Jan 27 (Wed), 2021 | Harry Li

CMPE240 Welcome to 240 Section 2

Email: hua.li@sjsu.edu
Office Hours: M.W. 4:30-5:30 pm.
~~Zoom Based~~
Greensheet github/hualili
CMPE240/2018F

References:
1. Greensheet on github
(650) 400-1116 Text message
Prerequit Requirements 180D

Advanced Microprocessor Systems
Smart phones & RISC Architecture
5G, Edge AI → IoT, AI → GPU Architectural Aspects

Prototype System
Fully Functional Microprocessor System

Action Items:
1° github/hualili/cmpe240
2° Pre-requit Requirements, 180D
3° LPC1769 CPU module
digi-key.com OR mouser.com

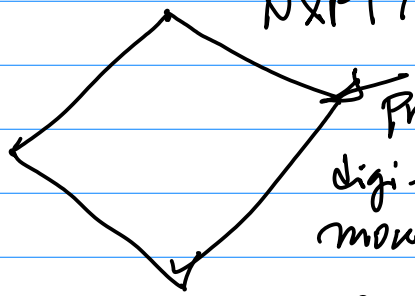
Handson: multiple projects, 3 mile stones

4° CPU Datasheet 5° MCU expresso

CMPE240 Feb1 (Mon) 1/

Todays Topics: 1° System Level Architecture — LPC1769
Ref: github/hualili/cmpe240/2018F-102 2° CPU Data-Sheet

Example:
1° CPU Module @ center of the System Layout Design
NXP1769



2° Wirewrapping Board
Dimension: 6"x4"
with Through-Holes, and one side of Board whose through-Holes with metal plating; But not the entire Board (just the throug-Holes)

3° PWR CKT: J1 Connector
Right Angle Connector; S/W Toggle Switch; IC Regulator 7805, 1117
Red LED & 10mA Resistor, Cap. (LPF)
Note: Debug/Development No External PWR CKT
NXP.com

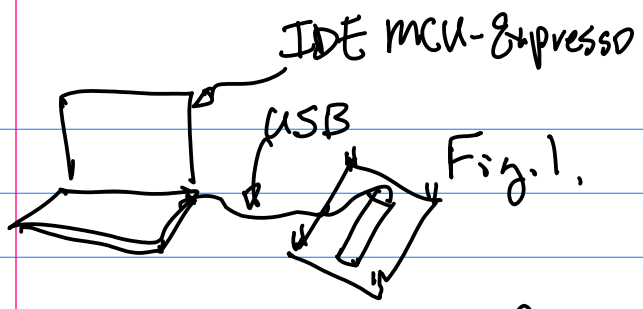
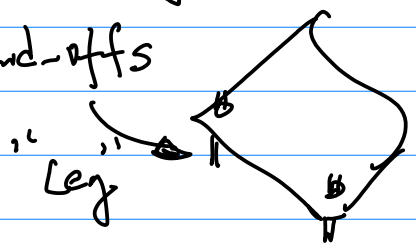


Fig. 1.



Ref: CPU module Design. pdf
github, Rev. D. or higher

From the Datasheet, Two tables on Each side of the CPU module
First, Left side, the outter table is for mbed, Not interested.

Naming Convention:

① CPU pin is named identical as the name appeared in the CPU Datasheet;

② The physical connector Label(s) are given on the SCH to match up the physical Board

Question: Find the pin for GND From SCH,

Connectivity Table

Connector J2 Position

Functional Pin	Physical Pin	Note
GND	J2-1	
VIN	J2-2	4.5~5.5VDC

Enumeration of the pin(s)
CPU pin (1st pin) starts as "0";

CPU Datasheet 2018S-3-UM10360
CPU Architecture, pp. 9

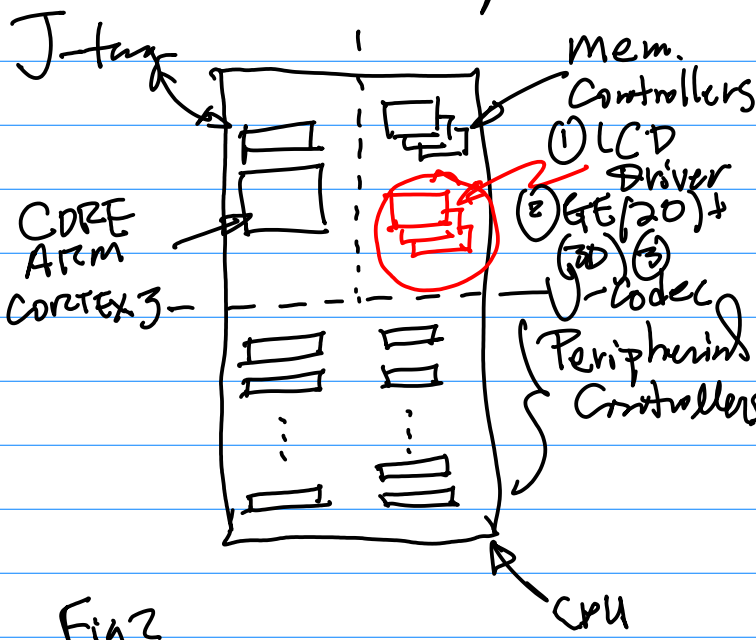


Fig 2.

Peripheral controllers (SCH, CPU Datasheet)

Feb 3. CMPE240 HARRY LI

Note: 1° Have your video when Participating the Class; 2° Honest Pledge Form/Sign CANVAS submission By Sat. 11:59 pm;

Today's Topics: 1° CPU Architecture/CPU Datasheet SCH; 2° Homework Design of prototype Board

Peripheral Controller

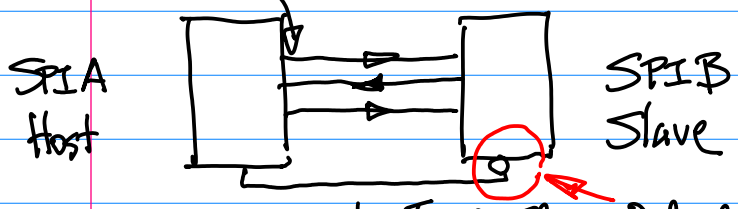
Ref: { ① CPU Datasheet, pp.9.
② SCH

Example: From SCH.

1° SPI Serial Peripheral Interface

Note: RESET pin has to Included in Our Prototype Design

MOSI: Master Output Slave Input



MISO: Master Input Slave Output

SCK: Serial Clock (O)

SSEL: Enable SPI Controller (ON the Slave Side, Active Low)

Note: $MOSI_x$, $MISO_x$, SCK_x, $SSEL_x$, where x stands for the xth of SPI Controller

Question: Find Number of SPI I/F for this LPC1769 CPU?

Two SPI I/F SPI0, SPI1

2° UART (Serial I/F), Note RS232

{ TX (Transmitter)
Rx (Receiver)

3rd pin has to Be a part of it: GND

P0.0; P0.1

multiplexing, TX/SDA

Init & Config for Special Purpose Registers will define which function the pin will I/O assume.

Action 1: Homework — Read SCH, generate a table for all Peripheral Controllers

{ Table: CPU Datasheet, pp.9
Table2: SCH, to find subset of the I/O Controller.

Advanced Feature:

G.E. Graphics Engine

Example: Broadcom

Pie3Bt, 4 → GE

NUDA: GPU (Graphics Processing Unit)

NAND 128 GPU, TX2, 64 GPU + 256 GPU

"3+1" Pins

{ MOSI/MISO/SCK
SSEL (Enable)

Software Implementation for G.E (2D, 3D)

Basic Concepts.

1° RISC (Reduced Instruction Set Computer) { ARM MIPS

32 Bit

Address Bus (Uni-direction) : 32 Bit
Data Bus (Bio-direction) : 32 Bit
Register Files { General Purpose Register
Special Purpose Reg.
SPRs

Memory : $2^{32} = 2^{10} \cdot 2^{10} \cdot 2^{10} \cdot 2^{2} = 4$ GB
1K 1M 1G Byte

Note:
1° Byte Addressable
Smallest memory with an Unique Address is a Single Byte

CMPE240 Feb 8 (Mon), HARRY LI

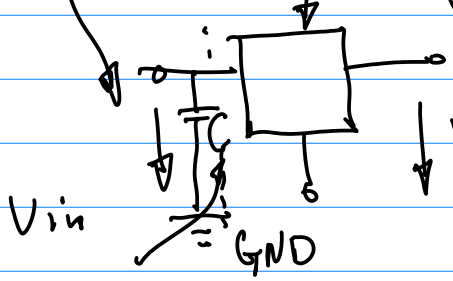
Today's Topics:
1° Bill of Material (github)
2° CPU Architecture
3° Preparation to Bring up your CPU, MCU Expresso

Two Step Process to Prepare for Prototype Board,
Step1: CPU module LPC1769
Wire wrapping (Prototype Board)
"Glue" Logic Resistor, Caps, PWR Regulator IC (7805), Red LED, Stand-offs

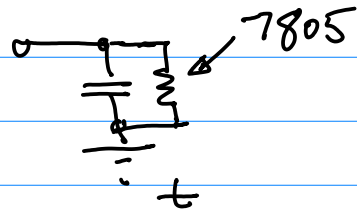
CMOS CPU ARM
Vout 3.3VDC, 10~30mA
Estimate 5.3V / 10mA, 3.3V / 30mA
3.3×10^{2} , 1.1×10^{2}
Ops.

Wall mount Adaptor
110ACV →
7.5VDC
1000~1500mA

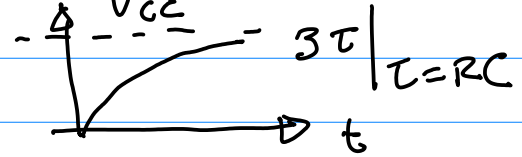
Note: PWR Regulator IC, 7805 (~2VDC Volt drop)



To Cap to Realize LPF.



$V_{cc}(1 - e^{-\frac{t}{RC}})$ ... (1)



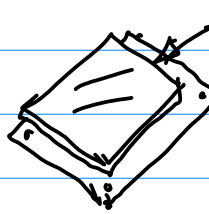
4.7 MF
2nd Part for Caps : 7805

Datasheet for 7805, Caps are those with Polarity "⊥" ⊤

Bill of the Material to Build G.E.

SPI I/F Based Color LCD Device. a SPI (Not I2C)

b Module — Connector Through-Hole

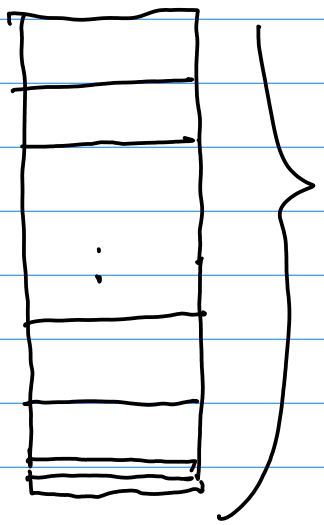


≅ ST7735R (IC) Software API

2-3 weeks

CPU Architecture Discussion

1° memory map



$2^{32} = 4\,GB$

2° Byte Addressable Machine ~ whose Smallest mem. cell With unique address is a Single Byte

3° memory Bank: 8

$4\,G/8 = 2^{10}\cdot 2^{10}\cdot 2^{10}\cdot 2^{2}/2^{3}$

$= 2^{9}\cdot 2^{10}\cdot 2^{10} = 512\,M$

Question: How many Bits from the Addr. Bus do we need to uniquely define Each memory Bank? 3 Bits

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

0 0 0 ⋮

1st BANK — LSBit

Starting Addr. of the 1st Bank, 0X0000_0000

Little Endian

What is the Starting Address of the 2nd BANK:

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

0 0 1 ⋮ 0

0X2000_0000 ; — 2nd BANK

0X4000_0000 ; — 3rd BANK

CmpE240 Feb10 (Wed)

Ref: github/hualili/cmpe240 ... 20|8F-107-LecGPP

Homework 1. Form 4-Person Team

First, Last Name, Last 4 Digits SID

E-mail Address → Submission

Via Email & Canvas By Thursday 11:59

Subject Title → Team Coordinator First_Last_CmpE240

Document Name

E-mail Submission.

Submission:
1° Project Exported in Zip format
2° White paper, Report
3° Video Clip (up to 5~7 seconds) short please.

Homework 2: (1 pt)

Requirements ① Build a prototype Board, ② Write a first program Turn On/off LED "Hello, the world"

Note: Use Prototype Board to Build GPP I/O to Drive LED ON/OFF

③ Build I/O Testing Circuit, the CKT input is from GPP (GPIO)

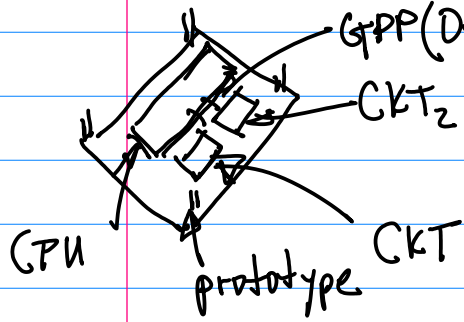


Fig 1.

CKT1: For Output Testing { "1" to Turn on LED; "0" to Turn off LED

CKT2: For Input Testing

{ Input "1", Toggle Switch to connect GPP Input to Vcc (via a Resistor)
Input "0", Toggle the switch to GND (via Resistor)

④ Write One page Report (IEEE paper format) White paper Due ON CANVAS

Due 2 weeks from Today, Feb 24.

Example: From PPT ON GPP I/F.

Identify CPU GPP pin

P0.3 J2-22 Input
P0.21 J2-23 Output

Consider the Output Testing CKT1

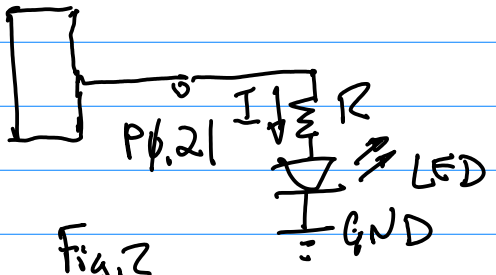


Fig 2

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

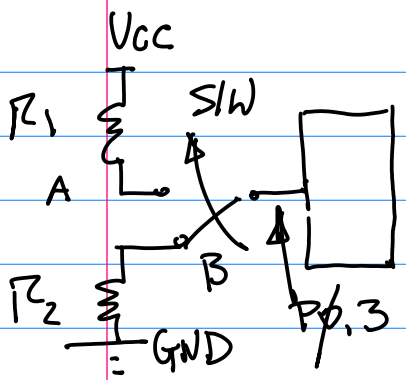
$V_{CC} = 3.3$ VDC

$I \cong 10$ mA, $V_{LED} = 1.2$ VDC

$3.3 = 10 \times 10^{-3} R + 1.2$

$R = (3.3 - 1.2) / 10^{-2} = 210$ Ω

Now, Let's Design CKT2



SW:
@ A

R_2: Output $V_{CC}=3.3$
$I = 4\,mA$
$R_2 = 3.3/4\times10^{-3}$
$\simeq 1K\Omega$

R_1: Output, GND
$V_{CC}/R_1 = 4\,mA$, $3.3/4\,mA \simeq 1K\Omega$

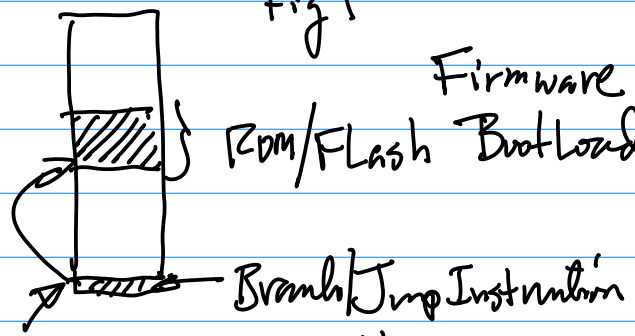
System Level & Software Design

1° CPU Architecture → mem. map
Peripheral Controllers
GPP Controller

BANK 0
a PWR-up Address
= 1st Address

0X0000_0000 CPU when powered up
it will fetch the 1st Executable Instruction at this Location

Fig 1



0X0000_0000 power up Addr.

Feb 15 Monday

Today's Topic: GPP I/O Testing
CPU Architecture & Special Purpose Registers.

Power-up Addr: Addr. when CPU is powered, it will fetch the 1st instruction to execute at this memory location.

Firmware — A program which has been burn into ROM/Flash

Question: How do you write C program to perform GPIO init & config? so that I/O I/F can be established?

Background:

Hardware: Peripheral Controllers
Software

GPP General Purpose Port (GPIO)

GPP Controller

Special Purpose Registers
32 bits:
Their function:
a Control function: Init & Config
b Data Exchange Function:
c pull up/Down

CMPE240

Naming convention:

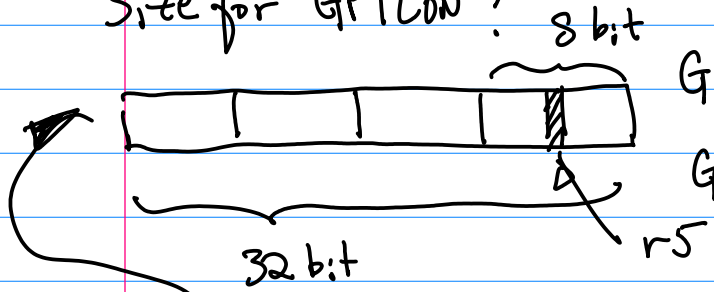
Prefix + Root + Postscript

3 3 (3)

GPx CON

General purpose GPxCON

GPP_Port 1 — What is the Size for GP1CON?



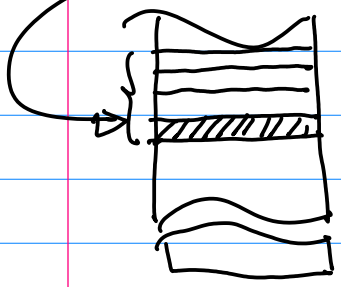
GPxCON[3:0]

GPxCON[5] = 1

= 0

Note: 1° 0x uuuu_uuuu

Suppose 0x4000_0200



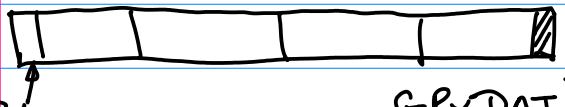
4 Address (Bytes) per SPR.

Fig2

Question: How many Control function can one Control Register define?

$2^{32} = 4G$

Now, GPxDAT, 0x4000_0204



GPxDAT[0] Fig3.

LSB (Little Endian)

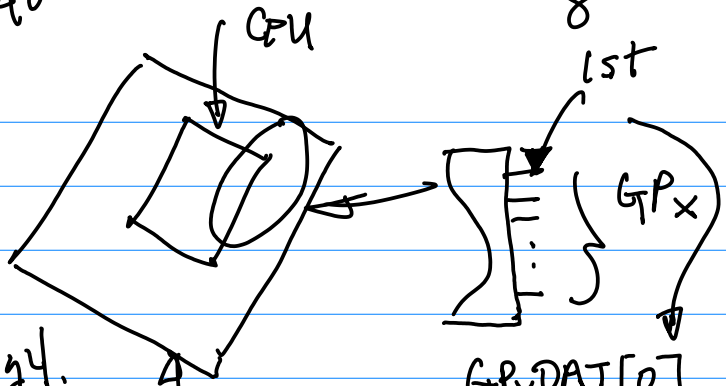


Fig4. 4 Prototype Board

GPxCON[0] = 1

Action 1: LPC1769 patch.zip

Import it as a project into your IDE

Test run blenky Program

To modify the GPP

So P0.2 & P0.3

Can be employed for I/O Testing

Note: 1° Project Panel on the top left of your IDE GUI → select "Blenky" By highlighting it.

CMPE240 Feb 17 (W)

Ref: 1° github/hwalsh/Cmpe240/2018S-11-GPIO2015-1-30.zip

2° ~/Cmpe240/2018-10-LCD_DrawLine.zip

3° ~/2018F/2018F-107-~
(For GPP)

4° ~/2018F/2018F-109
(For SPI LCD)

5° ~/cmpe240/20185-10-~
DrawLine.zip
(For 2D GE-Line Plot)

Topics: 1° SPRs for SPI LCD I/F; 2° SPRs for GPP.

Init & Config of Peripheral Controller(s)
- GPP
- SPI (Serial Peripheral Interface)

Note: External Connector → CPU GPP. J2-x → P0.2, etc. → SPR

Example:
SPI Interface for color LCD Display.
- Hardware Design
- Software Design

Consider Hardware Design Block Diagram

Step 1
SPI Host

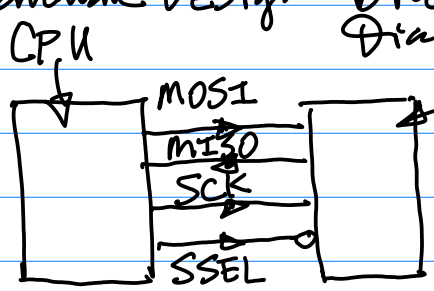


Fig1.

Pin Connectivity "3+1"

CPU	SPI(s)	Note
MOSI P0.9/J2-5	SI	CPU (O)
MISO P0.8/J2-6	SO	CPU (I)
SCK P0.7/J2-7	CLK	CPU (O)
SSEL P0.6/J2-8	$\overline{CS}$ (nCS)	CPU (O) Active Low

Table1: Connectivity Table

2018S-8-SPI (photo Board)

2018F-108-LCD-Connector

Note: Connectivity Table for CPU to SPI LCD Display Device.

Action 1: Solder up the SPI LCD Device;

Software Design

Background:
1° (0,0) X-Y Coordinate origin (0,0) upper Left Corner

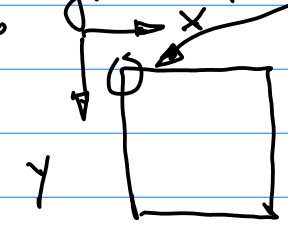


Fig2

Resolution: M x N

M: No. of pixels per row
N: No. of Rows per Frame

Example: 160 x 120
- 160: Width, No. of Pixels/Row
- 120: Height, No. of Rows

2° I(x,y) Image plane(s)

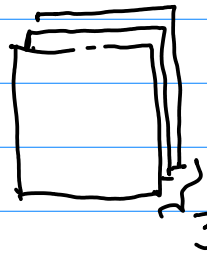


Fig.1

r: red;
g: green;
b: blue.

$I(x,y) = (r(x,y), g(x,y), b(x,y))$ ... (1)

Development Environment
IDE ≠ SDK
1° Ref: 2015-10bb-~
Note: IDE Not Backwards Compatible
Version 11.3.0 User Guide

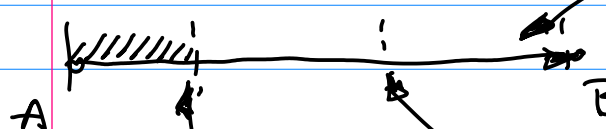
2° Selection of Target CPU, Target Board
LPC1769 → Board
(Picture of Board is provided)

Quick Start Panel → New Project → Search Target Board → C semi-hos
LPCXPESSO 1769

3° Ref: 2015-10b- homework

4° Report Template 2018-10~

Feb 22 (monday)



Discussion

GPP I/O Testing

① CPU Architecture
- CPU Block Diagram
- Memory map

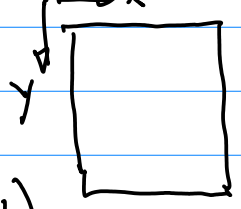
② SPR_Init&Config.
- Peripheral Controller
- GPP+SPI

③ Implementation

Discussion On Graphics

1° Graphics
- Pixel Graphics I(x,y)
- Vector Graphics → 2D & 3D G.E.

2° Display Device



a Scanning (Image Formation)
Start @ (0,0) → L2R one pixel at a time, → T2B one row at a time.

b Timing Calculation

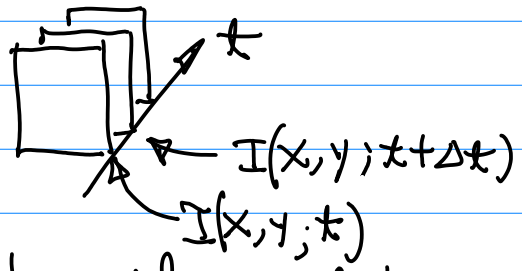
b1 Frame Rate FPS (Frame Per Second)

30 FPS

$f_F = 30\,Hz$; ... (1)

$T_F = 1/f_F = 33.3 \times 10^{-3}$ Sec.

Note: GPP PPT 2018F-107



b2 Horizontal Timing (clock)

Given Graphics/Image with M x N

$f_H(Sync_H) = N\, f_F(Sync_F)$ ... (2)

b3 Pixel Timing (clock, Data Clock)

$f_D(Sync_D) = M\, f_H(Sync_H)$ ... (3)

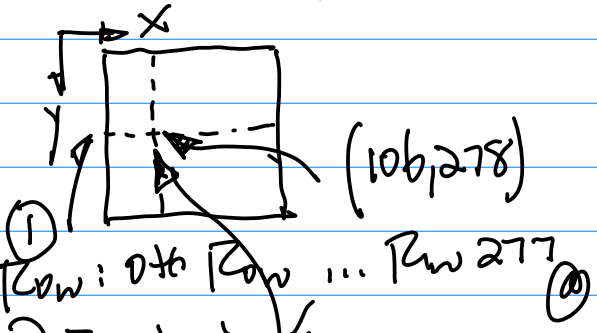
Example: Given Graphics I(x,y) with 1024 x 768, And fixed Timing for pixel I(106,278)

(m = 1024, N = 768)

Sol: $Sync_F = 30\,Hz\ (= f_F)$

$Sync_H = N \cdot Sync_F = 768\, Sync_F$

$Sync_D = M\, Sync_H = 1024\, Sync_H$

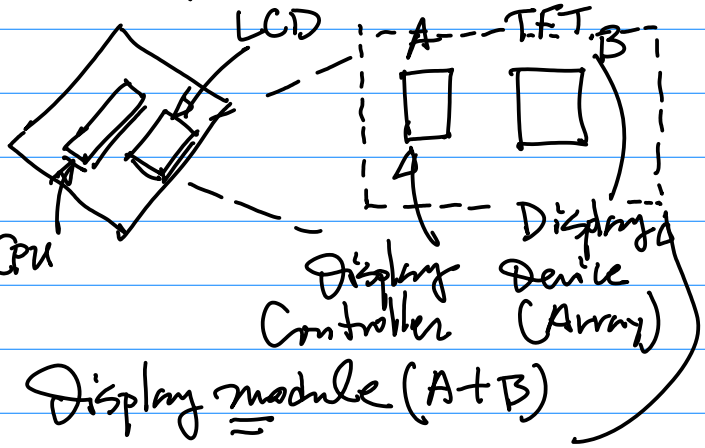


Therefore,

$T = T_{\Sigma H} + T_{\Sigma D}$

$= 278\,\frac{1}{Sync_H} + 106\,\frac{1}{Sync_D}$ //

3° Timing in terms of Graphics Display Controller Design.



Design of the Display Controller Timing Scheme

T.F.T. Display

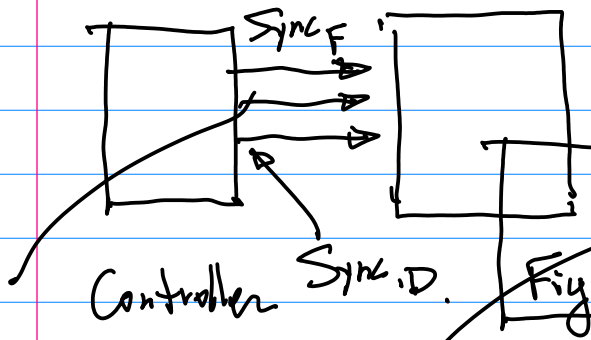


Fig. 1

Compile & Build → Run Debug

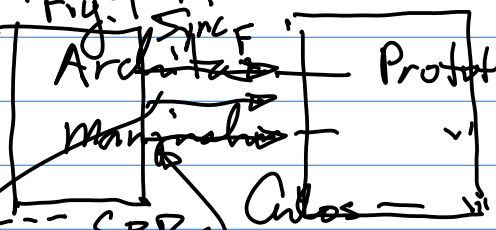
Drawline. OK

Probes Loading. error: migration is failed ... Cancel the Session

Try Again

(LPC1769 patch imputed already)

Software Aspects of SPI/Controlling Implementation



March 1

Fig. 1

Flow of the Discussion

C Implementation → Graphics Vector Graphics (C-Code) → SPI C-Code → SPRs

Drawline.zip (Project) MCUXpresso

Controller

SPRs ↔ CPU Datasheet

Topics: 1° Finish Hardware Design (Graphics Driver)

Example: Import/Build/Debug Drawline.zip Execute

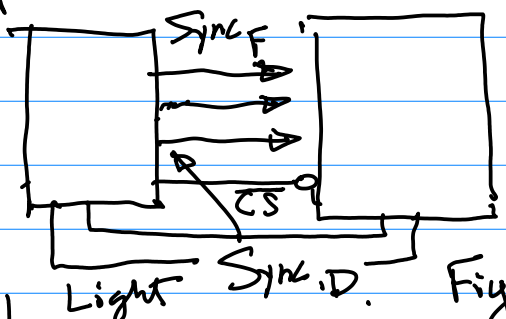


Fig. 16

Note:

1° MCUXpresso Version, 11.3.0 ✓

IDE { : SDKi (LPC1769) SDKj : }

a ≙ No SDK for LPC1769

b LPC1769 Pre-Built for IDE

Control Signal:

1° nCS ($\overline{CS}$) Controller Enable

2° Data/Command D/$\overline{C}$, or $\overline{C}$/D

No SDK for LPC1769 → NXP Open MCU → Check LPC1769 Pre-installed ✓ → Import LPC1769 Patch.

Compile & Build ← Import Drawline.zip ← github

Controller to LCD Display.

Part II:

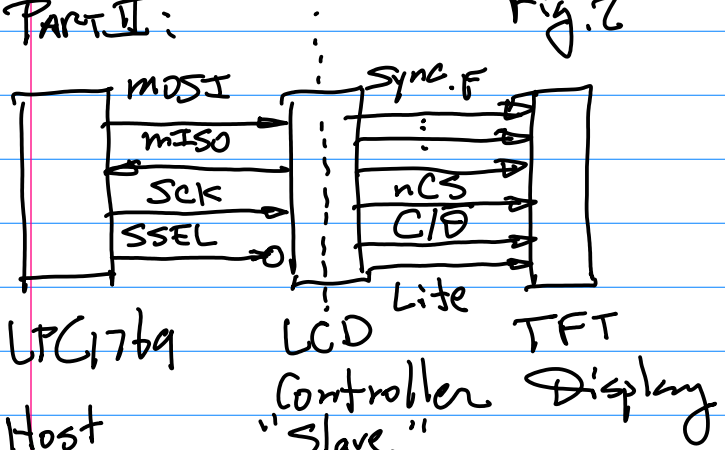


Fig. 2

2D Graphics

Action 1. LCD Wire wrapped on the prototype Board

Action 2. Test DrawLine Project

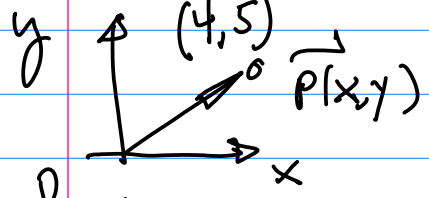
Note:

1° mathematical Formulation

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

2D vector

Simplified Short Hand Notation



$\vec{P}(4,5) = (4,5)$

2° Generate A screen Saver Based on G.E. (Vector Graphics)

Given a Set of vectors (Vertex)

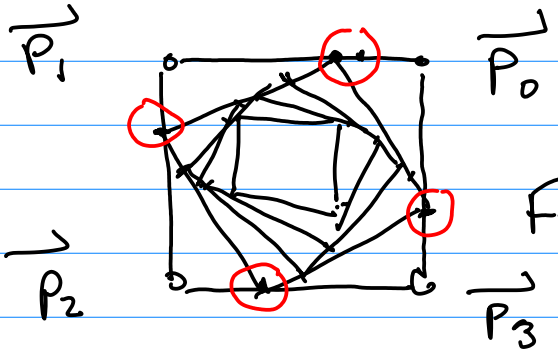
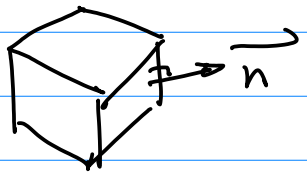


Fig. 3

Note: $\vec{P_0}, \vec{P_1}, \cdots, \vec{P_3}$ are arranged in the Counter Clockwise Direction C.C.W.



3° Vector Representation of line a fix pt. and a direction

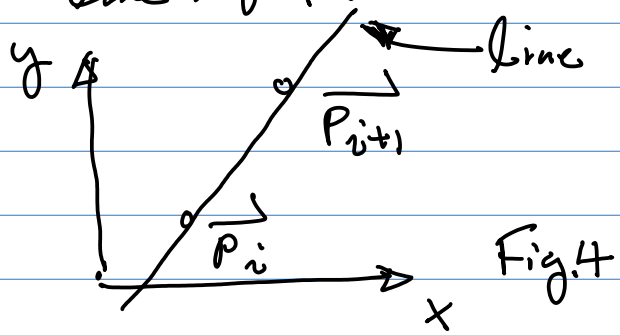


Fig. 4

Define Direction Vector

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

Example: Given $\vec{P_i}(3,4)$, $\vec{P_{i+1}}(1.5,-1.1)$

Find Direction Vector ?

Sol: $\vec{d}(x,y) \triangleq \vec{P_{i+1}}(1.5,-1.1) - \vec{P_i}(3,4)$

$= (1.5,-1.1) - (3,4) = (-1.5,-5.5)$

4° Define A Line Starting pt $\vec{P_i}$

Ending (2nd) pt $\vec{P_{i+1}}$,

$$\vec{P}(x,y) = \vec{P_i}(x_i,y_i) + \lambda \vec{d}(x,y) \quad \cdots (1)$$

$\vec{P}(x,y)$: Arbitrary Pt. on the Line; λ: Scaling factor; $\vec{d}(x,y)$: Direction Vector

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

From Eqn(1)

$\lambda = 0$, $\vec{P}(x,y) = \vec{P_i}(x_i,y_i)$

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

$0 < \lambda < 1$, Pt Between $\vec{P_i}$ and $\vec{P_{i+1}}$

5° See Fig 3, P.P. 13

From Eqn (1b)

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

for $i = 0, 1, 2, 3$

Introducing Super Script to Eqn (1b) for the Level of iterations

initially, Super Script Starts at ϕ. One Level higher.

$$\vec{P}_i^{1}(x,y) = \vec{P}_i^{0}(x_i^0, y_i^0) + \lambda\left(\vec{P}_{i+1}^{0}(x_{i+1}^0, y_{i+1}^0) - \vec{P}_i^{0}(x_i^0, y_i^0)\right) \quad (3)$$

↑ pt i @ next level, e.g. One level higher

Generalize it,

$$\vec{P}_i^{j+1} = \vec{P}_i^{j} + \lambda\left(\vec{P}_{i+1}^{j} - \vec{P}_i^{j}\right) \quad (4)$$

$$\vec{P}_i^{j+1}(x_i^{j+1}, y_i^{j+1}) = \vec{P}_i^{j}(x_i^j, y_i^j) + \lambda\left(\vec{P}_{i+1}^{j}(x_{i+1}^j, y_{i+1}^j) - \vec{P}_i^{j}(x_i^j, y_i^j)\right) \quad \cdots (4\text{-}1b)$$

$$\begin{cases} x_i^{j+1} = x_i^j + \lambda\left(x_{i+1}^j - x_i^j\right) & \cdots (5a) \\ y_i^{j+1} = y_i^j + \lambda\left(y_{i+1}^j - y_i^j\right) & \cdots (5b) \end{cases}$$