# CAN TEMPERATURE MONITOR

## CAN0 Internal Loopback on TM4C123GH6PM

ECE-4721 Embedded Systems
Yousif Fatohi, Andrew Kashat, Peter Younan

# PROJECT OVERVIEW

## Read Temperature

Internal ADC reads the TM4C123 die temperature sensor — no external hardware needed.

## Transmit via CAN

Temperature packed into a standard CAN frame (ID 0x100) and sent at 500 kbps through CAN0.

## Loopback Receive

CAN0 loopback mode echoes the frame back internally — proving TX and RX without a second node.

## Display on PC

UART0 prints TX vs RX values to a serial terminal (PuTTY) over the on-board USB debug link.

# HARDWARE & PIN MAP

## CAN0 Bus

PE5 (CAN0TX) → MCP2551 TXD

PE4 (CAN0RX) → MCP2551 RXD

500 kbps | 16 MHz system clock

Alternate function 8 on Port E

## UART0 (Debug)

PA1 (TX) → ICDI USB → PC

PA0 (RX) → ICDI USB → PC

9600 baud | 8-N-1

No external adapter needed

## CAN Frame Format

ID: 0x100 (standard 11-bit) | Byte 0: Temperature °C | DLC: 2 bytes

# WHY LOOPBACK MODE?

## THE PROBLEM

CAN1 uses PA0/PA1 — the same pins as UART0 (the on-board USB debug port).

Using CAN1 means losing the serial terminal, making debugging impossible.

A second CAN node would require another LaunchPad or transceiver board.

## THE SOLUTION

CAN0 loopback mode — the controller receives its own transmitted frames internally.

CAN0 uses PE4/PE5, which don't conflict with UART0.

The MCP2551 transceiver still drives the physical bus — external devices can still see the frames.

Single-board demo with full CAN TX + RX verification.

```
Key Register:  CANTEST bit 4 (Lback) = 1   →   TX fed back to RX internally
```

# ADC: INTERNAL TEMPERATURE SENSOR

## How It Works

1. ADC0 Sequencer 3 (SS3) configured for single-sample mode
2. Input mux set to internal temperature sensor (SSCTL3.TS0 = 1)
3. Software trigger initiates a conversion (PSSI register)
4. Poll RIS register until conversion complete
5. Read 12-bit raw value from SSFIFO3

```
TEMP (°C) = 147.5 – ( 75 × 3.3 × raw / 4096 )
```

# CAN0 INITIALIZATION

| | |
|---|---|
| **1. Clocks** | Enable CAN0 and GPIOE peripheral clocks via RCGCCAN and RCGCGPIO |
| **2. GPIO Config** | PE4/PE5 set to alternate function 8 (CAN0RX/TX), digital enable |
| **3. Init Mode** | Set INIT + CCE + TEST bits in CANCTL to enter configuration mode |
| **4. Loopback** | Set bit 4 (Lback) in CANTEST register for internal echo |
| **5. Bit Timing** | BRP=1, TSEG1=11, TSEG2=4, SJW=1 → 16 Tq/bit → 500 kbps |
| **6. Exit Init** | Clear INIT bit, keep TEST bit set so loopback stays active |
| **7. Msg Obj 1 TX** | Configure message object 1: ID 0x100, direction = transmit, DLC = 2 |
| **8. Msg Obj 2 RX** | Configure message object 2: ID 0x100 mask, direction = receive, DLC = 8 |

# TRANSMIT & RECEIVE FLOW

## CAN0_Transmit()

1. Write data bytes into IF1DA1
2. Set TXRQST + EOB + DLC in IF1MCTL
3. Write message object 1 to IF1CRQ
4. Poll until IF1CRQ.BUSY clears

## CAN0_Receive()

1. Read message object 2 via IF2CRQ
2. Check NEWDAT bit (IF2MCTL bit 15)
3. If set → extract data from IF2DA1
4. Return 1 (success) or 0 (timeout)

## Polling Strategy

After transmit, the main loop polls CAN0_Receive() up to 100 times with 50 µs delays.

At 500 kbps a standard CAN frame takes ~36 µs — 5 ms of polling gives plenty of headroom.

If no frame arrives, the terminal shows "RX timeout" instead.

# MAIN LOOP

1

## Read Temperature

ADC0_ReadTempC() samples the die sensor and returns °C

2

## Pack & Transmit

Temperature byte loaded into tx_data[0], sent via CAN0_Transmit()

3

## Poll for Loopback

CAN0_Receive() checked up to 100× (50 µs each) for the echoed frame

4

## Print to Terminal

printf() sends "TX Temp=28 C | RX Temp=28 C" over UART0 to PuTTY

5

## Delay 1 Second

delay_ms(1000) before repeating the cycle

# SERIAL OUTPUT

## PuTTY Setup

1. Plug LaunchPad USB into PC

2. Device Manager → Ports →
   "Stellaris Virtual Serial Port"

3. PuTTY → Serial → COMx
   9600 baud, 8-N-1

4. Press Reset on LaunchPad

## Expected Output

```
CAN Temp Monitor Ready
TX Temp=28 C  |  RX Temp=28 C
TX Temp=29 C  |  RX Temp=29 C
TX Temp=28 C  |  RX Temp=28 C
TX Temp=27 C  |  RX Temp=27 C
TX Temp=28 C  |  RX timeout
```

`fputc()` retargets C standard I/O — `printf()` calls `uart0_send()` for each character over UART0

# KEY TAKEAWAYS

- Loopback mode enables full CAN TX/RX testing on a single TM4C123 board
- Internal temperature sensor provides real-world data with zero external wiring
- CAN0 on PE4/PE5 avoids the PA0/PA1 conflict with UART0 debug output
- MCP2551 transceiver still active — frames visible to external CAN devices
- Modular code structure: ADC, CAN, UART each isolated in their own init/read/write functions