CMPE240
Spring2023

1/

Jan25,23.

Today's Topics:

1° Syllabus, "Greensheet" of the Class.

San José State University
College of Engineering/Computer Engineering Department
CMPE240 Advanced Microcomputer Design

S2023

Note: 1° Attendance; 2° Bring your Laptop Computer; 3° Prototype Board in Class Use/Inspection.

Professor Hua Harry Li
Engineering Building, Rm 267A
Phone: (650) 400-1116 for Text Message Only
Email hua.li@sjsu.edu

Class Time: Mondays and Wednesdays 1:30-2:45 PM

Prototype System { Board A: CPU Module. Board B: For Graphics Display

Office Hours: Mondays and Wednesdays 4:30 – 5:30 PM Zoom
Zoom link: Join Zoom Meeting https://us04web.zoom.us/j/9841607683?pwd=UlA3aEk1TnV4bjNLQk5CQkw0dDk4UT09 Meeting ID: 984 160 7683 Passcode: 121092
Lecture Room: Engineering Building Room 303
Lab facility: Engineering Building Room 268

4° Board A: CPU NXP LPC11C24 to Replace LPC1769 (Near the End of its Life, By 2024)

5° Good Throughout the entire School Session. But it will expire By the Last Day of the Class. (May 15th).

6° Lab Access Code/Policy for the Lab Usage.

Prerequisites

CmpE 180D for non CMPE or non EE undergraduate major. Students who do not provide documentation of satisfied the class prerequisite requirements by the second class meeting will be dropped from the class.

Faculty Web Page and MYSJSU Messaging (Optional)

Copies of the course materials such as the syllabus, major assignment handouts, etc. can be found from github https://github.com/hualili/CMPE240-Adv-Microprocessors/tree/master/2018F and on SJSU CANVAS.

Note: Homeworks/Projects Announcement will be Posted on CANVAS. Submission on CANVAS only.

CMPE240
Spring23

2/

Course Description

Architecture of a computing system including system bus, memory subsystems and peripherals. Uni-directional and bidirectional bus architectures, SRAM and FLASH memories and their interfaces with the system bus. Design of Graphics Processing Engines, interrupt controller, transmitte timers, display adapter, and other system peripherals and bus interfaces.

Required Texts/Readings

Note: Find the Datasheet on the Class github. CMPE244

Textbook

- NXP LPC17xx datasheets;
- LPC1768/1769 CPU Module schematics;
- Dave Jaggar, ARM Architectural Reference Manual, Prentice Hall, ISBN 0-13-736299-4;
- Reference: ARM11 data sheets and on-line web materials on line https://github.com/hualili/ , or at the SJSU CANVAS provided copyright permitted;
- (Optional) Nvidia Jetson NANO datasheet and user menu (online from Nvidia developer website);
- (Optional) RISC-V tutorial (the link to be given in the lecture) and FPGA verilog implementation guide (the link to be given in the lecture).

Other Readings

- The reference material for ARM CPU hardware features, application notes, class handouts and lab assignments and reports, please see Professor Li's lecture notes, PPT, sample C code etc on line https://github.com/hualili/CMPE240-Adv-Microprocessors ;
- Professor Li's book materials, ARM Microprocessor Systems (in preparation for publication) https://github.com/hualili/CMPE240-Adv-Microprocessors

Other equipment / material requirements

32Bit RISC Prototype/Development Board.

Ref:

1° CPU Datasheets

2021F-107-lpc-cpu-UM... Add files vi

2° "SCH" Design.

2021F-107b-sch-#LPC... Add file

3° Lecture Notes

2022F-101-notes-cmpe240-2022-11-30.pdf

CMPE240
Spring23

deadlines and penalties for adding and dropping classes

Homework/Projects

Assignments and Grading Policy

Laboratory	30%
Midterm Examination	30%
Final	40%

0 to 59 F
60 to 69 D
70 to 79 C
80 to 89 B
90 to 100 A

Option 1. Target CPU Module Board
NXP LPC 11C24 ARM CPU Module (recommended a
this course), NXP LPC1769 ARM CPU Mc

Jan29 (Monday).

1. Check the CANVAS for Homework. Honesty Pledge

2. Target platform.

Background: x86. MIPS. ARM.
CISC RISC
RISC-V

NXP LPC Family.

LPC1769 — End of Life By 2024.

LPC1768 — mbed

LPC11C24

Option1: Jetson NANO 2gb.

Option2: RISC-V FPGA Board

BoM (Bill on Material) on github.

Anchor Elektronics.

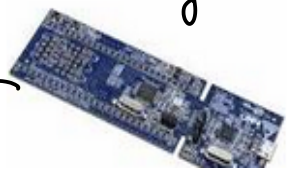
2021F-107-lpc-cpu-UM...

2021F-107b-sch-#LPC...

Example: SCH for LPC1768

Note1. CPU module, see Fig.1, its Two Connectors match the Pins in the Sch. Here

Fig.1



Note2: To make sure match the pins in the Sch to its physical Connector

Note3: Naming Convention — Enumeration Starts with Index 1 for the first pin.

J2 or J6

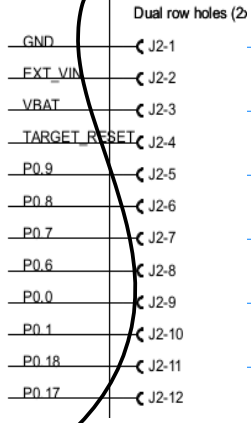
Note4: Pin Connectivity Information should be tied to its physical Device. Fig.1 And to its CPU Datasheet. ----→

→ Eventually to Software IDE.

Note5: J2-1, J2-2, ... ; And Chip pin Name & Number Such as P0.9, P0.8, ...

For LPC1768 is No Need

mbed	LPCXpresso
GND	GND
VIN (4.5-14V)	VIN (4.5-5.5V)
VB (battery supply)	VB (battery supply)
nR (reset)	RESET_N
SPI1-MOSI	P0.9 MOSI1
SPI1-MISO	P0.8 MISO1
SPI1-SCK	P0.7 SCK1
GPIO	P0.6 SSEL1
UART1-TX / I2C1-SDA	P0.0 TXD3/SDA1
UART1-RX / I2C1-SCL	P0.1 RXD3/SCL1
SPI2-MOSI	P0.18 MOSI0
SPI2-MISO	P0.17 MISO0



Functional Description of Each Pin, such as MOSI ... etc.

(Master Out Slave Input)

CMPE 240

Spring 23

Homework: Due A week from today.

1. Download NXP MCU Xpresso,
2. Install MCU Xpresso;
3. Start MCU Xpresso, then Screen Capture of your MCU Xpresso Start Page, make sure it has your Personal identifier on it.

Feb 1st (Wed)

Note: 1° Attendance Sheet

2° Ref. from the Class github.

2023S-102-MCUXpresso_IDE_Installation_...

3° LPC module to be finalized today by the Class. Purchasing CPU module by the end of the day today.

LPC11C24 / LPC1769

CPU module Board B — for Graphics Engine Design Emulation. pre-fab.

LPC1769 End of Life

Option1 Purchase Out-of-Stock. ~16 weeks.

Option2 Removal of All parts. Except the CPU module.

Example: Continuation of the SCH.

1. Pg. 9 → "P" Port, General Purpose port → ARM multiplexing. Each pin can have more than one functions. → Init & Config. → more than one port.

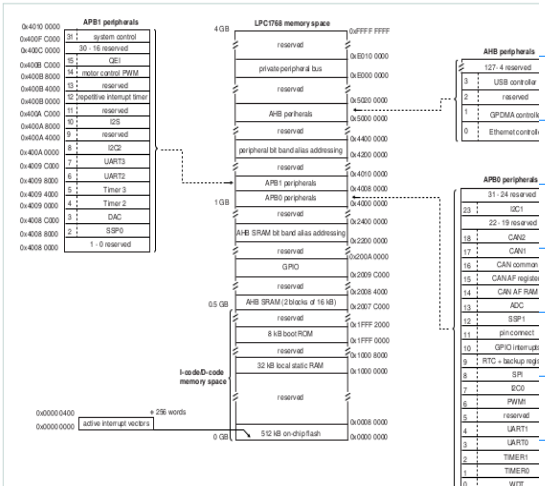
2. Connection to CPU Datasheet.

UM10360

LPC176x/5x User manual

Rev. 3.1 — 2 April 2014 User manual

PP.14 Memory map.



P12.

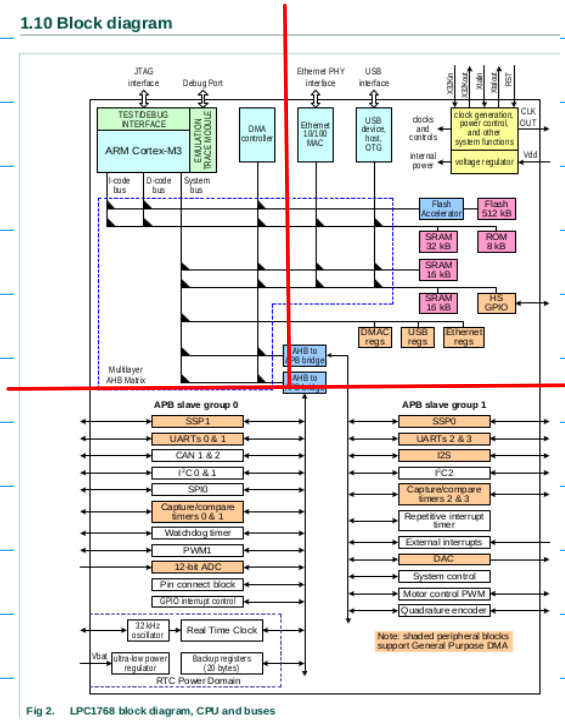
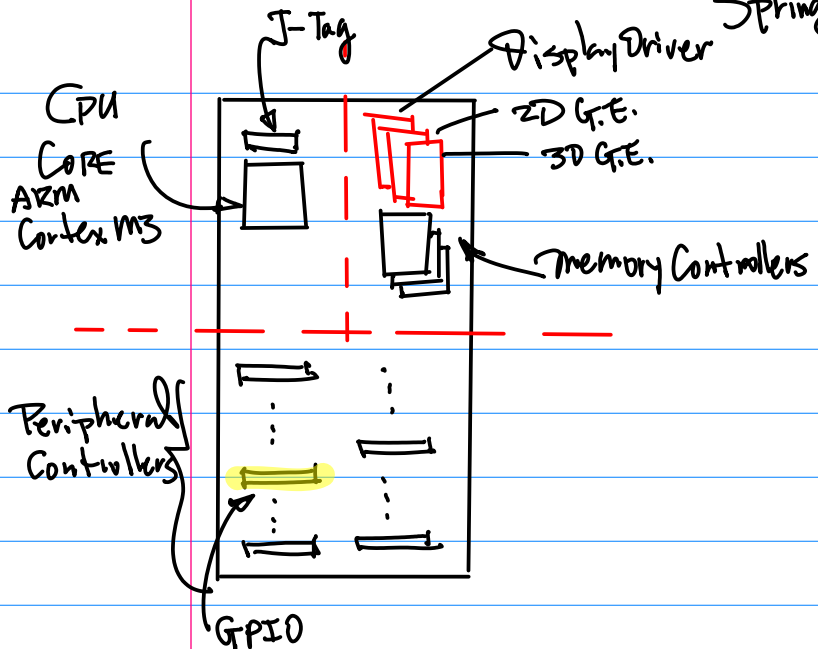


Fig 2. LPC1768 block diagram, CPU and buses



Now, find the pin(s) of the Port in the SCH.

Feb 6 (Monday).

Note: 1° CANVAS Submission of the prerequisite proof By Wednesday.

2°. Screen Capture of MCU Xpresso Installation. (Be Sure to have your Personal Identifier on the Screen).

3° Bring Your LPC CPU module to the class.

Example: Continuation of CPU Architecture

Ref. CPU Datasheet pp.13, pp.12.

Discussion On the CPU Block Diagram.

1° Display Driver.

2° 2D Graphics Processing Engine

3° 3D Graphics Processing Engine. J.T.U.

4° Video Codec. { Encoder / Decoder } MPEG4 / H.264 / 265 / 26X

Pixel Graphics.

Vector Graphics. (Display Drive, 2D G.E, 3D G.E.)

5° General Discussion of the CPU Architecture

(1) 32Bit RISC Architecture.

ALU. Bit width. 32bit

Register File { Special Purpose Registers. / General Purpose Registers. } 32bits

Any meaningful A/L Ops.

Init & Config.

32bit

Bus Systems { Addr ~ 32bit / Uni-Directional; Data ~ 32bit / Bi-directional; ~~Cont. Bus~~ }

(2) memory map. $2^{32} = 2^{10} \cdot 2^{10} \cdot 2^{10} \cdot 2^{2}$

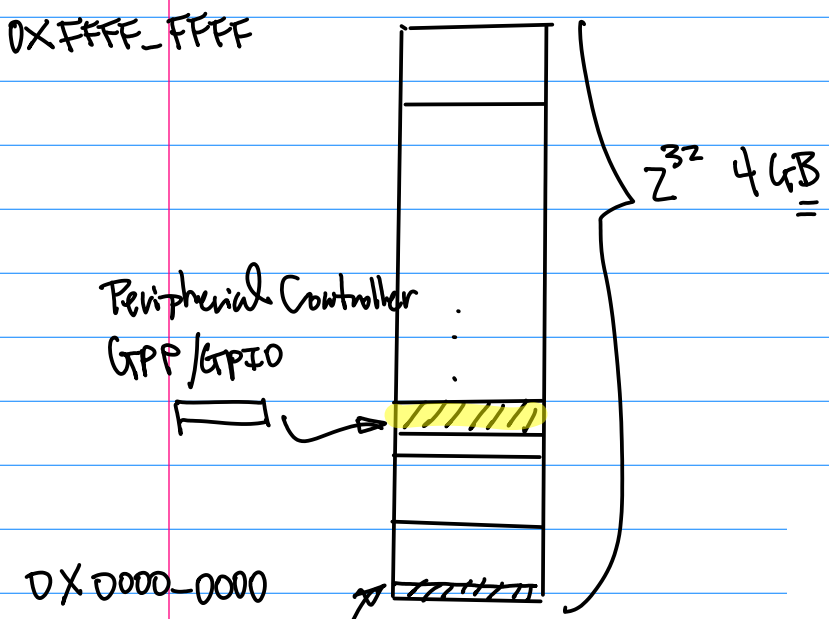
1K

1 Meg

1 Gig

4 G.B. Byte

Byte Addressable machine. ~ minimum mem. Cell with An unique addr. is a Single Byte.



a. Power-up Address: ~ when the CPU is powered up, it will go to this memory location to fetch the 1st Executable instruction

b. 8 memory Banks. Equal size of memory Blocks.

$$2^{32}/8 = 2^{32}/2^3 = 2^{29} = 2^9 \cdot 2^{20}$$

512 | Meg.

1st BANK is enumerated as BANK0, ..., the Last BANK is Bank7.

c. We use 3 Bits from the Address Bus to define the starting Addr. of Each Bank. " Little Endian "

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

$a_{31} a_{30} a_{29} : a_{28} \cdots a_1 a_0$

$a_{31} a_{30} a_{29} : a_{28} \cdots$

0 0 0 : 0 BANK0 0X0...
0 0 1 : 0 BANK1 0X2000_0000
0 1 0 : 0 BANK2 0X4000_0000
1 1 1 : 0

d.

UM10360
Chapter 2: LPC176x/5x Memory map
Rev. 3.1 — 2 April 2014 User manu

2.1 Memory map and peripheral addressing

The ARM Cortex-M3 processor has a single 4 GB address space. The following table shows how this space is used on the LPC176x/5x.

Table 3. LPC176x/5x memory usage and details

Address range	General Use	Address range details and description
0x0000 0000 to	On-chip non-volatile	0x0000 0000 - 0x0007 FFFF For devices with 512 kB of flash memory

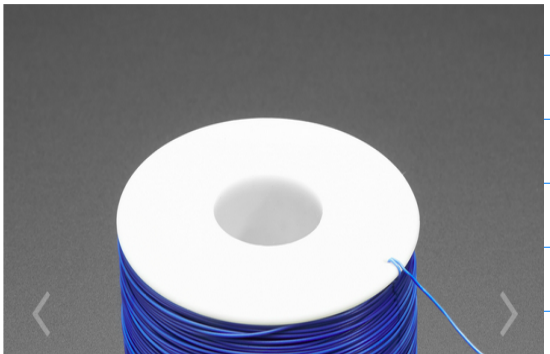
Address for GPIO Controller. 0X2...
0X4...

Feb 8. (Wed).

Note: Inspection of CPU module.
1° LPC1769 or LPC11C24;
2° Layout Design. CPU module Location Standoffs. Wire#

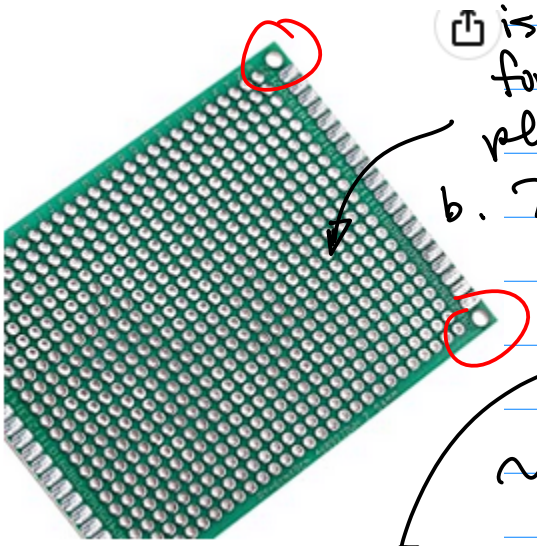
Prototyping wire
30AWG Blue 26~32 AWG

Product ID: 1446 $7.50



CMPE240
Spring 2023

Prototype Board, Note: a. Through-Hole is uniformly formed. with plate coating



b. Mounting holes at corners.

c. Standoffs.

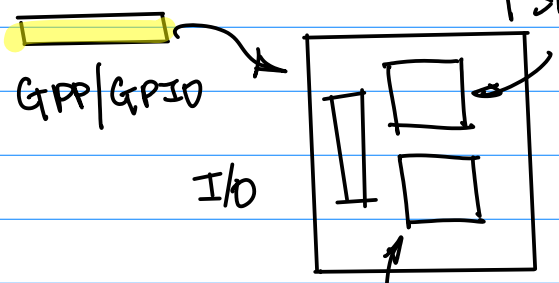
~ $5 - $15

Note: Please Bring 1° Your LPC CPU Module; 2° USB Cable; 3° Laptop with MCU Expresso for Inspection.

Example: Continuation of CPU Architecture Discussion

Note! 1° A GPP/GPIO Controller is mapped to the memory map. Read CPU Datasheet to find its Location;

2° Special Purpose Registers.



Special Purpose Registers.

Task 1: Init & Config.

Naming Convention: RISC (uniformity, Regularity, Orthogonality)

Root (3 Letters)

{ CON

Prefix 3 Letters + Root 3 Letters

GPI

↓

GPP

↓

GP0, GP1, GPx

GPPCON

Software to Drive the Peripheral Controller

↓

Compiler, gcc. Open Source → Porting Target CPU

CmpE240

Spring2023

2nd Special Purpose Register

GPx + DAT

Root.

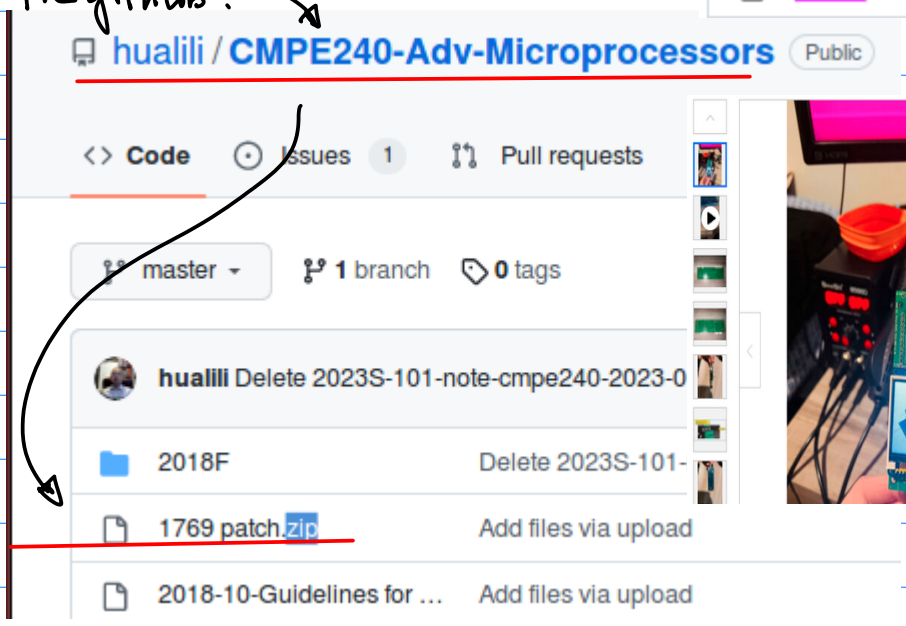
GPIODAT.

Feb13 (Monday).

Example: Start-up MCU Xpresso.

1. Install MCU Xpresso. Start the MCU. → Project Browser (Top Left Panel) → A pannel Beneath it, "Import Project", Double Click / Select to open it, then follow the steps to import 1769 patch.

2. the LPC1769 patch is posted on the github.



3. Connect your LPC GPU module (LPC1769) Browse the imported 1769 projects.

a/

find a project with a key word "Blinky" Then, use "Debug" option to Build it, And to observe the LED on the GPU module flashing.

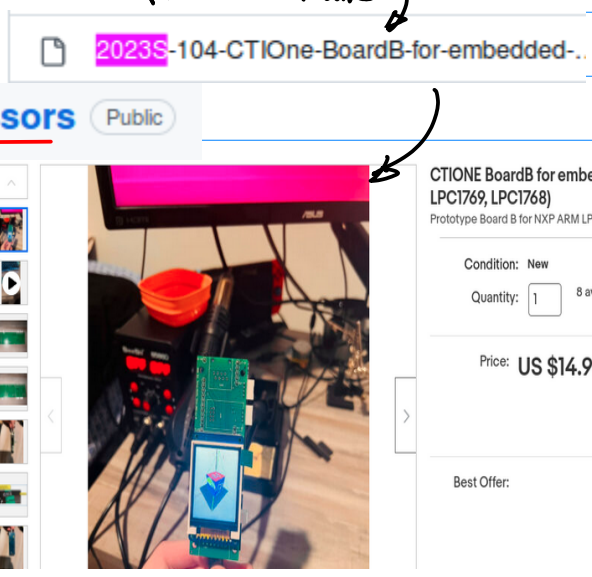
4. Then Browse the LPC1769 Patch, Locate 17xx.h which realizes the task of Porting the target platform (1769) to the IDE Software.

Resources: The posted zip files. (projects).

LPC1769 patch → gpio.zip (IIC24) → drawline.zip (IIC24)

2D Graphics → 2D+3D projects for IIC24.

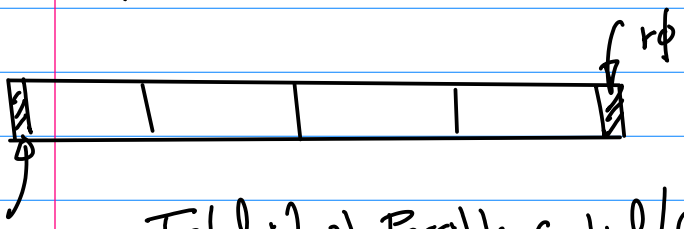
Note: for IIC24 Board)



https://www.digikey.com/en/products/detail/adafruit-industries-llc/358/5801368

Image shown is a representation only. Exact specifications should be obtained from the product data sheet.

Example: GPxCON 32 bit.



Total No. of Possible Control/Config

$OPS = 2^{32}$

From CPU Datasheets, Inspection:

from github ~ Cmpe244

Notation: Vector Notation

GPACON[3:0]

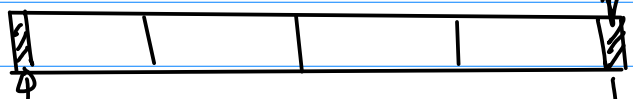
2021F-105-#0-cpu-arm...

GPACONSLP	0x7F00800C	R/W	Port A Sleep mode Co
GPAPUDSLP	0x7F008010	R/W	Port A Sleep mode Pu

GPACON	Bit	Description	
GPA0	[3:0]	0000 = Input 0010 = UART RXD[0] 0100 = Reserved 0110 = Reserved	0001 = Outpu 0011 = Reser 0101 = Reser 0111 = Exter
GPA1	[7:4]	0000 = Input 0010 = UART TXD[0] 0100 = Reserved	0001 = Outpu 0011 = Reser 0101 = Reser

The 1st Pin of the GPIO port.

GPADAT For I/O operation



To make GPADAT[0] as an output we will have to make GPACON[3:0] = 0x1

GPACON	Bit	Description	Initial State
GPA0	[3:0]	0000 = Input 0001 = Output 0010 = UART RXD[0] 0011 = Reserved 0100 = Reserved 0101 = Reserved 0110 = Reserved 0111 = External Interrupt Group 1 [0]	0000
GPA1	[7:4]	0000 = Input 0001 = Output	0000

From the SCH, we can find the physical pin to realize output function.

Source: 17xx.h

The mapping from CPU Architecture to the C-Code is defined by 17xx.h file with statements like:

LPC_GPIO0->FIODIR 0x.....

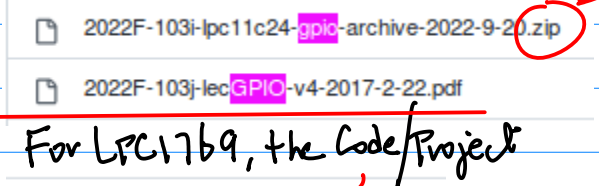
Memory Addr.

CPU Datasheet

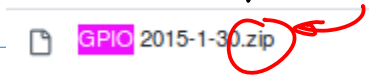
from *.h file

Feb 15 (Wed)

Ref: 1° GPIO Code for LPC11C24



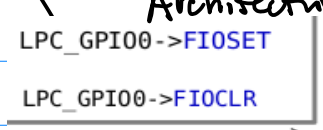
For LPC1769, the Code/Project

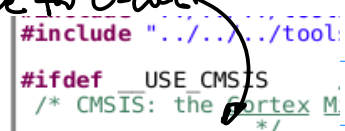


2° GPIO I/O Design.

Special purpose Register

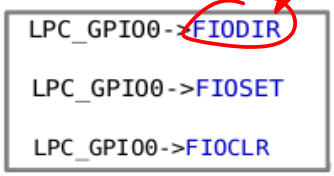
3°. Mapping/Porting Architecture to C-code.

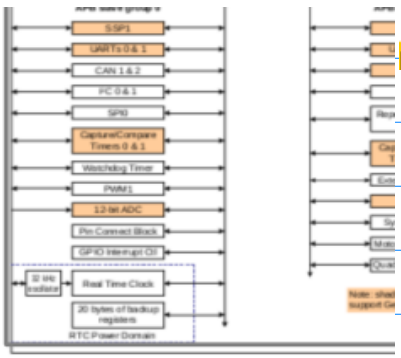




like the following
#define SPR 0x2000_0000
map the CPU architecture to the arm gcc compiler

Note: FIODIR, CPU Datasheet.





CPU

Background:

Exercise: Must Do

① Locate LPC17xx.h

② Find GPIO0 part of the Code.

③ Locate LPC_GPIO0->FIODIR

Find its Address.

2° Naming Convention.

Product family → Peripheral controller → Special Purpose Register

(4) Use CPU Datasheet to
Verify its Address in LPC17xx.h.

Continuation:
CPU Datasheet Table 102.
Discussion on Bit Settings.

Bit	Symbol	Value	Description
31:0	FIO0DIR FIO1DIR FIO2DIR FIO3DIR FIO4DIR		Fast GPIO Direction PORTx control bits. Bit 0 in FIOxDIR controls pin Px.0, bit 31 in FIOxDIR controls pin Px.31.
		0	Controlled pin is input.
		1	Controlled pin is output.

a.

Bit 3 for pin 3 Bit 2 for pin 2

b.


```
LPC_GPIO0->FIODIR = 0x4;
```

c.

Ref: LCD Display for LPC11C24

2022F-103f-LCD-connectivity-LPC11C24-hl-bj-2022-9-30.pdf

LPC11C24 Connectivity Table to

HL (2022-9-30) corrected this typo by replacing LPC 1769 to LPC11C24

Table 5. Connectivity Table of LPC11C24 and LCD

LPC11C24	Description	LCD
1. J6-28	3VOUT	VCC
2. J6-1	GND	GND
3. J6-8	SSEL0	TFT_CS
4. J6-14	RST	RESET
5. J6-13	D/C	A0
6. J6-5	MOSI0	SDA
7. J6-7	SCK0	SCK
8. J6-28	3VOUT	LED

Brand Walfront

TFT LCD Display:
Screen:https://www.amazon.com/gp/product/B07BFV69DZ/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1

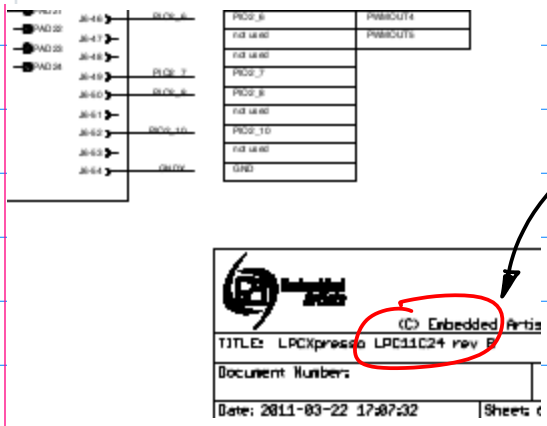
1.8 inch SPI TFT LCD Display Module for ST7735 128x160 51/AVR/STM32/ARM 8/16 bit

Visit the Walfront Store
20 ratings
Amazon's Choice in LCD Graphic Displays by Walfront

-10% $15.55
Price: $17.66
prime One-Day
& FREE Returns
Includes $1.71 Prime savings

Ref: CPU Datasheet for LLC24

1. 2022F-103g-lpc11c24-cpu-datasheet-UM10398.pdf Add files via upload
2. 2022F-103h-SCH-LPCXpressoLPC11C24revB.pdf Add files via upload
3. 2022F-103i-lpc11c24-gpio-archive-2022-9-20.zip Add files via upload



Feb 20 (Monday).

Note: 1° Homework Coming on CANVAS;

Today's Topics:
1° Summary on GPIO
2° Hardware for Graphics 2D Engine

Example: Ref CPU Datasheet Table 102. for FIODIR S.F.R.

Note: 1° Addr. for the S.F.R. Need to Be Cross-refer'd in MCUXpresso Code.

2° FIODIR Multiple. Registers

access to a group of bits in a single GPIO port independently from other bits in the same port.

Table 102. GPIO register map (local bus accessible registers - enhanced GPIO features)

Generic Name	Description	Access	Reset value[1]	PORTn Register Name & Address
FIODIR	Fast GPIO Port Direction control register. This register individually controls the direction of each port pin.	R/W	0	FIO0DIR - 0x2009 C000 FIO1DIR - 0x2009 C020 FIO2DIR - 0x2009 C040 FIO3DIR - 0x2009 C060 FIO4DIR - 0x2009 C080
FIOMASK	Fast Mask register for port. Writes, sets, clears, and reads to port (done via writes to FIOPIN, FIOSET, and FIOCLR, and reads of FIOPIN) alter or return only the bits enabled by zeros in this register.	R/W	0	FIO0MASK - 0x2009 C010 FIO1MASK - 0x2009 C030 FIO2MASK - 0x2009 C050 FIO3MASK - 0x2009 C070 FIO4MASK - 0x2009 C090

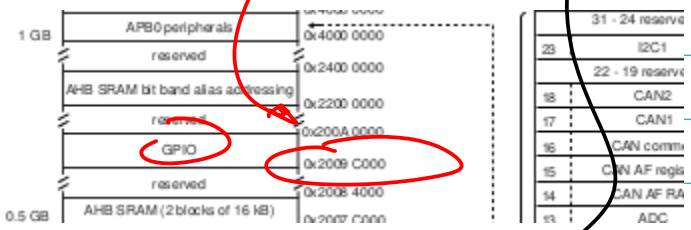
Connection To Coding: C/C++

FIO0DIR → LPC_GPIO0 → FIO0DIR

0X2009_C003 — r0

0X2009_C000 : — r3 | 0X2009_C000

Connection To Mem map



LPC1102 / OR 1769

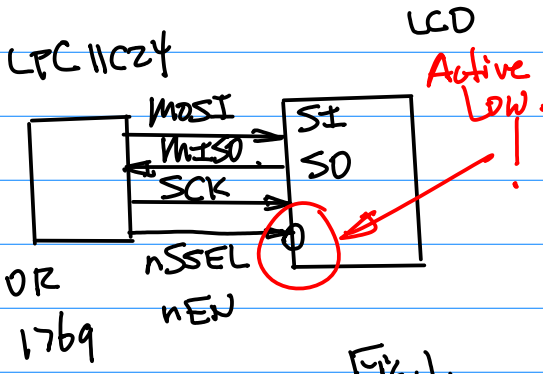


Fig.1.

30. Addr. for Byte Addressable Machine, 4 Addresses.

FIO0DIR[rx':rx], for Example

FIO0DIR[4:3]

40 Bit (Binary) Pattern for the SFR.

Table 105. Fast GPIO port Direction control byte and half-word accessible register description

Generic Register name	Description	Register length (bits) & access	Reset value	PORTn Register Address & Name
FIOxDIR0	Fast GPIO Port x Direction control register 0. Bit 0 in FIOxDIR0 register corresponds to pin Px.0 ... bit 7 to pin Px.7.	8 (byte) R/W	0x00	FIO0DIR0 - 0x2009 C000 FIO1DIR0 - 0x2009 C020 FIO2DIR0 - 0x2009 C040 FIO3DIR0 - 0x2009 C060 FIO4DIR0 - 0x2009 C080
FIOxDIR1	Fast GPIO Port x Direction	8 (byte)	0x00	FIO0DIR1 - 0x2009 C001

Responsible for P0.0 ~ P0.7 (8 pins).
"1" for Output, "0" for the input.

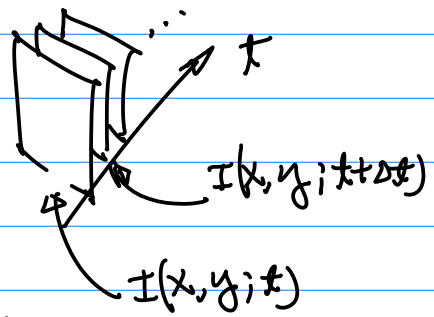
* Consider Hardware Design for LCD Display.

Background: SPI. Enable LCD Display Device.

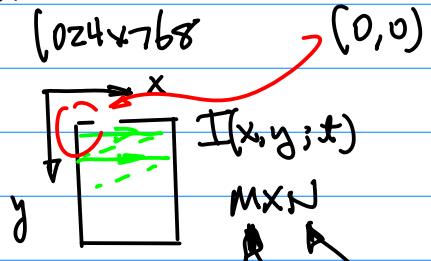
Serial Peripheral Interface.

Bit Rate ~ 100Mbps.

Note: Bit Rate Discussion.



Graphics Display Resolution: 1024x768



Scans:
Top Left Corner (0,0),
From Left to Right
Top to Bottom.
No. of pixels/Row
No. of Rows Per frame.

30 Frames Per Second (FPS)
24 Bits of color. (Color Depth)

$1024 \times 768 \times 30 \times 24$ bit/sec.

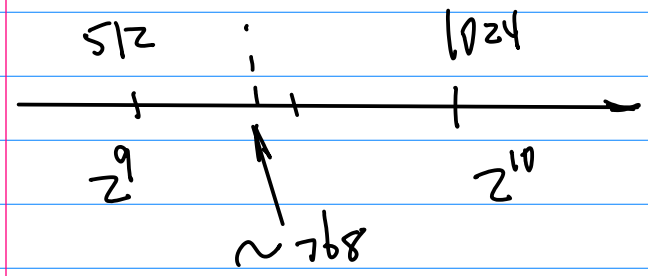
$2^{10} \quad 2^{9} \quad 2^{5} \quad 2^{5} = 1K \cdot 1K \cdot 2^{9}$

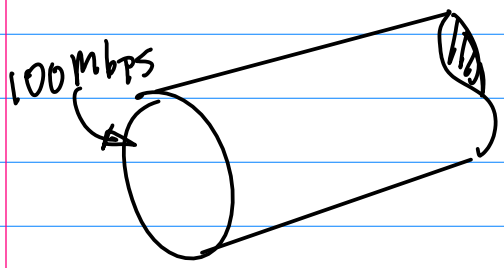
$2^{10} = 1\,Meg \times 512$

$1K \quad 1K = 512\,Mbps$ //

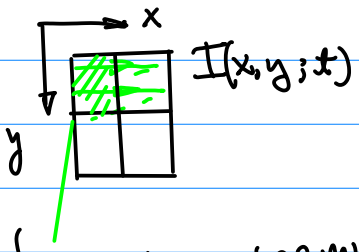
Now, Let's go to Sch of the Target GH.

2022F-103h-SCH-LPCXpressoLPC11C24revB.pdf





	LPCXpresso			Dual row holes (2x27), 100 mil spaci
	GND		GNDX	J6-1
	VIN (4.5-5.5V)		EXT_POWX	J6-2
	not used			J6-3
	PIO0_0	RESET	PIO0_0	J6-4
	PIO0_9	MOSI0/SWO	PIO0_9	J6-5
	PIO0_8	MISO0	PIO0_8	J6-6
	PIO2_11	SCK0	PIO2_11	J6-7
	PIO0_2	SSEL0	PIO0_2	J6-8
A	PIO1_7	TXD	PIO1_7	J6-9
L	PIO1_6	RXD	PIO1_6	J6-10
	PIO0_7		PIO0_7	J6-11
	PIO2_0		PIO2_0	J6-12
X	PIO2_1		PIO2_1	J6-13
	PIO2_2		PIO2_2	J6-14
	PIO0_11	AD0	PIO0_11	



Build Connectivity Table

CPU pin PinNo./Name	Connector Info	Note
PIO0-9/MOSI0	J6-5/MOSI0	

$1/4 \times 512 > 100\,Mbps$

1/4 of the Resolution $1024 \rightarrow 1024/2 = 512$

$768 \rightarrow 512/2 = 256$

SPI I/F: "3+1"

- MISO Mast Input / Slave Output
- MOSI master Out / Slave Input
- SCK (Clock/Output)

} 3

- SSEL (nEN: Active Low, Output from the master)

Feb 22 (Wed).

1° In-Class Homework 1pt.

Next Monday.

Homework: SPR & GPP.

In-Class.

~15 min.

Submission on-Line to CANVAS.

You will Need Laptop Computer.

CMPE240
Spring2023

Example: Continuation of Build
CPU to LCD Display Interface

Software Design: { Theoretical Background: CPU Architecture; Coding.

CPU pin PinNo./Name	Connector Info	Note LCD Device
P1.0-9/MOSI0	J6-5/MOSI0	SI

Note: for LCD pin Connectivity Information for LPC11C24

2022F-103f-LCD-connectivity-LPC11C24-hl-bj-2022-9-30.pdf

HL (2022-9-30) corrected this typo by replacing LPC 1769 to LPC11C24

Table 5. Connectivity Table of LPC11C24 and LCD

LPC11C24	Description	LCD
1. J6-28	3VOUT	VCC
2. J6-1	GND	GND
3. J6-8	SSEL0	TFT_CS
4. J6-14	RST	RESET
5. J6-13	D/C	A0
6. J6-5	MOSI0	SDA
7. J6-7	SCK0	SCK
8. J6-28	3VOUT	LED

Note: 1° PWR pins (Vcc=3.3V, GND)

2° Chip select

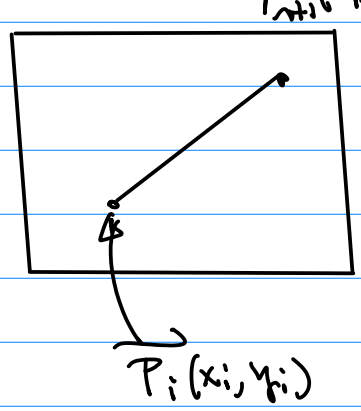
SI

3° Control Signal from the Master, CPU, D: Data; C: Command
Binary Value; Serial Input; 5° for Backlight

4° ~~SDA~~ is S.I.

Note: 8pin v.s. 10 pins. if for Wirewrapping Build, then you can use any one of the two. Check the Latest pin Counts for 11C24 Prefab'd Board.

Coding: MCU Xpresso Installation.
↓
SDK for the target CPU Configuration
↓
Import the existing code
↓ LPC1769 patch.
↓ LPC11C24 gpio.zip
↓
Test Drive, "Dry Run"
↓
Ready for "DrawLine" project (Code for LPC1769, and code for LPC11C24.)



CPU Datasheet, pp.431

Note: 1° Control Register, SSPφCR0

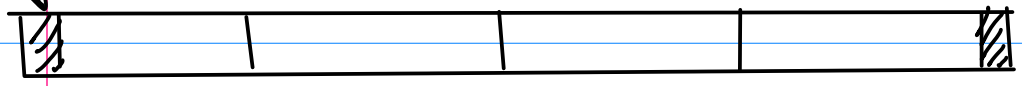
Table 371: SSPn Control Register 0 (SSP0CR0 - address 0x4008 8000, SS 0x4003 0000) bit description

Bit	Symbol	Value	Description
3:0	DSS		Data Size Select. This field controls the number of bi transferred in each frame. Values 0000-0010 are not and should not be used.
		0011	4-bit transfer

2° Addr. for SSPφCR0: 0x4008_8000.
Where is it on the memory map?
How is it reflected in C/C++ code, e.g. LPC17xx.h

3° SSPφCR0[3:0]
Least Significant Bit
Most Significant Bit

r31 "Little Endian" r0



SSPφCR0

SSPφCR0[3:0]: Data Size Select. (DSS).
For 8-bit Communication.

Bit	Symbol	Value	Description
3:0	DSS		Data Size Select. Thi transferred in each fr and should not be us
		0011	4-bit transfer
		0100	5-bit transfer
		0101	6-bit transfer
		0110	7-bit transfer
		0111	8-bit transfer
		1000	9-bit transfer
		1001	10-bit transfer
		1010	11-bit transfer
		1011	12-bit transfer
		1100	13-bit transfer

SSPφCR0[3:0] = 0111 = 0x7

Tech. Spec. (Technical Specification) → Binary Pattern for Init & Config.

For SPI
SSPφCR0[5:4] = 00

$$f_{SPI} = \frac{PCLK}{CPSDVSR * (SCR+1)}$$

(2~254) (0~255) ...(1)

Feb 27 (Mon)

Project 1. Due 2½ weeks.
Design & implement 2D Vector Graphics.
1° Screen Saver.

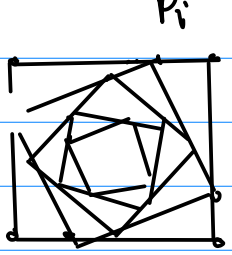


Fig1.

2°

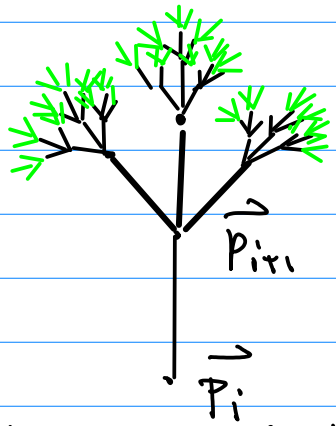


Fig2

Background 2D Vector Graphics

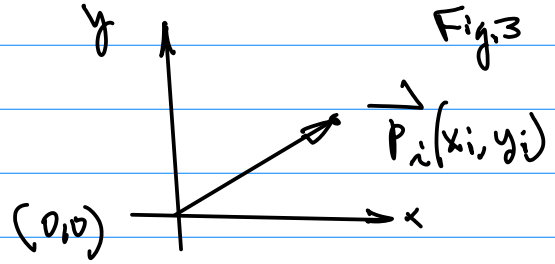


Fig3

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

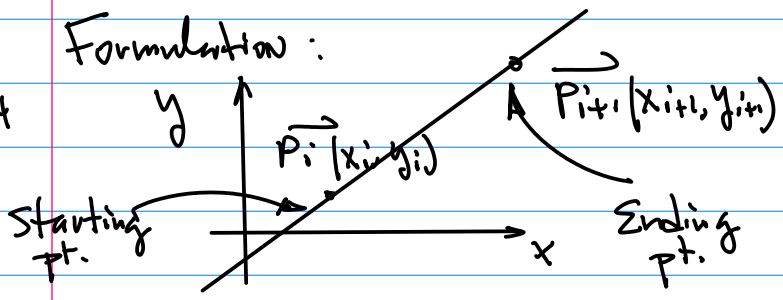
$\vec{P}(x,y) = (x,y)$

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

Visualize the vector $\vec{P_i}(x_i,y_i)$

Formulation:

Fig4



Given $\vec{P_i}(x_i,y_i)$ and $\vec{P_{i+1}}(x_{i+1},y_{i+1})$.

Define a directional vector.

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

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

Therefore,

$$\vec{P}(x,y) = \vec{P_i}(x_i,y_i) + \lambda \vec{d}(x,y) \quad \cdots (2\text{-}a)$$

$$\vec{P}(x,y) = \vec{P_i}(x_i,y_i) + \lambda\left(\vec{P_{i+1}}(x_{i+1},y_{i+1}) - \vec{P_i}(x_i,y_i)\right) \quad \cdots (2\text{-}b)$$

where $\lambda \in (-\infty, +\infty)$

Question1: Find the starting pt By Eqn(2-b)?

$\lambda = 0$

Question2: Find the ending pt By Eqn(2-b)?

$\lambda = 1$

Question: Find All the pts Between $\vec{P_i}$ and $\vec{P_{i+1}}$? $\quad 0 < \lambda < 1$

if $\lambda < 0$, then we have all the pts defined by Eqn(2-b).

if $\lambda > 1$, then we have all the pts beyond the Ending pt.

Marchlist (Wed)

Note: Project1 Due in 2 week 10 pts.

Example: Design Rotating Square

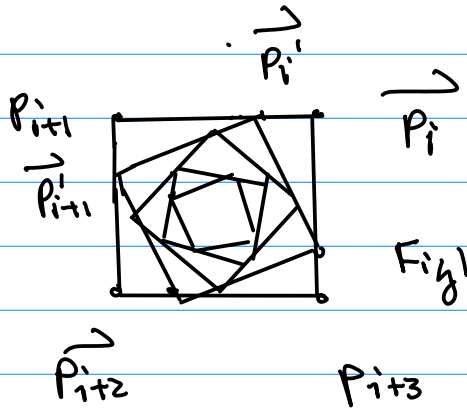


Fig1.

First, To create a set of points,

$$\left\{ \vec{P_i}(x_i,y_i) \mid i = 0,1,2,3 \right\} \quad \cdots (1)$$

Fig1

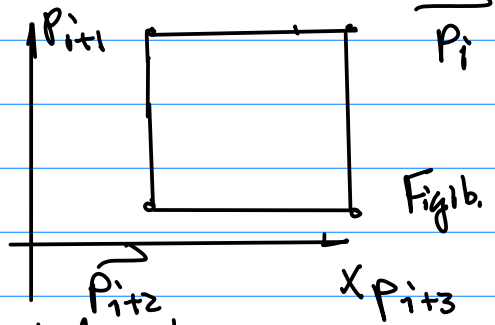


Fig1b.

Virtual Coordinate System

Let's choose $\vec{P_i}\Big|_{i=0} = (30,25)$

$\vec{P_1}(10,25)$, $\vec{P_2}(10,5)$, $\vec{P_3}(30,5)$

Then use DrawLine Sample code to Draw Lines to Connect Each pair of points.

$\vec{P_0},\vec{P_1}$; $\vec{P_1},\vec{P_2}$; $\vec{P_2},\vec{P_3}$; and $\vec{P_3},\vec{P_0}$

Note: in C/C++, use mod( ) Function.

Then, Define the 2nd set of points.

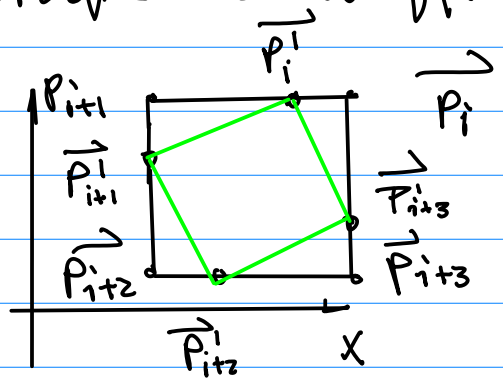


Fig. 2

Define A set of pts as

From Eqn. (2b), we have

$$\vec{P}(x,y) = \vec{P_i}(x_i,y_i) + \lambda\left(\vec{P_{i+1}}(x_{i+1},y_{i+1}) - \vec{P_i}(x_i,y_i)\right) \quad \ldots(2)$$

Substitute the given conditions into the Above equation, we have

$$\vec{P_0^1}(x_0^1,y_0^1) = \vec{P_0^0}(x_0^0,y_0^0) + \lambda\left(\vec{P_1^0}(x_1^0,y_1^0) - \vec{P_0^0}(x_0^0,y_0^0)\right) \quad \ldots(3)$$

Coding it in C/C++

$$\vec{P_0^1}(x_0^1,y_0^1) = \vec{P_0^0}(x_0^0,y_0^0) + \lambda\left(\vec{P_1^0}(x_1^0,y_1^0) - \vec{P_0^0}(x_0^0,y_0^0)\right)$$

$$= (x_0^0,y_0^0) + \lambda(x_1^0 - x_0^0, y_1^0 - y_0^0) \quad \ldots(3b)$$

Hence,

$$\begin{cases} x_0^1 = x_0^0 + \lambda(x_1^0 - x_0^0) & \ldots(4a) \\ y_0^1 = y_0^0 + \lambda(y_1^0 - y_0^0) & \ldots(4b) \end{cases}$$

lamda = 0.2

```
x[0][i] = x[0][0] + lamda * (
        x[1][0] - x[0][0]);   ...(5)
```

Next step, generalize it for level j+1 for all the points i, (i=0,1,2,3)

$$\vec{P_i^{j+1}} = \vec{P_i^j} + \lambda\left(\vec{P_{i+1}^j} - \vec{P_i^j}\right)$$

$$\vec{P_i^{j+1}}(x_i^{j+1},y_i^{j+1}) = \vec{P_i^j}(x_i^j,y_i^j) + \lambda\left(\vec{P_{i+1}^j}(x_{i+1}^j,y_{i+1}^j) - \vec{P_i^j}(x_i^j,y_i^j)\right) \quad \ldots(6)$$

for the coding:

$$\begin{cases} x_i^{j+1} = x_i^j + \lambda(x_{i+1}^j - x_i^j) & \ldots(7a) \\ y_i^{j+1} = y_i^j + \lambda(y_{i+1}^j - y_i^j) & \ldots(7b) \end{cases}$$

Example: Design & Implement Tree Drawing Algorithm

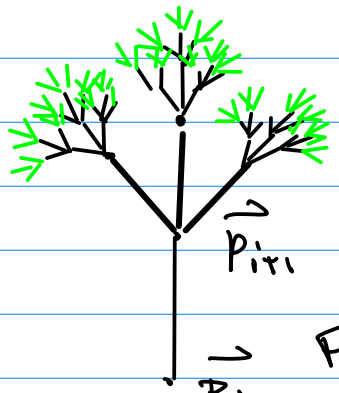


Fig.3a

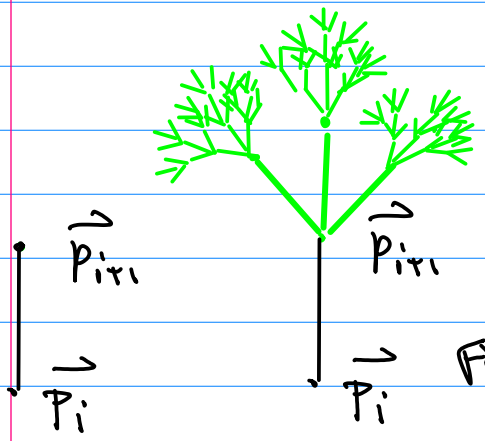
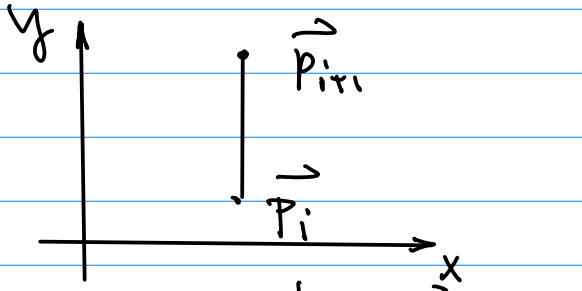


Fig.3b

Define $\vec{P}_i$ and $\vec{P}_{i+1}$ as points for the tree trunk.

$$\vec{P}_i(x_i, y_i)\Big|_{i=0} = \vec{P}_0(x_0, y_0) = (10, 2)$$



$$\vec{P}_{i+1}(x_{i+1}, y_{i+1})\Big|_{i=0} = \vec{P}_1(x_1, y_1) = (10, 12)$$

Fig.3c

Next step "grow" the tree to Level 1

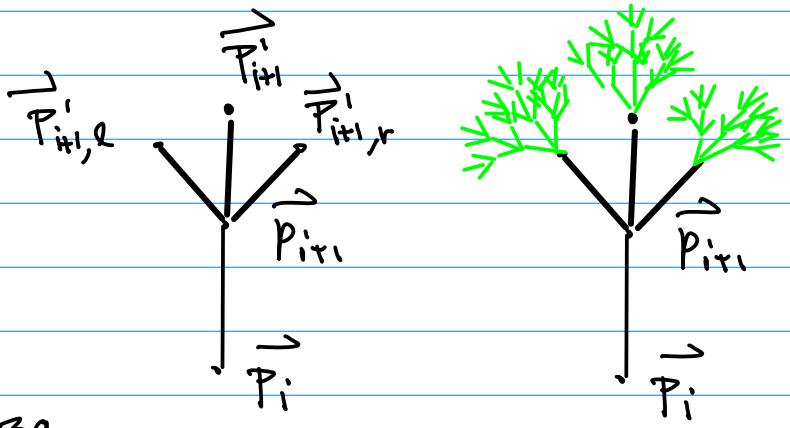


Fig.4.

where $\vec{P}'_{i+1}$ is the point with 20% Reduction of its previous magnitude.

$$\vec{P}'_{i+1} = \vec{P}_i + \lambda(\vec{P}_{i+1} - \vec{P}_i)\Big|_{\lambda=0.8}$$

To find pts to the right & Left, Lets define 2D Transforms.

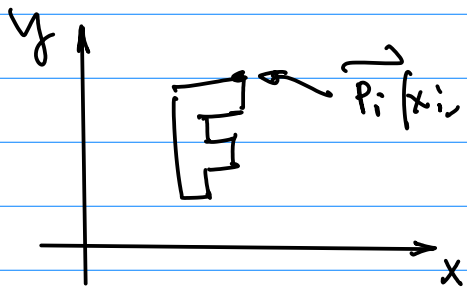


Fig.5

Define A pattern $\left\{\vec{P}_i(x_i, y_i)\Big| i=0,\cdots,M\right\}$

1. Translation.

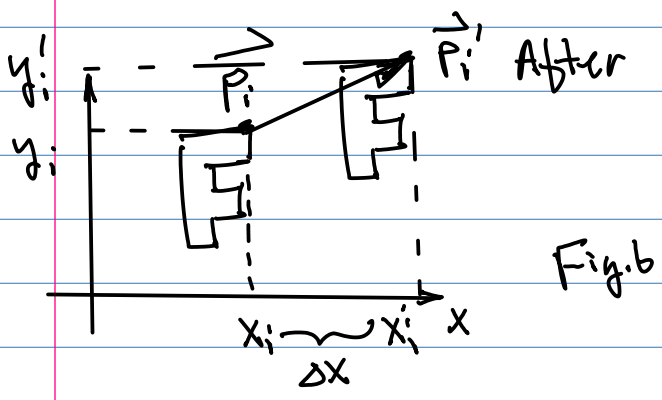


Fig.6

$$\begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \cdots (9)$$

Rotation.

$$\begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \cdots (10)$$

$$\{\vec{P_i}(x_i, y_i)\} \xrightarrow{\Phi} \{\vec{P_i'}(x_i', y_i')\}$$

Φ^{-1}

2. Rotation

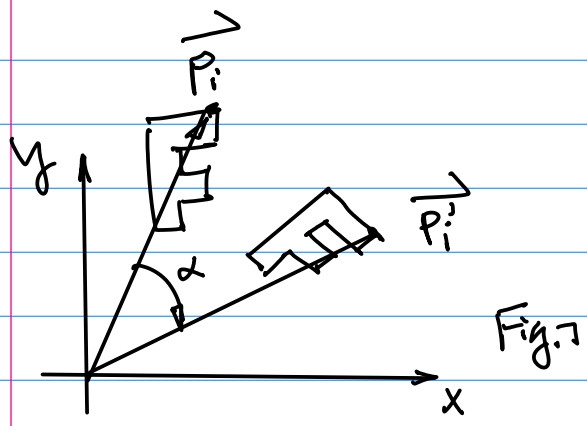


Fig.7

Clockwise Rotation $\alpha < 0$

Counter Clockwise Rotation $\alpha > 0$

Let's consider the translation.

$$(x_i, y_i) \xrightarrow{\Phi=?} (x_i', y_i')$$

Before After

$$x_i' \stackrel{?}{=} x_i + \Delta x \quad \cdots (8a)$$

Where $\Delta x = x_i' - x_i$

Similarly

$$y_i' = y_i + \Delta y \quad \cdots (8b)$$

2018F-112-lec4-2D-vectors-v3-2018-10-1.pdf

March 6 (Monday).

Note: for people who choose Pre-fab Board for the project, (1) you have 2 weeks to work on the project from the Receiving of the Board; (2) Do Submission of the photo of the prof.

Example: Continuation.

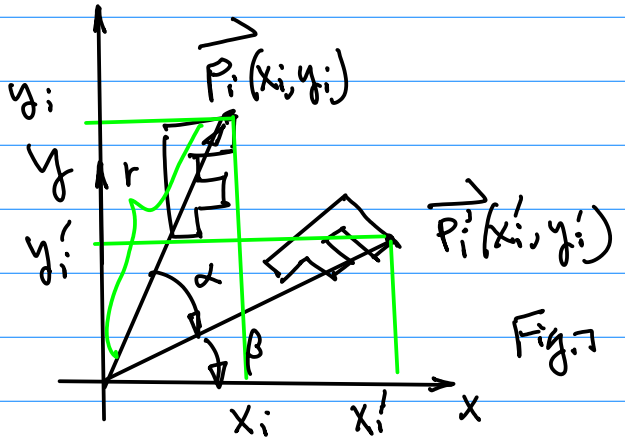


Fig.7

$$\begin{cases} x_i = r \cdot \cos(\alpha+\beta) & \cdots (1a) \\ y_i = r \cdot \sin(\alpha+\beta) & \cdots (1b) \end{cases}$$

$x_i = r\cdot\cos(\alpha+\beta)$

$= r\cos\alpha\cos\beta - r\sin\alpha\sin\beta$

where $r\cos\beta = x_i'$, $r\sin\beta = y_i'$ ... (2).

$$\begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \ldots(4)$$

where $\Delta x = -12$

$\Delta y = -20$. ... (5)

Question: How to Apply the Rotation matrix to find the pt. of the left Branch & the pt of the Right Branch.

$\vec{P}'_{i+1,L}$, $\vec{P}'_{i+1,R}$

Suppose $\vec{P}'_{i+1} = (12, 20)$

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

$R_{3\times3}$

$$\begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \ldots(2)$$

Where $\alpha = -30°$.

Note: The Rotation matrix is only good w.r.t. the reference point at the origin $(0,0)$.

Therefore, Translation of the pattern to the origin by Translation is Need.

The Translation matrix is from Eqn (9), pp.21.

$T_{3\times3}$ (Pre-Processing) ... (3)

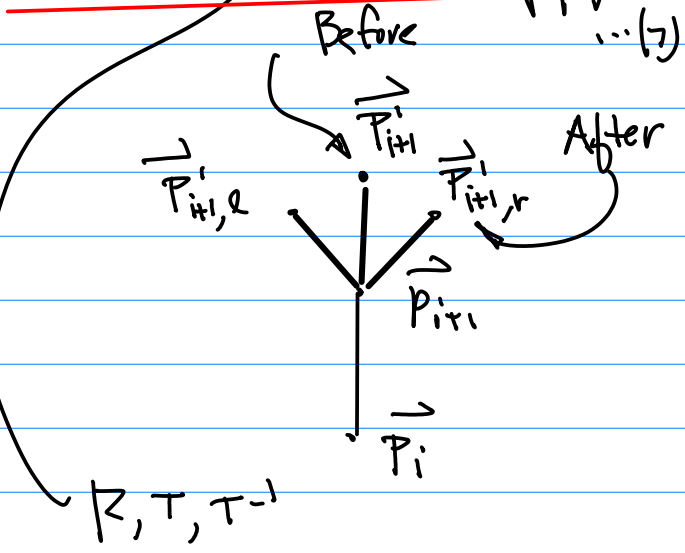
For the prepossing.

Post Processing using the following Math. Formulation

$$T^{-1}_{3\times3} = \begin{pmatrix} 1 & 0 & ? \\ 0 & 1 & ? \\ 0 & 0 & 1 \end{pmatrix} \begin{matrix} \leftarrow -\Delta x \\ \leftarrow -\Delta y \\ \\ \end{matrix} \quad \ldots(6)$$

Now, Let's combine the above steps.

After $\begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} \overset{?}{=} T^{-1} R T \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}$ Before ... (7)



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

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

$$= \begin{pmatrix} \cos\alpha & -\sin\alpha & \cos\alpha\cdot\Delta x - \sin\alpha\cdot\Delta y - \Delta x \\ \sin\alpha & \cos\alpha & \sin\alpha\cdot\Delta x + \cos\alpha\cdot\Delta y - \Delta y \\ 0 & 0 & 1 \end{pmatrix} \quad \cdots (8).$$

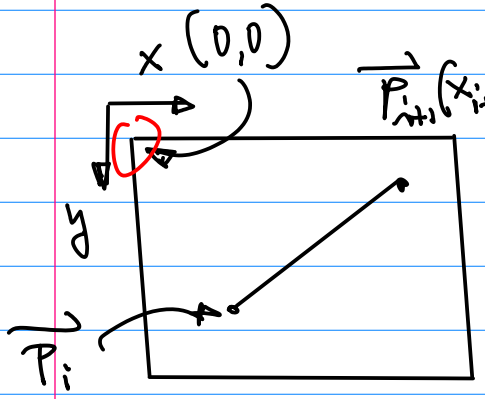
Now,

$$\begin{cases} x_i' = \cos\alpha\cdot x_i - \sin\alpha\cdot y_i + \cos\alpha\cdot\Delta x - \sin\alpha\cdot\Delta y - \Delta x & \cdots (aa) \\ y_i' = \sin\alpha\cdot x_i + \cos\alpha\cdot y_i + \sin\alpha\cdot\Delta x + \cos\alpha\cdot\Delta y - \Delta y & \cdots (ab). \end{cases}$$

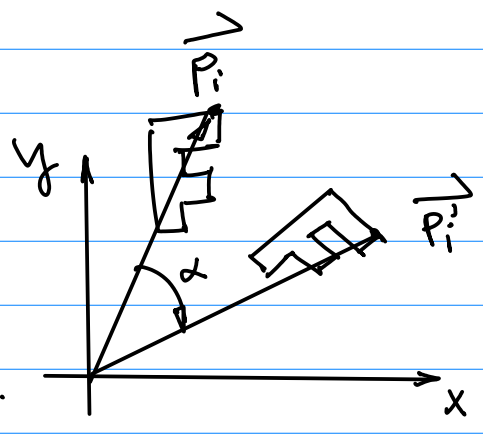
```
X_prim[i] = Cos(alpha) * x[i] - Sin(alpha) * y[i] +
            Cos(alpha) * delta_x - Sin(alpha) * delta_y - delta_x;
```

where $\Delta x = -12$, $\Delta y = -20$, $x_i = 12$, $y_i = 20$

Consider the Hardware of A Display Device.

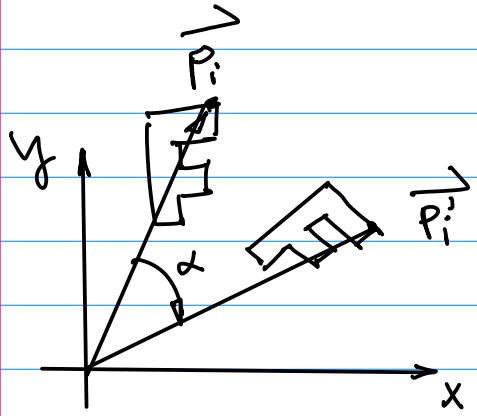


Physical Display ~ Coordinate System.



CMPE240

Spring2023



From pp.23

$x_p = x_v + \frac{M}{2}$ ...(7)

a Direction — Half of the Resolution along x-direction.

b offset

$y_p \stackrel{?}{=} -y_v + \frac{N}{2}$ ...(8)

(half of the No. of Rows)

$x \in (-\infty, +\infty), y \in (-\infty, +\infty)$

↓ Normalization.

$x \in (-1, +1), y \in (-1, +1)$

Virtual Display / Coordinate system

Physical Display (x_p, y_p) ⇄ Virtual Display (x_v, y_v)

After: y_p — Before: y_v

$y_p \stackrel{?}{=} y_v$

$y_p \stackrel{?}{=} -y_v$ Direction

offset

$y_p = -y_v + N/2$

Note: M x N Resolution

No. of Col. (x) — No. of Rows (y)

Ref:

2022F-101-notes-cmpe240-2022-11-30.pdf

March 8 (Wed)

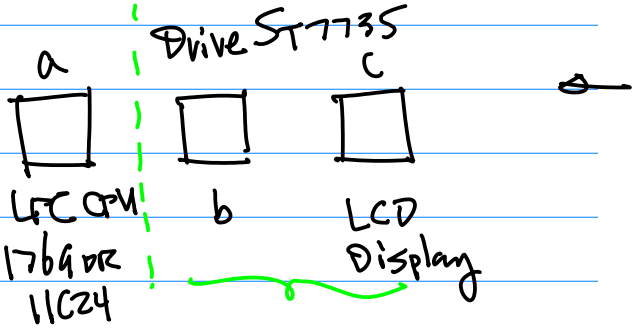
Midterm Exam is scheduled on March 22nd (Wed).

Example: Lecture Notes pp.23

Example: Sample Code Reading.

2021F-111b-handout-drawline-dotC.pdf

Note: 1° Resolution

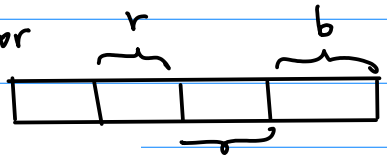



```
#define ST7735_TFTWIDTH 127
#define ST7735_TFTHEIGHT 159

#define ST7735_CASET 0x2A
#define ST7735_RASET 0x2B
#define ST7735_RAMWR 0x2C
#define ST7735_SLPOUT 0x11
#define ST7735_DISPON 0x29
```

CMPE249
Spring2023

25

Note: 24 Bit Color




```
// defining color values
#define LIGHTBLUE 0x00FFE0
#define GREEN 0x00FF00
#define DARKBLUE 0x000033
#define BLACK 0x000000
#define BLUE 0x0007FF
#define RED 0xFF0000
#define MAGENTA 0x00F81F
#define WHITE 0xFFFFFF
#define PURPLE 0xCC33FF
```

g
b
r

8 bits → [0,255]

Note: More discussion Later.

```
void lcd_init()
{
 int i;
 printf("LCD Demo Begins!!!\n");
 // Set pins P0.16, P0.21, P0.22 as output
 LPC_GPIO0->FIODIR |= (0x1<<16);

 LPC_GPIO0->FIODIR |= (0x1<<21);
 LPC_GPIO0->FIODIR |= (0x1<<22);

 // Hardware Reset Sequence
 LPC_GPIO0->FIOSET |= (0x1<<22);
 lcddelay(500);

 LPC_GPIO0->FIOCLR |= (0x1<<22);
 lcddelay(500);

 LPC_GPIO0->FIOSET |= (0x1<<22);
 lcddelay(500);
```

Note: SPI About Communication Between a & b.

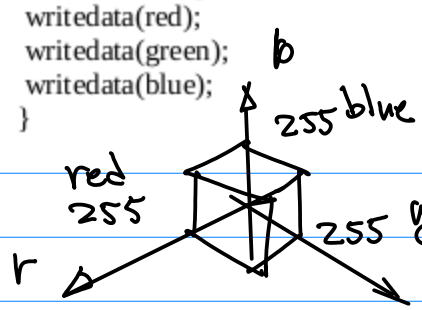
```
void spiwrite(uint8_t c)
{

 int pnum = 0;
 src_addr[0] = c;
 SSP_SSELToggle( pnum, 0 );
 SSPSend( pnum, (uint8_t *)src_addr, 1 );
 SSP_SSELToggle( pnum, 1 );
}
```

Note: Related to Color Space.

Note: Together with Draw Line function, we can build Different Apps.

```
void write888(uint32_t color, uint32_t repeat)
{
 uint8_t red, green, blue;
 int i;
 red = (color >> 16);
 green = (color >> 8) & 0xFF;
 blue = color & 0xFF;
 for (i = 0; i< repeat; i++) {
  writedata(red);
  writedata(green);
  writedata(blue);
 }
```

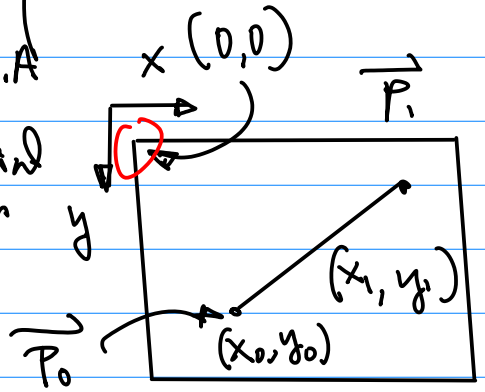
```
void drawPixel(int16_t x, int16_t y, uint32_t color)
{
 if ((x < 0) || (x >= _width) || (y < 0) || (y >= _height))
 return;

 setAddrWindow(x, y, x + 1, y + 1);
 writecommand(ST7735_RAMWR);
 write888(color, 1);
}
```




```
void drawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1, uint32_t color)
{
 int16_t slope = abs(y1 - y0) > abs(x1 - x0);
```

D.D.A = Digital Differential Algorithm

$y = ax + b$



Background ON 3D Graphics.

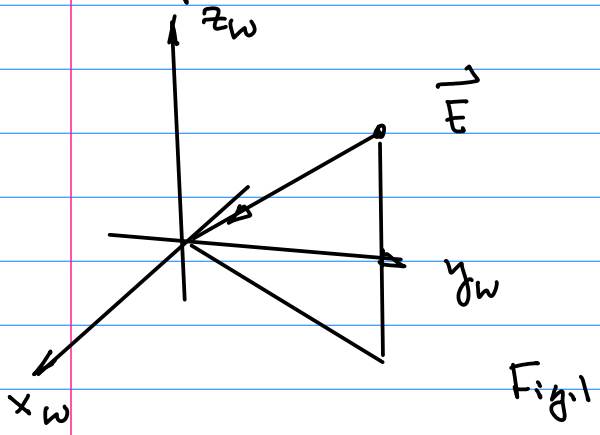


Fig.1

x_w-y_w-z_w World Coordinate System.

R.H. Right Hand System.

Define A virtual Camera:

its Location $\vec{E}(x_e, y_e, z_e)$

Direction of the Virtual Camera:

points to $(0,0,0)$;

Virtual Camera model: "Pin-Hole" model

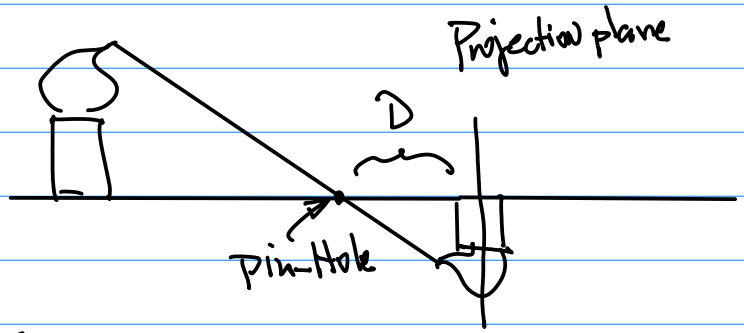


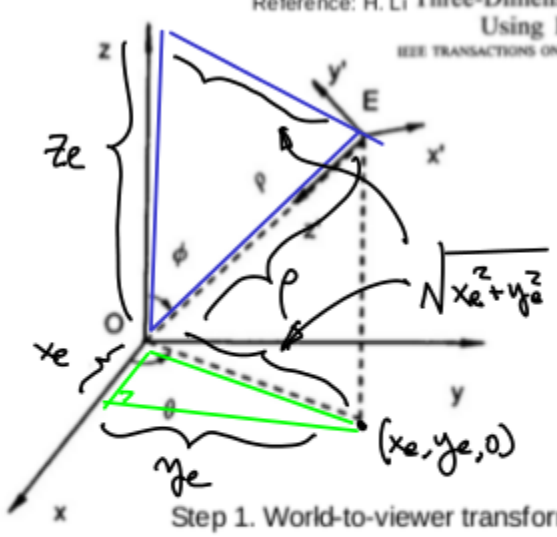
Fig.2

Design A Technique Called Transformation Pipeline to Be Able to generate 3D Graphics.

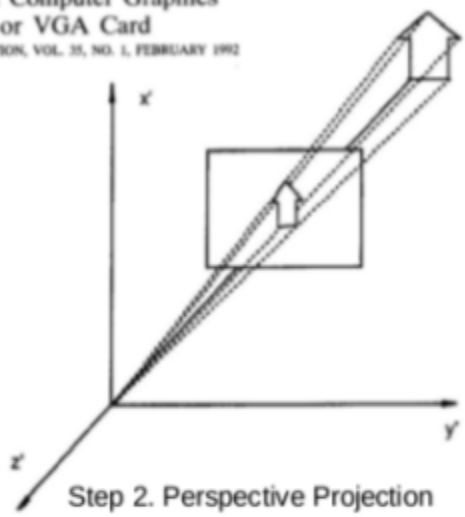
CMPE240
Spring2023

From: pp.32, github, Lecture Notes

Reference: H. Li Three-Dimensional Computer Graphics Using EGA or VGA Card
IEEE TRANSACTIONS ON EDUCATION, VOL. 35, NO. 1, FEBRUARY 1992



Step 1. World-to-viewer transform



Step 2. Perspective Projection

$$\mathbf{T} = \begin{bmatrix} -\sin\theta & \cos\theta & 0 & 0 \\ -\cos\phi\cos\theta & -\cos\phi\sin\theta & \sin\phi & 0 \\ -\sin\phi\cos\theta & -\sin\phi\cos\theta & -\cos\phi & \rho \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$x_p = x_e\left(\frac{D}{z_e}\right)$$
$$y_p = y_e\left(\frac{D}{z_e}\right)$$

Harry Li, Ph.D.