

August 23rd (mon).

CMPE240

HARRY LI.

E-mail: hual@sisw.edu

Text message (650) 400-1116

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

Advanced Microprocessor Systems

=

Prototype System
with a CPU module

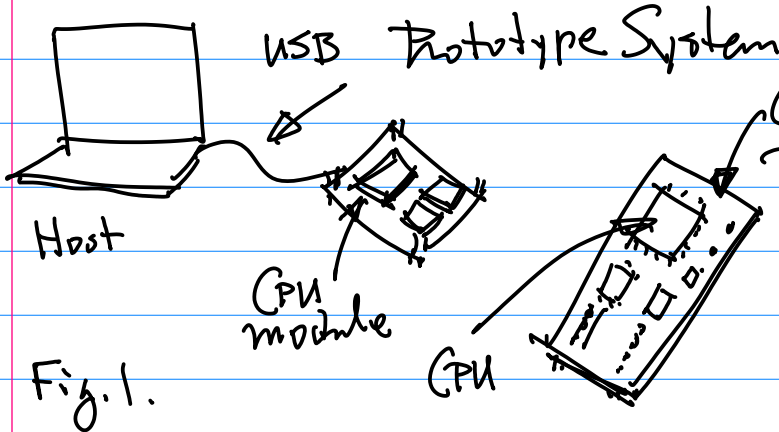


Fig. 1.

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

Text Books, References

1. NXP LPC1764 GPU Datasheet
800+ pages Homework: Download pdf. Before
Next Monday, Aug. 30th.

2. LPC1764 Schematics of the CPU module

3. Nvidia Jetson Nano Datasheet on TX2 (6 CPU + 256 GPU)
400+ pages. 5% Bonus.
(Optional)

4. TISC-V. Open Source Architecture, A Super Set of ARM, FPGA, Verilog, SoC. +RTOS. (Optional)

A Proposal (One +5% Paragraph) By Sept. 1st (Wed). Submit to my Email;

Note: Buy LPC1764 CPU module.

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

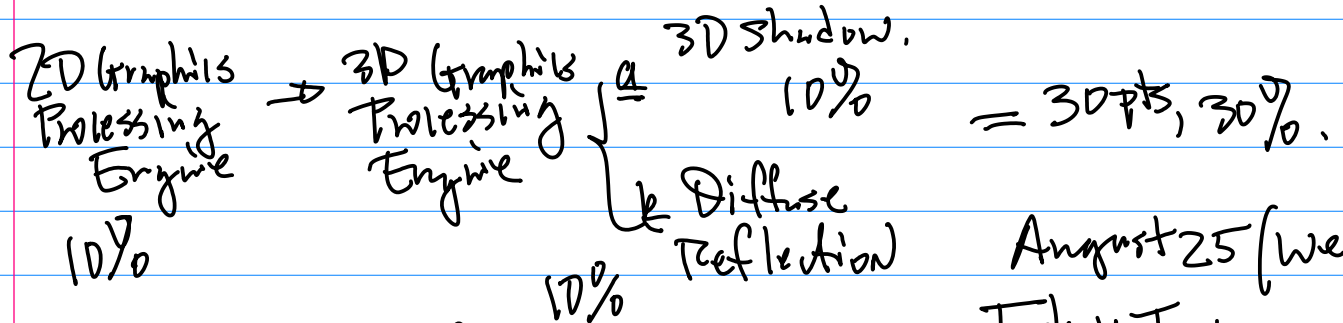
Grading Policy & Projects
2 projects (Phase I & II)

2D Graphics Processing Engine

3D Graphics Processing Engine

CMPE240

2.



midterm: 30%, Final 40% (Comprehensive)

August 25 (Wed)

Today's Topics:

Option 1. (5%+) NVDA NANO

1° Bill of material

a. Likely Devices Drivers, O.S. C/C++, Python.
b. I/O Interface: "EdgeAI"
GPIO, SPI.

Reference: github/hnukili/
CMPE240/2018F

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

The B.O.M.

1. CPU module NXP LPC1114

↓
3rd Party (Digital Art), module
to Distributors

Digkey.com, Mouser.com
etc.

Expecting Delays.
Lead Time over 8 weeks

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

Alternative { Re-use the previously
used module
Team (4 person)

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;

Each person will need to have
his/her Board;

3° Late Project, 10% per week;

Option 1: NANO. a 4400+
pages
"firmware" Datasheet

Tool for
Flashing the
CPU module

b Jetpack 4.3 or Higher
(O.S. + Libs. + Packages)

CMPE240

= Coding in Both user & kernel Spaces. \rightarrow O.S. Distr.

Tool chain, Device Driver Debugging & Development;

Option 2. ITC-V. verilog, FPGA.

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

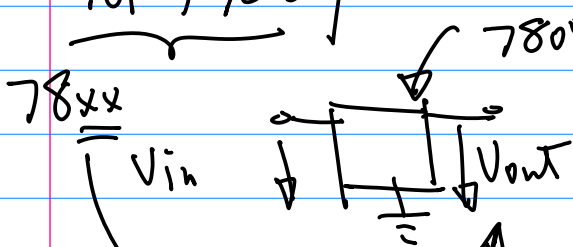


Fig. 1.

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

$V_{in} \geq V_{out} + 1.5 \text{ VDC}$... (i)
DC Voltage Source

a About 7805
1000 mW.

b 7.5 VDC

OR

9. VDC @ 1000 mW + 500 mW
= 1500 mW

= Why Do we use it?
Current Rating.
Rating

\rightarrow Deploy the System.

3 "Blue" Components Resistors a

3.
= LEDs (Red green) for Debugging purpose, for PWR.
(GPIO), $I_{LED} = 4 \text{ mA}$

= Connectors.

d J1 for PWR Input & pin

d2 IN-Line pins.
Breakable

to mount CPU module.

= 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.T. Lib.

5. "Other" thing.

RJ-45 Connector

Sept. 8 (W)

Topics: 1. "Hello, the world" program

Hardware Implementation

NXP MCU Xpresso.

a. Installation of MCU Xpresso.

b. [github/hualili/CMPE240/2018F](https://github.com/hualili/CMPE240/2018F)

LTC1769 Patch, Import this patch to your Xpresso.

Prototype Board Build Up

External Power CKT (Red LED should be included)

GPP Testing CKT

a. Wire Whipping Board (LTC/NAND Pie)

b. Stand-offs.

Implementation/Design of the CKT.

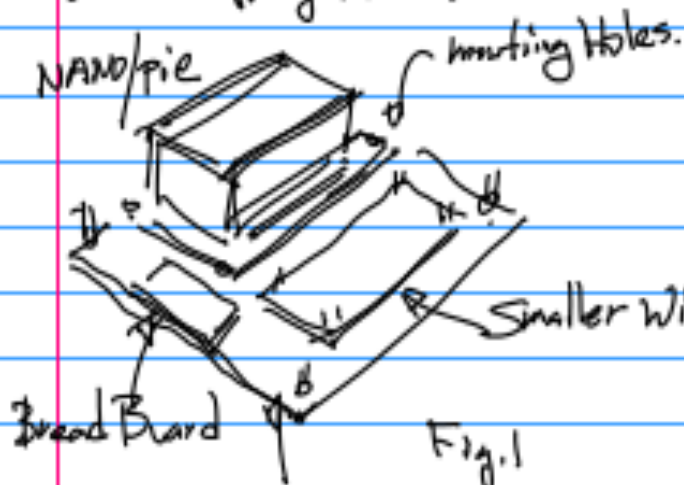
Architecture Aspects.

CPU Architecture, M. Map.

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

Note: Wire Whipping Board with "Stand-offs" (legs)

Homework: Next show-and-tell Wire Whipping Board;



Wire Whipping Board "Carrier" Board

"TAP Plastic"

On the Board: a. Stand-offs.

b. Connector(s) for External power

→ PIC (7805), with Red LED

CPU Architecture:

1. 32-bit Architecture

CPU Architecture

a. ALU 32bit Arithmetic/Logic Unit.

b. Register File,

A Bank of Registers. 32 bits GPRs

General Purpos Registers

Those Registers that can participate. Any meaningful

Arithmetic/Logic Operations.
Special Purpose Registers.

SPRs 32 bit

To Define/Determine the Behavior of peripheral
Naming Convention: Controllers.
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)

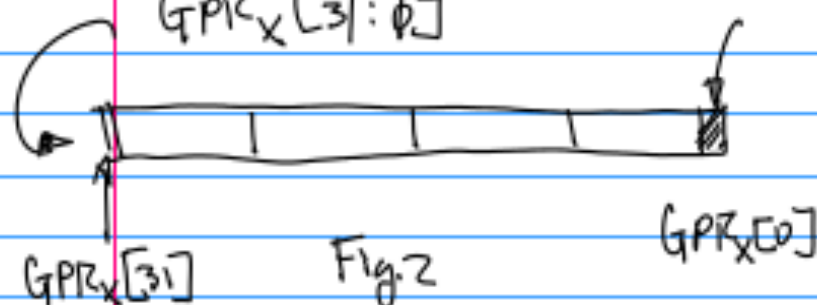
C. Data Bus "Bi-Directional" 32 bits
Information Flowing Both Directions.

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

Notation: 32 bit Register

$GPR_x[31:0]$

LSB



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

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

Note: "Little Endian"

LSB is a/b,

2° Byte Addressable machine
is a machine whose
Smallest memory cell
with an unique address
is a single Byte.

Total memory:

$$2^{32} = 2^2 \cdot 2^{16} \cdot 2^{16} \dots (1)$$

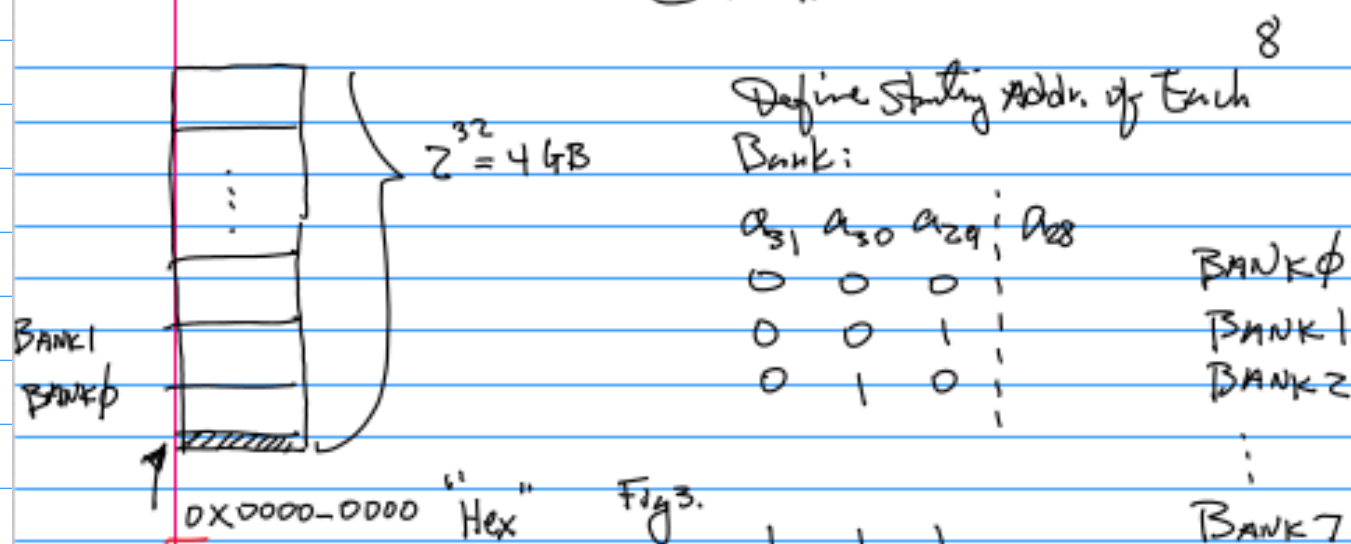
$$2^{10} = 1K, \quad 2^{20} = 2^{10} \cdot 2^{10} = 1M \dots (2) \quad \dots (3) \quad M$$

$$2^{30} = 1M \cdot 1K = 1Gig \dots (4)$$

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

3. Memory map.

CMPE240



32 bits for the Address
8 bits for this memory

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

a. Power-up Address:

CPU will fetch the 1st
Executable from this memory
Location.

→ 0x0000-0000
for ARM

For BANK0: 0x0000-0000

BANK1: 0x2000-0000

.. 2: 0x4000-0000

Note: for x86, the Power-up
Address: 0xFFFF-FFFF

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

b. Which memory Bank holds
this GPIO? BANK1
whose starting Address is
0x2009-C000

How many Bits Do we need to
Uniquely define Each Bank?

3 bits → $a_{31} a_{30} a_{29}$

Sept 13 (Mon)

1^o 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 \rightarrow 8 BANKS
PPL3

1st 256 KB
= Flash

0X0000-0000,

Rest of the Devices, such as Mem. Controller, Peripheral Controller

\downarrow
Peripheral controllers on the mem. map.

APB \neq APB

S.P.R.s.

"CON" 3 letter

a Naming conversion Prefix Root Postscript

3 letters 3 letters 3 letter

"SPICON" \rightarrow "SPICON001" for Example \rightarrow C compiler/C code

b Definition: Are those S.P.R.s for the init & Config of a Peripheral Controller.

Example. GPP

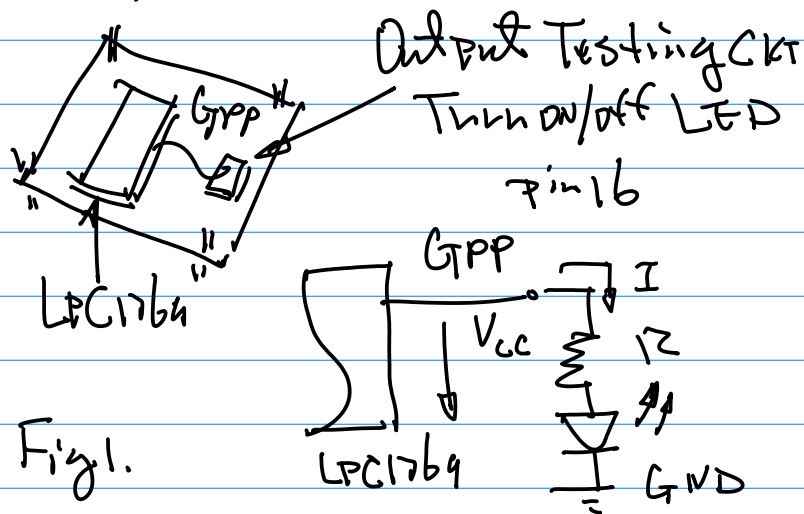


Fig.1.

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

$$I \geq 8mA, \rightarrow R \approx 2k\Omega \quad 300\Omega$$

$$V_{LED} \approx 1.8VDC$$

GPPCON

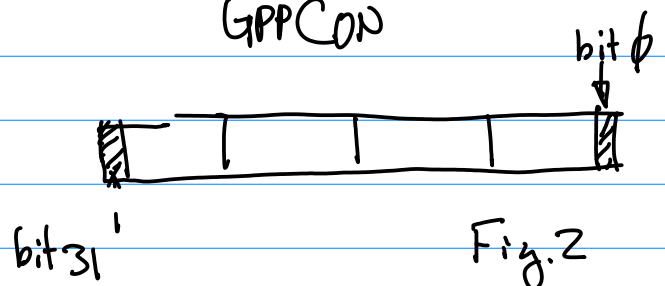


Fig.2

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

CMPE240

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

Reference: 1° 2021F-105 ~ on GP10
C-Code for Init & Config
is required

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

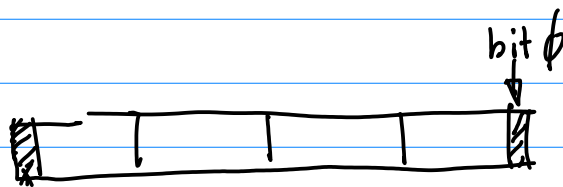
Example: Make GPP for
Input & Output

We have to use the following
init & Config pattern:

Testing.

Hardware

Software { NXP MCU Expresso
Import GPP Sample
"zip"



0X F2bb_F Fbb

To make pin 16 as an output.

Design Step 1.

Identify/select GPP/GPP-pins
Pb2, Pb3

(Connector → CPU → Selection
DataSheet

First, Port the Architecture
Compiler to the target

#define GPPCON

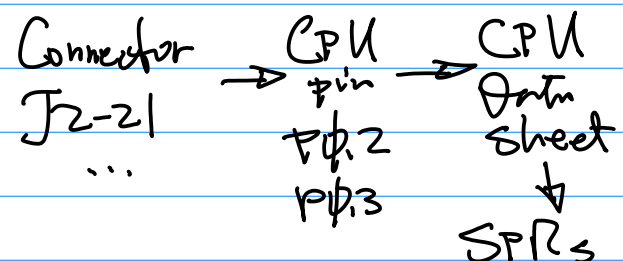
Step 2. Define Pb2 Output,
Pb3 As Input.

Design the Hardware

then, Copy 0X F2bb_F Fbb into
this memory location.

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

Homework: Show + Tell
By Next Week Installation of MCU
+ Import LFC1769.



Sept 15 (w) Architecture
Today's Topics: GP10 Design

Note: SPRs commonly defined/ utilized are

GPx CON

Where $x = A, B, C, D, \dots$

GPFCON

GPFDAT (32 bits \rightarrow 32 pins)

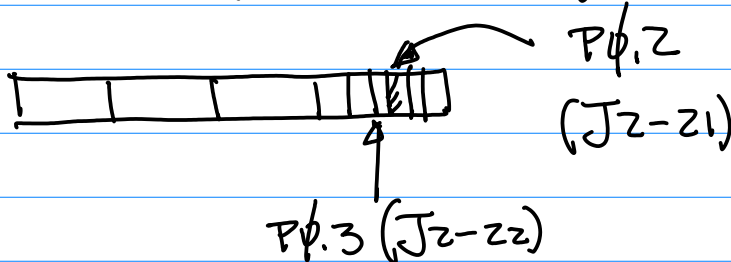
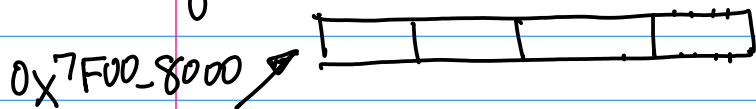


Fig.1

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

Example, Samsung ARM11 Datasheet
pp3/2

Fig.2



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

Sept. 20 (mon) Topics: 1^o GPFD

SPRs, IDE, Sample Code;

2^o 2D GrE.

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

1. Naming Convention in C Compiler

```
LPC_GPIO0->FIOCLR
LPC_GPIO0->FIOSET
LPC_GPIO0->FIODIR
```

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.

Chapter 9, pp 129

PINSEL[5:4] for P0,2

PINSEL[5:4] = 00 for I/O

PINSEL[5:4] = 01 for UART Tx

pp 133 FIODIR Example.

P0,3 output, Find SPR?

Define bit values
for the Output

Table Look up.

FIO0DIR0

FIOSET, CPU Datasheet
Look up.

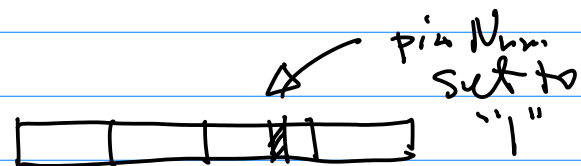
```
void GPIOinitOut(uint8_t portNum,
uint32_t pinNum)
```

```
{
if (portNum == 0)
```

```
{
LPC_GPIO0->FIODIR |= (1 <<
pinNum);
```

```
else if (portNum == 1)
```

```
{
LPC_GPIO1->FIODIR |= (1 <<
pinNum);
```



$1 \ll \text{pinNum}$ // set Direction
to pin Num

Logic Operation $\left\{ \begin{array}{l} 1 = \text{"Bitwise"} \\ \& = \text{"OR"} \end{array} \right.$

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);
    }
}
```

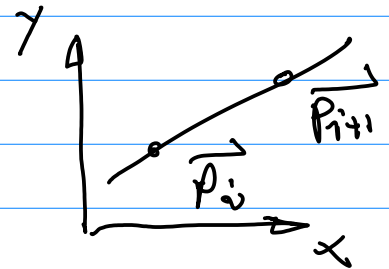
Turn ON LED
(Output "1")

$\overrightarrow{P}(x,y)$ Notation
 $\overrightarrow{P}(x,y) = (x,y)$
 $\overrightarrow{P}_i \rightarrow \overrightarrow{P}_i(x_i, y_i) \rightarrow (x_i, y_i)$
 point(s), Vertex, Vectors
 $\overrightarrow{P}_i = \overrightarrow{P}_i(x_i, y_i) = (x_i, y_i)$

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



Now, 2D Vector Graphics

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

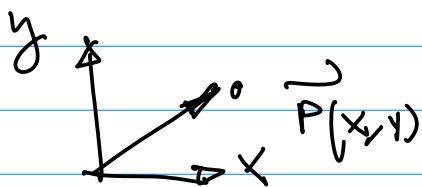
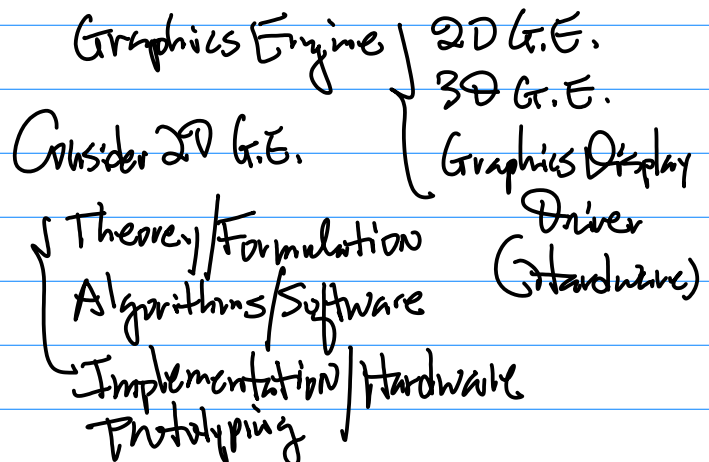


Fig.1

Sept. 22 (W)

Fig.2



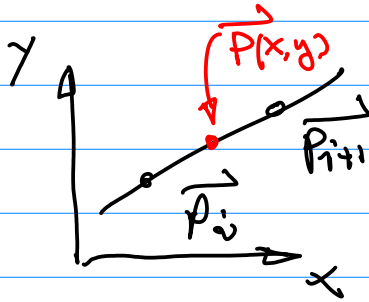


Fig. 2

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

Let's define a direction vector

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

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

Example: Suppose given a starting

$$\text{Pt. } \vec{P}_i(x_i, y_i) = (3, 4.5)$$

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

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

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

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

$$\begin{aligned} \text{direction-x} &= x[i+1] - x[i]; \\ \text{direction-y} &= y[i+1] - y[i]; \end{aligned}$$

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

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

\uparrow Starting pt \uparrow scalar \uparrow Directional vector

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

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

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

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

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

Screen Saver Design for LFC
2D G.E.

Rotating Squares And Trees.

Example: Design of Rotating Squares

Step 1. Defined 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



Fig.3a



Fig.3b

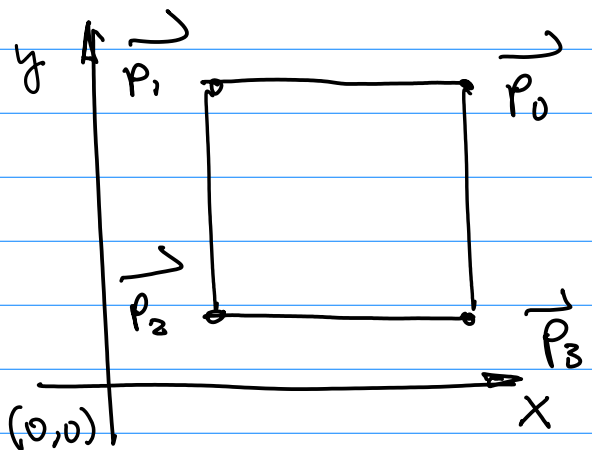


Fig.4

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

Step 2. use

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

Prepare: LCD Soldering on the WinWrapping Board, Input Single line Drawing Project.

Sept. 27 (Mon)

Homework, 2 pts. Due 1 week from Today

Topics: 2D Screen Saver Design

Requirements:

a. Build LCD Hardware

Interface;

b. Input Sample code from [github/malili/CmpE240](https://github.com/malili/CmpE240)

2018S-to-LCD-DrawLine.

Modify the code to Display 2D

Rotating Squares Using 2D
Vertex Equation;

Submission:

= c. Project (Zip, Exported)

d. Screen photo

Submission to CANVAS.

Announcement:

Office Hours — Tue 3:40-4:40pm.

Due to SJSL off - Campus
Program.

Example: Continued from pp15.

Step 2. Use Vertex Equation to find 4 pts

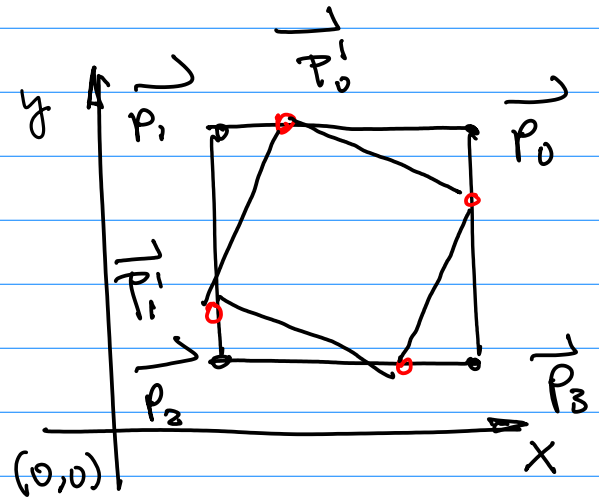


Fig. 1

Let $\lambda = 0.8$, for

line 1 (\vec{P}_1 and \vec{P}_0): 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)

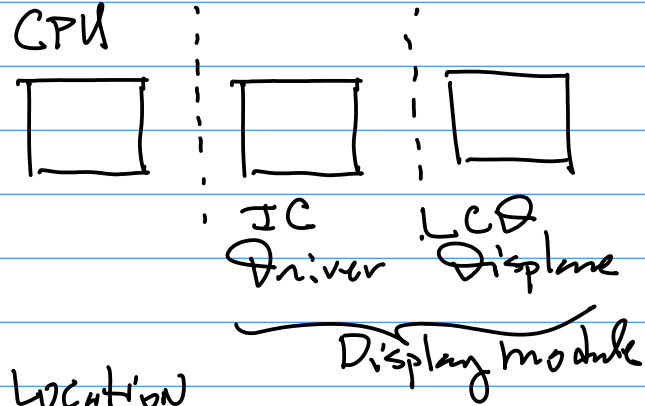
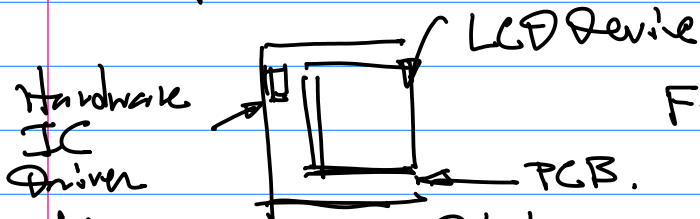
In Homework, level ≥ 10 .

Coding:

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

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

Hardware Implementation of LCD Interface.



- To Drive LCD Display To Display a pixel
 - To provide feedBack and Interface to CPU module.
- (SPI Interface)
- (x,y) Location
- I(x,y) Intensity, and color

To Establish Interface, SPI (Serial Peripheral Interface)

Hardware pins of SPI : 3+1.

Now, Consider the I/F to LCD module.

Ref: [github/mudili/cmpe240](https://github.com/mudili/cmpe240)
2018S-9-SPI LCD ...

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

LPC-1769	
Label	Pin
MOSI ✓	P0.18
MISO ✓	P0.17
SCK ✓	P0.15
CS Enable	P0.16
GPIO-DC (Data / command)	P0.21
GPIO-Reset	P0.22
3.3V	
GND	

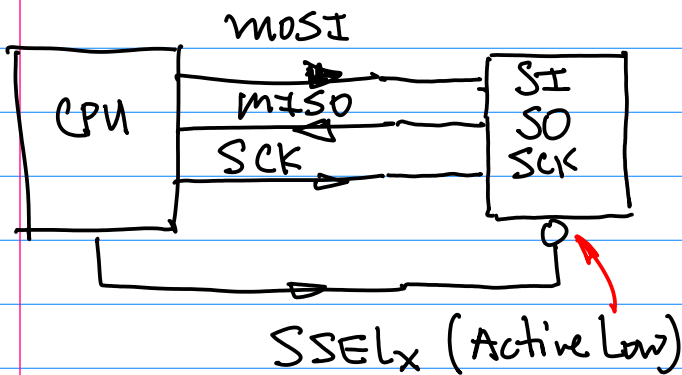


Fig. 3.

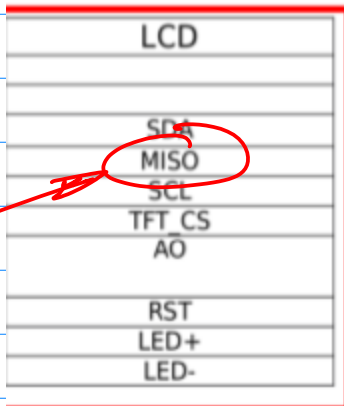
Note: Mark the Direction of the Signal

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

"SD"



Type: Change MISO on LCD to SD, etc.

Note: In addition to SPI interface, identify Command/Data Toggle Control pin, the label should be

C/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 $\begin{matrix} \uparrow \\ P_0 \text{ or } \\ P_i \end{matrix}$ $\begin{matrix} \uparrow \\ P_1 \text{ or } \\ P_{i+1} \end{matrix}$

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 \rightarrow 8 bit $\rightarrow 2^8 = 256$

Match to Eqn (14) & (16)
to Build a Square one
Line at a time.

2^o Bit Arrangement for the primitive

Colors: RGB = (2 hex)(2 hex)(2 hex)