I2C Protocol

What is I²C?

I²C is a **two-wire** communication protocol used for connecting low-speed peripherals to microcontrollers. It uses:

- **SCL (Serial Clock Line)** – clock signal generated by master.
- **SDA (Serial Data Line)** – bidirectional data line.

Key Features:

- Supports multiple masters and slaves.
- Each device has a unique address.
- Data is transferred in 8-bit format with ACK/NACK bits.
- Speeds: 100 kHz (Standard), 400 kHz (Fast), up to 3.4 MHz (High-speed).

I²C Pins on LPC1768:

The LPC1768 has multiple I2C interfaces:

- **I2C0**: P0.27 (SDA0), P0.28 (SCL0)
- **I2C1**: P0.0 (SDA1), P0.1 (SCL1)
- **I2C2**: P0.10 (SDA2), P0.11 (SCL2)

You can choose any one based on your requirement.

Implement **I2C communication** between the **LPC1768 microcontroller** and an **EEPROM** to:

1. **Write a single byte 'A'** to EEPROM at memory location 0x80.
2. **Read the same byte** back from that location.
3. **Display the read character** on an LCD.

In this I2C communication example using LPC1768, the **two devices involved** are:

1. Master Device:

→ LPC1768 Microcontroller (I2C0 peripheral)

- It initiates the communication.
- Sends START/STOP conditions, addresses, and data.
- Controls the clock (SCL) and data (SDA) lines.

2. Slave Device:

→ EEPROM Chip (e.g., 24C02/24C08/24C256, etc.)

- It responds to the master's address.
- Stores and provides data on request.
- Communicates only when addressed by the master.

I2C communication registers

1. LPC_SC->PCONP

Purpose:

Enables power to peripherals. If this bit isn't set, the I2C block will not work even if the pins and registers are configured.

Register: PCONP = Power Control for Peripherals

```
LPC_SC->PCONP |= (1 << 7); // Powerup I2C0
```

A: i2c_config();

2. LPC_PINCON->PINSEL1

Purpose: Selects alternate function for pins.

Use in Code:

```
LPC_PINCON->PINSEL1 = (1 << 22); // P0.27 as SDA0
LPC_PINCON->PINSEL1 |= (1 << 24); // P0.28 as SCL0
```

Pin	Bit Range	Setting	Function
P0.27	Bits 23:22	01	SDA0 (I2C Data)
P0.28	Bits 25:24	01	SCL0 (I2C Clock)

LPC_I2C0->I2SCLH and LPC_I2C0->I2SCLL

These two registers **set the I2C clock rate** — that is, how fast data is transferred over the I2C bus.

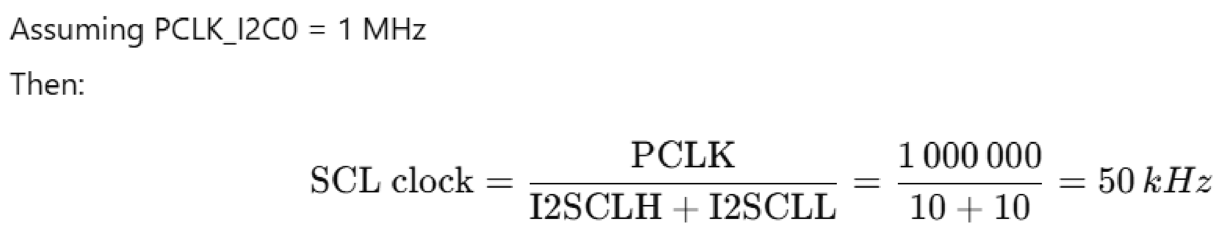
Register Roles:

Register	Purpose
I2SCLH	Sets HIGH time of SCL clock
I2SCLL	Sets LOW time of SCL clock

Total Clock Period = I2SCLH + I2SCLL

The values are counts of **PCLK_I2C0 cycles**.

```
LPC_I2C0->I2SCLH = 10;
LPC_I2C0->I2SCLL = 10;
```



Assuming PCLK_I2C0 = 1 MHz

Then:

$$\text{SCL clock} = \frac{\text{PCLK}}{\text{I2SCLH} + \text{I2SCLL}} = \frac{1\,000\,000}{10 + 10} = 50\,kHz$$

Which is a **safe speed** for EEPROMs and short-distance I2C devices.

4. LPC_I2C0->I2CONSET

This register is used to **set (enable) control bits** in the I2C controller. It **does not clear** bits — use I2CONCLR for that.

Bits Used in Your Code:

Bit	Mask	Meaning	In Your Code
6	(1<<6)	**I2EN** – I2C interface enable	Used in i2c_config()
5	(1<<5)	**STA** – Send START condition	Used in i2c_start()
4	(1<<4)	**STO** – Send STOP condition	Used in i2c_stop()
2	(1<<2)	**AA** – Assert ACK bit	Used in i2c_mem_read()
3	(1<<3)	**SI** – Interrupt flag	(status bit, not set manually)

B: LCD Initialization

```
lcd_init();  // Initializes LCD
```

C: Writing to EEPROM: i2c_start();

```
LPC_I2C0->I2CONSET = (1 << 5);  // STA: Set START condition
while((ic->I2CONSET & (1 << 3)) == 0);  // Waits until the START is transmitted and interrupt is generated.
LPC_I2C0->I2CONCLR = (1 << 5);  // Clear START bit
LPC_I2C0->I2CONCLR = (1 << 3);  // Clear interrupt (SI) bit
```

D: Write Slave Addr, Memory Addr, Data

```
i2c_mem_write(0xA0);  // SLA+W (write mode)

LPC_I2C0->I2DAT = d;  // Load data (slave addr, memory addr or data byte)
while((LPC_I2C0->I2CONSET & (1 << 3)) == 0);  // Wait for transmission
LPC_I2C0->I2CONCLR = (1 << 3);  // Clear SI

LPC_I2C0_mem_write(0x80);  // Memory address in EEPROM
LPC_I2C0_mem_write('A');   // Actual data byte
```

E: Stop Condition

```
i2c_stop();
LPC_I2C0->I2CONSET = (1 << 4);  // STO: Send STOP condition
while((ic->I2CONSET & (1 << 4)) == 1);  // Wait until stop bit clears
```

Stops I2C communication.

F: Reading from EEPROM

```
i2c_start();  // START
i2c_mem_write(0xA0);  // SLA+W
i2c_mem_write(0x80);  // Memory address
```

G: Repeat Start for Read

```
i2c_start();          // START again (repeated)
i2c_mem_write(0xA1);  // SLA+R
```

Repeated START without STOP. EEPROM expects this for reading.

H: Read Byte

```
dat = i2c_mem_read(0);
```

Inside i2c_mem_read(0):

```
// If ack == 0, send NACK
LPC_I2C0->I2CONCLR = (1 << 2);  // Clear AA bit (send NACK)

while((LPC_I2C0->I2CONSET & (1 << 3)) == 0);  // Wait for data (SI)

val = LPC_I2C0->I2DAT;  // Read received data byte

LPC_I2C0                    ->I2CONCLR = (1 << 3);  // Clear SI flag
```

I: Stop

```
i2c_stop();  // Again stops communication after read
```

Summary of Registers Used

Register	Function
PINSEL1	Select SDA and SCL functions (P0.27, P0.28)
I2SCLH, I2SCLL	Set I2C clock frequency
I2CONSET	Enable I2C, send START, STOP, ACK, and check SI
I2CONCLR	Clear START, STOP, SI, ACK flags
I2DAT	Load or read 8-bit data for I2C transmission

I²C Address Table for EEPROM

Purpose	Address Sent (Binary)	Hex Value	Explanation
Write Operation	1010 0000	0xA0	Slave Address: 0x50 (7-bit) + W=0 bit
Read Operation	1010 0001	0xA1	Slave Address: 0x50 (7-bit) + R=1 bit