Jan 27, 21

Welcom to

CMPE242   Harry Li

Embedded Hardware Systems

1. Green Sheet   github/hualili/cmpe242

E-mail: hua.li@sjsu.edu

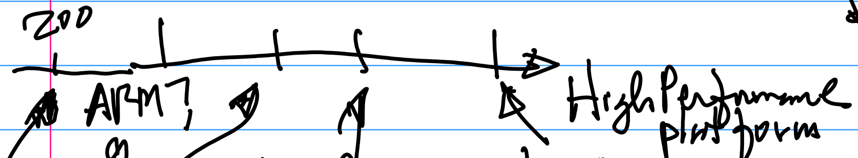
(650) 400-1116 Text message

2. Pre-requist Requirments 180A & D

Course Description

Hands-ON.

Target Development platform



NXP
LPC
17XX

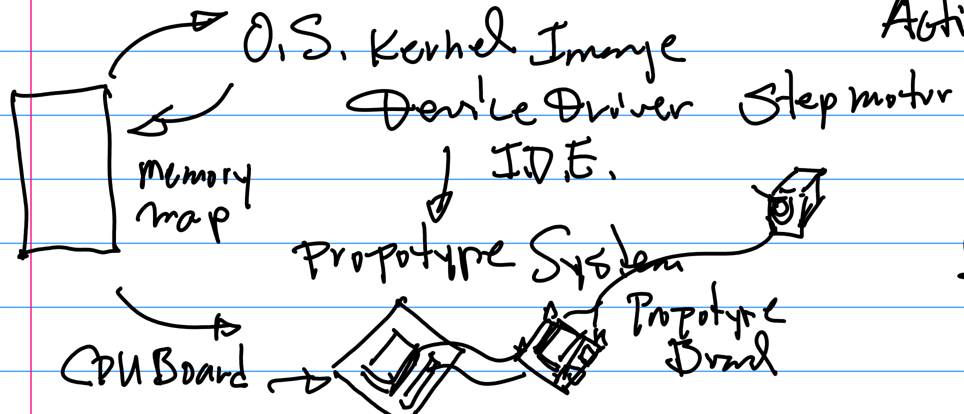
Samsung
ARM11
~10 yrs
800mHz
Linux
Kernel O.S.
Open Source

Broadcom
Pie 3B+
Pie 4
~$60
Edge AI

NVDA
(1) NANO ~$60
(2) TX2 ~$400
+ $150~$350
(3) TX1 ~$399

<— Basic Material

Scope of the Course { Device Driver ~~Dep~~ Development { C/C++, Python
O.S. Kernel



Before: 3 Labs/Projects 1/

Device Driver
Sensor LSM 303 → Feed Back Loop
P.I.D Controller
Convolution
to
Integration
(HDL)

I/F to Target Board
↓
Human Readable file.

Optional subjects

1° RISCV Project — FPGA

Device Driver Development <— O.S. Kernel Image

2° ROS (Robotics Operating System)
platform / Visualization Tool.

* Grading Policy
3 - 3 - 4 <— Final
Mid Projects

* CANVAS — EE242
{ (1) Assignments (NO)
(2) Collection / Submission of your class work

Action 1. github/hualili/cmpe242

2. Datasheet
{ Samsung ARM11 Datasheet
{ NXP LPC 1769 DataSheet

Architecture NXP LPC 17xx

CmpE242 2/1

Action 3. Target Platform Selection
a) Unix-Like OS. b) Edge AI Computing (Scalability) → GPU

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

CmpE242 Feb1. 2021

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

System Architecture

NVDA CPU/GPU platform { NANO TX1, TX2 }
BroadCom, Pi 3/4 G.E. (Graphics Engine)
Samsung ARM11 (9,7)

Optional Architecture RISC-V
↓
Common Characteristics RISC
Reduced Instruction Set Computer
↓ 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

Base Line Software { O.S. Kernel Aspects
Device Drivers.
↓
Linux Distribution Optimized for Embedded platform

Example: Datasheet
LPC1769 ARM Cortex M3
↓
Samsung ARM11
↓
NVDA ... Pi e.x

CPU Datasheet, LPC1769
20185-3-UM10360, P.P.9

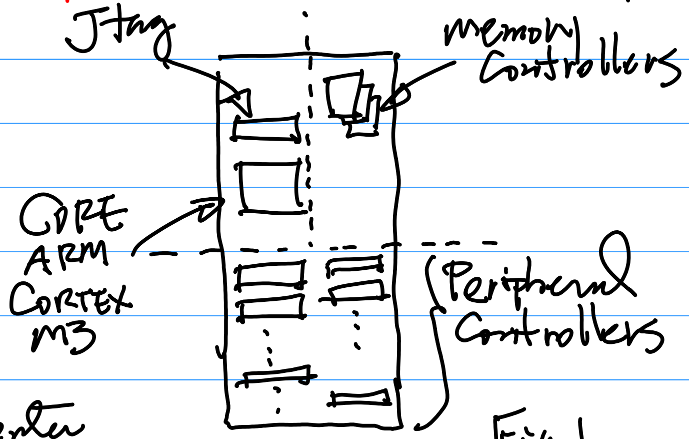


Fig 1.

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

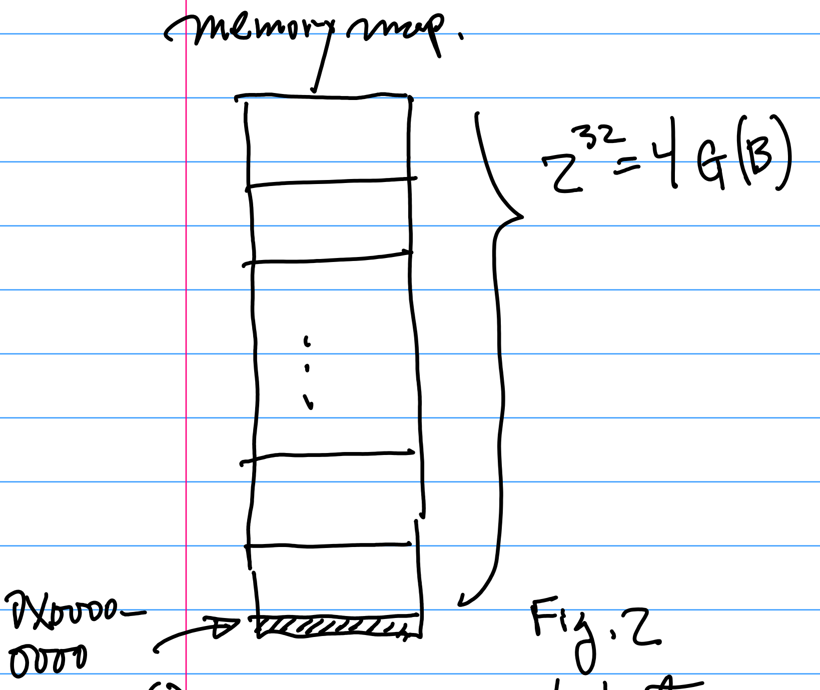


Fig. 2

① 32Bit RISC Architeture

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

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

$$= 4GB$$ (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: 4GB/8

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

$$= 512\,MB$$

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

$$\underbrace{a_{31}\, a_{30}\, a_{29}} \vdots\, a_{28} \cdots a_1 a_0$$

Little Endian

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

Find Starting Address for the 1st BANK:

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

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

0X0000_0000

2nd Bank's Starting Address

$$\underbrace{\begin{array}{cccc} a_{31} & a_{30} & a_{29} \vdots & a_{28} \\ 0 & 0 & 1 \vdots & 0 \end{array}}$$

0x2000_0000

3rd Banks Starting Address

$$\underbrace{\begin{array}{cccc} a_{31} & a_{30} & a_{29} \vdots & a_{28} \\ 0 & 1 & 0 & 0 \end{array}}$$

0 x 4000_0000

Now, Consider target Board Conditions to qualify the selection

① ARM Based ; ② UNIX-Like O.S. Linux { Kernel Image, Source Distro., Tool Chain

③ Establish eco-system Developer Base (~millions)

④ Technology Innovators / Leaders.

⑤ External Expansion Connectors

Feb 3rd (Wed) CMPE242

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

Today's Topics: 1° CPU Architecture

2° Target Board Selection — Bill of material

Ref: github

1-2020S-lect1_hardware board...

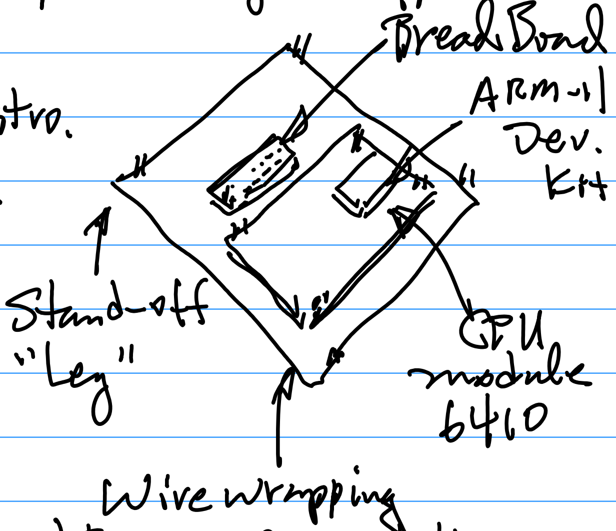
ARM-11 ① Coupled w/ Linux Open Source Distro, { Kernel Sources, ARM tool Chain

② Datasheet Baseline Reference Requirements, Exams

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

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

Good learning Tool 4/5



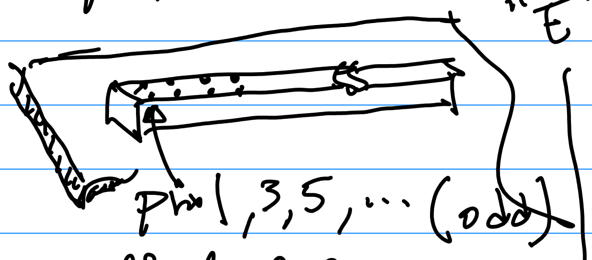
Note: Bread Limitation — Run High Speed ~20.MHz
Cannot

CON1,5 GPEx

Connector 1

General purpose (Port)

Pin Number
physical Localition



multiple G.P.P

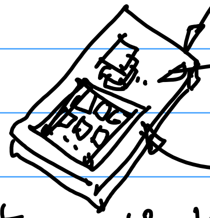
GPEx : GPE3

pin of that Port

Option (Pie Board 3B+, 4) $75 Pie-4, 8GB mem.

CMPE242

NVDA (Nvidia) { NANO — 128 GPU
TX2 System on module



6 CPU ARM A57?

256 GPU

Note: 1° NANO Expansion Connector, Yes
(Limited I/O Function)
Compare to ARM11 ← SPI ✓, I2C ?, PWM

2° Kernel Source Distribution, Yes
To Become a developer, to sign up
~$59

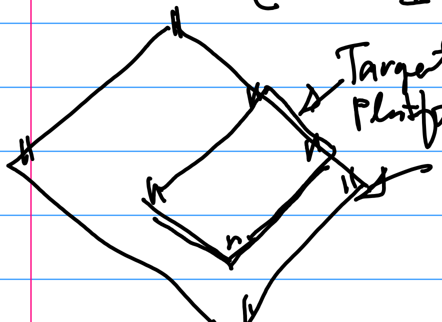
Option (NVDA TX2 System on Module)
1° Expensive! $299 + Carrier Board
$150 – 300+

Edge on AI

If this is seleted, then { 2~4 person to Share the Cost

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

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

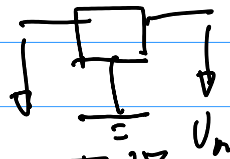


Target Platform

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

2° 4 Stand-offs ("Legs")

3° Componments to Build PWR CKT & CKT for I/O Testing ("Hello, the world").

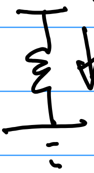


(To Target Board, and CKTs)

78xx 7805 7812
5VDC 12VDC

Red LED, 4~10mA

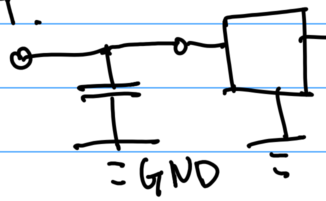
Resistors.



$RI = V$

$R = V/I = 5/4\times10^{-3}$
$= 1.25\,k\Omega$

Cap for your External POWER Input Ckt.



LPF (Low Pass Filter)

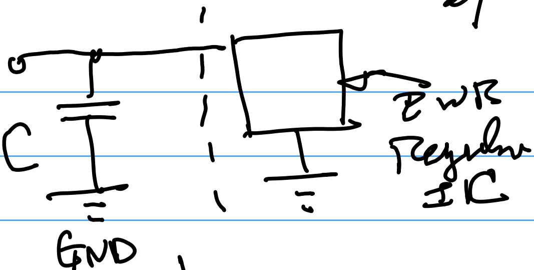
$C = 4.7\,\mu F$

$\tau = RC \rightarrow$
3dB

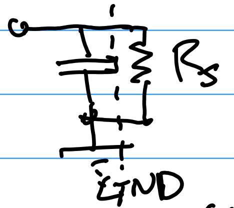
4° Toggle S/W

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

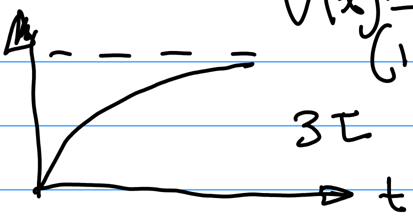


τ=RC



R=1KΩ
C=4.7uF

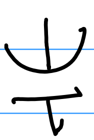
$V(t) = V_{DC}(1-e^{-\frac{t}{\tau}})$



Bill of material { Phase I: Target platform "Hello, the world"
Phase II: Sensors | Drive | Stepper motor | OpAmps

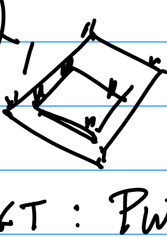
1. Target: ① Pie3B+, Pie4
② NANO, TX2 (Edge AI)
③ ARM11 — Base Line Reference platform

Caps for 7805 → Datasheet
Polarity "



"

2. Prototype Board, 6"x4"

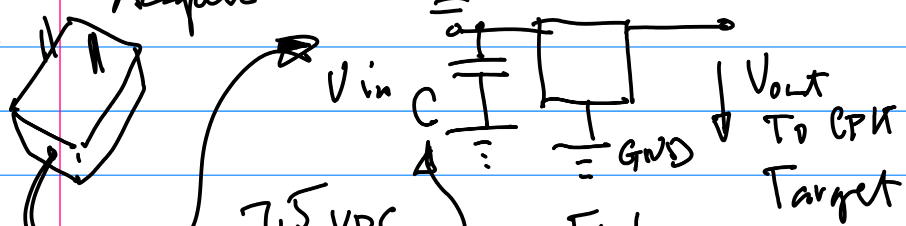


I/O CKT
Sensors { Analog, Digital }
Drive

74xx NAND, NOR,
LED, Red, yellow, green 4~10mA

POWER CKT : PWR Regulator IC

3. Wall mount Adapter LM7805 2VDC dropoff



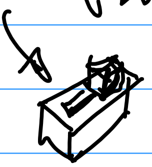
7.5 VDC ~1000-1500mA. Fig 1. 5VDC

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

Connectors { External PWR

4. "Glue" Logic { Assorted Resistors 200~5KΩ 10KΩ
Caps { 7805 { PWR Input Node
LPF (Low Pass Filter)
② PWR Distribution Node

Switch (Toggle Switch)



To Control PWR Source; To Build I/O Testing

Phase II (Bill of material):
Ref: github:

1. LSM303(I2C) → Autonomous Robot

2. optical Encoder (optional)

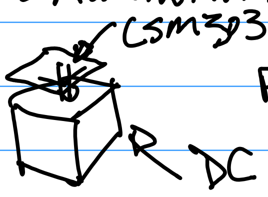
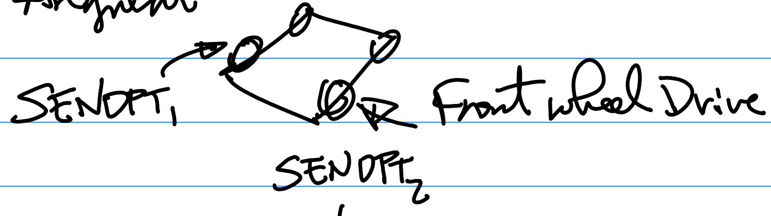
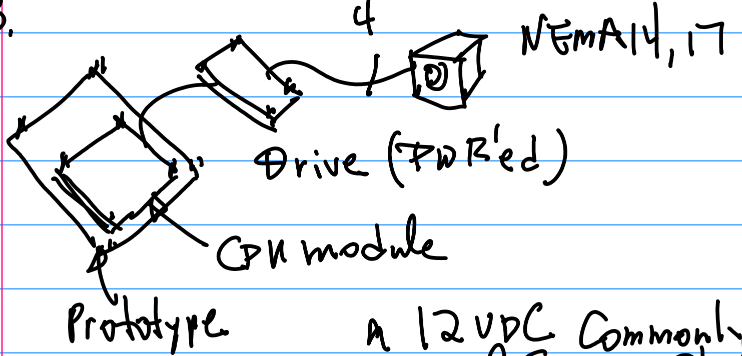


Fig2

Displacement measurement
Angular

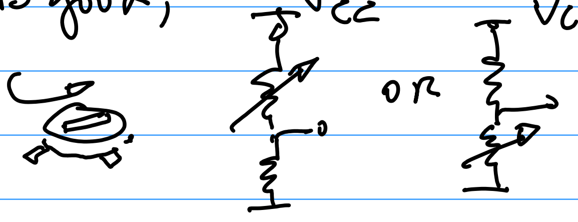


3.



a 12VDC Commonly used Some, Check Current Need;

b Buck Converter is good;



Consider Building "Hello, the World" Test CKT

Objectives 1: Bring up the target Board and Print "your Name, Last 4 Digits Student ID"

2° Test GPIO Interface

2.1 Send Logic "1" to Turn ON On-Board LED, then flash @ 1 Hz Frequeny

2.2 Send Logic "0" via gpio port to turn off OnBoard LED;

2.3 Read input via GPIO port, if the input is "1", then Send Command to Turn ON On-Board LED

2.4. Read Input via GPIO Port, if the input is "0" then, Send command to Turn off the onBoard LED.

Discussion ON CPU Architeture then ON this Assignment.

Action 1: Form 4 person Team, Send me email First, Last Name, SID

Feb10.

Homework1 Due Feb24(W) 11:59pm.

CmpE/EE242 CANVAS

EE242 Submission to E-mail:

Target Platforms	ARM11^a Architecture^b	O.S.^{c} Kernel	Driver
BaseLine → Pie	✓ (Not much)	✓NM	✓NM
NVDA NANO, TX2	✓	✓	✓

1° Target platform Built on Prototype Board

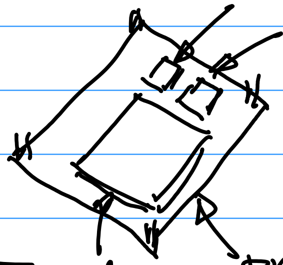


Fig.1

Ref: 1-lec1-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.

P0.3 GPP Input pin

P0.21 GPP Output Pin

NXP LPC 1769

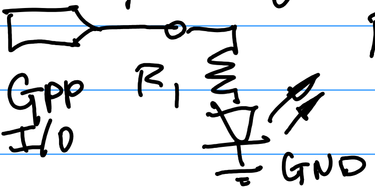


Fig2

$V_{GPP} = IR + V_{LED}$ ... (1)

CMOS $V_{GPP} = 3.3VDC$

$I \simeq 10mA$

$V_{LED} \simeq 1.2VDC$

$3.3 = 10\times10^{-3}R + 1.2$

$R = 3.3 - 1.2 / 10^{-2}$

$= 210\Omega$

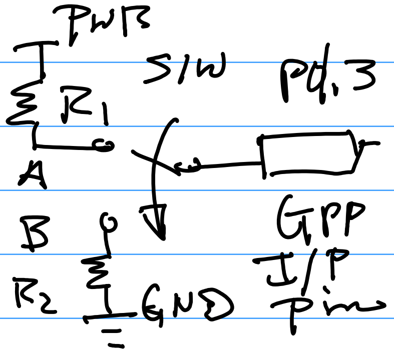
Note: Find the pin(s) from Target platform.

Pie3 GPIO14 — Pin8

Consider Input Testing CKT

Input Pin GPP { Read "1", PWR; Read "0", GND

with toggle switch



R1: $3.3/10mA = R_1$

$3.3/10\times10^{-3} = 330\Omega$

R2: $\frac{3.3 \text{(from GPP)}}{10\times10^{-3}} = 330\Omega$

CmpE242

System (Architecture) & Software Design. Boot Loader { O.S. , I/O Function

3° Source code, Binary → Zip.

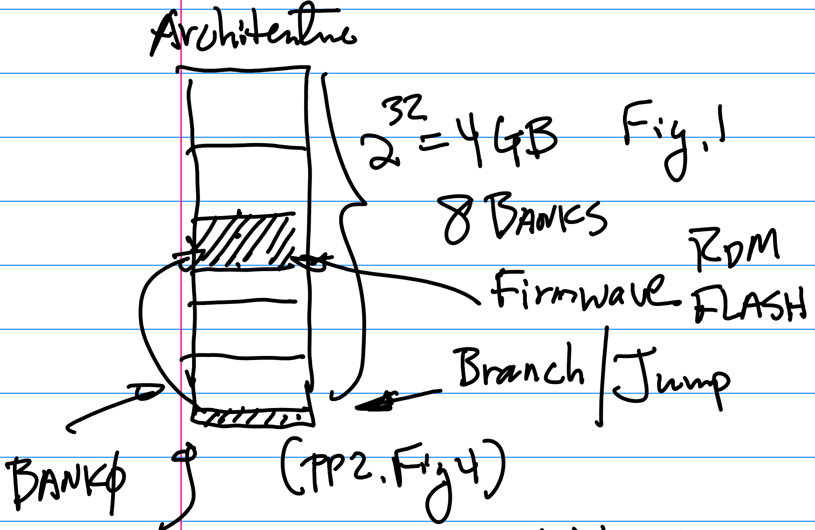
4° One page Report, IEEE paper format (Template is given on-Line) github

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.

Architecture



0X0000_0000 PWR-up Address

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

Feb 15, Monday

Topics: GPP I/O, Device Driver.

github: 2-2020S-lec2 ...

Example: 1) CPU Broadcom BCM2835

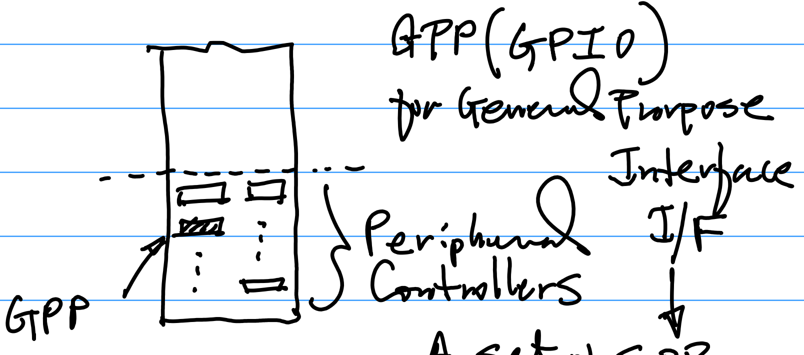
2) PWR 1, (3P3), 2, 4 (5VDC)

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

Concept of Device Driver

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

CPU Architecture.



From Fig 1. PP. 2

A set of SPRs (Special Purpose Registers)

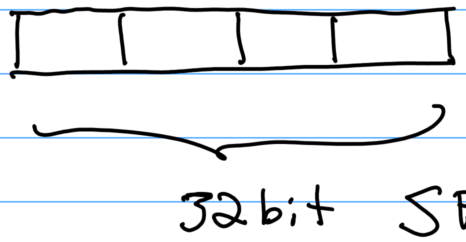
Note:

1° SPRs are 32 bits.

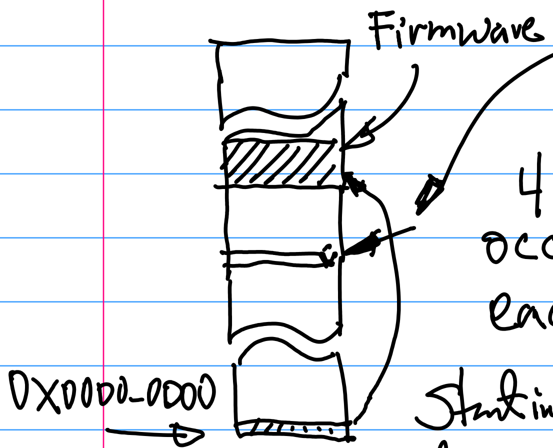
2° SPRs' Functions into 3 Categories

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

3° Map 32bit SPR onto ~~A~~ The memory



which is mapped to the memory location



4 Address occupied by each SPR.

Starting Addr. is always at multiple of 4

Fig 1

1° PWR-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)

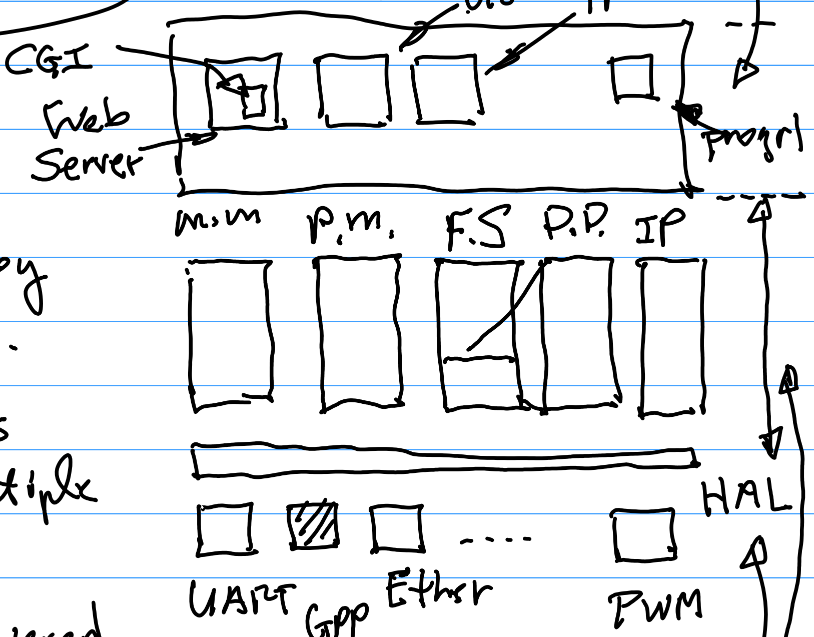
Ref: CPU Datasheets

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



Hardware Abst. Layer

Fig 2.

Kernel Space

Example: Prog1.c → USER Space

CMPE242

userSpace → fd = open("/dev/...");

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

read (fd, Buffer, Size)

write (fd, Buffer, size)

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 peripherial controller(s)

Init & Config of Special Purpose Registers

Example: Naming Convention for SPRs:

1. For Control Register

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

GPx CON

Question: How many control functions can one Control Register realize?

GPxCON

$2^{32} = 4G$

Little Endian

GPxCON

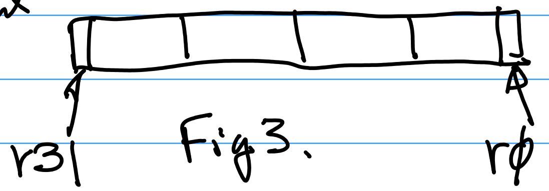


Fig3.

Test GPP Output GPx pin8 as an output pin

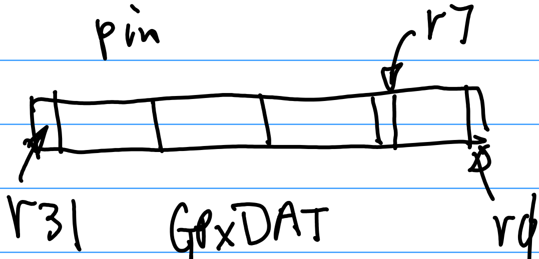


Fig4

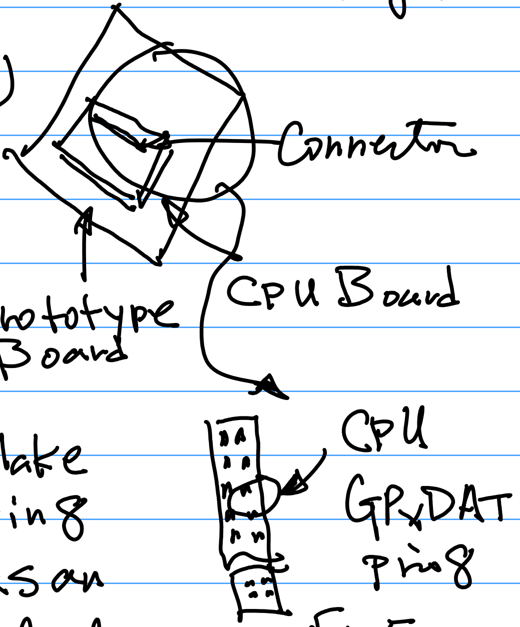


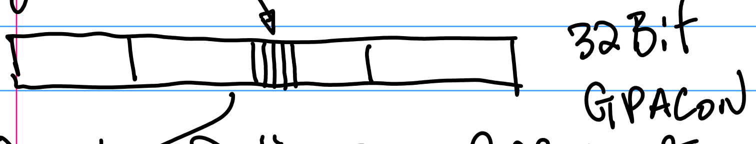
Fig5

Make pin8 as an output pin. → Set GPxCON pin8 to 1

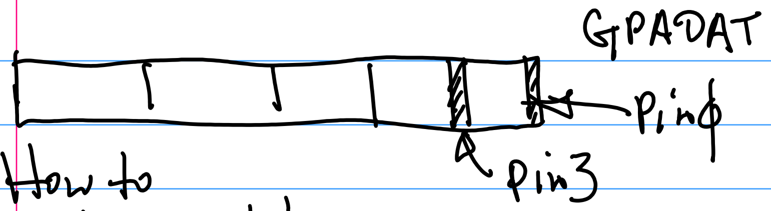
GPxCON[7] = 1

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 → GPA3 GPACON[15:12]

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 ?

0X0000_1000 ✓

move/Copy this Binary Pattern to 0X7F00_8000

Note: 1° All CON/DAT 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)
Pie 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 "...." Verbage
depends on	CPU_manufactor ID + Device ID
help	Text Info

Example of C code Driver.

printk( )
↑
Kernel Space

CmpE242

From Fig.3 (pp 11) GPxCON[7]=1

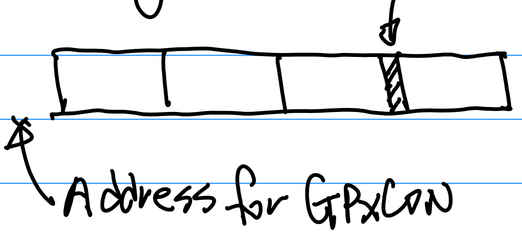
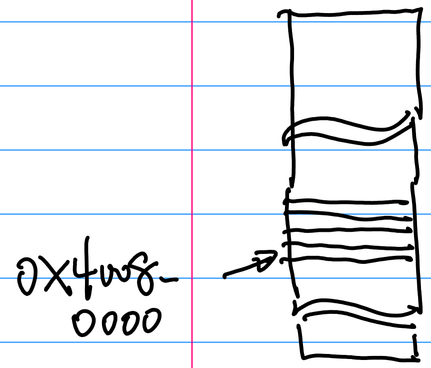


Fig.6

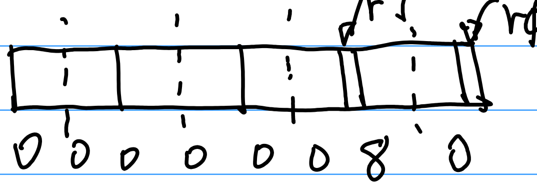
Suppose Addr. = 0X4008_0000

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

Note: GPxCON occupies 4 Bytes



Init & Config Pattern (Binary)

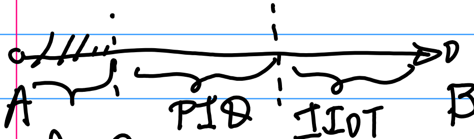


GPxCON[7]=1

0X80 → Define

HomeworK ON CANVAS & github

Feb 22nd (Monday)



Kernel + Device Drivers. Background → Architecture + mem.map

12

1° Kernel Space Driver Development

Hands-ON Experience.

Action 1

Kernel Source Distribution 250mB + Tool Chain ARM plus Document CPU

2° Perform Init & Configuration of SPRs

Define the Behavior of Peripheral Controller.

Example 1: Samsung CPU Arm-11

2018S-29-~ (Datasheet)

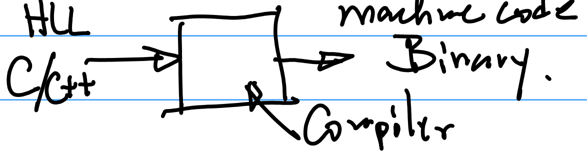
Chapter 10, GPA-GPQ Table Lookup

Section 10.4.1.

GPACON's Address.

0X7FD0_8000

ptr move/Copy Data 4 Bytes into this memory location



Architecture → IDE Kernel DD → Implementation

GPP I/O Testing

Section 10.5.1

Vector GPACON[3:0]

CmpE242

Feb24 (Wed)

Example: ARM ToolChain Based Enviroment → ARM11

Linux O.S. Source → Kernel Image → Target Source Code @ Kernel

Compiled & Built

Device Drivers.

tmp = readl(S3C64xx_GPKDAT);

IO

GPP Port K DATA Register.

*.h porting to make O.S. to Target CPU

#define S3C64xx_GPKDAT 0x nnnn-nnnn

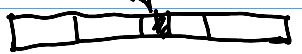
① O.S. Source Distribution — Pre-Built / Compiled + Built

{ ARM11
NVDA TX2

② ToolChain (Compiler + UI)

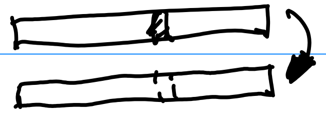
→ Example User Space Example folder (Program I/F to Device Drivers)

Example tmp &= ~(1<< (4+arg)); mask



"~" Negation → 0

"&" Bitwise AND



Char ← Drivers

↓

locate Drivers for the Target CPU "mini6410 ..."

Note: 1° Required to Be Able to Write a Simple Driver Test Code "hello.c"

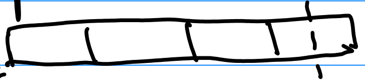
SPRx

mask → masking Logic Operation @ Bit wise Level

2° init, exit 2 modules

Example

4 bit



GPKCON

GPKCON[3:0]

to init GPKφ

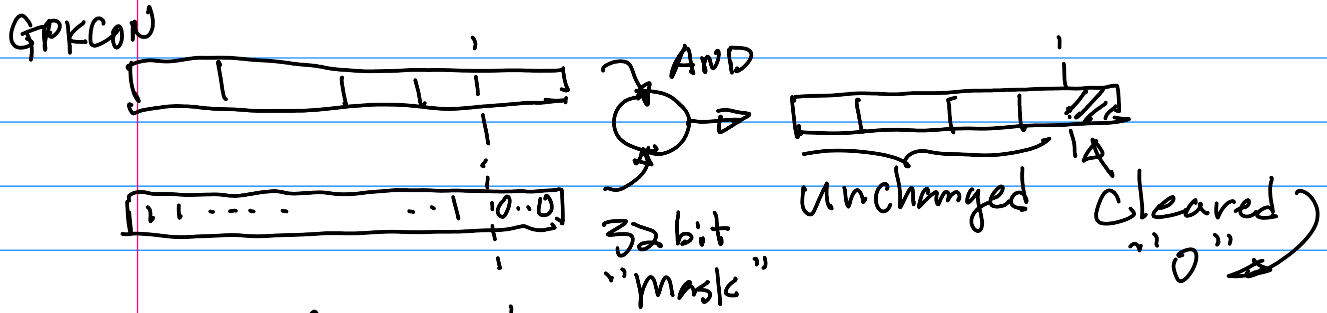
Clear GPKCON[3:0]

3° PrintK(" "); → Devices is installed invoked

↑ Kernel Space

Check the Source Code (to Be posted On the github)

* ioctl.h . #define DEVICE_NAME "ledsφ"



CPU Datasheet, ARM11,
Section 10.5.5, pp.320

(1) SPRs { CON ✓
DAT ✓
PUL

PWR management

(2) 0x7F008080 → GPECON

(3) GPE0~4 : Total 5 pins GPP I/F

GPE3 GPECON[15:12]

Once Driver Done → make menuconfig
↓ Kconfig

Load Kernel Image to the target Board ← module Built { a "M"
b "*"
c Void

↓
Done, Pie3B+/4
NAND
TX2

OR, Load Driver module
Copy Driver module to SD Card *.ko
↓
Target platform (O.S. Running Linux)

insmod : Install module

$ insmod myleddriver.ko ↓
$ "installed Driver"
printk(" ↓ ");
$ ./testmyLed ↓

rmmod to Remove the driver

Action 1: { (1) Download NVDA TX O.S. Distribution 250MB
(2) Download Toolchain
(3) Download Document

{ Linux O.S.
ARM Tool Chain → NVDA
→ Broadcom
→ Samsung
→ Google

March 1st (Monday)

Jonathon < B T Prototype
Akhil

PID Controller Design { a P.I.D. ~
b PWM Stepper motor
c LSM303 I/F

CmpE242

Ref: github: 2018S-12-...PID
2018S-14-...Stepper motor

Topics: 1° P.I.D Controller Design.
Proportional ~ Integral ~ Derivative ~

Consider Design CNC machine → Stepper motor Controll Core

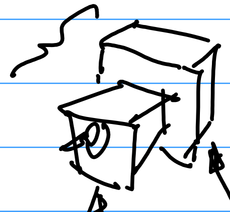
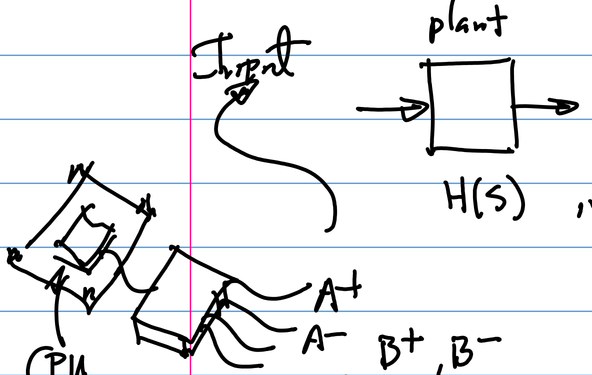


Fig 1. GearBox: Reduction Speed ↓ Gain Torques.



Output (speed, torque etc...)

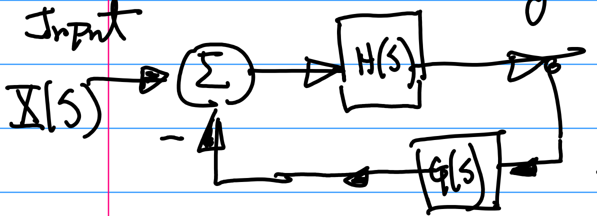
"Open-Loop" System → Challenges Uncertainty of the System.

Stepper motor Drive

Fig 1.b.

Note 1° Stepper Motor Drives Are Equiped with P.I.D. Control functions.

Fig 2



Output Y(S)

Sensor(s)/Transducer

Comparison of the Actual output with the Desired Output/Input

Note: $a-b$

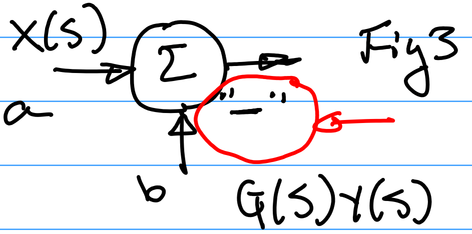


Fig 3

Comparison $\begin{cases} a-b \\ a/b \end{cases}$ ✓ Difference ↓ Error

Negative FeedBack Loop

Note 2° Performance Enhancement

✓ P: Proportional Controller
✓ I: Integral ~
✓ D: Derivative ~

Frequency Domain

$$E(S) = X(S) - G(S)Y(S) \quad \ldots (1)$$

$e(t)$, $e(\tau)$ Time Domain

Control Action Proportional to the error, in Time Domain

$K_p e(t)$ ... (2) (at this moment t)

Gain (Proportional Gain)

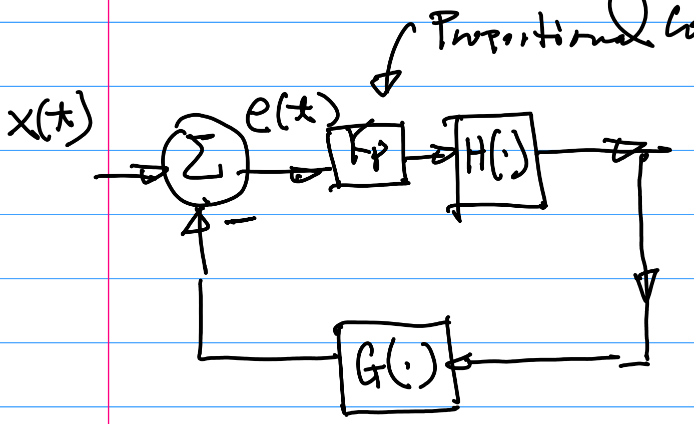


Fig4.

Now, Integral Controller

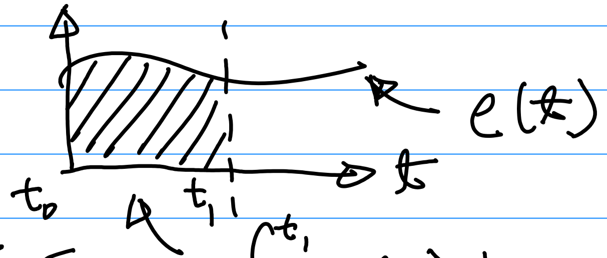


Fig5

$$\int_{t_0}^{t_1} e(\tau)d\tau \quad \ldots(3)$$

taking into the consideration of Controller's Accumulative Performance

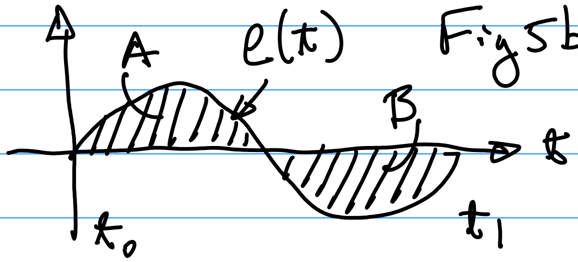


Fig5b

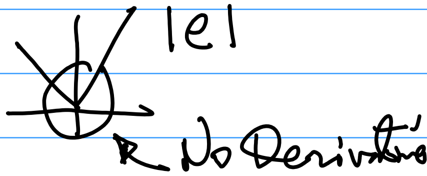
$$\int_{t_0}^{t_1} e(\tau)d\tau \simeq A-B=0 \quad \ldots(3b)$$

To solve Integration Cancellation Due,

Let's square the error $e(t)$, e.g. $e^2(t)$

Note: Don't use Absolute Value of the error.

Fig6.



Therefore

$$\int_T e^2(\tau)d\tau \quad \ldots(3c) \checkmark$$

Integral controller

$$K_I\int_T e^2(\tau)d\tau \quad \ldots(4)$$

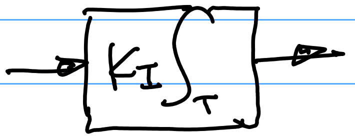
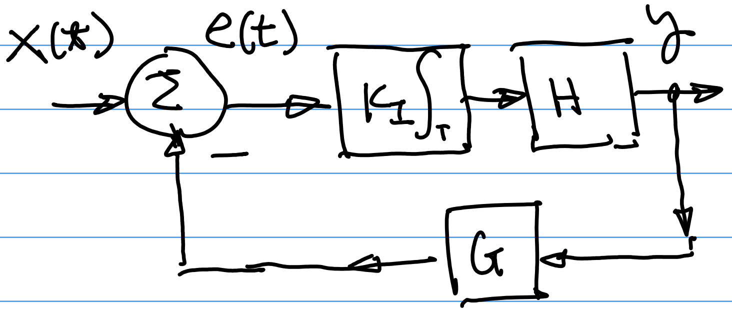


Fig6.

Hence,



Consider Derivatives

Background :

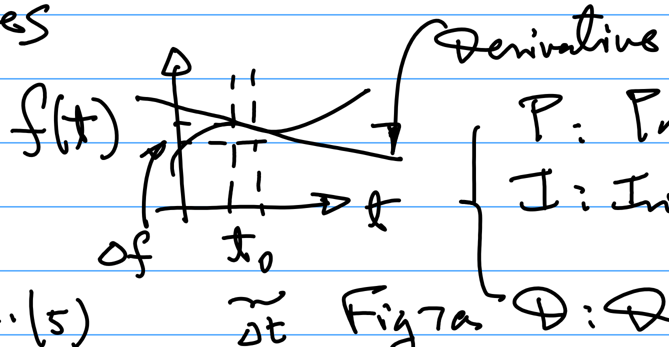


Fig 7a

P : Proportional

I : Integral (History Accumulative)

D : Derivative (Predictive)

$$\lim_{\Delta t \to 0} \frac{\Delta f}{\Delta t} = \frac{df}{dt} \quad \dots (5)$$

Changing Rate

$$\frac{df}{dt}\Big|_{t_0} > 0 \rightarrow f(t+t_0)\uparrow$$

$$\frac{df}{dt}\Big|_{t_0} = 0 \rightarrow \overset{f}{\text{unchanged}}$$

$$\frac{df}{dt}\Big|_{t_0} < 0 \rightarrow f(t+t_0)\downarrow$$

Action
1. Stepper Motor Drive Motor Ready
2. GPP + PWM Driver (Software)

Let $e(t) = f(t)$

$$\frac{d}{d\tau}e(\tau) = e'(\tau) \quad \dots (6)$$

Build derivative Controller

$$K_d \frac{d}{d\tau}e(\tau) \quad \dots (6b)$$

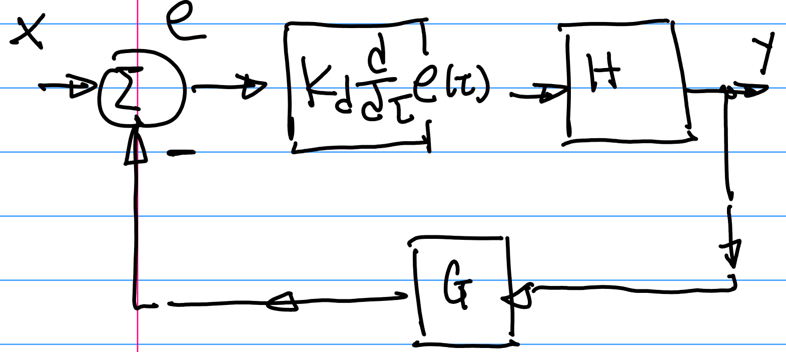


Fig 7b.

Predictive