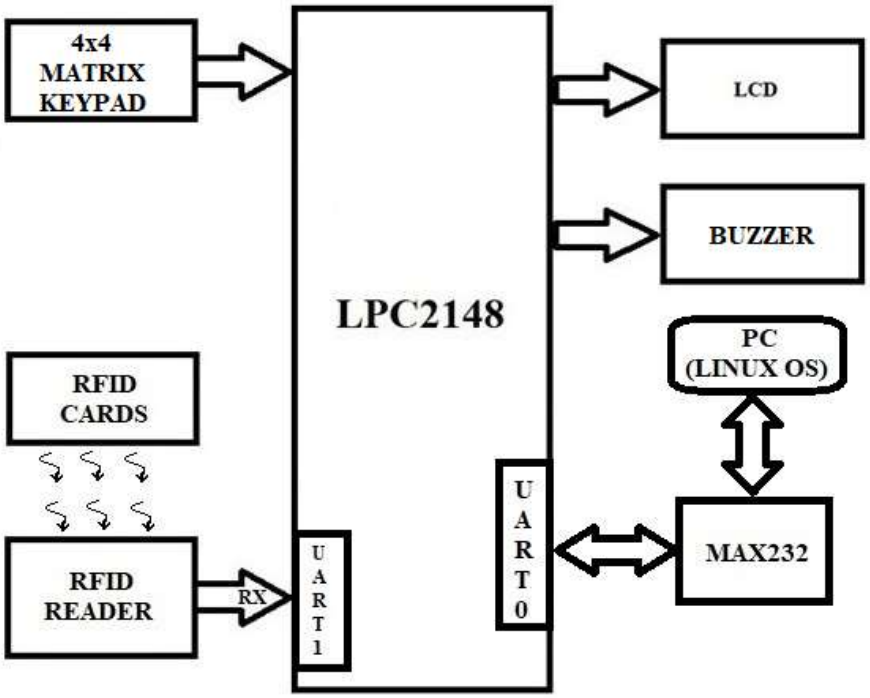
ATM SYSTEM DESIGN WITH DATABASE INTEGRATION

OBJECTIVE:

The main aim of this project is to develop a **secure ATM system** using **RFID authentication** and a **PIN-based interface**, with backend **banking database integration implemented in C using data structures** (instead of SQL-based systems).

BLOCK DIAGRAM:



REQUIREMENTS:

HARDWARE REQUIREMENTS:

- LPC2148 Microcontroller
- RFID Reader
- RFID Cards
- 16x2 LCD Display
- 4x4 Matrix Keypad
- MAX232 (for UART level shifting)
- USB-to-UART Converter/DB-9 Cable
- Buzzer

SOFTWARE REQUIREMENTS:

- EMBEDDED C – Programming
- KEIL uVision IDE
- FLASH MAGIC
- GCC Compiler

STEPS TO BE FOLLOWED TO COMPLETE THE PROJECT:

1. Create a new folder on your PC and name it with your project title.
2. This project includes two application programs:
    - One for the microcontroller board (ATM front-end)
    - One for the PC side (Linux), written in C, to simulate a banking database using data structures and file handling.
3. Copy and verify the following hardware interface modules into your MCU project folder:
    - lcd.c, lcd.h, delay.c, delay.h, uart.c, uart.h, keypad.c, keypad.h
4. Individually test all modules:
    - LCD: Display characters and strings.
    - Keypad: Read and display input values.
    - UART: Send/receive test strings using UART0 and UART1 (via interrupt).
    - RFID: Read card data via UART1 and display it on LCD.
5. Finalize your main microcontroller code projectmain.c:
    - Display project title briefly.
    - Continuously wait for RFID card to be presented.
    - When card is read, send the card number to PC over UART in the mentioned format. (#CARDNUMBER$)
    - Wait for PC to validate and reply.
    - If PC responds with success, display "Enter PIN" on LCD.
    - Read PIN from keypad.
    - Send card number and PIN to PC over UART in the format:

      #CARDNUMBER#PIN$
    - Wait for PC to validate and reply.

6. If PC responds with success, show the user menu on LCD:
    1. BALANCE
    2. DEPOSIT
    3. WITHDRAW
    4. PIN CHANGE
    5. MINI STATEMENT
    6. EXIT
        - User selects an option via keypad.
        - Send request in format: #ACTION#AMOUNT$

BALANCE ENQUIRY:

- User selects BALANCE from the menu via keypad.
- MCU sends request to PC in format: #TXN:BALANCE#REQ$
- PC identifies the user via RFID tag and retrieves current balance.
- PC sends response to MCU:

  @OK#BALANCE:XXXX.XX$
- MCU displays balance on LCD:

  Balance: XXXX.XX
- If retrieval fails, display “Unable to fetch balance” on LCD

DEPOSIT:

- User selects DEPOSIT from the menu via keypad.
- Prompt appears to enter deposit amount.
- User enters amount via keypad.
- MCU sends request to PC in format:

  #TXN:DEPOSIT#AMOUNT:1000$
- PC adds amount to the user's balance.
- PC logs transaction (Deposit, amount, timestamp) in linked list and file.
- PC sends updated balance to MCU:

  @OK#BALANCE:XXXX.XX$
- MCU displays:

  Deposit Success

New Balance: XXXX.XX

- If update fails, display: Deposit Failed

WITHDRAWAL:

- User selects WITHDRAW from the menu via keypad.
- Prompt appears to enter withdrawal amount.
- User enters amount via keypad.
- MCU sends request to PC in format:

  #TXN:WITHDRAW#AMOUNT:500$

- PC checks for sufficient balance.
- If balance is sufficient: Deducts amount, Logs transaction (Withdraw, amount, timestamp) and sends: @OK#BALANCE:XXXX.XX$
- MCU displays:

  Withdraw Success

  New Balance: XXXX.XX

- If insufficient balance, then sends: @ERR#INSUFFICIENT_BALANCE$
- MCU displays: Insufficient Balance
- If process fails for technical reasons: display Withdrawal Failed

ATM PIN CHANGE:

- Prompts user to enter the current PIN again.
- Asks for new PIN input twice for confirmation.
- Sends request to PC as: #PINCHANGE#NEWPIN$
- PC updates the PIN in the user database and sends response:

  @OK#PINCHANGED$ or @ERR#PINCHANGE_FAILED$

- If PIN change process failed for three times, then display “Contact nearest branch to change the PIN” on LCD.
- In this situation PIN modification allowed through database side

MINI STATEMENT (Last 3 Transactions):

- Sends request to PC: #MINISTMT#REQ$
- PC retrieves last 3 transactions from the linked list.
- Responds as: @OK#MINISTMT:TXN1|TXN2|TXN3$
- Displayed sequentially on LCD.

TIME LIMITS FOR KEYPAD INPUT:

After displaying the menu, the system starts a 30-second timer. If the user does not select any option within the timeout, the system resets to the initial state (waiting for Timer logic can be implemented using delay counters or timer interrupts)

7. Wait for PC to respond with success or error message.
    - Display response on LCD.
    - Loop back to initial state (waiting for new card).

PC-SIDE PROGRAM IN C (DATABASE USING DS):

1. Start Program and Initialize

- Open required files:
    - users.txt → Load user records into an array/list.
    - transactions.txt → Read existing transactions if needed.
- Initialize UART to continuously read input from MCU.

2. Receive and Parse RFID Card Number

- Wait for input string in format:

  #CARD:12345678$
- Parse the RFID value.
- Check if it exists in the user list.
    - If found: send @OK#VALID_CARD$
    - If not: send @ERR#INVALID_CARD$

3. Receive and Validate PIN

- Wait for:

  #CARD:12345678#PIN:4321$
- Match with stored user PIN.
    - If correct: send @OK#LOGIN_SUCCESS$
    - If incorrect: send @ERR#INVALID_PIN$

4. Handle Menu Selections

Wait for one of the following requests from the MCU:

A. BALANCE ENQUIRY

- Format: #TXN:BALANCE#REQ$

- Action:
    - Fetch user balance.
    - Send: @OK#BALANCE:XXXX.XX$

B. DEPOSIT

- Format: #TXN:DEPOSIT#AMOUNT:1000$
- Action:
    - Add amount to user's balance.
    - Log the transaction (type: Deposit, timestamp).
    - Update users.csv/users.txt and transactions.csv/transactions.txt.
    - Send: @OK#BALANCE:XXXX.XX$

C. WITHDRAWAL

- Format: #TXN:WITHDRAW#AMOUNT:500$
- Action:
    - Check if balance is sufficient.
    - If yes: deduct, log, and update files.
      Send: @OK#BALANCE:XXXX.XX$
      If not: send @ERR#INSUFFICIENT_BALANCE$

D. PIN CHANGE

- Format: #PINCHANGE#NEWPIN$
- Action:
    - Update the pin value in the user structure.
    - Write updated data to users.txt.
    - Send:
      On success: @OK#PINCHANGED$
      On failure: @ERR#PINCHANGE_FAILED$

E. MINI STATEMENT

- Format: #MINISTMT#REQ$
- Action:
    - Search the transaction linked list for the user.
    - Fetch last 3 transactions.
    - Format them as:
      @OK#MINISTMT:Deposit-100@10:00|Withdraw-50@12:00|Deposit-200@14:00$
    - Send this to MCU for LCD display.

5. Final Step

- After each transaction:

- Await next command or timeout.
- On session end or timeout from MCU, return to idle state.
- Wait for the next RFID input.

Example Structures:

```
typedef struct {
   char tag_id[20];
   char pin[10];
   char name[50];
   float balance;
} User;


typedef struct Transaction {
   char tag_id[20];
   char type[10]; // "Withdraw"/"Deposit"
   float amount;
   char timestamp[25];
   struct Transaction *next;
} Transaction;
```

4. Send the result of operations back to the MCU in format:
    - @OK#BALANCE:1234.56$
    - @ERR#INVALID_PIN$
    - @ERR#INSUFFICIENT_BALANCE$

MESSAGE FORMATS:

From MCU to PC:

#CARD:12345678#PIN:4321$

#TXN:DEPOSIT#AMOUNT:1000$

From PC to MCU:

@OK#BALANCE:5300$

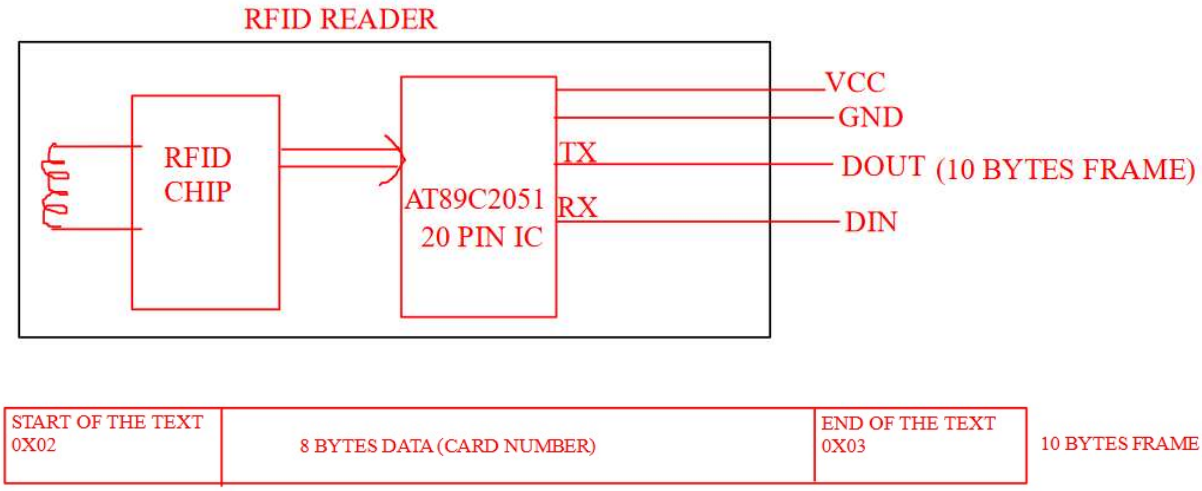
@ERR#INVALID_PIN$

@ERR#INSUFFICIENT_BALANCE$

This process provides a brief and structured outline of the project implementation. Based on specific application requirements, additional functionalities and enhancements may be incorporated into both the microcontroller and PC-side applications.

********** ALL THE BEST **********

RFID READER: RFID Reader will send the 10 bytes of data once card is placed nearer to the RFID Reader. Once check the below images to get some idea on RFID Reader block diagram. Refer the supported data sheets to get some more knowledge.



For example, card number is 12345678, then output of the RFID Reader is

0x02 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x03 (hex format)