Pac-Man, originally known as Puck Man in Japan, is a maze video game developed and released by Namco in 1980. The game was later released in North America by Midway Manufacturing. The inspiration for the Pac-Man character came from a pizza with a slice removed, and the game's characters were designed to be cute and colorful to appeal to younger players. In the game, Pac-Man eats the pills scattered throughout the maze to make points, while avoiding the ghosts that chase him. You can try online Pac-Man here.

Pac-Man on LandTiger Board

This project is an embedded system implementation of the classic Pac-Man game on the LandTiger development board. The objective is to recreate the Pac-Man gameplay while integrating multiple peripherals of the board to demonstrate their practical use in an interactive application.

The implementation was developed using Keil µVision IDE for ARM microcontrollers, with direct deployment and testing on the LandTiger hardware board. All code, project files, and configurations are prepared to run natively on the board (the emulator alone is not sufficient).

Peripherals and Features Implemented

- **LCD Display**
    - The Pac-Man maze, characters, and game status are drawn on the board's LCD screen, allowing real-time visualization of the game.
- **Input Controls (Joystick / Buttons)**
    - The player controls Pac-Man's movement using the on-board joystick and buttons, enabling smooth navigation across the maze.
- **AI-Controlled Ghost (Blinky)**
    - A red ghost with AI pursues Pac-Man using a pathfinding strategy.
    - Supports two behavior modes:
        - Chase Mode → follows Pac-Man directly.
        - Frightened Mode → activated when Pac-Man eats a Power Pill. The ghost flees from Pac-Man, turns blue, and can be eaten for bonus points. If defeated, it respawns at the center.
- **Sound System (Speaker)**
    - The speaker generates background music and sound effects for events such as eating pills, catching ghosts, and losing lives.
- **Timers**
    - Hardware timers manage countdowns, frightened-mode duration, and ghost respawn delays.
    - Provide accurate synchronization of game mechanics.
- **CAN Bus Communication**
    - Configured in loopback mode using CAN1 and CAN2.
    - Transmits a 32-bit encoded game status message containing:
        - Remaining Time (8 bits)
        - Remaining Lives (8 bits)
        - Current Score (16 bits)

    - Demonstrates board-level communication and real-time data sharing.
- Interrupts & System Control
    - Interrupts handle inputs, timer events, and peripheral operations efficiently, ensuring responsive gameplay.

CAN Bus Communication in Pac-Man Implementation

In this project, the **CAN (Controller Area Network) peripherals** of the LandTiger board are used to transmit game status information. The system is configured in **external loopback mode**, meaning that the board sends a message using **CAN1** and receives it back through **CAN2**. Even though the communication happens internally, this configuration mimics the behavior of two boards exchanging data.

Message Structure

Each CAN message encodes the following game parameters:

- **Remaining Time** → 8 bits
- **Remaining Lives** → 8 bits
- **Current Score** → 16 bits

These fields are packed into a **32-bit unsigned integer variable**, ensuring compact and efficient transmission.

Physical Connection

The CAN bus requires two signal lines:

- **CAN High (red line)**
- **CAN Low (green line)**

These lines must be connected as shown in the diagram, enabling proper communication between the CAN controllers.

Purpose in the Game

By sending game data (time, lives, and score) through the CAN bus, the project demonstrates how an embedded system can:

- Share real-time game status with another device.
- Test inter-board communication using loopback.
- Apply standard automotive/industrial communication protocols in a gaming context.

This integration highlights the versatility of the LandTiger board by combining **game logic with embedded communication standards**.

Remaining time	Remaining lives	Score
8 bits	8 bits	16 bits

Every **message** must be saved in a **32-bit unsigned int variable**

This project combines **classic game logic** with embedded systems design by leveraging the **LCD, joystick, timers, speaker, CAN bus, and interrupts** of the LandTiger board.
It was fully implemented and tested in **Keil µVision IDE**, showcasing how software and hardware interact to build a complete real-time embedded application.