KPIT

CASE STUDY DETAILS & GUIDELINES

Legends	Description
Document version	Version 1.0
Type of assessment	Case study
Title	Wiper System control
Skill / Domain	Embedded C
Difficulty level	Basic
Category	Embedded C Programming and circuit simulation
Edited by	
Reviewed by	

Design by: Raghav Ankur

KPIT Technologies Ltd.

KPIT

Contents

KPIT

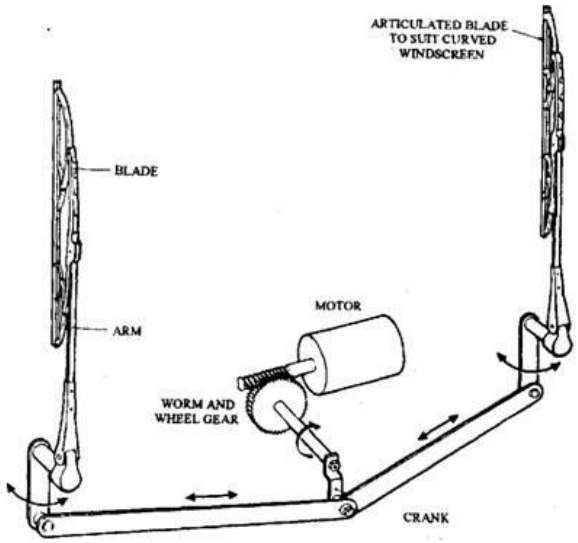
Usage Guidelines

Do not forward this document to any non-KPIT mail ID. Forwarding this document to a non-KPIT mail ID may lead to disciplinary action against you, including termination of employment.

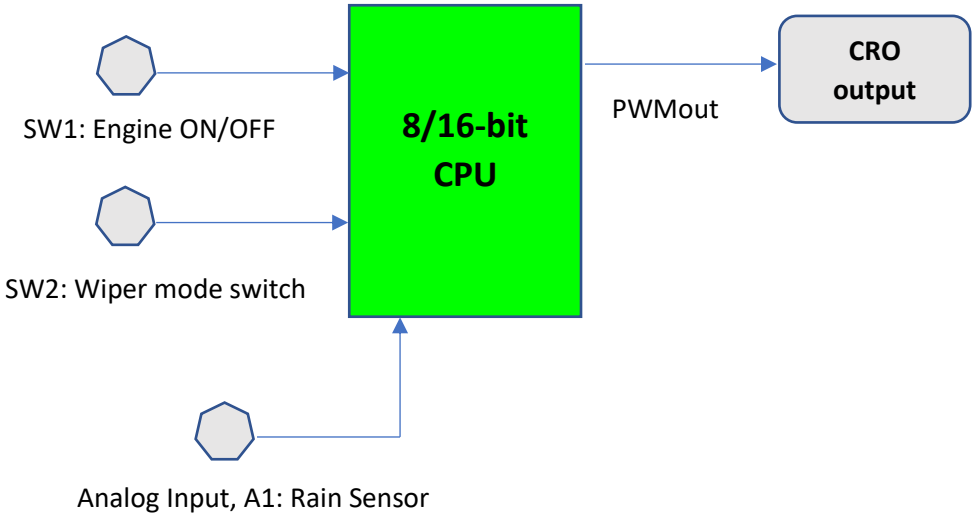
COPYRIGHT NOTICE

Introduction

In automotive applications, PWM technique is used to control the motor of the Wiper system. With the use of PWM in wiper control system, system is more flexible in term of controlling the speed of motor which in turn control the wiper system. In such system, motor is controlled by typically 8 or 16-bit microcontroller that has in-build PWM module that can generate pulses of different duty cycles at fixed frequency.



System Design Specifications



System inputs interface:

Switch SW1: Engine state is represented by SW1

Switch SW2: Wiper mode is represented by SW2

Analog input, A1: Analog value corresponding to rain intensity

Condition	User test input	Value or Range
Engine OFF	Switch SW1 open	High → 1
Engine ON	Switch SW1 close	Low → 0
Wiper mode- Auto	Switch SW2 close	Low →0
Wiper mode- Manual	Switch SW2 open	High →1
Intensity of rain	Potentiometer connected to Analog input A1	0 to 1023 for 10-bit ADC resolution

System outputs Interface:

PWMout signal: Generate PWM pulse of constant duty cycle on this pin as per required specification.

Feature1: Wiper state for Engine ON and Wiper is in auto mode

The wiper system will work under following conditions.

i. When automobile **engine is ON** and wiper mode is set to **auto mode**, wiper system will sense the analog input (intensity of rain) every 5 second corresponding to rain sensed by windshield. Depending upon the value of intensity of rain, the wiper is controlled at specific speed as specifies below:

Intensity of rain	Wiper system speed value	Description
No rain	Wiper off	DC motor off
Low	25% duty cycle	Low RPM corresponding to DC motor
Medium	50% duty cycle	Medium RPM corresponding to DC motor
High	75% duty cycle	High RPM corresponding to DC motor
Very high	100% duty cycle	Highest RPM

Duty cycle values corresponding to analog inputs are:

Analog input	User test output	Value of duty cycle
0 < rain_intesity <= 50	PWMout	0%
50 < rain_intensity <= 200	PWMout	25%
200 < rain_intensity <= 500	PWM Duty Cycle	50%
500 < rain_intensity <= 1000	PWM Duty Cycle	75%
rain_intensity>1000		100%

Feature2: Wiper state for Engine ON and Wiper is in manual mode
Under this condition system must generate default duty cycle of 50% on PWMout interface and shall not sample analog input (intensity of rain).

Feature3: Wiper state for Engine OFF
Under this condition system should not sense analog inputs as well as shall not generate any PWM signal.

Software design considerations

The use of external interrupts for Engine on/off and Wiper mode need to be considered for ISR based software design.

As per requirement wiper system must be active if engine ON state is detected. The system parameters, configuration settings need to be initialized before detecting the state of engine. If Engine is in ON state and wiper is in auto mode, the sampling process must start to process the data from rain sensor every 5 second. To sample sensor data ISR need to be designed for timer overflow interrupt. We also need to check timer overflow flag and based upon overflow count_timer need to be incremented within TimerA ISR. Once count_timer reaches to count_set (which coresponds to 5 second delay) we need to sample the ADC value and store it in the user defined global buffer. Depending upon the ADC value we need to generate duty cycle to control speed of motor.

Expected outcomes

O1: Able to write basic code for various CPU peripherals (Timers, GPIO and PWM)

O2: Understand various configuration registers and peripheral operations

O3: Able to Analyze application features / requirements

O4: Able to relate feature implementation with CPU peripherals

O5: Design algorithm and pseudo code

O6: Design Embedded C code based upon algorithm

O7: Able to design using modular approach

Task/Activities

Activity-I: Modular architecture and C code design

- Choose relevant software architecture and design modular C code for hardware platform.
- Ensure Usage of macros, enums, bit-fields, structures, function pointers to design code.
- Ensure correct usage of datatypes, Qualifiers like volatile, const, extern, static.

Activity-II: Circuit simulation

Design Test circuit to validate the functionality

Activity-III: Code quality and related language standards

Develop the code as per guidelines and related language standards.

Guidelines & Assessment Schema for case study-based projects

Total time duration 3 hours

Assignment submission packages

- Module Implementation files [.c files] and corresponding header files [.h files]
- Main program [.c file]
- .HEX file
- Simulation circuit [.simu file]

Resources and tools required

Tools required	Description	Download link
Code Block	IDE used to develop C/C++ based embedded applications	http://www.codeblocks.org/downloads/binaries
WinAVR	Open source gcc based tool chain for AVR micro-controllers	https://sourceforge.net/projects/winavr/
Simulide	Used for Circuit designing, uploading hex file, and simulating the application	https://sourceforge.net/projects/simulide/

Activity based assessment rubrics and associated weightage

Rubrics for measuring various activities	Rating Scale (0-10)
Rubrics for Activity-I	
Modular Programming and related guidelines	
Proper usage of variable naming conventions	
Relevant use of various storage classes	
Proper usage of datatypes, macros structs, enum, pointers, typedefs	
Overall Software Design and Interrupt usage	
Rubrics for Activity-II	
Overall functionality	
Ensure Proper test cases as per input - output requirements	
Rubrics for Activity-III	
Code optimizations	
Any coding guidelines MISRA, CERT or Secure coding	

Additional note

- Ensure the code works for the stated requirements.
- You are not supposed to switch case conditions.
- Use Timers to create PWM as well as for creating delay.
- Comply the code with **code style guidelines.**
- Develop an embedded C code for the above scenarios.
- Proper use of interrupt needs to be emphasized.