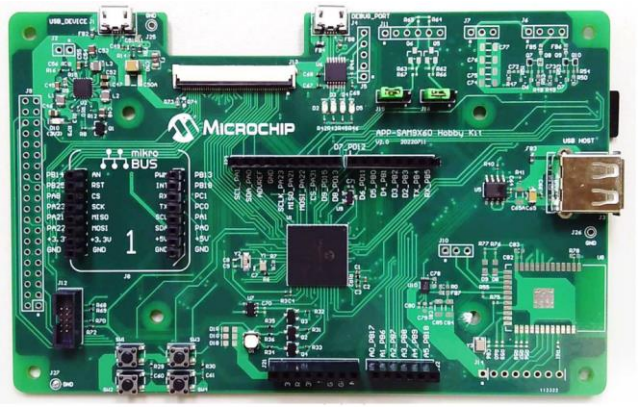
SAM9x60 Hobby Porting Guide

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



SMART | CONNECTED | SECURE

Goals

- How to modify customer board from official EVK
- How to modify Linux Kernel device tree
- How to build buildroot image
- How to make patch for releasing
- How to customize buildroot package
- How to use cross compiler to development application

Before Start

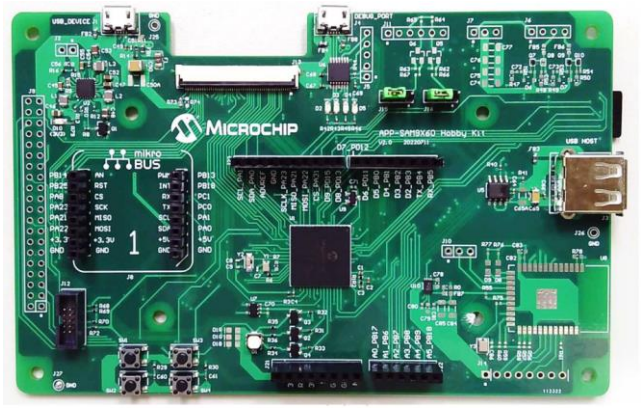
- **When you read this document, your will see two symbols, “$” and “#”**
- **Those two symbols are indicate the user account type you’re logged in to. The dollar sign means you’re the normal user. The hash sign means you’re system administrator.**
- **The following contents means the command that you need to input in host or device console.**
  - “$” means the command for ***host*** since default user is ***normal user***
  - “#” means the command for ***device*** since default user is ***superuser***

Make your board design based on official EVK

Customer Board

- **Most of customers referred to official EVK to development their boards.**
  - Reduce design risk
  - Improve development schedule
  - Peripheral components are validated

SAM9x60 Curiosity

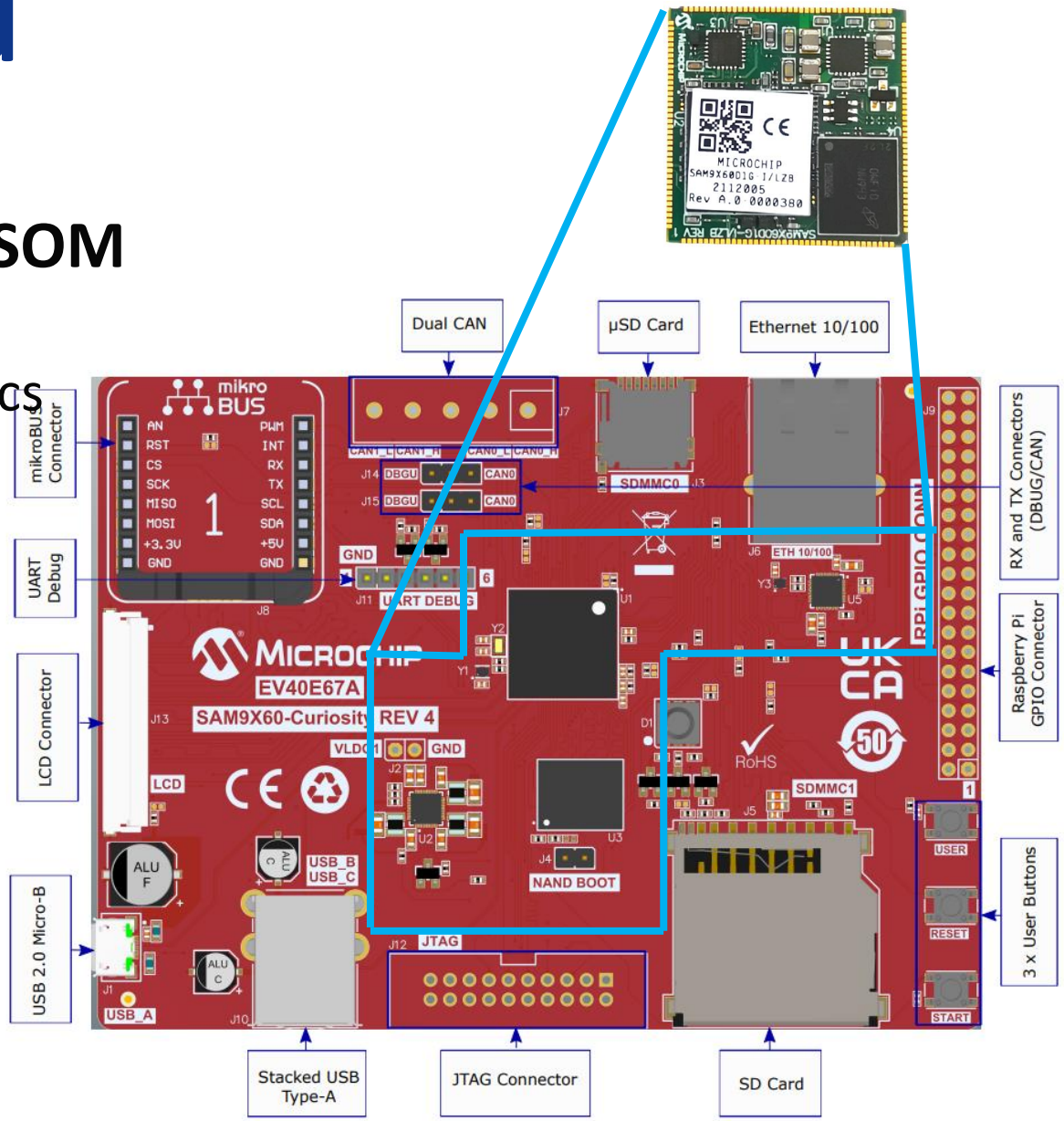


SAM9x60 Hobby

SAM9X60 Curiosity Board

Part Number: EV40E67A

- **100% Compatible with the SAM9X60 SOM**
  - Even if not using the SOM
  - Uses the same devices and same schematics
- **Development in progress**
  - Layout completed
  - Waiting BOM Compliance approval
- **Will allow evaluation of**
  - The SAM9X60
  - The SAM9X60-SIP
  - The SAM9X60-SOM
- **Main benefit**
  - One board to support and maintain



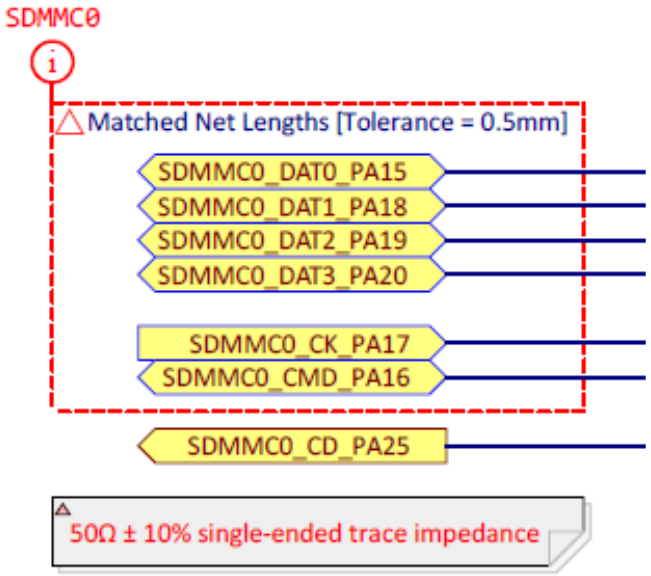
MICROCHIP

Comparison

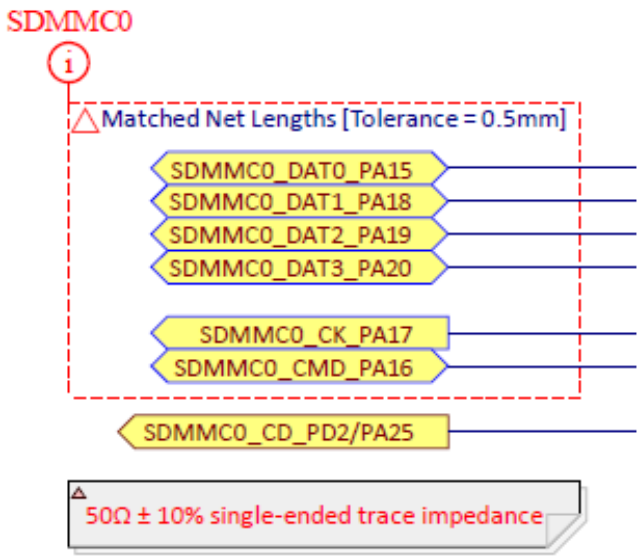
	SAM9x60 Curiosity	SAM9x60 Hobby
Chip	SAM9x60-D1G, 1Gb SiP	SAM9x60-D1G, 1Gb SiP
NAND Flash	4Gb	N/A
SD/eMMC	uSD/SD	uSD
Ethernet	10/100 PHY	N/A
Wireless	N/A	WILC3000 WiFi/Bt Combo
LCD	RGB parallel	RGB parallel
CAN	Dual CAN 2.0	N/A
Usb UART	N/A	MCP2221
Sensors	N/A	MCP9700/MCP9800
USB Host	X2	X1
I/O	RGB LEDs, User button x1 Raspberry pi GPIO connector, MicroBus	RGB LEDs, User button x2 Raspberry pi GPIO connector, MicroBus Arduino Shield
Audio	N/A	Class D

Differentiation

SD Card detect



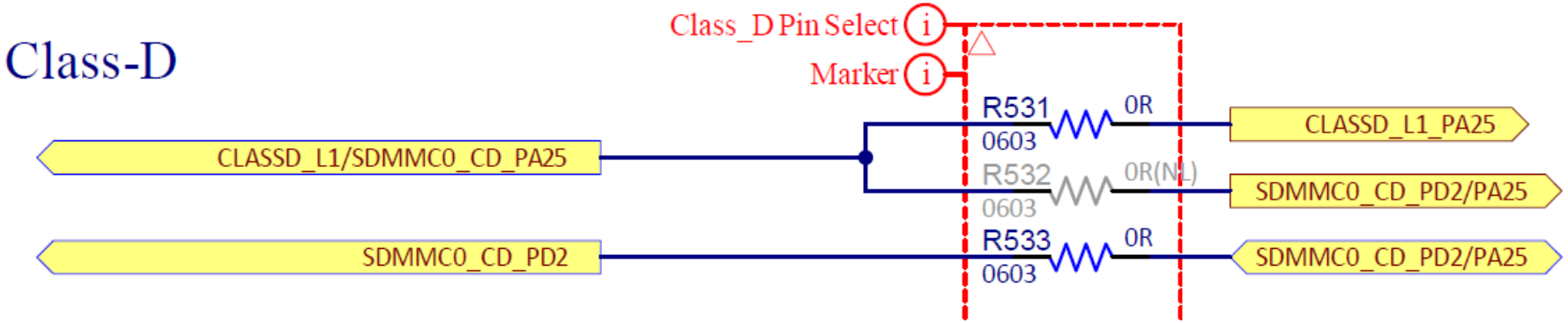
SAM9x60 Curiosity



SAM9x60 Hobby

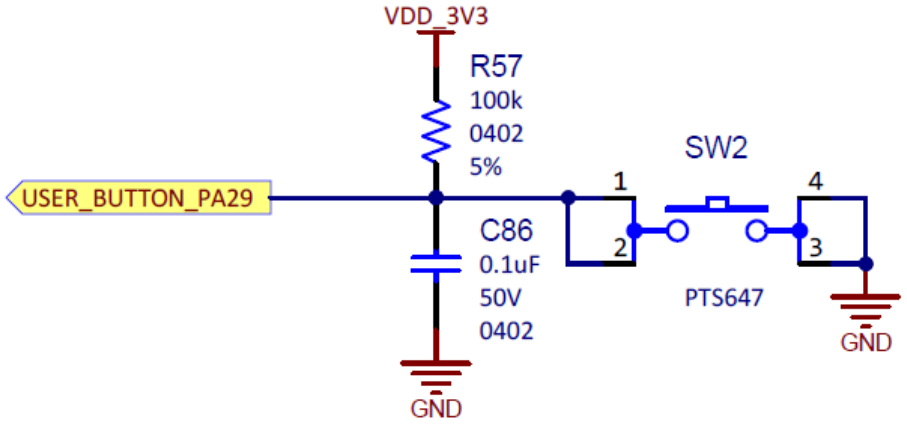
Idea

- The SD Card pin assignment are same basically.
- But PA25 will be used for Class-D.
- Used PD2 for SD card detect instead of PA25. PA25 will be used for Class-D

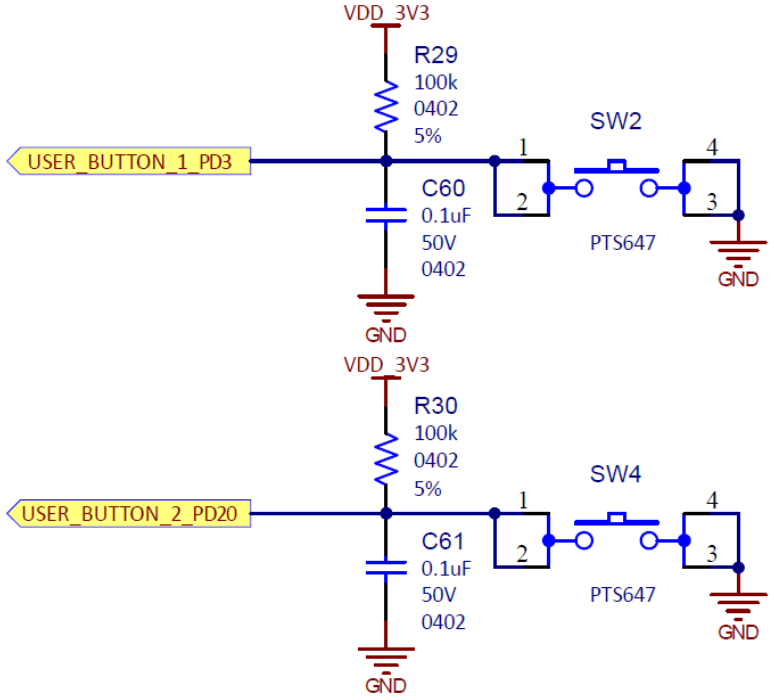


Differentiation

User Buttons



SAM9x60 Curiosity



SAM9x60 Hobby

Idea

- There is only one user button in Curiosity and PA29 is the I/O pin.
- Add one extra user button in Hobby and using PD3 and PD20 for I/O pins.

Modify Linux Kernel Device Tree for your board

Linux Kernel Device Tree & Device Tree Overlay

- The “Open Firmware Device Tree”, or simply Device tree (DT), is a data structure and language for describing hardware.
- More specifically, it is a description of hardware that is readable by an operating system so that the operating system doesn’t need to hard code details of the machine.
- A device tree overlay (DTO) enables a central device tree blob (DTB) to be overlaid on the device tree. A bootloader using DTO can maintain the system-on-chip (SoC) DT and dynamically overlay a device-specific DT, adding nodes to the tree and making changes to properties in the existing tree.

SAM9x60 Kernel Resource

- **Kernel Source Code**

  https://github.com/linux4microchip/linux

- **SAM9x60 Device Tree file**

  https://github.com/linux4microchip/linux/blob/linux-5.15-mchp/arch/arm/boot/dts/at91-sam9x60_curiosity.dts

- **Device Tree Overlay Source**

  https://github.com/linux4microchip/dt-overlay-mchp

Preprocessing

- **Learn how to install and setup development environment in your laptop**

  https://www.youtube.com/watch?v=8sDyjYb8OAY&t=613s

SD Card Detect

https://github.com/linux4microchip/linux/blob/linux-5.15-mchp/arch/arm/boot/dts/at91-sam9x60_curiosity.dts

```
&sdmmc0 {
        bus-width = <4>;
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl_sdmmc0_default &pinctrl_sdmmc0_cd>;
        cd-gpios = <&pioA 25 GPIO_ACTIVE_LOW>;
        disable-wp;
        status = "okay";
};
```

SAM9x60 Curiosity

```
&sdmmc0 {
        bus-width = <4>;
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl_sdmmc0_default &pinctrl_sdmmc0_cd>;
        cd-gpios = <&pioD 2 GPIO_ACTIVE_LOW>;
        disable-wp;
        status = "okay";
};
```

SAM9x60 Hobby

SD Card Detect

https://github.com/linux4microchip/linux/blob/linux-5.15-mchp/arch/arm/boot/dts/at91-sam9x60_curiosity.dts

```
pinctrl_sdmmc0_cd: sdmmc0_cd {
        atmel,pins =
                <AT91_PIOA 25 AT91_PERIPH_GPIO AT91_PINCTRL_NONE>;
};
```

SAM9x60 Curiosity

```
pinctrl_sdmmc0_cd: sdmmc0_cd {
        atmel,pins =
                <AT91_PIOD 2 AT91_PERIPH_GPIO AT91_PINCTRL_NONE>;
};
```

SAM9x60 Hobby

User Buttons

https://github.com/linux4microchip/linux/blob/linux-5.15-mchp/arch/arm/boot/dts/at91-sam9x60_curiosity.dts

```
gpio-keys {
        compatible = "gpio-keys";
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl_key_gpio_default>;
        status = "okay";

        button-user {
                label = "PB_USER";
                gpios = <&pioA 29 GPIO_ACTIVE_LOW>;
                linux,code = <KEY_PROG1>;
                wakeup-source;
        };
};
```

SAM9x60 Curiosity

```
gpio-keys {
        compatible = "gpio-keys";
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl_key_gpio_default>;
        status = "okay";
        sw2 {
                label = "PB_USER_SW2";
                gpios = <&pioD 3 GPIO_ACTIVE_LOW>;
                linux,code = <KEY_PROG1>;
                wakeup-source;
        };
        sw4 {
                label = "PB_USER_SW4";
                gpios = <&pioD 20 GPIO_ACTIVE_LOW>;
                linux,code = <KEY_PROG2>;
                wakeup-source;
        };
};
```

SAM9x60 Hobby

User Buttons

https://github.com/linux4microchip/linux/blob/linux-5.15-mchp/arch/arm/boot/dts/at91-sam9x60_curiosity.dts

```
gpio_keys {
        pinctrl_key_gpio_default: pinctrl_key_gpio {
                atmel,pins = <AT91_PIOA 29 AT91_PERIPH_GPIO AT91_PINCTRL_NONE>;
        };
    };
```

SAM9x60 Curiosity

```
gpio_keys {
        pinctrl_key_gpio_default: pinctrl_key_gpio {
                atmel,pins =
                <AT91_PIOD 3 AT91_PERIPH_GPIO AT91_PINCTRL_NONE
                AT91_PIOD 20 AT91_PERIPH_GPIO AT91_PINCTRL_NONE>;
        };
    };
```

SAM9x60 Hobby

How to Build Linux Kernel

- **Useful link:**

  https://www.linux4sam.org

  https://www.linux4sam.org/bin/view/Linux4SAM/LinuxKernel

- **Procedures**

  $ git clone https://github.com/linux4microchip/linux.git -b sam9x60-curiosity-2022.07

  $ cd linux

  Modify arch/arm/boot/dts/at91-sam9x60_curiosity.dts

  $ ARCH=arm make at91_dt_defconfig

  $ ARCH=arm make

  zImage will be placed at arch/arm/boot

How to Build Device Tree Overlay

- **The device tree overlay is a standalone project and will refer to the kernel path.**
- **Need to export KERNEL_DIR**

  // in kernel working path. For example, /home/USER_NAME/linux

  $ export KERNEL_DIR=$PWD

  Or use absolute path name

  $ export KERNEL_DIR=/home/USER_NAME/linux
- **Procedures**

  $ git clone https://github.com/linux4microchip/dt-overlay-mchp.git -b linux4microchip+sam9x60-curiosity-2022.07

  $ cd dt-overlay-mchp

  $ ARCH=arm make sam9x60_curiosity.dtbos

  $ ARCH=arm make sam9x60_curiosity.itb

Preparing SD Card Image

- Download SAM9x60 Curiosity prebuilt image

  Here

- Flash image to SD card: using Ether

  Download here

  How to do here

- There are two partitions in SD Card.
- The FAT16/32 partition is boot partition
- The EXT4 partition is Linux root filesystem partition
- Replace sam9x60_curiosity.itb in boot partition

Bootup

- **Insert uSD Card to uSD slot in Hobby board**
- **Connect uUSB cable between laptop and J4 in Hobby board**
  - J1: USB_DEVICE. Power source and SAM-BA monitor update
  - J4: DEBUG_PORT. Power source and debug log via MCP2221
  - Use terminal software, such as TeraTerm in laptop

COM10 - Tera Term VT

File Edit Setup Control Window Help

```
Starting iptables: OK
Starting bluetoothd: OK
Starting network: udhcpc: started, v1.35.0
macb f802c000.ethernet eth0: PHY [f802c000.ethernet-ffffffff:00] driver [Generic PHY] (irq=POLL)
macb f802c000.ethernet eth0: configuring for phy/rmii link mode
macb f802c000.ethernet eth0: Link is Up - 100Mbps/Full - flow control tx
IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
udhcpc: broadcasting discover
udhcpc: broadcasting discover
udhcpc: broadcasting discover
udhcpc: no lease, failing
Failed to bring up eth0.
FAIL
Starting dropbear sshd: OK
Starting radvd: OK
Starting sshd: OK
Starting DHCP server: FAIL
Starting EGT Application Launcher: OK

Welcome to the Microchip SAM9X60 CURIOSITY Demo
sam9x60_curiosity login:
```

Default username is root
No password

Testing

User buttons

- **If gpio-key driver hook well, there will be a input event file node in /dev/input/eventX**

```
# ls /dev/input/
by-path  event0
#
```

- **Check event type**

  # cat /proc/bus/input/devices

```
# cat /proc/bus/input/devices
I: Bus=0019 Vendor=0001 Product=0001 Version=0100
N: Name="gpio-keys"
P: Phys=gpio-keys/input0
S: Sysfs=/devices/platform/gpio-keys/input/input0
U: Uniq=
H: Handlers=kbd event0
B: PROP=0
B: EV=3
B: KEY=300000 0 0 0 0
```

Testing

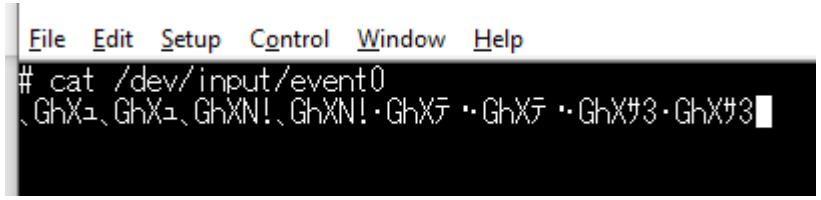
User buttons

- **Test key event**

  # cat /dev/input/event0

Press SW2 & SW4

See what does it happen in terminal?



Press ctrl+z exit

What is that?
Don't worry. We will talk about it later.
Make sure there is something shown when you press both keys.

Testing

LEDs

- **The LED devices are located at /sys/class/leds**

```
File  Edit  Setup  Control  Window  Help
# cd /sys/class/leds/
#
# ls
blue    green   mmc0::  red
#
```

- **Three LEDs have been integrated**
- **LED functions**
  - brightness: on/off LED
  - trigger: trigger mode. Use cat to check what modes are supported

Testing

LEDs

- **Turn on RED LED**

  # echo 1 >> /sys/class/leds/red/brightness

- **Turn off RED LED**

  # echo 0 >> /sys/class/leds/red/brightness

- **Stop BLUE trigger**

  # echo none >> /sys/class/leds/blue/trigger

- **Enable heartbeat trigger with BLUE**

  # echo heartbeat >> /sys/class/leds/blue/trigger

Generate Linux Kernel Patch

- Using git command to compare the different what we modified
- This patch can be released to others or used when building buildroot image

```
// in kernel folder
$ git add -A
$ git diff --cached >> PATCH_FILENAME
```

https://github.com/s887432/sam9x60_hobby-patch/blob/main/0000_sam9x60_hobby_linux_dt.oatch

Build Buildroot Image

Linux4Sam Buildroot

- https://www.linux4sam.org/bin/view/Linux4SAM/BuildRoot

- **Download source code**
  - $ git clone https://github.com/linux4sam/buildroot-at91.git -b sam9x60-curiosity-2022.07
  - $ git clone https://github.com/linux4sam/buildroot-external-microchip.git -b sam9x60-curiosity-2022.07
- **Apply kernel patch**
  - $ mkdir -p buildroot-external-microchip/patches/linux
  - $ cp PATCH_FILENAME buildroot-external-microchip/patches/linux

Linux4Sam Buildroot

- **Build image**

  $ cd buildroot-at91

  $ BR2_EXTERNAL=../buildroot-external-microchip make sam9x60_curiosity_graphics_defconfig

  $ make -jn

  where n could be double of CPU core

It will take long time to build image depends on CPU and network performance.

- **When build successed, the SD card image will be generated at buildroot-at91/output/images/sdcard.img**

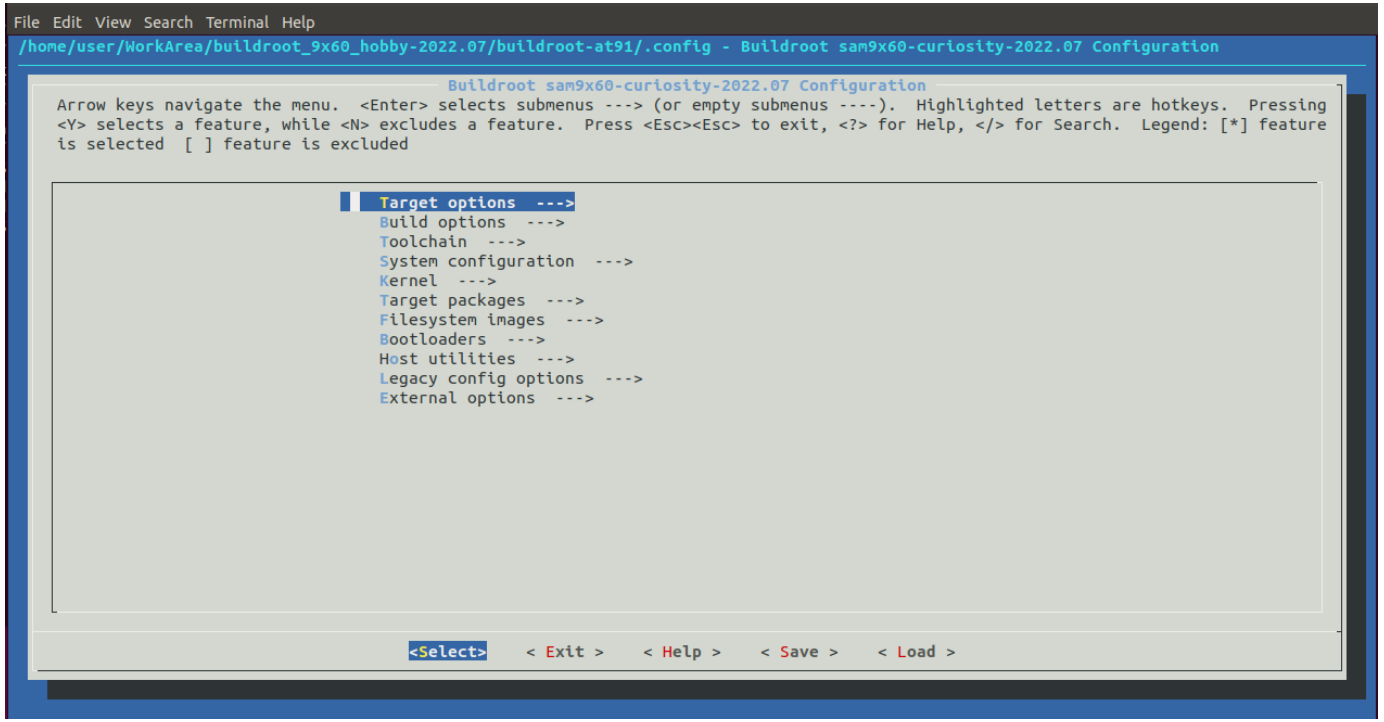
- **You can use etcher or Linux command to flash SD card**

  $ sudo dd if=output/images/sdcard.img of=/dev/sd**b** bs=4M

Customize your Buildroot

manuconfig and defconfig

- Manuconfig is used in Buildroot which provides an easy and simple way to configure your environment.
- The defconfig file describes the default setting and selected packages.
- Using arrow key to move cursor
- Use space to select or deselect



$ make menuconfig

Host name and welcome message

```
Welcome to the Microchip SAM9X60 HOBBY Demo
sam9x60_hobby login:
```

Welcome message

Host Name

- **Solution 1: defconfig**
  - https://github.com/linux4sam/buildroot-external-microchip/blob/master/configs/sam9x60_curiosity_graphics_defconfig

BR2_TARGET_GENERIC_HOSTNAME="**sam9x60_curiosity**"

BR2_TARGET_GENERIC_ISSUE="**Welcome to the Microchip SAM9X60 CURIOSITY Demo**"

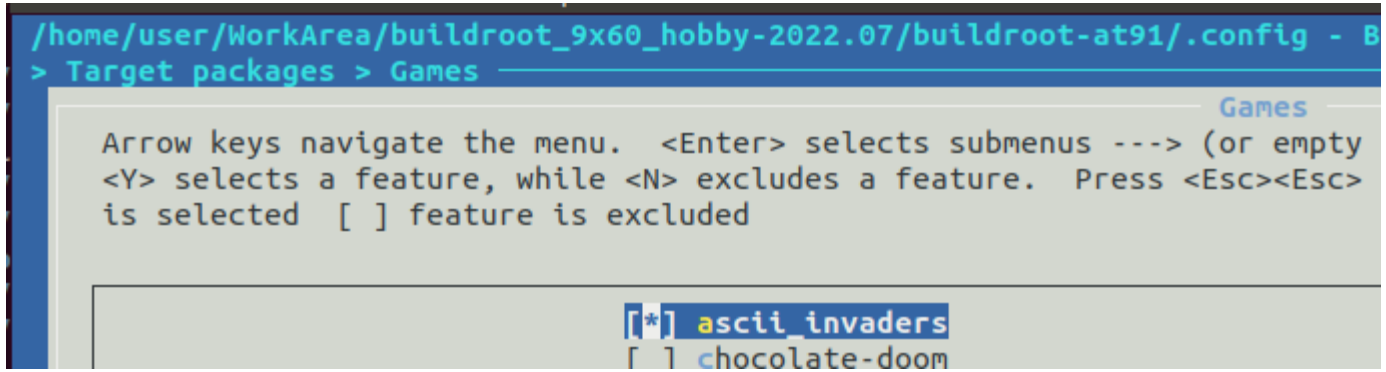
- **Solution 2: menuconfig**
  - Host name: [System configuration]➔[System hostname]
  - Welcome message: [System configuration]➔[System banner]

Add Game: ascii invaders

- **An ASCII-art game like Space Invaders using Curses.**
- **https://github.com/macdice/ascii-invaders**

- **Manuconfig**

  [Target packages]➔[Games]➔[ascii_invaders]



 # ascii_invaders

Build Application

Cross Compiler

- **Prebuilt cross compilers are workable. Check Linux4Sam for more detail. https://www.linux4sam.org/bin/view/Linux4SAM/Sam9x60CuriosityMainPage#Setup_ARM_Cross_Compiler**

- **More suitable is using the cross compiler which built by buildroot.**
  - Location: buildroot-at91/output/host
- **When added extra libraries, the header files and libraries will be installed. No need to build cross compiler version in your host**

Cross Compiler

- **For example. The buildroot folder is located at /home/user/buildroot-at91**
- **Export cross compiler**

  $ gedit ~/.bachrc

  Add following item in end of file

  export PATH=$PATH:/home/user/buildroot-at91/output/host/bin

  Test

  $ arm-buildroot-linux-gnueabi-gcc

```
File  Edit  View  Search  Terminal  Help
user@user-VirtualBox:~$ arm-buildroot-linux-gnueabi-gcc
arm-buildroot-linux-gnueabi-gcc.br_real: fatal error: no input files
compilation terminated.
user@user-VirtualBox:~$
```

Hello World: Cross-Compile

- Hello World

```
#include <stdio.h>
int main(int argc, char **argv)
{
        printf("Hello World!!!\r\n");
        return 0;
}
```

```
1 #include <stdio.h>
2
3 int main(int argc, char **argv)
4 {
5         printf("Hello World!!!\r\n");
6         return 0;
7 }
8
9 // end of file
10
```

- compile

$ arm-buildroot-linux-gnueabi-gcc -o helloworld helloworld.c

```
user@user-VirtualBox:~/WorkArea/app/hello$ arm-buildroot-linux-gnueabi-gcc -o helloworld helloworld.c
user@user-VirtualBox:~/WorkArea/app/hello$
user@user-VirtualBox:~/WorkArea/app/hello$ file helloworld
helloworld: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.3, for GNU/Linux 5.4.0,
 not stripped
user@user-VirtualBox:~/WorkArea/app/hello$
```

Download file to board

- **There are several ways to download compiled binary for files to board.**
  - via ethernet/WiFi
  - USB mass storage

Buildroot won't mount USB disk automatically in default. We need to do it manually.
Will introduce how to do auto-mount in coming training.
~ means current user folder

Host
Insert USB disk to host
Copy helloworld to SD card
Ex.
$ cp helloworld /media/USER_NAME/MyDisk

Device
Insert USB disk to USB host slot
Mount USB disk
mount –t vfat /dev/sdb1 /mnt
Copy file to root folder
cp /mnt/helloworld ~
Umount USB disk
umount /mnt

Ubuntu will mount USB mass storage automatically.
The mount point will be located at
/media/USER_NAME/DISK_LABEL

Launch your application in board

- When you want to launch program, you need to identify the program location unless it is located at /usr/bin or /sbin

- Launch application in current folder

./helloworld

./ means current path

- Launch application by absoluted filename

/root/helloworld

The root folder is located at /root

Summary

- **In this training, we learned**
  - Make a customize board from official EVK.
  - Modify Linux Kernel device tree for customize board
  - Build buildroot image and apply Kernel patch
  - Customize buildroot packages
  - Make an application with cross compiler