Phase 1: STM32 Microcontroller

Name: Avirup Bhunia

Date: 08/08/2025

Assignment: Firmware Intern

1.1 GPIO Configuration & Alternate Function Mapping

STM32's General Purpose Input/Output (GPIO) pins are highly flexible and can operate in multiple modes: input, output, alternate function (AF), and Analog.

Configuration involves setting the MODER, OTYPER, OSPEEDR, and PUPDR registers for each pin.

Alternate function mapping allows a pin to connect internally to peripherals like UART, SPI, I2C, or timers without external wiring changes.

Learned the process of mapping AF values using the AFRL/AFRH registers and checking the datasheet and reference manual for correct AF numbers.

1.2 RCC and Peripheral Clock Trees

The Reset and Clock Control (RCC) module manages clock sources, prescaler, and enables/disables peripheral clocks.

STM32 supports multiple clock sources: HSI, HSE, PLL.

Understanding the clock tree diagram is essential for ensuring peripherals receive stable clocks at required frequencies.

Learned to enable clocks for GPIO and communication peripherals via the RCC AHBxENR and RCC_APB1ENR/RCC_APB2ENR registers.

1.3 USART/UART Communication

UART is a full-duplex asynchronous serial protocol for point-to-point communication.

Key configuration parameters: baud rate, word length, stop bits, and parity.

Register-level setup includes USART_BRR for baud rate calculation and USART_CR1/CR2/CR3 for enabling TX/RX and interrupts.

Explored three methods of operation:

Polling — Simple but CPU-intensive.

Interrupt-driven — Efficient, uses RXNE/TXE interrupts.

DMA — Offloads transfer to DMA controller, improving efficiency.

1.4 DMA (Normal & Circular Mode, Peripheral–Memory Transfer)

The Direct Memory Access (DMA) controller enables high-speed data transfers without CPU intervention.

Normal mode: stops when the transfer count (NDTR) reaches zero.

Circular mode: automatically restarts the transfer, ideal for continuous data streaming (e.g., ADC sampling).

Learned about configuring DMA_SxCR, DMA_SxNDTR, DMA_SxPAR, and DMA_SxM0AR registers for peripheral–memory transfers.

1.5 Timers (Delay, PWM Generation, Input Capture)

STM32 timers can operate as: Basic timers for delays.

PWM timers for motor control or LED dimming.

Input capture for measuring external signal frequency/pulse width.

Configuration involves setting PSC (prescaler) and ARR (auto-reload register).

Learned to use CCRx registers for PWM duty cycle adjustment.

1.6 I2C and SPI Configuration and Use

I2C: Two-wire (SDA/SCL), supports multi-master and multi-slave configurations, uses addressing.

SPI: Full-duplex, master/slave, higher speed than I2C, uses MOSI, MISO, SCK, and optional NSS.

Learned to configure control registers (I2C_CR1/CR2, SPI_CR1/CR2) and handle ACK/NACK in I2C.

Master–slave communication tested with simple loopback setups.

1.7 NVIC Interrupt Structure and Vector Table Understanding

The Nested Vectored Interrupt Controller (NVIC) prioritizes and handles interrupts.

Learned about preemption priority and subpriority levels.

The vector table stores ISR addresses; each interrupt source has a fixed vector location.

2. Peripheral-Specific Challenges & Takeaways

1. GPIO — Challenge: Remembering AF mappings for each pin without confusion; takeaway: always cross-check with datasheet tables.

2. RCC — Challenge: Debugging clock misconfigurations when peripherals didn't work; takeaway: verify clock enable bits before peripheral configuration.

3. UART — Challenge: Ensuring correct baud rate with different oversampling modes (8 vs. 16); takeaway: precise calculation avoids framing errors.

4. DMA — Challenge: Handling incorrect NDTR settings that caused incomplete transfers; takeaway: initialize NDTR correctly before each transfer.

5. TIMERS — Challenge: Getting accurate delays without affecting other timers; takeaway: calculate prescaler and ARR values carefully.

6. I2C — Challenge: Debugging bus lock-ups when slaves didn't ACK; takeaway: implement timeout and bus reset mechanisms.

7. SPI — Challenge: Mismatch in CPOL/CPHA caused corrupted data; takeaway: ensure both devices share same mode.

8. NVIC — Challenge: Wrong priority configuration leading to delayed ISR execution; takeaway: understand preemption vs. subpriority rules.