I2C PROTOCOL

PROTOCOL OVERVIEW

- I^2C (Inter-Integrated Circuit) is a synchronous, multi-master/multi-slave, single ended, serial communication bus

- It is widely used for attaching lower-speed peripheral integrated circuits (ICs) to processors and microcontrollers.

- I^2C uses only two signals: serial data line (SDA) and serial clock line (SCL) Serial clock (SCL) – RC3/SCK/SCL • Serial data (SDA) – RC4/SDI/SDA .

- The I^2C reference design has a 7-bit address space, with a rarely used 10-bit extension

- Common I^2C bus speeds are the 100 kbit/s standard mode and the 400 kbit/s fast mode. There is also a 10 kbit/s low-speed mode

- Later revisions of I^{2}C can host more nodes and run at faster speeds .

Different I^{2}C Modes	
I^{2}C Mode	Maximum Bit Rate
Standard-mode	100kbps
Fast-mode	400kbps
Fast-mode Plus	1Mbps
High-speed mode	3.4Mbps
Ultra-Fast mode	5Mbps

- The bus has two roles for nodes:

- Controller (master) node: Node that generates the clock and initiates communication with targets (slaves).

- Target (slave) node: Node that receives the clock and responds when addressed by the controller (master).

PRACTICAL APPLICATION
Messaging example: 24C32 EEPROM

- These EEPROMs use two address bytes (High and Low) to specify memory locations

- Write Operation: After a START, the master sends the device address (write), the two address bytes, and the data bytes (within the same 32-byte page), then issues STOP. The EEPROM becomes busy while storing data and won't respond during this time.

- Read Operation: The master sends the device address (write) and two address bytes, then a repeated START with the read bit. The EEPROM returns sequential data bytes.The master ACKs each byte, sends NACK after the last, and then STOP.

KEY FEATURES:

- Half-duplex Communication Protocol - Bi-directional communication is possible but not simultaneously.

- Synchronous Communication - The data is transferred in the form of frames or blocks ,Can be configured in a multi-master configuration.

- Clock Stretching - Slave can hold the SCL line low to delay the master if it's not ready, pausing data transfer until ready.

- Arbitration - In multi-master setups, if a master detects SDA high when it should be low, it stops transmission, allowing the other master to continue.

- Serial transmission - I2C uses serial transmission for transmission of data.

- Used for low-speed communication.

REGISTERS ON PIC16F877A:

The MSSP module has six registers for I2C operation.
1. *MSSP Control Register (SSPCON)*
2. *MSSP Control Register 2 (SSPCON2)*
3. *MSSP Status Register (SSPSTAT)*
4. *Serial Receive/Transmit Buffer Register (SSPBUF)*
5. *MSSP Shift Register (SSPSR) – Not directly accessible*
6. *MSSP Address Register (SSPADD)*

- SSPCON, SSPCON2 and SSPSTAT are the control and status registers in I2C mode operation.

- SSPSR is the shift register used for shifting data in or out.

- SSPBUF is the buffer register to which data bytes are written to or read from.

- SSPADD register holds the slave device address when the SSP is configured in I2C Slave mode.

- In receive operations, SSPSR and SSPBUF together create a double-buffered receiver.

SSPSTAT: MSSP STATUS REGISTER

- BIT 7 -   SMP: Slew Rate Control bit
- BIT 6 - CKE: SMBus Select bit
- BIT 5 -   D/A : Data/Address bit
- BIT 4 -   P: Stop bit
- BIT 3 -   S: Start bit
- BIT 2 -   R/W : Read/Write
- BIT 1 -   UA: Update Address(10-bit Slave mode only)
- BIT 0 - BF: Buffer Full Status bit

SSPCON1: MSSP CONTROL REGISTER 1

- BIT 7 - WCOL: Write Collision Detect bit
- BIT 6 - SSPOV: Receive Overflow Indicator bit
- BIT 5 - SSPEN: Synchronous Serial Port Enable bit
- BIT 4 - CKP: SCK Release Control bit
- BIT (3 - 0) - SSPM3:SSPM0: Synchronous Serial Port Mode Select bits
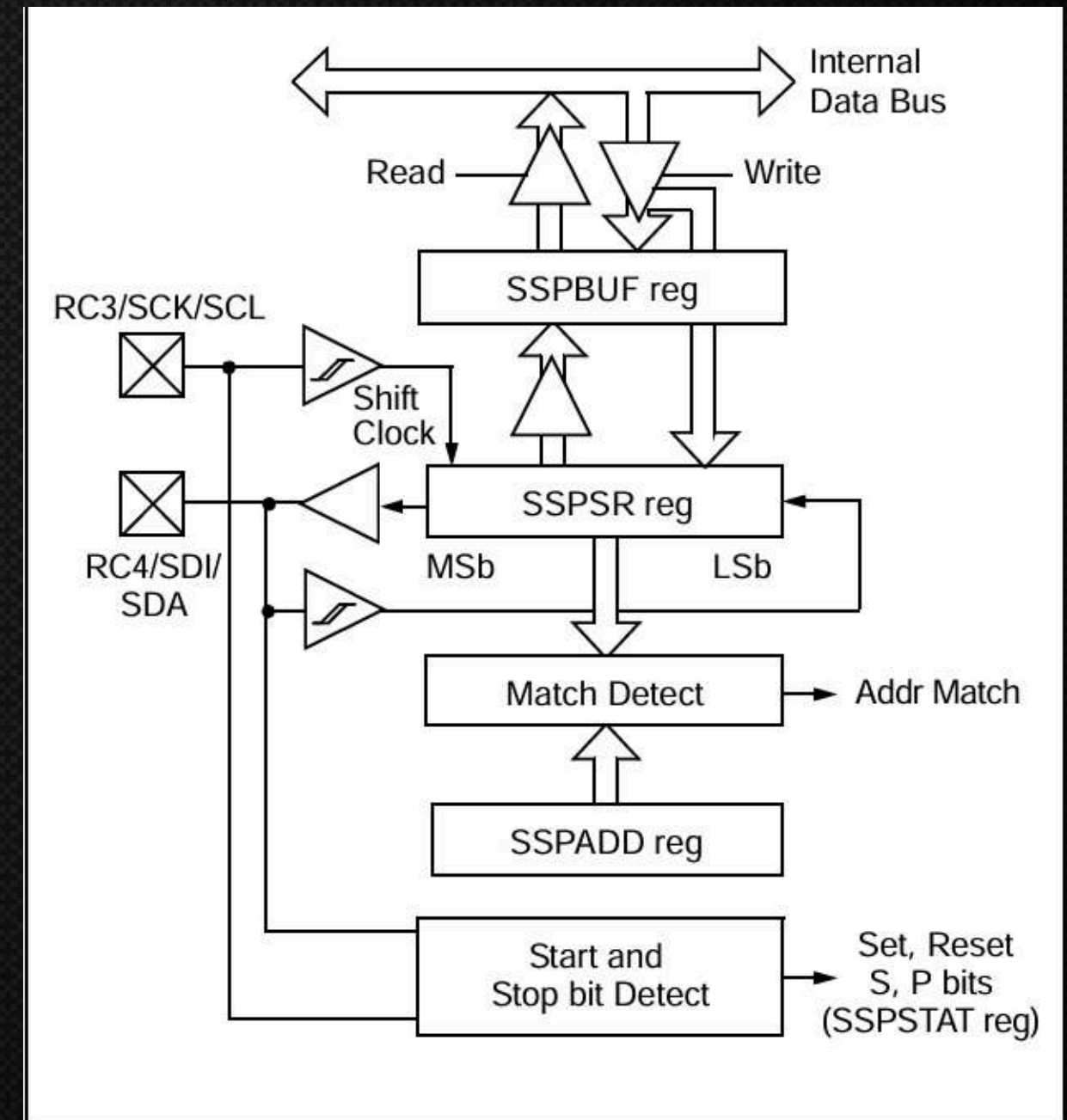
BLOCK DIAGRAM

SSPCON2: MSSP CONTROL REGISTER 2 :

- BIT 7 -   GCEN: General Call Enable bit (Slave mode only)

- BIT 6 -   ACKSTAT: Acknowledge Status bit (Master Transmit mode only)

- BIT 5 -   ACKDT: Acknowledge Data bit (Master Receive mode only)

- BIT 4 -   ACKEN: Acknowledge Sequence Enable bit (Master Receive mode only)

- BIT 3 -   RCEN: Receive Enable bit (Master mode only)

- BIT 2 -   PEN: Stop Condition Enable bit (Master mode only)

- BIT 1 -   RSEN: Repeated Start Condition Enabled bit (Master mode only)

- BIT 0 - SEN: Start Condition Enabled/Stretch Enabled bit

PIN CONFIGURATION:

- The I2C is a two-wire interface communication that comes with two main lines known as SDA and SCL

- The pin layout on the PIC16F877A is as follows:
1. Serial clock (SCL) – RC3/SCK/SCL
2. Serial data (SDA) – RC4/SDI/SDA

- SDA is serial data line that carries the data

- SCL is serial clock line that is used to synchronize all data transfers over the I2C bus
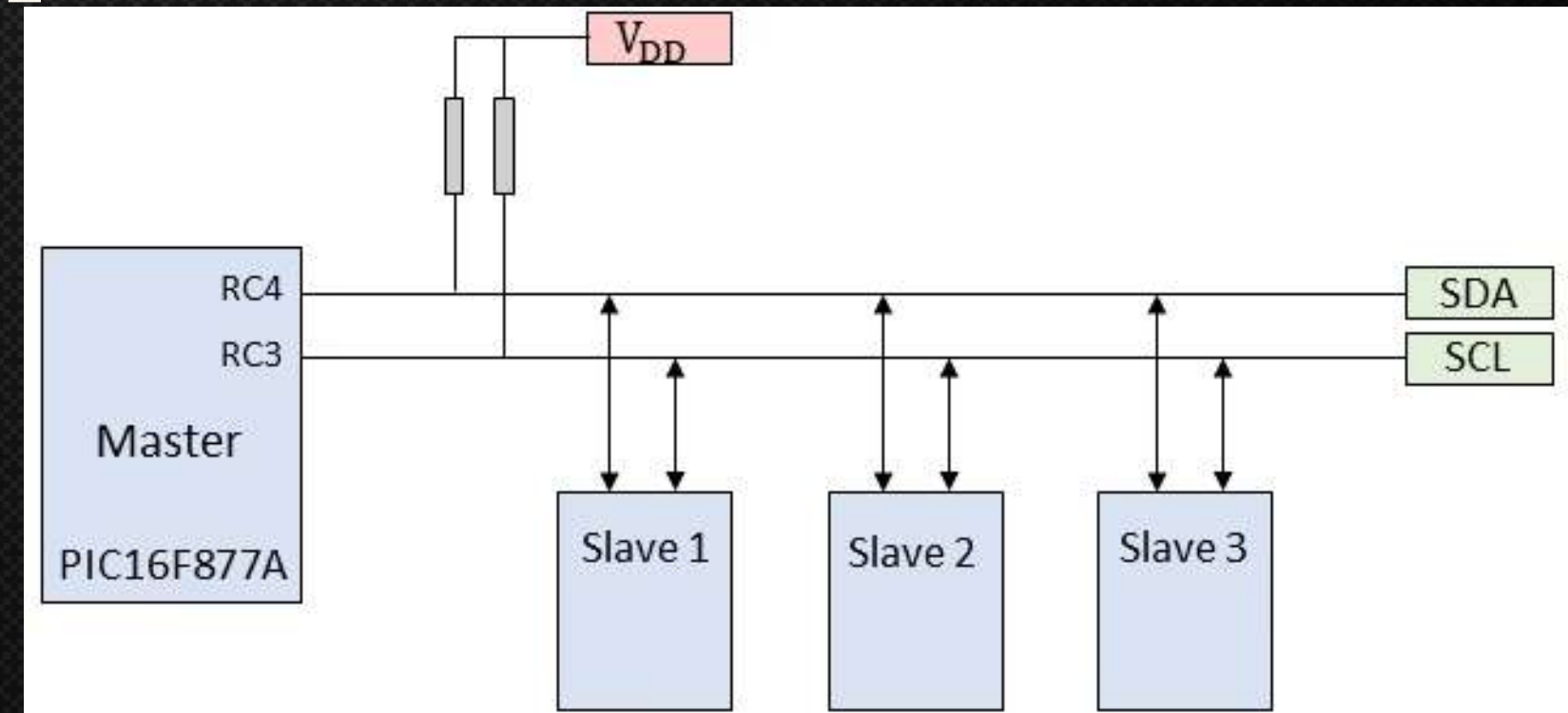
KEY ASPECTS OF I2C COMMUNICATION:

- Voltage Levels: I2C supports 5V and 3.3V levels. All devices on the bus must have compatible voltage levels.

- Open-Drain/Collector Outputs: I2C devices can only pull the line low (sink current), not drive it high, which is why pull-up resistors are required.

- Pull-Up Resistors: Used on SDA and SCL lines to keep them high when no device pulls them low.

HARDWARE REQUIREMENTS:

- Pull-Up Resistors:
1. The resistor range for SDA and SCL is limited by load and rise/fall time specs.

2. Minimum value: Set by the max current the device can handle – 3 mA (standard/fast mode) or 20 mA (fast mode+).

- Overcoming Capacitance Limits:

 If total bus capacitance exceeds 400 pF, use I2C multiplexers, switches, or buffers to split or isolate devices, reducing load and maintaining proper signal timing.

I²C Initialization Steps:

1. Enable I²C Peripheral:
   - Turn ON I²C module (SSPEN=1, PE=1) to activate SDA and SCL lines.

2. Select I²C Mode:
   - Configure control register for Master or Slave mode (SSPCON, I2C_CR1).

3. Set Clock Frequency:
   - For Master only: define SCL speed via baud register (SSPADD, I2C_CCR).
   - Common rates: 100 kHz (Standard), 400 kHz (Fast).

4.Configure SDA & SCL Pins:

Set as inputs (open-drain) and connect external pull-ups (4.7k–10k Ω).

5.Load Slave Address:

Assign 7-bit or 10-bit address in address register (SSPADD, OAR1).

6.Enable Acknowledge (ACK):

Enable ACK bit (ACKEN=1, ACK=1) for data acknowledgment.

7.Clear Status Flags:

Reset interrupt, buffer, and collision flags (SSPIF, BF, WCOL).

8.Enable Interrupts (Optional):

Activate I²C event/error interrupts (SSPIE=1, BCLIE=1).

9.Generate Start Condition:

Master sets start bit (SEN=1) to begin transmission.

10.Ready for Communication:

Exchange address + data frames; terminate with Stop Condition (P).

I²C Configuration Steps – PIC16F877A (Master Mode):

1. Set Up I²C Pins

```
TRISC3 = 1;    // SCL pin as input (RC3)
TRISC4 = 1;    // SDA pin as input (RC4)
```

- SDA and SCL must be inputs because I²C uses open-drain configuration.

2. Configure I²C Control Register (SSPCON)

```
SSPCON = 0b00101000;    // SSPEN=1 (Enable I²C), Master mode (Fosc / (4*(SSPADD+1)))
```

- Bit5 (SSPEN) → Enables I²C pins
- Bits3:0 → Select Master mode

3. Configure I²C Status Register (SSPSTAT)

```
SSPSTAT = 0b10000000;    // Slew rate control disabled for standard speed (100kHz)
```

- Bit7 = 1 → Slew rate control off (for ≤100 kHz)
- Bit6 = 0 → SMBus specific settings disabled

4. Set I²C Clock Frequency

```
SSPADD = ((FOSC / (4 * I2C_BAUD)) - 1);  // Example: FOSC=4MHz, I2C_BAUD=100kHz → SSPADD=9
```

5. Clear I²C Interrupt Flags

```
SSPIF = 0;    // Clear MSSP interrupt flag
BCLIF = 0;    // Clear bus collision flag
```

6. Enable I²C Interrupts

```
SSPIE = 1;    // Enable MSSP interrupt
BCLIE = 1;    // Enable bus collision interrupt
PEIE  = 1;    // Enable peripheral interrupts
GIE   = 1;    // Enable global interrupts
```

7. Module Ready for Communication

At this point, the I²C module is initialized.

You can now:

- Generate Start Condition → SEN = 1
- Send Address + R/W bit → write to SSPBUF
- Wait for ACK → check ACKSTAT
- Generate Stop Condition → PEN = 1

TESTING N DEBUGGING:

- Timing Issues: Verify SDA–SCL timing; insufficient or excessive delays may cause false START/STOP conditions, leading to missing ACKs.

- SCL Pulse Errors: Extra or missing clock pulses can corrupt data; issuing a new START condition can reset the slave.

- Incomplete 8-Bit Transfer: The slave responds with ACK only after receiving 8 bits; if incomplete, restart communication with a START.

- Missing Bytes: If fewer bytes are sent than expected, the slave waits indefinitely; reinitiate the transfer with a START.

- Incorrect Slave Response: Floating or wrongly connected address lines result in no ACK; ensure the correct slave address is used.

- Address Update Requirement: Changing the address bits requires updating the master's target address; otherwise, communication fails.

Resolving Address Conflicts on the I²C Bus:

- Multiple devices with the same address cause bus conflicts and NACKs.

- Assign unique addresses using programmable or pin-selectable address bits (A0, A1).

- Up to 4 devices can share a bus if unique addresses are configured.

- SOLUTIONS:
1. Reprogram device addresses (if supported).
2. Use an I²C multiplexer (e.g., PCA9544A) to switch active devices.
3. Use an I²C buffer (e.g., PCA9515A/PCA9517) to isolate segments or perform level shifting.