Task Management System with Synchronized Multi-Channel PWM Control

Ismail

January 20, 2026

1 Introduction

This report describes the design and implementation of a task management system for the Texas Instruments **C2000 F28003x** microcontroller. The system uses a simple asynchronous state machine with two operating modes: **IDLE** and **RUN**. The main objective is to generate up to six synchronized PWM signals with a configurable switching frequency and duty cycle while satisfying strict initialization and safety requirements.

2 System Requirements

The implemented software fulfills the following requirements:

- Asynchronous execution inside an infinite loop
- Startup in **IDLE** mode
- All initialization performed in **IDLE**
- No PWM signal generation during **IDLE**
- Generation of up to six independent PWM outputs
- Identical switching frequency for all PWM channels
- Switching frequency configured once in **IDLE**
- Frequency range from 50 Hz to 50 kHz
- Identical duty cycle for all PWM channels
- Duty cycle initialized to 0% in **IDLE**
- Duty cycle adjustable between 0% and 100% in **RUN**

3 System Architecture

3.1 State Machine

The software is structured around a two-state finite state machine:

- **STATE_IDLE**: Performs all initialization tasks and ensures PWM outputs are disabled.
- **STATE_RUN**: Starts synchronized PWM operation and allows duty cycle updates.

The state machine runs asynchronously inside the main loop, without using blocking delays or an RTOS.

4 PWM Configuration

4.1 PWM Channels

Six PWM channels are used:

- ePWM1A (GPIO0)
- ePWM2A (GPIO2)
- ePWM4A (GPIO6)
- ePWM5A (GPIO16)
- ePWM6A (GPIO10)
- ePWM7A (GPIO12)

Each PWM module is configured identically to guarantee synchronized operation.

4.2 Switching Frequency

The switching frequency is configured once during **IDLE**. The PWM period is calculated using:

$$TBPRD = \frac{f_{SYSCLK}}{2 \cdot f_{PWM}} \tag{1}$$

For a system clock of 120 MHz and a PWM frequency of 10 kHz:

$$TBPRD = \frac{120 \times 10^6}{2 \times 10^4} = 6000 \tag{2}$$

4.3 Duty Cycle

The duty cycle is implemented using the Compare A (CMPA) register:

$$CMPA = TBPRD \times \frac{Duty(\%)}{100} \tag{3}$$

All PWM channels share the same CMPA value to ensure identical duty cycles.

5 Important Code Sections

5.1 Asynchronous Task Execution

```
while(1)
{
    TaskManager_Run();
}
```

This loop ensures continuous, non-blocking execution of the state machine.

5.2 IDLE State Initialization

```
void System_StateIdle(void)
{
    if(!initComplete)
    {
        switchingFrequency = DEFAULT_FREQ;
        dutyCycle = INITIAL_DUTY_CYCLE;
        pwmPeriod = PWM_CalculatePeriod(switchingFrequency);
        pwmCompare = PWM_CalculateCompare(pwmPeriod, dutyCycle);
        PWM_ConfigureAllChannels(switchingFrequency);
        PWM_DisableOutputs();
        initComplete = true;
        systemState = STATE_RUN;
    }
}
```

Key points:

- PWM frequency and duty cycle are initialized
- PWM timers are configured but not started
- TBCLKSYNC remains disabled
- No PWM output is generated

5.3 RUN State Operation

```
void System_StateRun(void)
{
    static bool firstEntry = true;

    if(firstEntry)
    {
        PWM_StartAll();
        dutyCycle = 50.0f;
        PWM_SetDutyCycle(dutyCycle);
        firstEntry = false;
    }
}
```

Key points:

- PWM signals start synchronously
- Duty cycle is updated dynamically
- PWM operation is isolated from initialization

6 IDLE to RUN Timing Measurement

6.1 Measurement Method

The transition time from **IDLE** to **RUN** was measured using **CPU Timer 0**. The timer runs at the system clock frequency of 120 MHz, providing a resolution of 8.33 ns per cycle.

6.2 Measured Result

The measured IDLE-to-RUN transition time was:

$$t_{IDLE \rightarrow RUN} \approx 57.9\ \mu s \tag{4}$$

This time includes:

- PWM and GPIO configuration
- Period and compare calculations
- State transition logic

It excludes:

- PWM signal generation
- Real-time control execution

7 Important Design Considerations

- PWM outputs are strictly disabled during IDLE for safety
- TBCLKSYNC ensures synchronous PWM startup
- Global configuration guarantees identical frequency and duty cycle
- Initialization latency occurs only once at startup

8 Conclusion

A robust and deterministic task management system with synchronized multi-channel PWM control was successfully implemented. The design strictly separates initialization and runtime behavior, ensures safety during startup, and meets all specified functional requirements. The measured initialization latency confirms efficient execution and negligible impact on real-time performance.