August 23rd (Mon).

CMPE240 Section 1.

CMPE240

Harry Li.
E-mail: hual@sjsu.edu
Text message (650) 400-1116

Office Hours: M.W. 3:40-4:40 pm.

Advanced Microprocessor Systems = Prototype System with A CPU module

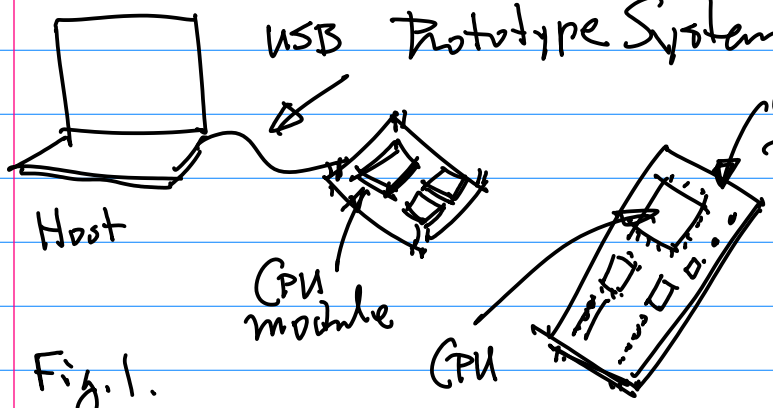


Fig. 1.

GPU (Graphics Processing Unit), Array of Processors, Machine Learning, AI. Autonomous Systems. Nvda Jetson TX2.

Textbooks, References

1° NXP LPC1769 CPU Datasheet 800+ pages Homework: Down Load pdf. Before Next Monday. Aug. 30th.

2. LPC1769 schematics of the CPU module

3. Nvda Jetson NANO Datasheet on TX2 (6 CPU + 256 GPU) 400+ pages. 5% Bonus. (optional)

4. RISC-V. Open Source Architecture, A super set of ARM. FPGA, Verilog. SoC. +RTOS. (optional) +5%

A proposal (One Paragraph) By Sept. 1st (Wed). submit to my Email;

Note: Buy LPC1769 CPU module.

digi-key.com, mouser.com, etc.

Grading Policy & Projects

2 projects (phase I & II)

2D Graphics Processing Engine → 3D Graphics Processing Engine } a le

2D Graphics Processing Engine → 3D Graphics Processing Engine { a. 3D shadow. 10% = 30pts, 30%.
b. Diffuse Reflection 10%
10%

midterm: 30%, Final 40% (Comprehensive)

Option 1. (5%+) NVDA NANO
{ a. Likely Devices Drivers, O.S. C/C++, Python.
b. I/O Interface: "EdgeAI" GPIO, SPI.

Option 2. (5%+) RISC-V Target SoC, FPGA Board,

Proposal (one paragraph), Submission By Sept 1st (Wed) via E-mail.

Policy ON Project Submission.
1°. Form 3-4 person Team.
2° No Source Code/Design material Can be Copied; All Course material has to be completed individually;
3° Late Project, 10% per week;

August 25 (Wed)
Today's Topics:
1° Bill of material
Reference: github/hwalili/Cmpe240/2018F

The B.O.M.
1. CPU module NXP LPC1769
↓
3rd Party (DigitalArt), module
To Distributors
Digikey.com, Mouser.com etc. Expecting Delays. Lead Time over 8 weeks
Alternative { Re-use the previously used module
Team (4 person)

Each person will need to have his/her Board;
Option 1: NANO. ≈ 4400+ pages Datasheet "firmware"
↓ Jetpack 4.3 or Higher
(O.S. + Libs. + Packages)

Tool for Flashing the CPU module

Cmpe240

c. Coding in Both user & kernel spaces. → O.S. Distr.
Toolchain, Device Driver Debugging & Development;

Option 2. RISC-V. verilog, FPGA.

2. Power Regulator IC. such as
7812, 7805 ... 1117

78xx = 7805

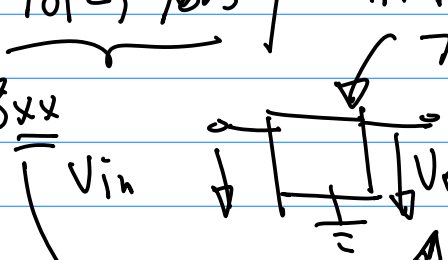


Fig.1. Fig.

"05": 5.0 VDC, "12": 12VDC

$V_{in} \geq V_{out} + 1.5\,VDC$ ... (1)

DC Voltage Source

a. About 7805
1000 mW.

b. 7.5VDC
OR
9. VDC

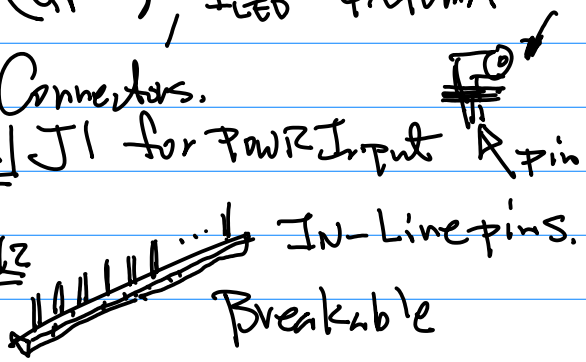
@ 1000mW + 500 mW
= 1500mW
Current Rating.

c. Why Do we use it?
↳ Deploy the System.

3. "Glue" Components / Resistors a.

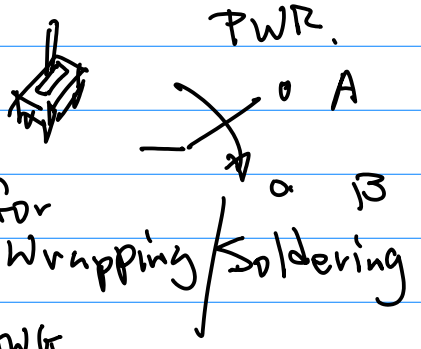
3.
c. LEDs. (Red, Green) for Debugging purpose, for PWR. (GPIO), I_{LED} = 4mA/10mA

d. Connectors.
d1 J1 for PWR Input



d2 to mount CPU module.

e. Switch. S/W1: to toggle PWR.
S/W2



f. Wire for Wire Wrapping / Soldering
28-30 AWG

4. Color LCD Display module
a. SPI (Serial Peripheral Interface)
b. Software Graphics (Driver) C/C++ Lib.
to Activate/Initialize LCD.
MCU Xpresso (I.D.E.)
↓
S.T. Lib.

5. "Other" thing.
RJ-45 Connector

Sept. 8 (W)

Topics: 1° "Hello, the world" program

- Hardware Implementation
  - LPC MCU Xpresso.
    - a. Installation of MCU Xpresso.
    - b. github.com/hualili/CMPE240/2018F
      LPC1769 patch, Import this patch to your Xpresso.
  - Prototype Board Build Up
    - a. Wire Wrapping Board (LPC/NANO/Pie)
    - b. Stand-offs.
  - External Power CKT (Red LED should be included)
  - GPP Testing CKT
    - Implementation/Design of the CKT.
    - Architecture Aspects, CPU Architecture, M. Map.

https://github.com/hualili/CMPE240-Adv-Microprocessors/blob/master/1769%20patch.zip

Note: Wire Wrapping Board with "Stand-off's (legs)

Homework: Next Show-and-tell Wire Wrapping Board;

NANO/pie

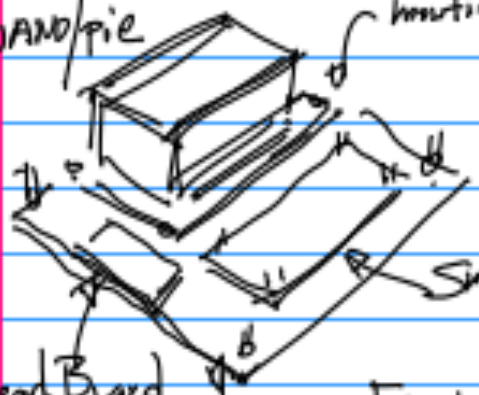


Fig. 1

Wire Wrapping Board "Carrier" Board

"Tap Plastic"

On the Board: a. Stand-offs.

b. Connector(s) for External PWR

→ PWR IC (7805), with Red LED

CPU Architecture:

1. 32-bit Architecture

CPU Architecture

a. ALU 32 bit Arithmetic/Logic Unit.

b. Register "File",

A Bank of Registers. 32 bits GPRs

General Purpose Registers

Those Registers that can participate Any meaningful

Arithmetic/Logic Operations.
Special-Purpose Registers.
SPRs 32 bit

Naming Convension: 6 letters

Common Design for SPRS:

1° Control Register(s) per Each Peripheral Controller
CON
Root (3 Letters)

2° Data Register, DAT

3° Pull-up/Down (Electric Characteristics)

4. Data Bus, "Bi-Directional" 32 bits
Information Flowing Both Directions.
Address, "Uni-directional" from CPU to the Outside. 32 bits

Notation: 32 bit Register
$GPR_X[31:0]$

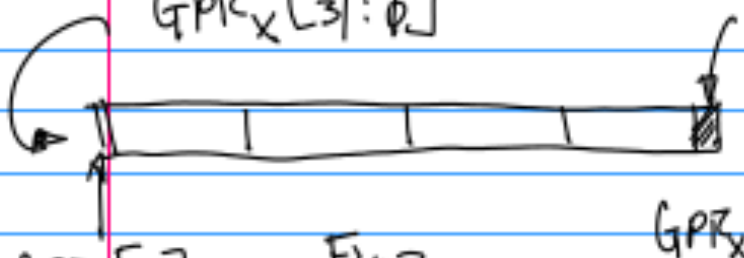


Fig. 2

For Address Bus, $Addr[31:0] =$
$a_{31} a_{30} \cdots a_1 a_0$

To Define/Determine the Behavior of Peripheral Controllers.

Note: "Little Endian"
LSB is a 0,

2. "Byte Addressable" machine
is a machine whose smallest memory cell with an unique address is a single Byte.

Total memory:
$2^{32} = 2^2 \cdot 2^{10} \cdot 2^{10} \cdot 2^{10}$ ...(1)
$2^{10} = 1K$, ...(2) $2^{20} = 2^{10} \cdot 2^{10} = 1M$ ...(3)
$2^{30} = 1M \cdot 1K = 1$ Gigan ...(4)
$2^{32} = 2^2 \cdot 2^{30} = 4 \cdot GB$

3. Memory Map.

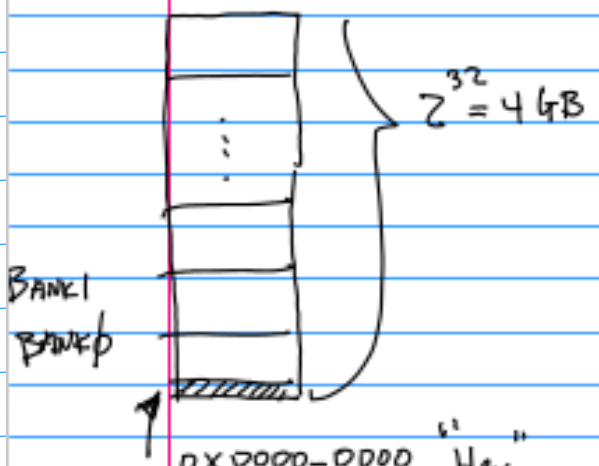


Fig 3.

32 bits for the Address
8 bits for this memory

a. PWR-up Address:
CPU will fetch the 1st Executable from this memory Location.
→ 0X0000-0000 for ARM

Note: for x86, the PWR-up Address: 0XFFFF_FFF0

b. BANKS. $2^{32}/8 = 2^{32}/2^3$
$= 2^{29} = 2^9 \cdot 2^{20} = 512$MB

How many Bits Do we need to uniquely define Each Bank?
3 bits → $a_{31} a_{30} a_{29}$

Define Starting Addr. of Each Bank:

a_{31}	a_{30}	a_{29}	a_{28}
0	0	0	⋮
0	0	1	⋮
0	1	0	⋮
⋮	⋮	⋮	⋮
1	1	1	⋮

BANK0
BANK1
BANK2
⋮
BANK7

Write the Address for Each Bank. "Starting" (32 bit)

For BANK0 : 0X0000_0000
BANK1 : 0X2000_0000
" 2 : 0X4000_0000

Example: CPU Datasheet pp. 13.
GPIO 0X2009_C000

a. Collection of SPRs are mapped to here, e.g. Addr. for SPRs are mapped to here

b. Which memory Bank holds this GPIO? BANK1 whose starting Address is 0X2009_C000

Sept 13 (Mon)

1° Today's Topics: Integrate Architecture Discussion with Software Development IDE. Objectives: To write first C program for testing purpose

Example: Starting from CPU memory map → 8 BANKS (pp13)

→ Populate 8 BANKS

1st 256 KB = Flash 0x0000_0000,

Rest of the Banks, such as mem. controller, Peripheral Controller

↓

Peripheral controllers on the mem. map. APB0 & APB1 → SSP1 (SPI) Peripheral Controller

↓

0x4003_0000 Size 4000

SPRs. ← "CON" 3 Letter

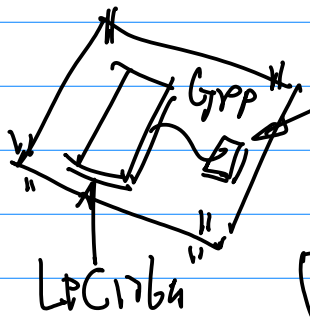
a = Naming convention: Prefix (Two) + Postscript

3 Letters 3 Letters 3 letter

"SPICON" → "SPICON001" for Example → C Compiler / C Code

b = Definition: Are those SPRs for the init & Config of a Peripheral Controller.

Example. GPP



Output Testing CKT Turn ON/off LED pin16

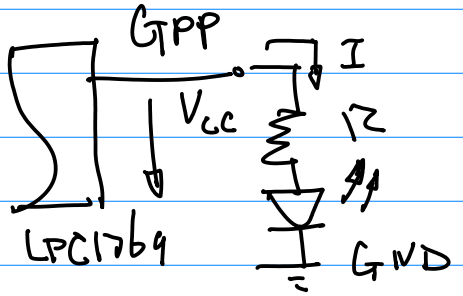


Fig 1.

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

$I \simeq 8mA$, → $R \simeq 2V / 8mA \simeq 300\Omega$

$V_{LED} \simeq 1.8VDC$

GPPCON

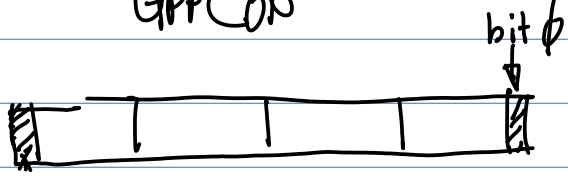


Fig. 2

Where to find GPPCON on the memory map? → Addr. of GPPCON is described on CPU Datasheet.

CmpE240

2^{32} Possible Combinations of Init & Config. Feature

Reference: 1° 2021F-105 ~ 10 ON GPIO

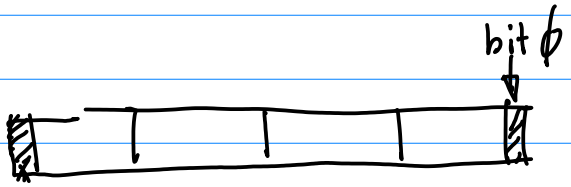
C-Code for Init & Config is required

Example: Make GPP for Input & Output Testing.

{ Hardware
{ Software { NXP MCU Xpresso
Import GPP Sample "zip"

GPP (General Purpose Port)
32 pins, Define pin 16 as output pin.

we have to use the following init + Config pattern:



0x F2 00_FF 00

To make pin 16 as an output.

First, "port" the Architecture Compiler to the target

#define GPPCON

Addr from the memory

then, Copy 0x F2 00_FF 00 into this memory Location.

Homework: Show + Tell
By Next Week Installation of MCU + Import LPC1769.

Sept 15 (W) Architecture
Today's Topics: GPIO Design

Design Step 1.
Identify/select GPP/GPP-pins
P0.2, P0.3
(Connector → CPU → Selection
DataSheet

Step 2. Define P0.2 Output, P0.3 As Input.
Design the Hardware

Step 3. SPRs (Special Purpose Registers) for the GPP peripheral Controller

Connector J2-21 ... → CPU pin P0.2 P0.3 → CPU DataSheet → SPRs

Note: SFRs commonly defined/utilized are

GPx CON

where x = A, B, C, D, ...

GPF CON

GPF DAT (32 bits → 32 pins)

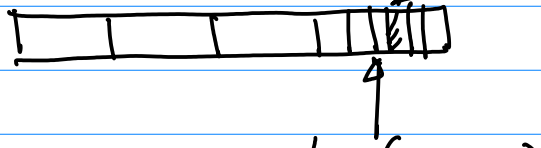
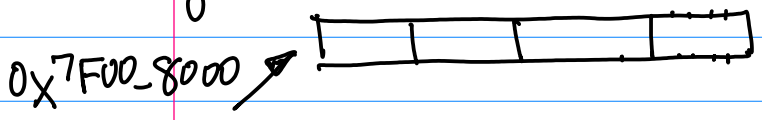


Fig.1

Find Table on CPU Datasheet to to define P0.2 output, P0.3 as input.

Example, Samsung Arm11 Datasheet pp312

Fig2



Question: Define Binary Pattern for GPA CON to make its pin 2 as an output?

Sept. 20 (Mon) Topics: 1° GPIO SFRs, IDE, Sample Code; 2° 2D G.E.

Example: git (class) 2021F-105 — GPP...

1. Naming Convention in C Compiler

```
LPC_GPIO0->FIODIR

LPC_GPIO0->FIOSET

LPC_GPIO0->FIOCLR
```

Target CPU (Family)

Peripheral Controller

Special purpose Register

From the Example, git, 2021F-105

From CPU datasheet, GPIOs are configured using the following registers:
1. Power: always enabled.
2. Pins: See Section 8.3 for GPIO pins and their modes.
3. Wake-up: GPIO ports 0 and 2 can be used for wake-up if needed, see (Section 4.8.8).
4. Interrupts: Enable GPIO interrupts in IO0/2IntEnR (Table 115) or IO0/2IntEnF (Table 117). Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.

Chapter9, pp129

PINSEL[5:4] for P0.2

PINSEL[5:4] = 00 for GPP P0.2

PINSEL[5:4] = 01 for UART TX

pp133 FIODIR Example.

P0.3 output, Find SPR?

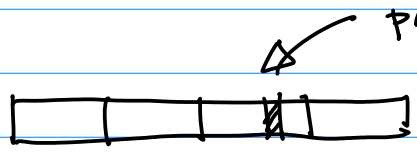
Define bit value for the Output

Table Lookup.

FIO0DIR0

FIOSET, CPU Datasheet Lookup.

```
void GPIOinitOut(uint8_t portNum,
uint32_t pinNum)
{
if (portNum == 0)
{
LPC_GPIO0->FIODIR |= (1 <<
pinNum);
}
else if (portNum == 1)
{
LPC_GPIO1->FIODIR |= (1 <<
pinNum);
```



1 << pinNum // set Direction to pinNum

Logic Operation { "|=" Bitwise "OR" "&=" ?

Example: Set pin

```
void setGPIO(uint8_t portNum,
uint32_t pinNum)
{
if (portNum == 0)
{
LPC_GPIO0->FIOSET = (1 <<
pinNum);   //1 as output
printf("Pin 0.%d has been set.\
n",pinNum);
}
```

Turn ON LED ("output 1")

Example: Clear the pin

```
void clearGPIO(uint8_t portNum, uint32_t
pinNum)
{
if (portNum == 0)
{
LPC_GPIO0->FIOCLR = (1 << pinNum);
printf("Pin 0.%d has been cleared.\n",
pinNum);
}
```

$\vec{P}(x,y)$ Notation

$\vec{P}(x,y) = (x,y)$

$\vec{P}_i \rightarrow \vec{P}_i(x_i,y_i) \rightarrow (x_i,y_i)$

point(s), Vertex, Vectors

$\vec{P}_i = \vec{P}_i(x_i,y_i) = (x_i,y_i)$

Formulation for a straight line

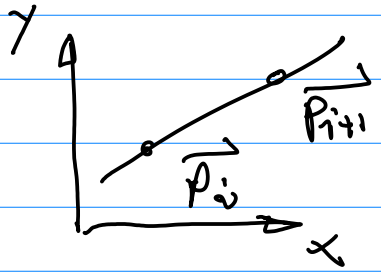


Fig.2

Now, 2D Vector Graphics

$\vec{P}(x,y)$ a point, vertex, a vector

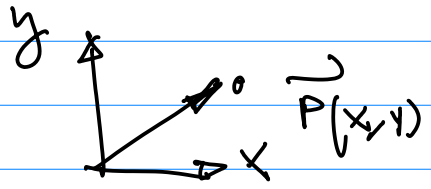


Fig.1

Sept. 22 (W)

Graphics Engine { 2D G.E. / 3D G.E. / Graphics Display Driver (Hardware)

Consider 2D G.E.

- Theory/Formulation
- Algorithms/Software
- Implementation/Hardware Prototyping

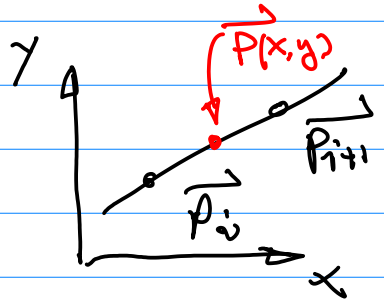


Fig. 2

Note: Need 2 points $\vec{P}_i$, $\vec{P}_{i+1}$ to define a line

Let's define a direction vector

$$\vec{d}(x_d, y_d) = \vec{P}_{i+1} - \vec{P}_i$$

$$= \vec{P}_{i+1}(x_{i+1}, y_{i+1}) - \vec{P}_i(x_i, y_i) \quad \dots (1)$$

Example: Suppose given a starting pt. $\vec{P}_i(x_i, y_i) = (3, 4.5)$

$\vec{P}_{i+1}(x_{i+1}, y_{i+1}) = (5.5, 6.3)$

Find direction vector?

Sol. By Eqn(1), we have

$$\vec{d}(x_d, y_d) = \vec{P}_{i+1} - \vec{P}_i$$

$$= ((x_{i+1}, y_{i+1}) - (x_i, y_i))$$

$$= (x_{i+1} - x_i, y_{i+1} - y_i)$$

Sub. the given condition into the directional vector, we have

$$\vec{d} = ((5.5-3), (6.3-4.5))$$

$$= (2.5, 1.8),$$

In C/C++ Coding, we use the following Equation, From Eqn (1), we have

$$\vec{d}(x_d, y_d) = (x_{i+1} - x_i, y_{i+1} - y_i) \quad \dots (1b)$$

OR,

$$\begin{cases} x_d = x_{i+1} - x_i \\ y_d = y_{i+1} - y_i \end{cases} \quad \dots (1c)$$

direction_x = x[i+1] - x[i];
direction_y = y[i+1] - y[i];

Let's uniquely define a line Need a pt $\vec{P}_i$, or $\vec{P}_{i+1}$; and directional vector

$$\vec{P}(x,y) = \vec{P}_i + \lambda(\vec{P}_{i+1} - \vec{P}_i) \quad \dots (2)$$

Starting pt, scalar, Directional vector

Let $\lambda = 0$, then $\vec{P}(x,y) = \vec{P}_i(x_i, y_i)$

Starting pt.

$\lambda = 1$, then

$\vec{P}(x,y) = \vec{P}_{i+1}(x_{i+1}, y_{i+1})$

$0 < \lambda < 1$, Any point $\vec{P}(x,y)$ Between $\vec{P}_i$ and $\vec{P}_{i+1}$.

$\lambda > 1$ Any pt: $\vec{P}(x,y)$ Beyond $\vec{P}_{i+1}(x_{i+1}, y_{i+1})$.

$\lambda < 0$, Any point Beneath $\vec{P}_i(x_i, y_i)$.

Fig3a

Fig.3b

Screen Saver Design for LPC 2D I.E.

Rotating Squares And Trees.

Example: Design A Rotating Squares

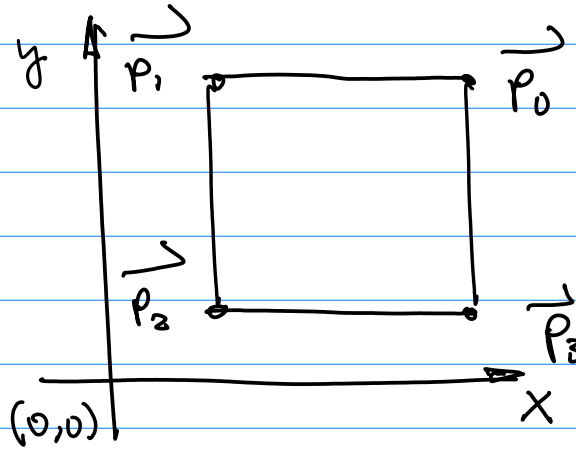
Step1. Define 4 vertices/pts

$\vec{P}_i$, $i = 0, 1, 2, 3$

$\vec{P}_0(x_0, y_0) = (60, 60)$, $\vec{P}_1(x_1, y_1) = (10, 60)$

$\vec{P}_2(x_2, y_2) = (10, 10)$, $\vec{P}_3(x_3, y_3) = (60, 10)$

Based on the physical display device



Note: Be sure to Arrange $\vec{P}_i$ in a Counter Clockwise direction. (for Later 3D Hidden Line/Surface Removal)

Step2. use

$$\vec{P}(x,y) = \vec{P}_i(x_i, y_i) + \lambda\left(\vec{P}_{i+1}(x_{i+1}, y_{i+1}) - \vec{P}_i(x_i, y_i)\right)$$

Purpose: LCD Soldering ON
the Wire-Wrapping Board,
Input Single line
Drawing project.