

Jan 27, 21
Welcom to

CMPE242

Harry Li

Embedded Hardware Systems

1. Green Sheet [github/hualili/cmpe242](https://github.com/hualili/cmpe242)

Email: hua.li@sisu.edu

(650) 400-1116 Text message

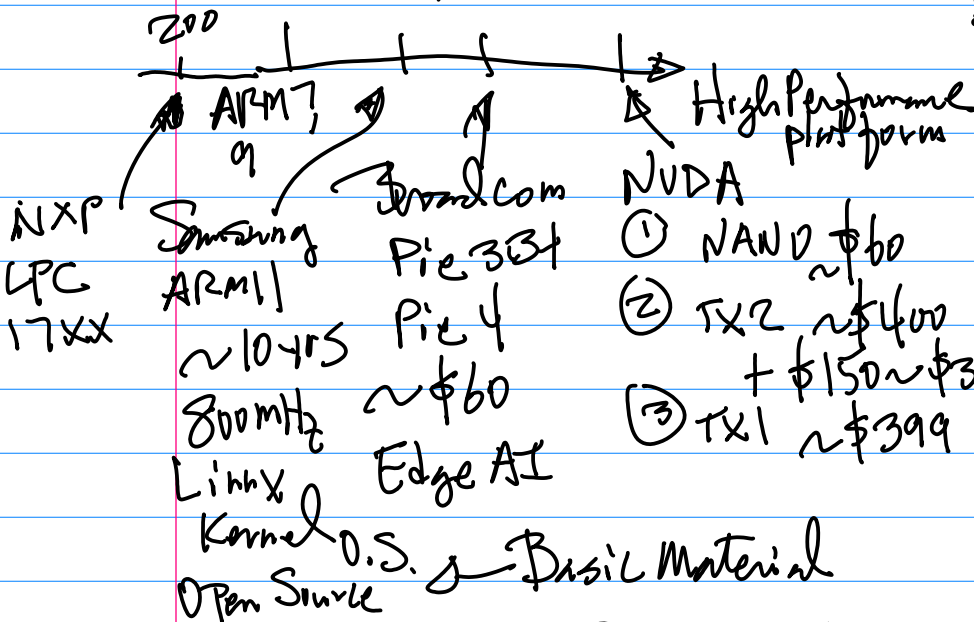
2. Pre-requist Requirements 180A & D

Course Description

Hands-ON.

Target Development Platform

200



Scope of the Course

Device Driver
Dep Development
C/C++ Python
O.S. Kernel

Grading Policy

3-3-4
mid Projects Final

CANVAS - EE242

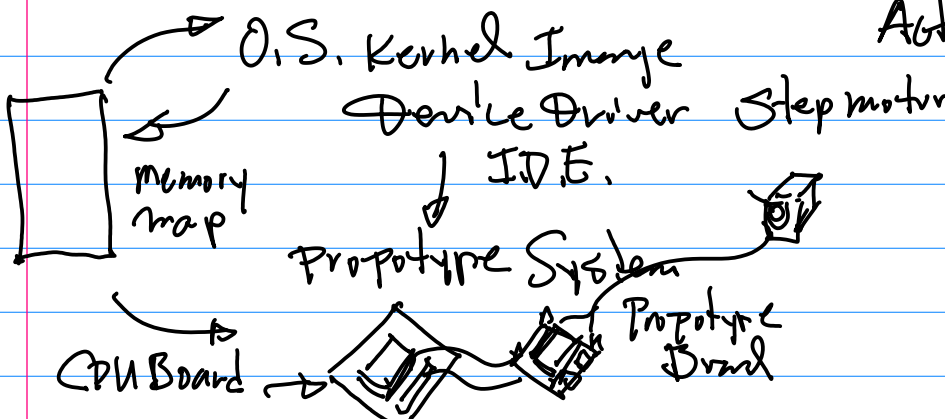
① Assignments (No)

② Submission / Submission of your class work

Action 1. [github/hualili/cmpe242](https://github.com/hualili/cmpe242)

2. Datasheet

Samsung ARM11 Datasheet
NXP LPC 1769 Datasheet
Architecture NXP LPC 17XX



Action 3. Target Platform Selection
of Unix-like OS. 6) Edge AI
Computing (Scalability) \Rightarrow GPU

Office Hours M.W. 4:30-5:30pm
ON Zoom.

CMPE242 Feb. 2021

Today's Topics: 1° System Architecture
Review, CPU Datasheet; 2° Target
platform

BaseLine Software

Linux Distribution
Optimized for Embedded
platform

Example: Datasheet

LPC1769 ARM Cortex M3

Samsung ARM11

NVDA ... Piex

System Architecture

NVDA CPU/GPU platform.
BroadCom, Piex 3/4 G.E. (Graphics
Engine)
Samsung ARM11 (9, 7)

NAJO
TX1, TX2
Graphics
Engine

CPU Datasheet, LPC1769

20185-3-UM10360, P.P. 9

Jtag

Optional Architecture RISC-V

Common Characteristics RISC

Reduced Instruction Set Computer

CORE
ARM
Cortex
M3

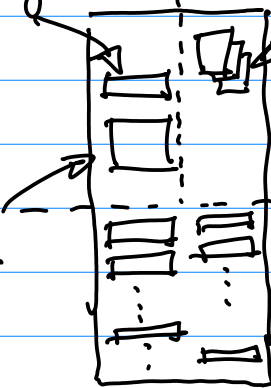


Fig 1.

Optimization Not Only on the Hardware
But On the Compiler Design, and
System Software Design.

MIPS, ARM (most widely Adopted)
RISC

ARM Architecture — Common Core

Base Line Hardware (Datasheet); ARM11

Fig 1. CPU Architecture
ARM Cortex M3

Note: To Be Able to Draw/
Design CPU Architecture
Either this OR Your Target
platform. (1769 + ARM11)
Base Line

memory map.

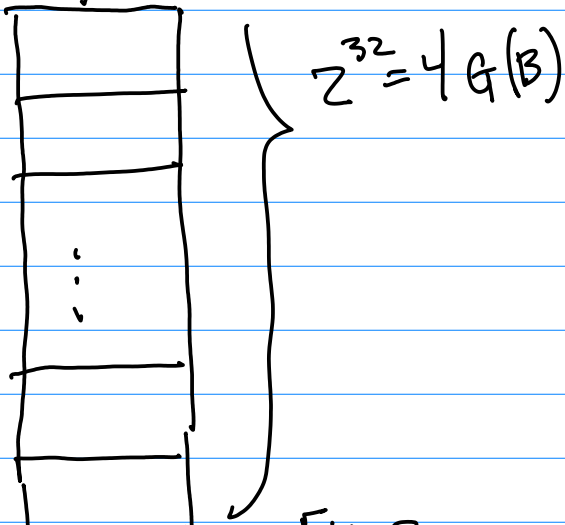


Fig. 2

(1) 32 Bit RISC Architecture

$$2^{32} = 2^{10} \cdot 2^{10} \cdot 2^{10} \cdot 2^2$$

$$= \underbrace{1K \cdot 1K \cdot 1K}_{1M} \cdot 2^2$$

$$= 4 \text{ GB}^1 \text{ (Byte)}$$

(2) Byte Addressable Machine
 ~ whose minimum memory cell with an unique address is a Single Byte

(3) Memory Banks, 8 BANKS

Size of Each Bank: $4 \text{ GB} / 8$

$$= 2^{32} / 2^3 = 2^{32-3} = 2^{29} = 2^9 \cdot 2^{20}$$

$$= 512 \text{ MB}$$

Starting Address of Each Bank
 Question: How many Bits needed to define the Starting Address of Each Bank? 3 bits, $2^3 = 8$

3 bits Needed

$$a_{31} a_{30} a_{29} : a_{28} \dots a_1 a_0$$

Little Endian

ARM CPU Can be configured at Boot Stage as either "Little Endian" or "Big Endian".

Find Starting Address for BANK the 1st

$$a_{29} = a_{30} = a_{31} = 0$$

a_{28} has to be added, to form a Hex

0x0000-0000

2nd Bank's Starting Address

$$a_{31} a_{30} a_{29} : a_{28}$$

$$0 \quad 0 \quad 1 : 0$$

0x2000-0000

3rd Bank's Starting Address

$$a_{31} a_{30} a_{29} : a_{28}$$

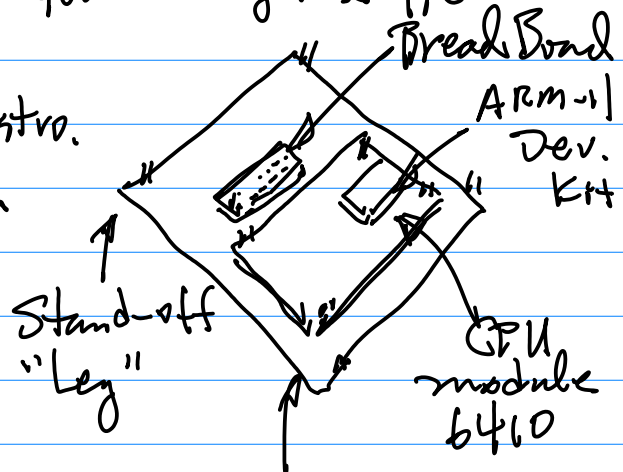
$$0 \quad 1 \quad 0 : 0$$

0x4000-0000

Now, Consider target Board
Conditions to qualify the selection

- ① ARM Based ; ② UNIX-Like O.S.
 - ③ Establish eco-system
Developer Base (~millions)
 - ④ Technology Innovators / Leaders.
 - ⑤ External Expansion Connectors
- Linux kernel
Image
Source Distro.
Tool chain

Plus: many examples
on Device Drivers (I2C,
PWM, SPI, UART, ...)



Feb 3rd (Wed) CMPE242

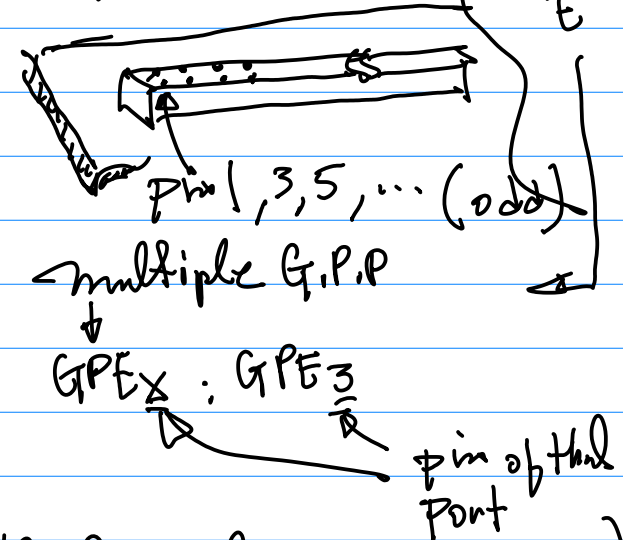
Note: 1° Submission of Honest Pledge
Form (Signed), CANVAS,
By Sat 11:59 PM; EE242 Submission
to e-mail;

Wire wrapping
Note: Bread Limitation -
Run High Speed
Cannot ~20 MHz

Today's Topics: 1° CPU Architecture
2° Target Board Selection - Bill of
material
Ref: github

CPU 1.5 GHz
Connector 1
General purpose (port)
Pin Number
Physical Location
"E"

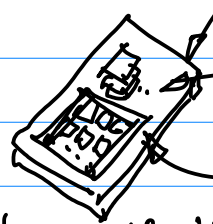
- 1-2020S - let A1 - hardware board...
- ARM-1 ① Compiled w/ Linux Open Source
Distro / Kernel Sources
ARM tool chain
 - ② Datasheet Baseline Reference
Requirements, Exams



DrawBack: Lack of the Ability to
handle Edge AI ; (G.E.) Tiny6410
Kit from Friendly ARM. Com, ~\$90

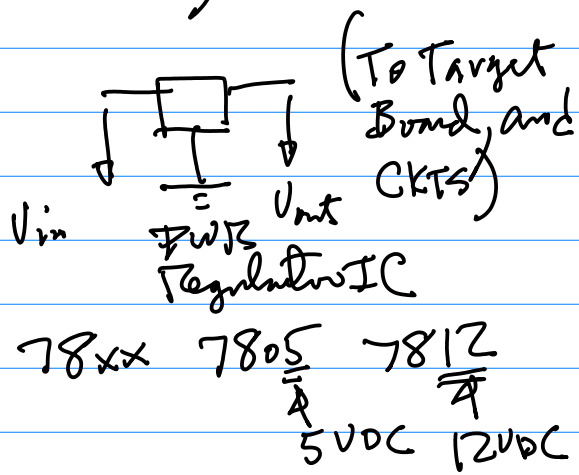
Option (Pie Board 3B+, 4)
675 Pie-4, 8GB mem.

OMP242
 NVDA (Nvidia) NANO — 128 GPU
 GTX2 System on module
 6 CPU ARM A57?
 256 GPU



2° 4 Stand-offs (Legs)
 3° Components to Build
 PWR CKT & CKT for
 I/O Testing ("Hello, the
 world").

Note: 1° NANO Expansion Connector, Yes
 (Limited I/O Function)
 Compare to ARM II — SPI, I2C, PWM
 2° Kernel Source Distribution, Yes
 To Become a developer, to Sign up
 ~\$59

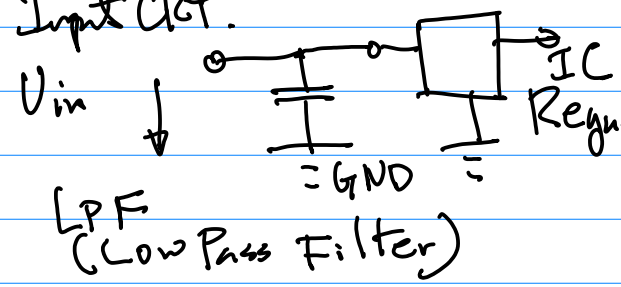


Option (NVDA GTX2 System on module)
 1° Expensive! \$299 + Carrier Board
 \$150 — 3v3t
 Edge on AI
 If this is selected, then 2~4 person
 to share the Cost

Red LED, 4~10mA
 Resistors. $V_{CC} = 5V$
 $I = 4 \sim 10mA$
 $R I = V$
 $R = V/I = 5 / 4 \times 10^{-3}$
 $= 1.25 K\Omega$

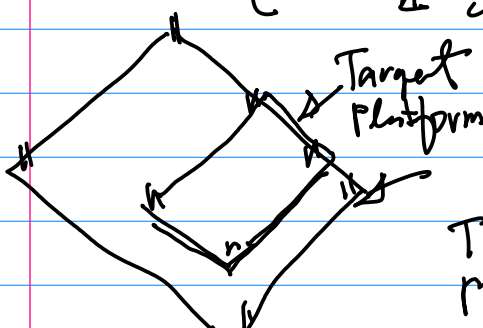
Note: 1° 4-Person Team By Next Week;
 2° Homework/Project has to
 Individual, Each person has
 your own Board;

Cap for your External Power
 Input Ckt.



$C = 4.7 \mu F$
 $T = RC \rightarrow 3dB$
 4° Toggle s/w

Bill of materials:
 Phase I & II { Phase I — HW1 — "Hello,""
 " II Sensors/Stepper motor Drive



1° Wirewrapping Board
 Through-Holes w/ metal plating (Not the entire side)

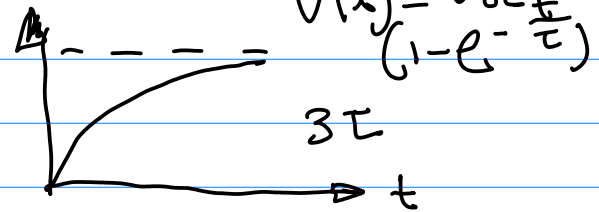
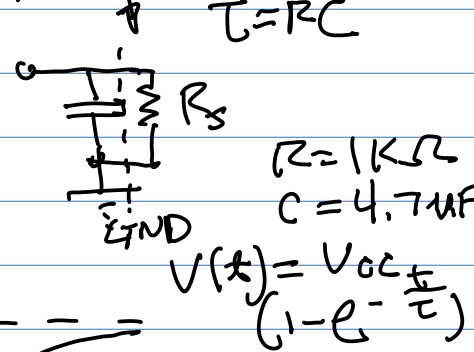
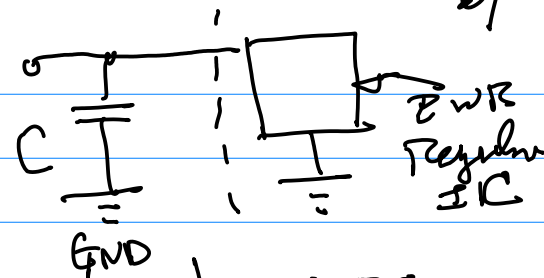
Feb 8, (Monday) CMPE242
HARRY LI

Today's Topics: 1° Bill of Material

2° Prepare for the 1st Assignment

"Hello, the world". 3° CPU Architecture

Bill of material { Phase I: Target platform
"Hello, the world"
Phase II: Sensors / Drive
Stepper motor /
OP Amps



Caps for 7805 → Datasheet

Polarity "⌋"

74KK NAND, NOR,

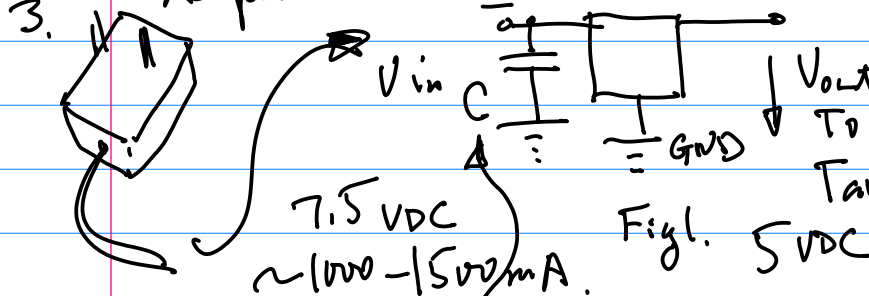
LED, Red, yellow/green
4~10mA

1. Target: ① Pic 3 Bt, Pic 4
② NA10, TX2 (Edge AI)
③ ARMII — Base Line Reference platform

2. Prototype Board, 6" x 6" I/O CKT
Sensors { Analog
Drive { Digital

POWER CKT: PWR Regulator IC

Wall mount Adapter LM7805 2VDC dropoff



Note: you may need higher voltage source to Stepper motor Drive.
LM7812?

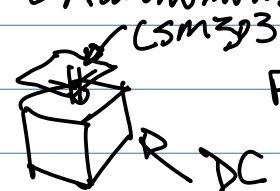
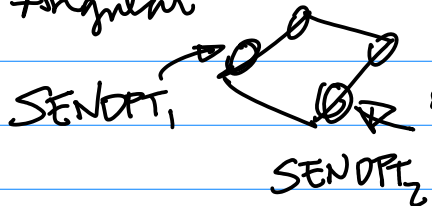
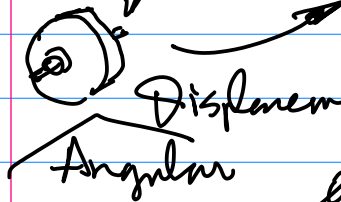
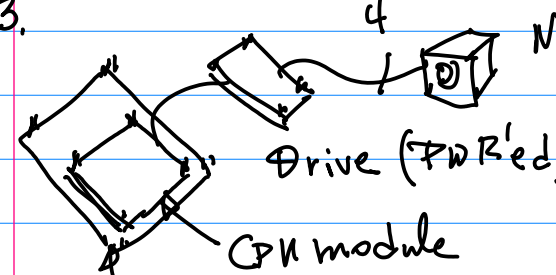
4. "Glue" Logic { Assembled ICs 200 ~ 5KΩ
Caps 7805 10KΩ
PWR Input Node
LPF (Low Pass Filter)

Connectors { External PWR
Switch (Toggle Switch)
To Control PWR Source;
To Build I/O Testing

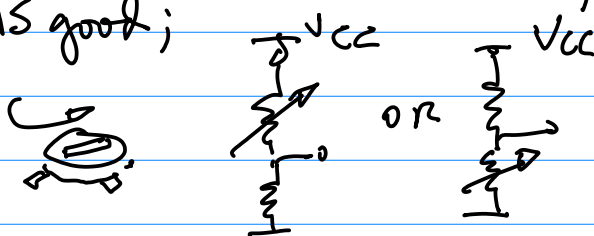
Phase II (Bill of Material):

Ref: git hub:

1. LSM303 (I2C) → Autonomous Robot

2. Optical Encoder (optional)  Fig 2 Front wheel Drive3.  NEMA14, 17

Prototype

b Bulk Converter
= is good;A 12VDC Commonly
= need some, Check
Current Need;Consider Building "Hello, the world"
Test CKT

Objectives

1. Bring up the target Board
and Print "your Name,
Last 4 Digits Student ID"2^o Test GPIO Interface2.1 Send Logic "1"
to Turn ON On-Board
LED, then flash @
1 Hz Frequency2.2 Send Logic "0" via
gpio port to turn off
on-Board LED;2.3 Read input via GPIO
port, if the input is
"1", then Send Command
to Turn ON On-Board LED2.4. Read Inputs via GPIO
Port, if the input is "0"
then, Send Command to
Turn off the on-Board LED.Discussion ON CPU Architecture
then ON this Assignment.

Action 1: Form 4 person

Team, Send me email
First, Last Name, SID

Feb 10.

Homework 1 Due Feb 24 (W) 11:59pm.

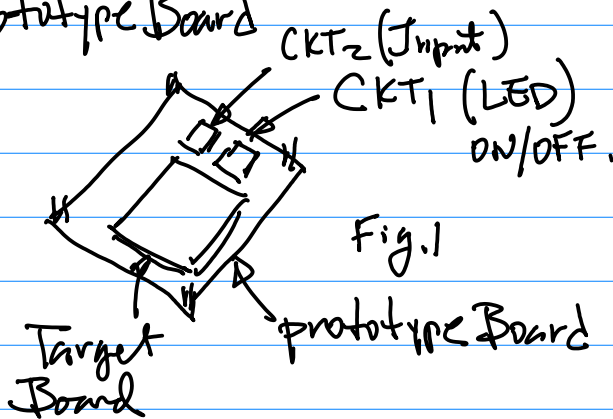
CMPE/EE242 CANVAS

EE242 Submission to E-mail:

Target Platforms: ARM11^a Architecture^b O.S.^c Driver
 Base Line: Pic ✓ (not mmh) ✓ NM ✓ NM ✓ NM
 NUDA ✓
 NANO, TX2 ✓
 $V_{GPP} = IR + V_{LED} \dots (1)$
 CMOS $V_{GPP} = 3.3 \text{ VDC}$
 $I \approx 10 \text{ mA}$
 $V_{LED} \approx 1.2 \text{ VDC}$
 $3.3 = 10 \times 10^{-3} R + 1.2$
 $R = (3.3 - 1.2) / 10^{-2}$
 $= 210 \Omega$

1° Target platform Built on

Prototype Board



Note: Find the pin(s) from Target platform.

Pic3 GPIO14 — Pin 8

Consider Input Testing CKT

Input pin { Read "1", PWR
 GPP { Read "0", GND
 with toggle switch

Ref: 1-lect- Hardware Board-

Print "Hello, the world", Drive LED ON/OFF

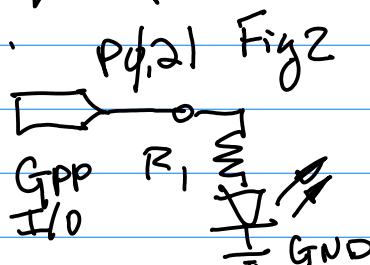
C/C++ or Python Has to be the LED ON your prototype

2° GPIO Testing
 GPP Output CKT₁ { "1" Turn ON LED
 "0" Turn OFF LED

Design:

Identify the GPIO pins for Input and Output Testing CKT.

Pd.3 GPP Input pin
 Pd.21 GPP Output pin



$$R1: 3.3 / 10 \text{ mA} = R1$$

$$3.3 / 10 \times 10^{-3} = 330 \Omega$$

$$R2: \frac{3.3 (\text{from GPP})}{10 \times 10^{-3}} = 330 \Omega$$

NXP
 LPC
 1769

System (Architecture) & Software Design. Boot Loader } 0.5.
I/O Function

3° Source code, Binary \rightarrow Zip.

4° One page Report IEEE paper format (Template is given on-line) github

5° Form 4-Person Team. Submit First, Last Name, 4 Digits SID E-mail Contact Information Indicate Coordinator of the Group By Thursday.

Note: All work has to be individual

6° Short Video Clip (5-10 seconds) Shows the Prototype Board and Screen Capture.

Feb 15, Monday

Topics: GPP I/O, Device Driver.

github: 2-2020S-lec2...

Example: 1) CPU Broadcom BCM2835

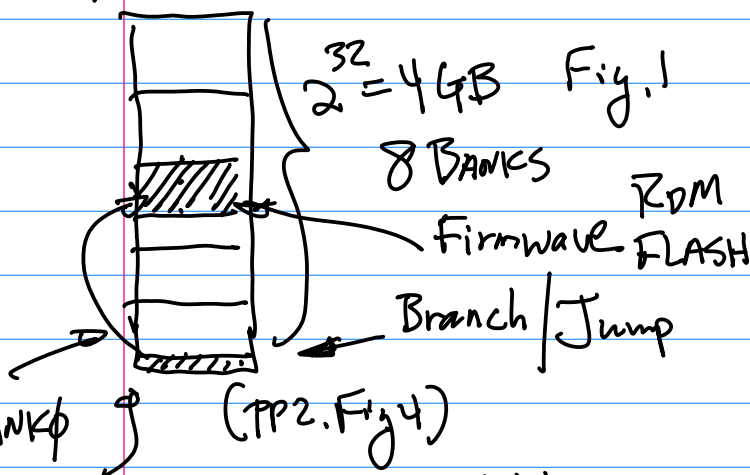
2) PWR1, (3P3), 2, 4 (5VDC)

3) GPIO for Homework \rightarrow Chose pin those not marked w/ other function

Concept of Device Driver

Architecture + memory map
Software { Kernel (OS)
Device Drivers

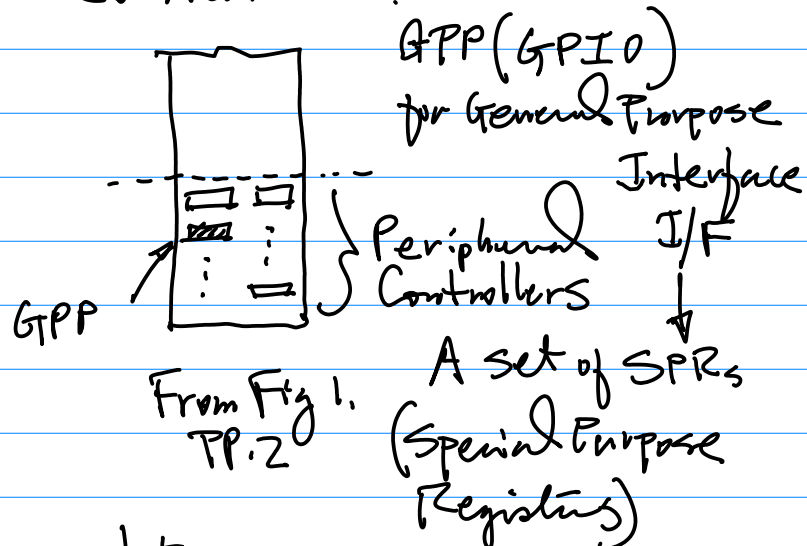
Architecture



0x0000_0000 PWR-up Address

Addr. when CPU is powered up, it will fetch the 1st Instruction from this addr.

GPU Architecture



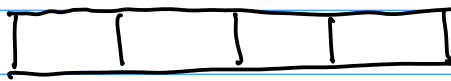
Note:

1° SPRs are 32 bits.

2° SPRs' Functions into 3 Categories

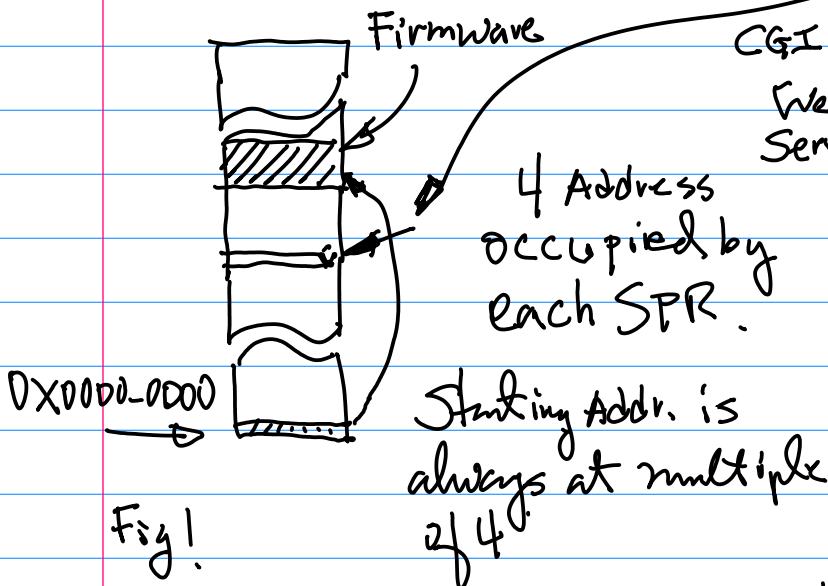
- a Control Function, Init & Config
- b Data
- c Pull up/Down

3° Map 32bit SPR onto ^{The} memory



32 bit SPR

which is mapped to the memory location



1° Power-up Address: ~ when CPU is powered up, it will fetch the 1st Instruction from this location
~ /cmpe242/2018S-29-CPU (ARM11 CPU Data Sheet)

Feb 17 (W)

Ref: CPU Data sheets

1° github ~ 2018S-29

2° Boot up Sequence

Firmware: ROM/FLASH { Boot loader I/F

3° OS Image is Being Loaded
Then USER Space program can be executed, And the Device(s) can be accessed via Device Driver in Kernel space.

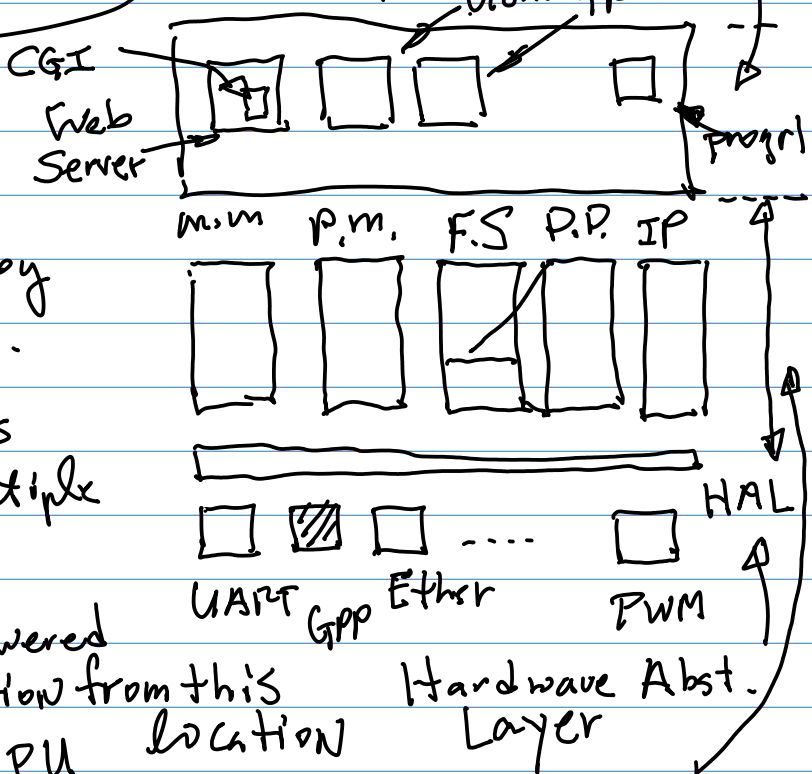
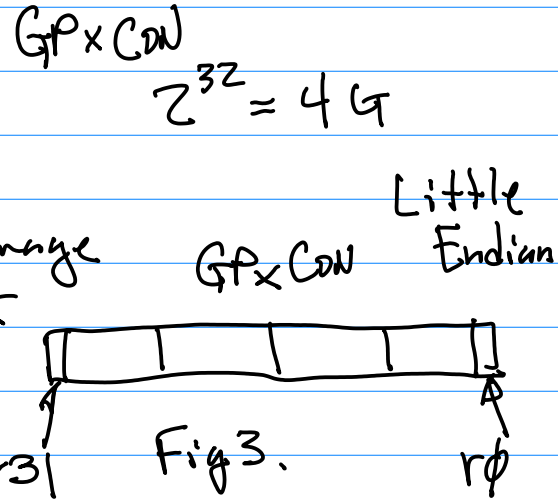


Fig 2. Kernel Space
Example: Prog1.C → USER Space

UserSpace \rightarrow `open("dev", ...);`
 $fd =$
 \downarrow
`read(fd, Buffer, size)`
 \downarrow
`write(fd, Buffer, size)`

path Device Driver location in the Kernel Image for GPP Input Testing



4. Kernel Space:

OS: manage ~ Resources & Policy

Device Driver(s): A collection of Program(s), manage/manipulate

Special Purpose Registers, to be able to utilize peripheral controller(s)

Init & Config of Special Purpose Registers

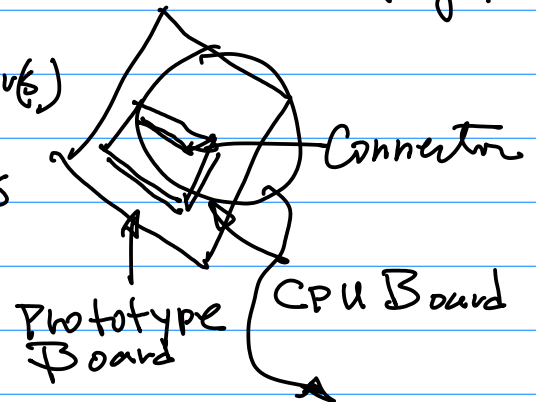
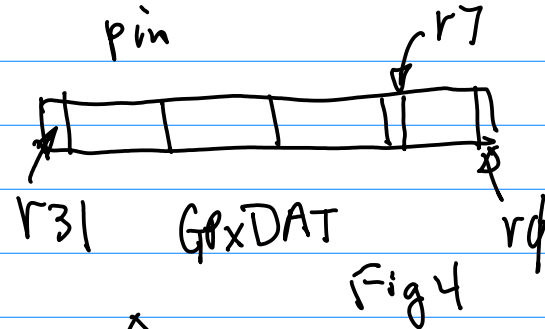
Example: Naming Convention for SPRs:

10 For Control Register

Prefix (3) + Root (3) + Postscript (3)

GPx CON

Test GPP Output GPx pin 8 as an output pin



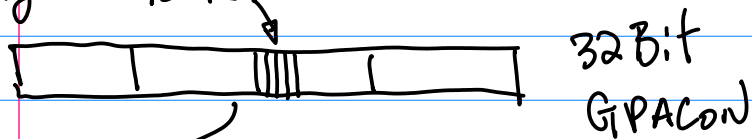
Make pin 8 as an output pin. \rightarrow Set GPxCON pin 8 to 1

CPU GPxDAT pin 8

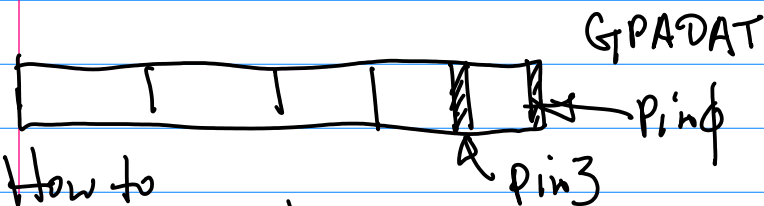
Fig 5

Question: How many control functions can one Control Register realize? $GPxCON[7] = 1$

GPACON[3:0] = 0000 Input
 Fig 1. 15:12 0001 Output



Question: Define Pin3 GPP Input.
 (Pin0, 1, 2, 3...)



How to
 Perform Init & Config?

First GPACON \Rightarrow GPA3 GPACON[15:12]

\downarrow
 GPACON[15:12] = 0000 for Input

GPACON[15:12] = 0001 for Output

Suppose that is task (Init & Config)

Question: Find Binary Pattern to
 Define GPACON?

0X00001000 ✓

Move/Copy this Binary Pattern
 to 0X7F00_8000

Note: 1° All GPADAT Registers
 for GPA ~ GPR

2° To Be Able to Define Input/
 Output Based on Table Look up.

3° To Be Able to Generate Binary Pattern for Init & Config.

Action 2: Read CPU Datasheet
 ARM11 (Samsung)

TX2 (Nvidia)

Pic 3/4 (BroadCom)

Note: Programming Kernel
 Source/Drivers.

4° make menuconfig

5° Kconf Script

Note: To Be Able to write
 Simple Script.

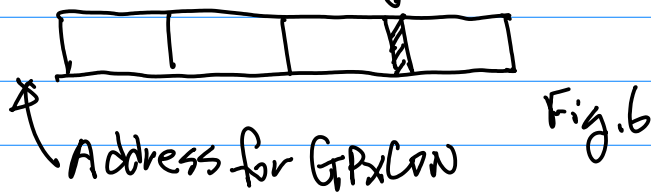
Config	MODULE NAME
tristate	(1) Define 3 options to build
	(2) followed by "...." Verboage
depends on	CPU_ <u>manufacturer ID</u> + Device ID
help	Text Info

Example of C Code Driver.

printf(
)
 Kernel Space

CmpE242

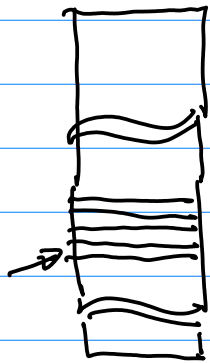
From Fig.3 (pp 11) $GPxCON[7] = 1$ ¹² Kernel Space Driver Development



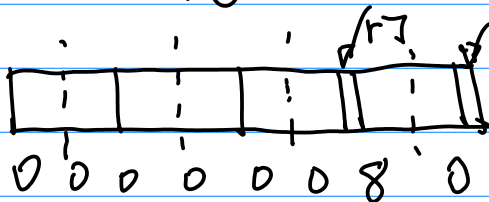
Suppose Addr. = 0X4008_0000

Find the BANK which holds this SPR, addr, GPx Controller

Note: GPxCON occupies 4 Bytes



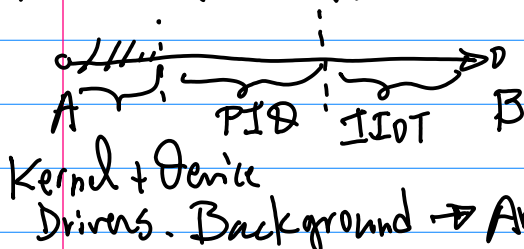
Init & Config Pattern (Binary)



0X80 → Define

Homework ON CANVAS & github

Feb 22nd (Monday)



Architecture → IDE → Implementation

Kernel DD Section 10.5.1
GPP I/O Testing verify GPACON[3:0]

Hands-On Experience.
Action 1
Kernel Source + Tool Chain
Distribution ARM
250MB plus
Document
CPU

2^o Perform Init & Configuration
Define the Behavior of
of Peripheral SPRs
Controller.

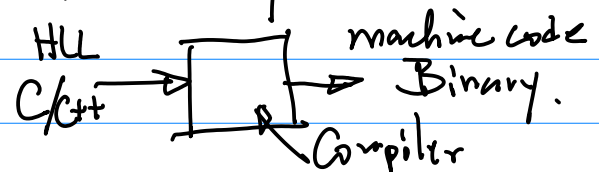
Example: Samsung CPU Arm-v1
2018S-29-~ (Datasheet)
Chapter 10, GPA-GPQ Table
Look up

Section 10.4.1.

GPACON's Address.

0X7F00_8000
ptr move/copy data
4 Bytes into

this memory location



Architecture → IDE → Implementation

Kernel DD Section 10.5.1
GPP I/O Testing verify GPACON[3:0]