LPC1768

- ARM Cortex-M3 based microcontroller
- Harvard architecture
- 3.3 V Supply
- Operate at up to 100 MHz frequency
- 512 K Flash memory, 64K SRAM, 8K ROM
- Ethernet MAC,
- USB interface➔ Host/ Device,
- 8 channel general purpose DMA controller
- 4 UARTs
- 2 CAN channels
- 2 SSP controllers, SPI interface
- 3 I2C interfaces
- 8 channel 12-bit ADC
- 10-bit DAC
- Motor control PWM
- Quadrature Encoder interface
- 4 general purpose timers
- 6-output general purpose PWM
- Ultra-low power RTC with separate battery supply
- Up to 70 general purpose I/O pins.

Applications

Automotive

1. Automotive Advanced Exterior Lighting
2. Heating Ventilation, and Air Conditioning (HVAC)
3. Motorcycle Engine Control Unit (ECU) and Small Engine Control

Industrial

1. 3-Phase AC Induction Motor
2. Air Conditioning (AC)
3. Electricity Grid and Distribution
4. Electricity Meter
5. Gas Meter
6. Heat Metering
7. Motion Control and Robotics
8. Motor Drives
9. Permanent Magnet Synchronous Motor (PMSM)
10. Smart Lighting
11. Smart Power Socket and Light Switch
12. Water Meter

Mobile

1. Hearables
2. Input Device (Mouse, Pen, Keyboard)
3. Smart Watch
4. Wristband

Smart City

1. Automatic Vehicle Identification
2. Transport Ticketing

Smart Home

1. Home Control Panel
2. Home Security and Surveillance
3. Major Home Appliances
4. Robotic Appliance
5. Small and Medium Appliances

Ref: https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/lpc1700-arm-cortex-m3/512-kb-flash-64-kb-sram-ethernet-usb-lqfp100-package:LPC1768FBD100

LPC1768 simplified block diagram

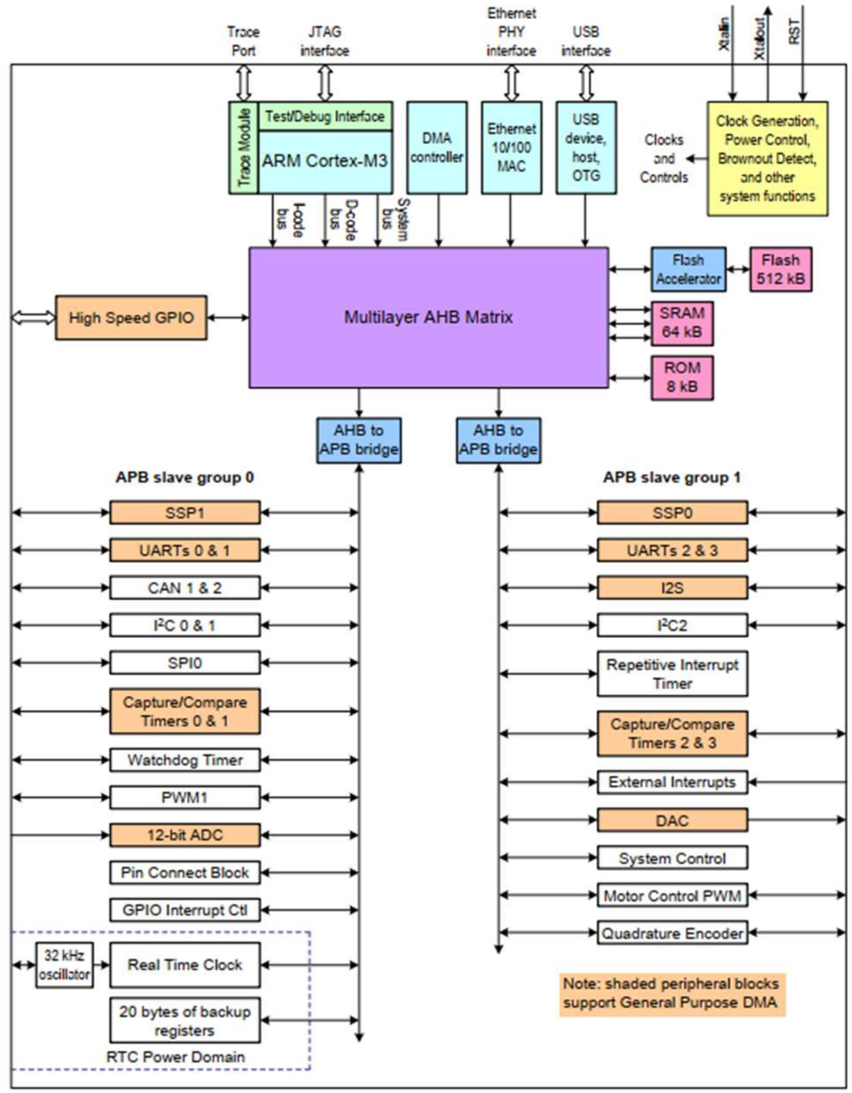
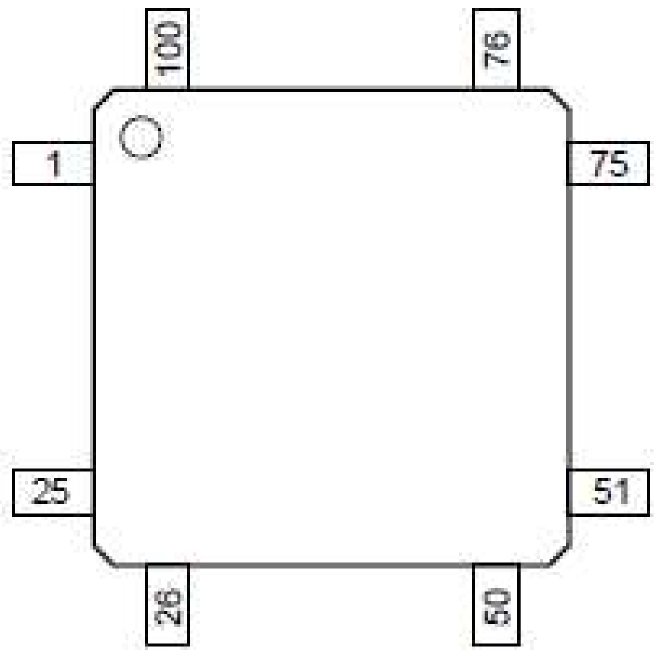


Fig 1. LPC1768 simplified block diagram

INPUT/OUTPUT (IO) PROGRAMMING

LPC1768 pin configuration

512 K Flash memory
64K SRAM
8K ROM
100 pin IC



What is IO programming?

- Through IO programming microcontroller can be used to control other devices such as sensors, displays, On-chip modules etc.
- It is done through GPIO (General purpose Input Output).

- GPIO is a pin on an IC (Integrated Circuit). It can be either input pin or output pin, whose behavior can be controlled at the run time. It's a standard interface used to connect microcontrollers to other electronic devices.
- In GPIO operation we perform general purpose operation to read from the port and to write to the port.

GPIO in LPC1768

- 5 general purpose bidirectional digital IO ports:
  - Port 0, Port 1, Port 2, Port 3 and Port 4.
  - Each port has 32 pins→ total 160 pins.
  - But only 100/160 pins are available
  - Supports fast GPIO i.e., enhanced /Accelerated Features
  - In addition Port 0 and Port 2 pins can provide a single interrupt

- Applications of GPIO
  - Driving LEDs/other indicators, Controlling off-chip devices, Sensing digital inputs

- Port 0: Here 28 pins can be used as GPIO
  - Pins 12, 13, 14 & 31 are not available
- Port 1: Here 24 pins can be used as GPIO
  - Pins 2, 3, 7, 6, 5, 11, 12, & 13 are not available
- Port 2: Here 14 pins can be used as GPIO
  - only pins 0 to 13 are available and rest are reserved
- Port 3 : Here 2 pins can be used as GPIO
  - only pins 25,26 are available and rest are reserved
- Port 4 : Here 2 pins can be used as GPIO
  - only 28,29 are available and rest are reserved
- In total 70 GPIO pins are available for the user.

Pin connect block

- How to identify/name the Pins?!!!
  - The naming convention for port pins is ‘**Px.y**’,
    - where ‘**x**’ is port number and ‘**y**’ is in number.
  - For example : **P0.7** refers to Pin number **7** of Port **0**
  - **P2.11** refers to Pin number **11** in Port **2**.

LPC1768 microcontroller

– 100 pin IC provided by NXP semiconductor

– Five ports, Port 0, Port 1, Port 2, Port3 and Port 4

– Each pin can have maximum four functions

Pin Connect Block Registers

- The pin connect block allows to have more than one function for a pin.
- Configuration registers control the multiplexers to allow connection between the pin and the on-chip peripherals.

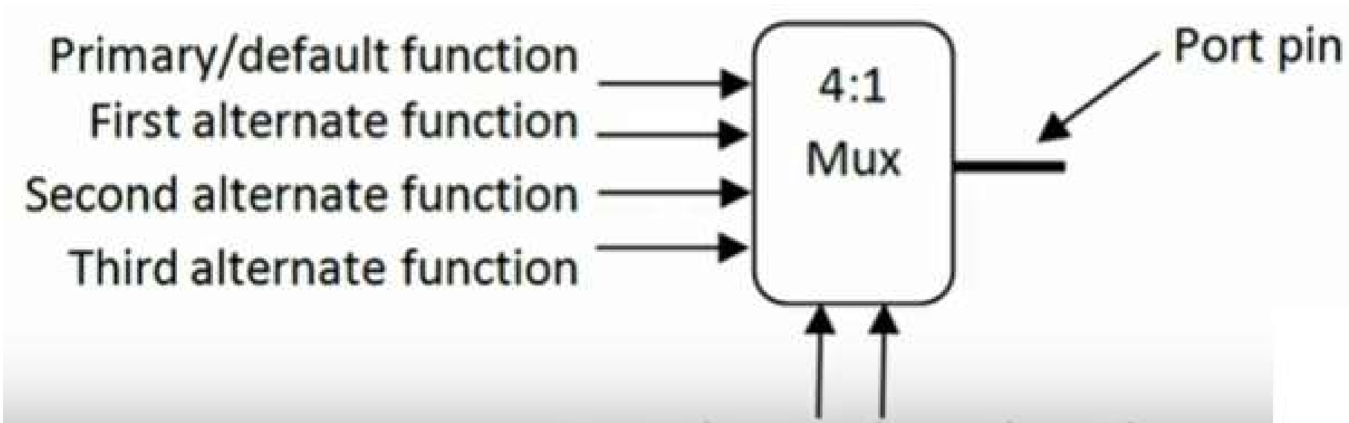
Pin function select registers: Total 9

- It controls the function of port pins
- Memory Mapped registers➔ We can access these registers using memory address
- Allows the connection b/w on-chip peripherals and pins
- Controls MUX like structure (To select a particular function)

Note: All registers are 32 wide

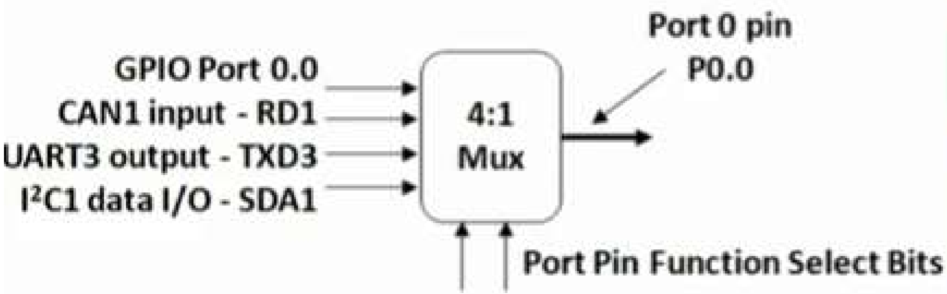
Summary of PINSEL registers	
Register	Controls
PINSEL0	P0[15:0]
PINSEL1	P0 [31:16]
PINSEL2	P1 [15:0] (Ethernet)
PINSEL3	P1 [31:16]
PINSEL4	P2 [15:0]
PINSEL5	P2 [31:16]
PINSEL6	P3 [15:0]
PINSEL7	P3 [31:16]
PINSEL8	P4 [15:0]
PINSEL9	P4 [31:16]
PINSEL10	Trace port enable

- WKT each pin can have 4 functions ➔ There are 4 inputs and 1 output
- ➔ 4:1 MUX structure
- ➔ There 2 select lines
- The select line of the MUX are from PINSEL registers. i.e., Two bits in each PINSEL register are used to control a single port pin.
- For example, Bits 0 and 1 in PINSEL0 are used to configure the functionality of P0.0 pin.



PINSEL

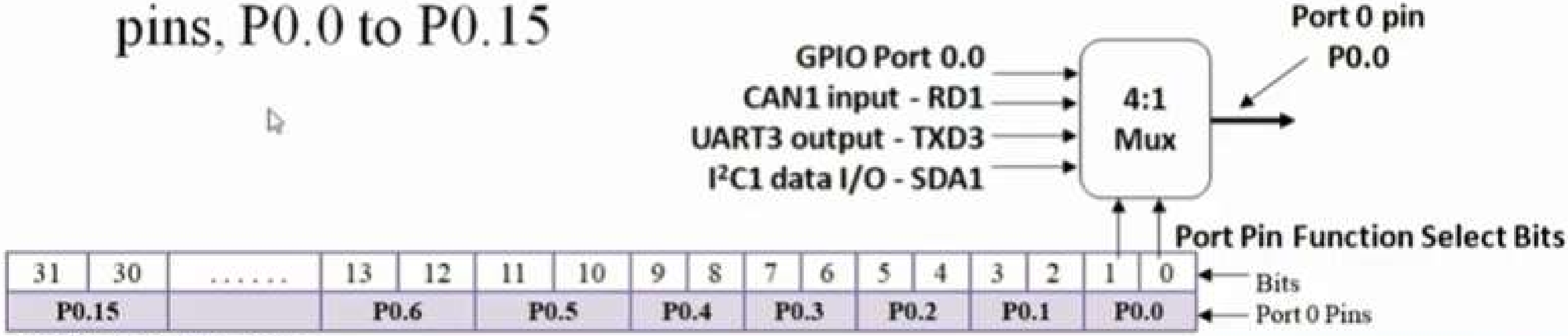
- Bit combinations to configure function of a port pin



Port Pin Function Select Bits	Function	Port0 pin P0.1 Function
00	Primary (default)	GPIO Port
01	First alternate	CAN1 – RD1
10	Second alternate	UART3 – TXD3
11	Third alternate	I^2C1 - SDA1

Example

- PINSEL0
  - 32 bit wide register controls the functions of the Port 0 pins, P0.0 to P0.15



PINSEL0 Register Format

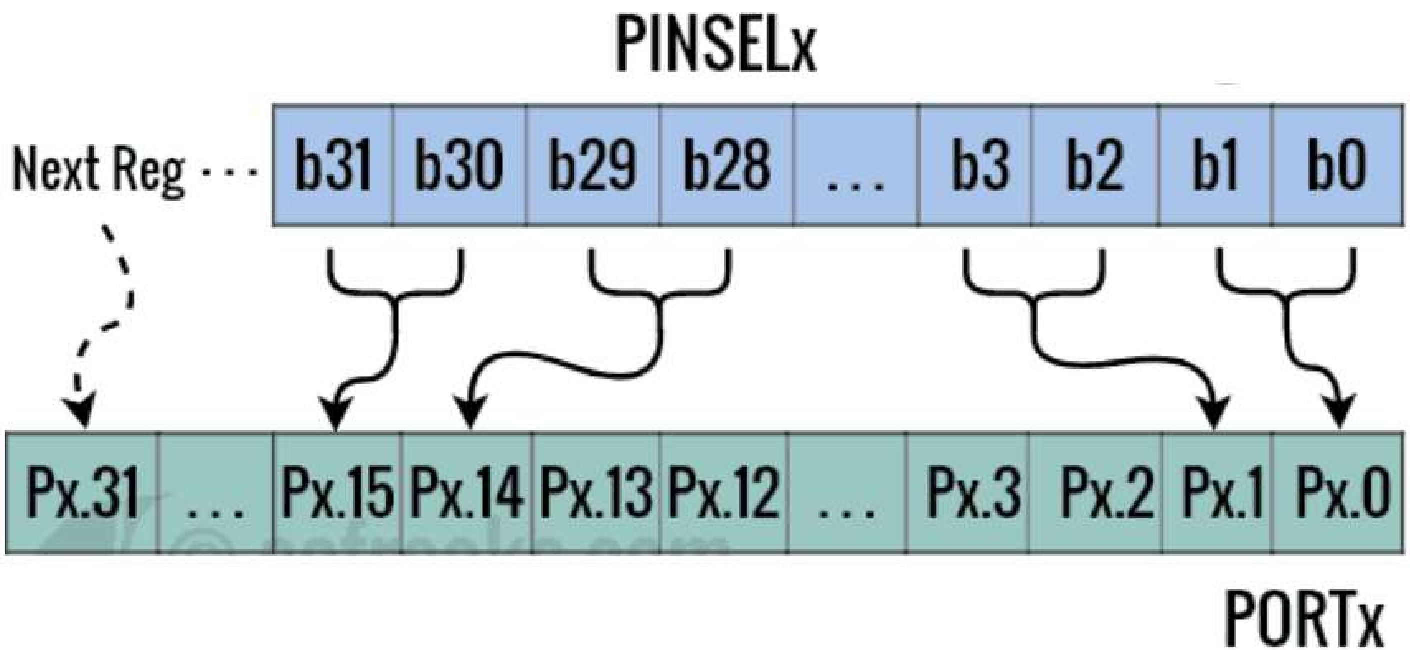
Table 80. Pin function select register 0 (PINSEL0 - address 0x4002 C000) bit description

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

- PINSEL1 - controls the functions of the Port 0 pins, P0.16 to P0.30
- PINSEL2 - controls the functions of the Port 1 pins, P1.0 to P1.15
- PINSEL3 - controls the functions of the Port 1 pins, P1.16 to P1.31
- PINSEL4 - controls the functions of the Port 2 pins, P2.0 to P2.13
- PINSEL7 - controls the functions of the Port 3 pins, P3.25 & P3.26
- PINSEL9 - controls the functions of the Port 4 pins, P4.28 & P4.29

- PINSEL10 - controls the Trace function on the Port 2 pins, P2.2 to P2.6

So the relation b/w PINSEL and port Pins can be shown as follows



GPIO Registers

- The ports are controlled by 5 registers.
  - (Note: PINSELx will only select the pins)
- These registers are present on Peripheral AHB bus (Advanced High performance Bus) for fast read/write timing.
- They can all be accessed in byte, half-word, and word sizes.
- Bit addressable register
- These GPIOs fast, hence the naming convention uses a prefix of “FIO” for all the registers related to GPIO.

1) FIODIR : This is the GPIO direction control register.
Here Bit 0➔ corresponding pin is Input.
Here Bit 1➔ corresponding pin is Output.

2) FIOMASK : This gives masking mechanism for any pin status it is used for Pin access control. Setting a bit to 0 means that the corresponding pin will be affected by changes to other registers like FIOPIN, FIOSET, FIOCLR. Writing a 1 means that the corresponding pin won't be affected by other registers.

3) FIOPIN : This register can be used to Read or Write values directly to the pins. Regardless of the direction set it gives the current status of the GPIO pin when read.

4) FIOSET : It is used to drive an 'output' configured pin to Logic 1 i.e HIGH. Writing Zero does NOT have any effect and hence it cannot be used to drive a pin to Logic 0 i.e LOW. For driving pins LOW FIOCLR is used which is explained below.

5) FIOCLR : It is used to drive an 'output' configured pin to Logic 0 i.e LOW. Writing Zero does NOT have any effect and hence it cannot be used to drive a pin to Logic 1.

Generic Name	Description	Access	Reset value[1]	PORTn Reg Name & Ad
FIODIR	Fast GPIO Port Direction control register. This register individually controls the direction of each port pin.	R/W	0	FIO0DIR - 0 FIO1DIR - 0 FIO2DIR - 0 FIO3DIR - 0 FIO4DIR - 0
FIOMASK	Fast Mask register for port. Writes, sets, clears, and reads to port (done via writes to FIOPIN, FIOSET, and FIOCLR, and reads of FIOPIN) alter or return only the bits enabled by zeros in this register.	R/W	0	FIO0MASK FIO1MASK FIO2MASK FIO3MASK FIO4MASK
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.	R/W	0	FIO0PIN - 0 FIO1PIN - 0 FIO2PIN - 0 FIO3PIN - 0 FIO4PIN - 0
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.	R/W	0	FIO0SET - (FIO1SET - (FIO2SET - (FIO3SET - (FIO4SET - (
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.	WO	0	FIO0CLR - (FIO1CLR - (FIO2CLR - (FIO3CLR - (FIO4CLR - (

Table 104. Fast GPIO port Direction register FIO0DIR to FIO4DIR - addresses 0x2009 C000 to 0x2009 C080) bit description

Bit	Symbol	Valu e	Description	Reset value
31:0	FIO0DIR FIO1DIR FIO2DIR FIO3DIR FIO4DIR		Fast GPIO Direction PORTx control bits. Bit 0 in FIOxDIR controls pin Px.0, bit 31 in FIOxDIR controls pin Px.31.	0x0
		0	Controlled pin is input.	
		1	Controlled pin is output.	

Represents Port No

Represents Pins of the Port No

GPIO Port direction control registers

- Word accessible FIOxDIR register (32 bit wide) controls the direction of each pin of Port x

Name	Description	Access	Reset value	Memory Address
FIO0DIR	Port 0 Direction control register	Read/Write	0x0000 0000	0x2009 C000
FIO1DIR	Port 1 Direction control register	Read/Write	0x0000 0000	0x2009 C020
FIO2DIR	Port 2 Direction control register	Read/Write	0x0000 0000	0x2009 C040
FIO3DIR	Port 3 Direction control register	Read/Write	0x0000 0000	0x2009 C060
FIO4DIR	Port 4 Direction control register	Read/Write	0x0000 0000	0x2009 C080

Memory mapped IO & Bit Addressable

GPIO Port direction control register

- WKT FIOxDIR is used to control/set the direction of registers
- To do that following steps needs to be followed (Note: these steps are Generalized)
  1) Logically AND the contents of FIOxDIR: To preserves directions of the other pins
     - Put 0's corresponding to the pins that needs to be changed and 1's for the rest
  2) Logically OR the result (from above step) with the desired direction value

- Let use see an example: We want to set P0.4 (Port 0, Pin4) as input

FIO0DIR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Pin
0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0	Predefined direction
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	AND
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	OR
0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	1	0	0	1	0	0	Final Value

FIO0DIR & = FFFF FFEF (AND)

FIO0DIR | = 0000 0000 (OR) // can be skipped

Set P0.5 as Output

FIO0DIR Register

Pin	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Predefined direction	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	0	1	0	1	0	0
AND	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
OR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	o	0	0	0	0
Final Value	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	1	1	0	1	0	0

FIO0DIR & = FFFF FFDF (AND) // can be skipped

FIO0DIR | = 0000 0020 (OR)

GPIO Port direction control register

- FIO0DIR - Selecting direction of Port0 pin P0.5 as output
    - FIO0DIR &= 0xFFFFFFDF;   //Can skip
    - FIO0DIR |= 0x00000020;

Ref:

https://www.youtube.com/watch?v=5ipqQsguqdE -- FIOxDIR

https://www.youtube.com/watch?v=WmZFfokztXg Pin Connect block

Chapter 8 and 9 in the user manual https://www.nxp.com/docs/en/user-guide/UM10360.pdf

GPIO port output Set register FIOxSET (FIO0SET to FIO4SET)

- This register is used to produce a HIGH level output at the port pins configured as GPIO in an OUTPUT mode.
- Writing 1 produces a HIGH level at the corresponding port pins.
- Writing 0 has no effect.
- If any pin is configured as an input or a secondary function, writing 1 to the corresponding bit in the FIOxSET has no effect.
- Access to a port pin via the FIOxSET register is conditioned by the corresponding bit of the FIOxMASK register

Bit	Symbol	Value	Description	Reset value
31:0	FIO0SET FIO1SET FIO2SET FIO3SET FIO4SET		Fast GPIO output value Set bits. Bit 0 in FIOxSET controls pin Px.0, bit 31 in FIOxSET controls pin Px.31.	0x0
		0	Controlled pin output is unchanged.	
		1	Controlled pin output is set to HIGH.	

GPIO port output Clear register FIOxCLR (FIO0CLR to FIO4CLR)

- This register is used to produce a LOW level output at port pins configured as GPIO in an OUTPUT mode.
- Writing 1 produces a LOW level at the corresponding port pin and clears the corresponding bit in the FIOxCLR register.
- Writing 0 has no effect. If any pin is configured as an input or a secondary function, writing to FIOxCLR has no effect.
- Access to a port pin via the FIOxCLR register is conditioned by the corresponding bit of the FIOxMASK register

Table 108. Fast GPIO port output Clear register (FIO0CLR to FIO4CLR- addresses 0x2009 C01C to 0x2009 C09C) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FIO0CLR FIO1CLR FIO2CLR FIO3CLR FIO4CLR		Fast GPIO output value Clear bits. Bit 0 in FIOxCLR controls pin Px.0, bit 31 controls pin Px.31.	0x0
		0	Controlled pin output is unchanged.	
		1	Controlled pin output is set to LOW.	

GPIO port Pin value register FIOxPIN (FIO0PIN to FIO4PIN)

- The FIOxPIN register is made up of individual bits, and each bit corresponds to a specific pin on the port. It can be configured as either input or output.
- **Reading(i/p):** When you read the FIOxPIN register, you get a value where each bit reflects the current state of the corresponding pin.
- **Writing (o/p):** If the pin is configured as an output, you can write to the FIOxPIN register to change the state of that pin. Setting a bit to 1 makes the pin high, and setting it to 0 makes it low.
- Access to a port pin via the FIOxPIN register is conditioned by the corresponding bit of the FIOxMASK register
- Only pins masked with zeros in the Mask register will be correlated to the current content of the Fast GPIO port pin value register.

Fast GPIO port Mask register FIOxMASK (FIO0MASK to FIO4MASK)

- This register is used to select port pins that will and will not be affected by write accesses to the FIOxPIN, FIOxSET or FIOxCLR register. Mask register also filters out port’s content when the FIOxPIN register is read.
- A zero in this register’s bit enables an access to the corresponding physical pin via a read or write access.
- If a bit in this register is one, corresponding pin will not be changed with write access and if read, will not be reflected in the updated FIOxPIN register.

Bit	Symbol	Value	Description	Reset value
31:0	FIO0MASK FIO1MASK FIO2MASK FIO3MASK FIO4MASK		Fast GPIO physical pin access control.	0x0
		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 port output Set register FIOxSET (FIO0SET to FIO4SET)

Example #1)

Configure Pin 4 of Port 0 i.e P0.4 as Output and SET it High (Logic 1)

LPC_GPIO0->FIODIR = 0x0000 0010; // in 32 bit form 0000 0000 0000 0000 0000 0000 0001 0000; 4th bit is 1

Port Number

Pin number

Alternate way, is

LPC_GPIO0->FIODIR = (1<< 4); // 1 in 32 bit binary form left shift by 4 times.
0000 0000 0000 0000 0000 0000 0000 0001 left shit by 4 times results as
0000 0000 0000 0000 0000 0000 0001 0000. which Configures P0.4 as Ouput

LPC_GPIO0->FIOSET = (1<< 4); // SET High for P0.4 pin

LPC_GPIO0->FIODIR = 0x0000 0010; // omitting left side zeros ; we can write as 0x10 ; x stands for Hexadecimal number

LPC_GPIO0->FIODIR = 0x10; // Configure pin 4 of port 0 as Output

Lets say we want to set **PIN 3** on **Port 0 as output**. It can be done in following ways:

CASE 1. LPC_GPIO0->FIODIR = (1<<3); //(binary using left shift - direct assign: other pins set to 0)
CASE 2. LPC_GPIO0->FIODIR |= 0x0000008; // or 0x8; (hexadecimal - OR and assign: other pins not affected)
CASE 3. LPC_GPIO0->FIODIR |= (1<<3); //(binary using left shift - OR and assign: other pins not affected)

- Case 1 must be avoided since we are directly assigning a value to the register. So, while we are making P0.3 as '1' others are forced to be assigned a '0' which can be avoided by **ORing** and then assigning Value.
- Case 2 can be used when bits need to be changed in bulk and
- Case 3 when some or single bit needs to be changed.

Example #1)
Configure Pin 4 of Port 0 i.e P0.4 as Ouput and drive it High(Logic 1).

```
LPC_GPIO0->FIODIR |= (1<<4); // Config P0.4 as Output
LPC_GPIO0->FIOSET |= (1<<4); // Make output High for P0.4
```

Example #2)
Making output configured Pin 17 High of Port 0 i.e P0.17 and then Low

```
LPC_GPIO0->FIODIR |= (1<<17); // P0.17 is Output pin
LPC_GPIO0->FIOSET |= (1<<17); // Output for P0.17 becomes High
LPC_GPIO0->FIOCLR |= (1<<17); // Output for P0.17 becomes Low
```

Example #3)
Configuring 1st 8 Pins of Port 0 (P0.0 to P0.7) as Output and Setting them High:

```
LPC_GPIO0->FIODIR |= 0xFF; // Config P0.0 to P0.7 as Output
LPC_GPIO0->FIOSET |= 0xFF; // Make output High for P0.0 to P0.7
```

Example #4)
Configuring P0.5 and P0.11 as Output and Setting them High:

```
LPC_GPIO0->FIODIR |= (1<<5) | (1<<11);     // Config P0.5 and P0.11 as Ouput
LPC_GPIO0->FIOSET |= (1<<5) | (1<<11);     // Make ouput High for P0.5 and P0.11
```

Logical OR

```
0000 0000 0000 0000 0000 0000 0010 0000  [ 1<<5]
0000 0000 0000 0000 0000 1000 0000 0000  [ 1<<11]
-------------------------------------------------------------
0000 0000 0000 0000 0000 1000 0010 0000
```

GPIO port output Clear register FIOxCLR (FIO0CLR to FIO4CLR)

Configure Pin 17 of Port 1 [P1.17] as output and then SET that Pin and CLEAR it:

```
LPC_GPIO1->FIODIR  = (1<<17);        // configuring 17th pin of port 1 [P1.17] as Output pin
LPC_GPIO1->FIOSET  = (1<<17);        // Output for P1.17 becomes High
LPC_GPIO1->FIOCLR = (1<<17);         // Output for P1.17 becomes Low
```

Example #3) Configure Pin 3 and 5 of port 0 as output and SET them Logic High.

i.e Configuring P0.3 and P0.5 as Output and Setting them High:

LPC_GPIO0->FIODIR = (1<<3) | (1<<5) // 1 left shift 3 logically ORed with 1 left shift by 5

	0000 0000 0000 0000 0000 0000 0000 1000	**[1 << 3]**
LOGIC OR with	**0000 0000 0000 0000 0000 0000 0010 0000**	**[1 << 5]**
	0000 0000 0000 0000 0000 0000 0010 1000	**3rd and 5th bits are 1. So P0.3 and P0.5 configures as output**

Alternate way is

```
LPC_GPIO0->FIODIR =0x0000 0028;      // grouping above terms or we can also write as 0x28
LPC_GPIO0->FIOSET = (1<<3) | (1<<5)            // Make ouput High for P0.3 and P0.5
```

Example #5)

Configuring Pins of Port 0 (P0.4 to P0.11) as Output and Setting them High: and Low

```
LPC_GPIO0->FIODIR = 0xFF0;      // Config P0.4 to P0.11 as Ouput. 0000 0000 0000 0000 0000 1111 1111 0000
LPC_GPIO0->FIOSET = 0xFF0;                     //  1111 1111 0000  Make output High for P0.4 to P0.11
LPC_GPIO0->FIOCLR = 0xFF0;                     // Make output LOW for P0.4 to P0.11
```

Example #6) Configuring Pins of Port 1 (P1.14 to P0.21) as Input :

```
LPC_GPIO1->FIODIR &= ~(0xFF0);                       // Config P1.14 to P1.21 as Input. ~ means complement
LPC_GPIO1->FIODIR &=~(1<<14);                  // only Pin p1.14 is configured as input
```

Note: On reset by default all registers are zero. So all port's all the pins acts as input. We can skip setting i/p lines

GPIO port FIOxPIN

- Example :
  - Produce **HIGH** logic on pin **0, 1, 2, 3** of port 0
  - Produce **LOW** logic on pin **4, 5, 6, 7** of port 0

LOW HIGH

FIO0PIN

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	1	1	1

- **OR** operation for logic level **HIGH**
- **AND** operation for logic level **LOW**
- FIO0PIN |= 0x0000000F;
- FIO0PIN &= 0xFFFFFF0F;

Write an embedded c program to turn on and off the led connected to P0.2 in LPC1768

```
#include <lpc17xx.h>
void delay_ms(unsigned int ms);
int main(void) {
   LPC_PINCON->PINSEL0 &= FFFFFFCF ; // Configure P0.2 as GPIO
   LPC_GPIO0->FIODIR |= (1 << 2) ; // Configure P0.2 as output
   while (1) {
   LPC_GPIO0->FIOSET = (1 << 2) ; // Turn ON LED (Set P0.2 to HIGH)
   delay_ms(500);  // Random Delay

   LPC_GPIO0->FIOCLR = (1 << 2) ; // Turn OFF LED (Clear P0.2 to LOW)
   delay_ms(500);  // Random Delay
   }
}
void delay_ms(unsigned int ms) {
   unsigned int i, j;
   for (i = 0; i < ms; i++) {
      for (j = 0; j < 4000; j++);  // Approximate delay
   }
}
```

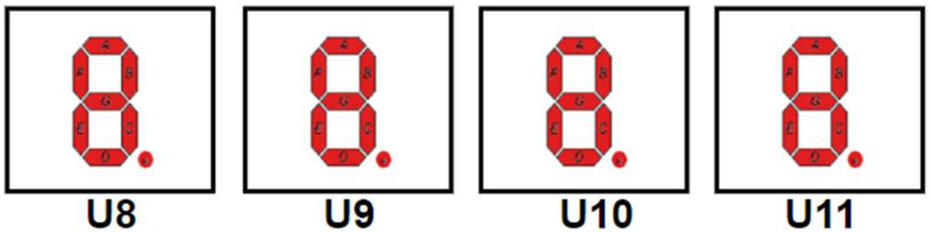
Write an embedded c program for BCD up counter using LEDs connected to P1.0 – P1.3 in LPC1768

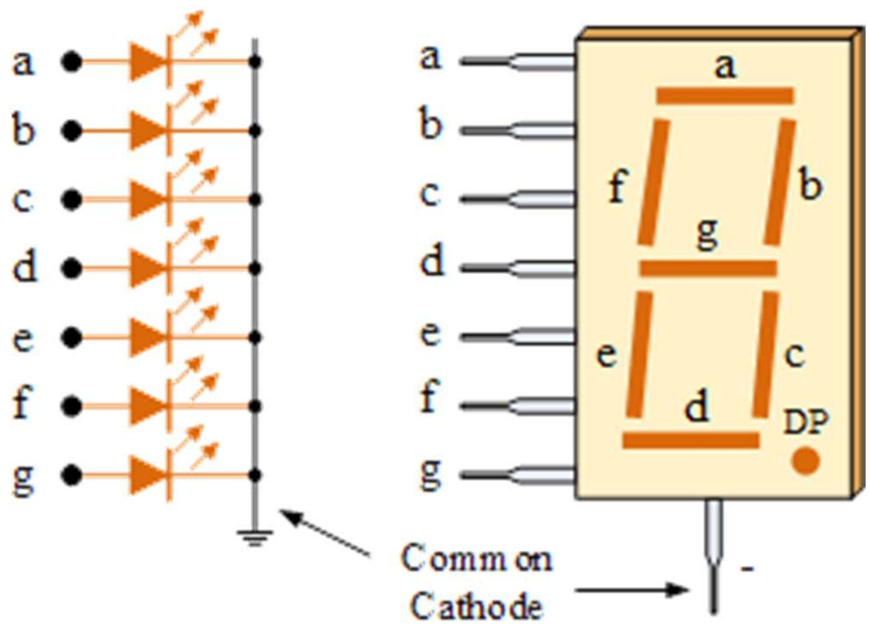
```
#include <lpc17xx.h>  // Include LPC1768 header file
#define LED  0x0F  // Mask for P1.0 to P1.3 (4-bit BCD)
void delay_ms(unsigned int ms);

int main(void) {
   unsigned int count = 0;
   // Configure P1.0 to P1.3 as output
   LPC_GPIO1->FIODIR |= LED;
   while (1) {
      // Clear previous value and set new BCD value
      LPC_GPIO1->FIOCLR = LED;  // Turn off all LEDs
      LPC_GPIO1->FIOSET = (count & LED);  // Set new BCD value
      delay_ms(1000);  // Delay of 1 second
      count++;  // Increment counter
      if (count > 9) {
         count = 0;  // Reset count after 9
      }
   }
}
```

```
void delay_ms(unsigned int ms) {
   unsigned int i, j;
   for (i = 0; i < ms; i++) {
      for (j = 0; j < 4000; j++);  // Approximate delay for 1ms
   }
}
```

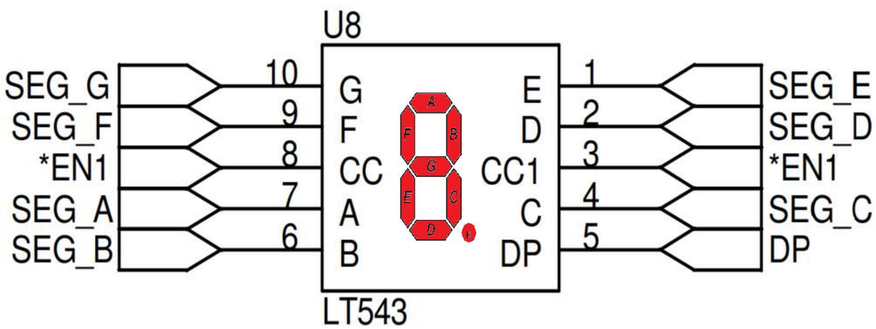
Seven segment display :

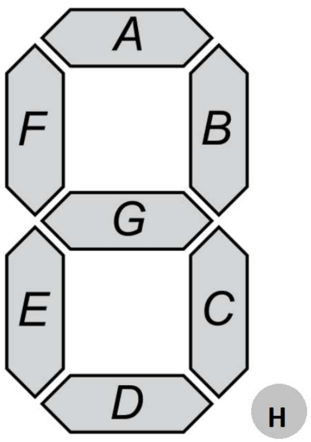
- There are four multiplexed 7-segment display units (**U8, U9, U10 and U11**) on the board.
- Each display has 8-inputs (Diodes).
    - SEG_A, SEG_B, SEG_C, SEG_D, SEG_E, SEG_F, SEG_G and SEG_H (DP) and Common Cathode CC.





- Each display has 8-inputs (Diodes).
  - SEG_A (Pin-7), SEG_B (Pin-6), SEG_C (Pin-4), SEG_D (Pin-2), SEG_E (Pin-1), SEG_F (Pin-9), SEG_G (Pin-10) and SEG_H (Pin-5) and the remaining pins pin-3 & pin-8 are Common Cathode CC.





	h	g	f	e	d	c	b	a	hex value
0	0	0	1	1	1	1	1	1	3F
1	0	0	0	0	0	1	1	0	06
2	0	1	0	1	1	0	1	1	5B
3	0	1	0	0	1	1	1	1	4F
4	0	1	1	0	0	1	1	0	66
5	0	1	1	0	1	1	0	1	6D
6	0	1	1	1	1	1	0	1	7D
7	0	0	0	0	0	1	1	1	07
8	0	1	1	1	1	1	1	1	7F
9	0	1	1	0	1	1	1	1	6F
A	0	1	1	1	0	1	1	1	77
b	0	1	1	1	1	1	0	0	7C
C	0	0	1	1	1	0	0	1	39
d	0	1	0	1	1	1	1	0	5E
E	0	1	1	1	1	0	0	1	79
F	0	1	1	1	0	0	0	1	71

- As there are four multiplexed 7-segment displays, A 1-of-10 Decoder/Driver (U7) is used to select/enable one of the 7-segment displays
  - Note: The 1-of-10 Decoder/Driver is a 4:16 decoder only. Here only 10/16 outputs are utilized, hence the name 1-of-10. However, in our MCU kit only 4/10 outputs are utilized.