CMPE242
Spring2023
PART II

April 3rd (Monday)

Road map for the 2nd half of the Semester:

ITDT (Industrial IDT):

- ADC Data Validation → ① F.F.T. (Fast Fourier Transform)
- Analog Sensor interface Design.
  - ② Power Spectrum Technique.
  - ③ Hardware Architecture Aspects
  - Ion Selective Electrode Sensors
    - Many Applications in different Industry Sector.

Homework Extension Next Monday with Demo.

Project: Due the End of the Semester.

- Implementation (1/3) 33%
  - PID.
  - I2C Sensor
  - PWM motor Control
    - Pre-processing C.E.T.
    - ADC
- Research Part: (1/3)
  - ① P.P.T Presentation on State-of-the-Art Technology in the Embedded world.
  - ② Report (Guideline)
  - ③ Proposal (One page), Submit to the CANVAS. for Approval. (No more than)
    - By Wednesday | Monday Next
- (1/3) Demo & Presentation: By the end of the Semester.

Example: for Analog ISE Sensor interface Design.

Ref:

2018S-20-NH3- 4Design2018-1-16.pdf

ammonia/ammonium ele

Cat No. S-05722-16 model 9512BNWP

Ammonia (NH3)
Ammonium (NH4+)

1. For both drinking water and was
2. EPA-approved for ISE analysis
3. The Orion ammonia electrode is with a chemical-resistant transluce
4. The easy-to-fill electrode comes to avoid overfilling and to monitor t without disassembling the electrod
5. Membrane replacement options loose membranes or preassemble membrane for the convenience of your own membrane.

20 replacement membranes, preas body with membrane, 60-mL of filli cable with BNC connector.

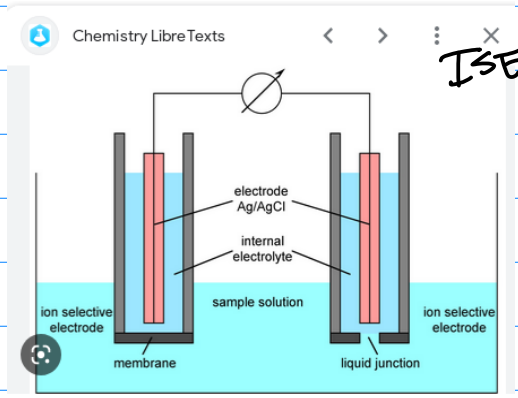
620

Fig.1

google Definition:

① Principle of ion-selective electrode (I.S.E.) An ideal I.S.E. consists of a thin membrane across which only the intended ion can be transported. The transport of ions from a high conc. to a low one through a selective binding with some sites within the membrane creates a potential difference.

→ Example of A Battery



ISE sensor

Fig.2

Working Principle of Battery - Electrical E... Visit

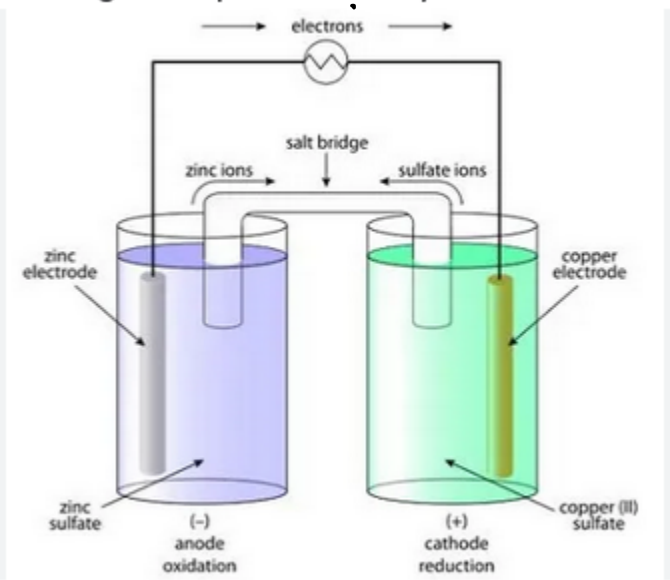


Fig. 3

Observation: Use Battery As An Example to Demonstrate Ion Selective Electrode Sensor. See NH_3/NH_4^+ sensor in Fig. 1.

Note 2: For the Non Linear Part, Let's perform Linearization — By using Piece-wise Linear Lines.

{ Piece-wise Line 1.
Piece-wise Line 2.

Next Step is to formulate Each Line by using Linear Equation.

$$\frac{y_2-y_1}{x_2-x_1}=\frac{y-y_1}{x-x_1} \quad \ldots (1)$$

Solve for $y=bx+c$ (see the previous Notes).

With Simplification By Removing Very Low Concentration Part, we have

Characteristic

Typical NH3 Calibration Curve

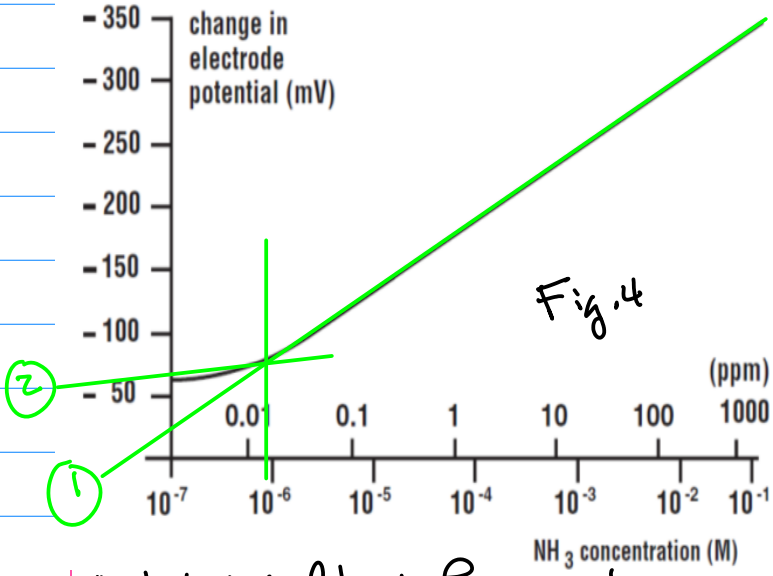


Fig. 4

Note 1. We like to have the Linear Characteristics from the Calibration Curve, such as [1,10], [10,100], [100,1000], etc.

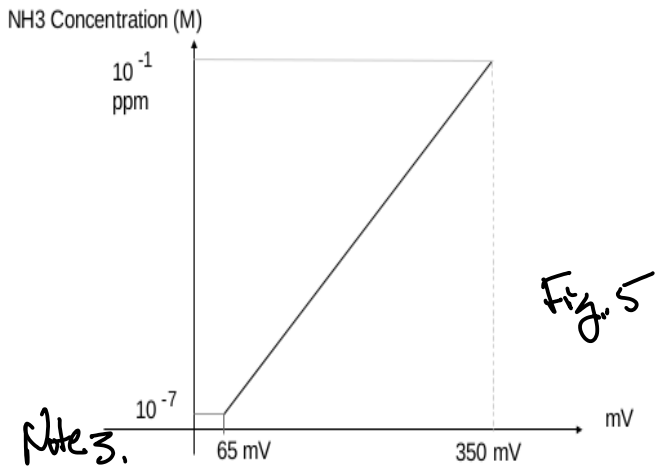


Fig. 5

Note 3. Then, Change the Cal-Curve to the Characteristic Curve e.g. Horizontal Axis is voltage for the Design of interface.

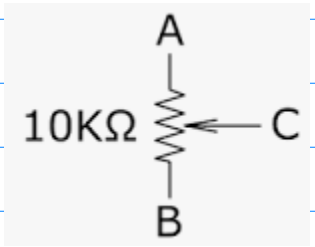


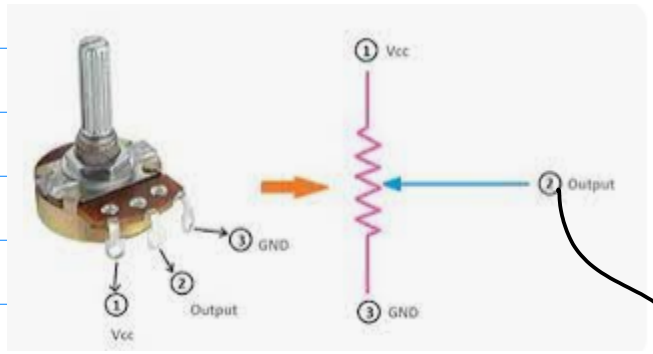
Fig.1

Industrial Analog Sensor Interface Protocol:
1° Output has to be defined by Current. 2° The Range 4~20mA.

Control.com
https://control.com › Technical Articles

Why is 4-20 mA Current Used for Industrial Analog Sensors?

Note 4. Fit to the Dynamic Range of your target platform.
↓
External ADC Needed
CMOS Range ADC [0,3.3]
TTL " " [0, 5.0]



April 5th (Wed)
Note 1° New updated Due Date for the Motor Control, please use PWM from your target platform as the input to the Motor Controller.

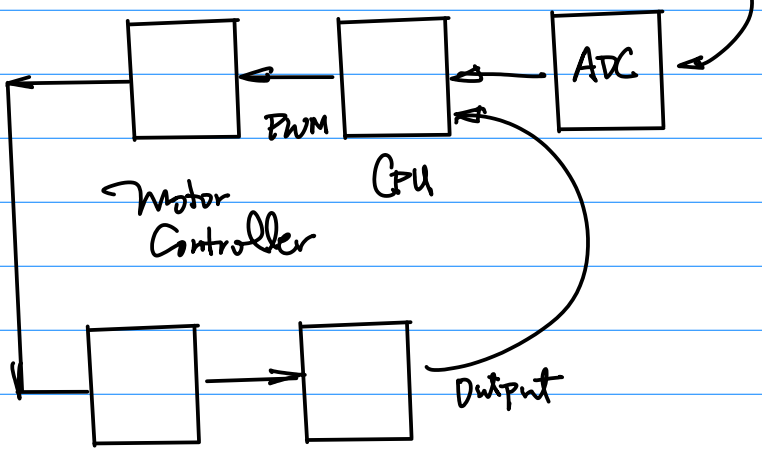


Fig.2

2° ADC Unit for the project
Selection Guide:
1° Interface protocol
I2C; ~4 mbps
2° 0.5~2 MSPS
(Million Samples per Second)
3° 10~12 bits per Sample.
Example: P.O.T.

https://www.adafruit.com/product/1083

ADS1015 12-Bit ADC - 4 Channel with Programmable Gain Amplifier - STEMMA QT / Qwiic

Product ID: 1083 **$9.95**

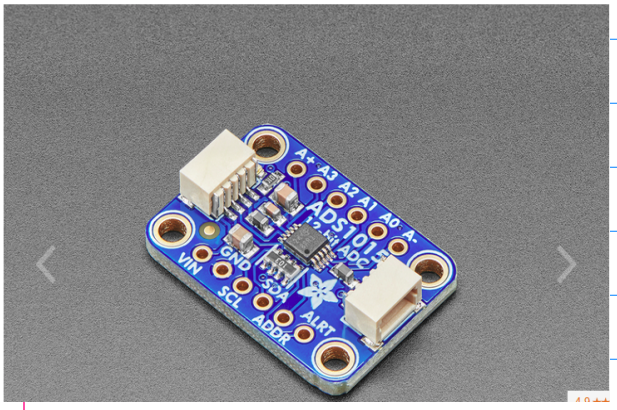


Fig.3

Note: Input Voltage Range

TEXAS INSTRUMENTS

ADS1013
ADS1014
ADS1015

www.ti.com

SBAS473C – MAY 2009 – REVISED OCTOBER 2009

ELECTRICAL CHARACTERISTICS

All specifications at –40°C to +125°C, VDD = 3.3V, and Full-Scale (FS) = ±2.048V, unless otherwise noted.
Typical values are at +25°C.

PARAMETER	TEST CONDITIONS	ADS1013, ADS1014, ADS1015 MIN	TYP	MAX	UNIT
ANALOG INPUT					
Full-scale input voltage[1]	$V_{IN} = (AIN_P) - (AIN_N)$		±4.096/PGA		V
Analog input voltage	AIN_P or AIN_N to GND	GND		VDD	V
Differential input impedance			See Table 2		
Common-mode input impedance	FS = ±6.144V[1]		10		MΩ
	FS = ±4.096V[1], ±2.048V		6		MΩ
	FS = ±1.024V		3		MΩ
	FS = ±0.512V, ±0.256V		100		MΩ

Example: Design Objective.
1° Selection of An ADC. 1024
Construct the Characteristic Curve of the ADC

V_{max} = 3.3V for CMOS

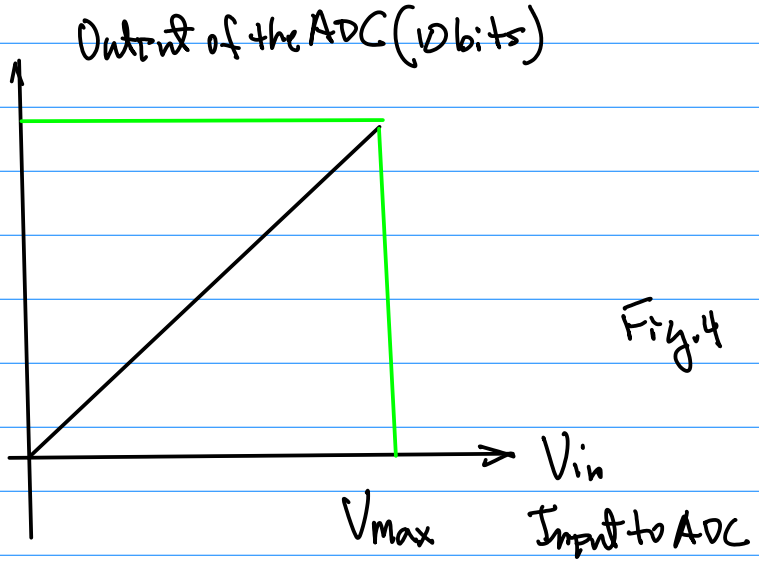


Fig.4

2° Design Objective: To Design A pre-process unit to make the Analog Sensor Output match to the ADC input dynamic Range.

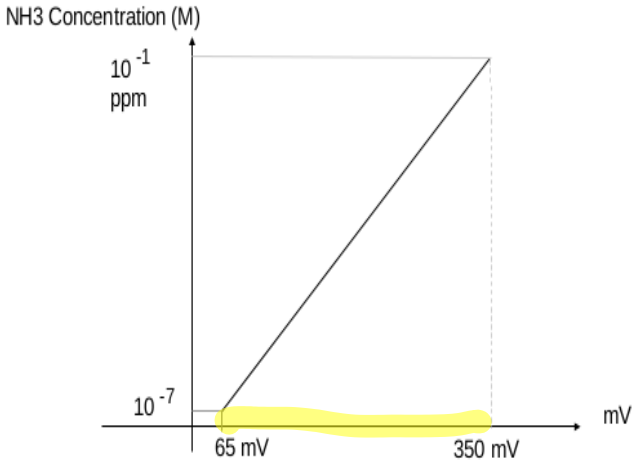


Fig.5

Step 1. 65mV → 0
350mV → V_{MAX}
In general,
$V_{s,min} \rightarrow V_{ADC,min}$ ... (1)
$V_{s,max} \rightarrow V_{ADC,max}$ ... (2)

Apply to (max).

Example: Continuation of the Preprocessing Design.

Note 1. Check 1015 ADC Dynamic Range for the Input. [0, 3.3V]. Verification is Needed

Theoretical Analysis:

Step 1. Provide "offset" to shift → The Sensor dynamic Range, Subtraction Can be utilized for this purpose. e.g.

$V_{sen} - V_{offset}$ ... (1)

which will Lead to the Result Below.

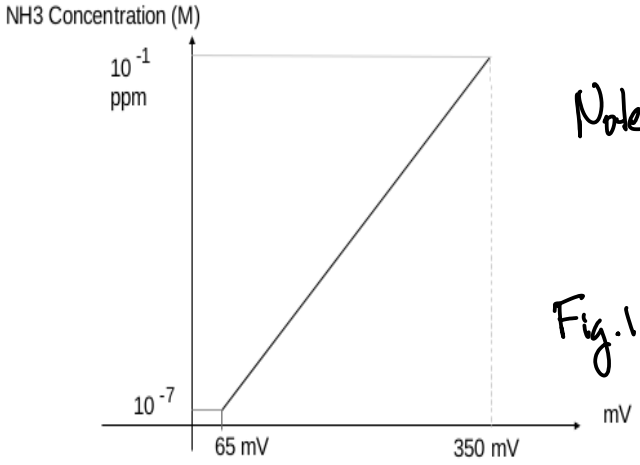


Fig.1a

Note1. For more generalized Case, Let
$V_{min} = 65mV$, $V_{max} = 350mV$.
So, the offset $= -V_{min}$.
then, the Upper Bound after offset is
$V_{max} - V_{min}$

↓ Subtraction/offset

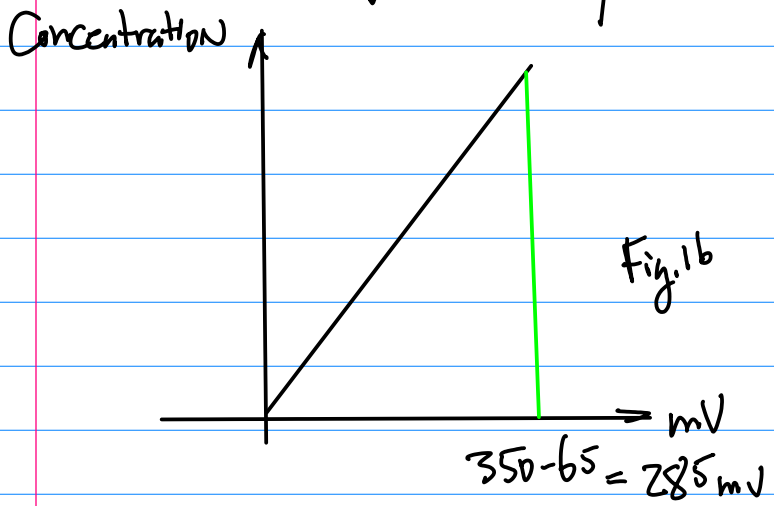


Fig.1b

Step2. To magnify the sensor Output Range to match the entire Dynamic Range of the ADC.

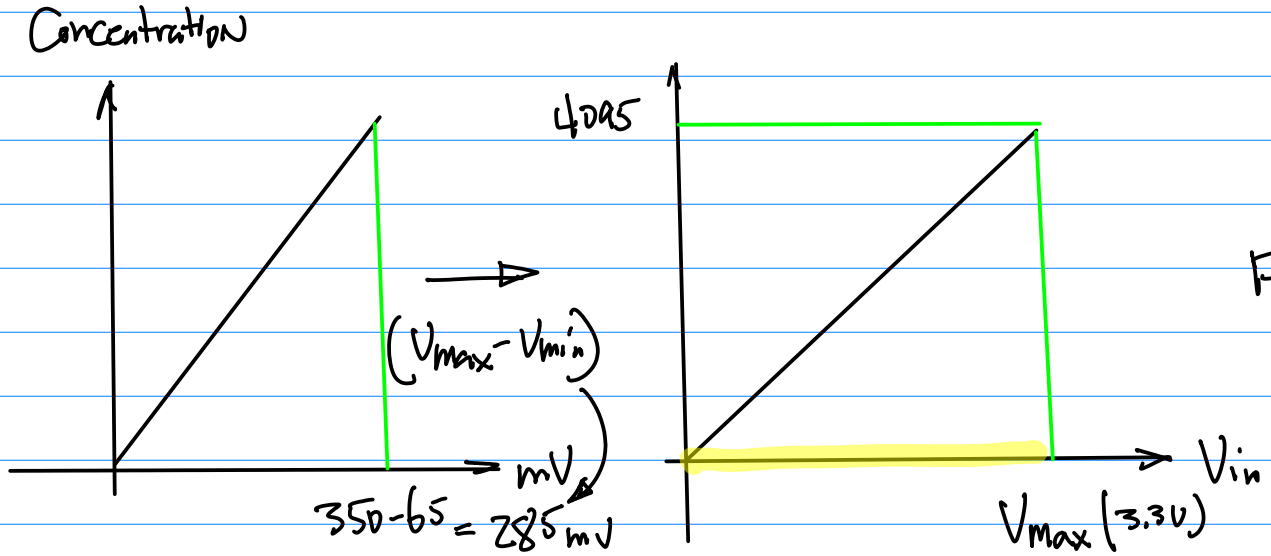


Fig.2

Find the Gain for the Magnification

$$A = \frac{V_{Output\ Range}}{V_{Input\ Range}} = \frac{3.3}{285 \times 10^{-3}} \cong 11.58$$

where 3.3 VDC is from 10 bit ADC for Example.

Example: Hardware Design for the Pre-processing.

Ref.

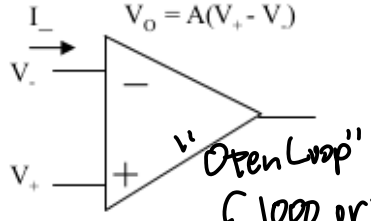
1-0lecture10_OpAmp Circuits.pdf

Note1: Using OpAmp for Pre-processing Not for the Buffering.

OpAmp Device As a Buffering Stage

Both Analog and Digital Circuit

Note2: Background



(1) To protect the previous stage's output signal, which is the input to the next stage while sampling/connecting the signal to its next stage logic circuit. (2) Unit gain non-inverting OpAmp configuration is an excellent choice.

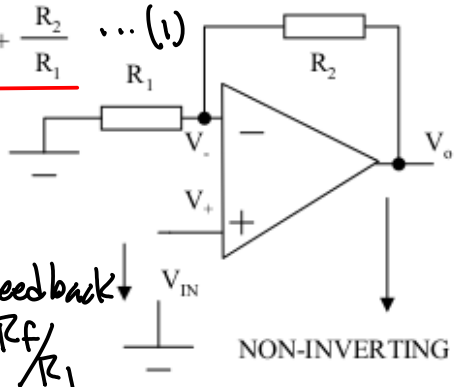
1000 or Bigger

1×10^{-9} A or Smaller

Ideal OpAmp Properties: (1) very large gain, A>>M; (2) draws very little current, I ~0, e.g., very high impedance; (3) $V_O = A(V_+ - V_-)$ is finite range, which leads to $V_+ = V_-$.

for Example 100 MΩ or higher.

$\frac{V_O}{V_{IN}} = 1 + \frac{R_2}{R_1}$ ... (1)

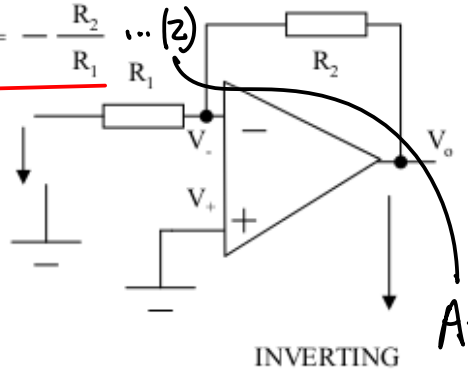


NON-INVERTING

R_2 for feedback

so $A = 1 + R_f/R_1$

$\frac{V_O}{V_{IN}} = -\frac{R_2}{R_1}$ ... (2)



INVERTING

$A = -\frac{R_f}{R_1}$

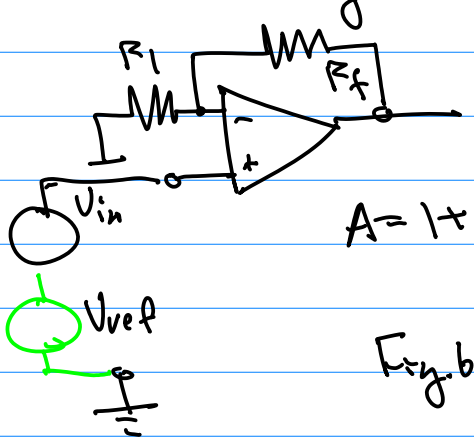
Fig.3

V_1 $I \ll 0$ V_2

Now, Consider the design Implementation for 2-Step Process.

Tools { Non-Inverting ~ | Inverting ~
Linear System: Super Impose One Signal (offset) on to the other Signal (Input).

For Non-Inverting



$A = 1 + \frac{R_f}{R_1}$

Fig.6a

"Black Box"

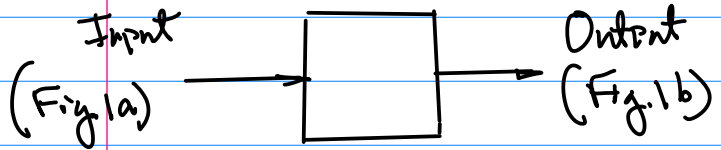
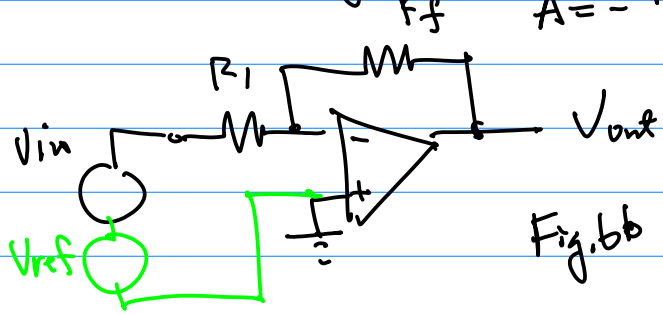


Fig.4.

Linear System. "Super Impose Signal for Offset"

For Inverting.



$A = -\frac{R_f}{R_1}$

Fig.6b

Note: Selection of Non-~ v.s. Inverting configuration Depends on your design Need. This Design is an illustration of Super Imposing an "offset", e.g. V_{ref}.

V_{in}.

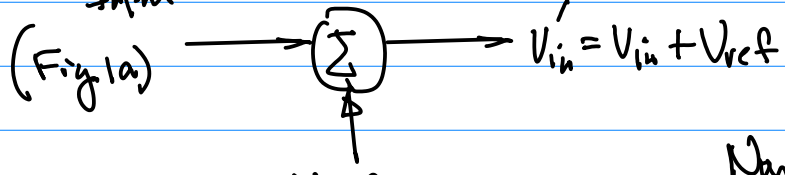


Fig.5

In the Circuit Design. Just Connect 2 inputs together.

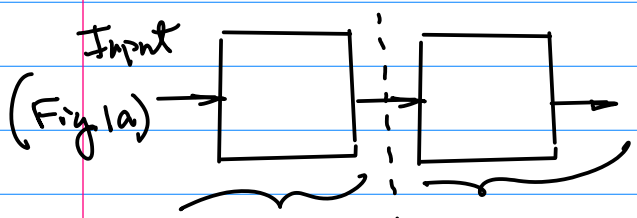
Now, to Magnify the Signal to match ADC's Dynamic Range.
Choose Non-Inverting configuration (see Fig.2)

Hence

$$A = 1 + \frac{R_f}{R_1} = 11.58$$

Choose $R_1 = 1\ K\Omega$ (10.58 $K\Omega$?)
Solve for $R_f \simeq 7\ K\Omega$
please verify it !

Stage1 Stage2
Box1 . Box2



Analog Devices
https://www.analog.com › ltspice-simulator
LTspice Information Center
LTspice® is a powerful, fast, and **free SPICE simulator** software, schematic capture and waveform viewer with enhancements and models for improving the ...

Free for Download, "Originated from "Linear" A Silicon Valley Company.

EasyEDA
https://easyeda.com
EasyEDA - Online PCB design & circuit simulator
EasyEDA is a **free** and easy to use circuit design, **circuit simulator** and in your web browser.

Requirements: To Be Able to Run SPICE Simulator.

April 12 (Wed).

Note 1. The Last Project Preparation.
(Requires the Semester End Presentation).

Note 2. Implementation of ADC Unit.
P.O.T. 47K or 470K or Similar.
↓
ADC → Target → PWM → Controller
↑
LSM303 ← Motor ←

Note 3. ADC Data Validation
FFT. Power Spectrum.

Note 4. Road Map.
IIOT (Analog Sensors. 4~20mA)
↓
Preprocessing
↓
OpAmp. (On-Line Simulation Tool)

Example:

ISE

(1) Analog Sensor

Stage 1 Stage 2

a. ≃285mV Box 1 Box 2

Input

b. Output ≃ 3.3V (CMOS)

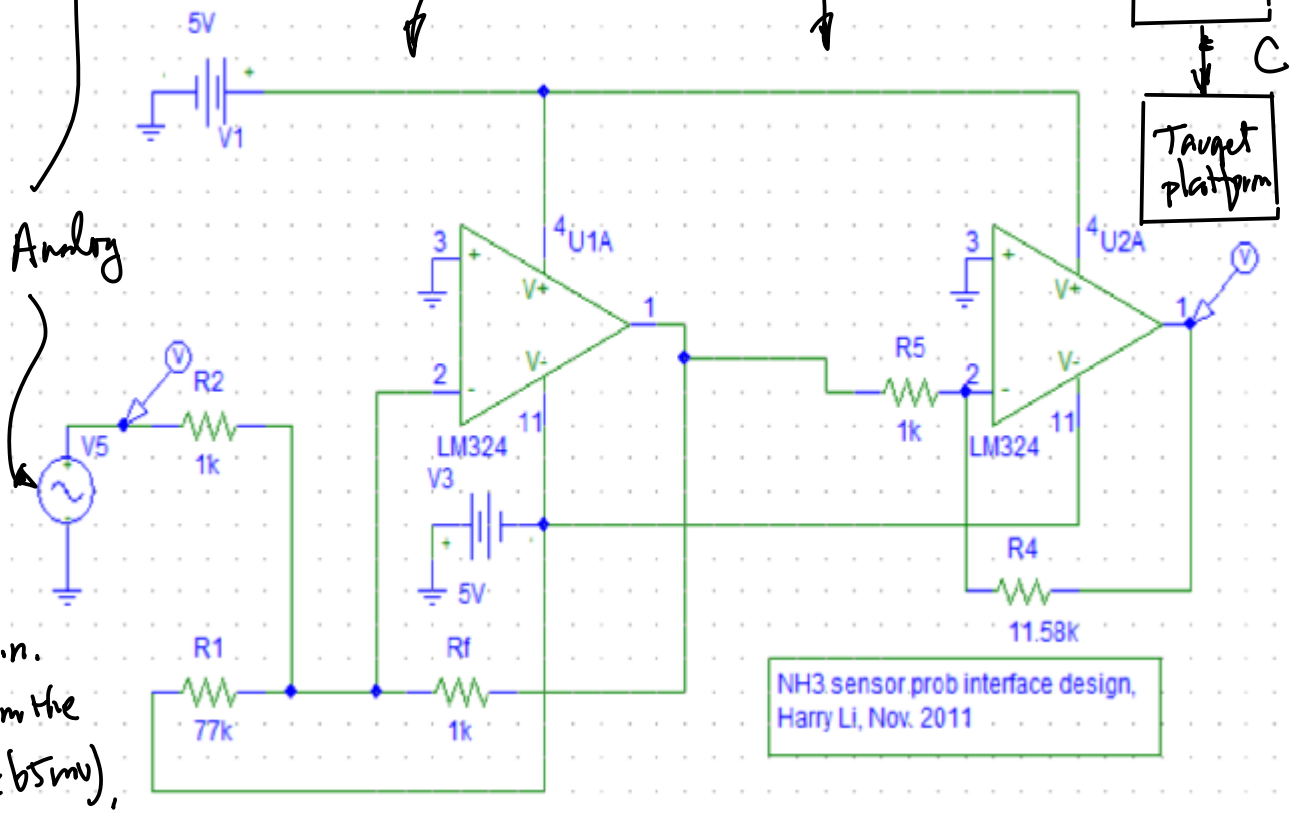
offset Gain Amplification

ADC

12bit ADC

c. Output ≃4095

Target platform

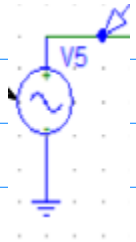


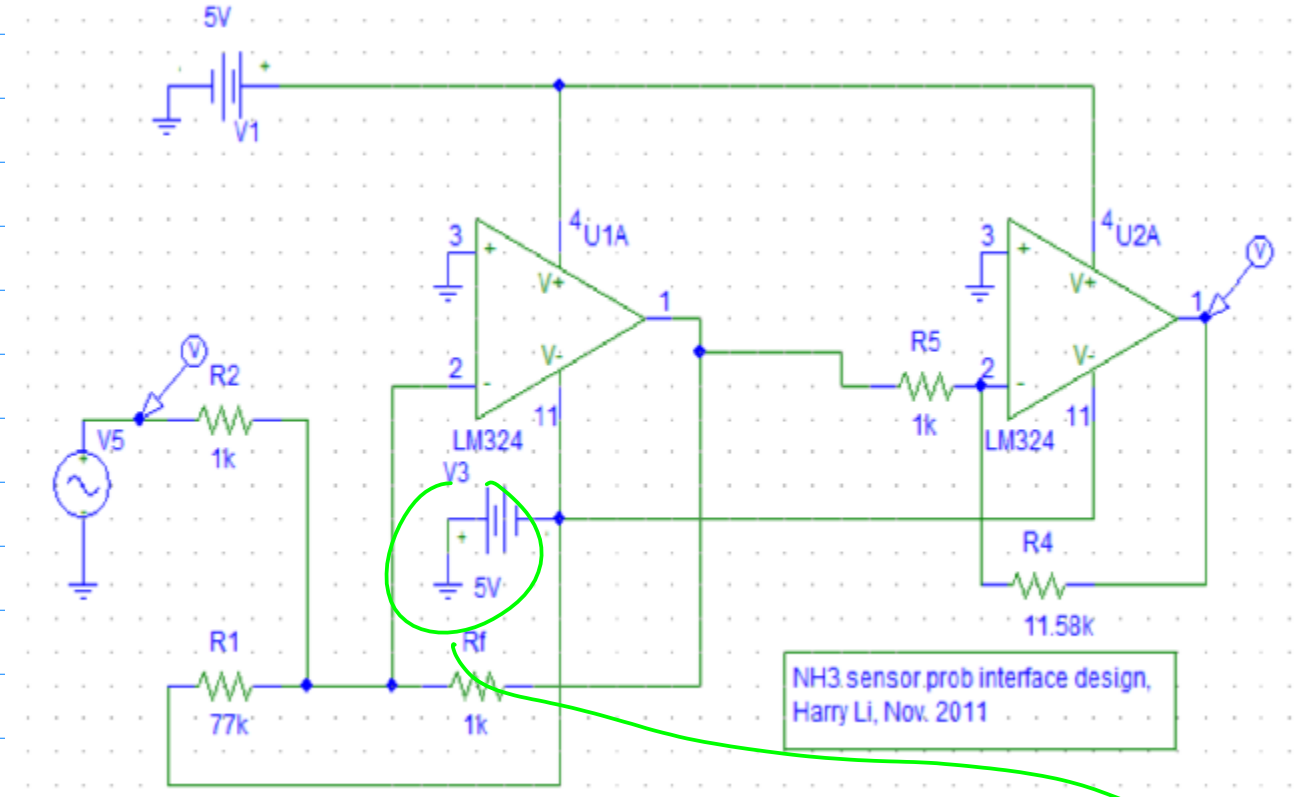
Note: 1° Analog Sensor Signal Source sine Wave Signal

2° for min. Input from the Sensor. (≃65mv), we want to get ADC Output = 0.

for max Input from the Sensor, we want to get Output: (3.3VDC) 4095 ADC

Note: 3. Simulation of the Sensor Output as the input to the pre-processing Circuit.





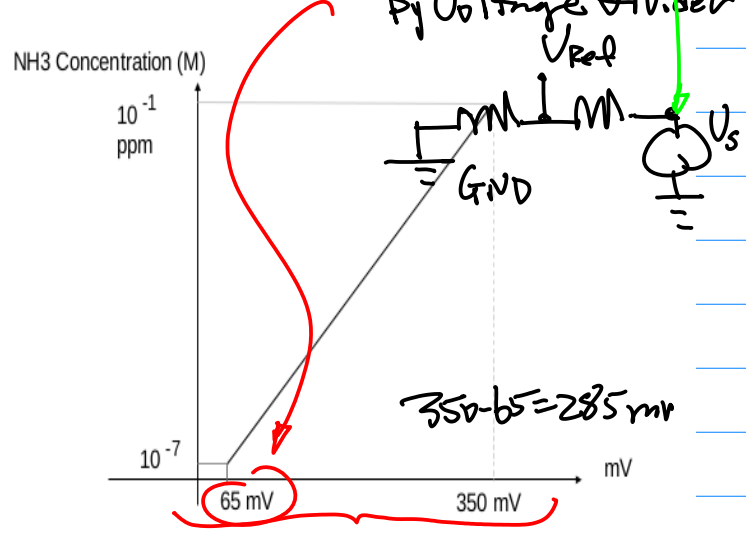
April 17 (Monday).

1° Project (Integration of Homework + ADC). Due May 7 (Sunday) plus Research Part / Presentation.

2° Bonus Points (5%) for BLDC Motor Control; 3-phase (U, V, W) Motor Control.

3° SPICE Simulation for Pre-processing Circuit Simulation.

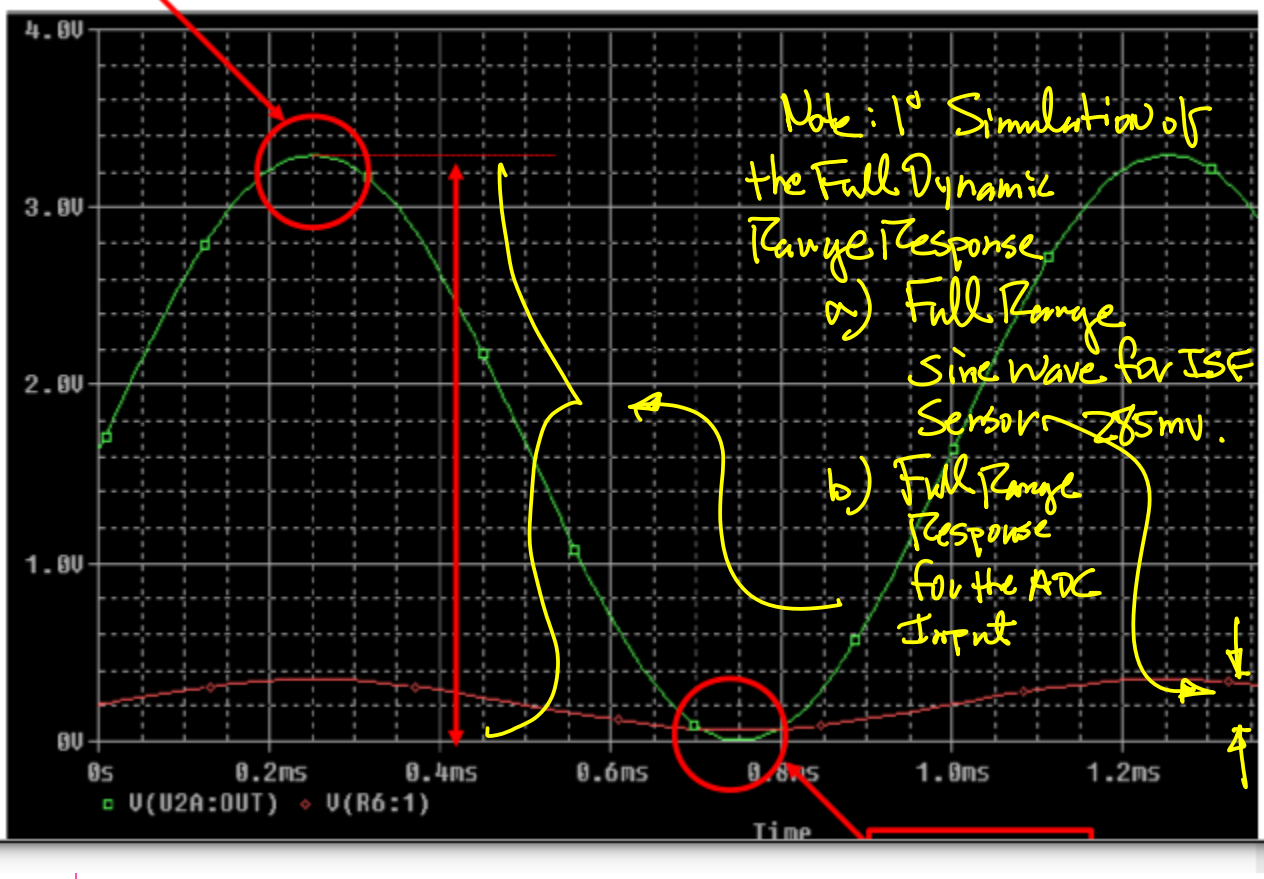
Example: Continuation. Provide "offset" By Voltage Divider



2018S-20-NH3- 4Design2018-1-16.pdf

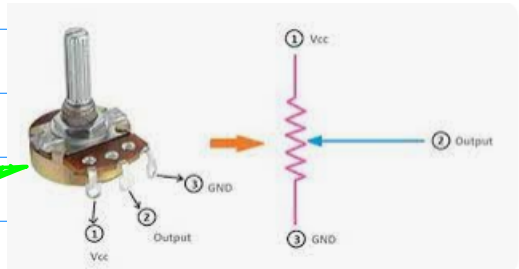
Simulation Result

The output: 3.3V



ADC Data Validation

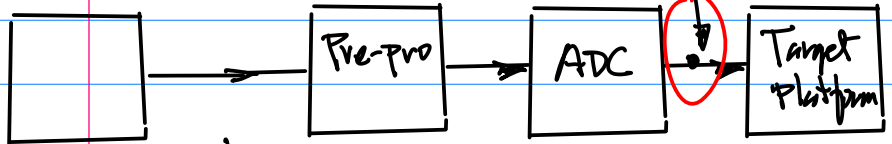
Tool: Fourier Spectrum Analysis.



Data Validation

P.O.T. Human Operator

See PP.48 For Details.



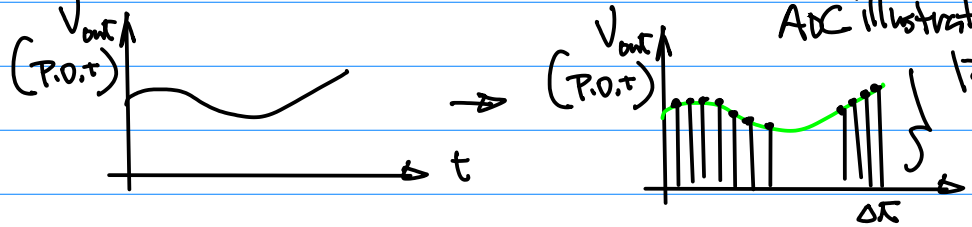
ADC Output

$\{x(n) | n=0,1,2\ldots\}$

ADC illustration

12bit ADC.

[0, 4095]



Background / Formulation.

To validate $\{x(n)\}$, or $\{x(n) | n=0,1,2,\cdots,\}$

$x(n)$.

D.F.T (Discrete Fourier Transform) is defined as follows.

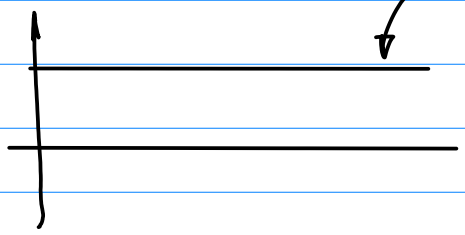
$$X(m) = \frac{1}{N}\sum_{n=0}^{N-1} x(n) e^{-j2\pi \frac{mn}{N}} \quad \cdots (1)$$

(Time Index)

physical meaning: $X(m)$, Discrete Fourier Transform.

m: Frequency Index.

$m=0$, D.C. index; $X(0)$ D.C. Component.



$m=1$, $X(1)$ Fundamental Frequency Component.

N: One Period. Total No. of Points per a Period. Such as $N=1024, 2048, 4096$, etc.

$N=2^x$ for FFT Only.

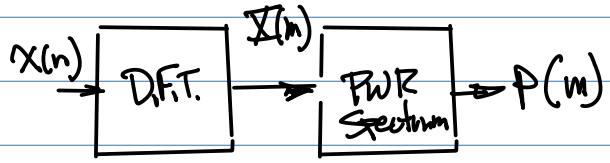
(Fast Fourier Transform)

$$e^{-j2\pi \frac{mn}{N}} = \cos 2\pi \frac{mn}{N} - j \sin 2\pi \frac{mn}{N} \quad \cdots (2)$$

Euler Formula

$$e^{j\phi} = \cos\phi + j\sin\phi \quad \cdots (3)$$

Power Spectrum of $X(m)$.



Let's Define the Power Spectrum as:

$$P(m) = \sqrt{Re^2[X(m)] + Im^2[X(m)]} \quad \cdots (4)$$

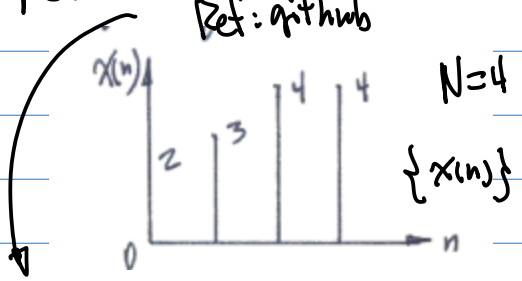
Where

$$Re[X(m)] = Re\left[\frac{1}{N}\sum_{n=0}^{N-1} x(n) e^{-j2\pi \frac{mn}{N}}\right]$$

$$Im[X(m)] = Im\left[\frac{1}{N}\sum_{n=0}^{N-1} x(n) e^{-j2\pi \frac{mn}{N}}\right]$$

Example:

Ref: github



$N=4$

$\{x(n)\}$

CMPE242-Embedded-Systems- / 2018S-26-1D-DFTv2.pdf

$x(0)=2$, $x(1)=3$, $x(2)=4$, $x(3)=4$.

Find $X(m)$ D.F.T.

$$X(m) = \frac{1}{4}\sum_{n=0}^{3} x(n) e^{-j2\pi \frac{mn}{4}} \quad \cdots (5)$$

Next. From Eqn(5)

For m=0,

$$X(0)=\frac{1}{4}\sum_{n=0}^{3}x(n)e^{-j\frac{2\pi}{N}0n}=\frac{1}{4}\sum_{n=0}^{3}x(n)$$

$$=\frac{1}{4}\left(x(0)+x(1)+x(2)+x(3)\right) \quad \cdots(6a)$$

For m=1

$$X(1)=\frac{1}{4}\sum_{n=0}^{3}x(n)e^{-j2\pi\cdot\frac{1\cdot n}{4}}$$

$$=\frac{1}{4}\left[x(0)\cdot 1+x(1)e^{-j2\pi\frac{1}{4}}+x(2)e^{-j2\pi\frac{2}{4}}+x(3)e^{-j2\pi\frac{3}{4}}\right) \quad \cdots(6b)$$

April 19 (Wed)

Final Exam Schedule: 18th (Thur) 12:15-2:30 pm

Group I Classes

Group I classes are those classes which meet M, W, F, MTW, MWR, MTWF, MWRF, MTWRF, MW, WF, MWF, MF, TW, WR, MT, WS.

Regular Class Start Times	Final Examination Days	Final Examination Time
7:00 through 8:25 AM	Friday, May 19	7:15-9:30 AM
8:30 through 9:25 AM	Tuesday, May 23	7:15-9:30 AM
9:30 through 10:25 AM	Thursday, May 18	7:15-9:30 AM
10:30 through 11:25 AM	Monday, May 22	9:45 AM-12:00 PM
11:30 AM through 12:25 PM	Wednesday, May 17	9:45 AM-12:00 PM
12:30 through 1:25 PM	Friday, May 19	12:15-2:30 PM
1:30 through 2:25 PM	Tuesday, May 23	12:15-2:30 PM
2:30 through 3:25 PM	Thursday, May 18	12:15-2:30 PM
3:30 through 4:25 PM*	Monday, May 22	2:45-5:00 PM
4:30* through 5:25 PM*	Wednesday, May 17	2:45-5:00 PM

Note: Final Exam is in the same format as the midterm. Be Sure to Bring your Prototype System.

Note2:

Project ON CANVAS. Part I & Part II. 25pts.

Part I: Integration of PID Control with ADC

Part II: Research. PPT.

Note3: Presentation Date

8th (Monday) | 10th (Wed) | Final (18th, Th)

(1) Presentation. 12:15-2:30 pm.

(2) In-Class Demo of the Project.

Last Day of Class 15th. (Monday) Review.

Example: Continuation of D.F.T. Example.

For m=2, from Eqn(4) pp58

$$X(m)=\frac{1}{4}\sum_{n=0}^{3}x(n)e^{-j2\pi\frac{2\cdot n}{4}}$$

$$m=2$$

$$=\frac{1}{4}\left[x(0)e^{-j0}+x(1)e^{-j2\pi\frac{1\cdot 2}{4}}+x(2)e^{-j2\pi\frac{2\cdot 2}{4}}+x(3)e^{-j2\pi\frac{3\cdot 2}{4}}\right] \quad \cdots(6c)$$

for m=3,

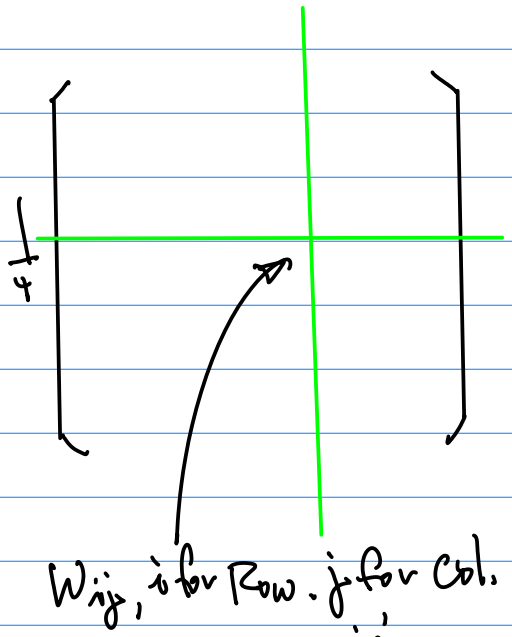
$$X(m)=\frac{1}{4}\sum_{n=0}^{3}x(n)e^{-j2\pi\frac{3\cdot n}{4}}$$

$$m=3$$

$$=\frac{1}{4}\left[x(0)e^{-j\cdot 0}+x(1)e^{-j2\pi\frac{3\cdot 1}{4}}+x(2)e^{-j2\pi\frac{3\cdot 2}{4}}+x(0)e^{-j2\pi\frac{3\cdot 3}{4}}\right] \quad \cdots(6d)$$

Hence, we can form a Col. Vector
By Arranging $\underline{X}(0)$,
$\underline{X}(1), \cdots, \underline{X}(3)$ as follows.

$$\begin{bmatrix} \underline{X}(0) \\ \underline{X}(1) \\ \underline{X}(2) \\ \underline{X}(3) \end{bmatrix} = \frac{1}{4} E_{4\times 4} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \quad \cdots (1)$$



W_{ij}, i for Row, j for Col.

where

$$E_{4\times 4} = \begin{bmatrix} W_{00} & W_{01} & W_{02} & W_{03} \\ W_{10} & W_{11} & W_{12} & W_{13} \\ W_{20} & W_{21} & W_{22} & W_{23} \\ W_{30} & W_{31} & W_{32} & W_{33} \end{bmatrix}_{4\times 4} \quad \cdots (2)$$

where $W_{ij} = e^{-j2\pi \frac{i\cdot j}{N}}$ ..(4)

From Euler Equation

$$e^{-j2\pi \frac{i\cdot j}{N}} = \cos 2\pi \frac{i\cdot j}{N} - j \sin 2\pi \frac{i\cdot j}{N} \quad \cdots (5)$$

Find the Entry of E matrix at 1st Row, 1st Col. Location.

Since, we have Eqn (5).

$$e^{-j2\pi \frac{i\cdot j}{N}} = e^{-j2\pi \frac{i\cdot j}{4}}$$

$$= e^{-j2\pi \frac{i\cdot j}{4}}\Big|_{\substack{i=0=1 \\ j=0}}$$

Note: $W_{i\,j}$ ... (3)

Index for Row. Index for Col.

From Eqn (1) & (2), we have

$W_{ij}\big|_{i=1,j=2}$

$$\begin{bmatrix} \underline{X}(0) \\ \underline{X}(1) \\ \underline{X}(2) \\ \underline{X}(3) \end{bmatrix}_{m\times 1} = \frac{1}{4} \begin{bmatrix} W_{00} & W_{01} & W_{02} & W_{03} \\ W_{10} & W_{11} & W_{12} & W_{13} \\ W_{20} & W_{21} & W_{22} & W_{23} \\ W_{30} & W_{31} & W_{32} & W_{33} \end{bmatrix}_{m\times n} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}_{n\times 1} \quad \cdots (3)$$

for the 3rd Row, 2nd col.

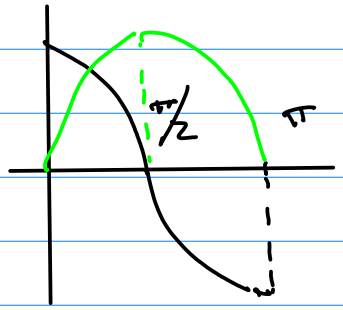
$i=2$, $j=1$

From

$$e^{-j2\pi \frac{i\cdot j}{N}}\Big|_{\substack{i=2,j=1 \\ N=4}} = e^{-j2\pi \frac{2}{4}}$$

$e^{-j2\pi\frac{2}{4}} = e^{-j\pi} = \cos\pi - j\sin\pi = -1$

$P[m] = \sqrt{Re^2[X[m]] + Im^2[X[m]]}$



$P[0] = \sqrt{Re^2[X[0]] + Im^2[X[0]]}$
$= \sqrt{3.25^2 + 0^2} = 3.25$

Now, Consider the Last Row, Last Col. $i = N-1 = 3$, $j = N-1 = 3$.

From

$e^{-j2\pi\frac{3\cdot3}{4}} = e^{-j\frac{9\pi}{2}} = e^{-j\frac{8\pi}{4} - j\frac{\pi}{4}}$

$= e^{-j4\pi - j\frac{\pi}{2}} = 1\cdot e^{-j\frac{\pi}{2}} =$

$= \cos\frac{\pi}{2} - j\sin\frac{\pi}{2} = 0 - j\cdot 1 = -j$

$\cos\frac{4}{4}\pi - j\sin\frac{4}{4}\pi = 1$ ✓

$X[0] = \frac{1}{4}(2+3+4+4) = 3.25$
$X[1] = \frac{1}{4}(2-3j-4+4j) = \frac{1}{4}(-2+j)$
$X[2] = \frac{1}{4}(2-3+4-4) = \frac{1}{4}(-1)$
$X[3] = \frac{1}{4}(2+3j-4-4j) = \frac{1}{4}(-2-j)$

for $m=1$,

$P[1] = \sqrt{Re^2[X[1]] + Im^2[X[1]]}$
$= \sqrt{(-\frac{2}{4})^2 + (\frac{1}{4})^2}$ finish the evaluation

April 24 (Monday).

From the Handout, Ref from the github.

Example: FFT.C

CMPE242-Embedded-Systems- / 2018S-26-fft.c

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}\begin{bmatrix} 2 \\ 3 \\ 4 \\ 4 \end{bmatrix}$$

So, ... Step. ...(3*)

Step 1. Find $E_{N\times N}$. 2. Find D.F.T. using $E_{N\times N}$ And $x(n)$.
Step 3. Find $P(m)$ for $m = 0, 1, \cdots, N-1$

CmpE242
Spring 2023

10

Note: You can Compile/Build for Your Laptop platform. No Need for Arm_Linux-gcc Cross Compiler.

62

```
1  /*****************************************************************
2  * Program is for CMPE class use, see Dr. Harry Li's lecture notes for details *
3  * Reference: Digital Signal Processing, by A.V. Oppenhaim;
4  * fft.c for calculting 4 points input, but you can easily expand this to 2^x inputs;
5  * : x0.1;              Date: Sept, 2009;
6  * Note: cross compiled for arm-linux-gcc, be sure to modify make file
7  to link math lib when compiling, by adding -lm
8  This code then was tested on ARM11 board. Feb 2015.
9  *****************************************************************/
```

20 Link Math Library

~/Desktop/SJSU/befor2018/EE264/EE264Ubuntu/OpenGL/CT/lec22FFTIFFT$ ./fft

```
harry@harry-laptop:~/Desktop/SJSU/befor2018/EE264/
IFFT$ ls
fft  fft_cpp.cpp  fft_ifft_cpp.cpp  ifft_cpp.cpp
harry@harry-laptop:~/Desktop/SJSU/befor2018/EE264/
IFFT$ ./fft
*********Before*********
X[1]:real == 2.000000 imaginary == 0.000000
X[2]:real == 3.000000 imaginary == 0.000000
X[3]:real == 4.000000 imaginary == 0.000000
X[4]:real == 4.000000 imaginary == 0.000000


**********After*********
X[1]:real == 13.000000 imaginary == 0.000000
X[2]:real == -2.000000 imaginary == 1.000000
X[3]:real == -1.000000 imaginary == 0.000000
X[4]:real == -2.000000 imaginary == -1.000000
It took me 145 clicks (0.000000 seconds).
harry@harry-laptop:~/Desktop/SJSU/befor2018/EE264/
IFFT$
```

```
/*****************************************************
*
//This program is converted from a FORTRAN program
g *
//book by A.V. Oppenhei; *
//Status: Tested; *
//gcc fft.cpp -o fft *
//Additional information: See Dr. Hua Harry Li's ha
//fft.cpp for calculting 4 pts input, but can easil
*
//*****************************************************
```

Note: To Compile -lm ON x86 platform.

April 26 (Wed)

Example: Data Validation using Power Spectrum of the F.F.T.

Note1: Baseline Code 128 pts.

Power Spectrum.

Nyquest Sampling Theorem:

$f_{sampling} \geq 2 f_{max}$ ... (1)

Create 128 pts Data As a Baseline Test.

Note: $P(m) = P(m + KN)$ ... (1)

Periodical Function.

Period = N

$P(m) = P(-m)$ ... (2)

Even function

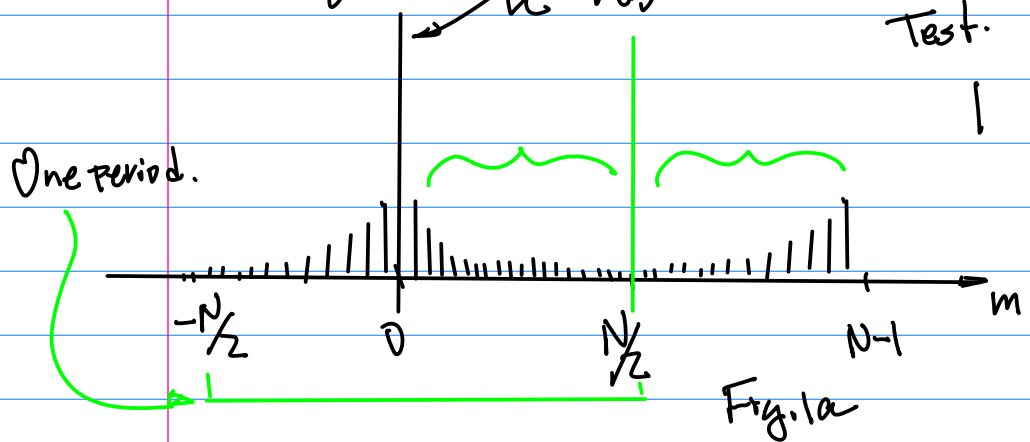


Fig.1a

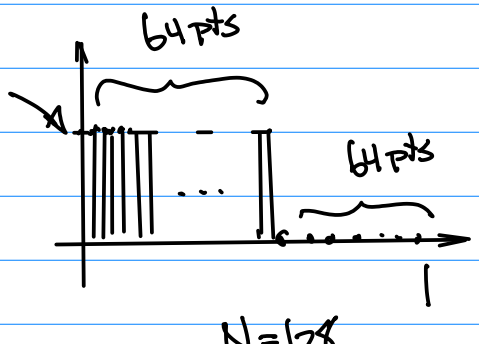


Fig.1b

CMPE242
Spring2023

Execution of the F.F.T and Computation of the Power Spectrum, Lead to the Similar plot as in Fig.1a. plot1.

Discussion:

$P(m)\big|_{m=N/2}$ Freq. Component at the highest Frequency Index m.

Now, Modify the Input Data to 64pts "1"s + 32pts "1". Leave 32pts "0".

e.g. 96 pts "1"s plus 32 pts "0".

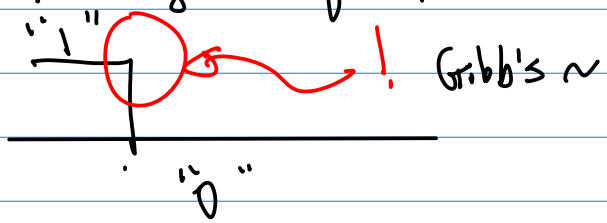
plot the P(m). plot.2

Observation: plot1 (64pts "1"s) has more Higher Frequency Components. OR, more precisely, more energy in the higher Frequency Range.

Create 3rd Data set, 96+16 = 112 "1"s.

↓

fewer higher Frequency Comp.

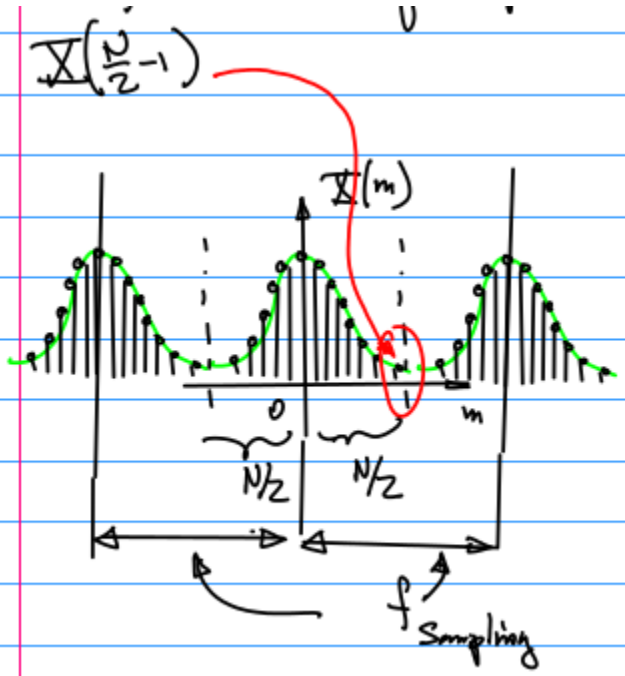


Remark: The Signal Energy Distribution in the higher frequency Range can be demonstrated with These 3 plots. Therefore the Sampling frequency for each of them satisfies the following condition

$$f_{Sampling_1} \geq f_{Sampling_2} \geq f_{Sampling_3} \quad \ldots (2)$$

Ref → pp25

2022S-101-note-part2-cmpe242-2022-05-9.pdf



Note1: The $f_{sampling}$ sets Apart of each Period of the P(m);

Note2: In The Higher Frequency Range, the energy distribution of Non-zero P(m) Contributes to "Aliasing".

Note3: To Validate ADC Data, we define the following index

$$\eta = \sum_{m \in \Omega_1} P(m) \Big/ \sum_{m=0}^{N-1} P(m) \quad \ldots (5)$$

Where Ω_1 : higher frequency Range.
Such as

$$N_{Low} \le m \le N/2 \quad \ldots (6)$$

↑ user defined Lower Bound of the high Freq. Range.

Example, for N = 128. N/2 highest Freq. Index, $\frac{N}{2} = 64$, then depending on the Application We can have $N_{Low} = 50$

$$2 \sum_{m=50}^{64} P(m) \quad \ldots (7)$$

May 1st (Monday)
Example: Compute power spectrum for the data validation using N=4;
Sol:

From the power spectrum
P(0), P(1), P(2), P(3) for the entire period N;

Suppose we want validate the date based on equation (5), assuming the frequency index m = 1, and 2;

eta M <= 20%

Hence,
The total energy = P(0) + P(1) ... + P(3);
(please perform the calculation off line)

The energy in the higher frequncy range:

P(1) + P(2) + P(3)

Then the ration eta:

$$\text{eta} = \frac{P(1) + P(2) + P(3)}{P(0) + P(1) + P(2) + P(3)}$$

Comparing this result to the limit set by the application requirement 20%

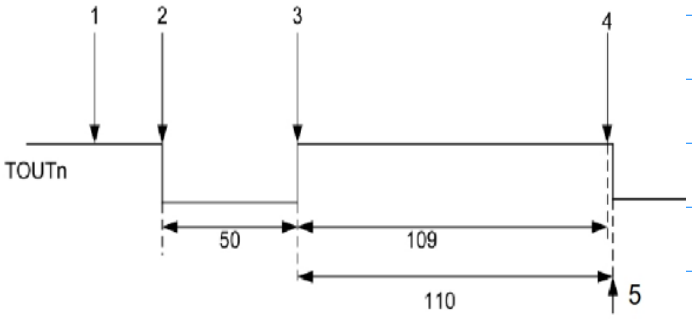
If not, then increase the sampling frequency per Nyquest Theorem.

Now consider the last part of the class, PWM and ADC Archietecture.

Example:
Ref. Timing/SPRs for PWM

/ 2022S / 2022S-107e-pwm-waveform-v3-2018-3-4.jpg

PWM Operation



FromSamsung ARM11 data sheet;

Ref: pp. 1

2022S-101-note-part2-cmpe242-2022-05-9.pdf

Ref: Samsung ARM 11 CPU datasheet

2021F-105-#0-cpu-a... Add files

Using the example from the lecture note, pp. 1

32 PWM TIMER

This chapter describes the functions and usage of PV

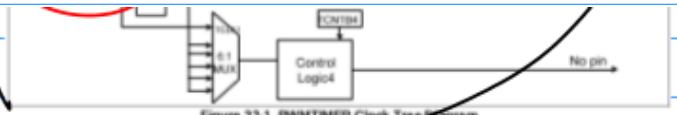


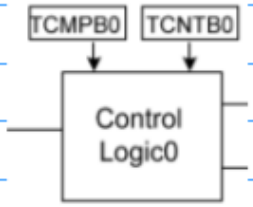
Figure 32-1. PWMTIMER Clock Tree Diagram

a. Input: PCLK peripheral clock.
1/2, 1/4, 1/8, ... of the System Clock

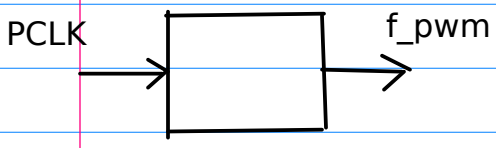
b. Output (2 outputs)

d. Special Purpose Registers

CONF (Configuration Register)
CNT (Control — "Count")
CMP (Comparision)
→ define f_{pwm} & Duty Cycle.



.



$T = 1/f_{PWM} \cdots (2)$, $f_{PWM} = \frac{PCLK}{(Prescaler+1)DIV} \cdots (3)$

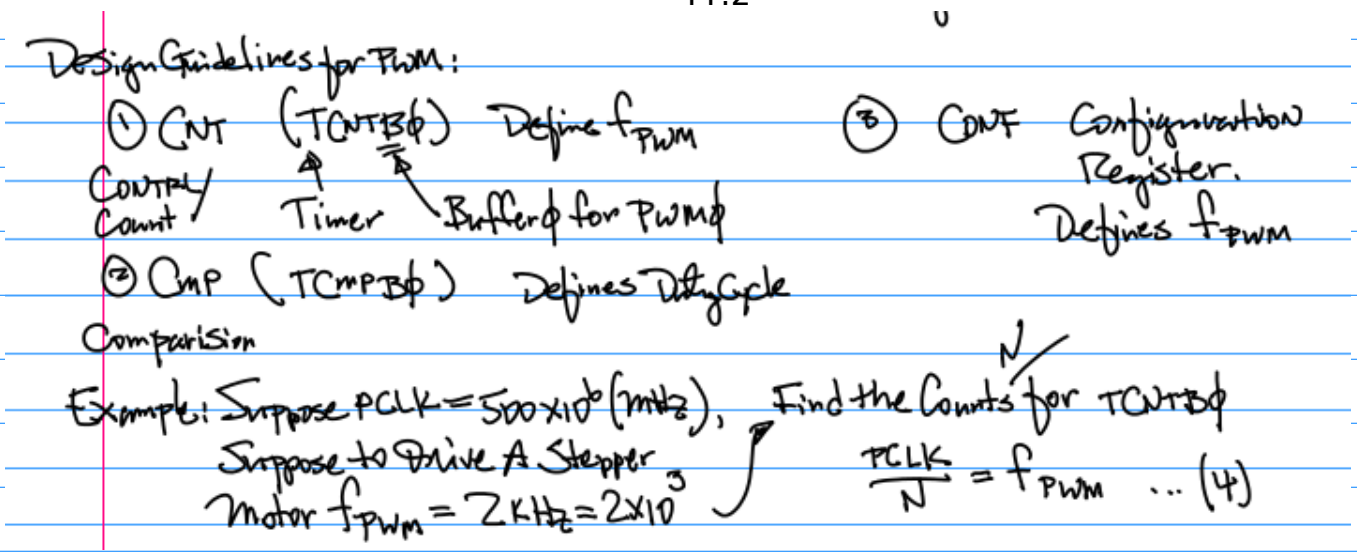
Frequency = PCLK / ({prescaler value + 1}) / {divider value}

Suppose PCLK = 1/4 (System Clock) = 1/4 (800 x 10^6)

= 200 x 10^6

Defined By Special Purpose Register.

PP.2

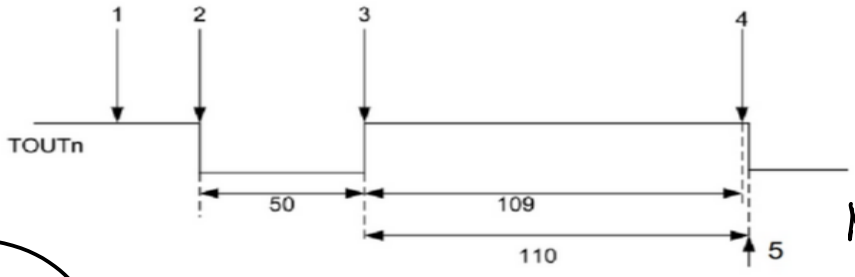


2022S / 2022S-107e-pwm-waveform-v3-2018-3-4.jpg

Five steps for defining (1) frequency of the PWM output; (2) the duty cycle;
Note 1: CNT is responsible for defining f_pwm, the calculation of N, e.g., the number of counts is given in Equation (4) on this pape.
Note 2: f_pwm is achieved by counter down counting from N, once it reached to 0, then interrupt is triggered and then repeating the process, so f_pwm is achieved.
Note 3: for duty cycle definition, we compute 2 parts, the first one is the calculation of the counts number from the percentage of the desired duty cycle. Example: 69% duty cycle, 69% * N, if N in the waveform example is 159, then M = 109;

In summary, the 5 steps process of setting up f_pwm and duty_cycle is given below:

PWM Operation



Note 1: $N = \frac{PCLK}{f_{PWM}}$ ... (1)

Note 2: TCFG0

Note 3: TCMPB0 $M = \left(\frac{\%\ of\ Duty}{Cycle}\right) \cdot N$ ... (2)

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 cycl

May 3rd (Wed).
Example: Continuation of PWM.

Ref: Special Purpose Register

2022S / 2022S-107f-pwm-specialPurposeRegister-v3-2018-3-4.jpg

TCFG0, TCNTBn and TCMPBn

SFR, see C code Next Page.

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 Regi

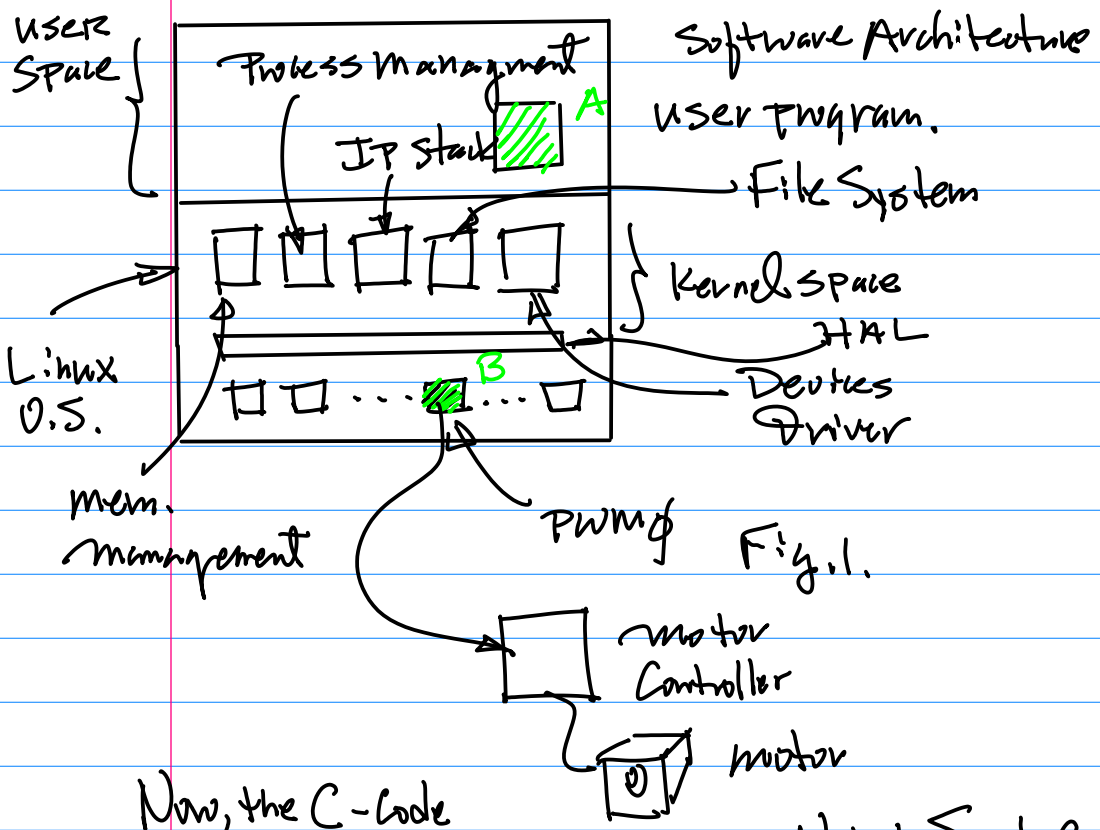
Register	Offset	R/W	Description
TCMPB0	0x7F006010	R/W	Timer 0 Compare Buffer Register

Harry Li, Ph.D.

Example: PWM Device Driver

```
harry@harry-laptop: /opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char
hpet.c                          mwave                       synclink.c
hw_random                       mxser.c                     synclink_gt.c
i8k.c                           mxser.h                     synclinkmp.c
ip2                             nozomi.c                    tb0219.c
ipmi                            nsc_gpio.c                  tlclk.c
isicom.c                        nvram.c                     toshiba.c
istallion.c                     nwbutton.c                  tpm
Kconfig                         nwbutton.h                  ttyprintk.c
lp.c                            nwflash.c                   uv_mmtimer.c
Makefile                        pc8736x_gpio.c              viotape.c
Makefile-backup                 pcmcia                      virtio_console.c
mbcs.c                          ppdev.c                     vme_scc.c
mbcs.h                          ps3flash.c                  xilinx_hwicap
mem.c                           ramoops.c
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char$ ls
 mini64*
mini6410_adc.c              mini6410_hello_module.mod.c  mini6410_leds.o
mini6410_adc.mod.c          mini6410_hello_module.mod.o  mini6410_pwm2.c
mini6410_adc.o              mini6410_hello_module.o      mini6410_pwm2.mod.c
mini6410_buttons.c          mini6410_leds.c              mini6410_pwm.c
mini6410_buttons.o          mini6410_leds.ko             mini6410_pwmHarry.c
mini6410_hello_module.c     mini6410_leds.mod.c          mini6410_pwm.mod.c
mini6410_hello_module.ko    mini6410_leds.mod.o
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char$
```

PWM Driver illustrated in Fig. 1 as B.



Now, the C-Code for Part A & B.

Note 1. Sample Code, Driver Program.

harry@harry-laptop: /opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char

```
32 #include <mach/gpio-bank-e.h>
33 #include <mach/gpio-bank-f.h>
34 #include <mach/gpio-bank-k.h>
35
36 #define DEVICE_NAME     "pwm"
37
38 #define PWM_IOCTL_SET_FREQ              1
39 #define PWM_IOCTL_STOP                  0
40
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 )  //Harry: add "duty"
49 {
50         unsigned long tcon;
51         unsigned long tcnt;
52         unsigned long tduty; //Harry
53         unsigned long tcfg1;
54         unsigned long tcfg0;
                                                    49,1          19%
```

The code with Duty Cycle Added.

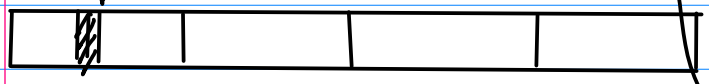
Note 2:

Special Purpose Register, see CPU Datasheet

```
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers$ c
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/ch
 mini64*.c
mini6410_adc.c          mini6410_hello_module.mod.c  mini6410_pwm2.mod.c
mini6410_adc.mod.c      mini6410_leds.c              mini6410_pwm.c
mini6410_buttons.c      mini6410_leds.mod.c          mini6410_pwmHarry.c
mini6410_hello_module.c mini6410_pwm2.c              mini6410_pwm.mod.c
```

Note: Duty Cycle functionality is added by this code

```
60
61          tmp = readl(S3C64XX_GPFCON);
62          tmp &= ~(0x3U << 28);
63          tmp |=  (0x2U << 28);
64          writel(tmp, S3C64XX_GPFCON);
```



3U → 11 2Bit
"~" Negate to 00;
then "&=" AND to clear/reset the 2 bits.
2 = 10 (Binary). "|=" OR to the 2 bits at that location.

```
66          tcon = __raw_readl(S3C_TCON);
67          tcfg1 = __raw_readl(S3C_TCFG1);
68          tcfg0 = __raw_readl(S3C_TCFG0);
69
70          //prescaler = 50
71          tcfg0 &= ~S3C_TCFG_PRESCALER0_MASK;
72          tcfg0 |= (50 - 1);
73
74          //mux = 1/16
75          tcfg1 &= ~S3C_TCFG1_MUX0_MASK;
76          tcfg1 |= S3C_TCFG1_MUX0_DIV16;
77
78          __raw_writel(tcfg1, S3C_TCFG1);
79          __raw_writel(tcfg0, S3C_TCFG0);
```

Note 1: CPU Datasheet

CMPE244 / 2021F-105-#0-cpu-arm11-2018S-29-CPU_S3C6410X.pdf

Note 2: SPI's.

32.4 SPECIAL FUNCTION REGISTERS

32.4.1 REGISTER MAP

Register	Offset	R/W	Description
TCFG0	0x7F006000	R/W	Timer Configuration Register 0 that confi two 8-bit Prescaler and DeadZone Lengt
TCFG1	0x7F006004	R/W	Timer Configuration Register 1 that contr and DMA Mode Select Bit
TCON	0x7F006008	R/W	Timer Control Register
TCNTB0	0x7F00600C	R/W	Timer 0 Count Buffer Register
TCMPB0	0x7F006010	R/W	Timer 0 Compare Buffer Register
TCNTO0	0x7F006014	R	Timer 0 Count Observation Register

PP1118. Note1: 32bit SPR; 2. 8-bit Prescaler for Each PWM

$$f_{PWM} = \frac{PCLK}{(Prescaler+1) * Div}$$

TCFG0	Bit	R/W	Description	Initial State
Reserved	[31:24]	R	Reserved Bits	0x00
Dead zone length	[23:16]	R/W	Dead zone length	0x00
Prescaler 1	[15:8]	R/W	Prescaler 1 value for Timer 2, 3 and 4	0x01
Prescaler 0	[7:0]	R/W	Prescaler 0 value for timer 0 & 1	0x01

PCLK → f_{PWM} → N, → M → Perform Init & Config. Device Driver User Program

for TCNTBp　　for TCMPBp

Note3. TCNTBp, for f_{PWM}, and for Duty Cycle. $M = \left(\frac{\%\ of\ the\ Duty\ Cycle}{}\right) * N.$

```
81      clk_p = clk_get(NULL, "pclk");
82      pclk  = clk_get_rate(clk_p);
83      tcnt  = (pclk/50/16)/freq;
84      tduty  = tcnt*duty;      //Harry: duty is x% of the period, duty cycle
```

Now, to Add Duty Cycle Control Based on Line #38

```
37
38 #define PWM_IOCTL_SET_FREQ      1
39 #define PWM_IOCTL_STOP          0
40
```

→ #define PWM_IOCTL_SET_DUTY 1