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
$\neq$
Prototype System
with A CPU module

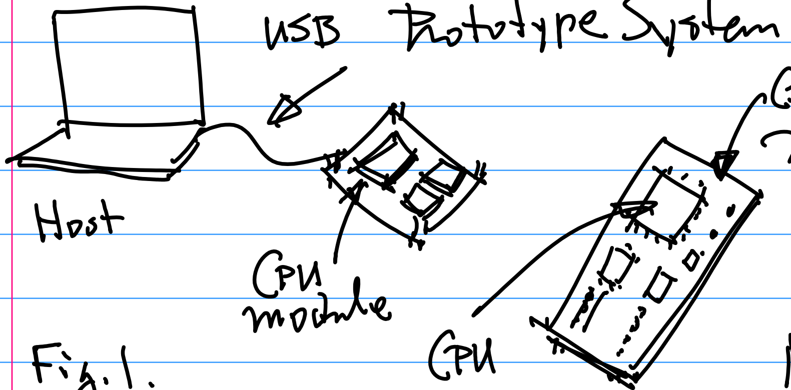


Fig. 1.

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)
400+ 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)

$$\begin{tikzcd} \text{2D Graphics Processing Engine} \arrow[r] & \text{3D Graphics Processing Engine} \end{tikzcd}$$

a
b

CmpE240

2D Graphics Processing Engine 10% → 3D Graphics Processing Engine { a̲ 3D shadow. 10% ; b̲ 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;

Tool for Flashing the CPU module → b̲ Jetpack 4.3 or Higher (O.S. + Libs. + Packages)

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 (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. a̲ 4400+ pages
"firmware" Datasheet

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

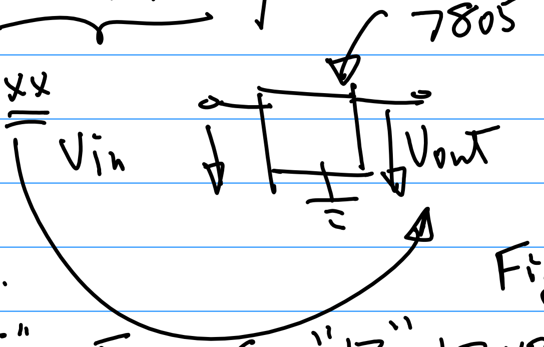


Fig. 1.

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?

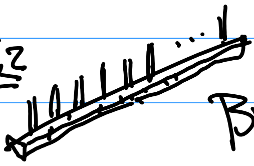
→ Deploy the System.

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

d Connectors.

d1 J1 for PWR Input 4 pin

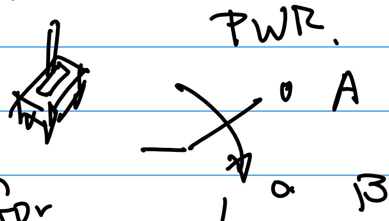
d2 IN-Line pins. Breakable



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

CMPE240

Sept. 8 (W)

Topics: 1° "Hello, the world" program

Hardware Implementation: 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.

NXP MCU Xpresso.

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

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

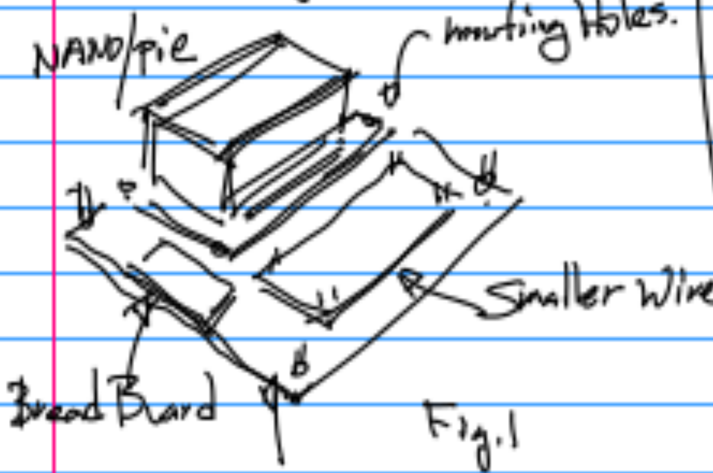


Fig. 1

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 Purpos Registers
Those Registers that can Participate Any meaningful

CMPE240

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 Charateristics)

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

$GPR_x[31:\phi]$

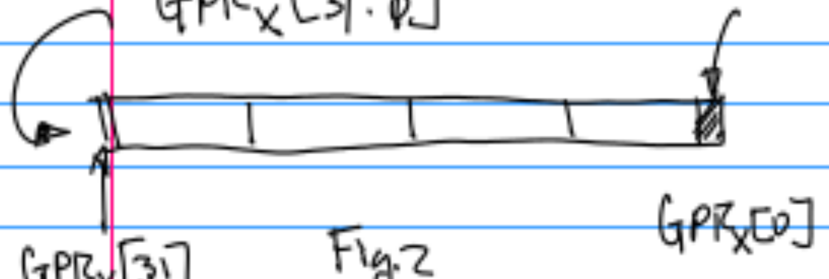


Fig.2

For Address Bus, $Addr[31:\phi]=$

$a_{31}a_{30}\cdots a_1a_0$

Note: "Little Endian"
LSB is a_ϕ,

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

Total memory:

$2^{32}=2^2\cdot2^{10}\cdot2^{10}\cdot2^{10}$ ...(1)

$2^{10}=1K$, ...(2) $2^{20}=2^{10}\cdot2^{10}=1M$ ...(3)

$2^{30}=1M\cdot1K=1giga$ ...(4)

$2^{32}=2^2\cdot2^{30}=4GB$

3. memory map.

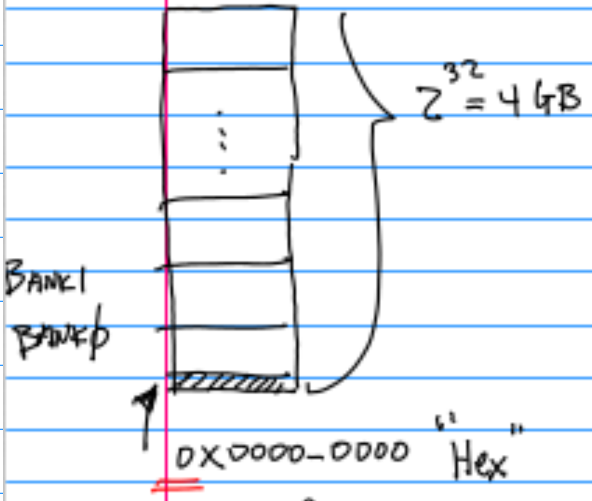


Fig 3.

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_FFFφ

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		BANKφ
0	0	1		BANK1
0	1	0		BANK2
				⋮
1	1	1		BANK7

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

For BANKφ: 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

CmpE240

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

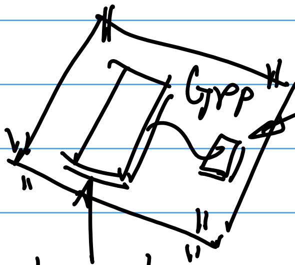
1st 256 KB = Flash
0x0000_0000,
Rest of the Devices, such as Mem. Controller, Peripheral Controller
↓
Peripheral controllers on the mem map. → SSP1 (SPI) Peripheral Controller
APB0 & APB
↓
S.P.R.s. ← 0x4003_0000
Size 4000
"CON" 3 Letter

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



Output Testing CKT
Turn on/off LED
pin 16

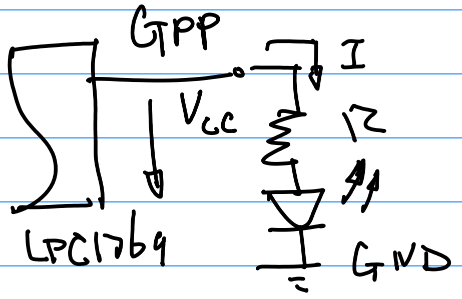


Fig 1.

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

$I \simeq 8mA$, → $R \simeq 200$ ~ 300Ω
$V_{LED} \simeq 1.8VDC$

GPPCON

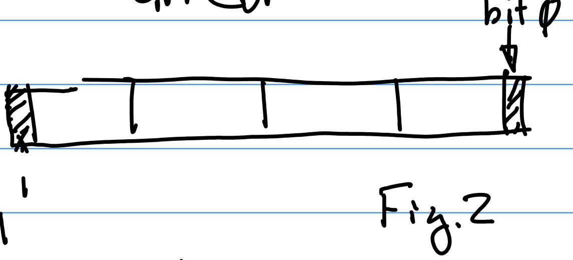


Fig. 2

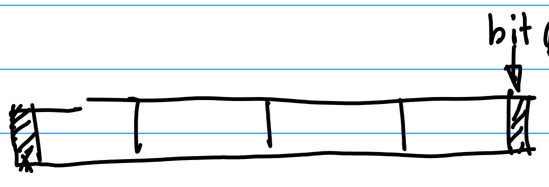
Where to find GPPCON on the memory map? → Addr. of GPPCON is described on CPU Datasheet.

CmpE240

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

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

we have to use the following init + Config pattern:



bit 31

0X F2$\phi\phi$_FF$\phi\phi$

To make pin 16 as an output.

First, "port" the Architecture Compiler to the target

#define GPPCON $\rightarrow$ Addr. from the memory

then, Copy 0XF2$\phi\phi$_FF$\phi\phi$ 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
  - Input GPP Sample ".zip"

Design Step 1.

Identify/select GPP/GPP-pins Pϕ.2, Pϕ3

(Connector $\rightarrow$ CPU Datasheet $\rightarrow$ Selection

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

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

$$\begin{tikzcd} \text{Connector J2-21 } \ldots \arrow[r] & \text{CPU pin P}\phi.2\text{ P}\phi.3 \arrow[r] & \text{CPU Data Sheet} \arrow[d] \\ & & \text{SPRs} \end{tikzcd}$$

Note: SPRs commonly defined/ utilized are

GPx CON

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

GPFCON

GPFDAT (32 bits ⇒ 32 pins)

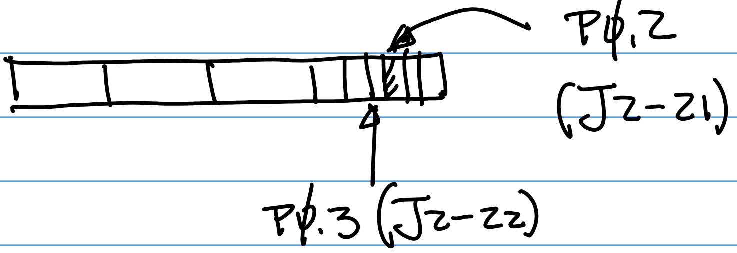
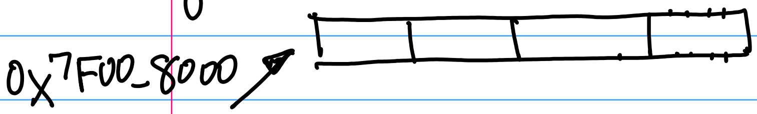


Fig. 1

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

Example, Samsung Arm11 Datasheet pp312

Fig2

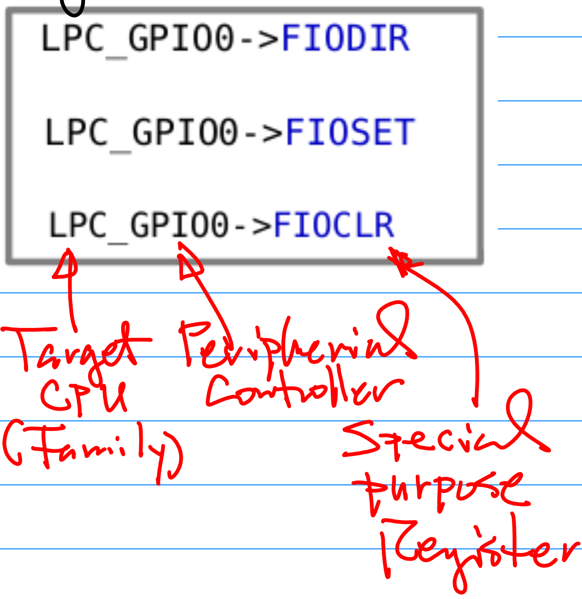


Question: ~~Define~~ Find Binary Pattern for GPA CON to make its pin 2 as an output?

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

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

1. Naming Convention in C Compiler



```
LPC_GPIO0->FIODIR

LPC_GPIO0->FIOSET

LPC_GPIO0->FIOCLR
```

CmpE240

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

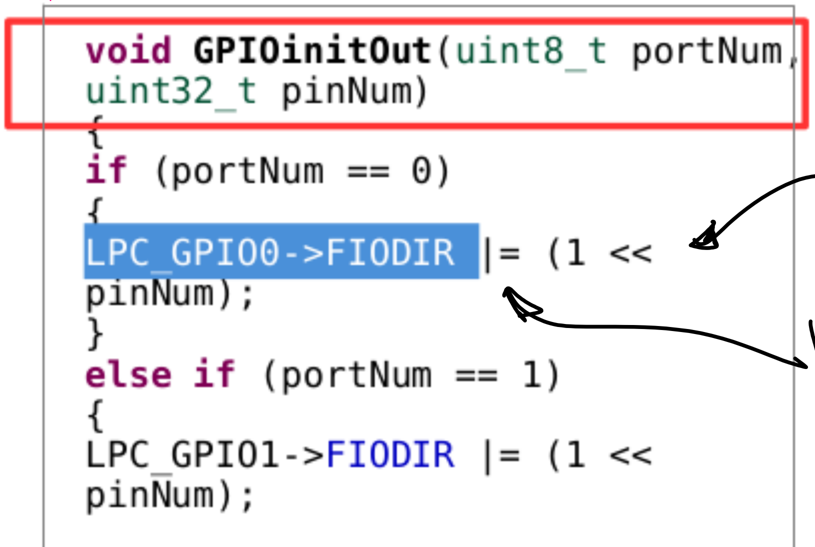
pp 133 FIODIR Example.

P0,3 output, Find SPR? Define bit Value for the Output

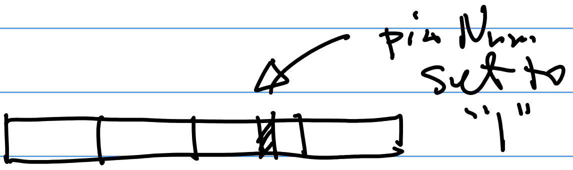
Table Look up.

FIO0DIR0

FIOSET, CPU Datasheet Look up.



```
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);
```

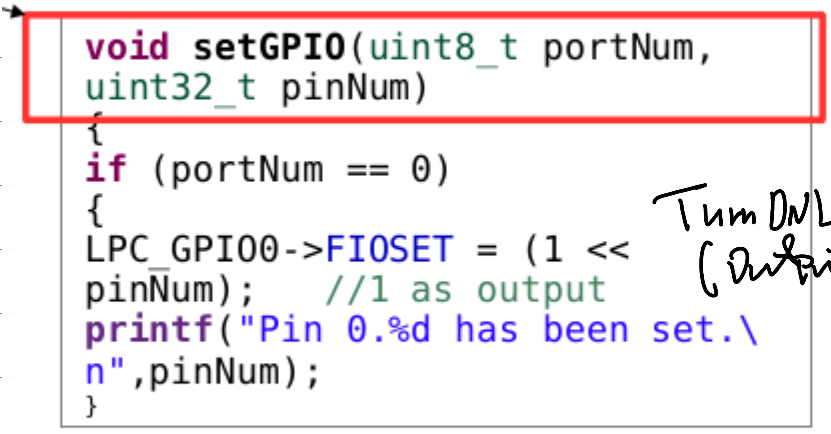


1 << pinNum // set Direction to pin Num

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

CmpE240

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")

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

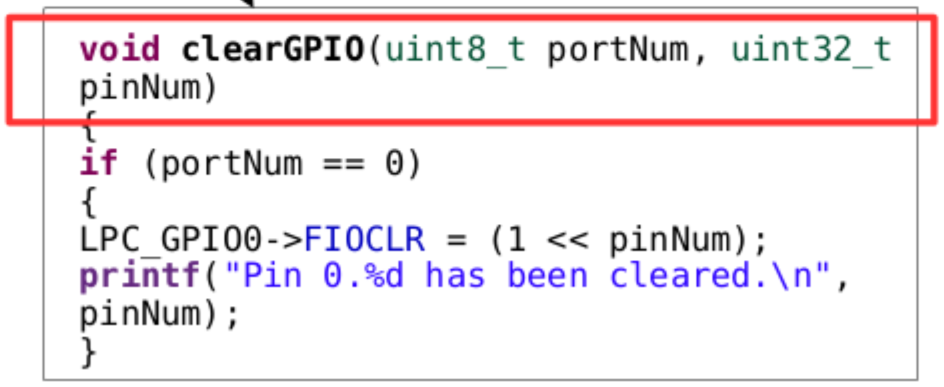
$\vec{P}(x,y) = (x,y)$

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

point(s), Vertex, Vectors

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

Example: Clear the pin



```
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

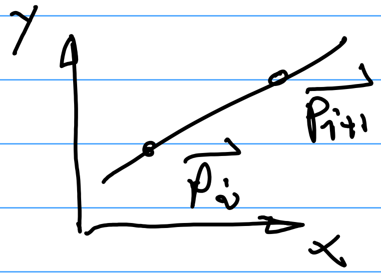


Fig. 2

Now, 2D Vector Graphics

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

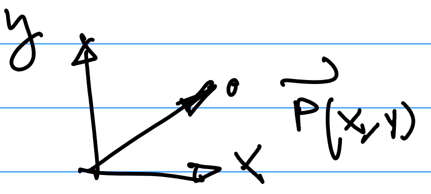


Fig. 1

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

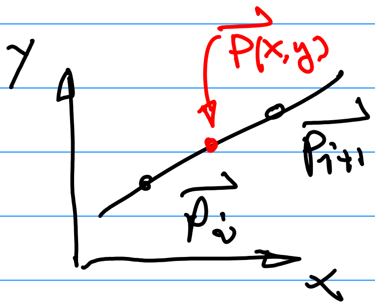


Fig.2

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

Let's define a direction vector

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

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

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

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

$$= (2.5, 1.8) \,_{\prime\prime}$$

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 $\vec{P_i}$, or $\vec{P_{i+1}}$; and directional vector

$$\vec{P}(x,y) = \underset{\text{Starting pt}}{\vec{P_i}} + \underset{\text{scalar}}{\lambda} \underset{\text{Directional vector}}{(\vec{P_{i+1}} - \vec{P_i})} \quad \ldots (2)$$

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

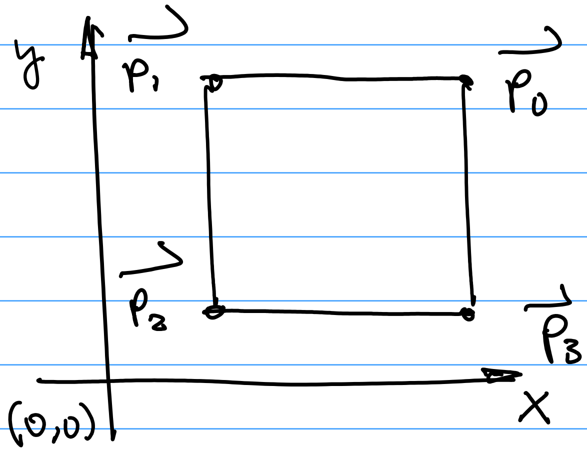
Fig3a

Fig.3b

Screen Saver Design for LPC
2D G.E.
Rotating Squares And Trees.

Example: Design A Rotating Squares

Step1. 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

Fig4


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

Step2. Use

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

Prepare: LCD Soldering ON the Wire-Wrapping Board, Import Single line Drawing project.

Sept. 27 (Mon)

Homework, 2pts. Due 1week from Today

Topics: 2D Screen Saver Design

Requirements:

a. Build LCD Hardware Interface;

b. Import 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

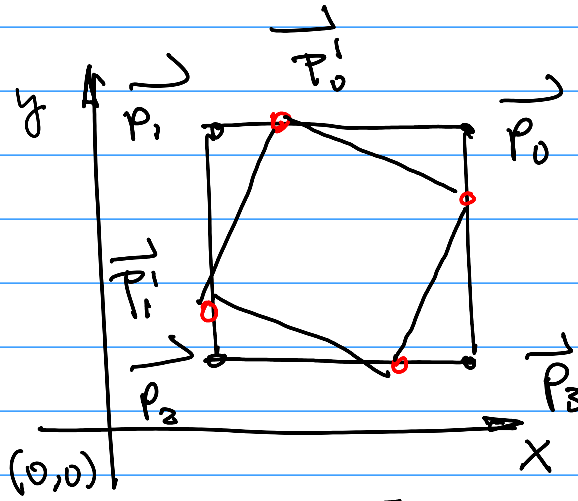


Fig. 1

Let $\lambda = 0.8$, for line1 ($\vec{P_0}$ 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 ($\vec{P_2}$ & $\vec{P_3}$)

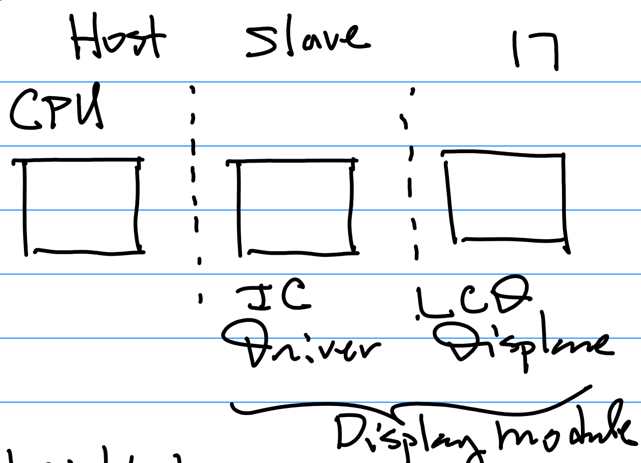
line 4 ($\vec{P_3}$ & $\vec{P_0}$)

In Homework, level ≥ 10.

Coding:

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

Hardware Implementation of LCD Interface.



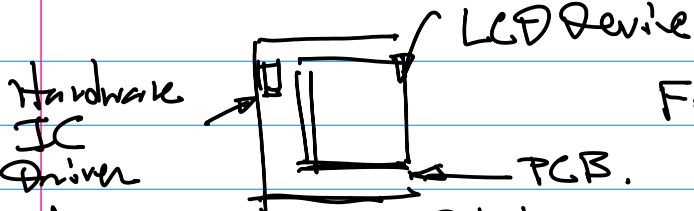


Fig.2

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)

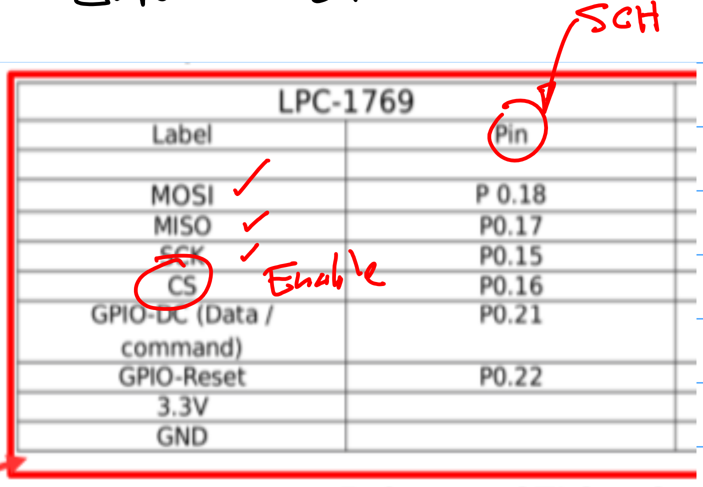
To Establish Interface, SPI (Serial Peripheral Interface)

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 ...



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	

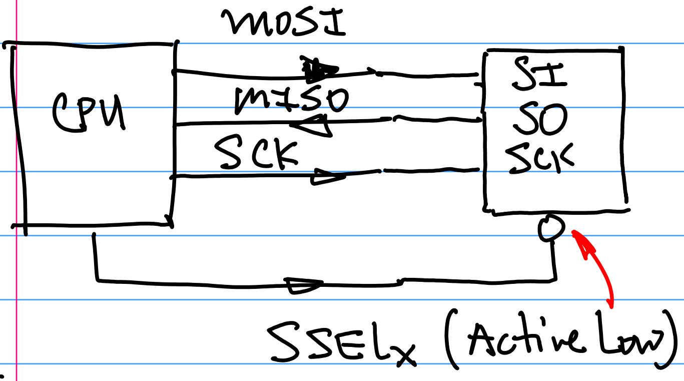
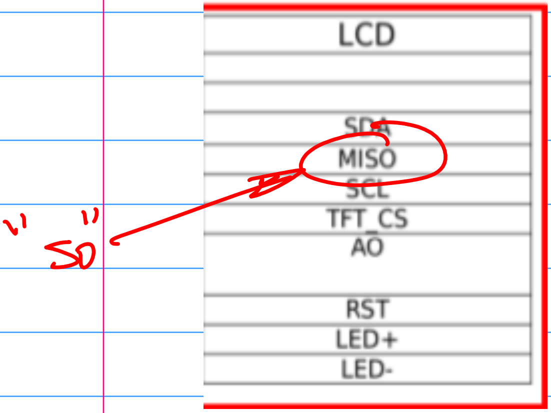


Fig.3.

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/$\overline{D}$ 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 (red, green, blue)
Primitive Colors: r, g, b
2 hex Digits: min. 0 max: 255
2 hex ⇒ 8 bit ⇒ $2^8 = 256$

Match to Eqn (14) & (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

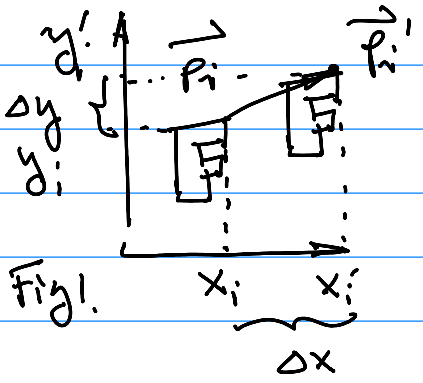


Fig 1.

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

Establish Translation Matrix T.

$\vec{P_i}(x_i, y_i)$ Before ; $\vec{P_i'}(x_i', y_i')$ After

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

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

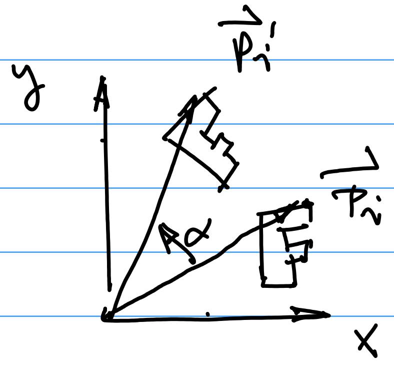


Fig. 2

Note: Counter Clockwise Rotation "Positive" Angle α

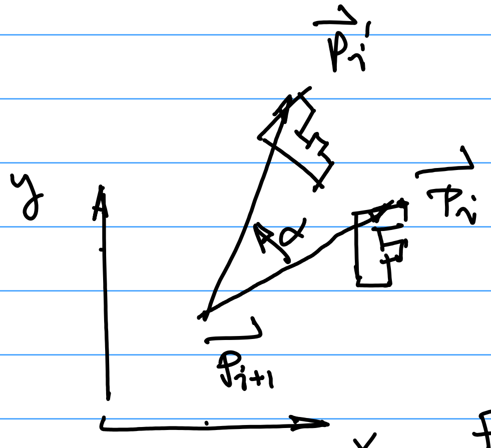


Fig 4.

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

After / Before

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

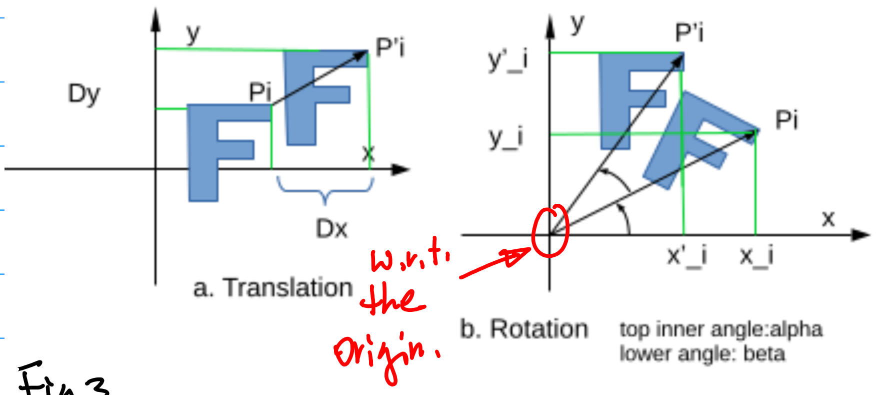


Fig. 3

After / Before

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

From Eqn (4)

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

where

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

Fig.5

Example: Use 2D Transforms to Create Trees shown Above

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

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

$\vec{P_0}(x_0, y_0) = (10,10)$, $\vec{P_1}(x_1, y_1) = (10,20)$

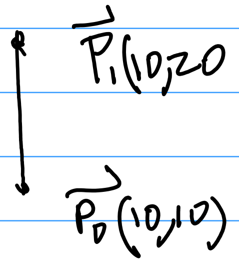


Fig.6

Step 2. Use Vector Eqn to create next Level main/major Branch $\vec{P_1'}(x_1', y_1')$ as in Fig.

$\vec{P_1'}(x_1', y_1') = P_0(x_0, y_0) + \lambda(\vec{P_1}(x_1, y_1) - \vec{P_0}(x_0, y_0))$

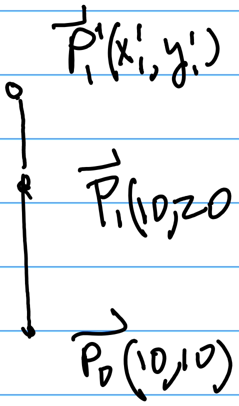
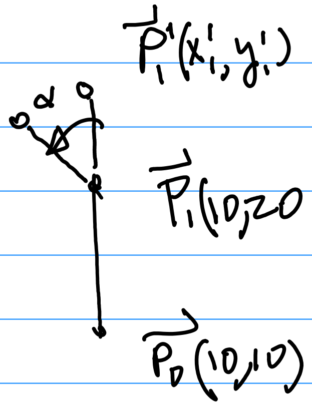


Fig.7

make $\lambda = 0.8$.

Step 3. Rotation of $\vec{P_1'}$ Counter clockwise to Create Left Branch.



$\vec{P_1'}(x_1', y_1')$
α
$\vec{P_1}(10,20)$
$\vec{P_0}(10,10)$