CMR GROUP OF INSTITUTIONS
EXPLORE TO INVENT

CMR Engineering College

UGC AUTONOMOUS

KANDLAKOYA (VI), MEDCHAL ROAD, HYDERABAD-501401

Department of Electronics and Communication Engineering

ARM Cortex-M3 Control Boards

It’s a great addition to your lab facilities! ARM Cortex-M3 boards (like the **STM32F103** or **LPC1768**) are industry-standard tools for teaching embedded systems because they bridge the gap between simple 8-bit microcontrollers and high-performance 64-bit processors.

Below is a draft you can use for your college website. It is structured to be professional, informative for students, and attractive for accreditation (like ABET or NBA) purposes.

Facility Spotlight: Cortex-M3 Embedded Systems Experments

Overview

Our laboratory is now equipped with **ARM Cortex-M3 Control Boards**, providing students with a high-performance 32-bit prototyping platform. These boards are designed to handle the complex, real-time processing demands of modern embedded applications, ranging from Industrial Automation and Robotics to IoT (Internet of Things) devices.[1]

Core Technical Specifications

The Cortex-M3 processor is the industry leader for deterministic, low-power applications. Key features available on our lab units include:

- **Architecture:** 32-bit RISC ARMv7-M with a 3-stage pipeline (Harvard Architecture).[2]
- **Performance:** Up to 1.25 DMIPS/MHz, allowing for complex mathematical operations and fast data processing.[3]
- **Interrupt Handling:** Features the **Nested Vectored Interrupt Controller (NVIC)**, which reduces interrupt latency and provides high responsiveness for real-time tasks.[4]
- **Memory:** Integrated Flash memory and high-speed SRAM, supported by an optional **Memory Protection Unit (MPU)** for robust software execution.[5]

- **Peripherals:** On-board support for USB, CAN, UART, I2C, SPI, and High-speed ADCs for sensor interfacing.[6]

Educational Objectives

The introduction of these boards allows students to move beyond basic electronics and master professional-grade engineering concepts:

1. **Real-Time OS (RTOS) Implementation:** Learning to manage multitasking and priority-based scheduling.
2. **Advanced Interfacing:** Connecting high-resolution displays, industrial sensors, and communication modules (Ethernet/Bluetooth).
3. **Power Optimization:** Exploring hardware-controlled sleep modes and low-power system design.
4. **Hardware-Software Co-Design:** Utilizing professional IDEs like **Keil MDK**, **STM32CubeIDE**, or **IAR Embedded Workbench**.

Applications in Student Projects

- **Robotics:** Precision motor control using dedicated PWM channels.[7]
- **Smart Systems:** Developing Smart Home nodes or wearable health monitors.[8]
- **Industrial IoT:** Monitoring sensor data and transmitting it via wired or wireless gateways.[9]

Note for Students: Technical manuals, sample codes, and pin-out diagrams for the specific board variants (e.g., STM32 / LPC series) are available google drve Folder.

Quick-Start Guide: ARM Cortex-M3 Control Board

Welcome to the Embedded Systems Lab! This guide will walk you through setting up your workspace and running your first "Blinky" program on the Cortex-M3 platform.

Step 1: The Development Environment

We primarily use **Keil MDK-ARM** or **STM32CubeIDE** for development.

1. **Open the IDE:** Launch Keil μVision from the desktop.
2. **Install Device Packs:** Ensure the specific device family (e.g., STM32F1xx or LPC17xx) is installed via the **Pack Installer**.
3. **Create a Project:**
   - Go to Project -> New uVision Project.
   - Select your specific chip model from the device database.
   - When prompted to "Copy Startup Code," select **Yes**. This file handles the initial boot sequence.

Step 2: Hardware Setup

1. **Connect the Debugger:** Plug the **ST-Link** or **J-Link** debugger into the board’s JTAG/SWD header.
2. **Power Up:** Connect the board to your PC via USB. Ensure the "Power" LED on the board is lit.
3. **Driver Check:** Open *Device Manager* on your PC to ensure the debugger is recognized under "Universal Serial Bus devices."

Step 3: Writing Your First Program (Blinky)

Create a new file main.c and add the following logic (syntax may vary slightly by board manufacturer):

C

```
#include "stm32f10x.h" // Replace with your board's specific header
```

```
void Delay(uint32_t count) {
    for(uint32_t i = 0; i < count; i++);
}

int main(void) {
    // 1. Enable Clock for the GPIO Port
    RCC->APB2ENR |= (1 << 4); // Example: Enable Port C

    // 2. Configure Pin as Output
    GPIOC->CRH &= ~(0xF << 20); // Clear bits
    GPIOC->CRH |= (0x3 << 20);  // Set as Output (50MHz)

    while(1) {
        GPIOC->ODR ^= (1 << 13); // Toggle LED on Pin 13
        Delay(1000000);
    }
}
```

Step 4: Compiling and Flashing

1. **Build:** Click the **Build** icon (or press F7). Look for "0 Error(s), 0 Warning(s)" in the output window.
2. **Target Settings:** * Go to Options for Target -> Debug tab.
    - Select your debugger (e.g., ST-Link Debugger) from the drop-down menu.
    - Under Settings -> Flash Download, check the box **Reset and Run**.
3. **Flash:** Click the **Download** button (or press F8). Your LED should start blinking!

Common Troubleshooting

- **"Target Not Found":** Check your USB cable and ensure the debugger is firmly connected to the board.
- **Build Errors:** Ensure you have included the correct header file for your specific Cortex-M3 variant.
- **LED Not Blinking:** Check the board schematic to verify which GPIO pin the physical LED is actually connected to.

Cortex-M3 Lab Experiment Series

Experiment 1: Digital I/O and External Interrupts

- **Objective:** Understand GPIO (General Purpose Input/Output) configuration and the Nested Vectored Interrupt Controller (NVIC).
- **Task:** Interface a push-button to an external interrupt (EXTI) line. When pressed, the interrupt should toggle an LED.
- **Learning Outcome:** Students learn the difference between "polling" a button and using hardware interrupts for efficiency.

Experiment 2: High-Resolution Timing with SysTick

- **Objective:** Master the system timer (SysTick) dedicated to the Cortex-M3 core.
- **Task:** Create a precise 1-millisecond delay timer using the SysTick register instead of "dummy" for-loops.
- **Learning Outcome:** Understanding the internal system clock and how to create time-deterministic software.

Experiment 3: Analog-to-Digital Conversion (ADC)

- **Objective:** Interface the physical world with 32-bit digital processing.
- **Task:** Connect a potentiometer or temperature sensor to an ADC channel. Read the voltage and output the value to a computer via UART (Serial Communication).
- **Learning Outcome:** Learning about sampling rates, resolution (12-bit), and data serialization.

Experiment 4: PWM Generation for Motor Control

- **Objective:** Use advanced timers to generate Pulse Width Modulation (PWM) signals.
- **Task:** Control the brightness of an LED or the speed of a small DC motor by varying the duty cycle of a timer output.
- **Learning Outcome:** Understanding how digital processors control analog power levels.

Experiment 5: Multi-Tasking with a Real-Time OS (RTOS)

- **Objective:** Transition from "Super-loop" programming to professional multitasking.

- **Task:** Use **FreeRTOS** to run two concurrent tasks: one task blinking an LED and another task reading sensor data, managed by a scheduler.
- **Learning Outcome:** Learning task priorities, context switching, and resource management in complex systems.

Resource Kit for Labs

To support these experiments, the following documentation is available in the lab:

- **The Definitive Guide to ARM Cortex-M3** (Joseph Yiu) – The "Bible" for this architecture.
- **Datasheets & Reference Manuals:** Specific to the chip manufacturer (ST, NXP, or TI).
- **Pin-out Maps:** Laminated sheets for each workbench to prevent incorrect wiring.

.