

**UNIVERSIDADE FEDERAL DO MARANHÃO – UFMA**

**CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA – CCET**

**DISCIPLINA: ENGENHARIA ELÉTRICA**

NERVAL DE JESUS SANTOS JUNIOR

Discente

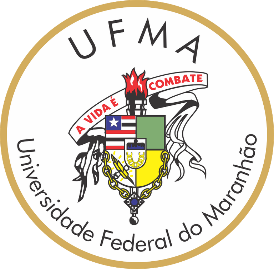
DENIVALDO LOPES

Docente Responsável

**RELATÓRIO DA UNIDADE II SOBRE HCS12**

São Luís - MA

2023



NERVAL DE JESUS SANTOS JUNIOR

Discente

**RELATÓRIO DA UNIDADE II SOBRE HCS12**

Relatório apresentado(a) para disciplina **DEEE0187 – INTROD.À ARQUITET. DE COMPUTADORES (EE) (2023.1-T01)** da Universidade Federal do Maranhão – UFMA, ministrada pelo(a) docente doutor(a), Denivaldo Lopes.

São Luís - MA

2023

**Sumário**

[1. Introdução 4](#_heading=h.2s8eyo1)

[**2. Metodologia 5**](#_heading=)

[3. Objetivos 5](#_heading=h.17dp8vu)

[4. Materiais 6](#_heading=h.3rdcrjn)

[5. Procedimentos 7](#_heading=h.26in1rg)

[6. Resultados e Discussões 28](#_heading=h.lnxbz9)

[7. Conclusão 42](#_heading=h.35nkun2)

[Referências Bibliográficas 43](#_heading=h.1ksv4uv)

# Introdução

Nos últimos anos, os microcontroladores têm desempenhado um papel fundamental no avanço da tecnologia, encontrando aplicações em diversos setores, desde a indústria até dispositivos médicos. Entre os microcontroladores amplamente utilizados, o HCS12 se destaca por sua versatilidade e poder de processamento. Com sua arquitetura robusta e conjunto abrangente de recursos, o HCS12 oferece um ambiente propício para o desenvolvimento de sistemas embarcados. No livro "Embedded Systems: Introduction to ARM Cortex-M Microcontrollers" de Valvano 2012, é apresentada uma introdução abrangente sobre microcontroladores, destacando o HCS12 como um componente central no campo dos sistemas embarcados.

Este trabalho tem como objetivo explorar as funcionalidades e características do microcontrolador HCS12, bem como apresentar soluções para problemas encontrados durante o desenvolvimento de projetos utilizando o ambiente de desenvolvimento Codewarrior.

O microcontrolador HCS12 é um dispositivo eletrônico que tem sido amplamente utilizado em sistemas embarcados, como automação industrial e robótica. Para explorar suas capacidades, é necessário utilizar um ambiente de desenvolvimento integrado (IDE) como o Codewarrior. No entanto, durante o processo de desenvolvimento, é comum encontrar problemas que precisam ser solucionados para garantir o bom funcionamento do sistema. Neste trabalho, o objetivo é demonstrar como é possível utilizar o HCS12 em sistemas embarcados e como solucionar problemas no ambiente de desenvolvimento, utilizando o manual do usuário do HCS12 e a comunidade NXP, que oferecem suporte e soluções para problemas relacionados ao Codewarrior.

Durante o decorrer deste trabalho, serão abordados diferentes aspectos do microcontrolador HCS12, desde sua arquitetura e recursos até a implementação de soluções práticas para problemas encontrados no Codewarrior. Serão exploradas técnicas e estratégias para lidar com erros de compilação, depuração de código, otimização de desempenho e integração de bibliotecas externas.

Ao final deste trabalho, espera-se que o entendimento aprofundado sobre o microcontrolador HCS12 seja sanado, suas características-chave e possíveis soluções para os desafios encontrados no ambiente Codewarrior. Além disso, espera-se que os exemplos práticos apresentados inspirem e auxiliem outros desenvolvedores a explorarem ainda mais as capacidades do HCS12 e a aprimorarem suas habilidades de programação em sistemas embarcados.

# Metodologia

Para a realização adequada do experimento e compreensão do mesmo, serão seguidos os seguintes passos:

1. Primeiramente será utilizado o *software* Logisim e excel, com eles será implementado a tabela-verdade referente ao problema ao qual será feita a solução.
2. Com a tabela pronta, será gerada, através de minimização por mapa de Karnaugh, a expressão booleana correspondente.
3. Com o auxílio da ferramenta online, Draw.io, será feito o diagrama de blocos que servirá como modelo para construção do programa que será executado no microcontrolador.
4. O *software* computacional CodeWarrior será o ambiente de programação, na linguagem C, para implementação do algoritmo de controle.
5. O programa será carregado e executado no microcontrolador HCS12, a fim de ser testado e observado os resultados finais.
6. A nota será baseada na solução dos problemas postos, incluindo execução, teste e análise dos programas feitos em *assembly* e/ou C para HCS12 (conforme especificado em cada questão).

# Objetivos

O objetivo deste relatório é apresentar de forma clara e objetiva os procedimentos e programas que foram desenvolvidos em aula prática, dentro da disciplina de laboratório de arquitetura de computadores. Será feita também a exposição dos resultados obtidos com o experimento e uma conclusão geral a respeito do trabalho realizado.

**Objetivos Específicos**

* Explorar as funcionalidades e características do microcontrolador HCS12.
* Demonstrar a utilização do ambiente de desenvolvimento Codewarrior para projetos com o microcontrolador HCS12.
* Solucionar problemas encontrados durante o desenvolvimento de projetos no ambiente Codewarrior.
* Apresentar implementações práticas que solucionem problemas específicos no ambiente Codewarrior.
* Mostrar a criação de fluxogramas para representar a lógica dos programas desenvolvidos.
* Utilizar a tabela verdade para analisar e implementar expressões booleanas relacionadas aos problemas propostos.
* Aplicar o mapa de Karnaugh para realizar a minimização das expressões booleanas.
* Desenvolver programas em linguagem C e assembly para o microcontrolador HCS12 no ambiente Codewarrior.
* Realizar testes e análises dos programas implementados em relação aos resultados esperados.
* Apresentar os resultados obtidos, demonstrando o sucesso na solução dos problemas propostos utilizando o microcontrolador HCS12 e o ambiente Codewarrior.

Esses objetivos específicos serão alcançados através da metodologia proposta, que envolve o uso do software excel para a implementação da tabela-verdade, a aplicação do mapa de Karnaugh para a minimização das expressões booleanas, o uso do Draw.io para criar o diagrama de blocos e a programação em linguagem C e assembly no ambiente Codewarrior. A execução dos programas no microcontrolador HCS12 permitirá testar e analisar os resultados finais, contribuindo para a obtenção de um entendimento aprofundado sobre o microcontrolador e o ambiente de desenvolvimento. Ao final do trabalho, os resultados serão apresentados de forma clara e objetiva, juntamente com uma conclusão geral sobre o experimento realizado.

# Materiais

Nesse experimento usamos os seguintes materiais:

* **Hardware:**
* - Microcomputador (processador Core i7, RAM 8GB, HD 1TB, Monitor 16”);
* - Kit de desenvolvimento para HCS12: APS12C128SLK [Ref1];
* - Kit de desenvolvimento para HCS12 (placa de expansão): SLK0109UG [Ref2];
* **Software:**
* - Sistema operacional: Windows 10 64 bits;
* - Ambiente de desenvolvimento integrado (IDE): CodeWarrior versão 5.1;

# Procedimentos

Primeiramente, fez-se a seleção das questões 1, 3 e 4 para serem solucionadas.

**Questão 1**

Na questão 1 pede-se que seja feito um programa em *assembly* para HCS12 que corresponda ao programa em linguagem de alto nível a seguir. (Pontos: 3,33)

OBS: Utilizar instruções para manipular valores inteiros de 16 bits (1 word).

**Programa em linguagem de alto nível**

//Questão 1 Pede-se que seja feito um programa em assembly para HCS12 que corresponda ao programa em linguagem de alto nível a seguir: Programa em linguagem de alto nível

void main(){

int X = readPortB();     // X é uma variável que armazena valores inteiros lidos da PortaB

int Y = -3;

int Z = 7;

int W=0;

int R=0;

int T=0;

int K=0;

int S=0;

R = 3\*X + 2 \* Y + Z;

    T =   R +2\*X - Z;       K = 2\*(X + Z);

if(K  <= T){

        S = 2 \* Y + X;

           W= funcA(S,R);

    }else {

        S = funcB(X,T);

        W = funcC(X,S,T); }

}

int funcA(int *parA1*, int *parA2*){

      return 3\* *parA1* + *parA2*;

}

int funcB(int *parB1*, int *parB2*){

        return *parB1* +  *parB2* + 3;

}

int funcC(int *parC1*, int *parC2*, int *parC3*){

        return 2\**parC1* + *parC2* - *parC3*;

}

código em linguagem c- IDE do vscode

Nessa questão pede-se para ser criado um programa dentro do codeWarrior que faça os processos de acordo com o descrito na linguagem de alto nível.

Primeiramente, foi realizada a análise do código de maneira a criar um fluxograma que expressasse o código em assembly. Depois de fazer essa análise é necessário criar o arquivo com todas as dependências que o codewarrior oferece para a simulação. Depois desse processo faz-se necessário desenvolver a mesma lógica do código em linguagem c dentro do arquivo (.asm) que tem a o escopo inicial da estrutura do código em assembly. Então foi necessário fazer a criação de alguns componentes que como a porta B, para ser realizada a leitura da porta depois da compilação.

Foi de fundamental importância os arquivos sobre HCS12 que o professor mandou para funcionar como base. O Datasheet foi muito útil principalmente para entendermos a estrutura eletrônica do HSC12 e poder aplicar de acordo com os comandos em assembly.

**Questão 3**

Na questão 3 pede-se para ser feito uma elaboração de um programa em C para HCS12 que leia um valor short (8 bits) em complemento de dois pela porta A. Depois, calcule o valor fatorial deste valor e o apresente na porta B. (Pontos: 3,33)

No geral, o programa realiza a leitura de um valor short na porta A do microcontrolador, interpreta esse valor como complemento de dois, calcula o fatorial desse valor e apresenta o resultado na porta B. A implementação detalhada desse programa em linguagem C para o microcontrolador HCS12 dependerá das especificações e recursos disponíveis nesse microcontrolador, bem como das bibliotecas e funções disponíveis para a linguagem de programação utilizada.

satisfazendo o que está escrito a seguir foi possível solucionar o problema

* Leia um valor short de 8 bits pela porta A do microcontrolador.
* O valor lido na porta A é interpretado como um número em complemento de dois.
* Calcule o valor fatorial com base no valor posto na porta A.
* Apresente o resultado do fatorial na porta B do microcontrolador.
* Para entender melhor como o programa funciona, vamos analisar cada etapa separadamente.

O microcontrolador HCS12 possui portas de entrada e saída para comunicação com o mundo externo. No programa, a leitura do valor short é realizada pela porta A, que é uma porta de entrada. Ao ler o valor, o programa armazena esse valor em uma variável na memória do microcontrolador para posterior processamento.

O complemento de dois é uma técnica para representar números negativos em binário. Neste caso, o valor lido pela porta A é interpretado como um número em complemento de dois, o que significa que se for um valor negativo, ele será representado de maneira apropriada.

O próximo passo é calcular o valor fatorial do valor lido pela porta A. O fatorial de um número é o produto de todos os números inteiros positivos menores ou iguais a esse número. Portanto, o programa precisa implementar um algoritmo para calcular o valor fatorial do valor lido.

Após calcular o fatorial, o programa deve apresentar o resultado na porta B do microcontrolador. A porta B é uma porta de saída que permite que o microcontrolador envie dados para dispositivos externos.

Para solução desta questão foi necessário fazer a tabela verdade da terceira questão como mostrado na Tabela 1 a seguir. Por meio dela foi possível observar como a relação entre o valor de entrada que é lido pela porta A e sai depois de passar pela função de fatorial os resultados obtidos na porta B.

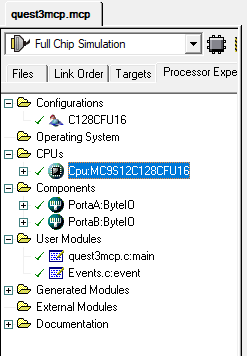
Tabela 1-Tabela verdade da questão de fatorial

|  |  |
| --- | --- |
| **Tabela da Verdade da terceira Questão** | |
| **valor (A)** | **fatorial (A!)** |
| 0 | 1 |
| 1 | 1 |
| 2 | 2 |
| 3 | 6 |
| 4 | 24 |
| 5 | 120 |
| 6 | 720 |
| 7 | 5040 |
| 8 | 40320 |
| 9 | 362880 |
| 10 | 3628800 |

Fonte: Autoria própria.

A partir da tabela verdade mostrada anteriormente foi possível fazer a criação do arquivo para simulação do programa dentro do CodeWarrior. Foi criado então o arquivo mostrado na Figura 1 a seguir. Neste arquivo podemos ver algumas configurações iniciais necessárias para solução do problema. primeiramente foi criado os componentes de entrada de sinal (Porta A) e saída de sinal (Porta B).

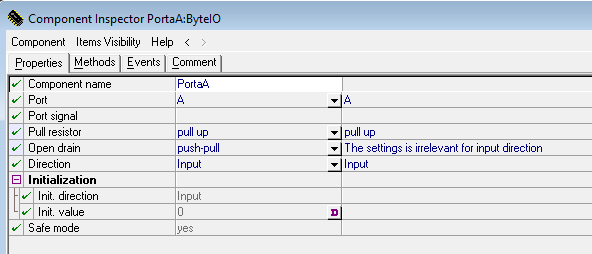
Figura 1- arquivo da questão fatorial



Fonte: Autoria própria.

O Codewarrior é uma poderosa plataforma de desenvolvimento integrado (IDE) projetada especificamente para atender às necessidades dos desenvolvedores de sistemas embarcados. Ele oferece uma interface intuitiva e uma ampla gama de recursos para facilitar o processo de desenvolvimento, como compiladores, depuradores e emuladores. No entanto, assim como em qualquer ambiente de programação, os desenvolvedores podem se deparar com desafios e problemas ao utilizar o Codewarrior.

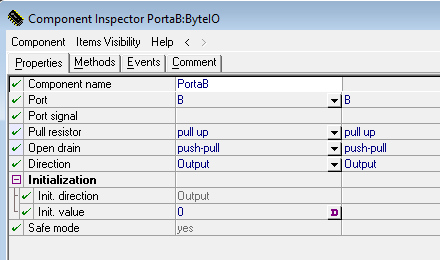
Nas Portas foram feitas algumas alterações quanto ao nome da porta e o tipo de função que ela vai executar, se é do tipo input ou output como mostrado na Figura 2.

Figura 2- configuração das Portas

Fonte: Autoria própria.

Na figura 3 a seguir é possível ver a configuração de saída para Porta B.

Figura 3- configuração da Porta B.



Fonte: Autoria própria.

A seguir o código completo da questão de número 3 implementado na linguagem de programação de alto nível, linguagem C,por ser uma bem mais próxima da linguagem de máquina e pedida no problema em questão.

**Programa em linguagem de alto nível**

/\*\* ###################################################################

\*\* Filename : quest3mcp.c

\*\* Project : quest3mcp

\*\* Processor : MC9S12C128CFU16

\*\* Version : Driver 01.14

\*\* Compiler : CodeWarrior HC12 C Compiler

\*\* Date/Time : 01/07/2023, 21:00

\*\* Abstract :

\*\* Main module.

\*\* This module contains user's application code.

\*\* Settings :

\*\* Contents :

\*\* No public methods

\*\*

\*\* ###################################################################\*/

/\* MODULE quest3mcp \*/

/\*

UNIVERSIDADE FEDERAL DO MARANHÃO - UFMA

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA - CCET

DISCIPLINA: ENGENHARIA ELÉTRICA

NERVAL DE JESUS SANTOS JUNIOR

Discente

DENIVALDO LOPES

Docente Responsável

RELATÓRIO DA UNIDADE II SOBRE HCS12

São Luís - MA

2023

\*/

/\* Including needed modules to compile this module/procedure \*/

#include "Cpu.h"

#include "Events.h"

#include "PortaA.h"

#include "PortaB.h"

/\* Include shared modules, which are used for whole project \*/

#include "PE\_Types.h"

#include "PE\_Error.h"

#include "PE\_Const.h"

#include "IO\_Map.h"

/\*\*\* End of Processor Expert internal initialization. \*\*\*/

/\* User includes (#include below this line is not maintained by Processor Expert) \*/

#include <math.h>

void main(void)

{

/\* Write your local variable definition here \*/

/\* put your own code here \*/

/\* Write your local variable definition here \*/

short valueA;

short factorial = 1;

int i;

/\*\*\* Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! \*\*\*/

PE\_low\_level\_init();

/\*\*\* End of Processor Expert internal initialization.\*\*\*/

/\* Write your code here \*/

valueA = PortaA\_GetVal(); // Ler o valor da porta A

if (valueA < 0) {

// Converter o valor em complemento de dois para valor positivo

valueA = valueA & 0xFF; // Limpar os bits de sinal (mais significativo)

valueA = -valueA; // Inverter o sinal

}

// Calcular o fatorial do valor

for (i = 1; i <= valueA; i++) {

factorial \*= i;

}

PortaB\_PutVal(factorial);

for(;;){}

/\*\*\* Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! \*\*\*/

} /\*\*\* End of main routine. DO NOT MODIFY THIS TEXT!!! \*\*\*/

/\* END quest3mcp \*/

/\*

\*\* ###################################################################

\*\*

\*\* This file was created by Processor Expert 3.05 [04.46]

\*\* for the Freescale HCS12 series of microcontrollers.

\*\*

\*\* ###################################################################

\*/

código em linguagem c- IDE do vscode

**Questão 4**

Na questão de número 4 pede-se a implementação de um sistema de software/hardware baseado no HCS12 deve obter as raízes de uma equação do 2º grau, dada por a\*x2+b\*x+c=0, onde a é lido pela Porta A, b é lido pela porta B e c é lido pela Porta T. Os valores de a, b e c são lidos como valores inteiros em complemento de dois. As raízes desta equação devem ser armazenadas em variáveis denominadas x1 e x2. Elabore este programa em linguagem C.

Resposta :

A premissa do programa é desenvolver um sistema de software/hardware baseado no microcontrolador HCS12 que obtenha as raízes de uma equação do 2º grau, representada por ax^2 + bx + c = 0. Nesse sistema, os valores de a, b e c são lidos pelas portas A, B e T, respectivamente, e são interpretados como valores inteiros em complemento de dois. As raízes da equação devem ser armazenadas nas variáveis x1 e x2.

No geral, o programa implementa um sistema de software/hardware no microcontrolador HCS12 para obter as raízes de uma equação do 2º grau. Ele realiza a leitura dos valores de a, b e c pelas portas A, B e T, interpreta esses valores como complemento de dois, calcula as raízes da equação e as armazena nas variáveis x1 e x2. As saídas do programa dependem dos requisitos específicos e podem envolver exibição ou envio das raízes para dispositivos externos. A implementação detalhada desse programa em linguagem C para o microcontrolador HCS12 dependerá das especificações e recursos disponíveis nesse microcontrolador, bem como das bibliotecas e funções disponíveis na linguagem de programação.

A seguir, vamos analisar como cada aspecto do programa funcionaria em relação à linguagem de programação e ao microcontrolador HCS12:

Leitura dos valores de a, b e c pelas portas A, B e T:

O programa deve realizar a leitura dos valores de a, b e c pelas portas A, B e T, respectivamente, do microcontrolador HCS12. Cada valor lido será armazenado em uma variável na memória do microcontrolador para uso posterior.

Interpretação dos valores lidos como complemento de dois:

Os valores de A, B e C lidos pelas portas A, B e T são interpretados como valores inteiros em complemento de dois. Isso significa que, se algum desses valores for negativo, eles serão representados adequadamente nesse formato.

Cálculo das raízes da equação do 2º grau:

O próximo passo é calcular as raízes da equação do 2º grau usando os valores de a, b e c obtidos. A fórmula geral para calcular as raízes é x = (-b ± √(b^2 - 4ac)) / (2a). O programa deve implementar essa fórmula para encontrar as raízes da equação e armazená-las nas variáveis x1 e x2.

Armazenamento das raízes nas variáveis x1 e x2:

As raízes da equação calculadas no passo anterior devem ser armazenadas nas variáveis x1 e x2. Essas variáveis serão utilizadas posteriormente para qualquer processamento ou exibição dos resultados.

Saídas do programa:

Após calcular as raízes e armazená-las nas variáveis x1 e x2, o programa pode realizar diversas ações com esses valores, dependendo dos requisitos do sistema. Por exemplo, pode-se exibir as raízes em um display, enviá-las para um dispositivo externo ou utilizá-las para outras operações no próprio microcontrolador.

* O sistema de software/hardware é baseado no HCS12.
* A equação do 2º grau, ax2+bx+c=0, deve ser resolvida.
* Os valores dos coeficientes a, b e c são lidos pelas portas A, B e T, respectivamente.
* Os valores lidos são inteiros em complemento de dois.
* As raízes da equação devem ser armazenadas nas variáveis x1 e x2.
* O programa deve ser elaborado em linguagem C.

Quanto a uma tabela, você pode criar uma tabela mostrando os valores lidos pelos registradores de porta A, B e T, e os valores calculados das raízes x1 e x2. Algo assim:

Tabela 2 - Tabela verdade questão 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabela da Verdade da quarta Questão** | | | | |
| **Porta A** | **Porta B** | **Porta T** | **X1** | **X2** |
| **A** | **B** | **C** | **saida x1** | **saida x22** |
| leitura da porta | leitura da porta | leitura da porta | raiz 1 | raiz 2 |

Fonte: Autoria própria

Com base na tabela verdade foi feita a criação do diagrama de bloco dentro do Draw.io da questão 4, ela foi de fundamental importância para criação da lógica de programação que soluciona o problema.

A seguir o código inteiro da resolução da questão 4.

**Programa em linguagem de alto nível**

/\*\* ###################################################################

\*\* Filename : HelloWorld\_C.c

\*\* Project : HelloWorld\_C

\*\* Processor : MC9S12C128CFU16

\*\* Version : Driver 01.14

\*\* Compiler : CodeWarrior HC12 C Compiler

\*\* Date/Time : 30/06/2023, 18:30

\*\* Abstract :

\*\* Main module.

\*\* This module contains user's application code.

\*\* Settings :

\*\* Contents :

\*\* No public methods

\*\*

\*\* ###################################################################\*/

/\* MODULE HelloWorld\_C \*/

/\*

UNIVERSIDADE FEDERAL DO MARANHÃO - UFMA

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA - CCET

DISCIPLINA: ENGENHARIA ELÉTRICA

NERVAL DE JESUS SANTOS JUNIOR

Discente

DENIVALDO LOPES

Docente Responsável

RELATÓRIO DA UNIDADE II SOBRE HCS12

São Luís - MA

2023

\*/

/\* Including needed modules to compile this module/procedure \*/

#include "Cpu.h"

#include "Events.h"

/\* Include shared modules, which are used for whole project \*/

#include "PE\_Types.h"

#include "PE\_Error.h"

#include "PE\_Const.h"

#include "IO\_Map.h"

/\* User includes (#include below this line is not maintained by Processor Expert) \*/

#include <math.h>

float x1, x2,delta;

float a,b,c;

float x1Imag, x2Imag;

void main(void)

{

/\* Write your local variable definition here \*/

/\*\*\* Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! \*\*\*/

PE\_low\_level\_init();

/\*\*\* End of Processor Expert internal initialization. \*\*\*/

/\* Write your code here \*/

a=PORTA;

b=PORTB;

c=PTT;

x1=0;

x2=0;

delta=0;

x1Imag=0;

x2Imag=0;

// função do delta da equação do segundo grau

delta=b\*b - 4\*a\*c;

if(delta ==0){

// raizes reais iguais

x1=((float)-b)/(2.0\*a);

x2=x1;

}else if(delta > 0){

x1= (-b + sqrtf(delta))/(2.0f\*a);

x2= (-b - sqrtf(delta))/(2.0f\*a);

}else{

x1= (-b)/(2.0f\*a);

x2= (-b)/(2.0f\*a);

x1Imag= -(sqrtf(delta))/(2.0f\*a);

x2Imag= (sqrtf(delta))/(2.0f\*a);

}

/\*\*\* Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! \*\*\*/

for(;;){}

/\*\*\* Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! \*\*\*/

} /\*\*\* End of main routine. DO NOT MODIFY THIS TEXT!!! \*\*\*/

/\* END HelloWorld\_C \*/

/\*

\*\* ###################################################################

\*\*

\*\* This file was created by Processor Expert 3.05 [04.46]

\*\* for the Freescale HCS12 series of microcontrollers.

\*\*

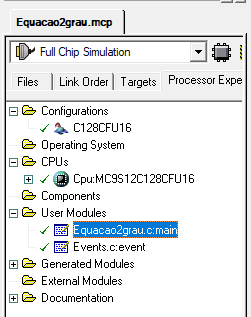
\*\* ###################################################################

\*/

código em linguagem c- IDE do vscode

No programa foi necessário criar alguns componentes de entradas das variáveis que passaram pela função para obter os resultados das raízes da equação do segundo grau. Na Figura 4 a seguir temos o escopo com as configurações iniciais do programa dentro do CodeWarrior.

Figura 4- arquivo da questão da equação do segundo grau



Fonte: Autoria própria.

**Questão 5**

Na questão de número 5 pede-se para ser feito um programa em linguagem C para o HCS12 que faça o controle da seguinte planta. (Pontos: 3,33)

O terceiro experimento consiste em fazer o controle de um tanque conforme o enunciado. A seguir é possível ver a imagem de ilustração do tanque.

Tabela verdade e obtenção da expressão simplificada:

O primeiro passo é construir a tabela verdade que relaciona as entradas (sensores de nível A, B, C e D e o interruptor I) com as saídas (ligamento das eletroválvulas Evs e Eve). Com base na tabela verdade, é possível obter a expressão simplificada por meio do mapa de Karnaugh. Essa expressão simplificada será utilizada para controlar o ligamento das eletroválvulas.

Entrada dos sinais dos sensores de nível e interruptor I:

Os sensores de nível A, B, C e D, juntamente com o interruptor I, são utilizados como entradas para o programa. Os sensores de nível são lidos pelas portas PB0, PB1, PB2 e PB3 do microcontrolador HCS12, respectivamente, enquanto o interruptor I é lido pela porta PB0. Esses valores são interpretados como sinais de entrada no programa para determinar as condições de controle.

Ligamento das eletroválvulas de saída (Evs e Eve):

Com base nas condições especificadas, o programa controla o ligamento das eletroválvulas de saída Evs (ligada à porta PB4) e Eve (ligada à porta PB5). O ligamento das eletroválvulas é determinado pelas condições estabelecidas na premissa, como o nível do líquido e o estado do interruptor I. O programa liga ou desliga as eletroválvulas de acordo com essas condições.

Sinal de alerta para mau funcionamento do sistema:

Além do controle das eletroválvulas, o programa também utiliza a porta PB7 para emitir um sinal de alerta em caso de mau funcionamento do sistema. Isso é feito com base nas condições especificadas, como sensores ativados ou desativados de forma inadequada. O programa monitora essas condições e emite um sinal de alerta, se necessário.

O programa em linguagem C para o microcontrolador HCS12 deve ser implementado seguindo o fluxograma que representa a solução do problema. A implementação detalhada desse programa dependerá das especificações e recursos disponíveis no microcontrolador HCS12, bem como das bibliotecas e funções disponíveis na linguagem de programação utilizada.

Figura 5- arquivo da questão fatorial



Fonte: Autoria própria.

Um reservatório para armazenar um determinado produto líquido deve ser controlado da seguinte forma:

* + - Ao pressionar o botão *I*, o sistema deve ser acionado;
    - Uma vez que o sistema tenha sido acionado, a eletroválvula *Evs* deve ser ligada se, e somente se, o líquido estiver no nível A ou nível B. Se o líquido estiver no nível D, então a eletroválvula *Evs* deve ser desligada até que o nível B seja atingido;
    - A eletroválvula *Eve* deve ser ligada se, e somente se, o botão *I* estiver ligado e o nível do líquido estiver no nível D ou C ou B. No nível A, a eletroválvula *Eve* deve ser desligada;
    - Caso o sistema esteja ligado e seja pressionado o botão *I* para desligá-lo, então a Eletroválvula *Eve* deve ser desligada. Em seguida, todo o líquido deve ser escoado pela Eletroválvula *Evs*, até que o sensor D acuse reservatório vazio (sensor desativado).

OBS: Utilize os seguintes bits das portas:

* + - PB0 como entrada para I;
    - PP0 como entrada para A;
    - PB1 como entrada para B;
    - PB2 como entrada para C;
    - PB3 como entrada para D;
    - PB4 como saída para Evs;
    - PB5 como saída para Eve;
    - PB7 como sinal de alerta para mau funcionamento do sistema, por exemplo: sensor A e B ativos, mas sensor C desativado; ou sensor A ativado, mas sensores B e C desativados; ou B ativado, mas sensor C desativado.

OBS: Faça a tabela da verdade, depois obtenha a expressão simplificada pelo mapa de Karnaugh, faça o fluxograma da solução do problema, escreva o programa em linguagem C para HCS12.

Código completo da questão 5 a seguir:

**Programa em linguagem de alto nível**

/\*\* ###################################################################

\*\* Filename : tanque.c

\*\* Project : tanque

\*\* Processor : MC9S12C128CFU16

\*\* Version : Driver 01.14

\*\* Compiler : CodeWarrior HC12 C Compiler

\*\* Date/Time : 19/04/2023, 14:13

\*\* Abstract :

\*\* Main module.

\*\* This module contains user's application code.

\*\* Settings :

\*\* Contents :

\*\* No public methods

\*\*

\*\* ###################################################################\*/

/\* MODULE tanque \*/

/\*

UNIVERSIDADE FEDERAL DO MARANHÃO - UFMA

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA - CCET

DISCIPLINA: ENGENHARIA ELÉTRICA

NERVAL DE JESUS SANTOS JUNIOR

Discente

DENIVALDO LOPES

Docente Responsável

RELATÓRIO DA UNIDADE II SOBRE HCS12

São Luís - MA

2023

\*/

/\* Including needed modules to compile this module/procedure \*/

#include "Cpu.h"

#include "Events.h"

#include "I.h"

#include "A.h"

#include "B.h"

#include "C.h"

#include "D.h"

#include "E.h"

#include "Erro.h"

#include "S.h"

/\* Include shared modules, which are used for whole project \*/

#include "PE\_Types.h"

#include "PE\_Error.h"

#include "PE\_Const.h"

#include "IO\_Map.h"

/\* User includes (#include below this line is not maintained by Processor Expert) \*/

int valueE;

int valueS;

int error;

int SYSTEM;

void main(void)

{

/\* Write your local variable definition here \*/

int valueSensorA;

int valueSensorB;

int valueSensorC;

int valueSensorD;

/\*\*\* Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! \*\*\*/

PE\_low\_level\_init();

/\*\*\* End of Processor Expert internal initialization. \*\*\*/

/\* Write your code here \*/

while(1)

{

SYSTEM = I\_GetVal();

valueSensorA = A\_GetVal();

valueSensorB = B\_GetVal();

valueSensorC = C\_GetVal();

valueSensorD = D\_GetVal();

if(!SYSTEM){

if ((((!valueSensorA && !valueSensorB && !valueSensorD) || (!valueSensorA && valueSensorC && !valueSensorD))) != 0)

{

E\_ClrVal();

}

else

{

E\_SetVal();

}

if ((valueSensorB && valueSensorC && !valueSensorD) != 0)

{

S\_ClrVal();

}

else

{

S\_SetVal();

}

if ((valueSensorD || (valueSensorB && !valueSensorC) || (valueSensorA && !valueSensorB)) != 0)

{

Erro\_ClrVal();

}

else

{

Erro\_SetVal();

}

}

}

/\*\*\* Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! \*\*\*/

for(;;){}

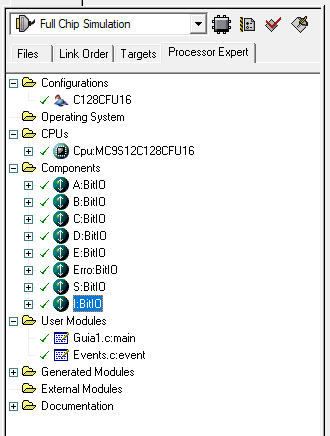
/\*\*\* Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! \*\*\*/

}

código em linguagem c- IDE do vscode

Foi necessário para a questão 5 fazer algumas configurações e criação de componentes, os sensores que temos descritos a seguir na Figura 6, sensores A,B,C e D, interruptor I, e os atuadores Evs e Eve. Além disso, temos o erro como um sinal de alerta.

Figura 6- arquivo da questão fatorial

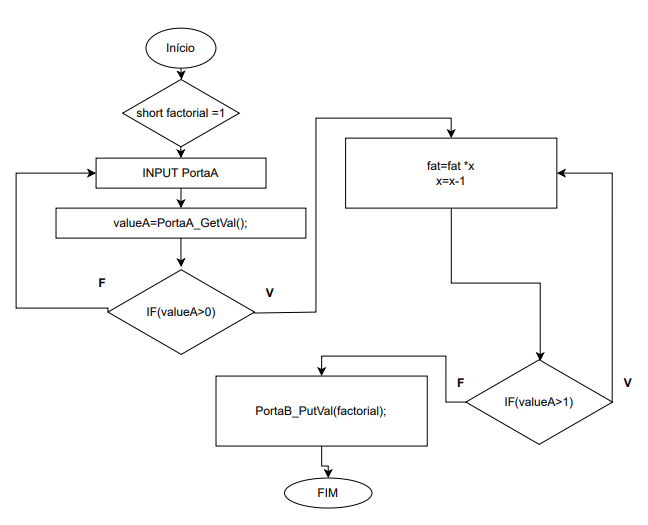


Fonte: Autoria própria.

# Resultados e Discussões

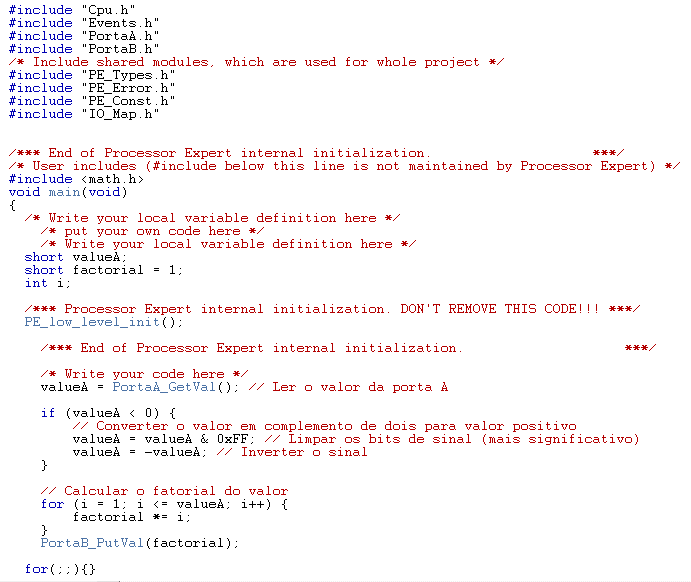
**Questão 3**

Antes de ser implementada a lógica de programação dentro do CodeWarrior foi criado um fluxograma que expressasse a resolução do problema em questão. A seguir na Figura 7 é possível visualizar o fluxograma desenvolvido dentro do Draw.io para criar o diagrama de blocos.

Figura 7- Fluxograma questão 3

Fonte: Autoria própria

Depois de feitas as configurações foi necessário implementar o código pensado no fluxograma como mostrado acima. o código dentro do codeWarrior está implementado como mostrado na Figura 8.

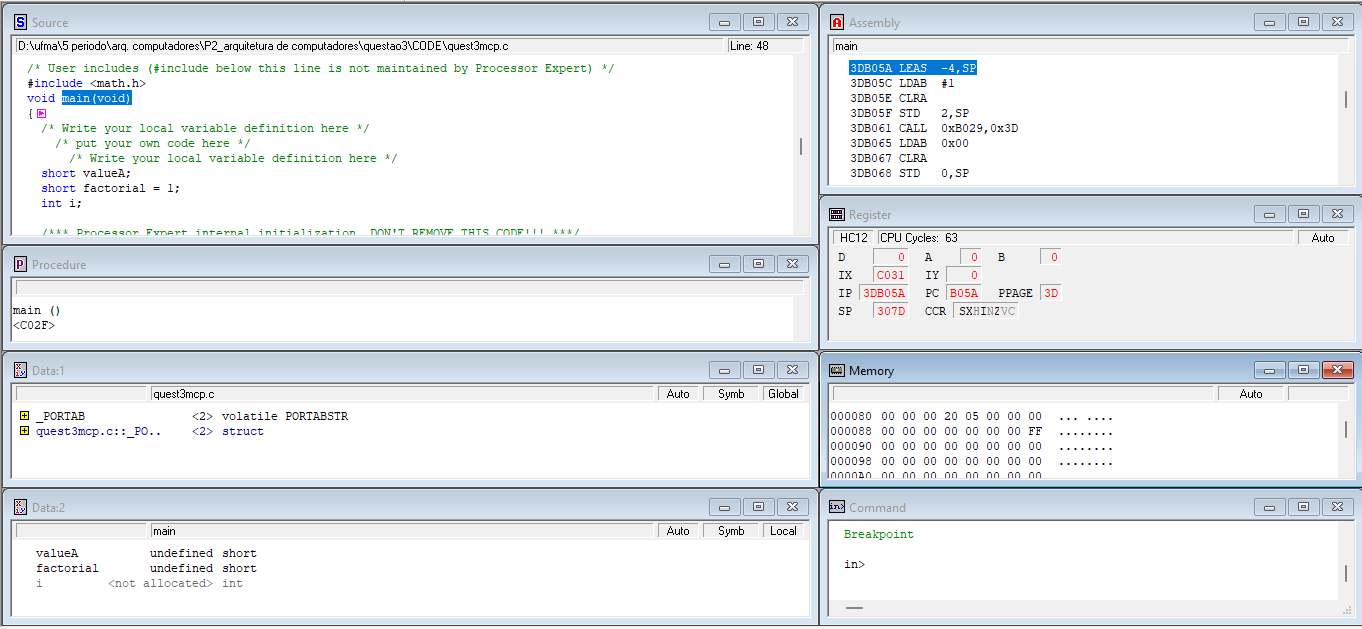
Figura 8 - Código do cálculo de fatorial 

Fonte: Autoria própria.

Depois o código foi executado no microcontrolador e os resultados foram apurados.

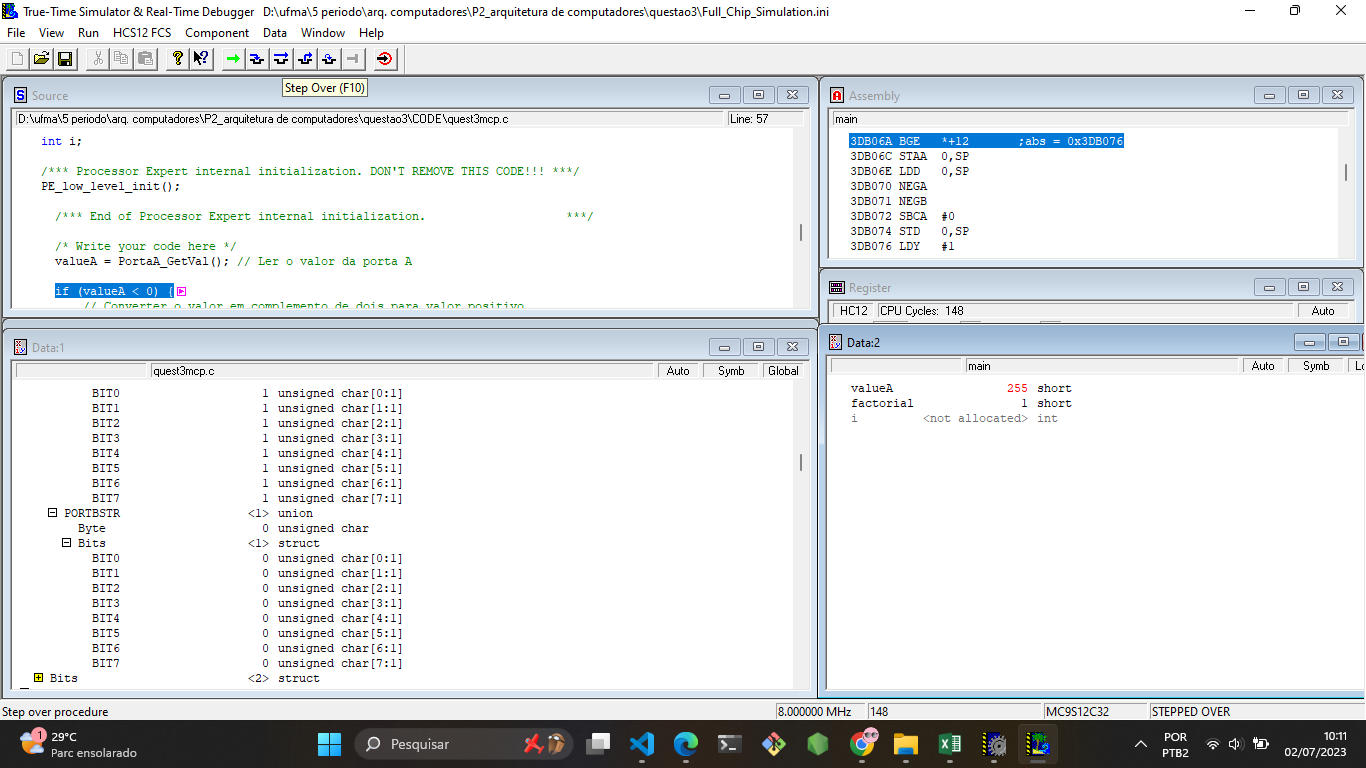
A seguir na Figura 9 temos a simulação do programa.

Figura 9- simulação do fatorial.



Fonte: Autoria própria.

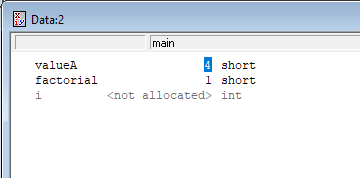
Como podemos observar na figura 10 a seguir podemos apertar a tecla F10 para rodarmos nosso programa passo a passo.

Figura 10- rodando o programa do fatorial

Fonte: Autoria própria.

Antes de chegar na parte de condições estabelecidas podemos modificar o valor da variável de entrada da Porta A como mostrado na Figura 11 a seguir. Como teste coloquei o valor 4 na Porta A.

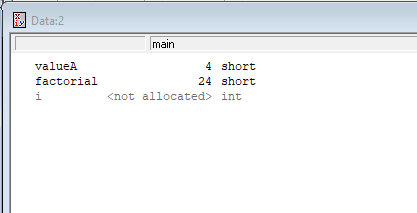
Figura 11-alterando o valor de entrada da Porta A



Fonte: Autoria própria.

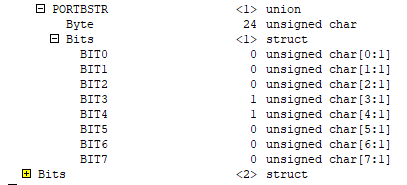
Depois de rodar o programa ele executa o fatorial e retorna o fatorial do valor 4 na Porta B que é igual a 24 como mostrado na Figura 12.

Figura 12- obtendo o resultado do fatorial na Porta B



Fonte: Autoria própria.

Além disso, se observarmos os bits da Porta B podemos perceber que eles também modificam de acordo com o valor de saída na Porta B, onde podemos colocar o microcontrolador, por exemplo, para ligar leds como sinal de saída para termos o valor fatorial em um exemplo prático. Na figura 13 podemos ver os valores de saída de cada Bit na Porta B.

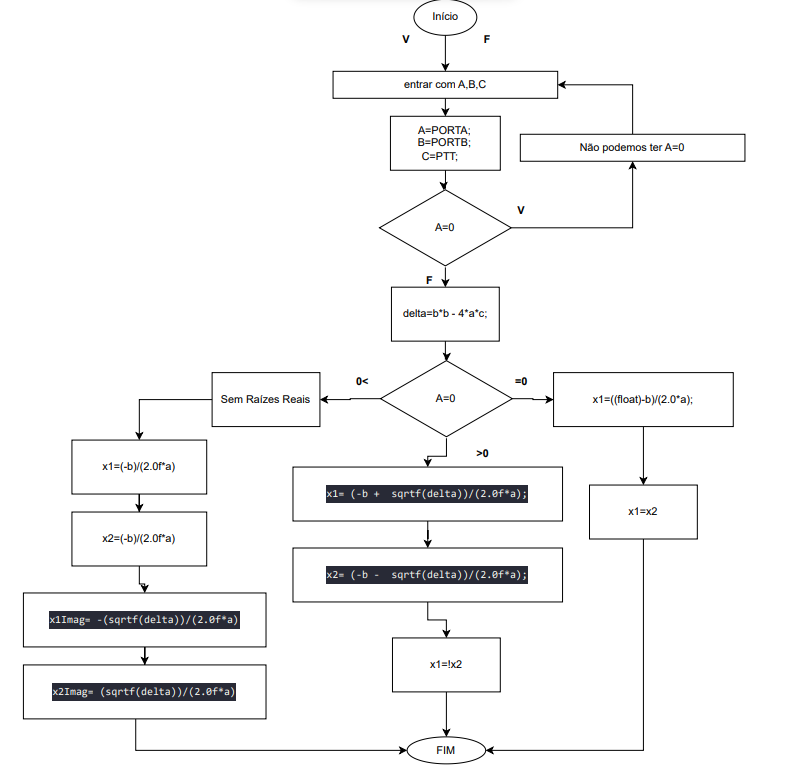
Figura 13- saídas dos bits referente ao resultado do fatorial

Fonte: Autoria própria.

**Questão 4**

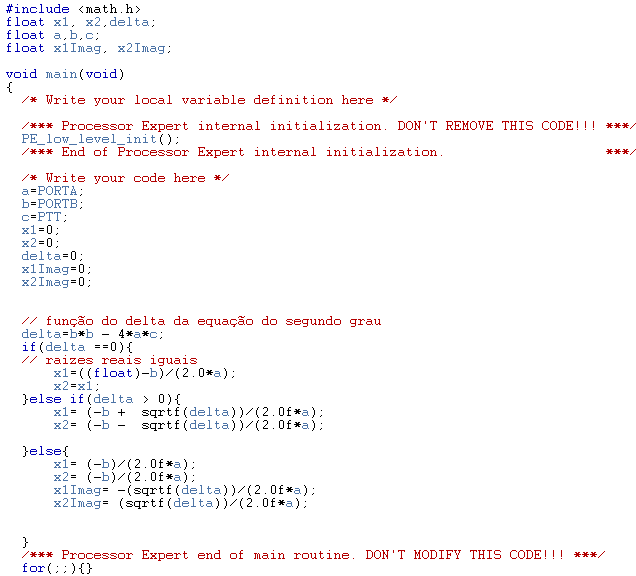
A seguir na Figura 14 podemos ver o diagrama de blocos da questão 4. De posse da mesma foi possível entender de maneira mais ampla como seria a resolução de uma equação do segundo grau, para que posteriormente fosse possível desenvolver ela para um programa em HSC12, o qual teríamos as saídas em binário expressos nas portas de saída.

Figura 14- Fluxograma questão 4



Fonte: Autoria própria

Na figura 15 é possível ver o código com as variáveis necessárias para equação do segundo grau, temos os valores de entrada A,B e C e os valores de saída x1,x2 e delta . O código em linguagem C será o seguinte:

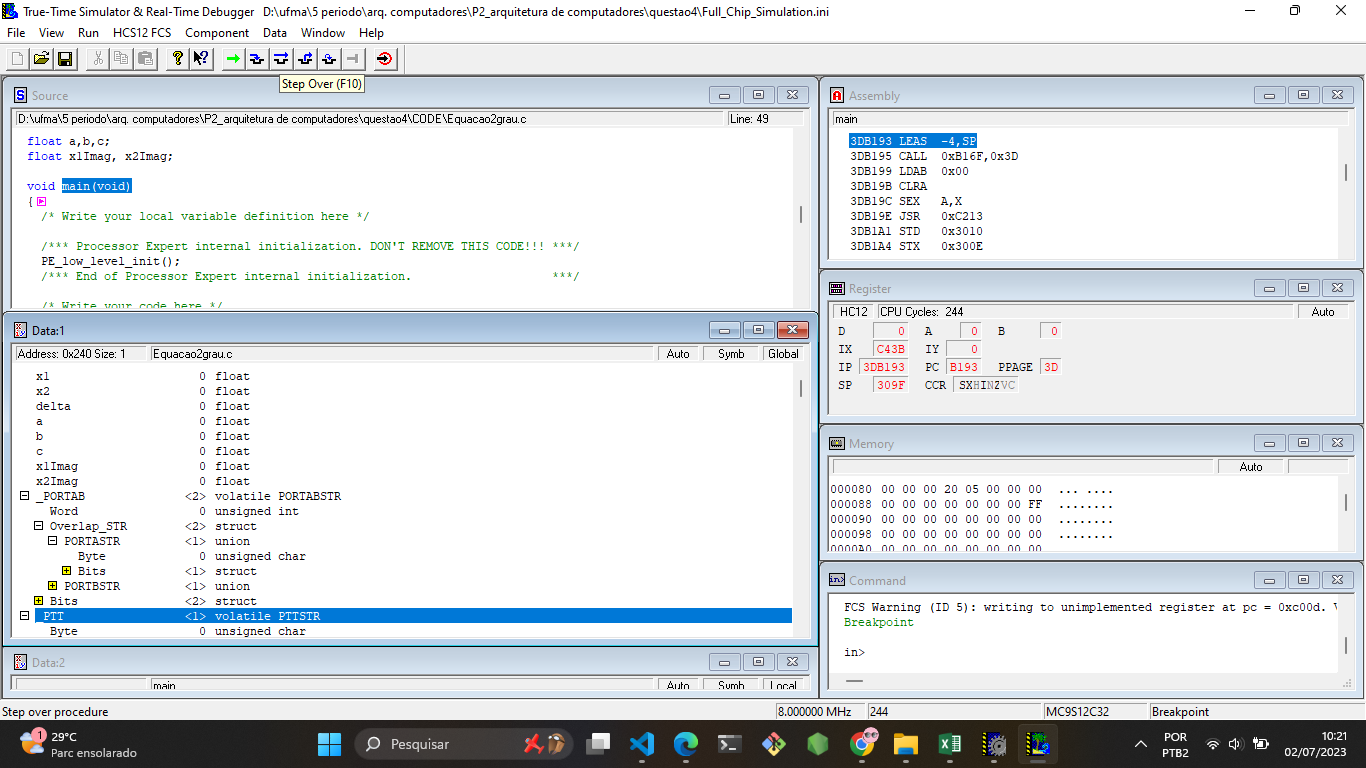
Figura 15 - código da questão da equação do segundo grau

Fonte: Autoria própria.

Depois o código foi executado no microcontrolador e os resultados foram apurados.

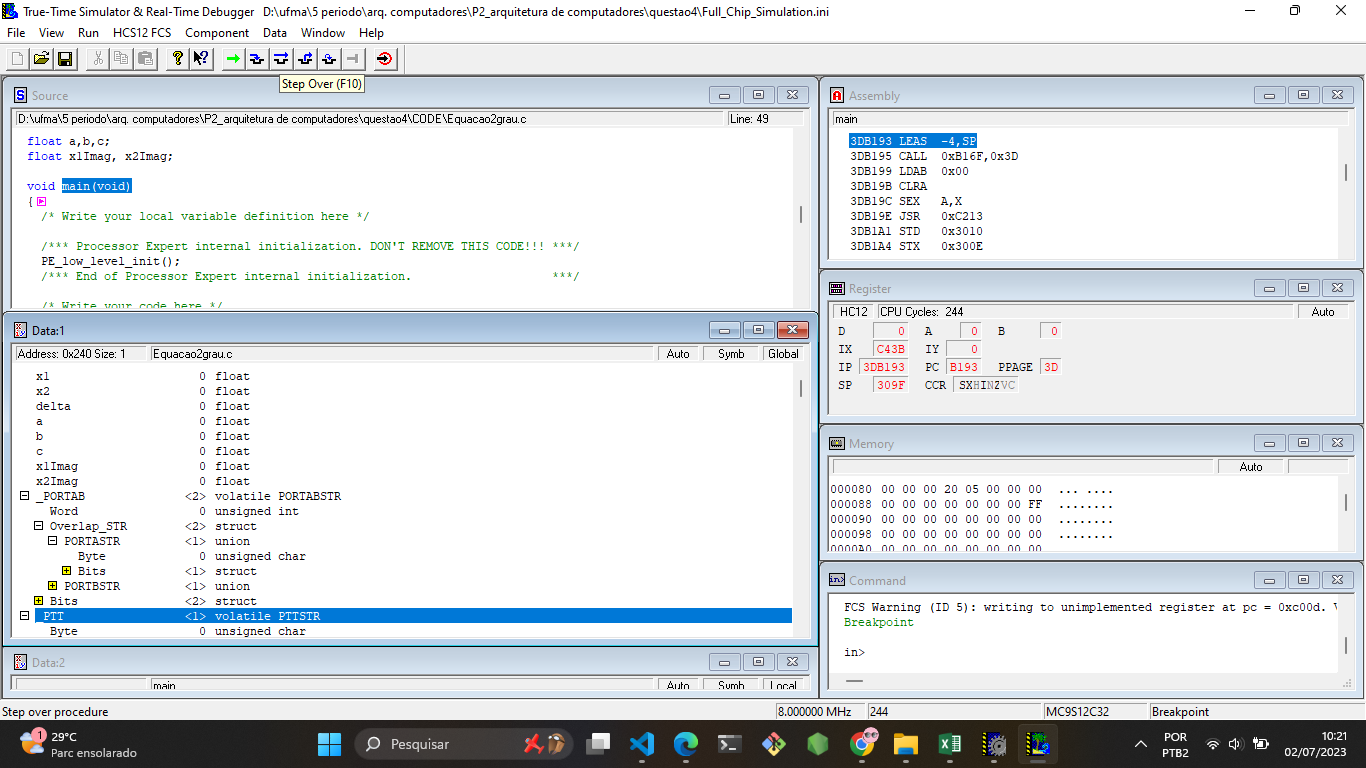
Na figura 16 podemos ver a compilação do programa dentro do codewarrior.

Figura 16- simulação da equação do segundo grau



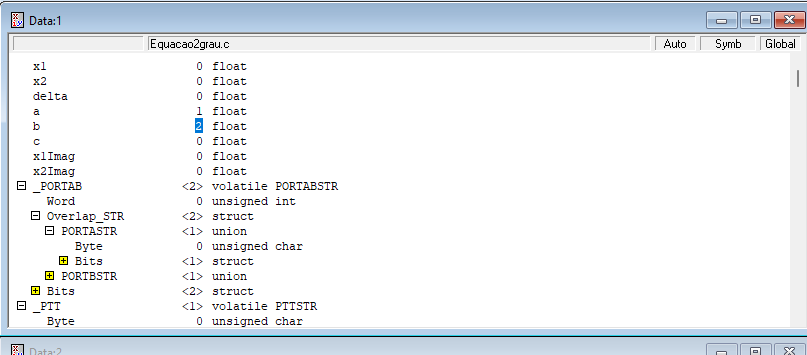
Fonte: Autoria própria.

Apertando a tecla F10 conseguimos rodar as linhas de código passo a passo como mostrado na Figura 17 a seguir.

Figura 17 - rodando o programa 

Fonte: Autoria própria.

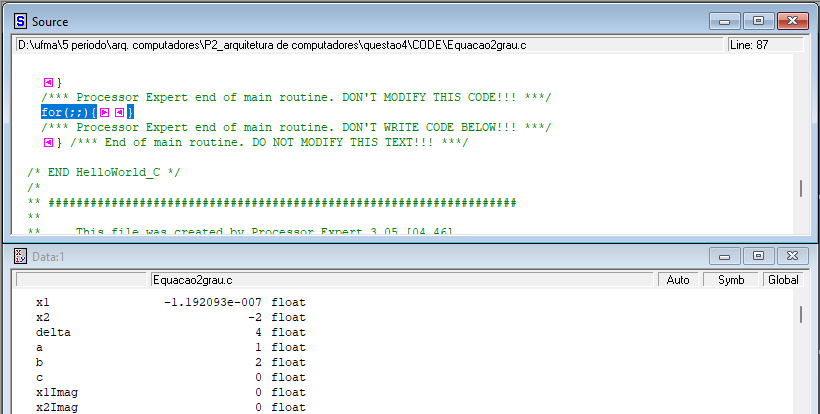
Na figura 18, foram colocados valores nas portas A,B e T como pedido na questão. A partir delas, ao rodar o programa ele vai retornar na saída do x1,x2 e delta de acordo com a equação do segundo grau.

Figura 18 - modificando entradas a,b,c

Fonte: Autoria própria.

Depois de modificado os valores de entrada, podemos continuar rodando o programa até que ele execute totalmente e mostre os valores de saída como mostrado na Figura 19.

Figura 19- saídas nas raízes delta, x1,x2



Fonte: Autoria própria.

**Questão 5**

Na Tabela 3 abaixo conseguimos ver os resultados da tabela verdade que soluciona o problema da questão 5. As saídas e entradas estão descritas como pedidas pela questão.

Tabela 3 - arquivo da questão do Tanque

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Tabela da Verdade da quinta Questão** | | | | | | | | |
| **PB0(I)** | **PB3(D)** | **PB2(C)** | **PP1(B)** | **PP0(A)** | **PB4(Evs)** | **PB5(Eve)** | **PB7(Erros)** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

Fonte: Autoria própria.

Expressões extraídas através do mapa de Karnaugh:

Entrada:

Tabela 4 – Mapa de Karnaugh de Eve(PB5)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **\*Mapa de Karnaugh de Eve(PB5):** | | | **IC'B'A'+IDCA'** | | |
| I | | CD | | | |
| 00 | 01 | 11 | 10 |
| AB | 00 | 1 | 1 | 1 | 0 |
| 01 | 0 | 0 | 1 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

Fonte: Autoria própria.

Saída:

Tabela 5 – Mapa de Karnaugh de Evs(PB4)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **\*Mapa de Karnaugh de Evs(PB4):** | | | **I'D+DCB** | | |
|  | | CB | | | |
| 00 | 01 | 11 | 10 |
| ID | 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 1 | 1 | 1 |
| 11 | 0 | 0 | 1 | 0 |
| 10 | 0 | 0 | 0 | 0 |

Fonte: Autoria própria.

Erro:

Tabela 6 – Mapa de Karnaugh de Erro(PB7)

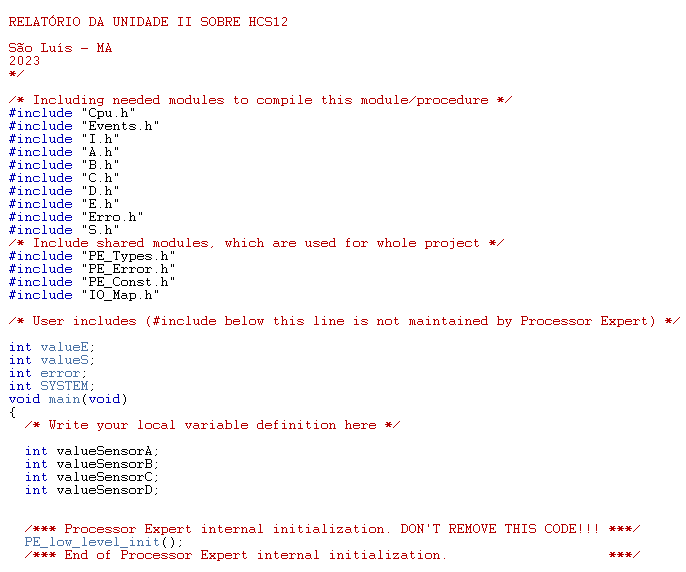
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **\*Mapa de Karnaugh de Erros(PB7):** | | | **ID'C+IC'B+IB'A** | | |
| I | | CD | | | |
| 00 | 01 | 11 | 10 |
| AB | 00 | 0 | 0 | 0 | 1 |
| 01 | 1 | 1 | 0 | 1 |
| 11 | 1 | 1 | 0 | 1 |
| 10 | 1 | 1 | 1 | 1 |

Fonte: Autoria própria.

O código em linguagem C será o seguinte:

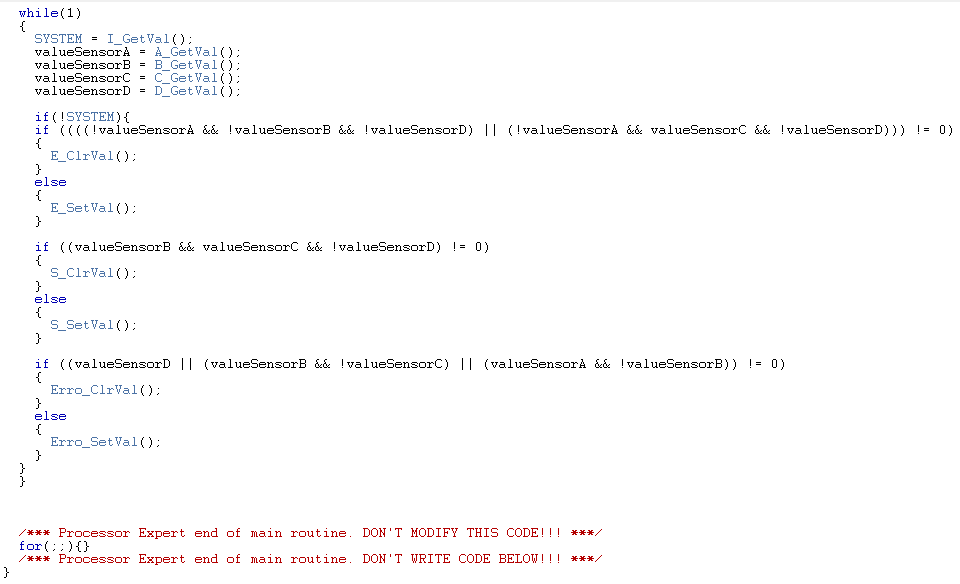
Na Figura 20 a seguir é possível ver o código em linguagem c, linguagem de alto nível dentro do codewarrior.

Parte 1:

Figura 20- arquivo da questão fatorial

Fonte: Autoria própria.

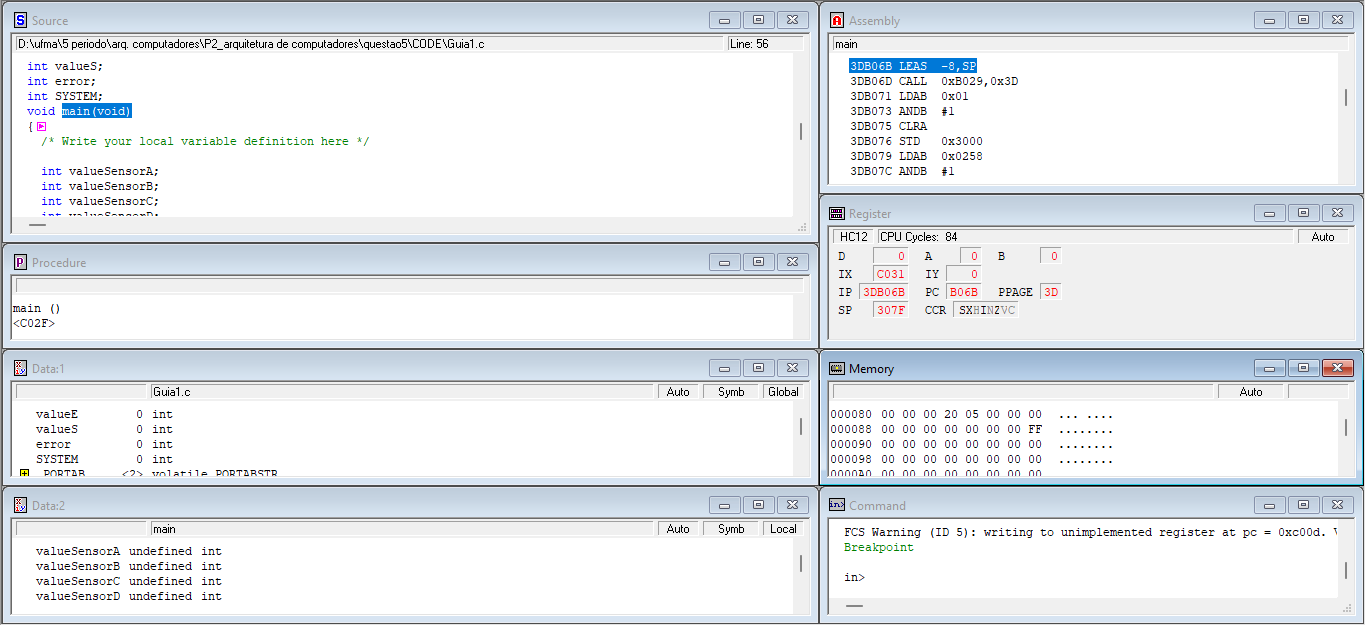
Parte 2:

Figura 21- arquivo da questão fatorial

Fonte: Autoria própria.

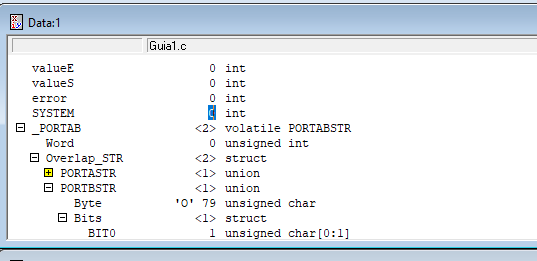
Depois o código foi executado no microcontrolador e os resultados foram apurados.

Na figura 22 temos a compilação do programa da quinta questão.

Figura 22 - arquivo da questão do tanque

Fonte: Autoria própria.

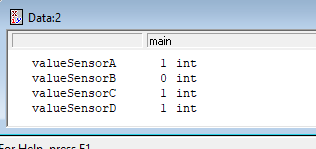
Para esse programa temos o interruptor colocado no programa como “system”, o qual faz o acionamento do sistema. então para os testes, poderemos modificar as seguintes variáveis: I(liga o sistema), A(sensor de nível), B(sensor de nível), C(sensor de nível) e D(sensor de nível). Na figura 23 é possível observarmos o acionamento do sistema que é ligado ao ser colocado o valor 0 dentro da variável.

Figura 23 - modificação das entradas de sinal 

Fonte: Autoria própria.

Na figura 24 podemos ver a entrada de sinal dos sensores que tambem são ativos ao ser colocado o valor 0 em cada uma delas.

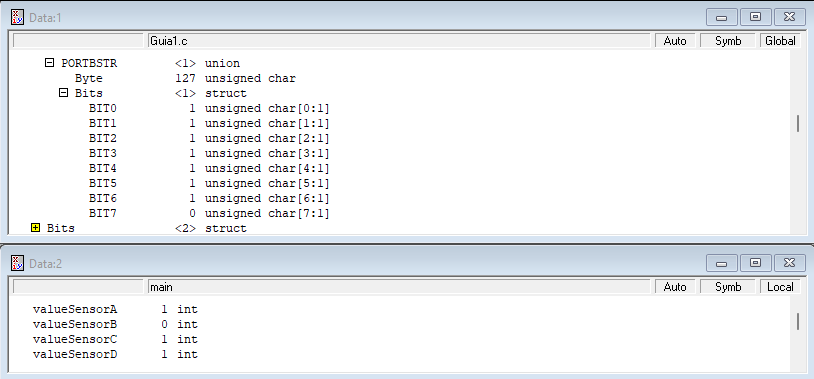
Figura 24 - entradas dos sensores



Fonte: Autoria própria.

Podemos ver a seguir a saída de acordo com a entrada de sinal dos sensores de acordo com o exemplo feito. Na figura 25 podemos ver os bits de saída da Porta B.

Figura 25 - saída na Porta B



Fonte: Autoria própria.

# Conclusão

Ao longo deste trabalho, foram exploradas as funcionalidades e características do microcontrolador HCS12, juntamente com a apresentação de soluções para os problemas encontrados durante o desenvolvimento de projetos utilizando o ambiente de desenvolvimento Codewarrior. Os experimentos e questões foram solucionados com sucesso, resultando nos resultados esperados obtidos nas simulações dentro do Codewarrior.

Durante o trabalho, foram abordados diferentes aspectos do HCS12, desde sua arquitetura e recursos até a implementação de soluções práticas para problemas no Codewarrior. Foram exploradas técnicas e estratégias para lidar com erros de compilação, depuração de código, otimização de desempenho e integração de bibliotecas externas.

Através dos experimentos foi possível compreender de forma clara e prática os conhecimentos adquiridos de forma teórica na disciplina de arquitetura de computadores. Todos os passos seguidos e as dificuldades para concluir os projetos serviram de grande aprendizado de como é o dia a dia do engenheiro eletricista e como deve-se agir diante de tais situações. Com os experimentos trazidos pelo professor ficou evidente as diversas aplicações de microcontroladores no cotidiano de diversas empresas e como é possível aproveitar situações vivenciadas na indústria para implementar tecnologias que auxiliem um usuário e dê a ele uma maior praticidade para realizar suas tarefas.

Em resumo, os objetivos propostos foram alcançados com sucesso, demonstrando a eficácia do microcontrolador HCS12 e do ambiente Codewarrior na criação e desenvolvimento de sistemas embarcados. Os resultados obtidos nas simulações confirmaram as expectativas e destacaram a importância dessas tecnologias no avanço da tecnologia em diversas áreas industriais e médicas.

# Referências Bibliográficas

[1] Draw.io: Software para construção de diagrama de blocos. Disponível em: https://draw.io. Acesso em: 02 de julho de 2023.

[2] NXP. CodeWarrior Development Studio para Microcontroladores HCS12 (Classic IDE) versão 5.2, 2016.

[3] Logisim: Software de simulação de circuitos digitais. Versão estável: 2.7.1, 2011; Disponível em:<http://www.cburch.com/logisim/pt/index.html> . Acesso em: 21 jan 2023.

[4] HCS12 Microcontroller and Embedded Systems: Using Assembly and C for Freescale HCS12. Muhammad Ali Mazidi, Janice Gillispie Mazidi, Rolin D. McKinlay.

[5] NXP Community. Disponível em: https://community.nxp.com/. Acesso em 01 de julho de 2023.

[6] Valvano, J.2012. Embedded Systems: Introduction to ARM Cortex-M Microcontrollers. Local de publicação: Editora.

[7] Microchip Technology. Disponível em: <www.microchip.com>. Acesso em: 02/07/2023.

[8] Embedded.com. Disponível em: <www.embedded.com>. Acesso em: 02/07/2023.