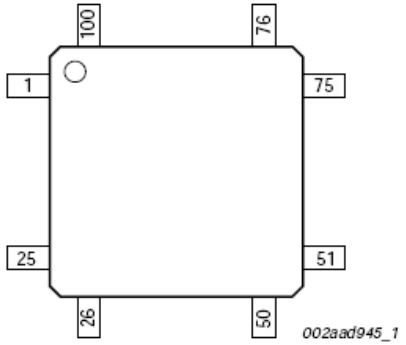
Embedded Modules and C programmig

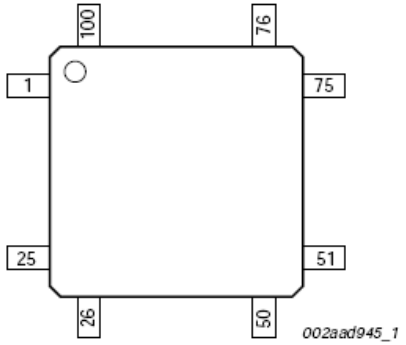
LPC 1768 PIN CONFIGURATION



PX.Y – Port X Pin Y

P0[0] to P0[31]	I/O	**Port 0:** Port 0 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 0 pins depends upon the pin function selected via the pin connect block. Pins 12, 13, 14, and 31 of this port are not available.
P1[0] to P1[31]	I/O	**Port 1:** Port 1 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 1 pins depends upon the pin function selected via the pin connect block. Pins 2, 3, 5, 6, 7, 11, 12, and 13 of this port are not available.
P2[0] to P2[31]	I/O	**Port 2:** Port 2 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 2 pins depends upon the pin function selected via the pin connect block. Pins 14 through 31 of this port are not available.
P3[0] to P3[31]	I/O	**Port 3:** Port 3 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 3 pins depends upon the pin function selected via the pin connect block. Pins 0 through 24, and 27 through 31 of this port are not available.
P4[0] to P4[31]	I/O	**Port 4:** Port 4 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 4 pins depends upon the pin function selected via the pin connect block. Pins 0 through 27, 30, and 31 of this port are not available.

LPC 1768 PIN CONFIGURATION



PX.Y – Port X Pin Y

P0[0] to P0[31]	I/O	**Port 0:** Port 0 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 0 pins depends upon the pin function selected via the pin connect block. Pins 12, 13, 14, and 31 of this port are not available.
P1[0] to P1[31]	I/O	**Port 1:** Port 1 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 1 pins depends upon the pin function selected via the pin connect block. Pins 2, 3, 5, 6, 7, 11, 12, and 13 of this port are not available.
P2[0] to P2[31]	I/O	**Port 2:** Port 2 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 2 pins depends upon the pin function selected via the pin connect block. Pins 14 through 31 of this port are not available.
P3[0] to P3[31]	I/O	**Port 3:** Port 3 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 3 pins depends upon the pin function selected via the pin connect block. Pins 0 through 24, and 27 through 31 of this port are not available.
P4[0] to P4[31]	I/O	**Port 4:** Port 4 is a 32-bit I/O port with individual direction controls for each bit. The operation of port 4 pins depends upon the pin function selected via the pin connect block. Pins 0 through 27, 30, and 31 of this port are not available.

PIN CONNECT BLOCK

Register	Controls
PINSEL0	P0[15:0]
PINSEL1	P0 [31:16]
PINSEL2	P1 [15:0] (Ethernet)
PINSEL3	P1 [31:16]
PINSEL4	P2 [15:0]
PINSEL5	P2 [31:16]
PINSEL6	P3 [15:0]
PINSEL7	P3 [31:16]
PINSEL8	P4 [15:0]
PINSEL9	P4 [31:16]

PINSEL0 to PINSEL9 Values	Function
00	Primary (default) function, typically GPIO port
01	First alternate function
10	Second alternate function
11	Third alternate function

PIN CONNECT BLOCK

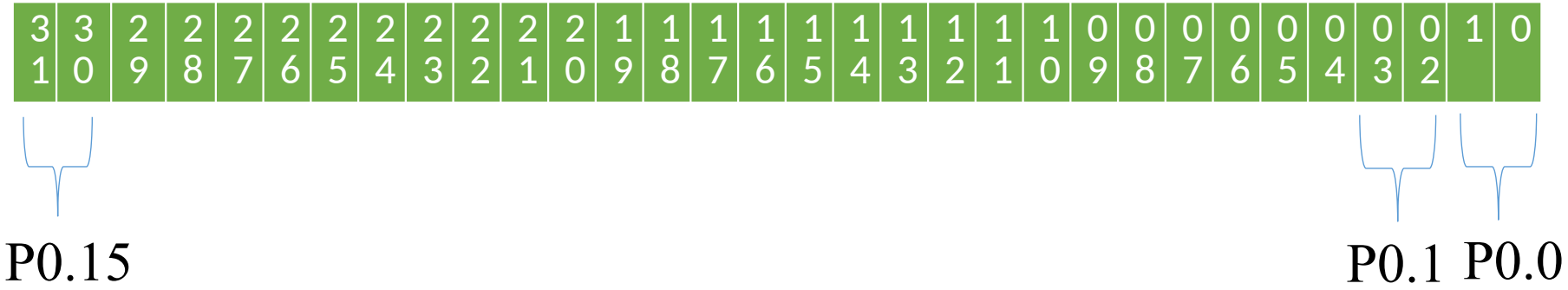
PINSEL0	Pin name	Function when 00	Function when 01	Function when 10	Function when 11
1:0	P0.0	GPIO Port 0.0	RD1	TXD3	SDA1
3:2	P0.1	GPIO Port 0.1	TD1	RXD3	SCL1
5:4	P0.2	GPIO Port 0.2	TXD0	AD0.7	Reserved
7:6	P0.3	GPIO Port 0.3	RXD0	AD0.6	Reserved
9:8	P0.4[1]	GPIO Port 0.4	I2SRX_CLK	RD2	CAP2.0
11:10	P0.5[1]	GPIO Port 0.5	I2SRX_WS	TD2	CAP2.1
13:12	P0.6	GPIO Port 0.6	I2SRX_SDA	SSEL1	MAT2.0
15:14	P0.7	GPIO Port 0.7	I2STX_CLK	SCK1	MAT2.1
17:16	P0.8	GPIO Port 0.8	I2STX_WS	MISO1	MAT2.2
19:18	P0.9	GPIO Port 0.9	I2STX_SDA	MOSI1	MAT2.3
21:20	P0.10	GPIO Port 0.10	TXD2	SDA2	MAT3.0
23:22	P0.11	GPIO Port 0.11	RXD2	SCL2	MAT3.1
29:24	-	Reserved	Reserved	Reserved	Reserved
31:30	P0.15	GPIO Port 0.15	TXD1	SCK0	SCK

PIN CONNECT BLOCK

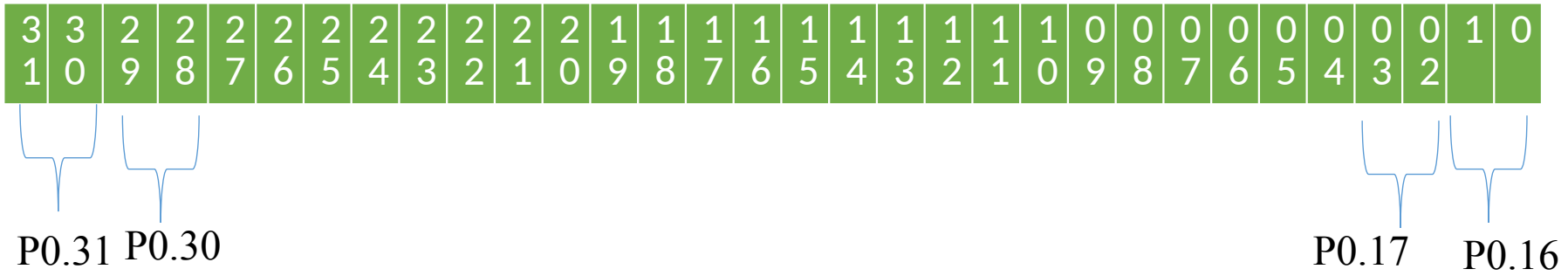
PINSEL1	Pin name	Function when 00	Function when 01	Function when 10	Function when 11
1:0	P0.16	GPIO Port 0.16	RXD1	SSEL0	SSEL
3:2	P0.17	GPIO Port 0.17	CTS1	MISO0	MISO
5:4	P0.18	GPIO Port 0.18	DCD1	MOSI0	MOSI
7:6	P0.19[1]	GPIO Port 0.19	DSR1	Reserved	SDA1
9:8	P0.20[1]	GPIO Port 0.20	DTR1	Reserved	SCL1
11:10	P0.21[1]	GPIO Port 0.21	RI1	Reserved	RD1
13:12	P0.22	GPIO Port 0.22	RTS1	Reserved	TD1
15:14	P0.23[1]	GPIO Port 0.23	AD0.0	I2SRX_CLK	CAP3.0
17:16	P0.24[1]	GPIO Port 0.24	AD0.1	I2SRX_WS	CAP3.1
19:18	P0.25	GPIO Port 0.25	AD0.2	I2SRX_SDA	TXD3
21:20	P0.26	GPIO Port 0.26	AD0.3	AOUT	RXD3
23:22	P0.27[1][2]	GPIO Port 0.27	SDA0	USB_SDA	Reserved
25:24	P0.28[1][2]	GPIO Port 0.28	SCL0	USB_SCL	Reserved
27:26	P0.29	GPIO Port 0.29	USB_D+	Reserved	Reserved
29:28	P0.30	GPIO Port 0.30	USB_D−	Reserved	Reserved
31:30	-	Reserved	Reserved	Reserved	Reserved

PIN CONNECT BLOCK

PINSEL0



PINSEL1



Configure P0.1 with Function-01 and P0.15 with Function-02, Value to be loaded to PINSEL0:
10000000000000000000000000000100 = 0x80000004 = (2<<30) | (1<<2)= (2<<(15-0)*2 | (1<<(1-0)*2)
Configure P0.17 with Function-03 and P0.30 with Function-01, Value to be loaded to PINSEL0:
00010000000000000000000000001100 = 0x1000000C = (1<<28) | (3<<2)
= (1<< (30-16)*2) | (3<<(17-16)*2)

GPIO (General Purpose Input/Output)

FIODIR	Fast GPIO Port Direction control register. This register individually controls the direction of each port pin.
FIOPIN	Fast Port Pin value register using FIOMASK. The current state of digital port pins can be read from this register, regardless of pin direction or alternate function selection (as long as pins are not configured as an input to ADC). The value read is masked by ANDing with inverted FIOMASK. Writing to this register places corresponding values in all bits enabled by zeros in FIOMASK. **Important:** if an FIOPIN register is read, its bit(s) masked with 1 in the FIOMASK register will be read as 0 regardless of the physical pin state.
FIOSET	Fast Port Output Set register using FIOMASK. This register controls the state of output pins. Writing 1s produces highs at the corresponding port pins. Writing 0s has no effect. Reading this register returns the current contents of the port output register. Only bits enabled by 0 in FIOMASK can be altered.
FIOCLR	Fast Port Output Clear register using FIOMASK. This register controls the state of output pins. Writing 1s produces lows at the corresponding port pins. Writing 0s has no effect. Only bits enabled by 0 in FIOMASK can be altered.
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.

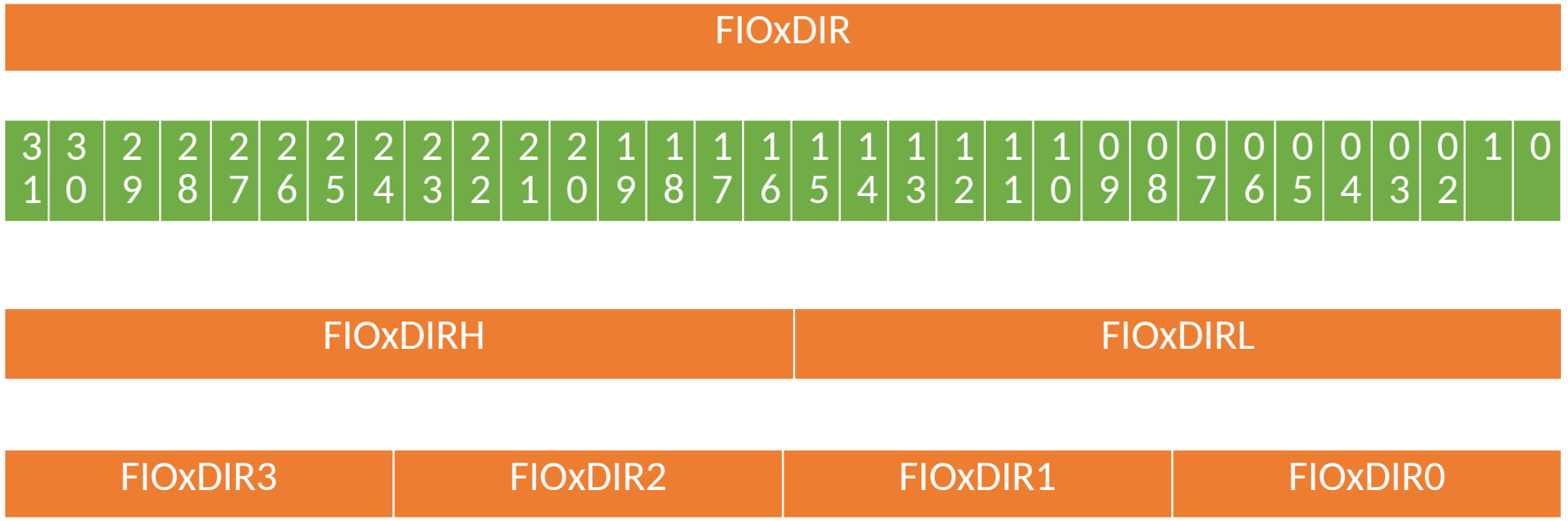
0	Controlled pin is input.
1	Controlled pin is output.

0	Controlled pin output is unchanged.
1	Controlled pin output is set to HIGH.

0	Controlled pin output is unchanged.
1	Controlled pin output is set to LOW.

0	Controlled pin is affected by writes to the port's FIOxSET, FIOxCLR, and FIOxPIN register(s). Current state of the pin can be read from the FIOxPIN register.
1	Controlled pin is not affected by writes into the port's FIOxSET, FIOxCLR and FIOxPIN register(s). When the FIOxPIN register is read, this bit will not be updated with the state of the physical pin.

GPIO (General Purpose Input/Output)



Same concept applicable to FIOxSET, FIOxCLR, FIOxPIN, FIOxMASK (Ex: For Port-1 , we can have FIO1SET, FIO1SETH, FIO1SETL, FIO1SET3, FIO1SET2, FIO1SET1, FIO1SET0……

GPIO (General Purpose Input/Output)

Ex: Send 0xA5 to P0.15-P0.8 without affecting values on the remaining pins.
This can be accomplished in several ways

```
FIO0MASK = 0xFFFF00FF ;
FIO0PIN  = 0x0000A500;
```

Solution 2: using 16-bit (half-word) accessible fast GPIO registers

```
FIO0MASKL = 0x00FF;
FIO0PINL  = 0xA500;
```

Solution 3: using 8-bit (byte) accessible fast GPIO registers

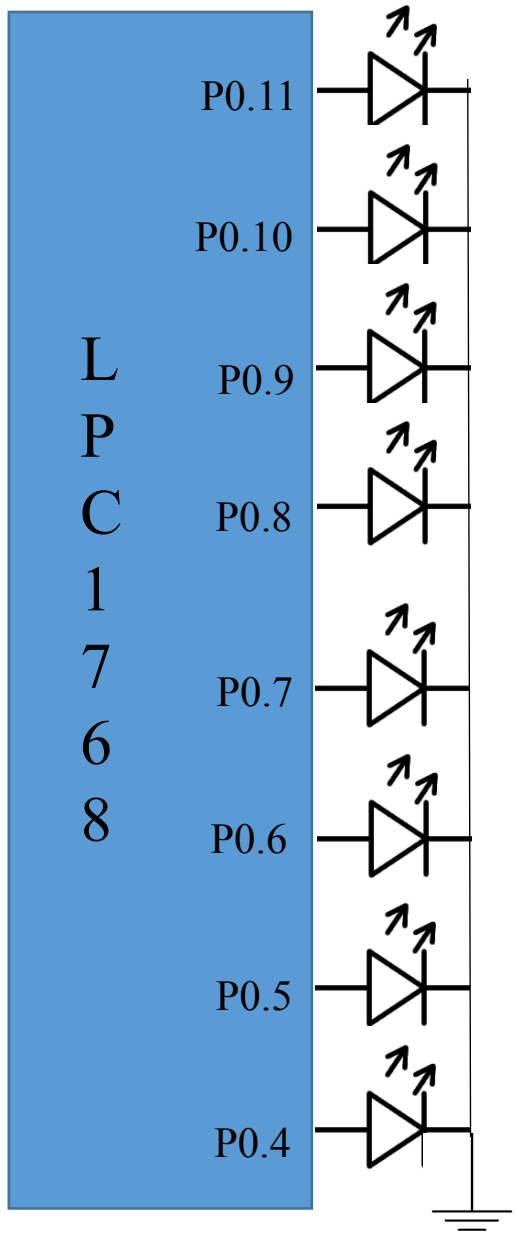
```
FIO0PIN1  = 0xA5;
```

GPIO (General Purpose Input/Output)

Write an embedded C program to turn ON and OFF LEDs connected to P0.11 – P0.4

```
#include <LPC17xx.h>
 unsigned int j;
 unsigned long LED = 0x00000FF0;

 int main(void)
 {
                SystemInit();
                SystemCoreClockUpdate();

                LPC_PINCON->PINSEL0 =  0x00000000; // P0.15-P0.0 GPIO
                LPC_GPIO0->FIODIR = 0x00000FF0; // P0.11-P0.4 as output

                while(1)
                {
                        LPC_GPIO0->FIOSET = LED; // SET P0.11-P0.4
                        for(j=0;j<10000;j++); // Delay

                        LPC_GPIO0->FIOCLR = LED; // CLEAR P0.11-P0.4
                        for(j=0;j<10000;j++); //Delay

                }
}
```

```
LPC_GPIO0->FIOPIN= ~(LPC_GPIO0->FIOPIN & 0x00000FF0);
for(j=0;j<10000;j++); //Delay
```

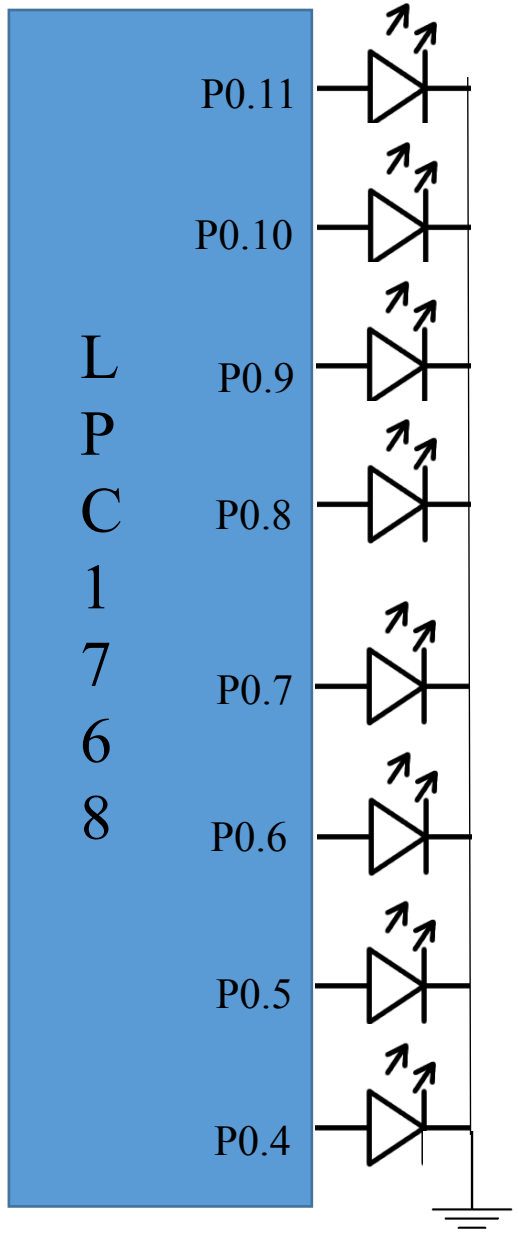


GPIO (General Purpose Input/Output)

Write an embedded C program to turn ON and OFF LEDs connected to P0.11 – P0.4

```
#include <LPC17xx.h>
 unsigned int j;
 unsigned int LED = 0x0FF0;

 int main(void)
 {
        SystemInit();
        SystemCoreClockUpdate();

        LPC_PINCON->PINSEL0 =  0x00000000; // P0.15-P0.0 GPIO
        LPC_GPIO0->FIODIRL = 0x0FF0; // P0.11-P0.4 as output

        while(1)
        {
            LPC_GPIO0->FIOSETL = LED; // SET P0.11-P0.4
            for(j=0;j<10000;j++); // Delay

            LPC_GPIO0->FIOCLRL = LED; // CLEAR P0.11-P0.4
            for(j=0;j<10000;j++); //Delay

        }
}
```

```
LPC_GPIO0->FIOPINL= ~(LPC_GPIO0->FIOPINL & 0x0FF0);
for(j=0;j<10000;j++); //Delay
```



GPIO (General Purpose Input/Output)

8 bit Johnson Counter on LEDs

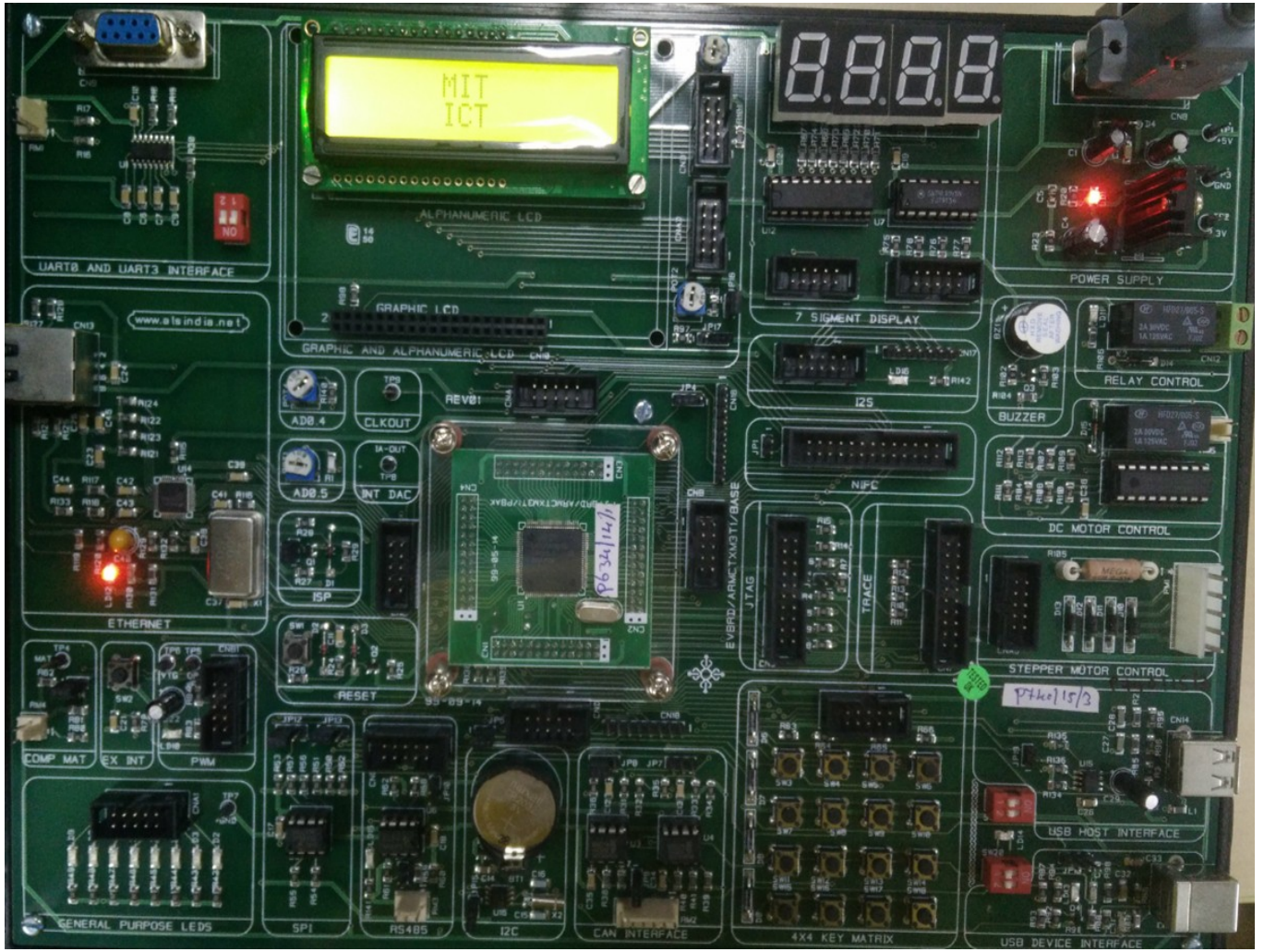
```
#include <LPC17xx.h>

unsigned int i,j;
unsigned long LED = 0x00000010;

int main(void)
{
  SystemInit()

  SystemCoreClockUpdate();


  LPC_PINCON->PINSEL0 = 0
                              ;Configure  Port0 pins  P0.4-P0.11 ;as GPIO

  LPC_GPIO0->FIODIR = 0x00000FF0;
                                        ;Configure P0.4-P0.11 as output
```

00000000
00000001
00000011
00000111
00001111
00011111
00111111
01111111
11111111
11111110
11111100
11111000
11110000
11100000
11000000
10000000
00000000

GPIO (General Purpose Input/Output)

```
while(1)
      {
                LED = 0x00000010; Initial value on LED
                for(i=1;i<9;i++)            //ON the LED's serially
                {
                  LPC_GPIO0->FIOSET = LED;

                  for(j=0;j<10000;j++);
                   LED <<= 1;
                }


                     LED = 0x00000010;

                   for(i=1;i<9;i++)         //OFF the LED's serially
                               {
                   LPC_GPIO0->FIOCLR = LED
                          for(j=0;j<10000;j++);
                   LED <<= 1;
                               }

             }
              }
```

GPIO (General Purpose Input/Output)



GPIO (General Purpose Input/Output)

Pin CNA	PIN LPC1768	Description
1	81	P0.4/I2SRX_CLK/RD2/CAP2.0
2	80	P0.5/I2SRX_WS/TD2/CAP2.1
3	79	P0.6/I2SRX_SDA/SSEL1/MAT2.0
4	78	P0.7/I2STX_CLK/SCK1//MAT2.1
5	77	P0.8/I2STX_WS/MISO1/MAT2.2
6	76	P0.9/I2STX_SDA/MOSI1/MAT2.3
7	48	P0.10/TXD2/SDA2/MAT3.0
8	49	P0.11/RXD2/SCL2/MAT3.1
9	-	No connection
10	-	Ground

Pin CNB	Pin LPC1768	Description
1	37	P1.23/MCI1/PWM1.4/MISO0
2	38	P1.24/MCI2/PWM1.5/MOSI0
3	39	P1.25/MCOA1/MAT1.1
4	40	P1.26/MCOB1/PWM1.6/CAP0.0
5	53	P2.10/EINT0/NMI
6	52	P2.11/EINT1/I2STX_CLK
7	51	P2.12/EINT2/I2STX_WS
8	50	P2.13/EINT3/I2STX_SDA
9	-	No connection
10	-	Ground

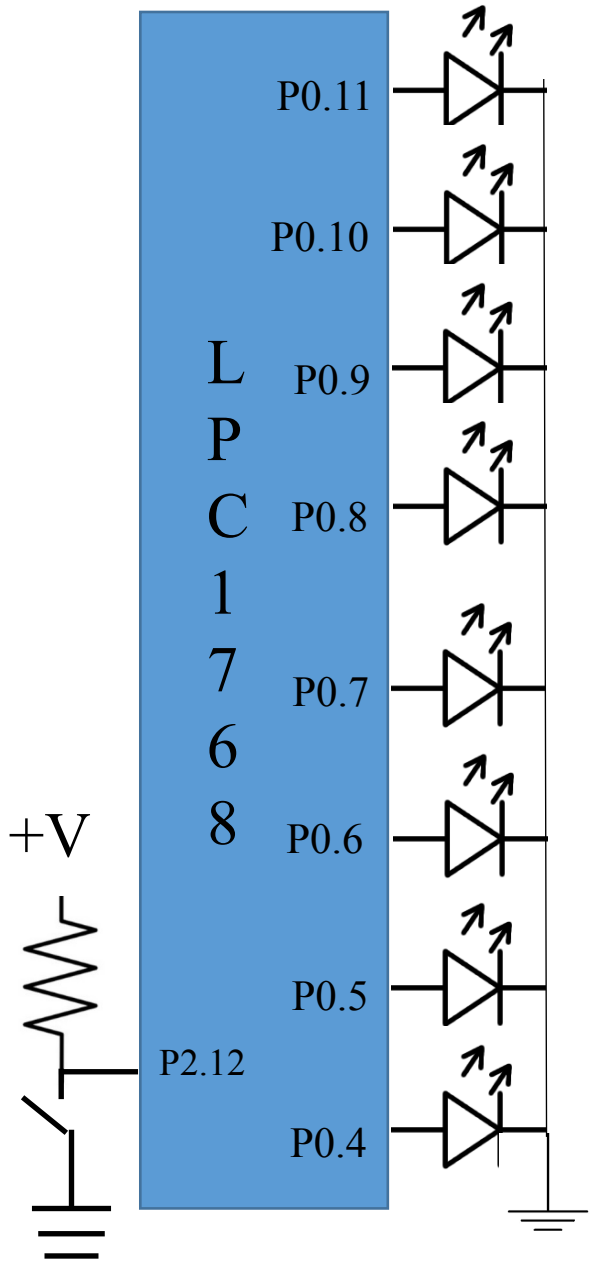
Pin CNC	Pin LPC1768	Description
1	62	P0.15/TXD1/SCK0/SCK
2	63	P0.16/RXD1/SSEL0/SSEL
3	61	P0.17/CTS1/MISO0/MISO
4	60	P0.18/DCD1/MOSI0/MOSI
5	59	P0.19/DSR1/SDA1
6	58	P0.20/DTR1/SCL1
7	57	P0.21/RI1/RD1
8	56	P0.22/RTS1/TD1
9	50	P2.13/I2STX_SDA
10	-	Ground

Pin CND	Pin LPC1768	Description
1	9	P0.23/AD0.0/I2SRX_CLK/CAP3.0
2	8	P0.24/AD0.1/I2SRX_WS/CAP3.1
3	7	P0.25/AD0.2/I2SRX_SDA/TXD3
4	6	P0.26/AD0.3/AOUT/RXD3
5	25	P0.27/SDA0/USB/SDA
6	24	P0.28/SCL0/USB_SCL
7	75	P2.0/PWM1.1/TXD1
8	74	P2.1/PWM1.2/RXD1
9	-	No connection
10	-	Ground

GPIO (General Purpose Input/Output)

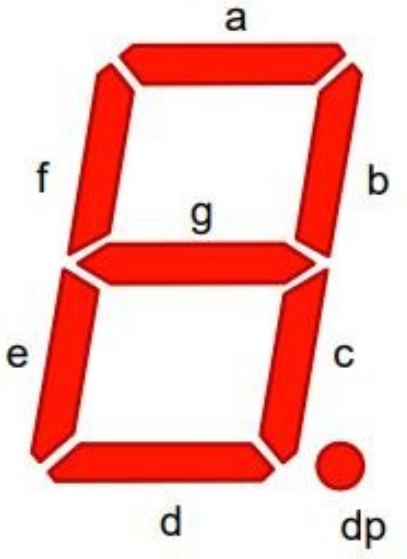
Write an embedded C program to turn ON LEDs connected to P0.11 – P0.4 when key connected to P2.12 pressed, else turn OFF.

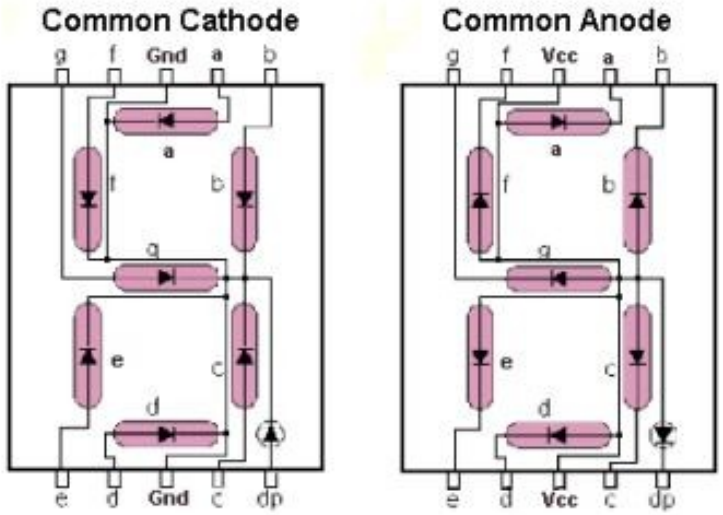
```
#include <LPC17xx.h>
 unsigned int j;
 unsigned long LED = 0x0000FF00;

 int main(void)
 {
            SystemInit();
            SystemCoreClockUpdate();

            LPC_PINCON->PINSEL0 =  0x00000000; // P0.15-P0.0 GPIO
            LPC_GPIO0->FIODIR = 0x00000FF0; // P0.11-P0.4 as output

            while(1)
            {
                if ( !(LPC_GPIO2->FIOPIN & 1<<12))

                        LPC_GPIO0->FIOSET = LED; // SET P0.11-P0.4
                else

                        LPC_GPIO0->FIOCLR = LED; // CLEAR P0.11-P0.4

            }
}
```



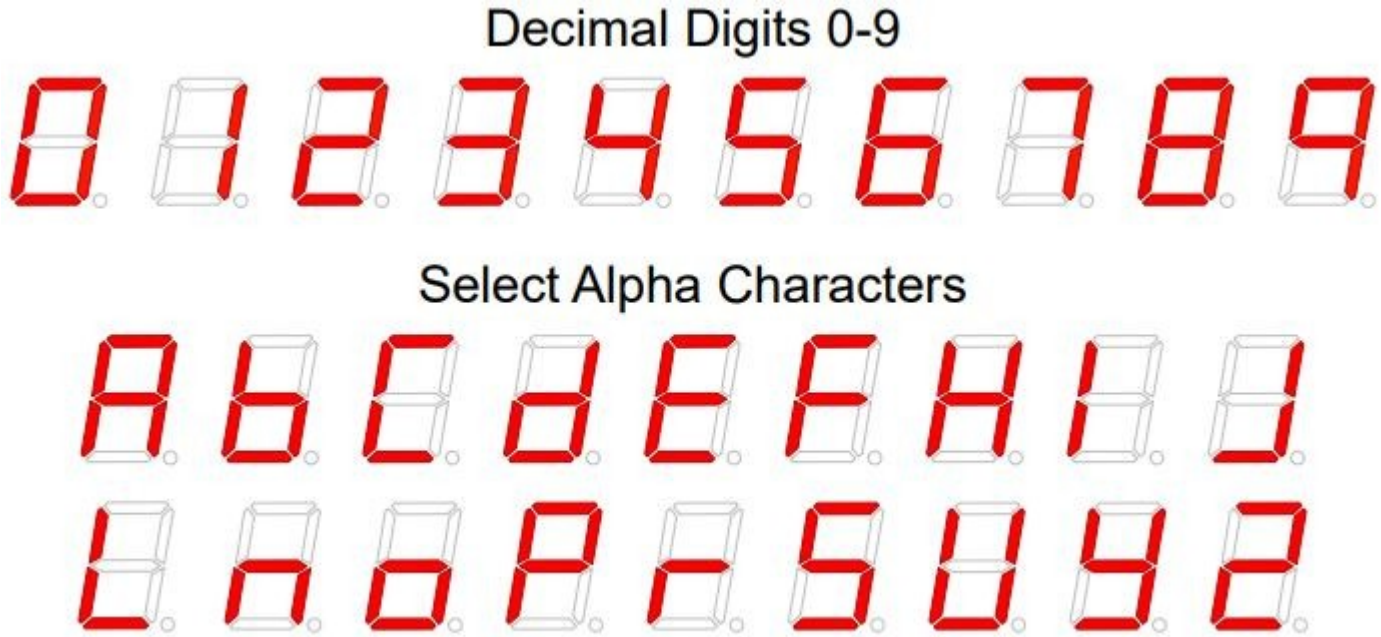
Seven Segment Display

- Common Cathode (all LED cathodes are connected)
- Common Anode (all LED anodes are connected)





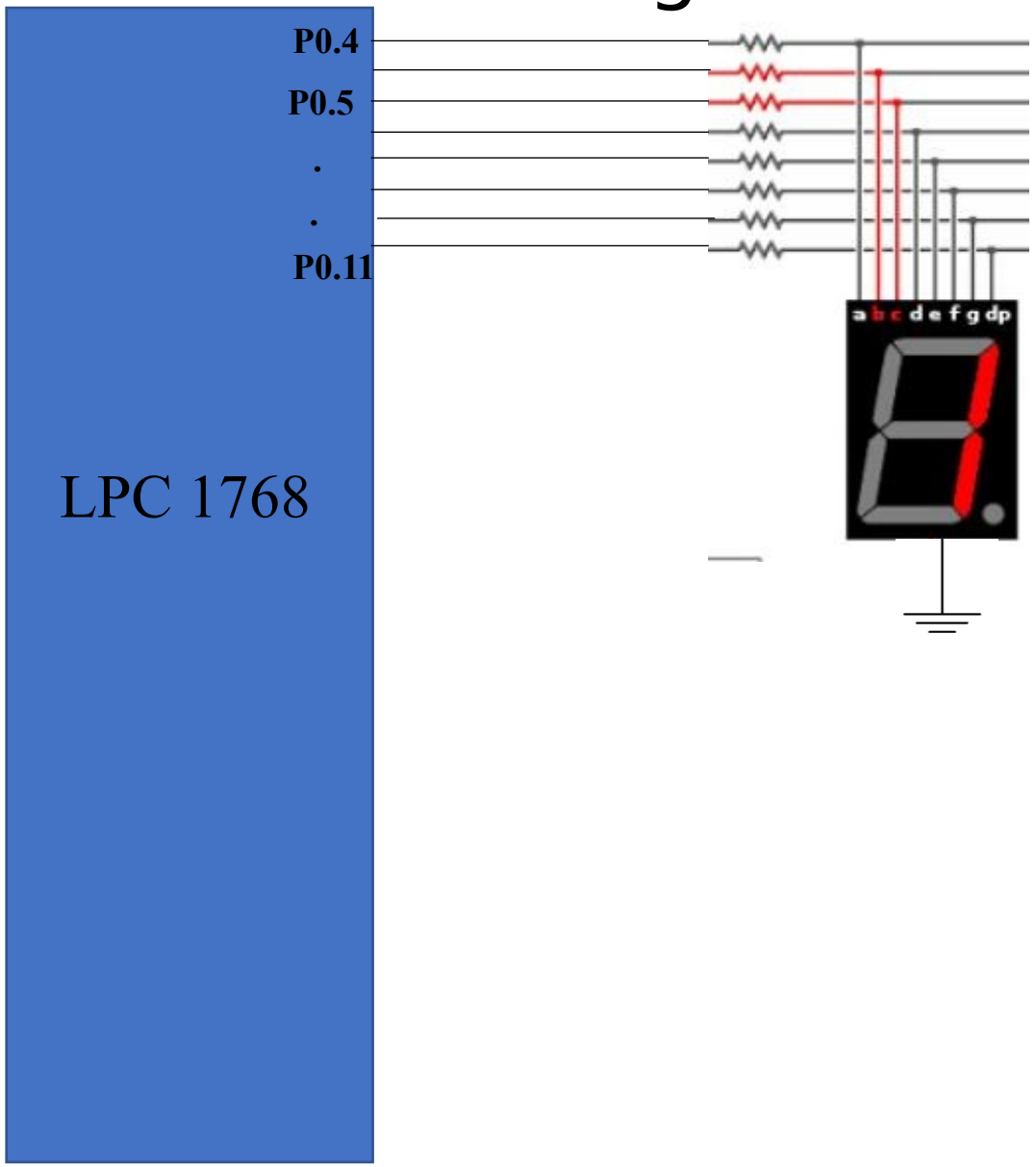
Seven Segment Display



Seven Segment Display

Number	h g f e d c b a	Hex Code
0	0 0111111	3F
1	0 0000110	06
2	1011011	5B
3	1001111	4F
4	1100110	66
5	1101101	6D
6	1111101	7D
7	0000111	07
8	1111111	7F
9	1001111	4F

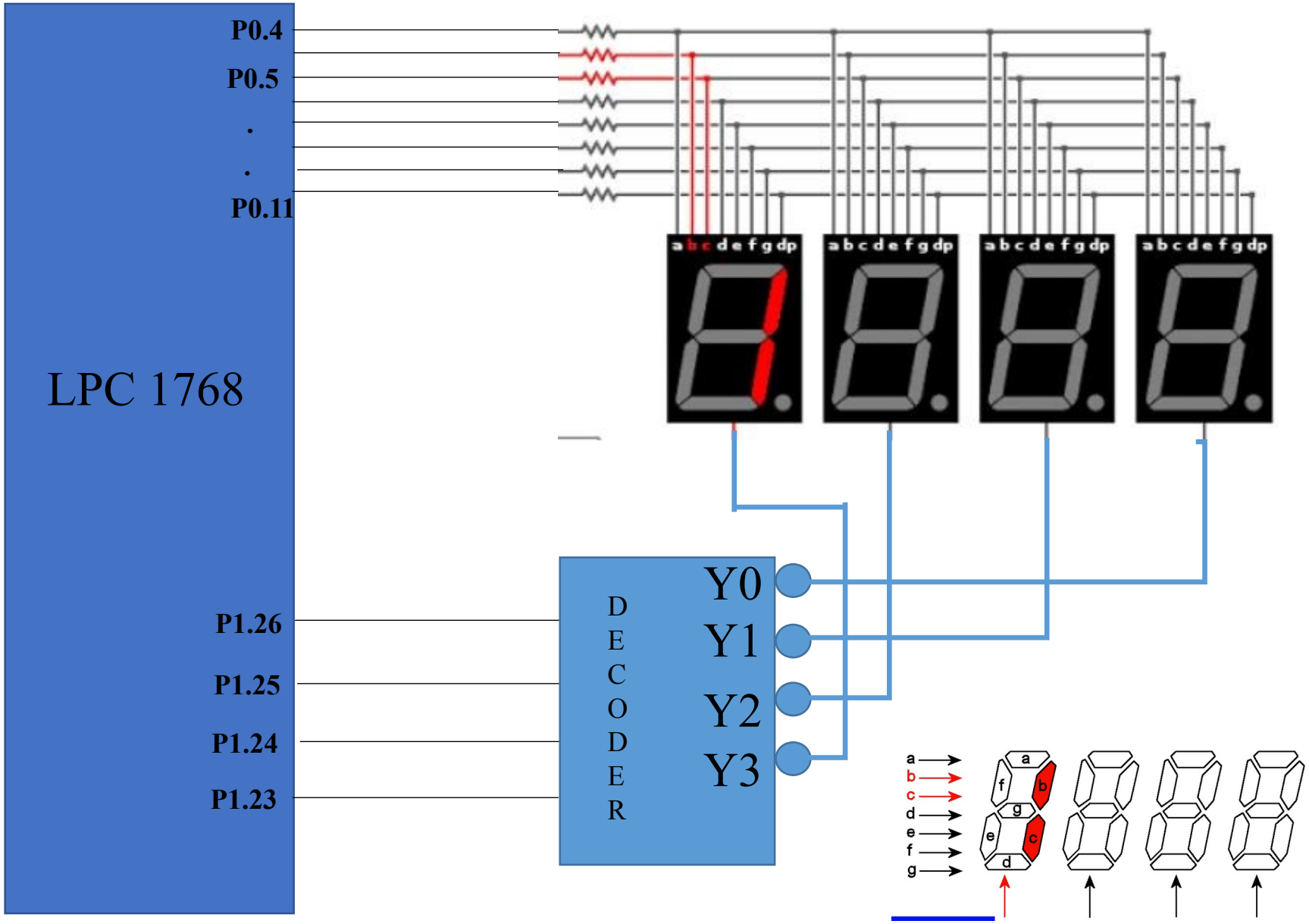
Seven Segment Display Interfacing



Seven Segment Display Interfacing

```
#include<lpc17xx.h>
unsigned char seven_seg[10]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};
unsigned int i,j;
void delay(void);
int main(void)
 {
            SystemInit();
            SystemCoreClockUpdate();

            LPC_PINCON->PINSEL0 = 0      //P0.4 to P0.11 GPIO data lines
            LPC_GPIO0->FIODIR |= 0x00000FF0;       //P0.4 to P0.11 output
            while (1)
              {
                  for(i=0; i<10; i++)
                    {
                     LPC_GPIO0->FIOPIN = seven_seg[i ] << 4;
                     delay();
                    }
              }
    }
void  delay(void)
{
 for(j=0;j<10000;j++);
}
```

Multiplexed Seven Segment Display



Multiplexed Seven Segment Display

```
#include <LPC17xx.h>
#include <stdio.h>

#define   FIRST_SEG         0<<23
#define   SECOND_SEG        1<<23
#define   THIRD_SEG         2<<23
#define   FOURTH_SEG        3<<23


unsigned int dig_count;
unsigned int digit_value = {0, 4, 3, 2, 1}
unsigned int select_segment = {0, 0 << 23, 1<<23. 2<<23. 3<<23};
unsigned char seven_seg[10]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};
unsigned long int temp1,temp2 ,i=0;

void Display(void);
void delay(void);
int main(void)
{
        SystemInit();
        SystemCoreClockUpdate();

        LPC_PINCON->PINSEL0 = 0;    //P0.4 to P0.11 GPIO data lines
        LPC_PINCON->PINSEL3 = 0;    //P1.23 to P1.26 GPIO enable lines

        LPC_GPIO0->FIODIR = 0x00000FF0;       //P0.4 to P0.11 output
        LPC_GPIO1->FIODIR = 0x07800000;       //P1.23 to P1.26 output
```

Multiplexed Seven Segment Display

```
        while(1)
        {
                delay();

                dig_count +=1;
                if(dig_count == 0x05)
                        dig_count = 0x00;

                Display();

        } //end of while(1)

}//end of main


 void Display(void)     //To Display on 7-segments
{

        LPC_GPIO1->FIOPIN = select_segment[dig_count];
        LPC_GPIO0->FIOPIN = seven_seg[digit_value[dig_count]] << 4;
        for(i=0;i<500;i++);
        LPC_GPIO0->FIOCLR = 0x00000FF0;
}
void delay(void)
{
for i=0;i<500;i++);
}
```

4-digit BCD upcounter

```
        while(1)
        {
                delay();

                dig_count +=1;
                if(dig_count == 0x05)
                        dig_count = 0x00;

                Display();

        } //end of while(1)

}//end of main


 void Display(void)     //To Display on 7-segments
{

        LPC_GPIO1->FIOPIN = select_segment[dig_count];
        LPC_GPIO0->FIOPIN = seven_seg[digit_value[dig_count]] << 4;
        for(i=0;i<500;i++);
        LPC_GPIO0->FIOCLR = 0x00000FF0;
}
void delay(void)
{
for i=0;i<500;i++);
}
```

For every second update the digits

After one second, set Flag

4-digit BCD upcounter

```
if(flag == 0xFF)
        {
                flag = 0;
                digit_value[1] +=1;

                if(digit_value[1] == 0x0A)
                {
                        digit_value[1] = 0;
                        digit_value[2] +=1;

                        if(digit_value[2] == 0x0A)
                        {
                                digit_value[2]= 0;
                                digit_value[3] +=1;

                                if(digit_value[3] == 0x0A)
                                {
                                        digit_value[3] = 0;
                                        digit_value[4] += 1;

                                        if(digit_value[4]== 0x0A)
                                        {
                                                digit_value[4]= 0;
                                        } //end of dig4

                                } //end of dig3

                        } //end of dig2

                } //end of dig1

        } //end of one_sec if
```

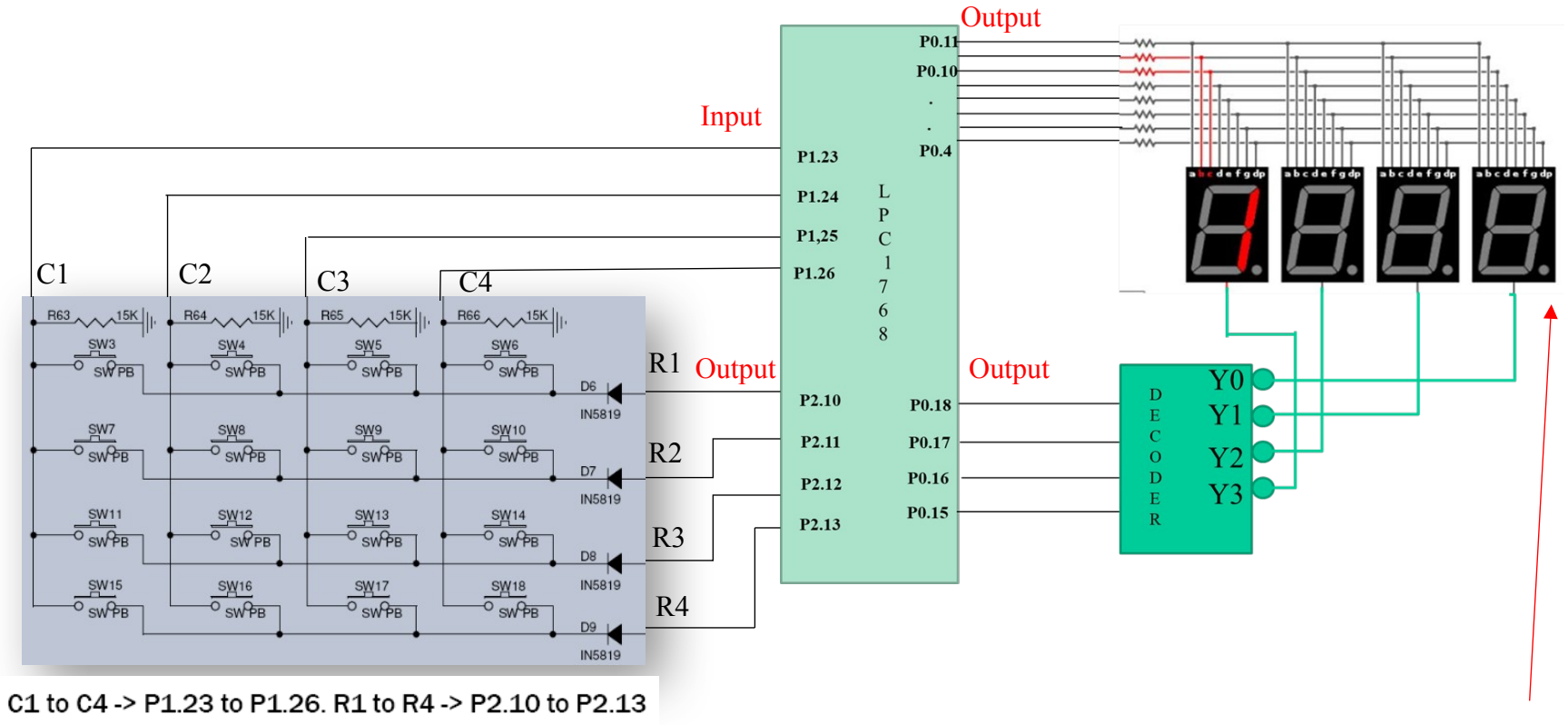
For every second update the digits

4-digit BCD upcounter

```
void delay(void)
{
for i=0;i<500;i++);

if(count ==N)
            {
                        flag = 0xFF;
                        count = 0;
            }
            else count += 1;
}
```

After one second, set Flag

Matrix Keyboard Interfacing



C1 to C4 -> P1.23 to P1.26. R1 to R4 -> P2.10 to P2.13

Display keycode on Digit1 (0 to F)

Matrix Keyboard Interfacing

```
#include <LPC17xx.h>
#define   FIRST_SEG           0xFFF87FFF
void scan(void);

unsigned char col,row,flag;
unsigned long int i,var1,temp,temp3,temp2;
unsigned char SEVEN_CODE[4][4] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0x77,0x7c,0x58,0x5e,0x79,0x71};

int main(void)
{
        SystemInit();
        SystemCoreClockUpdate();
        LPC_PINCON->PINSEL0 = 0;    //P0.4 to P0.11 GPIO data lines
        LPC_GPIO0->FIODIR = 0xFFFFFFFF;      //Port 0 output
        LPC_PINCON->PINSEL3 = 0; //P1.23 to P1.26 MADE GPIO
        LPC_PINCON->PINSEL4 = 0; //P2.10 t P2.13 made GPIO
        LPC_GPIO2->FIODIR = 0x00003C00; //made output P2.10 to P2.13 (rows)
        LPC_GPIO1->FIODIR =0; //made input P1.23 to P1.26 (cols)
```

Matrix Keyboard Interfacing

```
        while(1)
        {
                for(row=0;row<4;row++)
                        {
                                if(row == 0)
                                temp = 1<<10;
                                else if(row == 1)
                                temp = 1<<11;
                                else if(row == 2)
                                temp = 1<<12;
                                else if(row == 3)
                                temp = 1<<13;

                                LPC_GPIO2->FIOPIN = temp;
                                flag = 0;
                                scan();
                                if(flag == 1)
                                {
                                temp2 = SEVEN_CODE[row][col];
                                LPC_GPIO0->FIOMASK=0xFFF87FFF;
                                LPC_GPIO0->FIOPIN = FIRST_SEG;
                                temp2 = temp2 << 4;
                                LPC_GPIO0->FIOMASK=0xFFFFF00F;
                                LPC_GPIO0->FIOPIN = temp2;              // Taking Data Lines for 7-Seg
                                break;
                                }
                        } //end for(row=1;row<5;row++)

                }//end while 1

}//end main
```

Matrix Keyboard Interfacing

```
void scan(void)
{
        unsigned long temp3;

        temp3 = LPC_GPIO1->FIOPIN;
        temp3 &= 0x07800000;
        if(temp3 != 0x00000000)
        {
                                flag = 1;
                        if (temp3 == 1<<23)
                                col=0;
                        else if (temp3 == 1<<24)
                                col=1;
                        else if (temp3 == 1<<25)
                                col=2;
                        else if (temp3 == 1<<25)
                                col=3;

        }//1st if(temp3 != 0x00000000)
}//end scan
```

Timer/Counter Programming

Timer - Interval Timer to generate the intended delay . The Timer is designed to count cycles of the peripheral clock (PCLK)
Counter - Counting internal events. or an externally-supplied clock

There are four 32-bit Timers in LPC1768:
Timer0, Timer1, Timer2 and Timer3.

Timer/Counter Programming

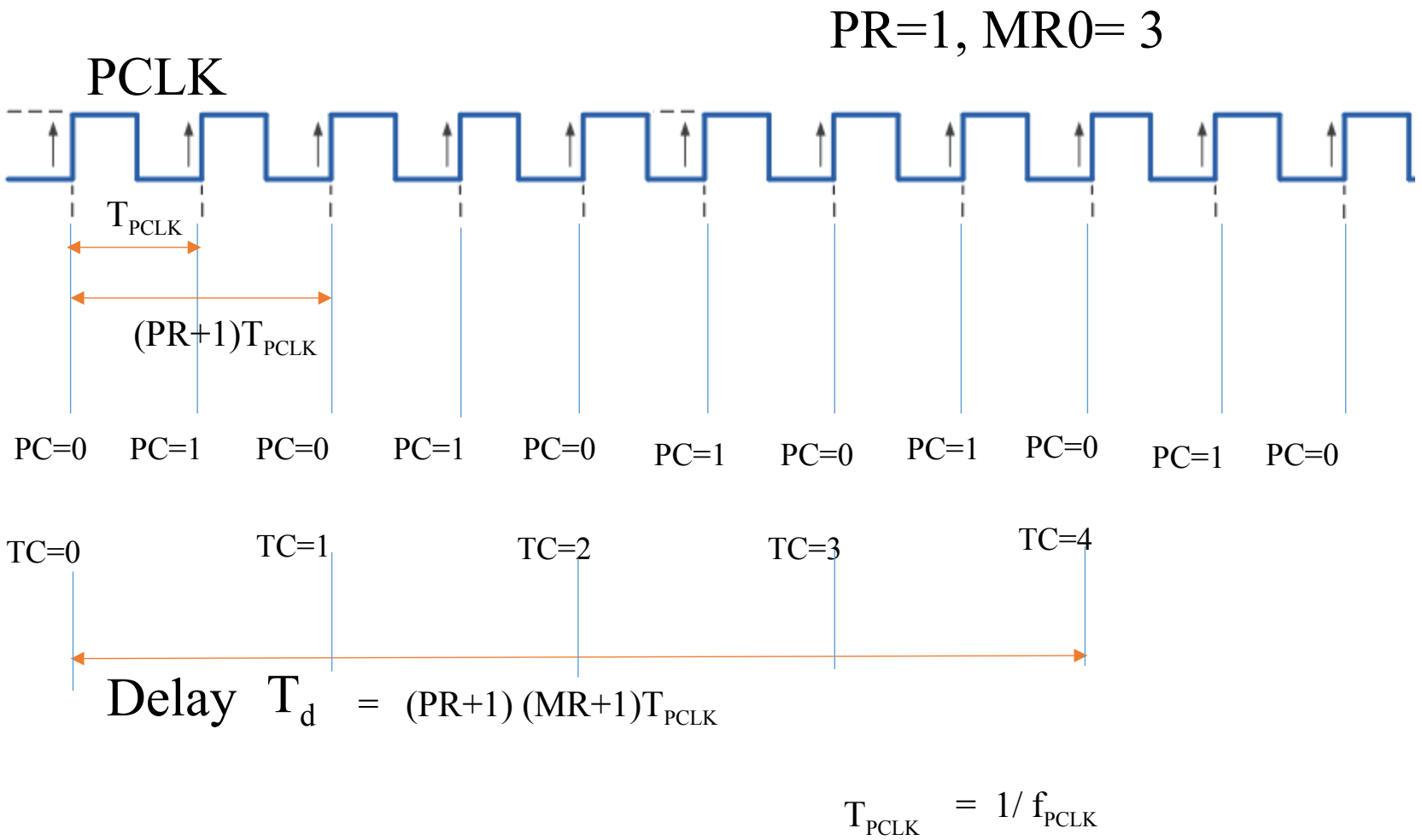
PC – Prescale Counter Register: It is a 32-bit register. The value in PC is incremented on every PCLK cycle and when its value matches the value in PR, the TC is incremented and the value in PC is RESET on the next PCLK cycle.

PR – Prescale Register: It is a 32-bit register and specifies the maximum value for the Prescale Counter (PC).

TC – Timer Counter Register: It is a 32-bit register and is incremented at every PR+1 cycles of PCLK.

MR0 – MR3 – Match Registers: Contains user loaded values to be compared with TC. When value in Match Register matches with TC, appropriate action can be performed.

Timer/Counter Programming



Ex: For frequency 3MHz, to get 1 second delay: PR= 999, MR0= 2999

PR= 2999, MR0= 999 etc.

Timer/Counter Programming

TCR – Timer Control register: Used to control Timer Counter functions i.e. enable, disable and reset.

Bit 0	Counter Enable	When 1, TC and PC are enabled for counting. When 0, TC and PC are disabled.
Bit 1	Counter Reset	When 1, TC and PC are reset on next positive edge of PCLK. The counters remain reset until this bit is returned to 0.

Timer/Counter Programming

CTCR (Count Control Register) is used to select between Timer and Counter mode, and in Counter mode to select the pin and edge(s) for counting.

Bit	Symbol	Value	Description
1:0	Counter/ Timer Mode		This field selects which rising PCLK edges can increment the Timer's Prescale Counter (PC), or clear the PC and increment the Timer Counter (TC).
		00	Timer Mode: the TC is incremented when the Prescale Counter matches the Prescale Register. The Prescale Counter is incremented on every rising PCLK edge.
		01	Counter Mode: TC is incremented on rising edges on the CAP input selected by bits 3:2.
		10	Counter Mode: TC is incremented on falling edges on the CAP input selected by bits 3:2.
		11	Counter Mode: TC is incremented on both edges on the CAP input selected by bits 3:2.
3:2	Count Input Select		When bits 1:0 in this register are not 00, these bits select which CAP pin is sampled for clocking.
		00	CAPn.0 for TIMERn
		01	CAPn.1 for TIMERn
		10	Reserved
		11	Reserved

Timer/Counter Programming

CAP0.0	P1.26
CAP0.1	P1.27
CAP1.0	P1.18 / P1.28 / P2.6
CAP1.1	P1.19 / P1.29
CAP2.0	P0.4
CAP2.1	P0.5
CAP3.0	P0.23
CAP3.1	P0.24
MAT0.0	P1.28 / P3.25
MAT0.1	P1.29 / P3.26
MAT1.0	P1.22
MAT1.1	P1.25
MAT2.0	P0.6 / P4.28
MAT2.1	P0.7 / P4.29
MAT2.2	P0.8
MAT2.3	P0.9
MAT3.0	P0.10
MAT3.1	P0.11

Timer/Counter Programming

Match Registers (MR0 - MR3)

- The Match register values are continuously compared to the Timer Counter value.
- When the two values are equal, actions can be triggered automatically.
- The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

Timer/Counter Programming

Match Control Register (MCR)

The Match Control Register is used to control what operations are performed when one of the Match Registers matches the Timer Counter.

Bit	Symbol	Value	Description
0	MR0I	1	Interrupt on MR0: an interrupt is generated when MR0 matches the value in the TC.
		0	This interrupt is disabled
1	MR0R	1	Reset on MR0: the TC will be reset if MR0 matches it.
		0	Feature disabled.
2	MR0S	1	Stop on MR0: the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC.
		0	Feature disabled.
3	MR1I	1	Interrupt on MR1: an interrupt is generated when MR1 matches the value in the TC.
		0	This interrupt is disabled
4	MR1R	1	Reset on MR1: the TC will be reset if MR1 matches it.
		0	Feature disabled.
5	MR1S	1	Stop on MR1: the TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC.
		0	Feature disabled.

Timer/Counter Programming

MR2I	1	Interrupt on MR2: an interrupt is generated when MR2 matches the value in the TC.
	0	This interrupt is disabled
MR2R	1	Reset on MR2: the TC will be reset if MR2 matches it.
	0	Feature disabled.
MR2S	1	Stop on MR2: the TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC.
	0	Feature disabled.
MR3I	1	Interrupt on MR3: an interrupt is generated when MR3 matches the value in the TC.
	0	This interrupt is disabled
MR3R	1	Reset on MR3: the TC will be reset if MR3 matches it.
	0	Feature disabled.
MR3S	1	Stop on MR3: the TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC.
	0	Feature disabled.
-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

Timer/Counter Programming

External Match Register (EMR): It provides Match Outputs. Also, the External Match Register provides both control and status of the external match pins.

Bit	Symbol	Description
0	EM0	External Match 0. When a match occurs between the TC and MR0, this bit can either toggle, go low, go high, or do nothing, depending on bits 5:4 of this register. This bit can be driven onto a MATn.0 pin, in a positive-logic manner (0 = low, 1 = high).
1	EM1	External Match 1. When a match occurs between the TC and MR1, this bit can either toggle, go low, go high, or do nothing, depending on bits 7:6 of this register. This bit can be driven onto a MATn.1 pin, in a positive-logic manner (0 = low, 1 = high).
2	EM2	External Match 2. When a match occurs between the TC and MR2, this bit can either toggle, go low, go high, or do nothing, depending on bits 9:8 of this register. This bit can be driven onto a MATn.2 pin, in a positive-logic manner (0 = low, 1 = high).
3	EM3	External Match 3. When a match occurs between the TC and MR3, this bit can either toggle, go low, go high, or do nothing, depending on bits 11:10 of this register. This bit can be driven onto a MATn.3 pin, in a positive-logic manner (0 = low, 1 = high).
5:4	EMC0	External Match Control 0. Determines the functionality of External Match 0. Table 433 shows the encoding of these bits.
7:6	EMC1	External Match Control 1. Determines the functionality of External Match 1. Table 433 shows the encoding of these bits.
9:8	EMC2	External Match Control 2. Determines the functionality of External Match 2. Table 433 shows the encoding of these bits.
11:10	EMC3	External Match Control 3. Determines the functionality of External Match 3. Table 433 shows the encoding of these bits.

Timer/Counter Programming

EMR[11:10], EMR[9:8], EMR[7:6], or EMR[5:4]	Function
00	Do Nothing.
01	Clear the corresponding External Match bit/output to 0 (MATn.m pin is LOW if pinned out).
10	Set the corresponding External Match bit/output to 1 (MATn.m pin is HIGH if pinned out).
11	Toggle the corresponding External Match bit/output.

MAT0.0	P1.28 / P3.25
MAT0.1	P1.29 / P3.26
MAT1.0	P1.22
MAT1.1	P1.25
MAT2.0	P0.6 / P4.28
MAT2.1	P0.7 / P4.29
MAT2.2	P0.8
MAT2.3	P0.9
MAT3.0	P0.10
MAT3.1	P0.11

Timer/Counter Programming

```
#include<stdio.h>
#include<LPC17xx.h>
void delay(void)
{
        LPC_TIM0->TCR = 0x00000002;     // Timer0 Reset
        LPC_TIM0->EMR = 0X20;//Set match bit upon match
        LPC_TIM0->PR = 1000; /
        LPC_TIM0->MR0 = 3000;          //for 1 second
        LPC_TIM0->MCR = 0x00000004;     // stop PC and TC on MR0
        LPC_TIM0->TCR = 0x00000001;     // Timer0 Enable
        while ( !(LPC_TIM0->EMR & 0x01)); // wait until match

        return;
        }


int main(void)
{
        LPC_GPIO0->FIODIR=0x00000004;

                while(1)
        {


                LPC_GPIO0->FIOSET=0x4;
                delay();

                LPC_GPIO0->FIOCLR=0x4;
                delay();
        }
}
```

Toggle LED connected to P0.2 every second.i.e generate square wave with period 2 seconds

```
LPC_GPIO0->FIOPIN=~(LPC_GPIO0->FIOPIN & 0x00000004);
Delay();
```

Timer/Counter Programming

Generate square wave of period 2 seconds with 75% duty cycle on P0.2

```
#include<stdio.h>
#include<LPC17xx.h>
void delay(void)
{
		//LPC_SC->PCONP |= (1<<1);    //powers the T0
		LPC_TIM0->TCR = 0x00000002;     // Timer0 Reset
		LPC_TIM0->EMR = 0X20; //Set EM0 upon match
		LPC_TIM0->PR = 2999;
		LPC_TIM0->MCR = 0x00000004;     // stop PC and TC on MR0
		LPC_TIM0->TCR = 0x00000001;     // Timer0 Enable
		while ( !(LPC_TIM0->EMR & 0x01));// Wait until EM0 is set
		return;
		}
int main(void)
{
		LPC_GPIO0->FIODIR=0x00000004;
				while(1)
		{
				LPC_GPIO0->FIOPIN=0x00000004;
				LPC_TIM0->MR0 = 1500; //For 1.5 seconds
				delay();
				LPC_GPIO0->FIOPIN=0x00000000;
				LPC_TIM0->MR0 = 500; //For 0.5 seconds
				delay();
}
}
```

Timer/Counter Programming

Square waveform on MAT 0.0 output line by taking EM0 on the output pin.

```
#include<stdio.h>
#include<LPC17xx.h>
void delay(void)
{
        LPC_TIM0->TCR = 0x00000002;     // Timer0 Reset
        LPC_TIM0->CTCR = 0x00000000;
        LPC_TIM0->EMR = 0X30;//Toggle bit upon match
        LPC_TIM0->PR = 0; //
        LPC_TIM0->MR0 = 3000000;        //
        LPC_TIM0->MCR = 0x00000002;     // Reset TC
        LPC_TIM0->TCR = 0x00000001;     // Timer0 Enable
        return;
        }
int main(void)
{
                LPC_PINCON->PINSEL3 |= (3<<24);//Get EM0 on MAT0.0 (P1.28) line
                delay();
                while(1);
}
```

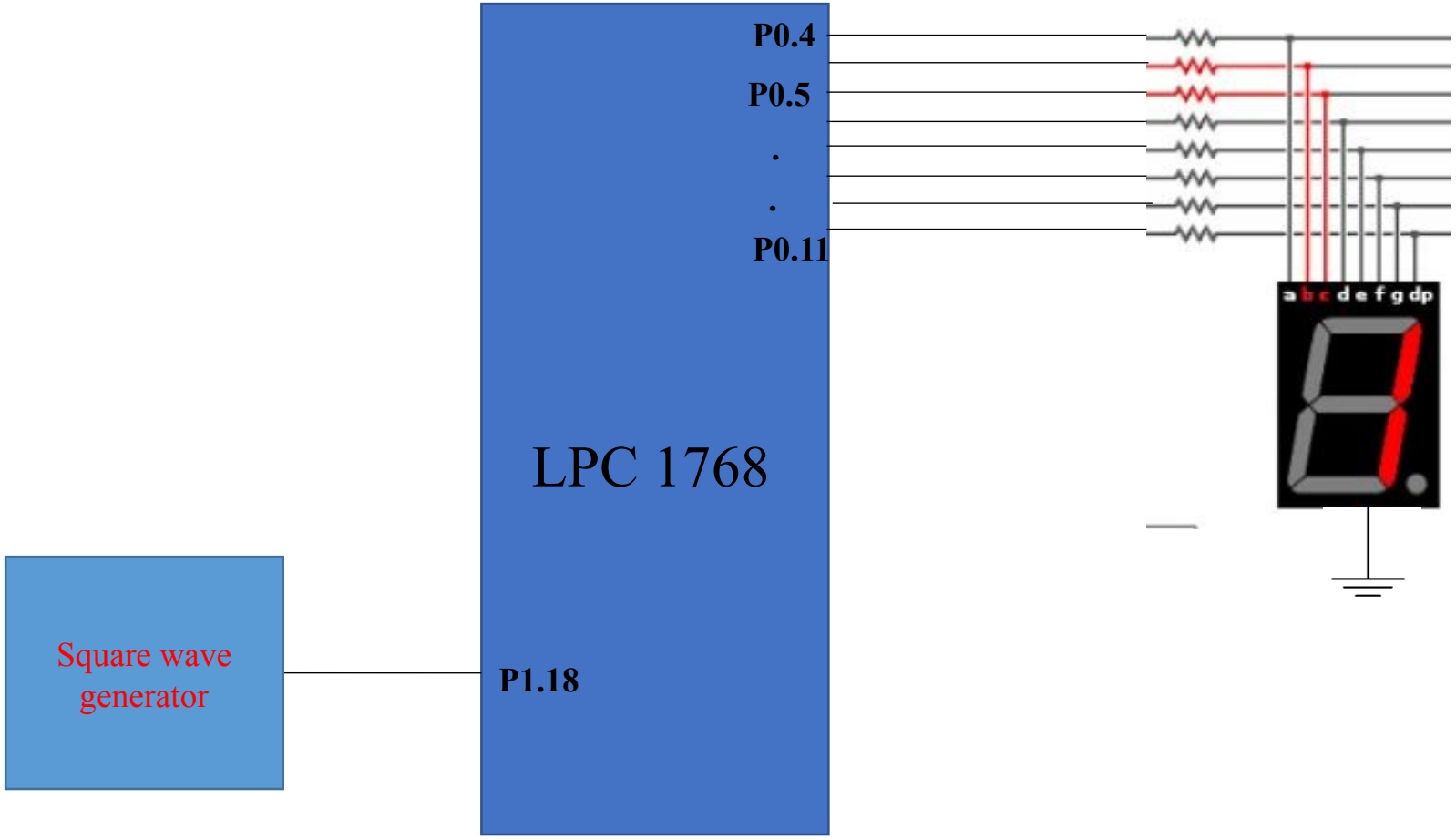
Timer/Counter Programming

MAT 1.1(P1.25) toggles whenever count reaches 3. CAP 1.0 (P1.18) is counter clock. i.e Divide the frequency of the square waveform input at P1.18 by a factor of 8 on P1.25

```
#include<stdio.h>
#include<LPC17xx.h>
void init_timer1(void)
{
        LPC_PINCON->PINSEL3 |=(3<<18 | 3<<4);//  MAT 1.1(P1.25) and CAP 1.0 (P1.18)
        LPC_TIM1->TCR=2;//Reset Counter1
        LPC_TIM1->CTCR = 0x2; // Counter at -ve edge of CAP1.0
        LPC_TIM1->MR1=0x03; //To count 4 clock pulses
        LPC_TIM1->MCR=0x10;//Clear TC upon Match1
        LPC_TIM1->EMR=0xC0;//Toggle EM1 upon Match
        LPC_TIM1->TCR=1;//Start Counter1
}
int  main(void)
{
        init_timer1();
        while(1);
}
```

Timer/Counter Programming

Assume that output of a square wave generator (Frequency < 10Hz) is connected to P1.18 (CAP1.0, Function-3) , write aprogram to display the frequency of this square waveform on the seven segment connected to P011-P0.4.



Timer/Counter Programming

```
#include<stdio.h>
#include<LPC17xx.h>
unsigned char seven_seg[10]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};
void delay(void)
{
        LPC_TIM0->TCR = 0x00000002;     // Timer0 Reset
        LPC_TIM0->EMR = 0X20;//Set match bit upon match
        LPC_TIM0->PR = 3000; //for 1 ms
        LPC_TIM0->MR0 = 1000;               //for 1 second
        LPC_TIM0->MCR = 0x00000004;     // stop PC and TC on MR0
        LPC_TIM0->TCR = 0x00000001;     // Timer0 Enable
        while ( !(LPC_TIM0->EMR & 0x01)); // wait until match
        }
void init_counter1(void)
{
        LPC_PINCON->PINSEL3 = (3<<4);//  cap 1.0 (P1.18)
        LPC_TIM1->CTCR = 0x01; // Counter at +ve edge of CAP1.0
}
```

Timer/Counter Programming

```
int  main(void)
{
        LPC_PINCON->PINSEL0 = 0          //P0.4 to P0.11 GPIO data lines
        LPC_GPIO0->FIODIR = 0x00000FF0;          //P0.4 to P0.11 output
        init_counter1();
while(1)
       {
        LPC_TIM1->TCR=2;//Reset Counter1
        LPC_TIM1->TCR=1;//Start Counter1
        Delay(); // wait for 1 second
        LPC_GPIO0->FIOPIN = seven_seg[LPC_TIM1->TC ] << 4; Counter1 on the
seven segment
        }

}
```

Timer/Counter Programming

Capture Registers (CR0 - CR1)

Each Capture register is associated with a device pin and may be loaded with the Timer Counter value when a specified event occurs on that pin.

The settings in the Capture Control Register determine whether the capture function is enabled, and whether a capture event happens on the rising edge of the associated pin, the falling edge, or on both edges.

Timer/Counter Programming

Capture Control Register (CCR)

The Capture Control Register is used to control whether one of the four Capture Registers is loaded with the value in the Timer Counter when the capture event occurs, and whether an interrupt is generated by the capture event.

Timer/Counter Programming

Capture Control Register (CCR)

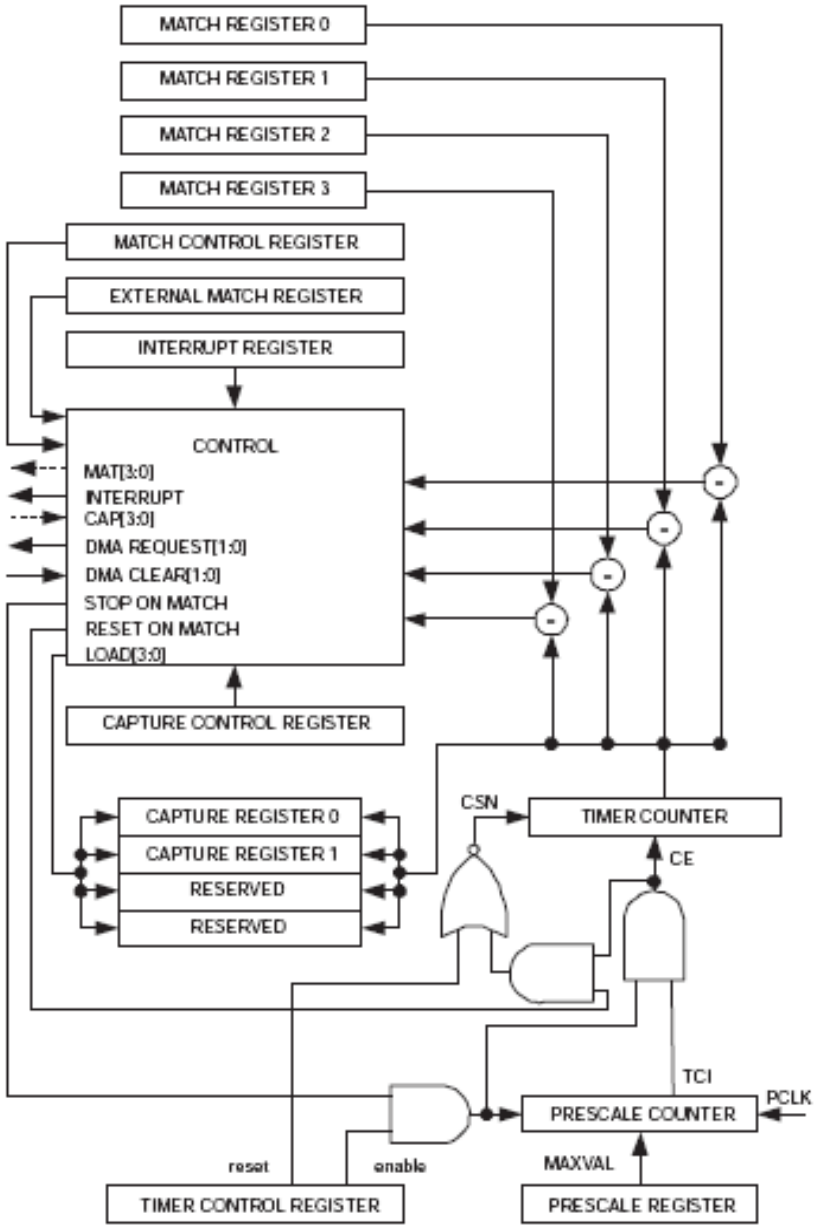
Bit	Symbol	Value	Description
0	CAP0RE	1	Capture on CAPn.0 rising edge: a sequence of 0 then 1 on CAPn.0 will cause CR0 to be loaded with the contents of TC.
		0	This feature is disabled.
1	CAP0FE	1	Capture on CAPn.0 falling edge: a sequence of 1 then 0 on CAPn.0 will cause CR0 to be loaded with the contents of TC.
		0	This feature is disabled.
2	CAP0I	1	Interrupt on CAPn.0 event: a CR0 load due to a CAPn.0 event will generate an interrupt.
		0	This feature is disabled.
3	CAP1RE	1	Capture on CAPn.1 rising edge: a sequence of 0 then 1 on CAPn.1 will cause CR1 to be loaded with the contents of TC.
		0	This feature is disabled.
4	CAP1FE	1	Capture on CAPn.1 falling edge: a sequence of 1 then 0 on CAPn.1 will cause CR1 to be loaded with the contents of TC.
		0	This feature is disabled.
5	CAP1I	1	Interrupt on CAPn.1 event: a CR1 load due to a CAPn.1 event will generate an interrupt.
		0	This feature is disabled.

Timer/Counter Programming

Capture TC into TC when +ve edge is applied to CAP0.0 (P1.26) or CAP0.1(P1.27)

```
#include<stdio.h>
#include<LPC17xx.h>
void delay(void)
{
        //LPC_SC->PCONP |= (1<<1);    //powers the T0
        LPC_TIM0->CCR=9;//capture on positive edge
        LPC_TIM0->TCR = 0x00000002;     // Timer0 Reset
        LPC_TIM0->EMR = 0X20;//Set match bit upon match
        LPC_TIM0->PR = 3000; //for 1 ms
        LPC_TIM0->MR0 = 1000;                 //for 1 second
        LPC_TIM0->MCR = 0x00000004;     // stop PC and TC on MR0
        LPC_TIM0->TCR = 0x00000001;     // Timer0 Enable
        while ( !(LPC_TIM0->EMR & 0x01)); // wait until match
        return;
        }

int main(void)
{
        LPC_GPIO0->FIODIR=0x00000004;
        LPC_PINCON->PINSEL3 |=(3<<20) | (3<<22);//select cap 0.0 and cap 0.1
                                while(1)
        {
                LPC_GPIO0->FIOPIN=~(LPC_GPIO0->FIOPIN & 0x00000004);//toggle p0.2
                delay();

        }
}
```

Timer/Counter Programming



Timer Block diagram