Users can interact with the system through the command line.



Components of Embedded Linux System for MPU

Components of Embedded Linux System for MPU

- We need to obtain components suitable for MPU to complete the startup process and launch the Embedded Linux System.
- To do this, we need to understand how to obtain these components of Embedded Linux System . We can obtain the source code of each component from the GitHub Repository from Microchip ([linux4microchip](#) and [linux4sam](#)) and compile it to get the required parts.



Linux common commands

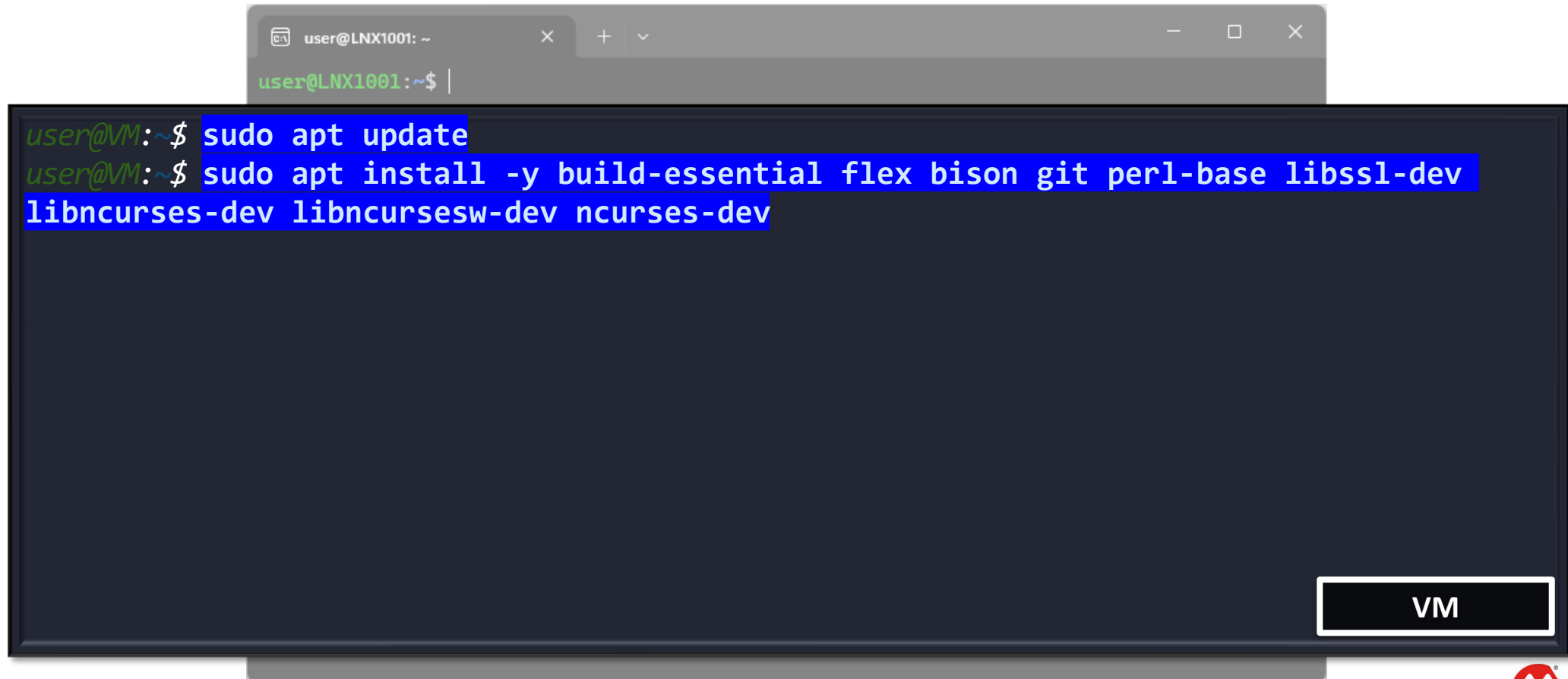
Hints

- **ls** : List all files in the current directory.
- **cd** : Enter directory.
- **pwd** : Show current path.
- **mkdir** : Create folder.
- **rm** : Delete Files, add -r to delete the folder.
- **cp** : Copy files.
- **mv** : Move files.
- **sudo** : Temporarily grant users privileged access to system.
- **apt** : Advanced Package Tool, automating the retrieval, configuration and installation of software packages.
- **git clone** : Clone repository.
- **git checkout**: Switch branches.

Setup Required Packages

Prepare

- Download and install essential host packages on your build host.

A terminal window with a dark background. The title bar shows 'user@LNX1001: ~'. The prompt is 'user@LNX1001:~\$'. Two commands are entered and highlighted in blue: 'sudo apt update' and 'sudo apt install -y build-essential flex bison git perl-base libssl-dev libncurses-dev libncursesw-dev ncurses-dev'. A 'VM' button is in the bottom right corner.

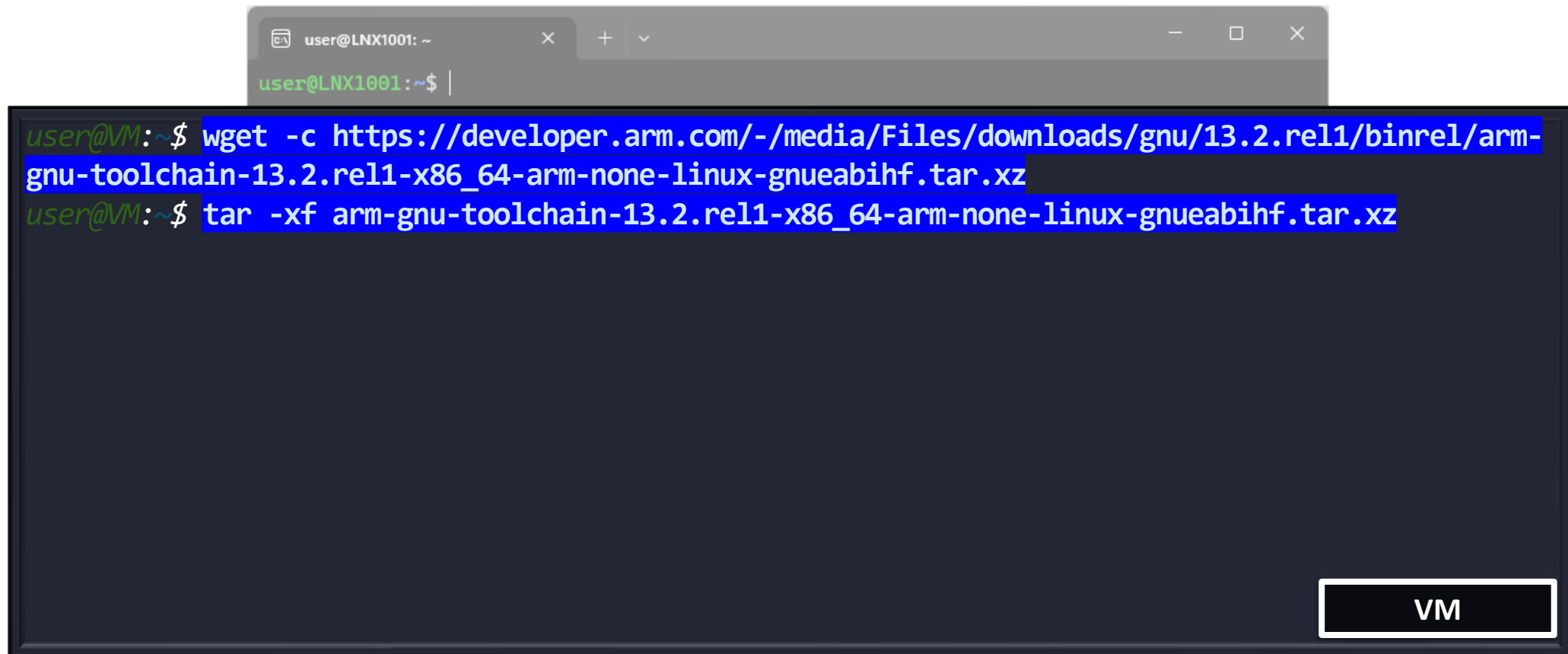
```
user@LNX1001: ~  
user@LNX1001:~$  
user@VM:~$ sudo apt update  
user@VM:~$ sudo apt install -y build-essential flex bison git perl-base libssl-dev  
libncurses-dev libncursesw-dev ncurses-dev
```

Setup ARM Cross Compiler

Cross-compiler

Prepare

- 1. Download and extract the ARM GNU Toolchain.



A terminal window with a dark background and light green text. The window title bar shows 'user@LNX1001: ~'. The terminal content shows two commands being executed: `wget -c https://developer.arm.com/-/media/Files/downloads/gnu/13.2.rel1/binrel/arm-gnu-toolchain-13.2.rel1-x86_64-arm-none-linux-gnueabihf.tar.xz` and `tar -xf arm-gnu-toolchain-13.2.rel1-x86_64-arm-none-linux-gnueabihf.tar.xz`. The prompt is `user@VM:~$`. A small 'VM' label is in the bottom right corner of the terminal window.

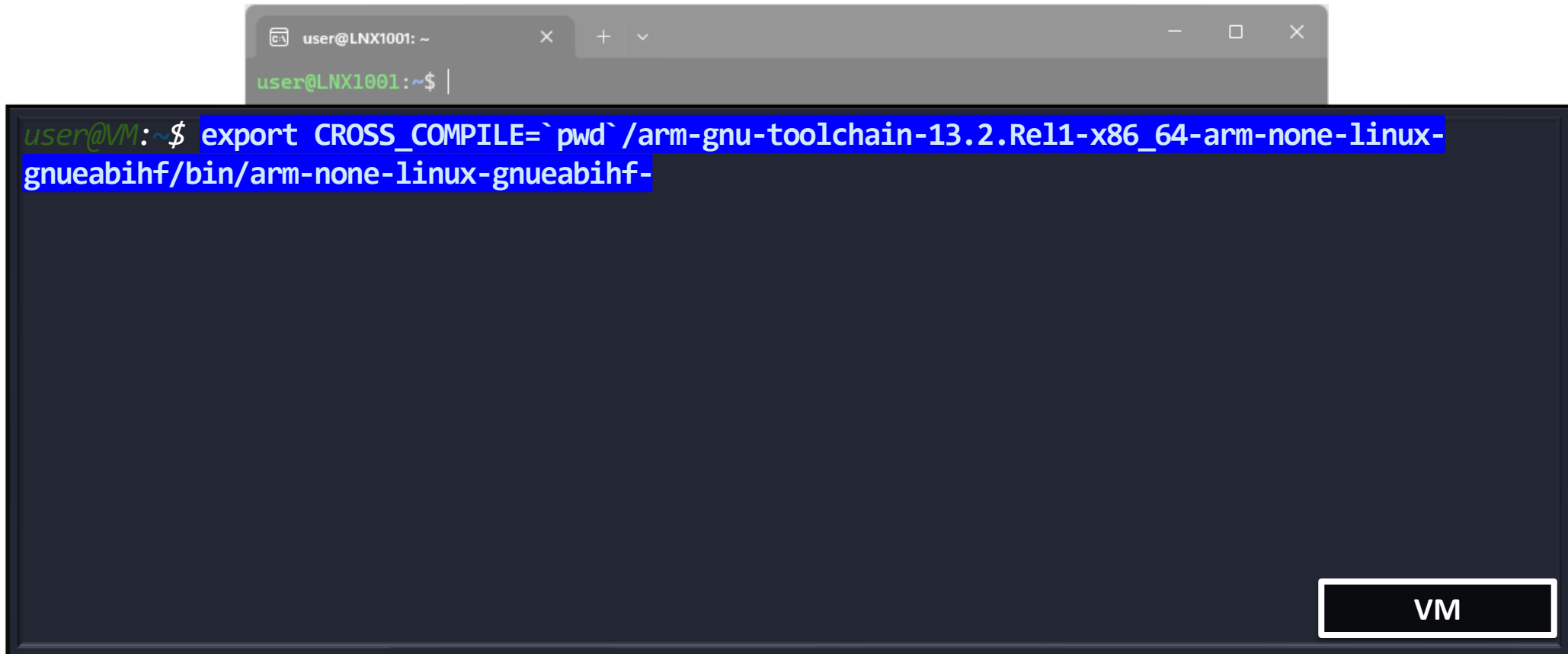
```
user@LNX1001: ~  
user@LNX1001:~$  
user@VM:~$ wget -c https://developer.arm.com/-/media/Files/downloads/gnu/13.2.rel1/binrel/arm-  
gnu-toolchain-13.2.rel1-x86_64-arm-none-linux-gnueabihf.tar.xz  
user@VM:~$ tar -xf arm-gnu-toolchain-13.2.rel1-x86_64-arm-none-linux-gnueabihf.tar.xz  
VM
```

Setup ARM Cross Compiler

Cross-compiler

Prepare

- 2. Add the ARM GNU Toolchain into your system.



A terminal window with a dark background. The title bar shows 'user@LNX1001: ~'. The prompt is 'user@LNX1001:~\$'. The command being entered is 'export CROSS_COMPILE=`pwd`/arm-gnu-toolchain-13.2.Rel1-x86_64-arm-none-linux-gnueabihf/bin/arm-none-linux-gnueabihf-'. The command is highlighted in blue. A 'VM' button is visible in the bottom right corner of the terminal window.

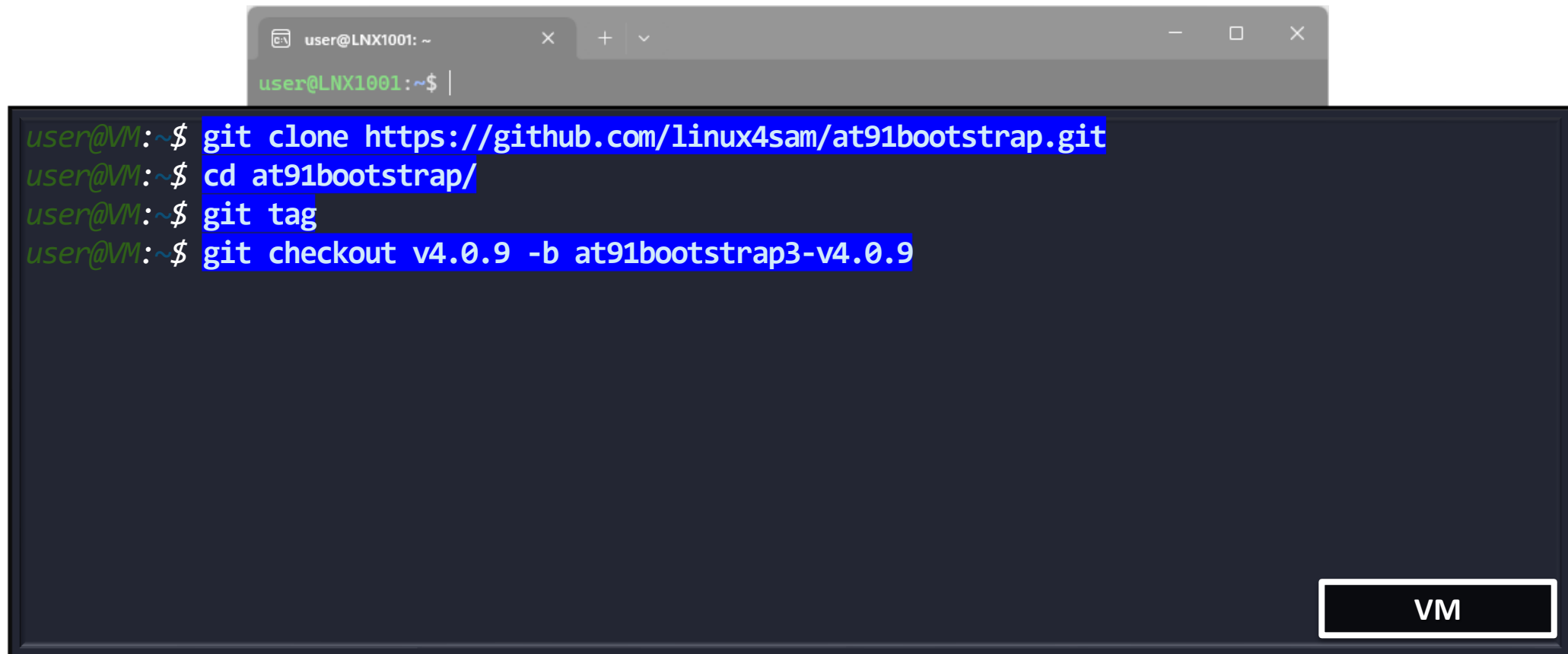
```
user@LNX1001: ~  
user@LNX1001:~$  
user@VM:~$ export CROSS_COMPILE=`pwd`/arm-gnu-toolchain-13.2.Rel1-x86_64-arm-none-linux-gnueabihf/bin/arm-none-linux-gnueabihf-
```

Build AT91Bootstrap

Cross-compiler

AT91Bootstrap

- 1. Get AT91Bootstrap Source Code and Switch branch to "v4.0.9" tag.



```
user@LNx1001: ~  
user@LNx1001:~$  
user@VM:~$ git clone https://github.com/linux4sam/at91bootstrap.git  
user@VM:~$ cd at91bootstrap/  
user@VM:~$ git tag  
user@VM:~$ git checkout v4.0.9 -b at91bootstrap3-v4.0.9
```

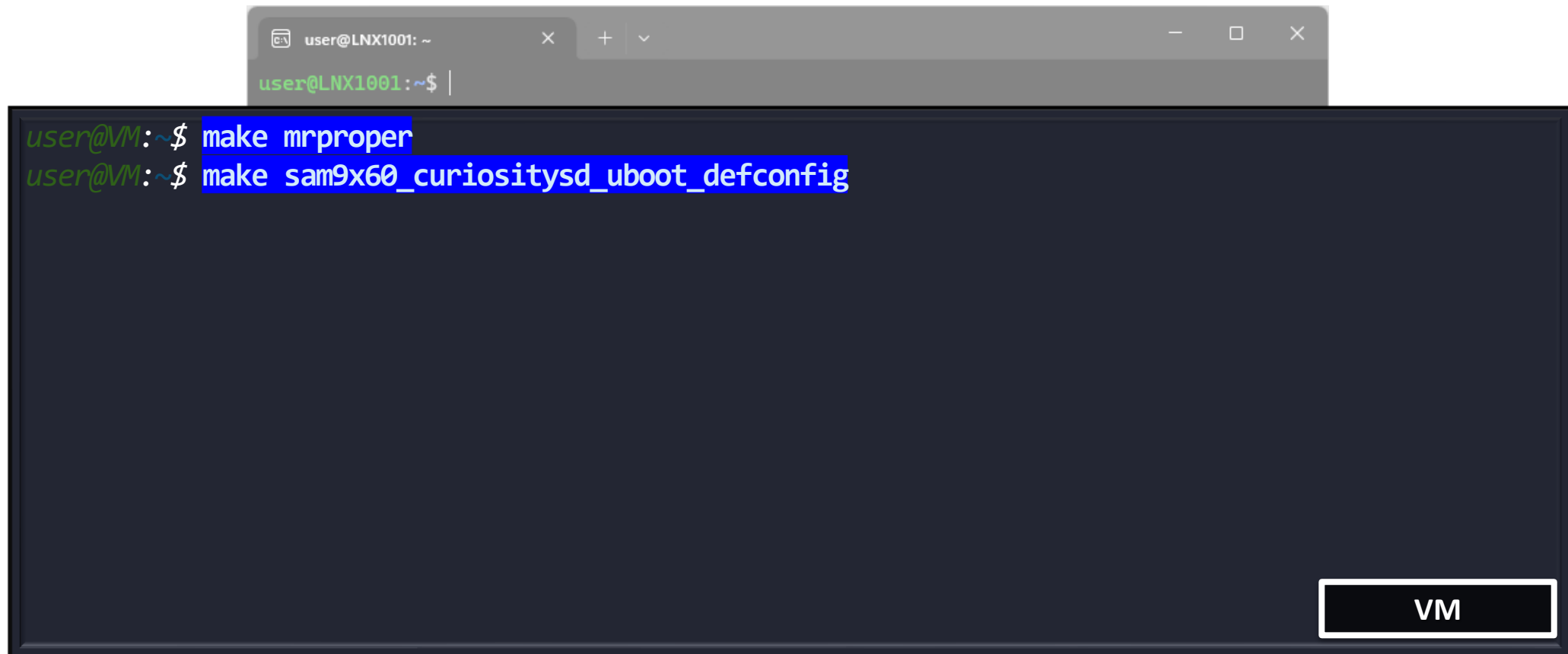
VM

Build AT91Bootstrap

Cross-compiler

AT91Bootstrap

- 2. Configure AT91Bootstrap to load U-Boot binary from SD Card.



A terminal window with a dark background. The title bar shows 'user@LNX1001: ~'. The prompt is 'user@LNX1001:~\$'. Two commands are entered and highlighted in blue: 'make mrproper' and 'make sam9x60_curiositysd_uboot_defconfig'. A 'VM' button is in the bottom right corner.

```
user@LNX1001: ~  
user@LNX1001:~$  
user@VM:~$ make mrproper  
user@VM:~$ make sam9x60_curiositysd_uboot_defconfig
```

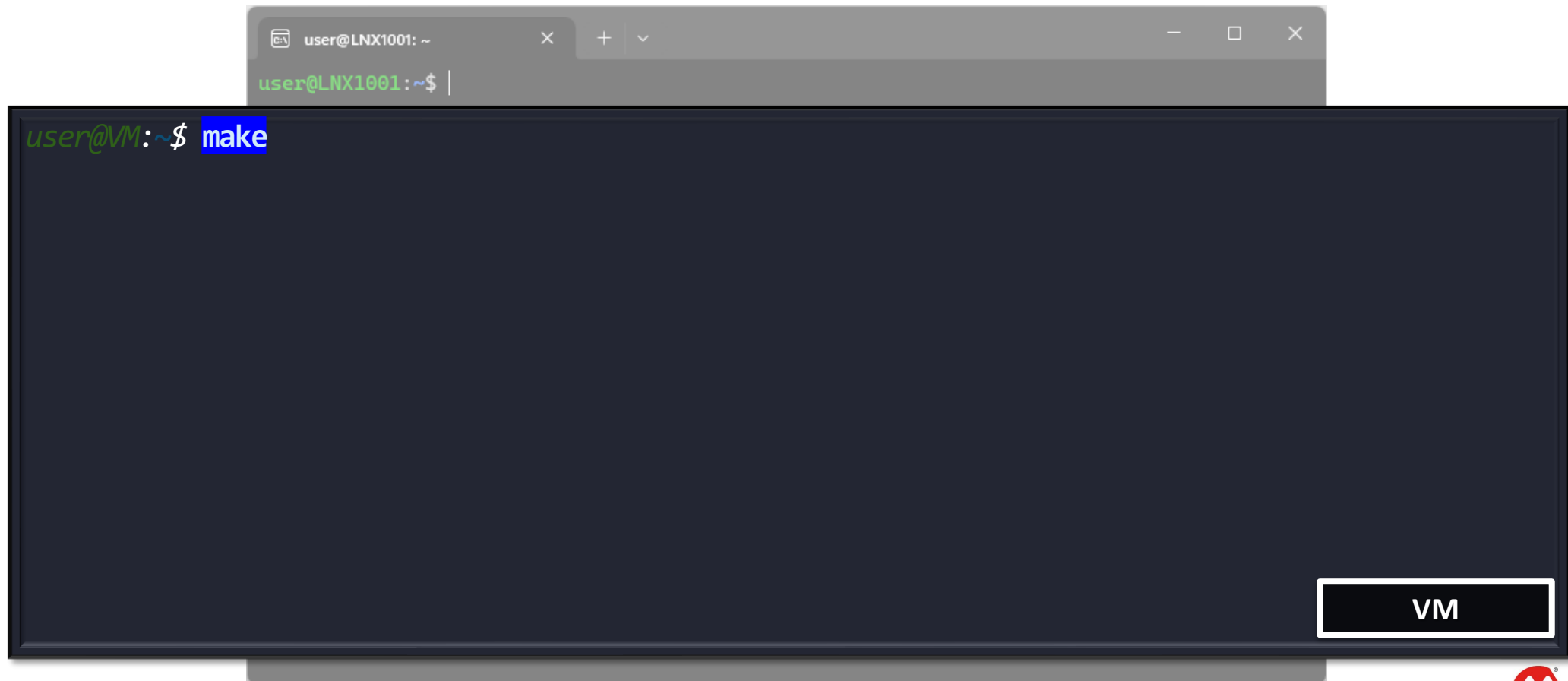
VM

Build AT91Bootstrap

Cross-compiler

AT91Bootstrap

- 3. Build the AT91Bootstrap.



A terminal window with a dark background. The title bar shows 'user@LNX1001: ~'. The prompt is 'user@LNX1001:~\$'. The command 'make' is entered and highlighted in blue. A small 'VM' label is in the bottom right corner of the terminal area.

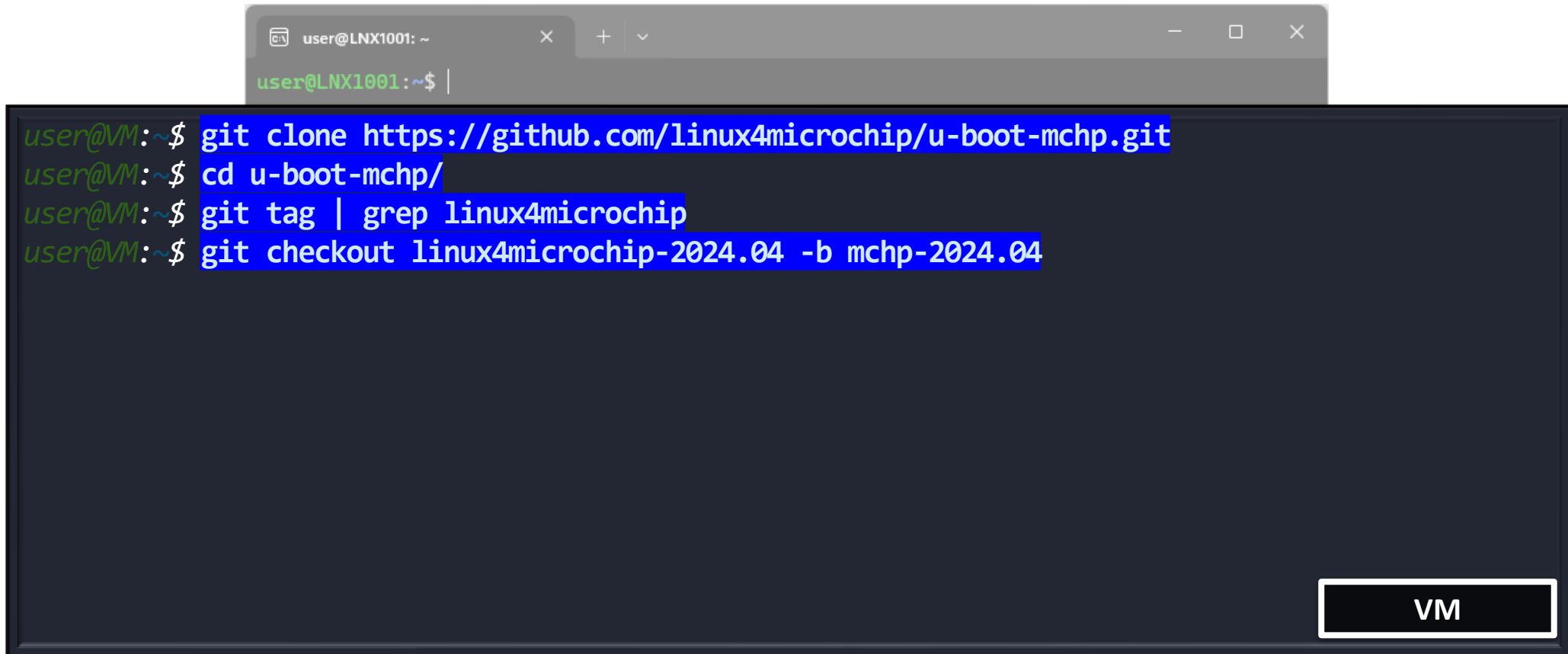
```
user@LNX1001: ~  
user@LNX1001:~$ make
```

Build U-Boot

Cross-compiler

U-Boot

- 1. Get U-Boot sources and Switch branch to "linux4microchip-2024.04" tag.



```
user@LNX1001: ~  
user@LNX1001:~$  
user@VM:~$ git clone https://github.com/linux4microchip/u-boot-mchp.git  
user@VM:~$ cd u-boot-mchp/  
user@VM:~$ git tag | grep linux4microchip  
user@VM:~$ git checkout linux4microchip-2024.04 -b mchp-2024.04
```

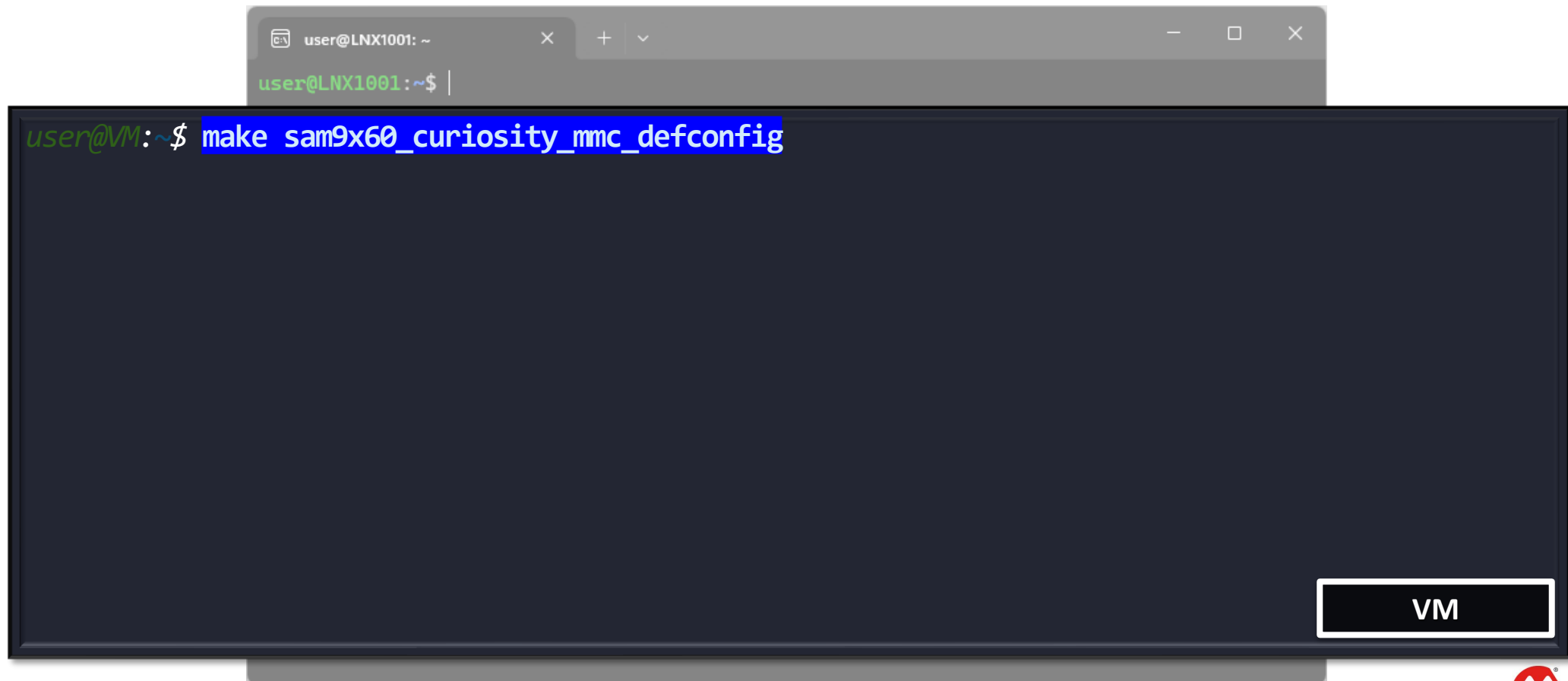
VM

Build U-Boot

Cross-compiler

U-Boot

- 2. Configure U-Boot to boot from SD Card.



A terminal window with a dark background. The title bar shows 'user@LNX1001: ~'. The prompt is 'user@LNX1001:~\$'. The command 'make sam9x60_curiosity_mmc_defconfig' is entered and highlighted in blue. A small 'VM' label is in the bottom right corner of the terminal window.

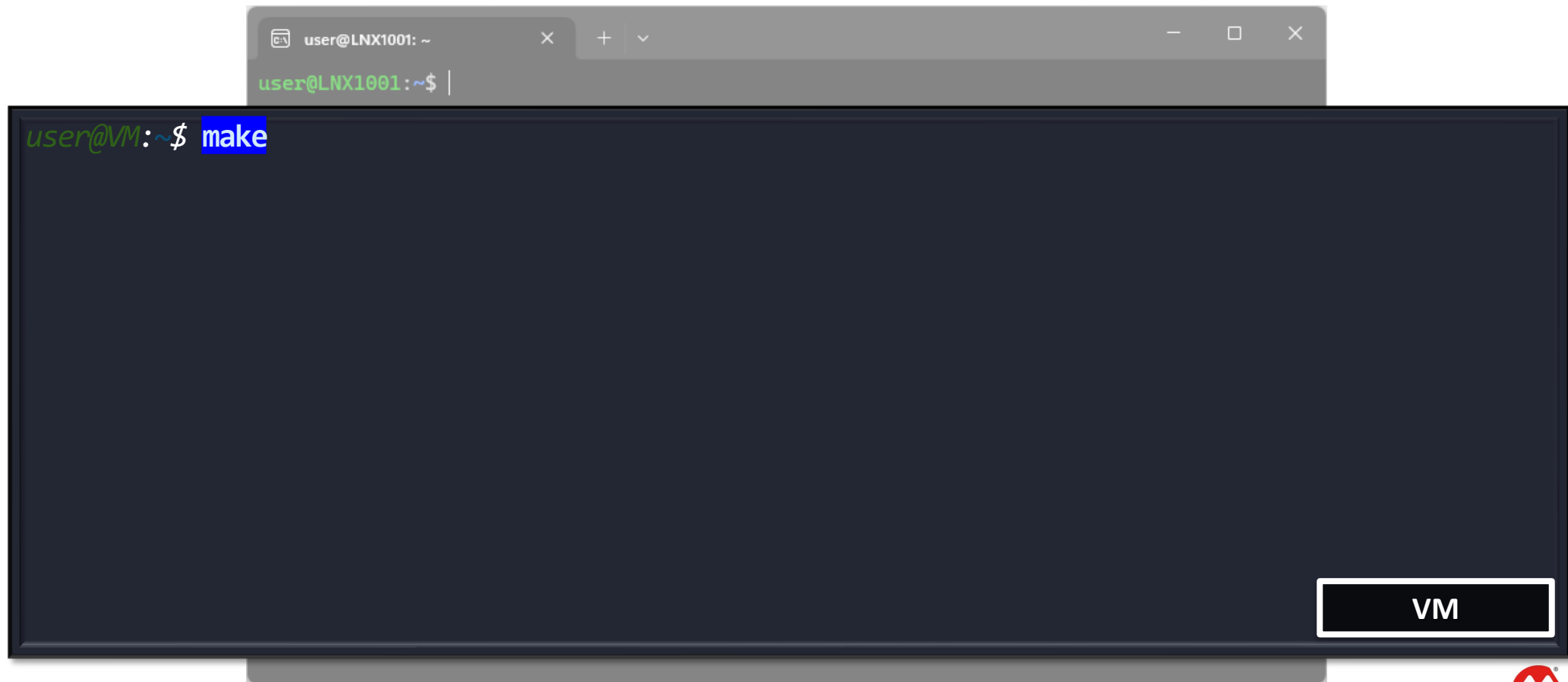
```
user@LNX1001: ~  
user@LNX1001:~$ |  
user@VM:~$ make sam9x60_curiosity_mmc_defconfig
```

Build U-Boot

Cross-compiler

U-Boot

- 1. Build the U-Boot.

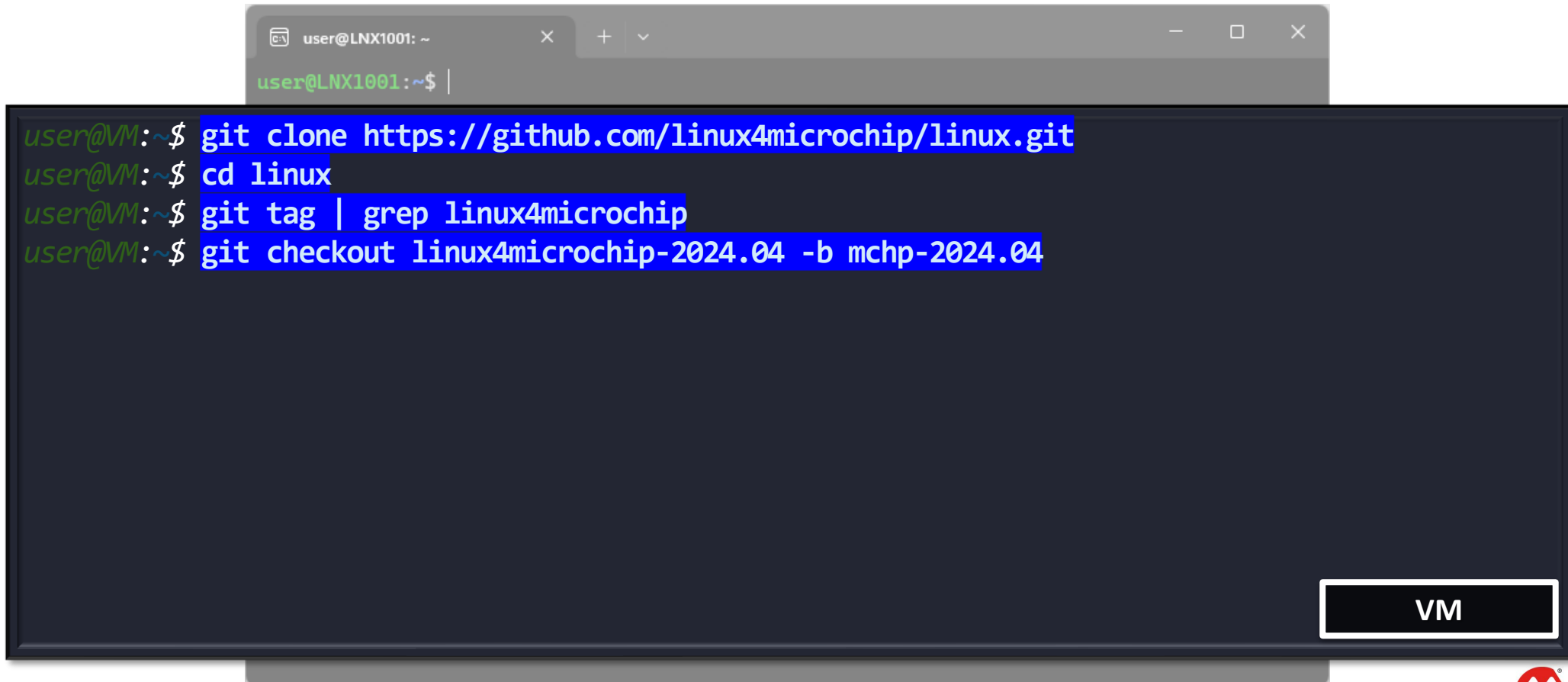


Build Kernel

Cross-compiler

Linux Kernel &
Device Tree

- 2. Get Kernel sources and Switch branch to "linux4microchip-2024.04" tag.



```
user@LNX1001: ~  
user@LNX1001:~$  
user@VM:~$ git clone https://github.com/linux4microchip/linux.git  
user@VM:~$ cd linux  
user@VM:~$ git tag | grep linux4microchip  
user@VM:~$ git checkout linux4microchip-2024.04 -b mchp-2024.04
```

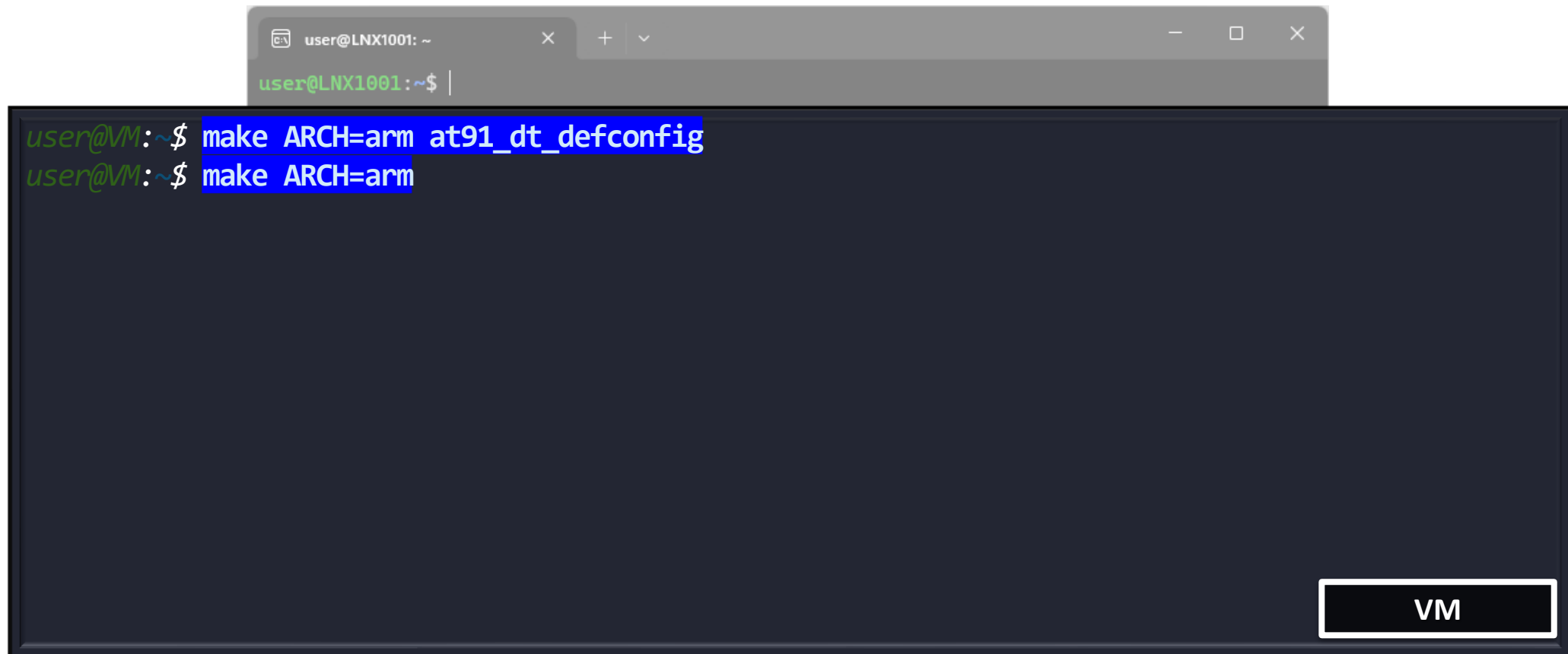
VM

Build Kernel

Cross-compiler

Linux Kernel &
Device Tree

- 3. Build the Linux kernel.



A terminal window with a dark background. The title bar shows 'user@LNX1001: ~'. The prompt is 'user@LNX1001:~\$'. Two commands are entered: 'make ARCH=arm at91_dt_defconfig' and 'make ARCH=arm'. The second command is highlighted in blue. A 'VM' button is in the bottom right corner.

```
user@LNX1001: ~  
user@LNX1001:~$  
user@VM:~$ make ARCH=arm at91_dt_defconfig  
user@VM:~$ make ARCH=arm
```

VM

Build Root File System

- **Root File System (rootfs)** is the foundation of Linux System. It contains the minimal set of files required to boot and run a system, such as the kernel, init scripts, libraries, and basic user utilities.
- **There are several ways to create a rootfs:**
 - **Building from scratch :**
This is a flexible but also the most time-consuming method.
 - **Based on an existing system :**
Copy an existing Linux distribution's rootfs and customize it.
 - **Using Build System tools :**
Use tools to automatically create embedded Linux system rootfs.

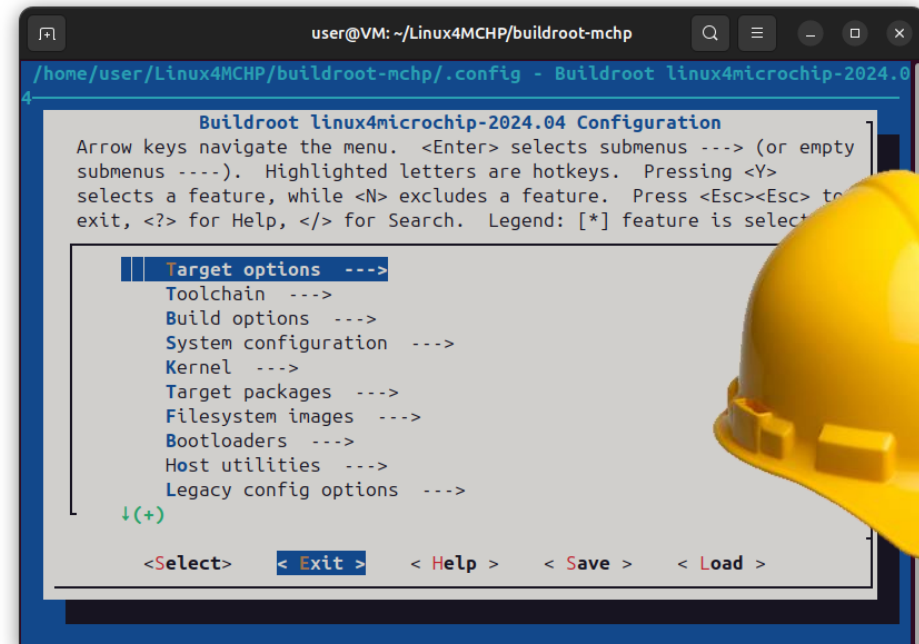
Build System and Buildroot

Build System

- Embedded Linux systems often contain more than hundreds of files. Manually compiling each file and managing dependencies is very time-consuming and error-prone, build system can simplify these processes.
- Build System **is a collection of tools and processes that automates the compilation, linking, and packaging of source code into programs, is essential for creating a system image by combining the Linux kernel, device drivers, applications, and other components.**
- **Common Build Systems for Embedded Linux :**
 - **Buildroot** : One of the most widely used build systems for embedded Linux, it provides a highly configurable environment that allows users to easily select packages, configure the kernel, and generate custom Linux distributions.
 - **Yocto Project** : The Yocto Project is a more complex and powerful build system that offers finer-grained control and is suitable for large and complex embedded systems.

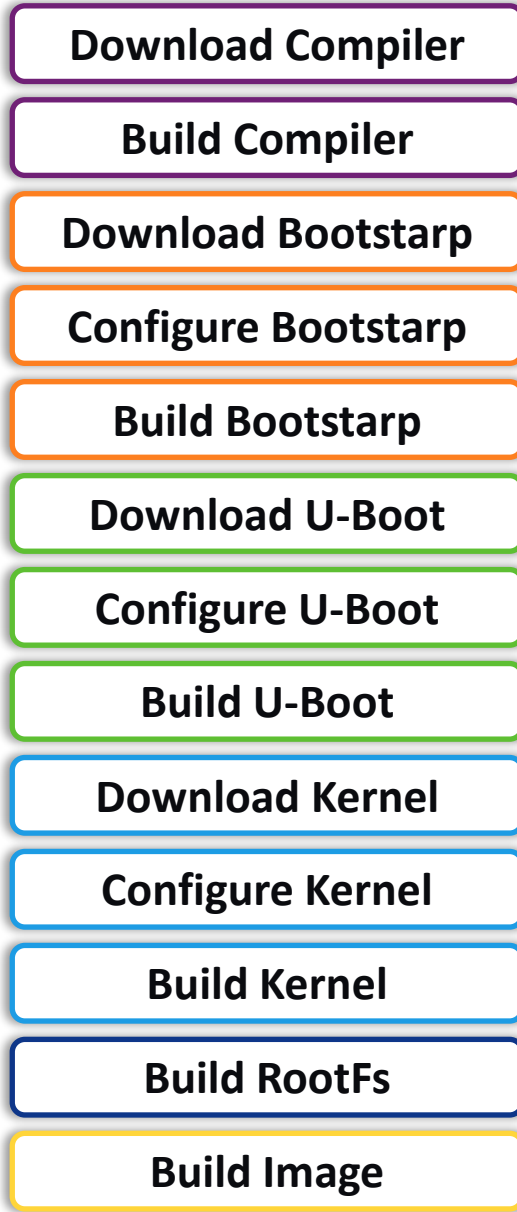
Buildroot

- **Buildroot** is a versatile and efficient build system designed for Embedded Linux Systems. It provides **a comprehensive set of tools and features to simplify the process** of creating customized Linux distributions for various embedded devices.
- It uses the Linux **Kconfig** configuration system, allowing easy selection of system content, and manage the compilation process through **Makefile**.
- Part of components currently the buildroot-mchp for Microchip can build:
 - Toolchain
 - Bootloader for Microchip SoC (AT91Bootstrap)
 - U-Boot
 - Linux
 - Device-Tree
 - Root File System
 - Embedded Linux OS Image



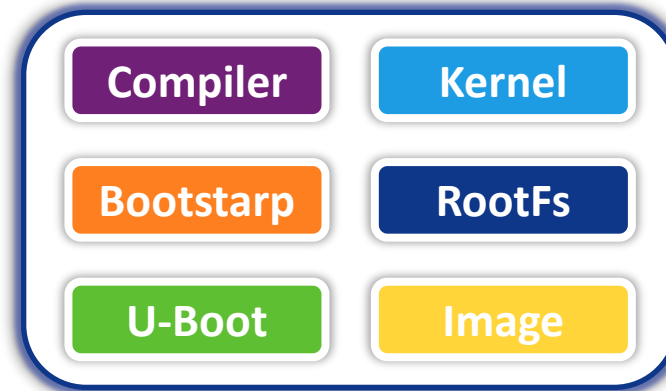
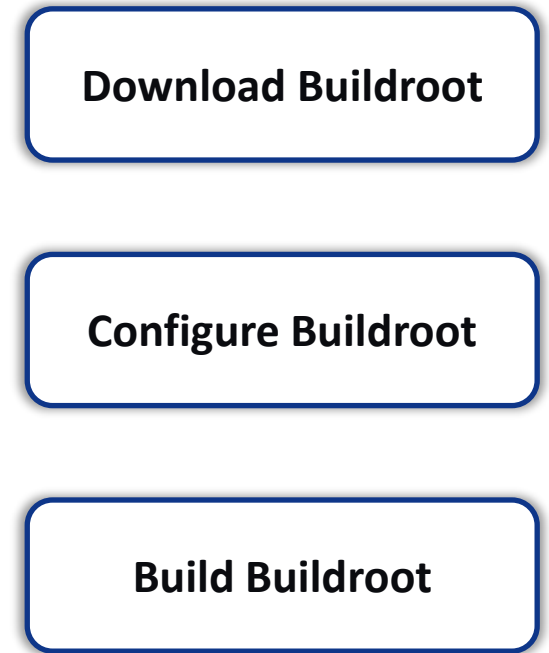
Buildroot vs Legacy

Legacy



VS

Buildroot



Build Buildroot

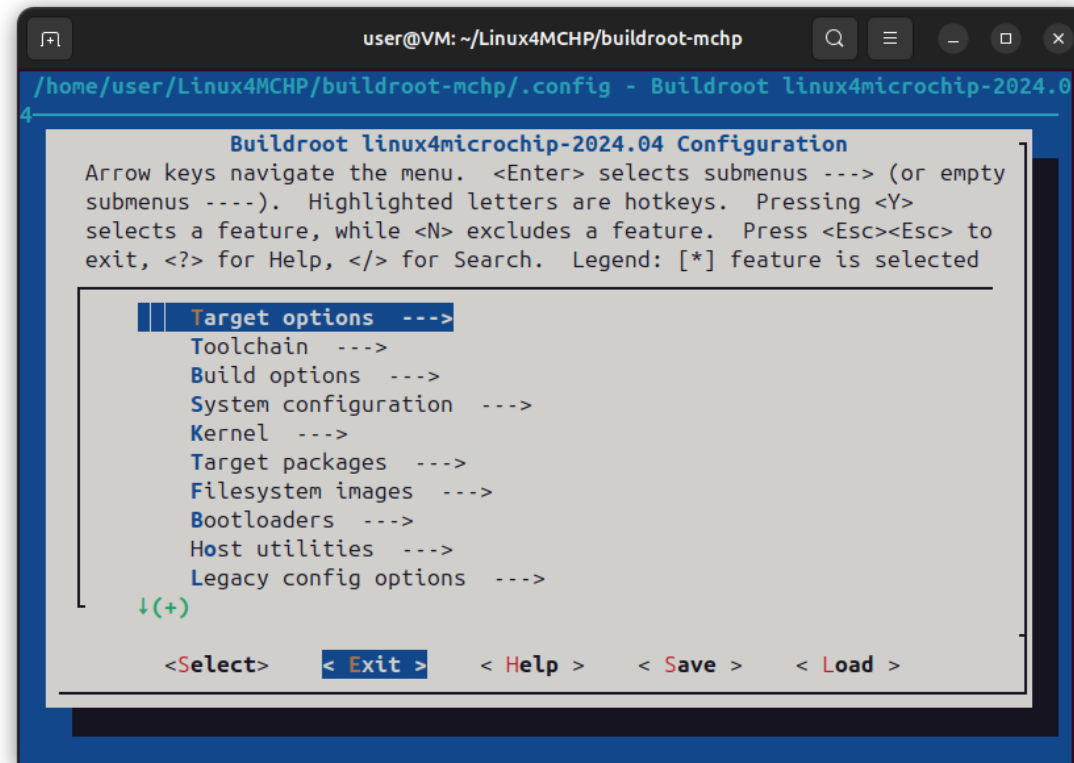
- We can use Buildroot to automatically build the required components according to the configuration.
- How to build Buildroot for Microchip SoC :
 1. Install the dependencies for Host Linux system.
 2. Get Buildroot sources. (e.g., **git clone**)
 3. Configure what you need. (e.g., **make menuconfig**)
 4. Execute the command and start automatic build Buildroot. (e.g., **make**)
 - a. Download required resources.
 - b. Compile and Linking.
 - c. Merge and output.
 5. Get something of what you need in Buildroot.

Makefile

- When you execute the make command, Buildroot will start analyzing the Makefile and execute the corresponding rules one by one according to the specified target.
- The entire construction process will be completed automatically. You only need to modify the configuration in the Makefile to customize the embedded system that meets your needs.
- **Basic Buildroot instructions :**
 - **make help** : Display all available targets.
 - **make menuconfig** : Configure toolchain, packages and kernel.
 - **make** or **make all** : Begin the build process.
- **Package-specific for Buildroot :**
 - **make <pkg>** : Build and install <pkg> and all its dependencies.
 - **make <pkg>-dirclean** : Remove <pkg> build directory.
 - **make <pkg>-extract** : Extract <pkg> sources.
 - **make <pkg>-rebuild** : Restart the build from the build step.

Configure Menu

- Buildroot provides a graphical configuration interface to configure and customize your embedded Linux system.
- We can execute the "**make menuconfig**" command to open the Buildroot Configure Menu, and then we can view the current configuration or modify.



```
user@VM: ~/Linux4MCHP/buildroot-mchp
/home/user/Linux4MCHP/buildroot-mchp/.config - Buildroot linux4microchip-2024.04
4
Buildroot linux4microchip-2024.04 Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected

Target options --->
  Toolchain --->
  Build options --->
  System configuration --->
  Kernel --->
  Target packages --->
  Filesystem images --->
  Bootloaders --->
  Host utilities --->
  Legacy config options --->
↓(+)

<Select>  <Exit>  <Help>  <Save>  <Load>
```

Lab1 – Prepare Buildroot and First Build

Lab1 – Prepare Buildroot and First Build

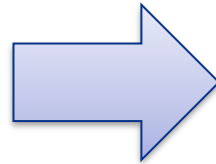
- We will prepare the Linux OS on the computer to create an Embedded Linux OS image, then try to build an environment suitable for developing SAM9X60 and an image of the Curiosity Board.
- Follow the steps below to complete the above requirements :
(The process requires the Internet !!)
 1. Prepare a WSL2 with a Linux OS installed on the computer and Launch it.
 2. Install the dependencies of Buildroot.
 3. Clone Buildroot source from GitHub repository.
 4. Switch Buildroot source branch.
 5. Modify the configuration.
 6. To start the build process.
 7. Check the output files.
 8. Flash OS images to SD card and boot.

Let's go!

Lab1 – Prepare Buildroot and First Build

Step 1

- Launch Ubuntu on the WSL.
([Refer the document to install Linux on Windows with WSL](#))

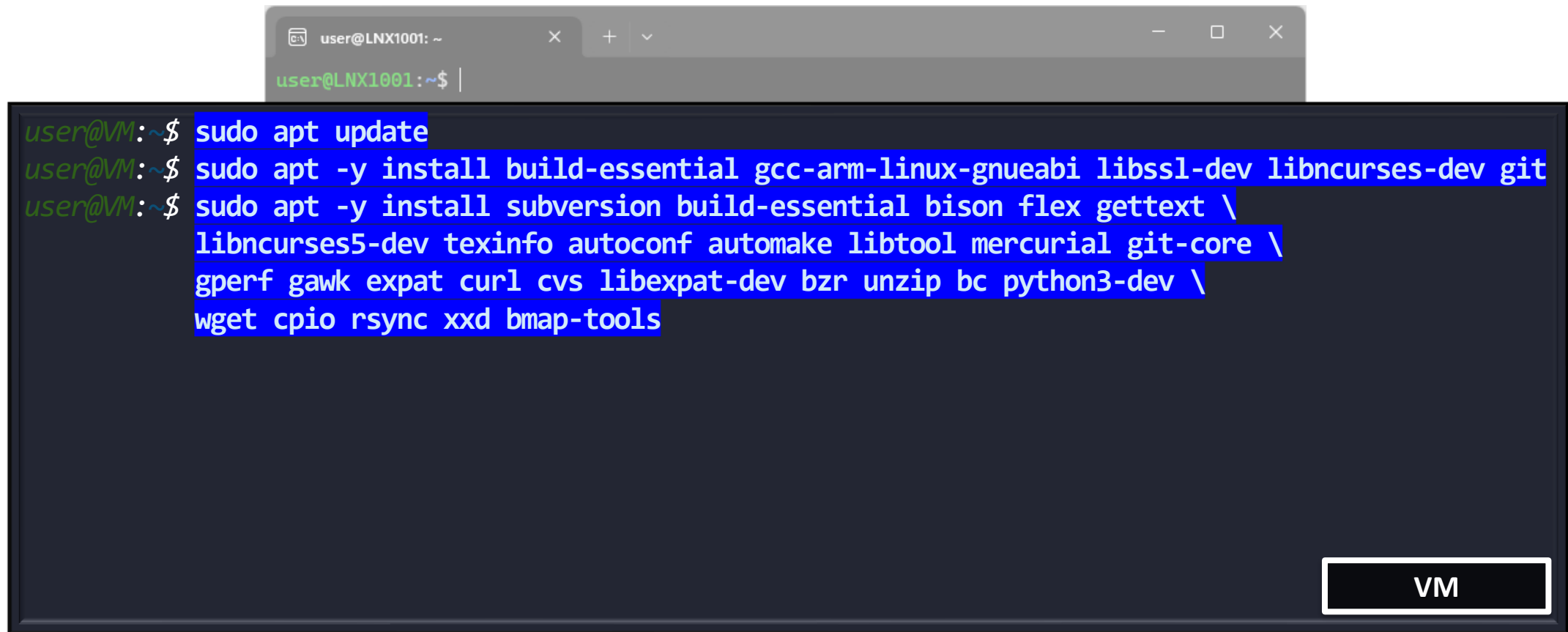


```
user@LNX1001: ~  
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.6.87.2-microsoft-standard-WSL2 x86_64)  
  
* Documentation: https://help.ubuntu.com  
* Management:   https://landscape.canonical.com  
* Support:      https://ubuntu.com/pro  
  
System information as of Wed Aug 6 14:37:06 CST 2025  
  
System load: 0.0      Processes:            79  
Usage of /:  3.2% of 1006.85GB Users logged in:          0  
Memory usage: 2%  
Swap usage:  0%  
  
* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s  
just raised the bar for easy, resilient and secure K8s cluster deployment.  
  
https://ubuntu.com/engage/secure-kubernetes-at-the-edge  
  
This message is shown once a day. To disable it please create the  
/home/user/.hushlogin file.  
user@LNX1001:~$
```

Lab1 – Prepare Buildroot and First Build

Step 2

- **Install the dependencies** of Buildroot for Host Linux system.



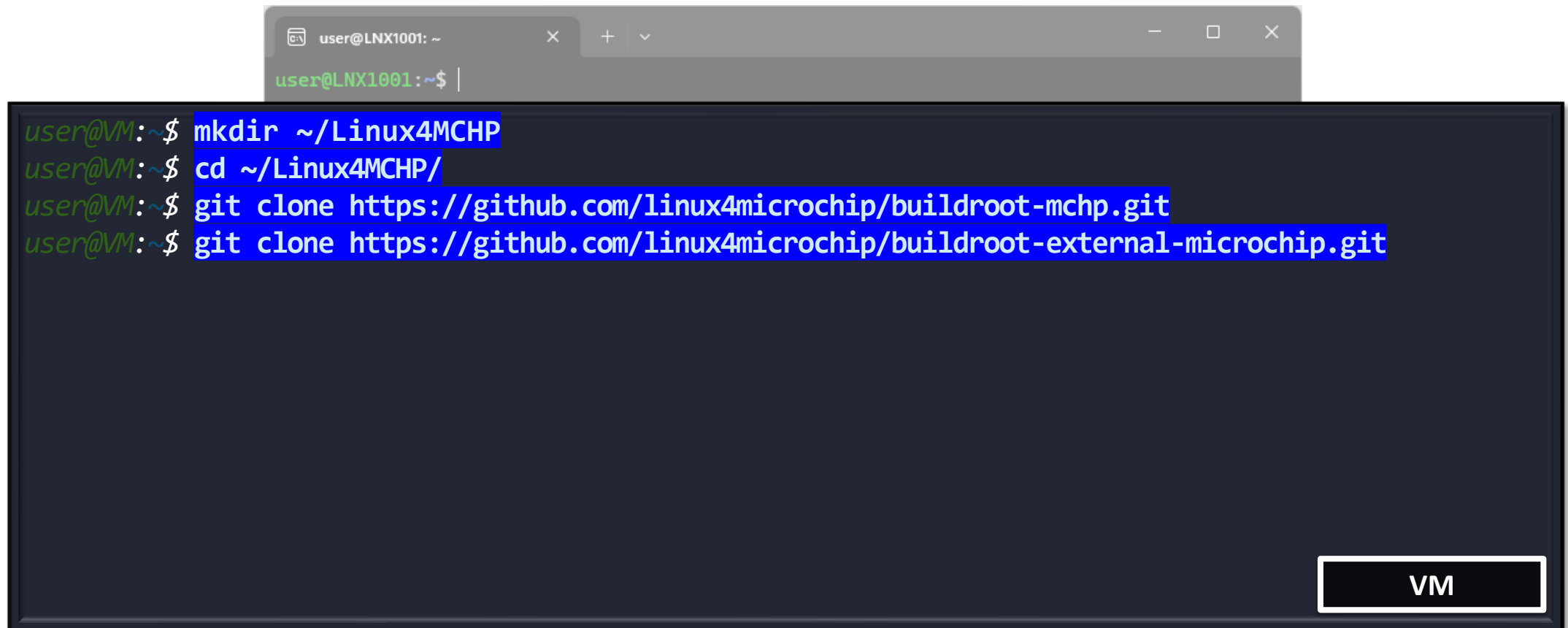
```
user@LNx1001: ~  
user@LNx1001:~$  
user@VM:~$ sudo apt update  
user@VM:~$ sudo apt -y install build-essential gcc-arm-linux-gnueabi libssl-dev libncurses-dev git  
user@VM:~$ sudo apt -y install subversion build-essential bison flex gettext \  
libncurses5-dev texinfo autoconf automake libtool mercurial git-core \  
gperf gawk expat curl cvs libexpat-dev bzip2 unzip bc python3-dev \  
wget cpio rsync xxd bmap-tools
```

VM

Lab1 – Prepare Buildroot and First Build

Step 3

- **Clone Buildroot source from GitHub repository.**



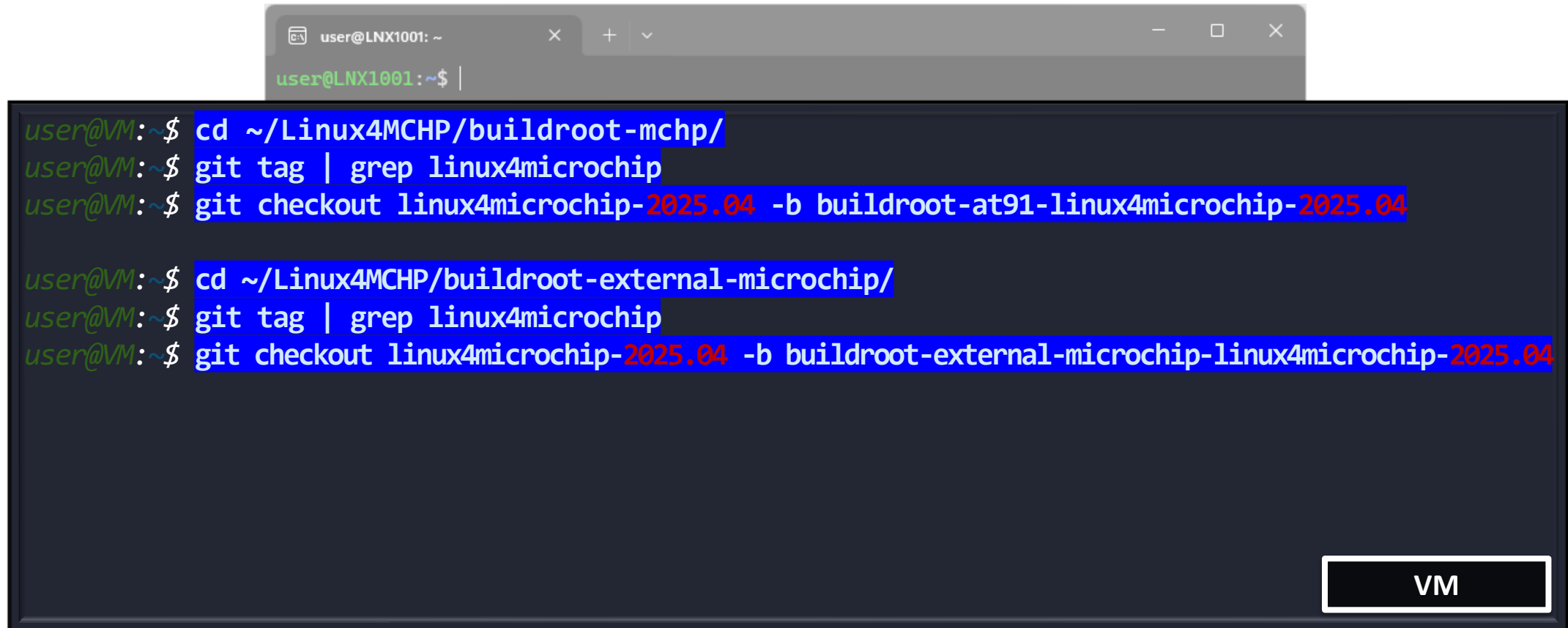
A terminal window with a dark background and light green text. The window title bar shows 'user@LNx1001: ~'. The terminal content shows four commands being entered at the 'user@VM:~\$' prompt. The first two commands are 'mkdir ~/Linux4MCHP' and 'cd ~/Linux4MCHP/'. The next two commands are 'git clone https://github.com/linux4microchip/buildroot-mchp.git' and 'git clone https://github.com/linux4microchip/buildroot-external-microchip.git'. The text for the last two commands is highlighted in blue. A 'VM' button is visible in the bottom right corner of the terminal window.

```
user@LNx1001: ~  
user@LNx1001:~$  
user@VM:~$ mkdir ~/Linux4MCHP  
user@VM:~$ cd ~/Linux4MCHP/  
user@VM:~$ git clone https://github.com/linux4microchip/buildroot-mchp.git  
user@VM:~$ git clone https://github.com/linux4microchip/buildroot-external-microchip.git
```

Lab1 – Prepare Buildroot and First Build

Step 4

- **Switch Buildroot source branch.** (buildroot and buildroot-external)



A terminal window with a dark background and light green text. The window title bar shows 'user@LNX1001: ~'. The terminal content shows two sets of commands. The first set is for the 'buildroot-mchp/' directory, and the second set is for the 'buildroot-external-microchip/' directory. In both cases, the user navigates to the directory, lists tags containing 'linux4microchip', and then checks out the 'linux4microchip-2025.04' branch. The branch names are highlighted in blue. A 'VM' button is visible in the bottom right corner of the terminal window.

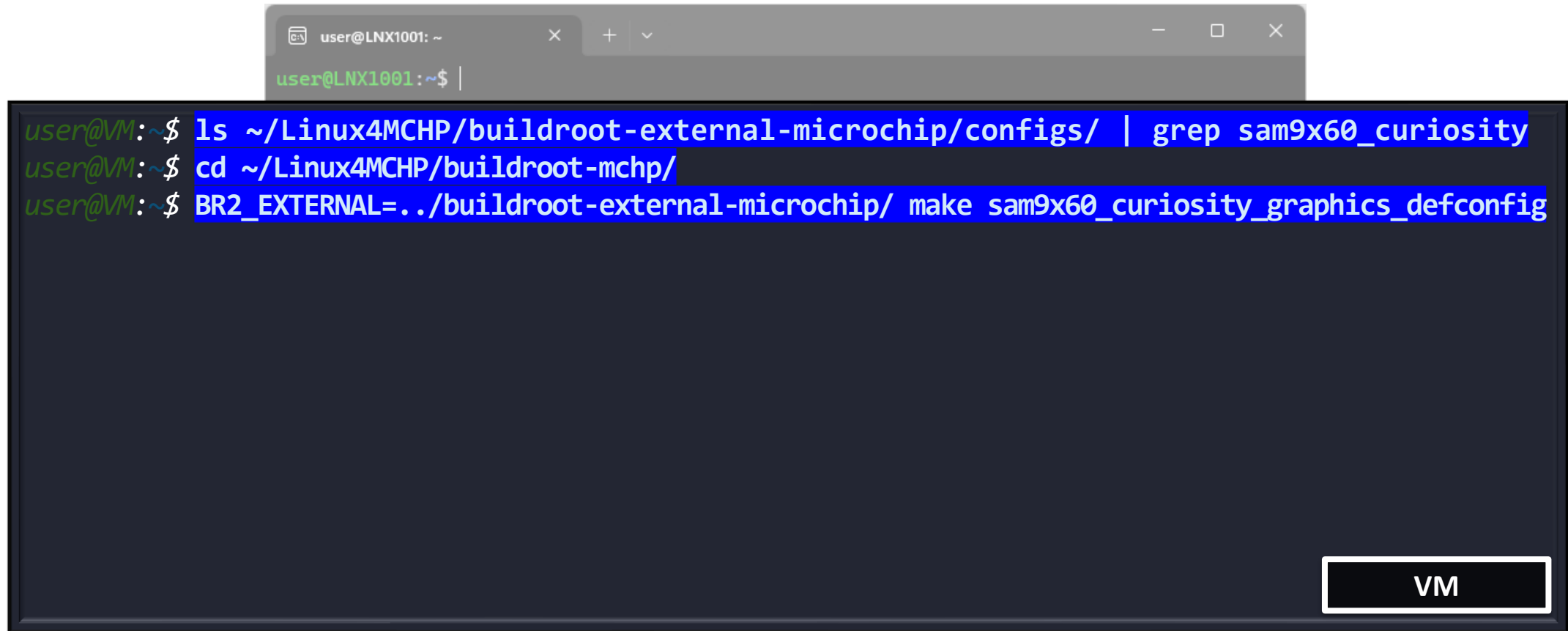
```
user@LNX1001: ~  
user@LNX1001:~$  
  
user@VM:~$ cd ~/Linux4MCHP/buildroot-mchp/  
user@VM:~$ git tag | grep linux4microchip  
user@VM:~$ git checkout linux4microchip-2025.04 -b buildroot-at91-linux4microchip-2025.04  
  
user@VM:~$ cd ~/Linux4MCHP/buildroot-external-microchip/  
user@VM:~$ git tag | grep linux4microchip  
user@VM:~$ git checkout linux4microchip-2025.04 -b buildroot-external-microchip-linux4microchip-2025.04
```

VM

Lab1 – Prepare Buildroot and First Build

Step 5

- **Modify the configuration with BR2-External Project.**
(Default configuration for SAM9X60 Curiosity Board with graphics)



A terminal window with a dark background and light green text. The window title bar shows 'user@LNX1001: ~'. The terminal content shows three commands being entered at the 'user@VM:~\$' prompt. The first two commands are highlighted in blue. The third command is not highlighted. A 'VM' button is visible in the bottom right corner of the terminal window.

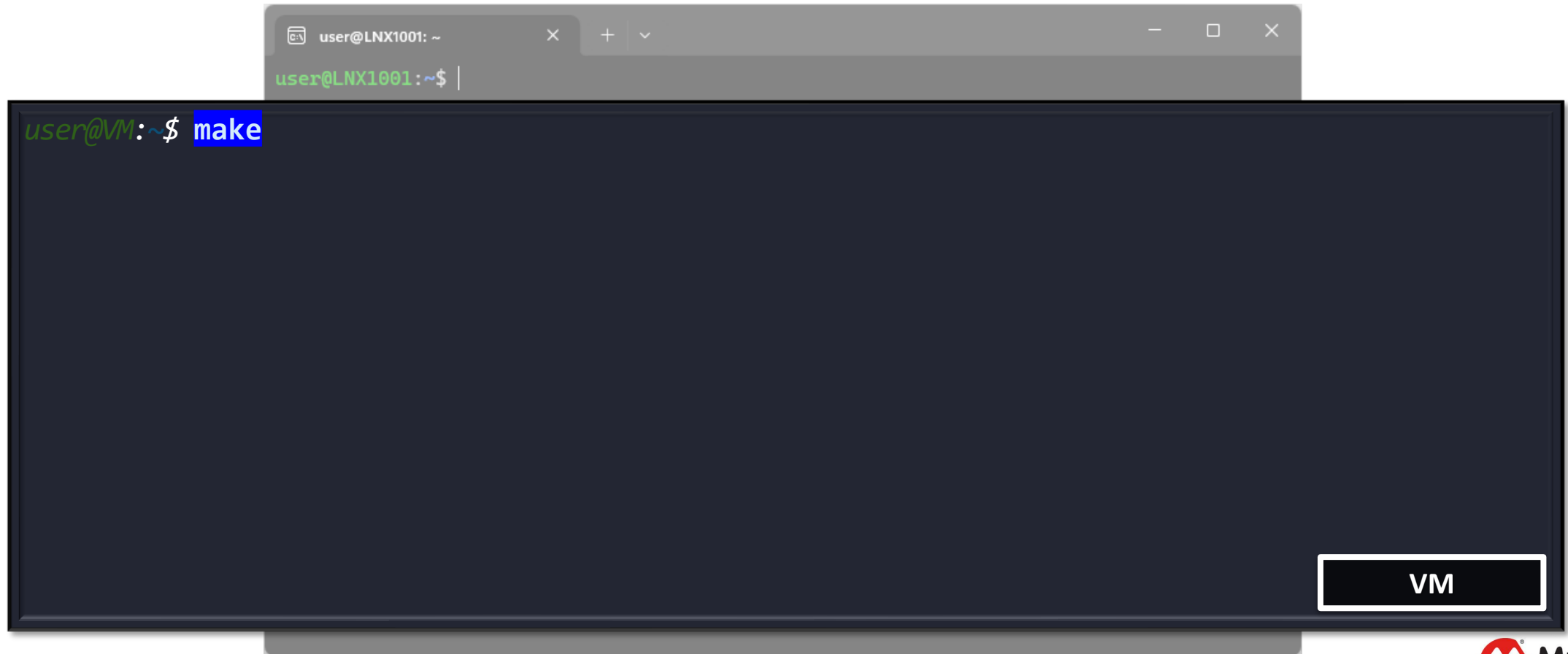
```
user@LNX1001: ~  
user@LNX1001:~$  
user@VM:~$ ls ~/Linux4MCHP/buildroot-external-microchip/configs/ | grep sam9x60_curiosity  
user@VM:~$ cd ~/Linux4MCHP/buildroot-mchp/  
user@VM:~$ BR2_EXTERNAL=../buildroot-external-microchip/ make sam9x60_curiosity_graphics_defconfig
```

VM

Lab1 – Prepare Buildroot and First Build

Step 6

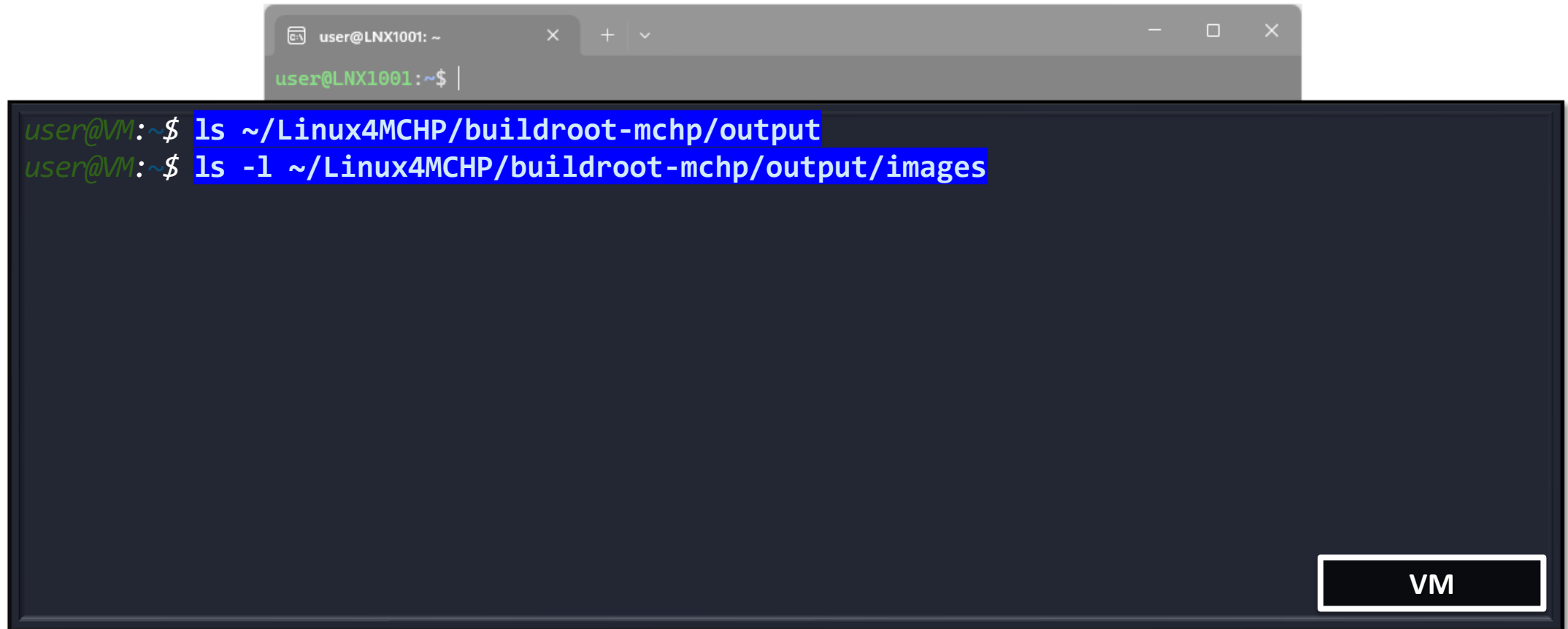
- To start the build process.



Lab1 – Prepare Buildroot and First Build

Step 7

- **Check the output files** in the BuildRoot directory.



A terminal window with a dark background. The title bar shows 'user@LNX1001: ~'. The prompt is 'user@LNX1001:~\$'. Two commands are entered and highlighted in blue: 'ls ~/Linux4MCHP/buildroot-mchp/output' and 'ls -l ~/Linux4MCHP/buildroot-mchp/output/images'. A 'VM' button is in the bottom right corner.

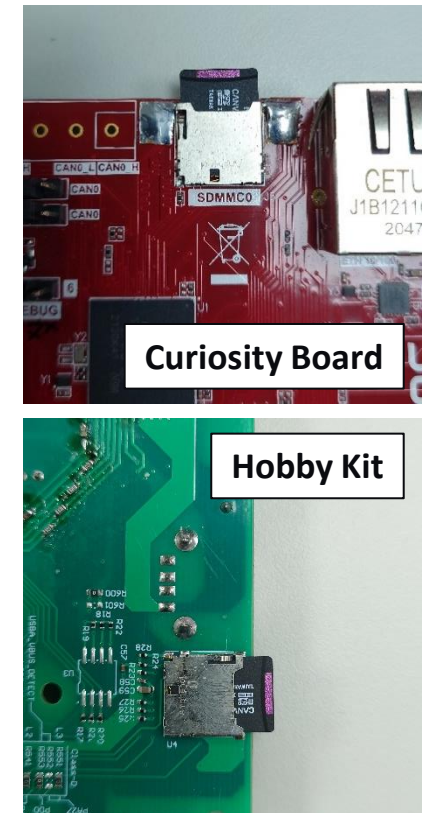
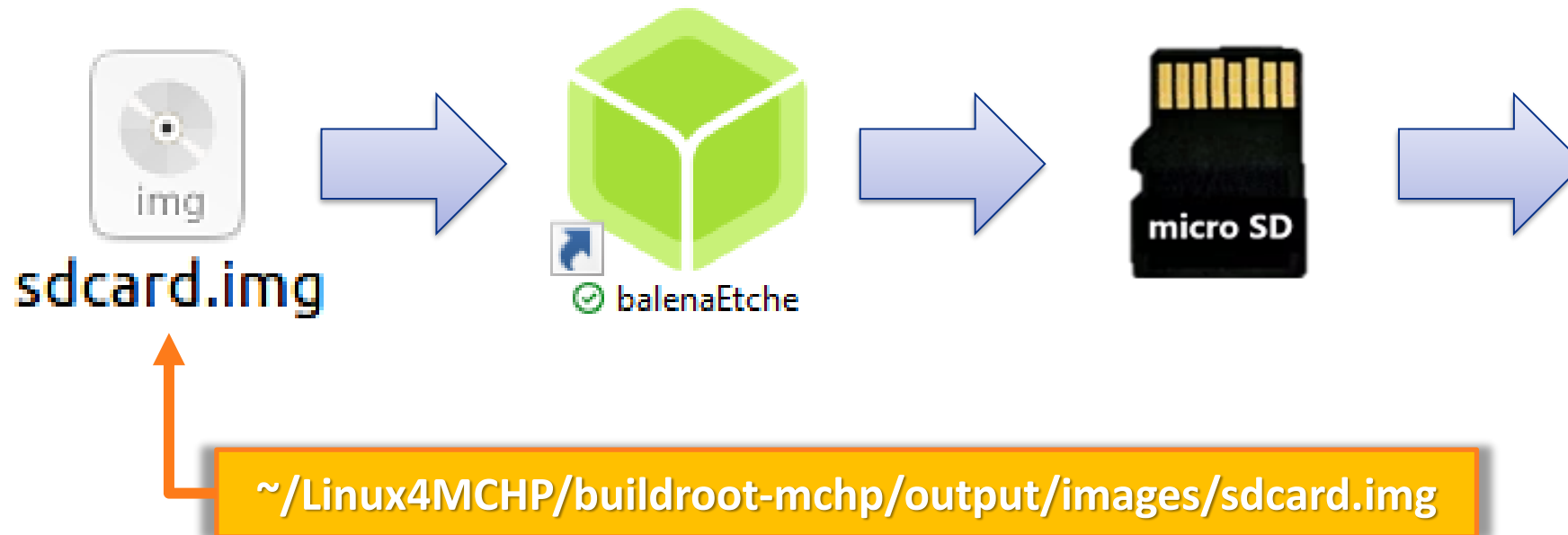
```
user@LNX1001: ~  
user@LNX1001:~$  
user@VM:~$ ls ~/Linux4MCHP/buildroot-mchp/output  
user@VM:~$ ls -l ~/Linux4MCHP/buildroot-mchp/output/images
```

VM

Lab1 – Prepare Buildroot and First Build

Step 8

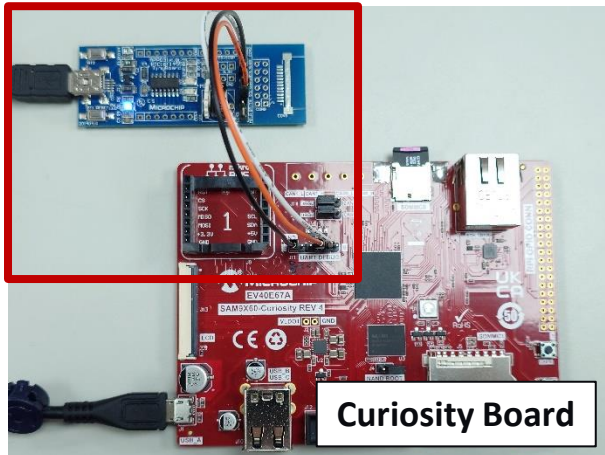
- **Prepare a SD Card** with Embedded Linux OS and mount it to EVB.
(Refer to the appendix to complete the Flash SD Card)



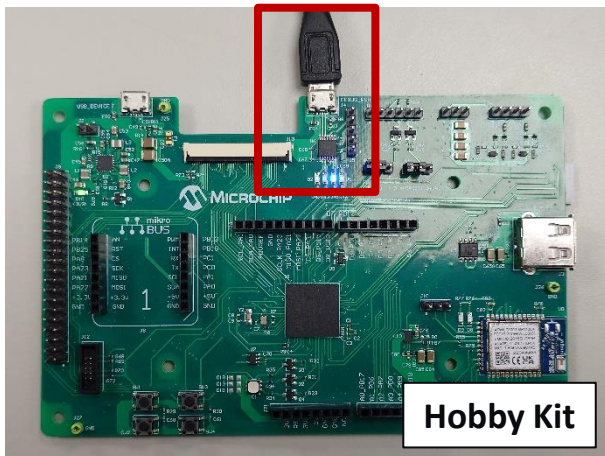
Lab1 – Prepare Buildroot and First Build

Step 9

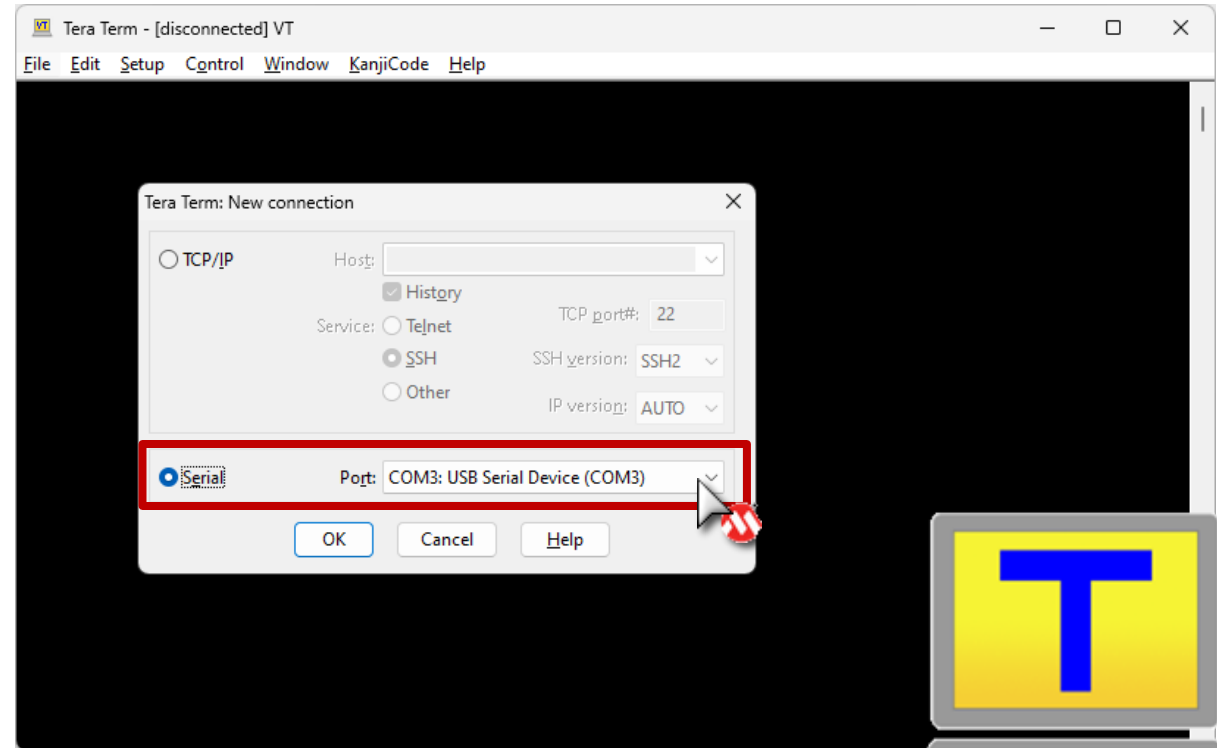
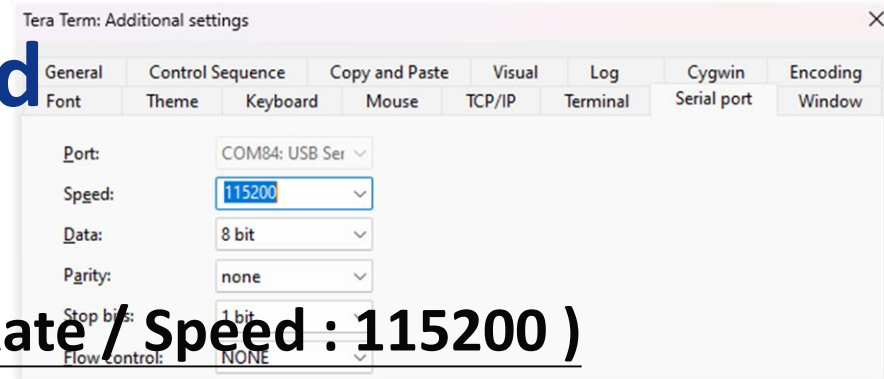
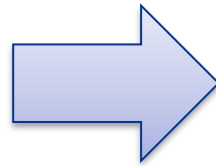
- **Connect the Debug port** of EVB and Tera Term. (**Baud Rate / Speed : 115200**)
(Select the **COM Port** that appears after the connection)



Curiosity Board



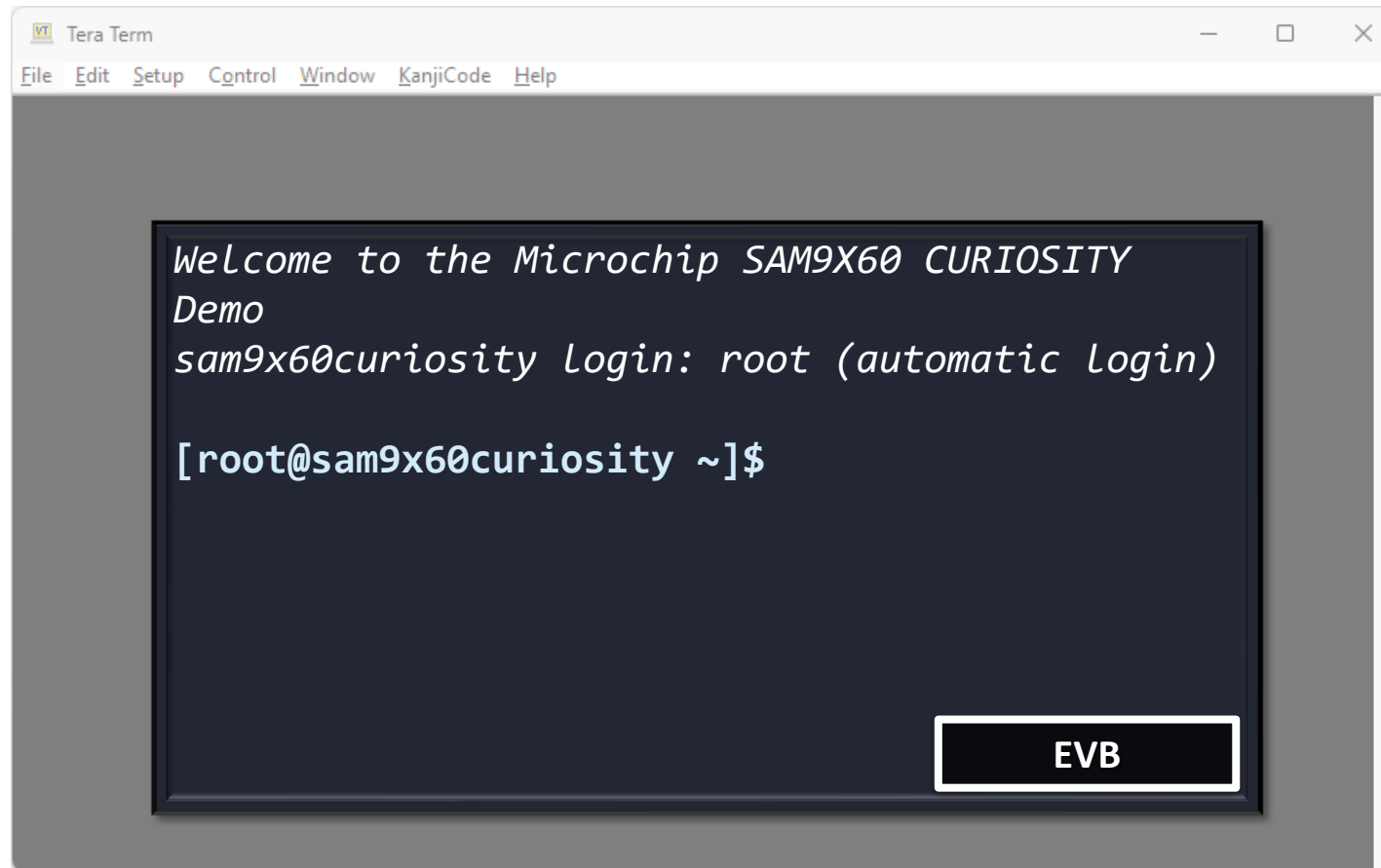
Hobby Kit



Lab1 – Prepare Development Tools

Step 10 (Embedded Linux OS of EVB will be used later)

- Use a Terminal (Tera Term) to **login** the Embedded Linux OS of EVB.



```
VT Tera Term
File Edit Setup Control Window KanjiCode Help

Welcome to the Microchip SAM9X60 CURIOSITY
Demo
sam9x60curiosity login: root (automatic login)

[root@sam9x60curiosity ~]$
```

EVB

Lab1 – Prepare Buildroot and First Build

Result

- Buildroot output is stored in a single directory, "**output/**". This directory contains several subdirectories.
- After mount the OS image and boot it, you will see the login screen of the Embedded Linux system running on the SAM9X60 MPU.

```
user@LNX1001: ~/Linux4MCHP x + v
user@LNX1001:~/Linux4MCHP/buildroot-mchp$ ls output/
build host images staging target
user@LNX1001:~/Linux4MCHP/buildroot-mchp$ ls -l output/images/
total 1310552
-rwxr-xr-x 1 user user 37072 Sep 9 03:36 at91-sam9x60_curiosity.dtb
-rwxr-xr-x 1 user user 18929 Sep 9 03:37 at91bootstrap.bin
-rwxr-xr-x 1 user user 18929 Sep 9 03:37 boot.bin
-rw-r--r-- 1 user user 16777216 Sep 9 04:02 boot.vfat
-rw-r--r-- 1 user user 943718400 Sep 9 04:01 rootfs.ext2
lrwxrwxrwx 1 user user 11 Sep 9 04:01 rootfs.ext4 -> rootfs.ext2
-rw-r--r-- 1 user user 425656320 Sep 9 04:02 rootfs.tar
-rwxr-xr-x 1 user user 18929 Sep 9 03:37 sam9x60-sdcardboot-uboot-4.0.11.bin
-rw-r--r-- 1 user user 5631664 Sep 9 03:38 sam9x60_curiosity.itb
-rw-r--r-- 1 user user 2053 Sep 9 03:38 sam9x60_curiosity.its
-rw-r--r-- 1 user user 4868 Sep 9 03:38 sam9x60_curiosity_pda5.dtbo
-rw-r--r-- 1 user user 1534 Sep 9 03:38 sam9x60_curiosity_wilc3000.dtbo
-rw-r--r-- 1 user user 961544192 Sep 9 04:02 sdcard.img
-rw-r--r-- 1 user user 458952 Sep 9 03:31 u-boot.bin
-rwxr-xr-x 1 user user 16384 Sep 9 03:31 uboot-env.bin
-rw-r--r-- 1 user user 5585520 Sep 9 03:36 zImage
user@LNX1001:~/Linux4MCHP/buildroot-mchp$ |
```

```
COM18 - Tera Term VT
File Edit Setup Control Window KanjiCode Help
[OK] Finished Virtual Console Setup.
[OK] Reached target System Initialization.
[OK] Started Discard unused filesystem blocks once a week.
[OK] Started Daily Cleanup of Temporary Directories.
[OK] Reached target Timer Units.
[OK] Listening on D-Bus System Message Bus Socket.
[OK] Listening on OpenSSH Server Socket (sshd, AF_UNIX Local).
[OK] Listening on Hostname Service Socket.
[OK] Reached target Socket Units.
[OK] Reached target Basic System.
[OK] Starting D-Bus System Message Bus...
[OK] Started Hardware RNG Entropy Gatherer Daemon.
[OK] Started Serial Getty on ttyS0.
[OK] Reached target Login Prompts.
[OK] Starting OpenSSH server daemon...
[OK] Started D-Bus System Message Bus.
[OK] Started OpenSSH server daemon.
[OK] Reached target Multi-User System.

Welcome to the Microchip SAM9X60 CURIOSITY Demo
sam9x60curiosity login: root (automatic login)

[root@sam9x60curiosity ~]$
```

Modify the Configuration of Buildroot for Hobby Kit

Buildroot Source Tree

- **Buildroot Source Tree** is a well-organized directory structure that contains all the source code, scripts, and configuration files necessary to construct a complete embedded system.
- **Key Directories in the Source Tree:**
 - **arch/** : configuration scripts for various CPU architectures such as ARM, MIPS, etc.
 - **linux/** : automated build scripts for the Linux kernel.
 - **fs/** : source code for various file systems such as ubi, ext2, etc.
 - **package/** : configuration files for all the software packages supported by Buildroot.
 - **dl/** : downloaded source code packages and application archives.
 - **configs/** : configuration parameters for development boards.
 - **board/** : board-specific configuration.
 - **output/** : output files from the build process.
 - **others**

Buildroot Source Tree

- In the course, we will build Buildroot and obtain resources suitable for the evaluation board, such as a Linux OS image.
- We can find the output files in the "**output/images/**" directory from the build process which we needs.
- During the build process, source code packages and application archives based on your configuration will be downloaded through the Internet. These files will be stored in the "**dl/**" folder, and the source code will be extract to the "**output/build/**" folder.
- Directory during build process and output :
 - **output/build/** : Source code for source code packages and application archives.
 - **output/images/** : Output files during build process.

Lab2 – Modify the Configuration of Buildroot and Rebuild

Lab2 – Modify the Configuration of Buildroot and Rebuild

- Before starting this Lab, you need to complete the previous Lab.
- The image we compiled previously was suitable for the SAM9X60 Curiosity Board. In this Lab we will try to **create an image suitable for the Hobby Kit**, we need to modify some configurations to complete it.
- Follow the steps below to complete the above requirements :
 1. Modify the configuration of Buildroot for SAM9X60 Hobby Kit.
 2. Open the Device-Tree-Source file.
 - a. Modify the default LED heartbeat color to Green.
 - b. Modify the User-Button pin for Hobby Kit.
 - c. Modify the Device-Tree-Source to create SPI node.
 - d. Modify the Device-Tree-Source for SD Card Detect.
 3. Rebuild the Buildroot for Hobby Kit.
 4. Flash OS images to SD card.
 5. Check the modified part in the Embedded Linux System.

Let's go!

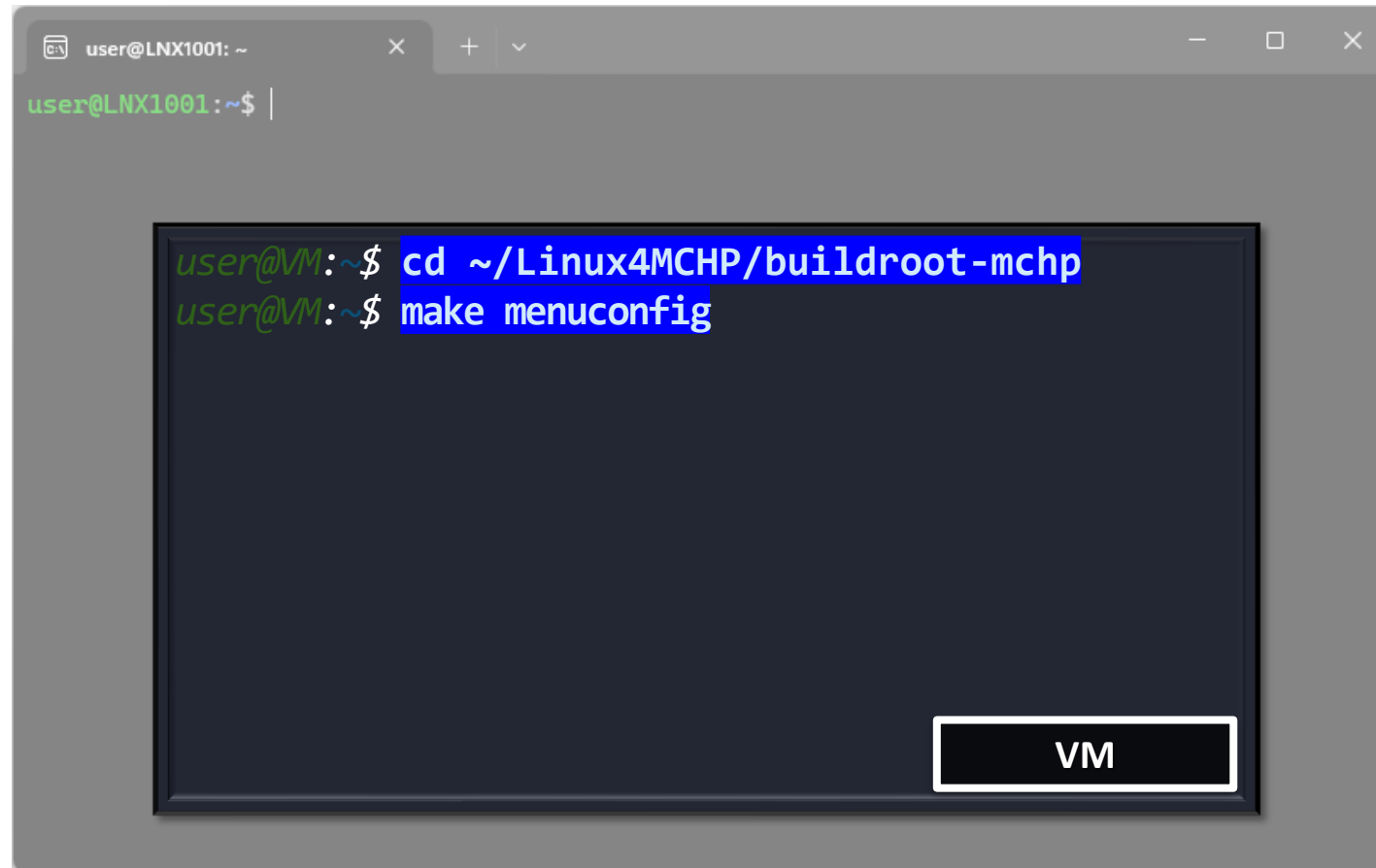
Modify the configuration of Buildroot for SAM9X60 Hobby Kit

Lab2 – Modify the Configuration of Buildroot and Rebuild

Lab2 – Modify the Configuration of Buildroot and Rebuild

Step 1.1

- Use the menu to configure Buildroot.

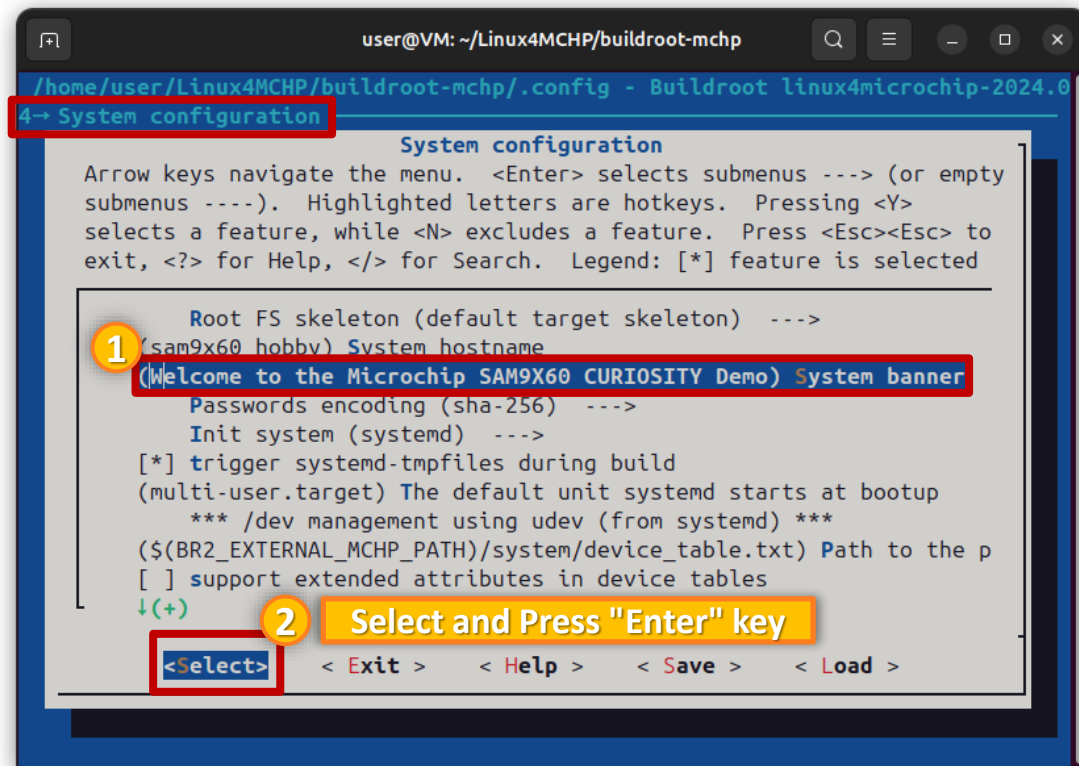
A terminal window with a grey title bar showing 'user@LNX1001: ~'. The main terminal area has a dark background. It shows two commands being entered: 'cd ~/Linux4MCHP/buildroot-mchp' and 'make menuconfig'. Both command lines are highlighted with a blue selection box. A small white box with the text 'VM' is located in the bottom right corner of the terminal window.

```
user@LNX1001: ~$ cd ~/Linux4MCHP/buildroot-mchp
user@VM:~$ make menuconfig
```

Lab2 – Modify the Configuration of Buildroot and Rebuild

Step 1.2

- **Modify** the **Login Welcome Message** for SAM9X60 Hobby Kit.
(SAM9X60 CURIOSITY Demo → **SAM9X60 Hobby Kit**)



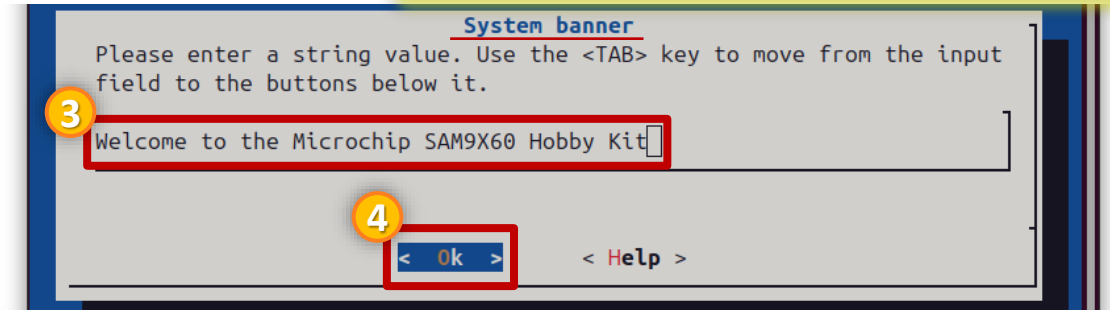
```
user@VM: ~/Linux4MCHP/buildroot-mchp
/home/user/Linux4MCHP/buildroot-mchp/.config - Buildroot linux4microchip-2024.0
4→ System configuration

System configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected

1 Root FS skeleton (default target skeleton) --->
  (sam9x60 hobby) System hostname
  (Welcome to the Microchip SAM9X60 CURIOSITY Demo) System banner
  Passwords encoding (sha-256) --->
  Init system (systemd) --->
  [*] trigger systemd-tmpfiles during build
  (multi-user.target) The default unit systemd starts at bootup
  *** /dev management using udev (from systemd) ***
  ($(BR2_EXTERNAL_MCHP_PATH)/system/device_table.txt) Path to the p
  [ ] support extended attributes in device tables
  ↓(+)
```

2 Select and Press "Enter" key

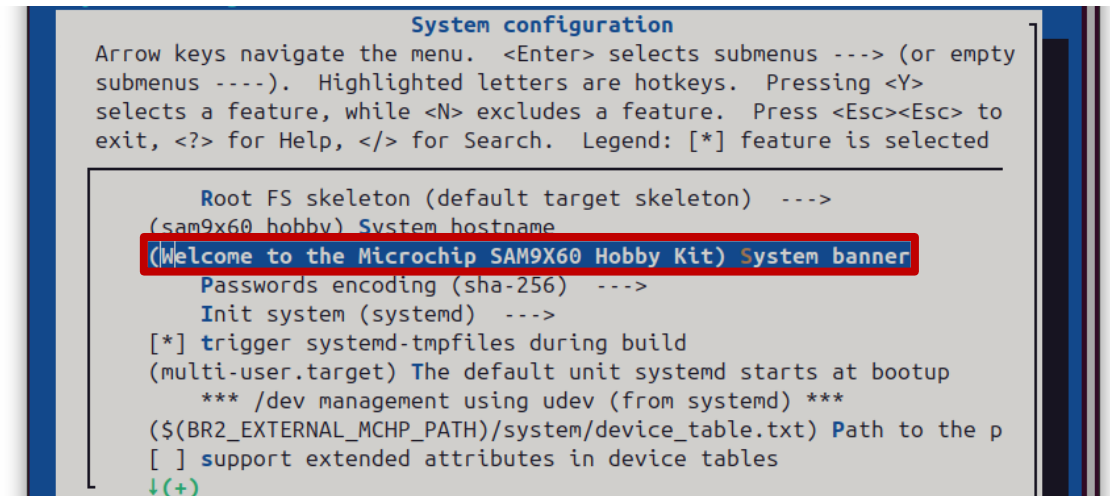
<Select> <Exit> <Help> <Save> <Load>



```
System banner
Please enter a string value. Use the <TAB> key to move from the input
field to the buttons below it.

3 Welcome to the Microchip SAM9X60 Hobby Kit

4 < Ok > < Help >
```



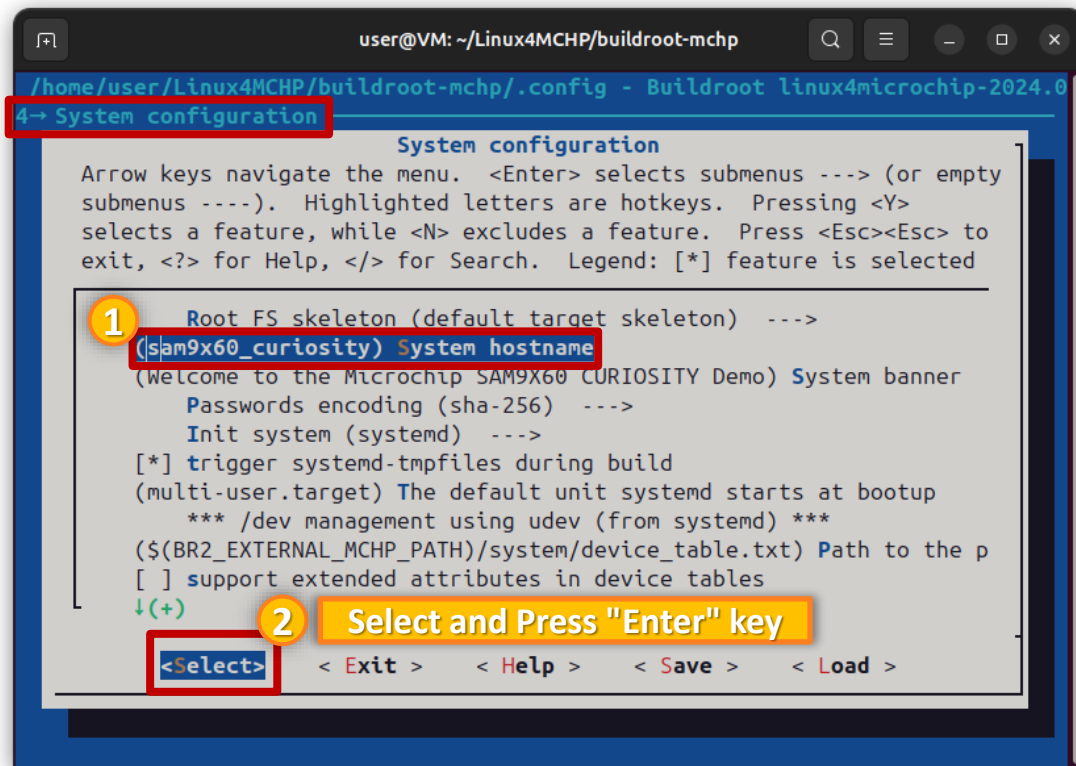
```
System configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected

1 Root FS skeleton (default target skeleton) --->
  (sam9x60 hobby) System hostname
  (Welcome to the Microchip SAM9X60 Hobby Kit) System banner
  Passwords encoding (sha-256) --->
  Init system (systemd) --->
  [*] trigger systemd-tmpfiles during build
  (multi-user.target) The default unit systemd starts at bootup
  *** /dev management using udev (from systemd) ***
  ($(BR2_EXTERNAL_MCHP_PATH)/system/device_table.txt) Path to the p
  [ ] support extended attributes in device tables
  ↓(+)
```

Lab2 – Modify the Configuration of Buildroot and Rebuild

Step 1.3

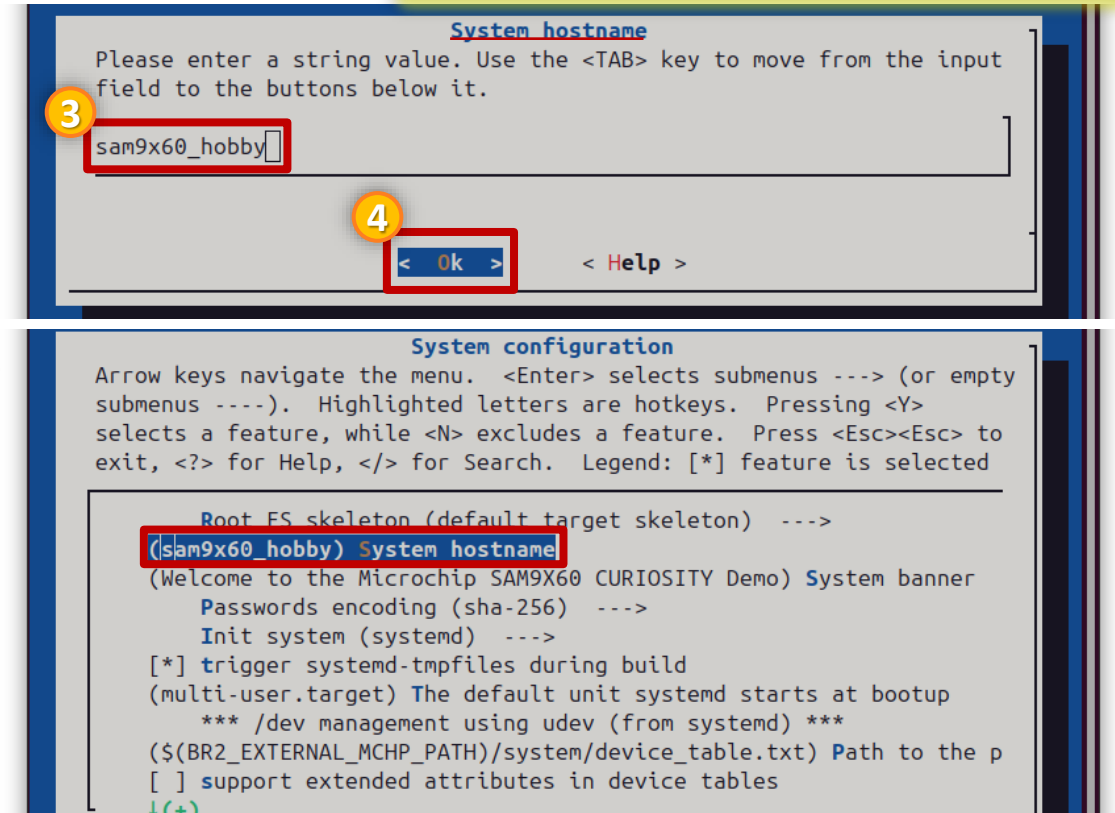
- **Modify** the **System Hostname** for SAM9X60 Hobby Kit.
(sam9x60_curiosity → sam9x60_hobby)



```
user@VM: ~/Linux4MCHP/buildroot-mchp
/home/user/Linux4MCHP/buildroot-mchp/.config - Buildroot linux4microchip-2024.0
4→ System configuration

System configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected

1 (sam9x60_curiosity) System hostname
(Welcome to the Microchip SAM9X60 CURIOSITY Demo) System banner
  Passwords encoding (sha-256) --->
  Init system (systemd) --->
  [*] trigger systemd-tmpfiles during build
  (multi-user.target) The default unit systemd starts at bootup
  *** /dev management using udev (from systemd) ***
  ($ (BR2_EXTERNAL_MCHP_PATH)/system/device_table.txt) Path to the p
  [ ] support extended attributes in device tables
  ↓(+ )
2 <Select> <Exit> <Help> <Save> <Load>
```



```
Welcome to the Microchip SAM9X60 Hobby Kit
sam9x60hobby login:

System hostname
Please enter a string value. Use the <TAB> key to move from the input
field to the buttons below it.

3 sam9x60_hobby
4 <Ok> <Help>

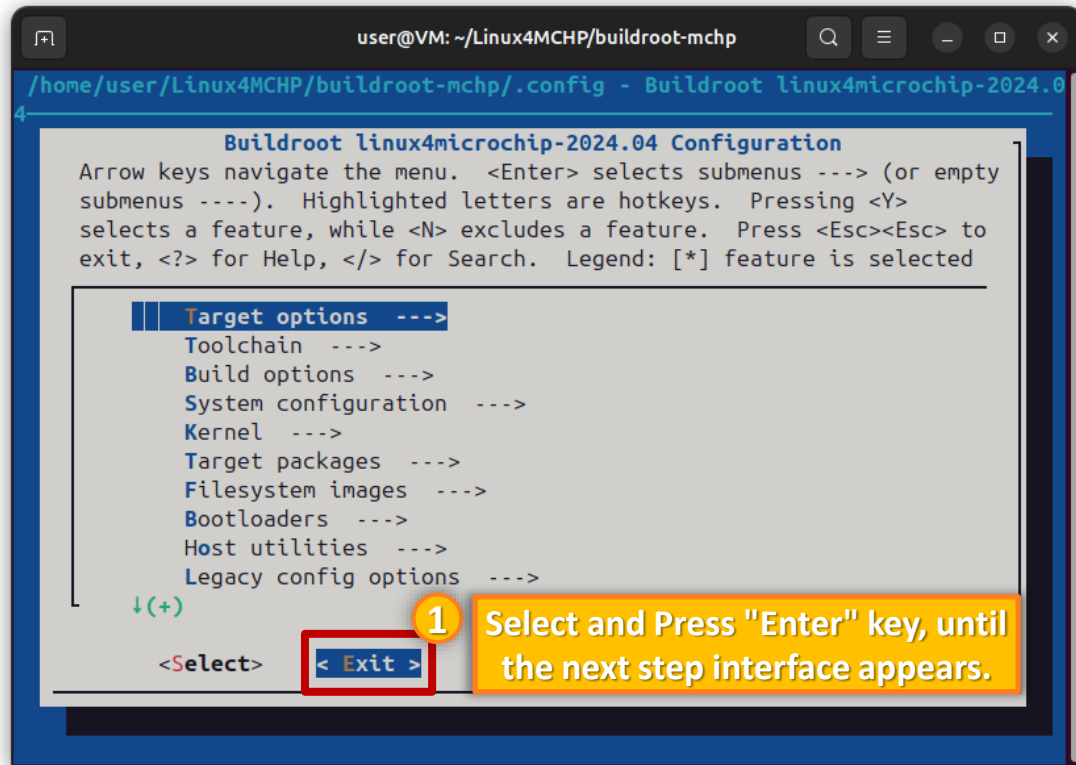
System configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected

1 (sam9x60_hobby) System hostname
(Welcome to the Microchip SAM9X60 CURIOSITY Demo) System banner
  Passwords encoding (sha-256) --->
  Init system (systemd) --->
  [*] trigger systemd-tmpfiles during build
  (multi-user.target) The default unit systemd starts at bootup
  *** /dev management using udev (from systemd) ***
  ($ (BR2_EXTERNAL_MCHP_PATH)/system/device_table.txt) Path to the p
  [ ] support extended attributes in device tables
  ↓(+ )
```

Lab2 – Modify the Configuration of Buildroot and Rebuild

Step 1.4

- **Exit** the menu of configure and **Save** new configuration to Buildroot.



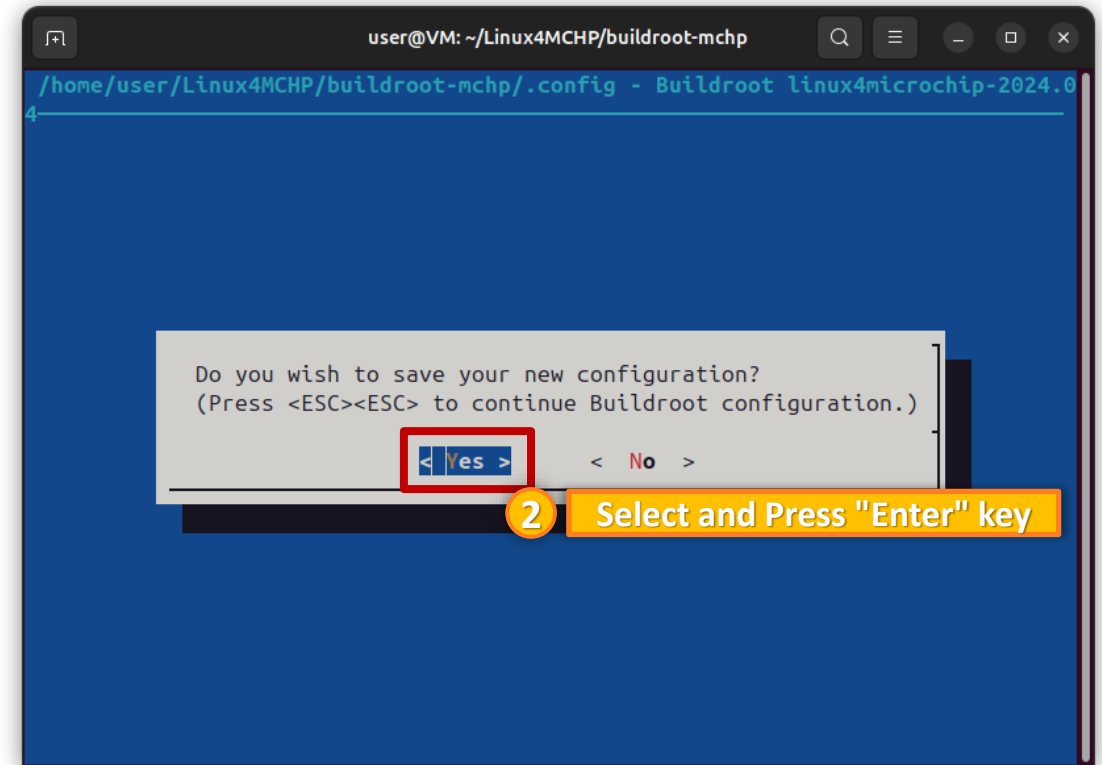
```
user@VM: ~/Linux4MCHP/buildroot-mchp
/home/user/Linux4MCHP/buildroot-mchp/.config - Buildroot linux4microchip-2024.04

Buildroot linux4microchip-2024.04 Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> selects a feature, while <N> excludes a feature. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] feature is selected

Target options --->
Toolchain --->
Build options --->
System configuration --->
Kernel --->
Target packages --->
Filesystem images --->
Bootloaders --->
Host utilities --->
Legacy config options --->

↓(+)
```

1 Select and Press "Enter" key, until the next step interface appears.



```
user@VM: ~/Linux4MCHP/buildroot-mchp
/home/user/Linux4MCHP/buildroot-mchp/.config - Buildroot linux4microchip-2024.04

Do you wish to save your new configuration?
(Press <ESC><ESC> to continue Buildroot configuration.)

< Yes > < No >
```

2 Select and Press "Enter" key

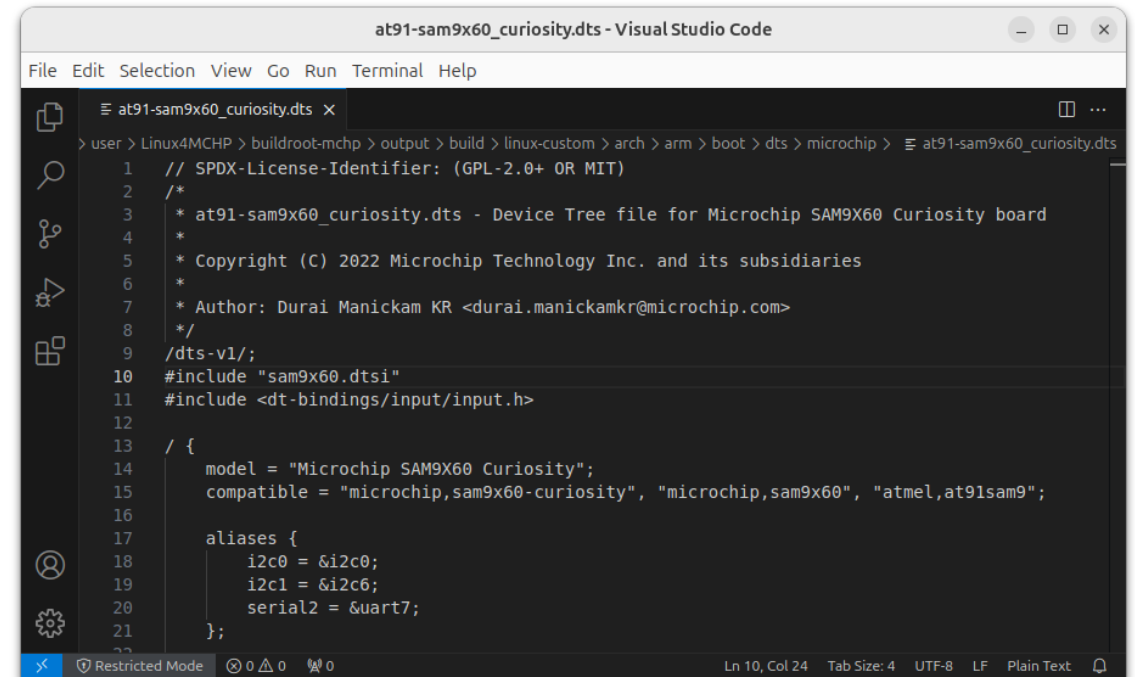
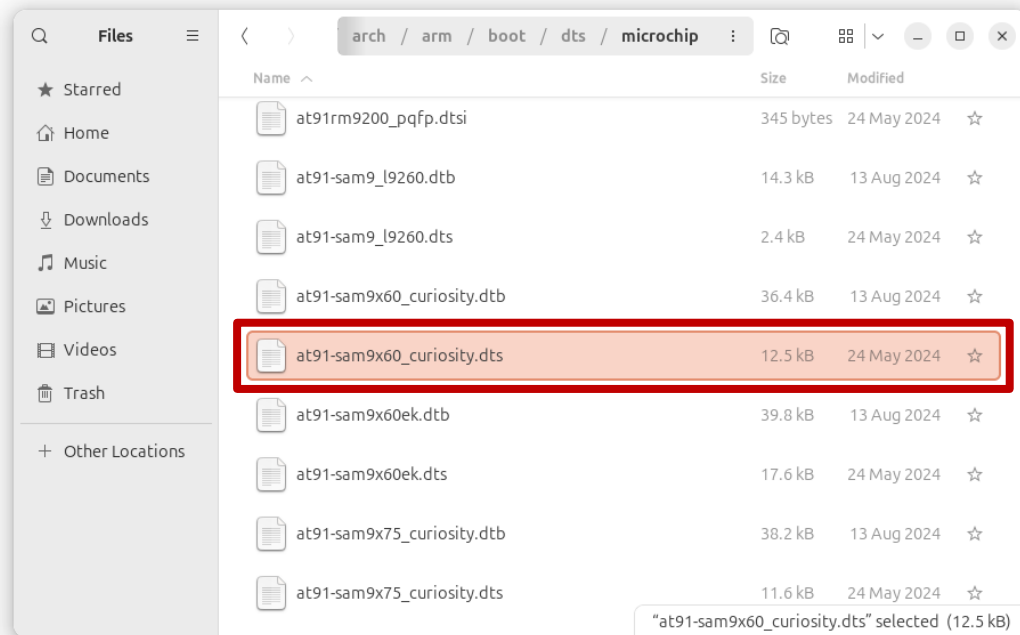
Open and Modify the Device-Tree-Source file for SAM9X60 Hobby Kit

Lab2 – Modify the Configuration of Buildroot and Rebuild

Lab2 – Modify the Configuration of Buildroot and Rebuild

Step 2.1

- **Open** the Device-Tree-Source file.
(Use GUI or Command to open the file with Visual Studio Code)

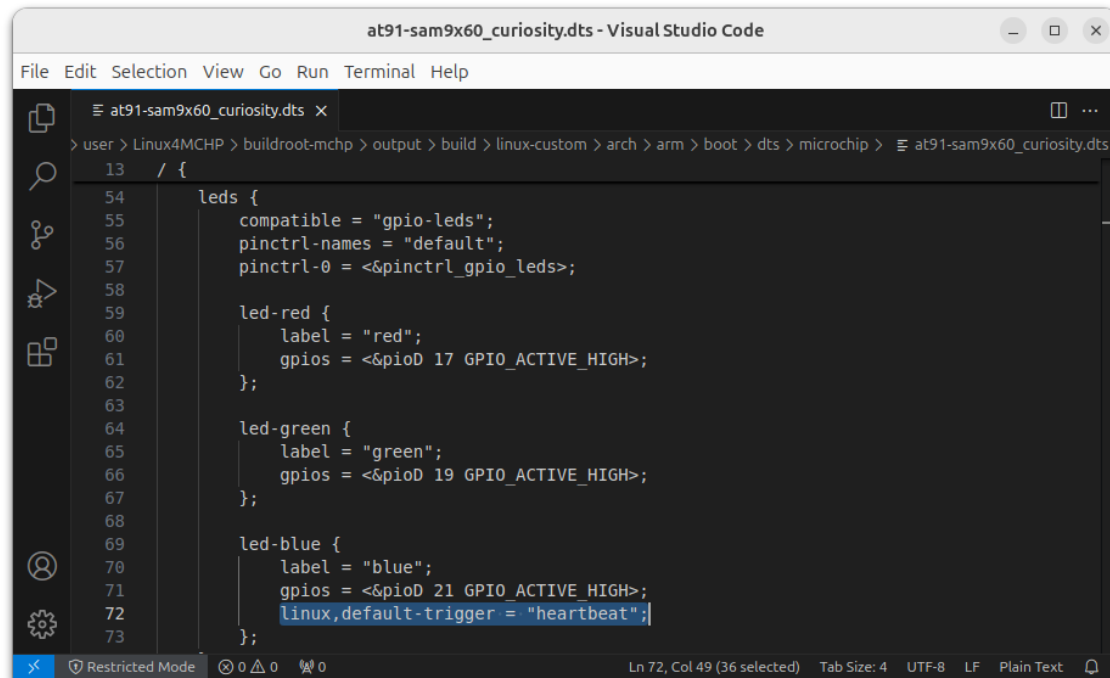


```
user@VM:~$ code ~/Linux4MCHP/buildroot-mchp/output/build/linux-custom/arch/arm/boot/dts/microchip/at91-sam9x60_curiosity.dts
```

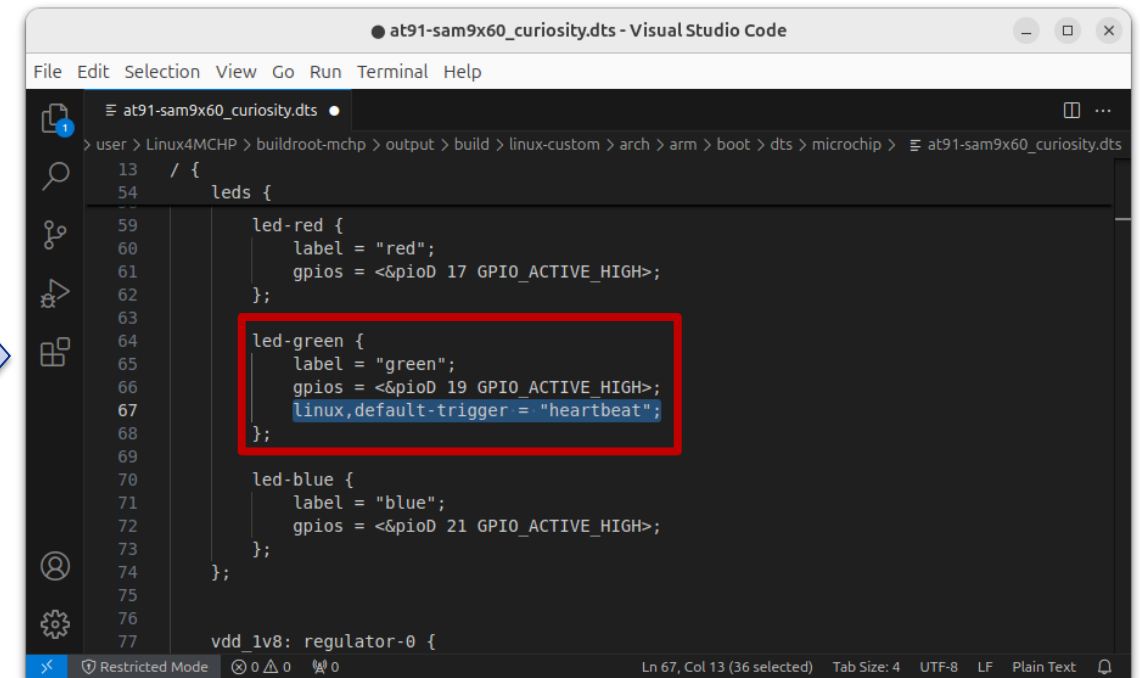
Lab2 – Modify the Configuration of Buildroot and Rebuild

Step 2.2

- **Modify** the default LED heartbeat color to Green.
(Blue → Green)



```
at91-sam9x60_curiosity.dts - Visual Studio Code
File Edit Selection View Go Run Terminal Help
> user > Linux4MCHP > buildroot-mchp > output > build > linux-custom > arch > arm > boot > dts > microchip > at91-sam9x60_curiosity.dts
13 / {
54     leds {
55         compatible = "gpio-leds";
56         pinctrl-names = "default";
57         pinctrl-0 = <&pinctrl_gpio_leds>;
58
59         led-red {
60             label = "red";
61             gpios = <&pioD 17 GPIO_ACTIVE_HIGH>;
62         };
63
64         led-green {
65             label = "green";
66             gpios = <&pioD 19 GPIO_ACTIVE_HIGH>;
67         };
68
69         led-blue {
70             label = "blue";
71             gpios = <&pioD 21 GPIO_ACTIVE_HIGH>;
72             linux,default-trigger = "heartbeat";
73         };
};
```

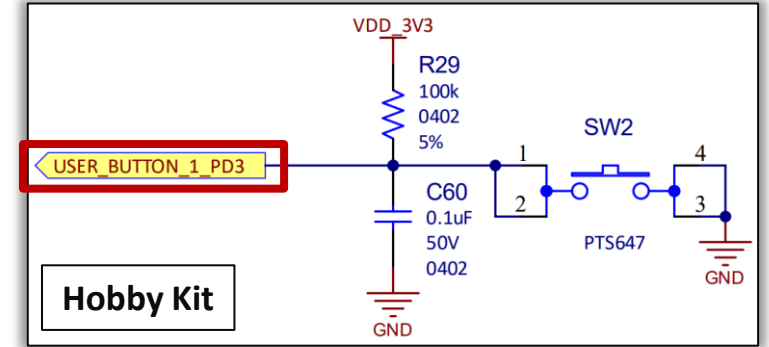


```
at91-sam9x60_curiosity.dts - Visual Studio Code
File Edit Selection View Go Run Terminal Help
> user > Linux4MCHP > buildroot-mchp > output > build > linux-custom > arch > arm > boot > dts > microchip > at91-sam9x60_curiosity.dts
13 / {
54     leds {
59         led-red {
60             label = "red";
61             gpios = <&pioD 17 GPIO_ACTIVE_HIGH>;
62         };
63
64         led-green {
65             label = "green";
66             gpios = <&pioD 19 GPIO_ACTIVE_HIGH>;
67             linux,default-trigger = "heartbeat";
68         };
69
70         led-blue {
71             label = "blue";
72             gpios = <&pioD 21 GPIO_ACTIVE_HIGH>;
73         };
74     };
75
76     vdd_lvs: regulator-0 {
77
```

Lab2 – Modify the Configuration of Buildroot and Rebuild

Step 2.3

- **Modify** the User-Button pin for Hobby Kit.
(PA29 → **PD3**)



```
at91-sam9x60_curiosity.dts - Visual Studio Code
File Edit Selection View Go Run Terminal Help
> user > Linux4MCHP > buildroot-mchp > output > build > linux-custom > arch > arm > boot > dts > microchip > at91-sam9x60_curiosity.dts
13 / {
41     gpio-keys {
42         compatible = "gpio-keys";
43         pinctrl-names = "default";
44         pinctrl-0 = <&pinctrl_key_gpio_default>;
45     };
46     button-user {
47         label = "PB_USER";
48         gpios = <&pioA 29 GPIO_ACTIVE_LOW>;
49         linux,code = <KEY_PROG1>;
50         wakeup-source;
51     };
52 };
```



```
at91-sam9x60_curiosity.dts - Visual Studio Code
File Edit Selection View Go Run Terminal Help
> user > Linux4MCHP > buildroot-mchp > output > build > linux-custom > arch > arm > boot > dts > microchip > at91-sam9x60_curiosity.dts
13 / {
41     gpio-keys {
42         compatible = "gpio-keys";
43         pinctrl-names = "default";
44         pinctrl-0 = <&pinctrl_key_gpio_default>;
45     };
46     button-user {
47         label = "PB_USER";
48         gpios = <&pioD 3 GPIO_ACTIVE_LOW>;
49         linux,code = <KEY_PROG1>;
50         wakeup-source;
51     };
52 };
```

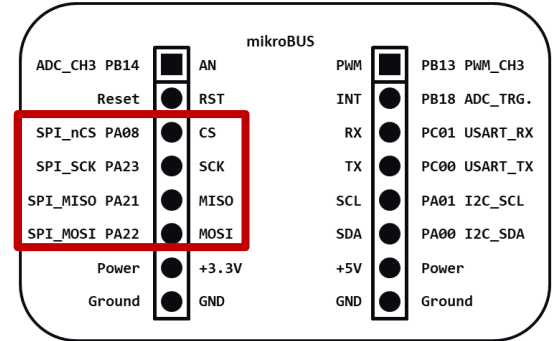
```
at91-sam9x60_curiosity.dts - Visual Studio Code
File Edit Selection View Go Run Terminal Help
> user > Linux4MCHP > buildroot-mchp > output > build > linux-custom > arch > arm > boot > dts > microchip > at91-sam9x60_curiosity.dts
257 &pinctrl {
333     gpio-keys {
334         pinctrl key_gpio_default: pinctrl-key-gpio {
335             atmel,pins = <AT91_PIOA 29 AT91_PERIPH_GPIO AT91_PINCTRL_NONE>;
336         };
337     };
338 };
```

```
at91-sam9x60_curiosity.dts - Visual Studio Code
File Edit Selection View Go Run Terminal Help
> user > Linux4MCHP > buildroot-mchp > output > build > linux-custom > arch > arm > boot > dts > microchip > at91-sam9x60_curiosity.dts
257 &pinctrl {
333     gpio-keys {
334         pinctrl key_gpio_default: pinctrl-key-gpio {
335             atmel,pins = <AT91_PIOD 3 AT91_PERIPH_GPIO AT91_PINCTRL_NONE>;
336         };
337     };
338 };
```

Lab2 – Modify the Configuration of Buildroot and Rebuild

Step 2.4

- **Modify** the Device-Tree-Source to create SPI node.
(**Create** the nodes for **Flexcom 5**, **SPI 5** and **SPI Device 0**)



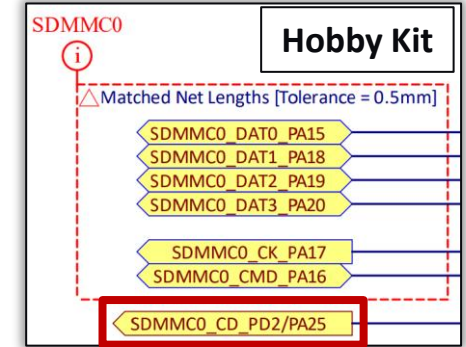
```
at91-sam9x60_curiosity.dts - Visual Studio Code
File Edit Selection View Go Run Terminal Help
> user > Linux4MCHP > buildroot-mchp > output > build > linux-custom > arch > arm > boot > dts > microchip > at91-sam9x60_curiosity.dts
214 };
215
216 &flx5 {
217     atmel,flexcom-mode = <ATMEL_FLEXCOM_MODE_SPI>;
218     status = "okay";
219
220     spi5: spi@400 {
221         dmas = <0>, <0>;
222         pinctrl-names = "default";
223         pinctrl-0 = <&pinctrl_flx5 default>;
224         status = "okay";
225
226         spidev@0 {
227             compatible = "rohm,dh2228fv";
228             reg = <0>;
229             spi-max-frequency = <1000000>;
230         };
231     };
232 };
233
234 &flx6 {
```

```
at91-sam9x60_curiosity.dts - Visual Studio Code
File Edit Selection View Go Run Terminal Help
> user > Linux4MCHP > buildroot-mchp > output > build > linux-custom > arch > arm > boot > dts > microchip > at91-sam9x60_curiosity.dts
275 &pinctrl {
276
277     flexcom {
278         pinctrl_flx0_default: flx0-twi {
279             atmel,pins =
280                 <AT91_PIOA 0 AT91_PERIPH_A AT91_PINCTRL_PULL_UP
281                 AT91_PIOA 1 AT91_PERIPH_A AT91_PINCTRL_PULL_UP>;
282         };
283
284         pinctrl_flx5_default: flx5_spi {
285             atmel,pins =
286                 <AT91_PIOA 22 AT91_PERIPH_B AT91_PINCTRL_PULL_UP
287                 AT91_PIOA 21 AT91_PERIPH_B AT91_PINCTRL_PULL_UP
288                 AT91_PIOA 23 AT91_PERIPH_B AT91_PINCTRL_PULL_UP
289                 AT91_PIOA 8 AT91_PERIPH_B AT91_PINCTRL_PULL_UP>;
290         };
291
292         pinctrl_flx6_default: flx6-twi {
293             atmel,pins =
294                 <AT91_PIOA 30 AT91_PERIPH_A AT91_PINCTRL_PULL_UP
295                 AT91_PIOA 31 AT91_PERIPH_A AT91_PINCTRL_PULL_UP>;
296         };
297     };
298 }
```

Lab2 – Modify the Configuration of Buildroot and Rebuild

Step 2.5

- **Modify** the Device-Tree-Source for SD Card Detect.
(PA25 → PD2)



```
at91-sam9x60_curiosity.dts - Visual Studio Code
File Edit Selection View Go Run Terminal Help
> user > Linux4MCHP > buildroot-mchp > output > build > linux-custom > arch > arm > boot > dts > microchip > at91-sam9x60_curiosity.dts
449 &sdmmc0 {
450     bus-width = <4>;
451     pinctrl-names = "default";
452     pinctrl-0 = <&pinctrl_sdmmc0 default &pinctrl_sdmmc0_cd>;
453     cd-gpios = <&pioA 25 GPIO_ACTIVE_LOW>;
454     disable-wp;
455     status = "okay";
456 };
457
```

```
257 &pinctrl {
396     sdmmc0 {
397         pinctrl_sdmmc0_default: sdmmc0 {
398             atmel,pins =
399                 <AT91_PIOA 17 AT91_PERIPH_A (AT91_PINCTRL_DRIVE_STRENGTH_HI) /*
400                 AT91_PIOA 16 AT91_PERIPH_A (AT91_PINCTRL_PULL_UP | AT91_PINCTRL_DRIVE_STRENGT
401                 AT91_PIOA 15 AT91_PERIPH_A (AT91_PINCTRL_PULL_UP | AT91_PINCTRL_DRIVE_STRENGT
402                 AT91_PIOA 18 AT91_PERIPH_A (AT91_PINCTRL_PULL_UP | AT91_PINCTRL_DRIVE_STRENGT
403                 AT91_PIOA 19 AT91_PERIPH_A (AT91_PINCTRL_PULL_UP | AT91_PINCTRL_DRIVE_STRENGT
404                 AT91_PIOA 20 AT91_PERIPH_A (AT91_PINCTRL_PULL_UP | AT91_PINCTRL_DRIVE_STRENGT
405             };
406         pinctrl_sdmmc0_cd: sdmmc0-cd {
407             atmel,pins =
408                 <AT91_PIOA 25 AT91_PERIPH_GPIO AT91_PINCTRL_NONE>;
409             };
410         };
411     };

```



```
at91-sam9x60_curiosity.dts - Visual Studio Code
File Edit Selection View Go Run Terminal Help
> user > Linux4MCHP > buildroot-mchp > output > build > linux-custom > arch > arm > boot > dts > microchip > at91-sam9x60_curiosity.dts
449 &sdmmc0 {
450     bus-width = <4>;
451     pinctrl-names = "default";
452     pinctrl-0 = <&pinctrl_sdmmc0 default &pinctrl_sdmmc0_cd>;
453     cd-gpios = <&pioD 2 GPIO_ACTIVE_LOW>;
454     disable-wp;
455     status = "okay";
456 };
457
```

```
257 &pinctrl {
396     sdmmc0 {
397         pinctrl_sdmmc0_default: sdmmc0 {
398             atmel,pins =
399                 <AT91_PIOA 17 AT91_PERIPH_A (AT91_PINCTRL_DRIVE_STRENGTH_HI) /*
400                 AT91_PIOA 16 AT91_PERIPH_A (AT91_PINCTRL_PULL_UP | AT91_PINCTRL_DRIVE_STRENGT
401                 AT91_PIOA 15 AT91_PERIPH_A (AT91_PINCTRL_PULL_UP | AT91_PINCTRL_DRIVE_STRENGT
402                 AT91_PIOA 18 AT91_PERIPH_A (AT91_PINCTRL_PULL_UP | AT91_PINCTRL_DRIVE_STRENGT
403                 AT91_PIOA 19 AT91_PERIPH_A (AT91_PINCTRL_PULL_UP | AT91_PINCTRL_DRIVE_STRENGT
404                 AT91_PIOA 20 AT91_PERIPH_A (AT91_PINCTRL_PULL_UP | AT91_PINCTRL_DRIVE_STRENGT
405             };
406         pinctrl_sdmmc0_cd: sdmmc0-cd {
407             atmel,pins =
408                 <AT91_PIOD 2 AT91_PERIPH_GPIO AT91_PINCTRL_NONE>;
409             };
410         };
411     };

```

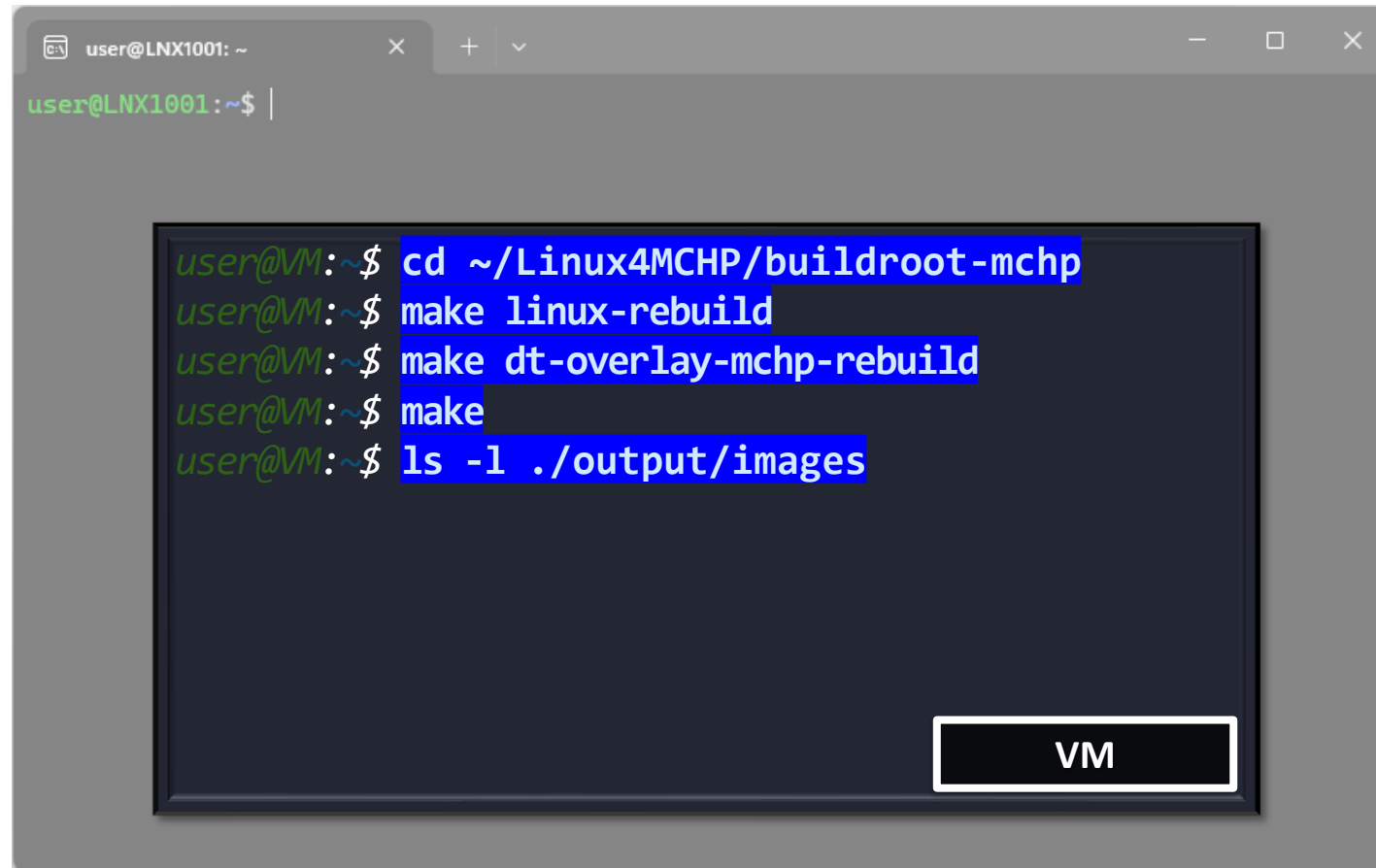
Rebuild the Buildroot and Prepare a SD Card with Embedded Linux OS

Lab2 – Modify the Configuration of Buildroot and Rebuild

Lab2 – Modify the Configuration of Buildroot and Rebuild

Step 3

- **Rebuild** the Buildroot for Hobby Kit.



A terminal window titled 'user@LNX1001: ~' with standard window controls. The prompt is 'user@LNX1001:~\$'. A dark rectangular box is overlaid on the terminal, containing a sequence of commands for rebuilding the buildroot. The commands are: 'cd ~/Linux4MCHP/buildroot-mchp', 'make linux-rebuild', 'make dt-overlay-mchp-rebuild', 'make', and 'ls -l ./output/images'. Each command is preceded by a green prompt 'user@VM:~\$'. The text in the box is highlighted in blue. A white button labeled 'VM' is located at the bottom right of the dark box.

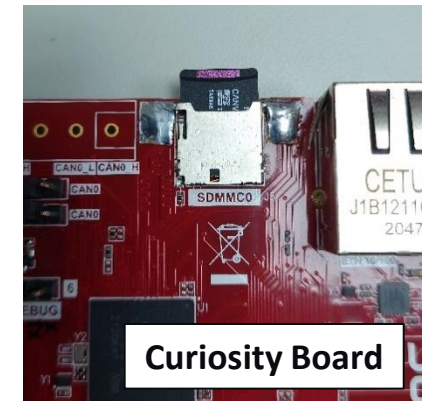
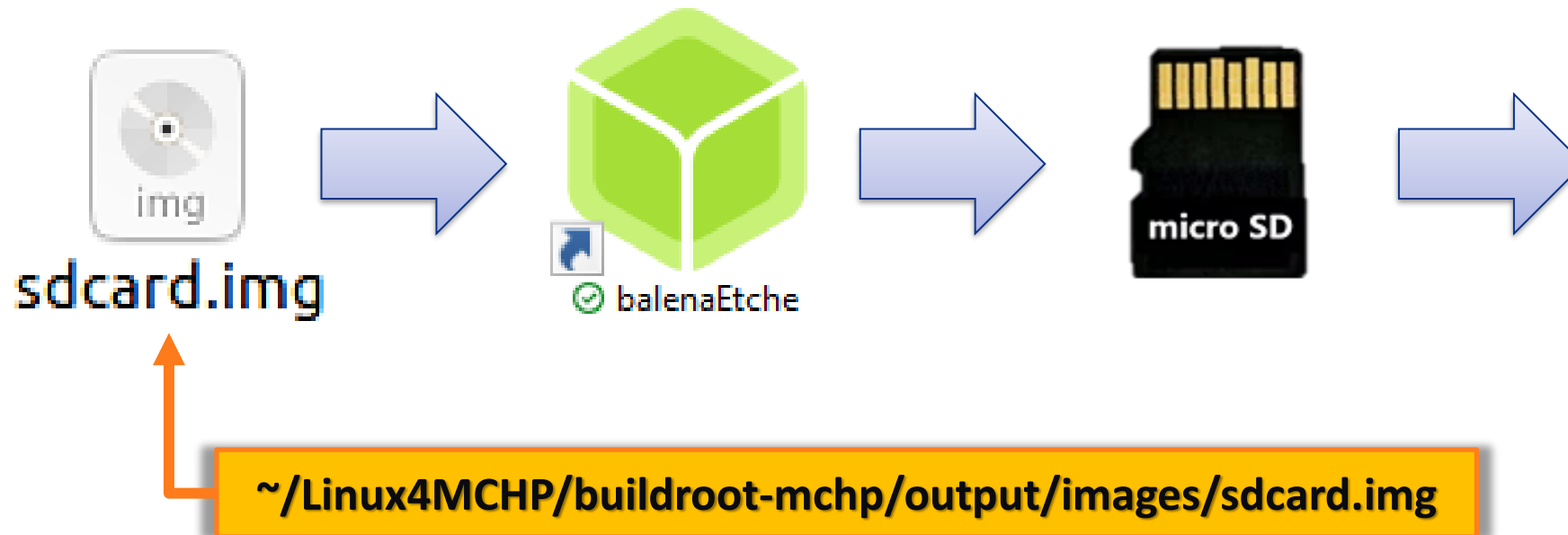
```
user@VM:~$ cd ~/Linux4MCHP/buildroot-mchp
user@VM:~$ make linux-rebuild
user@VM:~$ make dt-overlay-mchp-rebuild
user@VM:~$ make
user@VM:~$ ls -l ./output/images
```

VM

Lab2 – Modify the Configuration of Buildroot and Rebuild

Step 4

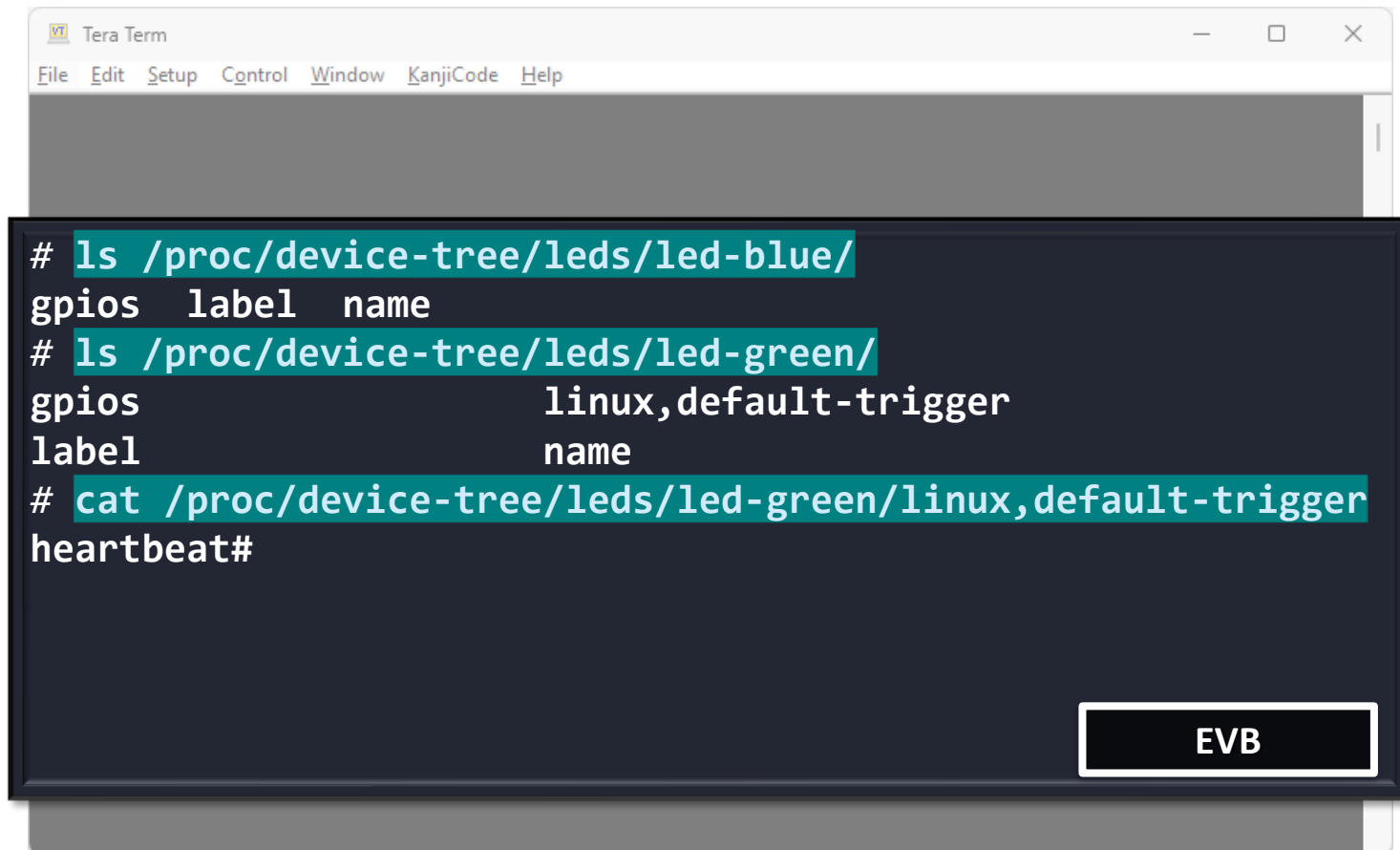
- Prepare a SD Card with Embedded Linux OS and mount it to EVB.
(Refer to the appendix to complete the Flash SD Card)



Lab2 – Modify the Configuration of Buildroot and Rebuild

Step 5.1

- **Check** the modified part in the Embedded Linux System.



A screenshot of a Tera Term terminal window. The window has a menu bar with 'File', 'Edit', 'Setup', 'Control', 'Window', 'KanjiCode', and 'Help'. The terminal content shows the following commands and their outputs:

```
# ls /proc/device-tree/leds/led-blue/
gprios label name
# ls /proc/device-tree/leds/led-green/
gprios                linux,default-trigger
label                name
# cat /proc/device-tree/leds/led-green/linux,default-trigger
heartbeat#
```

At the bottom right of the terminal window, there is a button labeled 'EVB'.

```
led-red {
    label = "red";
    gprios = <&pioD 17 GPIO_ACTIVE_HIGH>;
};

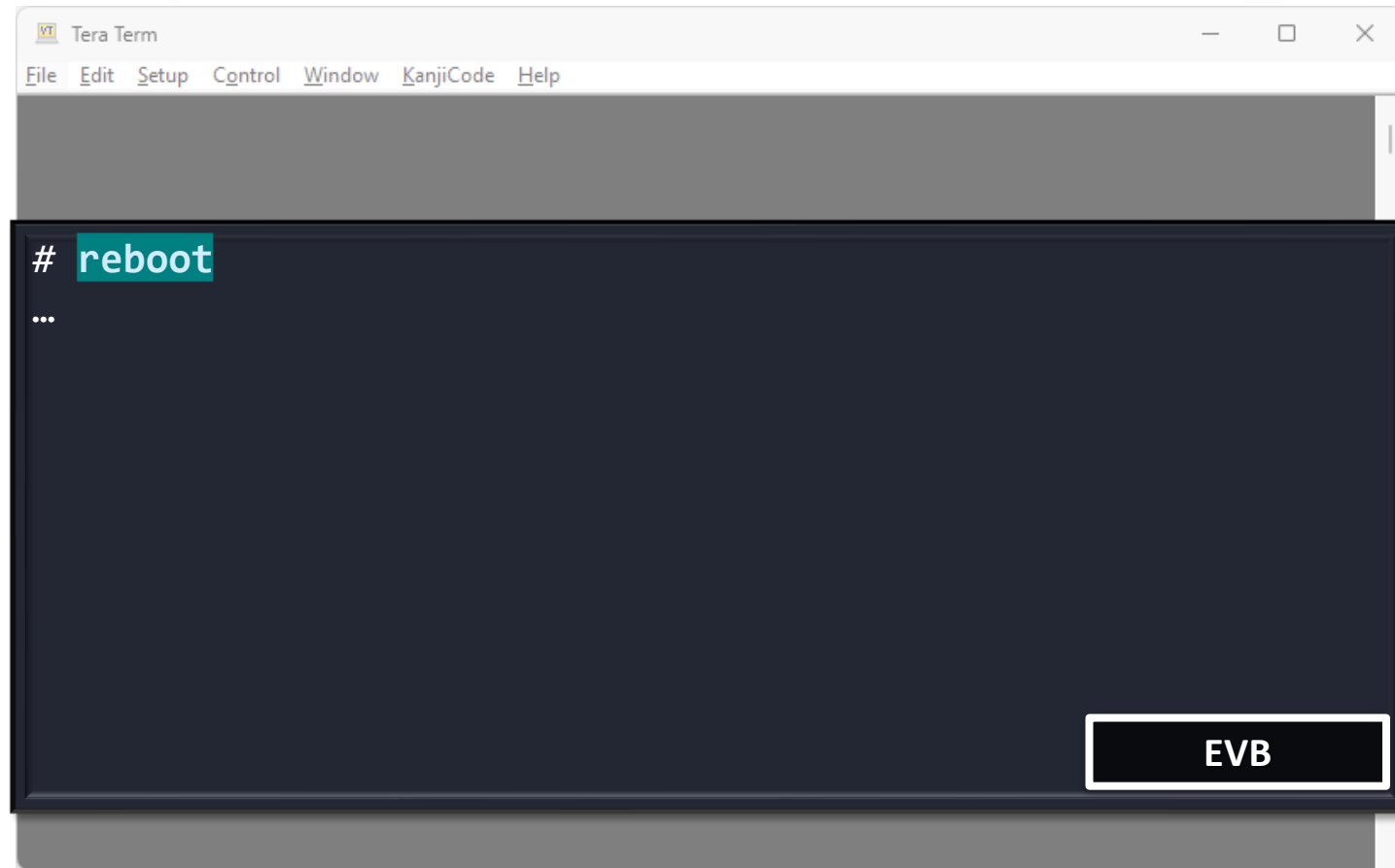
led-green {
    label = "green";
    gprios = <&pioD 19 GPIO_ACTIVE_HIGH>;
    linux,default-trigger = "heartbeat";
};

led-blue {
    label = "blue";
    gprios = <&pioD 21 GPIO_ACTIVE_HIGH>;
};
```

Lab2 – Modify the Configuration of Buildroot and Rebuild

Step 5.2

- **Check** the modified part in the Embedded Linux System.

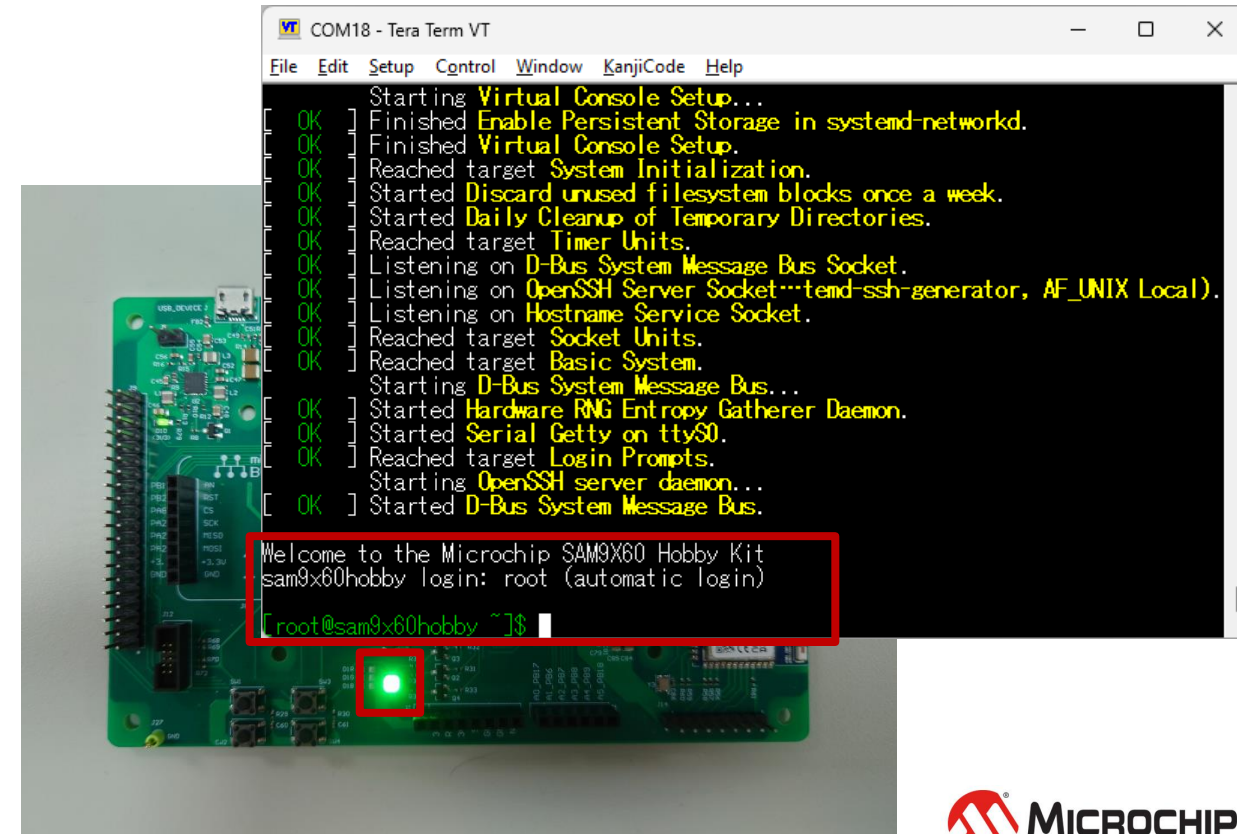


Lab2 – Modify the Configuration of Buildroot and Rebuild

Result

- Buildroot output is stored in a single directory, "**output/**". This directory contains several subdirectories, you can find newly modified files in here.
- After login the Embedded Linux system, you can find the modified information in the system and see the green led blinking.

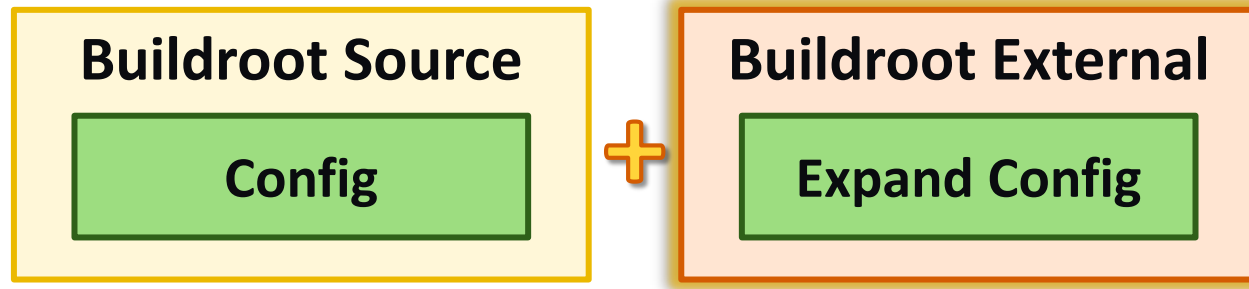
```
user@LNX1001: ~/Linux4MCHP x + v
user@LNX1001:~/Linux4MCHP/buildroot-mchp$ ls output/
build host images staging target
user@LNX1001:~/Linux4MCHP/buildroot-mchp$ ls -l output/images/
total 1310552
-rwxr-xr-x 1 user user 37072 Sep 9 03:36 at91-sam9x60_curiosity.dtb
-rwxr-xr-x 1 user user 18929 Sep 9 03:37 at91bootstrap.bin
-rwxr-xr-x 1 user user 18929 Sep 9 03:37 boot.bin
-rw-r--r-- 1 user user 16777216 Sep 9 04:02 boot.vfat
-rw-r--r-- 1 user user 943718400 Sep 9 04:01 rootfs.ext2
lrwxrwxrwx 1 user user 11 Sep 9 04:01 rootfs.ext4 -> rootfs.ext2
-rw-r--r-- 1 user user 425656320 Sep 9 04:02 rootfs.tar
-rwxr-xr-x 1 user user 18929 Sep 9 03:37 sam9x60-sdcardboot-uboot-4.0.11.bin
-rw-r--r-- 1 user user 5631664 Sep 9 03:38 sam9x60_curiosity.itb
-rw-r--r-- 1 user user 2053 Sep 9 03:38 sam9x60_curiosity.its
-rw-r--r-- 1 user user 4868 Sep 9 03:38 sam9x60_curiosity_pda5.dtbo
-rw-r--r-- 1 user user 1534 Sep 9 03:38 sam9x60_curiosity_wilc3000.dtbo
-rw-r--r-- 1 user user 961544192 Sep 9 04:02 sdcard.img
-rw-r--r-- 1 user user 458952 Sep 9 03:31 u-boot.bin
-rwxr-xr-x 1 user user 16384 Sep 9 03:31 uboot-env.bin
-rw-r--r-- 1 user user 5585520 Sep 9 03:36 zImage
user@LNX1001:~/Linux4MCHP/buildroot-mchp$ |
```



Customize External Tree Project for Buildroot

Buildroot External Tree

- Buildroot External Tree refers to a separate directory structure that contains custom packages, configurations, or patches that you want to incorporate into your main Buildroot project.
- This provides **a flexible way** to manage and maintain additional components outside of the standard Buildroot package structure.
- This method has the advantage of **keeping project-specific files separated from the main Buildroot source tree** while integrating them seamlessly into the build process. This method is called the **br2_external mechanism**.



Microchip Buildroot External

- Microchip Buildroot External includes Microchip packages, patches, setup, and configuration to work with Microchip provided software that is not included in mainline buildroot.
- We can find Buildroot External for Microchip SoC at "[linux4michip/buildroot-external-microchip](https://github.com/linux4michip/buildroot-external-microchip)" from the GitHub Repository.
- We can use simple command to build Buildroot Source with Microchip Buildroot External :

```
user@VM:~$ cd ~/Linux4MCHP/buildroot-mchp/  
user@VM:~$ BR2_EXTERNAL=../buildroot-external-microchip/ make sam9x60_curiosity_graphics_defconfig  
user@VM:~$ make
```

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

- In this Lab we will try to **create an image suitable for the Hobby Kit**, we will modify the **BR2-External Project** to complete it.
- In this way, we can build a Linux OS image suitable for Hobby Kit without modify the Buildroot source, and easily built again.
- Follow the steps below to complete the above requirements :
 1. Create configurations for Hobby Kit in BR2-External Project.
 2. Modify the configuration of BR2-External Project.
 - a. Change the name of OS Image and Image-Tree-Blob for Hobby Kit. (Modify the genimage.cfg)
 - b. Change the uboot-env.txt content to Image-Tree-Blob for Hobby Kit.
 3. Create a Device-Tree-Source for Hobby Kit in BR2-External Project.
 4. Modify the configuration of Buildroot. (make menuconfig)
 5. Save the configuration to the BR2-External Project.
 6. Make a difference patch of Image-Tree-Blob (.itb) for Hobby Kit.
 - a. Reset the dt-overlay-mchp in Buildroot.
 - b. Prepared the materials in a Patches directory.
 - c. Modify the Image-Tree-Source (.its) in dt-overlay-mchp directory.
 - d. Modify "Makefile" for Hobby Kit in dt-overlay-mchp directory.
 - e. Make a difference patch for BR2-External Project.
 7. Rebuild the Buildroot with BR2-External Project for Hobby Kit.
 8. Flash OS images to SD card.
 9. Check the modified part in the Embedded Linux System.

Let's go!

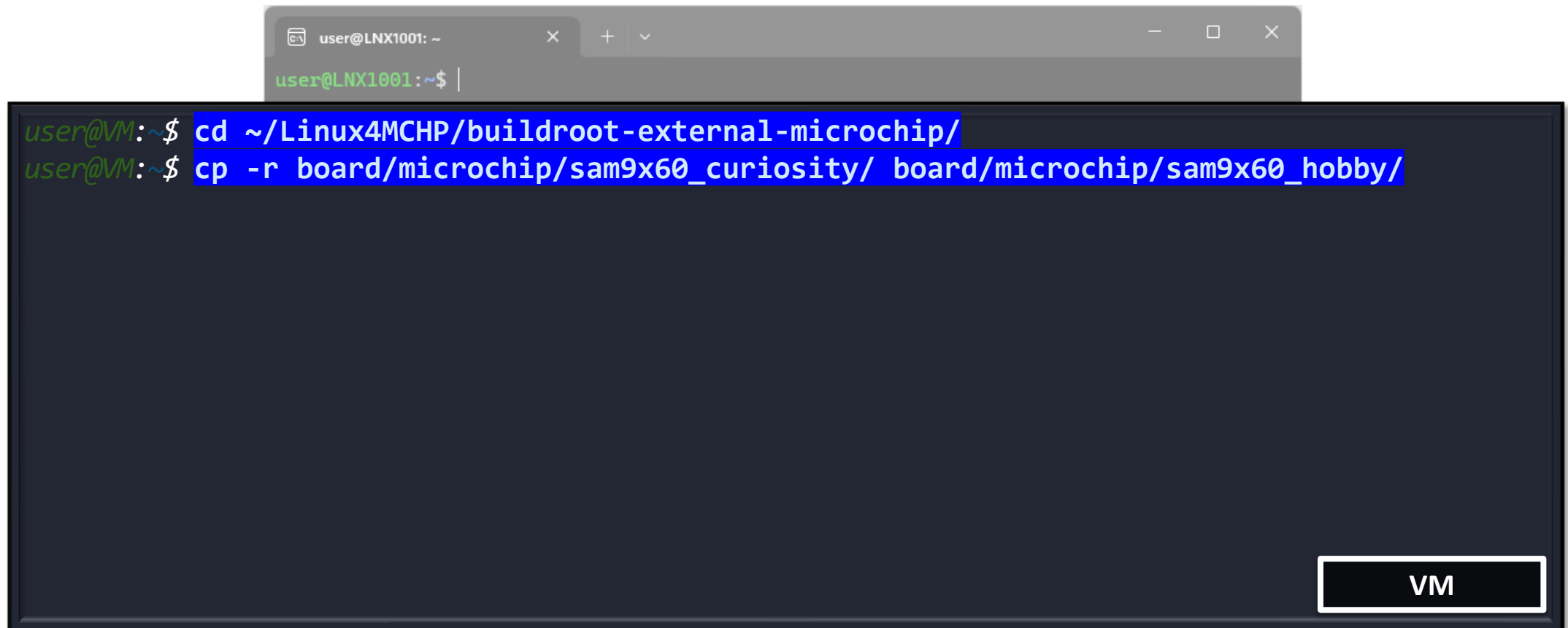
Create configurations for Hobby Kit in BR2-External Project

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 1

- **Create a SAM9X60 Hobby Kit directory** in **Buildroot External** for Microchip SoC.
(based on Curiosity Board's archives)



A terminal window with a dark background and light green text. The window title bar shows 'user@LNX1001: ~'. The terminal content shows two commands being executed at the 'user@VM:~\$' prompt. The first command is 'cd ~/Linux4MCHP/buildroot-external-microchip/' and the second is 'cp -r board/microchip/sam9x60_curiosity/ board/microchip/sam9x60_hobby/'. The second command and its output are highlighted in blue. A 'VM' button is visible in the bottom right corner of the terminal window.

```
user@VM:~$ cd ~/Linux4MCHP/buildroot-external-microchip/
user@VM:~$ cp -r board/microchip/sam9x60_curiosity/ board/microchip/sam9x60_hobby/
```

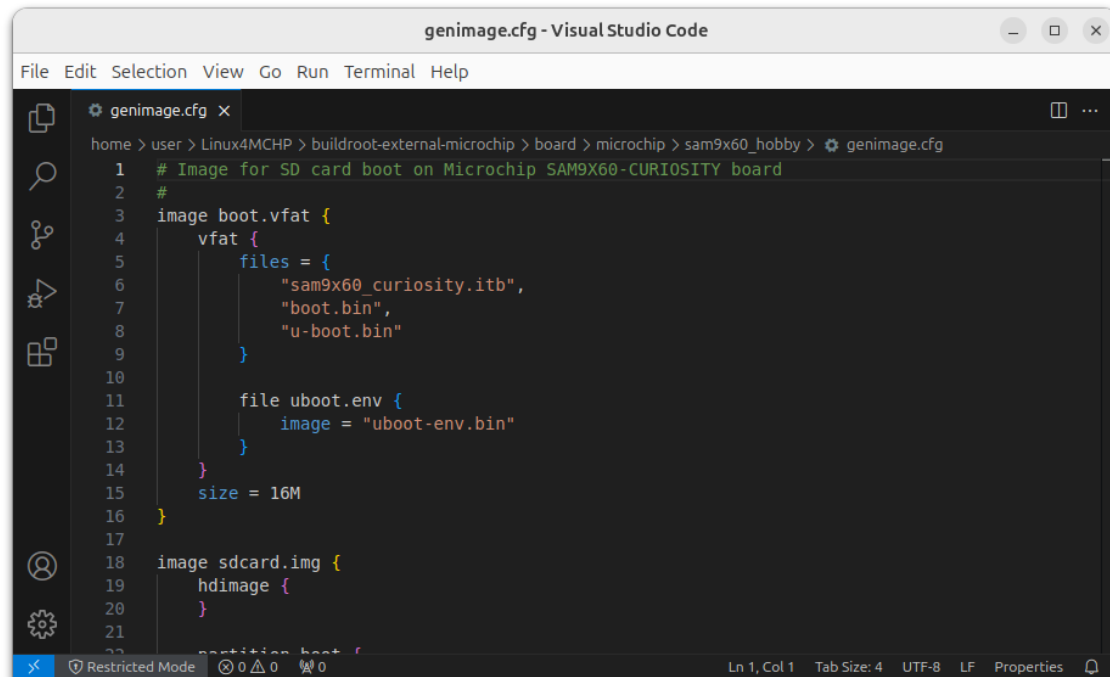
Modify the configuration of BR2-External Project

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

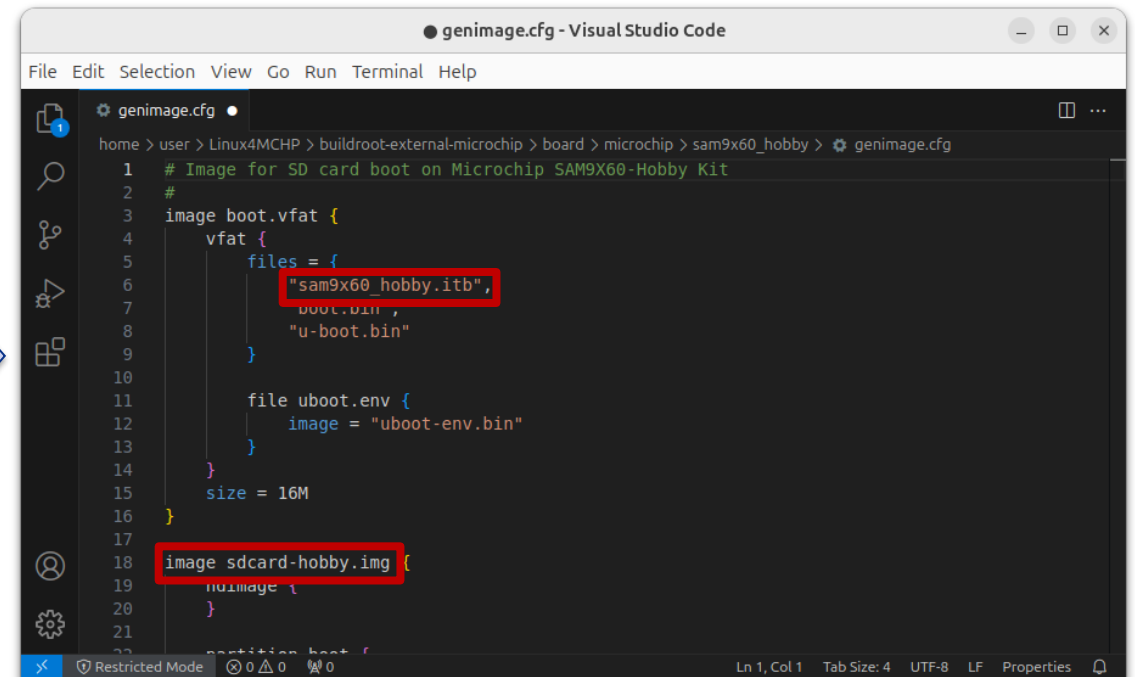
Step 2.1

- **Modify "genimage.cfg"** to create an image for Hobby Kit. (Prepare to generate sdcard-hobby.img)
(Use GUI or Command to open the file with Visual Studio Code)



```
genimage.cfg - Visual Studio Code
File Edit Selection View Go Run Terminal Help

home > user > Linux4MCHP > buildroot-external-microchip > board > microchip > sam9x60_hobby > genimage.cfg
1 # Image for SD card boot on Microchip SAM9X60-CURIOSITY board
2 #
3 image boot.vfat {
4     vfat {
5         files = {
6             "sam9x60_curiosity.itb",
7             "boot.bin",
8             "u-boot.bin"
9         }
10
11         file uboot.env {
12             image = "uboot-env.bin"
13         }
14     }
15     size = 16M
16 }
17
18 image sdcard.img {
19     himage {
20     }
21 }
```



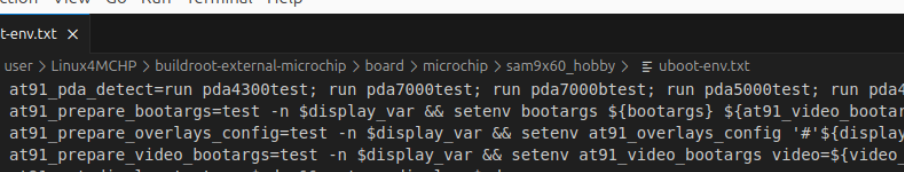
```
genimage.cfg - Visual Studio Code
File Edit Selection View Go Run Terminal Help

home > user > Linux4MCHP > buildroot-external-microchip > board > microchip > sam9x60_hobby > genimage.cfg
1 # Image for SD card boot on Microchip SAM9X60-Hobby Kit
2 #
3 image boot.vfat {
4     vfat {
5         files = {
6             "sam9x60_hobby.itb",
7             "boot.bin",
8             "u-boot.bin"
9         }
10
11         file uboot.env {
12             image = "uboot-env.bin"
13         }
14     }
15     size = 16M
16 }
17
18 image sdcard-hobby.img {
19     himage {
20     }
21 }
```

```
user@VM:~$ code ~/Linux4MCHP/buildroot-external-microchip/board/microchip/sam9x60_hobby/genimage.cfg
```

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

- **Change the "uboot-env.txt" content for Hobby Kit.**
(Use GUI or Command to open the file with Visual Studio Code)



```
uboot-env.txt - Visual Studio Code
File Edit Selection View Go Run Terminal Help

home > user > Linux4MCHP > buildroot-external-microchip > board > microchip > sam9x60_hobby > uboot-env.txt
1 at91_pda_detect=run pda4300test; run pda7000test; run pda7000btest; run pda5000test; run pda4301test; run p
2 at91_prepare_bootargs=test -n $display_var && setenv bootargs ${bootargs} ${at91_video_bootargs}
3 at91_prepare_overlays_config=test -n $display_var && setenv at91_overlays_config '#${display_var}
4 at91_prepare_video_bootargs=test -n $display_var && setenv at91_video_bootargs video=${video_mode}
5 at91_set_display=test -n $pda && setenv display $pda
6 video=${video_mode}
7 board_name=sam9x60_curiosity
8 bootargs=console=ttyS0,115200 root=/dev/mmcblk0p2 rw rootwait cma=16m rootfstype=ext4 atmel.pm_modes=standb
9 bootcmd=run at91_set_display; run at91_pda_detect; run at91_prepare_video_bootargs; run at91_prepare_bootar
10 bootcmd_boot=fatload mmc 0:1 ${loadaddr} ${board_name}.itb; bootm ${loadaddr}#kernel_dtb${at91_overlays_con
11 bootdelay=1
12 loadaddr=0x21000000
13 pda4300test=test -n $display && test $display = 4300 && setenv display_var 'pda4' && setenv video_mode ${vi
14 pda4301test=test -n $display && test $display = 4301 && setenv display_var 'pda4' && setenv video_mode ${vi
15 pda4301btest=test -n $display && test $display = 4301B && setenv display_var 'pda4' && setenv video_mode ${vi
16 pda5000test=test -n $display && test $display = 5000 && setenv display_var 'pda5' && setenv video_mode ${vi
17 pda7000btest=test -n $display && test $display = 7000B && setenv display_var 'pda7b' && setenv video_mode $
18 pda7000test=test -n $display && test $display = 7000 && setenv display_var 'pda7' && setenv video_mode ${vi
19 stderr=serial@fffee00
20 stdout=serial@fffee00
21 stdout=serial@fffee00
22 video_mode_pda4=Unknown-1:480x272-16
23 video_mode_pda5=Unknown-1:800x480-16
```

```
user@VM:~$ code ~/Linux4MCHP/buildroot-external-microchip/board/microchip/sam9x60_hobby/uboot-env.txt
```

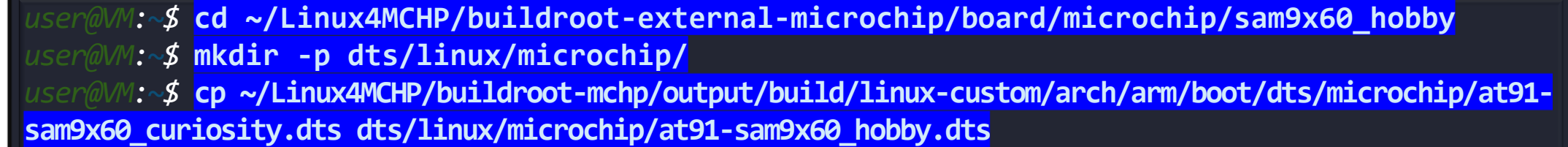
Create a Device-Tree-Source for Hobby Kit in BR2-External Project

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 3.1

- **Create a new Device-Tree-Source** for **Hobby Kit** in Buildroot External.
(based on Curiosity Board's archives)



```
user@LNK1001: ~  
user@LNK1001:~$  
user@VM:~$ cd ~/Linux4MCHP/buildroot-external-microchip/board/microchip/sam9x60_hobby  
user@VM:~$ mkdir -p dts/linux/microchip/  
user@VM:~$ cp ~/Linux4MCHP/buildroot-mchp/output/build/linux-custom/arch/arm/boot/dts/microchip/at91-  
sam9x60_curiosity.dts dts/linux/microchip/at91-sam9x60_hobby.dts
```

VM

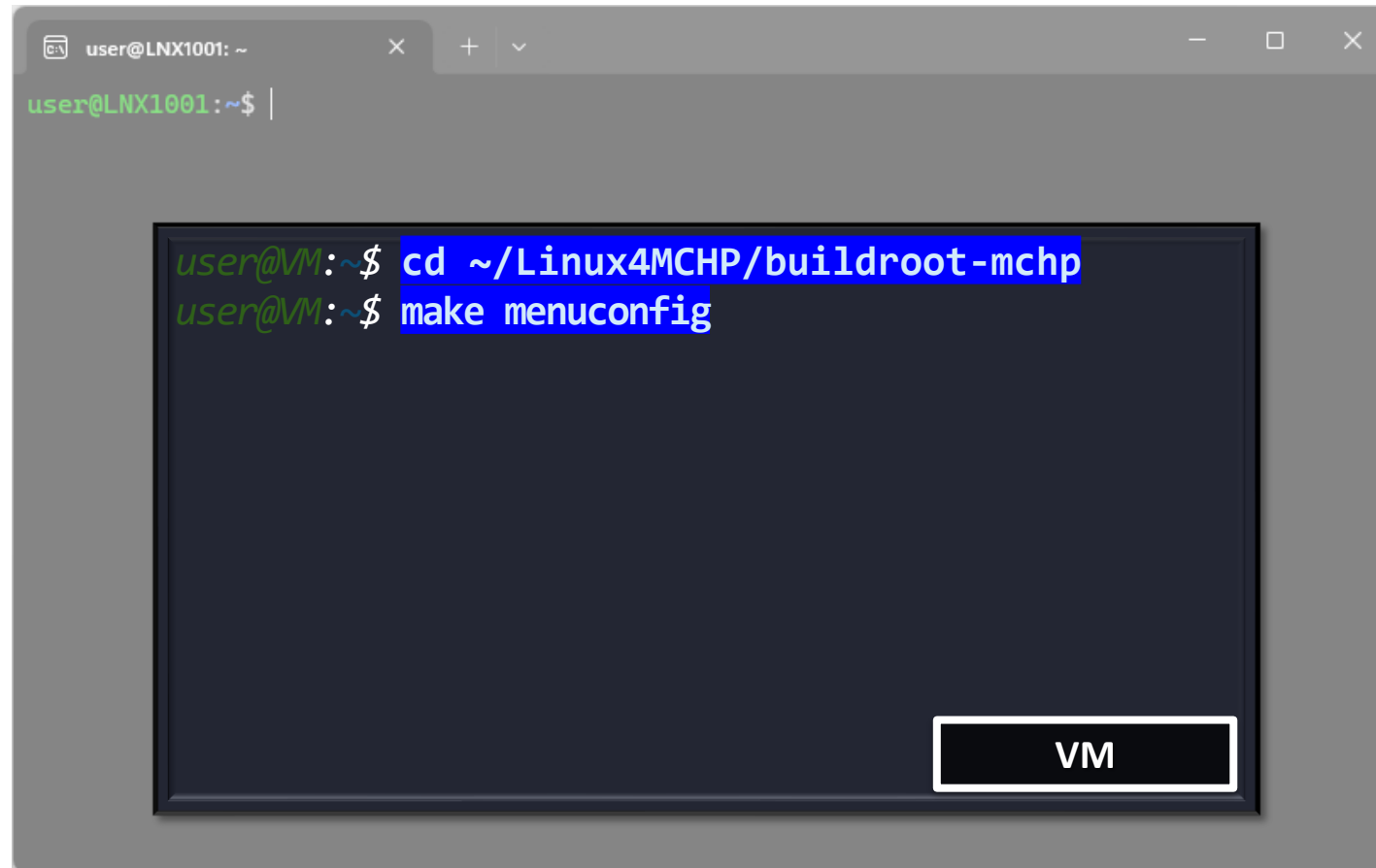
Modify the configuration of Buildroot

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 4.1

- Use the menu to configure Buildroot.

A terminal window with a grey title bar and window controls. The title bar shows 'user@LNX1001: ~'. The terminal content shows a prompt 'user@LNX1001:~\$' followed by a cursor. A dark blue rectangular box is overlaid on the terminal, containing two lines of text: 'user@VM:~\$ cd ~/Linux4MCHP/buildroot-mchp' and 'user@VM:~\$ make menuconfig'. The text in the box is white. In the bottom right corner of the dark blue box, there is a white rectangular button with the text 'VM' in black.

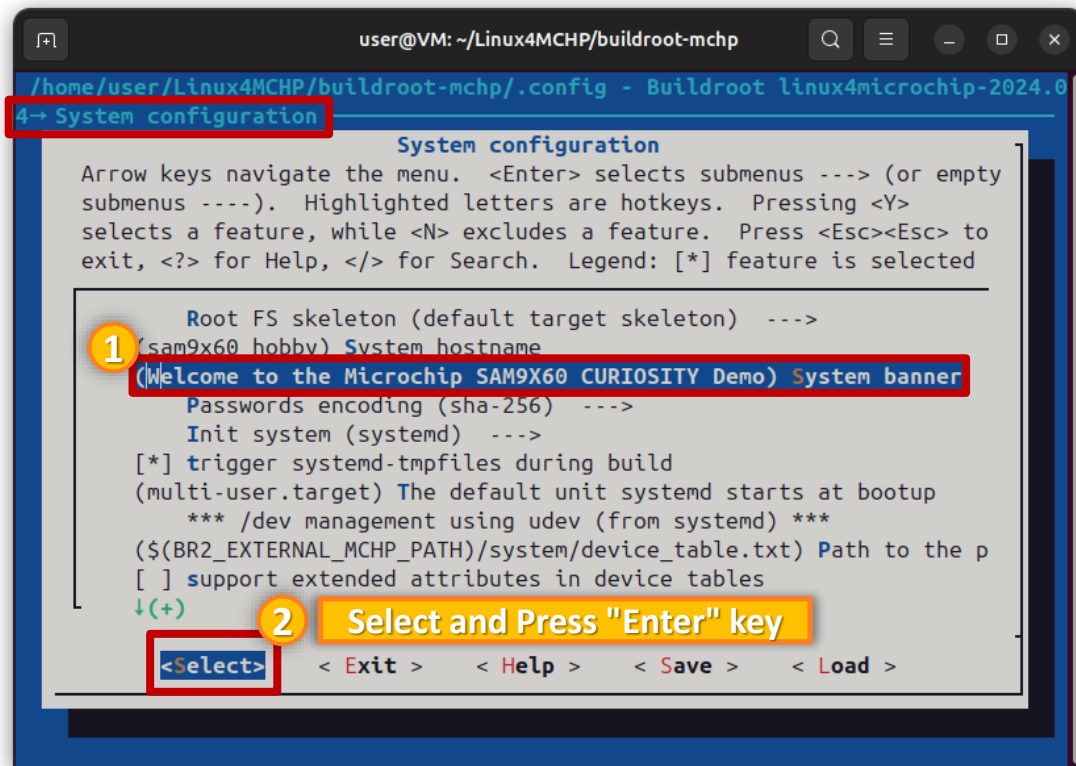
```
user@LNX1001: ~  
user@LNX1001:~$ |  
  
user@VM:~$ cd ~/Linux4MCHP/buildroot-mchp  
user@VM:~$ make menuconfig
```

VM

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 4.2

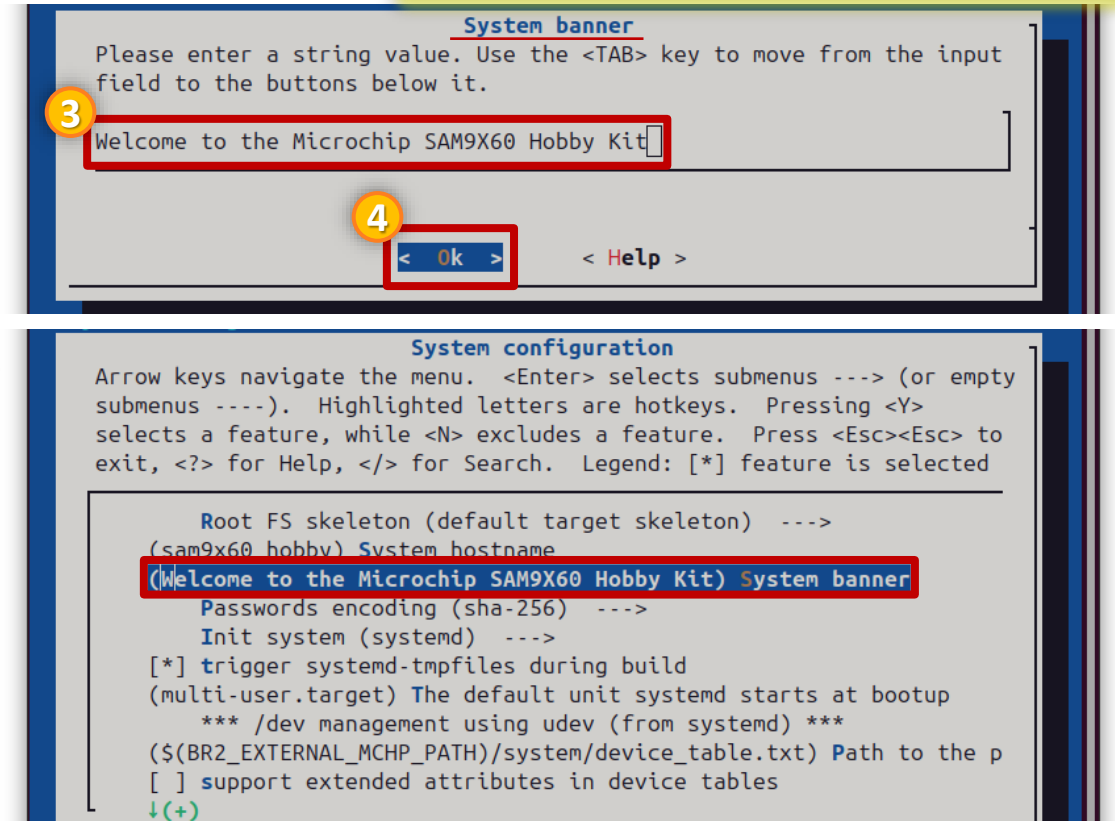
- **Check** the **Login Welcome Message** for SAM9X60 Hobby Kit.
(SAM9X60 CURIOSITY Demo → **SAM9X60 Hobby Kit**)



```
user@VM: ~/Linux4MCHP/buildroot-mchp
/home/user/Linux4MCHP/buildroot-mchp/.config - Buildroot linux4microchip-2024.0
4→ System configuration

System configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected

1 Root FS skeleton (default target skeleton) --->
  (sam9x60 hobby) System hostname
  (Welcome to the Microchip SAM9X60 CURIOSITY Demo) System banner
  Passwords encoding (sha-256) --->
  Init system (systemd) --->
  [*] trigger systemd-tmpfiles during build
  (multi-user.target) The default unit systemd starts at bootup
  *** /dev management using udev (from systemd) ***
  ($(BR2_EXTERNAL_MCHP_PATH)/system/device_table.txt) Path to the p
  [ ] support extended attributes in device tables
  ↓(+)
2 <Select> < Exit > < Help > < Save > < Load >
```



```
System banner
Please enter a string value. Use the <TAB> key to move from the input
field to the buttons below it.

3 Welcome to the Microchip SAM9X60 Hobby Kit

4 < Ok > < Help >

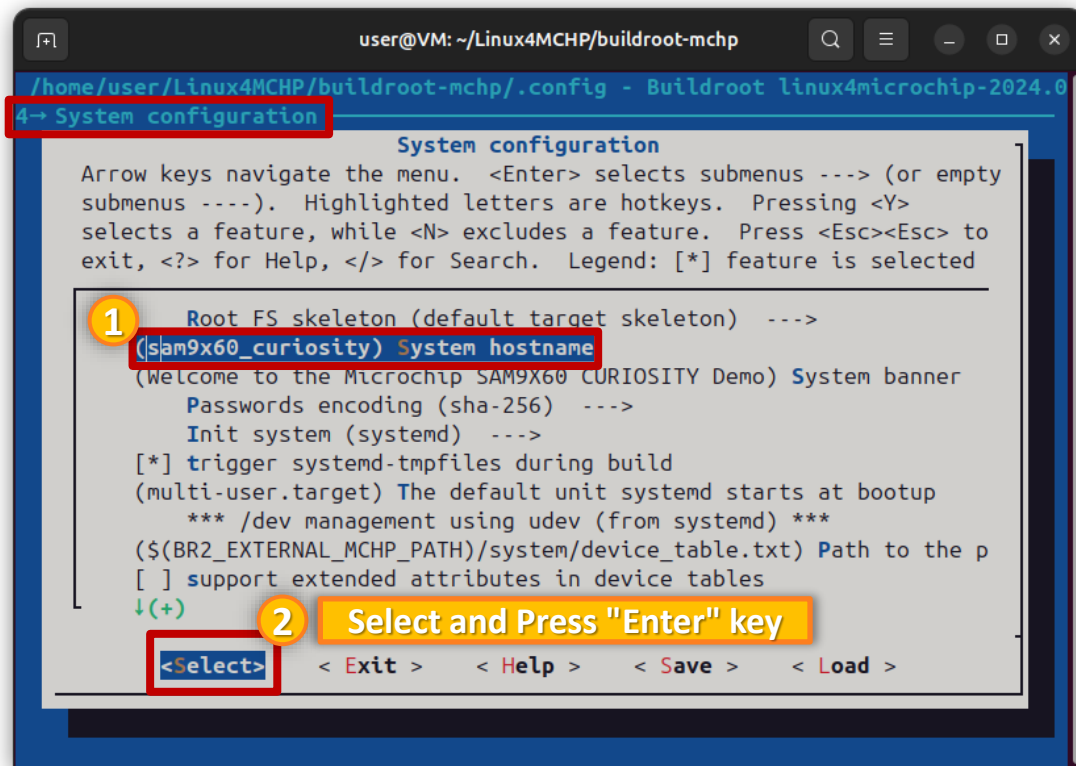
System configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected

Root FS skeleton (default target skeleton) --->
(sam9x60 hobby) System hostname
(Welcome to the Microchip SAM9X60 Hobby Kit) System banner
Passwords encoding (sha-256) --->
Init system (systemd) --->
[*] trigger systemd-tmpfiles during build
(multi-user.target) The default unit systemd starts at bootup
*** /dev management using udev (from systemd) ***
($(BR2_EXTERNAL_MCHP_PATH)/system/device_table.txt) Path to the p
[ ] support extended attributes in device tables
↓(+)
<Select> < Exit > < Help > < Save > < Load >
```

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 4.3

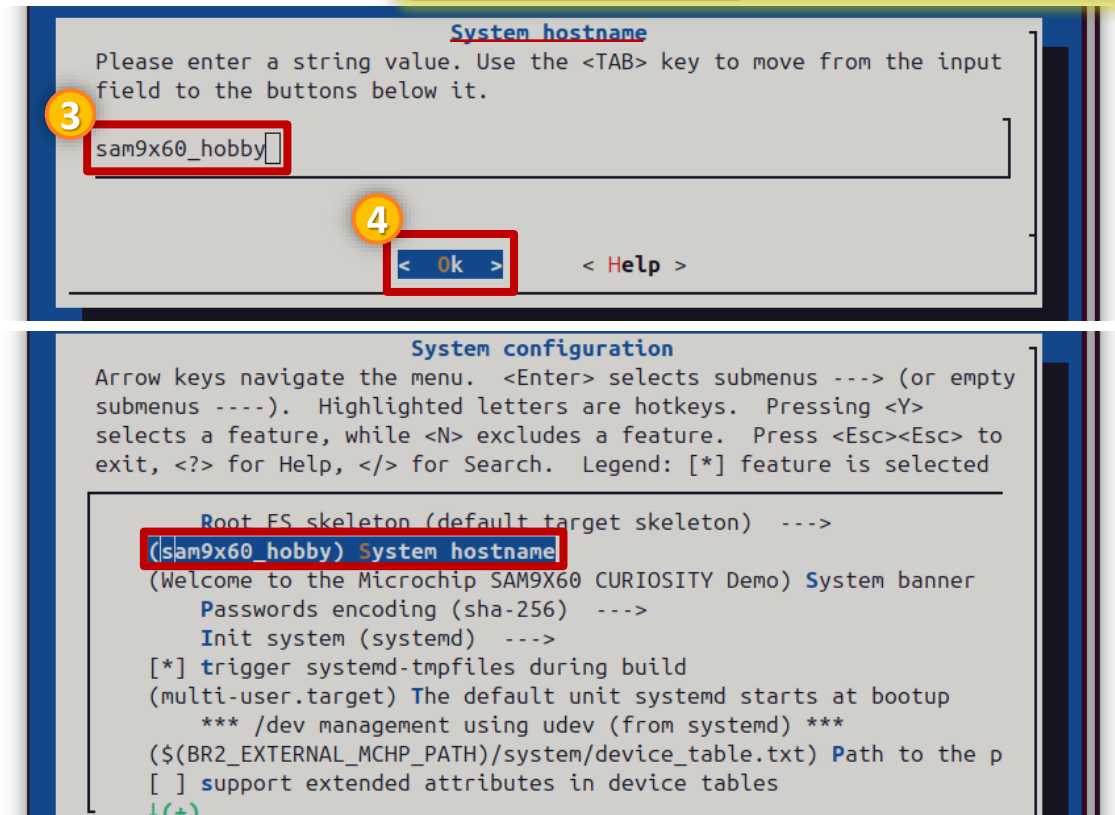
- Check the System Hostname for SAM9X60 Hobby Kit.
(sam9x60_curiosity → sam9x60_hobby)



```
user@VM: ~/Linux4MCHP/buildroot-mchp
/home/user/Linux4MCHP/buildroot-mchp/.config - Buildroot linux4microchip-2024.0
4→ System configuration

System configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected

1 Root FS skeleton (default target skeleton) --->
  (sam9x60_curiosity) System hostname
  (Welcome to the Microchip SAM9X60 CURIOSITY Demo) System banner
  Passwords encoding (sha-256) --->
  Init system (systemd) --->
  [*] trigger systemd-tmpfiles during build
  (multi-user.target) The default unit systemd starts at bootup
  *** /dev management using udev (from systemd) ***
  ($ (BR2_EXTERNAL_MCHP_PATH)/system/device_table.txt) Path to the p
  [ ] support extended attributes in device tables
  ↓(+)
2 Select and Press "Enter" key
  <select> < Exit > < Help > < Save > < Load >
```



```
Welcome to the Microchip SAM9X60 Hobby Kit
sam9x60hobby login:

System hostname
Please enter a string value. Use the <TAB> key to move from the input
field to the buttons below it.

3 sam9x60_hobby
4 < Ok > < Help >

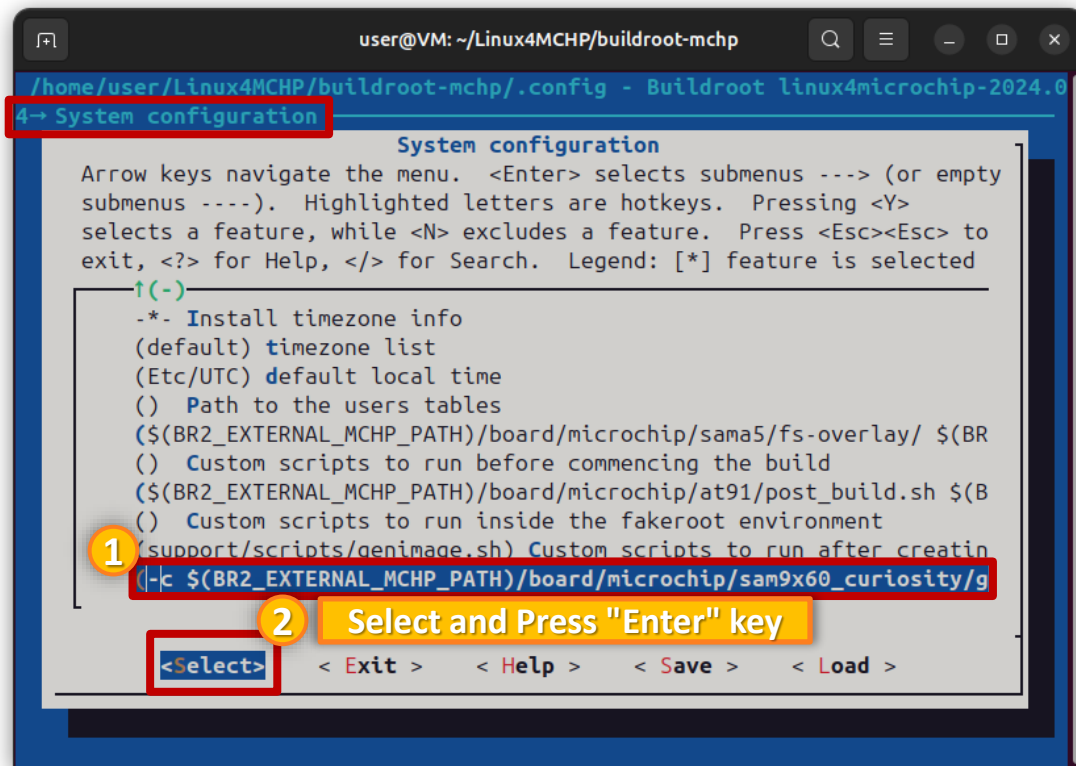
System configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected

Root FS skeleton (default target skeleton) --->
(sam9x60_hobby) System hostname
(Welcome to the Microchip SAM9X60 CURIOSITY Demo) System banner
Passwords encoding (sha-256) --->
Init system (systemd) --->
[*] trigger systemd-tmpfiles during build
(multi-user.target) The default unit systemd starts at bootup
*** /dev management using udev (from systemd) ***
($ (BR2_EXTERNAL_MCHP_PATH)/system/device_table.txt) Path to the p
[ ] support extended attributes in device tables
↓(+)
3
```

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 4.4

- **Modify** the **Generate Image Scripts Location** for SAM9X60 Hobby Kit.
(-c \$(BR2_EXTERNAL_MCHP_PATH)/board/microchip/**sam9x60_hobby/genimage.cfg**)



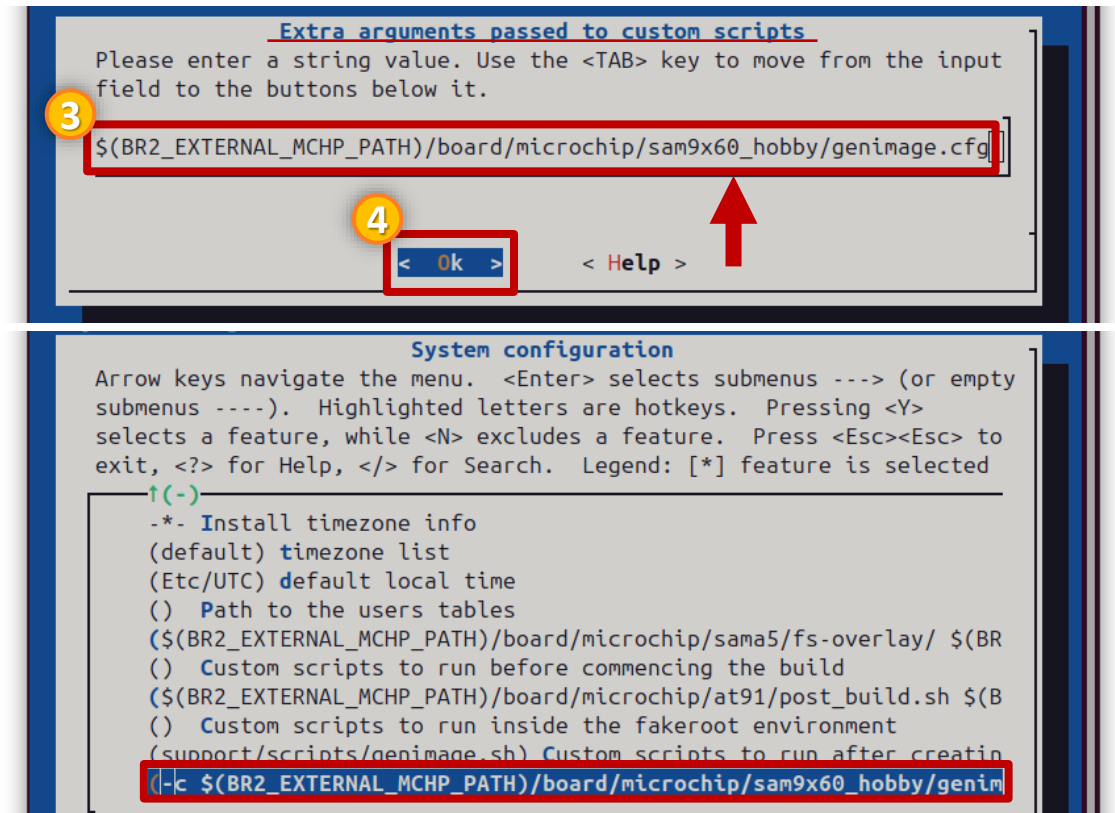
The screenshot shows the Buildroot configuration menu in a terminal window. The title bar indicates the user is at a VM with the path ~/Linux4MCHP/buildroot-mchp. The menu is titled '4 -> System configuration' and 'System configuration'. It provides instructions on how to navigate the menu using arrow keys, Enter, and other keys. A list of features is shown, including 'Install timezone info', 'Path to the users tables', and 'Custom scripts to run after creating the root filesystem'. The last option, 'Custom scripts to run after creating the root filesystem', is highlighted, and its value is set to '(-c \$(BR2_EXTERNAL_MCHP_PATH)/board/microchip/sam9x60_curiosity/g'. A red box highlights this path, and a yellow box with the number '2' points to the '<Select>' button at the bottom of the menu.

```
user@VM: ~/Linux4MCHP/buildroot-mchp
/home/user/Linux4MCHP/buildroot-mchp/.config - Buildroot linux4microchip-2024.0
4 -> System configuration

System configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected

↑(-)
-* Install timezone info
(default) timezone list
(Etc/UTC) default local time
() Path to the users tables
($(BR2_EXTERNAL_MCHP_PATH)/board/microchip/sama5/fs-overlay/ $(BR
() Custom scripts to run before commencing the build
($(BR2_EXTERNAL_MCHP_PATH)/board/microchip/at91/post_build.sh $(B
() Custom scripts to run inside the fakeroot environment
(support/scripts/genimage.sh) Custom scripts to run after creatin
(-c $(BR2_EXTERNAL_MCHP_PATH)/board/microchip/sam9x60_curiosity/g

1
2 Select and Press "Enter" key
<Select> < Exit > < Help > < Save > < Load >
```



The screenshot shows the Buildroot configuration menu in a terminal window. The title bar indicates the user is at a VM with the path ~/Linux4MCHP/buildroot-mchp. The menu is titled 'Extra arguments passed to custom scripts' and 'System configuration'. It provides instructions on how to navigate the menu using arrow keys, Enter, and other keys. A list of features is shown, including 'Install timezone info', 'Path to the users tables', and 'Custom scripts to run after creating the root filesystem'. The last option, 'Custom scripts to run after creating the root filesystem', is highlighted, and its value is set to '(-c \$(BR2_EXTERNAL_MCHP_PATH)/board/microchip/sam9x60_hobby/genimage.cfg'. A red box highlights this path, and a yellow box with the number '4' points to the '<Ok>' button at the bottom of the menu.

```
Extra arguments passed to custom scripts
Please enter a string value. Use the <TAB> key to move from the input
field to the buttons below it.

3
$ (BR2_EXTERNAL_MCHP_PATH)/board/microchip/sam9x60_hobby/genimage.cfg

4
< Ok > < Help >

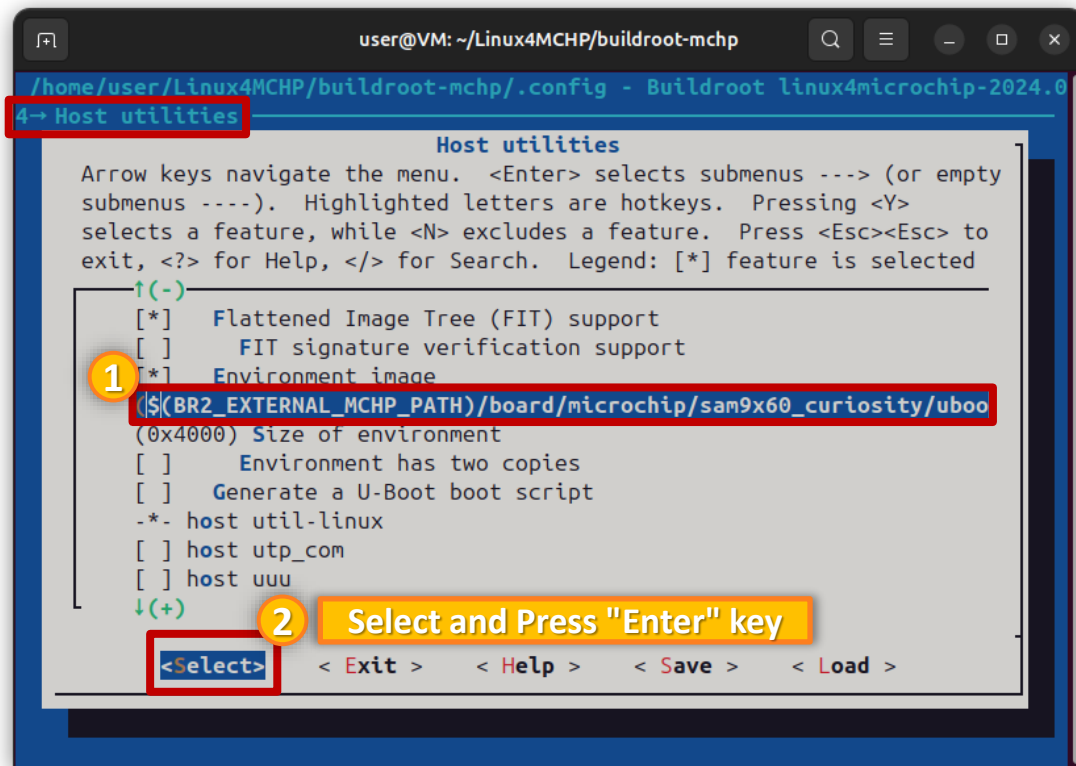
System configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected

↑(-)
-* Install timezone info
(default) timezone list
(Etc/UTC) default local time
() Path to the users tables
($(BR2_EXTERNAL_MCHP_PATH)/board/microchip/sama5/fs-overlay/ $(BR
() Custom scripts to run before commencing the build
($(BR2_EXTERNAL_MCHP_PATH)/board/microchip/at91/post_build.sh $(B
() Custom scripts to run inside the fakeroot environment
(support/scripts/genimage.sh) Custom scripts to run after creatin
(-c $(BR2_EXTERNAL_MCHP_PATH)/board/microchip/sam9x60_hobby/genim
```

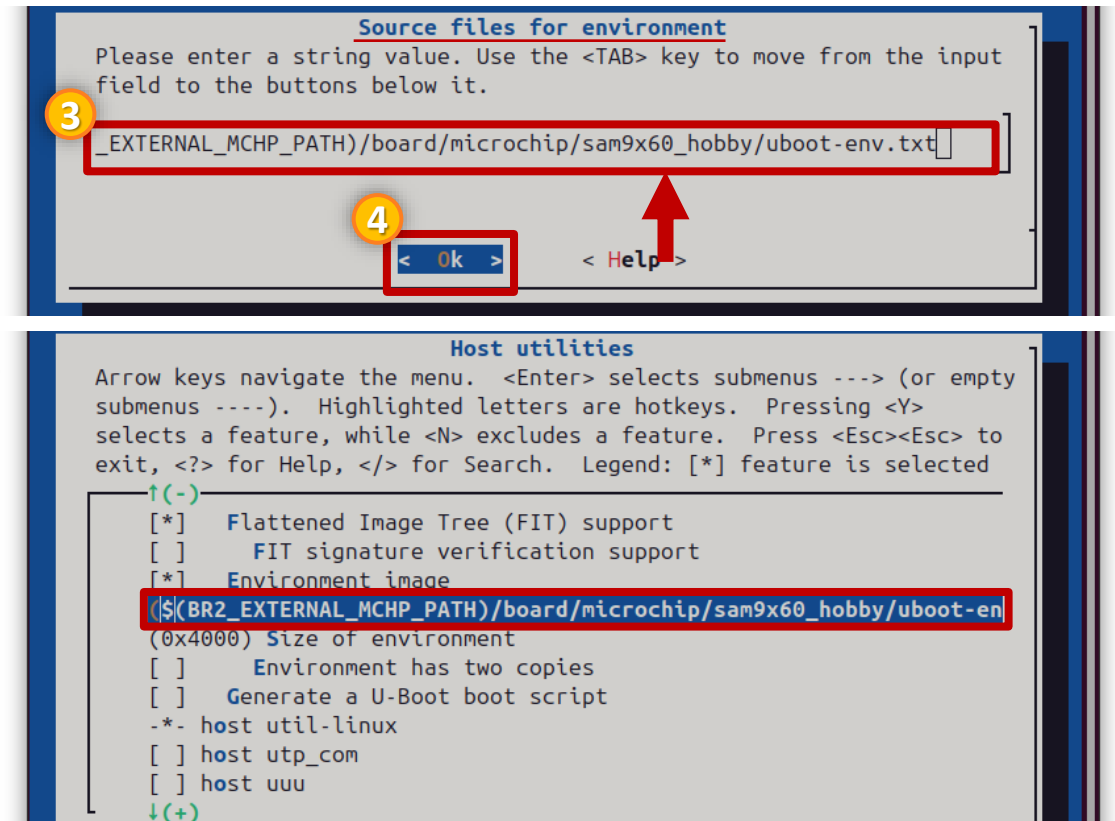
Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 4.5

- **Modify** the **U-boot Environment Variables Source Location** for SAM9X60 Hobby Kit.
(`$(BR2_EXTERNAL_MCHP_PATH)/board/microchip/sam9x60_hobby/uboot-env.txt`)



The screenshot shows the Buildroot configuration menu for Linux4MCHP. The title bar indicates the user is at a VM with the path ~/Linux4MCHP/buildroot-mchp. The main window title is "/home/user/Linux4MCHP/buildroot-mchp/.config - Buildroot linux4microchip-2024.0". The menu is titled "Host utilities" and includes instructions on navigation. A red box highlights the "Environment image" option, which is currently set to "(\$BR2_EXTERNAL_MCHP_PATH)/board/microchip/sam9x60_curiosity/uboot". A yellow callout with the number "1" points to this line. Another yellow callout with the number "2" points to the "<Select>" button at the bottom left of the menu.

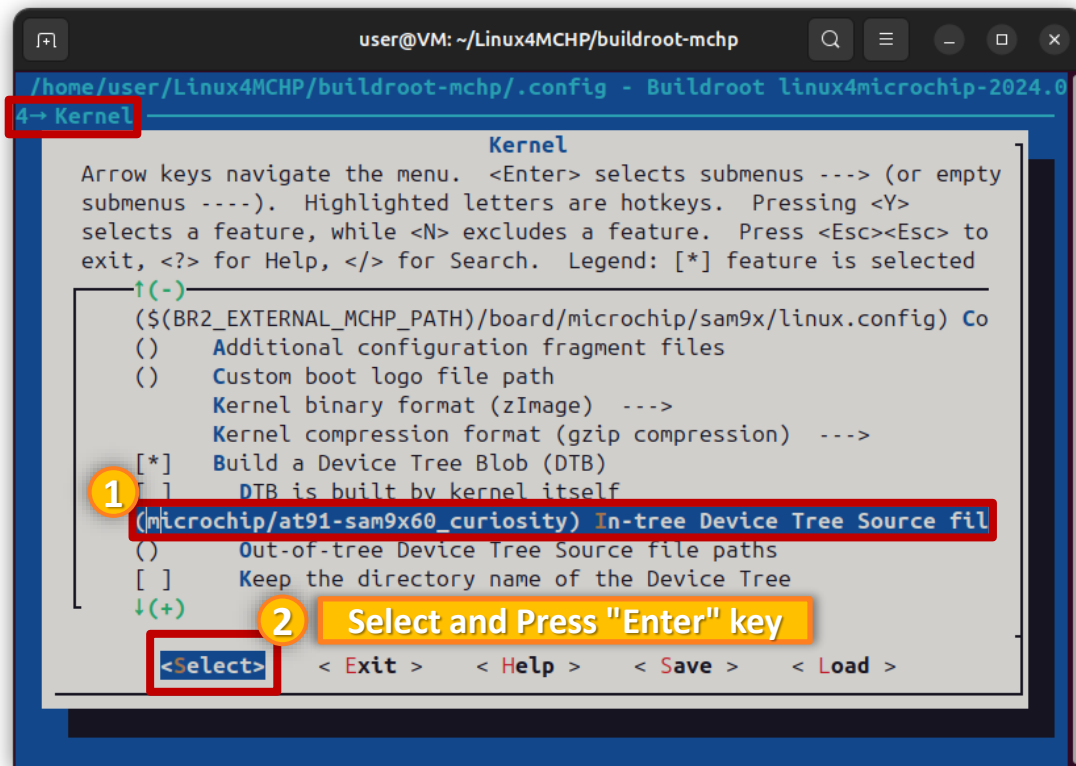


The screenshot shows the "Source files for environment" dialog box. It prompts the user to "Please enter a string value. Use the <TAB> key to move from the input field to the buttons below it." The input field contains the path "(\$BR2_EXTERNAL_MCHP_PATH)/board/microchip/sam9x60_hobby/uboot-env.txt". A red box highlights this input field, with a yellow callout "3" pointing to it. Below the input field, there are two buttons: "< Ok >" and "< Help >". A red box highlights the "< Ok >" button, with a yellow callout "4" pointing to it. A red arrow points from the "< Ok >" button back up to the input field.

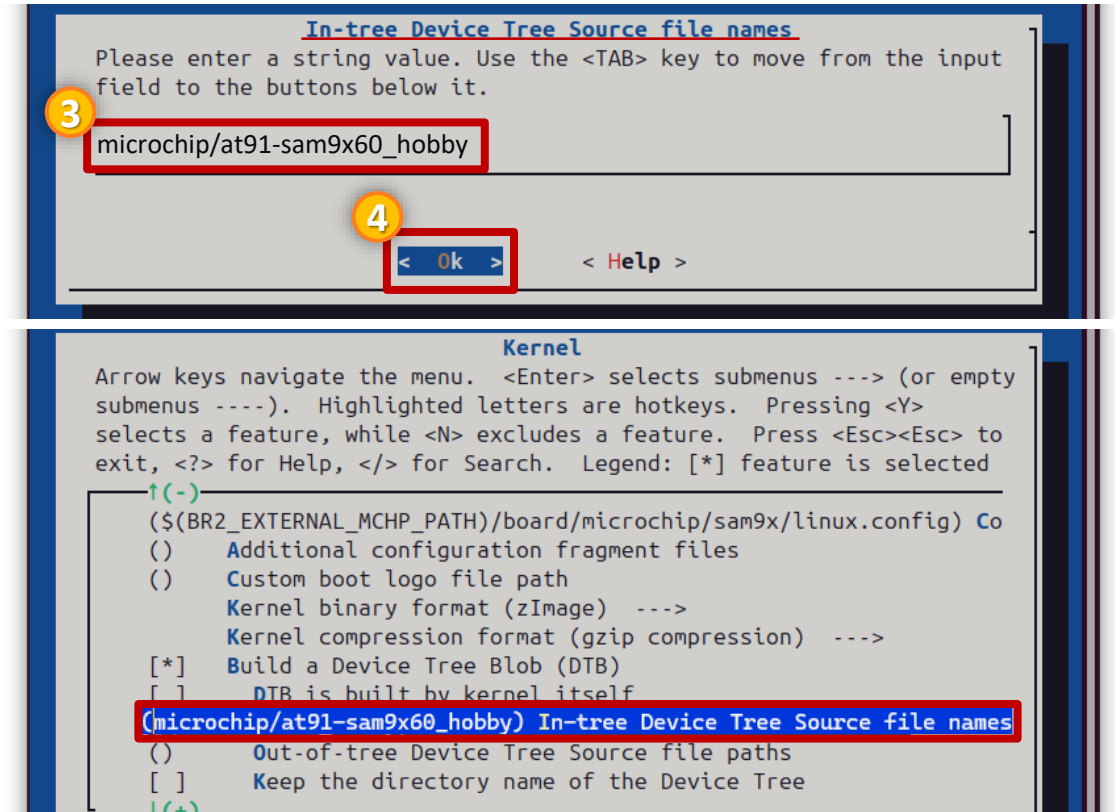
Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 4.6

- **Modify** the **Device-Tree-Source File Name** for SAM9X60 Hobby Kit for Next Step.
(microchip/at91-sam9x60_curiosity → microchip/at91-sam9x60_hobby)



Terminal window showing the Buildroot configuration menu. The title bar indicates the user is at a VM with the path ~/Linux4MCHP/buildroot-mchp. The prompt shows the current configuration file is /home/user/Linux4MCHP/buildroot-mchp/.config for Buildroot linux4microchip-2024.0. The menu is titled 'Kernel' and includes instructions on navigation. A red box highlights the 'Kernel' menu item, and a yellow box highlights the '4→ Kernel' prompt. The menu options include 'Additional configuration fragment files', 'Custom boot logo file path', 'Kernel binary format (zImage)', 'Kernel compression format (gzip compression)', and 'Build a Device Tree Blob (DTB)'. The 'DTB' option is selected, and the 'In-tree Device Tree Source file names' option is highlighted. A red box highlights the current value '(microchip/at91-sam9x60_curiosity)'. A yellow box highlights the instruction 'Select and Press "Enter" key'. The bottom of the screen shows navigation keys: '<Select>', '<Exit>', '<Help>', '<Save>', and '<Load>'.

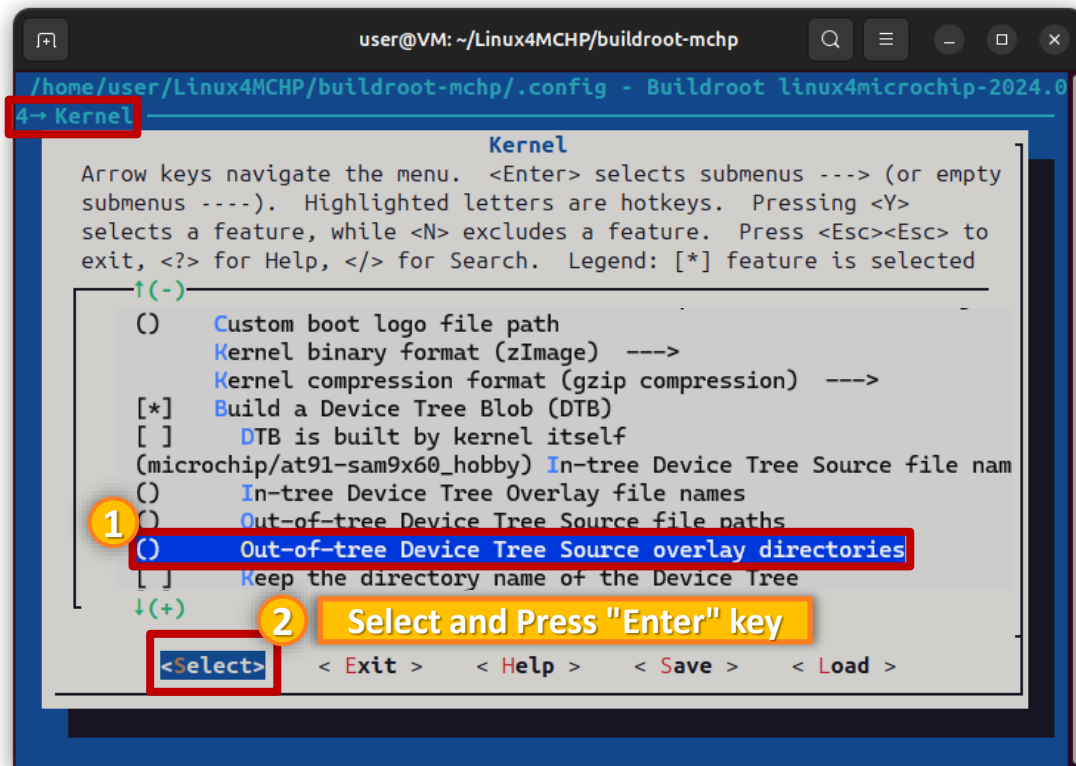


Terminal window showing the 'In-tree Device Tree Source file names' prompt. The prompt asks the user to enter a string value and use the <TAB> key to move from the input field to the buttons below it. A red box highlights the input field containing 'microchip/at91-sam9x60_hobby'. A yellow box highlights the '3' next to the input field. A red box highlights the '< Ok >' button, and a yellow box highlights the '4' next to the button. The bottom of the screen shows the same navigation keys as the previous window: '<Select>', '<Exit>', '<Help>', '<Save>', and '<Load>'.

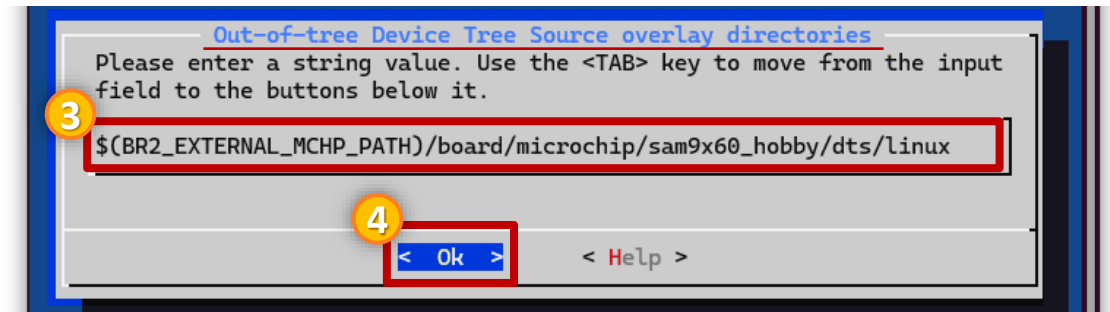
Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 4.7

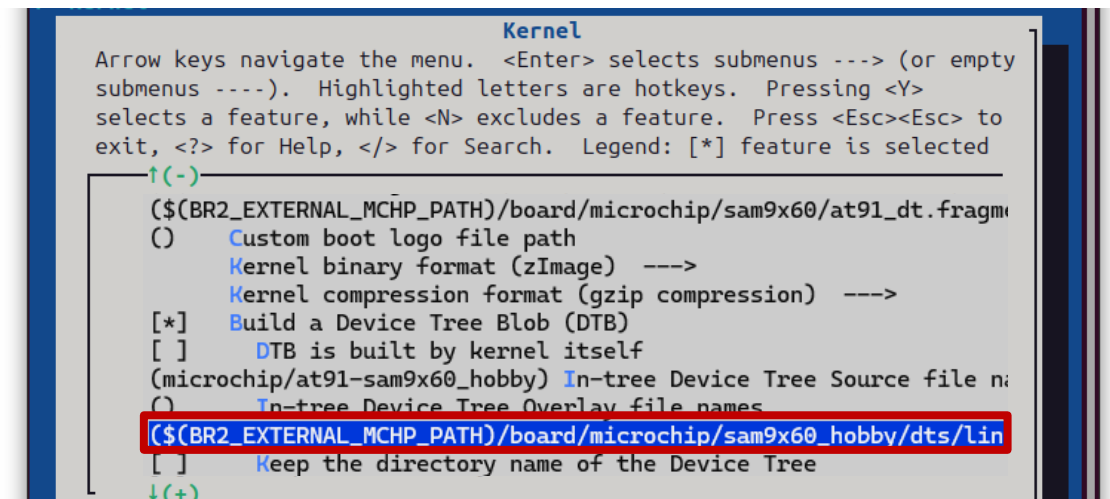
- **Modify** the **Out-of-tree Device Tree Source overlay directories** for SAM9X60 Hobby Kit.
(`$(BR2_EXTERNAL_MCHP_PATH)/board/microchip/sam9x60_hobby/dts/linux`)



```
user@VM: ~/Linux4MCHP/buildroot-mchp
/home/user/Linux4MCHP/buildroot-mchp/.config - Buildroot linux4microchip-2024.0
4→ Kernel
Kernel
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> selects a feature, while <N> excludes a feature. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] feature is selected
↑(-)
() Custom boot logo file path
Kernel binary format (zImage) ---->
Kernel compression format (gzip compression) ---->
[*] Build a Device Tree Blob (DTB)
[] DTB is built by kernel itself
(microchip/at91-sam9x60_hobby) In-tree Device Tree Source file name
() In-tree Device Tree Overlay file names
() Out-of-tree Device Tree Source file paths
1 ( ) Out-of-tree Device Tree Source overlay directories
[] Keep the directory name of the Device Tree
↓(+)
2 Select and Press "Enter" key
<Select> < Exit > < Help > < Save > < Load >
```



```
Out-of-tree Device Tree Source overlay directories
Please enter a string value. Use the <TAB> key to move from the input field to the buttons below it.
3 $(BR2_EXTERNAL_MCHP_PATH)/board/microchip/sam9x60_hobby/dts/linux
4 < Ok > < Help >
```



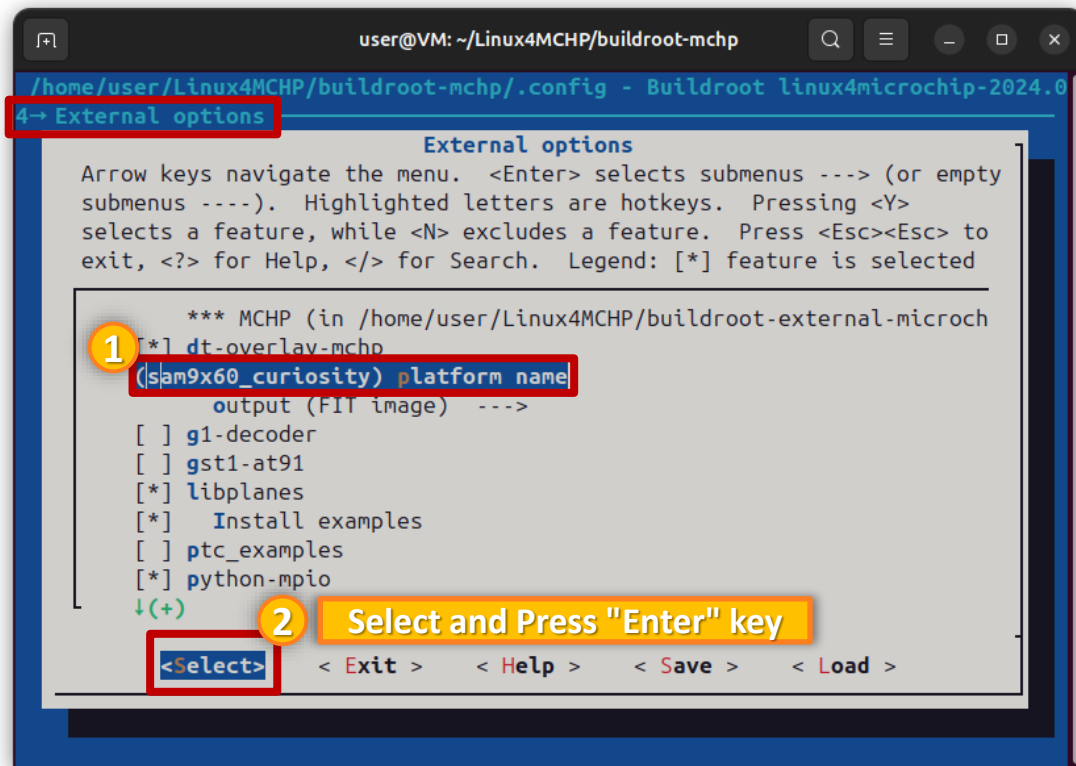
```
Kernel
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> selects a feature, while <N> excludes a feature. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] feature is selected
↑(-)
() Custom boot logo file path
Kernel binary format (zImage) ---->
Kernel compression format (gzip compression) ---->
[*] Build a Device Tree Blob (DTB)
[] DTB is built by kernel itself
(microchip/at91-sam9x60_hobby) In-tree Device Tree Source file name
() In-tree Device Tree Overlay file names
($BR2_EXTERNAL_MCHP_PATH)/board/microchip/sam9x60_hobby/dts/linux
[] Keep the directory name of the Device Tree
↓(+)

```

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 4.8

- **Modify** the **Platform Name** of **dt-overlay-mchp** for **SAM9X60 Hobby Kit**.
(**sam9x60_curiosity** → **sam9x60_hobby**)



```
user@VM: ~/Linux4MCHP/buildroot-mchp
/home/user/Linux4MCHP/buildroot-mchp/.config - Buildroot linux4microchip-2024.0
4→ External options

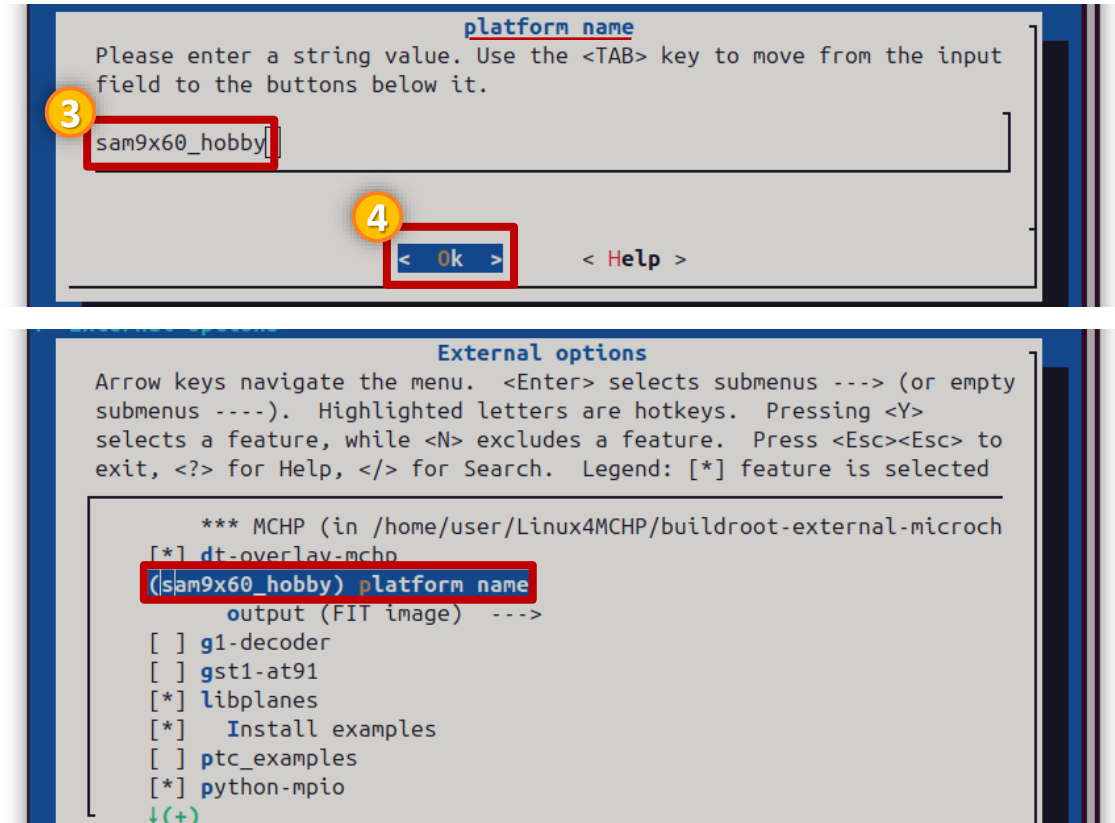
External options
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected

*** MCHP (in /home/user/Linux4MCHP/buildroot-external-microch
[*] dt-overlay-mchp
  (sam9x60_curiosity) platform name
    output (FIT image) --->
  [ ] g1-decoder
  [ ] gst1-at91
  [*] libplanes
  [*] Install examples
  [ ] ptc_examples
  [*] python-mpio
  ↓(+)
```

1

2 Select and Press "Enter" key

<Select> <Exit> <Help> <Save> <Load>



```
platform name
Please enter a string value. Use the <TAB> key to move from the input
field to the buttons below it.

3 sam9x60_hobby

4 <Ok> <Help>
```

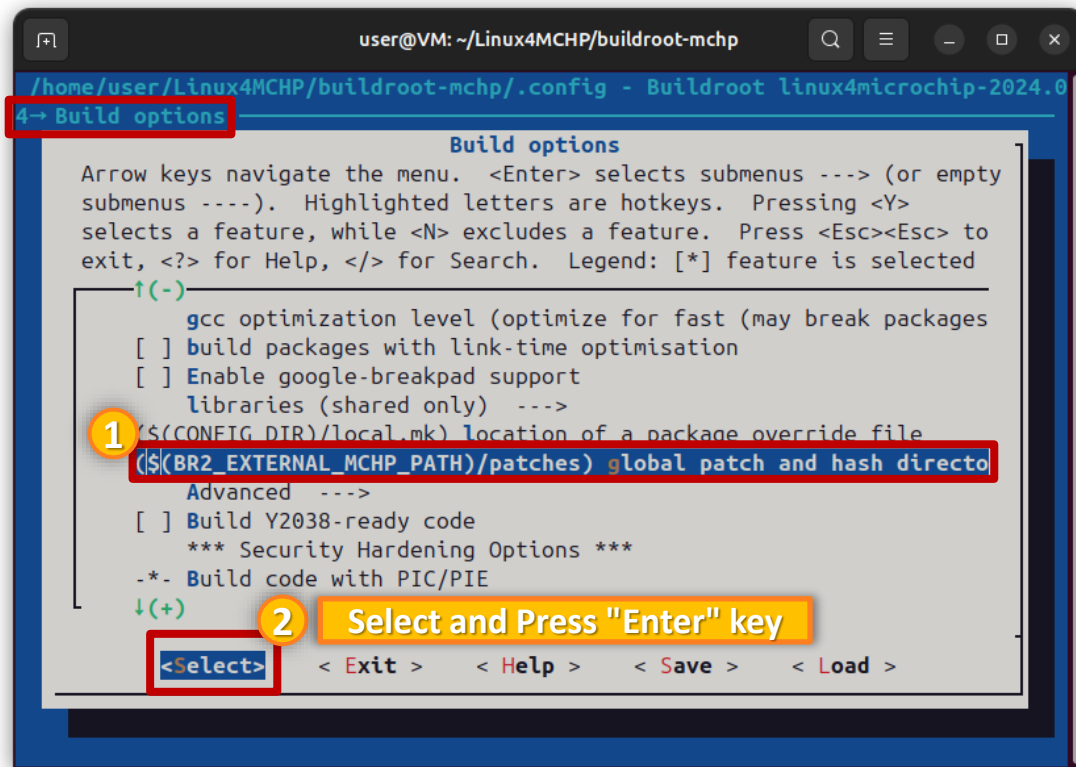
```
External options
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected

*** MCHP (in /home/user/Linux4MCHP/buildroot-external-microch
[*] dt-overlay-mchp
  (sam9x60_hobby) platform name
    output (FIT image) --->
  [ ] g1-decoder
  [ ] gst1-at91
  [*] libplanes
  [*] Install examples
  [ ] ptc_examples
  [*] python-mpio
  ↓(+)
```

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 4.9

- **Modify** the **Patches Location** in Buildroot External.
(`$(BR2_EXTERNAL_MCHP_PATH)/patches` `$(BR2_EXTERNAL_MCHP_PATH)/board/microchip/sam9x60_hobby/patches`)

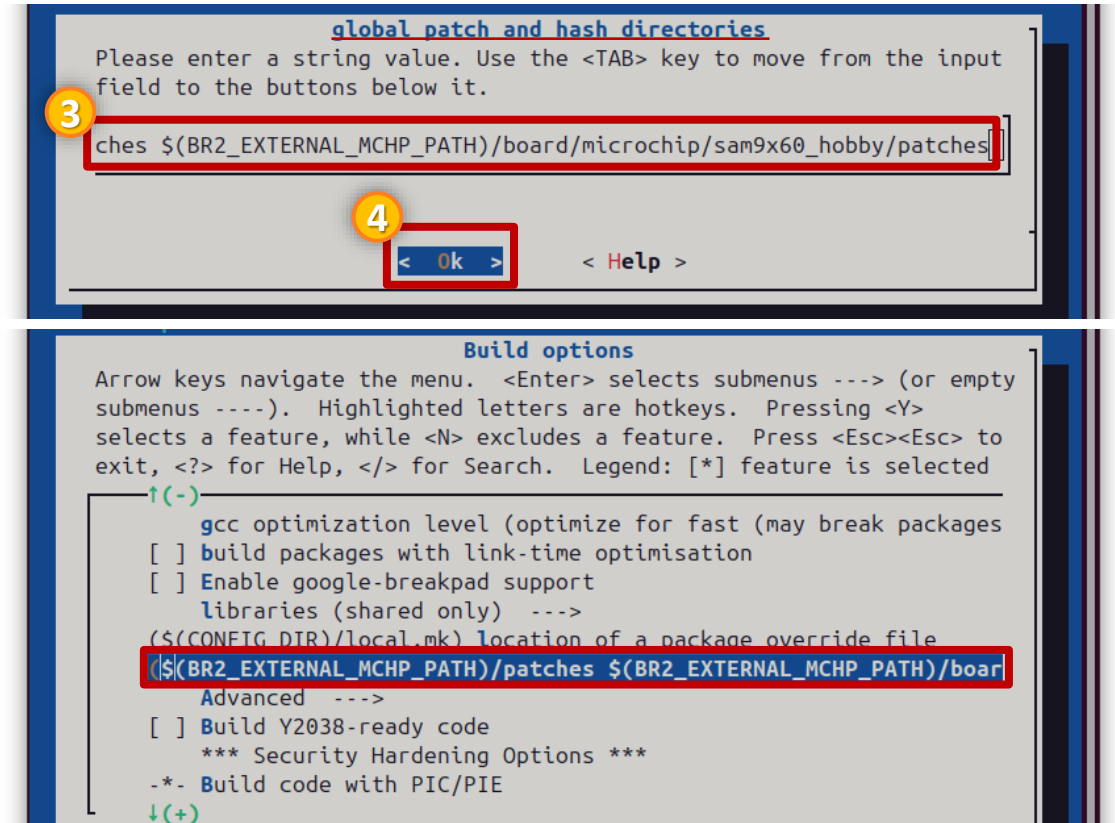


A terminal window showing the Buildroot configuration menu. The title bar reads 'user@VM: ~/Linux4MCHP/buildroot-mchp'. The menu is titled 'Build options' and contains instructions on how to navigate. A red box highlights the path '(\$ (BR2_EXTERNAL_MCHP_PATH)/patches) global patch and hash directories' with a yellow circle '1' next to it. Another red box highlights the '<select>' key with a yellow circle '2' and the text 'Select and Press "Enter" key'.

```
user@VM: ~/Linux4MCHP/buildroot-mchp
/home/user/Linux4MCHP/buildroot-mchp/.config - Buildroot linux4microchip-2024.0
4→ Build options

Build options
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
selects a feature, while <N> excludes a feature. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] feature is selected

↑(-)
gcc optimization level (optimize for fast (may break packages
[ ] build packages with link-time optimisation
[ ] Enable google-breakpad support
libraries (shared only) --->
1 ($ (CONFIG DIR)/local.mk) location of a package override file
($ (BR2_EXTERNAL_MCHP_PATH)/patches) global patch and hash directo
Advanced --->
[ ] Build Y2038-ready code
*** Security Hardening Options ***
-*- Build code with PIC/PIE
↓(+)
2 <select> < Exit > < Help > < Save > < Load >
```



A terminal window showing the Buildroot configuration menu. The title bar reads 'global patch and hash directories'. The menu is titled 'global patch and hash directories' and contains instructions on how to navigate. A red box highlights the path 'ches \$(BR2_EXTERNAL_MCHP_PATH)/board/microchip/sam9x60_hobby/patches' with a yellow circle '3' next to it. Another red box highlights the '< Ok >' key with a yellow circle '4'.

```
global patch and hash directories
Please enter a string value. Use the <TAB> key to move from the input
field to the buttons below it.
3 ches $(BR2_EXTERNAL_MCHP_PATH)/board/microchip/sam9x60_hobby/patches
4 < Ok > < Help >
```

Build options

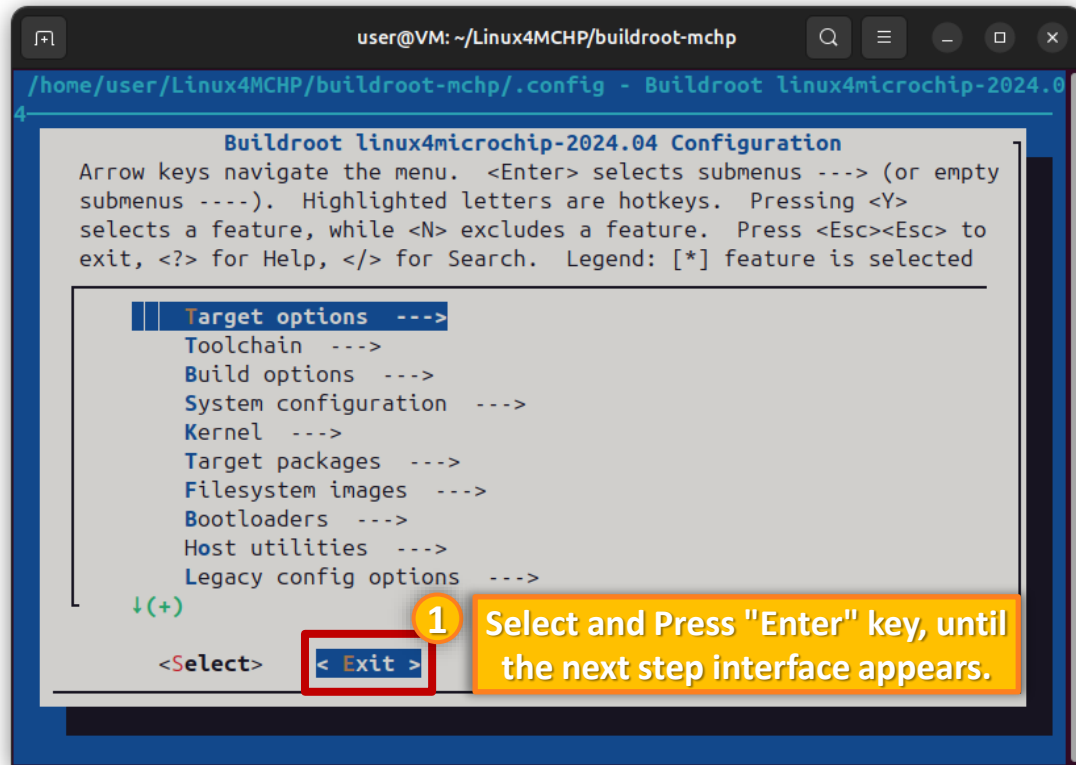
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> selects a feature, while <N> excludes a feature. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] feature is selected

```
↑(-)
gcc optimization level (optimize for fast (may break packages
[ ] build packages with link-time optimisation
[ ] Enable google-breakpad support
libraries (shared only) --->
($ (CONFIG DIR)/local.mk) location of a package override file
($ (BR2_EXTERNAL_MCHP_PATH)/patches $(BR2_EXTERNAL_MCHP_PATH)/boar
Advanced --->
[ ] Build Y2038-ready code
*** Security Hardening Options ***
-*- Build code with PIC/PIE
↓(+)
<select> < Exit > < Help > < Save > < Load >
```

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 4.10

- **Exit** the menu of configure and **Save** new configuration to Buildroot.



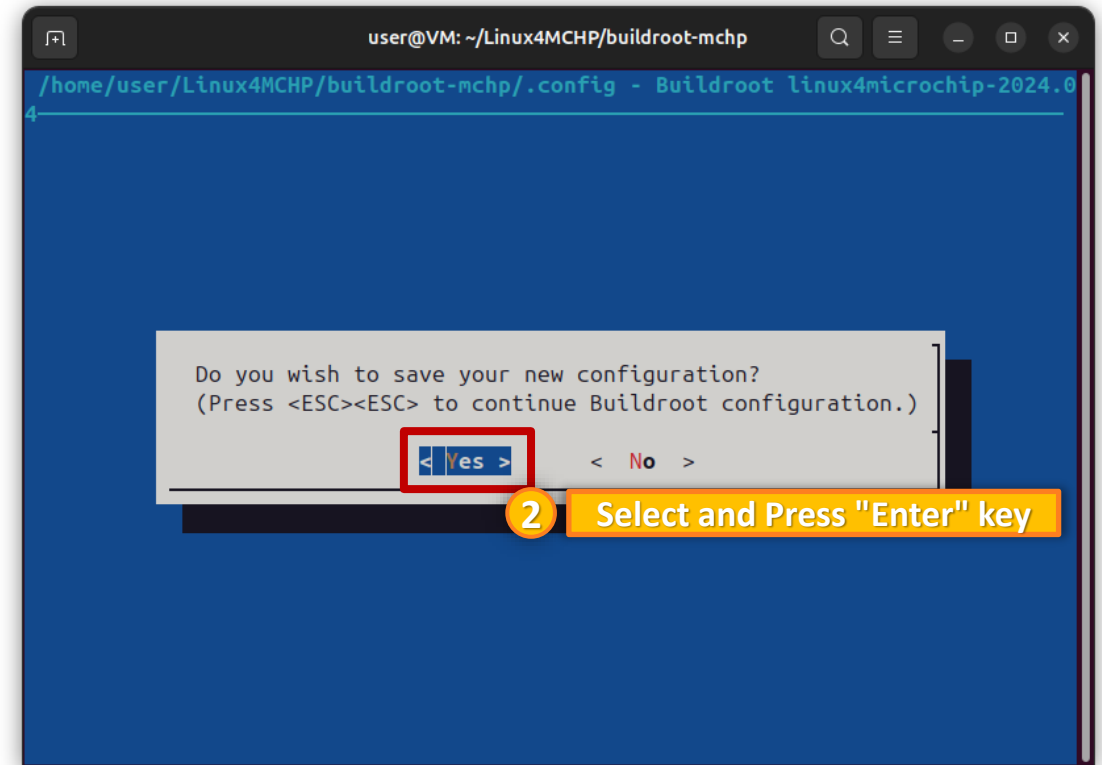
```
user@VM: ~/Linux4MCHP/buildroot-mchp
/home/user/Linux4MCHP/buildroot-mchp/.config - Buildroot linux4microchip-2024.04

Buildroot linux4microchip-2024.04 Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> selects a feature, while <N> excludes a feature. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] feature is selected

Target options --->
Toolchain --->
Build options --->
System configuration --->
Kernel --->
Target packages --->
Filesystem images --->
Bootloaders --->
Host utilities --->
Legacy config options --->

↓(+)
```

1 Select and Press "Enter" key, until the next step interface appears.



```
user@VM: ~/Linux4MCHP/buildroot-mchp
/home/user/Linux4MCHP/buildroot-mchp/.config - Buildroot linux4microchip-2024.04

Do you wish to save your new configuration?
(Press <ESC><ESC> to continue Buildroot configuration.)

< Yes > < No >
```

2 Select and Press "Enter" key

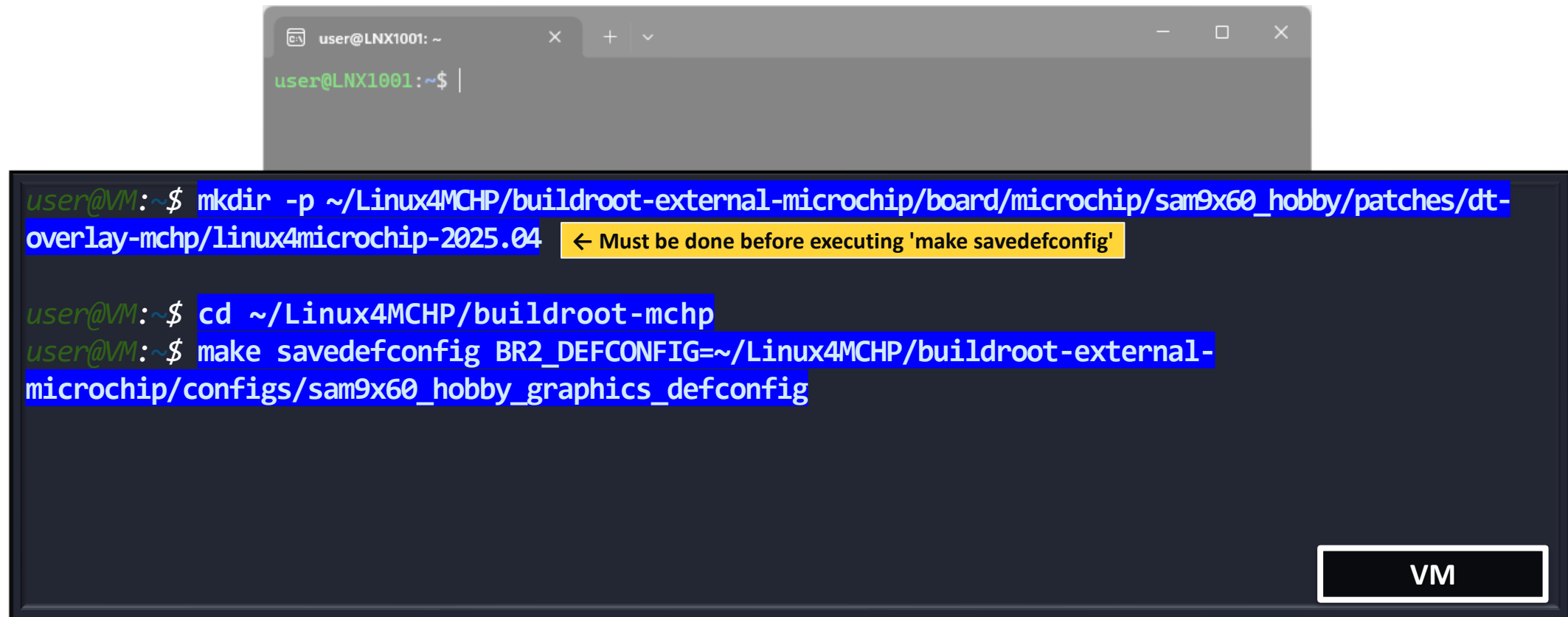
Save the Buildroot configuration to the BR2-External Project

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 5

- **Save the configuration** of SAM9X60 Hobby Kit to Buildroot External.



The image shows a terminal window with a dark background. The title bar of the window reads 'user@LNX1001: ~'. The terminal content shows a sequence of commands and their outputs. The first command is 'mkdir -p ~/Linux4MCHP/buildroot-external-microchip/board/microchip/sam9x60_hobby/patches/dt-overlay-mchp/linux4microchip-2025.04', which is highlighted in blue. A yellow tooltip with a left-pointing arrow and the text 'Must be done before executing 'make savedefconfig'' is positioned to the right of this command. The second command is 'cd ~/Linux4MCHP/buildroot-mchp', also highlighted in blue. The third command is 'make savedefconfig BR2_DEFCONFIG=~/Linux4MCHP/buildroot-external-microchip/configs/sam9x60_hobby_graphics_defconfig', highlighted in blue. In the bottom right corner of the terminal window, there is a white rectangular button with the text 'VM' in black.

```
user@LNX1001: ~  
user@LNX1001:~$  
  
user@VM:~$ mkdir -p ~/Linux4MCHP/buildroot-external-microchip/board/microchip/sam9x60_hobby/patches/dt-  
overlay-mchp/linux4microchip-2025.04 ← Must be done before executing 'make savedefconfig'  
  
user@VM:~$ cd ~/Linux4MCHP/buildroot-mchp  
user@VM:~$ make savedefconfig BR2_DEFCONFIG=~/Linux4MCHP/buildroot-external-  
microchip/configs/sam9x60_hobby_graphics_defconfig
```

VM

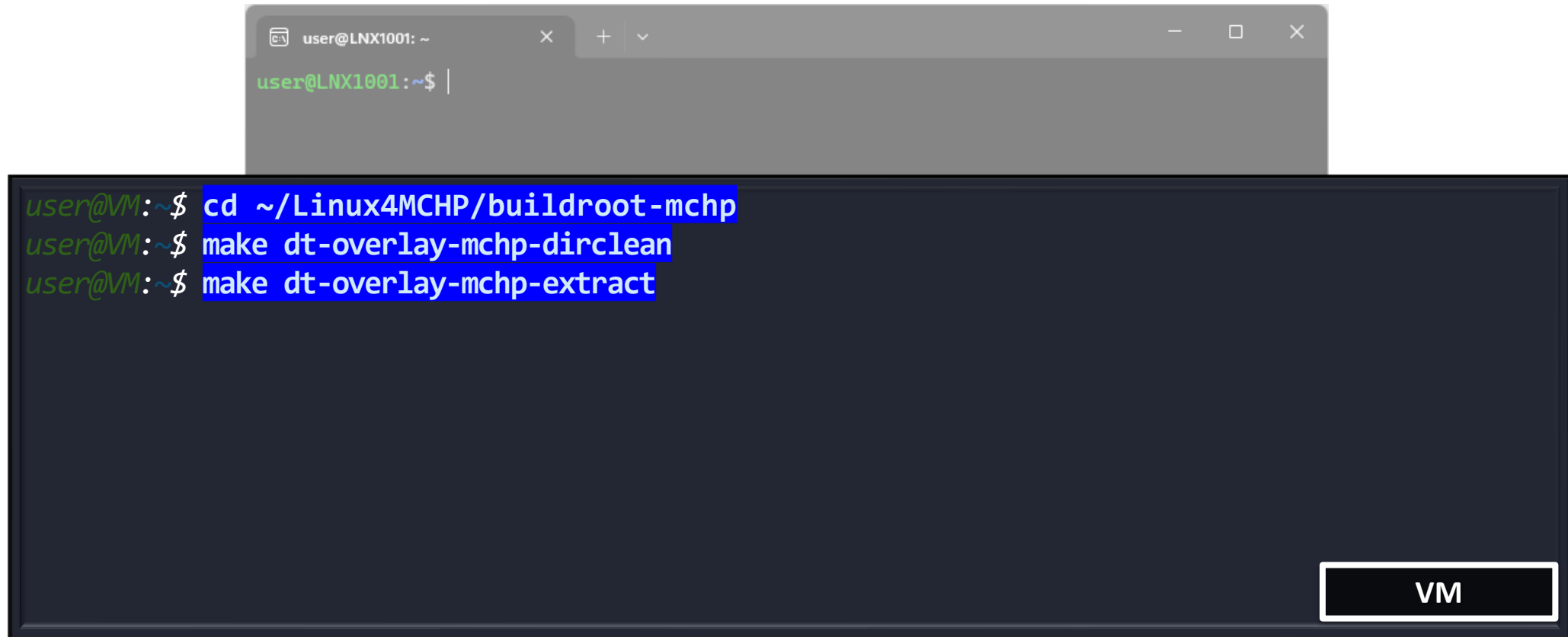
Make a difference patch for BR2-External Project

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 6.1

- **Reset** the **dt-overlay-mchp** in Buildroot.



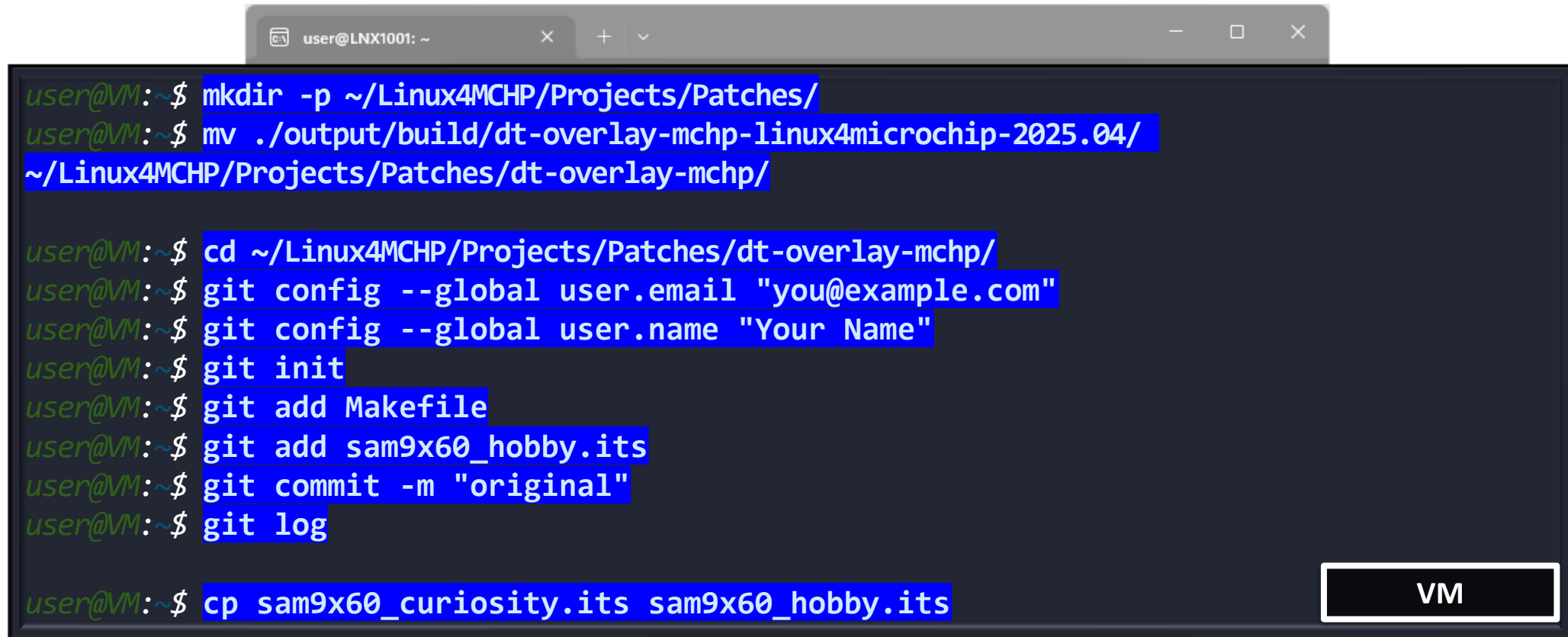
The image shows a terminal window with a dark background. The window title bar indicates the user is at 'user@LNX1001: ~'. The terminal content shows three commands being entered at the 'user@VM:~\$' prompt. The first command is 'cd ~/Linux4MCHP/buildroot-mchp', the second is 'make dt-overlay-mchp-dirclean', and the third is 'make dt-overlay-mchp-extract'. The command paths are highlighted in blue. A small 'VM' label is visible in the bottom right corner of the terminal window.

```
user@LNX1001: ~  
user@LNX1001:~$  
  
user@VM:~$ cd ~/Linux4MCHP/buildroot-mchp  
user@VM:~$ make dt-overlay-mchp-dirclean  
user@VM:~$ make dt-overlay-mchp-extract
```

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 6.2

- Prepared the materials in a Patches directory, we will need to modify the "Makefile" and "sam9x60_hobby.its" of dt-overlay-mchp.



```
user@VM:~$ mkdir -p ~/Linux4MCHP/Projects/Patches/  
user@VM:~$ mv ./output/build/dt-overlay-mchp-linux4microchip-2025.04/  
~/Linux4MCHP/Projects/Patches/dt-overlay-mchp/  
  
user@VM:~$ cd ~/Linux4MCHP/Projects/Patches/dt-overlay-mchp/  
user@VM:~$ git config --global user.email "you@example.com"  
user@VM:~$ git config --global user.name "Your Name"  
user@VM:~$ git init  
user@VM:~$ git add Makefile  
user@VM:~$ git add sam9x60_hobby.its  
user@VM:~$ git commit -m "original"  
user@VM:~$ git log  
  
user@VM:~$ cp sam9x60_curiosity.its sam9x60_hobby.its
```

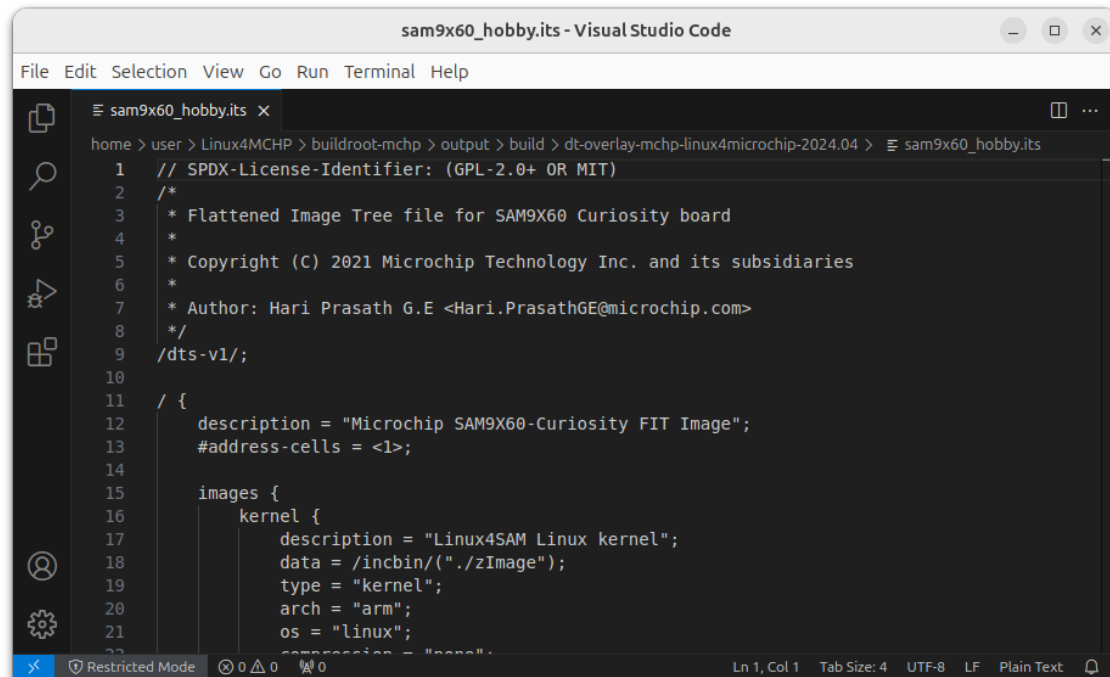
VM

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 6.3

- **Modify** "sam9x60_hobby.its" to create an image for Hobby Kit.

(Use GUI or Command to open the file with Visual Studio Code)



```
sam9x60_hobby.its - Visual Studio Code
File Edit Selection View Go Run Terminal Help
home > user > Linux4MCHP > buildroot-mchp > output > build > dt-overlay-mchp-linux4microchip-2024.04 > sam9x60_hobby.its
1 // SPDX-License-Identifier: (GPL-2.0+ OR MIT)
2 /*
3  * Flattened Image Tree file for SAM9X60 Curiosity board
4  * Copyright (C) 2021 Microchip Technology Inc. and its subsidiaries
5  *
6  * Author: Hari Prasath G.E <Hari.PrasathGE@microchip.com>
7  */
8 /dts-v1/;
9
10 / {
11     description = "Microchip SAM9X60-Curiosity FIT Image";
12     #address-cells = <1>;
13
14     images {
15         kernel {
16             description = "Linux4SAM Linux kernel";
17             data = /incbin/("./zImage");
18             type = "kernel";
19             arch = "arm";
20             os = "linux";
21             compression = "none";
22         };
23     };
24 }
```

```
/ {
    description = "Microchip SAM9X60-Hobby FIT Image";
    #address-cells = <1>;

    images {
        base_fdt {
            description = "SAM9X60-Hobby Flattened Device Tree blob";
            data = /incbin/("./at91-sam9x60_hobby.dtb");
            ...
        };
    };

    configurations {
        default = "kernel_dtb";

        kernel_dtb {
            description = "Linux kernel and base FDT blob for SAM9X60-Hobby board";
            kernel = "kernel";
            fdt = "base_fdt";
        };

        base_dtb {
            description = "Base FDT blob for SAM9X60-Hobby board ";
            fdt = "base_fdt";
        };
    };
};
```

←Remove fdt_pda5 and fdt_wilc3000 nodes

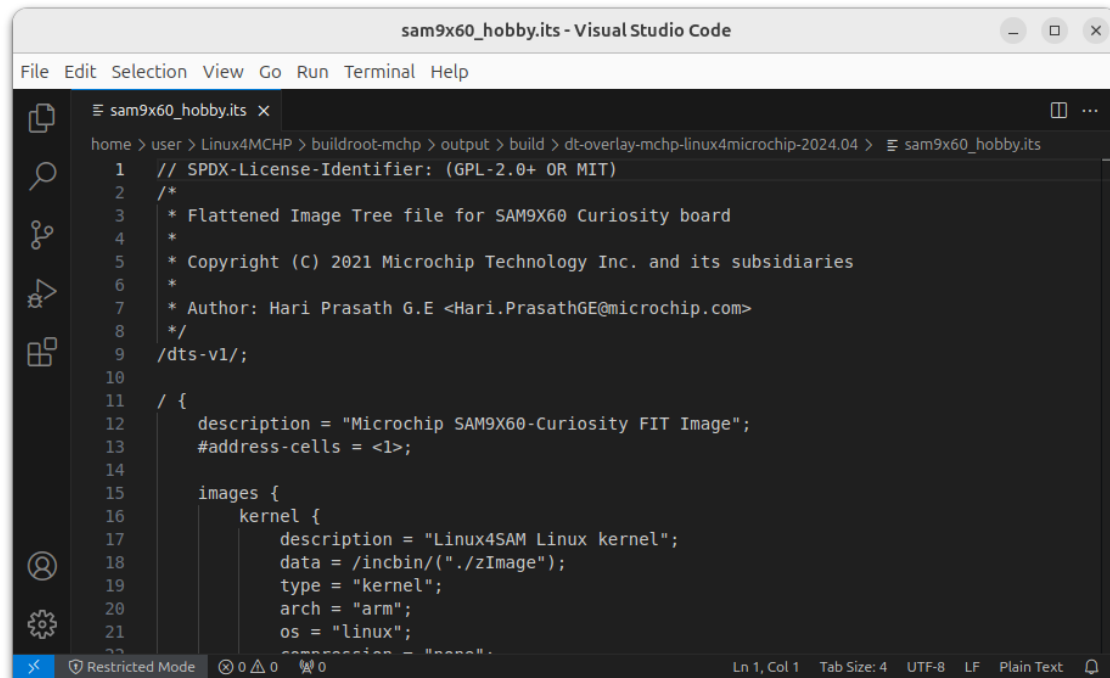
←Remove pda5 and wilc3000 nodes

```
user@VM:~$ code ~/Linux4MCHP/Projects/Patches/dt-overlay-mchp/sam9x60_hobby.its
```

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

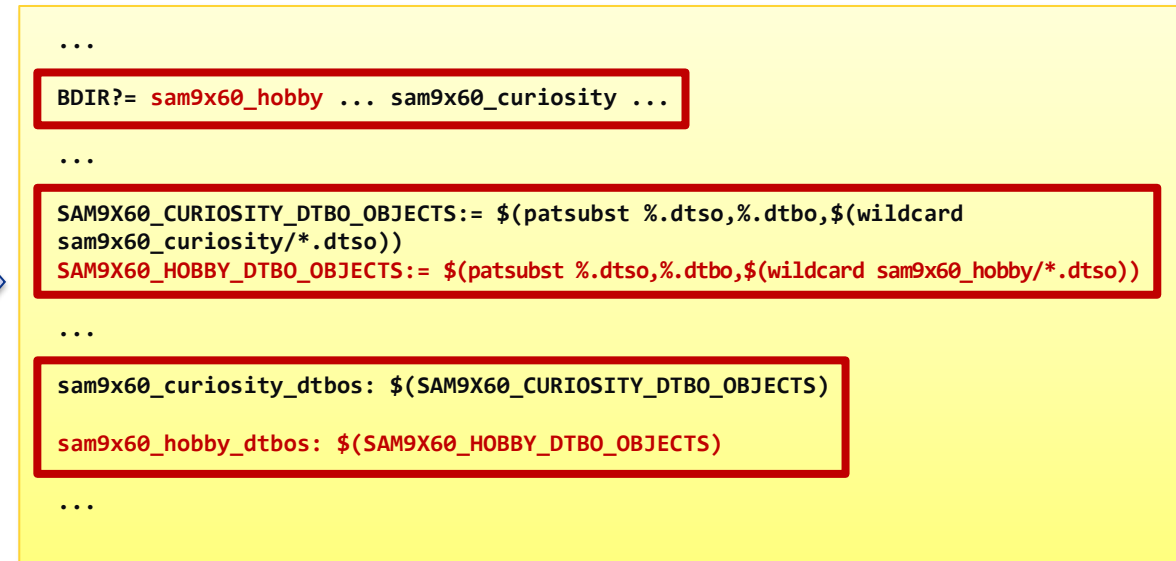
Step 6.4

- **Modify "Makefile"** to create an image for Hobby Kit.
(Use GUI or Command to open the file with Visual Studio Code)



The screenshot shows the Visual Studio Code editor with the file `sam9x60_hobby.its` open. The file content is as follows:

```
1 // SPDX-License-Identifier: (GPL-2.0+ OR MIT)
2 /*
3  * Flattened Image Tree file for SAM9X60 Curiosity board
4  * Copyright (C) 2021 Microchip Technology Inc. and its subsidiaries
5  *
6  * Author: Hari Prasath G.E <Hari.PrasathGE@microchip.com>
7  */
8 /dts-v1/;
9
10 / {
11     description = "Microchip SAM9X60-Curiosity FIT Image";
12     #address-cells = <1>;
13
14     images {
15         kernel {
16             description = "Linux4SAM Linux kernel";
17             data = /incbin("../zImage");
18             type = "kernel";
19             arch = "arm";
20             os = "linux";
21             compression = "none";
22         }
23     }
24 }
```



The excerpt shows the following modifications in the Makefile, highlighted with red boxes:

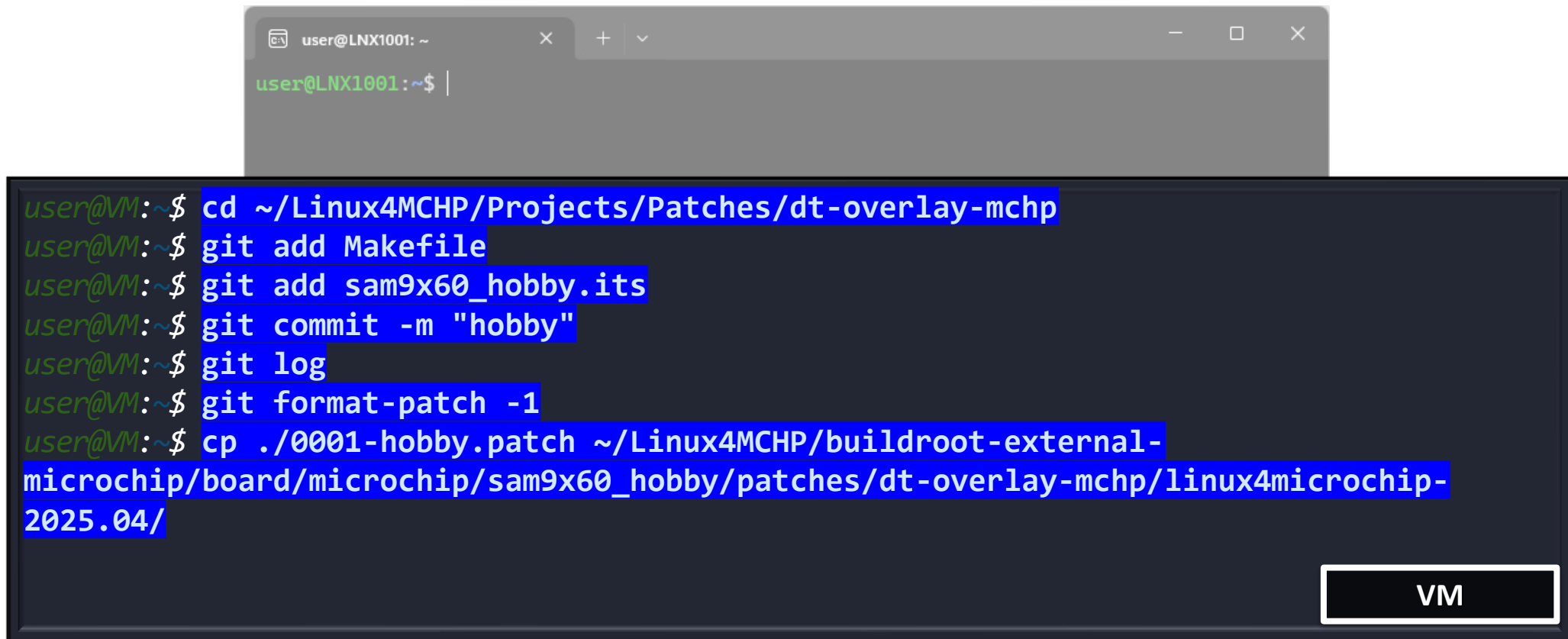
```
...
BDIR?= sam9x60_hobby ... sam9x60_curiosity ...
...
SAM9X60_CURIOSITY_DTBO_OBJECTS:= $(patsubst %.dtso,%.dtbo,$(wildcard
sam9x60_curiosity/*.dtso))
SAM9X60_HOBBY_DTBO_OBJECTS:= $(patsubst %.dtso,%.dtbo,$(wildcard sam9x60_hobby/*.dtso))
...
sam9x60_curiosity_dtbos: $(SAM9X60_CURIOSITY_DTBO_OBJECTS)
sam9x60_hobby_dtbos: $(SAM9X60_HOBBY_DTBO_OBJECTS)
...
```

```
user@VM:~$ code ~/Linux4MCHP/Projects/Patches/dt-overlay-mchp/Makefile
```

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 6.5

- **Make a difference patch** after modifying dt-overlay-mchp and place it in the Buildroot External directory of the SAM9X60 Hobby Kit.



```
user@LNX1001: ~  
user@LNX1001:~$  
  
user@VM:~$ cd ~/Linux4MCHP/Projects/Patches/dt-overlay-mchp  
user@VM:~$ git add Makefile  
user@VM:~$ git add sam9x60_hobby.its  
user@VM:~$ git commit -m "hobby"  
user@VM:~$ git log  
user@VM:~$ git format-patch -1  
user@VM:~$ cp ./0001-hobby.patch ~/Linux4MCHP/buildroot-external-  
microchip/board/microchip/sam9x60_hobby/patches/dt-overlay-mchp/linux4microchip-  
2025.04/
```

VM

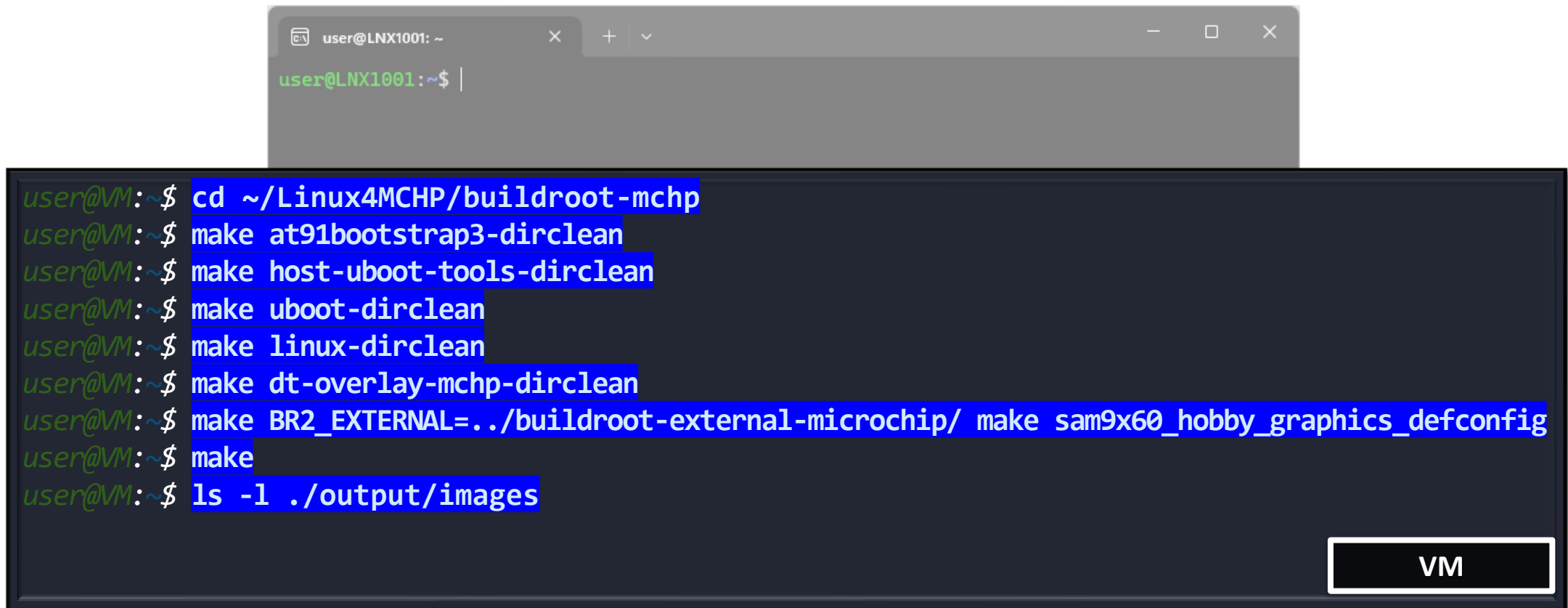
Rebuild the Buildroot and Prepare a SD Card with Embedded Linux OS

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 7

- **Rebuild** the Buildroot of the configuration and Buildroot External for SAM9X60 Hobby Kit.



The image shows a terminal window with a dark background. The window title bar indicates the user is 'user@LNX1001: ~'. The terminal content shows a series of commands being executed in a shell, with the prompt 'user@VM:~\$' at the start of each line. The commands are: 'cd ~/Linux4MCHP/buildroot-mchp', 'make at91bootstrap3-dirclean', 'make host-uboot-tools-dirclean', 'make uboot-dirclean', 'make linux-dirclean', 'make dt-overlay-mchp-dirclean', 'make BR2_EXTERNAL=../buildroot-external-microchip/ make sam9x60_hobby_graphics_defconfig', 'make', and 'ls -l ./output/images'. The output of the last command is not visible. A small 'VM' button is located in the bottom right corner of the terminal window.

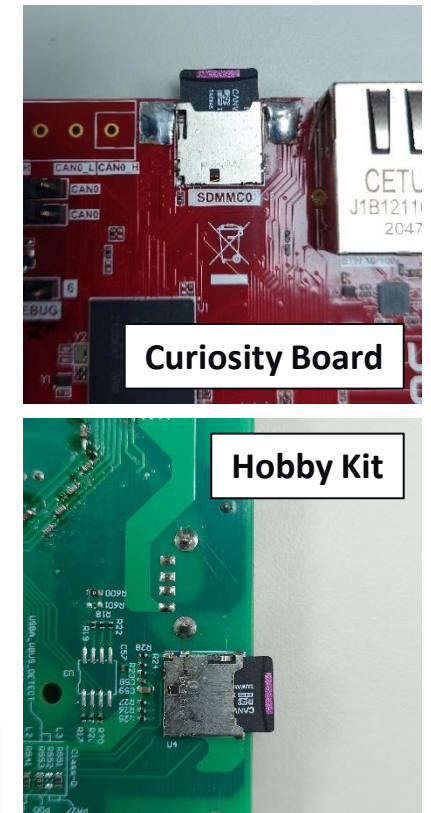
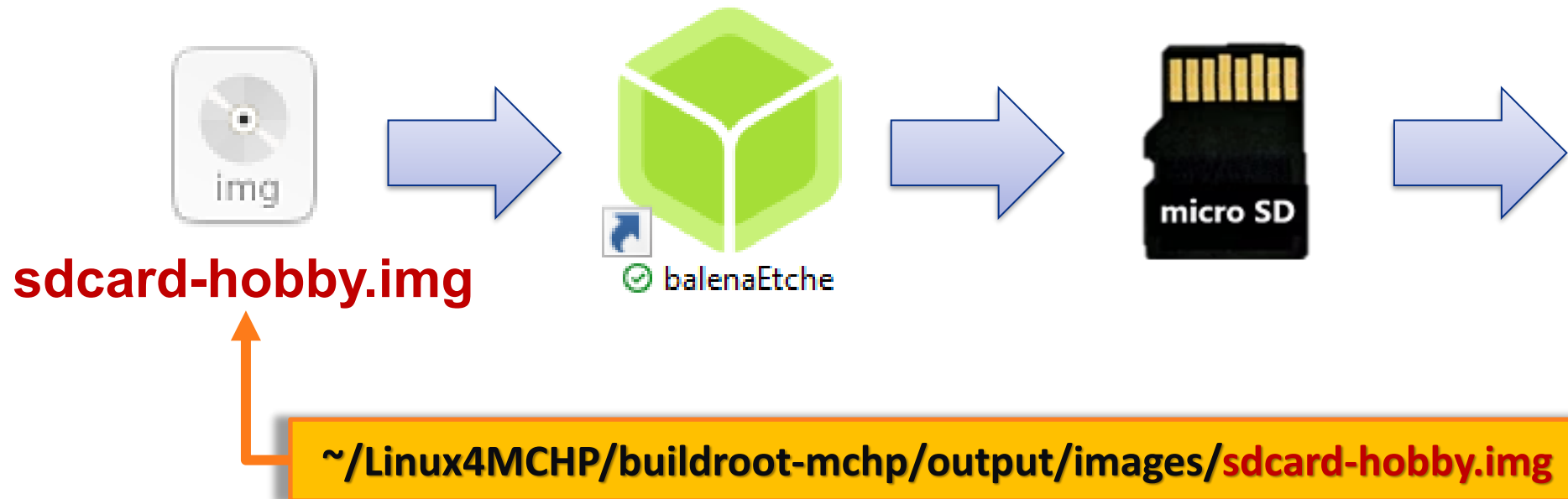
```
user@LNX1001: ~  
user@LNX1001:~$  
  
user@VM:~$ cd ~/Linux4MCHP/buildroot-mchp  
user@VM:~$ make at91bootstrap3-dirclean  
user@VM:~$ make host-uboot-tools-dirclean  
user@VM:~$ make uboot-dirclean  
user@VM:~$ make linux-dirclean  
user@VM:~$ make dt-overlay-mchp-dirclean  
user@VM:~$ make BR2_EXTERNAL=../buildroot-external-microchip/ make sam9x60_hobby_graphics_defconfig  
user@VM:~$ make  
user@VM:~$ ls -l ./output/images
```

VM

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 8

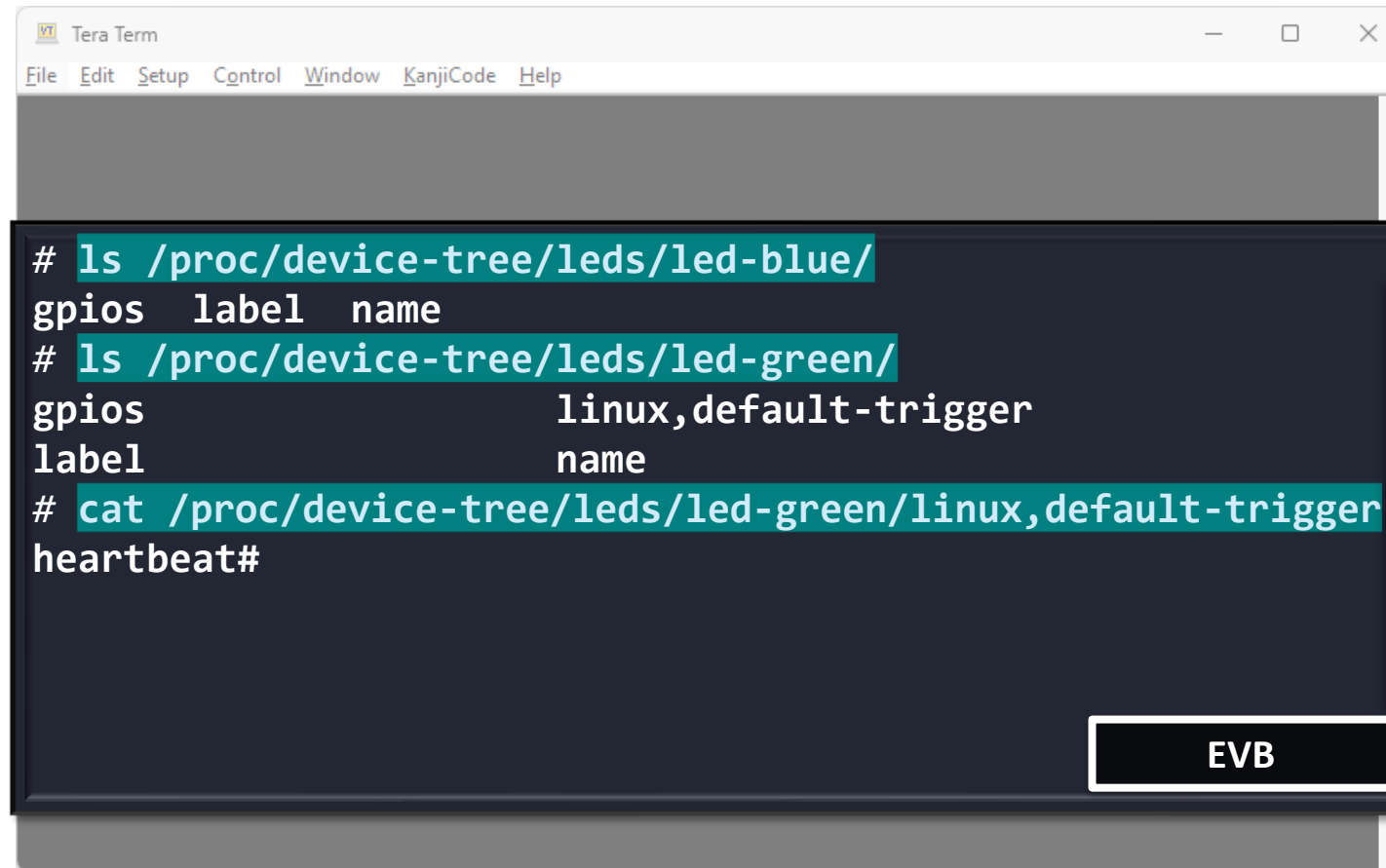
- Prepare a SD Card with Embedded Linux OS and mount it to EVB.
(Refer to the appendix to complete the Flash SD Card)



Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 9.1

- **Check** the modified part in the Embedded Linux System.



The screenshot shows a Tera Term terminal window with a menu bar (File, Edit, Setup, Control, Window, KanjiCode, Help). The terminal displays the following commands and output:

```
# ls /proc/device-tree/leds/led-blue/
gpio label name
# ls /proc/device-tree/leds/led-green/
gpio linux,default-trigger
label name
# cat /proc/device-tree/leds/led-green/linux,default-trigger
heartbeat#
```

A white box with the text "EVB" is located at the bottom right of the terminal window.

```
led-red {
    label = "red";
    gpios = <&pioD 17 GPIO_ACTIVE_HIGH>;
};

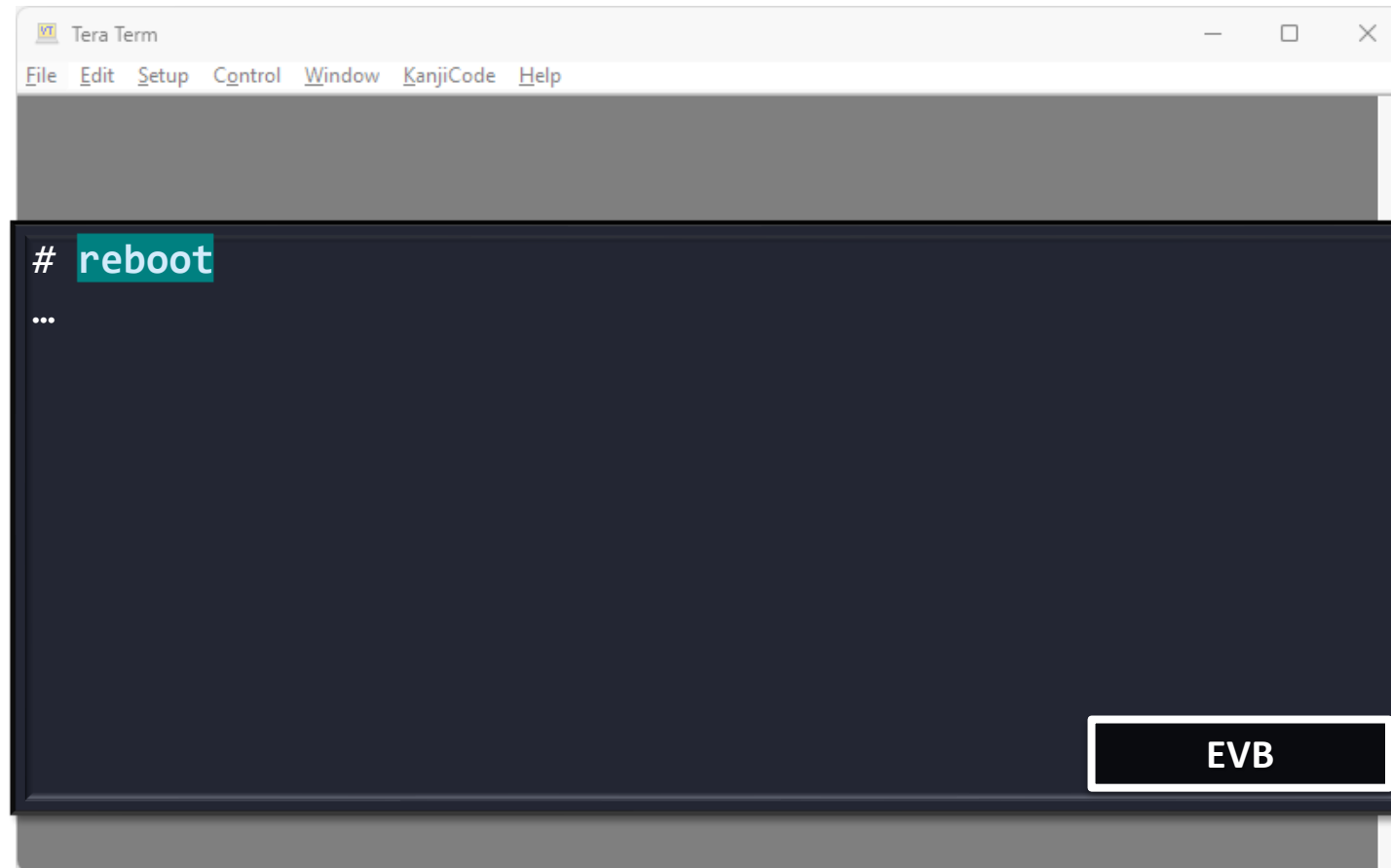
led-green {
    label = "green";
    gpios = <&pioD 19 GPIO_ACTIVE_HIGH>;
    linux,default-trigger = "heartbeat";
};

led-blue {
    label = "blue";
    gpios = <&pioD 21 GPIO_ACTIVE_HIGH>;
};
```

Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Step 9.2

- **Check** the modified part in the Embedded Linux System.

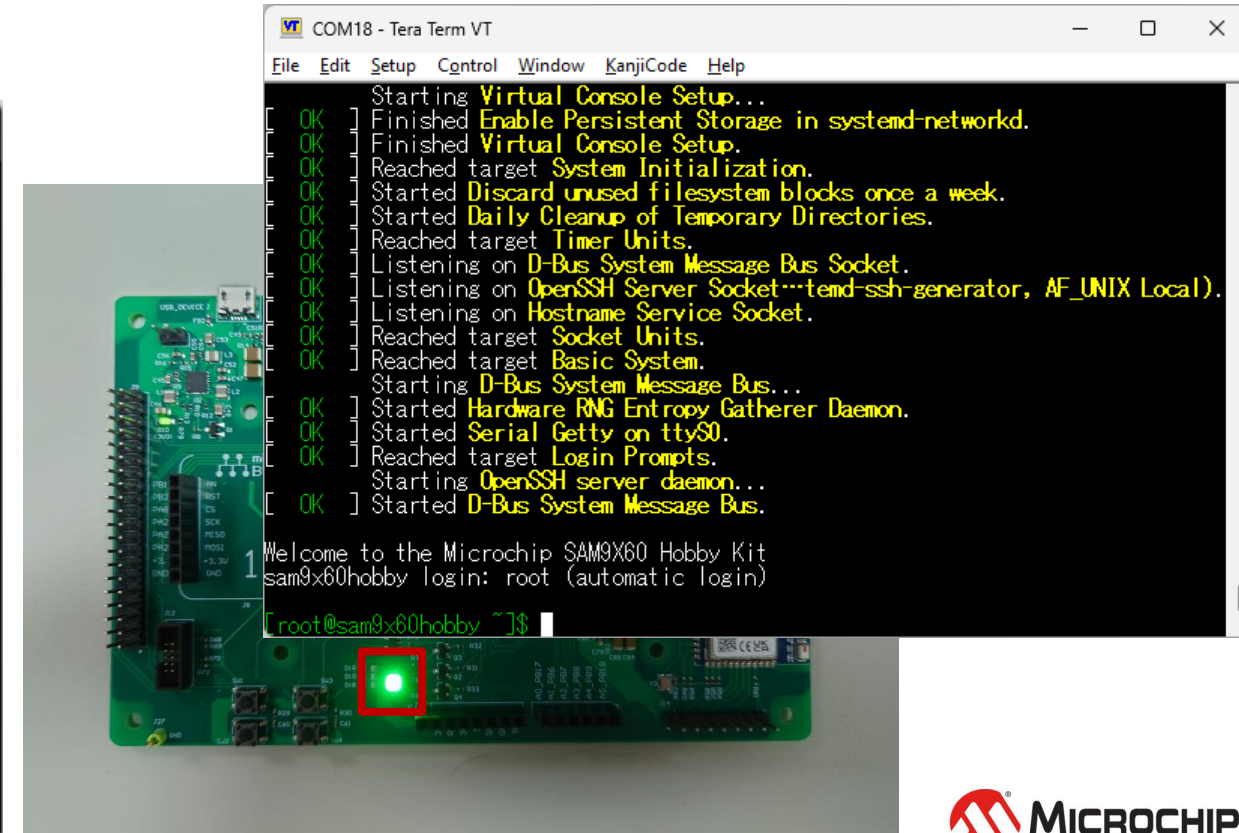


Lab3 – Modify br2-external-tree and Rebuild the Buildroot

Result

- We can find newly modified files containing Hobby Kit related archives in the "**output/**" directory.
- After login the Embedded Linux system, you can find the modified information in the system and see the green led blinking.

```
user@LNX1001: ~/Linux4MCHP x + v
user@LNX1001:~/Linux4MCHP/buildroot-mchp$ ls -l output/images/
total 1782368
-rwxr-xr-x 1 user user 37072 Sep  9 03:36 at91-sam9x60_curiosity.dtb
-rwxr-xr-x 1 user user 27372 Sep 11 07:12 at91-sam9x60_hobby.dtb
-rwxr-xr-x 1 user user 18929 Sep 11 07:12 at91bootstrap.bin
-rwxr-xr-x 1 user user 18929 Sep 11 07:12 boot.bin
-rw-r--r-- 1 user user 16777216 Sep 11 07:13 boot.vfat
-rw-r--r-- 1 user user 943718400 Sep 11 07:13 rootfs.ext2
lrwxrwxrwx 1 user user 11 Sep 11 07:13 rootfs.ext4 -> rootfs.ext2
-rw-r--r-- 1 user user 425656320 Sep 11 07:13 rootfs.tar
-rwxr-xr-x 1 user user 18929 Sep 11 07:12 sam9x60-sdcardboot-uboot-4.0.11.bin
-rw-r--r-- 1 user user 5631664 Sep  9 03:38 sam9x60_curiosity.itb
-rw-r--r-- 1 user user 2053 Sep  9 03:38 sam9x60_curiosity.its
-rw-r--r-- 1 user user 4868 Sep  9 03:38 sam9x60_curiosity_pda5.dtbo
-rw-r--r-- 1 user user 1534 Sep  9 03:38 sam9x60_curiosity_wilc3000.dtbo
-rw-r--r-- 1 user user 5614912 Sep 11 07:12 sam9x60_hobby.itb
-rw-r--r-- 1 user user 1219 Sep 11 07:12 sam9x60_hobby.its
-rw-r--r-- 1 user user 961544192 Sep 11 07:13 sdcard-hobby.img
-rw-r--r-- 1 user user 961544192 Sep  9 04:02 sdcard.img
-rw-r--r-- 1 user user 458952 Sep 11 07:31 u-boot.bin
-rwxr-xr-x 1 user user 16384 Sep 11 07:31 uboot-env.bin
-rw-r--r-- 1 user user 5585520 Sep 11 07:12 zImage
user@LNX1001:~/Linux4MCHP/buildroot-mchp$
```



Use MPIO to Test Peripherals

Microchip Peripheral I/O Python Package (MPIO)

- After we have built an Embedded Linux system suitable for a specific evaluation board, we can use some simple methods to test the peripherals are work normally.
- The [Microchip Peripheral I/O Python Package \(MPIO\)](#) provides a Python API, MPIO allows you to read and write to these hardware peripherals in an easy and consistent manner from Python.
- This package provides easy access to various hardware peripherals :
 - Input
 - LED
 - ADC
 - PWM
 - Serial
 - I2C
 - SPI
 - GPIO
 - Others

```
# python
Python 3.11.8 (main, Aug 13 2024, 15:05:30) [GCC 12.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from mpio import LED
>>> green = LED("green", True)
>>> green.brightness = False
>>> green.brightness = True
```

Lab4 – Use MPIO to Control Peripherals

Microchip Peripheral I/O Python Package

Lab4 – Use MPIO to Control Peripherals

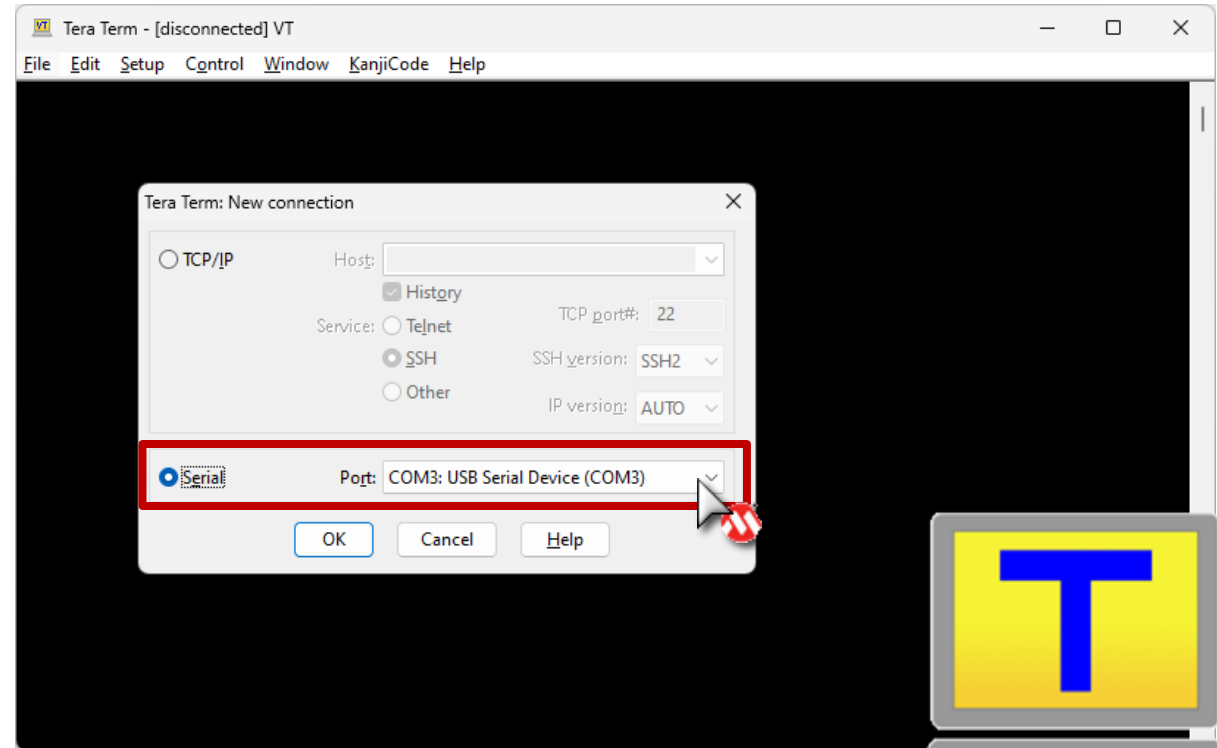
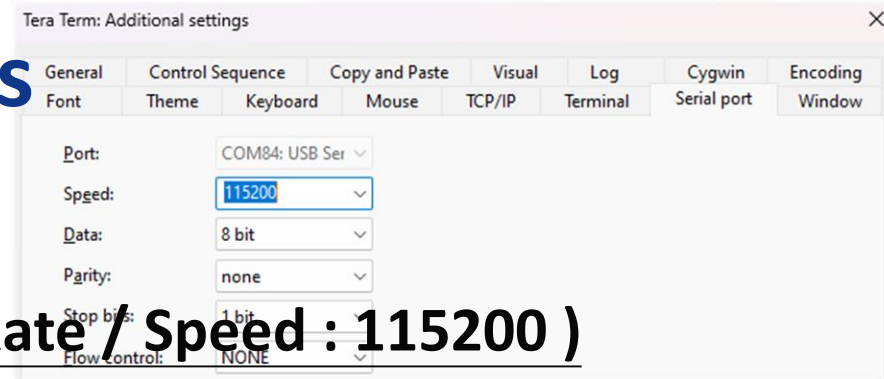
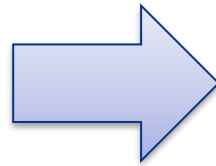
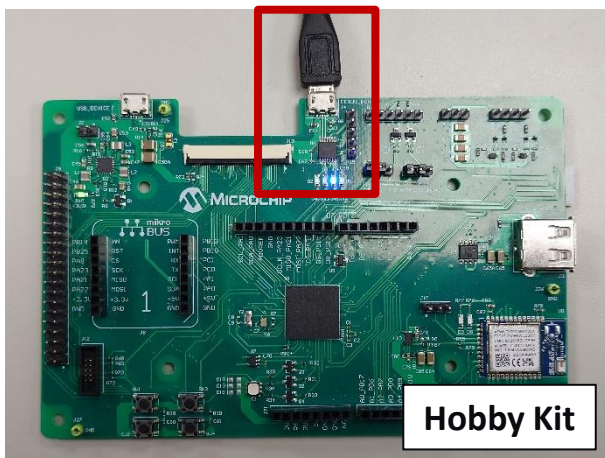
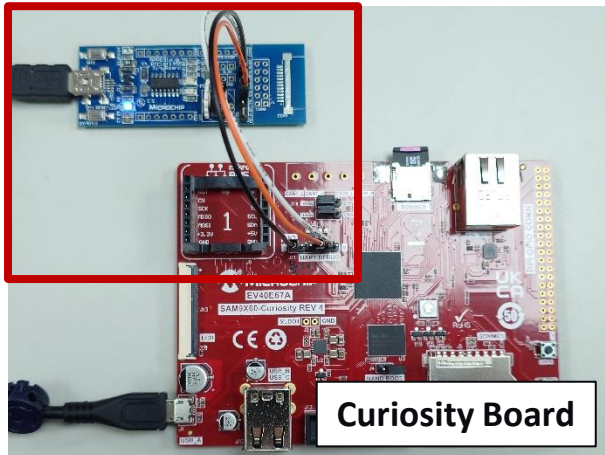
- Refer to the following instructions and examples to try using Microchip Peripheral I/O Python Package (MPIO) to control peripherals.
 - [linux4sam/mpio](https://linux4sam.com/mpio)
 - mpio.readthedocs.io

Let's go!

Lab4 – Use MPIO to Control Peripherals

Step 9

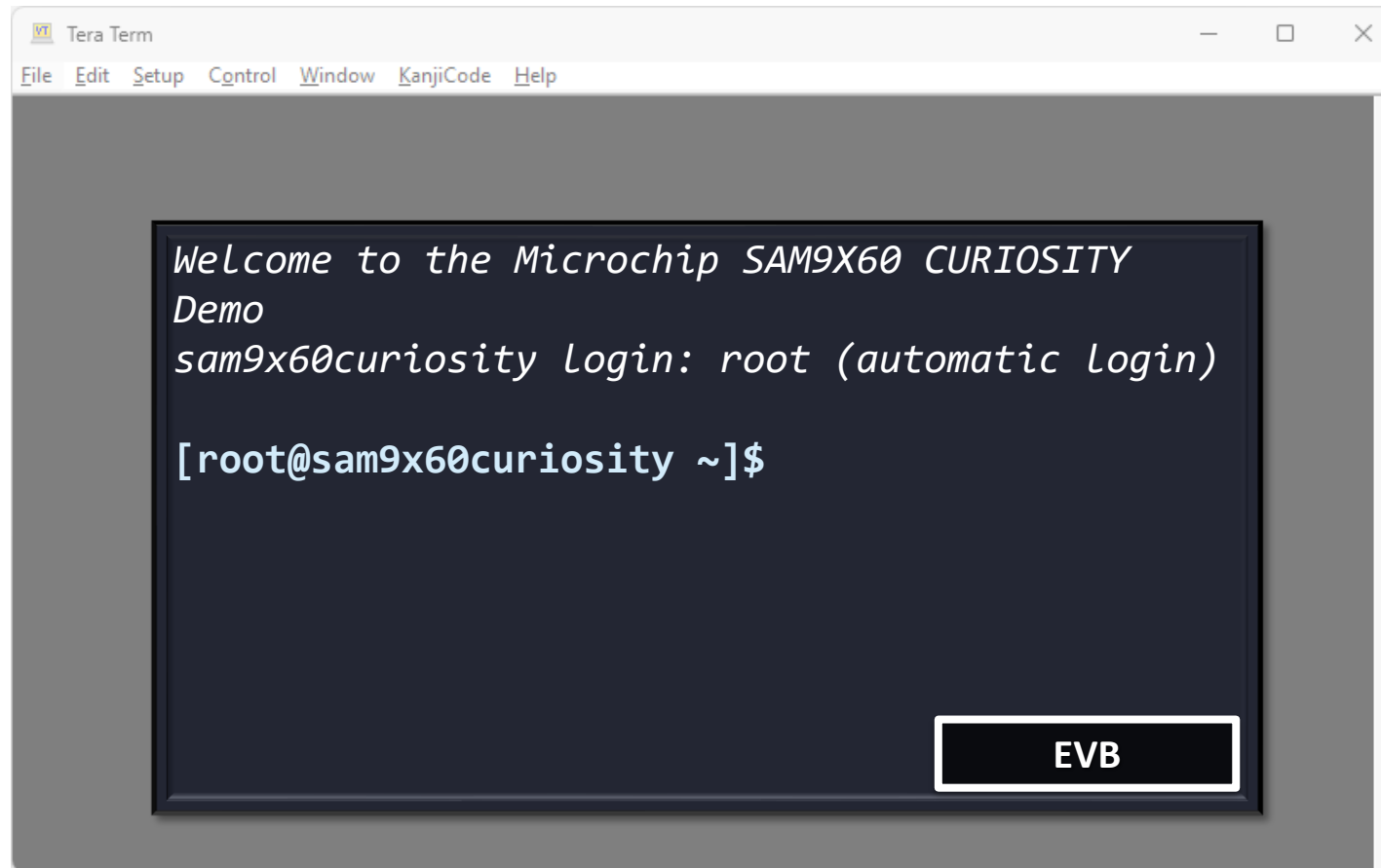
- **Connect the Debug port** of EVB and Tera Term. (**Baud Rate / Speed : 115200**)
(Select the **COM Port** that appears after the connection)



Lab1 – Prepare Development Tools

Step 2

- Use a Terminal (Tera Term) to **login** the Embedded Linux OS of EVB.



The image shows a screenshot of a Tera Term terminal window. The window has a title bar that says "Tera Term" and a menu bar with options: File, Edit, Setup, Control, Window, KanjiCode, and Help. The terminal content displays a welcome message: "Welcome to the Microchip SAM9X60 CURIOSITY Demo", followed by "sam9x60curiosity login: root (automatic login)", and then a shell prompt "[root@sam9x60curiosity ~]\$". A white rectangular box with the text "EVB" is positioned in the bottom right corner of the terminal area.

```
Welcome to the Microchip SAM9X60 CURIOSITY
Demo
sam9x60curiosity login: root (automatic login)

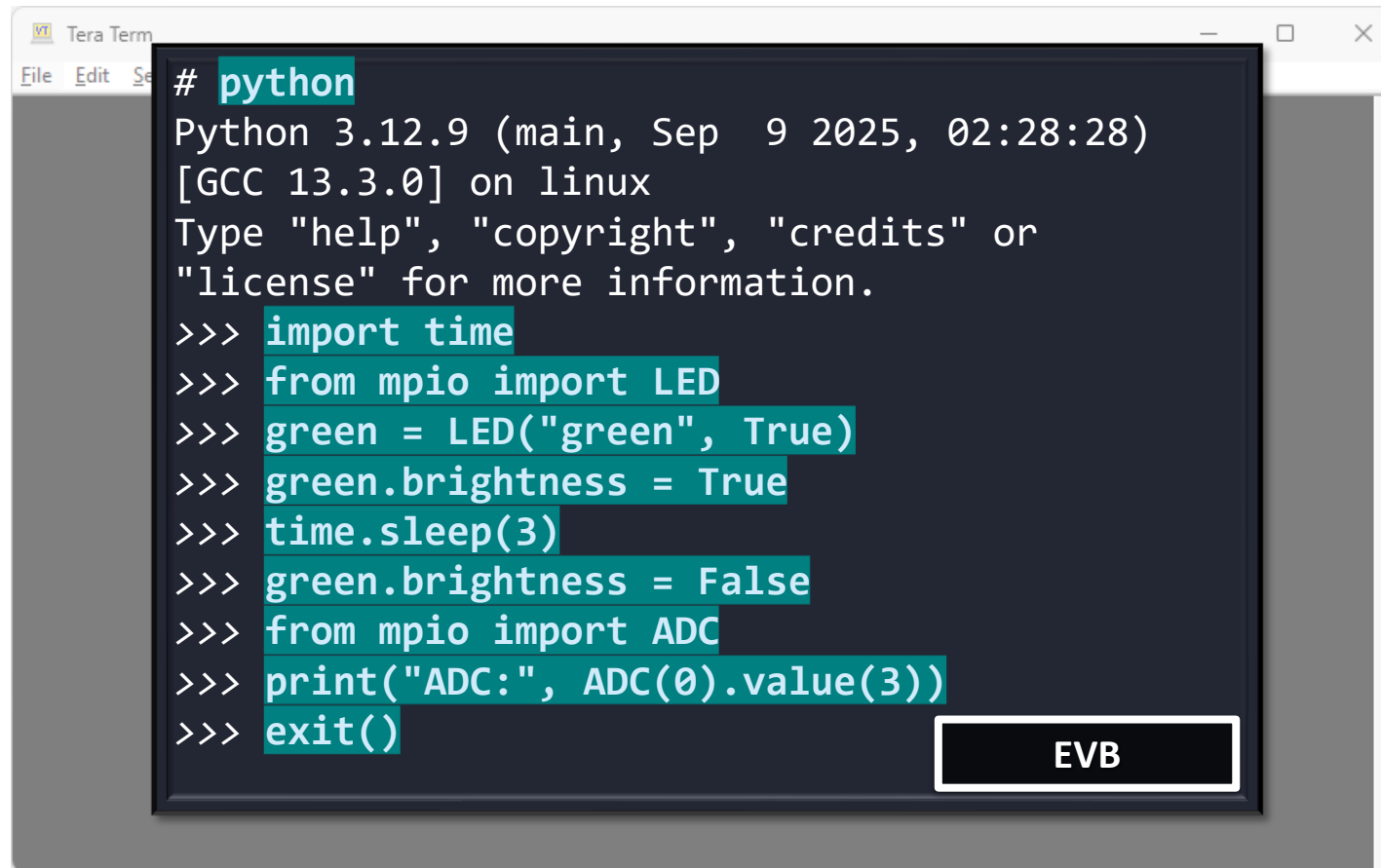
[root@sam9x60curiosity ~]$
```

EVB

Lab4 – Use MPIO to Control Peripherals

Step 3

- **Execute the python with MPIO in the Embedded Linux OS.**



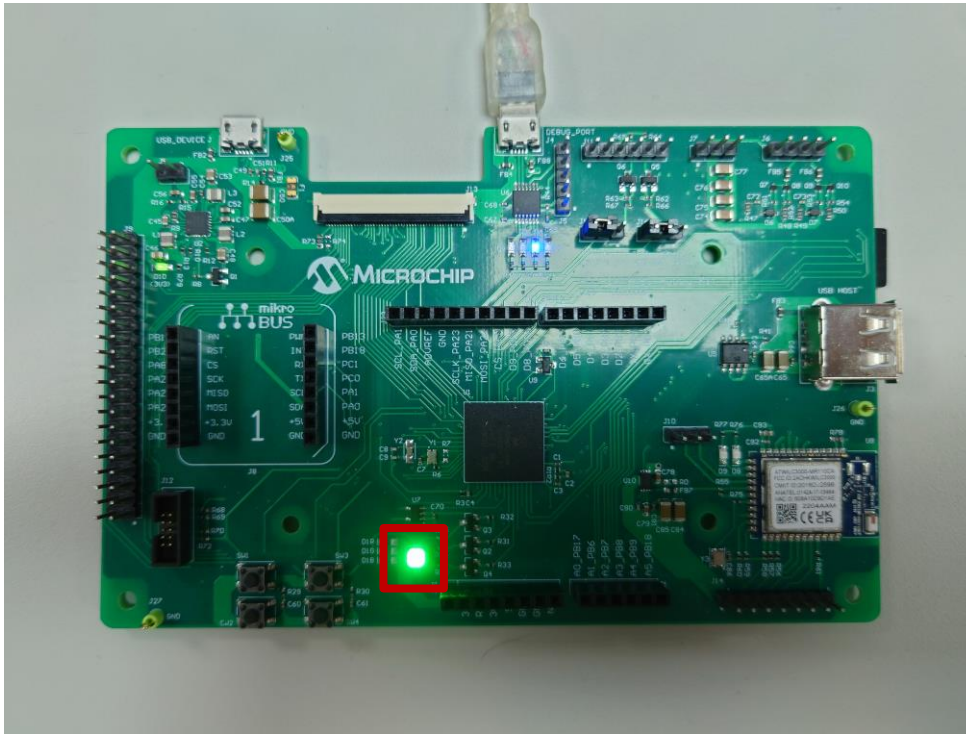
```
# python
Python 3.12.9 (main, Sep  9 2025, 02:28:28)
[GCC 13.3.0] on linux
Type "help", "copyright", "credits" or
"license" for more information.
>>> import time
>>> from mpio import LED
>>> green = LED("green", True)
>>> green.brightness = True
>>> time.sleep(3)
>>> green.brightness = False
>>> from mpio import ADC
>>> print("ADC:", ADC(0).value(3))
>>> exit()
```

EVB

Lab4 – Use MPIO to Control Peripherals

Result

- Using the MPIO, you can quickly test peripheral hardware.



```
user@VM: ~  
# python  
Python 3.11.8 (main, Aug 13 2024, 15:05:30) [GCC 12.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import time  
>>> from mpio import LED  
tness = True  
time.sleep(3)  
green.brightness = False  
from mpio import ADC  
print("ADC:", ADC(0).value(3))  
exit()>>> green = LED("green", True)  
>>> green.brightness = True  
>>> time.sleep(3)  
  
>>> green.brightness = False  
>>> from mpio import ADC  
>>> print("ADC:", ADC(0).value(3))  
ADC: 0  
>>> exit()  
#
```

Microchip Trademarks

<https://www.microchip.com/en-us/about/legal-information/microchip-trademarks>

Trademarks	Description/Appropriate Nouns	Notes
Adaptec® "a" logo		Registered in other countries (not US)
Adaptec®	products, solutions, cables, software, drivers, adapters, controllers, series, family	
Adjacent Key Suppression™	technology	
AgileSwitch®	digital programmable gate driver, gate driver, drivers, family, device	US Only
AKS™	technology	
Analogue-for-the-Digital Age™	AIPD family tagline	
Any Capacitor™	stable	
AnyIn™	feature, structure, capability	
AnyOut™	feature, capability	
Augmented Switching™	technology	
AVR®	devices, microcontrollers, MCUs, CPU, architecture, family	
AVR Logo		
AVR Freaks®	forum	
BesTime®	technology	
BitCloud®	SDK	
BlueSky™	firewall, subscription service, technology	
BodyCom™	technology, system, development kit	
ClockStudio™	application, software, program, utility	
ClockWorks®	Configurator, products, series, family, devices	US Only
CodeGuard™	security	
CryptoAuthentication™	development library, library, devices, ICs, family	
CryptoAutomotive™	technology, security ICs, development kits	
CryptoCompanion™	chip	
CryptoController™	solution, Trusted Platform Module (TPM)	
CryptoMemory®	cryptographic security ICs, ICs	

NOTE: *Always check with the Legal Department for the most current trademarks.

Trademarks	Description/Appropriate Nouns	Notes
PIC ³² Logo		
PICDEM™	demonstration board	
PICDEM.net™	Internet/Ethernet demonstration board	
PICKit™	starter kit, programmer, debugger, in-circuit debugger, development tool, tool, on-board	
picoPower®	technology	
PICSTART®	development programmer	
PICtail™	daughter board, board	
PolarFire®	FPGA, family, kits, devices, SoC, System-on-Chip	
Power MOS IV™		
Power MOS 7™		
Powermite 3®	package	US Only
PowerSmart™	digital control library designer, development suite	
Precision Edge®	family, product family	US Only
ProASIC®	FPGA, architecture, board, kit, family	US Only
ProASIC Plus®	FPGA, devices, family, architecture, switch	US Only
ProASIC Plus Logo		US Only
Prochip Designer®	software suite	
PureSilicon™	oscillator, technology	
QMatrix™	sensor, devices	
QTouch®	sensor, devices, library, tools, development platform, project builder, Composer, Project Builder, board, solution, technology, Configurator	
Quiet-Wire®	enhanced EMI technology, family, technology	
Ripple Blocker™	technology, attenuators, product line, family of linear regulators	
REAL ICE™	In-circuit emulator	
RTAX™	FPGAs	
RTG4™	FPGAs	
SAM-BA®	software, application, monitor	

NOTE: *Always check with the Legal Department for the most current trademarks.

Trademarks	Description/Appropriate Nouns	Notes
CryptoRF®	transponder, devices	
dsPIC®	Digital Signal Controllers (DSCs)	
dsPIC Logo		
dsPICDEM™	demonstration board, development board	
dsPICDEM.net™	board	
Dynamic Averaged Matching™ (DAM™)	architecture	
ECAN™	technology, solution	
Espresso T1S™	device	
EtherGREEN™	platform, technology, family	
EtherSynch®	platform, technology, family	US Only
EyeOpen™	cable compensation	
Flashtec™	controller, products, family	US Only
flexPVR®	technology	
Frequency on Demand®		Registered in other countries (not US)
GestIC®	technology, sensing, controllers	
GridTime™	device, time server, GNSS time server	
HELDO®	family, regulators	
Hyper Speed Control®	family, architecture	US Only
HyperLight Load®	mode	US Only
ICSP™ or In-Circuit Serial Programming™	programming capability	
IdealBridge™	rectifier, dual MOSFET-bridge rectifier	
IGAT™	board, demonstration board	
IGLOO®	FPGAs, family, devices, series	
INICNet™	technology, sniffer	
Intelligent Paralleling™	technology	
IntelliMOS™	family, power stage family	US Only
Inter-Chip Connectivity™	technology	
JitterBlocker™	family, attenuators	
JukeBlox®	technology, platform	

NOTE: *Always check with the Legal Department for the most current trademarks.

Trademarks	Description/Appropriate Nouns	Notes
SAM-ICE™	emulator	
SenGenuity®	sensors, products	
Serial Quad I/O™ or SQI™	family, flash device, product	
Silicon Storage Technology®		Japan only
SimpleMAP™	protocol	
SimpliPHY™	family, products, devices	
SmartBuffer™	devices, ICs	
SmartFusion®	FPGA, evaluation kit, SoC, System-on-Chip,	US Only
SmartHLS™	IDE, compiler software, compiler, software	
SMART-I.S.™	technology	
SpyNIC®	device	
SST®		
SST Logo		
storClad™	encryption technology, technology	
SuperFlash®	technology, kit, macro, device, memory	
SuperSwitcher™	family, regulator, buck regulator	
SuperSwitcher II™	family	
Switchtec™	technology, family, switch(es), product line	China only
Symmmcom®		
Symmetricon®		
SynchroPHY™	family, products, devices	
SyncServer®	server, instrument, clock, oscillator, device	
SyncWorld®	timing, Ecosystem Program	US only
Tachyon®	products, family, controllers, platform, technology	
The Embedded Control Solutions Company®	MarCom Use Only	
TimeCesium®	products, primary frequency reference	US only
TimeHub®	platform, system	US only
TimePictra®	software suite, platform, Synchronization Management System	US only

NOTE: *Always check with the Legal Department for the most current trademarks.

Trademarks	Description/Appropriate Nouns	Notes
JukeBlox with design	Registered word mark JukeBlox with a design	Image is not registered, this is a graphic treatment with the word mark
KEELOQ®	security ICs, protocol, technology	
Kleer®	technology	
Kleer with design	Registered word mark Kleer with a design	Image is not registered, this is a graphic treatment with the word mark
Knob-on-Display™/KoD™	technology, solution, HMI, user interface, touch controller, family, devices, Widget Configurator, Knob Designer, Position Wizard	
LANCheck®	online design review, online review, review	
Libero®	SoC Design Suite, tools, software, design files, IDE, designer	US registration
LinkMD®	cable diagnostics, technology, family, diagnostics engine	
MarginLink™	signal integrity testing	
maxCrypto™	controller-based encryption, encryption	
maXStylus™	solution	
maXTouch®	touchscreen controllers, technology, studio, family	
maxView™	storage manager, tools, drivers	
MediaLB®	bus, controller, device	
megaAVR®	devices, MCUs	
memBrain™	neuromorphic memory product, product, solution, technology	
Microchip Logo		
Microchip Name and Logo		
Microsemi Logo		
Microsem®		

NOTE: *Always check with the Legal Department for the most current trademarks.

Trademarks	Description/Appropriate Nouns	Notes
TimeProvider®	series, server, clock, products, grandmaster, ePRTC, enhanced PRTC	US only
TimeSource®	Enhanced PRTC	
tinyAVR®	microcontroller, MCU	
Total Endurance™	software model	
Trusted Time™	technology	
TSHARC™	touch screen controller, controller	
Turing™	hardware security module, programmer	
UNIQ®	memory chips, bus, hardware port, communication protocol, firmware	
USBCheck™	design review	
VariSense™	technology	
VectorBlox™	accelerator, software development kit, SDK, intellectual property, IP, solution, platform, tools, tool chain, tool flow	
Vector®		
VeriPHY™	cable diagnostics algorithm, cable diagnostics software suite	
ViewSpan™	technology	
WiperLock™	technology	
XMEGA®	microcontroller, MCU, devices, Custom Logic (XCL), family, series	
XpressConnect™	retimer	
ZENA™	software, analyzer or wireless adapter	
ZL®		

NOTE: *Always check with the Legal Department for the most current trademarks.

Microchip Technology Inc. Servicemarks*

Trademarks	Description/Appropriate Nouns	Notes
SQTP™	Serial Quick Turn Programming capability	
NOTE: *Always check with the Legal Department for the most current servicemarks.		
MICROCHIP RESERVES THE RIGHT TO CHANGE THESE GUIDELINES SOLELY AT ITS DISCRETION.		

Trademarks	Description/Appropriate Nouns	Notes
MindI™	analog simulator, analog simulator tool	
MIWI™	wireless networking protocol, protocol, network, stack, software, applications, mesh, coordinator	
MOST®	technology, NetServices, evaluation kit, board, system, device, product, network, specification, platform, cooperation, design	
MOST Logo		
motorBench®	Development Suite	US Only
MPASM™	assembler	
MPF™	products, devices	
MPLAB®	Integrated Development Environment, In-Circuit Debugger, In-Circuit Emulator, IDE, ICD, REAL ICE™ in-circuit emulator, starter kit, compiler, XC compiler, C compiler, Harmony, Integrated Programming Environment, IPE, Xpress, Code Configurator, development ecosystem, PICkit™, server, library, Snap, Connect Configurator, Network Creator, Analog Designer, Discover, MindI™, Analysis Tool Suite, Discover, Universal Device Programmer, Machine Learning Development Suite, PTG, Programmer-to-Go, Programmer-to-Go Mobil App, SiC Power Simulator	
MPLAB Certified Logo		
MPLIB™	object librarian or librarian	
MPLINK™	object linker or linker	
mSIC™	MOSFETs, diodes, products, gate drivers, modules, bare die, technology, solutions	
mTouch®	solution, sensing solution, evaluation kit, technology, development kit, library	US Only
MultiTRAK™	technology	
NetDetach™	technology	
Omniscient Code Generation™ (OCG)	compilation technology	
OptoLyzr®	tool, network analysis tool, Studio	
PIC®	microcontroller, MCU, device(s), assembler	

NOTE: *Always check with the Legal Department for the most current trademarks.

