Enhance The Turtle Documentation

Hazem Essam

September 3, 2024

Contents

1 Introduction

This document provides a detailed description of the code and libraries used in the project. The project involves motor control using an ATmega328P microcontroller, with various components and peripherals configured for operation.

2 Code Overview

2.1 ADC Library

```
#ifndef ADC_H
#define ADC_H

#include <xc.h> // include processor files - each
   processor file is guarded.
#include <avr/io.h>  // Include AVR IO library for
   register definitions

// Function prototypes
void ADC_Init(void);
uint16_t ADC_Read(uint8_t channel);

// GPIO-related prototypes
void GPIO_SetPinAsInput(uint8_t port, uint8_t pin);
void GPIO_DisableDigitalInput(uint8_t adc_channel);

#endif /* ADC_H */
```

Functionality: The ADC library provides functions to initialize the ADC, read values from the ADC channels, and manage GPIO pin settings related to ADC operations.

2.2 GPIO Library

```
#ifndef GPIO_H
#define GPIO_H
```

```
3
4 #include <avr/io.h>
5
6 // Function prototypes
7 void GPIO_Init(void);
8
9 #endif // GPIO_H
```

Functionality: The GPIO library initializes GPIO pins for various functions, such as PWM output and interrupt input.

2.3 Interrupt Library

```
1 #ifndef INTERRUPT_H
2 #define INTERRUPT_H
3
4 #include <avr/io.h>
5 #include <avr/interrupt.h>
6
7 void EXT_INT_Init(void);
8
9 #endif
```

Functionality: This library sets up external interrupts, specifically for handling encoder pulses.

2.4 SPI Library

```
1 #include <avr/io.h>
2 #define SS PB2
3 #define MOSI PB3
4 #define MISO PB4
5 #define SCK PB5
6
7 void SPI_MASTER(void);
8 void transmit(uint8_t);
```

Functionality: The SPI library sets up SPI communication as a master and provides functions to transmit data over SPI.

2.5 System Library

```
#ifndef XC_HEADER_TEMPLATE_H
#define XC_HEADER_TEMPLATE_H

#include <xc.h> // include processor files - each
   processor file is guarded.
#include "Timer.h"
#include "interrupt.h"
#include "GPIO.h"
#include <avr/io.h>
#include <avr/interrupt.h>
#include "ADC.h"
#include "spi.h"

void Sys_Init(void);

void motor_control(void);

#endif  /* XC_HEADER_TEMPLATE_H */
```

Functionality: The system library initializes all components and peripherals, and includes the motor control function.

2.6 Timer Library

```
#ifndef TIMER_H
#define TIMER_H

#include <avr/io.h>

void Timer0_Init(void);
void Timer1_Init(void);

```

```
#endif
```

Functionality: This library initializes timers for PWM and other time-based tasks.

2.7 ADC Implementation

```
#include "ADC.h"  // Include the corresponding header file
void ADC_Init(void) {
    GPIO_SetPinAsInput('C', 0);
    GPIO_DisableDigitalInput(0);
    ADMUX |= (1 << REFS0);
    ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
    ADCSRA |= (1 << ADEN);
}

uint16_t ADC_Read(uint8_t channel) {
    ADMUX = (ADMUX & 0xF8) | (channel & 0x07);
    ADCSRA |= (1 << ADSC);
    while (ADCSRA & (1 << ADSC));
    uint16_t ADCRead = ADCL;
    ADCRead |= (ADCH << 8);
    return ADCRead;
}

void GPIO_SetPinAsInput(uint8_t port, uint8_t pin) {
    switch(port) {
        case 'B':
            DDRB &= ~(1 << pin);
            break;
        case 'C':
            DDRC &= ~(1 << pin);
            break;
        case 'D':
            DDRD &= ~(1 << pin);
            break;
```

```
    }
}

void GPIO_DisableDigitalInput(uint8_t adc_channel) {
    switch(adc_channel) {
        case 0:
            DIDR0 |= (1 << ADC0D);
            break;
    }
}
```

Functionality: This file contains the implementation of ADC initialization, reading values, and GPIO configuration functions related to ADC.

2.8 GPIO Implementation

```
#include "GPIO.h"

void GPIO_Init(void) {
    DDRD |= (1 << PD6);
    DDRD &= ~(1 << PD2);
    PORTD |= (1 << PD2);
}
```

Functionality: Initializes the GPIO pins, configuring PD6 for PWM output and PD2 for encoder input.

2.9 Interrupt Implementation

```
#include "interrupt.h"

void EXT_INT_Init() {
    cli();
    EICRA = (1 << ISC01) | (0 << ISC00);
    EIMSK = (1 << INT0);
    sei();
}
```

Functionality: Configures external interrupts for handling encoder pulses.

2.10 Main Program

```
#include "system.h"
#include "spi.h"
#define PPR  10
#define F_CPU 16000000UL

volatile uint16_t encoder_counts = 0;
volatile uint16_t rpm = 0;
volatile uint16_t rpmToSend = 0;

ISR(TIMER1_COMPA_vect)
{
    rpm = (encoder_counts * 60) / PPR;
    rpmToSend = rpm/2;
    encoder_counts = 0;
    transmit(rpmToSend >> 8);
    transmit(rpmToSend & 0xFF);
}
ISR(INT0_vect)
{
    encoder_counts++;
}

void main(void)
{
    Sys_Init();

    while(1)
    {
        motor_control();
    }
    return;
}
```

Functionality: The main program initializes the system and continuously performs motor control based on ADC readings.

2.11 SPI Implementation

```
#include "spi.h"

void SPI_MASTER()
{
  DDRB |= (1<<MOSI) | (1<<SS) | (1<<SCK);
  DDRB &= ~(1<<MISO);
  SPCR = (1<<SPE) | (1<<MSTR) | (1<<SPR0);
}

void transmit(uint8_t data)
{
  SPDR = data;
  while(!(SPSR & (1<<SPIF)));
}
```

Functionality: Configures SPI as a master and provides functions to transmit data.

2.12 System Implementation

```
#include "system.h"

void Sys_Init()
{
    ADC_Init();
    GPIO_Init();
    EXT_INT_Init();
    TIMER0_Init();
    Timer1_Init();
    SPI_MASTER();
}

```

```
void motor_control()
{
    uint8_t duty_cycle = (uint8_t)((ADC_Read(0) *
       255UL) / 1023UL);
    OCR0A = duty_cycle;
}
```

Functionality: Initializes all system components and implements motor control based on ADC readings.

2.13 Timer Implementation

```
#include"Timer.h"

void TIMER0_Init() {
    TCCR0A = (1 << WGM01) | (1 << WGM00) | (1 <<
       COM0A1);
    TCNT0=0;
    TCCR0B = (1 << CS01) | (1 << CS00);  //
       Prescaler = 64
    OCR0A = 0;  // Set initial duty cycle to 0
}

void Timer1_Init(){
    TCCR1B |= (1<< WGM12); // CTC mode
    TCCR1B |= (1<<CS12) | (1<<CS10);
    OCR1A=15624;
    TIMSK1 |= (1 << OCIE1A);
}
```

Functionality: Sets up Timer0 for PWM and Timer1 for generating interrupts at a specified frequency.

Pin	Function
PD6 (OC0A)	Output pin for PWM signal generation
PD2 (INT0)	Input pin for reading encoder pulses with a pull-up resistor
Port B (MOSI)	Master Out Slave In pin for SPI communication
Port B (MISO)	Master In Slave Out pin for SPI communication
Port B (SS)	Slave Select pin for SPI communication
Port B (SCK)	Clock pin for SPI communication
AREF	Analog reference pin
AVCC	Analog power supply pin
PC0	Analog pin for potentiometer

Table 1: Pins Used in the Project

3 Pins Used

4 Components Used

- **Flyback Diode:** Protects circuit components from inductive loads and MOSFET discharging.
- **1MOhm Resistor:** Used for protecting the MCU from the discharging of MOSFET.
- **100nF Capacitor:** Provides noise immunity as per the datasheet.
- **Motor with Encoder:** Drives the mechanical load and provides feedback.
- **NMOSFET 2N6782:** Used for switching.
- **Potentiometer:** Adjusts the input to the ADC for control purposes.

5 Microcontroller Configuration (ATmega328P)

- **Package:** SPDIL28
- **Primitive:** DIGITAL, ATMEGA328P
- **MODDLL:** AVR2.DLL

- **ITFMOD:** AVR
- **Program:** Path to the compiled program
- **Trace Default:** 1
- **Codegen:** AVRASM2
- **Clock Div 8:** 1
- **CKOUT:** 1
- **RSTDISBL:** 1
- **WDTON:** 1
- **Bootrst:** 1
- **Cksel:** 2
- **Bootsz:** 0
- **Sut:** 2
- **Moddata:** 1024, 255
- **Disasm Bin:** 0

6 SPI Debugger Configuration

- **Primitive:** DIGITAL
- **MODDLL:** spidebug.dll
- **Mode:** Monitor
- **Clock Frequency:** 16MHz
- **Idle State:** 1
- **Word Length:** 16
- **Autoload:** 0
- **Loopback:** 0