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

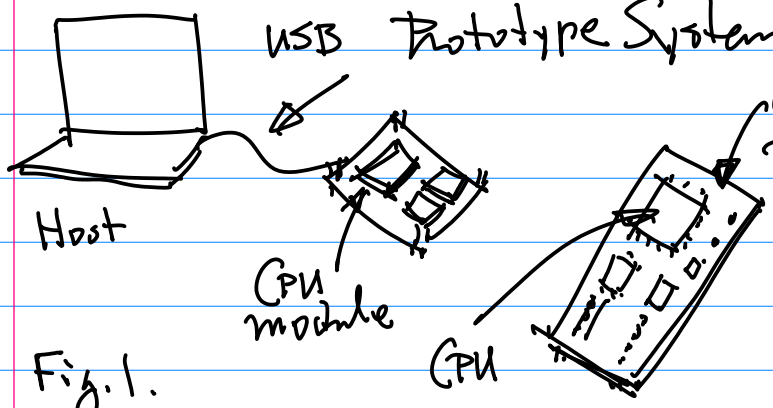


Fig. 1.

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

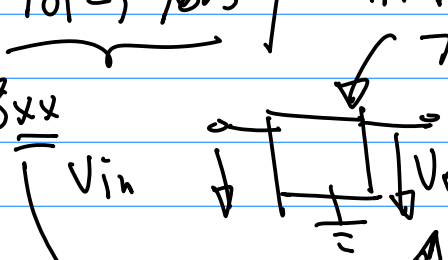


Fig.1. Fig.

"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

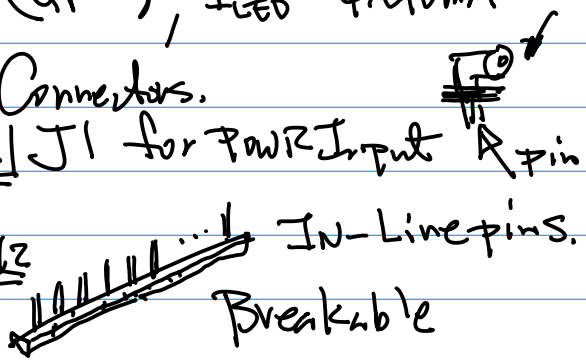
@ 1000mW + 500 mW
= 1500mW
Current Rating.

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

3. "Glue" Components / Resistors a.

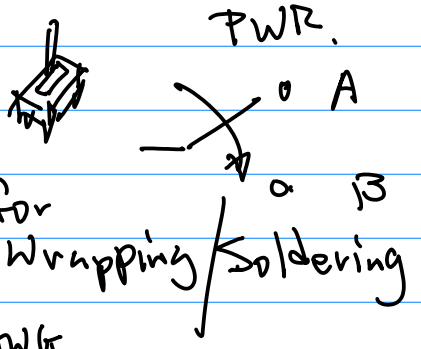
3.
c. LEDs. (Red, Green) for Debugging purpose, for PWR. (GPIO), I_{LED} = 4mA/10mA

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

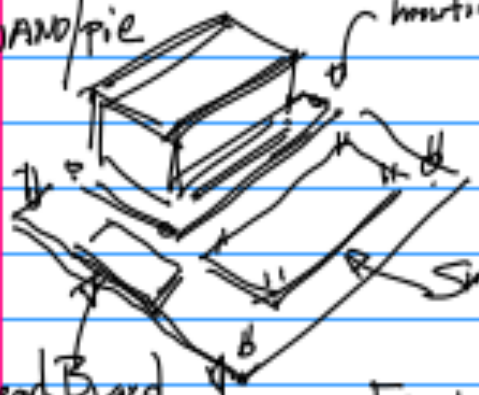


Fig. 1

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
$GPR_X[31:0]$

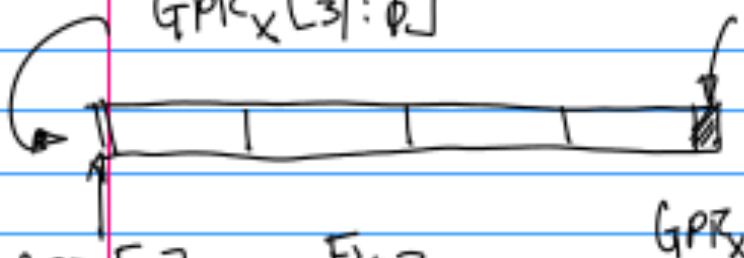


Fig. 2

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.

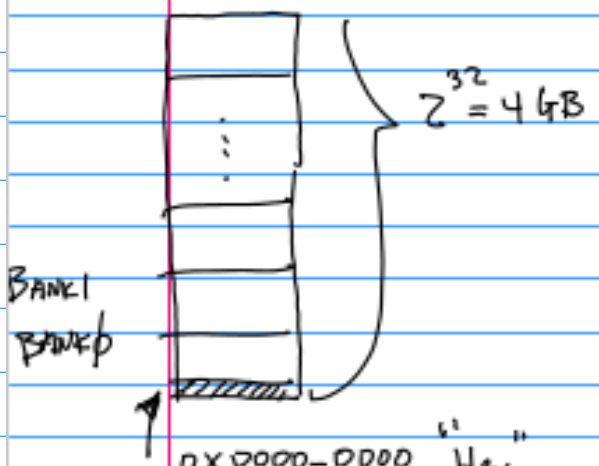


Fig 3.

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

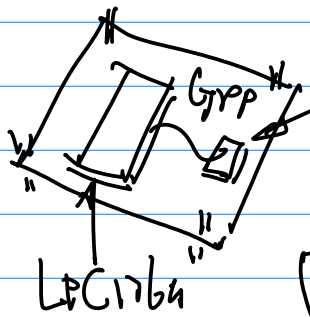
a = Naming convention: Prefix (Two) + Postscript

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

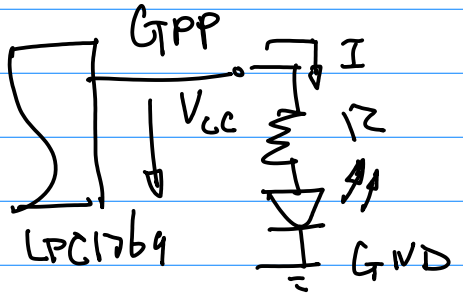


Fig 1.

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

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

$V_{LED} \simeq 1.8VDC$

GPPCON

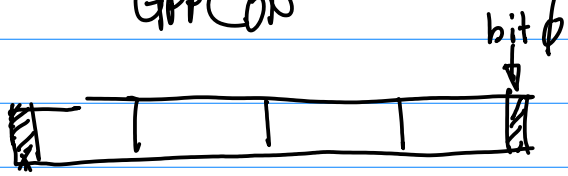


Fig. 2

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

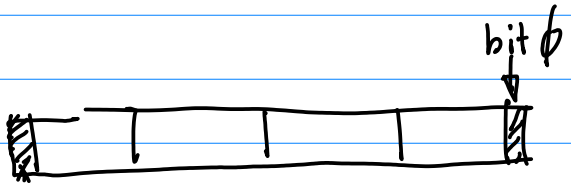
C-Code for Init & Config is required

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)

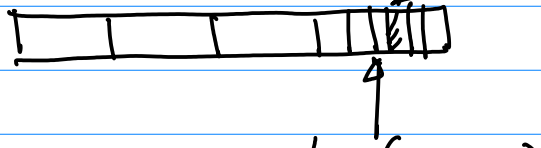
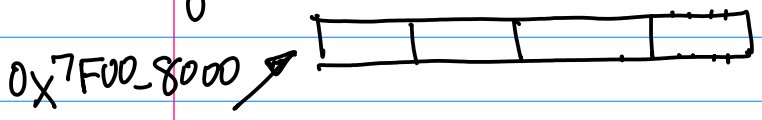


Fig.1

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?

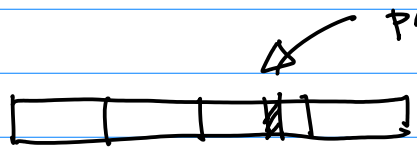
Define bit value for the Output

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" "&=" ?

$\vec{P}(x,y)$ 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: 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);
}
```

Formulation for a straight line

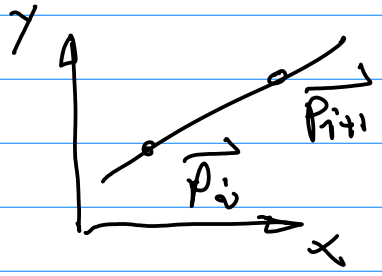


Fig.2

Now, 2D Vector Graphics

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

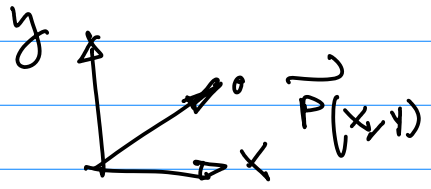


Fig.1

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

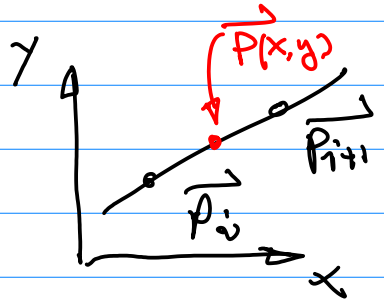


Fig. 2

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 k.E.

Rotating Squares And Trees.

Example: Design A Rotating Squares

Step1. Define 4 vertices/pts

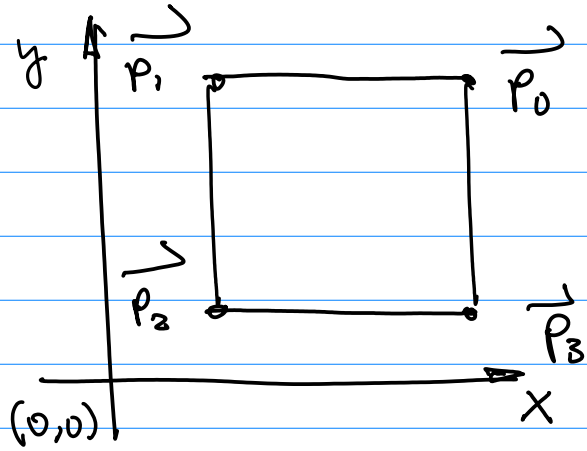
$\vec{P}_i$, $i = 0, 1, 2, 3$

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

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

Based on the physical display device

Fig4



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

Step2. use

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

Prepare: LCD Soldering ON the Wirewrapping 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 Simple code from github/hualili/Cmpe240 20185-to-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 — Tue 3:40-4:40pm.

Due to SJSU off-Campus Program.

Example: Continued from pp15.

Step 2. Use Vector Equation to find 4 pts

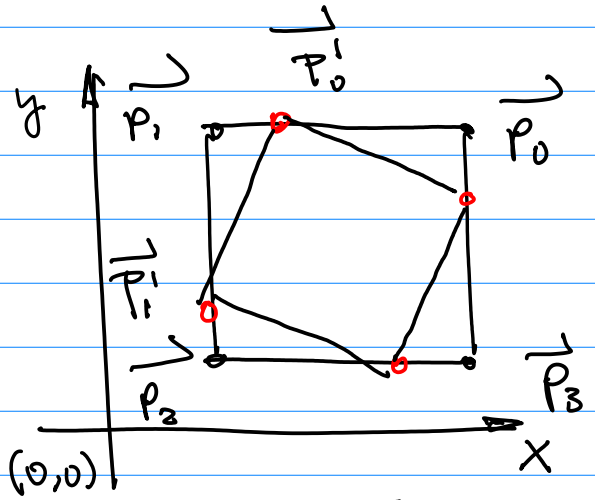


Fig. 1

Let $\lambda = 0.8$, for

line1 ($\vec{P_1}$ 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:

$$x = x_i + \lambda (x_{i+1} - x_i) \quad \ldots (1a)$$

$$y = y_i + \lambda (y_{i+1} - y_i) \quad \ldots (1b)$$

Hardware Implementation of LCD Interface.

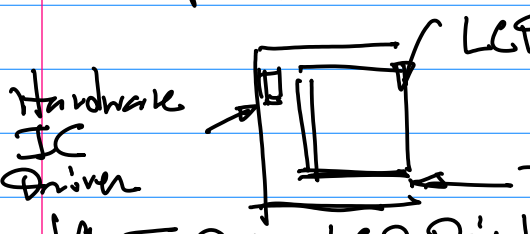
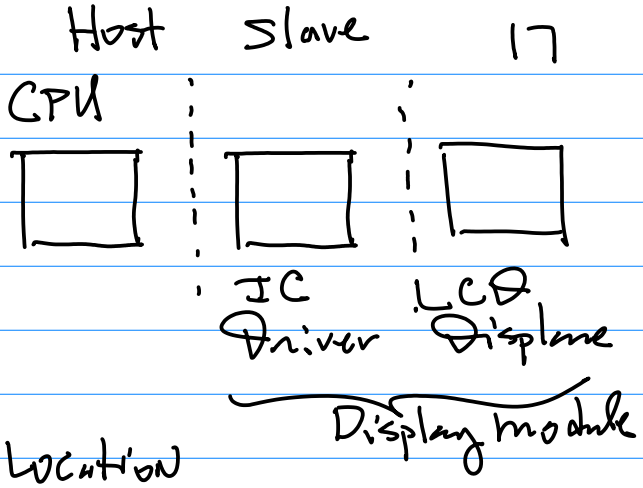


Fig.2



a. To Drive LCD Display. To Display a pixel } (x, y) Location
b. To provide feedBack and Interface to CPU module. } I(x, y) Intensity, and color
(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/hanbili/cmpe240 2018S-9-SPILCD ...

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

SCH

Enable

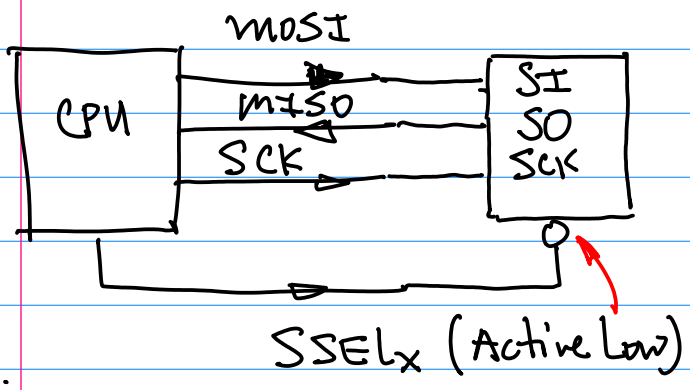


Fig.3.

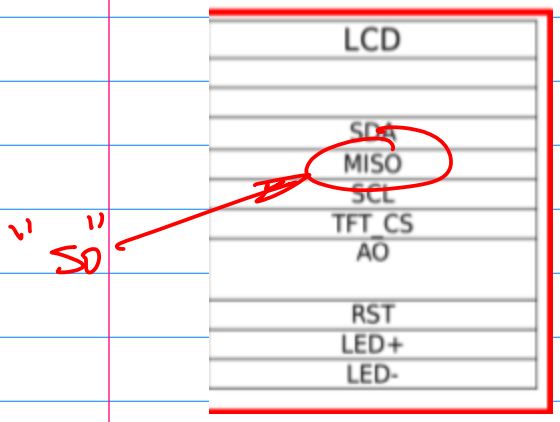
Note: mark the Direction of the Signal

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

Labels from the LCD Display.

SPI pins: SDA, MOSI.
MISO
⋮



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/D̄ Depending on the Signal Level, the Communication From CPU to LCD is interpreted by LCD either as a Command or Data.

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}

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

Example: Draw A Line Code

1. Color Definition. Hex Digits
2 hexs for Each primitive color (red, green, blue)

Primitive Colors: r, g, b

2 Hex Digits: min. 0, max: 255

2 hex → 8 bit → $2^8 = 256$

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

Match to Eqn (3a) & (4b) 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

However Team work is encourayed.

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

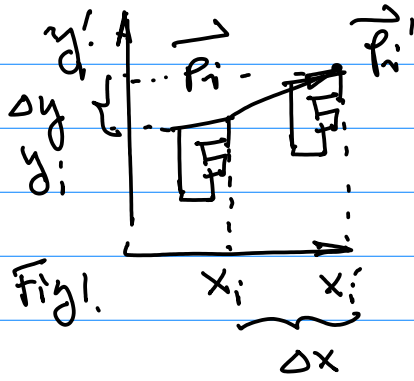


Fig1.

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

$$x_i' \stackrel{?}{=} x_i + \Delta x \quad \ldots (1)$$

After Before

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

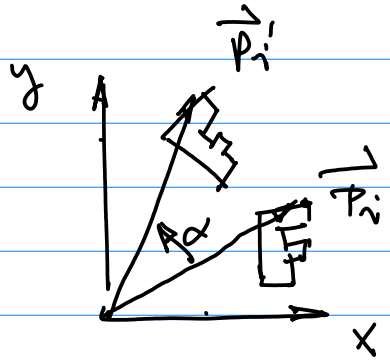


Fig.2

Note: Counter Clockwise Rotation "Positive" Angle α

After Before

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

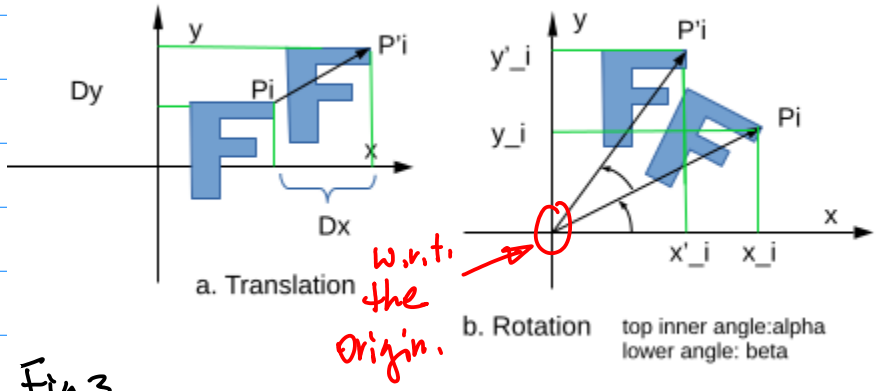


Fig.3

From Eqn (4)

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

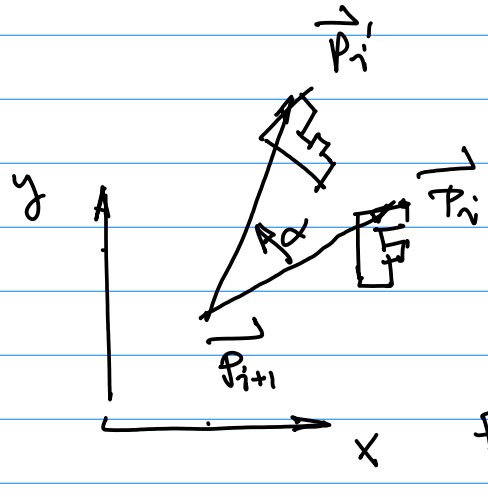


Fig4.

Note: for Rotations in Fig4, 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

After Before

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

where

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

Fig.5

Example: Use 2D Transforms to Create Trees shown Above

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

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

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

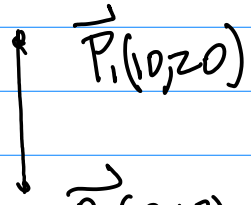


Fig.6

Step 2. Use Vector Eqn to create next level main/major Branch

$\vec{P}_1'(x_1',y_1')$ as in Fig.

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

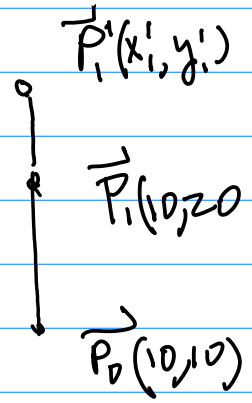


Fig.7

make $\lambda=0.8$.

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

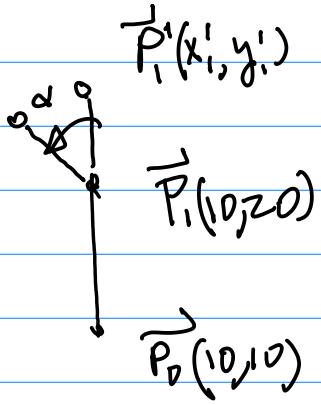


Fig.8

Oct.4 (Monday)

Topics: 1° 2D Example for trees

2° Virtual Display v.s. physical Display, Implementation

Example: Continued from Step 3.

First, Preprocess

$$T=\begin{pmatrix}1 & 0 & \Delta x\\ 0 & 1 & \Delta y\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}1 & 0 & -10\\ 0 & 1 & -20\\ 0 & 0 & 1\end{pmatrix}$$

$\Delta x=-10$

$\Delta y=-20$

Now, Similarly,

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:

$$T^{-1}=\begin{pmatrix}1 & 0 & -\Delta x\\ 0 & 1 & -\Delta y\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}1 & 0 & 10\\ 0 & 1 & 20\\ 0 & 0 & 1\end{pmatrix}$$

From Eqn (5)-(7):

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

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

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

Then for $\vec{P}_1(10,36)$

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

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

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

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

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

$$T^{-1}RT=$$

$$\begin{pmatrix}1 & 0 & -\Delta x\\ 0 & 1 & -\Delta y\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\alpha & -\sin\alpha & 0\\ \sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & \Delta x\\ 0 & 1 & \Delta y\\ 0 & 0 & 1\end{pmatrix}$$

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

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

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

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

Physical Display Coordinate v.s. Virtual Display Coordinate

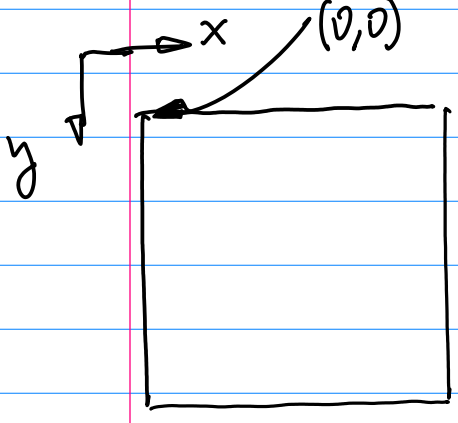


Fig.1

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

b. Resolution is m×N

m×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, ⋯, m-1

y: (Top down) 0, 1, 2, ⋯, N-1

Limitation:

1° No Negative Value in the System.

2° Tied to the physical Device with Resolution m×N.

3° Not Portable

Now, Virtual Display coordinate

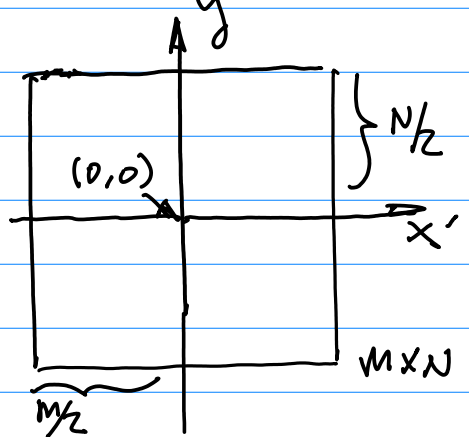


Fig.2

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) mapto your physical display.

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

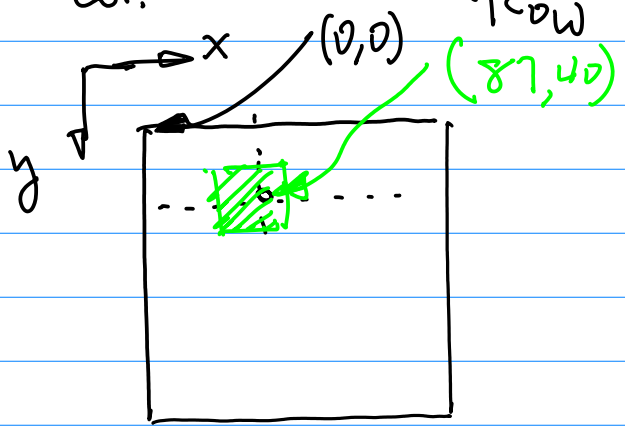
1° Write C code to Realize Eqn (1) & (2).

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

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



3D Graphics Processing Engine Introduction: World Coordinate System

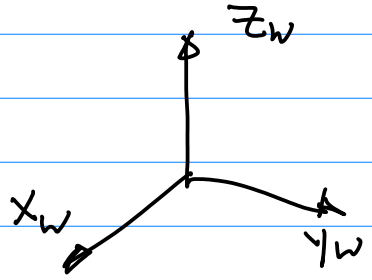


Fig. 1

Outline (Wed)

Topics: 1° Transformation pipeline

Note: 1° Office Hour Update Posted on Lit;

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.

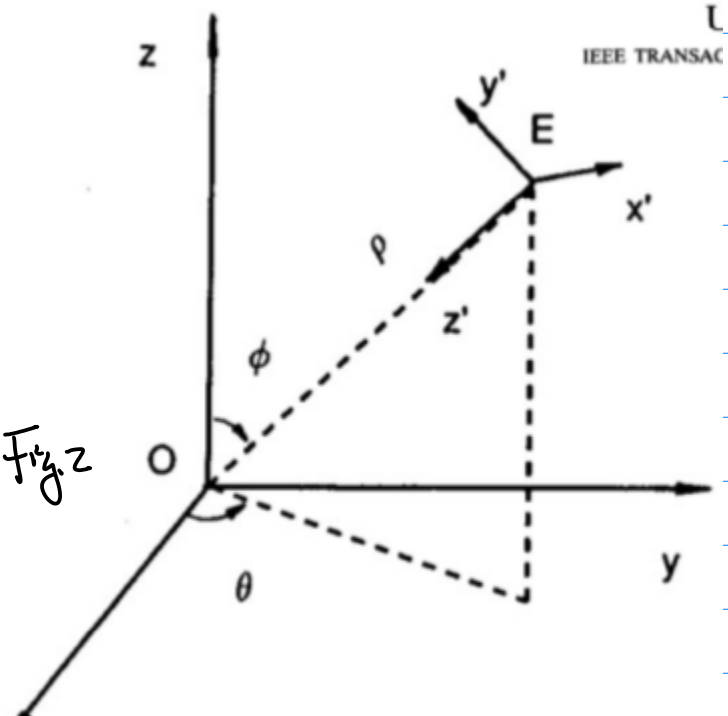
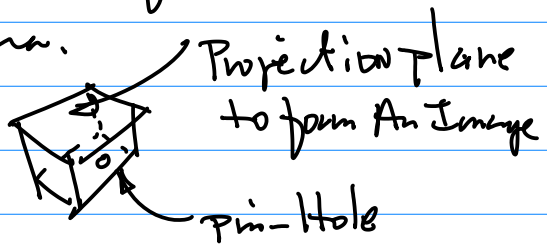


Fig. 2

a. "Pin-Hole" model.

Diameter of the Lens ("Hole") is very small, $d \ll s$.

Enclose to form A virtual Camera.



→ 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

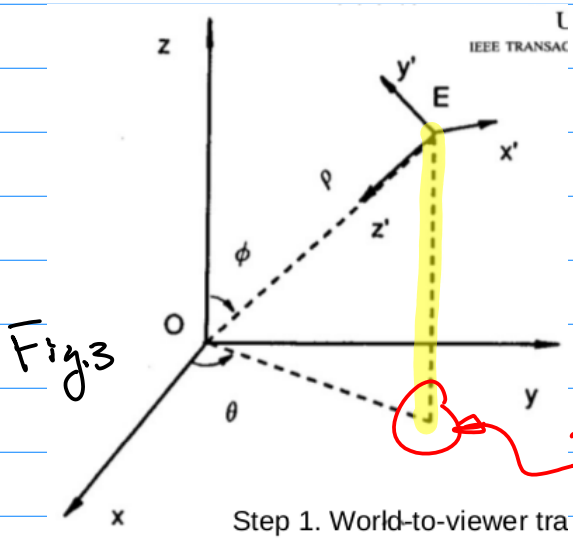


Fig.3

Step 1. World-to-viewer tra

Intersection Pt. $\vec{E}'$

Draw 2nd Line, passes $\vec{E}'$ (on X_w-y_w) to connect to the Origin (0,0,0) of X_w-y_w-z_w. as in Fig

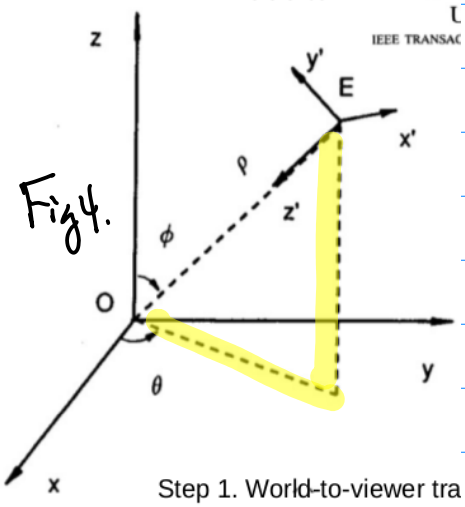


Fig4.

Step 1. World-to-viewer tra

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

Angle

Angle φ (phi): Formed Between $\overrightarrow{OE}$ (Not Exactly the $\overrightarrow{ED}$) 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:

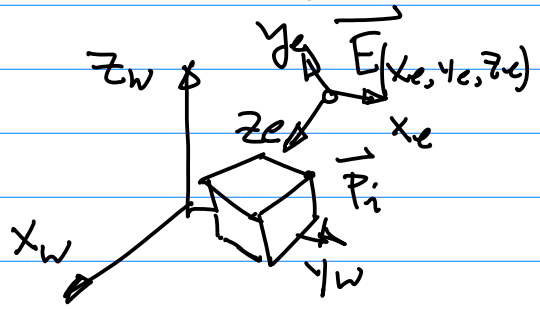


Fig.5

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

$\vec{P}_i$ defined in the X_w-y_w-z_w

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

$$\begin{pmatrix} x_i' \\ y_i' \\ z_i' \\ 1 \end{pmatrix} = \begin{bmatrix} -\sin\theta & \cos\theta & 0 & 0 \\ -\cos\phi\cos\theta & -\cos\phi\sin\theta & \sin\phi & 0 \\ -\sin\phi\cos\theta & -\sin\phi\cos\theta & -\cos\phi & \rho \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \quad \ldots(2b)$$

$\vec{P_i'}$ $\vec{P_i}$

After "in X_e-y_e-z_e System" Camera

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

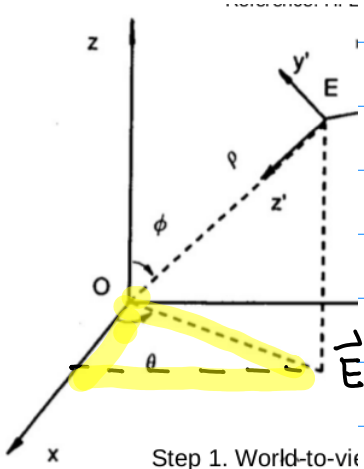
$$\cos\theta = \frac{x_e}{\sqrt{x_e^2+y_e^2}} = \frac{200}{200\sqrt{2}} = \sqrt{2}/2 \quad \ldots(3)$$

(From Fig4, 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

Out.11 (monday)

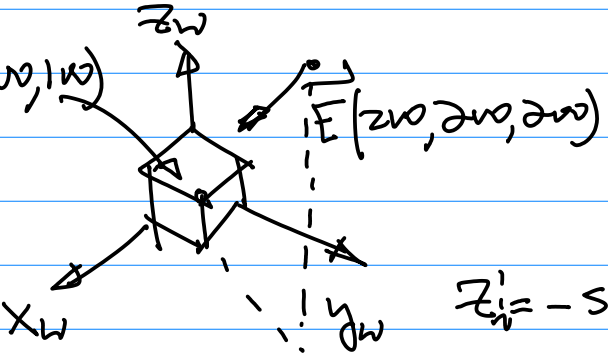
Note: 1° 3Handout material, C Program, with 2021F-111x

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

b. Draw line.c: Architecture → Write a Pixel (location, color/Intensity). Vector Line

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

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



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

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

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

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

$$= 100\sqrt{6}/3$$

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

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

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

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

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

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

(x_i', y_i', z_i')

$= (0, 100\sqrt{6}/3, 100\sqrt{3})$

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

Handout 2021F-11C 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-11Z-~ Review

2D Vector Graphics. to be posted on Line. Question 1 & Question 2.

Homework: DW github/CANVAS

One 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.

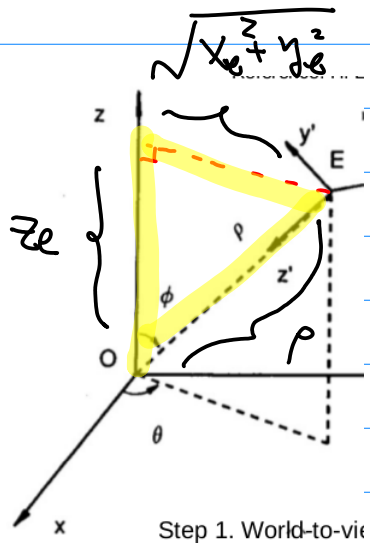
Format of the lectures.

CPU Architectures → CPU Datasheet

Coding Implementation

And Prototype Implementation

Today's Topics: Transformation pipeline.



Step 1. World-to-vie

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.

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

$$= \frac{\sqrt{x_e^2+y_e^2}}{\sqrt{x_e^2+y_e^2+z_e^2}} \Big|_{x_e=y_e=z_e=200}$$

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

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

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!)

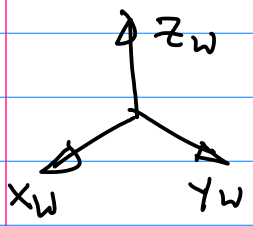
$$x_i'' = \frac{D}{z_i'} x_i' \quad \cdots (2a)$$

$$y_i'' = \frac{D}{z_i'} y_i' \quad \cdots (2b)$$

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

D : focal length.

$D \simeq 20-30$



Draw on 2D Display.

Tree Creation

Oct. 13 (Wed) Class Repo

Ref: go github/hualili/cmpe240

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

Road map Before the midterm

CPU Architecture → GPU → 2D Vector Graphics

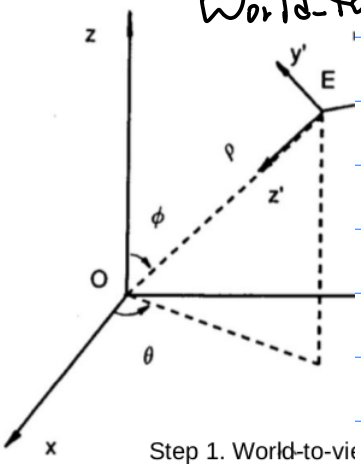
Display 2D G.E. ← Graphics Processing Engine

Driver Design

↓ Hardware → S.P.I. I/F Coding/Programming

3D Introduction

Example:



Step 1. World-to-vie

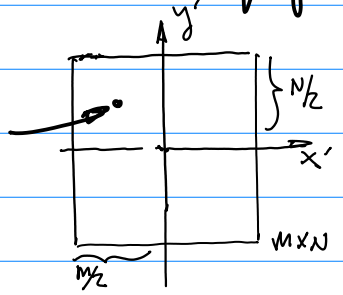
World-to-Viewer Transform

$\vec{P_i} \rightarrow \vec{P_i'}$ (3D)

↓ P.P.

$\vec{P_i''}$ (2D)

Step 2. Perspective projection



$\vec{P_i''}$

(x_i'', y_i'')

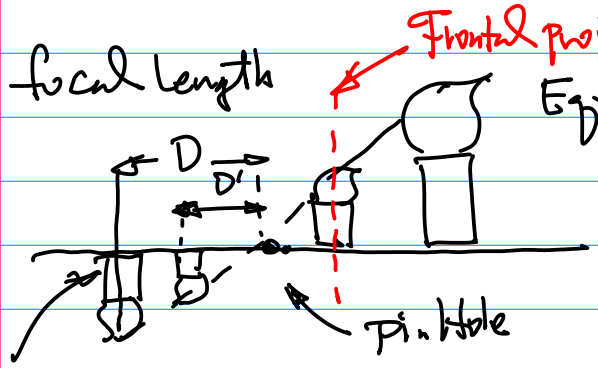


Fig.1.

Eqn(2a), (2b)

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

Back projection plane

Design of Display Driver Hardware & Architecture

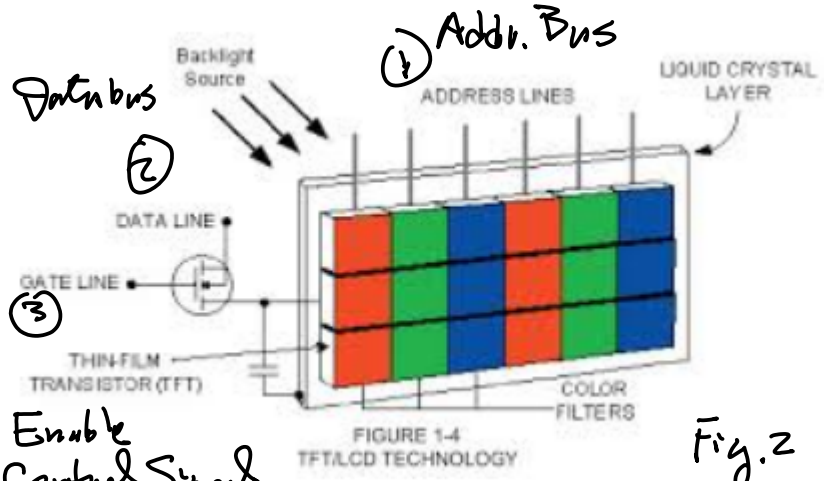


Fig.2

Enable Control Signal

Compute Perspective projection

Let choose $D=20$.

And

$P_i'(x_i', y_i', z_i') =$

$\left(0, 1\text{W}\frac{\text{J}}{\text{s}}, 1\text{W}\frac{\text{J}}{\text{s}}\right)$ PP2.7

Find Perspective Projection from Eq(2a)

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

$D=20$

$$= \left(\frac{20}{1\text{W}\frac{\text{J}}{\text{s}}}\right) \cdot x_i' = \left(\frac{20}{1\text{W}\frac{\text{J}}{\text{s}}}\right) \cdot 0 = 0$$

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

$$= \left(\frac{20}{1\text{W}\frac{\text{J}}{\text{s}}}\right) \cdot 1\text{W}\frac{\text{J}}{\text{s}} \frac{1}{2}$$ In Virtual Display coordinate

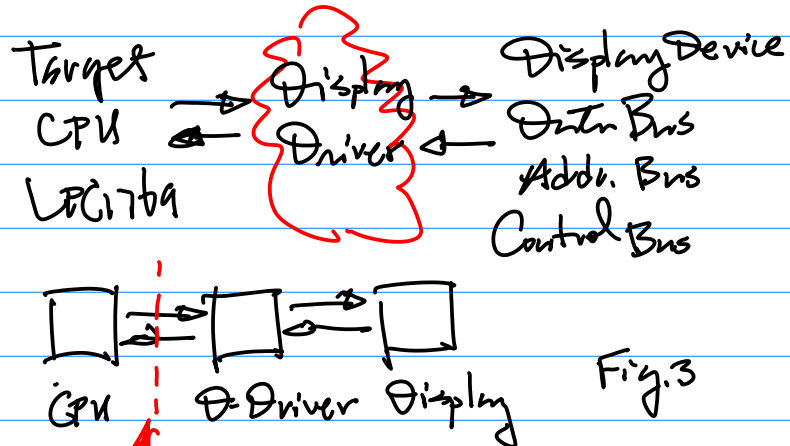


Fig.3

Step 3. For Implementation.

Use Virtual_To_Physical

"3+1" MOSI : Master Out Slave In

MISO : Master In Slave Out

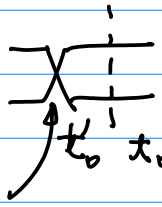
SCK :

CmpE240

SCK: Serial clock.

SSEL: Enable

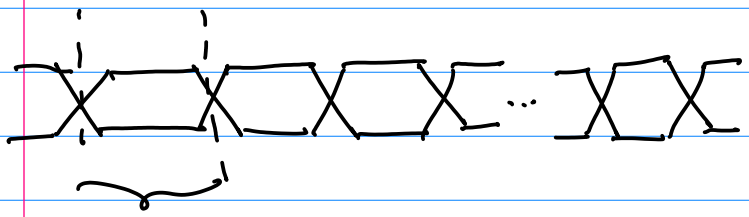
SPI Waveform (Protocol)



"Don't Care"

Fig4.

at t_0' time instant, "changing state".



8 bits

Fig5

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.

Notes: Each Field, for Example

Opcode,

0x9f

To ask for manufacturer's ID & Product ID.

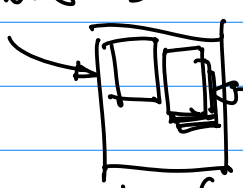
Figure 9-3. Program Sector Protection Register

SPI protocol, 3 Fields

- Opcode : Field1. Instruction Field
- Address Field, Field2
- Data (payload), Field3, Data Field

Controller (MCU)

Control Register



S.P.I. Device (Flash)

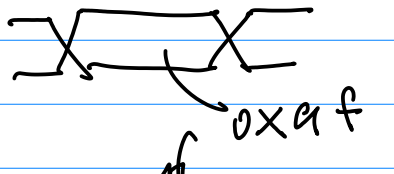
Flash memory

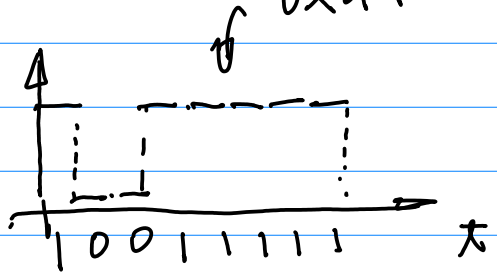
Slave (client)

0x9f

1001 1111

↓ Waveform





Suppose Display Device

Example: Suppose a Display Device

① with the following Resolution 320 x 240

(m x n)

m: Col. per row / pixels per row

n: No. of Rows

Col/Row: 320

Row: 240

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?