CMPE240
Section 1.

August 23rd (mon).

CMPE240

Harry Li.

E-mail: hua.li@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

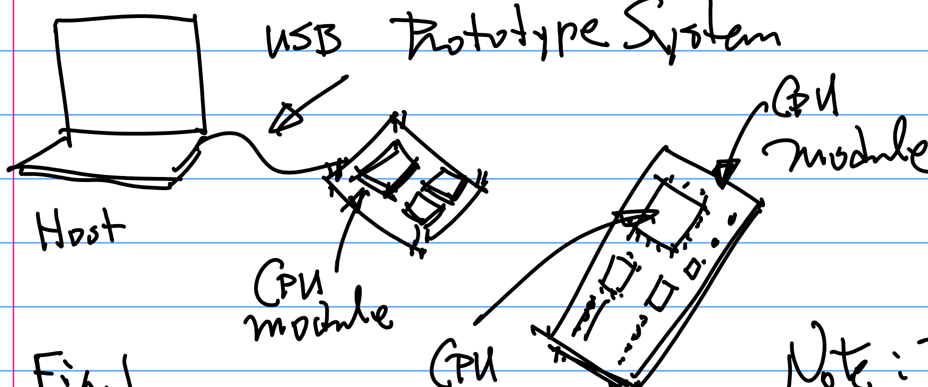


Fig. 1.

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

Text Books, 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) 4000+ pages. (optional) 5% Bonus.
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, b

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

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/hualili/Cmpe240/2018F

The B.O.M.

1. CPU module NXP LPC1769
↓
3rd Party (Digital Art), 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. a 4400+ pages "firmware" Datasheet

b 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.

Tool chain, Device Driver Debugging & Development;

Option 2. RISC-V. verilog, FPGA.

2. Power Regulator IC such as

7812, 7805 / ... 1117

78xx

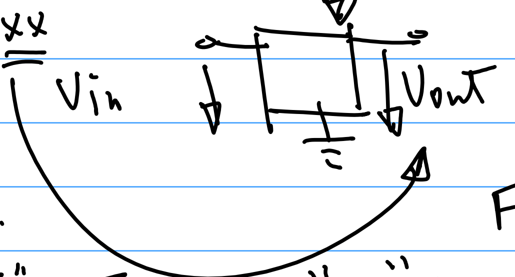


Fig.1.

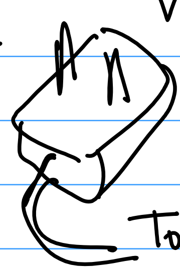
Fig.

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

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

DC Voltage Source

Wall mount Adaptor



To Vin of 7805

a About 7805 1000 mW.

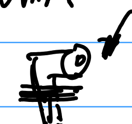
b 7.5VDC OR 9.VDC @ 1000mW + 500 mW = 1500mW

c Current ~~Reading~~ Rating

Why Do we use it? → Deploy the System.

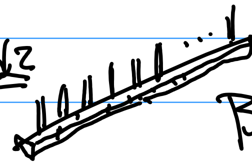
c LEDs. (Red, Green) for Debugging purpose, for PWR. (GPIO), $I_{LED} = 4 \sim 10\,mA$

d Connectors.



d1 J1 for PWR Input 2 pin

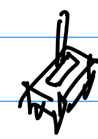
d2

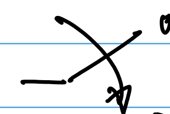


IN-Line pins. Breakable to mount CPU module.

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/Interface LCD.

MCU Xpresso (I.D.E.) → S.J. Lib.

5. "Other" thing.

RJ-45 Connector

3. "Glue" Components / Resistors a

Sept. 8 (W)

Topics: 1° "Hello, the world" program

Hardware Implementation
- NXP MCU Xpresso.

Prototype Board Build Up
- a. WireWrapping 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.

a. Installation of MCU Xpresso.

b. github hualili/CMPE240/2018F

LPC1769 patch, Import this patch to your Xpresso.

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

Note: WireWrapping Board with "Stand-off's" (Legs)

Homework: Next Show-and-tell

WireWrapping Board;

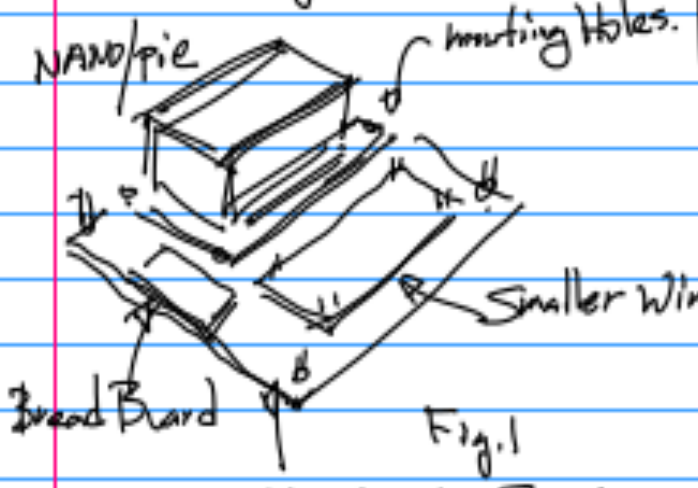


Fig. 1

"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. To Define/Determine the Behavior of peripheral Controllers.

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)

C. Data Bus, "Bi-Directional" 32 bits

Information Flowing Both Directions.

Address, "Uni-directional" from CPU to the Outside. 32 bits

Notation: 32 bit Register

$GPR_x[31:0]$

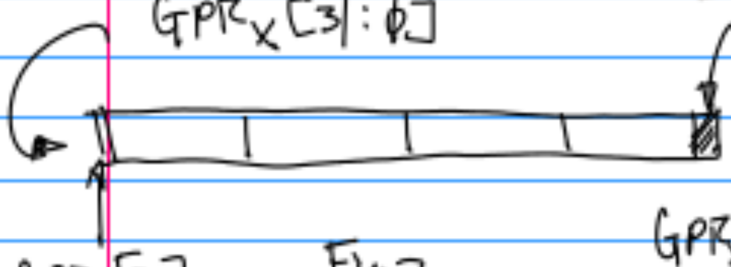


Fig.2

For Address Bus, $Addr[31:0] =$

$a_{31} a_{30} \cdots a_1 a_0$

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)

$2^{30} = 1M \cdot 1K = 1$ giga ...(4)

$2^{32} = 2^2 \cdot 2^{30} = 4$ GB

3. Memory map.

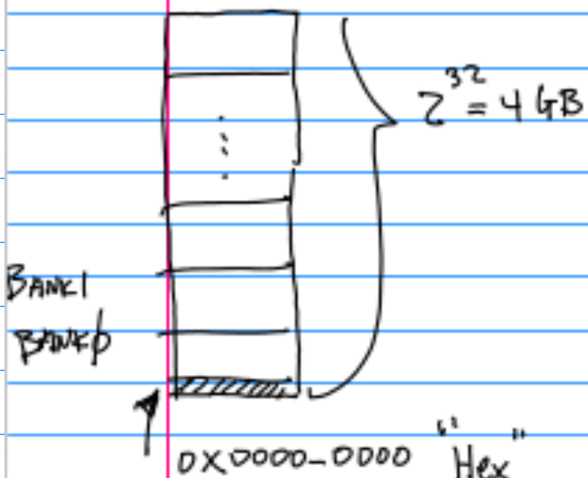


Fig 3.

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. $2^{32}/8 = 2^{32}/2^3$

$= 2^{29} = 2^9 \cdot 2^{20} = 512MB$

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		BANK0
0	0	1		BANK1
0	1	0		BANK2
				⋮
1	1	1		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 Devices, such as mem. Controller, Peripheral Controller

↓

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

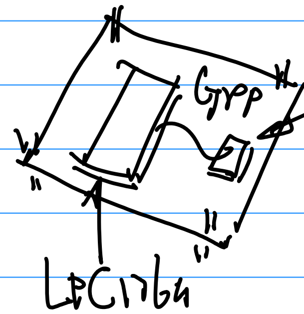
↓

S.P.Rs. ← 0X4003_0000
"CON" 3 Letter Size 4000

a. Naming conversion Prefix + Root + 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



pin 16

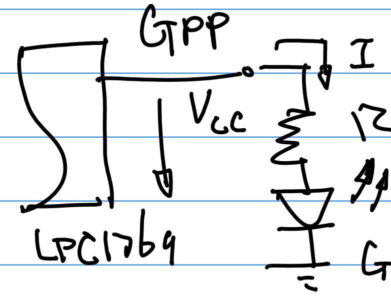


Fig 1.

$$V_{CC} = IR + V_{LED} \quad \cdots (1)$$

$I \simeq 8mA$, → $R \simeq 200\Omega$
300 Ω

$V_{LED} \simeq 1.8 VDC$

GPPCON

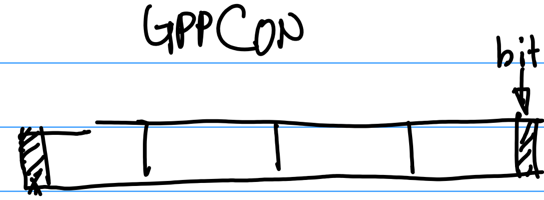


Fig. 2

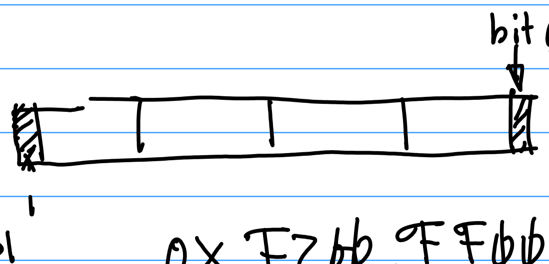
Where to find GPPCON on the memory map? → Addr. of GPPCON is described on CPU Datasheet.

CmpE240

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

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

we have to use the following init + Config pattern:



0X F200_FF00

To make pin 16 as an output.

First, "port" the Architecture Compiler to the target

#define GPPCON Addr. from the memory

then, Copy 0XF200_FF00 into this memory Location.

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

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

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"

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

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 Data Sheet → SPRs

CMPE240

11

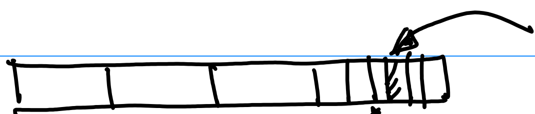
Note: SPRs commonly defined/ utilized are

GPx CON

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

GPFCON

GPFDAT (32 bits → 32 pins)



P0.2 (J2-21)

Fig. 1

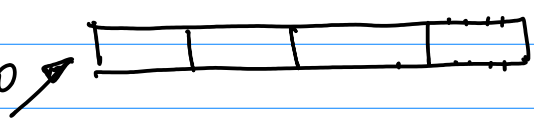
P0.3 (J2-22)

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

Example, Samsung Arm11 Datasheet pp312

Fig 2

0x7F00_8000



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

Sept. 20 (Mon) Topics: 1° GPIO SPRs, IDE, Sample Code; 2° 2D GE.

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

CmpE240

12

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.

Chapter 9, pp 129

PINSEL[5:4] for P0,2

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

PINSEL[5:4] = 01 for UART Tx

pp 133 FIODIR Example.

P0,3 output, Find SPR ?

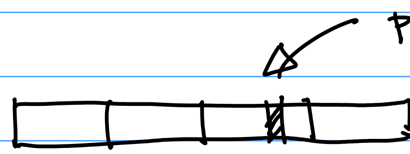
Define bit Value for the Output

Table Look up.

FIO0DIR0

FIOSET, CPU Datasheet Look up.

pin Num set to "1"



```
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 pin Num

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

Example: Set Pin

```c
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")

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

$\vec{P_i} = \vec{P_i}(x_i, y_i) = (x_i, y_i)$

Example: Clear the pin

```c
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);
}
```

Formulation for a straight line

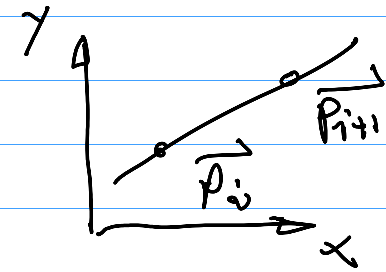


Fig.2

Now, 2D Vector Graphics

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

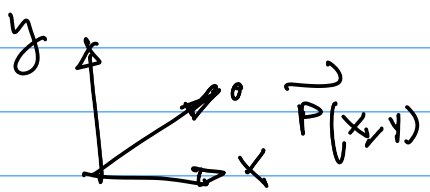


Fig.1

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

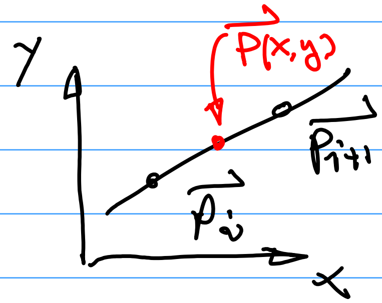


Fig. 2

Note: Need 2 points $\overrightarrow{P_i}$, $\overrightarrow{P_{i+1}}$ to define a line

Let's define a direction vector

$$\vec{d}(x_d, y_d) = \overrightarrow{P_{i+1}} - \overrightarrow{P_i}$$

$$= \overrightarrow{P_{i+1}}(x_{i+1}, y_{i+1}) - \overrightarrow{P_i}(x_i, y_i) \quad \ldots (1)$$

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

$\overrightarrow{P_{i+1}}(x_{i+1}, y_{i+1}) = (5.5, 6.3)$

Find direction vector?

Sol. By Eqn (1), we have

$$\vec{d}(x_d, y_d) = \overrightarrow{P_{i+1}} - \overrightarrow{P_i}$$

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

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

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

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

OR,

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

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

Let's uniquely define a line

Need a pt $\overrightarrow{P_i}$, or $\overrightarrow{P_{i+1}}$; and directional vector

$$\vec{P}(x,y) = \overrightarrow{P_i} + \lambda(\overrightarrow{P_{i+1}} - \overrightarrow{P_i}) \quad \ldots (2)$$

Starting pt — $\overrightarrow{P_i}$; scalar — λ; Directional vector — $(\overrightarrow{P_{i+1}} - \overrightarrow{P_i})$

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 $\vec{P_i}(x_i, y_i)$.

Fig 3a

Fig. 3b

Screen Saver Design for LPC 2D G.E.

Rotating Squares And Trees.

Example: Design A Rotating Squares

Step 1. Define 4 vectors/pts

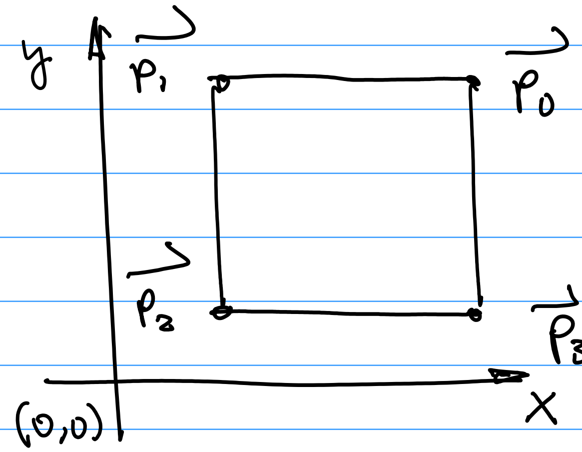
$\vec{P_i}$, $i = 0, 1, 2, 3$

$\vec{P_0}(x_0, y_0) = (60, 60)$, $\vec{P_1}(x_1, y_1) = (10, 60)$

$\vec{P_2}(x_2, y_2) = (10, 10)$, $\vec{P_3}(x_3, y_3) = (60, 10)$

Based on the physical display device

Fig 4



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

Step 2. Use

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

Prepare: LCD Soldering ON the Wire-wrapping Board, Input Single line Drawing project.

Sept. 27 (Mon)

Homework, 2pts. Due 1 week from Today

Topics: 2D Screen Saver Design

Requirements:

a. Build LCD Hardware Interface;

b. Input Sample code from github/hualili/Cmpe240 2018S-10-LCD-DrawLine.

Modify the code to Display 2D Rotating Squares Using 2D Vector equation;

Submission:

c. Project (Zip, Exported)

d. Screenphoto

Submission to CANVAS.

Announcement:

Office Hours — Thu 3:40-4:40pm. Due to SJSU off-Campus Program.

Example: Continued from pp15.

Step 2. Use Vector Equation to find 4 pts

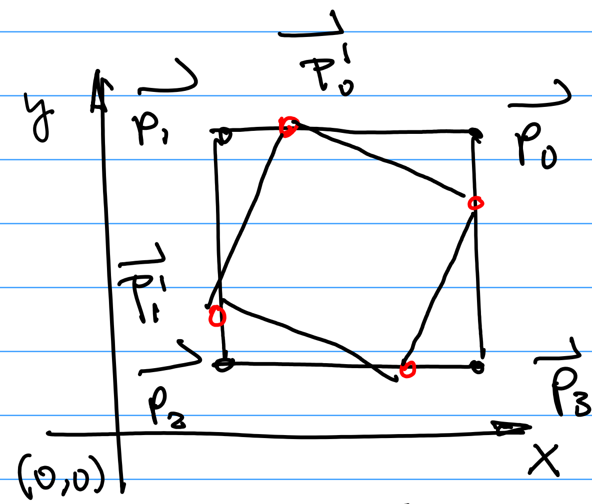


Fig. 1

Let $\lambda = 0.8$, for line 1 ($\vec{P_0}$ and $\vec{P_1}$): Eqn(1), pp15.

Calculate a point $\vec{P_0'}$

Superscript: the Level of iteration;

for line 2, 3, and 4, we do the Same.

line 2 ($\vec{P_1}$ & $\vec{P_2}$), line 3 ($\vec{P_2}$ & $\vec{P_3}$)

line 4 ($\vec{P_3}$ & $\vec{P_0}$)

In Homework, level ≥ 10.

Coding:

$$\begin{cases} x = x_i + \lambda (x_{i+1} - x_i) & \dots (1a) \\ y = y_i + \lambda (y_{i+1} - y_i) & \dots (1b) \end{cases}$$

Hardware Implementation of LCD Interface.

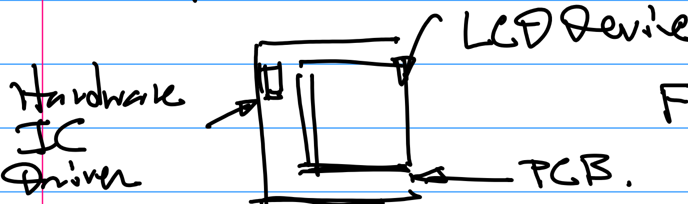
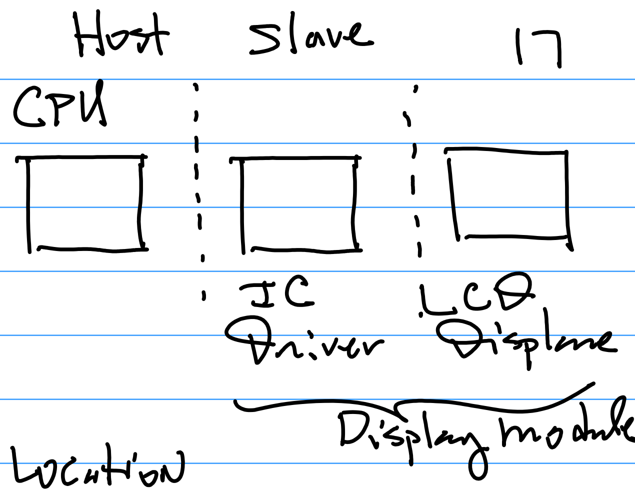


Fig.2



a. To Drive LCD Display. To Display a pixel: (x, y) Location; I(x, y) Intensity, and color

b. To provide feedBack and Interface to CPU module. (SPI Interface)

To Establish interface, SPI (Serial Peripheral Interface)

Hardware pins of SPI : 3+1.

- MOSI (Master Output Slave Input)
- MISO (Master Input Slave Output)
- SCK (SPI clock)
- SSELx (SPI Enable)

Now, Consider the I/F to LCD module.

Ref: github/hualili/cmpe240 2018S-9-SPILCD ...

SCH

LPC-1769	
Label	Pin
MOSI ✓	P 0.18
MISO ✓	P0.17
SCK ✓	P0.15
CS Enable	P0.16
GPIO-DC (Data / command)	P0.21
GPIO-Reset	P0.22
3.3V	
GND	

Fig.3.

Note: mark the Direction of the Signal

a. Identify all pins on CPU for SPI I/F;

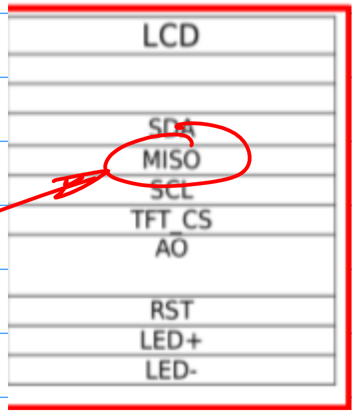
b. Identify all pins on LCD for SPI I/F, Correct matching the host/master and Slave

CmpE240

18

Labels from the LCD Display.

SPI Pins: SDA, MOSI.
MISO
⋮



"SO"

Typo: Change MISO on LCD to SO.
etc.

Note: In Addition to SPI interface, Identify Command/Data Toggle Control pin, the Label should be C/$\overline{D}$ Depending on the Signal Level, the Communication From CPU to LCD is interpreted by LCD either as a Command or Data.

Now, Software Part
github/hualili/Cmpe240
2018S-10-DrawLine

Identify module @ Line 285

Parameters (x_0, y_0), (x_1, y_1) and color
P_0 or P_i P_1 or P_{i+1}

Example: Draw A Line Code

1. Color Definition. Hex Digits

2 hexs for Each primitive color

Primitive Colors: r, g, b (red, green, blue)

2 hex Digits: min. 0, max: 255

2 hex → 8 bit → $2^8 = 256$

Match to Eqn (15) & (16) to Build a Square one Line at time.

Sept. 29 (Wed)

Project 1. (10 pts) Due Oct. 18th Before the Class.

Requirements:

1° All work including prototype Board, Programs, Report

2° Bit Arrangement for the primitive Colors: RGB = 2(hex)(2 hex)(2 hex)

However Team work is encouraged.

2° Implement Hardware LCD Display. a Rotation of sets of Squares, b Create trees to forest; c 3D World Coordinate System Visualization;

Submission:

1° Formal written Requirements Rubrics will be posted on Line.

2° Submission on CANVAS.

3° Source code/Binary have to Exported project, in Zip.

4° Project Report (5 pages) in IEEE format;

5° 5 Seconds video

a Entire System Setting, Host + Prototype Board.

b Screen of the Animated Display; c Show the Prototype Board.

2D Transforms

Mathematical Formulation

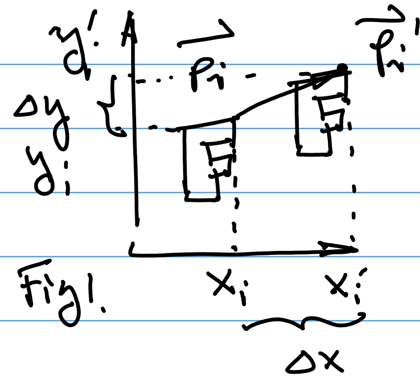


Fig 1.

Given 2D pattern $\{\vec{P}_i(x_i, y_i) \mid i = 0, 1, \ldots N-1\}$

Establish Translation Matrix T.

$\vec{P}_i(x_i, y_i)$ Before ; $\vec{P}'_i(x'_i, y'_i)$ After

$$\underset{\text{After}}{x'_i} \stackrel{?}{=} \underset{\text{Before}}{x_i + \Delta x} \quad \ldots (1)$$

Similarly

$$y'_i = y_i + \Delta y \quad \ldots (2)$$

After Before

$$\begin{pmatrix} x'_i \\ y'_i \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \ldots (3)$$

Let's consider Rotation

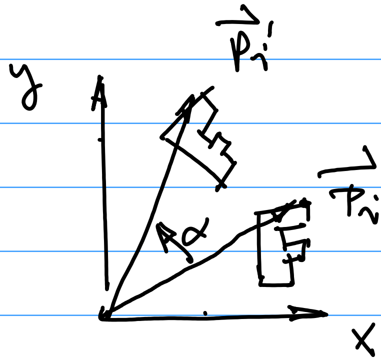


Fig. 2

Note: Counter Clockwise Rotation

"Positive" Angle α

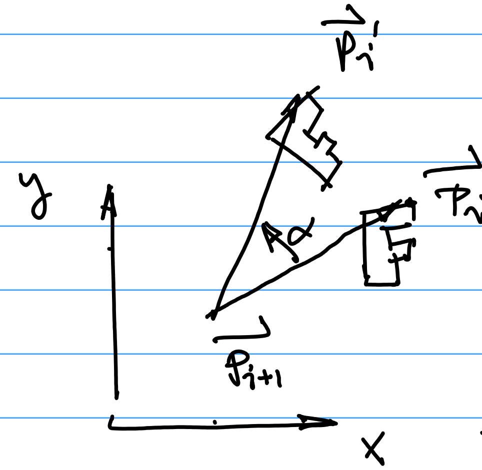


Fig 4.

After Before

$$\begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \ldots(4)$$

Note: for Rotations in Fig 4, we will have to conduct

Pre-processing to Translate the reference point $\vec{P}_{i+1}$ to origin (0,0)

Then, perform Rotation;

Finally, post processing. Translate the rotated pattern Back to its Original Location

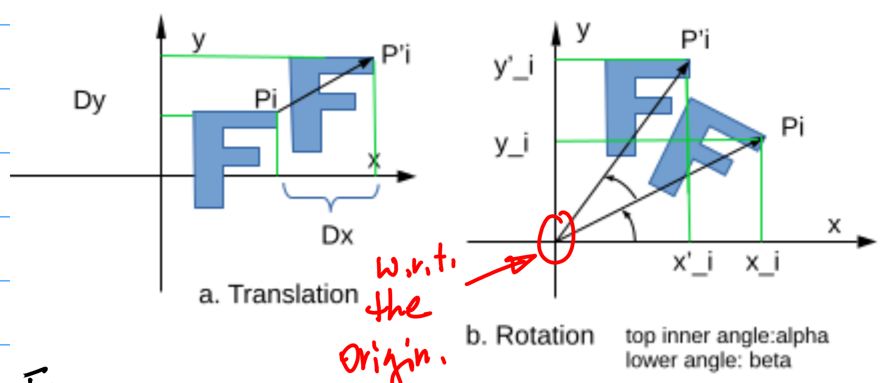


Fig. 3

After Before

$$\begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = T^{-1} R T \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \ldots(6)$$

From Eqn (4)

$$\begin{cases} x_i' = x_i \cos\alpha - y_i \sin\alpha & \ldots(5a) \\ y_i' = x_i \sin\alpha + y_i \cos\alpha & \ldots(5b) \end{cases}$$

where

$$T^{-1} = \begin{pmatrix} 1 & 0 & -\Delta X \\ 0 & 1 & -\Delta Y \\ 0 & 0 & 1 \end{pmatrix} \quad \ldots(7)$$

Fig.5

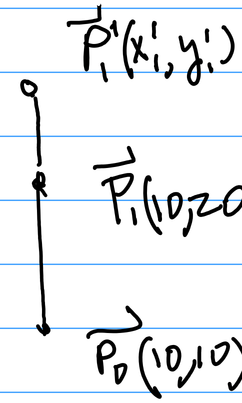


Fig.7

make $\lambda = 0.8$.

Example: Use 2D Transforms to Create Trees shown Above

Step 1. Define Initial Pair of Points to give a tree trunk.

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

$\vec{P}_0(x_0, y_0) = (10,10)$, $\vec{P}_1(x_1, y_1) = (10,20)$

Step 3. Rotation of $\vec{P}_1'$ Counter clockwise to Create Left Branch.

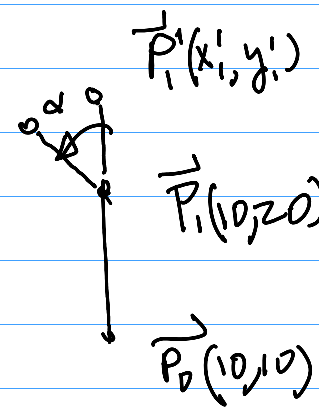


Fig.8

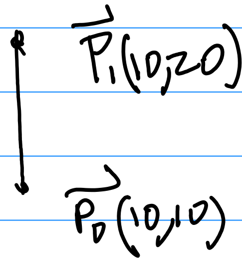


Fig.6

Step 2. Use Vector Eqn to Create next Level main/major Branch $\vec{P}_1'(x_1', y_1')$ as in Fig.

$$\vec{P}_1'(x_1', y_1') = P_0(x_0, y_0) + \lambda(\vec{P}_1(x_1, y_1) - \vec{P}_0(x_0, y_0))$$

Oct.4 (Monday)

Topics: 1° 2D Example for trees
2° Virtual Display v.s. physical Display, Implementation

Example: Continued from Step 3.

First, Preprocess

$$T=\begin{pmatrix}1 & 0 & DX\\ 0 & 1 & DY\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}1 & 0 & -10\\ 0 & 1 & -20\\ 0 & 0 & 1\end{pmatrix}$$

$DX=-10$
$DY=-20$

Next Rotation,

$$R=\begin{pmatrix}\cos\alpha & -\sin\alpha & 0\\ \sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{pmatrix},\quad \alpha=30.$$

Post-Processing:

$$T^{-1}=\begin{pmatrix}1 & 0 & -DX\\ 0 & 1 & -DY\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}1 & 0 & 10\\ 0 & 1 & 20\\ 0 & 0 & 1\end{pmatrix}$$

From Eqn(5)-(7):

$$\begin{pmatrix}1 & 0 & 10\\ 0 & 1 & 20\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\alpha & -\sin\alpha & 0\\ \sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & -10\\ 0 & 1 & -20\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}10\\ 20\\ 1\end{pmatrix}$$

$$=\begin{pmatrix}1 & 0 & 10\\ 0 & 1 & 20\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\alpha & -\sin\alpha & 0\\ \sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}0\\ 0\\ 1\end{pmatrix}$$

$$=\begin{pmatrix}1 & 0 & 10\\ 0 & 1 & 20\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}0\\ 0\\ 1\end{pmatrix}=\begin{pmatrix}10\\ 20\\ 1\end{pmatrix}$$

Then for $\vec{P_1}(10,36)$

Now, Similarly,

$$\begin{pmatrix}1 & 0 & -10\\ 0 & 1 & -20\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}10\\ 36\\ 1\end{pmatrix}$$

$$=\begin{pmatrix}1 & 0 & 10\\ 0 & 1 & 20\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\alpha & -\sin\alpha & 0\\ \sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}0\\ 16\\ 1\end{pmatrix}$$

$$=\begin{pmatrix}1 & 0 & 10\\ 0 & 1 & 20\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}-16\sin\alpha\\ 16\cos\alpha\\ 1\end{pmatrix}$$

$$=\begin{pmatrix}-16\sin\alpha+10\\ 16\cos\alpha+20\\ 1\end{pmatrix}\begin{matrix}\leftarrow \text{New } x\\ \leftarrow \text{New } y\\ \end{matrix}$$

Summary: Put Together Eqn (6) to form the Rotation Algorithm.

$$T^{-1}RT=\begin{pmatrix}1 & 0 & -DX\\ 0 & 1 & -DY\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\alpha & -\sin\alpha & 0\\ \sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & DX\\ 0 & 1 & DY\\ 0 & 0 & 1\end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & -\Delta X \\ 0 & 1 & -\Delta Y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\alpha & -\sin\alpha & \Delta X\cos\alpha - \Delta Y\sin\alpha \\ \sin\alpha & \cos\alpha & \Delta X\sin\alpha + \Delta Y\cos\alpha \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cos\alpha & -\sin\alpha & \Delta X\cos\alpha - \Delta Y\sin\alpha - \Delta X \\ \sin\alpha & \cos\alpha & \Delta X\sin\alpha + \Delta Y\cos\alpha - \Delta Y \\ 0 & 0 & 1 \end{pmatrix}$$

Therefore

$$\begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & \Delta X\cos\alpha - \Delta Y\sin\alpha - \Delta X \\ \sin\alpha & \cos\alpha & \Delta X\sin\alpha + \Delta Y\cos\alpha - \Delta Y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}$$

C/C++ Implementation

$$\begin{cases} x_i' = \cos\alpha \cdot x_i - \sin\alpha \cdot y_i + \Delta X\cos\alpha - \Delta Y\sin\alpha - \Delta X \\ y_i' = \sin\alpha\, x_i + \cos\alpha\, y_i + \Delta X\sin\alpha + \Delta Y\cos\alpha - \Delta Y \end{cases} \quad \ldots (6*)$$

Physical Display Coordinate
V.S. Virtual Display Coordinate

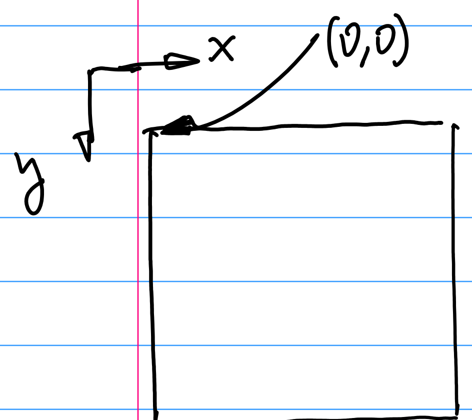


Fig.1

a. Origin (0,0) at Top left hand corner

b. Resolution is $m \times N$

$m \times N$

No. of Pixels/Row

No. of Rows/frame

Suppose LCD Resolution is 120×100

Number of Pixels/Row

100: Rows.

x: Left To Right $0, 1, 2, \ldots, m-1$

y: (Top down) $0, 1, 2, \ldots, N-1$

Limitation:

1° No Negative Value in the System.

2° Tied to the physical Device with Resolution $m \times N$.

3° Not Portable

Now, Virtual Display Coordinate

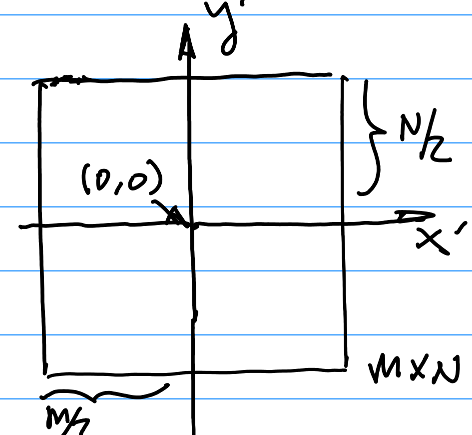


Fig.2

Set (0,0) at the center of the display device.

Transform physical Display to

Virtual Display.

$$\begin{cases} x = x' + M/2 & \dots (1) \\ y = -y' + N/2 & \dots (2) \end{cases}$$

Note: Verify Eqn (1) & (2).

How to use Eqn (1) and (2).

Conduct Computation in Virtual coordinate (x', y'), make sure to Scale the result in $x' \in [-\frac{M}{2}, \frac{M}{2}]$ M Total No. of Col.

$y' \in [-\frac{N}{2}, \frac{N}{2}]$, N : Total No. of Rows.

Then, use Eqn (1) & (2) map to your physical display.

Homework (Due 1 week Oct. 11, Monday) Virtual to physical Transform.

1° Write C code to Realize Eqn (1) & (2).

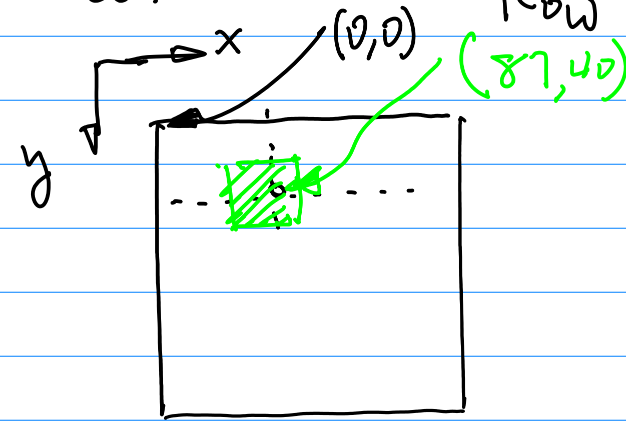
2° Prompt the user for input (x, y) value in Virtual coordinate System,

Then you compute Eqn (1) & (2) to find physical display Coordinate, plot (Draw) 5x5 Patch with its center pixel Equal to the Computation Result.

Example: Computation Result

$(x, y) = (87, 40)$

Col. Row



3D Graphics Processing Engine Introduction : World Coordinate System

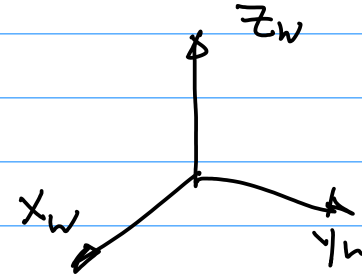


Fig. 1

Outline (Wed)

Topics: 1° Transformation pipeline

Note: 1° Office Hour Update Posted on Git;

2° Handout on GPIO Code, please read it, understand the code. It is required.

Continued ON 3D World-Coordinate System. X_w-Y_w-Z_w, Subscript: "w" for world coordinate System.

a. Right Hand System.

X_w Cross-product with Y_w to produce Z_w.

b. Everything is defined in the World Coordinate System.

Virtual Camera $\vec{E}(x_e, y_e, z_e)$ "Eye" location.

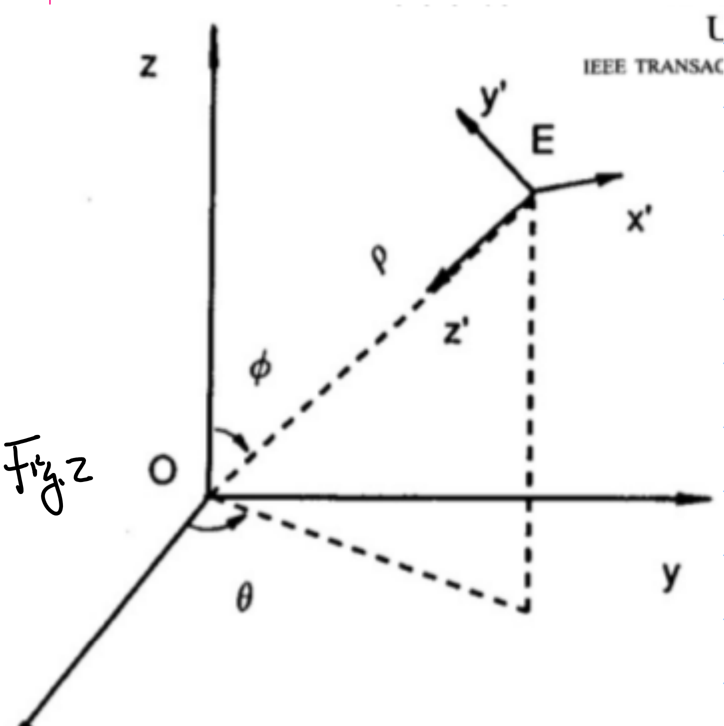
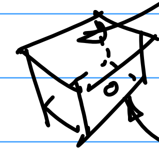


Fig.2

a "pin-Hole" model. Diameter of the Lens ("Hole") is very small, $d \ll S$.

Enclose to form A virtual Camera.



Projection plane to form An Image

pin-Hole

$\vec{ED}$ fixation direction of the Virtual Camera

Note: This is just ONE formulation Among other 3 Additional Possible Formulations

1° Viewer Coordinate System X_e-Y_e-Z_e, sub "e" for "Eye"/Camera Location;

2° Left Hand System.

Relationship B/W X_w-Y_w-Z_w and X_e-Y_e-Z_e Systems to Allow the Definition of Viewing 3D Objects in a different perspective.

Example: Draw A line, passes $E(x_e, y_e, z_e)$ Perpendicular to x_w-y_w plane

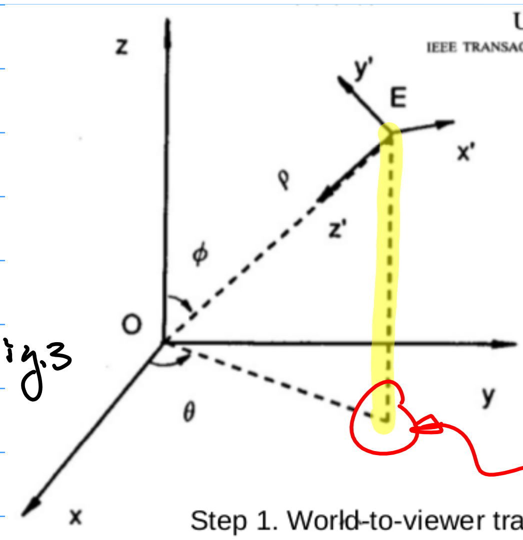


Fig.3

Intersection pt. $\vec{E'}$

Draw 2nd Line, passes E' (on x_w-y_w) to connect to the origin $(0,0,0)$ of x_w-y_w-z_w. as in Fig

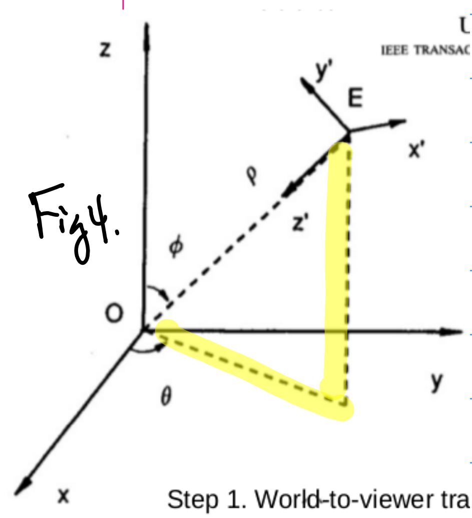


Fig4.

Angle θ: (Theta) formed Between the 2nd Line and x_w.

Angle ϕ (phi): Formed Between $\vec{OE}$ (Not Exactly the $\vec{EO}$) and z_w

Distance ρ (rho): from $\vec{E}(x_e, y_e, z_e)$ to the origin $(0,0,0)$:

$$\rho = \sqrt{x_e^2 + y_e^2 + z_e^2} \quad \ldots (1)$$

Example: A cube given in the following figure:

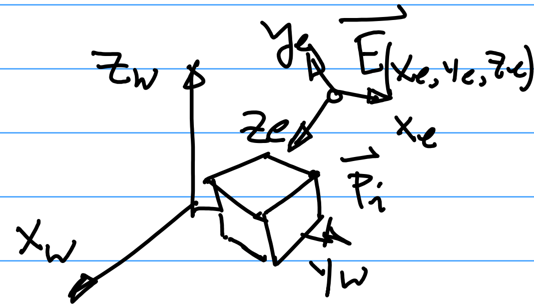


Fig.5

$$\mathbf{T} = \begin{bmatrix} -\sin\theta & \cos\theta & 0 & 0 \\ -\cos\phi\cos\theta & -\cos\phi\sin\theta & \sin\phi & 0 \\ -\sin\phi\cos\theta & -\sin\phi\cos\theta & -\cos\phi & \rho \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \ldots (2)$$

$\vec{P_i}$ defined in the x_w-y_w-z_w

$\vec{P_i'}$ defined in the x_e-y_e-z_e

$$\begin{pmatrix} x_i' \\ y_i' \\ z_i' \\ 1 \end{pmatrix} = \begin{bmatrix} -\sin\theta & \cos\theta & 0 & 0 \\ -\cos\phi\cos\theta & -\cos\phi\sin\theta & \sin\phi & 0 \\ -\sin\phi\cos\theta & -\sin\phi\cos\theta & -\cos\phi & \rho \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \quad \ldots (26)$$

$\vec{P_i'}$ After "in X_e-Y_e-Z_e System" Camera

$\vec{P_i}$ Before "In X_w-Y_w-Z_w" World

Example: Given A Virtual Camera E(200,200,200) Find Transformation Matrix T (World-To-Viewer) Transform Matrix.

Sol. For θ (theta) Angle

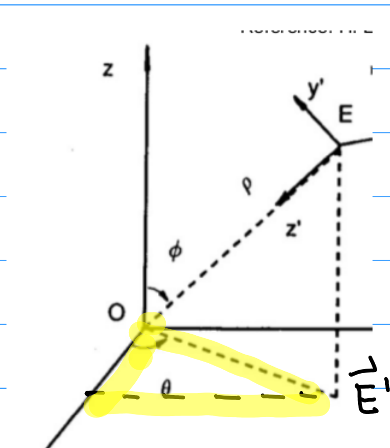
$$\cos\theta = \frac{x_e}{\sqrt{x_e^2+y_e^2}} = \frac{200}{200\sqrt{2}} = \sqrt{2}/2 \quad \ldots (3)$$

(From Fig 4, pp. 26)

$$\sin\theta = \frac{y_e}{\sqrt{x_e^2+y_e^2}} = \frac{200}{200\sqrt{2}} = \sqrt{2}/2 \quad \ldots (4)$$

Consider $\sin\phi$

Draw A Line Passing $\vec{E'}$ Perpendicular to X_w to form A Triangle


Step 1. World-to-vie

Oct. 11 (monday)

Note: 1° 3 Handout material, C Program, with 2021F-111x

a gpio.c ; Architecture → CPU Block Diagram → SPRs → CPU Datasheet Init & Config.

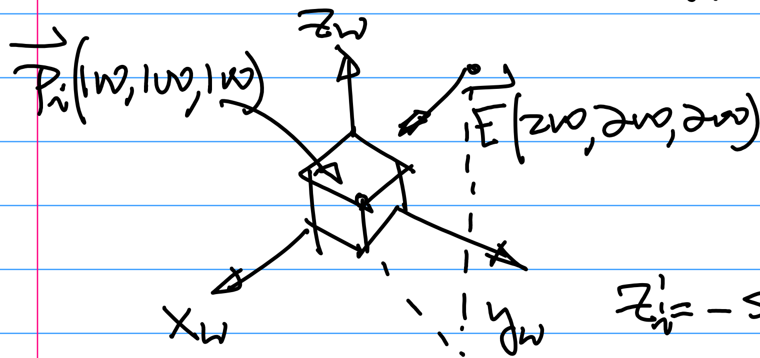
b DrawLine.c : Architecture → Write a pixel (location, color/Intensity). Vector Line

c ssp.c SPI I/F. to Be discussed;

CmpE240

28-1

Example: Suppose a given Cub with $\vec{P}_i(100,100,100)$, And Virtual Camera $\vec{E}(200,200,200)$. Find Viewer Coordinate of this Cub. $P_i' = ?$



From Eqn (26) PP. 27

$$\begin{pmatrix} x_i' \\ y_i' \\ z_i' \\ 1 \end{pmatrix} = \begin{bmatrix} -\sin\theta & \cos\theta & 0 & 0 \\ -\cos\phi\cos\theta & -\cos\phi\sin\theta & \sin\phi & 0 \\ -\sin\phi\cos\theta & -\sin\phi\cos\theta & -\cos\phi & \rho \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix}$$

$$\begin{cases} x_i' = -\sin\theta\, x_i + \cos\theta\, y_i \\ y_i' = -\cos\phi\cos\theta\, x_i - \cos\phi\sin\theta\, y_i + \sin\phi\, z_i \\ z_i' = -\sin\phi\cos\theta\, x_i - \sin\phi\cos\theta\, y_i - \cos\phi\, z_i + \rho \end{cases} \quad \cdots (1)$$

C/C++ Coding Based on Eqn (1).

From the given Condition: $\sin\theta = \cos\theta = \sqrt{2}/2$, $x_i = y_i = z_i = 100$

$x_i' = -\sqrt{2}/2 \cdot 100 + \sqrt{2}/2\, 100 = 0$

$y_i' = -\frac{\sqrt{3}}{3}\cdot\frac{\sqrt{2}}{2}\cdot 100 - \frac{\sqrt{3}}{3}\frac{\sqrt{2}}{2}100 + \frac{\sqrt{6}}{3}\cdot 100$

$\cos\phi = \sqrt{3}/3$, $\sin\phi = \sqrt{6}/3$

$= 100\sqrt{6}/3$

$y_i' = -\cos\phi\cos\theta\, x_i - \cos\phi\sin\theta\, y_i + \sin\phi\, z_i$

$z_i' = -\sin\phi\cos\theta\, x_i - \sin\phi\cos\theta\, y_i - \cos\phi\, z_i + \rho$

$= -\frac{\sqrt{6}}{3}\cdot\frac{\sqrt{2}}{2}100 - \frac{\sqrt{6}}{3}\frac{\sqrt{2}}{2}100 - \frac{\sqrt{3}}{3}100 + 200\sqrt{3}$

$-\frac{\sqrt{12}}{3}100 - \frac{\sqrt{3}}{3}100 + 200\sqrt{3}$

$= \left(-\frac{2\sqrt{3}}{3} - \frac{\sqrt{3}}{3}\right)100 + 200\sqrt{3}$

$= -100\sqrt{3} + 200\sqrt{3} = 100\sqrt{3}$

(x_i', y_i', z_i')

$= \left(0, 100\frac{\sqrt{6}}{3}, 100\sqrt{3}\right)$

Optional Homework: Write C code for Eqn (1).

Handout 2021F-111C SSP.c
Required (Homeworks, Test)

Handout
SSP.c

```
/****************************************************
 *   $Id:: ssp.c 5804 2010-12-04 00:32:12Z usb00423
 *   Project: NXP LPC17xx SSP example
 *
 *   Description:
```

Handout 2021F-112 ~ Review

2D Vector Graphics. to Be posted onLine. Question 1 & Question 2.

Homework: On github/CANVAS
Due A Week from Today 10/18. Submission on CANVAS.

Note: Midterm Scheduled Tentatively on the 1st week of Nov. However, it will be finalized Based on the progress in Projects and Lecture, a Review (15-20 min. in Class), b One Week Ahead Notice.

Format of the lectures.

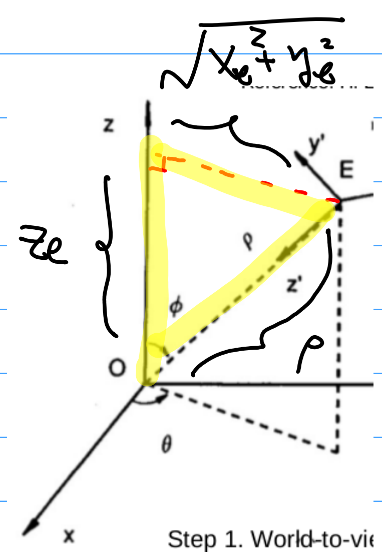
CPU Architectures → CPU Datasheet

Coding Implementation

And Prototype Implementation

Today's Topics: Transformation

Pipeline.



Note: Draw a line pass the Vector $\vec{E}$ location. make Sure a perpendicular to Z_w b in parallel with the line on X_w-Y_w plane

Example: Continue from previous discussion.

$$\sin\phi = \sqrt{x_e^2+y_e^2}/\rho = \frac{\sqrt{x_e^2+y_e^2}}{\sqrt{x_e^2+y_e^2+z_e^2}}$$

$x_e=y_e=z_e=200$

$$= \frac{\cancel{200}\sqrt{2}}{\cancel{200}\sqrt{3}} = \frac{\sqrt{2}\sqrt{3}}{3} = \sqrt{6}/3$$

$$\cos\phi = \frac{z_e}{\rho} = \frac{200}{200\sqrt{3}} = \sqrt{3}/3$$

2nd Step for Transformation Pipeline ⇒ Perspective Projection

Assume the point Before $\vec{P}_i'(x_i', y_i', z_i')$

After Perspective Project,

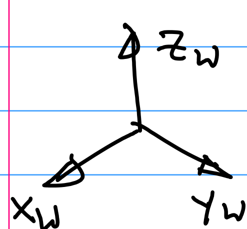
$\vec{P}_i''(x_i'', y_i'')$ (2D pt!)

$$\begin{cases} x_i'' = \frac{D}{z_i'} x_i' & \dots \underline{\underline{(2a)}} \\ y_i'' = \frac{D}{z_i'} y_i' & \dots \underline{\underline{(2b)}} \end{cases}$$

(x_i'', y_i'') on 2D LCD Display With Depth Perception

D: focal length.

$D \simeq 20\text{–}30$



Draw on 2D Display.

Tree Creation

Oct. 13 (Wed) Class Repo

Ref: [1] github/hualili/Cmpe240

2021F-113-LCD-TFT (ThinFilmTransistor).jpg

2021F-114-display-NEC-3P5-LCD-68775.pdf

2021S-100-accessible_CMPE240-S21-v2-HarryLi.pd

Architecture Aspects on Display Driver Design (Ref 1, Ref 2) → S.P.I. (Architecture + Software Coding, SPRs (CPU Datasheet)) → Handout on SPI program.

Road map Before the midterm

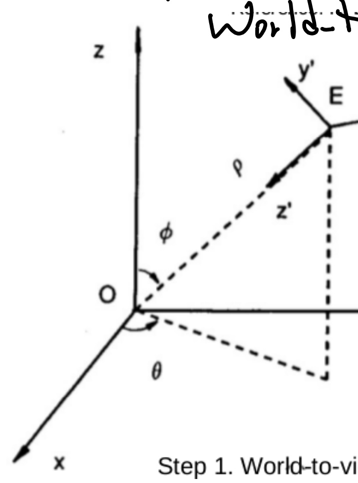
CPU Architecture → GPIO → 2D Vector Graphics

Display Driver Design ← 2D G.E. (Graphics Processing Engine) ←

↓

Hardware → S.P.I. I/F Coding/Programming → 3D Introduction

Example:



Step 1. World-to-vie

World-to-Viewer Transform

$\vec{P}_i \to \vec{P}_i'$ (3D)

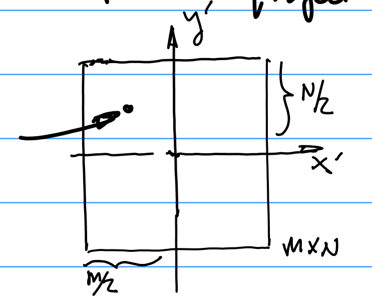
↓ P.P.

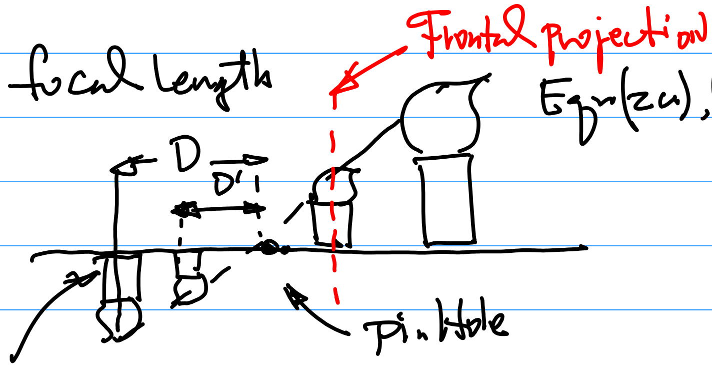
$\vec{P}_i''$ (2D)

Step 2. Perspective Projection

$\vec{P}_i''$

(x_i'', y_i'')





Eqn (2a), (2b)

Display coordinate Transform to find physical display value for your target platform.

Fig. 1.

Compute Perspective projection

Let choose D = 20.

And

$P_i'(x_i', y_i', z_i') = \left(0, 100\frac{\sqrt{6}}{3}, 100\sqrt{3}\right)$, PP2.7

Find Perspective Projection From Eq (2a)

$$x_i'' = \left(\frac{D}{z_i}\right) x_i' \underset{D=20}{=} \left(\frac{20}{z_i'}\right) x_i'$$

$$= \left(\frac{20}{100\sqrt{3}}\right) \cdot x_i' = \left(\frac{20}{100\sqrt{3}}\right) \cdot 0 = 0$$

$$y_i'' = \left(\frac{D}{z_i'}\right) y_i' = \left(\frac{20}{z_i'}\right) y_i'$$

$$= \left(\frac{20}{100\sqrt{3}}\right) \cdot 100\sqrt{6}\,\frac{1}{2}$$

In Virtual Display Coordinate

Step 3. For Implementation.

Use Virtual-To-Physical

Design of Display Driver Hardware & Architecture

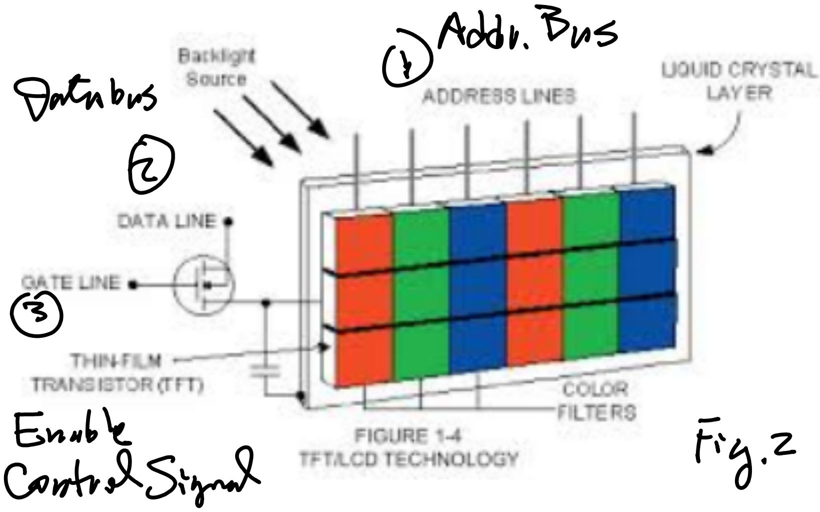


Fig. 2

Target CPU LPC1769 ⇄ Display Driver ⇄ Display Device: Data Bus, Addr. Bus, Control Bus

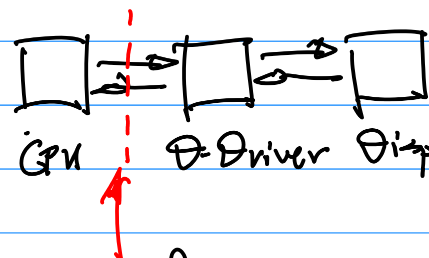


Fig. 3

Hardware, "3+1"

Software

"3+1" MOSI: Master Out Slave Input

MISO: Master IN Slave Out

SCK:

CmpE240

31

SCK: Serial clock.

SSEL: Enable

SPI Waveform (Protocol)

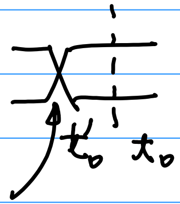


Fig 4.

at t_0' time instant, "changing state".

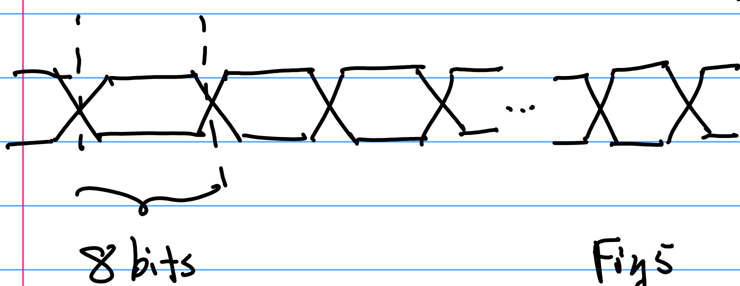


Fig 5

Depends on Application Need, we could have just Instruction Field for SPI. Communication.

Or.

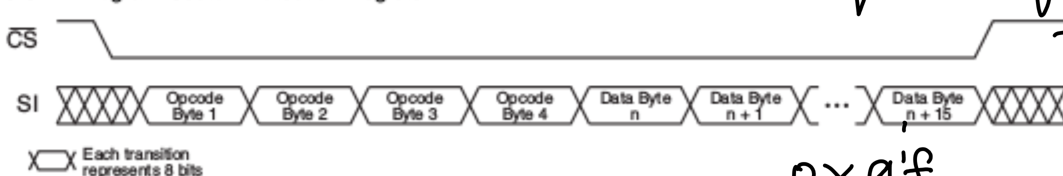
we could have complete 3 Fields, e.g. Opcode (Instruction) Field, Addr. Field, and Data Field.

Use Logic Analyzer or Osc. to Capture the Waveform for debugging.

Note: Each Field, for Example Opcode 0x9f

To ask for manufacturer's ID & Product ID.

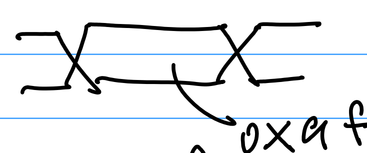
Figure 9-3. Program Sector Protection Register



0x9 : f

1001 : 1111

Waveform

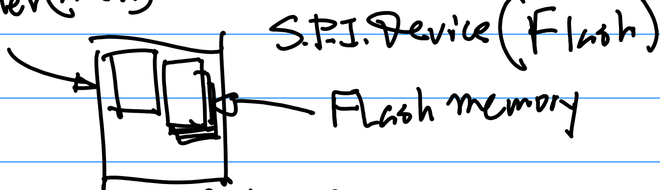


SPI protocol, 3 Fields

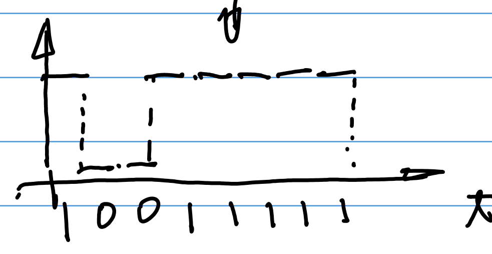
- Opcode : Field 1, Instruction Field
- Address Field, Field 2
- Data (Payload), Field 3, Data Field

Controller (MCU)

Control Register



Slave (client)



Suppose Display Device

Example: Suppose a Display Device

① with the following Resolution 320 x 260
(m x N)

Col / Row   Row

Col. per row
pixels per row   No. of Rows

NEC NEC LCD Technologies, Ltd.

2. GENERAL SPECIFICATIONS

Display area	53.64 (H) × 71.52 (V) mm
Diagonal size of display	8.9cm (3.5 inches)
Drive system	a-Si TFT active matrix
Display color	262,144 colors
Pixel	240 (H) × 320 (V) pixels
Pixel arrangement	RGB (Red dot, Green dot, Bl

② And Display 30 FPS (Frame/second)

③ Pixel depth (color) 24 Bits
r, g, b (red, green, blue) 8 bits each.

Find 1. Bit Rate for SPI Interface?

Oct 18 (monday)   Homework Due 1 week
Oct. 25th

2021F-111-handout-gpio-dotC.pdf

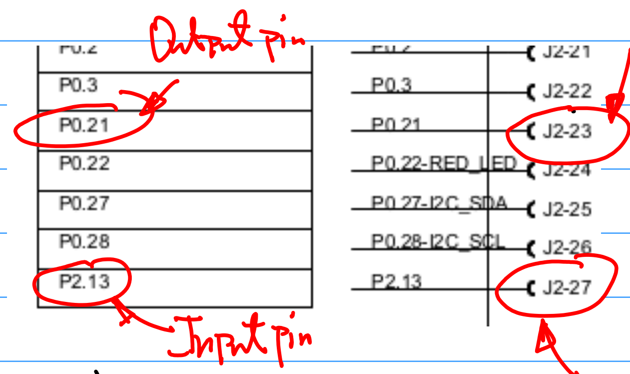
```
//Initialize the port and pin as outputs.
void GPIOinitOut(uint8_t portNum, uint32_t pinNum)
{
        if (portNum == 0)
        {
                LPC_GPIO0->FIODIR |= (1 << pinNum);
        }
        else if (portNum == 1)
        {
                LPC_GPIO1->FIODIR |= (1 << pinNum);
        }
        else if (portNum == 2)
```

Homework Requirements

1° Read the source code then, from SCH of the LPC1769 Design, find the proper value to perform init & Config. to make J2-27 GPIO pin as an input, J2-23 " " " Output.

LPCXpresso1769_CD_revD(1).pdf

NEC 3PF LCD C0775.pdf



Submission.

1° SCH shows GPIO Input & Output Pin

2° Output Testing CKT (LED, Resistor, Resistor Value Calculation)

3° Show main program

4° Photo of GPIO Testing (Output testing Result)

5° Submission to CANVAS.

2. midterm scheduled on Nov.10th (Wed), Close Book, Close Notes.

Requirements: a On-Line, Video has to be at all time, b Scan/Take photos of your papers, Convert to One pdf file. c Naming for the midterm paper:

first_Last_4DigitsID_mid_Cmpe240.pdf

d Submission to CANVAS with DeadLine

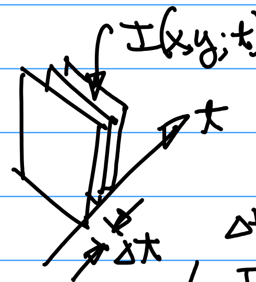
e Exam is One Home Example.

Example: 1° Find SPI Clock Rate (Continued from PP33).

2° Find SFR (Special Purpose Register) Responsible for Init & Config to fulfill this task?

3° Define the init & Config Binary Pattern, Write/modify C-Code to Realize this function.

Sol 1° Find SPI Clock Rate?

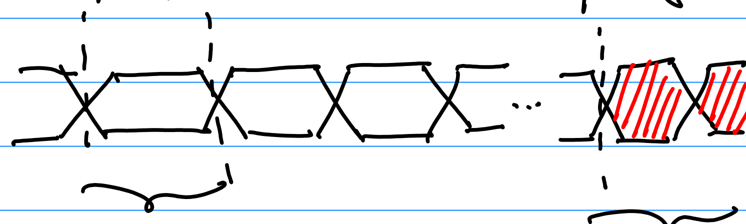


Δt is Related to the Frame Rate

Frame rate: 30 FPS (Frame per Second)

$$\Delta t = \frac{1}{f} = \frac{1}{30} = 33.3\times10^{-3}\ \text{Sec.}$$

SPI protocol: 3 Fields $\rightarrow$ Payload (3rd)

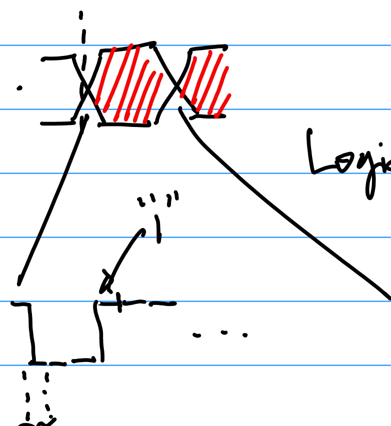


8 bits

Payload

Graphical Data Display

Fig 1.



Logic Analyzer

1 bit, bit "0"

Fig 1b.

1 bit transmitted per time needed to fulfill Requirements from 1° to 3°

So,

$$\underbrace{320\times240}_{\text{Resolution}} \times \underbrace{30}_{\text{FPS}} \times \underbrace{24}_{\text{Color}} = 2^8\cdot2^8\cdot2^5\cdot2^5 = 2^{16}\cdot2^{10} = 2^{10}\cdot2^{10}\cdot2^6$$

Resolution $\simeq 2^8\times2^8$, FPS 2^5, Color (r, g, b)

2^{10}: 1K, $2^{10}\cdot2^{10}$: 1meg.

$$= 2^6\cdot \text{mbps}$$

$$= 64\ \text{mbps}$$

(Need to transfer 64 meg. bits Data to the Display Device) from LPC1769

From Fig 16, One bit Per Clock, therefore SPI Clock rate:

$$f_{SPI} = 64\,\text{MHz} \cong 64 \times 10^6 \quad \ldots(1)$$

Note: If Resolution is Reduced from 360x260 to 160x120 for Example then Bit Rate Roughly is Reduced to its quart of the original.

$$f'_{SPI} \cong f_{SPI}/4 = \frac{64}{4}\,\text{MHz} = 16\,\text{MHz}$$

2° Find SPR Responsible, PP431, Datasheet

Control Register, $\underline{\underline{CR0}}$

Table 371: SSPn Control Register 0 (SSP0CR0 - address 0x4008 8000, SSP1CR0 - 0x4003 0000) bit description

① Super Set for SPI

$\underbrace{\text{SSP0}}_{\text{prefix}}\ \underbrace{\text{CR0}}_{\text{Root}}$: 0x4008_8000

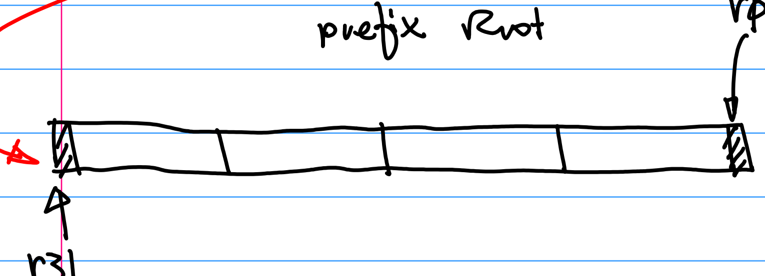


Fig. 2

SSP0CR0[3:0] : DSS (Data Size Select), 8 bits

SSP0CR0[3:0] = 0111 (B)

SSP0CR0[5:4] = 00 (B) for SPI.

Next 2 Bits are Set to 0 By default.

SSP0CR0[6] = SSP0CR0[7] = 0

Serial Clock Rate

SCR 8 bits, 256 Different Combinations.

		the clock line.
15:8	SCR	Serial Clock Rate. The number of prescaler-output clocks per bit on the bus, minus one. Given that CPSDVSR is the prescale divider, and the APB clock PCLK clocks the prescaler, the bit frequency is PCLK / (CPSDVSR × [SCR+1]).
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

$$f_{SPI} = \frac{PCLK}{CPSDVSR\,(SCR+1)} \quad \ldots(4)$$

Oct. 20 (Wed)

Note: Midterm Scheduled on Nov. 10 (We). Close Book / Close Notes One page Formula Allowed

Example: CPU Datasheet. PP431. Suppose CPU Operates 200MHz. SPR Setting for PCLK is set to deliver 1/4 of System Clock, →

$$PCLK = \frac{CLKs}{4} = \frac{200}{4} \times 10^6 = 50\,\text{MHz}$$

$f_{SPI} = 16\,MHz$ (for 160x120 Resolution)

Find SCR Settings to Realize this SPI Clock (Bit Rate of 16 mbps)

Sol: from Eqn (4), pp35

$$f_{SPI} = \frac{PCLK}{CPSDVSR\,(SCR+1)} \quad \cdots (1)$$

Substitute the given conditions

$f_{SPI} = 16\times10^6$, $PCLK = 50\,MHz = 50\times10^6$

$$16\times10^6 = \frac{50\times10^6}{CPSDVSR\cdot(SCR+1)} \quad \cdots (1\text{-}b)$$

SCR field from SPI Control Register CR0 [15:8], And CPSDVSR is a Special Purpose Register [2,254]

1 pp433 2

Table 375: SSPn Clock Prescale Register (SSP0CPSR - address 0x4008 8010, SSP1CPSR - 0x4003 0010) bit description

Bit	Symbol	Description	Reset Value
7:0	CPSDVSR	This even value between 2 and 254, by which SSP_PCLK is divided to yield the prescaler output clock. Bit 0 always reads as 0.	0
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

3

Important: the SSPnCPSR value must be properly initialized or the SSP controller will not be able to transmit data correctly.

SSP0CPSR[7:0]

from Eqn (1-b): $$16 = \frac{50}{CPSDVSR(SCR+1)}$$

Design By selecting a meaningful combination of CPSDVSR, and SCR

CPSDVSR [2,254], SCR [0,255]

Trial-by-Error, Iteration.

Let CPSDVSR = 128, Hence

$$16 = \frac{50}{128(SCR+1)}, \text{ OR}$$

$$16 = \frac{50}{64(SCR+1)}$$

$$16 = \frac{25}{32(SCR+1)}$$

$$\frac{16\times32}{25} = \frac{1}{SCR+1}, \quad SCR+1 = \frac{25}{16\times32}$$

If $\frac{25}{16\times32}$ falls in [0,255]? Not Exactly, so and iteration, try to Redefine CPSDVSR try to make it equal to 2

from Eqn (1-b)

$$16 = \frac{50}{2(SCR+1)}$$

$$SCR+1 = \frac{50}{2\times16} = \frac{50}{32}$$

$$SCR = \frac{50-32}{32} = \frac{18}{32} = 0.56 \simeq 1$$

Compare $\frac{18}{32}$ to $\frac{25}{16 \times 32}$, this result is better.

$SCR \in [0, 255]$

Check if iteration 3 is needed, Since we have the CPSDVSR = 2 minimum, therefore no further iterations.

$\therefore SCR \cong 1$

$CR0[15:8] = 0X1 = (0X01)$

4 bits (upper)   4 Bits (Lower)

From CPU CR0.

$CR0[31:16] = 0X0$

PP431.

		the clock line.	
15:8	SCR	Serial Clock Rate. The number of prescaler-output clocks per bit on the bus, minus one. Given that CPSDVSR is the prescale divider, and the APB clock PCLK clocks the prescaler, the bit frequency is PCLK / (CPSDVSR × [SCR+1]).	0x00
31:8 (16 Typo)	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

.6.2 SSPn Control Register 1 (SSP0CR1 - 0x4008 8004, SSP1CR1 -

Hence, we have the following Technical Specification of this Design:

1° Support SPI Display Driver function with Bit Rate of 16 Mbps $\to f_{SPI} = 16 \times 10^6$ (MHz)

2° Supports Resolution 160x120, 30FPS, Pixel depth is 24bpp.

Hence, Control Register CR0 has to be set to the following binary pattern

CR0[31:0]:

$CR0[3:0] = 0111 = 0X7$ Bits/Trans..

$CR0[5:4] = 00$ for SPI

$CR0[7:6] = 00$ By default

$CR0[15:8] = 0000\,0001 = 0X1$

$CR0[31:16] = 0 \ldots 0 = 0X00$

Together

$CR0[31:0] = 0X0000\_0107$

$= 0X0107$

$= 0X107$

Note: Technical Spec. ↔ Binary Pattern for Init & Config.

Now, C/C++ Implementation (Example from DrawLine.c)

CmpE240

38

Required, Init & Config. By Control Registers CR0, CR1

SSP1CR0, SSP1CR1

```
218 /******************************************************************************/
219 void SSP1Init( void )
220 {
221   uint8_t i, Dummy=Dummy;
222
223   /* Enable AHB clock to the SSP1. */
224   LPC_SC->PCONP |= (0x1<<10);
225
226   /* Further divider is needed on SSP1 clock. Using default divided by 8  */
227   LPC_SC->PCLKSEL0 |= (0x3<<20);
228   /* P0.6~0.9 as SSP1 */
229   LPC_PINCON->PINSEL0 &= ~((0x3<<12)|(0x3<<14)|(0x3<<16)|(0x3<<18));
230   LPC_PINCON->PINSEL0 |= ((0x2<<12)|(0x2<<14)|(0x2<<16)|(0x2<<18));
231
232 //#if !USE_CS
233   LPC_PINCON->PINSEL0 &= ~(0x3<<12);
234   LPC_GPIO0->FIODIR |= (0x1<<6);        /* P0.6 defined as GPIO and Outputs */
235 //#endif
236
237   /* Set DSS data to 8-bit, Frame format SPI, CPOL = 0, CPHA = 0, and SCR is 15 */
238   LPC_SSP1->CR0 = 0x0707;
239
240   /* SSPCPSR clock prescale register, master mode, minimum divisor is 0x02 */
241   LPC_SSP1->CPSR = 0x2;
```

(1) (2) (3) (4) (5)

LPC_SC -> PCONP

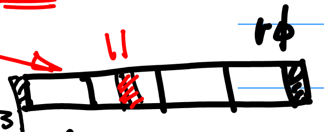
product ID

SPR: peripheral Controller Power

i.e. Control.

a CPU Datasheet 0x1<<10 b "|=" Bitwise OR operation

SPR: PCLKSEL0 for SSP1Clock.



a "~" Negation, 0x3 → 11 → "~" of 11 → 00

Clear 2 bits ← AND "&=" ↲

Set 2 bits as "10" (=0x2)

Example: for PINSEL0, P.113 CPU Datasheet

8.1 How to read this chapter

Table 75 shows the functions of the PINSEL registers in the LPC176x/5x

Table 75. Summary of PINSEL registers

Register	Controls	Table
PINSEL0	P0[15:0]	Table 80
PINSEL1	P0 [31:16]	Table 81
PINSEL2	P1 [15:0] (Ethernet)	Table 82
PINSEL3	P1 [31:16]	Table 83
PINSEL4	P2 [15:0]	Table 84
PINSEL5	P2 [31:16]	not used
PINSEL6	P3 [15:0]	not used

P.117 (1) PINSEL0 → Select/Define Multiplexed functions

Table 80. Pin function select register 0 (PINSEL0 - address 0x4002 C000) bit des

PINSEL0	Pin name	Function when 00	Function when 01	Function when 10	Function when 11
1:0	P0.0	GPIO Port 0.0	RD1	TXD3	SDA1
3:2	P0.1	GPIO Port 0.1	TD1	RXD3	SCL1
5:4	P0.2	GPIO Port 0.2	TXD0	AD0.7	Reserved
7:6	P0.3	GPIO Port 0.3	RXD0	AD0.6	Reserved
9:8	P0.4[1]	GPIO Port 0.4	I2SRX_CLK	RD2	CAP2.0
11:10	P0.5[1]	GPIO Port 0.5	I2SRX_WS	TD2	CAP2.1
13:12	P0.6	GPIO Port 0.6	I2SRX_SDA	SSEL1	MAT2.0
15:14	P0.7	GPIO Port 0.7	I2STX_CLK	SCK1	MAT2.1
17:16	P0.8	GPIO Port 0.8	I2STX_WS	MISO1	MAT2.2

(2)

Program line 230.

b13 b12

P0.6 → SPI (SSEL1)

b13=1, b12=0

Oct. 25 (Monday).

Note: 1° Prep. for Midterm Exam. Nov. 10 (Wed). on Zoom.

(1) Video on the entire Session. Mandatory. In emergency, Text to (650) 400-1116.

(2) One page formula is allowed. Submit the formula sheet together with exam papers By the end of Exam. No verbal description, No Block Diagrams allowed. Put your First, Last Name with 4 Digits of Student ID. on the sheet.

Close Book, Close Note, Datasheet if needed, it will be provided together within the exam paper.

(3) Use Blank Printer Paper, No Class Notepad is allowed.

1° Put your First, Last Name on the top right corner on Each page; together with 4 Digits ID.

2° Scan paper(s) and then place them together, use software to form/generate ONE pdf document. !

https://www.google.com/search?client=ubuntu&channel=fs&q=convert+multiple+jpegs+to+pdf&ie=utf-8&oe=utf-8

Example:

https://jpg2pdf.com

JPG to PDF – Convert JPG Images

This free online **JPG to PDF converter** allows to c

3° 5-10 minutes Extra time for preparing CANVAS Submission.

4° One pdf Document, then zip it. With the Name as follows:

FirstName-LastName-4Digits-mid-Cmpe240-yy-mm-dd.zip.

Exercise on Wednesday. Bring a Blank paper to Class, Ready for Scaning, Submission to CANVAS.

Continue on SPI. program (SPR. CR0 & 1) Init & Config.
Handout of the programs.

Example. Program Line 238, pp. 38

① SPI's Special Purpose Register, Control Registers { CR0 → 1st control register; CR1 }

```
232 //#if !USE_CS
233   LPC_PINCON->PINSEL0 &= ~(0x3<<12);
234   LPC_GPIO0->FIODIR |= (0x1<<6);        /* P0.6 defined as GPIO and Outputs */
235 //#endif
236
237   /* Set DSS data to 8-bit, Frame format SPI, CPOL = 0, CPHA = 0, and SCR is 15 */
238 ① LPC_SSP1->CR0 = 0x0707;
239
240   /* SSPCPSR clock prescale register, master mode, minimum divisor is 0x02 */
241   LPC_SSP1->CPSR = 0x2;
```

② Naming Convention Revisit: LPC_SSP1 → CR0

- LPC: Product Family
- SSP1: Peripheral Controller; "1" Enumeration → more than one SSPx
- → CR0: Pointer (Addr.) to CR0

③ Binary Pattern for SSP1 init & Config.

Technical Specification (Verbalized or Tabulation) ⇄ Binary Pattern

LPC_SSP1 → CR0 = 0x0707; (= 0x0000 0707)

- 0x0000: Upper 16 bits Unused. CR0[31:16]
- 0000 0111 0000 0111: 0000 0111 = SCR; 0000 0111 = 8 bit Packet Data

$$f_{SPI} = \frac{PCLK}{CPSDVSR\,(SCR+1)}$$ Eqn(1), P36.

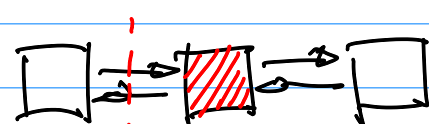
Line 241, Defines Prescaler Needed in f_{SPI}.

Summary: SPI CR0 Init & Config. Code Done. (SSPx)

Handout on the Class github.

- ✓ 2021F-111-handout-gpio-dotC.pdf
- ✓ 2021F-111b-handout-drawline-dotC.pdf
- ✓ 2021F-111c-handout-ssp-dotC.pdf

Example: Now, Consider the Display Driver Design task:



CPU | Driver | Display (P30)

Fig 1.

CMPE240

41

Design Aspect 1: Interface Between the Device Driver and the CPU.

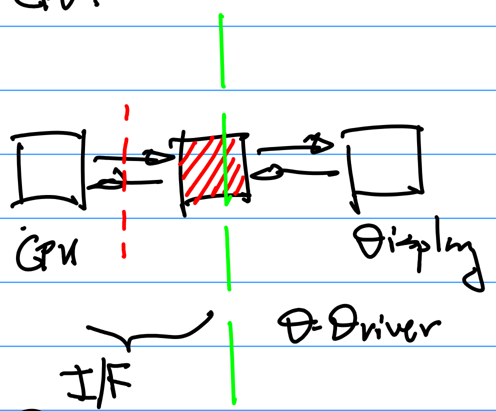
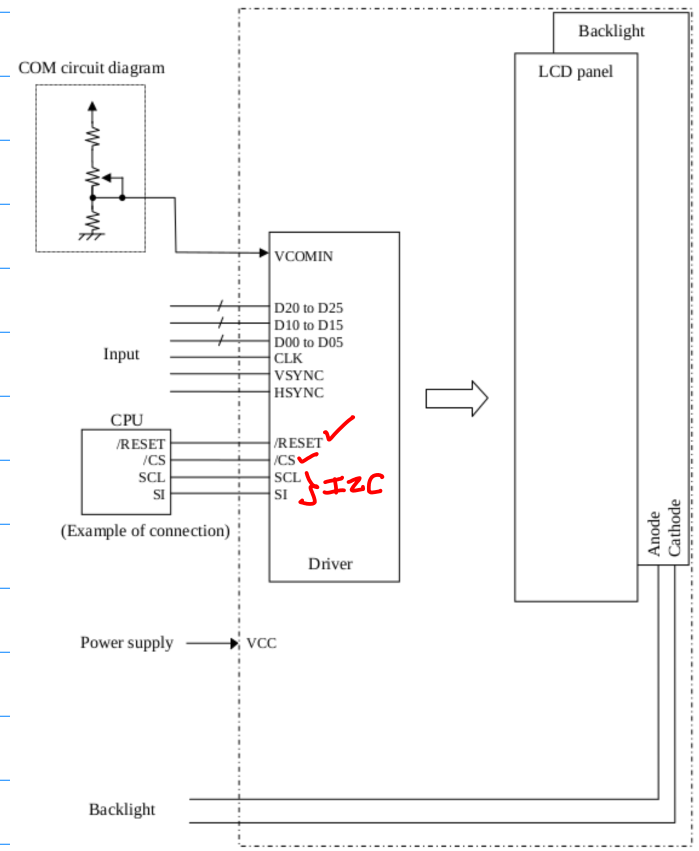


Fig 1b



Finish this Design Based on SPI Protocol "3+1"

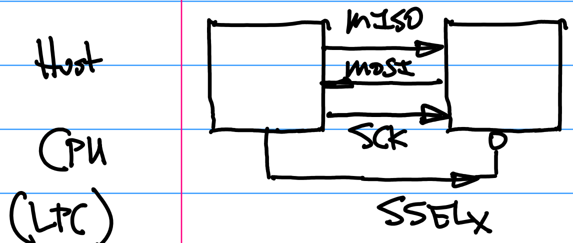


Fig. 1c

Note:
1° Display panel consists of the following Pins (e.g. Subsystems)
1.1° Data Bus (Bi-Directional) 16 bits to 32 bits
1.2° Address Bus (optional)
Timing Signals to Serve the purpose of Address Bus.
a Frame-Sync.
b H-Sync. (Horizontal)
c Data-Sync.

The 2nd Aspect of the Driver Design is Related Display Panel.

2021F-114-display-NEC-3P5-LCD-68775.pdf

NEC NEC LCD Technologies, Ltd.

TFT COLOR LCD MODULE

NL2432HC22-40J

8.9cm (3.5 Type)
QVGA

DATA SHEET
DOD-PP-0129 (3rd edition)

2021F-115b-homework-review1-cmpe240-…

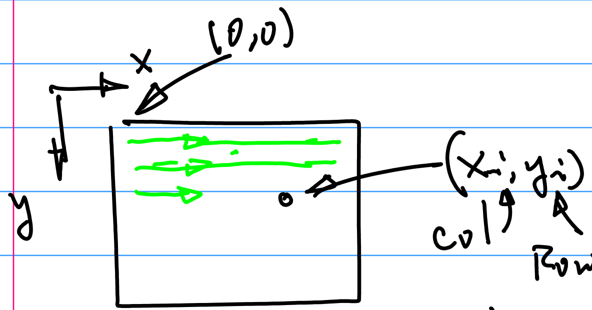
Oct. 27 (W)

Next Monday, Nov 1st. Do Exercise to Practice Submission of paper as a part of Prep for the midterm.

Example: Design for Display Device.

- Data Bus
- Address Bus — To give Location of a pixel or a collection of pixels on the display.
- Control — To give location of Control Unit inside Display Device

Enhanced Display Device(s), there is no Addr. Bus, But Timing Signals Instead.



M x N

Fig. 1

No. of Pixels/Row

No. of Rows

Timing Signals for the definition of Pixel location.

First, Frame Rate: 30 FPS (Frame Per Second).

$f_F = 30$ Hz. $Sync_F = 30$ Hz. ... (1a)

$T_F = 1/f_H$ ... (1b)

Next Timing Signal Horizontal Sync.

Note: (0,0): Origin, physical display device.

Display pixels from Top left hand corner, (0,0), one pixel at time from Left to Right.

Speed of the display is defined by horizontal scanning rate → $Sync_H$. (Horizontal Sync).

$f_H = N f_F$ ... (2a)

$T_H = \frac{1}{N} T_F$ ... (2b)

$Sync_H = f_H$

Timing Signal for Each Pixel.

$f_D = M f_H$ ... (3a)

$T_D = \frac{1}{M} T_H$ ... (3b)

$Sync_D$

From Hardware Design Aspect, we have the following 3 Timings

$Sync_F$, $Sync_H$, $Sync_D$

Example: Suppose, we want to display a pixel at (56,173) using $Sync_F$, $Sync_H$, $Sync_D$.
Find out the Time (clocks) Needed for the Display Device Driver.

Sol.

Suppose we have a display Device shown below with Resolution $M \times N$ (320×260), Find time Needed to display @ (56, 173).

$$\underset{\substack{M \\ \text{Col.}}}{56}, \quad \underset{\substack{N \\ \text{Row.}}}{173}$$

Step1. get the Row Number, Col. Number.

Col = 56.
Row = 173.

Step2.
Scanning the Number of Rows From the given condition
Row = 173

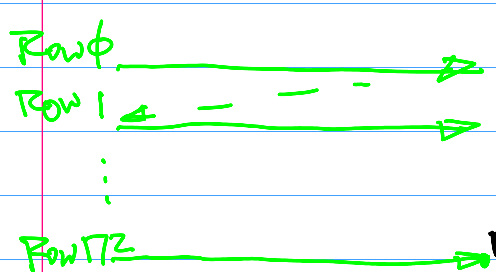


Fig. 2

Hence, the time for Scanning the Row is $173\,T_H$;

Step 3. For pixel Timing on the 173th Row.

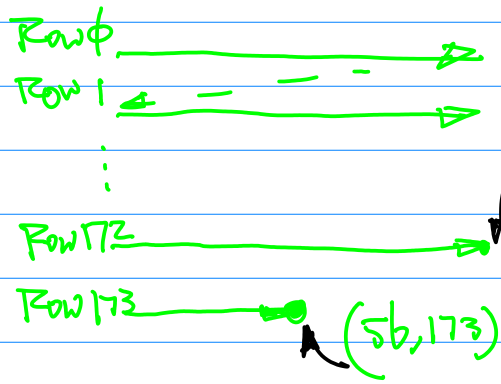


Fig. 3

From left to Right, one pixel at time
pixel0, pixel1, ..., pixel55, pixel56
By the Time of Completion.
$56+1=57$

$57\,T_D$

Step4. Combine Step 2 and 3 together

$$T_\Sigma = 173\,T_H + 57\,T_D \quad \ldots (4)$$

$$T_H = \frac{1}{N} T_F, \quad T_D = \frac{1}{M} T_H$$

$$T_F = \frac{1}{f_F} = \frac{1}{30} = 33 \times 10^{-3}$$

Hence,
$$T_\Sigma = 173\,T_H + 57\,T_D$$
$$= 173 \cdot \frac{1}{N} T_F + 57 \cdot \frac{1}{M} T_H$$
$$= 173 \frac{T_F}{N} + 57 \frac{1}{M} \cdot \frac{1}{N} T_F$$

$$T_\Sigma = 173\frac{T_F}{260} + 57\cdot\frac{1}{320}\cdot\frac{1}{260}\cdot T_F = \left(\frac{173}{260} + \frac{57}{320\times 260}\right)\times 33\times 10^{-3}$$

$T_F = 33\times 10^{-3}$

Example: 2nd part of the Design

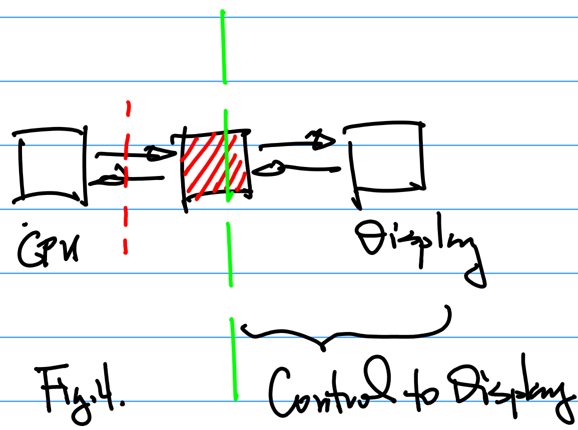
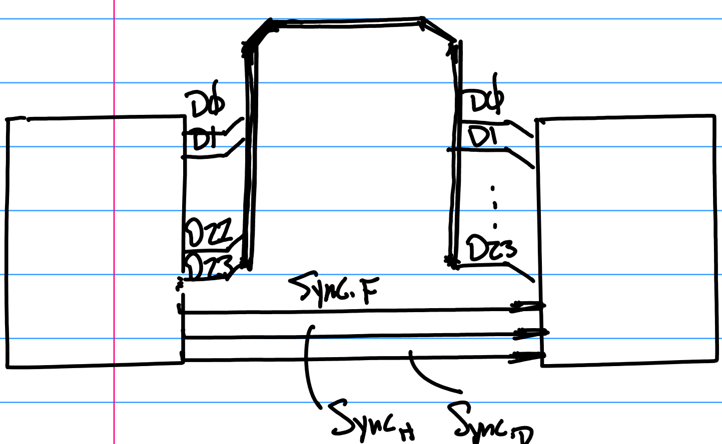


Fig.4.



Controller

Display Driver

Fig 4-b

Nov. 1st (Monday)

Note: Midterm on Nov. 10th (W).

Close Book/Close notes

One page formula is allowed.

Example: Control Bus/Signals for the Display Device

nCS (Chip Select "n" — Active Low)

nREST (Reset, Active Low)

D/C Data (Active High)

Command (Active Low, "n")

Therefore, Fig 4-b becomes

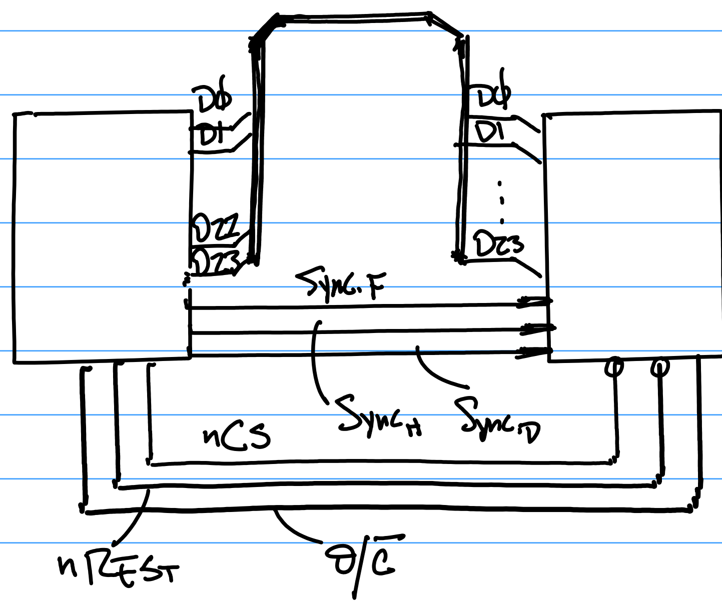


Fig 5

Note 1°: Note the Above Design Only Shows the I/F to the TFT Display panel. For the I/F to Host CPU, see the Design Fig 1c, pp. 41.

The Integrated Design is given here:

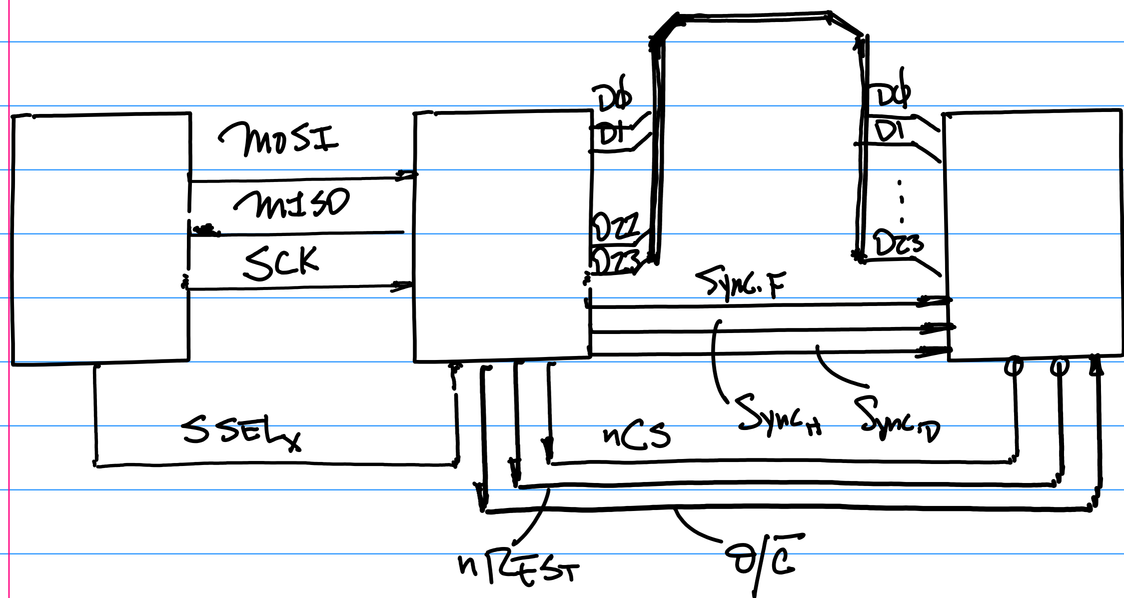
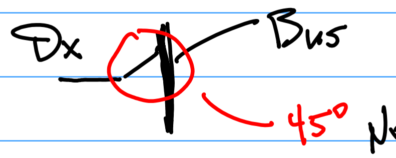


Fig.6

2° Label Each pin is Required. Direction of a signal flow has to Be indicated on SCH. (Note CAD Tool has software definition for Each pin, But not visible on a Printed file).

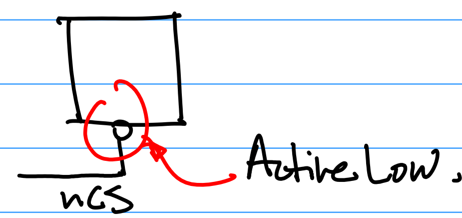
3° Bus has to Be Drawn as a Solid thick line to indicate it's a vector D[23:0]

Connection of a Single bit (e.g. a Single wire) to the Bus is Drawn as follows



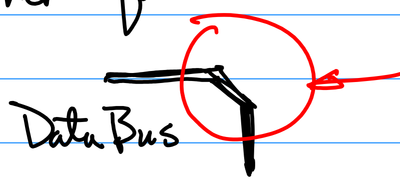
No Particular function, But rather "Artistic" presentation. Can be up angle, or down angle, or straight.

4° Active Low Signal is illustrated here with a little circle.



However, CAD Tool Does not show this Circle, Since its software defines the attributes of Each pin Already.

5° Bus line(s) with 45° Angle at the Corner is for "Artistic" presentation.



Example: Design 3D Algorithm and Implement it on LPC platform.

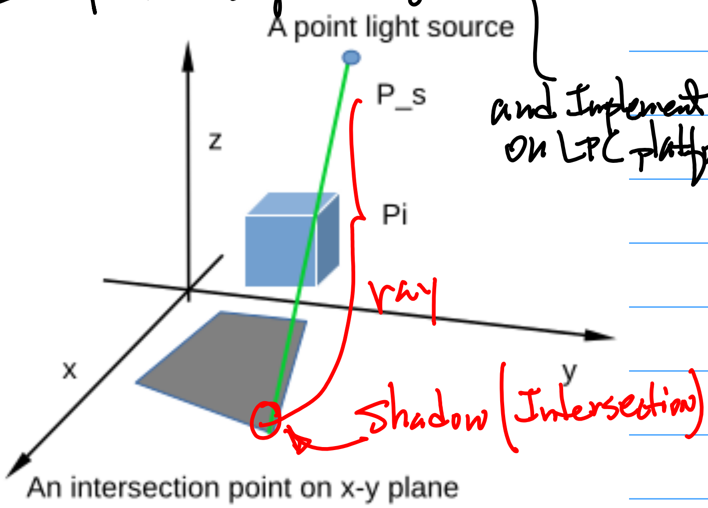


Fig 11.1.

ARM Microprocessor Systems

A Practical Hands-On Guide with Graphics Processings

Harry Li, Ph.D., Professor
Computer Engineering
San Jose State University

Objective: To create shadow(s) from point light source(s).

Step 1.

(1) Set up X_w-Y_w-Z_w World Coordinate System.

(2) Define a point light source $\vec{P_s}(x_s, y_s, z_s) = (-50, 100, 200)$

(3) Define A cube with Side = 100. And floating Above X_w-Y_w plane.

Step 2. Ray Equation.

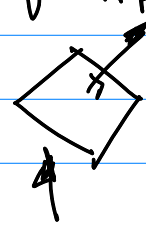
$\vec{r}(x_r, y_r, z_r) = \vec{P_i}(x_i, y_i, z_i) + \lambda(\vec{P_s}(x_s, y_s, z_s) - \vec{P_i}(x_i, y_i, z_i))$ ... (2)

$\vec{P_s}(x_s, y_s, z_s)$: A single pt. Light Source.

A shadow Point: A common point shared by Ray Equation $\vec{r}(x_r, y_r, z_r)$ and a plane, x_w-y_w plane. $\Rightarrow$ the Intersection point Between Ray Equation and the plane.

Step 3. Plane Equation.

Define A plane.



Normal vector $\vec{n}(n_x, n_y, n_z)$ or (x_n, y_n, z_n)

Normal vector $\vec{n}(x_n, y_n, z_n)$ is Perpendicular to "π" (plane).

$$P_1(x_1, y_1, z_1) = (0, 0, 10), P_2(x_2, y_2, z_2) = (100, 0, 10),$$
$$P_3(x_3, y_3, z_3) = (100, 100, 10), P_4(x_4, y_4, z_4) = (0, 100, 10),$$
$$P_5(x_5, y_5, z_5) = (0, 0, 110), P_6(x_6, y_6, z_6) = (100, 0, 110),$$
$$P_7(x_7, y_7, z_7) = (100, 100, 110), P_8(x_8, y_8, z_8) = (0, 100, 110). \quad (11.1)$$

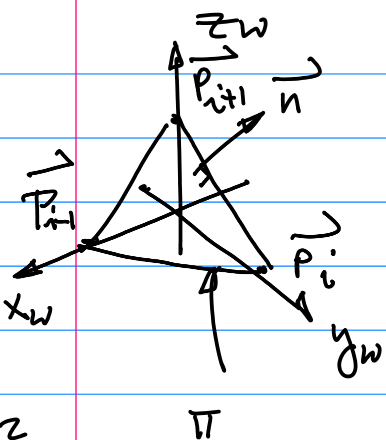


Fig2

(Note: $\vec{P}_{i-1}, \vec{P}_i, \vec{P}_{i+1}$ Arranged in A Counter Clockwise direction for the future use)

Form/Construct 2 Vectors, $\vec{P}_i - \vec{P}_{i-1}$ and $\vec{P}_{i+1} - \vec{P}_{i-1}$

Vector Cross product:

$$\vec{n} = \frac{(\vec{P}_i - \vec{P}_{i-1}) \times (\vec{P}_{i+1} - \vec{P}_{i-1})}{\|(\vec{P}_i - \vec{P}_{i-1}) \times (\vec{P}_{i+1} - \vec{P}_{i-1})\|} \quad \cdots (3)$$

Once the Normal vector is defined, we need a known point on the plane to uniquely define the plane.

Pick an arbitrary Known Point $\vec{a}$

Pick 2nd arbitrary point on the plane, which is an unknown point. $\vec{v}$

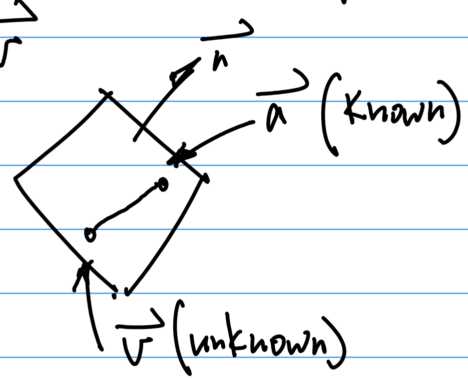


Fig.3.

$\vec{v} - \vec{a}$ a line

Nov 3rd (Wed)

Note: Midterm Exam Scheduled A week from today, Nov. 10th (Wed)

Review.

1. Scope of the exam: From the Beginning of the Semester to 3D G.E. (Including Shadow)

2. One page formula is allowed
   No System Block Diagrams, No Schematics Allowed; No verbal Description.
   Just math Formula
   Close Book / Close Notes.
   Datasheet will be provided.

3. Exam will be carried out on Zoom. Your Camera has to be on throughout the entire session of the exam. Video Recording is on.

4. Blank paper(s) are needed to write your answer.

   Each page.

   First Name - Last Name - 4 Digits ID - Cmpe240 mid

   Page Number.

Don't use Notebook,
Do not use Notebook papers.

5. Exam paper will be posted on Canvas,

1 hr. Exam. Starts @ 2:15 pm.

15 min. Extra to prepare the submission of your paper.

Smart phone APP (such as Microsoft office Lens)

Scan each page, then integrate them into One pdf Document.

Naming Convention:

FirstName_LastName_4Digits ID_240_mid.Zip

(please Zip your pdf)

Submission to CANVAS.

CANVAS on from 2:15—3:30
- 2:15 - 3:15
- 3:15 - 3:30 prep. for Submission.

6. Please Bring my attention By rising your hand when ready for Scanning your paper.

7. Formula sheet has to be Submitted together w/ your exam paper, Put your First, Last Name, ID, CmpE240 on the sheet.

8. If emergency, text me @ 650-400-1116.

About 3 Questions.

A Question Related to Basic Concepts.

CPU Architecture — Memory map. (Partition into memory Banks, Starting Addr. of Each Bank, to find Locations of Peripheral Controller, Special Purpose Registers) — LPC17xx.h (mapping)

handouts:
- GPIO, SPR/Datasheets.
- SPI I/F. CR0, CR1
- DrawLine

Tech. Spec. ⇄ Binary Pattern

Clock Rate for Engineering Design Applications.

8 Bits in CR0

CR0[15:8]

Design + Implementation Related Questions

From Simple GPIO Testing
- Input
- Output

Implementation for LCD Interface.
(SPI I/F)

2D Vector Graphics
Vector Equation, λ
Rotate Squares
Composition of 2D Transforms.

$T^{-1} R T$ $\quad \Delta X = ?$
$\Delta Y = ?$
Angle $\alpha = ?$

Rotation is defined wrt the origin of a coordinate system.

Screen Saver: implementation of trees.

3D Graphics.

Transformation Pipeline,

World-to-Viewer Transform
$T_{4\times4}(\theta, \phi; \rho)$; $\vec{E}(x_e, y_e, z_e)$

Perspective Projection:
3D Graphics to Be Drawn on 2D Display.

$$\begin{cases} x_i'' = \dfrac{D}{z_i'} x_i' \\ y_i'' = \dfrac{D}{z_i'} y_i' \end{cases}$$

Virtual Display $\rightleftarrows$ Physical Display.

Architectural Aspect of the Display Driver Design.

- Display Driver I/F to CPU
- Display Driver I/F to Display Panel

SPI Based I/F.

SPI protocol (Waveform)

Command (Instruction) Field + Address Field + Payload Field

↓

One Clock One Bit Transmitted

↓

Design for 30 F.P.S. with Resolution 160x120.

Bit Rate, CR0. Program

Display Panel Part: schematics.

Data Bus (pixel depth.)
bpp (bit per pixel)

Timing/Sync.
- Sync. F.
- Sync. H
- Sync. D

3D Shadow.

Nov. 8 (Monday) Midterm is scheduled on Wed (the 10th)

CmpE240

Example: Finding shadow(s).

$\vec{n}(n_x, n_y, n_z) = (0, 0, 1)$ ... (1)

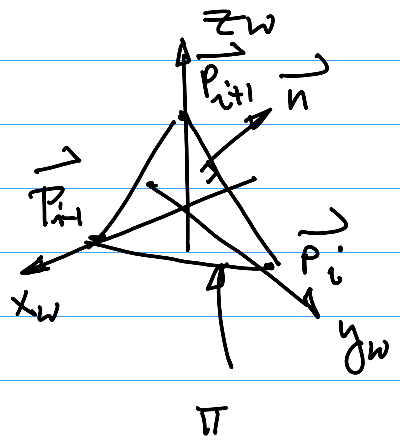


Fig 1. Defining A Normal vector. By Vector Cross Product.

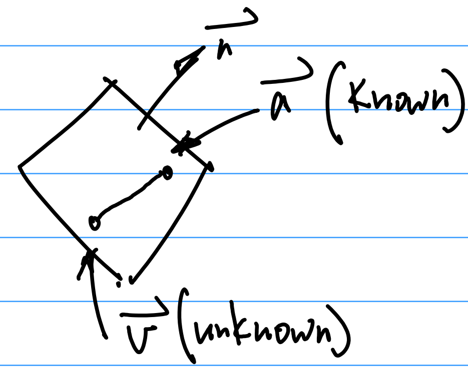


Fig. 2.

3 vertices $\vec{P}_{i-1}, \vec{P}_i, \vec{P}_{i+1}$ → 2 vectors $\left\{ \begin{array}{l} \vec{P}_i - \vec{P}_{i-1} \\ \vec{P}_{i+1} - \vec{P}_{i-1} \end{array} \right.$ → Cross Product $\vec{n}$

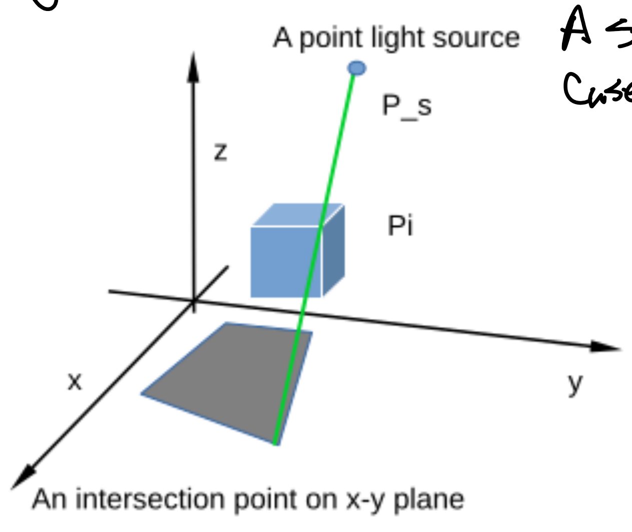
Now, 2 Vectors, $\vec{a}(a_x, a_y, a_z)$ a known point on the plane ($x_w - y_w$). And $\vec{v}(x, y, z)$ Arbitrary pt on the Same plane.

$\vec{v} - \vec{a}$ forms a line move the line in a control fashion e.g., its movement is always perpendicular to $\vec{n}$. As a result it forms a plane π.

$\vec{n} \cdot (\vec{v} - \vec{a}) = 0$ ... (2)

Note eqn(2) is general equation, And it is good for Any plane. if $\vec{n}$ is (0,0,1) And $\vec{a}, \vec{v}$ are from $x_w - y_w$ plane, then Eqn(2) is for $x_w - y_w$ plane

Comparison with Shadow example here the general plane π → $x_w - y_w$ plane, A special Case of π



The Normal vector for $x_w - y_w$ plane, A vector Perpendicular to $x_w - y_w$ plane