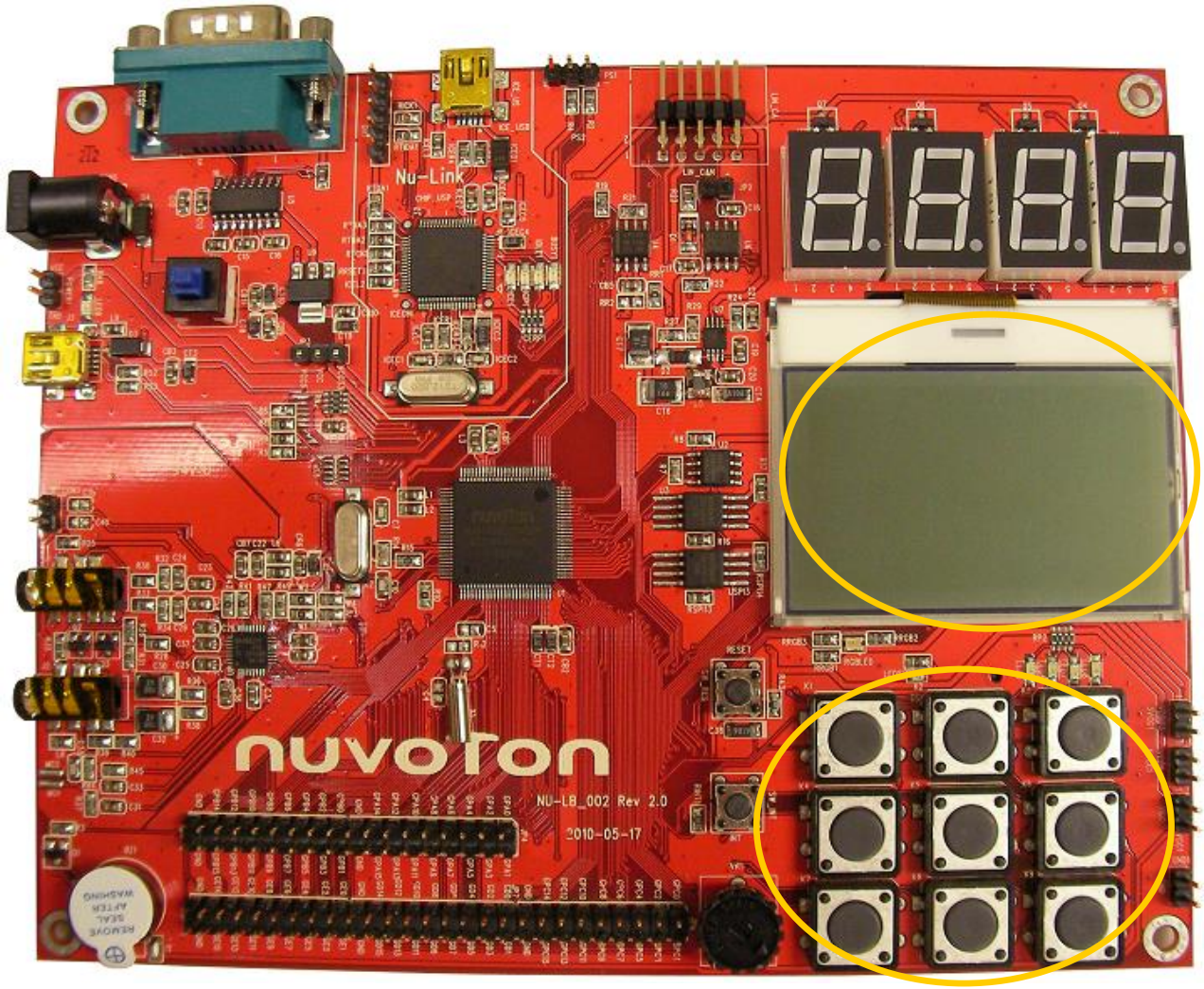
LCD液晶顯示

- 2013/4/9

課程大綱 (Lesson Outline)

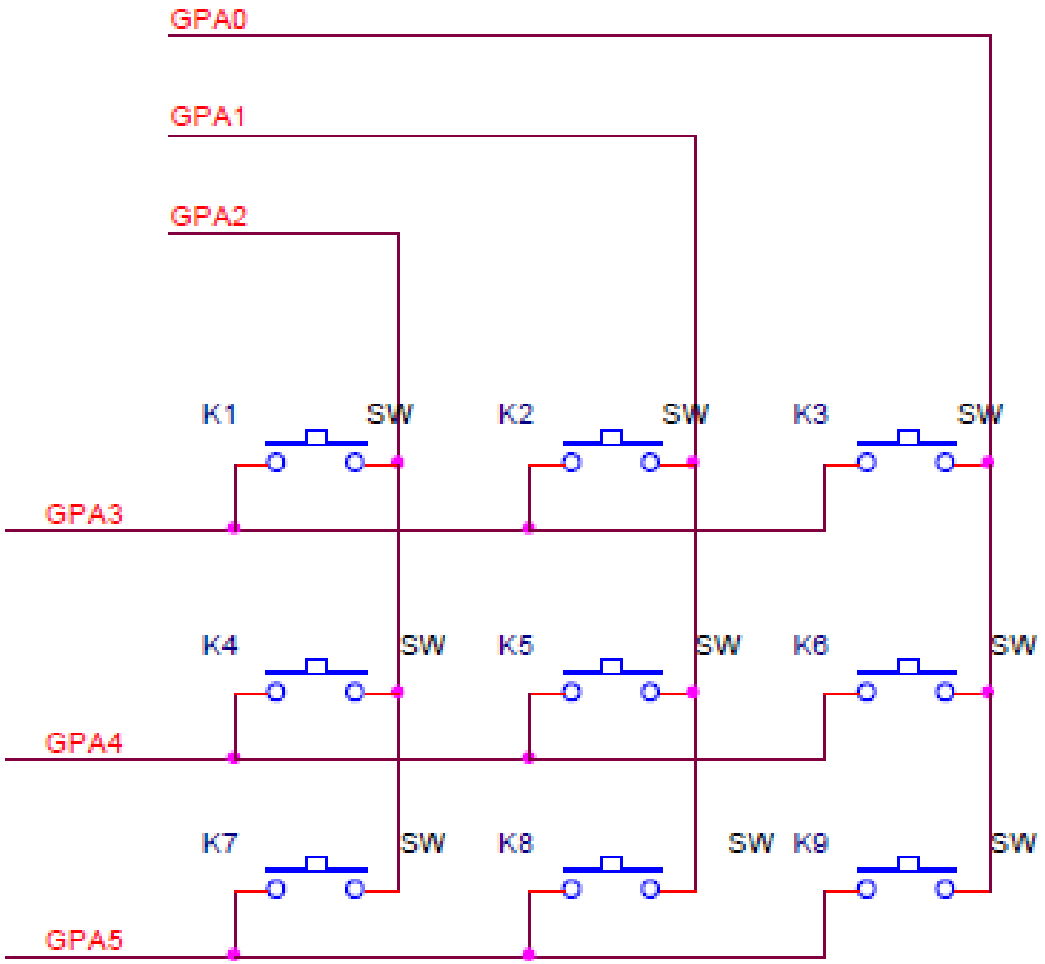
- LCD之基本輸出介紹
- 範例1 : LCD 顯示文字 (Smpl_LCD_TEXT)
- 範例2 : LCD 顯示keypad輸入 (Smpl_LCD_Keypad)
- 範例3 : LCD 顯示bitmap圖形 (Smpl_LCD_Bmp)
- 範例4 : LCD 顯示繪圖 (Smpl_LCD_Graphics)

Nu-LB-NUC140 開發板



學習板電路圖 Keypad circuit

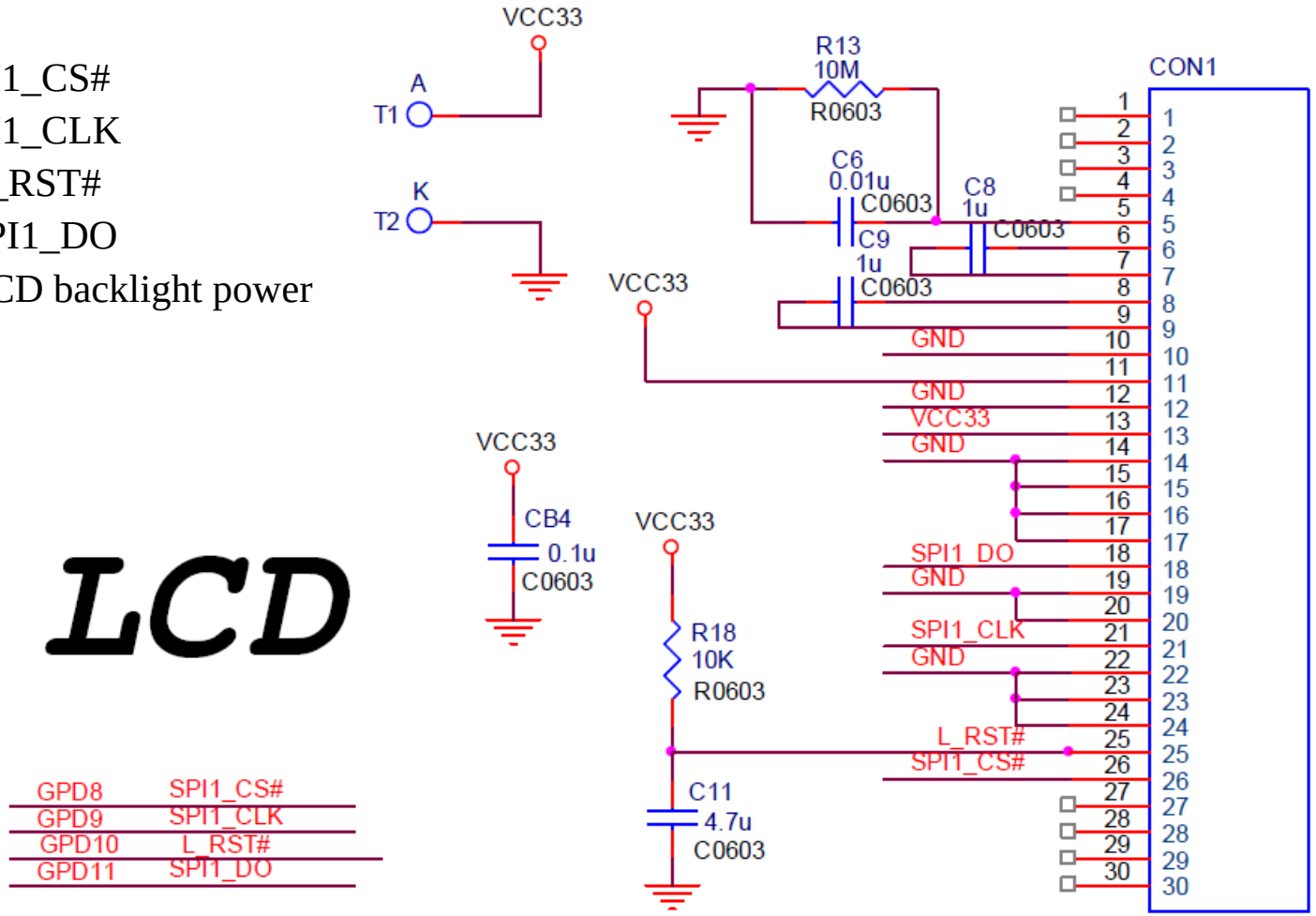
- SW Int: GPB15
  - 0=pressed,
  - 1= not pressed



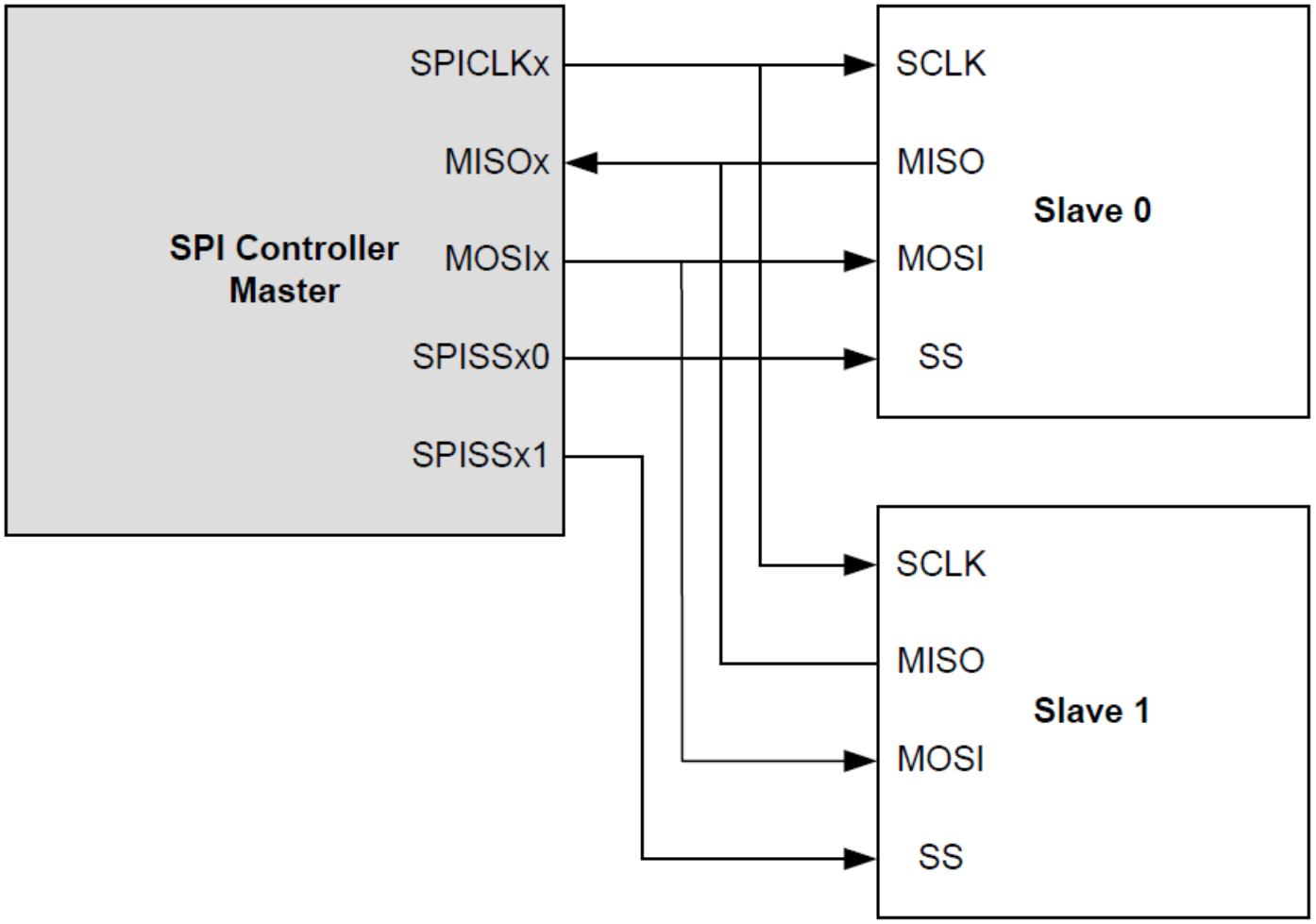
學習板電路圖 LCD circuit

SPI Interface

- GPD8 : SPI1_CS#
- GPD9 : SPI1_CLK
- GPD10 : L_RST#
- GPD11 : SPI1_DO
- GPD14 : LCD backlight power



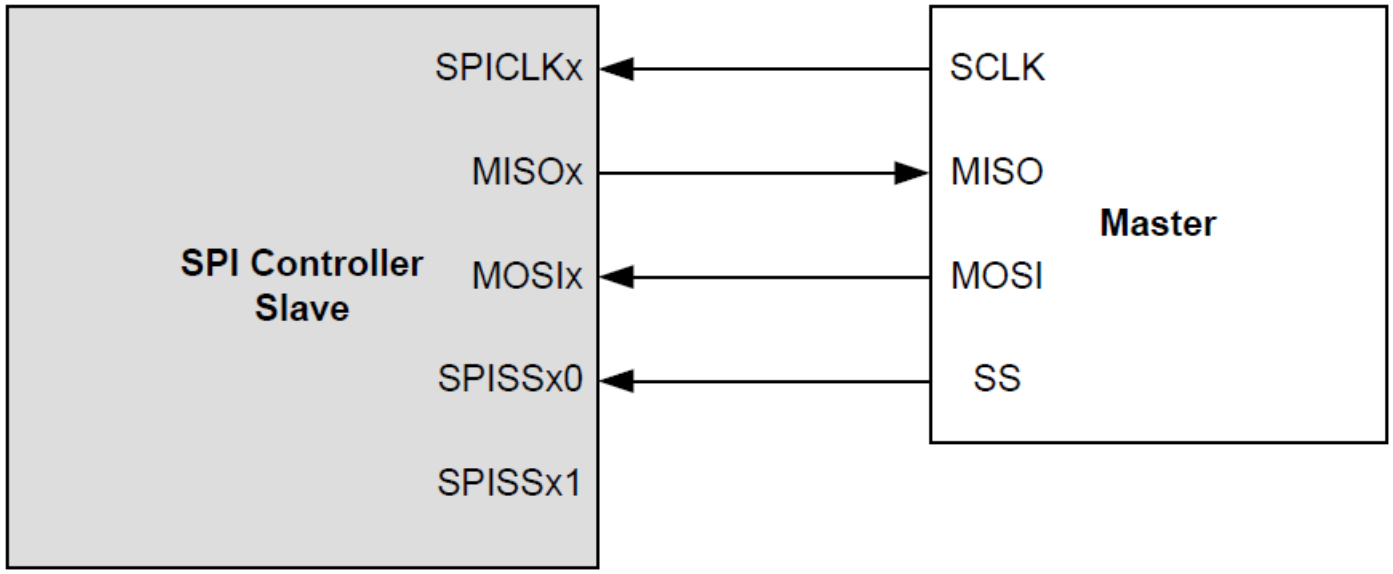
SPI 主機模式應用


SPICLKx
MISOx
SPI Controller
Master
MOSIx
SPISSx0
SPISSx1
SCLK
MISO
Slave 0
MOSI
SS
SCLK
MISO
Slave 1
MOSI
SS

SPI 主機模式應用

- 在主機模式下，SPI 控制器能通過從機選擇輸出腳 SPISSx0 與 SPISSx1 來選擇驅動兩個作為從機的外圍設備。
- 在主機/從機模式下，從機選擇信號的有效電平可以通過編程 SS_LVL 位 (SPI_SSR[2]) 來設定低有效或高有效，SS_LTRIG 位 (SPI_SSR[4]) 定義從機選擇信號 SPISSx0/1 為電平觸發或邊沿觸發。觸發條件的選擇取決於所連接的從機/主機設備的類型。
- 在主機模式下，如果置位AUTOSS (SPI_SSR[3])，將自動產生從機選擇信號

SPI 從機模式應用


SPICLKx
MISOx
MOSIx
SPISSx0
SPISSx1
SPI Controller
Slave
SCLK
MISO
MOSI
SS
Master

SPI 從機模式應用

- 在從機模式下，片外主機設備通過 SPISSx0 輸入端口驅動從機選擇信號到 SPI 控制器。
- 從機模式下，如果 SS_LTRIG 位被配置成電平觸發，則 LTRIG_FLAG 位 (SPI_SSR[5]) 用來表示事務完成後接收數目和接受位是否符合TX_NUM and TX_BIT_LEN 的設定要求。

LCD_Driver.c – Initial_panel() 1/x

```
void Initial_pannel(void)
{

    SYSCLK->APBCLK.SPI3_EN  =1;   //enable spi function
    SYS->IPRSTC2.SPI3_RST   =1;       //reset spi function
    SYS->IPRSTC2.SPI3_RST   =0;

    /* set GPIO to SPI mode*/
    SYS->GPDMFP.SPI3_SS0   =1;
    SYS->GPDMFP.SPI3_CLK  =1;
    //SYS->GPDMFP.SPI3_MISO0        =1;
    SYS->GPDMFP.SPI3_MOSI0        =1;
```

LCD_Driver.c – Initial_panel() 2/x

```
//CLKP HIGH IDLE
SPI_PORT[eDRVSPI_PORT3]->CNTRL.CLKP = 1;
//TX LEGTH 9
SPI_PORT[eDRVSPI_PORT3]->CNTRL.TX_BIT_LEN = 9;
//SET TX_NEG
SPI_PORT[eDRVSPI_PORT3]->CNTRL.TX_NEG = 1;
//SET DIV
SPI_PORT[eDRVSPI_PORT3]->DIVIDER.DIVIDER=0X03;
//ENABLE SLAVE SELECT
SPI_PORT[eDRVSPI_PORT3]->SSR.SSR=1;
```

LCD_Driver.c – Initial_panel() 3/x

```
// Set BR
SPI_PORT[eDRVSPI_PORT3]->TX[0] =0xEB;
SPI_PORT[eDRVSPI_PORT3]->CNTRL.GO_BUSY = 1;
while ( SPI_PORT[eDRVSPI_PORT3]->CNTRL.GO_BUSY == 1
);

// Set PM
SPI_PORT[eDRVSPI_PORT3]->SSR.SSR=0;
SPI_PORT[eDRVSPI_PORT3]->SSR.SSR=1;
```

LCD_Driver.c – Initial_panel() 4/x

```
//outp32(SPI3_Tx0, 0x81);
SPI_PORT[eDRVSPI_PORT3]->TX[0] =0x81;
SPI_PORT[eDRVSPI_PORT3]->CNTRL.GO_BUSY = 1;
while ( SPI_PORT[eDRVSPI_PORT3]->CNTRL.GO_BUSY == 1
);
SPI_PORT[eDRVSPI_PORT3]->TX[0] =0xa0;
SPI_PORT[eDRVSPI_PORT3]->CNTRL.GO_BUSY = 1;
while ( SPI_PORT[eDRVSPI_PORT3]->CNTRL.GO_BUSY == 1
);
SPI_PORT[eDRVSPI_PORT3]->SSR.SSR=0;
SPI_PORT[eDRVSPI_PORT3]->SSR.SSR=1;
```

LCD_Driver.c – Initial_panel() 5/x

```
    //outp32(SPI3_Tx0, 0xC0);
    SPI_PORT[eDRVSPI_PORT3]->TX[0] =0xc0;
    SPI_PORT[eDRVSPI_PORT3]->CNTRL.GO_BUSY = 1;
   while ( SPI_PORT[eDRVSPI_PORT3]->CNTRL.GO_BUSY == 1
    );

    // Set Display Enable
    SPI_PORT[eDRVSPI_PORT3]->SSR.SSR=0;
    SPI_PORT[eDRVSPI_PORT3]->SSR.SSR=1;
    SPI_PORT[eDRVSPI_PORT3]->TX[0] = 0XAF;
    SPI_PORT[eDRVSPI_PORT3]->CNTRL.GO_BUSY = 1;
   while ( SPI_PORT[eDRVSPI_PORT3]->CNTRL.GO_BUSY == 1
    );
    SPI_PORT[eDRVSPI_PORT3]->SSR.SSR=0;
}
```

LCD_Driver.c – WriteData() 1/x

```
void WriteData(unsigned char data)
{
    // Write Data
    SPI_PORT[eDRVSPI_PORT3]->SSR.SSR=1;              //chip select
    SPI_PORT[eDRVSPI_PORT3]->TX[0] =0x100 | data;
          //write data
    SPI_PORT[eDRVSPI_PORT3]->CNTRL.GO_BUSY = 1;
   while ( SPI_PORT[eDRVSPI_PORT3]->CNTRL.GO_BUSY == 1
    ); //check data out?
    SPI_PORT[eDRVSPI_PORT3]->SSR.SSR=0;
}
```

LCD_Driver.c – SetPACA() 1/x

```
void SetPACA(unsigned char PA, unsigned char CA)
{
    // Set PA

    SPI_PORT[eDRVSPI_PORT3]->SSR.SSR=1;
    SPI_PORT[eDRVSPI_PORT3]->TX[0] = 0xB0 | PA;
    SPI_PORT[eDRVSPI_PORT3]->CNTRL.GO_BUSY = 1;
   while ( SPI_PORT[eDRVSPI_PORT3]->CNTRL.GO_BUSY == 1
    );      //check data out?
    // Set CA MSB
```

LCD_Driver.c – SetPACA() 2/x

```
    // Set CA MSB
    SPI_PORT[eDRVSPI_PORT3]->SSR.SSR=0;
    SPI_PORT[eDRVSPI_PORT3]->SSR.SSR=1;
    SPI_PORT[eDRVSPI_PORT3]->TX[0] =0x10 |(CA>>4)&0xF;
    SPI_PORT[eDRVSPI_PORT3]->CNTRL.GO_BUSY = 1;
   while ( SPI_PORT[eDRVSPI_PORT3]->CNTRL.GO_BUSY == 1 );
            //check data out?
    // Set CA LSB
    SPI_PORT[eDRVSPI_PORT3]->SSR.SSR=0;
    SPI_PORT[eDRVSPI_PORT3]->SSR.SSR=1;
    SPI_PORT[eDRVSPI_PORT3]->TX[0] =0x00 | (CA & 0xF);
    SPI_PORT[eDRVSPI_PORT3]->CNTRL.GO_BUSY = 1;
   while ( SPI_PORT[eDRVSPI_PORT3]->CNTRL.GO_BUSY == 1 );
            //check data out?
    SPI_PORT[eDRVSPI_PORT3]->SSR.SSR=0;
}
```

LCD_Driver.c – Show_Word() 1/x

```
void Show_Word(unsigned char x, unsigned char y,unsigned char
    ascii_word)
{
    int i=0,k=0;
    unsigned char temp;

      k=(ascii_word-32)*16;

      for(i=0;i<8;i++)
      {
                    SetPACA((x*2),(129-(y*8)-i));
                    temp=Ascii[k+i];
     WriteData(temp);
      }
```

LCD_Driver.c – Show_Word() 2/x

```
 for(i=0;i<8;i++)
       {
                        SetPACA((x*2)+1,(129-(y*8)-i));
      temp=Ascii[k+i+8];
                        WriteData(temp);
       }
}
```

LCD_Driver.c – print_lcd() 1/x

```
void print_lcd(unsigned char line, char *str)
{
    int i=0;
  do{
    Show_Word(line,i,*str++);
    i++;
    if(i>15)  break;
  }  while(*str!='\0');
}
```

LCD_Driver.c – clr_all_pannal() 1/x

```
void clr_all_pannal(void)
{
    int i=0;

    /*CLEAR ALL PANNAL*/
    SetPACA(0x0, 0x0);

    for (i = 0; i < 132 *8; i++)
    {
        WriteData(0x00);
    }
    WriteData(0x0f);
}
```

LCD_Driver.c – draw_LCD() 1/x

```
void draw_LCD(unsigned char *buffer)
{
    int X_max = 64;
    int Y_max =128;
    int x=0;
    int y=0;
    for (y=0; y< Y_max; y++)
    {
        for (x=0; x< (X_max/8); x++)
        {
            SetPACA(x,(129-y));
            WriteData(buffer[y*8+x]);
        }
    }
}
```

Ascii_Table.c

‘0’=0x30
‘1’=0x31
‘2’=0x32
‘3’=0x33
‘4’=0x34
‘5’=0x35
‘6’=0x36
‘7’=0x37
‘8’=0x38
‘9’=0x39
‘A’=0x41
‘Z’=0x5A
‘a’=0x61
‘z’=0x7A

表2-14 七位元的ASCII碼

列	行 0	1	2	3	4	5	6	7
	BITS 4321 765	000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P	、	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	”	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
10	1010	LT	SUB	*	:	J	Z	j	z
11	1011	VT	ESC	+	;	K	〔	k	{
12	1100	FF	FS	,	<	L	\	l	|
13	1101	CR	GS	-	=	M	〕	m	}
14	1110	SO	RS	.	>	N	(↑)	n	~
15	1111	SI	US	/	?	O	←(←)	o	DEL

7-bit ASIC Code

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Ascii_Table.c

```
unsigned char Ascii[]={
/*   */
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
/* ! */
0xF0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xCF,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
/* " */
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xF0,0x00,0x00,0xF0,0x00,0x00,0x00,0x00,
/* # */
0x00,0x00,0x00,0xC0,0x30,0xC0,0x30,0x00,0xC8,0x39,0xCF,0x39,0x0F,0x09,0x01,0x00,
/* $ */
0xE0,0x10,0x10,0xF8,0x10,0x10,0x60,0x00,0x30,0x41,0x42,0xFF,0x42,0x44,0x38,0x00,
/* % */
0xC0,0x20,0x10,0xE0,0x20,0xE0,0x18,0x00,0x01,0xC2,0x31,0x0C,0x73,0x88,0x44,0x38,
/* & */
0x00,0xE0,0x10,0x10,0xE0,0x00,0x00,0x00,0x78,0x85,0x82,0x8D,0x70,0xA2,0x9E,0x42,
/* ' */
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xF0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
```

LCD1: Smpl_LCD_TEXT

LCD可以顯示4列，每列16字元

使用函數print_lcd將字串顯示在LCD的指定列

範例： print_lcd(0, "1st line--------");

將字串顯示在第0列

LCD1: Smpl_LCD_TEXT

```
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "NUC1xx-LB_002\LCD_Driver.h“
int main(void)
{
    Initial_pannel();  //Initial LCD,968clock
    clr_all_pannal(); //clear LCD ,192911clock
    print_lcd(0, "1st line--------");  // show on 1st line,192650clock
    print_lcd(1, "2nd line--------"); // show on 2nd line
    print_lcd(2, "3th line--------");  // show on 3th line
    print_lcd(3, "4th line--------"); // show on 4th line
}
```

LCD2: Smpl_LCD_KeyPad

LCD可以顯示4列，每列16字元

使用函數print_lcd將字串顯示在LCD的指定列

使用函數輸入KeyPad按鍵的輸入

LCD2:Smpl_LCD_Keypad 1/x

```
#include "NUC1xx.h"
#include "Seven_Segment.h"
#include "ScanKey.h"
#include "LCD_Driver.h"
#include "Driver/DrvGPIO.h“
int32_t main (void)
{
        int8_t number;
        char text;
        Initial_pannel(); //977clock
        clr_all_pannal(); //192914clock
```

LCD2:Smpl_LCD_Keypad 2/x

```
        OpenKeyPad(); //505clock
        while(1)
        {
                do{
                        number = Scankey();
                }while (!number);
                close_seven_segment(); //200clock
                show_seven_segment(0,number); //573clock
                text=bin2ascii(number); //28clock
                print_lcd(0, &text);        //11442clock
        }
}
```

LCD2:Smpl_LCD_Keypad 3/x

```
char bin2ascii(uint8_t number)
{
        if(number <10)
                return number+'0';
        else
                return number-10+'A';
}
```

LCD3: Smpl_LCD_Bmp

將bitmap顯示在LCD

1.使用小畫家產生bitmap檔案

1. 設定圖片大小(像素)：寬度128，高度64；色彩：黑白
2. 存檔類型：單色點陣圖
3. 檔案名稱：

2. 使用bmp2asm將bitmap轉換成HEX檔

執行bmp2asm.exe

選取目錄，選取檔案，點選[Convert]

將8行的HEX碼複製到宣告的陣列

3. 編輯HEX碼，置入C陣列作為顯示緩衝區

LCD3: Smpl_LCD_Bmp 1/x

```
#include "NUC1xx.h"
#include "LCD_Driver.h"
#include "Driver/DrvGPIO.h"

unsigned char DisplayBuf [8*128];
int32_t main (void)
{
        uint8_t x, y;
        unsigned char Nuvoton[128*8] = {
        ...
};
```

字型資料採用小畫家製做128x64.bmp, 再由bmp2asm轉出hex貼到smp_LCD_Bmp.c

LCD3: Smpl_LCD_Bmp 2/x

```
        Initial_pannel();//978clock
        clr_all_pannal(); //192909clock
        for (y=0; y<128; y++)
        {
                for (x=0; x<8; x++)
                {   DisplayBuf[y*8+x] = Nuvoton[y+x*128];   }
        }
        draw_LCD(DisplayBuf);  // render to LCD,719541clock
}
```

字型資料採用小畫家製做128x64.bmp, 再由bmp2asm轉出hex貼到smp_LCD_Bmp.c

LCD4: Smpl_LCD_Graphics

在LCD顯示圖形

繪線 LineBresenham(int x1, int y1, int x2, int y2, int color)

繪線 LineOptimized(int x1, int y1, int x2, int y2, int color)

繪圓 CircleBresenham(int xc, int yc, int r, int color)

繪圓 CircleMidpoint(int xc, int yc, int r, int color)

繪圓 CircleOptimized(int xc, int yc, int r, int color)

繪四方形 RectangleDraw(int x0, int y0, int x1, int y1, int color)

填滿四方形 RectangleFill(int x0, int y0, int x1, int y1, int color)

繪三角形 Triangle (int x0, int y0, int x1, int y1, int x2, int y2, int color)

畫點函式 (in 2D_Graphic_Driver.c) 1/x

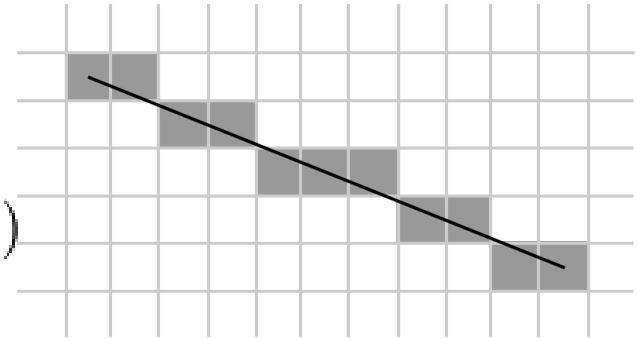
```
// draw pixel into Display Buffer
void draw_pixel(int x, int y, int color)
{
    if (color==0) {
        switch (x%8) {
        case 0: DisplayBuf[(x/8)+y*8] &= 0xFE;  break;
        case 1: DisplayBuf[(x/8)+y*8] &= 0xFD;  break;
        case 2: DisplayBuf[(x/8)+y*8] &= 0xFB;  break;
        case 3: DisplayBuf[(x/8)+y*8] &= 0xF7;  break;
        case 4: DisplayBuf[(x/8)+y*8] &= 0xEF;  break;
        case 5: DisplayBuf[(x/8)+y*8] &= 0xDF;  break;
        case 6: DisplayBuf[(x/8)+y*8] &= 0xBF;  break;
        case 7: DisplayBuf[(x/8)+y*8] &= 0x7F;  break;
        default: break; }
        }
```

畫點函式 (in 2D_Graphic_Driver.c) 2/x

```
    else if (color==1) {
            switch (x%8) {
            case 0: DisplayBuf[(x/8)+y*8] |= 0x01;  break;
            case 1: DisplayBuf[(x/8)+y*8] |= 0x02;  break;
            case 2: DisplayBuf[(x/8)+y*8] |= 0x04;  break;
            case 3: DisplayBuf[(x/8)+y*8] |= 0x08;  break;
            case 4: DisplayBuf[(x/8)+y*8] |= 0x10;  break;
            case 5: DisplayBuf[(x/8)+y*8] |= 0x20;  break;
            case 6: DisplayBuf[(x/8)+y*8] |= 0x40;  break;
            case 7: DisplayBuf[(x/8)+y*8] |= 0x80;  break;
            default: break;   }
            }
}
```

Bresenham Line-Drawing Algorithm

- The Bresenham Line-Drawing Algorithm
- 布雷森漢姆直線演算法
- Line equation : $y - y_0 = \frac{y_1 - y_0}{x_1 - x_0}(x - x_0)$



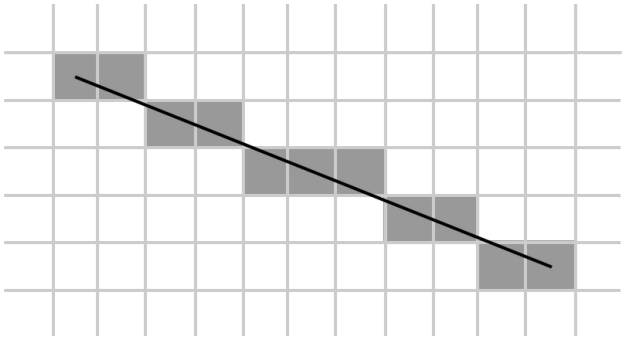
  - so, $y = \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + y_0$

- 因為x及y皆為整數，但並非每一點x所對應的y皆為整數，故此沒有必要去計算每一點x所對應之y值。反之由於此線之斜率介乎於1至0之間，故此我們只需要找出當x到達那一個數值時，會使y上升1，若x尚未到此值，則y不變。
- 至於如何找出相關的x值，則需依靠**斜率**

  由於此值不變，故可於運算前預先計算，減少運算次數。

Bresenham Line-Drawing Algorithm

- The Bresenham Line-Drawing Algorithm
- 布雷森漢姆直線演算法
- Line equation : $y - y_0 = \frac{y_1 - y_0}{x_1 - x_0}(x - x_0)$



  - so, $y = \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + y_0$

- 要實行此演算法，我們需計算每一像素點與該線之間的誤差。於上述例子中，誤差應為每一點x中，其相對的像素點之y值與該線實際之y值的差距。每當x的值增加1，誤差的值就會增加m。每當誤差的值超出0.5，線就會比較靠近下一個映像點，因此y的值便會加1，且誤差減1。

LineBresenham() – 畫斜線函式 1/x

```
void LineBresenham(int x1, int y1, int x2, int y2, int color)
{
    int dy = y2 - y1;
    int dx = x2 - x1;
    int stepx, stepy;

    if (dy < 0) { dy = -dy;  stepy = -1; } else { stepy = 1; }
    if (dx < 0) { dx = -dx;  stepx = -1; } else { stepx = 1; }
    dy <<= 1;        // dy is now 2*dy
    dx <<= 1;        // dx is now 2*dx

    draw_pixel(x1,y1, color);
    if (dx > dy)
    {
```

LineBresenham() – 畫斜線函式 2/x

```
int fraction = dy - (dx >> 1);  // same as 2*dy - dx
        while (x1 != x2)
        {
          if (fraction >= 0)
          {
             y1 += stepy;
             fraction -= dx;          // same as fraction -= 2*dx
          }
          x1 += stepx;
          fraction += dy;              // same as fraction -= 2*dy
          draw_pixel(x1, y1, color);
        }
```

http://willperone.net/Code/codeline.php

LineBresenham() – 畫斜線函式 3/x

```
} else {
        int fraction = dx - (dy >> 1);
        while (y1 != y2) {
          if (fraction >= 0) {
              x1 += stepx;
              fraction -= dy;
          }
          y1 += stepy;
          fraction += dx;
          draw_pixel(x1, y1, color);
        }
    }
}
```

CircleBresenham() – 畫圓函式 1/x

```
void CircleBresenham(int xc, int yc, int r, int color)
{
    int x = 0;
    int y = r;
    int p = 3 - 2 * r;
    if (!r) return;
    while (y >= x) // only formulate 1/8 of circle
    {
        draw_pixel(xc-x, yc-y, color);//upper left left
        draw_pixel(xc-y, yc-x, color);//upper upper left
        draw_pixel(xc+y, yc-x, color);//upper upper right
        draw_pixel(xc+x, yc-y, color);//upper right right
```

http://willperone.net/Code/codecircle.php

CircleBresenham() – 畫圓函式 2/x

```
draw_pixel(xc-x, yc+y, color);//lower left left
        draw_pixel(xc-y, yc+x, color);//lower lower left
        draw_pixel(xc+y, yc+x, color);//lower lower right
        draw_pixel(xc+x, yc+y, color);//lower right right
        if (p < 0) p += 4*x++ + 6;
            else p += 4*(x++ - y--) + 10;
    }
}
```

http://willperone.net/Code/codecircle.php

範例: Smp_LCD_Graphics (繪圖:斜線+圓圈)

```
int32_t main (void)
{
    int color = 1;
     Initial_panel();
     clr_all_panel();
     color = 1;
     // draw rectangle
     RectangleDraw(0,0,63,127, color);
     // draw box
     RectangleFill(10,10,20,20, color);
```

範例: Smp_LCD_Graphics (繪圖:斜線+圓圈)

```
    // draw circle
    CircleBresenham(30,22,20,color);
    CircleMidpoint(30,63,20,color);
    CircleOptimized(30,105,20,color);
    LineBresenham(0,20,50,60,color);
    LineOptimized(20,30,40,100,color);
    // draw triangle
    Triangle(30,90,60,30,50,60,color);

    // render to LCD
    draw_LCD(DisplayBuf);
}
```

General Disclaimer

The Lecture is strictly used for educational purpose.

MAKES NO GUARANTEE OF VALIDITY

- **The lecture cannot guarantee the validity of the information found here.** The lecture may recently have been changed, vandalized or altered by someone whose opinion does not correspond with the state of knowledge in the relevant fields. Note that most other encyclopedias and reference works also have similar disclaimers.

No formal peer review

- The lecture is not uniformly peer reviewed; while readers may correct errors or engage in casual peer review, they have no legal duty to do so and thus all information read here is without any implied warranty of fitness for any purpose or use whatsoever. Even articles that have been vetted by informal peer review or *featured article* processes may later have been edited inappropriately, just before you view them.

No contract; limited license

- Please make sure that you understand that the information provided here is being provided freely, and that no kind of agreement or contract is created between you and the owners or users of this site, the owners of the servers upon which it is housed, the individual Wikipedia contributors, any project administrators, sysops or anyone else who is in *any way connected* with this project or sister projects subject to your claims against them directly. You are being granted a limited license to copy anything from this site; it does not create or imply any contractual or extracontractual liability on the part of Wikipedia or any of its agents, members, organizers or other users.
- There is **no agreement or understanding between you and the content provider** regarding your use or modification of this information beyond the Creative Commons Attribution-Sharealike 3.0 Unported License (CC-BY-SA) and the GNU Free Documentation License (GFDL);

General Disclaimer

Trademarks

- Any of the trademarks, service marks, collective marks, design rights or similar rights that are mentioned, used or cited in the lectures are the property of their respective owners. Their use here does not imply that you may use them for any purpose other than for the same or a similar informational use as contemplated by the original authors under the CC-BY-SA and GFDL licensing schemes. Unless otherwise stated , we are neither endorsed by nor affiliated with any of the holders of any such rights and as such we cannot grant any rights to use any otherwise protected materials. Your use of any such or similar incorporeal property is at your own risk.

Personality rights

- The lecture may portray an identifiable person who is alive or deceased recently. The use of images of living or recently deceased individuals is, in some jurisdictions, restricted by laws pertaining to personality rights, independent from their copyright status. Before using these types of content, please ensure that you have the right to use it under the laws which apply in the circumstances of your intended use. *You are solely responsible for ensuring that you do not infringe someone else's personality rights.*