Sept.7.

Note: 1° LPC1769 from 2022S Semester, WaitingList. CANVAS. Scott.

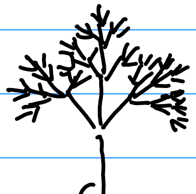
2° LPC1768 pin-to-pin Compatable. (Mbed)

a. Step1. MCUXpresso IDE
1768 Binary Code.

Step2. "Firmware" Upload the binary file to the Flash. Need a prob

Step3. Interactive Debugging.

3° LPC11C24 Digi-Key in Stock.
LPC1114 Size
GPP/SPI, FLASH (ON-Chip)
1/8 of the size
Comparing to LPC1768/9



Homework (0 pt)

1. Form A Team By Wednesday.
4-Person

2. Select/Finalize your target platform. By the end of the week.

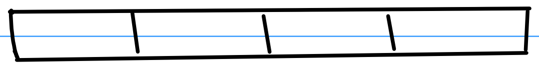
Example: Register File

Special Purpose Registers
General Purpose Register

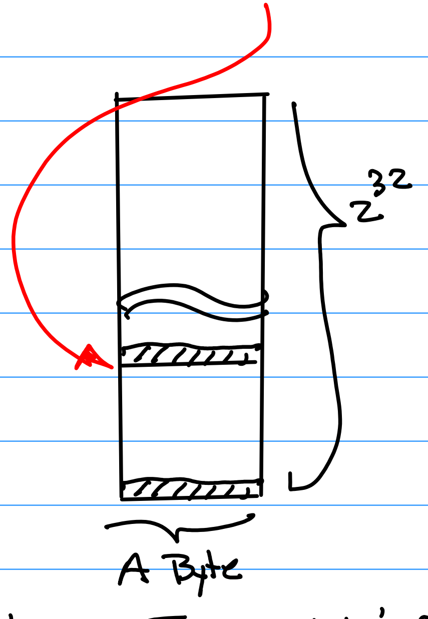
GPx CON

Prefix for Port "x", x=0,1,2,3

Root 3 Letters



GPxCON its address is 32 Bits, it maps to the memory map.



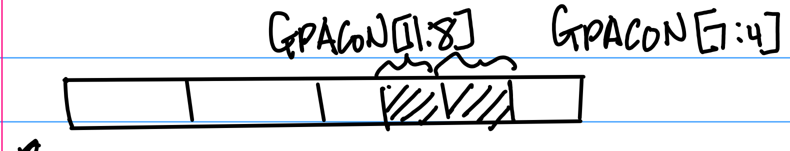
Note: The Task of Init & Config Can be realized by using HLL (High Level Languange), C/C++, to deposit A Binary Pattern to that memory Location (Addr. is a Pointer)

CmpE240

Sept.7

11

For Example for Samsung ARM-11.



GPxCON its address is 32 Bits, it maps to the memory map.

Consider: Power up Address + Power Process. Booting.

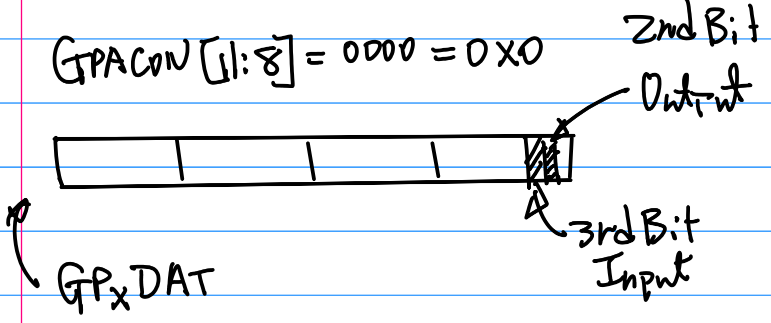
Design Requirements (Spec.)

1. 2nd Bit As An Output
2. 3rd Bit As An Input

To perform Init & Config.

GPACON[7:4] = 0001 = 0x1

GPACON[11:8] = 0000 = 0x0



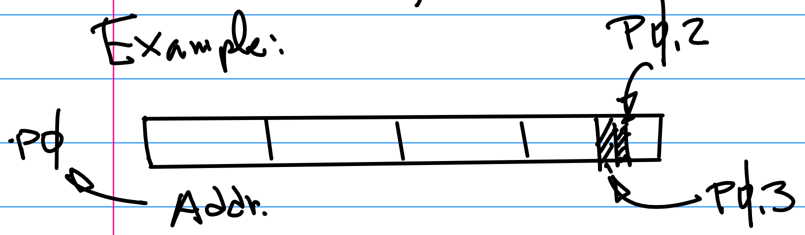
GPACON[3:0] = 0x10

Sept. 12 (monday)

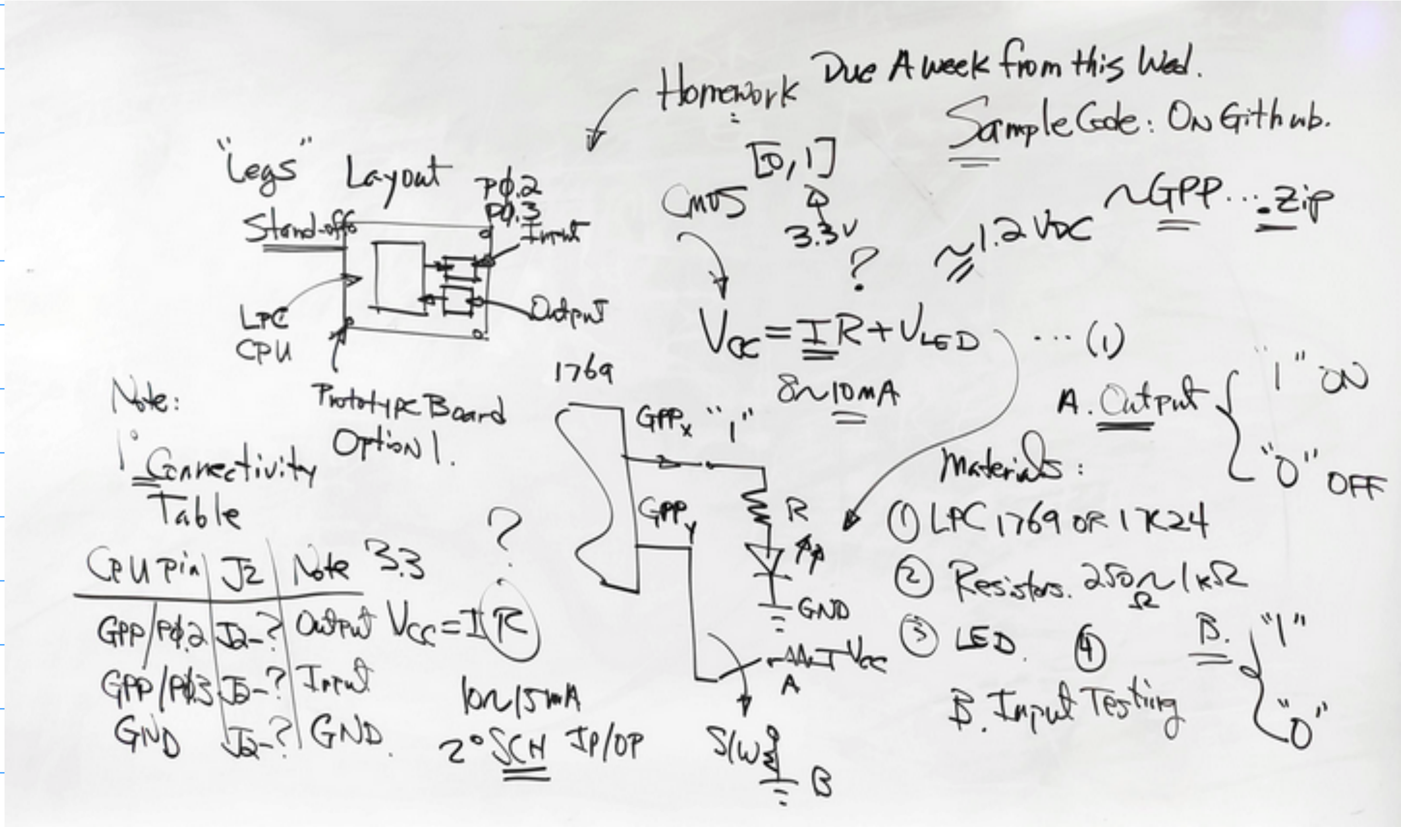
1. Homework (2pts)
2. Special Purpose Register

Note: Target platform.
LPC1768, LPC11C24

Example:



Sept.12(monday)



Homework: Due A week from this Wed.

Sample Code: On Github.

~GPP... .zip

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

8~10mA

Note:

1° Connectivity Table

CPU Pin	J2	Note
GPP/Pϕ.2	J2-?	Output
GPP/Pϕ.3	J2-?	Input
GND	J2-?	GND.

$V_{cc} = I\,R$

10~15mA

2° SCH IP/OP

A. Output { "1" ON / "0" OFF

Materials:

(1) LPC 1769 OR 1K24

(2) Resistors. 250Ω~1kΩ

(3) LED.

(4)

B. Input Testing

B. { "1" / "0"

Sept.14 (Wed).

Note: 1° Check Homework Assignment ON CANVAS.

Two Options { Prototype Board / e-Bay, Board B.

Topics today: IDE

1° GPP Software/Program

2° 2D Graphics Processing Engine Design.

Example: Setup the Expresso. 2

Key points:

1° Make Sure Select Target Board LPC1769. (Ref. on github, 35 slides)

2° C/C++ Project Settings. →

"Semihost"

3° Import LPC1769 patch.

1769 patch.zip

Note: 1° LPC1769 Patch is already Config by NXP.

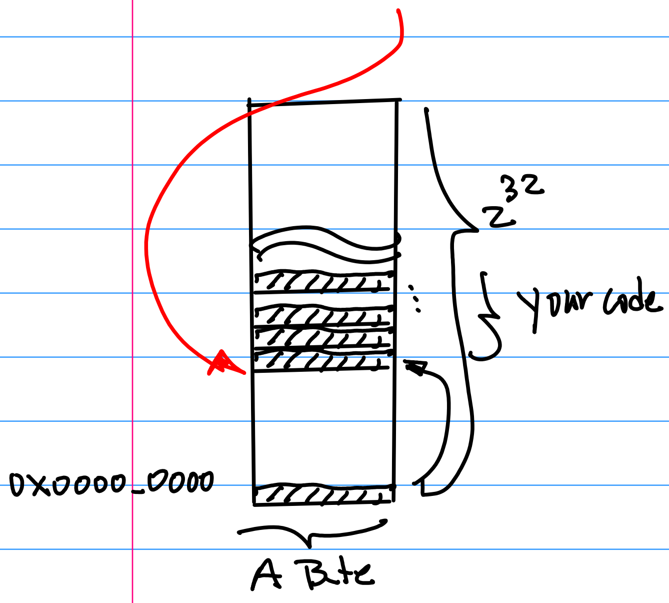
2° prob issue → fix:

Reconnect OR Reboot.

Import GPIO project to your MCU Xpresso.

CmpE240
Sept.14,22

Run "Debug", Once prob is detected then Your Program (Binary)



Consider G.E. Design.

Math. Formulation, for Vector Graphics.

Example.

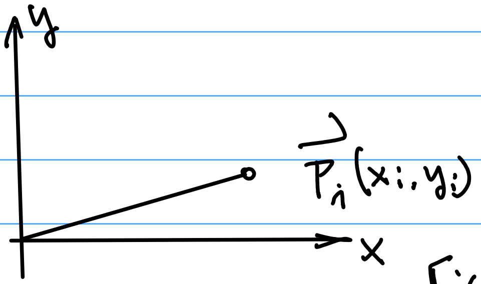


Fig.1.

$\vec{P}$ A Point → $\vec{P}(x,y)$ → $\vec{P_i}(x_i,y_i)$
for $i = 0,1,2,\ldots$
↓
(x_i, y_i)

Also, a line

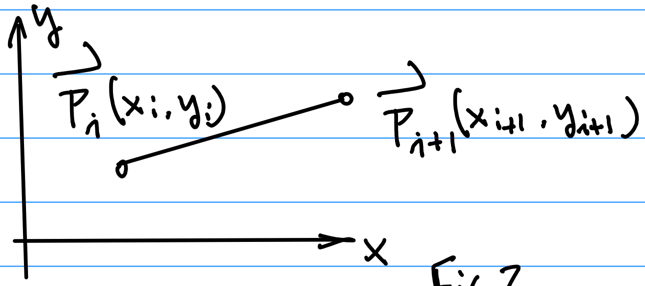


Fig.2.

Note: To Uniquely Define A Line, we can add a directional vector, Denoted as

Definition 1:

$$\vec{d}(x,y) \triangleq \underset{\overset{\uparrow}{\text{Ending pt.}}}{\vec{P}_{i+1}(x_{i+1},y_{i+1})} - \underset{\overset{\uparrow}{\text{Starting pt.}}}{\vec{P_i}(x_i,y_i)} \quad \ldots(1)$$

Definition 2: (Line Eqn. in Vector Form)

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

$$-\infty < \lambda < +\infty$$

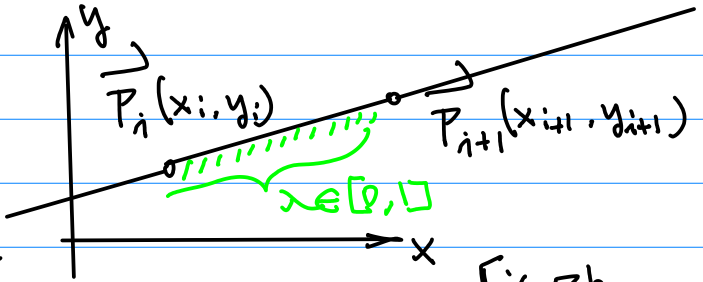


Fig. 2b.

Observation 1:

When $\lambda = 0$, Eqn(2) gives the Starting pt. $\vec{P_i}(x_i,y_i)$

$\lambda = 1$, " " Ending point $\vec{P}_{i+1}(x_{i+1},y_{i+1})$

$0 < \lambda < 1$, $\vec{P}(x,y)$ Any Arbitrary pt Between $\vec{P_i}(x_i,y_i)$ and $\vec{P}_{i+1}(x_{i+1},y_{i+1})$

Sept. 21. Today's Topics:

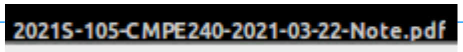
1° Check CANVAS for the Submission of the In-Class Exercise;

2° Graphics Lib has been ported to LPC11C24, Ref. Code & PPT will be provided.

$M \times N$

No. of Rows

No. of Pixels per Row (col).

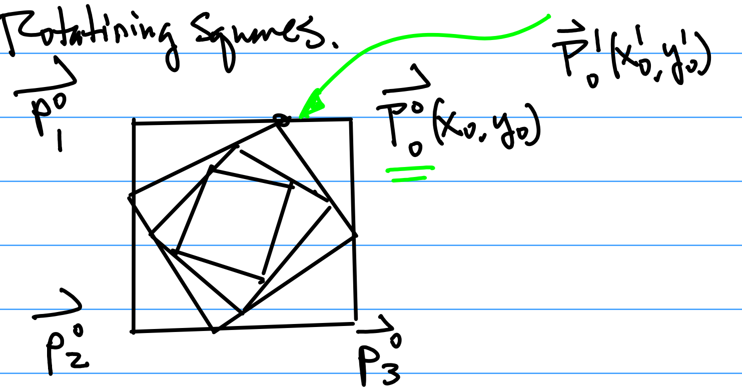

2021S-105-CMPE240-2021-03-22-Note.pdf

$\vec{P}_0^0(x_0^0,y_0^0), \vec{P}_1^0(x_1^0,y_1^0), \dots, \vec{P}_3^0(x_3^0,y_3^0)$ are defined as the same data pts on pp15.

Example: Given the equation Below,

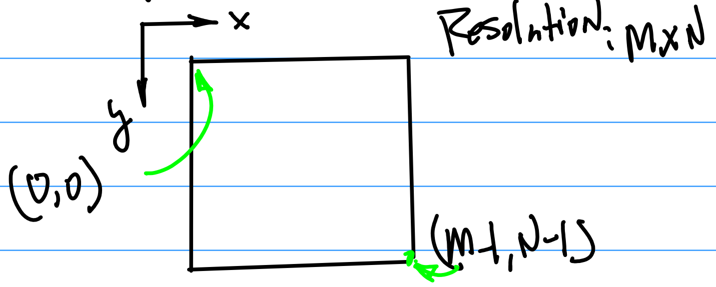
$$\vec{P}_i^{j+1}(x,y) = \vec{P}_i(x_i^j,y_i^j) + \lambda\left(\vec{P}_{i+1}(x_{i+1}^j,y_{i+1}^j) - \vec{P}_i(x_i^j,y_i^j)\right) \dots (1)$$

Let's Design A Graphic Algorithm to Create A Set of Counter Clockwise (CCW) Rotatining Squares.



Step 1. Define the initial set of Data Points $\{\vec{P}_i(x_i,y_i) \mid i=0,1,\dots,3\}$ for Resolution of A Display Device defined as $M \times N$

$\text{Max}(M,N) \le 200$, Select the Size of the Cube $\simeq 50$



Step 2. Get Line Equation, Based on Eqn (1). First pt on Level 1

$$\vec{P}_0^1(x_0^1,y_0^1) = \vec{P}_0^0(x_0^0,y_0^0) + \lambda\left(\vec{P}_1^0(x_1^0,y_1^0) - \vec{P}_0(x_0^0,y_0^0)\right)$$

$$= (60,60) + \lambda\left((10,60) - (60,60)\right)$$

$$= (60,60) + \lambda(-50,0)$$

From the Given Condition, CCW Rotation Let $\lambda = 0.2$

Note: Based on the Above Calculation Can be Conducted for the Rest of the pts, And Rest of the Levels.

Step 3. Suppose we want to generate 10 Levels of the Rotating Squares. Let Write C/C++ for this Purpose.

From Eqn(1), we have

$$\begin{cases} x_i^{j+1} = x_i^j + \lambda (x_{i+1}^j - x_i^j) & \ldots (2a) \\ y_i^{j+1} = y_i^j + \lambda (y_{i+1}^j - y_i^j) & \ldots (2b) \end{cases}$$

```
x[i][j+1] = x[i][j] + lamda*(x[i+1][j] - x[i][j]);
y[i][j+1] = y[i][j] + lamda*(y[i+1][j] - y[i][j]);
```

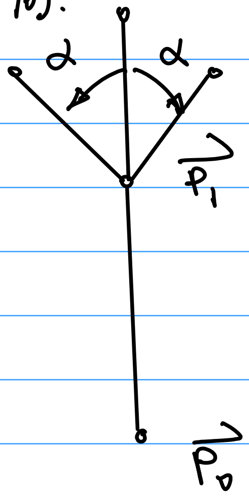
Consider Creating A Screen Saver
By Generating A tree.

Note: Level 0: Tree Truck;

Level 1: $\begin{cases} \text{Branch (Main)}: \text{Mag. Reduction By } 20\% \quad \lambda = 0.8 \\ \text{Side Branch} \begin{cases} L(CCW), \text{ Rotation by } \alpha \ (\pi/6). \\ R(CW), \ \alpha < 0 \end{cases} \end{cases}$

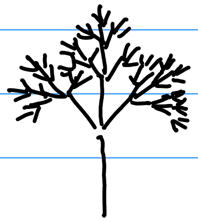
Same Direction, Directional Vect. Same.

α: Positive $\alpha \geq 0$



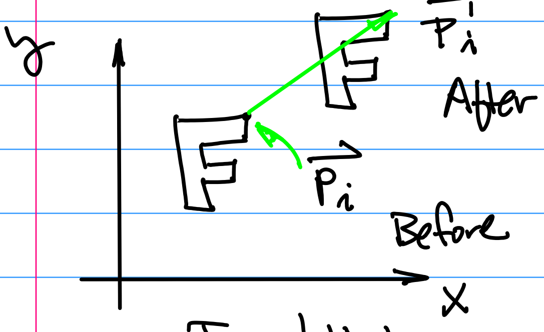
level 2, 3, ..., k.

Repeat Level 1 with Reference Vector (pt)
Updated Accordingly.

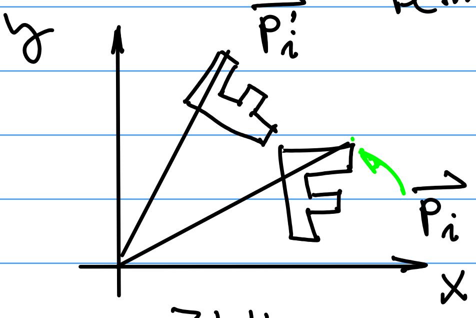


Background (2D Transformations)

Note: <u>Mapping</u> Between $\vec{P_i}$ & $\vec{P_i'}$, e.g.
Before & After.



Translation



Rotations

Sept.26.

Rotation: 1° Positive Angle is defined as a Counter ClockWise Rotation;

2° Reference pt is defined as the Origin.

3° physical Display (Coordinate System) v.s. Virtual Display (virtual Coordinate System).

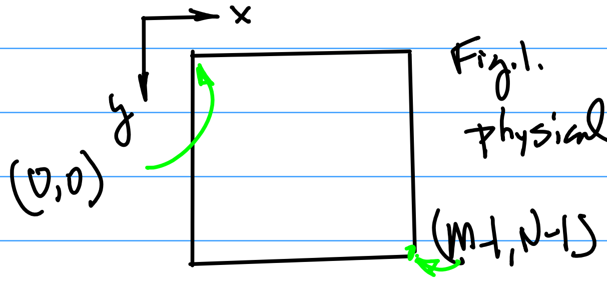


Fig.1. physical Display

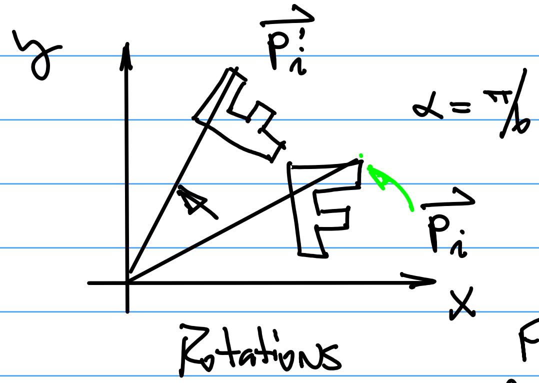


Fig.2 Virtual Display.

Example: for the Rotation illustrated in Fig.2.

pt (After) pt (Before) $\vec{P_i}$

$$\begin{pmatrix}x_i'\\y_i'\\1\end{pmatrix} \overset{?}{=} \Phi \begin{pmatrix}x_i\\y_i\\1\end{pmatrix} \quad \cdots (1)$$

pt (After) pt (Before) $\vec{P_i}$

$$\begin{pmatrix}x_i'\\y_i'\\1\end{pmatrix} = \begin{pmatrix}a_{11}&a_{12}&a_{13}\\a_{21}&\cdots&a_{23}\\a_{31}&\cdots&a_{33}\end{pmatrix}_{3\times3}\begin{pmatrix}x_i\\y_i\\1\end{pmatrix} \quad \cdots (1b)$$

3D Vector, With One "Dummy" Dimension.

Rotation Matrix for Eqn (1b)

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

$$\begin{cases} x_i' = x_i\cos\alpha - y_i\sin\alpha & \cdots (2\text{-}b)\\ y_i' = x_i\sin\alpha + y_i\cos\alpha & \cdots (2\text{-}c)\end{cases}$$

```
X_Prim[i] = X[i]*Cos(alpha)
                 - y[i]*Sin(alpha)
```

for the Reference of Doing C/C++ Coding.

Note: Reference pt. for the Definition of Rotation. This Rotation Can not be Simply defined by Eq(2)

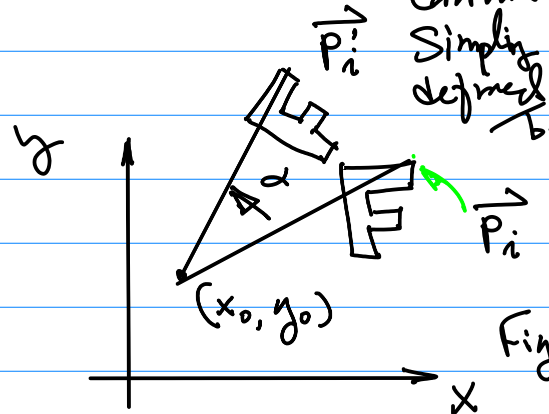


Fig.3

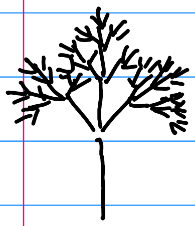


Fig.4a

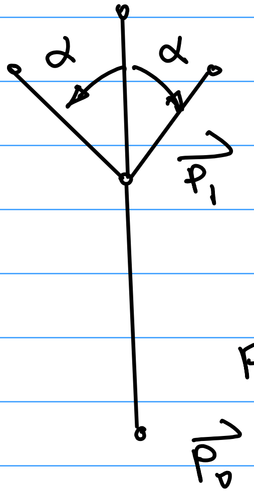
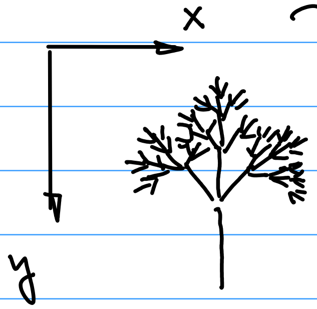


Fig.4b.

These are different Rotations than that in Eqn(2).

Consider the Composition of 2D Transforms.

Example: Build/Design to Realize a 2D Tree Pattern in Fig.4.



Mapping. ϕ_1

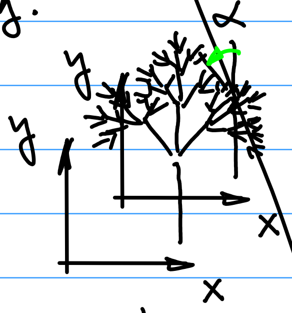


Fig.5a. Physical Display

Fig.5b. Virtual Display

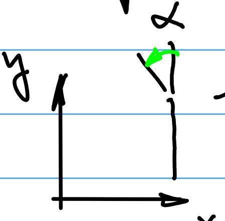
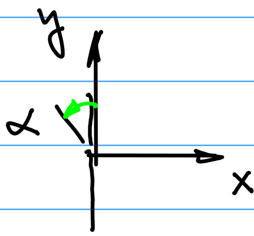


Fig.5C. Rotation By α. CCW.

Mapping: Preprocessing —



Translation. ϕ_2

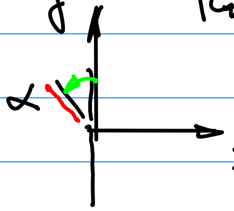
"M" matrix.

$T_{3\times3}$

Fig.5.d. Preprocessing By Translation.

↓

Rotation Eqn.(2).



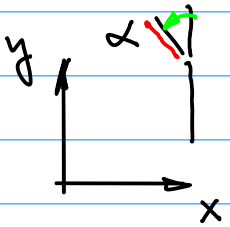
Rotation: ϕ_3

$R_{3\times3}$

Fig.5e. Rotation: Done

↓

Post Processing. To "move Back" to its original Position.



ϕ_4 Undo Preprocessing

$\phi_4 = \phi_2^{-1}$

or.

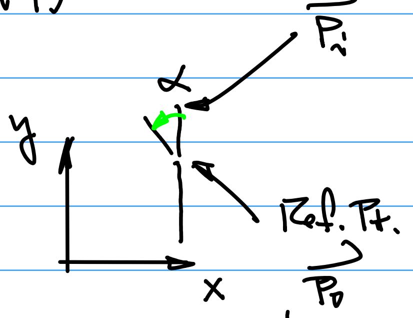
Fig.5f.

$$\phi_4 = \phi_2\Big|_{\phi_2 = T_{3\times3}} = T_{3\times3}^{-1} \quad \ldots(3)$$

Based on Step by step Analysis. (Analyze "Before" and "After" Relationship).

Start at given pt. to Be Rotated

$$\vec{P_i} = \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \ldots (4)$$



Next. Preprocessing ϕ_2, $T_{3\times3}$

To make $\vec{P_0}(x_0, y_0)$ to overlap with the origin (0,0).

$$T_{3\times3} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \quad \ldots (5)$$

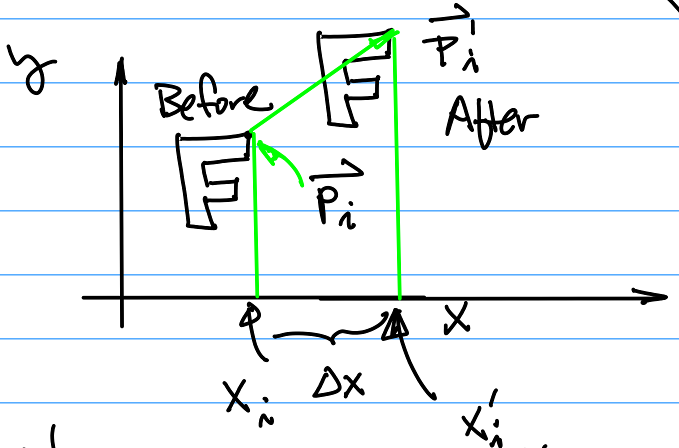
$$\begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = T_{3\times3} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \ldots (6)$$

P_i' After Before

$$= \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix}$$

$$x_i' = a_{11}x_i + a_{12}y_i + a_{13}\cdot 1 \quad \ldots (6b)$$

where $a_{13} = \Delta X$



Hence, $a_{12} = 0$, $a_{11} = 1$, then

So

$$x_i' = x_i + 0 + \Delta X = x_i + \Delta X \quad \ldots (6c)$$

$$y_i' = y_i + \Delta Y \quad \ldots (6d)$$

Therefore

$$T_{3\times3} = \begin{pmatrix} 1 & 0 & \Delta X \\ 0 & 1 & \Delta Y \\ 0 & 0 & 1 \end{pmatrix} \quad \ldots (7).$$

Note: $\Delta X = -(x_i - x_0)$,

$\Delta Y = -(y_i - y_0)$.

in Our Fig5. Series.

Now, Rotation, $R_{3\times3}$.

Then, Post Processing. ϕ_4

$$M_{\phi_3} = T^{-1}_{3\times3} = \begin{pmatrix} 1 & 0 & -\Delta X \\ 0 & 1 & -\Delta Y \\ 0 & 0 & 1 \end{pmatrix} \quad \ldots (8)$$

CmpE240
Sept.26,22

Put All these together, we have

$$\begin{pmatrix} x_i'' \\ y_i'' \\ 1 \end{pmatrix} = T_{3\times3}^{-1} R_{3\times3} T_{3\times3} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \ldots (9)$$

Rotated Post-Processed Point.

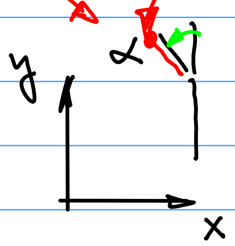


Fig.7

Sept.28 Wed.

Note: Conclusion on Composition of 2D Transforms is given in the Above Eqn.(9), Coding/Implementation in C/C++.

$$\begin{pmatrix} x_i'' \\ y_i'' \\ 1 \end{pmatrix} = \overbrace{T_{3\times3}^{-1} R_{3\times3} T_{3\times3}}^{T_\Sigma} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}$$

T_Σ is 3x3 matrix

↓

Last Row of $T_{\Sigma(3\times3)}$ is (0,0,1) No Need for Coding.

↓

Only the top 2 Rows from the matrix matter.

Step by step Derivation is also Provided at the Lecture Notes.

$$XL[i] = Cos\alpha * X[i] - Sin\alpha * y[i] + \Delta X Cos\alpha - \Delta y Sin\alpha - \Delta X$$

delta_x

Similarly for y_L',

$$Y_L[i] = Sin\alpha * X[i] + Cos\alpha * y[i] + \Delta X Sin\alpha + \Delta y Cos\alpha - \Delta y$$

delta_y

Note: Work on Virtual to physical mapping

Notation is introduced to Keep track if the new point is generated By CCW or CW

$\vec{P}_i^j$ ← Level
i ← Enumeration of the Points.

$\vec{P}_{i,l}^j$ or $\vec{P}_{i,r}^j$

(l – left – CCW,
r – Right – CW)

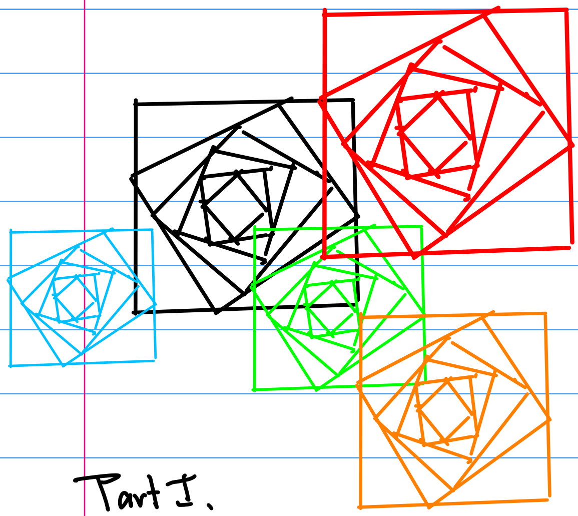
1st Project, 2D Graphics Engine

2D Graphics Screen Saver
a collection of Rotating Squares.

CmpE240

Sept. 28, 22

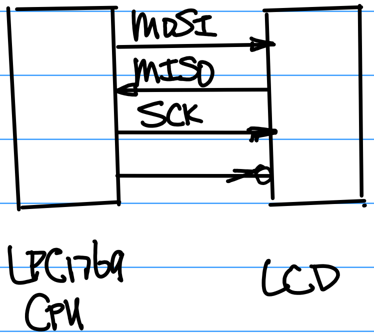
Interface Between the LPC1769 and LCD is By S.P.I (Serial Peripheral Interface)



Part I.

Fig.1

(Level 10 or higher)



Part II. Trees. Optional for Artistic Presentation.

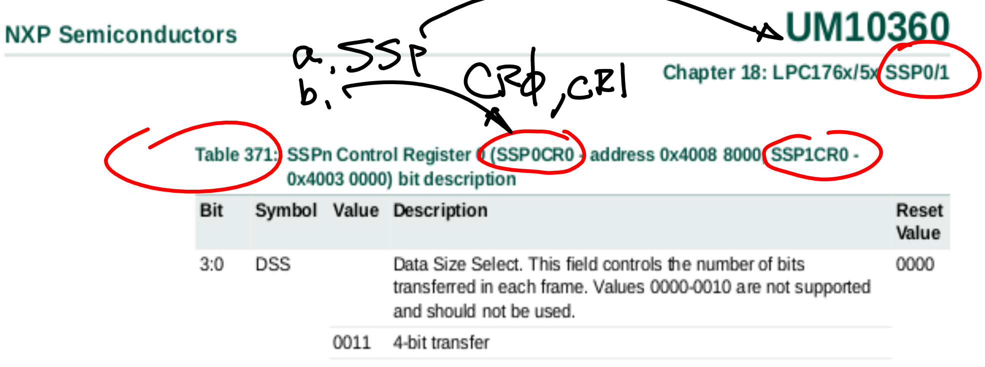
Fig.2

"3+1" pins for S.P.I.

1. MOSI Output
2. MISO Input
3. SCK (Clock) Output
4. SSEL (CS) Output

Now, consider LPC1769, CPU Datasheet pp431. CR0 2021F-107b-sch-#LPC...

Consider the Graphics Engine Design from System View.



a. SSP

b. CR0, CR1

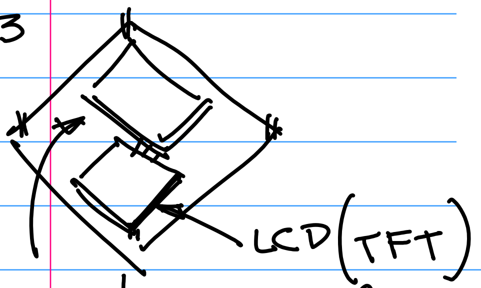
NXP Semiconductors **UM10360**

Chapter 18: LPC176x/5x SSP0/1

Table 371: SSPn Control Register 0 (SSP0CR0 - address 0x4008 8000) SSP1CR0 - 0x4003 0000) bit description

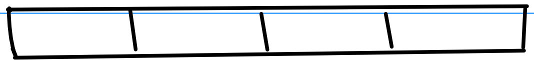
Bit	Symbol	Value	Description	Reset Value
3:0	DSS		Data Size Select. This field controls the number of bits transferred in each frame. Values 0000-0010 are not supported and should not be used.	0000
		0011	4-bit transfer	

Fig.3

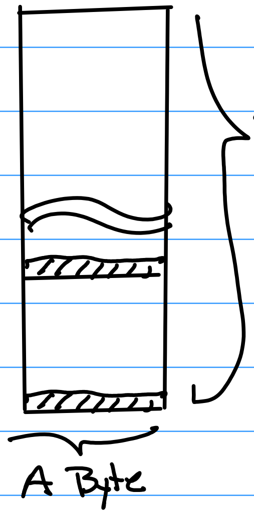


(SSP0CR0)

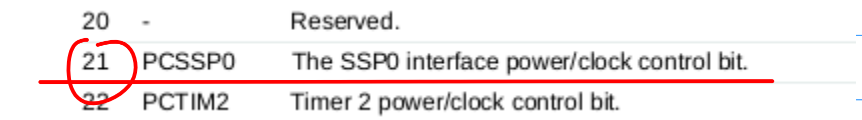
CR0



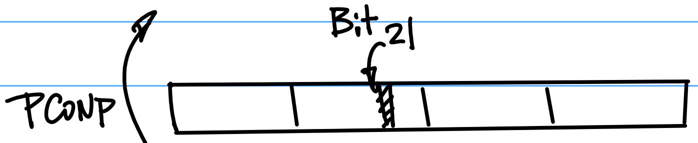
Addr. 0x4008_0000



Find its Memory Location at Which memory Bank?



20	-	Reserved.
21	PCSSP0	The SSP0 interface power/clock control bit.
22	PCTIM2	Timer 2 power/clock control bit.



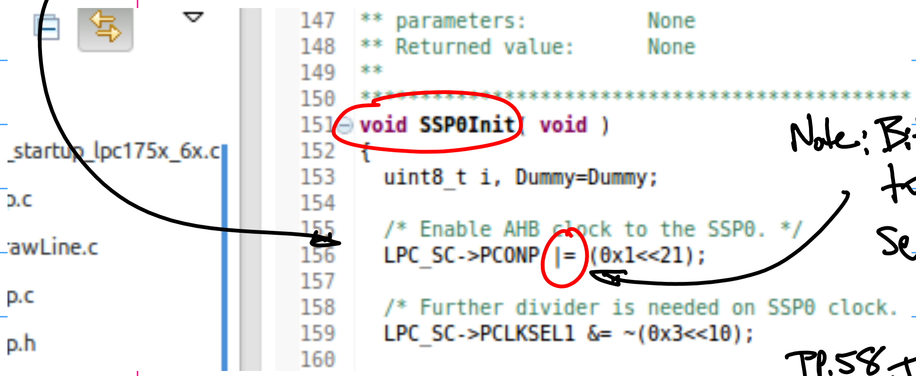
Oct.5 (Wed)

Example: SPR (for SSP/SPI) CR0

Note: 2nd SPR: PCLKSEL

a. PCONP, Naming Convention "LPC_SC->PCONP"

Source Code from DrawLine~ (github)



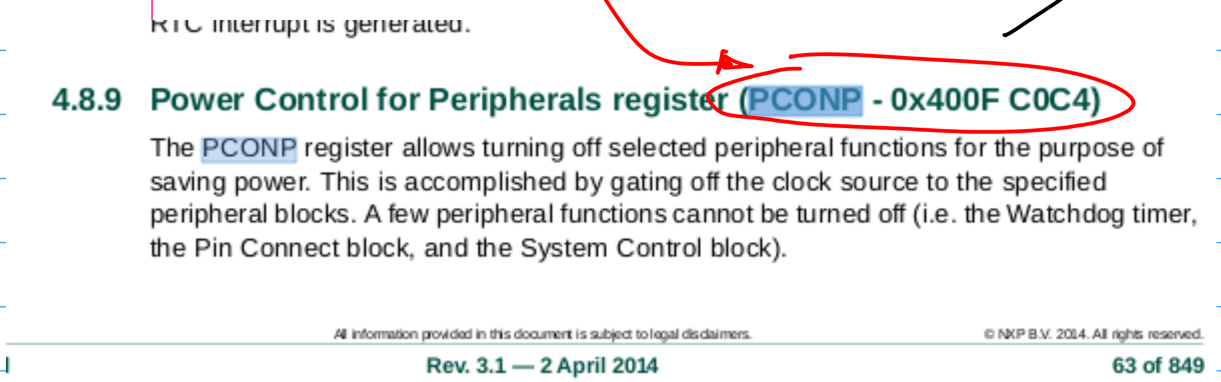
```
147 ** parameters:          None
148 ** Returned value:      None
149 **
150 *******************************************************
151 void SSP0Init( void )
152 {
153   uint8_t i, Dummy=Dummy;
154
155   /* Enable AHB clock to the SSP0. */
156   LPC_SC->PCONP |= (0x1<<21);
157
158   /* Further divider is needed on SSP0 clock. */
159   LPC_SC->PCLKSEL1 &= ~(0x3<<10);
160
```

Note: Bitwise op to just (only) Set Bit 21

PP.58, Table 41

b. CPU Datasheet, for PCONP.

C. Code (156)



RTC interrupt is generated.

4.8.9 Power Control for Peripherals register (PCONP - 0x400F C0C4)

The PCONP register allows turning off selected peripheral functions for the purpose of saving power. This is accomplished by gating off the clock source to the specified peripheral blocks. A few peripheral functions cannot be turned off (i.e. the Watchdog timer, the Pin Connect block, and the System Control block).

CmpE240
Oct.5th,2

22

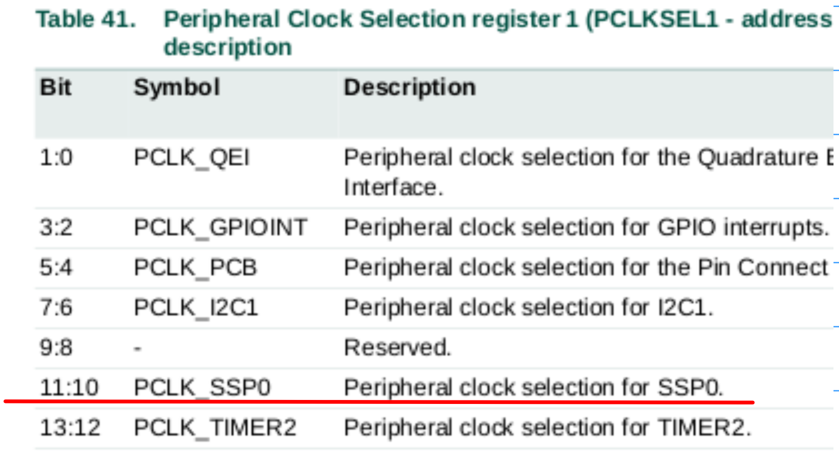
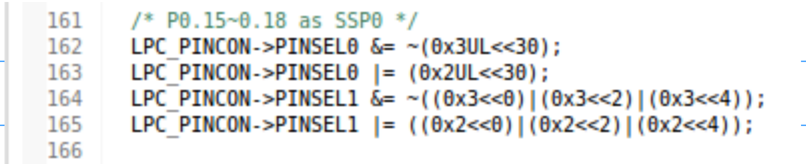


Table 41. Peripheral Clock Selection register 1 (PCLKSEL1 - address
description

Bit	Symbol	Description
1:0	PCLK_QEI	Peripheral clock selection for the Quadrature E Interface.
3:2	PCLK_GPIOINT	Peripheral clock selection for GPIO interrupts.
5:4	PCLK_PCB	Peripheral clock selection for the Pin Connect
7:6	PCLK_I2C1	Peripheral clock selection for I2C1.
9:8	-	Reserved.
11:10	PCLK_SSP0	Peripheral clock selection for SSP0.
13:12	PCLK_TIMER2	Peripheral clock selection for TIMER2.

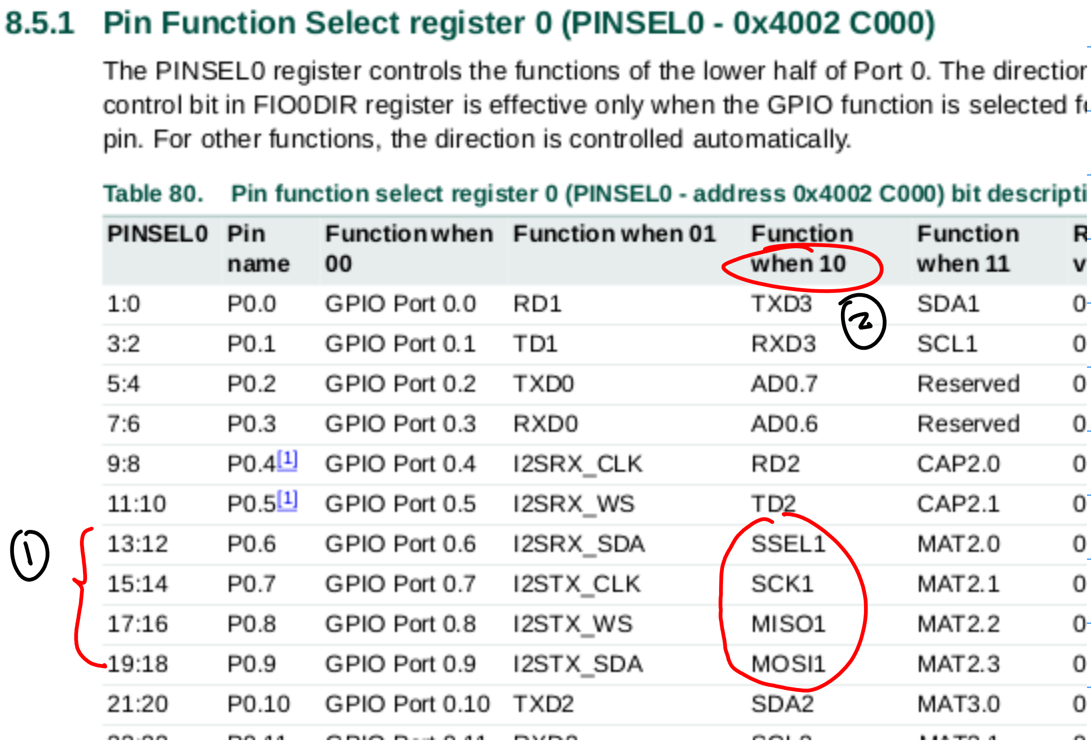
Note: PINSEL



```
/* P0.15~0.18 as SSP0 */
LPC_PINCON->PINSEL0 &= ~(0x3UL<<30);
LPC_PINCON->PINSEL0 |= (0x2UL<<30);
LPC_PINCON->PINSEL1 &= ~((0x3<<0)|(0x3<<2)|(0x3<<4));
LPC_PINCON->PINSEL1 |= ((0x2<<0)|(0x2<<2)|(0x2<<4));

```

pp117.



8.5.1 Pin Function Select register 0 (PINSEL0 - 0x4002 C000)

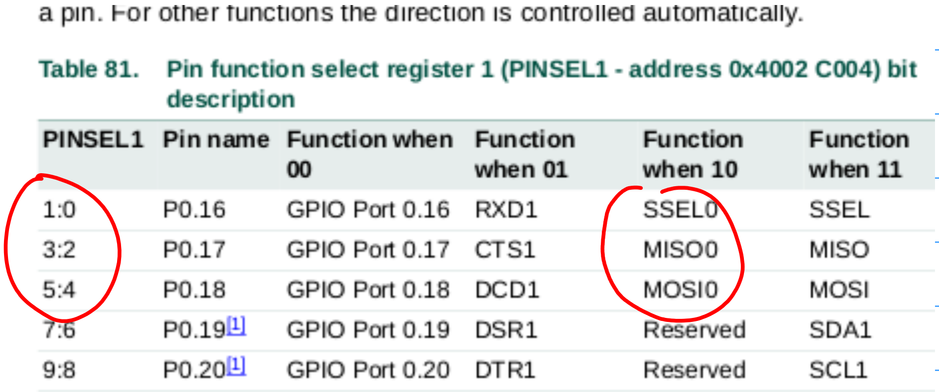
The PINSEL0 register controls the functions of the lower half of Port 0. The directior control bit in FIO0DIR register is effective only when the GPIO function is selected fo pin. For other functions, the direction is controlled automatically.

Table 80. Pin function select register 0 (PINSEL0 - address 0x4002 C000) bit descripti

PINSEL0	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	R v
1:0	P0.0	GPIO Port 0.0	RD1	TXD3	SDA1	0
3:2	P0.1	GPIO Port 0.1	TD1	RXD3	SCL1	0
5:4	P0.2	GPIO Port 0.2	TXD0	AD0.7	Reserved	0
7:6	P0.3	GPIO Port 0.3	RXD0	AD0.6	Reserved	0
9:8	P0.4[1]	GPIO Port 0.4	I2SRX_CLK	RD2	CAP2.0	0
11:10	P0.5[1]	GPIO Port 0.5	I2SRX_WS	TD2	CAP2.1	0
13:12	P0.6	GPIO Port 0.6	I2SRX_SDA	SSEL1	MAT2.0	0
15:14	P0.7	GPIO Port 0.7	I2STX_CLK	SCK1	MAT2.1	0
17:16	P0.8	GPIO Port 0.8	I2STX_WS	MISO1	MAT2.2	0
19:18	P0.9	GPIO Port 0.9	I2STX_SDA	MOSI1	MAT2.3	0
21:20	P0.10	GPIO Port 0.10	TXD2	SDA2	MAT3.0	0
23:22	P0.11	GPIO Port 0.11	RXD2	SCL2	MAT3.1	0

①

②

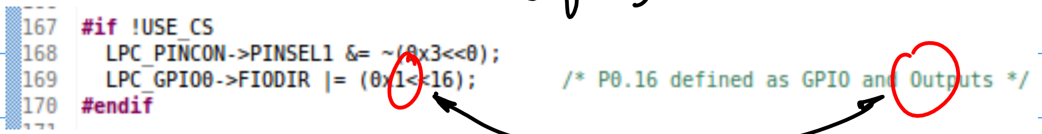


a pin. For other functions the direction is controlled automatically.

Table 81. Pin function select register 1 (PINSEL1 - address 0x4002 C004) bit description

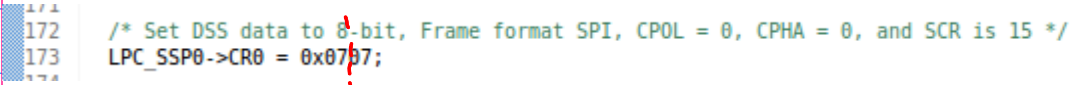
PINSEL1	Pin name	Function when 00	Function when 01	Function when 10	Function when 11
1:0	P0.16	GPIO Port 0.16	RXD1	SSEL0	SSEL
3:2	P0.17	GPIO Port 0.17	CTS1	MISO0	MISO
5:4	P0.18	GPIO Port 0.18	DCD1	MOSI0	MOSI
7:6	P0.19[1]	GPIO Port 0.19	DSR1	Reserved	SDA1
9:8	P0.20[1]	GPIO Port 0.20	DTR1	Reserved	SCL1

Note: FIODIR (Direction, e.g. Input/Output is defined)



```
#if !USE_CS
  LPC_PINCON->PINSEL1 &= ~(0x3<<0);
  LPC_GPIO0->FIODIR |= (0x1<<16);        /* P0.16 defined as GPIO and Outputs */
#endif
```

Note: a. Code → Tech. Spec. (Interpretation)



```
  /* Set DSS data to 8-bit, Frame format SPI, CPOL = 0, CPHA = 0, and SCR is 15 */
  LPC_SSP0->CR0 = 0x0707;
```

$$\underbrace{0000\,0111}_{SCR}\;\overbrace{00}^{SPI}00\;\underbrace{0111}_{8\ bit}$$

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

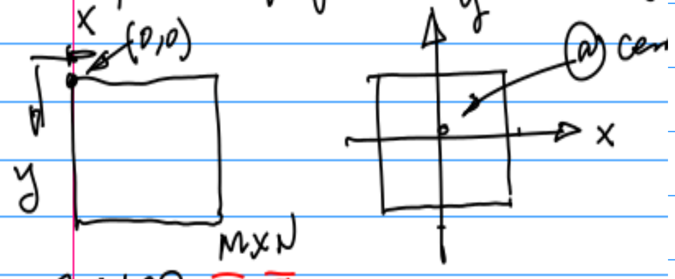
Consider Display Device Interface.

Background: Low Level Design of GE
{ Hardware Driver Design.
{ Physical Display v.s. Virtual Display.

Example: Physical Display v.s. Virtual Display.

pp19. from Ref.
(github. Notes ~105~)

Physical Display v.s. Virtual Display



SPI LCD
physical

Mapping Funct

M: No. of Row
N: Col. No. of.

Fig.1.

(x_v, y_v) for Virtual ~
(x_p, y_p) for Physical ~

F / 2021S-105-CMPE240-2021-03-22-Note.pdf

$$x_p = x_v + M/2 \quad \ldots(7)$$

$\left\{ \begin{array}{l} \underline{a} \text{ Direction} \\ \underline{b} \text{ Offset} \end{array} \right.$

$$y_p \overset{?}{=} -y_v + \frac{N}{2} \quad \ldots(8)$$

(half of the No. of Rows)

$$\begin{cases} x_p = x_v + M/2 & \ldots(7) \\ y_p = -y_v + N/2 & \ldots(8) \end{cases}$$