Universidade Tecnológica Federal do Paraná

Ciência da Computação

Rafael Henrique Sutil

Resumo do Funcionamento do código em C para o microcontrolador PIC

Ponta Grossa

2023

Funções:

StarButton_RTDS(): Aciona o LED1 e incrementa controle1 se o valor do sensor de freio for maior ou igual a 450. Limpa a flag de interrupção do botão de partida.

ParametrizacaoRetas(): Calcula os coeficientes das retas do acelerador com base em várias variáveis, armazenando os resultados em fAcelerador1 e fAcelerador2. Verifica se o valor do sensor TPS1 é maior que uiVglFinal e atribui 4444 a fAcelerador2 se verdadeiro.

CalculoExtremos(): Calcula os pontos no domínio onde a imagem (torque) é igual a 150, armazenando os resultados em fVglExtremo e fDplExtremo.

CalculoConstante(): Calcula a diferença entre os coeficientes angulares das retas dos sensores TPS.

CalculoConstanteReferenciaLeitura(float fDiferencaAngular): Calcula uma constante de referência a ser comparada com ConstanteReferenciaLeitura, usando várias variáveis.

CalculoConstanteReferenciaEstatica(): Calcula uma constante de referência estática com base em operações matemáticas simples e variáveis.

Aplications(): Loop infinito que executa uma sequência de operações. Realiza leituras analógicas, chama outras funções, realiza operações com bits e envia mensagens CAN.

Lê o valor de iMemoryControl da memória EEPROM e executa um conjunto de operações caso o valor seja igual a 1 e iControleEntradaMemoria seja igual a 0. Em seguida, o código realiza leituras de sensores, cálculos e verificações para processar os dados. Dependendo dos resultados, são definidos valores para variáveis, controlados LEDs e enviadas mensagens CAN. O loop infinito permite a repetição dessas operações com um intervalo de espera de 50ms. Em resumo, a função lida com a leitura de sensores, processamento de dados, controle de LEDs e envio de mensagens CAN.

SetPedalSettings(): Lida com a configuração e coleta de dados dos pedais. Possui quatro casos principais dentro de um switch-case, lendo valores iniciais e finais dos pedais, realizando operações de armazenamento na memória EEPROM e reiniciando as variáveis relacionadas aos pedais.

TimerRTDStart(): Inicia um temporizador (Timer0) e limpa a flag de interrupção correspondente quando ocorre uma interrupção externa no pino INT2.

RTDSTime(): Tratador de interrupção do Timer0. Aciona o LED4 e incrementa iContEstRTDS. Desliga o relé RTDS, LED4 e interrompe o Timer0 quando iContEstRTDS atinge 450.

APPSImplausibility(): Tratador de interrupção do Timer1. Alterna o estado do LED1 e incrementa iContEstAPPSImp. Define iControleTimer2Acel como 1 e redefine iContEstAPPSImp como 0 quando iContEstAPPSImp atinge um determinado valor. Essa função lida com a detecção de implausibilidade nos sensores do acelerador e toma ações correspondentes para corrigir ou lidar com essa situação.

Main(): A função SYSTEM_Initialize() é chamada para inicializar o sistema.
As interrupções globais são habilitadas com INTERRUPT_GlobalInterruptEnable().

São definidos os tratadores de interrupção para os pinos de interrupção INT1, INT0 e INT2 com as funções StarButton_RTDS(), SetPedalSettings() e TimerRTDStart(), respectivamente.

São configuradas as bordas de subida das interrupções externas EXT_INT0, EXT_INT1 e EXT_INT2.

O Timer0 é inicializado com TMR0_Initialize() e é definido o tratador de interrupção RTDSTime() para ele.

O Timer1 é inicializado com TMR1_Initialize() e é definido o tratador de interrupção APPSImplausibility() para ele.

Os timers TMR0 e TMR1 são parados com TMR0_StopTimer() e TMR1_StopTimer().

São realizadas algumas configurações relacionadas às interrupções e à recepção de mensagens CAN.

Em um loop infinito while (1), o código verifica a condição (iControleHabilita != 1) && (controle1 != 0) e executa um bloco de código caso essa condição seja verdadeira.

Dentro desse bloco de código, é enviada uma mensagem CAN, ocorre uma alternância do estado do LED2 e é feita uma pausa de 50ms. Em seguida, outra mensagem CAN é enviada.

Após o bloco de código anterior, o estado do LED3 é alternado quatro vezes com intervalos de 30ms e depois ocorre um intervalo maior de 300ms.

Se controle2 for igual a 1, a função Aplications() é chamada.

O loop while(1) continua executando repetidamente os passos de verificação da condição, alternância do LED3 e chamada da função Aplications().

Função principal do programa. Realiza a inicialização do sistema, incluindo interrupções, timers e configurações do controlador CAN. Em um loop infinito, verifica se

iControleHabilita é igual a 1 e controle1 é diferente de 0. Se a condição for verdadeira, o programa envia mensagens CAN e realiza operações relacionadas aos LEDs.

Resumindo, o código possui funções para lidar com a configuração e coleta de dados dos pedais, inicialização de temporizadores, tratadores de interrupção para os temporizadores e um loop principal para enviar mensagens CAN e controlar os LEDs. O programa também inclui cálculos de coeficientes de retas, cálculos de extremos, cálculos de constantes de referência e outras operações relacionadas ao controle do sistema.