

August 21 (Monday)

Organizational Meeting.

1. The "GreenSheet" is posted on the github

Note: Bring your Laptop Computer to the Class.

<https://github.com/hualili/CMPE244>

## Course and Contact Information

Instructor:	Harry Li, Ph.D. Professor, Computer Engineering Dept., San Jose State University
Office Location:	Engineering Building 267A
Telephone:	(408) 924-4060 (650) 400-1116
Email:	hua.li@sjtu.edu
Class Days/Time:	Mondays and Wednesdays, 4:30 pm – 5:45 pm, August 21, 2023
Office Hours:	Mondays and Wednesdays, 3:00 pm – 4:00 pm
Classroom:	Engineering Building Room 295
Prerequisites:	CMPE 180A and CMPE 180D, classified standing, computer science majors or Artificial Intelligence or Computer Engineering or Software Engineering majors only.

2. Emphasis on Posix O.S. Linux OpenSource O.S. & Device Drivers Programming and Development. Scalability & Vertical Solution.

## Course Description

Experiments dealing with advanced embedded software programming concepts, interfacing techniques, hardware organization, and software development using embedded systems. Individual projects.

3. Course Format: In-Person.

Hands-on Class. Prototype System

Optional. NVIDIA Jetson Nano. GPU (128)

4 GB Version      CPU      JetPack

Option 2. Broadcom Piex3B+, Piex4.

Option 3. RISC-V FPGA Dev. Board + FreeRTOS

Selection Decision in 1 week

Option 4. NXP LPC11G24 or

LPC1768, RTOS. NXP

Dev. Forum.

has limited Processing Power.

May Not Meet the Need for Our Project

#### 4. Textbook & References

Set I: Datasheets(s), CPU Datasheet, Developer Guide; Set II: NVIDIA Developer Forum. Set III: PPTs, Sample Code, Handouts in the Class GitHub.

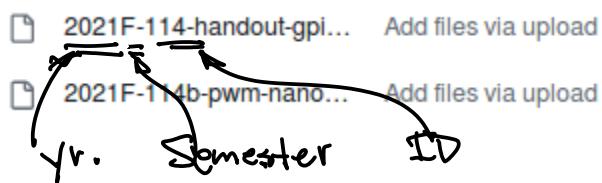
#### Course Materials

Instructor's teaching materials and online resources.

1. Professor's git: <https://github.com/hualili/CMPE244>
2. Jetson NANO Jetpack download <https://developer.nvidia.com/embedded/downloads>

#### Other Equipment / Material

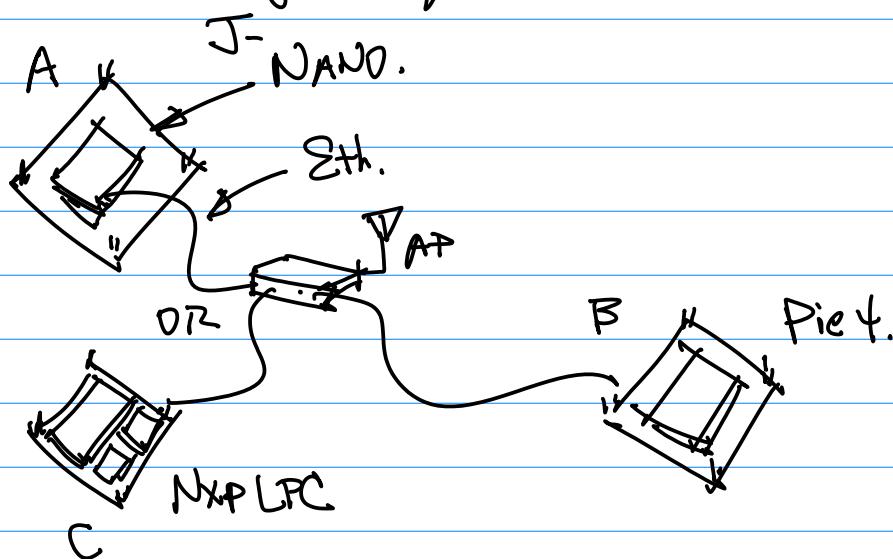
1. Hardware Equipment: You may choose any one of the following options. For detailed selection information, I will cover it in the introduction session of the class. Option 1. Nvidia Jetson NANO Board with minimum 2 GB RAM; or Option 2. Pi 3B+, or Pi 4; Option 3: Nvidia Jetson Tx2 developer kit; or Option 4: LPC1769 CPU Module: [https://www.mouser.com/NXP-Semiconductors/Embedded-Solutions/Engineering-Tools/Embedded-Processor-Development-Kits/Development-Boards-Kits-ARM/\\_/N-cxd2t2P=1z0jm4m&Keyword=LPC1769&FS=True&gclid=Cj0KCQjwqKuKBhCRAIIsACf4XuHyN8WfqtQ24WGgt0MdKd6n-k17c-YNz-r1hTcPt0ErdZN62jrM0mgaAtXZEALw\\_wcB](https://www.mouser.com/NXP-Semiconductors/Embedded-Solutions/Engineering-Tools/Embedded-Processor-Development-Kits/Development-Boards-Kits-ARM/_/N-cxd2t2P=1z0jm4m&Keyword=LPC1769&FS=True&gclid=Cj0KCQjwqKuKBhCRAIIsACf4XuHyN8WfqtQ24WGgt0MdKd6n-k17c-YNz-r1hTcPt0ErdZN62jrM0mgaAtXZEALw_wcB) or Option 5: Samsung ARM11 developer platform.
2. Linux Host Machine (Ubuntu 18.04).



Naming Convention:

A & B  
A & C

Note: Regarding the Selection of  
A Target Platform:



## 5. Grading Policy

Project Assignment (Two Projects)	
Phased	
Assignments and projects:	15% (pts) for the assigned projects.
Midterm Exam:	30%
Final Exam:	30%
Total:	40%
	100%

August 23rd (Wed)

Introduction

Note: Rm268

Ref:

Datasheets.



bcm



lpc



nvda



sam

Broadcom

Pi e

Linux O.S.

NXP

LPC1769

RTOS

IP Stack

Micro Web Server

Jetson

NAND.

JetPack O.S.

Linux (Ubuntu)

+ Additional  
Packages.

Samsung

ARM11

2021F-107-lpc-cpu-  
UM10360.pdf

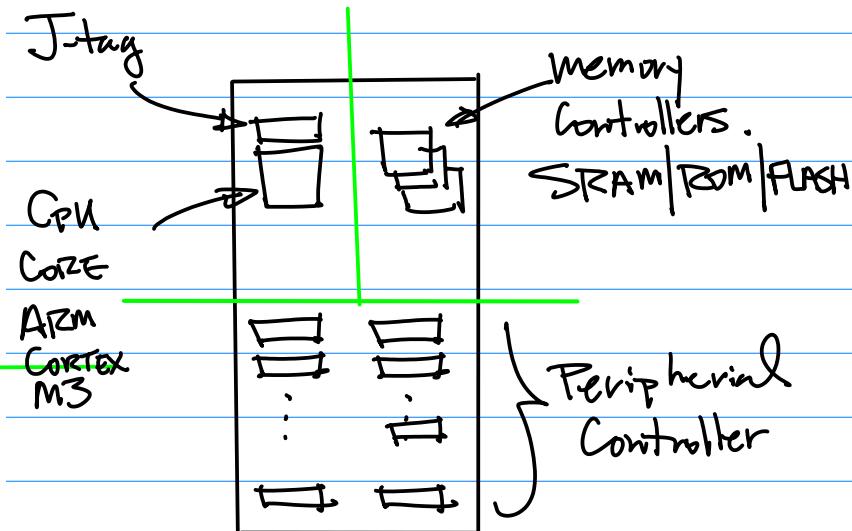
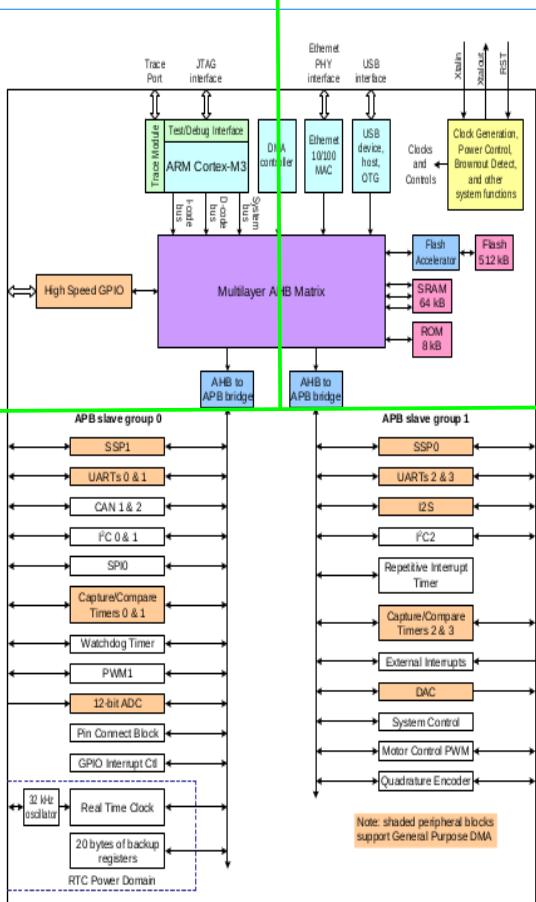
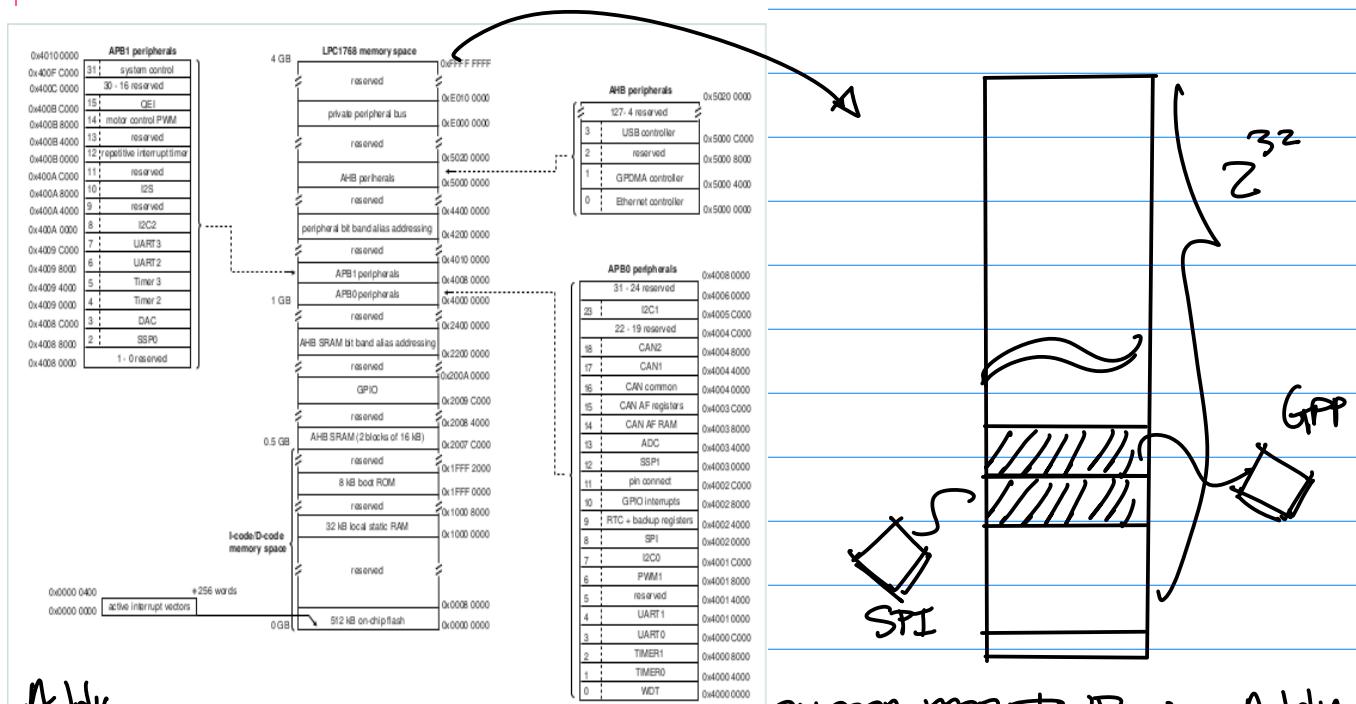


Fig. 1

Note: The CPU Block Diagram for LPC1769 is a Sample for the Rest of the target platforms.

e.g. Pic34 ; Sam's ARM 11 ;  
NVIDIA Jetson NANO

Note 2 :



Addrs.

$$\begin{aligned} Z^{32} &= Z^0 \cdot Z^1 \cdot Z^2 \cdot Z^3 \\ 1024 &: : : 4 \\ 1K &: : : \\ 1\text{ meg.} &: : : \\ 1\text{ G.} &: : : \end{aligned}$$

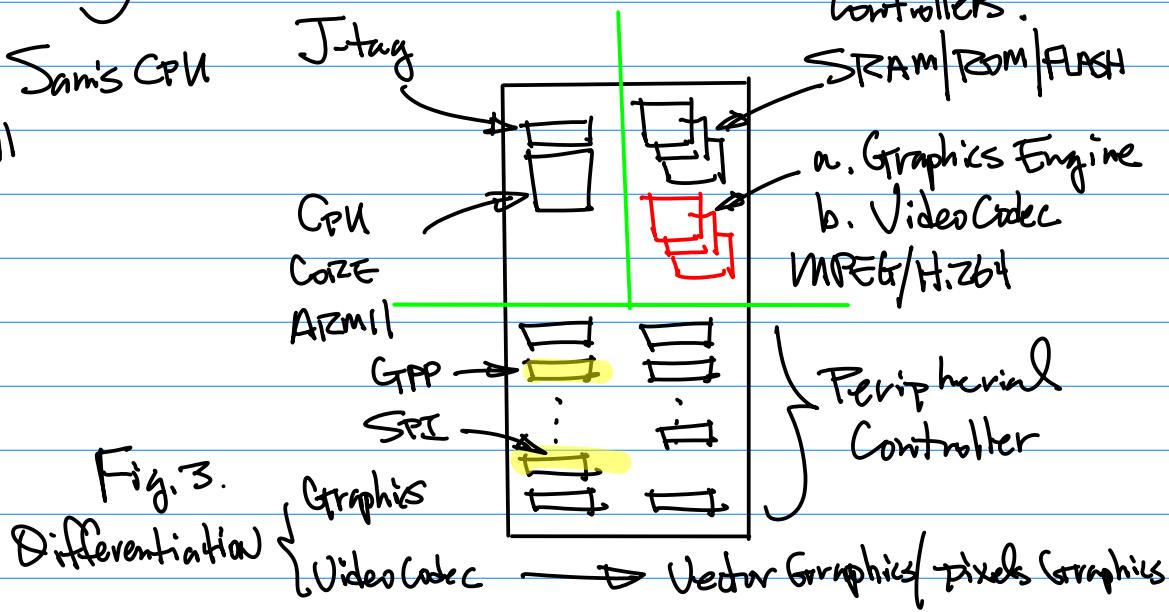
4 G Addresses.

Example: "B", Sam's CPU  
ARM11

Fig. 2

0x0000-0000 - PWR-up Addr.

Datasheets.

2021F-105-#0-cpu-arm11-  
2018S-29-CPU\_S3C6410X.  
pdfLocate the page with the top level  
Description of the CPU Architecture

Example: Connection to (Embedded) Software Architecture

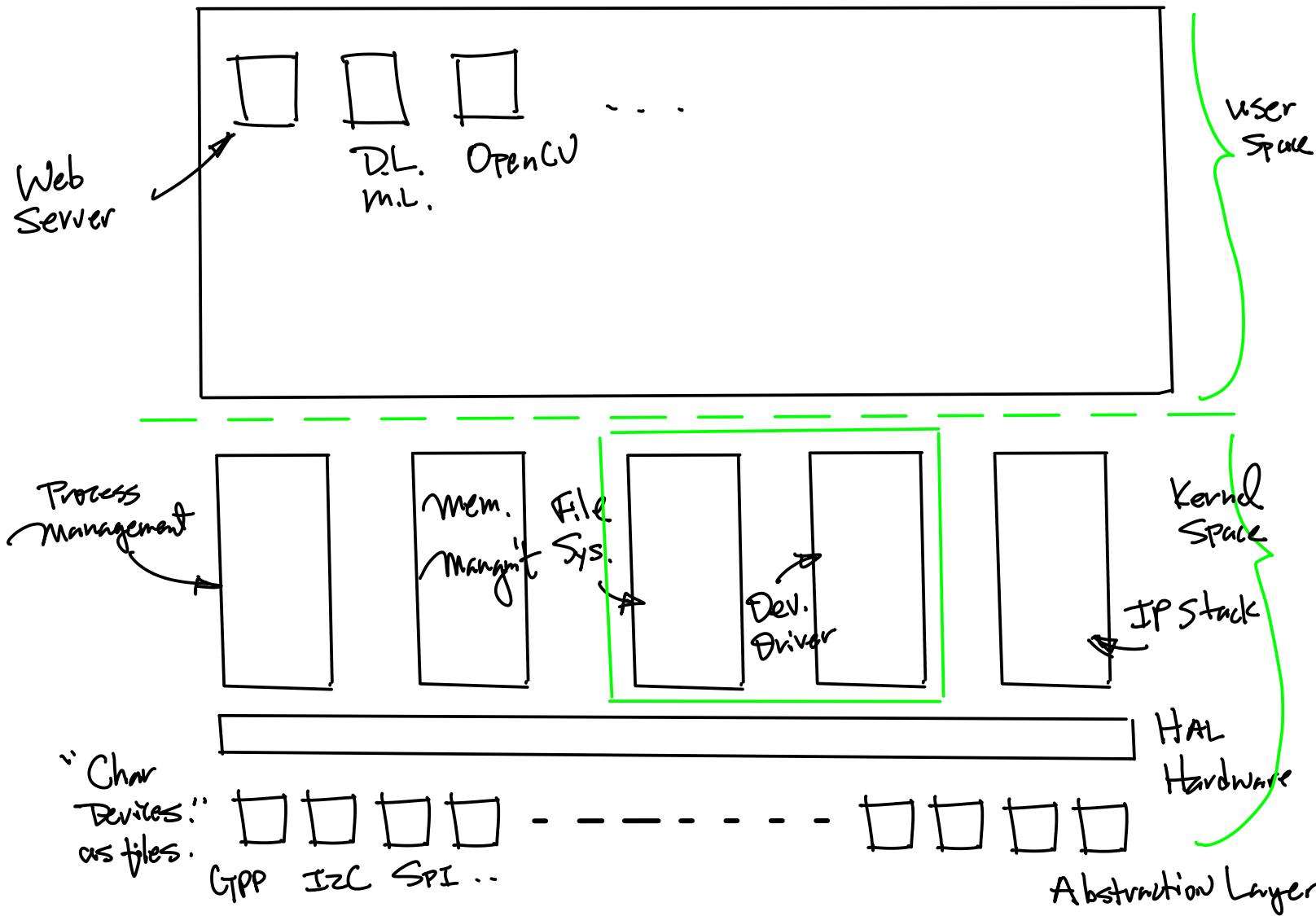


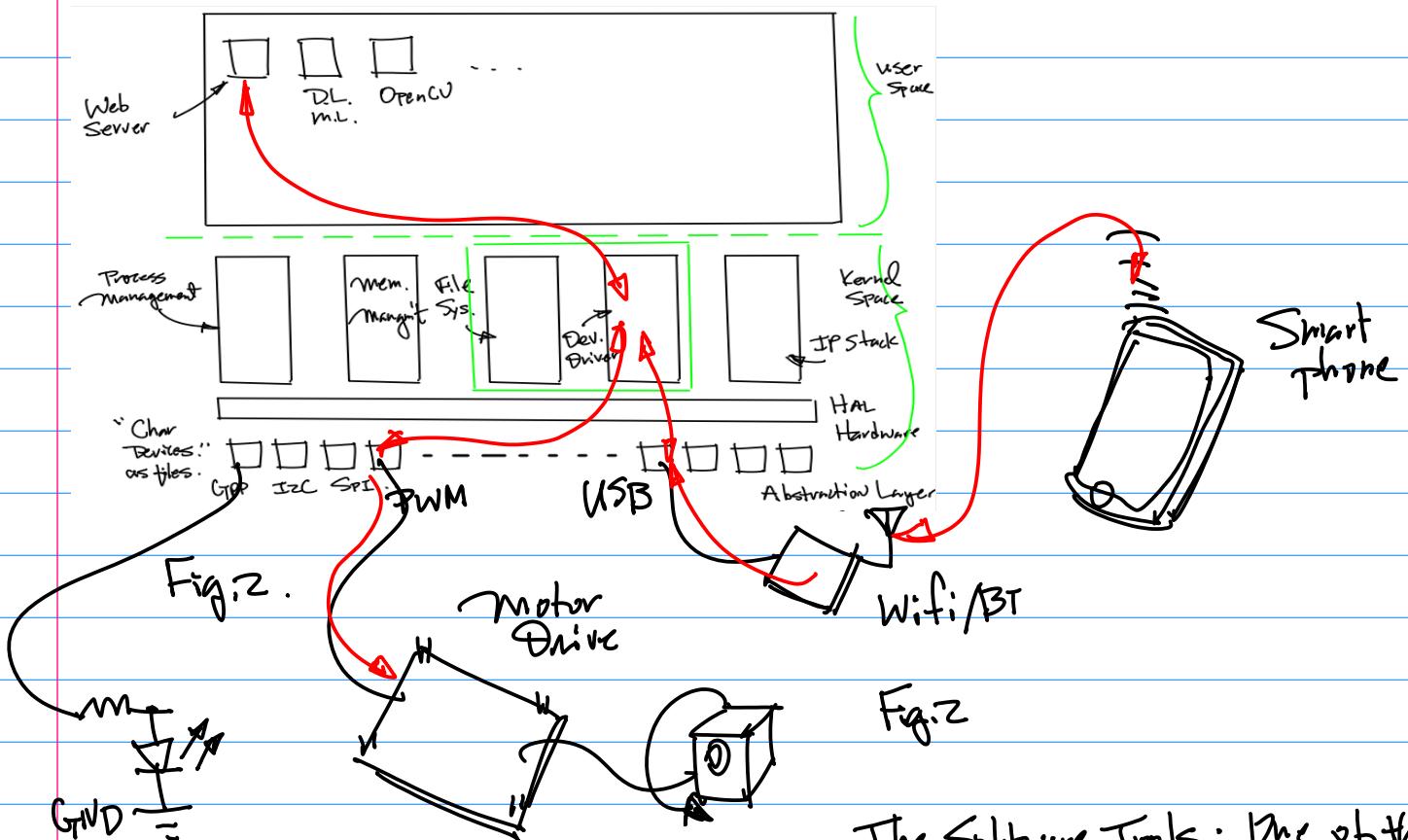
Fig.1.

Note: Data Size for 1080P  
Image OR 720P

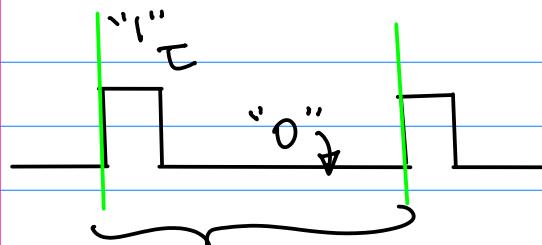
August 28 (Monday)  
Note: 1<sup>o</sup> Brief Description D.V.  
the Scope of Semester-Long  
Project.

- Embedded Software; Kernel D.S.
- Device Driver → APPS for iPhone/Android phone
- 2<sup>o</sup> CANVAS is up.
- Honesty pledge
- 3<sup>o</sup> Target platform → Minor upgrade to Enable RTC by Adding On-Board Battery

Example: Continuation of the Introduction/Embedded Software Architecture.



Note: PWM — Pulsewidth Modulation.



$T$  One Period

$$\left\{ \begin{array}{l} \text{Duty Cycle} = \frac{T}{f_{\text{PWM}}} \dots (1) \\ f_{\text{PWM}} \dots (2) \end{array} \right.$$

The Software Tools: One of them is open source gcc, or g++ Compiler.



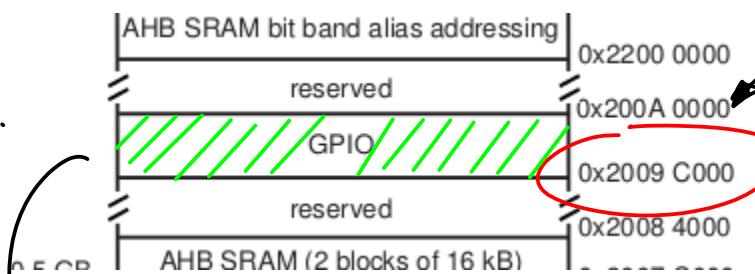
Porting. → Match to the Core  
(ISA: Instruction Set Architecture)  
Device Drivers Customization.

✓ Peripheral Controller  
A Set of Special Purpose Registers.

Most Likely this SPR has  
the addv in the Block.

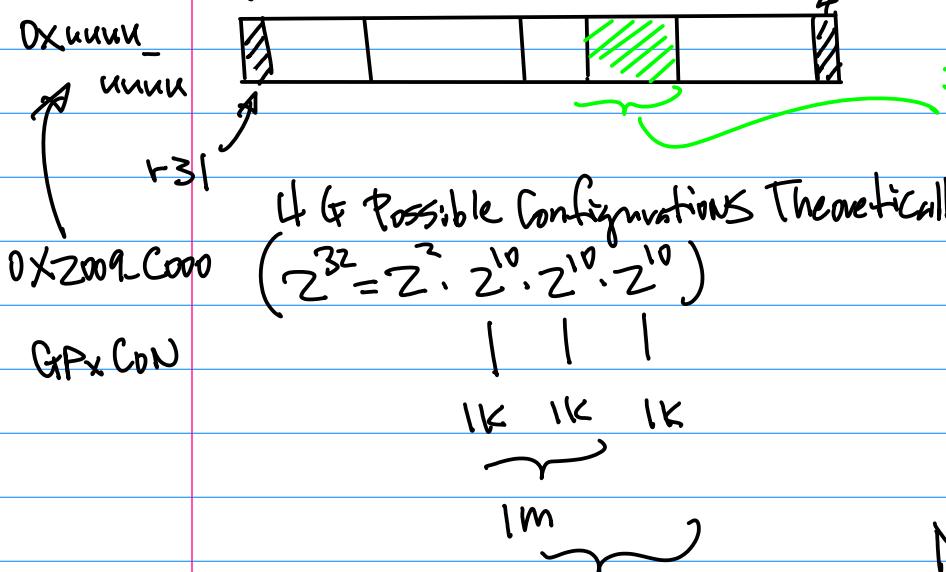
Identify A peripheral controller, GPP

Fig. 3.



→ Memory is dedicated for  
SPR's (Special Purpose Registers)

## Control & Configuration Register



It has its unique address. (at the multiple of 4).

## Ext. Naming Convention

Guideline :

Timeline:  
RISC → UC Berkeley David Patterson  
Stanford, John Hennessy

August 30 (Wed)

Note: 1<sup>o</sup> CANVAS is up.

Honesty Pledge to Be Signed  
And Submit on CANVAS  
By this Friday 11:59 pm.

2<sup>o</sup> Please Bring the target platform to the Class. Next Wednesday.

3<sup>o</sup> (Written Requirements) <sup>in 2 weeks</sup>

Bring up your target platform  
By Downloading Kernel OS.

Image to A micro-SD Card,  
then boot the System.

then Screen Capture with your personal identifier, Submit it on CANVAS.

4<sup>o</sup> Create A ChatGPT Account, Python interface to ChatGPT (3.5) API.

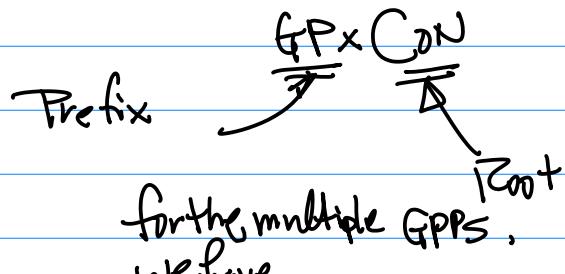
Example: Continued.

Naming Convention of the Control & Configuration Register.

By John Hennessy . Golden Rules

Uniformity, "3+3"  
Regularity,  
Orthogonality Naming Convention

Prefix + Root  
3 letters 3 letters.

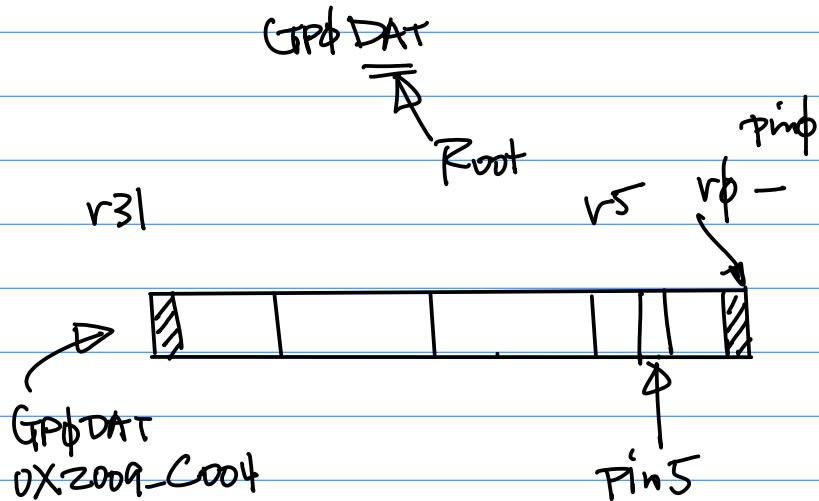


for the multiple GPPS,  
we have

GPPCON, GPICON, etc.

Suppose we want to use GP0 pin 5 as an output to turn on/off LED.

Design 2nd SPR.



place "1" @ r5 to Output logical high.  
"0" ..  
Low.

#define GP0CON  
GP0DAT ..

Porting Porting  
gcc/g++ → ARM → CORTEX

Porting ↓  
Board

Sept 6 (Wed).

Note: 1° Target Board  
Inspection:

Purpose: J41 Connector

RTC Battery      2021F-114~  
Ref: on the github.

Harry Li, Ph.D.

2021F-114-gpio-nano-v2-h1-2021-10-20.pdf

160.87% □

## NVIDIA Jetson Nano J41 Header Pinout

<https://www.jetsonhacks.com/nvidia-jetson-nano-j41-header-pinout/>

Note: I2C and UART pins are connected to hardware and should not be reassigned. By default, all other pins (except power) are assigned as GPIO. Pins labeled with other functions are recommended functions if using a different device tree.

GPIO	Name	Pin	Pin	Name	Sysfs GPIO
	3.3 VDC Power	1	2	5.0 VDC Power	
	I2C_2_SDA I2C Bus 1	3	4	5.0 VDC Power	
	I2C_2_SCL I2C Bus 1	5	6	GND	
gpio216	AUDIO_MCLK	7	8	UART_2_TX /dev/ttyTHS1	
	GND	9	10	UART_2_RX /dev/ttyTHS2	
gpio50	UART_2_RTS	11	12	I2S_4_SCLK	gpio79
gpio14	SPI_2_SCK	13	14	GND	
gpio194	LCD_TE	15	16	SPI_2_CS1	gpio232
	3.3 VDC Power	17	18	SPI_2_CS0	gpio15
gpio16	SPI_1_MOSI	19	20	GND	
gpio17	SPI_1_MISO	21	22	SPI_2_MISO	gpio13
gpio18	SPI_1_SCK	23	24	SPI_1_CS0	gpio19
	GND	25	26	SPI_1_CS1	gpio20

Diagram showing the pinout for the NVIDIA Jetson Nano J41 Header. The diagram is a grid where rows represent pins and columns represent functional groups. The first column is labeled with GPIO numbers (e.g., gpio149, gpio200, gpio38, gpio76, gpio12). The second column is labeled with function names like GND, I2C\_1\_SDA/I2C\_1\_SCL, CAM\_AF\_EN, GPIO\_PZ0, GPIO\_PE6, I2S\_4\_LRCK, SPI\_2\_MOSI, and another GND. The third column is labeled with pin numbers (e.g., 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40). The fourth column is labeled with pin numbers (e.g., 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26). The fifth column is labeled with function names like SPI\_1\_CS1, I2C\_1\_CS1, LCD\_BL\_PWM, I2S\_4\_SDIN, I2S\_4\_SDOUT, and another GND. The sixth column is labeled with Sysfs GPIO numbers (e.g., gpio20, gpio168, gpio51, gpio77, gpio78, gpio20).

Note: 1° Power Pins

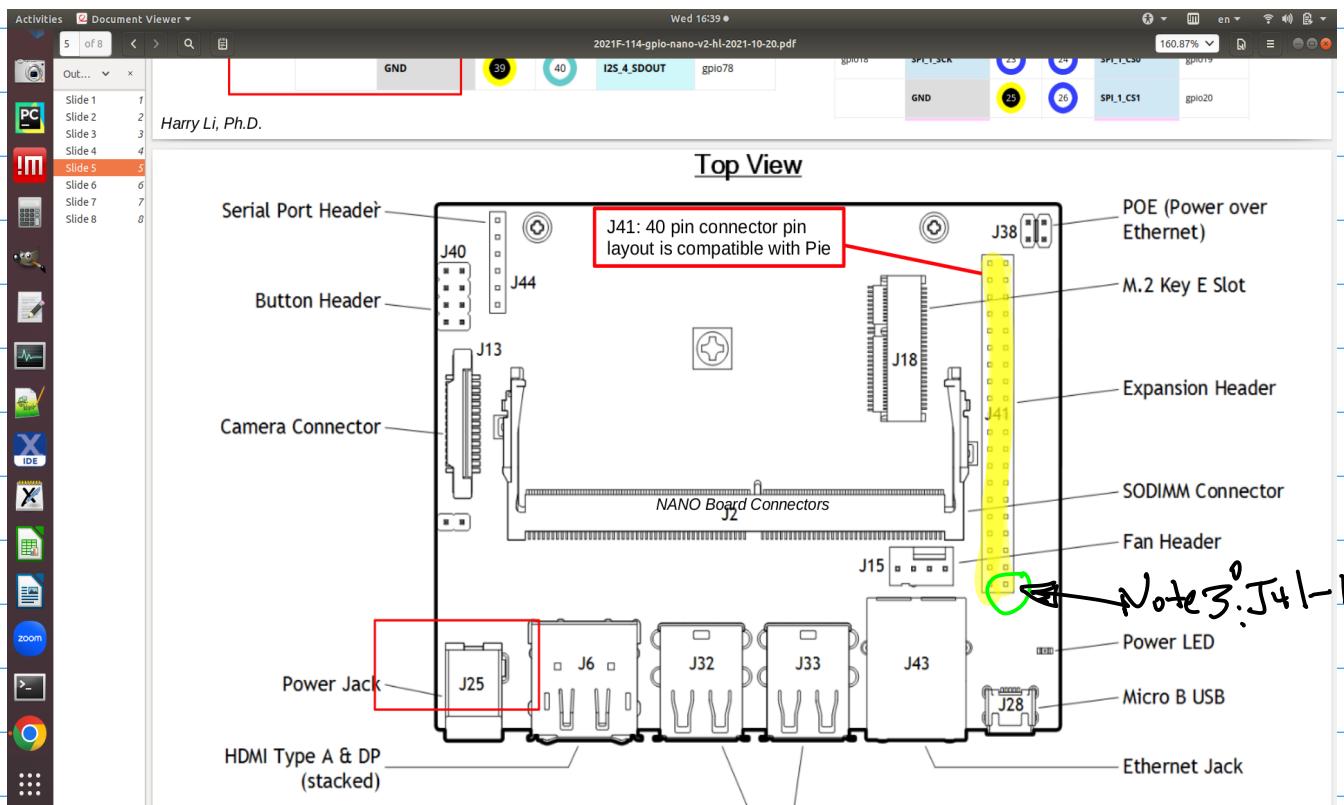
{ GND : b19/25/3a  
Vout: 3.3VDC/5VDC  
Pin 1, Pin 2, 4.  
Vin: J25 (5A or higher  
@ 5VDC)

Note 2° GPIO

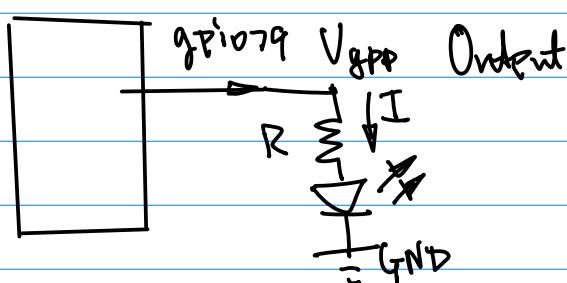
J41-Pins  
J41-12  
J41-40

CPU Functionality

gpio79  
gpio78



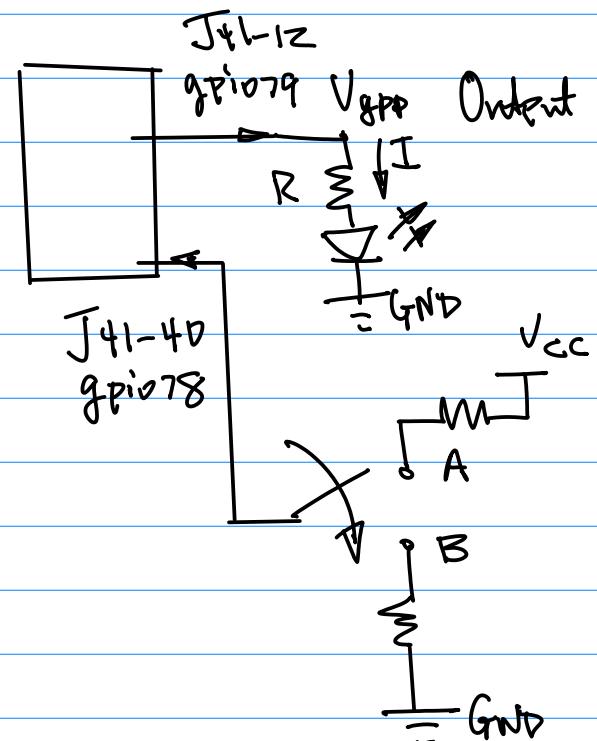
Note 4<sup>o</sup>: GPIO Input/Output Testing Ckt  
Build the following Testing Ckt.



Let  $I = 4 \text{ mA}$ ,  $V_{\text{LED}} \approx 1.8 \text{ V}$

$$V_{\text{DD},\text{H}} = IR + V_{\text{LED}} \dots (1)$$

w/o Resistor With Proper Selected LED.

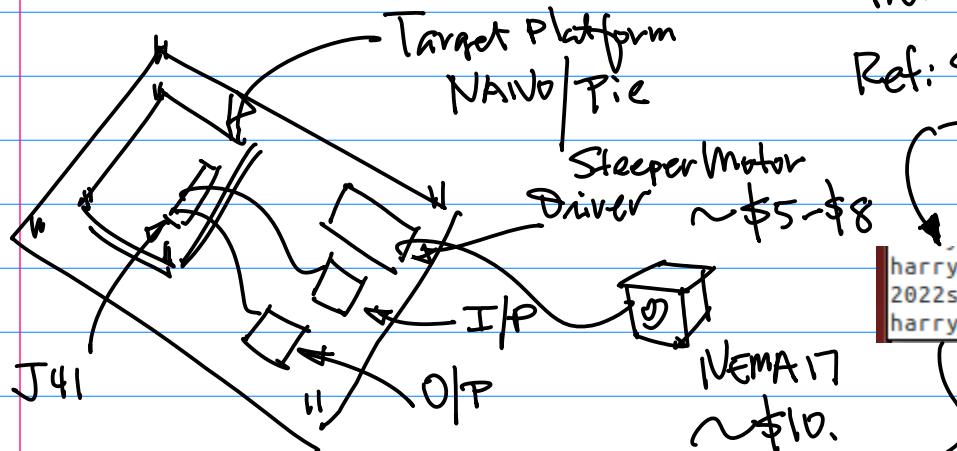


GPIO J41 Pins	CPU Functionality	Note
J41-12	gpio79	Output
J41-40	gpio78	Input

User Space Code  
Kernel Space Code

Take a Reference Design  
from Arm11, Samsung CPU.

Ref: Sample code has been  
Posted on the github.



```
harry@harry-laptop:/opt/FriendlyARM/min
2022s-104d-userSpace-gpio.c led led.c
harry@harry-laptop:/opt/FriendlyARM/min
```

#### CMPE242-Embedded-Systems- / 2022S / 2022S-104d-userSpace-gpio.c

hualili Add files via upload

Code	Blame	37 lines (30 loc) · 642 Bytes	Code 55% faster with Git
------	-------	-------------------------------	--------------------------

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/ioctl.h>
5 #include <sys/types.h>
6 #include <sys/stat.h>
7 #include <fcntl.h>
```

Note: Form 2 person Team for  
A Semester Long Project.

Sharp: Hardware Layer  
(Sensors/Actuators)  
↓  
"Security"  
Device Driver/Kernel Space  
↓  
Process Management

↓  
Web Server

↓  
Smartphone APPs.

ChatGPT 3.5 API + Python Interface

Example: Sample Code for GPIO Device  
Driver

Sept. 11 (Monday)

Notel: Homework Due in 1½ weeks.

To Be Posted on Line today;

a. Bring up the target, Screen  
Capture with Personal Identifier.

b. GPIO Testing (Python + Hardware  
Circuit). (1) Input Testing (KT).

Output Testing (KT). (2) Coding:

Python.

Note 2: B.o.M. (Bill of Material)

for the class/Project

a. Motor (Stepper Motor, NEMA.

17



Nema17  
Stepper Motor  
\$8.99  
Amazon.com

(2) BLDC  
Brushless DC  
motor

c. Smartphone (iPhone

OR Android phone

Swift Mac OS. as Development  
platform.

Ubuntu Linux.  
18.04.

NVIDIA JetPack (OS.)

Note:

With GPIO Testing

Ckt.

(Work-In-  
progress)

Target  
Board

NVIDIA Jetson Nano Bread Board/

OR Wine Wrapping  
Board.



350W Brushed  
Electric Motor



10 Inch Hub  
Motor 1000w, ...



48V 500W  
Wheel Motor ...

b. Motor Drive Unit.



EASON  
Stepper Motor  
\$9.69  
Amazon.com



SparkFun  
Electronics ...  
\$12.09  
DigiKey



WWZMDiB  
A4988 Stepper  
\$7.99  
Amazon.com



Pololu  
Corporation  
\$8.49  
DigiKey



[STEPPERONLINE](#)  
[CNC Stepper Motor](#)  
[Driver 1.0-4.2A](#)  
20-50VDC 1/128

## Example: Continuation on Linux D.D.

on ARM-11 (Samsung). Note! userSpace Code Samples, Kernel Space code

```
linux
harry@harry-laptop:/opt/FriendlyARM/mini6410$ cd linux/
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux$ ls
arm-qte-4.7.0      examples          u-boot-mini6410
arm-qt-extended-4.4.3  linux-2.6.38    x86-qte-4.6.1
arm-qtopia          rootfs_qtopia_qt4  x86-qt-extended-4.4.3
busybox-1.17.2     rootfs_qtopia_qt4-s x86-qtopia
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux$
```

Ref. Sample code on github.

```
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux/examples/leds$ ls
2022s-104d-userSpace-gpio.c  led  led.c  led.c~  Makefile  Makefile~
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux/examples/leds$
```

```
22
23      fd = open("/dev/leds0", 0);
24  if (fd < 0) {
25      fd = open("/dev/leds", 0);
26  }
27  if (fd < 0) {
28      perror("open device leds");
29      exit(1);
30  }
31
32      ioctl(fd, on, led_no);
33  close(fd);
34 }
```

Note1: "Char" Device. Open the Device just like a file.

Kernel (OS. Image)  
path to the Device.  
The Device Driver can be either integrated as the whole kernel image or module (installed/removed)

Note2 for passing control parameter(s) to the Device for Control Action.

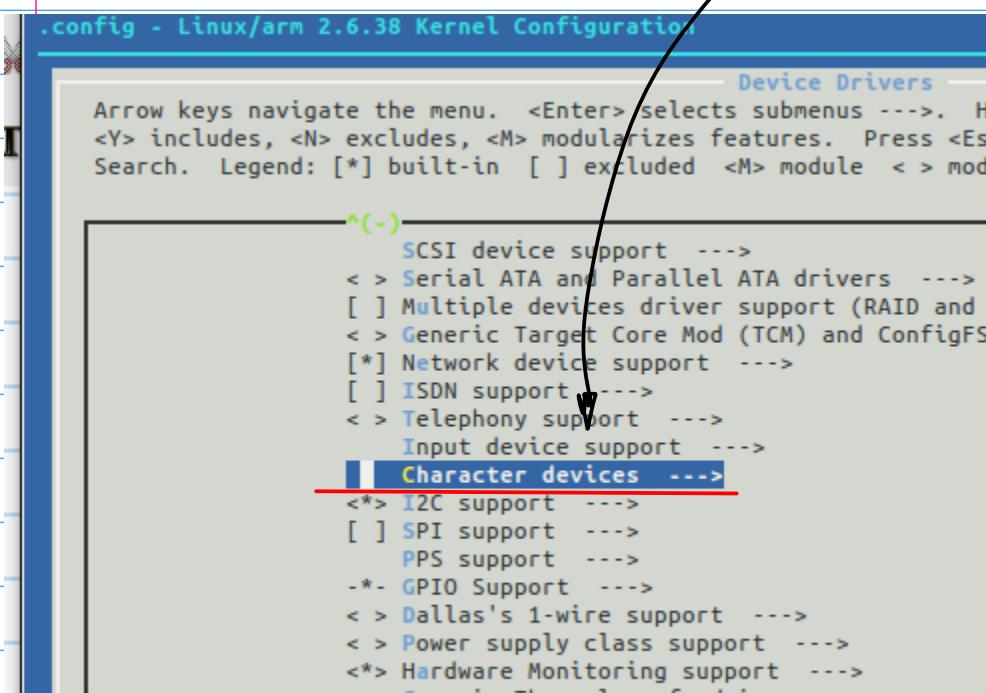
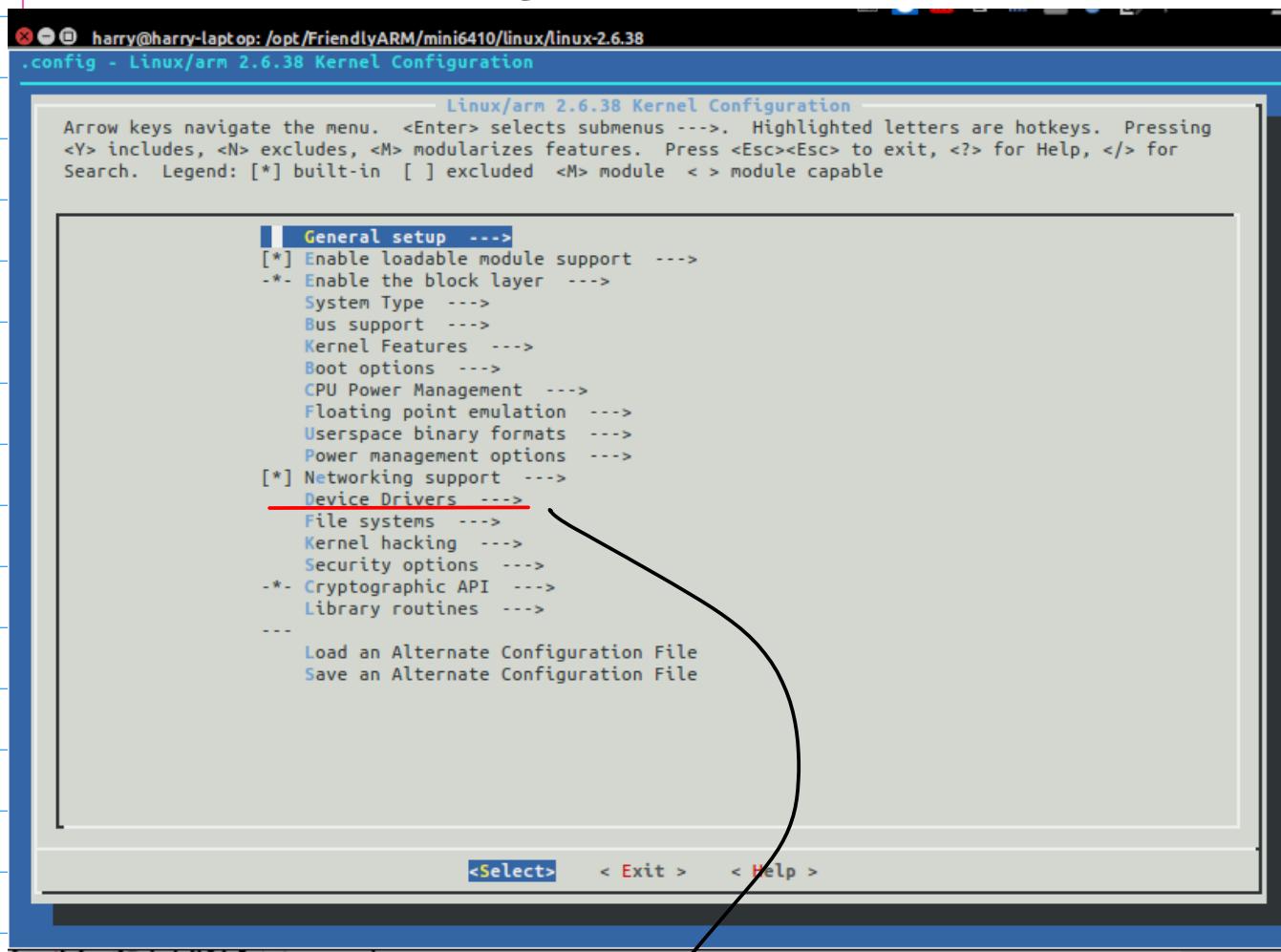
Note3. Close it, when done!

Build

Kernel image using

"menuconfig" → NVDA, Broadcom, Smart phones  
Embedded

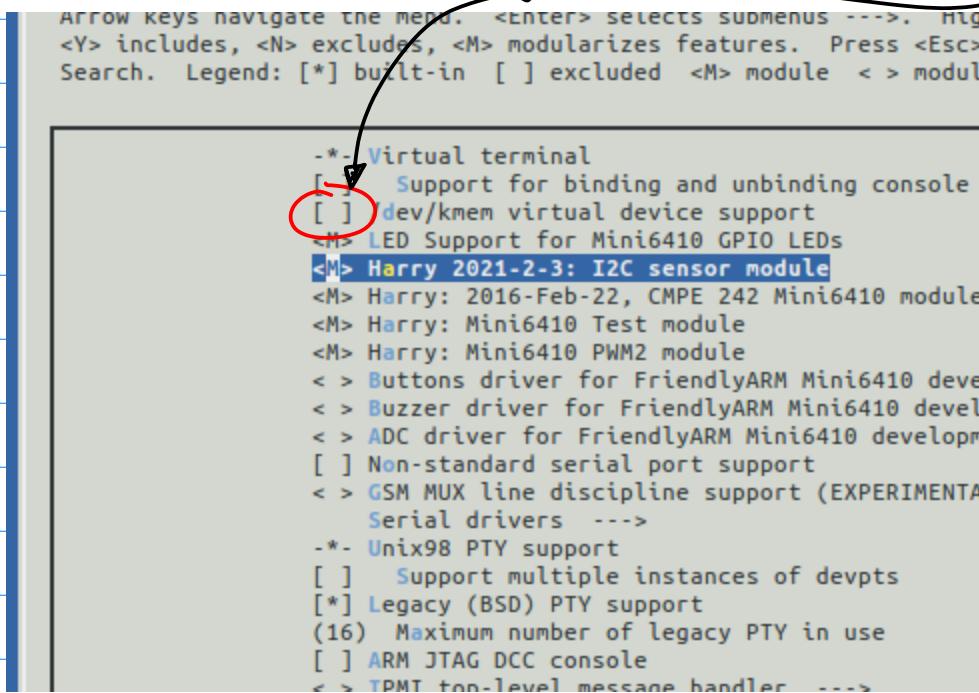
Android.



Note 4. menutools controls how the Kernel Image is built.  
 Here, the "Char" Device Driver can be Selected/Deselected.  
 b. Use "Space Bar" to toggle between 3 options.

```
Arrow keys navigate the menu. <Enter> selects submenus ---. Hig
<Y> includes, <N> excludes, <M> modularizes features. Press <Esc>
Search. Legend: [*] built-in [ ] excluded <M> module <> modul

```



(Non | M | \*)  
↓  
Module  
Integrated Kernel Image

Note 5. Folder for the "Char" D.D., The "gpio" (leds) Device Driver

```
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers$ cd char
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char$ ls
20-2021S-9-mini6410_pwmHarry.c  ip2          misc.c      rtc.i
2q                         ipmi         misc.o      scc.i
agg                        isicom.c    mmtimer.c  scx2i
amiserial.c                istallion.c modules.builtin ser_i
apm-emulation.c            Kconfig     modules.order moxa.c
applicom.c                 lp.c        moxa.h      moxa.h
applicom.h                 Makefile   mspec.c      mspec.c
bfin_jtag_comm.c           Makefile-backup  mwave       mxser.c
bfin-otp.c                 mbcs.c     mxser.h      mxser.h
briq_panel.c               mbcs.h      nozomi.c    nsc_gpio.c
bsr.c                      mem.c       nvram.c      nwbutton.c
built-in.o                  mem.o       nwbutton.h   nwbutton.h
cd1865.h                   mini6410_adc.c  nvram.c      nwflash.c
cyclades.c                 mini6410_adc.mod.c mini6410_adc.o pc8736x_gpio.c
digi1.h                     mini6410_buttons.c mini6410_buttons.o pxmcia
digiFep1.h                 mini6410_buttons.o mini6410_hello_module.c ppdev.c
digiPCI.h                  mini6410_hello_module.c mini6410_hello_module.mod.c ps3flash.c
ds1302.c                   mini6410_hello_module.ko mini6410_hello_module.mod.o ramosops.c
ds1620.c                   mini6410_hello_module.ko mini6410_hello_module.o random.c
dsp56k.c                   mini6410_hello_module.ko mini6410_leds.c random.o
dtlk.c                      mini6410_hello_module.ko mini6410_leds.ko tb02
efirtc.c                   mini6410_hello_module.ko
epca.c                      mini6410_hello_module.ko
epcaconfig.h               mini6410_hello_module.ko
```

2022S-104e  
(CMPE242)

## Example: Kernel Space

Device Driver Code Requirements: Code Spec.

Note1:  
 Connect the Code to the CPU  
 Datasheet  
 ↓  
 Debug

```

66 static int __init dev_init(void)
67 {
68     int ret;
69
70     {
71         unsigned tmp;
72         tmp = readl(S3C64XX_GPECON);
73         tmp = (tmp & ~(0xffffU<<16))|(0x1111U<<16);
74         writel(tmp, S3C64XX_GPECON);
75
76         tmp = readl(S3C64XX_GPEDAT);
77         tmp |= (0xF << 4);
78         writel(tmp, S3C64XX_GPEDAT);
79     }
80
81     ret = misc_register(&misc);
82
83     printk (DEVICE_NAME"\nHarry: PGE initialized\n");
84 }
```

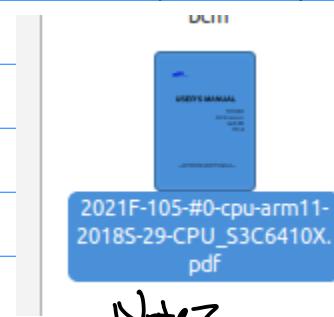
Note2. Read long 32bit

Naming: ID — Peripheral Control  
 (Manufacturer Part) GPP



Memory map.

Ref: CPU Datasheet.



Note2.

TP320.

a) GPE Con

b) Addr.

/ On the  
memory.

c) GPEDAT

d. Two Addition types:

Pull up  
down.

GPEPUD

GPECONSPL  
GPEPUDSLPPower  
Control

Register	Address	R/W	Description	Reset Value
GPECON	0x7F008080	R/W	Port E Configuration Register	0x00
GPEDAT	0x7F008084	R/W	Port E Data Register	Undefined
GPEPUD	0x7F008088	R/W	Port E Pull-up/down Register	0x00000155
GPECONSPL	0x7F00808C	R/W	Port E Sleep mode Configuration Register	0x0
GPEPUDSLP	0x7F008090	R/W	Port E Sleep mode Pull-up/down Register	0x0

GPDCON	Bit	Description		Initial State
GPE0	[3:0]	0000 = Input 0010 = PCM SCLK[1] 0100 = AC97 BITCLK 0110 = Reserved	0001 = Output 0011 = I2S CLK[1] 0101 = Reserved 0111 = Reserved	0000
GPE1	[7:4]	0000 = Input 0010 = PCM EXTCLK[1] 0100 = AC97 RESETn 0110 = Reserved	0001 = Output 0011 = I2S CDCLK[1] 0101 = Reserved 0111 = Reserved	0000

Note 3. From the D.D. Code, GPE1 is utilized for the I/O function

$$GPECON[7:4] = 0001$$

5  
GPIO  
pins

GPDCON	Bit	Description		Initial State
GPE0	[3:0]	0000 = Input 0010 = PCM SCLK[1] 0100 = AC97 BITCLK 0110 = Reserved	0001 = Output 0011 = I2S CLK[1] 0101 = Reserved 0111 = Reserved	0000
GPE1	[7:4]	0000 = Input 0010 = PCM EXTCLK[1] 0100 = AC97 RESETn 0110 = Reserved	0001 = Output 0011 = I2S CDCLK[1] 0101 = Reserved 0111 = Reserved	0000
GPE2	[11:8]	0000 = Input 0010 = PCM FSYNC[1] 0100 = AC97 SYNC 0110 = Reserved	0001 = Output 0011 = I2S LRCLK[1] 0101 = Reserved 0111 = Reserved	0000
GPE3	[15:12]	0000 = Input 0010 = PCM SIN[1] 0100 = AC97 SDI 0110 = Reserved	0001 = Output 0011 = I2S DI[1] 0101 = Reserved 0111 = Reserved	0000
GPE4	[19:16]	0000 = Input 0010 = PCM SOUT[1] 0100 = AC97 SDO 0110 = Reserved	0001 = Output 0011 = I2S DO[1] 0101 = Reserved 0111 = Reserved	0000

How To

Question: Make GPE1 as an output pin,  
But keep the rest unchanged?

Sept. 18 (Monday)

Note: 1<sup>o</sup> Homework Posted on the  
CANVAS.

2<sup>o</sup> Homework, Due Sept. 27  
(Wed), PWM Testing.

① Enable PWM Device  
Driver via Driver mapping  
utility By NVIDIA

② Test PWM Output By  
Changing f<sub>PWM</sub> from 2 kHz  
to 50 Hz; By changing  
Duty Cycle from 5% to 90%.

Then, Observe its Output

Note: Output with 2222 npn Transistor  
to drive a LED. (see the  
details in the Class PPT on  
github). (U.S.)

Note: Prepare the Source Distribution  
Download, to build Kernel  
Image from the distribution.  
(1 ~ 1½ week).

Note: Ref. on Smartphone Apps.  
Android APPs Development.

## Note: Device Driver Sample Code

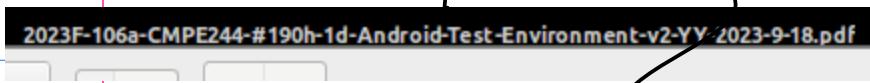
<input type="checkbox"/> 2023F-104-gpio-command-line-2023...	Add files via upload
<input checked="" type="checkbox"/> 2023F-105a-2022s-104d-userSpace...	Add files via upload
<input checked="" type="checkbox"/> 2023F-105b-mini6410_leds.mod.c	Add files via upload
<input type="checkbox"/> 2023F-106a-CMPE244-#190h-1d-A...	Add files via upload

README.md

Update README.md

APPS .

Note: Install Android Studio  
ON your Laptop .



## Install Android Studio on Ubuntu 18.04 (9/15-8/3/2023)

nnn-n-Android-Development-HelloWorld-Calculator-v2-XW-2023-7-7.odt

### 1. Check Java version

```
nicole@nicole-GS65:~$ java --version
openjdk 11.0.19 2023-04-18
OpenJDK Runtime Environment (build 11.0.19+7-post-Ubuntu-0ubuntu118.04.1)
OpenJDK 64-Bit Server VM (build 11.0.19+7-post-Ubuntu-0ubuntu118.04.1, mixed mode, sharing)
nicole@nicole-GS65:~$
```

3. Once installed, double click the icon to start the studio



### 2. Install Android Studio using snap

```
$ snap find "android-studio"
$ sudo snap install android-studio --classic
```

```
nicole@nicole-GS65:~$ snap find "android-studio"
Name          Version   Publisher      Notes           Summary
android-studio 2022.2.1.19  snapcrafters  classic  The IDE for Android
android-studio-canary 2022.1.1.7  snapcrafters  classic  The IDE for Android
(Canary build)
nicole@nicole-GS65:~$ sudo snap install android-studio --classic

(base) harry@harrys-gpu-laptop:~$ sudo snap install android-studio --classic
[sudo] password for harry:
android-studio 2022.3.1.18 from Snapcrafters installed
```



Note: Team Project (1) Formation of  
The Team ; (2) Selection of Smart phone  
for the APP. (3) Stepper motor  
Drive Board and Stepper motor.

Example: GPIO & PWM .

2023F-104

GPIO Command Line  
<https://jetsonhacks.com/2019/06/07/jetson-nano-gpio/>

```
# Map GPIO Pin
# gpio79 is pin 12 on the Jetson Nano
$ echo 79 > /sys/class/gpio/export
# Set Direction
$ echo out > /sys/class/gpio/gpio79/direction
# Bit Bangin'!
$ echo 1 > /sys/class/gpio/gpio79/value
$ echo 0 > /sys/class/gpio/gpio79/value
# Unmap GPIO Pin
$ echo 79 > /sys/class/gpio/unexport
# Query Status
$ cat /sys/kernel/debug/gpio
```

PWM

2021F-114b - ~

Note: Establish Remote Access to your target, e.g. Laptop to Access your target Board.

→ Next Monday, Show+Tell in class

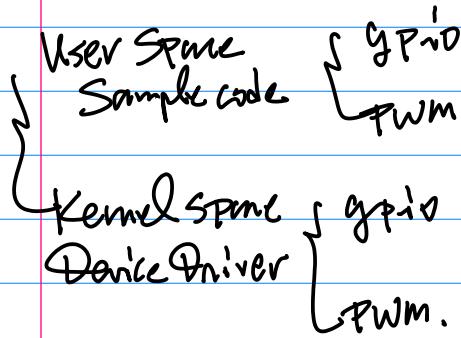
Sept 25 (Monday).

Note: 1<sup>o</sup> Updated github Sample code.

(1) 2023F-107.

Booting Jetson Nano without HDMI Connection.

(2) 2023F-105a to 2023F-105f.



Target Board Connection

Requirements.

a. The Demo/Show+Tell in Class is required, Also

Midterm Exam, and Final Exam are Based on Analyzing & Executing your Implementation Code on the target platform;

b. Bring gpio Homework with the target Board to the Classroom

Wednesday for Quick Inspection,

Show+Tell; (Sept. 27, Wednesday).

Note: HDMI Monitor is optional, To Be Able to Boot your System is the requirement.

Consider the github Sample code.

Ref: -105a, User Space program.

for GPIO.   
 {  
 open()  
 ioctl()  
 close()  
 }

-105b & c

```

1 #include <mach/gpio-bank-k.h>
2
3 #define DEVICE_NAME "leds0"
4
5 static long sbc2440_leds_ioctl(struct file *filp, unsigned
6                                long arg)
7 {
8     switch(cmd) {
9         case 0:
10            unsigned tmp;
11
12            make
13            memory
14            for the Kernel
15            Driver
16            Configuration.
17
18            make
19            memory
20            for the Kernel
21            Driver
22            Configuration.
23
24            make
25            memory
26            for the Kernel
27            Driver
28            Configuration.
29
30            make
31            memory
32            for the Kernel
33            Driver
34            Configuration.
35
36            make
37            memory
38            for the Kernel
39            Driver
40            Configuration.
41
42            make
43            memory
44            for the Kernel
45            Driver
46            Configuration.
47
48            make
49            memory
50            for the Kernel
51            Driver
52            Configuration.
53
54            make
55            memory
56            for the Kernel
57            Driver
58            Configuration.
59
60            make
61            memory
62            for the Kernel
63            Driver
64            Configuration.
65
66            make
67            memory
68            for the Kernel
69            Driver
70            Configuration.
71
72            make
73            memory
74            for the Kernel
75            Driver
76            Configuration.
77
78            make
79            memory
80            for the Kernel
81            Driver
82            Configuration.
83
84            make
85            memory
86            for the Kernel
87            Driver
88            Configuration.
89
90            make
91            memory
92            for the Kernel
93            Driver
94            Configuration.
95
96            make
97            memory
98            for the Kernel
99            Driver
100           Configuration.
101
102           make
103           memory
104           for the Kernel
105           Driver
106           Configuration.
107
108           make
109           memory
110           for the Kernel
111           Driver
112           Configuration.
113
114           make
115           memory
116           for the Kernel
117           Driver
118           Configuration.
119
120           make
121           memory
122           for the Kernel
123           Driver
124           Configuration.
125
126           make
127           memory
128           for the Kernel
129           Driver
130           Configuration.
131
132           make
133           memory
134           for the Kernel
135           Driver
136           Configuration.
137
138           make
139           memory
140           for the Kernel
141           Driver
142           Configuration.
143
144           make
145           memory
146           for the Kernel
147           Driver
148           Configuration.
149
150           make
151           memory
152           for the Kernel
153           Driver
154           Configuration.
155
156           make
157           memory
158           for the Kernel
159           Driver
160           Configuration.
161
162           make
163           memory
164           for the Kernel
165           Driver
166           Configuration.
167
168           make
169           memory
170           for the Kernel
171           Driver
172           Configuration.
173
174           make
175           memory
176           for the Kernel
177           Driver
178           Configuration.
179
180           make
181           memory
182           for the Kernel
183           Driver
184           Configuration.
185
186           make
187           memory
188           for the Kernel
189           Driver
190           Configuration.
191
192           make
193           memory
194           for the Kernel
195           Driver
196           Configuration.
197
198           make
199           memory
200           for the Kernel
201           Driver
202           Configuration.
203
204           make
205           memory
206           for the Kernel
207           Driver
208           Configuration.
209
210           make
211           memory
212           for the Kernel
213           Driver
214           Configuration.
215
216           make
217           memory
218           for the Kernel
219           Driver
220           Configuration.
221
222           make
223           memory
224           for the Kernel
225           Driver
226           Configuration.
227
228           make
229           memory
230           for the Kernel
231           Driver
232           Configuration.
233
234           make
235           memory
236           for the Kernel
237           Driver
238           Configuration.
239
240           make
241           memory
242           for the Kernel
243           Driver
244           Configuration.
245
246           make
247           memory
248           for the Kernel
249           Driver
250           Configuration.
251
252           make
253           memory
254           for the Kernel
255           Driver
256           Configuration.
257
258           make
259           memory
260           for the Kernel
261           Driver
262           Configuration.
263
264           make
265           memory
266           for the Kernel
267           Driver
268           Configuration.
269
270           make
271           memory
272           for the Kernel
273           Driver
274           Configuration.
275
276           make
277           memory
278           for the Kernel
279           Driver
280           Configuration.
281
282           make
283           memory
284           for the Kernel
285           Driver
286           Configuration.
287
288           make
289           memory
290           for the Kernel
291           Driver
292           Configuration.
293
294           make
295           memory
296           for the Kernel
297           Driver
298           Configuration.
299
299
300           make
301           memory
302           for the Kernel
303           Driver
304           Configuration.
305
306           make
307           memory
308           for the Kernel
309           Driver
310           Configuration.
311
312           make
313           memory
314           for the Kernel
315           Driver
316           Configuration.
317
318           make
319           memory
320           for the Kernel
321           Driver
322           Configuration.
323
324           make
325           memory
326           for the Kernel
327           Driver
328           Configuration.
329
330           make
331           memory
332           for the Kernel
333           Driver
334           Configuration.
335
336           make
337           memory
338           for the Kernel
339           Driver
340           Configuration.
341
342           make
343           memory
344           for the Kernel
345           Driver
346           Configuration.
347
348           make
349           memory
350           for the Kernel
351           Driver
352           Configuration.
353
354           make
355           memory
356           for the Kernel
357           Driver
358           Configuration.
359
360           make
361           memory
362           for the Kernel
363           Driver
364           Configuration.
365
366           make
367           memory
368           for the Kernel
369           Driver
370           Configuration.
371
372           make
373           memory
374           for the Kernel
375           Driver
376           Configuration.
377
378           make
379           memory
380           for the Kernel
381           Driver
382           Configuration.
383
384           make
385           memory
386           for the Kernel
387           Driver
388           Configuration.
389
390           make
391           memory
392           for the Kernel
393           Driver
394           Configuration.
395
396           make
397           memory
398           for the Kernel
399           Driver
400           Configuration.
401
402           make
403           memory
404           for the Kernel
405           Driver
406           Configuration.
407
408           make
409           memory
410           for the Kernel
411           Driver
412           Configuration.
413
414           make
415           memory
416           for the Kernel
417           Driver
418           Configuration.
419
420           make
421           memory
422           for the Kernel
423           Driver
424           Configuration.
425
426           make
427           memory
428           for the Kernel
429           Driver
430           Configuration.
431
432           make
433           memory
434           for the Kernel
435           Driver
436           Configuration.
437
438           make
439           memory
440           for the Kernel
441           Driver
442           Configuration.
443
444           make
445           memory
446           for the Kernel
447           Driver
448           Configuration.
449
450           make
451           memory
452           for the Kernel
453           Driver
454           Configuration.
455
456           make
457           memory
458           for the Kernel
459           Driver
460           Configuration.
461
462           make
463           memory
464           for the Kernel
465           Driver
466           Configuration.
467
468           make
469           memory
470           for the Kernel
471           Driver
472           Configuration.
473
474           make
475           memory
476           for the Kernel
477           Driver
478           Configuration.
479
480           make
481           memory
482           for the Kernel
483           Driver
484           Configuration.
485
486           make
487           memory
488           for the Kernel
489           Driver
490           Configuration.
491
492           make
493           memory
494           for the Kernel
495           Driver
496           Configuration.
497
498           make
499           memory
500           for the Kernel
501           Driver
502           Configuration.
503
504           make
505           memory
506           for the Kernel
507           Driver
508           Configuration.
509
510           make
511           memory
512           for the Kernel
513           Driver
514           Configuration.
515
516           make
517           memory
518           for the Kernel
519           Driver
520           Configuration.
521
522           make
523           memory
524           for the Kernel
525           Driver
526           Configuration.
527
528           make
529           memory
530           for the Kernel
531           Driver
532           Configuration.
533
534           make
535           memory
536           for the Kernel
537           Driver
538           Configuration.
539
540           make
541           memory
542           for the Kernel
543           Driver
544           Configuration.
545
546           make
547           memory
548           for the Kernel
549           Driver
550           Configuration.
551
552           make
553           memory
554           for the Kernel
555           Driver
556           Configuration.
557
558           make
559           memory
560           for the Kernel
561           Driver
562           Configuration.
563
564           make
565           memory
566           for the Kernel
567           Driver
568           Configuration.
569
570           make
571           memory
572           for the Kernel
573           Driver
574           Configuration.
575
576           make
577           memory
578           for the Kernel
579           Driver
580           Configuration.
581
582           make
583           memory
584           for the Kernel
585           Driver
586           Configuration.
587
588           make
589           memory
590           for the Kernel
591           Driver
592           Configuration.
593
594           make
595           memory
596           for the Kernel
597           Driver
598           Configuration.
599
599
600           make
601           memory
602           for the Kernel
603           Driver
604           Configuration.
605
606           make
607           memory
608           for the Kernel
609           Driver
610           Configuration.
611
612           make
613           memory
614           for the Kernel
615           Driver
616           Configuration.
617
618           make
619           memory
620           for the Kernel
621           Driver
622           Configuration.
623
624           make
625           memory
626           for the Kernel
627           Driver
628           Configuration.
629
630           make
631           memory
632           for the Kernel
633           Driver
634           Configuration.
635
636           make
637           memory
638           for the Kernel
639           Driver
640           Configuration.
641
642           make
643           memory
644           for the Kernel
645           Driver
646           Configuration.
647
648           make
649           memory
650           for the Kernel
651           Driver
652           Configuration.
653
654           make
655           memory
656           for the Kernel
657           Driver
658           Configuration.
659
660           make
661           memory
662           for the Kernel
663           Driver
664           Configuration.
665
666           make
667           memory
668           for the Kernel
669           Driver
670           Configuration.
671
672           make
673           memory
674           for the Kernel
675           Driver
676           Configuration.
677
678           make
679           memory
680           for the Kernel
681           Driver
682           Configuration.
683
684           make
685           memory
686           for the Kernel
687           Driver
688           Configuration.
689
690           make
691           memory
692           for the Kernel
693           Driver
694           Configuration.
695
696           make
697           memory
698           for the Kernel
699           Driver
700           Configuration.
701
702           make
703           memory
704           for the Kernel
705           Driver
706           Configuration.
707
708           make
709           memory
710           for the Kernel
711           Driver
712           Configuration.
713
714           make
715           memory
716           for the Kernel
717           Driver
718           Configuration.
719
720           make
721           memory
722           for the Kernel
723           Driver
724           Configuration.
725
726           make
727           memory
728           for the Kernel
729           Driver
730           Configuration.
731
732           make
733           memory
734           for the Kernel
735           Driver
736           Configuration.
737
738           make
739           memory
740           for the Kernel
741           Driver
742           Configuration.
743
744           make
745           memory
746           for the Kernel
747           Driver
748           Configuration.
749
750           make
751           memory
752           for the Kernel
753           Driver
754           Configuration.
755
756           make
757           memory
758           for the Kernel
759           Driver
760           Configuration.
761
762           make
763           memory
764           for the Kernel
765           Driver
766           Configuration.
767
768           make
769           memory
770           for the Kernel
771           Driver
772           Configuration.
773
774           make
775           memory
776           for the Kernel
777           Driver
778           Configuration.
779
779
780           make
781           memory
782           for the Kernel
783           Driver
784           Configuration.
785
786           make
787           memory
788           for the Kernel
789           Driver
790           Configuration.
791
792           make
793           memory
794           for the Kernel
795           Driver
796           Configuration.
797
798           make
799           memory
800           for the Kernel
801           Driver
802           Configuration.
803
804           make
805           memory
806           for the Kernel
807           Driver
808           Configuration.
809
810           make
811           memory
812           for the Kernel
813           Driver
814           Configuration.
815
816           make
817           memory
818           for the Kernel
819           Driver
820           Configuration.
821
822           make
823           memory
824           for the Kernel
825           Driver
826           Configuration.
827
828           make
829           memory
830           for the Kernel
831           Driver
832           Configuration.
833
834           make
835           memory
836           for the Kernel
837           Driver
838           Configuration.
839
840           make
841           memory
842           for the Kernel
843           Driver
844           Configuration.
845
846           make
847           memory
848           for the Kernel
849           Driver
850           Configuration.
851
852           make
853           memory
854           for the Kernel
855           Driver
856           Configuration.
857
858           make
859           memory
860           for the Kernel
861           Driver
862           Configuration.
863
864           make
865           memory
866           for the Kernel
867           Driver
868           Configuration.
869
870           make
871           memory
872           for the Kernel
873           Driver
874           Configuration.
875
876           make
877           memory
878           for the Kernel
879           Driver
880           Configuration.
881
882           make
883           memory
884           for the Kernel
885           Driver
886           Configuration.
887
888           make
889           memory
890           for the Kernel
891           Driver
892           Configuration.
893
894           make
895           memory
896           for the Kernel
897           Driver
898           Configuration.
899
899
900           make
901           memory
902           for the Kernel
903           Driver
904           Configuration.
905
906           make
907           memory
908           for the Kernel
909           Driver
910           Configuration.
911
912           make
913           memory
914           for the Kernel
915           Driver
916           Configuration.
917
918           make
919           memory
920           for the Kernel
921           Driver
922           Configuration.
923
924           make
925           memory
926           for the Kernel
927           Driver
928           Configuration.
929
930           make
931           memory
932           for the Kernel
933           Driver
934           Configuration.
935
936           make
937           memory
938           for the Kernel
939           Driver
940           Configuration.
941
942           make
943           memory
944           for the Kernel
945           Driver
946           Configuration.
947
948           make
949           memory
950           for the Kernel
951           Driver
952           Configuration.
953
954           make
955           memory
956           for the Kernel
957           Driver
958           Configuration.
959
960           make
961           memory
962           for the Kernel
963           Driver
964           Configuration.
965
966           make
967           memory
968           for the Kernel
969           Driver
970           Configuration.
971
972           make
973           memory
974           for the Kernel
975           Driver
976           Configuration.
977
978           make
979           memory
980           for the Kernel
981           Driver
982           Configuration.
983
984           make
985           memory
986           for the Kernel
987           Driver
988           Configuration.
989
990           make
991           memory
992           for the Kernel
993           Driver
994           Configuration.
995
996           make
997           memory
998           for the Kernel
999           Driver
1000           Configuration.
1001
1002
1003           make
1004           memory
1005           for the Kernel
1006           Driver
1007           Configuration.
1008
1009           make
1010           memory
1011           for the Kernel
1012           Driver
1013           Configuration.
1014
1015           make
1016           memory
1017           for the Kernel
1018           Driver
1019           Configuration.
1020
1021           make
1022           memory
1023           for the Kernel
1024           Driver
1025           Configuration.
1026
1027           make
1028           memory
1029           for the Kernel
1030           Driver
1031           Configuration.
1032
1033           make
1034           memory
1035           for the Kernel
1036           Driver
1037           Configuration.
1038
1039           make
1040           memory
1041           for the Kernel
1042           Driver
1043           Configuration.
1044
1045           make
1046           memory
1047           for the Kernel
1048           Driver
1049           Configuration.
1050
1051           make
1052           memory
1053           for the Kernel
1054           Driver
1055           Configuration.
1056
1057           make
1058           memory
1059           for the Kernel
1060           Driver
1061           Configuration.
1062
1063           make
1064           memory
1065           for the Kernel
1066           Driver
1067           Configuration.
1068
1069           make
1070           memory
1071           for the Kernel
1072           Driver
1073           Configuration.
1074
1075           make
1076           memory
1077           for the Kernel
1078           Driver
1079           Configuration.
1080
1081           make
1082           memory
1083           for the Kernel
1084           Driver
1085           Configuration.
1086
1087           make
1088           memory
1089           for the Kernel
1090           Driver
1091           Configuration.
1092
1093           make
1094           memory
1095           for the Kernel
1096           Driver
1097           Configuration.
1098
1099           make
1100           memory
1101           for the Kernel
1102           Driver
1103           Configuration.
1104
1105           make
1106           memory
1107           for the Kernel
1108           Driver
1109           Configuration.
1110
1111           make
1112           memory
1113           for the Kernel
1114           Driver
1115           Configuration.
1116
1117           make
1118           memory
1119           for the Kernel
1120           Driver
1121           Configuration.
1122
1123           make
1124           memory
1125           for the Kernel
1126           Driver
1127           Configuration.
1128
1129           make
1130           memory
1131           for the Kernel
1132           Driver
1133           Configuration.
1134
1135           make
1136           memory
1137           for the Kernel
1138           Driver
1139           Configuration.
1140
1141           make
1142           memory
1143           for the Kernel
1144           Driver
1145           Configuration.
1146
1147           make
1148           memory
1149           for the Kernel
1150           Driver
1151           Configuration.
1152
1153           make
1154           memory
1155           for the Kernel
1156           Driver
1157           Configuration.
1158
1159           make
1160           memory
1161           for the Kernel
1162           Driver
1163           Configuration.
1164
1165           make
1166           memory
1167           for the Kernel
1168           Driver
1169           Configuration.
1170
1171           make
1172           memory
1173           for the Kernel
1174           Driver
1175           Configuration.
1176
1177           make
1178           memory
1179           for the Kernel
1180           Driver
1181           Configuration.
1182
1183           make
1184           memory
1185           for the Kernel
1186           Driver
1187           Configuration.
1188
1189           make
1190           memory
1191           for the Kernel
1192           Driver
1193           Configuration.
1194
1195           make
1196           memory
1197           for the Kernel
1198           Driver
1199           Configuration.
1199
1200           make
1201           memory
1202           for the Kernel
1203           Driver
1204           Configuration.
1205
1206           make
1207           memory
1208           for the Kernel
1209           Driver
1210           Configuration.
1211
1212           make
1213           memory
1214           for the Kernel
1215           Driver
1216           Configuration.
1217
1218           make
1219           memory
1220           for the Kernel
1221           Driver
1222           Configuration.
1223
1224           make
1225           memory
1226           for the Kernel
1227           Driver
1228           Configuration.
1229
1230           make
1231           memory
1232           for the Kernel
1233           Driver
1234           Configuration.
1235
1236           make
1237           memory
1238           for the Kernel
1239           Driver
1240           Configuration.
1241
1242           make
1243           memory
1244           for the Kernel
1245           Driver
1246           Configuration.
1247
1248           make
1249           memory
1250           for the Kernel
1251           Driver
1252           Configuration.
1253
1254           make
1255           memory
1256           for the Kernel
1257           Driver
1258           Configuration.
1259
1260           make
1261           memory
1262           for the Kernel
1263           Driver
1264           Configuration.
1265
1266           make
1267           memory
1268           for the Kernel
1269           Driver
1270           Configuration.
1271
1272           make
1273           memory
1274           for the Kernel
1275           Driver
1276           Configuration.
1277
1278           make
1279           memory
1280           for the Kernel
1281           Driver
1282           Configuration.
1283
1284           make
1285           memory
1286           for the Kernel
1287           Driver
1288           Configuration.
1289
1290           make
1291           memory
1292           for the Kernel
1293           Driver
1294           Configuration.
1295
1296           make
1297           memory
1298           for the Kernel
1299           Driver
1300           Configuration.
1300
1301           make
1302           memory
1303           for the Kernel
1304           Driver
1305           Configuration.
1306
1307           make
1308           memory
1309           for the Kernel
1310           Driver
1311           Configuration.
1312
1313           make
1314           memory
1315           for the Kernel
1316           Driver
1317           Configuration.
1318
1319           make
1320           memory
1321           for the Kernel
1322           Driver
1323           Configuration.
1324
1325           make
1326           memory
1327           for the Kernel
1328           Driver
1329           Configuration.
1329
1330           make
1331           memory
1332           for the Kernel
1333           Driver
1334           Configuration.
1335
1336           make
1337           memory
1338           for the Kernel
1339           Driver
1340           Configuration.
1341
1342           make
1343           memory
1344           for the Kernel
1345           Driver
1346           Configuration.
1347
1348           make
1349           memory
1350           for the Kernel
1351           Driver
1352           Configuration.
1353
1354           make
1355           memory
1356           for the Kernel
1357           Driver
1358           Configuration.
1359
1360           make
1361           memory
1362           for the Kernel
1363           Driver
1364           Configuration.
1365
1366           make
1367           memory
1368           for the Kernel
1369           Driver
1370           Configuration.
1371
1372           make
1373           memory
1374           for the Kernel
1375           Driver
1376           Configuration.
1377
1378           make
1379           memory
1380           for the Kernel
1381           Driver
1382           Configuration.
1383
1384           make
1385           memory
1386           for the Kernel
1387           Driver
1388           Configuration.
1389
1390           make
1391           memory
1392           for the Kernel
1393           Driver
1394           Configuration.
1395
1396           make
1397           memory
1398           for the Kernel
1399           Driver
1400           Configuration.
1400
1401           make
1402           memory
1403           for the Kernel
1404           Driver
1405           Configuration.
1406
1407           make
1408           memory
1409           for the Kernel
1410           Driver
1411           Configuration.
1412
1413           make
1414           memory
1415           for the Kernel
1416           Driver
1417           Configuration.
1418
1419           make
1420           memory
1421           for the Kernel
1422           Driver
1423           Configuration.
1424
1425           make
1426           memory
1427           for the Kernel
1428           Driver
1429           Configuration.
1429
1430           make
1431           memory
1432           for the Kernel
1433           Driver
1434           Configuration.
1435
1436           make
1437           memory
1438           for the Kernel
1439           Driver
1440           Configuration.
1441
1442           make
1443           memory
1444           for the Kernel
1445           Driver
1446           Configuration.
1447
1448           make
1449           memory
1450           for the Kernel
1451           Driver
1452           Configuration.
1453
1454           make
1455           memory
1456           for the Kernel
1457           Driver
1458           Configuration.
1459
1460           make
1461           memory
1462           for the Kernel
1463           Driver
1464           Configuration.
1465
1466           make
1467           memory
1468           for the Kernel
1469           Driver
1470           Configuration.
1471
1472           make
1473           memory
1474           for the Kernel
1475           Driver
1476           Configuration.
1477
1478           make
1479           memory
1480           for the Kernel
1481           Driver
1482           Configuration.
1483
1484           make
1485           memory
1486           for the Kernel
1487           Driver
1488           Configuration.
1489
1490           make
1491           memory
1492           for the Kernel
1493           Driver
1494           Configuration.
1495
1496           make
1497           memory
1498           for the Kernel
1499           Driver
1500           Configuration.
1500
1501           make
1502           memory
1503           for the Kernel
1504           Driver
1505           Configuration.
1506
1507           make
1508           memory
1509           for the Kernel
1510           Driver
1511           Configuration.
1512
1513           make
1514           memory
1515           for the Kernel
1516           Driver
1517           Configuration.
1518
1519           make
1520           memory
1521           for the Kernel
1522           Driver
1523           Configuration.
1524
1525           make
1526           memory
1527           for the Kernel
1528           Driver
1529           Configuration.
1529
1530           make
1531           memory
1532           for the Kernel
1533           Driver
1534           Configuration.
1535
1536           make
1537           memory
1538           for the Kernel
1539           Driver
1540           Configuration.
1541
1542           make
1543           memory
1544           for the Kernel
1545           Driver
1546           Configuration.
1547
1548           make
1549           memory
1550           for the Kernel
1551           Driver
1552           Configuration.
1553
1554           make
1555           memory
1556           for the Kernel
1557           Driver
1558           Configuration.
1559
1560           make
1561           memory
1562           for the Kernel
1563           Driver
1564           Configuration.
1565
1566           make
1567           memory
1568           for the Kernel
1569           Driver
1570           Configuration.
1571
1572           make
1573           memory
1574           for the Kernel
1575           Driver
1576           Configuration.
1577
1578           make
1579           memory
1580           for the Kernel
1581           Driver
1582           Configuration.
1583
1584           make
1585           memory
1586           for the Kernel
1587           Driver
1588           Configuration.
1589
1590           make
1591           memory
1592           for the Kernel
1593           Driver
1594           Configuration.
1595
1596           make
1597           memory
1598           for the Kernel
1599           Driver
1600           Configuration.
1600
1601           make
1602           memory
1603           for the Kernel
1604          
```

## Kconf

Kernel configuration for make menuconfig.

```

80
81 source "drivers/hid/Kconfig"
82
83 source "drivers/usb/Kconfig"
84
85 source "drivers/uwb/Kconfig"
86
87 source "drivers/mmc/Kconfig"
88
89 source "drivers/memstick/Kconfig"
90
91 source "drivers/leds/Kconfig" Red circle around this line
92
93 source "drivers/nfc/Kconfig"
94
95 source "drivers/accessibility/Kconfig"
96

```

Kernel folder/driver folder

```

if (argc != 3 || sscanf(argv[1], "%d", &led_no)
    on < 0 || on > 1 || led_no < 0
    fprintf(stderr, "Usage: leds led_no 0|1"
    exit(1);
}

fd = open("/dev/leds0", 0); Red circle around /dev/leds0
if (fd < 0) {
    fd = open("/dev/leds", 0);
}
if (fd < 0) {
    perror("open device leds");
}

```

Note: 1°

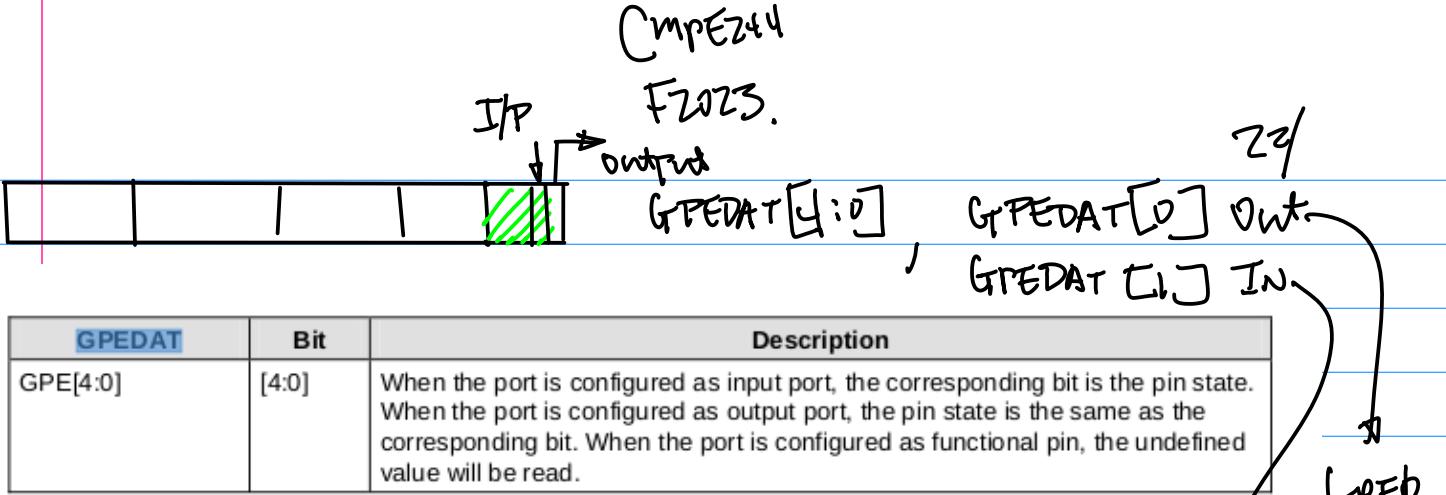
Note 2° SPR. Naming convention/addr. e.g.  
pointed from CPU Data sheet.

```

35 static long sbc2440_leds_ioctl(struct file *filp, unsigned int cmd, unsigned long arg)
36 {
37     switch(cmd) {
38         unsigned tmp;
39     case 0:
40     case 1:
41         if (arg > 4) {
42             return -EINVAL;
43         }
44         tmp = readl(S3C64XX_GPKDAT);
45         tmp &= ~(1 <(4 + arg));
46         tmp |= ((!cmd) <(4 + arg));
47        	writel(tmp, S3C64XX_GPKDAT);
48         //printk (DEVICE_NAME": %d %d\n", arg, cmd);
49         return 0;
50     default:
51         return -EINVAL;
52     }
53 }

```

↳ Bitwise Operations.



Note: Each bit (pin) can be set as an input or output by GPECON.

PP 320.

GPDCON	Bit	Description	Initial State
GPE0	[3:0]	0000 = Input 0010 = PCM SCLK[1] 0100 = AC97 BITCLK 0110 = Reserved	0000
GPE1	[7:4]	0000 = Input 0010 = PCM EXTCLK[1] 0100 = AC97 RESETn 0110 = Reserved	0000
GPE2	[11:8]	0000 = Input 0010 = PCM FSYNC[1] 0100 = AC97 SYNC 0110 = Reserved	0000
GPE3	[15:12]	0000 = Input 0010 = PCM SIN[1] 0100 = AC97 SDI 0110 = Reserved	0000
GPE4	[19:16]	0000 = Input 0010 = PCM SOUT[1] 0100 = AC97 SDO 0110 = Reserved	0000

Find the Binary Pattern to set these 2 pins.

$$GPECON[3:4] = 0001 \quad (O/P)$$

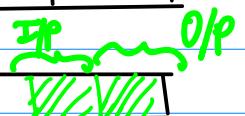
$$GPECON[7:4] = 0000 \quad (I/P)$$

for init & config.

Define Bitwise Operation mask

Design the Binary Pattern.

GPECON



mask to be designed

static void \_\_exit dev\_exit(void)

CmPE244

F2023

23/

Bitwise "OR"

Note: 1° Init module  
2°

```
66 static int __init dev_init(void)
67 {
68     int ret;
69
70     {
71         unsigned tmp;
72         tmp = readl(S3C64XX_GPECON);
73         tmp = (tmp & ~(0xffffU<<16)) | (0x1111U<<16);
74        	writel(tmp, S3C64XX_GPECON);
75
76         tmp = readl(S3C64XX_GPEDAT);
77         tmp |= (0xF << 4);
78        	writel(tmp, S3C64XX_GPEDAT);
79     }
80
81     ret = misc_register(&misc);
82
83     printk (DEVICE_NAME"\Harry: PGE initialized\n");
84
85     return ret;
86 }
```

From 2023F-105c-mini6410\_leds.c

#define DEVICE\_NAME "leds0"

static long sbc2440\_leds\_ioctl(struct file \*filp, unsigned int cmd, unsigned long arg)

static int \_\_init dev\_init(void)

static void \_\_exit dev\_exit(void)

GPIO SPI for Configuration

Hardware  
Software  
User Code  
Driver Code  
CPU  
Datasheet

Sept. 27 (Wed).

Note 1: Homework 1 #2.

Note 2: Inspection of the Prototype System

GPIO Testing.

Example: Architecture

Device Driver

OS Kernel Image

Note: PWM + I2C  
for the future discussion

Let's consider Configure + Build OS. Kernel Image.

Step 1. Download OS Distribution

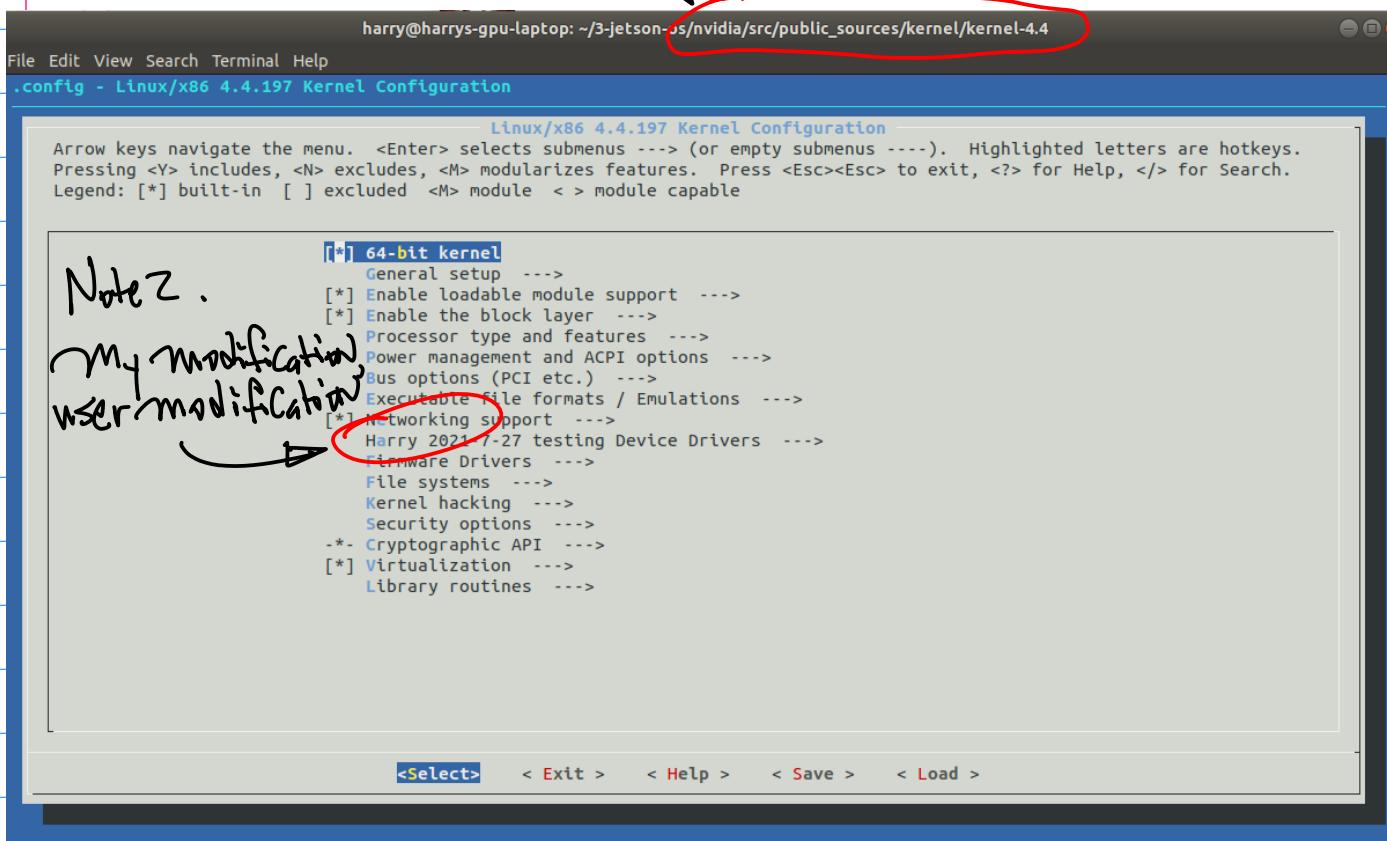
↓  
Step 2. Menutools to Build OS.

By the manufacturer's Default  
setting



Step 3. Create User-Defined  
Kernel image.

Note1: menuconfig UI



Now, Similar Setup for Sam's Arm11, Check Kconf for UI Customization  
at the Root folder.)

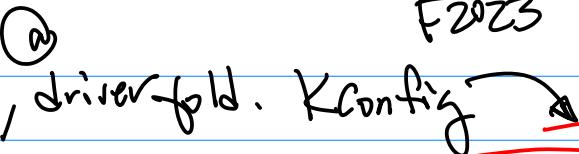
```

x harry@harry-laptop: /opt/FriendlyARM/mini6410/linux/linux-2.6.38
1 #
2 # For a description of the syntax of this configuration file,
3 # see Documentation/kbuild/kconfig-language.txt.
4 #
5 mainmenu "Linux/$ARCH $KERNELVERSION Kernel Configuration"
6
7 config SRCARCH
8     string
9     option env="SRCARCH"
10
11 source "arch/$SRCARCH/Kconfig"
~
```

CmREZt4

F2023

25/

Next Level, driver-fold. Kconfig 

```
x - harry@harry-laptop: /opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers
1 menu "Device Drivers"
2
3 source "drivers/base/Kconfig"
4
5 source "drivers/connector/Kconfig"
6
7 source "drivers/mtd/Kconfig"
8
9 source "drivers/of/Kconfig"
10
11 source "drivers/parport/Kconfig"
12
13 source "drivers/pnp/Kconfig"
14
15 source "drivers/block/Kconfig"
16
17 # misc before ide - BLK_DEV_SGII0C4 depends on SGI_IOC4
18
19 source "drivers/misc/Kconfig"
20
21 source "drivers/ide/Kconfig"
22
```

Next Level to "Char" folder, KConfig.

```
x - harry@harry-laptop: /opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char
1 #
2 # Character device configuration
3 #
4
5 menu "Character devices"
6
7 config VT
8     bool "Virtual terminal" if EXPERT
9         depends on !S390
10        select INPUT
11        default y
12        ---help---
13            If you say Y here, you will get support for terminal devices with
14            display and keyboard devices. These are called "virtual" because you
15            can run several virtual terminals (also called virtual consoles) on
16            one physical terminal. This is rather useful, for example one
17            virtual terminal can collect system messages and warnings, another
18            one can be used for a text-mode user session, and a third could run
19            an X session, all in parallel. Switching between virtual terminals
20            is done with certain key combinations, usually Alt-<function key>.
21
22            The setterm command ("man setterm") can be used to change the
23            properties (such as colors or beeping) of a virtual terminal. The
```

```

100 #
109 # Harry: Feb 17, 2016
110 #
111 config MINI6410_I2CSEN_MODULE
112     tristate "Harry 2021-2-3: I2C sensor module"
113     depends on CPU_S3C6410
114     help
115         I2C sensor module Feb 17, 2016.
116 #
117 #
118 #
119 # Harry: Here is my modification
120 #
---[END_OF_KERNEL_MODULE]

```

Ref: 2023 F-109, Readme.

Oct. 2nd (Monday) . Oct. 8 (Sun)

Note1. Homework, Due 1 week

Continuation from the Homework  
of O.S. Kernel Source Distribution  
installation.

Modify Kconfig, to

1° Add user defined device  
Driver Option as a char  
device.

The option appears as.

FirstName\_LastName\_CMPE244\_driverX, see Ref Below.

```

harry@harry-laptop: /opt/FriendlyARM/mini6410/linux/linux-2.6.38
.config - Linux/arm 2.6.38 Kernel Configuration

Character devices

Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded [M] module < > module capable

-# Virtual terminal
[ ] Support for binding and unbinding console drivers
[ ] /dev/kmem virtual device support
<M> LED Support for Mini6410 GPIO LEDs
<M> Harry 2021-2-3: I2C sensor module
<M> Harry: 2016-Feb-22, CMPE 242 Mini6410 module sample
<M> Harry: Mini6410 Test module

```

2° Provide Screen Capture  
of your UI.

3° provide Kconfig file  
which realizes this  
function.

4° Hardware Side.  
Motor Drive Board.

Note: place comments to highlight the modification

```

105      This option enables support for LEDs connected to GPIO lines
106      on Mini6410 boards.
107
108 #-----#
109 # Harry: Feb 17, 2016
110 #
111 config MINI6410_I2CSEN_MODULE
112     tristate "Harry 2021-2-3: I2C sensor module"
113     depends on CPU_S3C6410
114     help
115         I2C sensor module Feb 17 2016

```

Option to consider.

Be Careful, Power budget  
for A4988 is  
16mA.



A4988 Stepper  
Motor Driver...

\$5.89

Amazon.com  
Free shipping

Note 2: Quiz is scheduled A  
week from today Oct. 9th  
(Monday).

1° Bring the target Board &  
Prototype Board to the Class

2° Be sure to have a way  
to capture the UI.

3° Scope of the Quiz:

CPU Datasheet, CPU Architecture  
Memory-map, SPRs (Special  
Purpose Registers) for GPP.

→ ARM11, Background Info  
from LPC;

4° Execute GPIO Code

5° take a photo of your  
entire System Setup.

Cmpe244

F2023

28/

6° Screen Capture from your target Board to show the execution of the code with Personal ID is in the Screen Capture.

Time to Complete: 20 min  
for the Quiz, 15 min. to  
Prepare the Submission

Example: Continuation on KConfig.

7° Python Code.

8° Submission.

→ photos in png/jpg.

Code

A piece of paper with  
handwritten Answer.



Take a photo to Capture  
your answer sheet, then  
Convert it to pdf.

Then, place all the files  
(2 photos, 1 pdf, 1 code)

into one package, Zip it.

Naming the file:

FirstName - LastName - SID - 244.zip.  
(4 digits)

Submission to the CANVAS.

CMPE242

F2023.

29/

Objectives: To Be Able to Customize OS, Kernel, e.g. by modifying KConfig to have the following feature.

The screenshot shows a terminal window titled ".config - Linux/arm 2.6.38 Kernel Configuration". The menu path is ". Character devices". A callout points to the "Virtual terminal" section, which lists various serial port and console drivers. The "Harry" module is listed under "Harry: 2016-Feb-22, CMPE 242 Mini6410 module sample". The bottom of the window has buttons for "<Select>", "< Exit >", and "< Help >".

Note 1. The Architecture of the KConfig.

```
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38$ cd drivers/char/
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char$ ls
Kconfig
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char$ vi Kconfig
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char$ cd ..
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38$ cd ..
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38$ ls Kconfig
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38$ cd drivers/
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38$ ls Kconfig
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38$ cd char
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38$ ls Kconfig
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38$
```

a) at the Root

b. drivers

c. Char. ==

Note: To Create an entry, use "Config"

```

107
108 #
109 # Harry: Feb 17, 2016
110 #
111 config MINI6410_I2CSEN_MODULE
112 tristate "Harry 2021-2-3: I2C sensor module"
113 depends on CPU_S3C6410
114 help
115     I2C sensor module Feb 17, 2016.
116 #
117
118 #

```

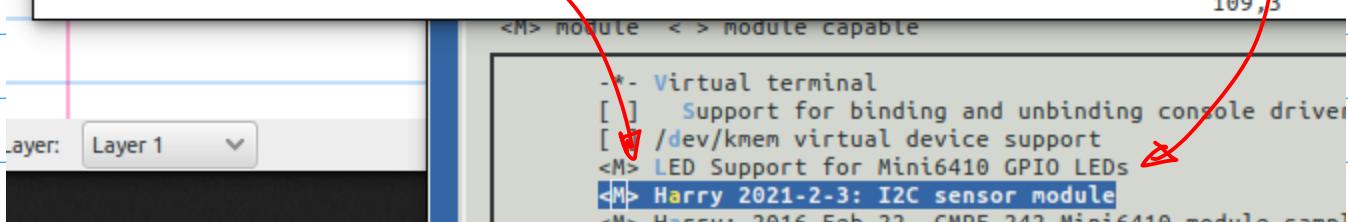
Note: To define the target CPU, use "depends on" followed by Manufacturer ID & Device ID.

```

107
108 #
109 # Harry: Feb 17, 2016
110 #
111 config MINI6410_I2CSEN_MODULE
112 tristate "Harry 2021-2-3: I2C sensor module"
113 depends on CPU_S3C6410
114 help
115     I2C sensor module Feb 17, 2016.
116 #
117
118 #
119 # Harry: Here is my modification
120 #

```

Note:



Note 2: To pre-select. use default "y"

```

70
91 config DEVKMEM
92     bool "/dev/kmem virtual device support"
93     default y
94     help
95         Say Y here if you want to support the /dev/kmem device. The
96         /dev/kmem device is rarely used, but can be used for certain
97         kind of kernel debugging operations.
98         When in doubt, say "N".

```

Note: To provider Description  
for the option, use  
"help"

Once the KConfig is updated with matching  
Name of the Device Driver, then \$make all

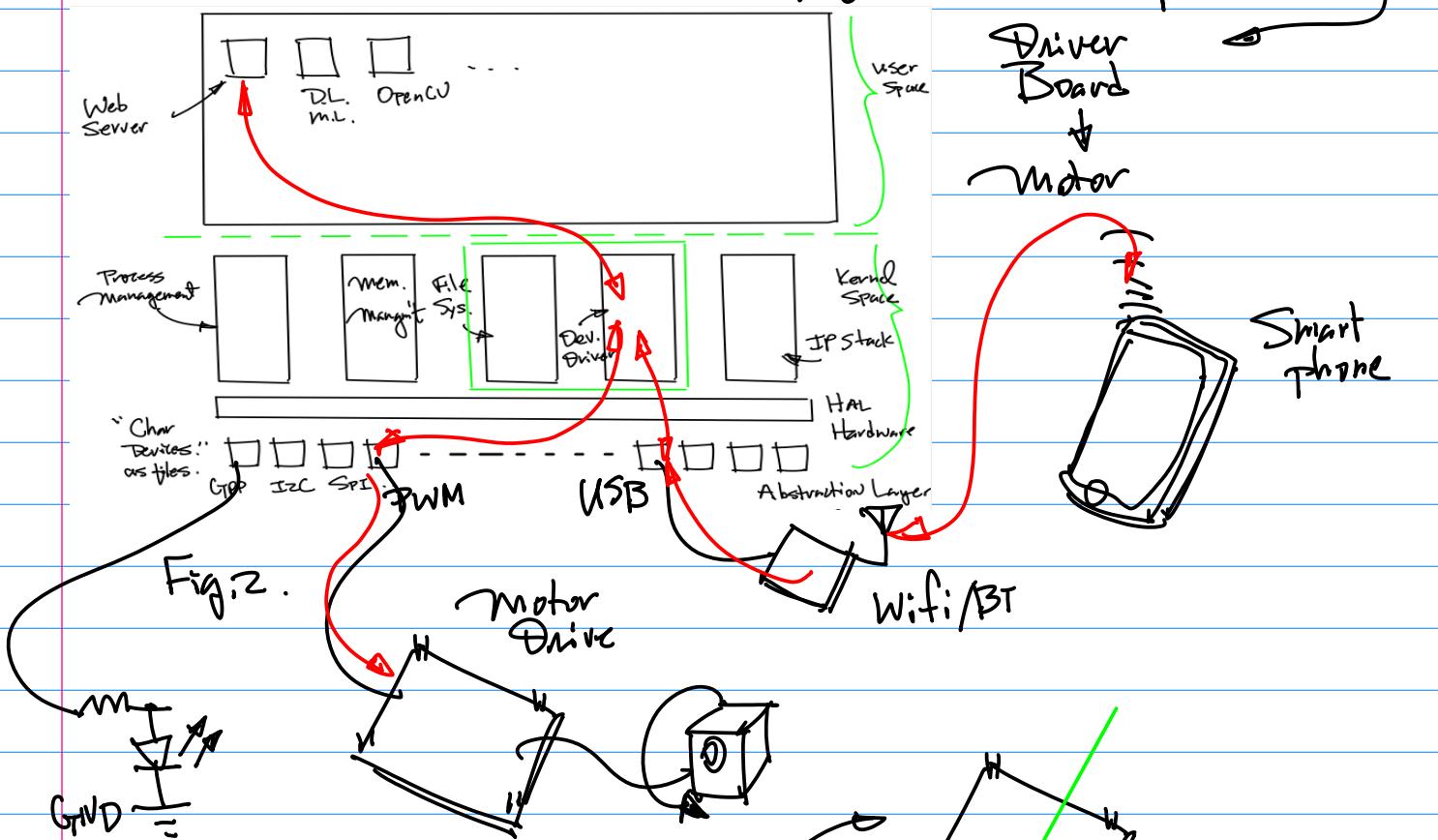
\$insmod *driverName*  
\$printk( )  
\$rmmod *driverName*

Oct.4 (Wed).

Consider Debugging the Device  
Driver(s) from manufacturer's  
distribution.

Example: Embedded Software in this

Example: User Smartphone → Jetson Nano → USER APP. → Device  
Web Server Web Server Program Driver



Background On the  
motor & motor Driver Board.

1° Stepper motor pins. (4)

A+, A-, B+, B-

2° Driver Board Pins

Input Group

Output Group. (4 pins)

ENABLE>Select.

Speed Input Pin. PWM f<sub>PWM</sub>

Direction Control. GPP

Duty Cycle  
Others. { Configuration Pins. x

Others... /

2x No. of  
Config. Patterns

Table 1. Connectivity Table  
Driver Board to the Stepper Motor.

Driver Board	NEMA 17
A+	A+
A-	A-
B+	B+
B-	B-

Now, for the Configuration of the driver board.

Background On the Stepper motor Drive Board.

Note: 1 Full Step of the Stepper Motor.

$$\frac{2\pi}{200} \Rightarrow \frac{360}{200}$$

1.8 degree/step

1 half Step. 0.9 Degree/  
Half Step

$\frac{1}{4}$  Step 0.45 Degree

$\frac{1}{8}$  Step 0.225 Degree.

:

A typical Driver Board consists of 4 pins for Configuration Conn1, Conn2, Conn3, Conn4

Example:  $f_{pwm}$  is set to its target frequency,  $f_{pwm} = 2 \times 10^3$

And Duty Cycle 5% to 95% Allows the Control of the Speed.

Sample code to Realize this Control function.

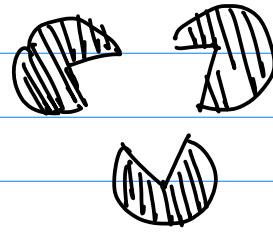
a. Set f<sub>pwm</sub>

b. Set & Change Duty Cycle.

Ref: Ref code to Be posted on line. To Be Referenced for your design & debugging

Note 1.

```
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/examples/pwm$ makefile pwm_test pwm_test.c pwm_testNew.c temp.c
```



CMPE244

F2023

34/

```

99     open_buzzer();
100
101    printf( "\nBUZZER TEST ( PWM Control )\n" );
102    printf( "Press +/- to increase/reduce the frequency of the BUZZER\n" );
103    printf( "Press 'ESC' key to Exit this program\n\n" );
104
105
106    while( 1 )

```

Note: Set  $f_{PWM}$ , But Need to set duty cycle as well.

```

109
110    set_buzzer_freq(freq);
111    printf( "\tFreq = %d\n", freq );
112
113    key = getch();
114
115    switch(key) {

```

Note: Access to the Device Driver By "fd" (file Descriptor).

```

4/
48 static int fd = -1;
49 static void close_buzzer(void);
50 static void open_buzzer(void)
51 {
52     fd = open("/dev/pwm", 0);
53     if (fd < 0) {
54         perror("open pwm_buzzer");
55         exit(1);
56     }
57 }

```

## Oct 8th (Monday)

Example: Debugging PWM Driver (ATMEL Example).

References with C code have been posted, And IDs are provided to the class.

C } user Space Code.  
  } Kernel Space Code.

Objective: To modify the Device Driver Code to include Duty function.

To upgrade the user Code to Allow Both  $f_{PWM}$  and

$f_{PWM}$

$f_{PWM}$ .

modification of this module?  
Add duty cycle if possible.

Duty Cycle to Be Defined and Selected by user.

Step 1. Modify / Upgrade User Space Code, to Add duty function.

Step 2. Modify / Upgrade Kernel Space Code, e.g., device Driver.

Control / Config.  $f_{PWM}$ .

Control / Config. Duty Cycle

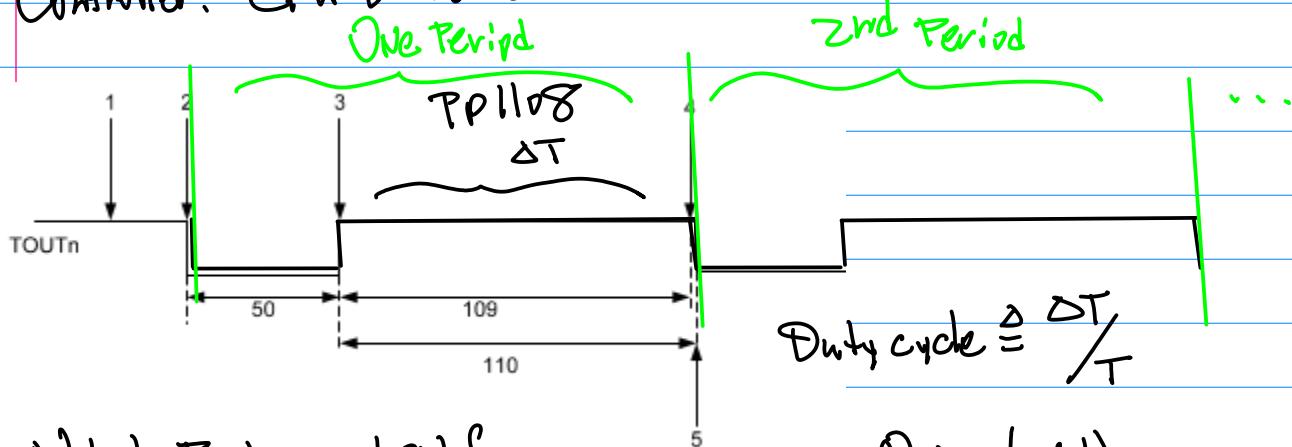
Step 3. Inspection of Control / Configuration Registers, e.g., SFRs to Allow Duty to Be a part of the Driver function.

CPU Data sheet.

Step 4. Option, But if the Name of the Device Driver is Changed One to

the upgrade, then we need to change/update the script in "KConfig".

## Discussion on PWM Peripheral Controller. CPU Datasheet



Note 1. Background on f<sub>PWM</sub>

System Clock  $\rightarrow$  PCLK  $\rightarrow$  PWM  
 $f_{sys}$   $\xrightarrow{\text{peripheral clock.}}$   $f_{PCLK}$   $\xrightarrow{\text{f}_{PWM}}$

SPR(s) to Control/Config.

PCLK.

$\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \dots$

16 bits.

$$f_{PWM} = \frac{f_{PCLK}}{DUSR * \cancel{\text{Prescaler}}} \dots (1)$$

Not necessarily tied to a commercial product.

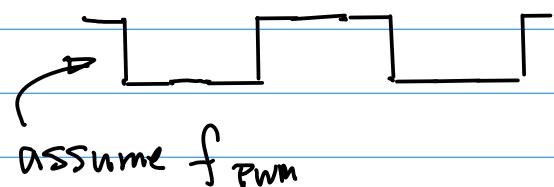
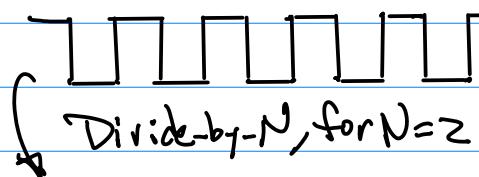
Oct. 11 (Wed)

Continuation on PWM Driver Debugging & Update

$f_{PWM}$   
Duty

Note 1. Waveform for Divide-by-N Counter

Input CLK  
PCLK



If  $N$  is 8 bits, then

$$N \in [0, 255]$$

Question: How many different frequencies for  $f_{\text{PWM}}$  can we have with this  $N$ ?

Alternative to Increase the No. of  $f_{\text{PWM}}$  by introducing a Product

$$N = N_1 * N_2$$

where  $N_1$  is 8 bit,  $N_2$  is 8 bit

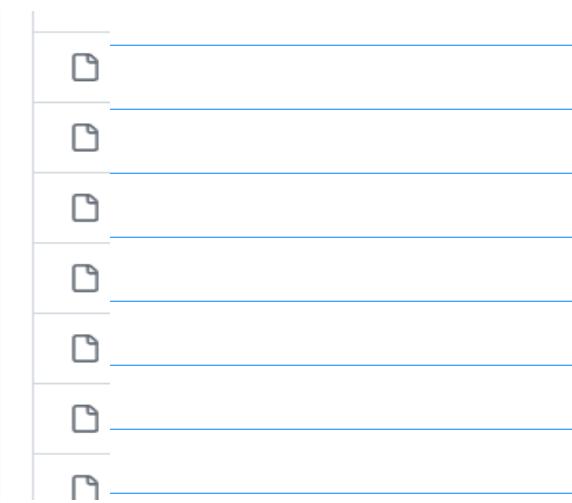
To Achieve the increase the No. of  $f_{\text{PWM}}$ .

Consider A Duty Cycle Implementation

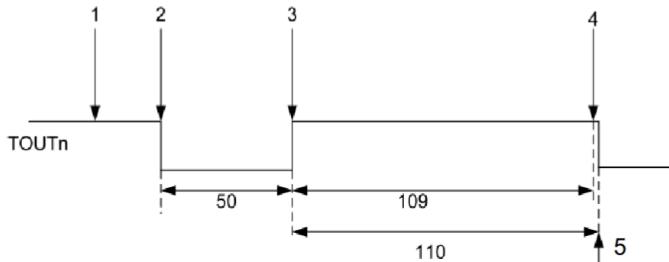
We need { A Counter { Count Up  
Count Down.  
} Interrupt }

Ref. from github for CmPEZ42

- 2022S-107a-PID-v5-2019-4-25.pdf
- 2022S-107b-pwm-pid.pdf
- 2022S-107c-pwm-config-v4-hl-2022-3-3.pdf
- 2022S-107d-pwm-nano-coding-hl-2021-12-19.pdf
- 2022S-107e-pwm-waveform-v3-2018-3-4.jpg
- 2022S-107f-pwm-specialPurposeRegister-v3-20...
- 2022S-107g-pwm-calculation-v3-2018-3-4.pdf
- 2022S-107h-spec-motor-drive-WS55-1800 (copy...)
- 2022S-108-L5M2023IUC.pdf



## PWM Operation



Example:

1. Initialize the TCNTBn with 159(50+109) and the TCMPBn with 109.
2. Start Timer by setting the start bit and manual update bit off.
- The TCNTBn value of 159 is loaded into the down-counter, the output is driven low.
3. When down-counter counts down to the value in the TCMPBn register 109, the output is changed from low to high.
4. When the down-counter reaches 0, the interrupt request is generated.
5. The down-counter is automatically reloaded with TCNTBn, which restarts the cycle.

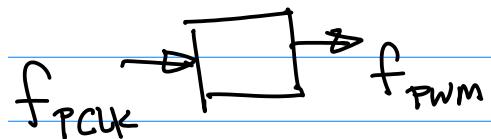
Example: Given  $f_{sys} = 800 \text{ MHz}$ .

$$\text{assume } f_{PCLK} = 800/8 = 100 \text{ MHz}$$

assume

$$f_{PWM} = 2 \text{ kHz}$$

find



Try:

$$TCNTB\phi = \frac{f_{PCLK}}{f_{PWM}}$$

from the given condition

## TCFG0, TCNTBn and TCMPBn

### 32.4.1.1 TCFG0 (Timer Configuration Register), pp 1118

Register	Offset	R/W	Description
TCFG0	0x7F006000	R/W	Timer Configuration Register 0 that configures the two 8-bit Prescaler and DeadZone Length

Timer input clock Frequency = PCLK / ( {prescaler value + 1} ) / {divider value}  
{prescaler value} = 1~255  
{divider value} = 1, 2, 4, 8, 16, TCLK

### 32.4.1.2 TCFG1 (Timer Configuration Register)

Register	Offset	R/W	Description
TCFG1	0x7F006004	R/W	Timer Configuration Register 1 that controls 5 MUX and DMA Mode Select Bit

### 32.4.1.3 TCON (Timer Control Register)

Register	Offset	R/W	Description
TCON	0x7F006008	R/W	Timer Control Register

### 32.4.1.5 TCMPB0 (Timer0 Compare Register)

Register	Offset	R/W	Description
TCMPB0	0x7F006010	R/W	Timer 0 Compare Buffer Register

Harry Li, Ph.D.

## 32.4 SPECIAL FUNCTION REGISTERS

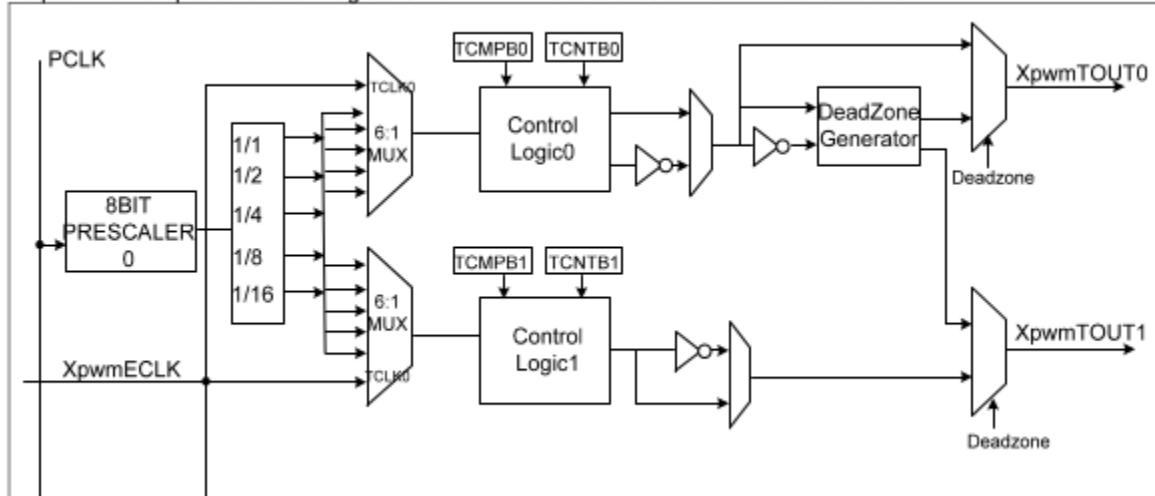
## 32.4.1 REGISTER MAP

Note1. TCNTB0PP1117  
CNTB : Count Buffer

Register	Offset	R/W	Description	Reset Value
TCFG0	0x7F006000	R/W	Timer Configuration Register 0 that configures the two 8-bit Prescaler and DeadZone Length	0x0000_0101
TCFG1	0x7F006004	R/W	Timer Configuration Register 1 that controls 5 MUX and DMA Mode Select Bit	0x0000_0000
TCON	0x7F006008	R/W	Timer Control Register	0x0000_0000
<u>1</u> TCNTB0	<u>0x7F00600C</u>	R/W	<u>Timer 0 Count Buffer Register</u>	0x0000_0000
<u>2</u> TCMPB0	<u>0x7F006010</u>	R/W	<u>Timer 0 Compare Buffer Register</u>	0x0000_0000
TCNTO0	0x7F006014	R	Timer 0 Count Observation Register	0x0000_0000
TCNTB1	0x7F006018	R/W	Timer 1 Count Buffer Register	0x0000_0000
TCMPB1	0x7F00601c	R/W	Timer 1 Compare Buffer Register	0x0000_0000
TCNTO1	0x7F006020	R	Timer 1 Count Observation Register	0x0000_0000
TCNTB2	0x7F006024	R/W	Timer 2 Count Buffer Register	0x0000_0000
TCNTO2	0x7F00602c	R	Timer 2 Count Observation Register	0x0000_0000
TCNTB3	0x7F006030	R/W	Timer 3 Count Buffer Register	0x0000_0000
TCNTO3	0x7F006038	R	Timer 3 Count Observation Register	0x0000_0000
TCNTB4	0x7F00603c	R/W	Timer 4 Count Buffer Register	0x0000_0000
TCNTO4	0x7F006040	R	Timer 4 Count Observation Register	0x0000_0000
TINT_CSTAT	0x7F006044	R/W	Timer Interrupt Control and Status Register	0x0000_0000

Reference for  $f_{Pwm}$  formula.

output. The 16 special function registers within PWM are accessed via APB transactions.



Oct 16 (mon).

Note1. Homework Assignment

On CANVAS:

## PWM Device Driver + Board Implementation.

### CMPE 244 Homework PWM Driver Debugging and Stepper Motor Actuation Demo

#### Part I:

1. Download the PWM user program and driver program from the class github, see the figure below for the document ID:

2023F-105d-user-p...	Add files via upload	3 weeks ago
2023F-105e-user-p...	Add files via upload	3 weeks ago
2023F-105f-#20-20...	Add files via upload	3 weeks ago

The github for the pwm user space program:

[https://github.com/hualili/CMPE244/blob/main/2023F-105d-user-pwm\\_test.c](https://github.com/hualili/CMPE244/blob/main/2023F-105d-user-pwm_test.c)

Example: Continuation of the PWM modification.

SPR.

Define the Counts value to Achieve Required Duty Cycle.

$T_{CNTBn}$  is compared against  $T_{CMPBn} = 109$ .

Then, if equal, trig the waveform change from the default "0" to high "1"

Design Requirements for the example:

Suppose 33% Duty Cycle.

Initialize  $T_{CMPBn} = ?$

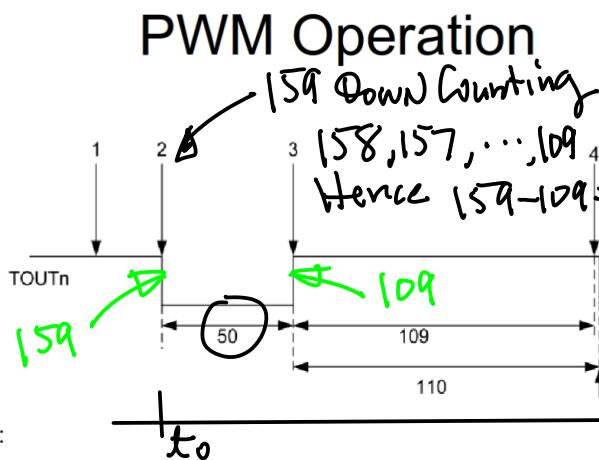
$$\therefore T_{CNPBn} = 50 \times 10^3$$

We can now,

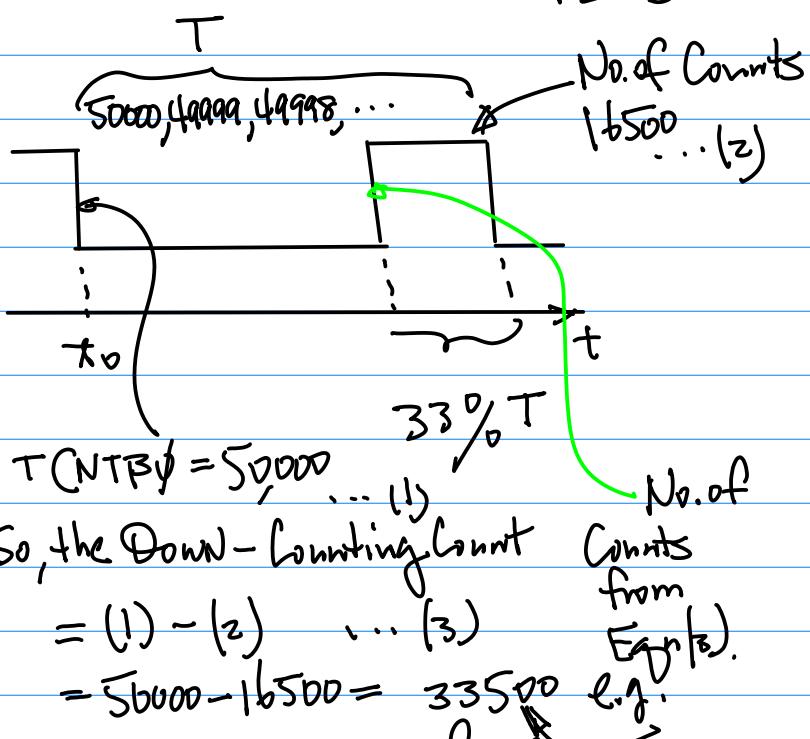
$$0.33 \times 50 \times 10^3$$

$$= 16500$$

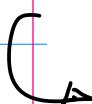
Now, to visualize it, we have



1. Initialize the  $TCNTBn$  with 159(50+109) and the  $TCMPBn$  with 109.
2. Start Timer by setting the start bit and manual update bit off.
3. When down-counter counts down to the value in the  $TCMPBn$  register 109, the output is changed from low to high.
4. When the down-counter reaches 0, the interrupt request is generated.
5. The down-counter is automatically reloaded with  $TCNTBn$ , which restarts the cycle.



Now, Read C Code of Device Driver.



On the Class  
github.

2023F-105F-#20-2021S-9-  
mini6410\_pwmHarry (1).c

```

multi-thread.py 2022S-103c-#nn_sample_2022.py 2023F-105F-#20..._pwmHar
1  /*
2   * Program: mini6410_pwmHarry.c;
3   * Modified by: HL; Date: Feb 10, 2016 *
4   * Purpose: To add PWM duty cycle change function *
5   *           by altering CMP compare register. *
6   */
7
8 #include <linux/module.h>
9 #include <linux/kernel.h>
10 #include <linux/fs.h>
11 #include <linux/init.h>
12 #include <linux/delay.h>
13 #include <linux/poll.h>
14 #include <asm/irq.h>
15 #include <asm/io.h>
16 #include <linux/interrupt.h>
17 #include <asm/uaccess.h>
18 #include <mach/hardware.h>
19 #include <plat/regs-timer.h>
20 #include <mach/regs-irq.h>
21 #include <asm/mach/time.h>
22 #include <linux/clk.h>
23 #include <linux/cdev.h>
24 #include <linux/device.h>
25 #include <linux/miscdevice.h>
26
27 #include <mach/map.h>
28 #include <mach/regs-clock.h>
29 #include <mach/regs-gpio.h>
```

Note 1. Driver Name in the User Space,

```

35 #define DEVICE_NAME "pwm"
36
37 #define PWM_IOCTL_SET_FREQ 1
38 #define PWM_IOCTL_STOP 0
39
40 static struct semaphore lock;
41
42

```

`OPEN("drivers/pwm" ...)`

Initially Enabled.

Note 2. Add the function  
to change duty cycle.  
`PWM_IOCTL_SET_DUTY`  
for Example.

```

41 static struct semaphore lock;
42
43 /* freq: pclk/50/16/65536 ~ pclk/50/16
44 * if pclk = 50MHz, freq is 1Hz to 62500Hz
45 * human ear : 20Hz~ 20000Hz
46 */
47 //static void PWM_Set_Freq( unsigned long freq)
48 static void PWM_Set_Freq( unsigned long freq, duty ) //Hz
49 {
50     unsigned long tcon;
51     unsigned long tcnt;
52     unsigned long tduy; //Harry
53     unsigned long tcfg1;
54     unsigned long tcfg0;
55
56     struct clk *clk_p;
57     unsigned long pclk;
58
59     unsigned tmp;
60
61     tmp = readl(S3C64XX_GPFCON);
62     tmp &= ~(0x3U << 28);
63     tmp |= (0x2U << 28);
64    	writel(tmp, S3C64XX_GPFCON);
65
66     tcon = __raw_readl(S3C_TCON);
67     tcfg1 = __raw_readl(S3C_TCFG1);
68     tcfg0 = __raw_readl(S3C_TCFG0);
69

```

Note 3. Preserve the module Name  
for minimum Modification  
or, update to  
`PWM_Set_Freq_Duty( )`.  
New Parameter for the  
Duty Cycle is added.

#### 32.4.1.3 TCON (Timer Control Register)

Register	Offset	R/W	Descr
TCON	0x7F006008	R/W	Timer Control Register
TCON	Bit	R/W	

TP 1119  
Read Datasheet, for Example "iveshot"  
V.S. "Auto-Reload" etc.

Oct. 18 (Wed)

Note: Roadmap of the Class



Introduction.

CPU/SPR/Driver

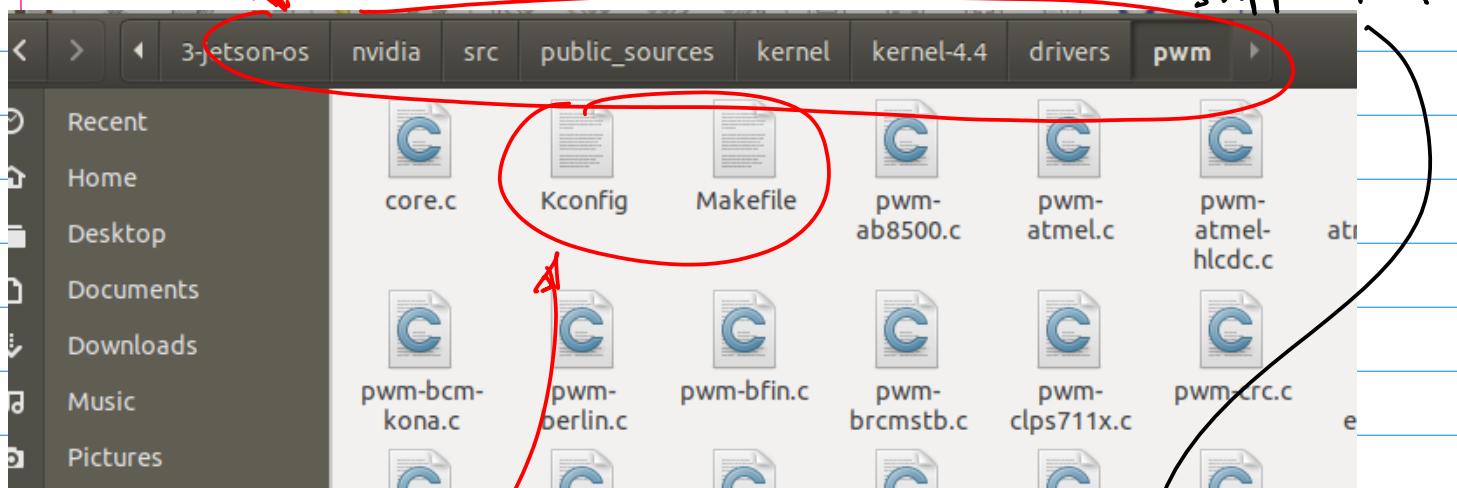
Device Driver  
Debug Development  
PWM

In About a week Time frame,  
in-Class Brief presentation (Updates, e.g.  
one paragraph Description for the  
Semester Long Project.)

Oct. 25 (Wed)

2 Person Team OR  
individual.

Q & A : Note 1. Device Driver folder, PWM Device Driver, Target Platform Device Driver: Note 2. Kernel 4.4 is chosen for my need with Ubuntu 18.04 on Jetson NAND. To Support AI.



Note 3. Two files.

KConfig → UI for menuconfig

Script per the  
Developer choice  
To Build Kernel  
image and Device  
Drivers.

makefile.

Per NVIDIA Requirements.

```

1 menuconfig PWM
2   bool "Pulse-Width Modulation (PWM) Support"
3   help
4     Generic Pulse-Width Modulation (PWM) support.
5
6     In Pulse-Width Modulation, a variation of the width of pulses
7     in a rectangular pulse signal is used as a means to alter the
8     average power of the signal. Applications include efficient
9     power delivery and voltage regulation. In computer systems,
10    PWMs are commonly used to control fans or the brightness of
11    display backlights.
12
13    This framework provides a generic interface to PWM devices
14    within the Linux kernel. On the driver side it provides an API
15    to register and unregister a PWM chip, an abstraction of a PWM
16    controller, that supports one or more PWM devices. Client
17    drivers can request PWM devices and use the generic framework
18    to interface with them directly.

```

Some support Target CPU.

```

44 config PWM_ATMEL
45     tristate "Atmel PWM support"
46     depends on ARCH_AT91 || AVR32
47     help
48         Generic PWM framework driver for Atmel SoC.
49
50         To compile this driver as a module, choose M here: the module
51         will be called pwm-atmel.
52

```

Note: Broadcom

```

77 config PWM_BCM_KONA
78     tristate "Kona PWM support"
79     depends on ARCH_BCM_MOBILE
80     help
81         Generic PWM framework driver for Broadcom Kona PWM block.
82
83         To compile this driver as a module, choose M here: the module
84         will be called pwm-bcm-kona.
85
86 config PWM_BCM2835
87     tristate "BCM2835 PWM support"
88     depends on ARCH_BCM2835
89     help
90         PWM framework driver for BCM2835 controller (Raspberry Pi)

```

Graphics Acceleration  
Unit.

Note: CPU from NXP

```

201
202 config PWM_LPC18XX_SCT
203     tristate "LPC18xx/43xx PWM/SCT support"
204     depends on ARCH_LPC18XX
205     help
206         Generic PWM framework driver for NXP LPC18
207         supports 16 channels.
208         A maximum of 15 channels can be requested
209         must have the same period.
210
211         To compile this driver as a module, choose
212         will be called pwm-lpc18xx-sct.
213
214 config PWM_LPC32XX
215     tristate "LPC32XX PWM support"
216     depends on ARCH_LPC32XX
217     help
218         Generic PWM framework driver for LPC32XX.

```

Activities Text Editor ▾

Open ▾

Wed 17:07 Kconfig -/3-jetson-os/nvidia/src/public\_sources/kernel/kernel-4.4/drivers/pwm Save

```

354 netp Generic PWM framework driver for STiH4xx SoCs.
355
356 To compile this driver as a module, choose M here: the module
357 will be called pwm-sti.
358
359
360 config PWM SUN4I
361 tristate "Allwinner PWM support"
362 depends on ARCH_SUNXI || COMPILER_TEST
363 depends on HAS_IOMEM && COMMON_CLK
364 help
365 Generic PWM framework driver for Allwinner SoCs.
366
367 To compile this driver as a module, choose M here: the module
368 will be called pwm-sun4i.
369
370 config PWM TEGRA
371 tristate "NVIDIA Tegra PWM support"
372 depends on ARCH_TEGRA
373 help
374 Generic PWM framework driver for the PWM controller found on NVIDIA
375 Tegra SoCs.
376
377 To compile this driver as a module, choose M here: the module
378 will be called pwm-tegra.
379
380 config PWM TEGRA_DFL
381 tristate "NVIDIA Tegra DFL PWM support"
382 depends on ARCH_TEGRA
383 help
384 PWM driver support for the Tegra DFL module found on NVIDIA
385 Tegra SoCs.
386
387 To compile this driver as a module, choose M here: the module
388 will be called pwm-tegra-dfl.
389
390 config PWM TEGRA_PMC_BLINK
391 tristate "NVIDIA Tegra PMC blink PWM support"
392 depends on ARCH_TEGRA
393 help
394 PWM driver support for the Tegra PMC blink module found on NVIDIA
395 Tegra SoCs.
396
397 To compile this driver as a module, choose M here: the module

```

*Note. TEGRA CPUs for Jetson NAND  
one of*

Plain Text ▾ Tab Width: 8 ▾ Ln 330, Col 19 ▾ INS

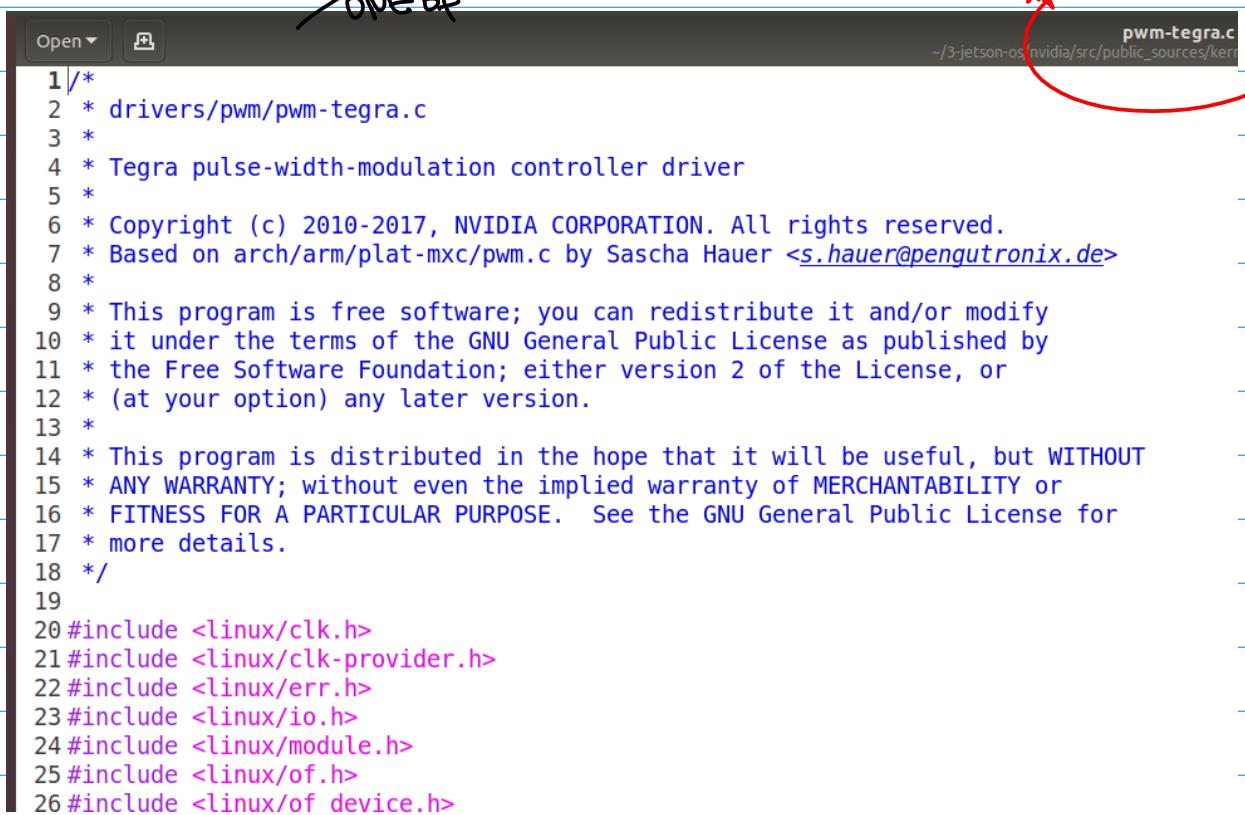
CMPE244

F2023

Note: makefile as the result output from KConf. Drivers Built 44|  
per Developer Selection.

```
File Edit View Search Terminal Help  
Subdir-ccflags-y += -Werror  
  
obj-$(CONFIG_PWM) += core.o  
obj-$(CONFIG_PWM_SYSFS) += sysfs.o  
obj-$(CONFIG_PWM_AB8500) += pwm-ab8500.o  
obj-$(CONFIG_PWM_ATMEL) += pwm-atmel.o  
obj-$(CONFIG_PWM_ATMEL_HLCD_C_PWM) += pwm-atmel-hlcdc.o  
obj-$(CONFIG_PWM_ATMEL_TCB) += pwm-atmel-tcb.o  
obj-$(CONFIG_PWM_BCM_KONA) += pwm-bcm-kona.o  
obj-$(CONFIG_PWM_BCM2835) += pwm-bcm2835.o  
obj-$(CONFIG_PWM_BERLIN) += pwm-berlin.o  
obj-$(CONFIG_PWM_BFIN) += pwm-bfin.o  
obj-$(CONFIG_PWM_BRCMSTB) += pwm-brcmstb.o  
obj-$(CONFIG_PWM_CLPS711X) += pwm-clps711x.o  
obj-$(CONFIG_PWM_CRC) += pwm-crc.o  
obj-$(CONFIG_PWM_EP93XX) += pwm-ep93xx.o  
obj-$(CONFIG_PWM_FSL_FTM) += pwm-fsl-ftm.o  
obj-$(CONFIG_PWM_IMG) += pwm-img.o  
obj-$(CONFIG_PWM_IMX) += pwm-imx.o  
obj-$(CONFIG_PWM_JZ4740) += pwm-jz4740.o  
obj-$(CONFIG_PWM_LP3943) += pwm-lp3943.o  
obj-$(CONFIG_PWM_LPC18XX_SCT) += pwm-lpc18xx-sct.o  
obj-$(CONFIG_PWM_LPC32XX) += pwm-lpc32xx.o  
obj-$(CONFIG_PWM_LPSS) += pwm-lpss.o  
obj-$(CONFIG_PWM_LPSS_PCI) += pwm-lpss-pci.o  
obj-$(CONFIG_PWM_LPSS_PLATFORM) += pwm-lpss-platform.o  
obj-$(CONFIG_PWM_MTK_DISP) += pwm-mtk-disp.o  
obj-$(CONFIG_PWM_MXS) += pwm-mxs.o  
obj-$(CONFIG_PWM_PCA9685) += pwm-pca9685.o  
obj-$(CONFIG_PWM_PUV3) += pwm-puv3.o  
obj-$(CONFIG_PWM_PXA) += pwm-pxa.o  
obj-$(CONFIG_PWM_RCAR) += pwm-rcar.o  
obj-$(CONFIG_PWM_RENESAS_TPU) += pwm-renesas(tpu).o  
obj-$(CONFIG_PWM_ROCKCHIP) += pwm-rockchip.o  
obj-$(CONFIG_PWM_SAMSUNG)
```

Note: Inspection of PWM Driver  
ONE OF



```

1 /*
2  * drivers/pwm/pwm-tegra.c
3  *
4  * Tegra pulse-width-modulation controller driver
5  *
6  * Copyright (c) 2010-2017, NVIDIA CORPORATION. All rights reserved.
7  * Based on arch/arm/plat-mxc/pwm.c by Sascha Hauer <s.hauer@pengutronix.de>
8  *
9  * This program is free software; you can redistribute it and/or modify
10 * it under the terms of the GNU General Public License as published by
11 * the Free Software Foundation; either version 2 of the License, or
12 * (at your option) any later version.
13 *
14 * This program is distributed in the hope that it will be useful, but WITHOUT
15 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
16 * FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for
17 * more details.
18 */
19
20 #include <linux/clk.h>
21 #include <linux/clk-provider.h>
22 #include <linux/err.h>
23 #include <linux/io.h>
24 #include <linux/module.h>
25 #include <linux/of.h>
26 #include <linux/of_device.h>
```

Example: Continued from

PWM Driver Code.

```

66 tcon = __raw_readl(S3C_TCON);
67 tcfg1 = __raw_readl(S3C_TCFG1);
68 tcfg0 = __raw_readl(S3C_TCFG0);
69 ... - - -
```

#### 32.4.1.3 TCON (Timer Control Register)

Register	Offset	R/W	Description
TCON	0x7F006008	R/W	Timer Control Register

TCON	Bit	R/W	Description
Reserved	[31:23]	R	Reserved Bits
Timer 4 Auto Reload on/off	[22]	R/W	0: One-Shot 1: Interval Mode(Auto-F
Timer 4 Manual Update	[21]	R/W	0: No Operation 1: Update TCNTB4
Timer 4 Start/Stoo	[20]	R/W	0: Stop 1: Start Timer 4