CÁSSIO DE CARVALHO BERNI

IMPLEMENTAÇÃO EM HARDWARE / FIRMWARE DE UM SENSOR VIRTUAL UTILIZANDO ALGORITMO DE IDENTIFICAÇÃO NEBULOSA

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia

São Paulo

2004

CÁSSIO DE CARVALHO BERNI

IMPLEMENTAÇÃO EM HARDWARE / FIRMWARE DE UM SENSOR VIRTUAL UTILIZANDO ALGORITMO DE IDENTIFICAÇÃO NEBULOSA

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia

Área de Concentração:
Engenharia de Sistemas

Orientador:
Prof. Dr. Claudio Garcia

São Paulo
2004

AGRADECIMENTOS

A Deus, por todas as minhas conquistas, por minha vida plena, por me manter íntegro e em busca da compreensão e aplicação de suas leis junto ao meu próximo.

A minha família, pelo inestimável amor, apoio e compreensão, em todos os momentos.

Ao Prof. Claudio Garcia, muito mais que um grande orientador, um grande amigo.

Ao mestre e amigo Vanderlei C. Parro, por ser para mim uma referência de ser humano, e por ter me encorajado a enfrentar mais esse desafio em minha vida.

À diretoria da Mosaico Engenharia Eletrônica, pela amizade e flexibilidade que me permitiram realizar esta jornada.

A todos os meus amigos em geral, que direta ou indiretamente colaboraram para que esse projeto se tornasse uma realidade.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

RESUMO

Este trabalho apresenta um projeto e implementação de um sensor virtual (soft sensor) "embedded", ou seja, um módulo de hardware genérico e autônomo, o qual pode ser utilizado em diversos processos complexos, nos quais uma determinada variável de interesse não possa ser diretamente medida por sensores convencionais.

O hardware é baseado em um processador digital de sinais da família TMS320, com a adição de circuitos de instrumentação, como interface de entrada e saída em padrões industriais (0..10V, 4..20mA), conversores A/D e D/A, entre outros.

A implementação do sensor virtual em hardware foi constituída basicamente de três etapas. A primeira consistiu no treinamento do modelo, realizado off-line, utilizando um microcomputador PC e o software Matlab. Neste trabalho, o modelo nebuloso obtido possui uma estrutura do tipo Takagi-Sugeno e os antecedentes são definidos com base no algoritmo de clusterização FCM.

Em uma segunda etapa, o modelo de sensor obtido foi validado no PC, comparando-se sua saída com a do sensor "real". Finalmente, a terceira etapa consistiu em traduzir e implementar o modelo no formato do processador digital de sinais. O firmware contém então o modelo obtido e opera on-line, estimando a variável desejada a partir de outra(s) variável(eis) disponível(eis).

Os testes foram realizados empregando-se uma planta simulada de neutralização de pH.

PALAVRAS-CHAVE: sensor, nebuloso, hardware, autônomo

"ABSTRACT"

This work presents a design and implementation of an embedded soft sensor, i. e., a generic and autonomous hardware module, which can be used in many complex plants, wherein a certain variable can not be measured directly by convencional sensors.

The hardware is based on a digital signal processor of TMS320 family, with the addition of instrumentation circuits, like industrial standard i/o interface (0..10V, 4..20mA), A/D and D/A converters, among other ones.

The implementation of the embedded soft sensor basically consisted of three steps. The first one was the model training, done off-line, using a PC microcomputer and Matlab software. In this work, the obtained fuzzy model has a Takagi-Sugeno structure and the antecedents are defined based on the FCM clustering algorithm.

In a second step, the obtained sensor model was validated in PC environment, comparing its output with the "real" sensor output. Finally, the third step consisted of translating and implementing the model to the digital signal processor format. The firmware so contains the obtained model and runs on-line, estimating the target variable from other available ones.

The tests have been performed using a simulated pH neutralization plant.

KEYWORDS: sensor, fuzzy, hardware, autonomous

1 INTRODUÇÃO

1.1 Objetivos

Na indústria em geral, particularmente na área química, existem determinados processos que possuem grandezas importantes não mensuráveis através de sensores convencionais. Um exemplo na área de tratamento de efluentes é a variável *DBO* (Demanda Bioquímica de Oxigênio), que é obtida apenas através de experimentos laboratoriais.

Um sensor virtual (conhecido também no meio acadêmico e industrial como *soft sensor*) é um modelo computacional que estima uma variável de interesse a partir de outra(s) variável(eis) medida(s) na planta.

O objetivo principal deste trabalho foi o de desenvolver um sensor virtual em hardware autônomo e genérico, ou seja, a partir de um modelo do sensor obtido em software, implementar este modelo em um sistema físico. Uma vez que os estudos de sensores virtuais estão atingindo um nível que permite sua aplicação na indústria, desejou-se com este trabalho viabilizar e concretizar essa aplicação.

Inicialmente, o sensor virtual de um processo simulado de neutralização de pH foi implementado em hardware / firmware e validado, comparando-se a saída de pH "real" com aquela fornecida pelo sensor virtual. Em uma segunda etapa, o sensor virtual obtido foi inserido junto à planta em malha fechada, com o intuito de fornecer ao controlador uma realimentação estimada, ou seja, ausência do sensor real. Nesta etapa, o controlador inferencial PID empregado foi implementado em ambiente Matlab / Simulink.

Finalmente, em uma terceira etapa, o próprio controlador PID foi embutido no hardware do sensor virtual, constituindo assim um sistema completo de controle da planta em questão.

Um objetivo adicional deste trabalho foi promover uma utilização industrial de sensores virtuais de maneira descentralizada, caracterizando o módulo autônomo de hardware como um instrumento "inteligente", aplicável em sistemas de controle distribuído.

1.2 Justificativas

A idéia principal em torno da utilização de um sensor virtual é a existência de situações industriais nas quais haja algum impedimento ao uso de sensores convencionais. Esse impedimento pode ser causado, por exemplo, por fatores como inexistência do sensor propriamente dito para uma variável específica, alto custo, imprecisão proibitiva à utilização desejada, tempo de resposta muito alto para aplicações em tempo real (controle) e impossibilidade de acesso ao local necessário para a instalação do sensor.

O leitor pode notar que duas questões importantes surgem a partir da idéia apresentada acima. São elas:

Q1. Como realizar a coleta de dados para levantar o modelo do processo / sensor virtual se não há o sensor real medindo a saída desejada?

Resposta possível: O modelo do processo / sensor virtual pode ser obtido por dados de experimentos laboratoriais.

Q2. Se já é conhecido o modelo do processo, por que desenvolver um sensor virtual para uma determinada variável de saída? O modelo do processo já não inclui o sensor virtual?

Resposta possível: Sim, porém o modelo do processo pode ficar muito complexo e lento para fornecer a resposta desejada, portanto é impraticável a sua utilização junto ao controlador. O sensor virtual a ser desenvolvido deve ser preciso e rápido, para que se justifique sua implantação como parte do sistema de controle real.

Uma das áreas mais críticas e com maiores possibilidades de uso de sensores virtuais é a de processos químicos complexos, como polimerização, colunas de destilação e bioprocessos. Há variáveis cujo sensor tradicional apresenta problemas de alto preço e baixa precisão (sensor de

concentração de substâncias específicas) ou problemas de atraso de medição, por exemplo, em análises laboratoriais de *DBO*, biomassa e Matérias Inibidoras (*MI*).

Sensores virtuais também têm sido usados para aumentar a confiabilidade e disponibilidade dos sensores tradicionais. Enquanto o sensor real está presente, o sensor virtual utiliza suas medidas para atualizar os parâmetros de seu modelo. Em caso de falha do sensor real, o sensor virtual assume o papel daquele, como contingência.

1.3 Metodologia

A primeira etapa para o desenvovimento do sensor virtual em hardware / firmware foi o estudo da teoria de identificação nebulosa baseada no modelo de Takagi-Sugeno (HELLENDOORN; DRIANKOV, 1997). A seguir, foi realizada uma minuciosa análise do artigo de Wang e Langari (1996), cujo algoritmo é a base do modelo de sensor virtual a ser adotado. Uma grande contribuição foi dada por Oliveira e Garcia (2003), o qual otimizou o modelo de Wang e Langari (1996), tornando o sensor virtual mais rápido e computacionalmente mais simples. O algoritmo desenvolvido por Oliveira e Garcia (2003) denomina-se "Regras Limitadas", e foi o algoritmo implementado no firmware.

Foram então coletados os dados para treinamento do modelo do sensor virtual. Isso foi feito considerando-se como processo o "benchmark" em Matlab / Simulink de uma planta simulada de neutralização de pH. A variável de interesse é justamente o pH e a variável de entrada é a vazão de base q_3 (vide capítulo 2). Com os dados do processo, foi feita a identificação do modelo do sensor virtual. Essa identificação foi realizada off-line, em ambiente Matlab / Simulink.

Uma vez obtidos os parâmetros do modelo do sensor, foi feita uma análise estrutural de software, necessária para uma adaptação do ambiente Matlab para o processador digital de sinais (DSP - Digital Signal Processor) TMS320F206 da Texas Instruments. Em paralelo, foi realizado o projeto do hardware completo, com a interface de condicionamento de sinais e conversores analógico-digital (A/D) e digital-analógico (D/A) interligados ao DSP. A idéia foi projetar um sistema "embedded", genérico, o qual pode ser utilizado em diversos tipos de processos. Para que

este módulo possa ser integrado a um sistema mais amplo de controle e supervisão, ele contém também uma interface de conexão para uma rede RS-485 (futuramente uma interface CAN - Controller Area Network - pode ser desenvolvida), rede essa que pode ter um microcomputador PC como elemento supervisório. Antes de programar o sistema com o software do modelo de sensor treinado, foram feitos testes para validação do hardware. Então, foi implementado o modelo do sensor virtual no hardware desenvolvido.

Para os testes de validação do sensor virtual foi utilizada uma placa de aquisição de dados da Lynx, modelo CAD12/36 (LYNX TECNOLOGIA ELETRÔNICA LTDA, 1993), com conversores A/D e D/A. A idéia foi exteriorizar os sinais de vazão de base e pH do "benchmark" em Matlab para o mundo real, emulando a planta física. A validação do sensor virtual foi feita em duas etapas. Na primeira, apenas o sensor "embedded" foi testado, isoladamente, em malha aberta, comparando-se a sua saída com a do sensor "real". Em uma segunda etapa, o sensor em hardware foi inserido junto ao processo, porém o fechamento da malha de controle do pH foi realizado através do sensor "real". A idéia foi variar o valor de referência do pH de diversas formas, como degrau, senóide, rampa, etc, e comparar as curvas de resposta do sensor em hardware com as do sensor "real".

Como ilustração adicional, foram realizados testes do sistema em malha fechada com realimentação pelo sensor "embedded", para os casos do controlador PID em ambiente Matlab e embutido no DSP (conjunto sensor virtual + PID).

1.4 Revisão bibliográfica

Pesquisas maciças envolvendo algoritmos de sensores virtuais no meio acadêmico são relativamente recentes (1998 em diante), sendo que uma característica comum a todas elas é a implementação de modelos computacionais em ambiente de microcomputador PC (ASPENTECH, 1998; YOKOGAWA, 2001). Os resultados são avaliados por simulações puramente em computador ("benchmarks" de plantas) ou através de interfaces de hardware junto ao processo real.

Atualmente, existem algumas empresas estrangeiras que possuem algumas soluções em sensores virtuais. É o caso da Pavilion, Emerson (Fisher), AspenTech, Matrikon, Honeywell e Yokogawa, entre outras (HAROLD, 2001). De forma geral, essas soluções são implementadas em software para ambiente Windows, ou seja, em microcomputador PC (ASPENTECH, 1998; YOKOGAWA, 2001).

Em geral, o sistema é constituído de dois módulos, o off-line e o on-line. O módulo off-line é o responsável pela geração do modelo de sensor virtual, a partir dos dados coletados da planta. Já o módulo on-line importa o modelo gerado para a utilização junto ao controlador. Nesse caso, as soluções encontradas utilizam como interface junto ao processo placas de aquisição de dados proprietárias. O padrão mais utilizado para integrar os sensores virtuais em PC à rede de controle é o OPC - **O**LE (Object Linking and Embedding) for **P**rocess **C**ontrol.

O diferencial apontado pelos fabricantes consiste em substituir as análises off-line em laboratório por estimadores "real time" dos índices de qualidade do processo (YOKOGAWA, 2001). Medidas de vazão, temperatura e pressão, entre outras, são utilizadas na estimação de variáveis complexas, como concentração de impurezas no butano, etc.

Em resumo, as soluções acadêmicas e industriais atuais em sensores virtuais concentram-se na utilização de microcomputadores, inclusive utilizando estações separadas para a geração do modelo e para a execução do mesmo. Não estão sendo utilizados módulos "embedded". É bem verdade que a flexibilidade no PC é grande, permitindo alteração imediata de parâmetros, análise via gráficos, etc. Em contrapartida, há a questão do alto custo de uma estação PC com hardware dedicado, além do item segurança, pois torna-se mais fácil a adulteração de software em um micro do que em um sistema "embedded". Salienta-se ainda que o módulo autônomo em questão não está isolado, e sim em rede, e pode ter seu firmware atualizado sem a necessidade de sua remoção da planta.

1.5 Estrutura do texto

O capítulo 2 apresenta uma breve descrição do processo de neutralização de pH, para

fornecer ao leitor subsídios mínimos que permitam a compreensão plena da sequência do trabalho.

Devido à utilização de técnicas de identificação nebulosa, o propósito do capítulo 3 é prover uma sustentação na teoria de lógica nebulosa, ainda que não seja um pré-requisito obrigatório. Considera-se útil para o entendimento da idéia central do algoritmo utilizado que se tenha um conhecimento básico de lógica nebulosa, conjuntos nebulosos, regras de inferência, entre outros itens. Ainda no capítulo 3, são apresentadas as técnicas de identificação nebulosa estudadas, o algoritmo final de treinamento e execução do modelo de sensor virtual e a estrutura de modelo utilizada.

O capítulo 4 apresenta um estudo da estrutura computacional do modelo obtido, estudo esse necessário para a escolha do hardware e ferramentas de software a serem utilizadas na "tradução" do modelo em PC para o modelo implantado em hardware / firmware. Alguns critérios para escolha da plataforma de hardware são analisados, como custo, tempos de processamento, etc.

Uma vez definida a estrutura de hardware e software, o capítulo 5 descreve em detalhes o projeto do sistema "embedded", batizado de VS01A. Primeiramente é apresentado o desenvolvimento de hardware, com a descrição de cada subsistema: fonte de alimentação, interface de entrada e saída, chip-select, conversão A/D, conversão D/A e o DSP. Com relação ao software, são descritos os programas de teste do hardware, o modelo com o algoritmo "Regras Limitadas" e o módulo de controlador PID, todos escritos em assembly.

O capítulo 6 apresenta os resultados comparativos entre o sensor "real" e o protótipo VS01A para diversas simulações, entre elas malha aberta, malha fechada com realimentação pelo sensor "real", malha fechada com realimentação pelo VS01A e PID integrado ao hardware. São feitos ainda comentários e observações pertinentes.

Por fim, são apresentadas no capítulo 7 algumas conclusões e sugestões para possíveis futuros trabalhos nesta área de pesquisa.

2 O PROCESSO DE NEUTRALIZAÇÃO DE pH

O pH - potencial hidrogeniônico - é uma grandeza que representa a concentração de íons hidrogênio H^+, fornecendo uma indicação sobre o grau de acidez, neutralidade ou alcalinidade de soluções aquosas. Sua faixa de valores está entre 0 (ácido) e 14 (base). Uma solução neutra possui o pH igual a 7.

A equação (2.1) apresenta a definição matemática do pH.

$$pH = -\log_{10}[H^+] \tag{2.1}$$

sendo $[H^+]$ a concentração de íons de hidrogênio, em mols/litro

Basicamente, o processo de neutralização de pH tem como elemento principal um tanque misturador, o qual geralmente é uma unidade CSTR: Continuous Stirred Tank Reactor. O processo é fortemente não linear, o que justifica em parte a utilização de lógica nebulosa para identificação e/ou controle, pois esta é uma das técnicas adequadas para sistemas não lineares (BABUSKA, 2001; HELLENDOORN; DRIANKOV, 1997).

Outra característica importante do processo é a sua invariância no tempo, sendo apenas dependente de perturbações externas. Dessa forma, foi possível a aplicação da técnica de identificação em batelada, ou seja, coleta de dados entrada-saída e identificação off-line (GARCIA, 2001).

A planta é sensível a perturbações próximo ao ponto de pH = 7 (neutralidade). Essa característica pode ser explicada da seguinte maneira: na condição de neutralidade, a adição de quantidades supostamente desprezíveis de ácido ou base conseguem tirar o sistema facilmente do ponto neutro, levando o pH a valores como 8, 9 (adição de base) ou 6, 5 (adição de ácido). Por outro lado, caso a solução já esteja com um pH = 4, por exemplo, a adição das mesmas quantidades de ácido ou base do caso anterior não causam mudanças perceptíveis no pH. São necessárias quantidades muito maiores de ácido ou base para que o sistema tenha o pH alterado significativamente. A curva de titulação apresentada na figura 2.1 ilustra o fato.

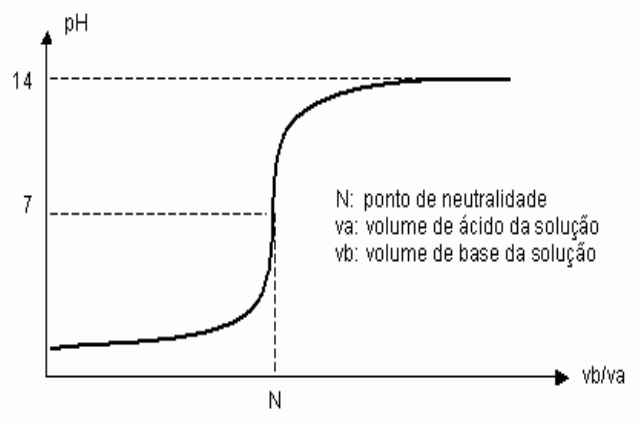


Figura 2.1 - Curva de titulação

Um diagrama esquemático simplificado do processo é mostrado na figura 2.2.

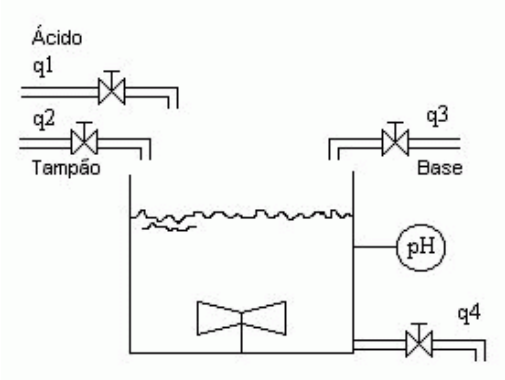


Figura 2.2 - Diagrama P & I do processo de neutralização de pH

sendo q_1 a vazão de ácido nítrico ($HN0_3$), q_2 a vazão de bicarbonato de sódio ($NaHC0_3$) - solução tampão, q_3 a vazão de hidróxido de sódio (NaOH) e q_4 a vazão de efluente - pH desejado = 7 (normalmente).

Os íons contidos nos fluxos reagentes são H^+ (íon hidrogênio), Na^+ (íon sódio), H_2CO_3 (ácido carbônico), CO_3^{2-} (íon carbonato), OH^- (íon hidróxido), NO_3^- (íon nitrato) e HCO_3^- (íon bicarbonato).

As principais reações químicas envolvidas são

$$H_20 \leftrightarrow OH^- + H^+$$

$$H_2C0_3 \leftrightarrow HCO_3^- + H^+$$

$$HC0_3^- \leftrightarrow CO_3^{2-} + H^+$$

A figura 2.3 permite uma visualização do tipo "caixa preta" do processo. Observa-se que a temperatura e as concentrações de ácido, tampão e base são consideradas perturbações ao sistema.

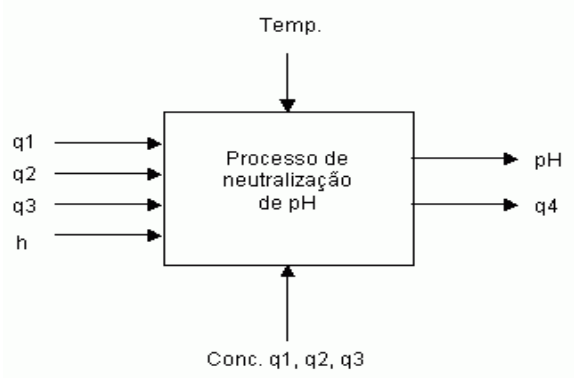


Figura 2.3 - Modelo "caixa preta" do processo de neutralização de pH

Não será detalhado aqui o processo de neutralização de pH, pois não é o foco deste trabalho. Para maiores detalhes sobre o processo em si, ver (COSTA, 2001).

Enfatiza-se que, atualmente, não seria necessário o desenvolvimento de um sensor virtual para a variável *pH*. O processo de neutralização de pH foi escolhido por sua simplicidade e pela existência do "benchmark" em ambiente Matlab / Simulink. O trabalho aqui proposto possui um caráter genérico, podendo ser aplicado em diversos processos industriais.

Para as simulações em geral, visando a obtenção do modelo do sensor virtual, será considerada variável de entrada a vazão de base q_3. As demais vazões de entrada serão mantidas constantes, e a vazão de saída q_4 será tal que o nível *h* do tanque seja mantido estável. A variável de saída será o *pH*.

O sensor virtual deverá ser capaz de estimar o *pH* atual baseado apenas na vazão de base q_3

atual, amostras anteriores de q_3 e amostras anteriores de pH estimado. Caracteriza-se assim a utilização da técnica de “simulação livre” ou de infinitos passos à frente, em que os regressores da saída são os valores previamente estimados e não valores medidos do processo (AGUIRRE, 2000).

3 LÓGICA NEBULOSA E ALGORITMOS DE IDENTIFICAÇÃO NEBULOSA

3.1 Lógica nebulosa - conceitos

A lógica nebulosa, também conhecida como lógica "fuzzy", surgiu por volta de 1965, com os estudos do professor Lotfi A. Zadeh, na Universidade de Berkeley, Califórnia. A característica fundamental dessa teoria é prover um método de tradução de expressões verbais, vagas, imprecisas e/ou qualitativas, comuns na comunicação humana, em valores numéricos. Assim, permite a um computador binário trabalhar de maneira mais próxima ao pensamento humano, o qual é intrinsicamente "nebuloso".

A lógica tradicional, booleana, apresenta apenas os valores “0” ou “1". Não há "meio termo". Para toda a faixa de operação da entrada, os dois únicos conjuntos aos quais a saída pode pertencer são "falso (0)" ou "verdadeiro (1)".

A palavra "fuzzy", do Inglês, significa nebuloso, não totalmente delineado. A variável de saída pode assumir qualquer valor entre “0” e “1”. Assim, é possível descrever a saída de uma forma mais qualitativa. Observa-se que a lógica booleana é um caso particular da lógica nebulosa.

A lógica nebulosa contempla aspectos imprecisos no raciocínio lógico utilizado pelos seres humanos. Sua aplicação é recomendada para o controle de sistemas os quais têm seu modelo desconhecido ou apenas qualitativo, ou sua obtenção analítica é inviável.

Um outro caso no qual a lógica nebulosa pode ser utilizada com eficiência é no controle e identificação de sistemas não lineares (SHAW; SIMÕES, 1999).

3.2 Conjuntos nebulosos

Para explicar o conceito de conjunto nebuloso utiliza-se o exemplo clássico da idade de uma pessoa, comparando-se a abordagem booleana, descrita pela figura 3.1, com a abordagem

nebulosa, descrita pela figura 3.2.

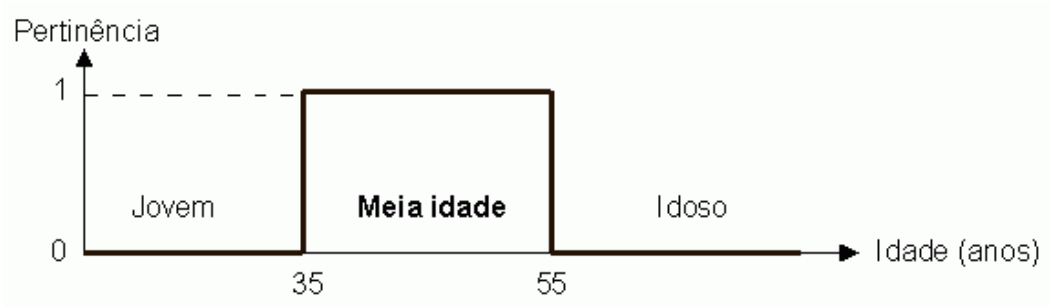


Figura 3.1 - Conjunto "meia idade" - abordagem booleana

Observa-se, na abordagem booleana, que as transições entre os conjuntos "jovem" e "meia idade", e entre os conjuntos "meia idade" e "idoso", são bruscas ou "crisp". Dessa forma, ao passar dos 34 para os 35 anos, uma pessoa que sequer pertencia ao conjunto "meia idade" passa a pertencer **totalmente** a ele. Essa abordagem é incompatível com a lógica de pensamento humano.

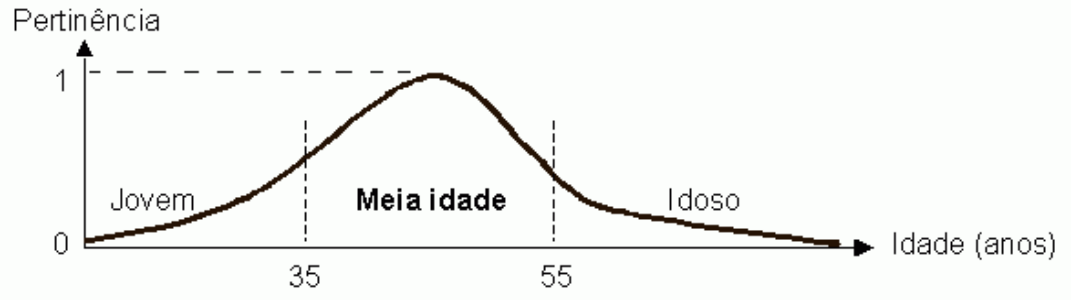


Figura 3.2 - Conjunto "meia idade" - abordagem nebulosa

Já na abordagem nebulosa, vê-se que as transições são suaves, sendo que um indivíduo com 35 anos pertence 50% ao conjunto "jovem" e 50% ao conjunto "meia idade".

É importante introduzir neste momento o conceito de grau de pertinência. O grau de pertinência define o “quanto” uma variável pertence a um determinado conjunto. Na lógica booleana existem somente dois graus de pertinência: 0% e 100%. Na lógica nebulosa pode ser usada toda a faixa entre 0% e 100%. A descrição formal de grau de pertinência, para o caso

nebuloso, é dada por

$$A = \{(x\ ,\ \mu_A(x) \quad |\quad x \in X)\} \tag{3.1}$$

sendo A o conjunto nebuloso, x a variável em questão, μ_A o grau de pertinência de x em A e X o universo de discurso (JANG; SUN; MIZUTANI, 1997; SHAW; SIMÕES, 1999).

Existem diversas operações elementares que podem ser realizadas com conjuntos nebulosos. As mais utilizadas são a **união** e **intersecção**, definidas respectivamente pelas equações (3.2) e (3.3).

$$C = A \cup B \Leftrightarrow \mu_C(x) = \max(\mu_A(x)\ ,\ \mu_B(x)) = \mu_A(x) \vee \mu_B(x) \tag{3.2}$$

$$C = A \cap B \Leftrightarrow \mu_C(x) = \min(\mu_A(x)\ ,\ \mu_B(x)) = \mu_A(x) \wedge \mu_B(x) \tag{3.3}$$

3.3 "Fuzzyficação" de variáveis

A "fuzzyficação" de uma variável é o processo de descrição da faixa de valores da mesma através de conjuntos nebulosos. Isso pode ser feito tanto para as variáveis de entrada como de saída, dependendo do tipo de modelo nebuloso, como será visto posteriormente.

Os ambientes de desenvolvimento de sistemas nebulosos, como Matlab, FuzzyTech, etc, trabalham com valores normalizados, por exemplo, entre −1 e +1, ou entre 0 e 1. Assim, um ponto importante na etapa de "fuzzyficação" é a normalização das variáveis.

Existem diversos tipos de funções de pertinência (SHAW; SIMÕES, 1999), porém os mais comuns são: triangular, trapezoidal e gaussiana. Como exemplo, tem-se na figura 3.3 a "fuzzyficação" de uma variável *erro* genérica.

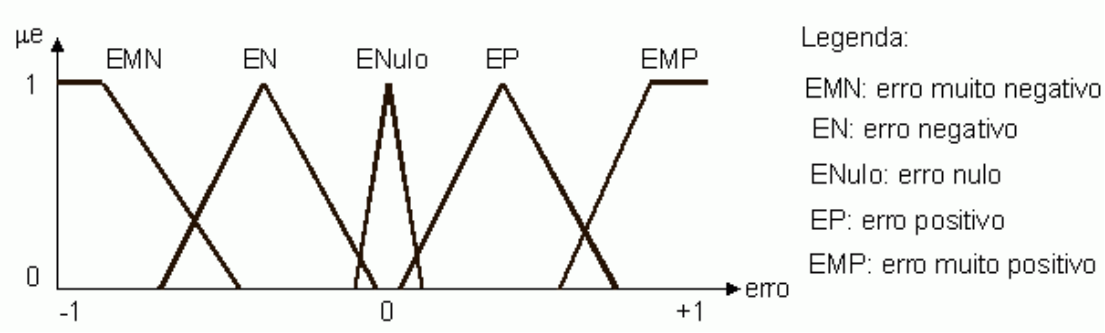


Figura 3.3 - "Fuzzyficação" de uma variável *erro* genérica

3.4 Regras de inferência

As regras de inferência são aquelas que definem o comportamento do sistema, ou seja, o mapeamento entrada(s)-saída(s). A partir de uma análise das entradas, são mapeados os conjuntos nebulosos ou as funções das saídas.

Como há uma proximidade muito grande entre a lógica nebulosa e o raciocínio lógico humano, o formato das regras é do tipo:

SE (antecedentes) *ENTÃO* (consequentes)

Um exemplo de regras de inferência, para o controle de um condicionador de ar, pode ser visto a seguir.

SE (temperatura = baixa) *E* (umidade = baixa) *ENTÃO* (potência do ventilador = muito baixa)
SE (temperatura = baixa) *E* (umidade = alta) *ENTÃO* (potência do ventilador = baixa)
SE (temperatura = média) *E* (umidade = baixa) *ENTÃO* (potência do ventilador = média)
SE (temperatura = média) *E* (umidade = alta) *ENTÃO* (potência do ventilador = média-alta)
SE (temperatura = alta) *E* (umidade = baixa) *ENTÃO* (potência do ventilador = alta)
SE (temperatura = alta) *E* (umidade = alta) *ENTÃO* (potência do ventilador = muito alta)

3.5 Modelo do tipo Mandani

O modelo nebuloso do tipo Mandani apresenta como característica básica o fato de tanto os antecedentes como os consequentes serem mapeados como conjuntos nebulosos. Exemplo: *Se* erro é "pequeno" *então* tensão é "baixa".

Para cada regra de inferência, caso tenhamos mais de uma variável de entrada, é necessário aplicar uma técnica de agregação dos conjuntos antecedentes, a fim de que seja gerado um conjunto consequente. A agregação dos conjuntos antecedentes geralmente é dada pelo operador intersecção (mínimo - ceifagem do conjunto consequente) ou pelo operador produto (“achatamento” do conjunto consequente) (JANG; SUN; MIZUTANI, 1997).

No caso de existirem *N* regras, serão gerados *N* conjuntos consequentes. A combinação desses conjuntos consequentes, a fim de que seja gerado o conjunto final de saída, é feita geralmente pelo operador união (máximo).

Como exemplo, ilustra-se na figura 3.4 uma tela do Toolbox Fuzzy do Matlab, para o caso de um controle de posição de um motor DC.

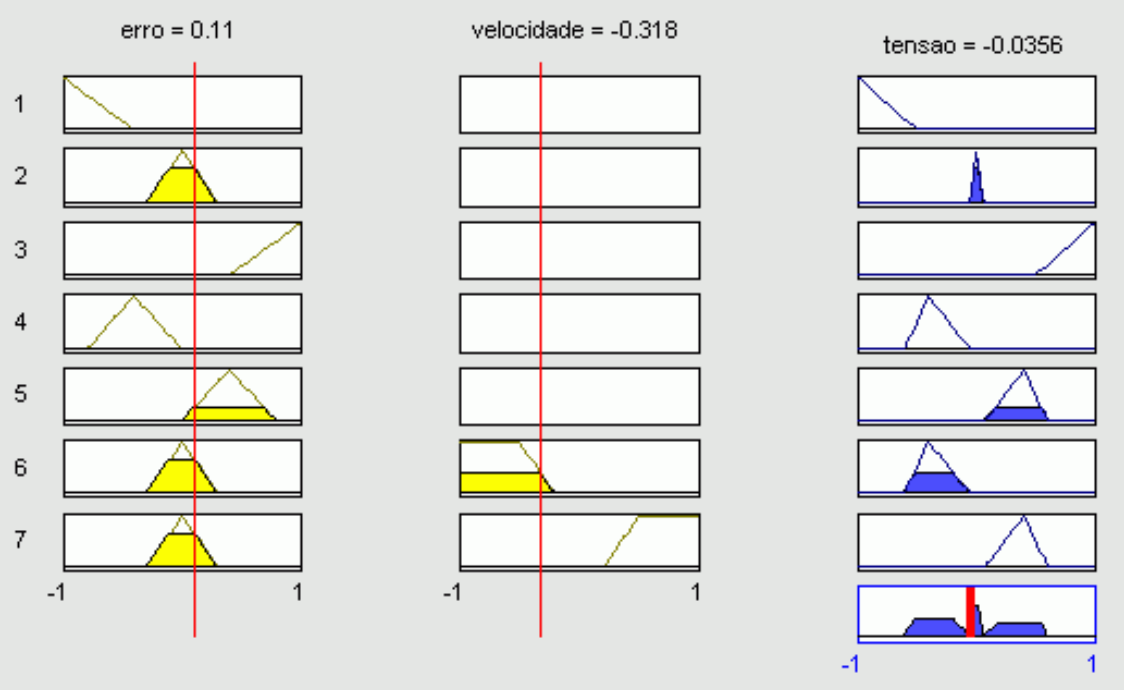


Figura 3.4 - Toolbox Fuzzy do Matlab - exemplo de inferência Mandani

Observa-se que as variáveis de entrada são o erro de posição e a velocidade do motor. A variável de saída é a tensão aplicada na armadura do motor. Todas as variáveis (entradas e saída) são descritas pelos seus respectivos conjuntos nebulosos, de acordo com a faixa de operação. Cada linha da figura 3.4 representa graficamente uma regra de inferência. Assim, verifica-se que existem 7 regras para esse exemplo, descritas a seguir.

1. *SE* (erro = muito negativo) *ENTÃO* (tensão = muito negativa)
2. *SE* (erro = nulo) *ENTÃO* (tensão = nula)
3. *SE* (erro = muito positivo) *ENTÃO* (tensão = muito positiva)
4. *SE* (erro = negativo) *ENTÃO* (tensão = meio negativa)
5. *SE* (erro = positivo) *ENTÃO* (tensão = meio positiva)
6. *SE* (erro = nulo) *E* (velocidade = negativa) *ENTÃO* (tensão = meio negativa)
7. *SE* (erro = nulo) *E* (velocidade = positiva) *ENTÃO* (tensão = meio positiva)

Analisando-se a regra 6 observa-se que o grau de pertinência da entrada atual *velocidade* ao conjunto "velocidade negativa" é menor que o grau de pertinência da entrada atual *erro* ao conjunto "erro nulo". Como a agregação dos antecedentes está sendo feita pelo operador mínimo, o grau de pertinência da saída atual *tensão* fica igual ao grau de pertinência da entrada atual *velocidade*, o qual é o menor valor.

A composição dos conjuntos consequentes (em azul) está sendo feita pelo operador máximo, gerando o conjunto consequente final na oitava linha da figura 3.4. A barra vertical vermelha no conjunto consequente final representa o valor numérico da tensão atual, resultante da "defuzzyficação" pelo cálculo do centróide (ver item 3.7).

3.6 Modelo do tipo Takagi-Sugeno

O modelo nebuloso do tipo Takagi-Sugeno apresenta como diferencial em relação ao modelo Mandani o fato dos consequentes não serem conjuntos, mas sim funções lineares das entradas. Um exemplo para o caso de uma variável de entrada é mostrado na figura 3.5.

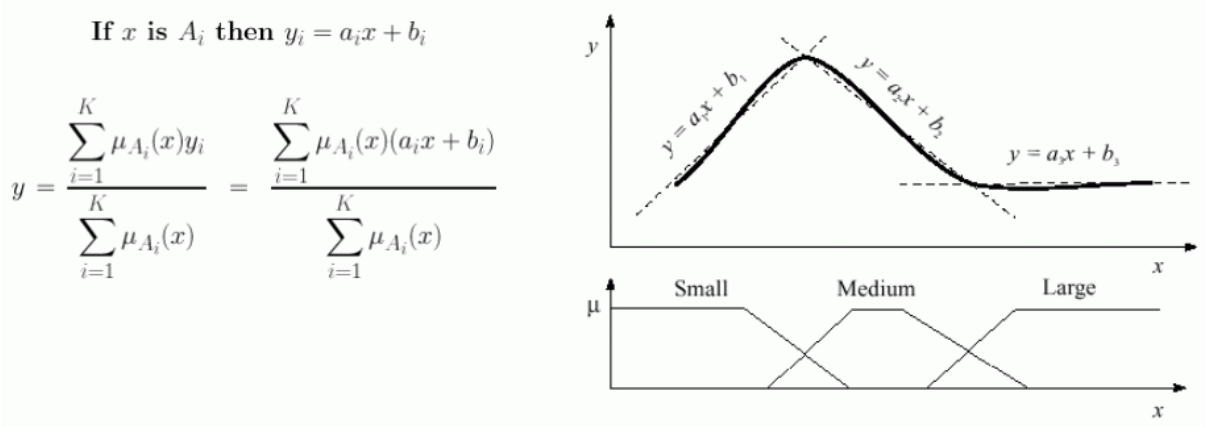


Figura 3.5 - Exemplo de inferência Takagi-Sugeno

Observa-se que, computacionalmente, todas as funções lineares das saídas parciais y_i são sempre calculadas. Evidentemente, essas saídas parciais terão seu "peso" definido pelo grau de pertinência da variável x a cada conjunto nebuloso definido por cada regra.

No caso de existir mais de uma variável de entrada, a escolha dos coeficientes de ponderação das saídas parciais deve ser feita com base em um critério de agregação nebulosa qualquer, como por exemplo, uma t-norma mínimo ou produto (SHAW; SIMÕES, 1999).

Quando as funções dos consequentes são polinômios de primeira ordem, o sistema de inferência nebulosa é chamado de Modelo Sugeno de Primeira Ordem.

Em relação ao modelo Mandani, o modelo Takagi-Sugeno apresenta um caráter menos linguístico, porém é mais simples para uso em identificação de sistemas, pois cada regra define um espaço diferente no qual a saída depende linearmente das entradas. Os parâmetros dos consequentes podem ser estimados pelo método dos mínimos quadrados (GARCIA, 2001; LJUNG, 1999).

Quando aplicado em identificação de sistemas, é comum o sistema de inferência Takagi-Sugeno levar em conta uma autoregressão da própria saída:

$$R_1 : se \quad y(k) \quad é \quad A_1 \quad E \quad u(k) \quad é \quad B_1 \quad então \quad y(k+1) = a_1 y(k) + b_1 u(k) + c_1$$
$$R_2 : se \quad y(k) \quad é \quad A_2 \quad E \quad u(k) \quad é \quad B_2 \quad então \quad y(k+1) = a_2 y(k) + b_2 u(k) + c_2$$

3.7 "Defuzzyficação"

Uma vez combinados os consequentes de cada regra, um único valor de saída deve ser gerado. O processo de geração desse valor é chamado "defuzzyficação".

O método mais comum de "defuzzyficação", para o modelo Mandani, é o cálculo do centro de área do conjunto consequente final. O método é normalmente chamado de cálculo do centróide. No caso do modelo Takagi-Sugeno, o processo de ponderação das saídas parciais pode ser comparado à "defuzzyficação" pelo centróide (SHAW; SIMÕES, 1999).

Após a "defuzzyficação" deve-se proceder a desnormalização das variáveis para as suas faixas de valores originais.

3.8 Identificação nebulosa

Nos métodos clássicos de identificação paramétrica de sistemas, é comum a aplicação do método dos mínimos quadrados (GARCIA, 2001). A idéia é minimizar uma função perda do tipo

$$V_N(Z_N\ ,\ \theta) = \frac{1}{N}\sum_{t=1}^{N}\frac{1}{2}(y(t) - \hat{y}(t/\theta))^2 = \frac{1}{N}(Y - \Phi\theta)^T(Y - \Phi\theta) \qquad (3.4)$$

sendo

$$Z_N = [\ (x(1), y(1))\ ,\ (x(2), y(2))\ , \ldots (x(N), y(N))\ ]$$

o conjunto de dados entrada-saída, N o número de pontos, $y(t)$ a saída real do sistema, $\hat{y}(t/\theta)$ a saída estimada e

$$\hat{\theta}(Z_N) = (\Phi^T \Phi)^{-1} \Phi^T Y \tag{3.5}$$

o estimador de mínimos quadrados.

Geralmente, para sistemas lineares, o modelo obtido apresenta coeficientes de regressão linear, como

$$\hat{y}(t/\theta) = b_0 + b_1 u_1 + b_2 u_2 + \ldots + b_p u_p + a_1 y_1 + a_2 y_2 + \ldots + a_n y_n \tag{3.6}$$

sendo

$$Z = \begin{bmatrix} u_1 \\ \vdots \\ u_p \end{bmatrix}_{p \times 1} = \text{vetor de entradas}$$

Porém, para sistemas não lineares, fica muito crítica a tentativa de se modelar toda a faixa de operação do sistema com apenas um equação de regressão linear, como mostra a figura 3.6.

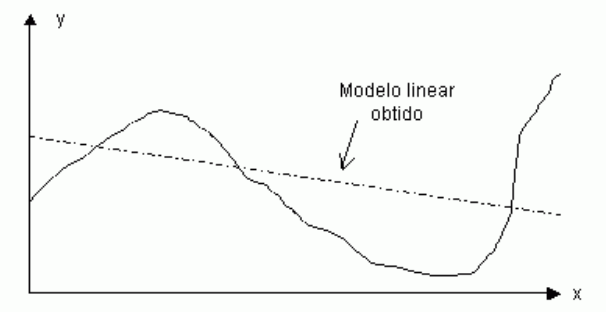


Figura 3.6 - Modelo linear único para toda a faixa de operação

Dessa forma, duas alternativas são usualmente utilizadas. Uma delas consiste na aplicação de técnicas não lineares de identificação, como por exemplo NARMAX (LJUNG, 1999), e a outra consiste na divisão da faixa de operação do sistema em trechos que possam ser modelados linearmente (HELLENDOORN; DRIANKOV, 1997).

A segunda técnica mostra-se muito interessante pela sua simplicidade, e é ilustrada na figura 3.7.

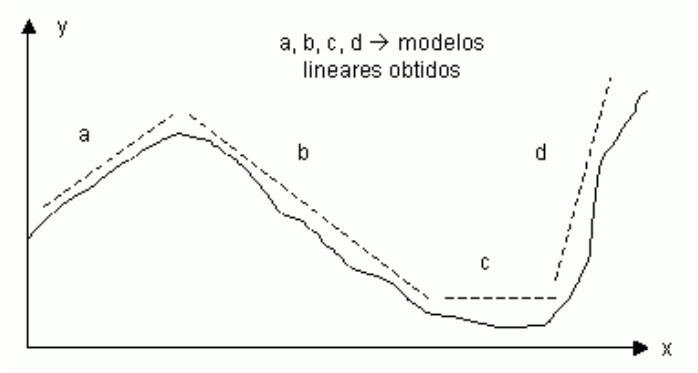


Figura 3.7 - Submodelos lineares parciais

Entretanto, percebe-se que há um problema sério nos pontos de transição entre os submodelos. Numa abordagem tradicional, as transições seriam bruscas ("booleanas", ou "crisp"), fazendo com que o modelo global divirga muito do sistema original nos pontos de transição (descontinuidades).

Takagi e Sugeno perceberam que seria prudente aplicar um pensamento nebuloso para as regiões de transição entre os submodelos. Assim, ocorreriam suavizações que tornariam o modelo global mais fiel ao sistema real (JANG; SUN; MIZUTANI, 1997). Já foi visto que isso é conseguido pelo sistema de inferência Takagi-Sugeno, com a ponderação das saídas parciais, cujo resultado está ilustrado na figura 3.8.

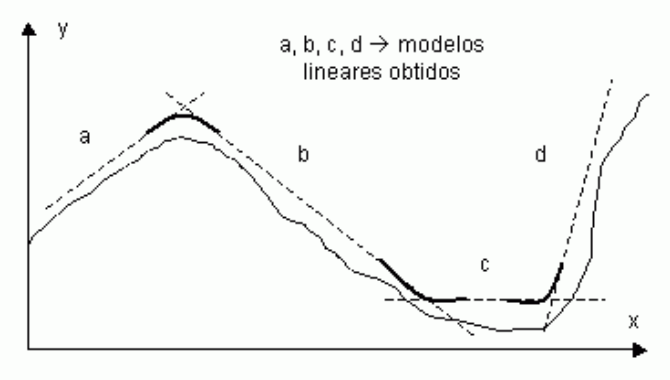


Figura 3.8 - Suavização dos pontos de transição entre os submodelos

Em suma, o procedimento de identificação nebulosa Takagi-Sugeno se inicia com o projeto do experimento e coleta de um conjunto de dados entrada-saída que seja significativamente representativo para toda a faixa de operação do sistema. Devem ser levantadas as variáveis relevantes de entrada. O sinal de excitação deve, se possível, percorrer todo o espaço entrada-saída, e ser rico em frequências diversas. Como exemplo, podemos utilizar rampas ascendentes e descendentes, cuja amplitude varra toda a faixa do sistema, e cuja frequência seja modulada por um segundo sinal, dente de serra ou triangular, para varredura de espectro. É possível (e recomendável) ainda que a rampa principal tenha uma adição de ruído branco gaussiano, tornando a excitação mais próxima da realidade. Como exemplo, o sinal de excitação utilizado no presente trabalho consistiu em uma sequência de duas rampas, a primeira ascendente e a segunda descendente, com amplitude em unidades de pH de 3 a 10. Esse sinal recebe a adição de uma perturbação na forma de ruído branco gaussiano, com potência relativa 0,001 (Matlab) e intervalo de amostragem de 0,167 seg, o qual é o mesmo do processo em si.

Em seguida, deve ser feita a escolha da estrutura do modelo. A idéia é identificar, entre outros parâmetros, a dinâmica do sistema, escolhendo a ordem através do número de regressores para cada entrada e para a saída.

O próximo passo é o particionamento do espaço entrada-saída. Dentre várias técnicas existentes, Wang e Langari (1996) e Oliveira e Garcia (2003) utilizaram a clusterização (ver item 3.9). Existem métodos para a determinação do número ótimo de clusters, como o "Validity Measures" (HELLENDOORN; DRIANKOV, 1997), ainda que isso possa ser feito empiricamente.

Na sequência, é feita a geração das regras. Para os antecedentes, tem-se o vetor com o centro dos clusters e, eventualmente, a matriz de partição nebulosa. Os parâmetros de cada consequente (submodelos de regressão linear) podem ser estimados pelo método dos mínimos quadrados. Por fim, o modelo obtido deve ser validado através de algum critério de desempenho.

3.9 Fuzzy Clustering

A idéia básica do processo de clusterização é identificar, em um conjunto de pontos coletados, agrupamentos cujos dados ilustrem regiões com contorno razoavelmente delineado. Cada agrupamento é chamado de cluster. Faz parte do processo de clusterização calcular o vetor *V* de centros de cada cluster, bem como a matriz de partição nebulosa *U*, conforme ilustra a figura 3.9.

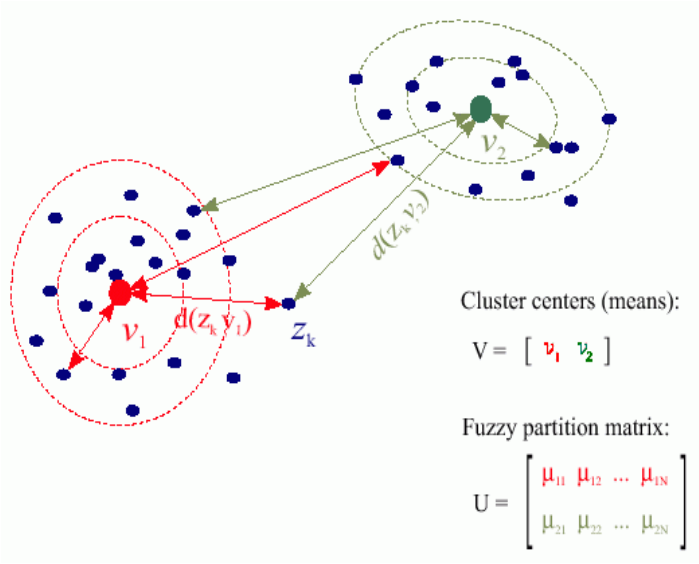


Figura 3.9 - Idéia geral do processo de clusterização

Observa-se que a matriz de partição nebulosa *U* apresenta os graus de pertinência de todos os pontos a todos os clusters. Para um novo ponto coletado Z_k são calculadas as distâncias desse ponto a cada cluster, definindo assim uma determinada coluna de *U*.

O objetivo principal da clusterização é identificar os conjuntos nebulosos dos antecedentes e consequentes (Mandani), para os casos onde não haja o conhecimento prévio do processo, o que permitiria construir esses conjuntos. Um exemplo de clusterização aplicada a um modelo do tipo Mandani está ilustrado na figura 3.10.

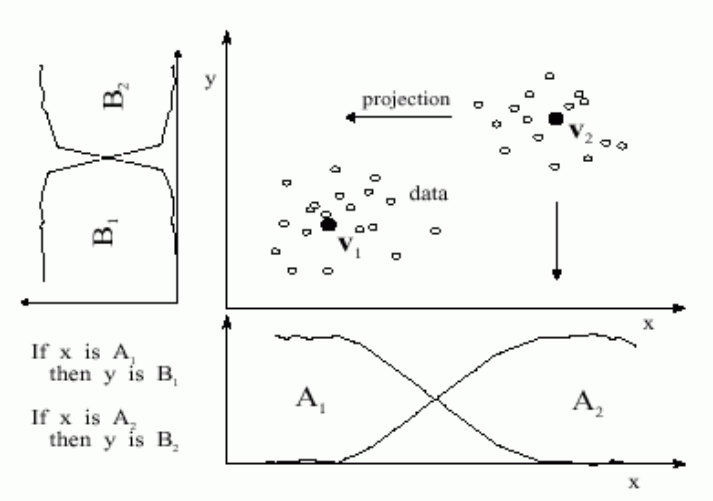


Figura 3.10 - Clusterização - Inferência Mandani

Observa-se que para um novo ponto x são calculadas as distâncias desse ponto aos centros v_1 e v_2 dos clusters. Isso equivale ao cálculo dos graus de pertinência μ_1 e μ_2 de x aos conjuntos A_1 e A_2, respectivamente. É importante observar que é necessário não apenas calcular o módulo da distância do ponto x ao centro de um cluster, mas também a direção, para que seja dada correta interpretação do grau de pertinência de y aos conjuntos consequentes.

O cálculo da saída y é feito da seguinte maneira:

$$y = \frac{x\mu_1 + x\mu_2}{\mu_1 + \mu_2} \tag{3.7}$$

Para o caso Takagi-Sugeno, cada cluster define um submodelo linear local. As saídas parciais são calculadas, e os "graus de pertinência" de x aos conjuntos A_1 e A_2 são os "pesos" utilizados na "defuzzyficação" (graus de ativação de regra). A figura 3.11 ilustra um exemplo para o caso de dois clusters.

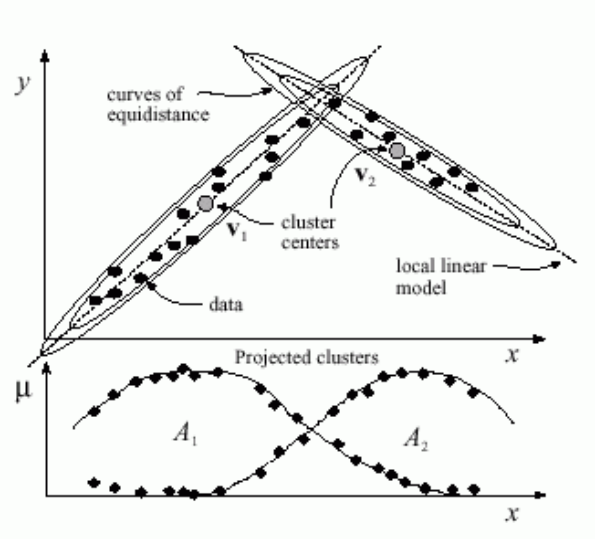


Figura 3.11 - Clusterização - Inferência Takagi-Sugeno

Observa-se que na área de transição entre os submodelos locais tem-se o equivalente a um ponto com 50% de pertinência ao conjunto A_1 e 50% de pertinência ao conjunto A_2. Assim, ocorrerá uma suavização da curva de saída, efeito altamente desejado.

Para sistemas com várias entradas, pode-se definir um tratamento vetorial a todas as variáveis envolvidas na clusterização (HELLENDOORN; DRIANKOV, 1997):

$$Z_k = \left[u_{1k} \; u_{2k} \; \ldots \; u_{pk} \right]^T \in R^p \quad , \quad k = 1 \ldots N$$

$$V = \left[v_1 \; v_2 \; \ldots \; v_c \right]^T \quad , \quad v_i \in R^p$$

$$U = \begin{bmatrix} \mu_{11} & \mu_{12} & \ldots & \mu_{1N} \\ \vdots & \vdots & \vdots & \vdots \\ \mu_{c1} & \mu_{c2} & \ldots & \mu_{cN} \end{bmatrix}$$

onde Z_k representa o vetor de entradas, N é o número de pontos, p é o número de entradas, V é a matriz de centros vetoriais dos clusters, c é o número de clusters, U é a matriz de partição nebulosa e μ_{ik} representa o grau de pertinência de Z_k ao cluster i.

3.9.1 Algoritmo FCM

Um dos algoritmos de clusterização mais utilizados é o FCM - Fuzzy C-Means. O objetivo do FCM, bem como de outros algoritmos de clusterização, é minimizar uma função do tipo

$$J = \sum_{i=1}^{c} \sum_{j=1}^{N} \mu_{ij}^{m} d^2(z_j, v_i) \tag{3.8}$$

onde m é um fator que controla a nebulosidade dos clusters. $m \geq 1$. Quanto maior seu valor, mais "nebulosas" ficam as regiões de transição entre os clusters. Um valor típico é $m = 2$. $d^2(z_j, v_i) = (z_j - v_i)^T (z_j - v_i)$ é a norma euclidiana que representa a distância entre o ponto z_j e o centro v_i do i-ésimo cluster. A variável livre no algoritmo é justamente v_i.

O FCM não leva em conta a direção do ponto em relação a um determinado cluster. Por isso, os agrupamentos são esféricos. No caso do algoritmo de Gustafson-Kessel, a pertinência de um ponto a um cluster tem sua variação dependente tanto da distância em si como da direção (sistema de coordenadas). Assim, os agrupamentos se tornam elipsóides (HELLENDOORN; DRIANKOV, 1997).

Para o FCM, as seguintes restrições têm de ser obedecidas

$$0 \leq \mu_{ij} \leq 1 \quad , \quad i = 1 \ldots c \quad , \quad j = 1 \ldots N \quad \Rightarrow \text{faixa do grau de pertinência;}$$

$$0 < \sum_{j=1}^{N} \mu_{ij} < N \quad , \quad i = 1 \ldots c \quad \Rightarrow \text{nenhum cluster pode ficar vazio; e}$$

$$\sum_{i=1}^{c} \mu_{ij} = 1 \quad , \quad j = 1 \ldots N \quad \Rightarrow \text{pertinência total unitária.}$$

A sequência do algoritmo FCM se inicia com a geração de uma matriz de partição nebulosa U inicial, de forma randômica, respeitando-se as restrições anteriores. Em seguida são calculados os centros dos clusters através de

$$v_i = \frac{\sum_{k=1}^{N} \mu_{i,k}^m z_k}{\sum_{k=1}^{N} \mu_{i,k}^m} \tag{3.9}$$

O cálculo das distâncias

$$d_{ik}^2(z_k, v_i) = (z_k - v_i)^T (z_k - v_i) \tag{3.10}$$

permite a determinação dos graus de pertinência que definem a nova matriz de partição nebulosa

$$\mu_{ik} = \frac{1}{\sum_{j=1}^{c} (d_{ik}/d_{jk})^{2/(m-1)}} \tag{3.11}$$

O próximo passo é calcular o erro entre a matriz U atual e a anterior. Caso $\|\Delta U\| < \epsilon$, finalizar o algoritmo. Caso contrário, voltar para o cálculo dos centros de clusters e repetir toda a sequência.

3.9.2 Product Space Clustering

Ao se utilizar um determinado método de clusterização, uma solução possível é determinar os agrupamentos das entradas, independentemente da saída. Embora satisfatória, essa solução não leva em conta a função entrada-saída. Assim, não há como diferenciar as regiões da faixa de operação que deveriam conter um maior número de clusters, por terem maior variação na saída. Similarmente, são alocados muito mais clusters para certas regiões do que realmente seriam necessários. Dessa forma, a distribuição de clusters ao longo da faixa de operação é uniforme e

não otimizada.

Uma forma mais eficiente de se efetuar a clusterização é utilizar não apenas as entradas mas também a saída na composição do vetor de dados. Assim, a clusterização leva em conta o formato da função entrada-saída. Para as regiões de maior variação na saída são alocados um maior número de clusters, enquanto que para as demais regiões são criados menos clusters. Essa técnica é chamada de Product Space Clustering (BABUSKA, 2001).

Uma desvantagem do Product Space Clustering em relação aos métodos "tradicionais" é a perda do caráter linguístico dos conjuntos nebulosos antecedentes. Por exemplo, não há mais como atribuir a um determinado conjunto as denominações "muito baixo", "baixo", "médio", "alto", etc., pois a distribuição dos clusters não é mais uniforme ao longo da faixa de operação.

3.10 Algoritmo de identificação proposto por Wang e Langari

No trabalho de Wang e Langari (1996), a técnica de clusterização utilizada é a FCM, sem utilizar o Product Space Clustering, ou seja, as entradas são clusterizadas de forma independente, sem considerar a saída.

A clusterização é realizada off-line, e o resultado desta fornece a matriz de partição nebulosa $U = [\mu_{ik}]_{c \times n}$, onde c = número de clusters e n = número de pontos, e o vetor de centros de clusters $V = [v_i]$, onde $i = 1 \ldots c$.

O cálculo dos graus de pertinência de U é feito pelas equações (3.10) e (3.11). É importante ressaltar que as entradas são tratadas de forma separada, ou seja, para p entradas, temos p matrizes de partição nebulosa U e p vetores V de centros de clusters.

Para um novo vetor de entrada, em operação on-line, é preciso calcular os novos graus de pertinência μ_{ik} através das equações (3.10) e (3.11), para cada entrada (total de cálculos = cp). Isso é feito para que se possa calcular a saída. Wang e Langari (1996) comentam que o cálculo dos novos graus de pertinência através das expressões (3.10) e (3.11) consomem muito tempo de processamento, e propõem uma técnica de interpolação, baseada nas pertinências calculadas durante o treinamento (off-line). Essa técnica não será abordada aqui. Para maiores detalhes, ver

(WANG; LANGARI, 1996).

O formato geral das regras é definido por

$$
\begin{aligned}
R_1 &: \text{se } u_1^k \text{ é } \mu_{i1}^k \text{ E } \ldots \text{ E } u_p^k \text{ é } \mu_{ip}^k \text{ então } y_1^k = b_{10} + b_{11}u_1^k + \ \ldots \ + b_{1p}u_p^k \\
R_2 &: \text{se } u_1^k \text{ é } \mu_{i1}^k \text{ E } \ldots \text{ E } u_p^k \text{ é } \mu_{ip}^k \text{ então } y_2^k = b_{20} + b_{21}u_1^k + \ \ldots \ + b_{2p}u_p^k \\
&\vdots \\
R_l &: \text{se } u_1^k \text{ é } \mu_{i1}^k \text{ E } \ldots \text{ E } u_p^k \text{ é } \mu_{ip}^k \text{ então } y_l^k = b_{l0} + b_{l1}u_1^k + \ \ldots \ + b_{lp}u_p^k
\end{aligned}
$$

onde $u_1^k \ldots u_p^k$ = entradas, $y_1^k \ldots y_l^k$ = saídas dos submodelos locais, μ_{ij}^k $(i = 1 \ldots c \ , \ j = 1 \ldots p)$ = graus de pertinência das entradas (matriz U), $k = 1 \ldots n$ = instante de amostragem e $l = c^p$ = número de regras.

É assumido que todas as entradas tenham o mesmo número c de clusters. Um ponto fundamental a ser observado é que as regras devem contemplar todas as combinações possíveis de graus de pertinência μ_{ij}^k. Assim, o número l de regras é igual a c^p. Para cada regra, é preciso interpretar cuidadosamente o índice i dos graus de pertinência. Esse índice não é fixo para uma regra, podendo assumir diferentes valores $(1..c)$ para cada antecedente **dentro da mesma regra**.

Para a obtenção da saída global é feita a "defuzzyficação", através da ponderação das saídas parciais pelos graus de ativação

$$
y_k = \frac{\sum_{i=1}^{l} w_i^k y_i^k}{\sum_{i=1}^{l} w_i^k} \tag{3.12}
$$

Sendo que os graus de ativação de cada regra são determinados por

$$
w_i^k = \mu_{j1}^k \wedge \mu_{j2}^k \wedge \ \ldots \ \wedge \mu_{jp}^k \quad , \quad i = 1 \ldots l \quad , \quad j = 1 \ldots c \tag{3.13}
$$

Uma solução possível para a t-norma $\wedge$ é a utilização do operador mínimo. Observe que para

obter cada grau de ativação w_i^k devemos aplicar a t-norma $\wedge$ para os p graus de pertinência que fazem parte da regra em questão. Novamente, é necessária uma atenção especial ao índice j, o qual pode assumir diferentes valores $(1..c)$ em cada grau de pertinência da equação (3.13), para **um** determinado grau de ativação.

Definindo-se um grau de ativação normalizado por

$$q_i^k \triangleq \frac{w_i^k}{\sum_{i=1}^{l} w_i^k} \tag{3.14}$$

Podemos escrever y_k da forma

$$yk = \sum_{i=1}^{l} q_i^k y_i^k = \sum_{i=1}^{l} q_i^k (b_{i0} + b_{i1} u_1^k + \ \ldots \ + b_{ip} u_p^k) \tag{3.15}$$

Para a estimação dos parâmetros b_{ij} $(i = 1 \ldots l \, , \, j = 0 \ldots p)$ é utilizado o método dos mínimos quadrados. A função custo a ser minimizada é

$$J = \sum_{k=1}^{n} (y^k - \sum_{i=1}^{l} q_i^k y_i^k)^2 = \| Y - \Phi b \|^2 \tag{3.16}$$

onde

$$Y = \left[y^1 \ y^2 \ \ldots \ y^n \right]^T \triangleq \left[y_1 \ y_2 \ \ldots \ y_n \right]^T$$

$$b = [b_{10} \ldots b_{l0} \ \ b_{11} \ldots b_{l1} \ b_{1p} \ldots b_{lp}]^T$$

$$\Phi = \begin{bmatrix} q_1^1 \ldots q_l^1 & q_1^1 u_1^1 \ldots q_l^1 u_1^1 & \cdots & q_1^1 u_p^1 \ldots q_l^1 u_p^1 \\ q_1^2 \ldots q_l^2 & q_1^2 u_1^2 \ldots q_l^2 u_1^2 & \cdots & q_1^2 u_p^2 \ldots q_l^2 u_p^2 \\ \vdots & \vdots & \vdots & \vdots \\ q_1^n \ldots q_l^n & q_1^n u_1^n \ldots q_l^n u_1^n & \cdots & q_1^n u_p^n \ldots q_l^n u_p^n \end{bmatrix}$$

Portanto, o vetor de parâmetros dos consequentes b é calculado pela equação (3.5)

$$b = (\Phi^T \Phi)^{-1} \Phi^T Y$$

Em face das dimensões do sistema acima, computacionalmente não é recomendada a implementação direta do método clássico de mínimos quadrados. Wang e Langari (1996) propõem um algoritmo recursivo

$$f_k = C_{k-1}^T \phi_k^T \tag{3.17}$$

$$\gamma_k = [1 + f_k^T f_k]^{1/2} \tag{3.18}$$

$$\alpha_k = \gamma_k + 1 \tag{3.19}$$

$$\beta_k = \frac{1}{(\gamma_k \alpha_k)} \tag{3.20}$$

$$\sigma_k = \beta_k C_{k-1} f_k \tag{3.21}$$

$$b_k = b_{k-1} + \sigma_k \left[\frac{\alpha_k}{\gamma_k} (y_k - \phi_k b_{k-1}) \right] \tag{3.22}$$

$$C_k = C_{k-1} - \sigma_k f_k^T \tag{3.23}$$

sendo ϕ_k a k-ésima linha de Φ. C_k é a decomposição de Cholesky de P_k, este último dado por

$$P_k = (\sum_{i=1}^{n} \phi_i^T \phi_i)^{-1} \tag{3.24}$$

Uma vez obtido o modelo off-line, o processo de estimação on-line da saída, a partir de um novo vetor de p entradas, se inicia com o cálculo, para cada entrada, dos novos graus de pertinência a cada cluster (total de cp cálculos).

Para cada regra ($l = c^p$) deve ser determinado o seu grau de ativação w_i , $i = 1 \ldots l$, através do operador t-norma $\wedge$. Em seguida são calculados os graus de ativação normalizados q_i.

O próximo passo é efetuar o cálculo de todas as saídas parciais y_i , $i = 1 \ldots l$, e então calcular a saída global y.

3.11 Proposta de otimização do trabalho de Wang e Langari: Regras Limitadas

O método de Wang e Langari (1996) apresenta dois grandes inconvenientes. O primeiro consiste em se efetuar a clusterização das p entradas de forma independente e não vetorial. Isso cria a necessidade de se ter c^p regras, levando o sistema a sofrer da maldição da dimensionalidade. O segundo inconveniente consiste em não se levar em conta a função entrada-saída, ou seja, distribuir os clusters de maneira uniforme ao longo da faixa de operação. Isso geralmente provoca um desperdício computacional e um maior erro na saída global.

Dessa forma, Oliveira e Garcia (2003) propõem uma otimização do método de Wang e Langari (1996), justamente corrigindo os inconvenientes deste último. Assim, o tratamento das variáveis passa a ser de forma vetorial. Têm-se um vetor de entradas de dimensão p e centros de clusters vetoriais de mesma ordem. O outro ponto de otimização é a realização da clusterização levando-se em conta a função entrada-saída. Assim, para um determinado instante k, o vetor que é passado ao FCM possui dimensão $p + 1$, sendo a última coluna a saída y_k. A distribuição de clusters vetoriais ao longo da faixa de operação não é mais uniforme, e sim otimizada de acordo com a amplitude e taxa de variação da saída.

Dessa forma, o número de regras l é o mesmo do número c de clusters. Daí vem o nome "Regras Limitadas" dado ao algoritmo.

O treinamento off-line do modelo se inicia com a coleta de n pontos. A matriz de entrada

assume o formato

$$Z = \begin{bmatrix} u_1^1 & \dots & u_p^1 & y^1 \\ \vdots & \vdots & \vdots & \vdots \\ u_1^n & \dots & u_p^n & y^n \end{bmatrix}$$

Em seguida deve ser definido o número de clusters e efetuada a clusterização via FCM. Os pontos possuem dimensão $p+1$. A matriz resultante de centros de cluster possui dimensão $c \times (p+1)$. Elimina-se a última coluna, pois refere-se à saída, que on-line deve ser estimada. Cada linha refere-se a um centro vetorial. O formato fica

$$V = \begin{bmatrix} v_{11} & \dots & v_{1p} \\ \vdots & \vdots & \vdots \\ v_{c1} & \dots & v_{cp} \end{bmatrix}$$

O próximo passo consiste na determinação da matriz de partição nebulosa U, calculando-se os graus de pertinência pelas equações (3.10) e (3.11), resultando em

$$U = \begin{bmatrix} \mu_{11} & \mu_{12} & \dots & \mu_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ \mu_{c1} & \mu_{c2} & \dots & \mu_{cn} \end{bmatrix}$$

Os graus de ativação de cada regra são os próprios graus de pertinência

$$q_i^k = \mu_i^k \quad , \quad i = 1 \dots c$$

Assim, é feito o cálculo da saída global, baseado nas saídas parciais e nos graus de ativação

$$yk = \sum_{i=1}^{l} q_i^k y_i^k = \sum_{i=1}^{l} q_i^k (b_{i0} + b_{i1} u_1^k + \ \ldots \ + b_{ip} u_p^k)$$

A estimação do vetor de parâmetros dos consequentes b segue a resolução já apresentada, através do método recursivo proposto por Wang e Langari (1996). Por fim, o modelo obtido deve ser validado através de algum índice de desempenho.

Para um novo vetor de entrada num instante k, o cálculo da saída estimada on-line se inicia com a determinação, para cada regra ou cluster, do vetor diferença entre o vetor de entrada e o respectivo centro de cluster. A seguir, são calculadas as normas euclidianas quadráticas dos vetores diferença mencionados, com o posterior cálculo dos graus de pertinência (graus de ativação das regras) do vetor de entrada a cada cluster ($c = l$ cálculos). Finalizando o algoritmo on-line, são calculadas as $c = l$ saídas parciais e a saída global y_k.

3.12 Comentários sobre a estrutura do modelo

Pode-se dividir o modelo estudado em duas partes, a parte antecedente e a parte consequente. Na parte antecedente, que pode ser chamada de parte nebulosa, estão os procedimentos que visam a obtenção dos graus de ativação de cada regra ou submodelo linear.

Na parte consequente estão exatamente os submodelos das saídas parciais, os quais compõem a saída global. Do ponto de vista de identificação de sistemas, é possível situar um submodelo qualquer dentro de uma estrutura clássica de modelo, como ARX, ARMAX, etc. As características básicas de um submodelo são a sua linearidade, é do tipo SISO, paramétrico, invariante no tempo, feito no domínio do tempo, em tempo discreto e a parâmetros concentrados.

A estrutura do i-ésimo submodelo linear é dada por

$$\begin{aligned} pH_i(k) = {} & b_{i0} + b_{i1} q_3(k-1) + b_{i2} q_3(k-2) + b_{i3} q_3(k-3) + \\ & b_{i4} q_3(k-4) + b_{i5} pH(k-1) + b_{i6} pH(k-2) \end{aligned} \qquad (3.25)$$

Como a equação (3.25) não contempla o erro de modelagem explicitamente, o que poderia caracterizá-la como uma estrutura ARX ou ARMAX (GARCIA, 2001), a saída parcial desse i-ésimo submodelo pode ser escrita por

$$y_i(k) = pH_i(k) + e(t) \tag{3.26}$$

A estrutura acima, com exceção do termo independente b_{i0}, é semelhante à estrutura OE - Output Error.

4 ESTRUTURA GERAL DE HARDWARE E SOFTWARE PARA A IMPLEMENTAÇÃO DO SENSOR VIRTUAL

4.1 Análise do modelo identificado - estrutura computacional

As simulações feitas por Oliveira e Garcia (2003) permitiram definir para o processo de neutralização de pH os parâmetros considerados ótimos:

- Número de regressores de vazão de base q_3: 4;
- Número de regressores de *pH* estimado: 2; logo, a dimensão do vetor de entradas é $p = 6$; e
- Número ótimo de clusters (e, por conseguinte, de regras): $c = l = 20..28$. Simulações com modelos de 20 a 28 clusters foram realizadas em Matlab, revelando a melhor relação custo/benefício para o modelo de 20 clusters.

Assim, o firmware do sensor virtual deve conter a matriz de centros de clusters com $c \times p = 20 \times 6 = 120$ elementos, bem como a matriz de parâmetros dos consequentes, com $(p + 1) \times c = (6 + 1) \times 20 = 140$ elementos. Logo, o total de elementos a serem armazenados são $120 + 140 = 260$.

O algoritmo de sensor virtual (Regras Limitadas) que deve ser implementado no firmware do DSP é dado por

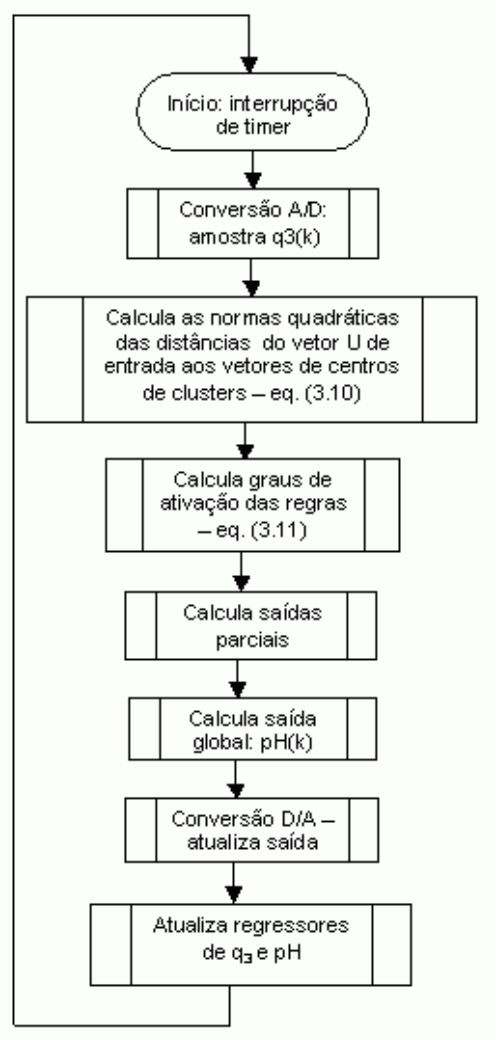


Figura 4.1 - Fluxograma do sensor virtual em firmware

Para uma visualização dos coeficientes a serem armazenados no DSP, o centro de cluster número 1 do modelo treinado em Matlab (4020 pontos, 20 clusters) é dado por

$$q_3(k-1) = 0,4048132844331$$
$$q_3(k-2) = 0,4039349568989$$
$$q_3(k-3) = 0,4035852862376$$
$$q_3(k-4) = 0,4047319753516$$
$$pH(k-1) = 6,3189378071355$$
$$pH(k-2) = 6,3198447348479$$

O submodelo linear número 1 é dado por

$$b_{10} = 0,33131256751638$$
$$b_{11} = 0,04913673996249$$
$$b_{12} = 0,02746331439865$$
$$b_{13} = -0,01721430276881$$
$$b_{14} = -0,01640534205397$$
$$b_{15} = 0,40540922782838$$
$$b_{16} = 0,53827977826054$$

4.2 Escolha da ferramenta de hardware / firmware - critérios adotados

A estrutura de modelo proposta para o sensor virtual é de entrada-saída, baseada em equação de diferenças. Dessa forma, naturalmente há um indicativo forte para a utilização de um processador digital de sinais (DSP), pois sua arquitetura é preparada para o processamento de equações de diferenças. O uso de microcontroladores de alta performance seria até possível, porém haveria necessidade de uma maior implementação de software, pelo fato de sua arquitetura de hardware não ser otimizada para operações típicas de processamento digital de sinais.

Um dos itens mais relevantes na implementação de um sistema em hardware é com relação à memória necessária. Os recursos disponíveis no hardware / firmware são **muito** menores que os de um microcomputador, daí a necessidade de se otimizar ao máximo toda a estrutura de software. O projetista não pode encarar o sistema apenas do ponto de vista do computador e do Matlab, e sim analisar cuidadosamente quais etapas podem ser otimizadas e, a partir daí, escolher o hardware mínimo necessário para a implementação do sistema com sucesso.

No caso em questão, optou-se pela utilização do processador digital de sinais da família TMS320C2XX, da Texas Instruments (TEXAS INSTRUMENTS, 1997), através de um starter kit, ou seja, uma placa com o DSP e alguns circuitos de apoio, utilizada para o desenvolvimento

de aplicações. O DSP presente no kit é o TMS320F206 (TEXAS INSTRUMENTS, 1998), de 16 bits, 20 Mips (milhões de instruções por segundo), ponto fixo e com memórias FLASH EEPROM e RAM internas.

Para o subsistema de aquisição de dados serão utilizados conversores A/D e D/A paralelos de 12 bits, ligados ao barramento de I/O do DSP.

Em resumo, a adoção do sistema de hardware / firmware descrito acima deveu-se ao fato do modelo de sensor virtual possuir estrutura tal que permita a aplicação eficiente de um DSP, bem como a disponibilidade da ferramenta de desenvolvimento, com ambiente de programação em assembly, e uma viabilidade comercial, uma vez que o hardware fica extremamente compacto e robusto. O custo é reduzido, por ser um processador em ponto fixo e de uma família já em amplo uso comercial e industrial (TMS320C2XX). Os detalhes de implementação de hardware e software são tratados no capítulo 5.

Um outro ponto importante na escolha do DSP em questão foi a facilidade de interconexão em rede (RS-485) através de comunicação serial assíncrona. Assim, é possível formar uma rede de instrumentos inteligentes sob a supervisão de um elemento mestre, como por exemplo um PC industrial.

4.3 Diagramas em blocos e em Simulink - validação e testes do sensor virtual em PC e em hardware

4.3.1 Diagramas do sistema de coleta de dados, identificação e validação do modelo em PC

A primeira etapa para a implementação do sensor virtual em hardware é efetuar a coleta de dados (batelada) para posterior aplicação dos algoritmos de identificação nebulosa off-line. A figura 4.2 ilustra o processo de coleta de dados e identificação.

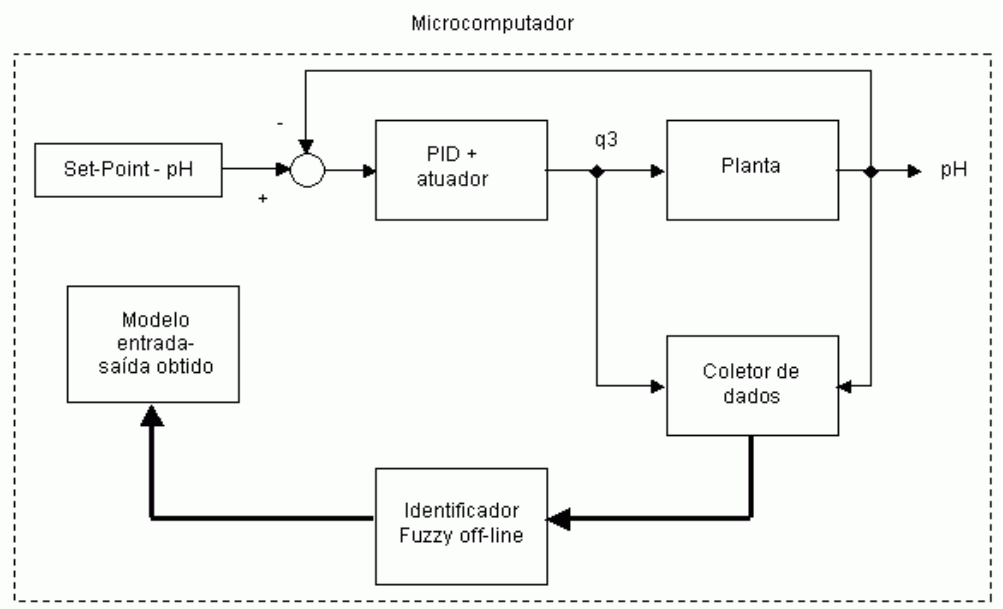


Figura 4.2 - Coleta de dados para identificação do sensor virtual

Observa-se que é utilizado um controle em malha fechada, sendo o valor de referência de pH constituído de duas rampas, em sequência, uma ascendente e outra descendente. A faixa de variação total do valor de referência é de 4 a 10. A idéia é fazer com que sejam coletados pontos em toda a faixa útil, a fim de que o modelo obtido possa ser utilizado em aplicações industriais. Caso a coleta seja feita em um setor restrito, é provável que o sensor virtual não trabalhe bem nos outros setores, com risco até de divergência total (instabilidade). A curva de treinamento é semelhante à da figura 6.5 (vide capítulo 6).

O subsistema em Simulink de coleta de dados é composto de 4 regressores de q_3 e dois regressores do *pH*, e está ilustrado na figura 4.3, consistindo-se no projeto do experimento, para situar o leitor em relação às etapas da teoria de identificação de sistemas (GARCIA, 2001).

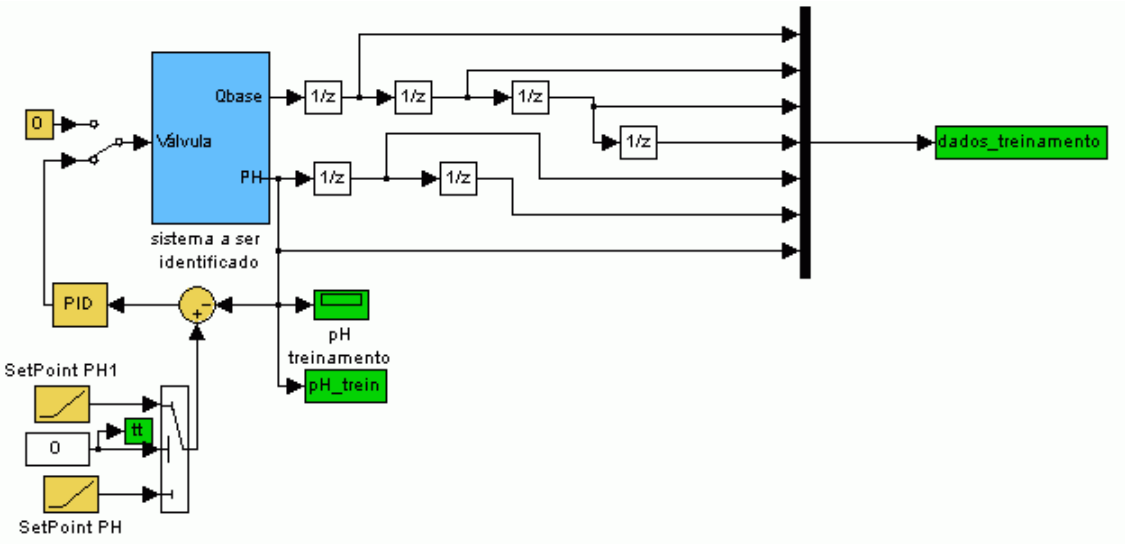


Figura 4.3 - Subsistema em Simulink para a coleta de dados para identificação

Após a realização da coleta de dados, é aplicado o algoritmo "Regras Limitadas", obtendo-se o modelo já descrito. Antes de uma implementação em hardware, é prudente efetuar testes de validação do modelo obtido ainda em ambiente Matlab / Simulink. Para isso, o algoritmo de execução on-line é passado para o Simulink através de um bloco "Matlab Function". Além disso, a validação é cruzada, ou seja, utilizam-se dados diferentes do treinamento, inclusive com valores de referência diferentes (fixo, degrau e senóide). A curva de pH estimado obtida do sensor virtual é comparada com a saída "real" do processo, e o índice de desempenho utilizado é o ISE - Integrated Square Error, ou erro quadrático integrado. Esse índice acumula, ponto a ponto, a diferença quadrática entre o pH estimado e o pH "real". O diagrama Simulink de validação do sensor virtual está ilustrado na figura 4.4.

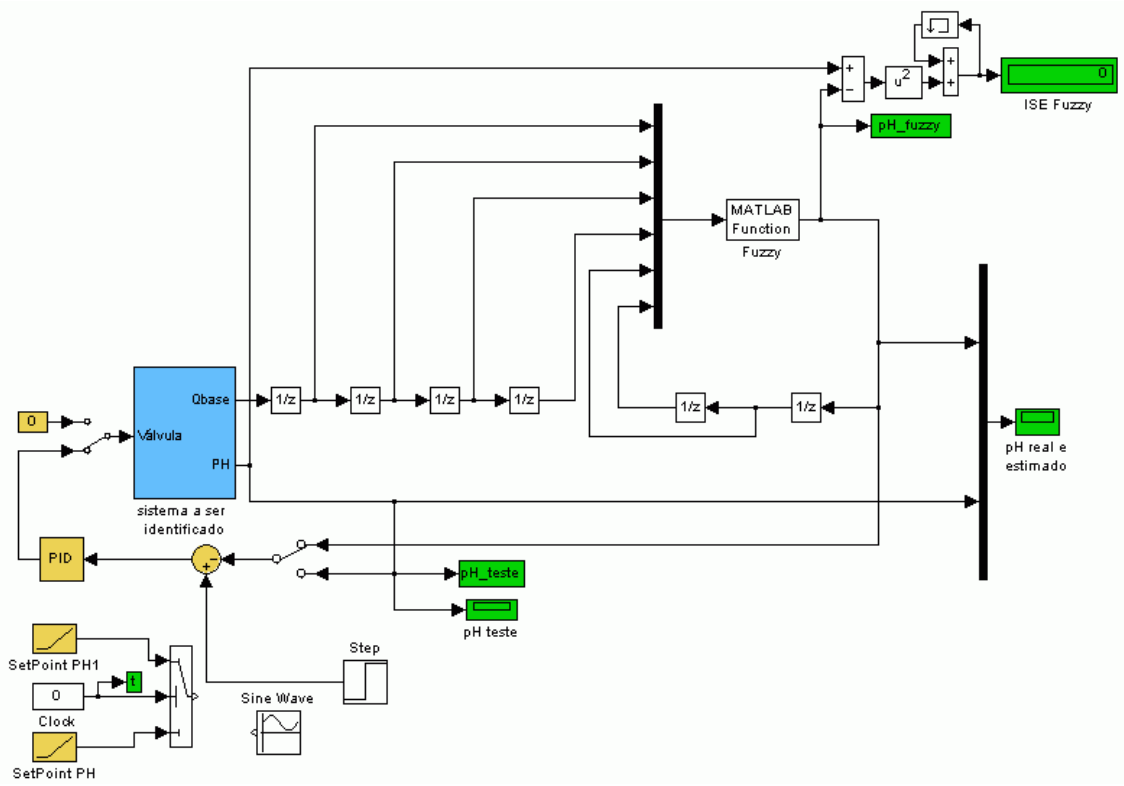


Figura 4.4 - Diagrama Simulink para validação do modelo obtido em PC

4.3.2 Diagramas de validação do sensor virtual em hardware

Uma vez validado o modelo de sensor em PC, torna-se segura a transcrição desse modelo para o firmware do DSP. A figura 4.5 ilustra o experimento realizado para a validação do sensor virtual desenvolvido em hardware.

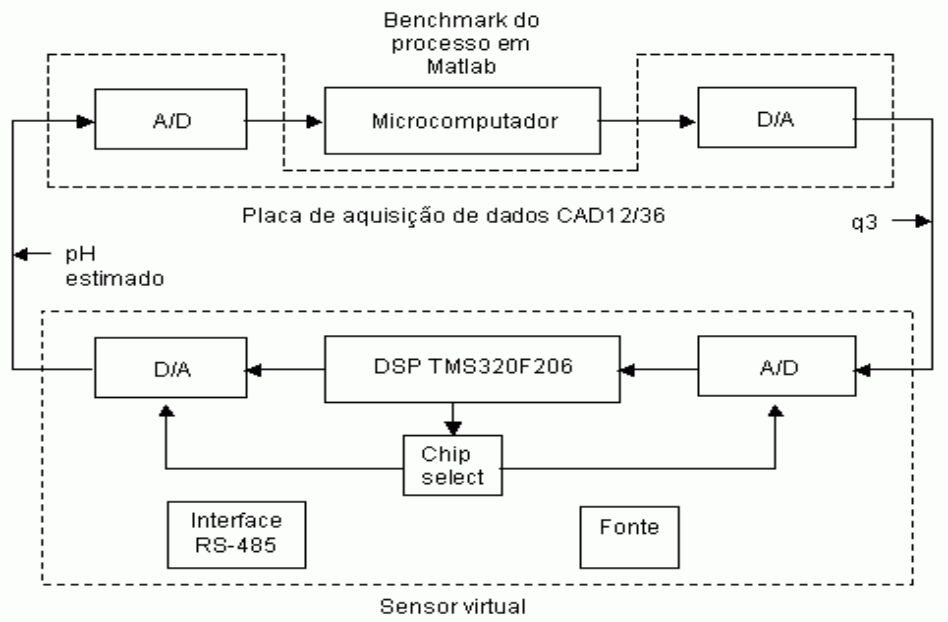


Figura 4.5 - Validação do sensor virtual em hardware

O microcomputador executa o "benchmark" do processo de neutralização de pH, em ambiente Matlab / Simulink, e os sinais são exteriorizados através de uma placa de aquisição de dados da Lynx Tecnologia, modelo CAD12/36 (LYNX TECNOLOGIA ELETRÔNICA LTDA, 1993).

O sensor virtual estima, em tempo real, o *pH*, baseado na vazão de base q_3. Ao ser coletado pela CAD12/36, a idéia é que o *pH* estimado seja comparado ao *pH* real, para fornecer um parâmetro de erro. O objetivo é minimizar este erro, a tal ponto que possa ser validado o sensor virtual.

Para a integração da placa CAD12/36 ao ambiente Matlab / Simulink foram utilizados drivers real time escritos em linguagem C pelo Prof. Dr. Ricardo Paulino Marques, do Laboratório de Automação e Controle - LAC - do departamento de Telecomunicações e Controle - PTC - da Escola Politécnica da Universidade de São Paulo - EPUSP.

A pimeira simulação comparativa foi em malha aberta. Assim, o modelo em PC (figura 4.4) foi testado desconectando-se a sua entrada q_3 do processo, e injetando-se nela o sinal desejado. Para o VS01A foi feito o mesmo teste, ilustrado na figura 4.6. Vê-se que o processo nem consta do diagrama.

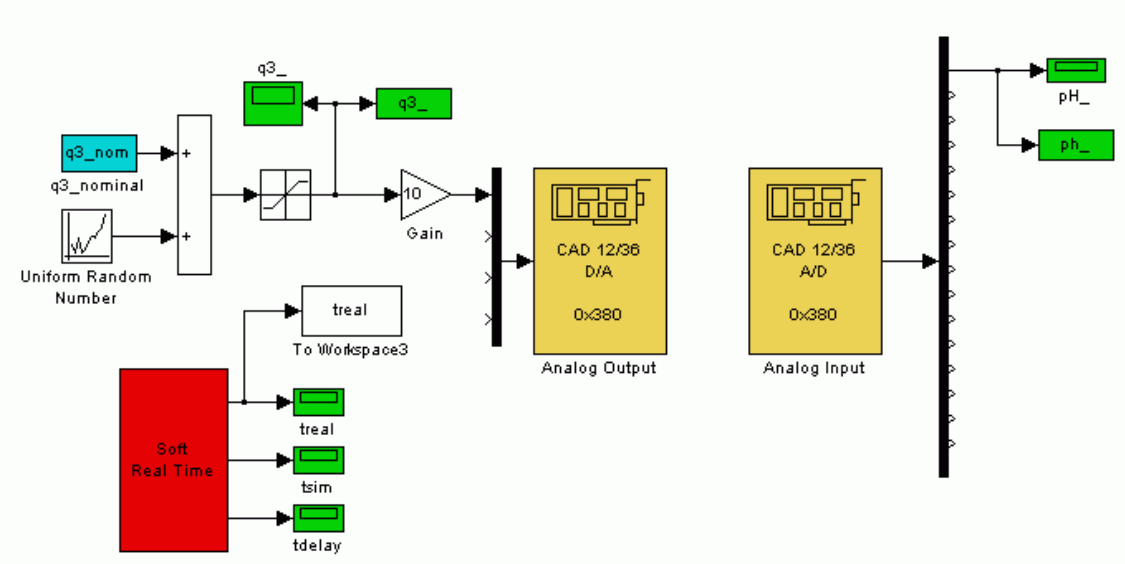


Figura 4.6 - Validação do VS01A em malha aberta

Feitas as simulações em malha aberta, o próximo passo foi verificar o comportamento dos sensores junto ao processo. Numa primeira etapa, o controle em malha fechada utilizou uma realimentação fornecida pelo sensor "real". O objetivo foi validar o VS01A para excursões de pH ao longo de toda a faixa de operação, com valores de referência distintos. A simulação do modelo em PC utilizou o diagrama da figura 4.4. Para o VS01A, a figura 4.7 ilustra o experimento em questão.

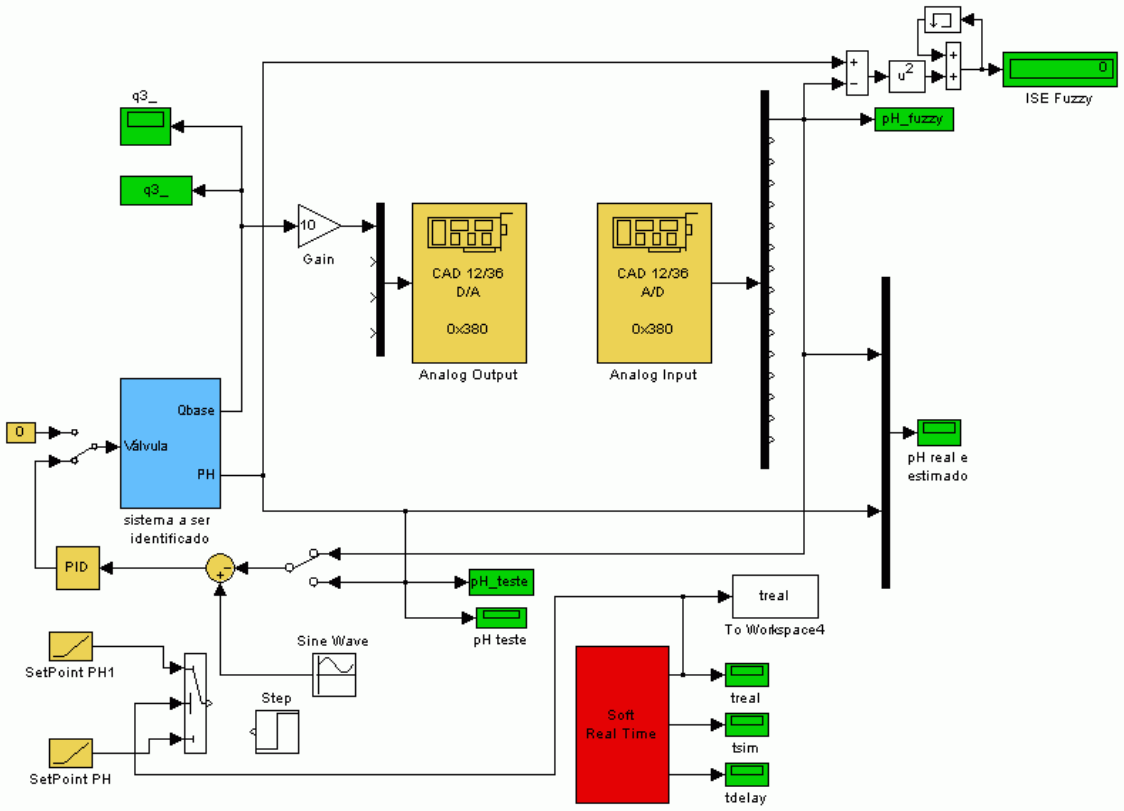


Figura 4.7 - Validação do VS01A junto ao processo

4.3.3 Diagrama de controle em malha fechada operando com o sensor virtual em hardware

Até o presente momento, os modelos de sensor em PC em hardware não foram utilizados como elos de realimentação para o controlador. Verificar tal comportamento é crucial, pois define a capacidade do modelo de atuar efetivamente como sensor. Assim, o próximo passo foi efetuar simulações tendo como elemento sensor o modelo treinado. A figura 4.8 ilustra o processo sendo controlado em malha fechada, utilizando-se o sensor virtual em hardware.

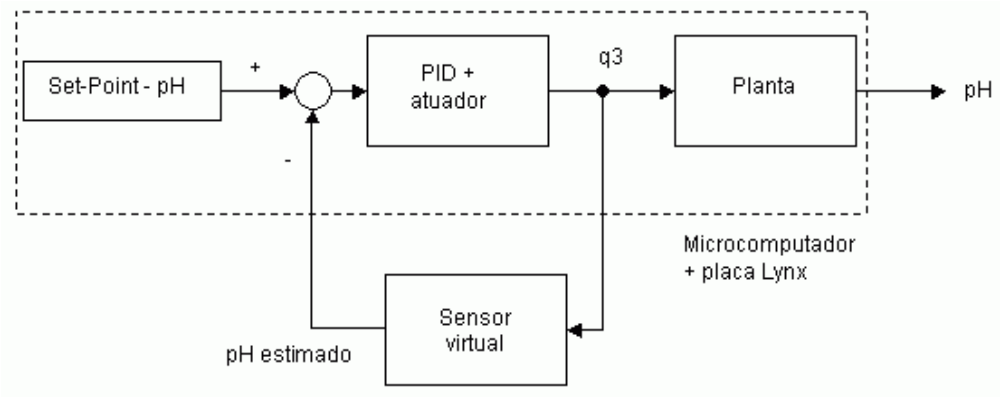


Figura 4.8 - Operação do sensor virtual em hardware em malha fechada

Observa-se que, do ponto de vista do controlador, não há diferença alguma entre a utilização do sensor real ou do sensor virtual.

4.3.4 Controle em malha fechada com o controlador e o sensor virtual em hardware

A figura 4.9 ilustra a última etapa proposta para este trabalho, a qual consistiu em incorporar um controlador PID ao hardware, juntamente com o sensor virtual.

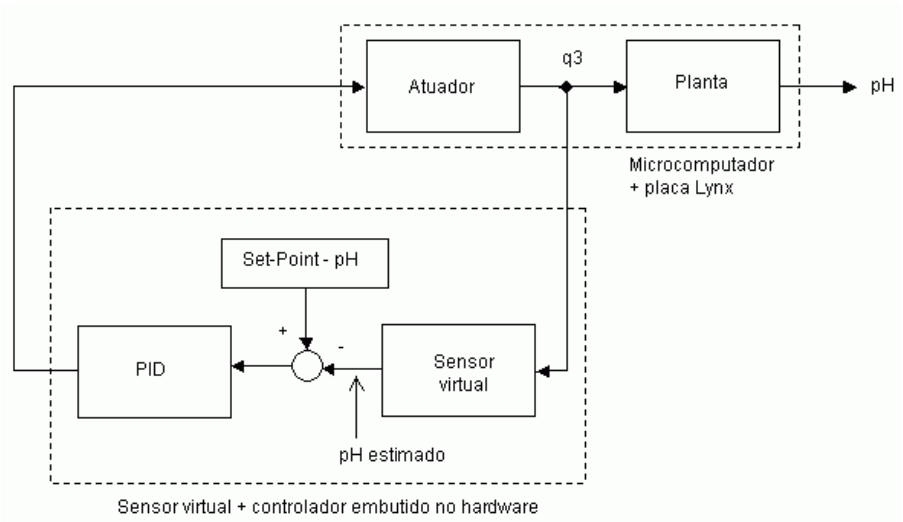


Figura 4.9 - Sensor virtual + PID embutido no hardware

Vê-se que a idéia é fornecer uma solução mais completa para o controle do processo. O projeto do PID, assim como o do próprio sensor, incorpora detalhes práticos, como range numérico, anti reset wind-up, etc. No caso do VS01A, a incorporação do PID junto ao sensor foi realizada em duas etapas. A primeira consistiu em testar isoladamente o PID em hardware, a fim de validá-lo previamente. A figura 4.10 ilustra o diagrama de simulação para o teste isolado do PID.

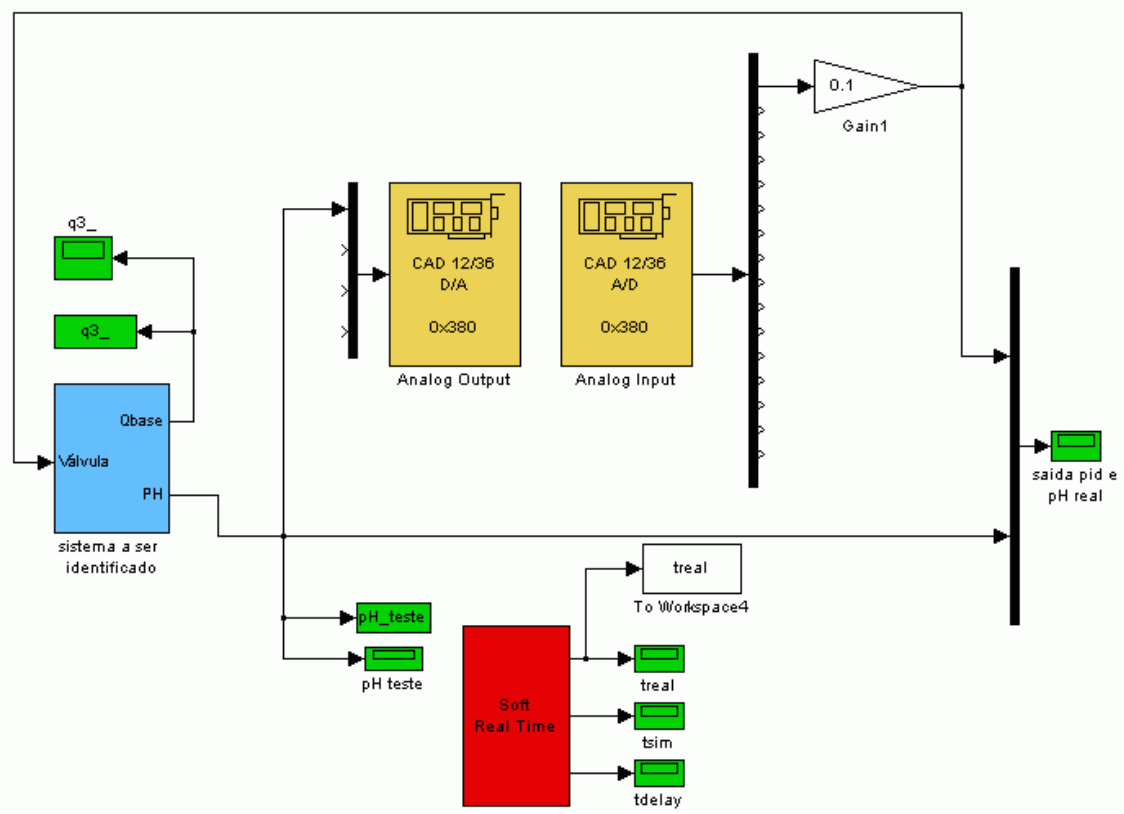


Figura 4.10 - Teste isolado do controlador PID embutido no VS01A

Com o PID validado, a etapa seguinte consistiu em sua incorporação ao firmware do sensor virtual. O teste inicial foi realizado com valor de referência fixo em pH = 7, e o diagrama de simulação está ilustrado na figura 4.11.

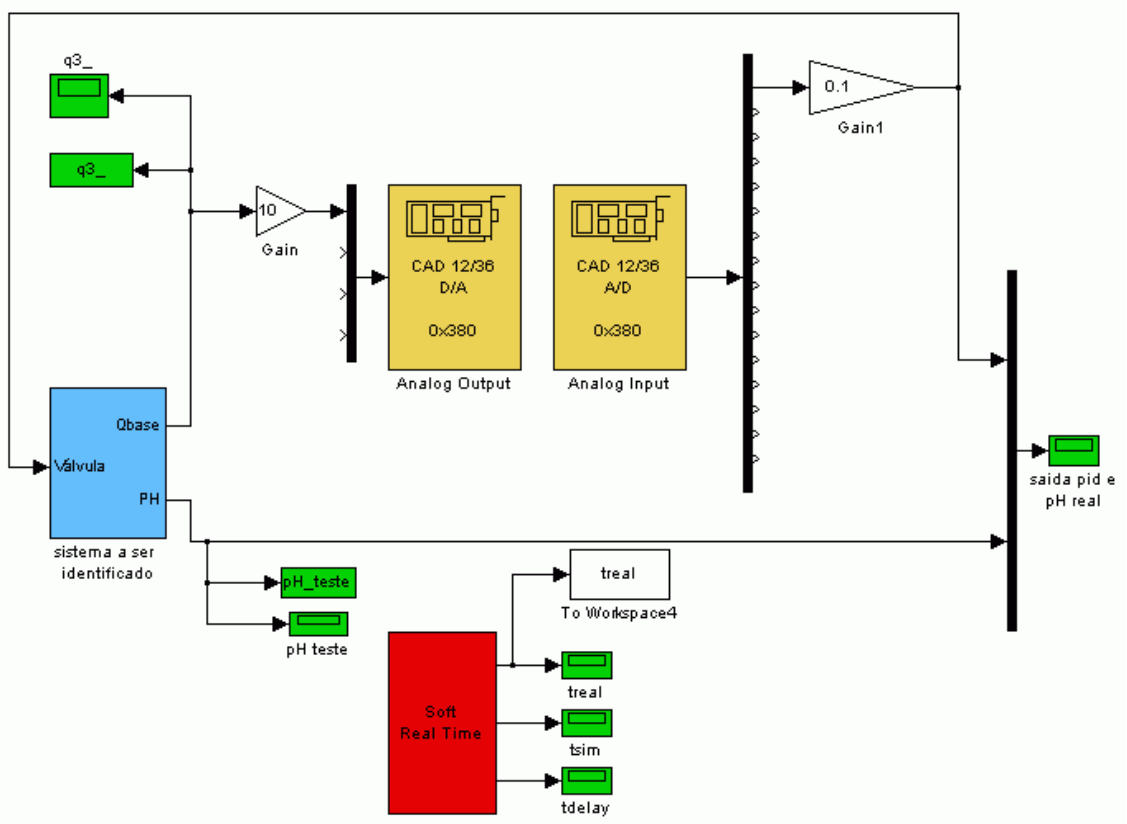


Figura 4.11 - Teste do VS01A + PID embutido

5 PROJETO DETALHADO DO PROTÓTIPO VS01A

5.1 Projeto de hardware

5.1.1 Descrição geral do sistema

Conforme já mencionado, a idéia principal em torno do hardware foi desenvolver um módulo "embedded", ou seja, autônomo e genérico, visando sua utilização em diversos processos. Para tanto, alguns requisitos devem ser obedecidos pelo sistema para a sua aceitação junto à indústria. Primeiro, deve-se ter padrões de entrada e saída compatíveis com os utilizados no mercado: 0..10V, 0..5V, 4..20mA, etc, uma vez que receberá sinais de sensores comerciais e fornecerá sinais para outros semelhantes. Segundo, deve preferencialmente possuir mais de uma entrada e saída, pois dependendo do processo, a estimação de uma variável de interesse pode depender de duas ou mais variáveis.

Como se trata basicamente de um sistema de aquisição, processamento e devolução de dados, a resolução dos conversores A/D e D/A deve ser tal que permita razoável precisão na discretização de variáveis externas. Em relação ao tempo de processamento, deve-se trabalhar via interrupção com taxa precisa e configurável, para que se tenha o algoritmo rodando numa taxa de amostragem confiável requerida pelo processo.

O protótipo desenvolvido, batizado de VS01A, apresenta os seguintes recursos:

- Processador digital de sinais TMS320F206, de 16 bits, com arquitetura em ponto fixo, 20 Mips, 32k words de memória de programa Flash interna e 544 words de RAM interna;
- Interrupção de timer programável, para configuração da taxa de amostragem;
- 4 canais analógicos de entrada, com resolução de 4096 pontos, e interface para

sinais 0..10V, 0..5V, 1..5V, 0..20mA e 4..20mA;

- Conversor A/D - analógico/digital - paralelo de 12 bits, por aproximações sucessivas, com taxa de conversão máxima de 156 KHz;
- Conversor D/A - digital/analógico - paralelo de 12 bits, com tempo de acomodação máximo de 10 us;
- 4 canais analógicos de saída, com resolução de 4096 pontos, e interface para sinais 0..10V, 0..5V e 1..5V; e
- Interface para rede de comunicação serial assíncrona no padrão RS-485.

5.1.2 Interfaces analógicas de entrada e saída

A interface de entrada está baseada no amplificador operacional OPA2704 da Texas Burr Brown, de alta performance, com design voltado à área de instrumentação. A impedância de entrada está na faixa de 4 Gohms. A seleção do tipo de entrada é feita por estrapes. Após o condicionamento do sinal, este passa por um filtro passa-baixas anti-aliasing e então é aplicado à entrada do conversor A/D, com range de 0 a 5V.

A interface de saída é baseada também no OPA2704. O sinal recebido da saída do conversor D/A possui range de 0 a 2,5V. Assim, a configuração do tipo de saída via estrapes efetua a seleção do ganho correto, para fundo de escala de 5V ou 10V. Antes de ser aplicado à interface de saída, o sinal do D/A passa por um filtro passa-baixas anti-imaging, de reconstituição do sinal.

O protótipo VS01A apresenta apenas uma interface de entrada e uma de saída, uma vez que essa configuração é suficiente para o sensor virtual aplicado ao processo de neutralização de pH. Para uma capacidade plena de utilização, os circuitos de entrada e saída devem ser reproduzidos para os três canais restantes.

5.1.3 Conversor A/D

A etapa de conversão A/D é baseada no conversor ADS7842 da Texas Burr Brown. Como o conversor é de 12 bits, isso garante ao sistema 4096 pontos de resolução. A tensão de referência é de 5V, gerada a partir de um subsistema regulador de 2,5V de precisão.

A interface digital é paralela, sendo a razão dessa escolha a existência de barramento de dados, endereço e controle no DSP. O A/D é encarado como um periférico de i/o do processador, e a transferência paralela de dados é bastante simples, exigindo apenas a correta manipulação de sinais de controle do barramento. Para uma visualização geral do processo de conversão A/D tem-se a carta de tempos ilustrada na figura 5.1.

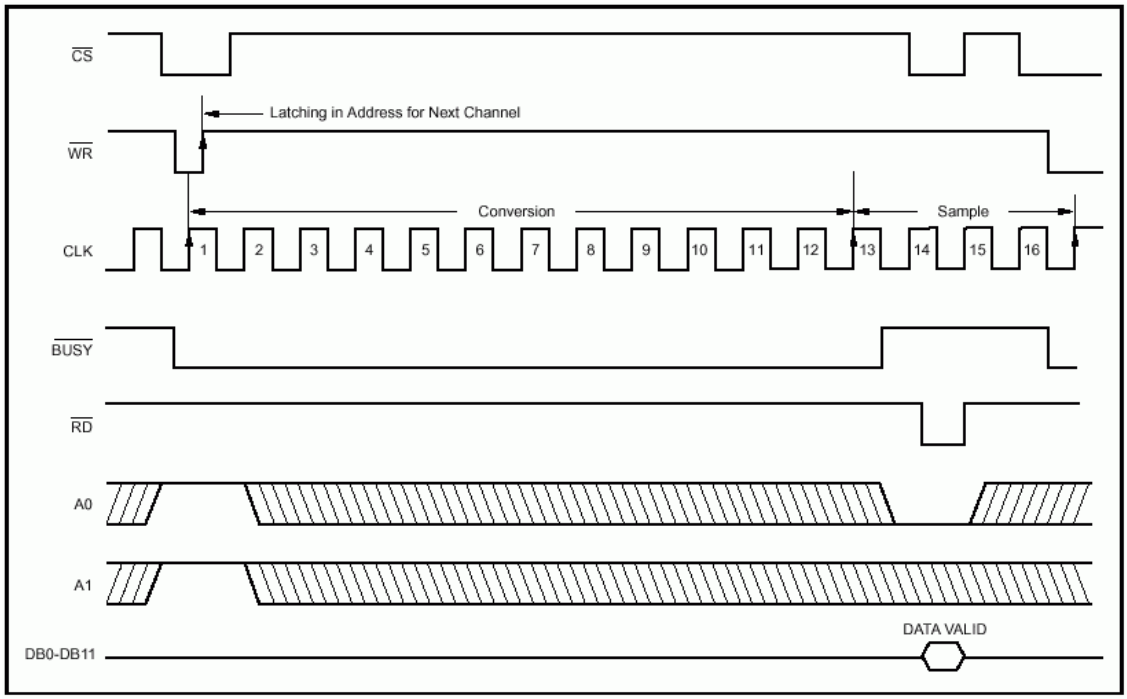


Figura 5.1 - Carta de tempos do conversor A/D ADS7842

O A/D é mantido sempre habilitado, através do aterramento de seu pino CS\. O DSP efetua o controle do disparo de conversão e da leitura do resultado através de 3 vias: WR\, RD\ e BUSY\. Observe que as operações dependem de um clock, cujo valor máximo é de 3,2 MHz. No caso, está sendo utilizado um sinal obtido a partir do clock do DSP (20 MHz), dividido por 8 (74LS93),

resultando assim em uma base de tempo de 2,5 MHz para o A/D. Uma das vantagens de se utilizar uma derivação do clock central do processador é a garantia de sincronismo das operações, já que todos os sinais de controle são temporizados pela mesma base de tempo.

As vias de endereço A1 e A0 para seleção do canal a ser convertido (0..3) estão aterradas, fixando a utilização do canal 0. Para uma configuração universal, essas vias devem ser diretamente conectadas às vias A1 e A0 do barramento de endereços do DSP.

O disparo de uma nova conversão se inicia com um bordo de subida do clock estando o sinal WR\ em zero. A queda de WR\ é gerada pelo DSP, através de um acesso de escrita de i/o. Ao detectar essa ocorrência, o A/D informa ao DSP o início da conversão, através da queda do sinal BUSY\. Este sinal é ligado a uma entrada lógica do processador. O DSP monitora o estado dessa entrada, sendo que a subida do sinal BUSY\ indica fim de conversão.

Para realizar a leitura do valor digital convertido, o processador executa um acesso de leitura de i/o, gerando uma queda do sinal RD\.

São necessários 12 ciclos de clock para uma conversão e 4 ciclos para a aquisição de nova amostragem, totalizando 16 ciclos de clock. Como a base de tempo é de 2,5 MHz, tem-se uma taxa de conversão máxima de 156,25 KHz.

O formato de saída do A/D é o chamado "straight binary", ou seja, vai de 000h (0V) a FFFh (4,99878V). O outro formato possível seria o complemento de 2, no qual o bit mais significativo representa o sinal, positivo ("0") ou negativo ("1"). Assim, a faixa positiva de valores (2,5V a 4,99878V) iria de 000h a 7FFh, e a faixa negativa (0V a 2,49878V) iria de 800h a FFFh.

Como a tensão de referência é de 5V, a resolução da entrada é de 5V/4096 = 1,22 mV. O valor digital de saída é dado por

$$N = \frac{V_{in} \times 4096}{5} \qquad (5.1)$$

sendo N a saída em 12 bits e V_{in} a tensão em volts aplicada à entrada do conversor.

5.1.4 Conversor D/A

A etapa de conversão D/A é baseada no conversor DAC7625 da Texas Burr Brown. Assim como o conversor A/D, este D/A é de 12 bits, garantindo ao sistema 4096 pontos de resolução. A tensão de referência é de 2,5V, fornecida por um regulador zener de precisão LM431.

A interface digital é paralela, pelos mesmos motivos descritos para o conversor A/D. Para uma visualização geral do processo de conversão D/A tem-se a carta de tempos ilustrada na figura 5.2.

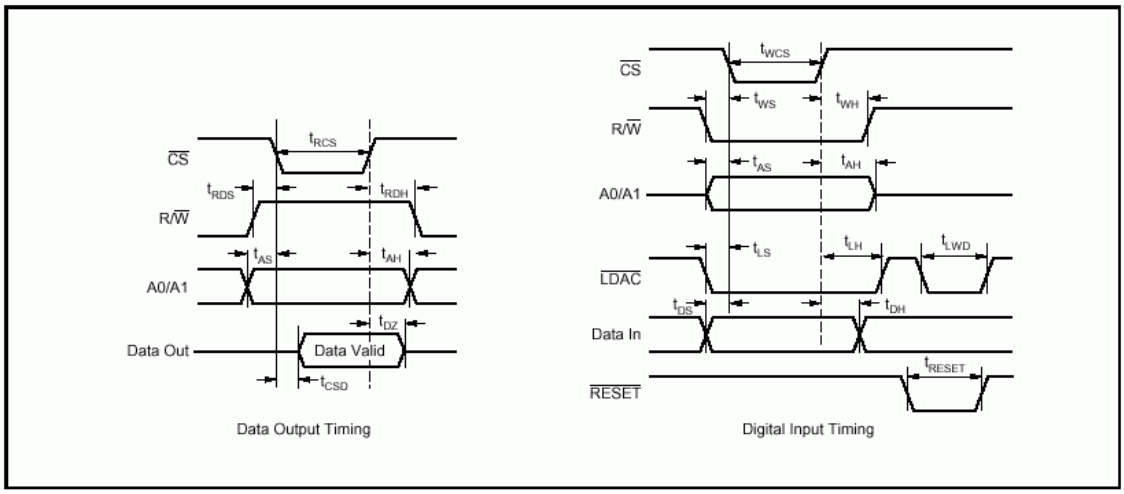


Figura 5.2 - Carta de tempos do conversor D/A DAC7625

Será feita uma análise apenas da carta de tempos da direita (figura 5.2), pois essa representa o disparo de conversão D/A. A carta de tempos da esquerda representa uma leitura do dado fornecido para conversão, ou seja, é possível para o processador confirmar se a informação enviada realmente se encontra nos registradores internos do DAC. Esse mecanismo de segurança não foi implementado no protótipo VS01A.

Os pinos R/W̄, CS\, A1 e A0 são aterrados, fixando a utilização do canal 0 para disparos de conversão. O único pino controlado pelo DSP é o LDAC\, o qual indo a zero faz com que o D/A converta o valor presente no barramento de dados em sua respectiva tensão analógica de saída.

O formato de saída do D/A também é "straight binary", ou seja, vai de 0V (000h) a 2,49939V (FFFh). A tensão de saída é dada por

$$V_{out} = \frac{N \times 2,5}{4096} \qquad (5.2)$$

sendo N a entrada em 12 bits e V_{out} a tensão de saída em volts.

5.1.5 Chip select

O circuito chip-select é o responsável pelo mapeamento dos conversores A/D e D/A no espaço de i/o do processador. Sua função é interfacear os sinais de controle do DSP com os sinais de comando dos conversores. O DSP possui instruções de acesso à memória interna, externa, i/o interno e externo. O chip-select limita o acesso aos conversores no espaço de i/o **externo**, evitando conflitos com os demais periféricos do processador.

A distinção entre a seleção do A/D e do D/A é feita pela saída XF (eXternal Flag) do processador, sendo essa uma sinalização tanto interna (flag em ram) como externa (pino de saída). Quando o sinal XF vai a zero, apenas o circuito de interface do A/D é habilitado, sendo inibido o acesso ao D/A (comando LDAC\), e vice-versa. Os demais sinais de controle advindos do DSP são: IS\ (I/O Space), R/$\bar{W}$ (Read/Write), RD\ (Read) e I/O0 (entrada de retorno do BUSY\).

Devido ao fato de que tanto os sinais de controle de barramento do DSP como os sinais de comando dos conversores serem habilitados em nível "0", o chip-select nada mais é do que um arranjo de lógica "OU" (74LS32), gerando um nível zero na saída desejada apenas quando todas as respectivas entradas forem a zero. Exemplo: o disparo de nova conversão A/D (pino WR\ do conversor = zero) depende da "queda" de três sinais de controle: IS\, R/$\bar{W}$ e XF. Logo, estes três sinais são entradas de uma porta "OU" cuja saída é o sinal WR\ para o conversor A/D.

5.1.6 Fonte de alimentação

A fonte de alimentação do VS01A é do tipo linear regulada, baseada nos integrados reguladores de tensão 7812 e 7805. São geradas então duas tensões para o sistema. A primeira, de 5V, destina-se a todo o sistema lógico, composto dos subsistemas: DSP, A/D, D/A e chip-select. A segunda, de 12V, destina-se às interfaces de entrada e saída, uma vez que os sinais externos podem chegar a 10V. O consumo da fonte de 12V está estimado em 20mA e o da fonte de 5V, em 100mA.

Um ponto importante na distribuição de alimentação para o circuito diz respeito à separação entre as partes analógica e digital. Cada parte deve ter sua conexão de alimentação e terra feita diretamente dos terminais da fonte, evitando a interferência mútua. Caso isso não seja feito, as correntes de chaveamento digital podem gerar espúrios na parte analógica, compromentendo o desempenho dos conversores A/D e D/A, bem como das interfaces de entrada e saída.

Especificamente em relação à parte analógica, é fundamental que qualquer ponto de terra tenha sua ligação feita diretamente do terminal terra da fonte. Isso evita a formação de loops de terra, os quais geram potenciais superiores a 0V, efeito que compromete o desempenho do sistema.

5.2 Projeto de software

5.2.1 Descrição da ferramenta de software para programação do DSP

A programação do DSP TMS320F206 foi realizada totalmente em assembly, ou seja, utilizando um compilador assembler, o qual efetua a tradução do código-fonte com os mnemônicos para o arquivo com a sequência de opcodes a serem executados (TEXAS INSTRUMENTS, 1997). A ferramenta de desenvolvimento utilizada não é IDE - Integrated

Development Environment - ou seja, o editor, o compilador e o simulador / emulador são pacotes separados.

Para a edição do código-fonte foi utilizado o software MultEdit, mas qualquer editor de texto poderia ser utilizado. O compilador utilizado foi o Tasm (WHITE MOUNTAIN DSP, 1997), em ambiente DOS. A ferramenta de depuração e gravação do firmware em memória ram chama-se Code Explorer (WHITE MOUNTAIN DSP, 1997). Este software exige para uso interno determinadas regiões de memória ram e de programa, para a depuração em tempo real do software que está sendo executado no DSP. Não há como simular apenas no computador o software. A depuração é feita pelo canal serial assíncrono, ligado via cabo DB-9 à porta serial do PC.

Os softwares escritos para o DSP foram testados inicialmente em memória ram. Após validados, foram geradas versões para execução na memória flash do processador. A ferramenta utilizada para a gravação do código na memória flash foi o software Serial Flash Loader (WHITE MOUNTAIN DSP, 1997).

5.2.2 Análise dos programas iniciais para testes do hardware

Antes de se iniciar a programação do algoritmo "Regras Limitadas" foi necessário testar e validar os subsistemas de conversão A/D e D/A. Para tanto, foram desenvolvidos softwares para conversões A/D e D/A isoladas. Em seguida, foi testado um software que, ao efetuar uma conversão A/D, transfere o valor convertido para o D/A, resultando em tensão de saída igual à entrada.

O "range" utilizado para o processo de neutralização de pH foi 0 a 10V, tanto para a entrada - vazão de base q_3 - como para a saída - pH estimado. O software de teste permitiu validar o desempenho dos conversores em toda a faixa de operação.

A placa de aquisição CAD12/36, responsável pela exteriorização dos sinais da planta em Simulink, passou por teste semelhante, de 0 a 10V. Além disso, teve a sua calibração de "zero" e "fundo de escala" revisada, tanto para o A/D canal 0 como para o D/A canal 0 (LYNX

TECNOLOGIA ELETRÔNICA LTDA, 1993).

5.2.3 Programação do algoritmo Regras Limitadas

O fluxograma descrito na figura 4.1 apresenta a sequência necessária para a implementação do sensor virtual no firmware. Porém, apesar de ser o mais importante passo, esse é apenas o ponto de vista teórico e genérico do projeto de software. A partir de agora, diversas outras análises devem ser feitas para se chegar a um código-fonte efetivo. A primeira delas consiste em se avaliar a faixa de valores das variáveis de interesse.

A vazão de base q_3 apresenta, segundo a planta, uma faixa de 0 a 1 dm^3/min. Já o *pH* possui uma faixa de 3 a 10. Os coeficientes de centros de clusters apresentam faixas similares de valores, pois são compostos de regressores de q_3 e do *pH* estimado. Já os coeficientes dos submodelos lineares têm faixa de -1 a +2. Os valores intermediários de todas essas variáveis devem apresentar pelo menos 4 casas decimais.

O processador em questão é uma máquina de ponto fixo, com barramento de dados de 16 bits. Sua arquitetura permite trabalhar em formato inteiro não-sinalizado (0 a 65535), inteiro sinalizado (complemento de 2: -32768 a 32767), e formatos fracionários, como o Q15. O formato Q15, ou 1.15, codifica o número de 16 bits com 1 casa inteira (sinal, no caso), e 15 casas decimais. Assim, a faixa suportada pelo Q15 vai de -1 (8000h) a +0,9999695 (7FFFh).

Veja que essa faixa do Q15 é suficiente para representar q_3, mas não representa o *pH* nem os coeficientes dos submodelos. Uma saída possível é utilizar formatos fracionários com mais casas inteiras, como 2.14, 3.13, e assim por diante. A desvantagem é a perda de resolução, pois transferem-se bits da parte fracionária para a parte inteira.

Um outra possibilidade é o escalonamento de variáveis, para que estas se enquadrem na faixa do formato Q15. Essa técnica é conhecida como "Scaling", e muito utilizada nos sistemas em ponto fixo. No caso em questão, todas as variáveis são divididas por 10, sendo então enquadradas no formato Q15. Observe que não há como normalizar apenas as variáveis que não se encaixam no Q15, pois isso causa uma alteração no modelo. A normalização deve ser homogênea, para que

se mantenha a coerência com o modelo original.

Se faz necessária uma análise mais aprofundada do processo de scaling do modelo. Em relação aos graus de ativação das regras (antecedentes), as distâncias do vetor de entradas

$$Z = \begin{bmatrix} q3(k-1) \\ q3(k-2) \\ q3(k-3) \\ q3(k-4) \\ pH(k-1) \\ pH(k-2) \end{bmatrix}$$

aos centros vetoriais de clusters serão diminuídas de um fator 10 e, por conseguinte, as normas quadráticas, de um fator 100. Porém, pela equação (3.11), vê-se que um grau de ativação é composto do inverso da soma das **relações** entre as normas quadráticas em questão, ou seja, o grau de ativação resultará no mesmo valor, com ou sem a normalização. Por enquanto, não está sendo levado em conta o prejuízo da perda de resolução decorrente da normalização.

Para a saída global $y = pH(k)$ tem-se

$$y = y_1 q_1 + y_2 q_2 + \ldots + y_c q_c \tag{5.3}$$

sendo y_i as saídas parciais e q_i os graus de ativação das regras.

Deseja-se obter $y = \frac{y}{10}$, ou seja, um pH normalizado. Assim, tem-se

$$\frac{y}{10} = \frac{1}{10}(y_1 q_1 + y_2 q_2 + \ldots + y_c q_c) = \frac{y_1}{10} q_1 + \frac{y_2}{10} q_2 + \ldots + \frac{y_c}{10} q_c \tag{5.4}$$

Logo, as saídas parciais devem ser normalizadas. Analisando uma saída parcial genérica tem-se

$$y_i = b_{i0} + b_{i1} q_3(k-1) + b_{i2} q_3(k-2) + b_{i3} q_3(k-3) +$$

$$+ b_{i4}q_3(k-4) + b_{i5}pH(k-1) + b_{i6}pH(k-2) \tag{5.5}$$

$\therefore$

$$\frac{y_i}{10} = \frac{1}{10}[b_{i0} + b_{i1}q_3(k-1) + b_{i2}q_3(k-2) + b_{i3}q_3(k-3) + b_{i4}q_3(k-4) + b_{i5}pH(k-1) + b_{i6}pH(k-2)] \tag{5.6}$$

$\therefore$

$$\frac{y_i}{10} = \frac{b_{i0}}{10} + \frac{[b_{i1}q_3(k-1)]}{10} + \frac{[b_{i2}q_3(k-2)]}{10} + \frac{[b_{i3}q_3(k-3)]}{10} + \frac{[b_{i4}q_3(k-4)]}{10} + \frac{[b_{i5}pH(k-1)]}{10} + \frac{[b_{i6}pH(k-2)]}{10} \tag{5.7}$$

Vê-se então que pode ser feita a normalização ou dos coeficientes dos submodelos lineares ou do vetor de entradas *Z*. O termo independente b_{i0} de cada submodelo linear tem a sua normalização obrigatória. Como o *pH* está fora da faixa do formato Q15, optou-se pela normalização do vetor *Z*. Entretanto, surge um problema: existem alguns coeficientes b_{i1} que são maiores do que 1. Assim, todos os coeficientes b_{i1} serão normalizados. Isso faz com que o termo $q_3(k-1)$ não seja normalizado para o cálculo das saídas parciais, mas apenas para o cálculo dos graus de ativação.

Uma vez acertado o scaling do modelo, analisa-se novamente a variável de entrada q_3. A faixa de tensão fornecida pela placa CAD12/36 na exteriorização de q_3 será 0 a 10V. Logo, a faixa convertida pelo A/D do VS01A será 0 a 4095. Como o formato interno ao DSP deve ser Q.15, ou seja, de -32768 a +32767, torna-se necessária a multiplicação do valor lido do A/D por 8. Observe o erro decorrente da quantização 12 bits para 16 bits (multiplicação por 8). Enquanto o fundo de escala real deve ser +32767, o máximo valor atingido na pós-quantização será $4095 \times 8 = 32760$.

Para a variável de saída *pH* estimado, a sua representação interna normalizada vai de 0 a 1, ou seja, 0 a +32767. Assim, dividindo seu valor por 8, obtém-se 0 a 4095. Essa faixa representa em tensão de saída 0 a 10V. Nesse caso, a correspondência entre unidades de pH e Volts é direta.

Toda a análise descrita anteriormente destina-se a uma implementação do modelo em ponto fixo. Porém, durante o desenvolvimento e testes preliminares, observou-se que, mesmo utilizando técnicas de scaling, os graus de ativação das regras apresentavam uma faixa dinâmica muito grande, sendo a sua representação em ponto fixo insuficiente. Dessa forma, a parte do modelo

referente ao cálculo dos graus de ativação foi adaptada para operações em ponto flutuante. Evidentemente, essas operações são tratadas apenas em níveis de software (CROWELL, 1989), já que a arquitetura do processador não suporta ponto flutuante em hardware. As consequências da utilização de rotinas de ponto flutuante são benéficas, pois o problema da baixa faixa dinâmica (ponto fixo) é solucionado, porém há desvantagens, como o aumento do tempo de processamento.

O firmware em questão, descrito no texto como VS01A - ponto fixo, é na verdade híbrido, pois apresenta partes de processamento em ponto fixo e partes em ponto flutuante.

Na tentativa de melhorar o desempenho do firmware, foi desenvolvido um segundo software, totalmente em ponto flutuante. Isso elimina o scaling de variáveis, mas aumenta o tempo de processamento do modelo. A planta de neutralização de pH apresenta um intervalo de amostragem de 1/6 de minuto, cujo valor para fins de simulação foi fixado no "benchmark" em 1/6 de segundo = 167 ms. O tempo total para o processamento do modelo no firmware VS01A - ponto fixo foi de 9,8ms. Já o modelo VS01A - ponto flutuante tem um tempo de processamento de 13ms. Observa-se que, mesmo o modelo integral em ponto flutuante está quase 13 vezes mais rápido que o necessário. Para uma correta operação do VS01A junto ao processo, via placa CAD12/36, sua taxa de interrupção foi configurada exatamente em 1/6 de segundo.

5.2.4 Implementação do controlador PID

A idéia em torno da realização do controlador PID integrado ao modelo de sensor virtual foi prover uma solução de controle mais completa ao processo de destino. No caso do VS01A, foi utilizado inicialmente um valor de referência fixo de pH = 7. Posteriormente, foram geradas alterações em degrau automáticas ao longo do tempo, simulando mudanças bruscas por parte do operador.

O algoritmo PID (ÅSTRÖM; WITTENMARK, 1997; LEONARDI, 2000) implementado no protótipo VS01A é dado por

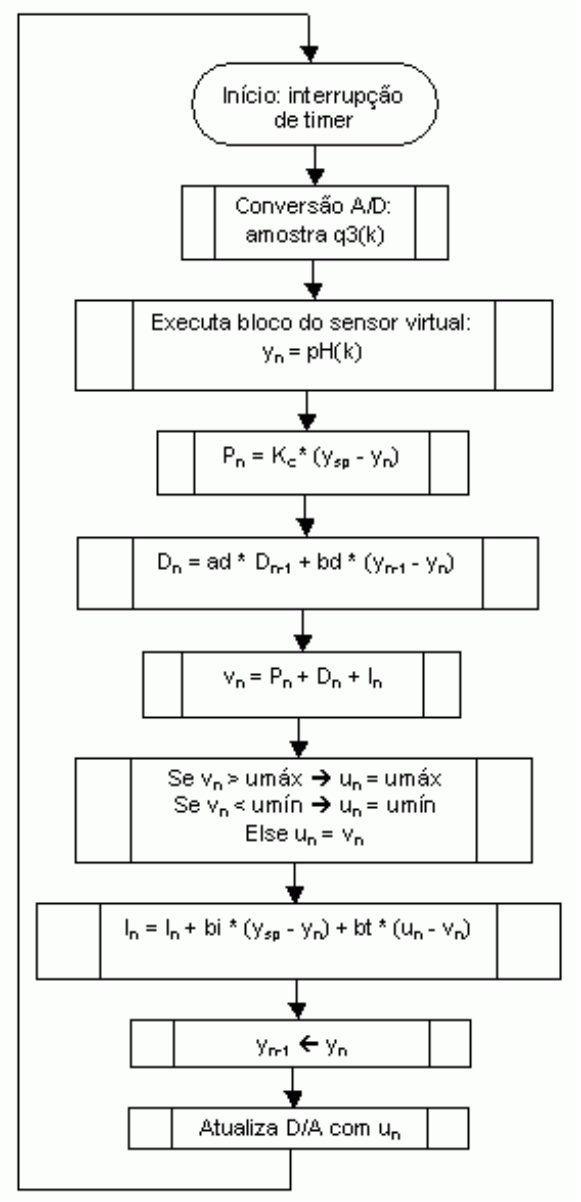


Figura 5.3 - Fluxograma do conjunto sensor virtual + PID

onde P_n, D_n e I_n são as contribuições dos modos proporcional, derivativo e integral, respectivamente, v_n é o esforço de controle sem saturação e u_n é o esforço de controle na saída, com saturação.

Os demais parâmetros envovidos são

$$ad = \frac{T_d}{NT + T_d} \tag{5.8}$$

$$bd = \frac{K_c N T_d}{NT + T_d} \tag{5.9}$$

$$bi = \frac{K_c T}{T_i} \tag{5.10}$$

$$bt = \frac{T}{T_t} \tag{5.11}$$

sendo T_d o tempo derivativo, N o ganho máximo do modo derivativo em altas frequências, T o intervalo de amostragem, K_c o ganho proporcional, T_i o tempo integral e T_t o tempo de trilhamento.

O controlador em questão incorpora o recurso de anti reset wind-up. Na inicialização, são zeradas as variáveis y_{n-1}, D_{n-1} e I_n.

6 RESULTADOS

As simulações realizadas podem ser divididas em três partes. A primeira consistiu na validação do sensor virtual obtido. Foram realizados testes em malha aberta e em malha fechada, com realimentação pelo sensor "real". O nível de ruído branco gaussiano adicionado à vazão de base q_3 foi "alto", mesmo nível utilizado no treinamento do sensor. O sinal de q_3 bastante ruidoso é útil para identificação (GARCIA, 2001), mas não é realista do ponto de vista de controle. Assim, foram incluídas diversas simulações com "baixo" ruído em q_3.

A segunda parte dos testes consistiu no fechamento da malha de controle pelo sensor virtual, ou seja, este atuando efetivamente como sensor de pH do processo. Para este caso, foi configurada uma vazão q_3 com "baixo" ruído. Todas as simulações da primeira e segunda partes apresentam resultados gráficos para o modelo em PC e para o VS01A - ponto flutuante. As tabelas comparativas apresentam ainda resultados para o VS01A - ponto fixo.

A última parte das simulações caracterizou-se por um teste adicional do conjunto sensor virtual + controlador PID em hardware.

6.1 Validação do sensor virtual obtido

6.1.1 Validação em malha aberta

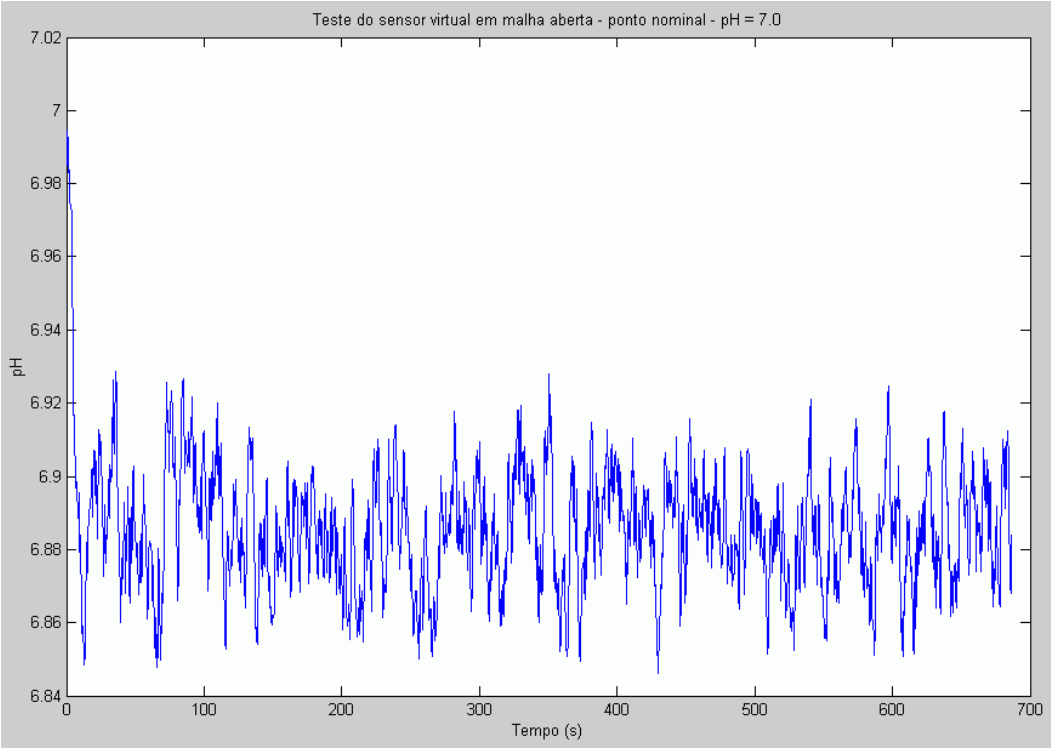


Figura 6.1 - Validação em malha aberta - modelo em PC

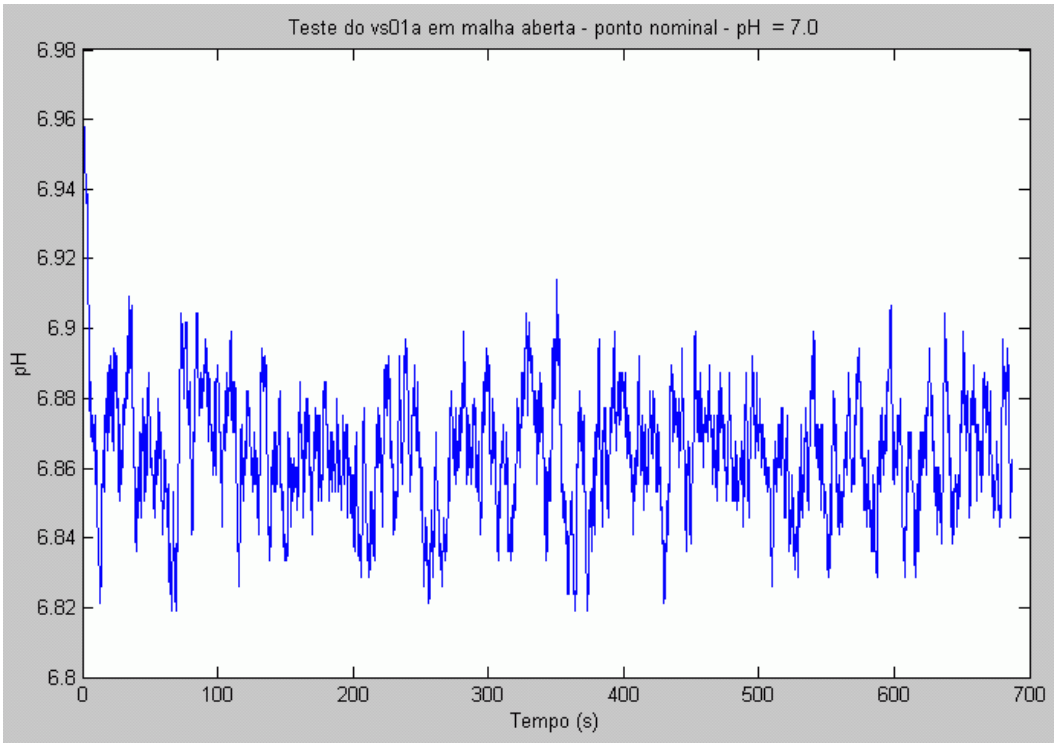


Figura 6.2 - Validação em malha aberta - VS01A

6.1.2 Comentários sobre os testes em malha aberta

Percebe-se pelas figuras 6.1 e 6.2 que ambos os modelos, em PC e no VS01A, apresentam um erro estacionário em relação ao valor desejado. O modelo em PC apresentou um valor médio de 6,88, enquanto o VS01A teve um valor médio de 6,86. O erro percentual, neste caso, foi de 2%, contra 1,71% do modelo em PC.

Uma possível justificativa para a presença desses erros foi o fato da etapa de treinamento do modelo em PC utilizar um valor de referência de pH com variação do tipo rampa, e não mantê-lo fixo em pH = 7 por um certo tempo de simulação. Assim, provavelmente a coleta de dados não concentrou a devida quantidade de pontos nessa região, estando bem abaixo da necessária para uma maior precisão na faixa de pH neutro.

6.1.3 Validação em malha fechada - realimentação pelo sensor "real"

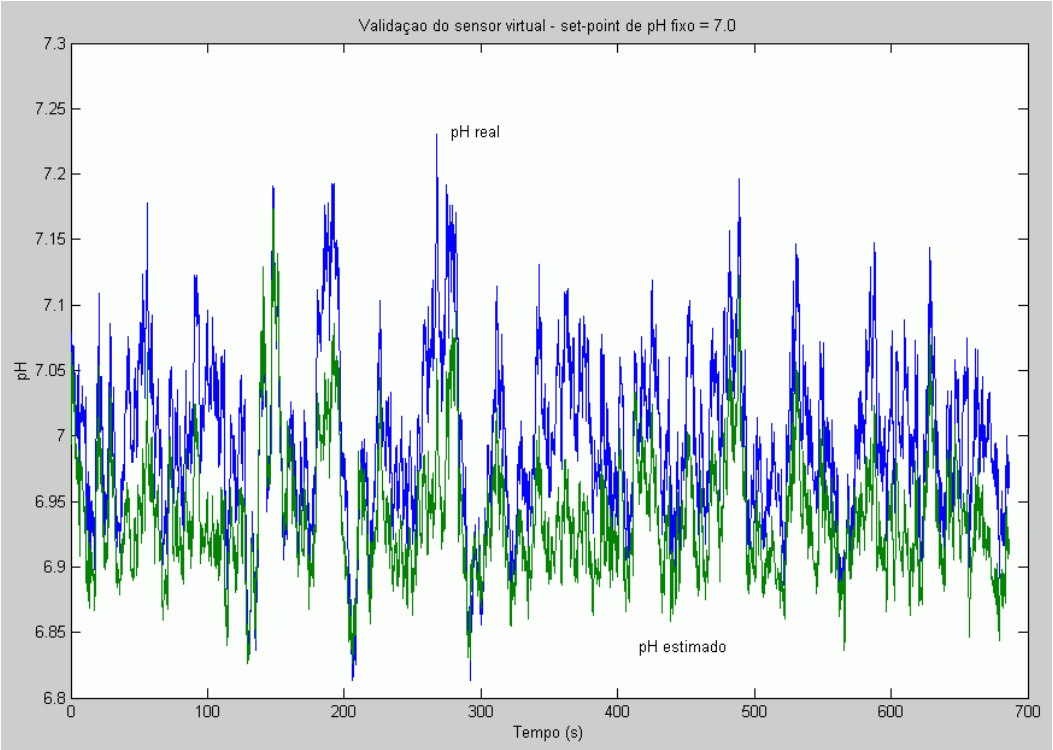


Figura 6.3 - Validação do modelo em PC - valor de referência fixo

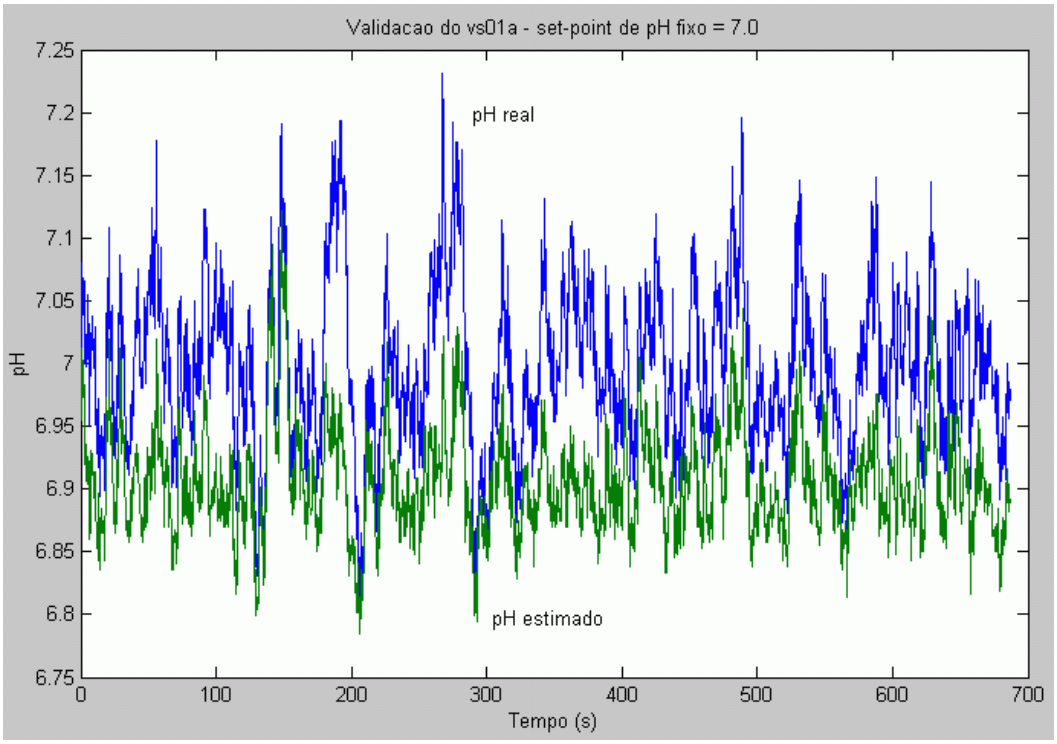


Figura 6.4 - Validação do VS01A - valor de referência fixo

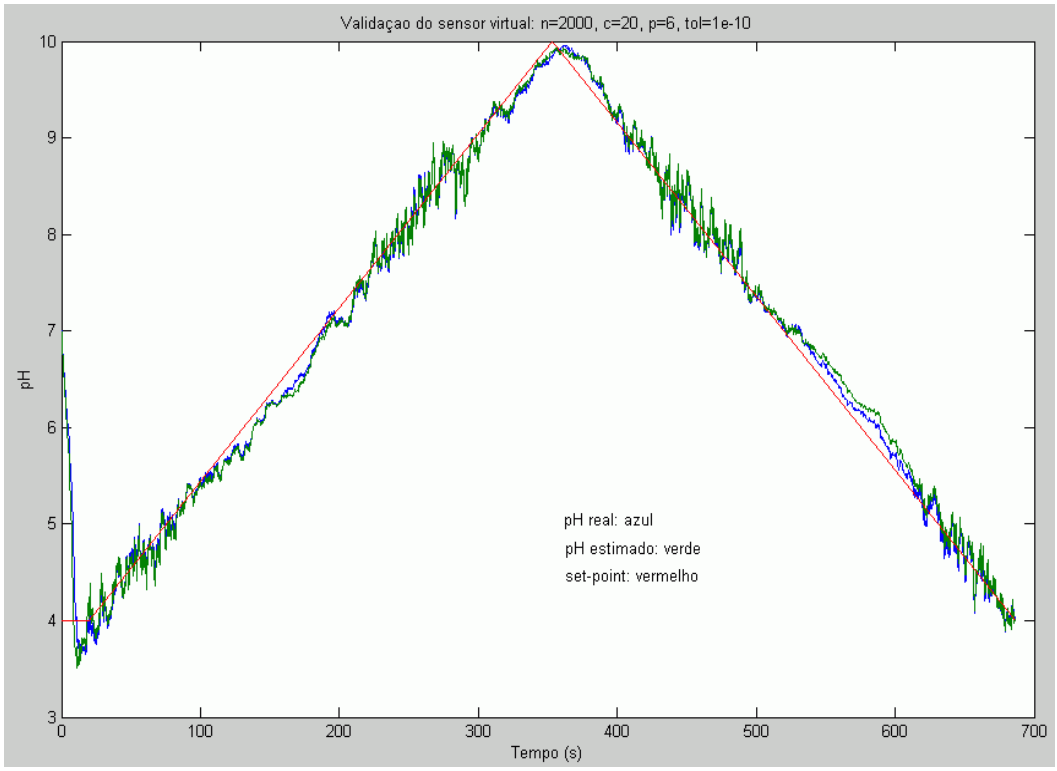


Figura 6.5 - Validação do modelo em PC - valor de referência em rampa

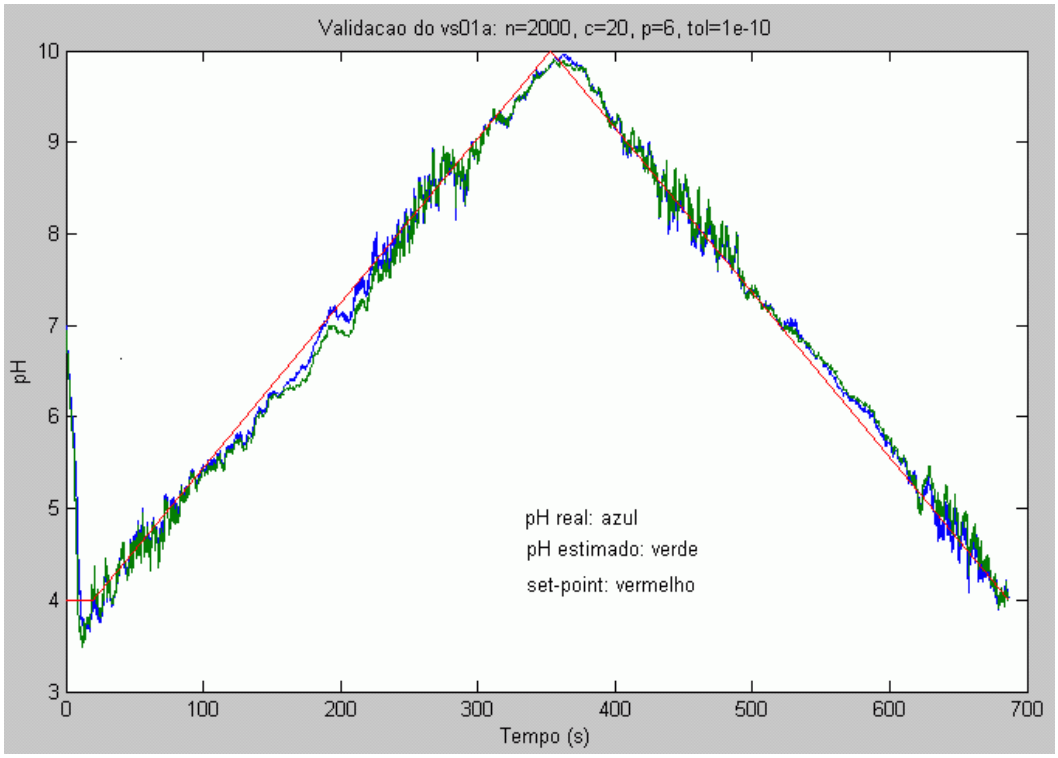


Figura 6.6 - Validação do VS01A - valor de referência em rampa

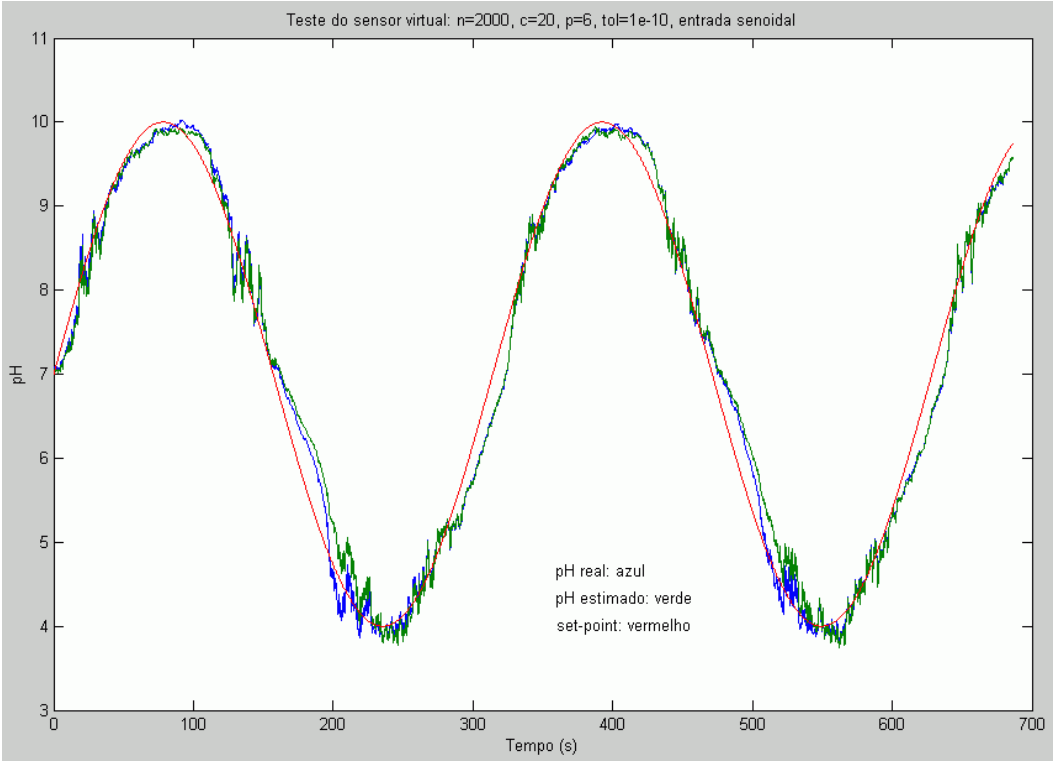


Figura 6.7 - Validação do modelo em PC - valor de referência senoidal

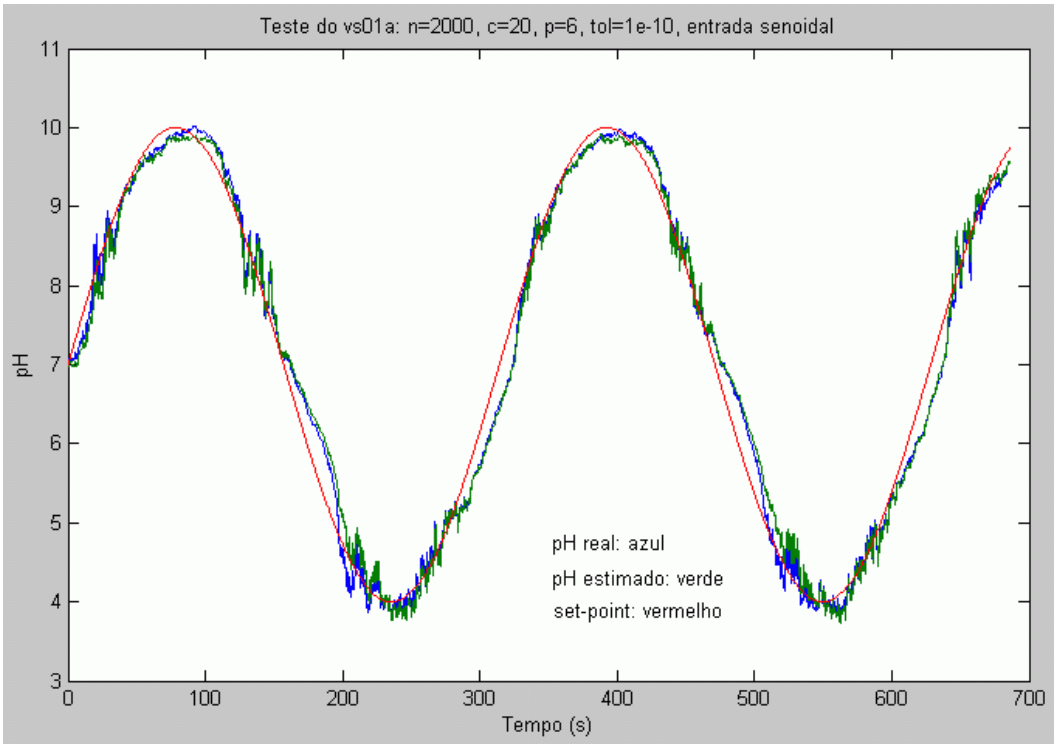


Figura 6.8 - Validação do VS01A - valor de referência senoidal

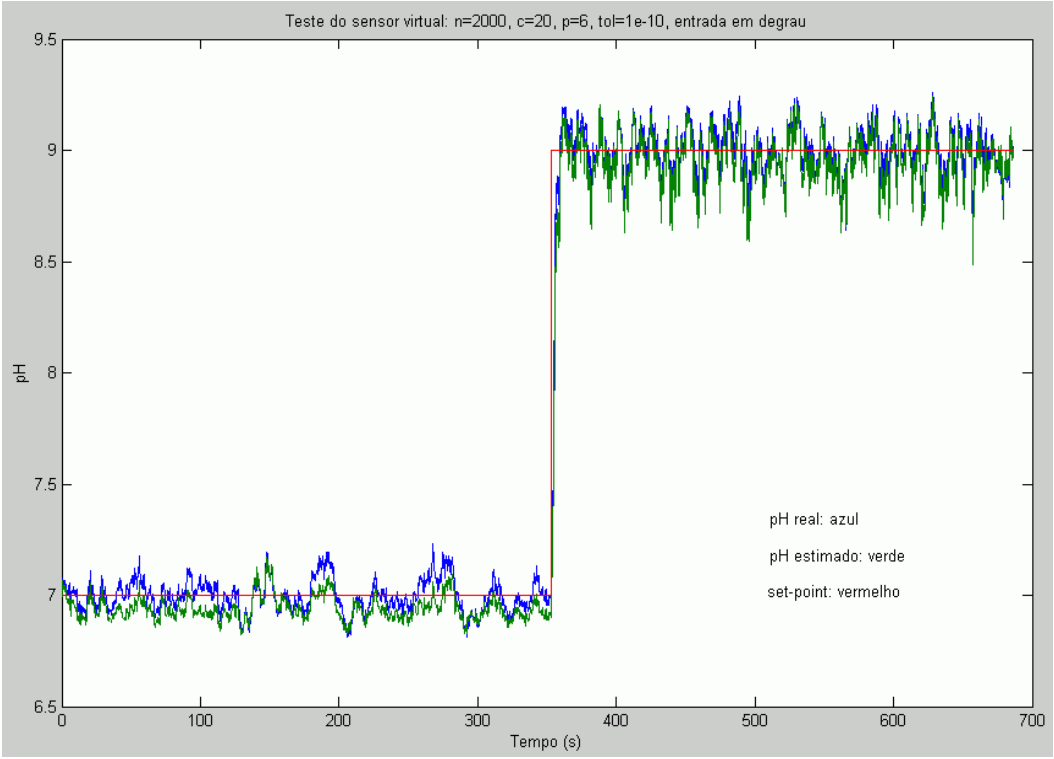


Figura 6.9 - Validação do modelo em PC - valor de referência em degrau 7 a 9

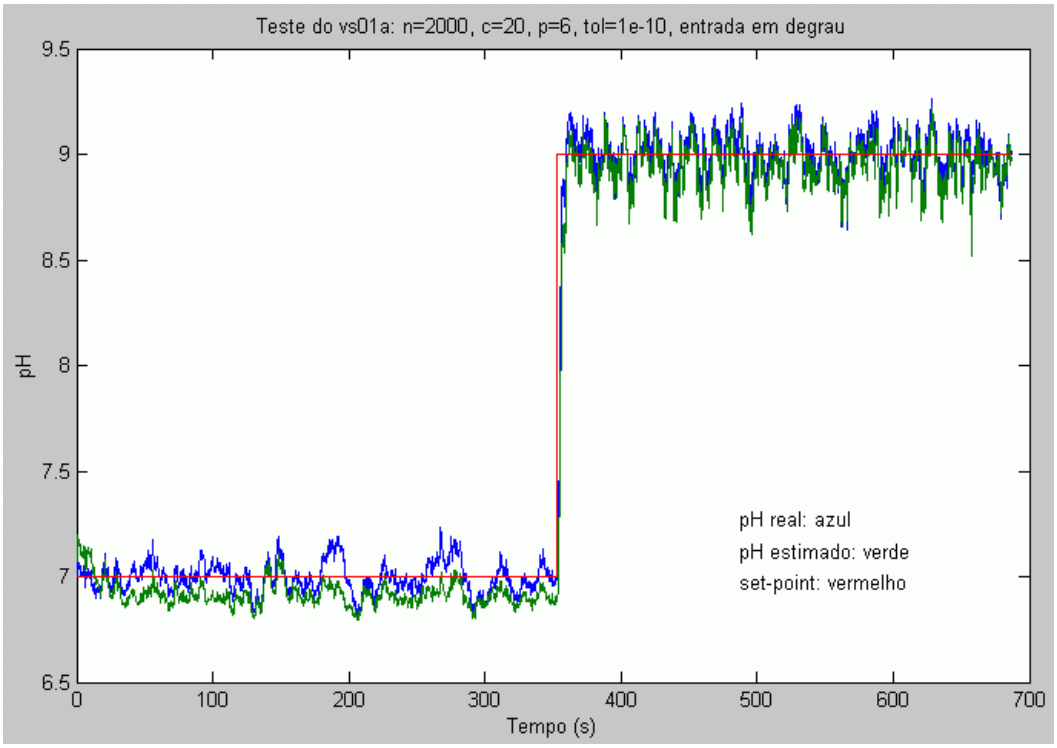


Figura 6.10 - Validação do VS01A - valor de referência em degrau 7 a 9

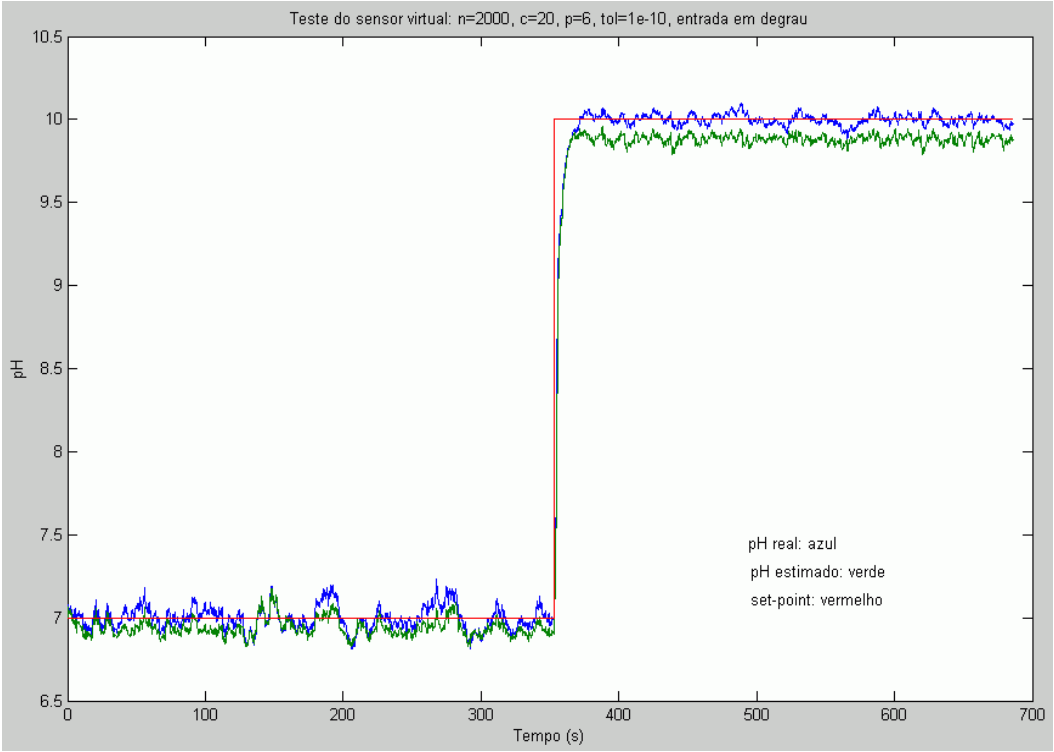


Figura 6.11 - Validação do modelo em PC - valor de referência em degrau 7 a 10

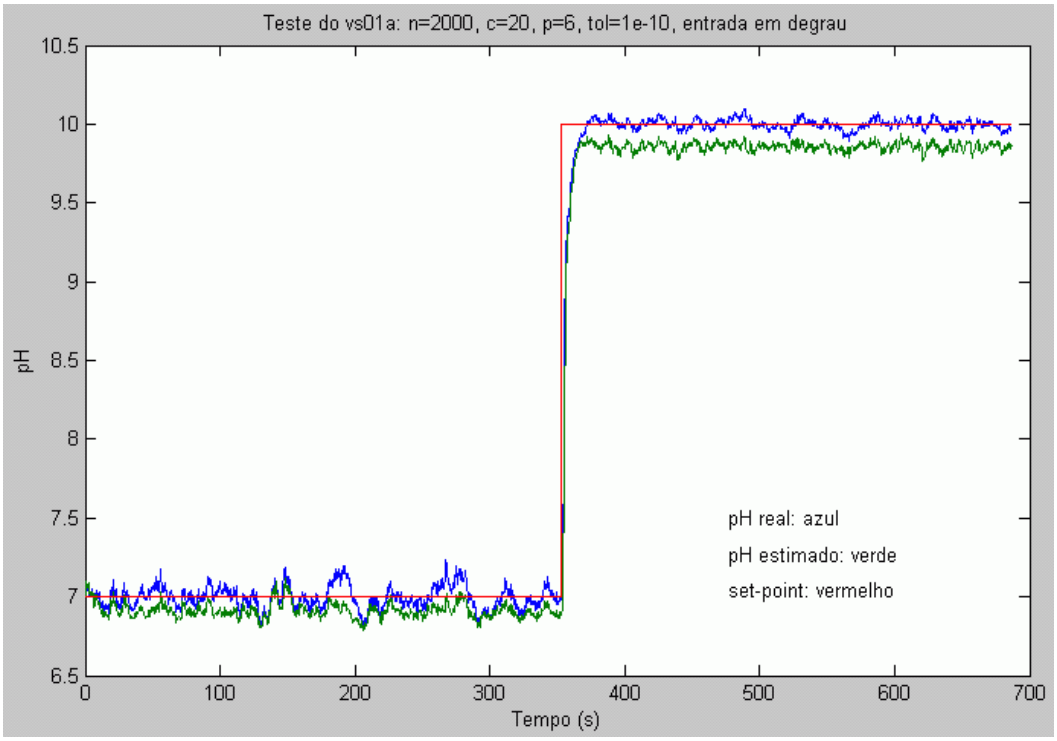


Figura 6.12 - Validação do VS01A - valor de referência em degrau 7 a 10

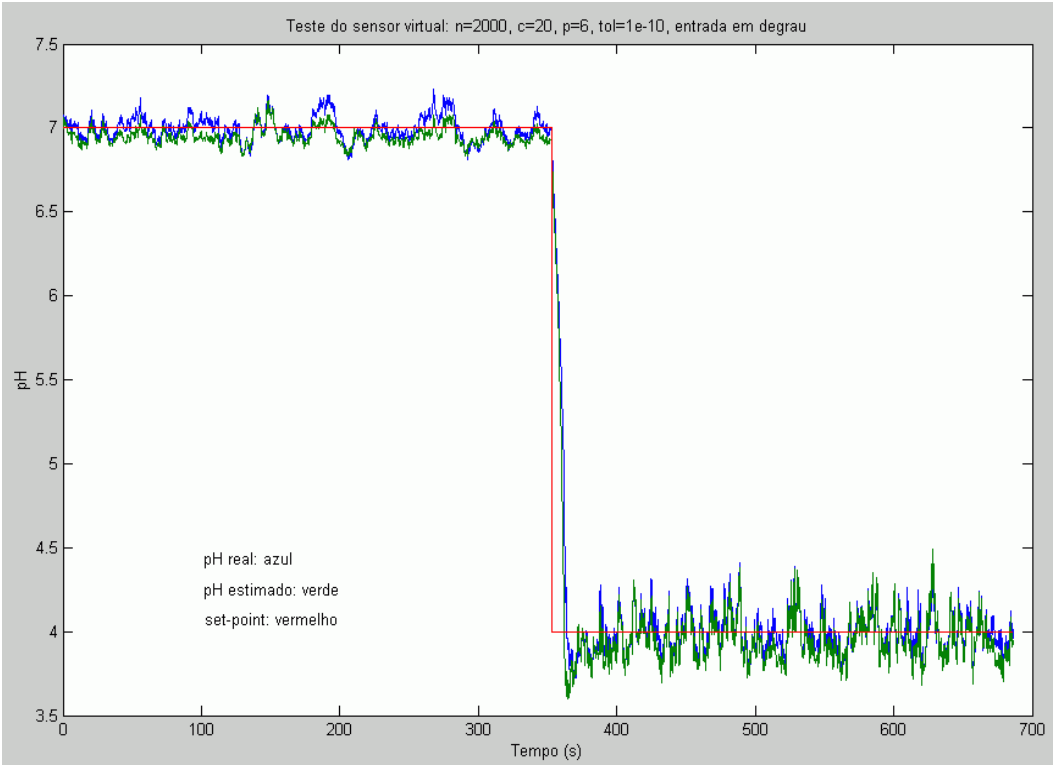


Figura 6.13 - Validação do modelo em PC - valor de referência em degrau 7 a 4

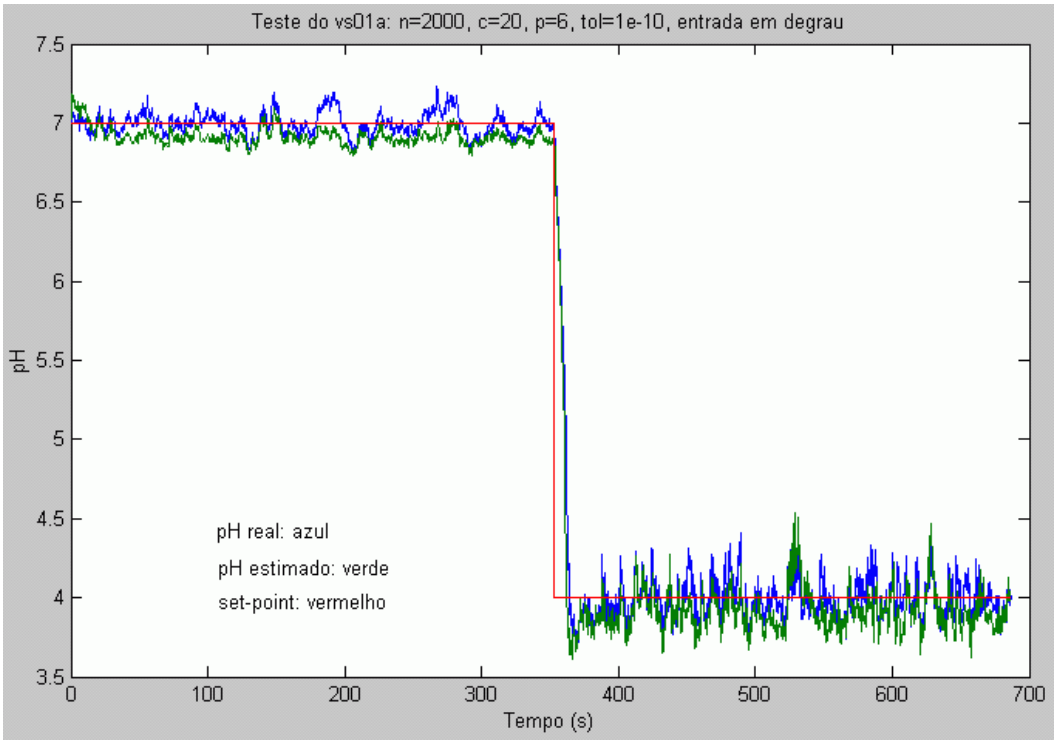


Figura 6.14 - Validação do VS01A - valor de referência em degrau 7 a 4

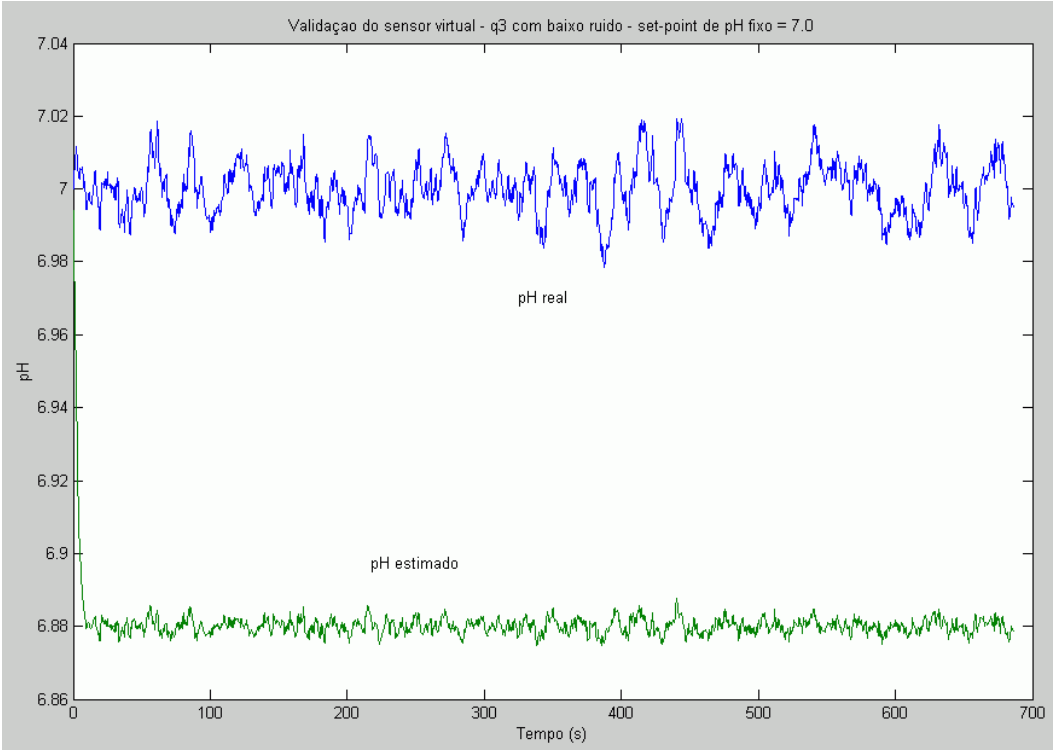


Figura 6.15 - Validação do modelo em PC - q_3 com baixo ruído - valor de referência fixo

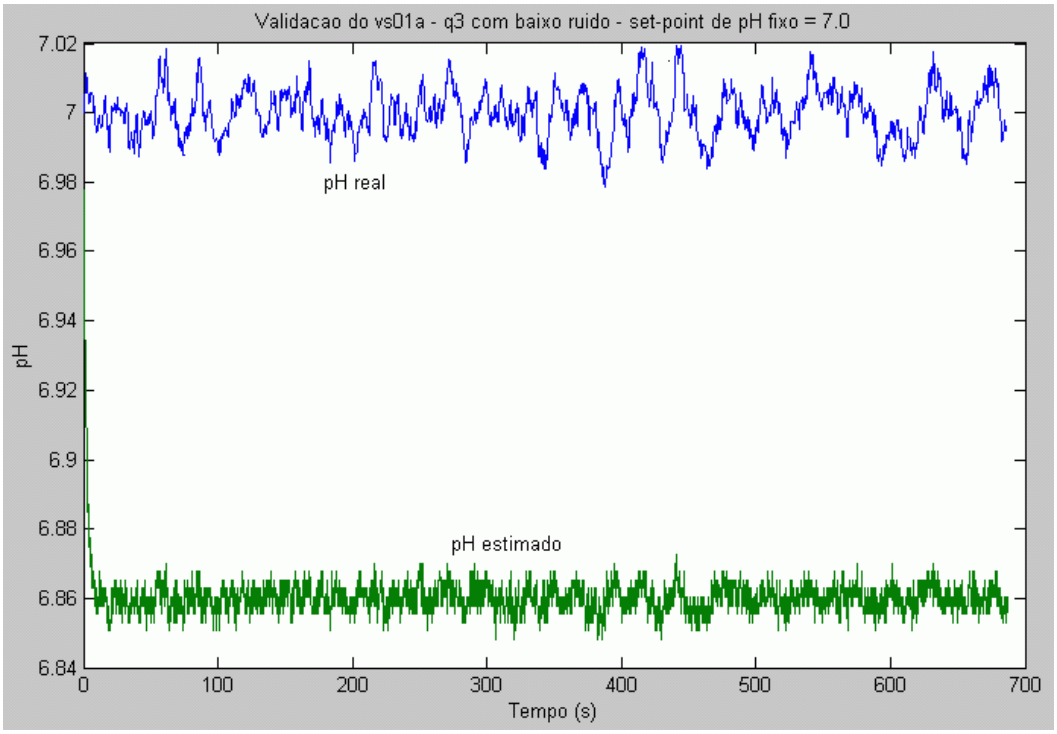


Figura 6.16 - Validação do VS01A - q_3 com baixo ruído - valor de referência fixo

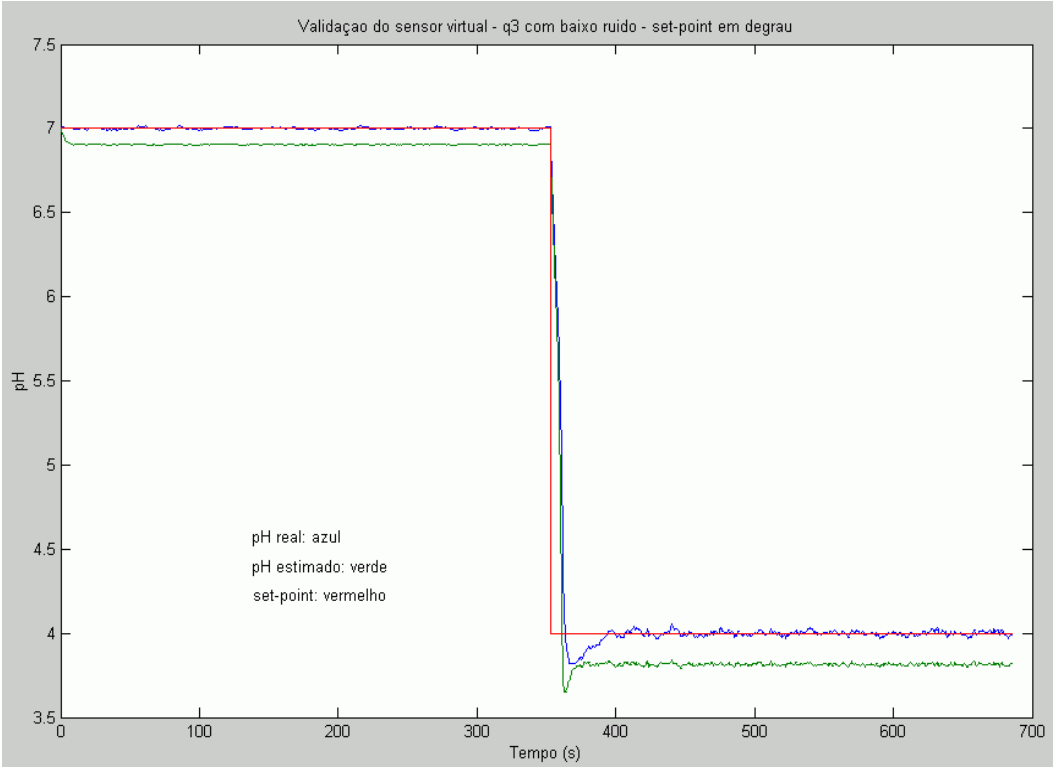


Figura 6.17 - Validação do modelo em PC - q_3 com baixo ruído - valor de referência em degrau 7 a 4

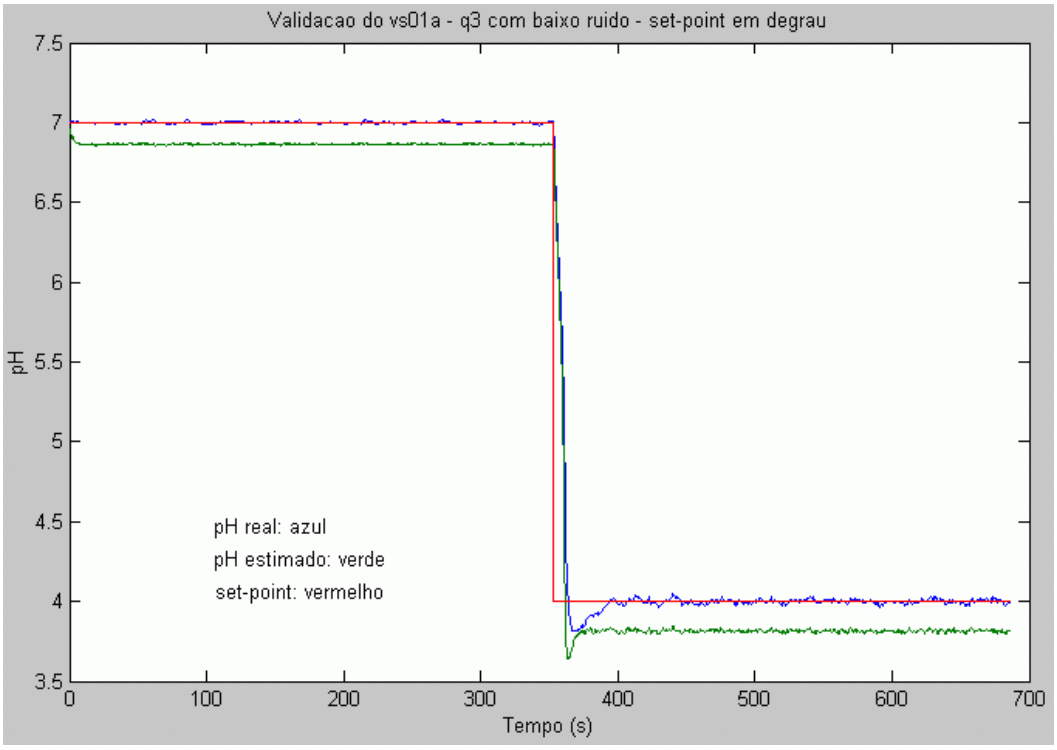


Figura 6.18 - Validação do VS01A - q_3 com baixo ruído - valor de referência em degrau 7 a 4

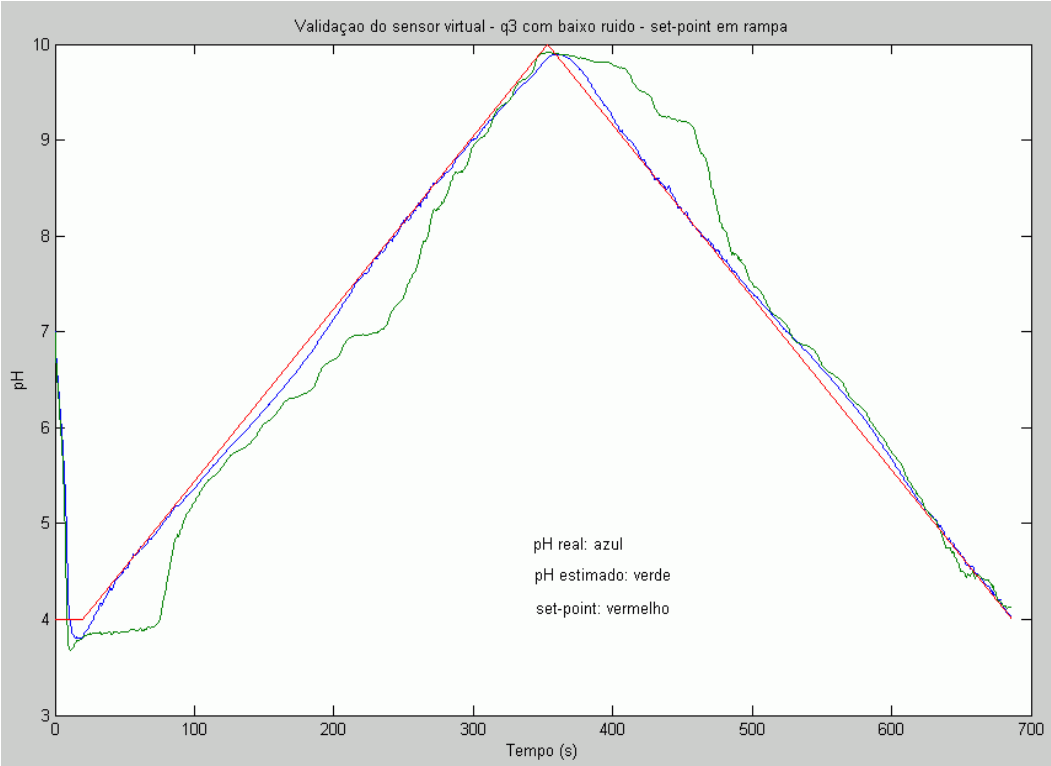


Figura 6.19 - Validação do modelo em PC - q_3 com baixo ruído - valor de referência em rampa

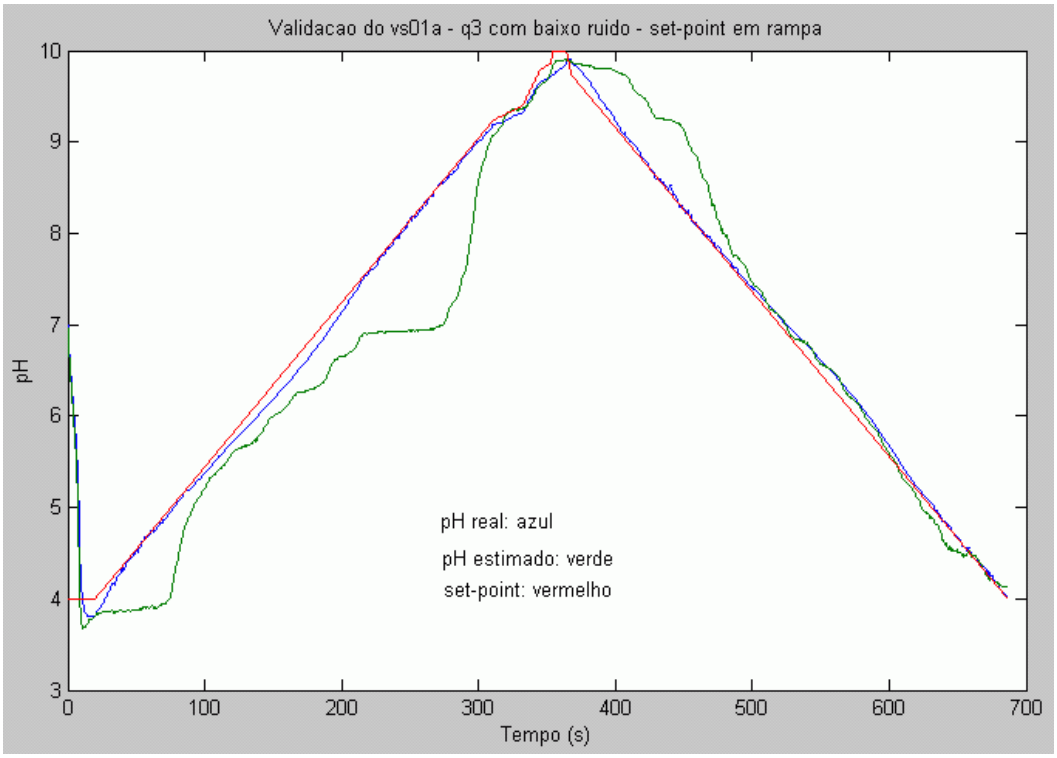


Figura 6.20 - Validação do VS01A - q_3 com baixo ruído - valor de referência em rampa

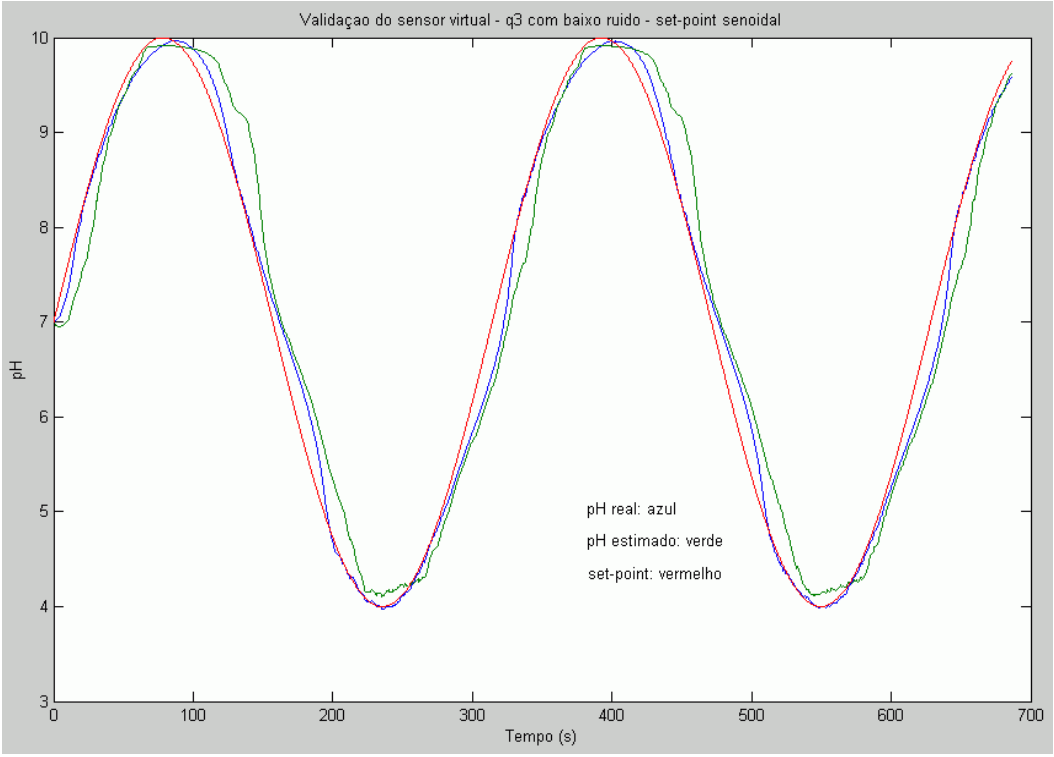


Figura 6.21 - Validação do modelo em PC - q_3 com baixo ruído - valor de referência senoidal

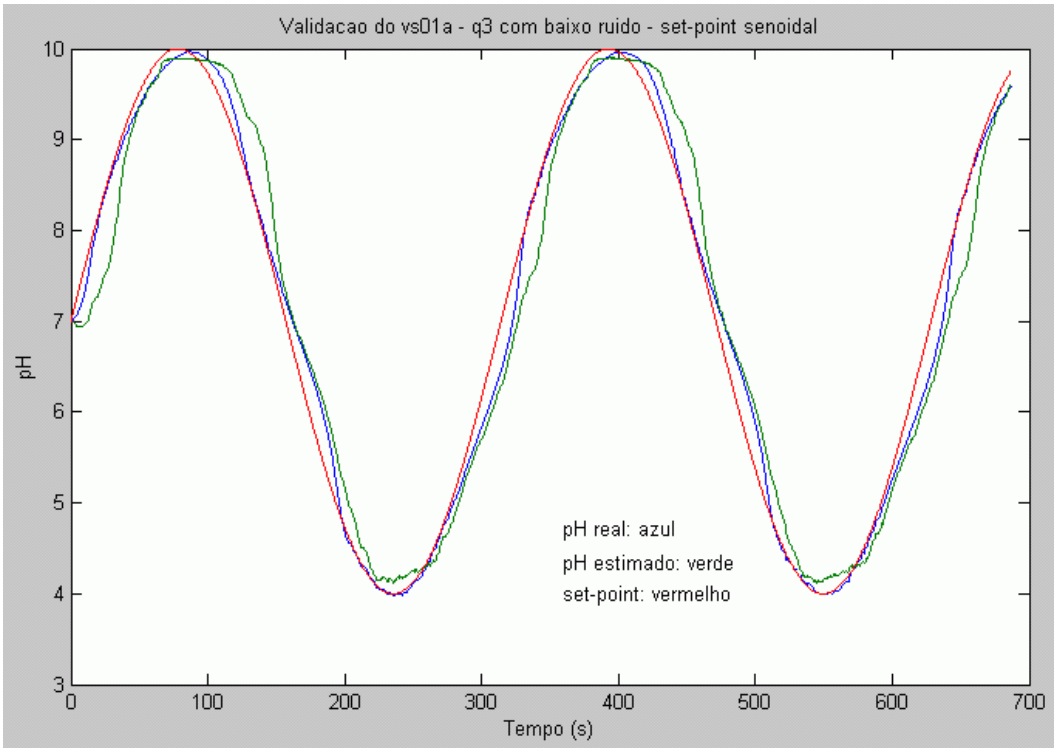


Figura 6.22 - Validação do VS01A - q_3 com baixo ruído - valor de referência senoidal

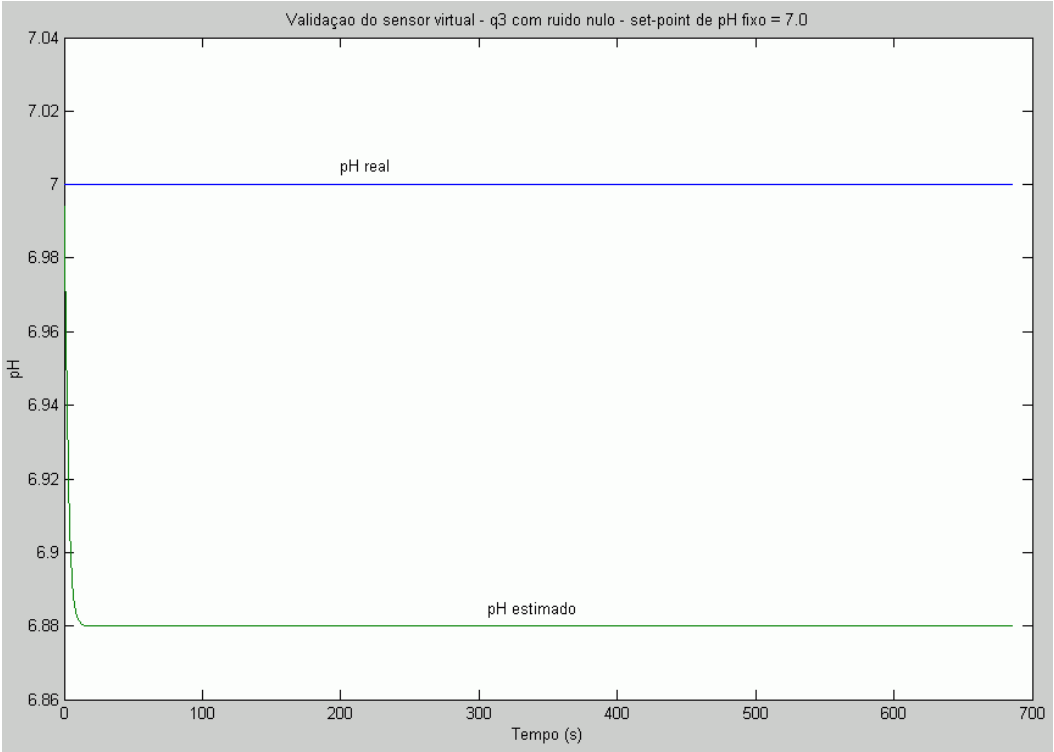


Figura 6.23 - Validação do modelo em PC - q_3 com ruído nulo - valor de referência fixo

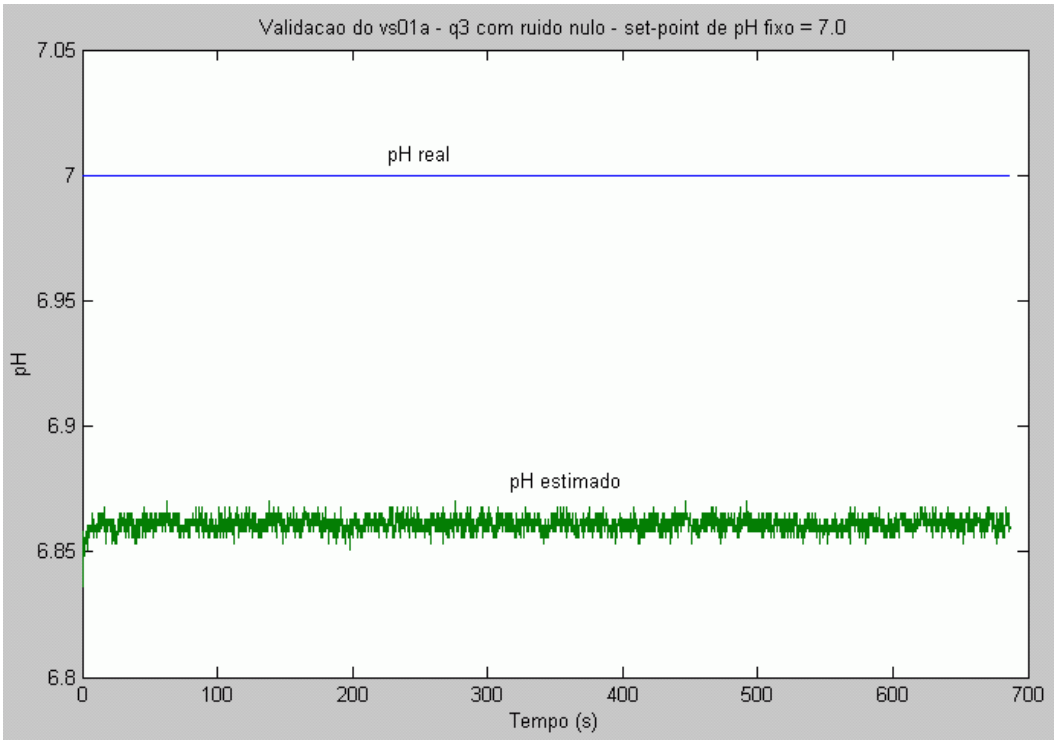


Figura 6.24 - Validação do VS01A - q_3 com ruído nulo - valor de referência fixo

6.1.4 Tabelas comparativas e comentários sobre os testes em malha fechada com realimentação pelo sensor "real"

Tabela 1 - ISE para q_3 com alto ruído - nível de identificação

Tipo de simulação	Matlab	VS01A - ponto fixo	VS01A - ponto flutuante
Valor de referência fixo - pH = 7	20,23	64,41	41,80
Valor de referência em rampa	22,45	55,26	41,50
Valor de referência senoidal	79,32	111,50	73,60
Valor de referência em degrau 7 –> 9	15,89	55,82	63,79
Valor de referência em degrau 7 –> 10	38,21	78,54	65,80
Valor de referência em degrau 7 –> 4	26,78	73,16	43,89

Tabela 2 - ISE para q_3 com baixo ruído - nível de controle

Tipo de simulação	Matlab	VS01A - ponto fixo	VS01A - ponto flutuante
Valor de referência fixo - pH = 7	59,06	93,05	80,6
Valor de referência em rampa	843,4	1442	543,5
Valor de referência senoidal	461,2	486,2	450,7
Valor de referência em degrau 7 –> 4	98,68	122,6	113,4

Observa-se que os índices ISE da tabela 2 aumentaram consideravelmente em relação aos da tabela 1. Isso porque o modelo identificado responde bem ao mesmo tipo de perturbação com a qual ele foi gerado. Neste caso, treinou-se o modelo com um nível de perturbação alto, e a tabela 2 refere-se a testes com baixo nível de perturbação.

Os índices de desempenho ISE do VS01A em geral foram maiores que os equivalentes do modelo em Matlab. Este aumento pode ser justificado pelas fontes de erro descritas abaixo:

- Quantização 12 para 16 bits na conversão A/D;
- Tolerância e “drift” de componentes eletrônicos, inclusive no circuito de tensão de

referência do A/D e D/A; e

- Placa em caráter de protótipo, sem lay-out de circuito impresso, desprovida de técnicas adequadas para instrumentação.

Conforme já visto, o valor convertido pelo A/D é multiplicado por 8, devido à representação em Q15. Assim, no fundo de escala (+32767), tem-se um erro de 7 unidades de A/D (4095 x 8 = 32760) para o VS01A e o mesmo valor para a placa CAD12/36. Logo, para os 4020 pontos de simulação, poder-se-ia ter, apenas pela quantização, um ISE máximo = 137,4 (14 x (10V / 4095 = 2,44 mV) x 4020).

Observa-se ainda um desempenho superior do VS01A - ponto flutuante em relação ao modelo em ponto fixo. Isso pode ser justificado pelo fato do modelo em ponto flutuante utilizar um formato numérico mais próximo do formato utilizado pelo PC, além de não existirem truncamentos e/ou arredondamentos em etapas intermediárias de cálculo. Já o modelo em ponto fixo, além de apresentar uma limitação de faixa dinâmica, o que exige técnicas de "scaling" que acabam reduzindo a precisão da representação, promove o acúmulo de erro em etapas intermediárias de cálculo, devido a arredondamentos. Estes são necessários pelo fato da utilização de simples precisão (16 bits) para as saídas parciais e os graus de ativação pós-convertidos para ponto fixo.

6.2 Desempenho do controle em malha fechada com realimentação pelo sensor virtual

6.2.1 Simulações

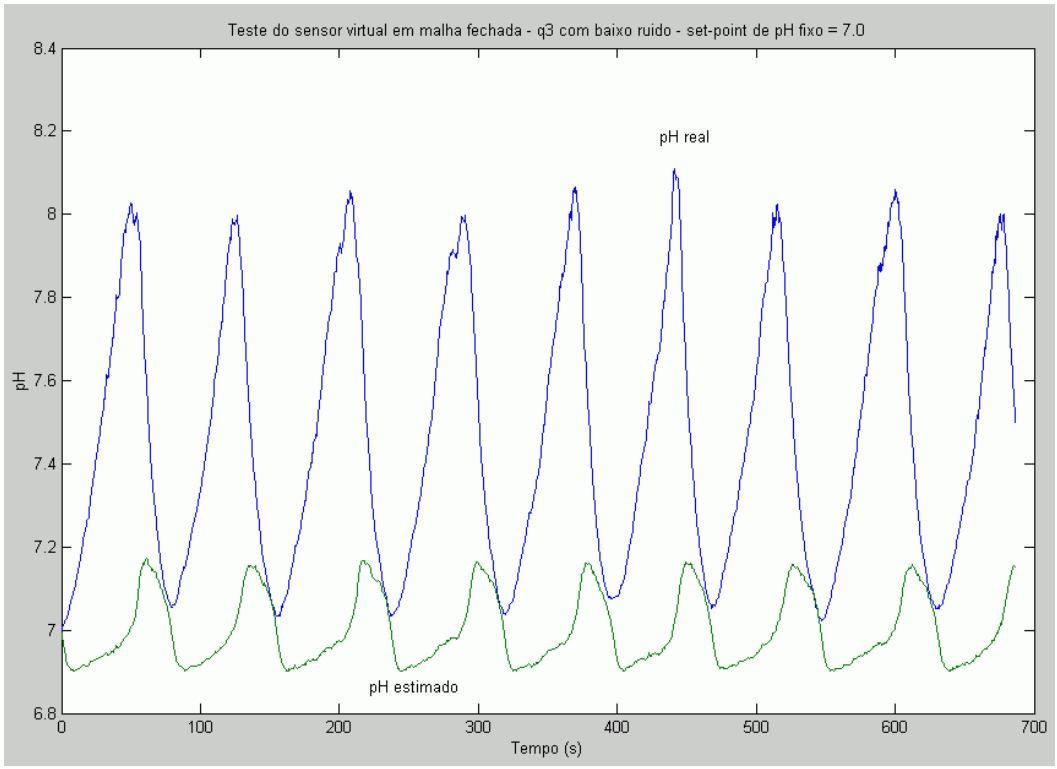


Figura 6.25 - Teste em malha fechada – modelo em PC - q_3 com baixo ruído - valor de referência fixo

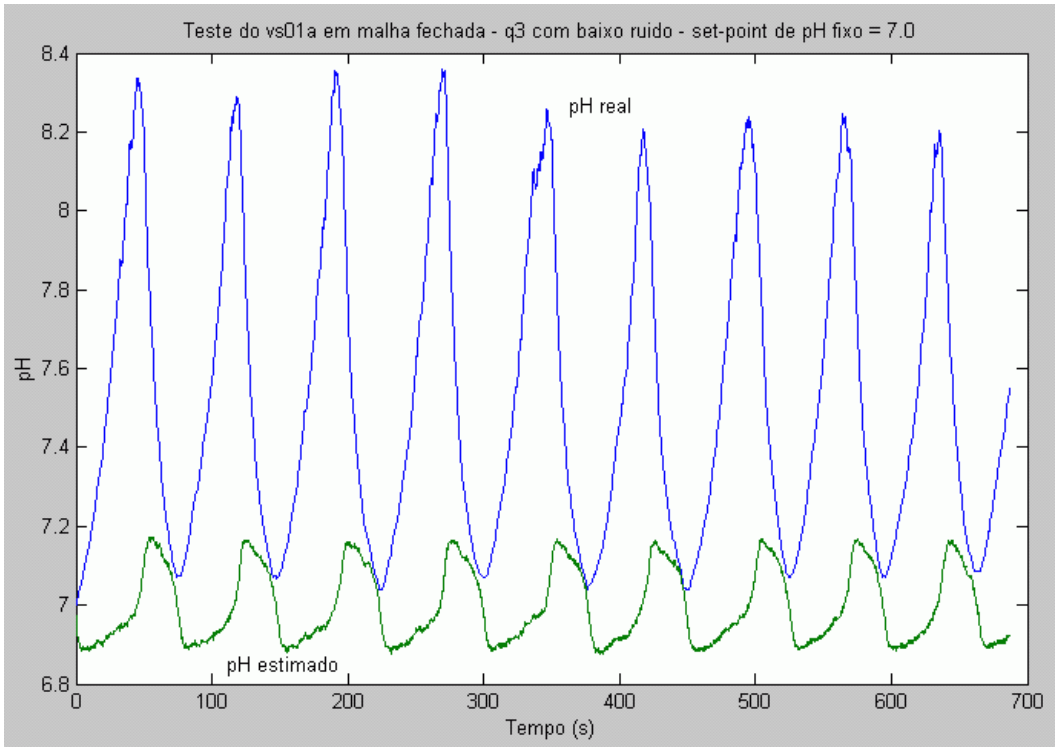


Figura 6.26 - Teste em malha fechada – VS01A - q_3 com baixo ruído - valor de referência fixo

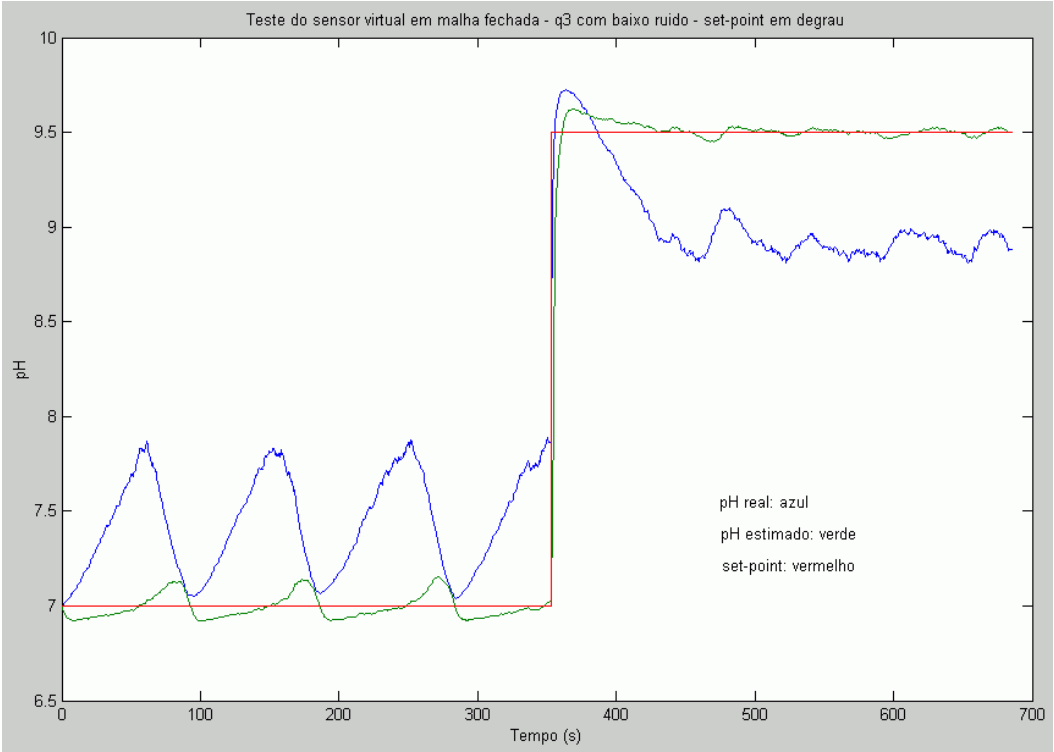


Figura 6.27 - Teste em malha fechada – modelo em PC - q_3 com baixo ruído - valor de referência em degrau 7 a 9,5

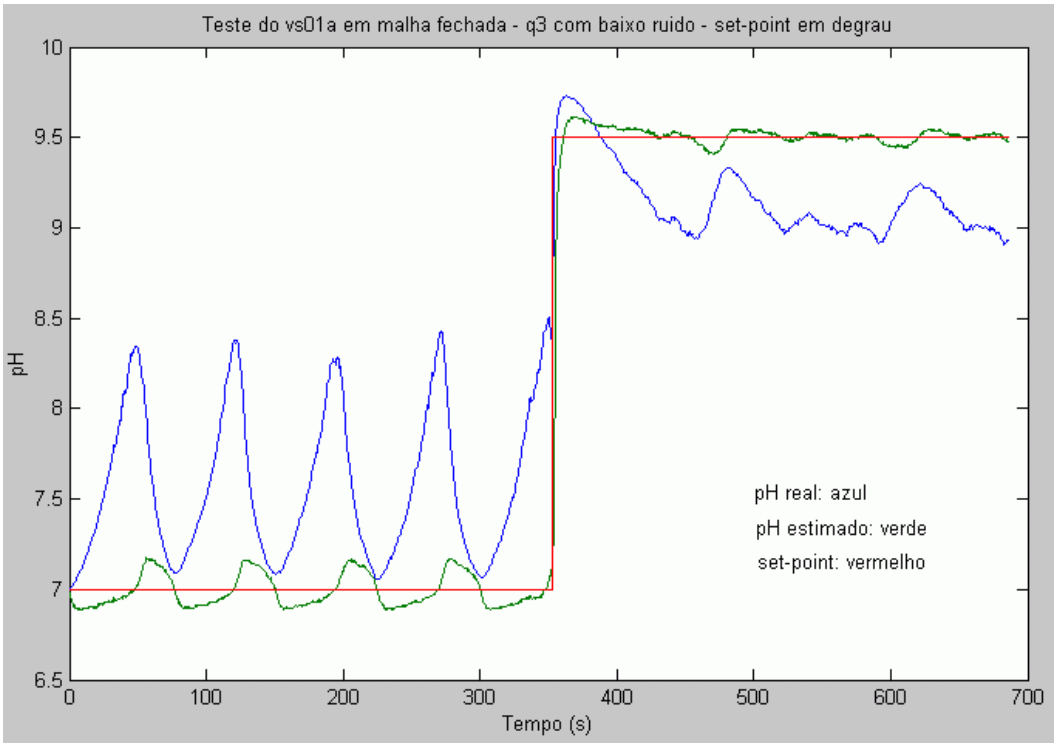


Figura 6.28 - Teste em malha fechada – VS01A - q_3 com baixo ruído - valor de referência em degrau 7 a 9,5

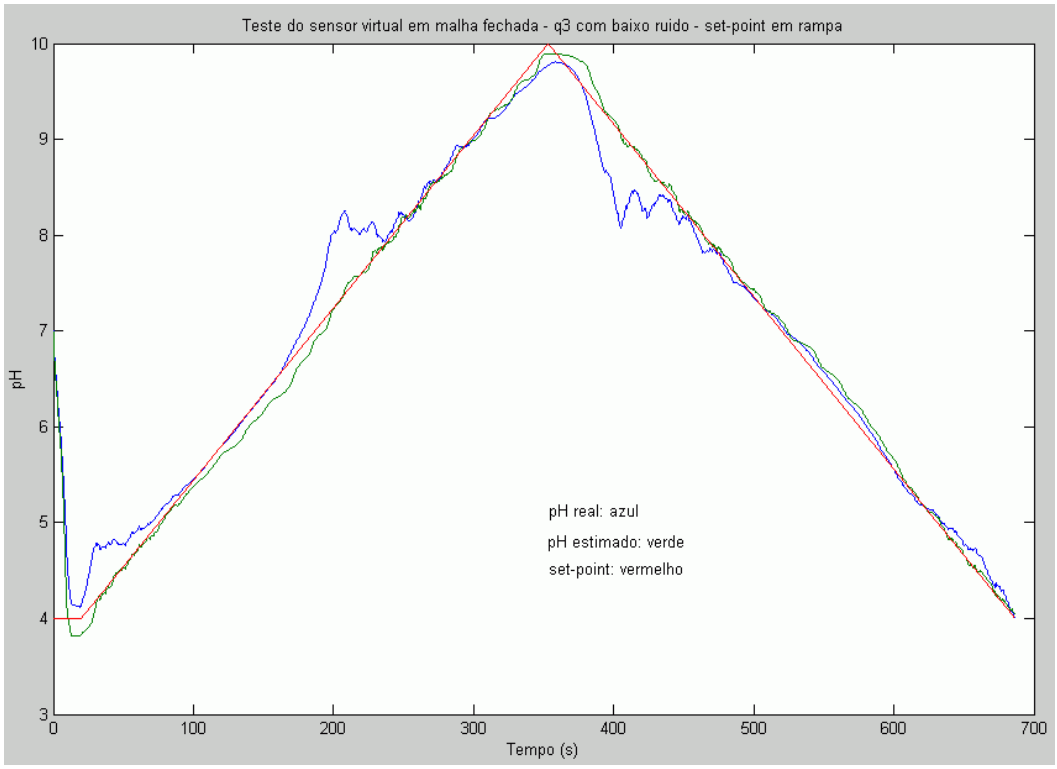


Figura 6.29 - Teste em malha fechada – modelo em PC - q_3 com baixo ruído - valor de referência em rampa

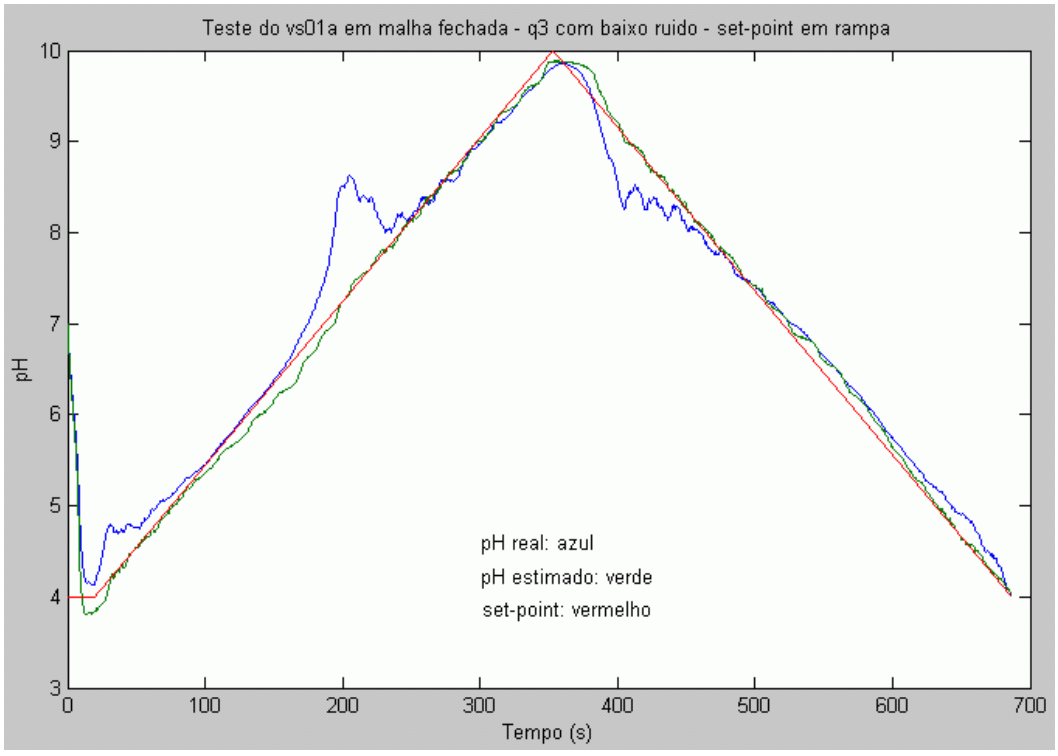


Figura 6.30 - Teste em malha fechada – VS01A - q_3 com baixo ruído - valor de referência em rampa

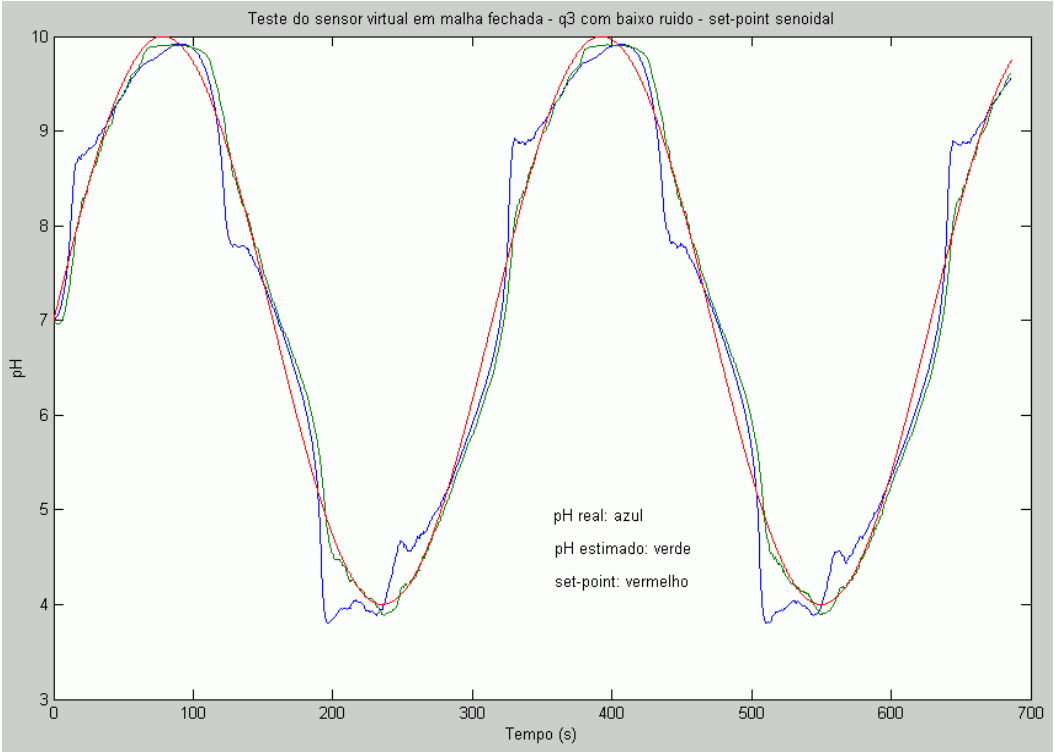


Figura 6.31 - Teste em malha fechada – modelo em PC - q_3 com baixo ruído - valor de referência senoidal

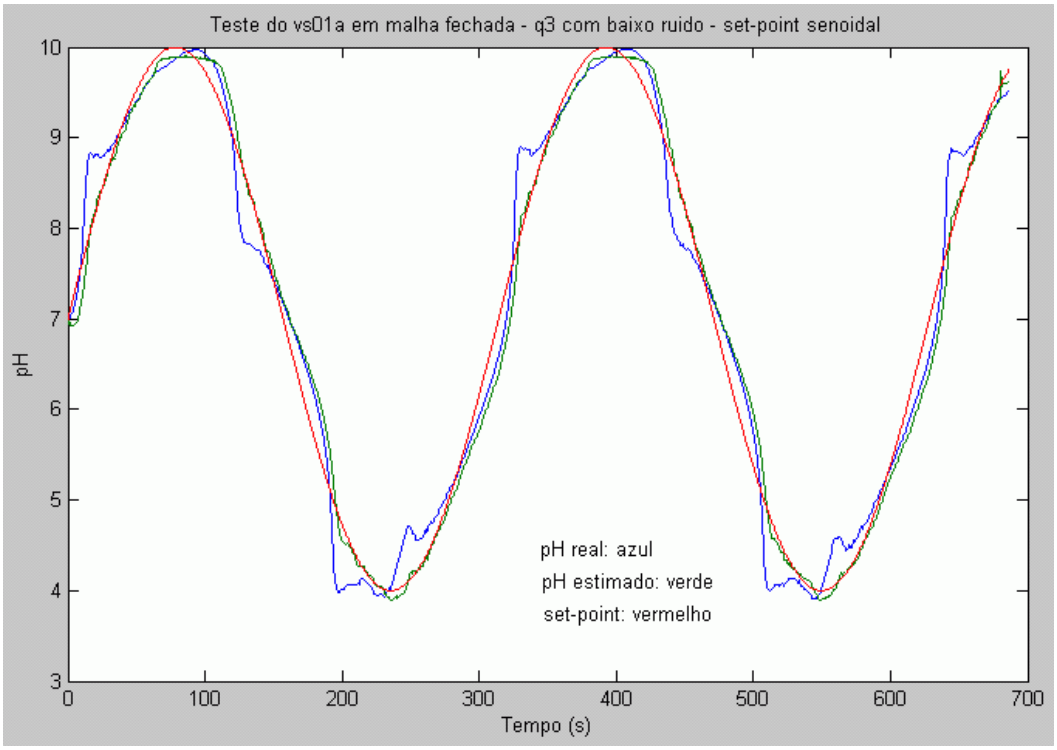


Figura 6.32 - Teste em malha fechada – VS01A - q_3 com baixo ruído - valor de referência senoidal

6.2.2 Tabela comparativa e comentários sobre os testes em malha fechada com realimentação pelo sensor virtual

Tabela 3 - ISE para q_3 com baixo ruído - realimentação pelo sensor virtual

Tipo de simulação	**Matlab**	**VS01A - ponto fixo**	**VS01A - ponto flutuante**
Valor de referência fixo - pH = 7	1317	2830	1743
Valor de referência em rampa	397	473,1	427,7
Valor de referência senoidal	507,6	350,3	394,4
Valor de referência em degrau 7 –> 9,5	917,6	1725	1453

Os índices de desempenho ISE do VS01A para simulações com sinal senoidal foram ligeiramente menores que os equivalentes do modelo em PC. Isso não pode ser interpretado como uma “vantagem” do VS01A em relação a seu modelo de origem. Provavelmente, o que ocorreu foi uma “contribuição” das fontes de erro do VS01A para uma aproximação do pH estimado de seu valor “real”, principalmente em torno do pH = 4 (vide figuras 6.31 e 6.32).

O desempenho com valor de referência fixo foi péssimo (figuras 6.25, 6.26, 6.27 e 6.28). Observa-se que o VS01A e o modelo original em Matlab ficaram igualmente ruins, o que comprova que a etapa de treinamento precisaria ser melhorada, com mais pontos coletados em torno do pH = 7.

6.3 Desempenho do conjunto controlador + sensor virtual em hardware em malha fechada

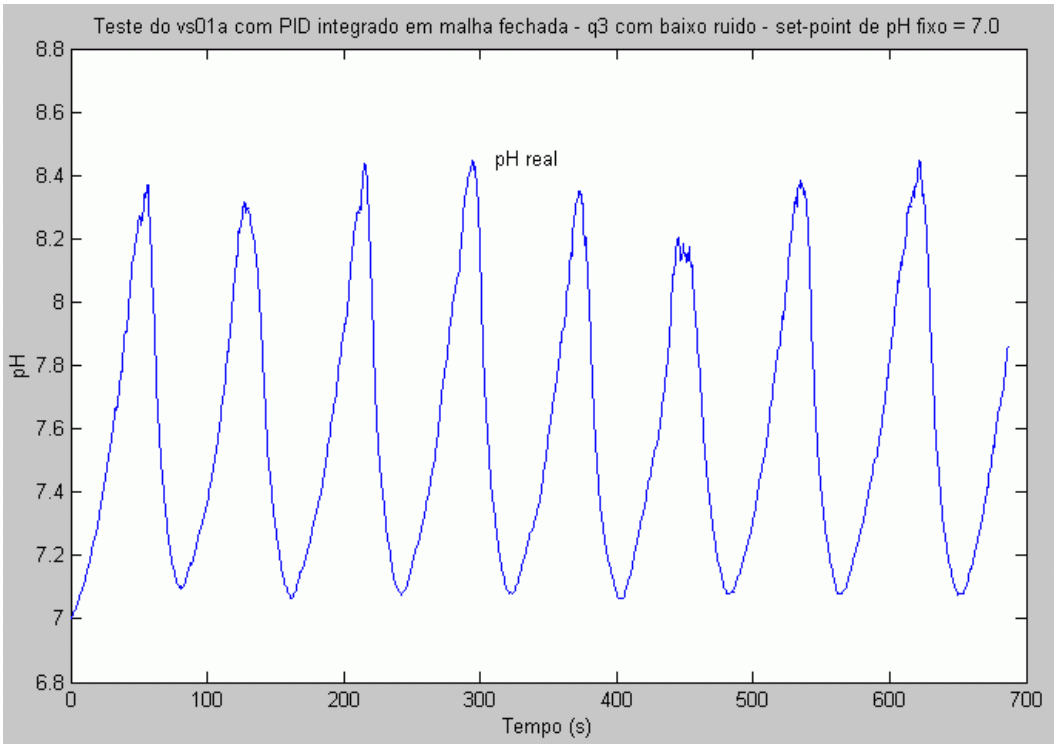


Figura 6.33 - Teste em malha fechada – VS01A + PID - q_3 com baixo ruído - valor de referência fixo

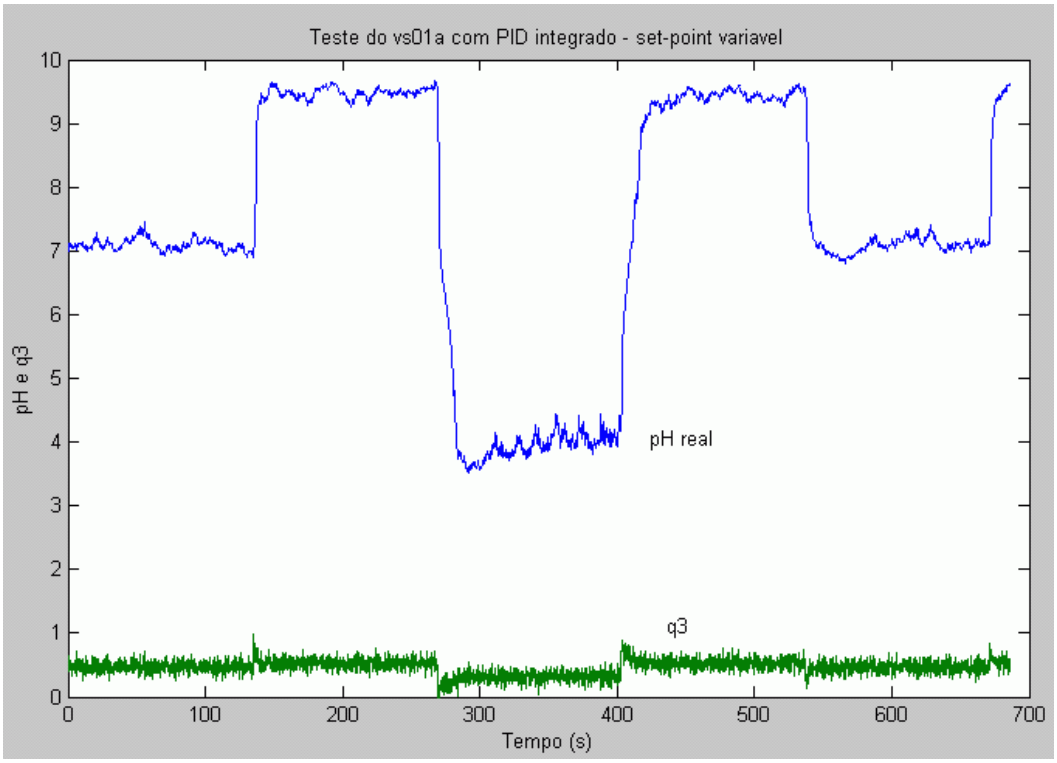


Figura 6.34 - Teste em malha fechada – VS01A + PID - q_3 com alto ruído - valor de referência variável

Observa-se pela figura 6.33 que a curva de pH do processo é praticamente a mesma daquela obtida com o PID no Matlab / Simulink (figura 6.26), o que comprova a eficácia do controlador PID no VS01A. Apenas como ilustração final, a figura 6.34 apresenta a curva de pH do processo para valor de referência variável em degrau, cujos valores são pH = 4, pH = 7 e pH = 9,5.

7 CONCLUSÕES E SUGESTÕES PARA FUTUROS TRABALHOS

7.1 Conclusões

Este trabalho descreveu o desenvolvimento, implementação e análise de desempenho de um protótipo de sensor virtual autônomo. Foi tido como propósito fundamental fornecer aos meios acadêmico e fabril um estudo preliminar sobre a efetiva aplicabilidade dessa ferramenta junto à indústria.

A ordem de grandeza dos índices de desempenho ISE obtidos com o protótipo VS01A foram semelhantes aos do modelo original em Matlab, comprovando a eficácia do sistema “embedded”. Esse sistema pode efetivamente ser utilizado em aplicações industriais, seja “apenas” como sensor, seja como solução mais completa: sensor + controlador. Evidentemente, nesse último caso, torna-se necessária uma Interface Homem - Máquina (IHM) para ajuste de sintonia, valor de referência, etc.

Salienta-se aqui o caráter genérico do hardware, o qual não precisa ser modificado para cada processo. Basta que as entradas e saídas em geral sejam compatíveis (faixa de tensão, corrente, etc) com o VS01A. O que caracteriza o VS01A como um determinado tipo de sensor virtual é o seu firmware, o qual deve ser reprogramado para atender a plantas distintas.

Técnicas de identificação nebulosa têm sido bastante estudadas para a geração de modelos. Porém, para o VS01A, é possível, no futuro, a utilização de outras técnicas, como por exemplo Redes Neurais Artificiais, uma vez que o firmware pode ser reprogramado.

Outro ponto importante é o fato do sistema "embedded" ser capaz de processar mais de um algoritmo de sensor virtual, ou seja, estimar on-line duas ou mais variáveis críticas de um processo, ou ainda variáveis de subprocessos distintos. A análise de viabilidade que deve ser feita é em relação aos tempos de amostragem exigidos por cada variável / subprocesso.

7.2 Sugestões para futuros trabalhos

Novos estudos na área de sensores virtuais "embedded" podem seguir por, no mínimo, quatro linhas de pesquisa. A primeira delas consiste na análise de novas tecnologias de hardware e software, como por exemplo microcontroladores, dispositivos de lógica programável (CPLD´s e FPGA´s) ou ainda DSP´s de alta performance. Cada uma dessas tecnologias possui inúmeras vantagens e desvantagens, dependendo do processo em que serão aplicadas.

Um segundo enfoque diz respeito à utilização de novos algoritmos para treinamento e execução do modelo do sensor. A mais emergente das técnicas atuais é a RNA - Redes Neurais Artificiais.

Uma terceira fonte de pesquisa é a implementação do VS01A junto a processos que possuam uma real necessidade de um sensor virtual. Um exemplo seria uma planta de tratamento de efluentes (SPERLING, 1996). A variável *DBO* (Demanda Bioquímica de Oxigênio) não é diretamente medida, mas apenas avaliada por experimentos laboratoriais, o que acarreta custos e atrasos no controle da qualidade da água.

Um estudo global de viabilidade para comercialização do sensor virtual "embedded" como produto é outro enfoque de pesquisa possível. Além do produto, deve estar agregado a ele o serviço de inspeção inicial da planta e a coleta de dados para geração do modelo.

REFERÊNCIAS BIBLIOGRÁFICAS

AGUIRRE, L. A. **Introdução à Identificação de Sistemas**: Técnicas Lineares e Não-Lineares Aplicadas a Sistemas Reais. Belo Horizonte: Ed. UFMG, 2000. 554p.

ASPENTECH. **Aspen IQ**: Powerful Inferential Sensor Modeling and Implementation Package. Cambridge, Massachusetts: Aspen Technology, Inc. 1998. Disponível em: <http://www.aspentech.com>. Acesso em: fev. 2004.

ÅSTRÖM, K. J; WITTENMARK, B. **Computer Controlled Systems**: Theory and Design. 3*rd* edition. New Jersey: Prentice Hall, 1997. 557p.

BABUSKA, R. **Lecture 3: Construction of Fuzzy Systems**. 2001. Disponível em: <http://lcewww.et.tudelft.nl/~babuska>. Acesso em: set. 2003.

COSTA, H. R. N. **Identificação Neuro-Fuzzy de um Processo de Neutralização de pH**. 2001. 143 f. Exame de Qualificação para Tese de Doutoramento. Escola Politécnica da Universidade de São Paulo, São Paulo, 2001.

CROWELL, C. **Floating-Point Arithmetic with the TMS32020**: Application Report SPRA011. Digital Signal Processing Solutions. Dallas: Texas Instruments Inc. 1989.

GARCIA, C. **Identificação de Sistemas**: Apostila do curso de pós graduação PTC-5719. São Paulo: Escola Politécnica da Universidade de São Paulo, 2001.

HAROLD, D. Process Control´s Latest Tool: Soft Sensors. **Control Engineering Europe**. Jun. 2001. Disponível em: <http://www.controleng.com>. Acesso em: jun. 2003.

HELLENDOORN, H.; DRIANKOV, D. (Eds.) **Fuzzy Model Identification**: Selected Approaches. Berlin: Springer-Verlag, 1997. 318p.

JANG, J. S. R.; SUN, C. T.; MIZUTANI, E. **Neuro-Fuzzy and Soft Computing**: A computational Approach to Learning and Machine Intelligence. New Jersey: Prentice Hall, 1997. 613p.

LJUNG, L. **System Identification**: Theory for the User. 2nd edition. New Jersey: Prentice Hall, 1999. 609p.

JOHANSEN, T. A. Robust Identification of Takagi-Sugeno-Kang Fuzzy Models using Regularization. **SINTEF Automatic Control**, N-7034 Trondheim, Norway, [1996?].

LEONARDI, F. **Notas de aula da disciplina "Projeto de Sistemas de Controle II"**. São Caetano do Sul: Escola de Engenharia Mauá do Instituto Mauá de Tecnologia, 2000.

LYNX TECNOLOGIA ELETRÔNICA LTDA. **CAD12/36**: Manual do usuário e de referência. Rev. 1.2. São Paulo. Nov. 1993.

OLIVEIRA, C. E. N.; GARCIA, C. Otimização do algoritmo de Wang e Langari - Regras Limitadas. **Relatório final de bolsa de iniciação científica FDTE**. São Paulo, 2003.

PERRELLA, N. **TMS320 One Day Workshop**. Santo André, maio de 2000.

SHAW, I. S.; SIMÕES, M. G. **Controle e Modelagem Fuzzy**. São Paulo: Edgarg Blucher, 1999. 165p.

SPERLING, M. V. **Princípios do tratamento biológico de águas residuárias**: Introdução à qualidade das águas e ao tratamento de esgotos. 2ª edição revisada. Belo Horizonte: Departamento de Engenharia Sanitária e Ambiental - DESA - Universidade Federal de Minas Gerais, 1996. 243p.

TEXAS INSTRUMENTS. **TMS320F206 Datasheet**: Literature number SPRS050A. Dallas: Texas Instruments Inc. Abril de 1998.

TEXAS INSTRUMENTS. **TMS320C2XX User´s Guide**: Literature number SPRU127B. Dallas: Texas Instruments Inc. Jan. 1997.

TEXAS INSTRUMENTS. **Digital Control Applications with the TMS320 Family**: Selected Application Notes. Literature number SPRA019. Dallas: Texas Instruments Inc. Maio de 1991.

TEXAS INSTRUMENTS. **Implementation of Fuzzy Logic**: Selected Applications. Literature number SPRA028. Dallas: Texas Instruments Inc. Jan. 1993.

WANG, L.; LANGARI, R. Complex Systems Modeling via Fuzzy Logic. **IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics**, v.26, n.1, p. 100-106, feb. 1996.

WHITE MOUNTAIN DSP. **Pathway 2xx (TMS320F206) DSP Starter Kit User´s Guide**. New Hampshire. Dez. 1997.

YOKOGAWA. **Exarque**: Advanced Process Control Solutions. Tokyo: Yokogawa Electric Corporation. Manual No TI36J06D20-01E. 2001.

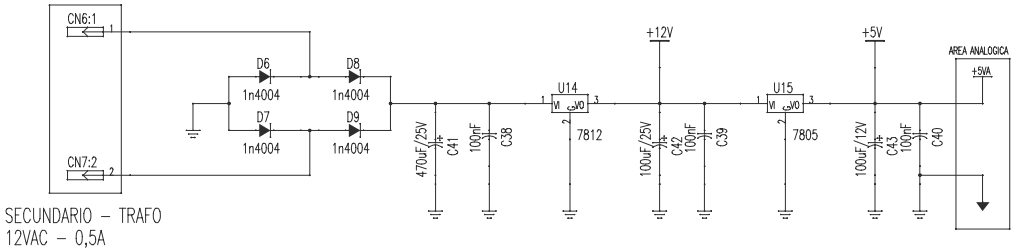
APÊNDICE A - Especificações técnicas básicas - DSP TMS320F206

- Tensão de alimentação: 5V;
- Velocidade de processamento: 20 Mips (50 ns de ciclo de instrução);
- Arquitetura Harward: barramentos independentes de dados e programa (ambos de 16 bits);
- Barramento de endereçamento externo de 16 bits;
- 544 x 16 words de memória RAM interna;
- 32K x 16 words de memória de programa FLASH EEPROM interna;
- 4K x 16 words de memória espelhada como dados ou programa;
- Endereçamento externo de programa: 64K words;
- Endereçamento externo de dados: 64K words;
- Endereçamento externo de I/O: 64K words;
- Acumulador de 32 bits;
- Unidade lógica e aritmética de 32 bits;
- Unidade MAC (multiplicação e acumulação) de 16 x 16, com resultado em 32 bits;
- 3 interrupções externas (INT1..INT3) e uma interrupção NMI;
- Um timer de 16 bits;
- 6 pinos de I/O de uso geral;
- 1 canal serial UART;
- 1 canal serial síncrono; e
- Encapsulamento TQFP - 100 pinos.

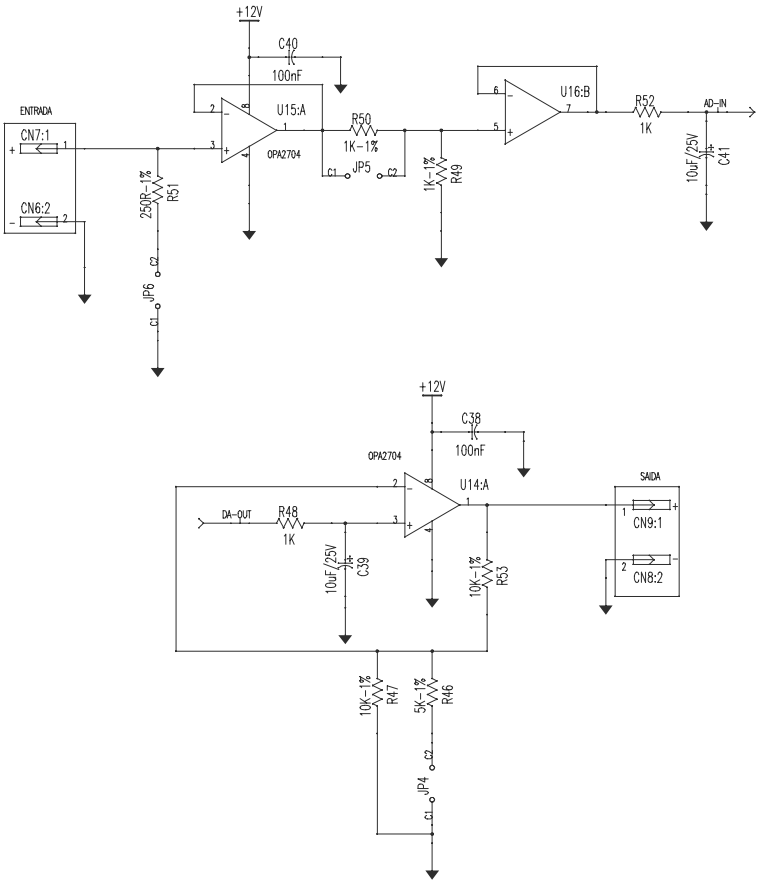
APÊNDICE B -Especificações técnicas básicas - placa CAD12/36

- Conversor A/D de 12 bits;
- Barramento principal do tipo ISA - 8 bits;
- 16 entradas analógicas simples ou 8 diferenciais multiplexadas;
- Suporte para interrupções;
- 16 entradas digitais;
- 16 saídas digitais;
- Suporta operação DMA;
- Ganhos programáveis;
- 3 temporizadores / contadores de 16 bits, para sincronismo de operações; e
- Módulo de expansão com conversor D/A de 12 bits e 4 saídas analógicas.

APÊNDICE C -Esquemas elétricos - VS01A



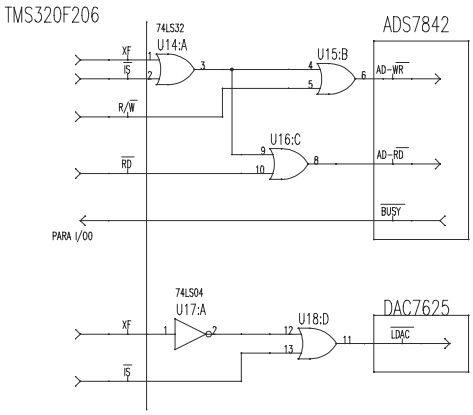
Title	FONTE DE ALIMENTACAO	
Size A3	Number VS01A	Rev 0.0
Date 19/01/2004	Drawn by CCB	
Filename VS01A.SCH	Sheet 1 of 6	

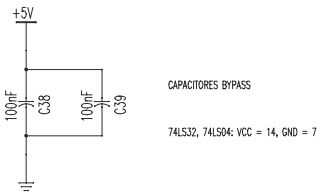


TIPO DE ENTRADA	JP1	JP2
0..10V	OFF	OFF
0..5V 1..5V	OFF	ON
0..20mA 4..20mA	ON	ON

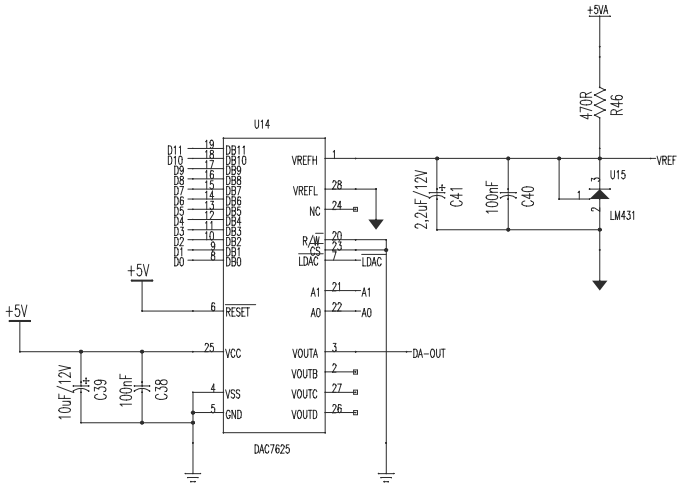
TIPO DE SAIDA	JP3
0..10V	ON
0..5V 1..5V	OFF

Title	INTERFACES DE ENTRADA E SAIDA ANALOGICAS	
Size A3	Number VS01A	Rev 0.0
Date 19/01/2004	Drawn by CCB	
Filename VS01A.SCH	Sheet 2 of 6	

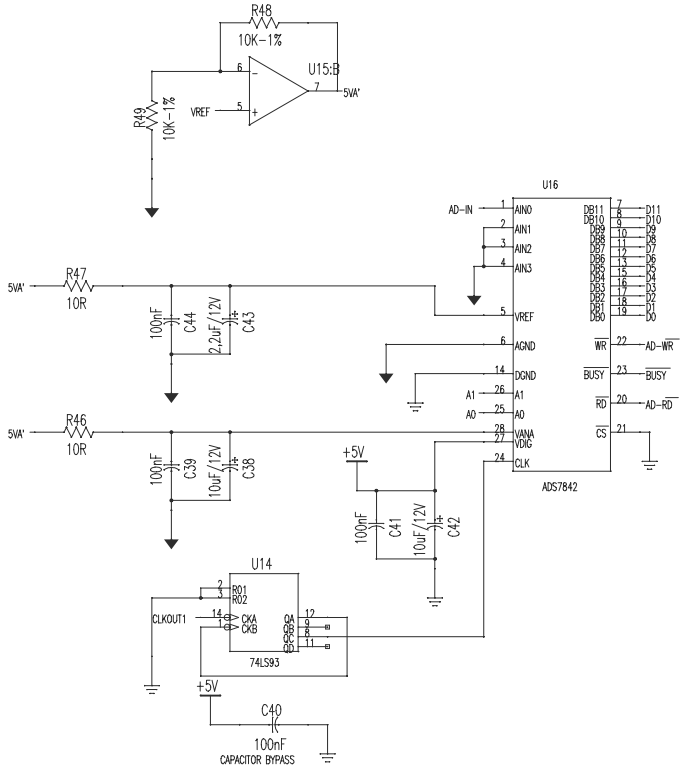




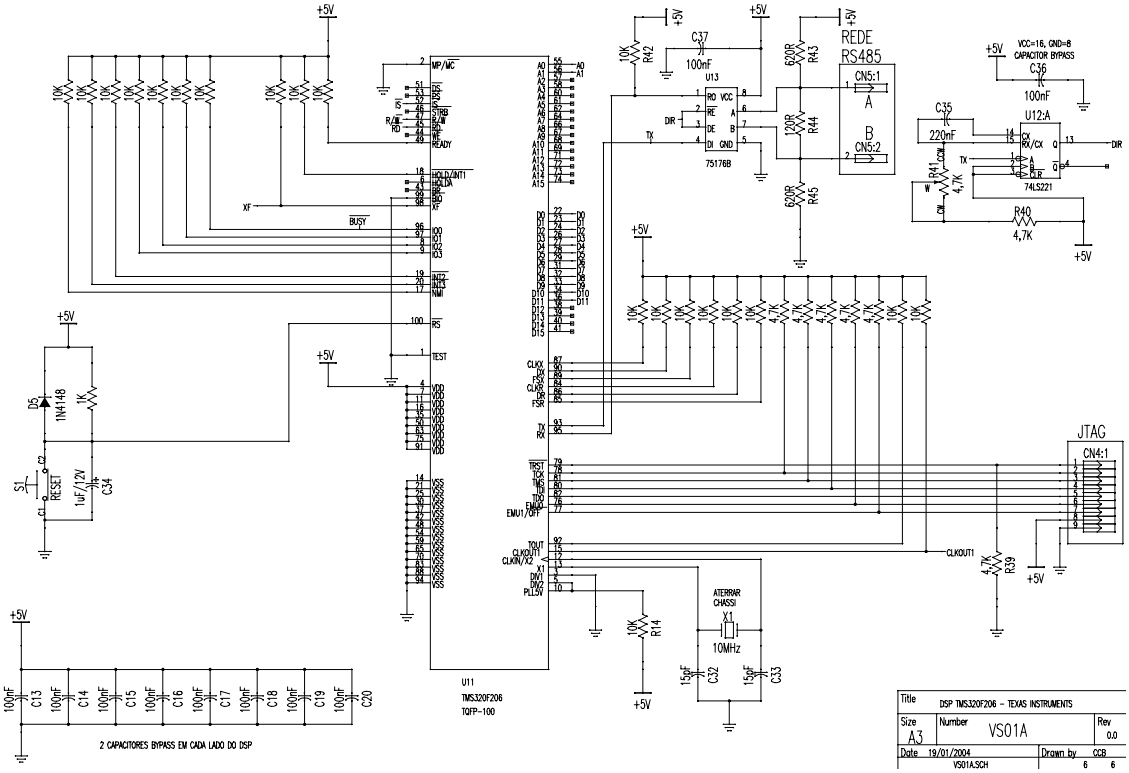
Title	CHIP SELECT – A/D E D/A	
Size A3	Number VS01A	Rev 0.0
Date 19/01/2004	Drawn by CCB	
VS01A.SCH	3	6



Title	CONVERSOR D/A	
Size A3	Number VS01A	Rev 0.0
Date 19/01/2004	Drawn by CCB	
Filename VS01A.SCH	Sheet 4 of 6	



Title	CONVERSOR A/D	
Size A3	Number VS01A	Rev 0.0
Date 19/01/2004	Drawn by CCB	
VS01A.SCH	5 6	



Title	DSP TMS320F206 - TEXAS INSTRUMENTS	
Size A3	Number VS01A	Rev 0.0
Date 19/01/2004	Drawn by CCB	
VS01A.SCH	6 6	

APÊNDICE D -Códigos-fonte em Matlab - treinamento e execução do modelo de sensor virtual em PC

D.1 Software de treinamento off-line

```
function [centros, WB]=rltreinar(num_clusters, wdados, tol, informacoes)
global num_var;
global n_clusters;
global centros;
global WB;
n_clusters = num_clusters;
[num_pontos, num_var]=size(wdados);
num_var=num_var-1;

% encontra os centros dos clusters para todas as variáveis
      if informacoes==1
      fprintf('Calculando os centros dos clusters.\n')
      end
centros1=fcm(wdados,num_clusters,[2 100 tol 0]);
centros=zeros(num_clusters,num_var);
centros(:,:)=centros1(1:num_clusters,1:num_var);                    % Projeta os centros no espaco das variáveis de entrada,
                                                                    % ou seja, ignora a última coordenada
clear centros1;

%% Início do procedimento de cálculo dos coeficientes 'b'
% Cálculo das pertinências de cada ponto a cada cluster
      if informacoes==1
      fprintf('Calculando as pertinencias de cada ponto a cada cluster.\n');
      end
pert=zeros(num_pontos, num_clusters);

for k=1:num_pontos

% calcula distâncias
dist2=[];                                                           % distância do ponto atual aos clusters
      for j=1:num_clusters
      z=wdados(k,1:num_var)-centros(j,:);
      dist2(j)=sum(z.^2);
      end
clear z;

% calcula pertinências
      for j=1:num_clusters
      pert(k,j)=1/sum( dist2(j)*(dist2.^-1) );
      end

end
```

```
Phi=pert;
    for i=1:num_var
    Phi=[Phi pert.*(wdados(:,i)*ones(1,num_clusters))];
    end
if informacoes==1
fprintf('Inicio da otimizacao.\n');
end
y=wdados(:,num_var+1);
pack;

% Método numérico para solução de mínimos quadrados
maxnr=num_clusters;
C=eye(maxnr*(num_var+1));
b=ones(maxnr*(num_var+1),1);
erro=1;
h=0;

while erro>=tol&h<1000

    for k=1:num_pontos
    fk=C'*Phi(k,:)';
    Gama=(1+fk'*fk)^(0.5);
    Alpha=Gama+1;
    Beta=1/(Gama*Alpha);
    Delta=Beta*C*fk;
    aux1=Delta*(Alpha/Gama*(y(k)-Phi(k,:)*b));
    b=b+aux1;
    C=C-Delta*fk';
    end;
erro=sum(aux1.*aux1);
h=h+1;
    if informacoes==1
    fprintf('Iteracao: %d\t Erro:%g\n',h,erro);
    end


end;

    if informacoes==1
    fprintf('Fim.\nGerando coeficientes\n');
    end
WB=zeros(num_var+1,maxnr);
nr=1;
    for i=1:num_var+1
        for j=1:maxnr
        WB(i,j)=b(nr);
        nr=nr+1;
        end
    end
centros
WB
return;
```

D.2 Software de execução on-line

```
function y=rlexecutar(U)
U=U';
global num_var;
global n_clusters;
global centros;
global WB;
temp=size(centros);
num_var=temp(2);
y=0;
    if [1, num_var]~=size(U)
    fprintf('Numero de variaveis diferente do treinamento');
    return;
    end

% calcula distâncias
dist2=[];                                   % distância do ponto atual aos clusters
    for j=1:n_clusters
    z=U(1:num_var)-centros(j,:);
    dist2(j)=sum(z.^2);
    end
clear z;

% calcula pertinências
pert=zeros(1,n_clusters);
    for j=1:n_clusters
    pert(j)=1/sum( dist2(j)*(dist2.^-1) );
    end
U=[1 U];
y=(U*WB)*pert';                             % U*WB resulta numa matriz contendo as saídas de cada regra
                                            % (U*WB)*pert' multiplica cada saída parcial pelo seu peso e soma
return;
```

APÊNDICE E -Código-fonte em Pascal - software de teste e calibração da placa CAD12/36

```
program ad_to_da;
uses crt;
var
    a,b,estado:byte;
    x: integer;
    y: boolean;
begin
clrscr;
port[$384]:=$03;
port[$385]:=$30;                    {programa modo 0}
port[$384]:=$00;
port[$385]:=$00;                    {registrador de limite}
port[$384]:=$01;
port[$385]:=$00;                    {acerta pont. mem. canais}
port[$384]:=$04;
port[$385]:=$8f;                    {programa mem. 0 - ver errata - manual}
port[$384]:=$06;
port[$385]:=$00;                    {auto-calibração}
delay(500);
    for x:=1 to 16 do
    begin
    a:=port[$384];
    b:=port[$385];                  {esvazia fifo}
    end;
y:=false;

repeat
port[$384]:=$01;
port[$385]:=$00;                    {pont. mem. canais = 0}
port[$384]:=$02;
port[$385]:=$00;                    {dispara conversão}
    repeat
    estado:=port[$383];
    estado:=(estado and $01);
    until estado = 1;               {aguarda término da conversão}
a:=port[$384];
b:=port[$385];                      {lê a/d}
port[$388]:=a;
port[$389]:=b;                      {escreve no d/a}
until y = true;
end.
```

APÊNDICE F -Códigos-fonte em C - driver real time da placa CAD12/36 para Matlab / Simulink

```
/* CAD1236.C
*
* Biblioteca básica de acesso à configuração da placa Lynx CAD12/36
*
* RPM - 19/02/1999
*
* CCB - 16/01/2004 - acertos para a placa CAD12/36 - Rev. 4
*/
/* Escolha do sistema operacional ********************/
#undef is_winNT /* Opcoes: *
* #define is_winNT Windows NT *
                    * (requer giveio.sys instalado) *
* #undef is_winNT Windows 9x */
/* Includes ****************************************/
#include "simstruc.h"
#include "time.h"
#include <stdio.h>
#ifdef is_winNT
#include <windows.h>
#endif
/* Macros ******************************************/
/*****************************
* Interface com o Simulink *
*****************************/
#define PARAM_STRLEN (128)
/***********************************
* Interface com a placa cad12/36 *
***********************************/
/* Endereços secundários */
#define LIM_REG_ADRS 0x0
#define MEM_PTR_ADRS 0x1
#define AD_CONV_ADRS 0x2
#define MOD_REG_ADRS 0x3
#define MEM_WRT_ADRS 0x4
#define SLF_CAL_ADRS 0x6
/* Programação de canais (8bits : [D7-D6-D5-D4-D3-D2-D1-D0])*/
/* D7 */
#define P1 0x00
#define P2 0x80
/* D6-D5-D4 */
#define G001 0x70
#define G002 0x60
#define G005 0x50
#define G100 0x30
/* D3-D2-D1-D0 */
#define Ch00 0x00
#define Ch01 0x01
#define Ch02 0x02
#define Ch03 0x03
#define Ch04 0x04
```

```
#define Ch05 0x05
#define Ch06 0x06
#define Ch07 0x07
#define Ch08 0x08
#define Ch09 0x09
#define Ch10 0x0a
#define Ch11 0x0b
#define Ch12 0x0c
#define Ch13 0x0d
#define Ch14 0x0e
#define Ch15 0x0f
/* Memória de canais */
#define mem00 0x0
#define mem01 0x1
#define mem02 0x2
#define mem03 0x3
#define mem04 0x4
#define mem05 0x5
#define mem06 0x6
#define mem07 0x7
#define mem08 0x8
#define mem09 0x9
#define mem10 0xa
#define mem11 0xb
#define mem12 0xc
#define mem13 0xd
#define mem14 0xe
#define mem15 0xf
/* Range do conversor D/A */
#define P2_10       1           /* -5V a +5V */
#define P2_05       2           /* -10V a +10V */
#define P2_25       3           /* -2.5V a +2.5V */
#define P1_05       4           /* 0V a +5V */
#define P1_10       5           /* 0V a +10V */
#define C_4_20      6           /* 4mA a 20mA */
#define C_0_20      7           /* 0mA a 20mA */
#define DA_RES 4096
#define TWOCPL      1           /* complemento de dois */
#define BINOFST     2           /* binario+offset */
/* Tipos e estruturas ********************************/
typedef struct
{
int Ctr0;
int Ctr1;
int Ctr2;
int Mode;
int Stat;
int ByteA;
int ByteB;
int Adrs;
int Data;
} CAD1236HwSetup;
/* Rotinas ***************************************/
/* Inicializa endereços */
static CAD1236HwSetup SetUpHwCAD1236(SimStruct *S)
```

```
{
CAD1236HwSetup       Board;
char_T       baseAddrStr[PARAM_STRLEN];
uint_T       baseAddr;
mxGetString(ssGetSFcnParam(S,0), baseAddrStr, PARAM_STRLEN);
baseAddr = (uint_T) strtoul(baseAddrStr, NULL, 0);
Board.Ctr0 = baseAddr;
Board.Ctr1 = baseAddr + 1;
Board.Ctr2 = baseAddr + 2;
Board.Mode = baseAddr + 3;
Board.Stat = baseAddr + 3;
Board.ByteA = baseAddr + 4;
Board.ByteB = baseAddr + 5;
Board.Adrs = baseAddr + 4;
Board.Data = baseAddr + 5;
return (Board);
}
/* Escrita no registrador secundário da CAD12/36 */
void WriteSecReg(CAD1236HwSetup Board, char_T RegAddress, char_T Data)
{
_outp (Board.Adrs,RegAddress);        /* 1o. passo: acessa o registrador secundário */
_outp (Board.Data,Data);                  /* 2o. passo: escreve o dado no registrador secundário */
}
/* Escrita na memória de canais da CAD12/36 */
void WriteChMem (CAD1236HwSetup Board, char_T Pos, char_T Data)
{
Data = ~Data;                                                    /* errata: placa cad12/36 rev. 4 - alteração do chip da memória de canais*/
WriteSecReg (Board, MEM_PTR_ADRS, Pos);              /* 1o. passo: escreve o endereço da posição no registrador de ponteiro */
WriteSecReg (Board, MEM_WRT_ADRS, Data);           /* 2o. passo: escreve o dado no endereço da posição */
}
/* Autocalibração da CAD12/36 */
void SelfCalibrate (CAD1236HwSetup Board)
{
clock_t t0;
/* Inicia auto calibração */
WriteSecReg (Board, SLF_CAL_ADRS, 0x0);
/* Aguarda 500ms para o término do processo de auto calibração */
t0 = clock();
while ( ((double)(clock()-t0))/CLOCKS_PER_SEC < 0.5 );
}
/* Esvazia FIFO da CAD1236 */
void EmptyFIFO (CAD1236HwSetup Board)
{
int i;
for (i = 0; i < 16; i++)
{
_inp (Board.ByteB);
_inp (Board.ByteA);
}
}
/* Codifica um número para uso do conversor DA */
unsigned short int double2bin (double num, int range)
{
double vmin, vmax;
unsigned short int outcome;
```

```
switch (range)
{
case P2_10:
      vmin = -10;
      vmax = 9.9951;
      break;
case P2_05:
      vmin = -5;
      vmax = 4.9976;
      break;
case P2_25:
      vmin = -2.5;
      vmax = 2.4988;
case P1_05:
      vmin = 0;
      vmax = 4.9988;
case P1_10:
      vmin = 0;
      vmax = 9.9976;
      break;
case C_4_20:
      vmin = 0.004;
      vmax = 0.0199961;
      break;
case C_0_20:
      vmin = 0;
      vmax = 0.0199951;
default:
vmin = 0;
vmax = 0;
}
if (num < vmin)
{ num = vmin; }
if (num > vmax)
{ num = vmax; }
outcome = (unsigned short int)((DA_RES-1)*(num-vmin)/(vmax-vmin));
return (outcome);
}
/* Escreve nas saídas analógicas */
void AnalogOutput (CAD1236HwSetup Board, int port, unsigned short int num)
{
int LSB,MSB;
MSB = num/16;
LSB = (num-MSB*16)*16;
_outp (Board.Ctr0+8+2*port,LSB);
_outp (Board.Ctr0+9+2*port,MSB);
}
/* Acesso ao driver GIVEIO (caso o sistema seja Windows NT) */
#ifdef is_winNT
int giveIOaccess (void)
{
HANDLE h;
h = CreateFile("\\\\.\\giveio", GENERIC_READ, 0, NULL,
OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
if(h == INVALID_HANDLE_VALUE)
```

```
{
printf("Couldn't access giveio device\n");
return -1;
}
CloseHandle(h);
return 0;
}
#else
int giveIOaccess (void)
{
return 1;
}
#endif
```

```
/* CAD1236_AD.C
*
* Implementa as rotinas de leitura da placa Lynx CAD12/36
*
* RPM - 20/05/1999
*
* CCB - 16/01/2004 - acertos para a placa CAD12/36 - Rev. 4
*/
#define S_FUNCTION_NAME cad1236_ad
#define S_FUNCTION_LEVEL 2
#include "cad1236.c"
/* Parâmetros */
#define NUMFMT_PARAM       ssGetSFcnParam(S,1)
#define RANGE_PARAM       ssGetSFcnParam(S,2)
/* Variáveis globais deste programa */
static      CAD1236HwSetup      Cad1236;      /* endereços da placa */
static      unsigned char      P;                    /* polaridade */
static      unsigned char      G;                    /* ganho */
/* Inicialização */
static void mdlInitializeSizes(SimStruct *S)
{
int rc; /* return code para giveIOaccess() */
ssSetNumSFcnParams(S,3);
if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) return;
if (!ssSetNumOutputPorts(S,1)) return;
ssSetOutputPortWidth(S, 0, DYNAMICALLY_SIZED);
if (!ssSetNumInputPorts(S,0)) return;
ssSetNumContStates(S,0);
ssSetNumDiscStates(S,0);
ssSetNumSampleTimes(S, 1);
ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
/*************************************************/
/* */
/* Caso o sistema utilizado seja Windows NT, */
/* o programa deverá abrir o device GIVEIO.SYS */
/* liberando o acesso para as portas de hardware. */
/* */
/* Usar com extremo cuidado! */
/* */
```

```
/* A funcao giveIOaccess() é compilada conforme */
/* a definição da macro is_winNT (veja o arquivo */
/* cad1236.c para mais informações) */
/* */
/************************************************/
rc = giveIOaccess();
/* Processamento dos parâmetros */
switch ((char_T) mxGetPr(RANGE_PARAM)[0])
{
case 1:
     P = P2; G = G001;
     break;
case 2:
     P = P2; G = G002;
     break;
case 3:
     P = P2; G = G005;
     break;
case 4:
     P = P2; G = G100;
     break;
case 5:
     P = P1; G = G001;
     break;
case 6:
     P = P1; G = G002;
     break;
case 7:
     P = P1; G = G005;
     break;
case 8:
     P = P1; G = G100;
default:
return;
}
/* Inicialização do cartão AD */
Cad1236 = SetUpHwCAD1236(S);                    /* define os endereços */
/* Programação da memória do cartão AD */
/* 1. Preparação (conforme Manual CAD12/36 pag.4-5 */
WriteSecReg(Cad1236,MOD_REG_ADRS,0x0);          /* prepara o modo de operação para programação da memória */
WriteSecReg(Cad1236,LIM_REG_ADRS,0xf);          /* Ajusta o registrador de limite */
/* 2. Programação de cada canal */
WriteChMem(Cad1236,mem00,P+G+Ch00);
WriteChMem(Cad1236,mem01,P+G+Ch01);
WriteChMem(Cad1236,mem02,P+G+Ch02);
WriteChMem(Cad1236,mem03,P+G+Ch03);
WriteChMem(Cad1236,mem04,P+G+Ch04);
WriteChMem(Cad1236,mem05,P+G+Ch05);
WriteChMem(Cad1236,mem06,P+G+Ch06);
WriteChMem(Cad1236,mem07,P+G+Ch07);
WriteChMem(Cad1236,mem08,P+G+Ch08);
WriteChMem(Cad1236,mem09,P+G+Ch09);
WriteChMem(Cad1236,mem10,P+G+Ch10);
WriteChMem(Cad1236,mem11,P+G+Ch11);
WriteChMem(Cad1236,mem12,P+G+Ch12);
```

```
WriteChMem(Cad1236,mem13,P+G+Ch13);
WriteChMem(Cad1236,mem14,P+G+Ch14);
WriteChMem(Cad1236,mem15,P+G+Ch15);
/* Auto calibração da placa */
SelfCalibrate(Cad1236);
/* Limpeza da FIFO */
EmptyFIFO(Cad1236);
/* Seleção do modo de operação */
WriteSecReg(Cad1236,MOD_REG_ADRS,0x0);
                        /* define o modo de operação da placa: *
                        * 0x0 = 00000000 *
                        * modo 0 (sem DMA) *
                        * BURST desligado *
                        * timers desabilitados */
}
static void mdlInitializeSampleTimes(SimStruct *S)
{
ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
ssSetOffsetTime(S, 0, 0.0);
}
/* Leitura */
static void mdlOutputs(SimStruct *S, int_T tid)
{
unsigned char i,j,k,st;
unsigned short int l;
real_T *y = ssGetOutputPortRealSignal(S,0);
    /* Reprocessamento dos parâmetros */
    /* (para impedir que outra pessoa os altere */
Cad1236 = SetUpHwCAD1236(S);
switch ((unsigned char) mxGetPr(RANGE_PARAM)[0])
{
case 1:
    P = P2; G = G001;
    break;
case 2:
    P = P2; G = G002;
    break;
case 3:
    P = P2; G = G005;
    break;
case 4:
    P = P2; G = G100;
    break;
case 5:
    P = P1; G = G001;
    break;
case 6:
    P = P1; G = G002;
    break;
case 7:
    P = P1; G = G005;
    break;
case 8:
    P = P1; G = G100;
default:
```

```
return;
}
/* Operação de leitura */
for (i=0; i < ssGetOutputPortWidth(S,0); i++)
{
        WriteSecReg (Cad1236,AD_CONV_ADRS, i);
        do {
        st = _inp(Cad1236.Stat);
        } while ((st & 0x01) == 0);
        j = _inp(Cad1236.ByteA);
        k = _inp(Cad1236.ByteB);
        EmptyFIFO(Cad1236);
        l = (j & 0x0000ff) | ((k<<8) & 0x00ff00);
        if (((unsigned char) mxGetPr(NUMFMT_PARAM)[0]) == TWOCPL) {
            switch (P+G)
            {
                case (P1+G001):
                    y[i] = (5*((real_T)l)/32768 + 5)/1;
                    break;
                case (P1+G002):
                    y[i] = (5*((real_T)l)/32768 + 5)/2;
                    break;
                case (P1+G005):
                    y[i] = (5*((real_T)l)/32768 + 5)/5;
                    break;
                case (P1+G100):
                    y[i] = (5*((real_T)l)/32768 + 5)/100;
                    break;
                case (P2+G001):
                    y[i] = (10*((real_T)l)/32768)/1;
                    break;
                case (P2+G002):
                    y[i] = (10*((real_T)l)/32768)/2;
                    break;
                case (P2+G005):
                    y[i] = (10*((real_T)l)/32768)/5;
                    break;
                case (P2+G100):
                    y[i] = (10*((real_T)l)/32768)/100;
                    break;
                default:
                    return;
            }
        }
        else {
            switch (P+G)
            {
                case (P1+G001):
                    y[i] = (10*((real_T)l)/65536)/1;
                    break;
                case (P1+G002):
                    y[i] = (10*((real_T)l)/65536)/2;
                    break;
                case (P1+G005):
                    y[i] = (10*((real_T)l)/65536)/5;
```

```
                    break;
                case (P1+G100):
                    y[i] = (10*((real_T)l)/65536)/100;
                    break;
                case (P2+G001):
                    y[i] = (20*((real_T)l)/65536 - 10)/1;
                    break;
                case (P2+G002):
                    y[i] = (20*((real_T)l)/65536 - 10)/2;
                    break;
                case (P2+G005):
                    y[i] = (20*((real_T)l)/65536 - 10)/5;
                    break;
                case (P2+G100):
                    y[i] = (20*((real_T)l)/65536 - 10)/100;
                    break;
                default:
                    return;
            }
        }
    }
}
static void mdlUpdate(SimStruct *S, int_T tid)
{
}
static void mdlDerivatives(SimStruct *S)
{
}
/* Finalização */
static void mdlTerminate(SimStruct *S)
{
}
/* posfácio obrigatório */
#ifdef MATLAB_MEX_FILE                  /* Is this file being compiled as a MEX-file? */
#include "simulink.c"                   /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"                    /* Code generation registration function */
#endif



/* CAD1236_DA.C
*
* Implementa as rotinas de escrita na placa CAD12/36
*
* RPM - 20/05/1999
*
* CCB - 17/01/2004 - Setagem do flag "Direct Feedthrough - Matlab 6.1"
*
*/
/* Macros Simulink */
#define S_FUNCTION_NAME cad1236_da
#define S_FUNCTION_LEVEL 2
#include "cad1236.c"
/* Parâmetros */
```

```
#define PARAM_RANGE       ssGetSFcnParam(S,1)
/* Variáveis globais deste programa */
static CAD1236HwSetup Cad1236;                /* endereços da placa */
static       char_T       rangeDA;                    /* range de saída */
/* Inicialização */
static void mdlInitializeSizes(SimStruct *S)
{
int rc; /* return code para giveIOaccess() */
ssSetNumSFcnParams(S, 2);
if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) return;
if (!ssSetNumOutputPorts(S, 0)) return;
if (!ssSetNumInputPorts(S, 1)) return;
ssSetInputPortWidth(S, 0, DYNAMICALLY_SIZED);
ssSetNumContStates(S,0);
ssSetNumDiscStates(S,0);
ssSetNumSampleTimes(S, 1);
ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
ssSetInputPortDirectFeedThrough(S, 0, 1);       /* need direct feedthrough */
/* Inicialização do cartão */
Cad1236 = SetUpHwCAD1236(S);                  /* define os endereços */
/*****************************************************/
/* */
/* Caso o sistema utilizado seja Windows NT, */
/* o programa deverá abrir o device GIVEIO.SYS */
/* liberando o acesso para as portas de hardware. */
/* */
/* Usar com extremo cuidado! */
/* */
/* A função giveIOaccess() é compilada conforme */
/* a definição da macro is_winNT (veja o arquivo */
/* cad1236.c para mais informações) */
/* */
/*****************************************************/
rc = giveIOaccess();

     /* Processamento dos parâmetros */
switch ((char_T) mxGetPr(PARAM_RANGE)[0])
{
case 1:
     rangeDA = P2_10;
     break;
case 2:
     rangeDA = P2_05;
     break;
case 3:
     rangeDA = P2_25;
     break;
case 4:
     rangeDA = P1_05;
     break;
case 5:
     rangeDA = P1_10;
     break;
case 6:
     rangeDA = C_4_20;
```

```
        break;
case 7:
        rangeDA = C_0_20;
        break;
default:
return;
}
}
static void mdlInitializeSampleTimes(SimStruct *S)
{
ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
ssSetOffsetTime(S, 0, 0.0);
}
/* Escrita */
static void mdlOutputs(SimStruct *S, int_T tid)
{
InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
int nout = ssGetInputPortWidth(S,0);
int i;
        /* Reprocessamento dos parâmetros */
        /* (para impedir que outra instância os altere */
Cad1236 = SetUpHwCAD1236(S);
switch ((char_T) mxGetPr(PARAM_RANGE)[0])
{
case 1:
        rangeDA = P2_10;
        break;
case 2:
        rangeDA = P2_05;
        break;
case 3:
        rangeDA = P2_25;
        break;
case 4:
        rangeDA = P1_05;
        break;
case 5:
        rangeDA = P1_10;
        break;
case 6:
        rangeDA = C_4_20;
        break;
case 7:
        rangeDA = C_0_20;
        break;
default:
return;
}
if (nout > 4)
{
nout = 4;
}
/* Operação de escrita */
for (i=0; i < nout; i++)
{
```

```
AnalogOutput(Cad1236,i,double2bin(*uPtrs[i],rangeDA));
}
}
/* Finalização */
static void mdlTerminate(SimStruct *S)
{
}
#ifdef MATLAB_MEX_FILE                    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"                     /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"                      /* Code generation registration function */
#endif



/*
* WAITFORTREAL.C
*
* S-function to sinchronize simulation clock with real time
*
* RP Marques (26/02/1999)
*
*/
#define S_FUNCTION_NAME waitfortreal
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"
#include <time.h>
static void mdlInitializeSizes(SimStruct *S)
{
ssSetNumSFcnParams(S, 0);
if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
return; /* Parameter mismatch will be reported by Simulink */
}
if (!ssSetNumInputPorts(S, 0)) return;
if (!ssSetNumOutputPorts(S,3)) return;
ssSetOutputPortWidth(S, 0, 1);
      ssSetOutputPortWidth(S, 1, 1);
      ssSetOutputPortWidth(S, 2, 1);
ssSetNumSampleTimes(S, 1);
}
static void mdlInitializeSampleTimes(SimStruct *S)
{
ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
ssSetOffsetTime(S, 0, 0.0);
}
/* Function: mdlOutputs =======================================================
* Abstract:
* The function waits for the real time clock to catch up with the
* simulation clock. Notice that the simulation must run faster than real
* time for this to work. Notice also that The simulation time may evolve
* faster than real time when needed
*
*/
static void mdlOutputs(SimStruct *S, int_T tid)
{
```

```
real_T       *treal= ssGetOutputPortRealSignal(S,0);
real_T       *tsim = ssGetOutputPortRealSignal(S,1);
real_T        *td = ssGetOutputPortRealSignal(S,2);
real_T       t = ssGetT(S) - ssGetTStart(S);
real_T next_t;
static clock_t       start_time;
if (t == 0) {
start_time = clock();
}
      /* freeze simulation in this line */
while ( ((real_T)(clock()-start_time))/CLOCKS_PER_SEC < t );
      /* Outputs */
      /* Real time */
      treal[0] = ((real_T)(clock()-start_time))/CLOCKS_PER_SEC;
      /* Simulation time */
      tsim[0] = t;
      /* Estimated delay */
      td[0] = ((real_T)(clock()-start_time))/CLOCKS_PER_SEC - t;
}
/* Function: mdlTerminate =====================================================
* Abstract:
* No termination needed, but we are required to have this routine.
*/
static void mdlTerminate(SimStruct *S)
{
}
#ifdef MATLAB_MEX_FILE                                  /* Is this file being compiled as a MEX-file? */
#include "simulink.c"                                    /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"                                     /* Code generation registration function */
#endif
```

APÊNDICE G -Fluxograma e código-fonte em Matlab - conversor ponto fixo para ponto flutuante

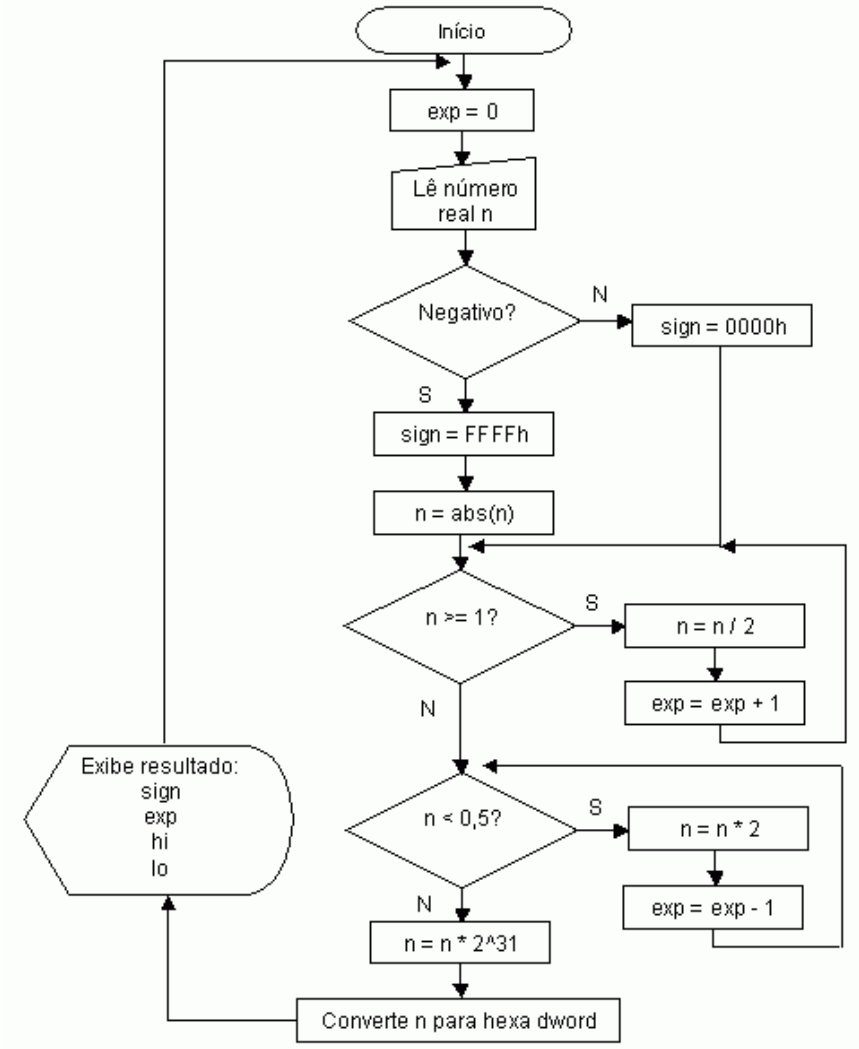


Figura G1 - Fluxograma de conversão ponto fixo para ponto flutuante

```
clear all;
clc
exp=0;
sign=0;
hi=0;
lo=0;
y=1;

while y==1

    exp=0;
```

```
    x = input('Digite o número: ');
        if x < 0
        sign=-1;
        x=abs(x);
        else
        sign=0;
        end
    while x>=1
    x=x/2;
    exp=exp+1;
    end
    while x<0.5
    x=x*2;
    exp=exp-1;
    end
x=x*2^31;
sign
exp
x=round(x);
x=dec2hex(x,8)

end
```

APÊNDICE H -Código-fonte em Assembly - VS01A + controlador PID

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * SENSOR VIRTUAL DE pH - VS-01A    *
; * MESTRADO - ESCOLA POLITÉCNICA DA USP   *
; * PTC - LAC - LABORATÓRIO DE AUTOMAÇÃO E CONTROLE  *
; * ORIENTADOR: PROF. DR. CLAUDIO GARCIA   *
; * DESENVOLVIDO POR CÁSSIO BERNI *
; * 12/2003 *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * HISTÓRICO *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
; - VS01_0.ASM ==> VERSÃO INICIAL - SENSOR VIRTUAL DE pH
; - VS01_1.ASM ==> FAMILIARIZAÇÃO COM INSTRUÇÕES PARA CÁLCULOS ARITMÉTICOS
; - VS01_2.ASM ==> IMPLEMENTAÇÃO DE ROTINAS DE CÁLCULOS DO MODELO
; - VS01_3.ASM ==> IMPLEMENTAÇÃO DO CÁLCULO DOS GRAUS DE ATIVAÇÃO DAS REGRAS
; - VS01_4.ASM ==> IMPLEMENTAÇÃO DO CÁLCULO DOS GRAUS DE ATIVAÇÃO DAS REGRAS - CONCLUSÃO DO MODELO
; - VS01_5.ASM ==> VERSÃO SEM INTERRUPÇÃO DE TIMER, PARA DEPURAÇÃO DO MODELO
; - VS01_6.ASM ==> IMPLEMENTAÇÃO DE ROTINAS EM PONTO FLUTUANTE –> USO EM ÁREAS CRÍTICAS DO MODELO (GRAUS DE ATIVAÇÃO)
; - VS01_7.ASM ==> CÁLCULO DOS GRAUS DE ATIVAÇÃO DAS REGRAS EM PONTO FLUTUANTE
; - VS01_8.ASM ==> MELHORIAS NO MODELO: CORREÇÃO DO NÚMERO "0" EM FP, ARREDONDAMENTOS
;                  E MAIOR PROTEÇÃO NA ROTINA FLOAT_TO_FIX
; - VS01_9.ASM ==> CORREÇÃO NA ROTINA DE DIVISÃO EM FP
; - VS01_10.ASM ==> VERSÃO VS01_9 COM INT. TIMER
; - VS01_11.ASM ==> VERSÃO VS01_9, PODENDO SELECIONAR FAIXA ÚTIL DE q3: 0,3..0,6 dm3/min OU 0..1 dm3/min
; - VS01_12.ASM ==> VERSÃO VS01_11 COM INT. TIMER
; - VS01_13.ASM ==> MELHORIA DA PRECISÃO MATEMÁTICA EM GERAL: MODELO INTEGRAL EM FP
; - VS01_14.ASM ==> VERSÃO VS01_13 COM INT. TIMER
; - VS01_15.ASM ==> VERSÃO VS01_14 EM FLASH
; - VS01_16.ASM ==> ALTERAÇÃO DA TAXA DE AMOSTRAGEM: 166,67 ms (A MESMA DA PLANTA)
; - VS01_17.ASM ==> VERSÃO VS01_12 (PONTO FIXO) EM FLASH E COM TAXA DE AMOSTRAGEM: 166,67 ms (A MESMA DA PLANTA)
; - VS01_18.ASM ==> VERSÃO VS01_16 COM CONTROLADOR PID EMBUTIDO
; - VS01_19.ASM ==> VERSÃO VS01_16 COM CONTROLADOR PID EMBUTIDO - SET-POINT VARIÁVEL

; OBSERVAÇÃO:
; PARA QUE O DEBUGGER (MONITOR) FUNCIONE, É NECESSÁRIO HABILITAR A
; INT. TXRXINT (COMUNICAÇÃO SERIAL ASSÍNCRONA)

; DADOS DO MODELO DE NEUTRALIZAÇÃO DE pH:
; Ts = INTERVALO DE AMOSTRAGEM = 1/6 SEG = 166,67 ms (DO PONTO DE VISTA DA PLANTA REAL, SERIA 1/6 MIN.)
; VAZÃO NOMINAL DE BASE q3 = 0,4684 dm3/min
; pH NOMINAL = 7

; VARIÁVEIS EM FP:
; XSIGN = SINAL: 0000h (+) OU FFFFh (-)
; xEXP = EXPOENTE: -127..+128
; xHI = MANTISSA HIGH
; xLO = MANTISSA LOW
; MANTISSA: FORMATO FRACIONÁRIO Q31 POSITIVO. PARA Q15, CARREGAR VALOR EM MANTISSA HIGH E ZERAR MANTISSA LOW

; DADOS DO CONTROLADOR PID:
; UTILIZA A MESMA SINTONIA DO PID EM MATLAB:
; KP = 0,1
; KI = 0,01
; KD = 0
; POSSUI MODO INTEGRAL COM ANTI-WINDUP. Tt = T
; N = 5 (MÁXIMO GANHO DERIVATIVO)

; OBSERVAÇÃO COM RELAÇÃO AO COMPILADOR TASM:
; HÁ UM ERRO NA COMPILAÇÃO DA INSTRUÇÃO SPLK #CTE,DMA
; PARA AS PÁGINAS 5 E 7 DA RAM DE DADOS (280H..2FFH E 380H..3FFH)
; UTILIZAR APENAS ENDEREÇAMENTO INDIRETO NESTAS PÁGINAS
```

```
.TITLE "SENSOR VIRTUAL DE pH - VS01A + PID - VERSÃO DE SOFTWARE: 0.0"
.INCLUDE "..\\..\\INCLUDE\\PATHWAY.INC"

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * CONSTANTES INTERNAS *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
TIMSTOP     .SET 0010H      ; ESCREVENDO ESTE VALOR NO TCR PÁRA O TIMER
TIMCFG0     .SET 0030H      ; ESCREVENDO ESTE VALOR NO TCR RECARREGA E PÁRA O TIMER
TIMCFG1     .SET 0000H      ; ESCREVENDO ESTE VALOR NO TCR LIGA O TIMER
TDDRVAL     .SET 000FH      ; TDDR (4-BITS) –> PRESCALER = 1:16
PRDVAL      .SET 5160H      ; PRD (16-BITS) –> PERÍODO

; TAXA DA INT. TIMER = FCLKOUT1 / [ (TDDR + 1) x (PRD + 1) ]
; = 60 Hz = 16,667 ms

VALOR_CONT    .SET 10       ; VALOR A SER CARREGADO NO CONTADOR DE INT.TIMER
T_MUDA_SP     .SET 804      ; A CADA T_MUDA_SP INTERRUPÇÕES, CHAMA ROTINA MUDA_SP

UM_SIGN     .SET 0000H
UM_EXP      .SET 1
UM_HI       .SET 4000H
UM_LO       .SET 0000H      ; 1 EM FP

; FATOR PARA CONVERSÃO pH(n) –> D/A, SEM SATURAÇÃO PARA pH(n) > 10
FAT_DA_SIGN     .SET 0000H
FAT_DA_EXP      .SET -6
FAT_DA_HI       .SET 6666H
FAT_DA_LO       .SET 6666H      ; 0,0125 EM FP - FATOR PARA CONVERSÃO pH(n) –> D/A

; FATOR PARA CONVERSÃO pH(n) –> D/A, COM SATURAÇÃO PARA pH(n) > 10
FAT1_DA_SIGN    .SET 0000H
FAT1_DA_EXP     .SET -3
FAT1_DA_HI      .SET 6666H
FAT1_DA_LO      .SET 6666H      ; 0,1 EM FP - FATOR PARA CONVERSÃO pH(n) –> D/A

; CONSTANTES PARA O CONTROLADOR PID:
KC_SIGN     .SET 0000H
KC_EXP      .SET -3
KC_HI       .SET 6666H
KC_LO       .SET 6666H          ; kc = 0,1 –> GANHO PROPORCIONAL

AD_SIGN     .SET 0000H
AD_EXP      .SET 0
AD_HI       .SET 0000H
AD_LO       .SET 0000H          ; ad = Td/(N*T + Td) = 0

BD_SIGN     .SET 0000H
BD_EXP      .SET 0
BD_HI       .SET 0000H
BD_LO       .SET 0000H          ; bd = (kc*N*Td)/(N*T + Td) = 0

BI_SIGN     .SET 0000H
BI_EXP      .SET -9
BI_HI       .SET 6D3AH
BI_LO       .SET 06D4H          ; bi = (kc*T)/Ti = 0,001667

BT_SIGN     .SET 0000H
BT_EXP      .SET 1
BT_HI       .SET 4000H
BT_LO       .SET 0000H          ; bt = T/Tt = 1

UMAX_SIGN     .SET 0000H
UMAX_EXP      .SET 0
UMAX_HI       .SET 440BH
UMAX_LO       .SET 7803H        ; umax = 0,5316

UMIN_SIGN     .SET 0FFFFH
```

```
UMIN_EXP        .SET -1
UMIN_HI         .SET 77E9H
UMIN_LO         .SET 0FF9H          ; umin = -0,4684

N_SIGN      .SET 0000H
N_EXP       .SET 3
N_HI        .SET 5000H
N_LO        .SET 0000H              ; N = 5

OFFSET_Q3_SIGN      .SET 0000H
OFFSET_Q3_EXP       .SET -1
OFFSET_Q3_HI        .SET 77E9H
OFFSET_Q3_LO        .SET 0FF9H  ; OFFSET_Q3 = 0,4684

SP_INI_SIGN     .SET 0000H
SP_INI_EXP      .SET 3
SP_INI_HI       .SET 7000H
SP_INI_LO       .SET 0000H          ; SET-POINT INICIAL = 7.0

SP_MAX_SIGN     .SET 0000H
SP_MAX_EXP      .SET 4
SP_MAX_HI       .SET 4C00H
SP_MAX_LO       .SET 0000H      ; SET-POINT MÁXIMO = 9.5

SP_MIN_SIGN     .SET 0000H
SP_MIN_EXP      .SET 3
SP_MIN_HI       .SET 4000H
SP_MIN_LO       .SET 0000H      ; SET-POINT MÍNIMO = 4.0

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * DEFINIÇÃO DAS VARIÁVEIS *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; MANTER ORDENAÇÃO EXATA (ENDEREÇAMENTO INDIRETO)
; 200H..27FH –> DATA PAGE 4:
; BUFFER AUXILIAR:
BUF_SIGN    .SET 200H
BUF_EXP     .SET 201H
BUF_HI      .SET 202H
BUF_LO      .SET 203H

; CONVERSÕES A/D E D/A:
A_D     .SET 204H
D_A     .SET 205H

; RESERVADA (UTILIZADA PELO MONITOR):
XXXX    .SET 206H

; AUXILIARES DE USO GERAL:
AUX     .SET 207H
AUX2    .SET 208H

; REGRESSORES - EM FP:
Q3_N_SIGN   .SET 209H
Q3_N_EXP    .SET 20AH
Q3_N_HI     .SET 20BH
Q3_N_LO     .SET 20CH           ; Q3(N)

Q3_N1_SIGN      .SET 20DH
Q3_N1_EXP       .SET 20EH
Q3_N1_HI        .SET 20FH
Q3_N1_LO        .SET 210H           ;Q3(N-1) - INÍCIO = 0,4684

Q3_N2_SIGN      .SET 211H
Q3_N2_EXP       .SET 212H
Q3_N2_HI        .SET 213H
Q3_N2_LO        .SET 214H           ; Q3(N-2) - INÍCIO = 0,4684
```

```
Q3_N3_SIGN      .SET 215H
Q3_N3_EXP       .SET 216H
Q3_N3_HI        .SET 217H
Q3_N3_LO        .SET 218H           ; Q3(N-3) - INÍCIO = 0,4684

Q3_N4_SIGN      .SET 219H
Q3_N4_EXP       .SET 21AH
Q3_N4_HI        .SET 21BH
Q3_N4_LO        .SET 21CH           ; Q3(N-4) - INÍCIO = 0,4684

PH_N1_SIGN      .SET 21DH
PH_N1_EXP       .SET 21EH
PH_N1_HI        .SET 21FH
PH_N1_LO        .SET 220H           ; PH(N-1) - INÍCIO = 7

PH_N2_SIGN      .SET 221H
PH_N2_EXP       .SET 222H
PH_N2_HI        .SET 223H
PH_N2_LO        .SET 224H           ; PH(N-2) - INÍCIO = 7

PH_N_SIGN       .SET 225H
PH_N_EXP        .SET 226H
PH_N_HI         .SET 227H
PH_N_LO         .SET 228H           ; PH(N)

; VETOR Z = (U-V). É TAMBÉM PARTE DOS GRAUS DE ATIVAÇÃO:
Z1_Q1_SIGN      .SET 229H
Z1_Q1_EXP       .SET 22AH
Z1_Q1_HI        .SET 22BH
Z1_Q1_LO        .SET 22CH

Z2_Q2_SIGN      .SET 22DH
Z2_Q2_EXP       .SET 22EH
Z2_Q2_HI        .SET 22FH
Z2_Q2_LO        .SET 230H

Z3_Q3_SIGN      .SET 231H
Z3_Q3_EXP       .SET 232H
Z3_Q3_HI        .SET 233H
Z3_Q3_LO        .SET 234H

Z4_Q4_SIGN      .SET 235H
Z4_Q4_EXP       .SET 236H
Z4_Q4_HI        .SET 237H
Z4_Q4_LO        .SET 238H

Z5_Q5_SIGN      .SET 239H
Z5_Q5_EXP       .SET 23AH
Z5_Q5_HI        .SET 23BH
Z5_Q5_LO        .SET 23CH

Z6_Q6_SIGN      .SET 23DH
Z6_Q6_EXP       .SET 23EH
Z6_Q6_HI        .SET 23FH
Z6_Q6_LO        .SET 240H

; CONTINUAÇÃO DOS GRAUS DE ATIVAÇÃO:
Q7_SIGN     .SET 241H
Q7_EXP      .SET 242H
Q7_HI       .SET 243H
Q7_LO       .SET 244H

Q8_SIGN     .SET 245H
Q8_EXP      .SET 246H
Q8_HI       .SET 247H
Q8_LO       .SET 248H
```

```
Q9_SIGN      .SET 249H
Q9_EXP       .SET 24AH
Q9_HI        .SET 24BH
Q9_LO        .SET 24CH

Q10_SIGN     .SET 24DH
Q10_EXP      .SET 24EH
Q10_HI       .SET 24FH
Q10_LO       .SET 250H

Q11_SIGN     .SET 251H
Q11_EXP      .SET 252H
Q11_HI       .SET 253H
Q11_LO       .SET 254H

Q12_SIGN     .SET 255H
Q12_EXP      .SET 256H
Q12_HI       .SET 257H
Q12_LO       .SET 258H

Q13_SIGN     .SET 259H
Q13_EXP      .SET 25AH
Q13_HI       .SET 25BH
Q13_LO       .SET 25CH

Q14_SIGN     .SET 25DH
Q14_EXP      .SET 25EH
Q14_HI       .SET 25FH
Q14_LO       .SET 260H

Q15_SIGN     .SET 261H
Q15_EXP      .SET 262H
Q15_HI       .SET 263H
Q15_LO       .SET 264H

Q16_SIGN     .SET 265H
Q16_EXP      .SET 266H
Q16_HI       .SET 267H
Q16_LO       .SET 268H

Q17_SIGN     .SET 269H
Q17_EXP      .SET 26AH
Q17_HI       .SET 26BH
Q17_LO       .SET 26CH

Q18_SIGN     .SET 26DH
Q18_EXP      .SET 26EH
Q18_HI       .SET 26FH
Q18_LO       .SET 270H

Q19_SIGN     .SET 271H
Q19_EXP      .SET 272H
Q19_HI       .SET 273H
Q19_LO       .SET 274H

Q20_SIGN     .SET 275H
Q20_EXP      .SET 276H
Q20_HI       .SET 277H
Q20_LO       .SET 278H

; CONTADOR PARA INT. TIMER = 166,67 ms:
CONT_TIMER     .SET 279H

; CONTADORES PARA GERAÇÃO DO SP VARIÁVEL:
CONT_FASE    .SET 27AH
FASE         .SET 27BH
```

```
;*****************************************************
; 300H..37FH –> DATA PAGE 6:
; AUXILIARES GERAIS:
TEMP1       .SET 300H
TEMP2       .SET 301H
TEMP3       .SET 302H
TEMP4       .SET 303H
TEMP5       .SET 304H
TEMP6       .SET 305H

; RESERVADA (UTILIZADA PELO MONITOR):
KKKKK       .SET 306H

; VARIÁVEIS DAS ROTINAS DE CÁLCULO EM FP:
ASIGN       .SET 307H
AEXP        .SET 308H
AHI         .SET 309H
ALO         .SET 30AH       ; "A" É UTILIZADA COMO PRIMEIRO ARGUMENTO EM TODAS AS ROTINAS FP

BSIGN       .SET 30BH
BEXP        .SET 30CH
BHI         .SET 30DH
BLO         .SET 30EH       ; "B" É UTILIZADA COMO SEGUNDO ARGUMENTO EM TODAS AS ROTINAS FP

CSIGN       .SET 30FH
CEXP        .SET 310H
CHI         .SET 311H
CLO         .SET 312H       ; C" É O RESULTADO EM TODAS AS ROTINAS FP

DSIGN       .SET 313H
DEXP        .SET 314H
DHI         .SET 315H
DLO         .SET 316H       ; "D" –> BUFFER TEMPORÁRIO

ESIGN       .SET 317H
EEXP        .SET 318H
EHI         .SET 319H
ELO         .SET 31AH       ; "E" –> BUFFER TEMPORÁRIO

; AUXILIARES PARA AS ROTINAS DE PONTO FLUTUANTE:
D           .SET 31BH
TEMP    .SET 31CH
RESID   .SET 31DH
THI     .SET 31EH
TLO     .SET 31FH
QM      .SET 320H
QL      .SET 321H
R1      .SET 322H
R2      .SET 323H
CL      .SET 324H

; NORMAS QUADRÁTICAS = [(U-V) X (U-V)´]. SÃO TAMBÉM AS SAÍDAS PARCIAIS Y1..Y20:
Y1_D1Q_SIGN     .SET 325H
Y1_D1Q_EXP      .SET 326H
Y1_D1Q_HI       .SET 327H
Y1_D1Q_LO       .SET 328H       ; REFERENTE AO CLUSTER 1

Y2_D2Q_SIGN     .SET 329H
Y2_D2Q_EXP      .SET 32AH
Y2_D2Q_HI       .SET 32BH
Y2_D2Q_LO       .SET 32CH       ; REFERENTE AO CLUSTER 2

Y3_D3Q_SIGN     .SET 32DH
Y3_D3Q_EXP      .SET 32EH
Y3_D3Q_HI       .SET 32FH
Y3_D3Q_LO       .SET 330H       ; REFERENTE AO CLUSTER 3

Y4_D4Q_SIGN     .SET 331H
```

```
Y4_D4Q_EXP      .SET 332H
Y4_D4Q_HI       .SET 333H
Y4_D4Q_LO       .SET 334H      ; REFERENTE AO CLUSTER 4

Y5_D5Q_SIGN     .SET 335H
Y5_D5Q_EXP      .SET 336H
Y5_D5Q_HI       .SET 337H
Y5_D5Q_LO       .SET 338H      ; REFERENTE AO CLUSTER 5

Y6_D6Q_SIGN     .SET 339H
Y6_D6Q_EXP      .SET 33AH
Y6_D6Q_HI       .SET 33BH
Y6_D6Q_LO       .SET 33CH      ; REFERENTE AO CLUSTER 6

Y7_D7Q_SIGN     .SET 33DH
Y7_D7Q_EXP      .SET 33EH
Y7_D7Q_HI       .SET 33FH
Y7_D7Q_LO       .SET 340H      ; REFERENTE AO CLUSTER 7

Y8_D8Q_SIGN     .SET 341H
Y8_D8Q_EXP      .SET 342H
Y8_D8Q_HI       .SET 343H
Y8_D8Q_LO       .SET 344H      ; REFERENTE AO CLUSTER 8

Y9_D9Q_SIGN     .SET 345H
Y9_D9Q_EXP      .SET 346H
Y9_D9Q_HI       .SET 347H
Y9_D9Q_LO       .SET 348H      ; REFERENTE AO CLUSTER 9

Y10_D10Q_SIGN    .SET 349H
Y10_D10Q_EXP     .SET 34AH
Y10_D10Q_HI      .SET 34BH
Y10_D10Q_LO      .SET 34CH      ; REFERENTE AO CLUSTER 10

Y11_D11Q_SIGN    .SET 34DH
Y11_D11Q_EXP     .SET 34EH
Y11_D11Q_HI      .SET 34FH
Y11_D11Q_LO      .SET 350H      ; REFERENTE AO CLUSTER 11

Y12_D12Q_SIGN    .SET 351H
Y12_D12Q_EXP     .SET 352H
Y12_D12Q_HI      .SET 353H
Y12_D12Q_LO      .SET 354H      ; REFERENTE AO CLUSTER 12

Y13_D13Q_SIGN    .SET 355H
Y13_D13Q_EXP     .SET 356H
Y13_D13Q_HI      .SET 357H
Y13_D13Q_LO      .SET 358H      ; REFERENTE AO CLUSTER 13

Y14_D14Q_SIGN    .SET 359H
Y14_D14Q_EXP     .SET 35AH
Y14_D14Q_HI      .SET 35BH
Y14_D14Q_LO      .SET 35CH      ; REFERENTE AO CLUSTER 14

Y15_D15Q_SIGN    .SET 35DH
Y15_D15Q_EXP     .SET 35EH
Y15_D15Q_HI      .SET 35FH
Y15_D15Q_LO      .SET 360H      ; REFERENTE AO CLUSTER 15

Y16_D16Q_SIGN    .SET 361H
Y16_D16Q_EXP     .SET 362H
Y16_D16Q_HI      .SET 363H
Y16_D16Q_LO      .SET 364H      ; REFERENTE AO CLUSTER 16

Y17_D17Q_SIGN    .SET 365H
Y17_D17Q_EXP     .SET 366H
Y17_D17Q_HI      .SET 367H
```

```
Y17_D17Q_LO       .SET 368H     ; REFERENTE AO CLUSTER 17

Y18_D18Q_SIGN     .SET 369H
Y18_D18Q_EXP      .SET 36AH
Y18_D18Q_HI       .SET 36BH
Y18_D18Q_LO       .SET 36CH     ; REFERENTE AO CLUSTER 18

Y19_D19Q_SIGN     .SET 36DH
Y19_D19Q_EXP      .SET 36EH
Y19_D19Q_HI       .SET 36FH
Y19_D19Q_LO       .SET 370H     ; REFERENTE AO CLUSTER 19

Y20_D20Q_SIGN     .SET 371H
Y20_D20Q_EXP      .SET 372H
Y20_D20Q_HI       .SET 373H
Y20_D20Q_LO       .SET 374H     ; REFERENTE AO CLUSTER 20

;*****************************************************************
; 380H..3FFH –> DATA PAGE 7 (UTILIZAR SOMENTE ENDEREÇAMENTO INDIRETO):
SP_SIGN     .SET 380H
SP_EXP      .SET 381H
SP_HI       .SET 382H
SP_LO       .SET 383H     ; SET-POINT

PN_SIGN     .SET 387H
PN_EXP      .SET 388H
PN_HI       .SET 389H
PN_LO       .SET 38AH     ; TERMO PROPORCIONAL

IN_SIGN     .SET 38BH
IN_EXP      .SET 38CH
IN_HI       .SET 38DH
IN_LO       .SET 38EH     ; TERMO INTEGRAL

DN_SIGN     .SET 38FH
DN_EXP      .SET 390H
DN_HI       .SET 391H
DN_LO       .SET 392H     ; TERMO DERIVATIVO

U_SIGN      .SET 393H
U_EXP       .SET 394H
U_HI        .SET 395H
U_LO        .SET 396H     ; VARIÁVEL MANIPULADA (SAÍDA DO ATUADOR, COM SATURAÇÃO)

V_SIGN      .SET 397H
V_EXP       .SET 398H
V_HI        .SET 399H
V_LO        .SET 39AH     ; V = P + I + D (ENTRADA DO ATUADOR, SEM SATURAÇÃO)

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * CONFIGURAÇÕES INICIAIS *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

    PS 4000H    ; ENDEREÇO 4000H NA MEMÓRIA DE PROGRAMA - INÍCIO DA PROGRAM FLASH
                ; VÁLIDO PARA SOFTWARES RODANDO EM VERSÃO FLASH

CONFIG:
    SETC INTM   ; INTM = 1 –> INTERRUPÇÕES MASCARÁVEIS DESABILITADAS
    CLRC CNF    ; CNF = 0 –> MAPEIA DARAM B0 COMO MEMÓRIA DE DADOS (0X200-0X2FF)
    SETC SXM    ; SMX = 1 –> UTILIZA ARITMÉTICA COM EXTENSÃO DE SINAL
    SETC OVM    ; OVM = 1 –> HABILITA SATURAÇÃO DO ACUMULADOR
    SPM 1       ; PM = 01 –> ULA REMOVE AUTOMATICAMENTE O SINAL EXTRA DAS CONTAS DE MULTIPLICAÇÃO EM MODO FRACIONÁRIO
Q15

    LDP #04H        ; ACERTA DATA PAGE POINTER PARA ÁREA DE RAM 200H..27FH
                    ; ENDEREÇAMENTO DIRETO
                    ; MANTÉM 7 WAIT STATES PARA ACESSO A I/O EXTERNO
```

```
                    ; (WSGR –> ESTADO DEFAULT NO RESET)
    CALL INITVECS     ; INSTALA A TABELA SECUNDÁRIA DE VETORES DE INT. EM 8000h
                      ; NA MEMÓRIA DE PROGRAMA

; CONFIGURA O TIMER:
    SPLK #TIMSTOP,AUX
    OUT AUX,TCR           ; PÁRA O TIMER
    SPLK #PRDVAL,AUX
    OUT AUX,PRD           ; INICIALIZA PERÍODO
    SPLK #TIMCFG1 | TDDRVAL,AUX
    OUT AUX,TCR           ; CONFIGURA PRESCALER E LIGA O TIMER

; INICIALIZA VARIÁVEIS EM GERAL:
    SPLK #VALOR_CONT,CONT_TIMER     ; INICIALIZA CONTADOR PARA INT. TIMER
    SPLK #T_MUDA_SP,CONT_FASE       ; INICIALIZA CONTADOR PARA FASES DO SET-POINT VARIÁVEL
    SPLK #0,FASE                    ; INICIALIZA FASE
    SPLK #0000H,BUF_SIGN
    SPLK #-1,BUF_EXP
    SPLK #77E9H,BUF_HI
    SPLK #0FF9H,BUF_LO          ; BUF <– 0,4684 EM FP
    LAR AR0,#BUF_SIGN
    LAR AR1,#Q3_N1_SIGN
    CALL TRANSFERE              ; INICIALIZA q3(n-1)
    LAR AR0,#BUF_SIGN
    LAR AR1,#Q3_N2_SIGN
    CALL TRANSFERE              ; INICIALIZA q3(n-2)
    LAR AR0,#BUF_SIGN
    LAR AR1,#Q3_N3_SIGN
    CALL TRANSFERE              ; INICIALIZA q3(n-3)
    LAR AR0,#BUF_SIGN
    LAR AR1,#Q3_N4_SIGN
    CALL TRANSFERE              ; INICIALIZA q3(n-4)
    SPLK #0000H,BUF_SIGN
    SPLK #3,BUF_EXP
    SPLK #7000H,BUF_HI
    SPLK #0000H,BUF_LO          ; BUF <– 7 EM FP
    LAR AR0,#BUF_SIGN
    LAR AR1,#PH_N1_SIGN
    CALL TRANSFERE              ; INICIALIZA pH(n-1)
    LAR AR0,#BUF_SIGN
    LAR AR1,#PH_N2_SIGN
    CALL TRANSFERE              ; INICIALIZA pH(n-2)
; PID:
; y(n-1) = pH(n-1) JÁ INICIALIZADO
    SPLK #0000H,BUF_SIGN
    SPLK #0,BUF_EXP
    SPLK #0000H,BUF_HI
    SPLK #0000H,BUF_LO          ; BUF <– 0 EM FP
    LAR AR0,#BUF_SIGN
    LAR AR1,#DN_SIGN
    CALL TRANSFERE              ; INICIALIZA D(n-1)
    LAR AR0,#BUF_SIGN
    LAR AR1,#IN_SIGN
    CALL TRANSFERE              ; INICIALIZA I(n)
    SPLK #SP_INI_SIGN,BUF_SIGN
    SPLK #SP_INI_EXP,BUF_EXP
    SPLK #SP_INI_HI,BUF_HI
    SPLK #SP_INI_LO,BUF_LO      ; BUF <– SET-POINT INICIAL EM FP
    LAR AR0,#BUF_SIGN
    LAR AR1,#SP_SIGN
    CALL TRANSFERE              ; INICIALIZA SET-POINT

; HABILITA INTERRUPÇÕES MASCARÁVEIS DESEJADAS:
    LDP #0H                     ; DATA PAGE = 0 (REGISTROS DE CONFIGURAÇÃO)
;*** SPLK #EN_TINT | EN_TXRXINT,IMR   ; HABILITA A INT. TIMER E COMUNICAÇÃO ASSÍNCRONA (MONITOR)
    SPLK #EN_TINT,IMR           ; HABILITA A INT. TIMER
    CLRC INTM                   ; HABILITAÇÃO GLOBAL DAS INTERRUPÇÕES
    LDP #04H                    ; ACERTA DATA PAGE POINTER PARA ÁREA DE RAM 200H..27FH

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
; * PROGRAMA PRINCIPAL *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
LOOP:
     IDLE              ; DORME –> MODO LOW POWER
     B LOOP        ; APÓS ACORDAR POR INT., VOLTA A DORMIR

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * TRATAMENTOS DE INTERRUPÇÕES *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;******** TIMER - OCORRE A CADA 16,67 ms
TIMERISR:
; CONTADOR DE INTERRUPÇÕES:
     LACC CONT_TIMER
     SUB #1
     BCND EXECUTA_MODELO,EQ          ; SE CONT_TIMER = 0 –> T = 10 x Tint. = 166,67 ms –> TRATA MODELO
     SACL CONT_TIMER                 ; DECREMENTA CONT_TIMER
     B FIM_TIMERISR                  ; FIM

EXECUTA_MODELO:
     SPLK #VALOR_CONT,CONT_TIMER       ; RECARREGA CONT_TIMER

; AMOSTRA q3(n), SÓ SERÁ UTILIZADA NA PRÓXIMA ITERAÇÃO:
     CALL LE_AD              ; FAZ CONVERSÃO A/D
     CALL CALC_Q3        ; CALCULA q3(n) EM FP

; CÁLCULO DO VETOR Z = (U - V) E DE SUA NORMA QUADRÁTICA PARA CADA CLUSTER:
     CALL CALC_Z_NORMA       ; CALCULA Z = (U - V) E A SUA NORMA QUADRÁTICA
                             ; PARA CADA CLUSTER, SALVANDO OS RESULTADOS EM FP

; CÁLCULO DOS GRAUS DE ATIVAÇÃO DE CADA REGRA:
     CALL CALC_Q             ; CALCULA GRAUS DE ATIVAÇÃO

; CÁLCULO DAS SAÍDAS PARCIAIS:
     CALL CALC_S_PARCIAIS       ; CALCULA SAÍDAS PARCIAIS

; CÁLCULO DA SAÍDA GLOBAL pH(N):
     CALL CALC_S_GLOBAL        ; CALCULA SAÍDA GLOBAL: pH(n)

; ALGORITMO PID:
     CALL PID                ; CONTROLADOR PID

; ATUALIZAÇÃO DO D/A:
     CALL PID_TO_DA          ; PREPARA VALOR PARA D/A
     CALL ESCREVE_DA         ; ATUALIZA D/A

; ATUALIZAÇÃO DOS REGRESSORES:
     CALL ATUALIZA_REGR        ; ATUALIZA REGRESSORES

; CHECA MUDANÇA DO SET-POINT:
     LACC CONT_FASE
     SUB #1
     BCND ALTERA_SP,EQ         ; SE CONT_FASE = 0 –> ALTERAR SET-POINT
     SACL CONT_FASE            ; DECREMENTA CONT_FASE
     B FIM_TIMERISR            ; FIM

ALTERA_SP:
     SPLK #T_MUDA_SP,CONT_FASE       ; RECARREGA CONT_FASE
     CALL MUDA_SP                    ; ALTERA SP, DE ACORDO COM A FASE
FIM_TIMERISR:
     LDP #0H                         ; PÁGINA 0 DA RAM DE DADOS
     CLRC INTM                       ; REABILITA INTERRUPÇÕES MASCARÁVEIS
     SPLK #CLR_TINT,IFR              ; ZERA FLAG - INTERRUPÇÃO TRATADA
     LDP #04H                        ; ACERTA DATA PAGE POINTER PARA ÁREA DE RAM 200H..27FH
     RET                             ; FIM

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * SUBROTINAS *
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * CONVERSÃO A/D *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
LE_AD:
    CLRC XF                         ; CHIP SELECT - A/D
    NOP
    SPLK #0000H,AUX
    OUT AUX,0000H                   ; DISPARA UMA CONVERSÃO A/D
    NOP
    NOP                             ; DELAY PARA GARANTIR LEITURA DO BUSY = 0 (INíCIO DE CONVERSÃO)
WAIT_END_CONV:
    IN AUX,IOSR                     ; LÊ PINOS DE I/O
    LACL AUX                        ; ACCL <– AUX
    AND #0001H                      ; TESTA BIT 0 DE IOSR (I/O ZERO)
    BCND WAIT_END_CONV,EQ           ; ENQUANTO I/O ZERO = 0 –> BUSY = 0 –> CONVERSÃO EM ANDAMENTO
; FIM DA CONVERSÃO
    NOP
    NOP
    IN AUX,0000H                    ; LÊ VALOR DA CONVERSÃO
    LACL AUX                        ; ACCL <– AUX
    AND #0FFFH                      ; MÁSCARA PARA ZERAR 4 BITS MAIS SIGNIFICATIVOS (12 BITS ÚTEIS)
    SACL A_D                        ; A_D <– ACCL. A_D CONTÉM VALOR DA CONVERSÃO
    RET                             ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * CONVERSÃO D/A *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
ESCREVE_DA:
    SETC XF                         ; CHIP SELECT - D/A
    NOP
    LACL D_A                        ; ACCL <– D_A
    AND #0FFFH                      ; MÁSCARA PARA 12 BITS MENOS SIGNIFICATIVOS
    SACL D_A                        ; D_A <– ACCL
    OUT D_A,0000H                   ; ESCREVE VALOR DESEJADO NO D/A
    RET                             ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * TRANSFERÊNCIA DE UM COEFICIENTE DE UM SUBMODELO LINEAR PARA O "A" *
; * PASSAR ORIGEM NO ACCL *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
SUBMODELO_TO_A:
    MAR *,AR0
    LAR AR0,#ASIGN
    RPT #3
    TBLR *+
    RET

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * TRANSFERÊNCIA DE UM COEFICIENTE DE UM CLUSTER PARA O "B" *
; * PASSAR ORIGEM NO ACCL *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
CLUSTER_TO_B:
    MAR *,AR0
    LAR AR0,#BSIGN
    RPT #3
    TBLR *+
    RET

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * TRANSFERÊNCIA GENÉRICA EM RAM - NÚMEROS EM FP *
; * PASSAR ORIGEM NO AR0 *
; * PASSAR DESTINO NO AR1 *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
TRANSFERE:
    LAR AR2,#3
    MAR *,AR0
    REP_TRANSF:
    LACL *+,AR1
```

```
    SACL *+,0,AR2
    BANZ REP_TRANSF,*-,AR0
    RET

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * CÁLCULO DE q3(n) *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
CALC_Q3:
    LACC A_D,16                 ; ACCH <– A_D
    LDP #06
    CALL FIX_TO_FLOAT           ; CONVERTE A/D PARA FP
    SPLK #0000H,BSIGN
    SPLK #04H,BEXP
    SPLK #4000H,BHI
    SPLK #0000H,BLO             ; B <– 8 EM FP
    CALL FLOAT_MULT             ; C <– A_D x 8
    LAR AR0,#CSIGN
    LAR AR1,#Q3_N_SIGN
    CALL TRANSFERE              ; SALVA q3(n)
    LDP #04
    RET

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * CÁLCULO DO VETOR Z = (U - V) E DE SUA NORMA QUADRÁTICA *
; * PARA TODOS OS CLUSTERS *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
CALC_Z_NORMA:
    LDP #06H
    LAR AR6,#Y1_D1Q_SIGN        ; INICIALIZA AR6 –> PONTEIRO DOS RESULTADOS DAS NORMAS QUADRÁTICAS
    LAR AR4,#19                 ; LOOP PARA MACRO REPETIÇÕES: 20 CLUSTERS
    LACC #Q3_N1_C1_SIGN
    SACL TEMP3                  ; TEMP3: PONTEIRO DE V –> BASTA INICIALIZÁ-LO AQUI
PROX_CLUSTER:
    LAR AR3,#5                  ; LOOP DE 6 REPETIÇÕES: TAMANHO DO VETOR Z
    LACC #Q3_N1_SIGN
    SACL TEMP2                  ; TEMP2: PONTEIRO DE U
    LACC #Z1_Q1_SIGN
    SACL TEMP4                  ; TEMP4: PONTEIRO DE Z
REP_Z:
    LAR AR0,TEMP2
    LAR AR1,#ASIGN
    CALL TRANSFERE              ; A <– COMPONENTE DO VETOR U
    LACL TEMP3
    CALL CLUSTER_TO_B           ; B <– COMPONENTE DO VETOR V
    CALL FLOAT_SUB              ; C <– A - B
    LAR AR0,#CSIGN
    LAR AR1,TEMP4
    CALL TRANSFERE              ; SALVA COMPONENTE DO VETOR Z
    LACL TEMP2
    ADD #4
    SACL TEMP2                  ; INCREMENTA PONTEIRO DE U
    LACL TEMP3
    ADD #4
    SACL TEMP3                  ; INCREMENTA PONTEIRO DE V
    LACL TEMP4
    ADD #4
    SACL TEMP4                  ; INCREMENTA PONTEIRO DE Z
    MAR *,AR3
    BANZ REP_Z                  ; LOOP PARA CÁLCULO DO VETOR Z = U - V

; CALCULA NORMA QUADRÁTICA EM PONTO FLUTUANTE:
    LACL #0H
    CALL FIX_TO_FLOAT
    CALL MOVAD                  ; INICIALIZA D = 0 EM FP
    LAR AR3,#5                  ; CONTADOR PARA 6 REPETIÇÕES
    LACC #Z1_Q1_SIGN
    SACL TEMP4                  ; TEMP4: PONTEIRO DE Z
LOOP_DQ:
    LAR AR0,TEMP4
    LAR AR1,#ASIGN
```

```
    CALL TRANSFERE          ; A <– COMPONENTE DO VETOR Z
    CALL MOVAB              ; B <– Zx
    CALL FLOAT_MULT         ; C <– Zx^2
    CALL MOVDA              ; A <– sum (Zi^2) , i = 1..(x-1) , PARA x = 1 –> A = 0
    CALL MOVCB              ; B <– Zx^2
    CALL FLOAT_ADD          ; C <– sum (Zi^2) , i = 1..x
    CALL MOVCD              ; D <– sum (Zi^2) , i = 1..x
    LACL TEMP4
    ADD #4
    SACL TEMP4              ; INCREMENTA PONTEIRO DE Z
    MAR *,AR3
    BANZ LOOP_DQ            ; AO FINAL DO LOOP: D <– Z1^2 + Z2^2 + Z3^2 + Z4^2 + Z5^2 + Z6^2
    LAR AR2,#3              ; CONTADOR PARA A TRANSFERÊNCIA DE D (FP) PARA DxQ (FP)
    LAR AR0,#DSIGN          ; AR0: PONTEIRO DE D
    MAR *,AR0
TRANSFERE_DQ:
    LACC *+,0,AR6           ; AR6 –> PONTEIRO DE DESTINO: DxQ
    SACL *+,0,AR2
    BANZ TRANSFERE_DQ,*-,AR0    ; CARREGA D EM DxQ
    MAR *,AR4
    BANZ PROX_CLUSTER           ; REPETE PROCEDIMENTO PARA TODOS OS CLUSTERS
    LDP #04H
    RET

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * CÁLCULO DE UM GRAU DE ATIVAÇÃO *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
CALC_Q:
    LDP #06H                    ; ENDEREÇAMENTOS DIRETOS: 300h..37Fh
    LAR AR4,#Z1_Q1_SIGN         ; AR4 –> PONTEIRO DE DESTINO: GRAUS DE ATIVAÇÃO DAS REGRAS
    LAR AR6,#Y1_D1Q_SIGN        ; AR6 –> PONTEIRO DE DxQ
    LACL #0
    SACL TEMP5                  ; ZERA CONTADOR PARA MACRO REPETIÇÕES: 20 GRAUS DE ATIVAÇÃO
PROX_GRAU:
    LAR AR2,#3                  ; CONTADOR PARA A TRANSFERÊNCIA DE DxQ (FP) PARA E (FP)
    LAR AR1,#ESIGN              ; AR1: PONTEIRO DE E
    MAR *,AR6                   ; AR6: PONTEIRO DE DxQ
TRANSF_TO_E:
    LACC *+,0,AR1
    SACL *+,0,AR2
    BANZ TRANSF_TO_E,*-,AR6     ; CARREGA DxQ EM E –> NUMERADOR
    LACL #0H
    CALL FIX_TO_FLOAT
    CALL MOVAD                  ; INICIALIZA D = 0 EM FP
    LAR AR1,#Y1_D1Q_SIGN        ; AR1: PONTEIRO DOS DxQ –> DENOMINADORES
    LAR AR3,#19                 ; AR3: 20 REPETIÇÕES –> 20 QUOCIENTES PARCIAIS PARA CALCULAR E SOMAR
CALC_DEN:
    LAR AR2,#3                  ; CONTADOR PARA A TRANSFERÊNCIA DE DxQ (FP) PARA B (FP)
    LAR AR0,#BSIGN              ; AR0: PONTEIRO DE B
    MAR *,AR1                   ; AR1: PONTEIRO DE DxQ
TRANSF_TO_B:
    LACC *+,0,AR0
    SACL *+,0,AR2
    BANZ TRANSF_TO_B,*-,AR1     ; CARREGA DxQ EM B = di2
    CALL MOVEA                  ; A <– dj2
    CALL FLOAT_DIV              ; C <– dj2/di2
    CALL MOVDA                  ; A <– sum (dj2/di2) , i = 1..(x-1) , PARA x = 1 –> A = 0
    CALL MOVCB                  ; B <– dj2/di2
    CALL FLOAT_ADD              ; C <– sum (dj2/di2) , i = 1..x
    CALL MOVCD                  ; D <– sum (dj2/di2) , i = 1..x
    MAR *,AR3
    BANZ CALC_DEN               ; REPETE LOOP 20 VEZES, OBTENDO O DENOMINADOR TOTAL
    SPLK #UM_SIGN,ASIGN
    SPLK #UM_EXP,AEXP
    SPLK #UM_HI,AHI
    SPLK #UM_LO,ALO             ; A <– 1
    CALL MOVDB                  ; B <– sum (dj2/di2) , i = 1..x
    CALL FLOAT_DIV              ; C <– 1 / sum (dj2/di2) , i = 1..x = GRAU DE ATIVAÇÃO Qj
    LAR AR2,#3                  ; CONTADOR PARA A TRANSFERÊNCIA DE C (FP) PARA Qj (FP)
    LAR AR0,#CSIGN              ; AR0: PONTEIRO DE C
    MAR *,AR0
```

```
TRANSFERE_Q:
    LACC *+,0,AR4                   ; AR4 –> PONTEIRO DE DESTINO: Q
    SACL *+,0,AR2
    BANZ TRANSFERE_Q,*-,AR0     ; CARREGA C EM Qj
    LACL TEMP5
    ADD #1
    SACL TEMP5
    XOR #20
    BCND PROX_GRAU,NEQ          ; REPETE PROCEDIMENTO PARA TODOS OS 20 GRAUS DE ATIVAÇÃo
    LDP #04H                    ; ENDEREÇAMENTOS DIRETOS: 200h..27Fh
    RET

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * CÁLCULO DAS SAÍDAS PARCIAIS *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
CALC_S_PARCIAIS:
    LDP #06H
    LAR AR6,#Y1_D1Q_SIGN        ; INICIALIZA AR6 –> PONTEIRO DOS RESULTADOS DAS SAÍDAS PARCIAIS
    LAR AR4,#19                 ; LOOP PARA MACRO REPETIÇÕES: 20 SAÍDAS PARCIAIS
    LACC #B10_SIGN
    SACL TEMP3                  ; TEMP3: PONTEIRO DOS SUBMODELOS –> BASTA INICIALIZÁ-LO AQUI
PROX_SAIDA:
    LAR AR3,#5                  ; LOOP DE 6 REPETIÇÕES: OPERAÇÕES DE MAC
    LACC #Q3_N1_SIGN
    SACL TEMP2                  ; TEMP2: PONTEIRO DE U
    LACL TEMP3
    CALL SUBMODELO_TO_A         ; A <– TERMO INDEPENDENTE - SUBMODELOS LINEARES
    CALL MOVAD                  ; D <– TERMO INDEPENDENTE - SUBMODELOS LINEARES
    LACL TEMP3
    ADD #4
    SACL TEMP3                  ; INCREMENTA PONTEIRO DOS SUBMODELOS
REP_S:
    LACL TEMP3
    CALL SUBMODELO_TO_A         ; A <– COMPONENTE DOS SUBMODELOS
    LAR AR0,TEMP2
    LAR AR1,#BSIGN
    CALL TRANSFERE              ; B <– COMPONENTE DO VETOR U
    CALL FLOAT_MULT             ; C <– A x B
    CALL MOVDA                  ; A <– bx0 + sum (bxi x Ui) , i = 1..(x-1) , PARA x = 1 –> A = bx0
    CALL MOVCB                  ; B <– bxi x Ui
    CALL FLOAT_ADD              ; C <– bx0 + sum (bxi x Ui) , i = 1..x
    CALL MOVCD                  ; D <– bx0 + sum (bxi x Ui) , i = 1..x
    LACL TEMP2
    ADD #4
    SACL TEMP2                  ; INCREMENTA PONTEIRO DE U
    LACL TEMP3
    ADD #4
    SACL TEMP3                  ; INCREMENTA PONTEIRO DOS SUBMODELOS
    MAR *,AR3
    BANZ REP_S                  ; LOOP PARA CÁLCULO DE UMA SAÍDA PARCIAL
    LAR AR2,#3                  ; CONTADOR PARA A TRANSFERÊNCIA DE D (FP) PARA Yx (FP)
    LAR AR0,#DSIGN              ; AR0: PONTEIRO DE D
    MAR *,AR0
TRANSFERE_S:
    LACC *+,0,AR6               ; AR6 –> PONTEIRO DE DESTINO: Yx
    SACL *+,0,AR2
    BANZ TRANSFERE_S,*-,AR0     ; CARREGA D EM Yx
    MAR *,AR4
    BANZ PROX_SAIDA             ; REPETE PROCEDIMENTO PARA TODOS AS SAÍDAS PARCIAIS
    LDP #04H
    RET

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * CÁLCULO DA SAÍDA GLOBAL *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
CALC_S_GLOBAL:
    LDP #06H
    LAR AR3,#19                 ; LOOP PARA MACRO REPETIÇÕES: 20 SAÍDAS PARCIAIS
    LACC #Y1_D1Q_SIGN
    SACL TEMP2                  ; TEMP2: PONTEIRO DAS SAÍDAS PARCIAIS
    LACC #Z1_Q1_SIGN
```

```
    SACL TEMP3              ; TEMP3: PONTEIRO DOS GRAUS DE ATIVAÇÃO
    LACL #0H
    CALL FIX_TO_FLOAT
    CALL MOVAD              ; INICIALIZA D = 0 EM FP
REP_GLOBAL:
    LAR AR0,TEMP2
    LAR AR1,#ASIGN
    CALL TRANSFERE          ; A <– SAÍDA PARCIAL
    LAR AR0,TEMP3
    LAR AR1,#BSIGN
    CALL TRANSFERE          ; B <– GRAU DE ATIVAÇÃO
    CALL FLOAT_MULT         ; C <– A x B
    CALL MOVDA              ; A <– sum (Yi x Qi) , i = 1..(x-1) , PARA x = 1 –> A = 0
    CALL MOVCB              ; B <– Yi x Qi
    CALL FLOAT_ADD          ; C <– sum (Yi x Qi) , i = 1..x
    CALL MOVCD              ; D <– sum (Yi x Qi) , i = 1..x
    LACL TEMP2
    ADD #4
    SACL TEMP2              ; INCREMENTA PONTEIRO DAS SAÍDAS PARCIAIS
    LACL TEMP3
    ADD #4
    SACL TEMP3              ; INCREMENTA PONTEIRO DOS GRAUS DE ATIVAÇÃO
    MAR *,AR3
    BANZ REP_GLOBAL         ; LOOP PARA CÁLCULO DA SAÍDA GLOBAL
    LAR AR0,#DSIGN          ; AR0: PONTEIRO DE D
    LAR AR1,#PH_N_SIGN      ; AR1: PONTEIRO DE pH(n)
    CALL TRANSFERE          ; SALVA pH(n)
    LDP #04H
    RET

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * ALGORITMO PID *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
PID:
    LDP #06H
; TERMO PROPORCIONAL:
    LAR AR0,#SP_SIGN        ; AR0: PONTEIRO DE YSP
    LAR AR1,#ASIGN          ; AR1: PONTEIRO DE A
    CALL TRANSFERE          ; A <– SET-POINT
    LAR AR0,#PH_N_SIGN      ; AR0: PONTEIRO DE Y(n)
    LAR AR1,#BSIGN          ; AR1: PONTEIRO DE B
    CALL TRANSFERE          ; B <– Y(n)
    CALL FLOAT_SUB          ; C <– YSP - Y
    CALL MOVCB              ; B <– YSP - Y
    SPLK #KC_SIGN,ASIGN
    SPLK #KC_EXP,AEXP
    SPLK #KC_HI,AHI
    SPLK #KC_LO,ALO         ; A <– KC
    CALL FLOAT_MULT         ; C <– KC * (YSP - Y)
    CALL MOVCE              ; E <– KC * (YSP - Y)
; TERMO DERIVATIVO:
    LAR AR0,#PH_N1_SIGN     ; AR0: PONTEIRO DE YA
    LAR AR1,#ASIGN          ; AR1: PONTEIRO DE A
    CALL TRANSFERE          ; A <– YA
    LAR AR0,#PH_N_SIGN      ; AR0: PONTEIRO DE Y(n)
    LAR AR1,#BSIGN          ; AR1: PONTEIRO DE B
    CALL TRANSFERE          ; B <– Y(n)
    CALL FLOAT_SUB          ; C <– YA - Y
    CALL MOVCB              ; B <– YA - Y
    SPLK #BD_SIGN,ASIGN
    SPLK #BD_EXP,AEXP
    SPLK #BD_HI,AHI
    SPLK #BD_LO,ALO         ; A <– BD
    CALL FLOAT_MULT         ; C <– BD * (YA - Y)
    CALL MOVCD              ; D <– BD * (YA - Y)
    SPLK #AD_SIGN,ASIGN
    SPLK #AD_EXP,AEXP
    SPLK #AD_HI,AHI
    SPLK #AD_LO,ALO         ; A <– AD
    LAR AR0,#DN_SIGN        ; AR0: PONTEIRO DE DA
    LAR AR1,#BSIGN          ; AR1: PONTEIRO DE B
```

```
    CALL TRANSFERE          ; B <– DA
    CALL FLOAT_MULT         ; C <– AD * DA
    CALL MOVCA              ; A <– AD * DA
    CALL MOVDB              ; B <– BD * (YA - Y)
    CALL FLOAT_ADD          ; C <– (AD * DA) + (BD * (YA - Y))
    LAR AR0,#CSIGN          ; AR0: PONTEIRO DE C
    LAR AR1,#DN_SIGN        ; AR1: PONTEIRO DE DN
    CALL TRANSFERE          ; SALVA DN
; CALCULA V:
    CALL MOVEA              ; A <– KC * (YSP - Y)
    CALL MOVCB              ; B <– (AD * DA) + (BD * (YA - Y))
    CALL FLOAT_ADD          ; C <– P + D
    CALL MOVCB              ; B <– P + D
    LAR AR0,#IN_SIGN        ; AR0: PONTEIRO DE IN
    LAR AR1,#ASIGN          ; AR1: PONTEIRO DE A
    CALL TRANSFERE          ; A <– IN
    CALL FLOAT_ADD          ; C <– P + I + D
    LAR AR0,#CSIGN          ; AR0: PONTEIRO DE C
    LAR AR1,#V_SIGN         ; AR1: PONTEIRO DE V
    CALL TRANSFERE          ; SALVA V
; CHECA SATURAÇÃO DE U:
    CALL MOVCD              ; D <– V
    CALL MOVCA              ; A <– V
    SPLK #UMAX_SIGN,BSIGN
    SPLK #UMAX_EXP,BEXP
    SPLK #UMAX_HI,BHI
    SPLK #UMAX_LO,BLO       ; B <– UMAX
    CALL FLOAT_SUB          ; C <– V - UMAX
    LACC CSIGN,16
    BCND SAT_MAX,EQ         ; V > UMAX –> FAZ U = UMAX
    CALL MOVDA              ; A <– V
    SPLK #UMIN_SIGN,BSIGN
    SPLK #UMIN_EXP,BEXP
    SPLK #UMIN_HI,BHI
    SPLK #UMIN_LO,BLO       ; B <– UMIN
    CALL FLOAT_SUB          ; C <– V - UMIN
    LACC CSIGN,16
    BCND SAT_MIN,NEQ        ; V < UMIN –> FAZ U = UMIN
    LAR AR0,#V_SIGN         ; AR0: PONTEIRO DE V
    LAR AR1,#U_SIGN         ; AR1: PONTEIRO DE U
    CALL TRANSFERE          ; U <– V
    B CALC_INTEGRAL         ; SEGUE
SAT_MAX:
    MAR *,AR3
    LAR AR3,#U_SIGN
    SPLK #UMAX_SIGN,*+
    SPLK #UMAX_EXP,*+
    SPLK #UMAX_HI,*+
    SPLK #UMAX_LO,*+        ; U <– UMAX
    B CALC_INTEGRAL         ; SEGUE
SAT_MIN:
    MAR *,AR3
    LAR AR3,#U_SIGN
    SPLK #UMIN_SIGN,*+
    SPLK #UMIN_EXP,*+
    SPLK #UMIN_HI,*+
    SPLK #UMIN_LO,*+        ; U <– UMIN
CALC_INTEGRAL:
    LAR AR0,#U_SIGN         ; AR0: PONTEIRO DE U
    LAR AR1,#ASIGN          ; AR1: PONTEIRO DE A
    CALL TRANSFERE          ; A <– U
    LAR AR0,#V_SIGN         ; AR0: PONTEIRO DE V
    LAR AR1,#BSIGN          ; AR1: PONTEIRO DE B
    CALL TRANSFERE          ; B <– V
    CALL FLOAT_SUB          ; C <– U - V
    CALL MOVCB              ; B <– U - V
    SPLK #BT_SIGN,ASIGN
    SPLK #BT_EXP,AEXP
    SPLK #BT_HI,AHI
    SPLK #BT_LO,ALO         ; A <– BT
    CALL FLOAT_MULT         ; C <– BT * (U - V)
    CALL MOVCD              ; D <– BT * (U - V)
```

```
    LAR AR0,#SP_SIGN            ; AR0: PONTEIRO DE YSP
    LAR AR1,#ASIGN              ; AR1: PONTEIRO DE A
    CALL TRANSFERE              ; A <– YSP
    LAR AR0,#PH_N_SIGN          ; AR0: PONTEIRO DE Y
    LAR AR1,#BSIGN              ; AR1: PONTEIRO DE B
    CALL TRANSFERE              ; B <– Y
    CALL FLOAT_SUB              ; C <– YSP - Y
    CALL MOVCB                  ; B <– YSP - Y
    SPLK #BI_SIGN,ASIGN
    SPLK #BI_EXP,AEXP
    SPLK #BI_HI,AHI
    SPLK #BI_LO,ALO             ; A <– BI
    CALL FLOAT_MULT             ; C <– BI * (YSP - Y)
    CALL MOVCB                  ; B <– BI * (YSP - Y)
    CALL MOVDA                  ; A <– BT * (U - V)
    CALL FLOAT_ADD              ; C <– BI * (YSP - Y) + BT * (U - V)
    CALL MOVCB                  ; B <– BI * (YSP - Y) + BT * (U - V)
    LAR AR0,#IN_SIGN            ; AR0: PONTEIRO DE IN
    LAR AR1,#ASIGN              ; AR1: PONTEIRO DE A
    CALL TRANSFERE              ; A <– IN
    CALL FLOAT_ADD              ; C <– IN + BI * (YSP - Y) + BT * (U - V)
    LAR AR0,#CSIGN              ; AR0: PONTEIRO DE C
    LAR AR1,#IN_SIGN            ; AR1: PONTEIRO DE IN
    CALL TRANSFERE              ; SALVA IN
    LDP #04H
    RET

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * PREPARA VALOR PARA D/A *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
PID_TO_DA:
    LDP #06H
    LAR AR0,#U_SIGN
    LAR AR1,#ASIGN
    CALL TRANSFERE              ; A <– u(n)
    SPLK #OFFSET_Q3_SIGN,BSIGN
    SPLK #OFFSET_Q3_EXP,BEXP
    SPLK #OFFSET_Q3_HI,BHI
    SPLK #OFFSET_Q3_LO,BLO      ; B <– 0,4684
    CALL FLOAT_ADD              ; C <– u(n) + offset_q3
    CALL FLOAT_TO_FIX
    LDP #04H
    CLRC SXM
    SFR
    SFR
    SFR
    SACH D_A                    ; CARREGA VALOR PARA O D/A
    SETC SXM
    RET

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * ATUALIZA REGRESSORES *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
ATUALIZA_REGR:
    LAR AR0,#PH_N1_SIGN
    LAR AR1,#PH_N2_SIGN
    CALL TRANSFERE          ; pH(n-2) <– pH(n-1)
    LAR AR0,#PH_N_SIGN
    LAR AR1,#PH_N1_SIGN
    CALL TRANSFERE          ; pH(n-1) <– pH(n)
    LAR AR0,#Q3_N3_SIGN
    LAR AR1,#Q3_N4_SIGN
    CALL TRANSFERE          ; q3(n-4) <– q3(n-3)
    LAR AR0,#Q3_N2_SIGN
    LAR AR1,#Q3_N3_SIGN
    CALL TRANSFERE          ; q3(n-3) <– q3(n-2)
    LAR AR0,#Q3_N1_SIGN
    LAR AR1,#Q3_N2_SIGN
    CALL TRANSFERE          ; q3(n-2) <– q3(n-1)
    LAR AR0,#Q3_N_SIGN
    LAR AR1,#Q3_N1_SIGN
```

```
    CALL TRANSFERE          ; q3(n-1) <– q3(n)
    RET

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * ALTERA SET-POINT *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
MUDA_SP:
    LACL FASE
    BCND FASE1,EQ                   ; FASE = 0, VAI PARA FASE1
    LACL FASE
    XOR #1
    BCND FASE2,EQ                   ; FASE = 1, VAI PARA FASE2
    LACL FASE
    XOR #2
    BCND FASE3,EQ                   ; FASE = 2, VAI PARA FASE3
    SPLK #0,FASE                    ; FASE = 3, RETORNA PARA ZERO
    SPLK #SP_INI_SIGN,BUF_SIGN
    SPLK #SP_INI_EXP,BUF_EXP
    SPLK #SP_INI_HI,BUF_HI
    SPLK #SP_INI_LO,BUF_LO          ; BUF <– SET-POINT INICIAL EM FP
    B MUDA_SP1
FASE1:
FASE3:
    LACL FASE
    ADD #1
    SACL FASE                       ; INCREMENTA A FASE
    SPLK #SP_MAX_SIGN,BUF_SIGN
    SPLK #SP_MAX_EXP,BUF_EXP
    SPLK #SP_MAX_HI,BUF_HI
    SPLK #SP_MAX_LO,BUF_LO          ; BUF <– SET-POINT MÁXIMO EM FP
    B MUDA_SP1
FASE2:
    LACL FASE
    ADD #1
    SACL FASE                       ; INCREMENTA A FASE
    SPLK #SP_MIN_SIGN,BUF_SIGN
    SPLK #SP_MIN_EXP,BUF_EXP
    SPLK #SP_MIN_HI,BUF_HI
    SPLK #SP_MIN_LO,BUF_LO          ; BUF <– SET-POINT MÍNIMO EM FP
MUDA_SP1:
    LAR AR0,#BUF_SIGN
    LAR AR1,#SP_SIGN
    CALL TRANSFERE                  ; ALTERA SET-POINT
    RET

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * ROTINAS DE PONTO FLUTUANTE *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * CONVERSÃO PONTO FIXO –> PONTO FLUTUANTE NORMALIZADO *
; * ENTRADA: ACCH (16 BITS - Q15) *
; * SAÍDA: ASIGN, AEXP, AHI, ALO *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
FIX_TO_FLOAT:
    BCND NEG_MANT,LT            ; NÚMERO NEGATIVO?
    SPLK #0000H,ASIGN           ; NÃO, CARREGA SINAL POSITIVO
    B FIX_TO_FLOAT1             ; SEGUE
NEG_MANT:
    SPLK #0FFFFH,ASIGN          ; CARREGA SINAL NEGATIVO
    NEG                         ; OBTÉM EQUIVALENTE POSITIVO
FIX_TO_FLOAT1:
    MAR *,AR5
    LAR AR5,#0                  ; INICIALIZA EXPOENTE
    RPT #13
    NORM *+                     ; 14 LEFT SHIFTS NO MÁXIMO: NORMALIZAÇÃO PARA 16 BITS SINALIZADO
    NOP
    NOP                         ; PROTEÇÃO P/ PIPELINE
                                ; ACCH <– MANTISSA
                                ; EXPOENTE <– -1 * AR5
```

```
FIX_TO_FLOAT2:
    SACH AHI                    ; SALVA MANTISSA NORMALIZADA
    SPLK #0000H,ALO             ; ZERA PARTE BAIXA
    SAR AR5,AEXP
    LACC AEXP,16
    NEG
    SACH AEXP                   ; CARREGA EXPOENTE <– -1 * AR5 (EXPOENTE ESTÁ SEM BIAS)
    RET

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * CONVERSÃO PONTO FLUTUANTE –> PONTO FIXO NORMALIZADO *
; * ENTRADA: CSIGN, CEXP, CHI, CLO *
; * SAÍDA: ACCH (16 BITS - Q15) *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
FLOAT_TO_FIX:
; DESNORMALIZAÇÃO:
    LACC CEXP,16                ; EXPOENTE NEGATIVO (RIGHT SHIFTS)?
    BCND EXP_POS,GEQ            ; NÃO (LEFT SHIFTS)
    NEG
    SACH CEXP                   ; SIM, OBTÉM CORRESPONDENTE POSITIVO
    LAR AR5,CEXP                ; CARREGA CONTADOR PARA RIGHT SHIFTS
    MAR *,AR5
    MAR *-
    LACC CHI,16
    ADDS CLO                    ; ACC <– MANTISSA NORMALIZADA
RIGHT_SHIFTS:
    SFR
    BANZ RIGHT_SHIFTS           ; DESNORMALIZA RESULTADO
    SACH CHI
    SACL CLO
    B FLOAT_TO_FIX2             ; SEGUE
EXP_POS:
    BCND FLOAT_TO_FIX2,EQ       ; EXP = 0
    LAR AR5,CEXP                ; CARREGA CONTADOR PARA LEFT SHIFTS
    MAR *,AR5
    MAR *-
    LACC CHI,16
    ADDS CLO                    ; ACC <– MANTISSA NORMALIZADA
LEFT_SHIFTS:
    SFL
    BANZ LEFT_SHIFTS            ; DESNORMALIZA RESULTADO
    SACH CHI
    SACL CLO
; ARREDONDAMENTO:
FLOAT_TO_FIX2:
    LACC CLO,16
    BCND FLOAT_TO_FIX3,GEQ
    SPLK #0001H,TEMP1
    LACC CHI,16
    ADD TEMP1,16                ; INCREMENTA ACCH –> SATURAÇÃO DEVE ESTAR HABILITADA!
    SACH CHI
; OVERFLOW / UNDERFLOW:
FLOAT_TO_FIX3:
    LACC CSIGN,16
    BCND CHECA_OVFLOW,GEQ       ; NÚMERO POSITIVO
                                ; NEGATIVO
    LACC CHI,16
    NEG
    BCND HOUVE_UDFLOW,GEQ       ; OCORRÊNCIA DE UNDERFLOW
    RET                         ; SEM UNDERFLOW, FIM
HOUVE_UDFLOW:
    SPLK #8000H,TEMP1
    LACC TEMP1,16               ; FORÇA VALOR MÍNIMO (-1)
    RET
CHECA_OVFLOW:
    LACC CHI,16
    BCND HOUVE_OVFLOW,LT        ; OCORRÊNCIA DE OVERFLOW: FORÇA ACCH = 7FFFh
    RET                         ; SEM OVERFLOW, FIM
HOUVE_OVFLOW:
    SPLK #7FFFH,TEMP1
    LACC TEMP1,16               ; FORÇA VALOR MÁXIMO (+1)
```

```
      RET

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * SUBTRAÇÃO EM PONTO FLUTUANTE: C = A - B *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
FLOAT_SUB:
      LACC BSIGN
      XOR #0FFFFH
      SACL BSIGN                    ; INVERTE O SINAL DE B E FAZ A SOMA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * SOMA EM PONTO FLUTUANTE: C = A + B *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
FLOAT_ADD:
      CLRC OVM                      ; OVM = 0 –> DESABILITA SATURAÇÃO DO ACUMULADOR
      MAR *,AR5
      LAR AR5,#0                    ; INICIALIZA EXPOENTE
      LACC AEXP
      SUB BEXP                      ; ACC = AEXP - BEXP –> ENCONTRA O MAIOR NÚMERO
      BCND AEQB,EQ                  ; SE SÃO IGUAIS, VAI PARA AEQB
      BCND ALTB,LT                  ; SE A É MENOR QUE B, VAI PARA ALTB
AGTB:
      NEG
      ADD #16                       ; ACC <– 16 - ACC
      BCND A1,LT                    ; SE A DIFERENÇA ENTRE OS EXPOENTES > 16, VAI PARA A1
; DIFERENÇA ENTRE OS EXPOENTES < 16:
      SACL D                        ; SALVA DIFERENÇA ENTRE OS EXPOENTES
      LT D
      LACT BHI                      ; BHI É SHIFTADO PARA DIREITA "D" VEZES
      SACH BHI
      SACL RESID                    ; MANTÉM BITS DE RESÍDUO
      LACT BLO                      ; BLO É SHIFTADO PARA DIREITA "D" VEZES
      SFL                           ; LIVRA-SE DO MSB
      SACH BLO
      LACC BLO
      OR RESID                      ; OBTÉM BITS SHIFTADOS DO BHI
      SACL BLO
      B A2                          ; SEGUE
; DIFERENÇA ENTRE OS EXPOENTES > 16:
A1:
      ADD #16
      BCND A3,LT                    ; SE A DIFERENÇA ENTRE OS EXPOENTES > 32, VAI PARA A3
      SACL D
      LT D
      LACT BHI
      SACH BLO
      LACC #0000H
      SACL BHI
A2:
      LACC ASIGN                    ; A É MAIOR QUE B
      SACL CSIGN                    ; PORTANTO, CSIGN <– ASIGN
      LACC AEXP
      SACL CEXP                     ; CEXP <– AEXP
      LACC ALO,1                    ; LIVRA-SE DO EXTRA BIT
      SACL ALO
      B CHKSGN                      ; VERIFICA SE AMBOS OS NÚMEROS TÊM MESMO SINAL
; A >>> B –> RESULTADO = A:
A3:
      LACC AHI
      SACL CHI
      LACC ALO,1
      SACL CLO
      LACC ASIGN
      SACL CSIGN                    ; A > B –> CSIGN = ASIGN
      LACC AEXP
      SACL CEXP
      B AROUND                      ; SEGUE
AEQB:
      LACC ASIGN                    ; SE OS SINAIS FOREM IGUAIS, CSIGN = ASIGN
      SACL CSIGN
      LACC ALO,1
```

```
        SACL ALO
        LACC BLO,1
        SACL BLO                ; ALINHA MANTISSAS
        LACC AEXP
        SACL CEXP               ; CEXP = AEXP
        B CHKSGN                ; VERIFICA SE AMBOS OS NÚMEROS TÊM MESMO SINAL
ALTB:
        ADD #16                 ; D = (16 - D)
        BCND B1,LT              ; SE A DIFERENÇA ENTRE OS EXPOENTES > 16, VAI PARA B1
        SACL D
        LT D
        LACT AHI                ; LEFT SHIFTA AHI "D" VEZES
        SACH AHI
        SACL RESID              ; MANTÉM EXTRA BITS
        LACT ALO                ; LEFT SHIFTA ALO "D" VEZES
        SFL                     ; LIVRA-SE DO MSB
        SACH ALO
        LACC ALO
        OR RESID                ; OBTÉM BITS RESIDUAIS
        SACL ALO
        B B2                    ; SEGUE
; DIFERENÇA ENTRE OS EXPOENTES > 16:
B1:
        ADD #16
        BCND B3,LT              ; SE A DIFERENÇA ENTRE OS EXPOENTES > 32, VAI PARA B3
        SACL D
        LT D
        LACT AHI
        SACH ALO
        LACC #0000H
        SACL AHI
B2:
        LACC BSIGN
        SACL CSIGN              ; B > A –> CSIGN = BSIGN
        LACC BEXP
        SACL CEXP               ; CEXP <– BEXP
        LACC BLO,1              ; LIVRA-SE DO EXTRA BIT
        SACL BLO
        B CHKSGN                ; VERIFICA SE AMBOS OS NÚMEROS TÊM MESMO SINAL
; B >>> A –> RESULTADO = B:
B3:
        LACC BHI
        SACL CHI
        LACC BLO,1
        SACL CLO
        LACC BSIGN
        SACL CSIGN              ; CSIGN <– BSIGN
        LACC BEXP
        SACL CEXP               ; CEXP <– BEXP
        B AROUND                ; SEGUE
CHKSGN:
        LACC ASIGN
        SUB BSIGN               ; ACC <– ASIGN - BSIGN
        BCND ADNOW,EQ           ; SINAIS IGUAIS, PODE SOMAR NÚMEROS
        BCND AISNEG,LT          ; A É NEGATIVO
BISNEG:
        LACC AHI,16
        ADDS ALO
        SUBS BLO
        SUB BHI,16
        BCND CZERO,EQ
        BCND CNEG,LT
        SACH CHI
        SACL CLO
        LACC #0000H
        SACL CSIGN
        B NORMAL                ; VAI PARA NORMALIZAÇÃO DO RESULTADO
AISNEG:
        LACC BHI,16
        ADDS BLO
        SUBS ALO
        SUB AHI,16
```

```
    BCND CZERO,EQ
    BCND CNEG,LT
    SACH CHI
    SACL CLO
    LACC #0000H
    SACL CSIGN
    B NORMAL            ; VAI PARA NORMALIZAÇÃO DO RESULTADO
CZERO:
    LACC #0000H
    SACL CEXP
    SACL CSIGN
    SACL CHI
    SACL CLO
    B AROUND            ; SEGUE
CNEG:
    ABS
    SACH CHI
    SACL CLO
    LACC #0FFFFH
    SACL CSIGN
    B NORMAL            ; VAI PARA NORMALIZAÇÃO DO RESULTADO
ADNOW:
    MAR *,AR7
    LAR AR7,#TEMP
    SST #0,*
    LACC TEMP
    AND #0EFFFH
    SACL TEMP
    LST #0,TEMP
    MAR *,AR5           ; GARANTE BIT OV ZERADO
    LACC AHI,16
    ADDS ALO
    ADDS BLO
    ADD BHI,16
    SACH CHI
    SACL CLO
    BCND OVFLOW,OV      ; TRATA OVERFLOW
    BCND CZERO,EQ       ; C = 0
; NORMALIZAÇÃO:
NORMAL:
    LACC CHI
    BCND LO1,EQ         ; CHI NÃO CONTÉM O MSB
    LACC CHI,16
    ADDS CLO
    RPT #13
    NORM *+
    B OUTPUT            ; SEGUE
LO1:
    LACC CLO,16         ; CLO CONTÉM O MSB
    LAR AR5,#16         ; OFFSET DE 16 NO EXPOENTE
    BCND NOFLOW,LT      ; NOVO OVERFLOW
    RPT #13
    NORM *+
    B OUTPUT            ; SEGUE
OVFLOW:
    CLRC SXM            ; RESET SIGN EXTENSION
    SFR
    SACH CHI
    SACL CLO
    LACC CEXP
    ADD #1
    SACL CEXP           ; INCREMENTA EXPOENTE
    B AROUND            ; SEGUE
NOFLOW:
    MAR *-              ; DECREMENTA AR5, O QUE GERA UM INCREMENTO EM CEXP
    CLRC SXM            ; RESET SIGN EXTENSION
    SFR
OUTPUT:
    SAR AR5,TEMP
    SACH CHI
    SACL CLO
    LACC CEXP
```

```
    SUB TEMP
    SACL CEXP           ; AJUSTA EXPOENTE
AROUND:
    SETC SXM            ; HABILITA SIGN EXTENSION
    SETC OVM            ; OVM = 1 –> HABILITA SATURAÇÃO DO ACUMULADOR
    RET                 ; FIM

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * MULTIPLICAÇÃO EM PONTO FLUTUANTE: C = A x B *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
FLOAT_MULT:
    MAR *,AR5
    LAR AR5,#0          ; INICIALIZA EXPOENTE
    LACC AEXP
    ADD BEXP
    SACL CEXP           ; CEXP <– AEXP + BEXP
    LT ALO
    MPY BHI
    PAC
    SACH THI
    SACL TLO            ; T <– ALO * BHI
    LT AHI
    MPY BLO             ; PREG <– AHI * BLO
    APAC
    APAC                ; ACC <– (AHI * BLO + ALO * BHI) * 2^-15
    ADD THI,16
    ADDS TLO
    SACH THI
    MPY BHI             ; AHI * BHI
    PAC
    ADDS THI
    SACH CHI
    SACL CLO            ; SALVA MANTISSA, EXTRA SIGN JÁ REMOVIDO (PM = 1)
    BCND OK,NEQ         ; RESULTADO <> 0
    LACC #0000H
    SACL CEXP           ; ZERA EXPOENTE
    B SETSIN            ; SEGUE
OK:
    LACC CHI,16
    ADDS CLO
    NORM *+             ; NORMALIZA MANTISSA
    SACH CHI
    SACL CLO            ; SALVA
    SAR AR5,TEMP
    LACC CEXP
    SUB TEMP
    SACL CEXP           ; AJUSTA EXPOENTE
SETSIN:
    LACL ASIGN
    XOR BSIGN           ; DESCOBRE O SINAL DO RESULTADO
    BCND NEG,NEQ        ; NEGATIVO
    LACC #0000H
    SACL CSIGN
    B OUTPUT1           ; POSITIVO
NEG:
    LACC #0FFFFH
    SACL CSIGN
OUTPUT1:
    RET

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * DIVISÃO EM PONTO FLUTUANTE: C = A / B *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
FLOAT_DIV:
    MAR *,AR5
    LAR AR5,#0          ; INICIALIZA EXPOENTE
    LACC #0000H
    SACL CEXP           ; ZERA CEXP
    LACC #0000H
    SACL CSIGN
    LACL ASIGN
```

```
        XOR BSIGN
        BCND OK1,EQ             ; CSIGN = +, CASO ASIGN = BSIGN
        LACC #0FFFFH
        SACL CSIGN              ; CSIGN = -1
OK1:
        LACC AHI,16
        ADDS ALO
        RPT #3
        SFR
        RPT #15
        SUBC BHI
        SACH R1
        SACL QM                 ; QM = AHI:ALO / BHI, R1 = RESTO
        LACC R1,15
        RPT #15
        SUBC BHI
        SACH R2
        SACL QL                 ; QL = (R1 * 2^15) / BHI, R2 = RESTO
        LT QM
        MPY BLO
        PAC
        RPT #15
        SUBC BHI
        SACL CL                 ; CL = (QM * BLO) / BHI
        LACC QM,16
        ADDS QL
        SUB CL
        SACL CLO
        SACH CHI
        LACL CHI
        AND #8000H
        BCND ESTOURO,NEQ
        LACL CHI
        AND #4000H
        BCND SOMA_3,NEQ
        LACL CHI
        AND #2000H
        BCND SOMA_2,NEQ
        LACL CHI
        AND #1000H
        BCND SOMA_1,NEQ     ; CHECAGENS DE OVERFLOW
        B NOOVF             ; SEM OVERFLOW
ESTOURO:
        LACC CHI,16
        ADDS CLO
        CLRC SXM
        SFR
        SACH CHI
        SACL CLO
        SETC SXM
        LACL #5
        SACL CEXP
        B NOOVF
SOMA_3:
        LACL #3
        SACL CEXP
        B NOOVF
SOMA_2:
        LACL #2
        SACL CEXP
        B NOOVF
SOMA_1:
        LACL #1
        SACL CEXP
NOOVF:
        LACC AEXP
        SUB BEXP
        ADD CEXP
        SACL CEXP
; NORMALIZAÇÃO:
NORMAL1:
        LACC CHI
```

```
    BCND LO2,EQ             ; CHI NÃO CONTÉM O MSB
    LACC CHI,16
    ADDS CLO
    RPT #13
    NORM *+
    B OUTPUT2               ; SEGUE
LO2:
    LACC CLO,16
    BCND NOFLOW1,LT         ; NOVO OVERFLOW
    RPT #13
    NORM *+
    B OUTPUT2               ; SEGUE
NOFLOW1:
    CLRC SXM
    SFR
OUTPUT2:
    SACH CHI
    SACL CLO
    SETC SXM
    RET

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * MOVIMENTAÇÃO ENTRE REGISTRADORES FP *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
MOVAB:
    BLDD #ASIGN,BSIGN
    BLDD #AEXP,BEXP
    BLDD #AHI,BHI
    BLDD #ALO,BLO
    RET

MOVAC:
    BLDD #ASIGN,CSIGN
    BLDD #AEXP,CEXP
    BLDD #AHI,CHI
    BLDD #ALO,CLO
    RET

MOVAD:
    BLDD #ASIGN,DSIGN
    BLDD #AEXP,DEXP
    BLDD #AHI,DHI
    BLDD #ALO,DLO
    RET

MOVAE:
    BLDD #ASIGN,ESIGN
    BLDD #AEXP,EEXP
    BLDD #AHI,EHI
    BLDD #ALO,ELO
    RET

MOVBA:
    BLDD #BSIGN,ASIGN
    BLDD #BEXP,AEXP
    BLDD #BHI,AHI
    BLDD #BLO,ALO
    RET

MOVBC:
    BLDD #BSIGN,CSIGN
    BLDD #BEXP,CEXP
    BLDD #BHI,CHI
    BLDD #BLO,CLO
    RET

MOVBD:
    BLDD #BSIGN,DSIGN
    BLDD #BEXP,DEXP
```

```
    BLDD #BHI,DHI
    BLDD #BLO,DLO
    RET

MOVBE:
    BLDD #BSIGN,ESIGN
    BLDD #BEXP,EEXP
    BLDD #BHI,EHI
    BLDD #BLO,ELO
    RET

MOVCA:
    BLDD #CSIGN,ASIGN
    BLDD #CEXP,AEXP
    BLDD #CHI,AHI
    BLDD #CLO,ALO
    RET

MOVCB:
    BLDD #CSIGN,BSIGN
    BLDD #CEXP,BEXP
    BLDD #CHI,BHI
    BLDD #CLO,BLO
    RET

MOVCD:
    BLDD #CSIGN,DSIGN
    BLDD #CEXP,DEXP
    BLDD #CHI,DHI
    BLDD #CLO,DLO
    RET

MOVCE:
    BLDD #CSIGN,ESIGN
    BLDD #CEXP,EEXP
    BLDD #CHI,EHI
    BLDD #CLO,ELO
    RET

MOVDA:
    BLDD #DSIGN,ASIGN
    BLDD #DEXP,AEXP
    BLDD #DHI,AHI
    BLDD #DLO,ALO
    RET

MOVDB:
    BLDD #DSIGN,BSIGN
    BLDD #DEXP,BEXP
    BLDD #DHI,BHI
    BLDD #DLO,BLO
    RET

MOVDC:
    BLDD #DSIGN,CSIGN
    BLDD #DEXP,CEXP
    BLDD #DHI,CHI
    BLDD #DLO,CLO
    RET

MOVDE:
    BLDD #DSIGN,ESIGN
    BLDD #DEXP,EEXP
    BLDD #DHI,EHI
    BLDD #DLO,ELO
    RET

MOVEA:
```

```
    BLDD #ESIGN,ASIGN
    BLDD #EEXP,AEXP
    BLDD #EHI,AHI
    BLDD #ELO,ALO
    RET

MOVEB:
    BLDD #ESIGN,BSIGN
    BLDD #EEXP,BEXP
    BLDD #EHI,BHI
    BLDD #ELO,BLO
    RET

MOVEC:
    BLDD #ESIGN,CSIGN
    BLDD #EEXP,CEXP
    BLDD #EHI,CHI
    BLDD #ELO,CLO
    RET

MOVED:
    BLDD #ESIGN,DSIGN
    BLDD #EEXP,DEXP
    BLDD #EHI,DHI
    BLDD #ELO,DLO
    RET

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * GRAVAÇÃO DA TABELA SECUNDÁRIA DE VETORES DE INT. EM 8000h - MEM. PROG. *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
INITVECS:
; FIRST FILL THE ENTIRE TABLE WITH 'B 0000H' INSTRUCTIONS, THIS WILL
; RESTART THE MONITOR ON ANY INTERRUPT NOT USED.
    LACC #8000H             ; ACCL = STARTING ADDRESS OF SECONDARY VECTOR TABLE
    LAR AR6,#AUX            ; AR6 = ADDRESS IN DATA MEMORY
    MAR *,AR6               ; ARP = AR6
    SPLK #7980H,AUX         ; STORE THE BRANCH OPCODE
    SPLK #0000H,AUX + 1     ; STORE THE RESET VECTOR (TO RESTART MONITOR)
; RESET VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; /HOLD//INT1 VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; /INT2//INT3 VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; TINT VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; RINT VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; XINT VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; TXRXINT VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
```

```
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; RESERVED VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; INT8 VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; INT9 VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; INT10 VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; INT11 VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; INT12 VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; INT13 VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; INT14 VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; INT15 VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; INT16 VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; TRAP VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; /NMI VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; RESERVED VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
; INT20 VECTOR
    TBLW *+                 ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                  ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                 ; WRITE VECTOR (0000H) TO TABLE
```

```
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
; INT21 VECTOR
    TBLW *+                     ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                     ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
; INT22 VECTOR
    TBLW *+                     ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                     ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
; INT23 VECTOR
    TBLW *+                     ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                     ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
; INT24 VECTOR
    TBLW *+                     ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                     ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
; INT25 VECTOR
    TBLW *+                     ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                     ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
; INT26 VECTOR
    TBLW *+                     ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                     ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
; INT27 VECTOR
    TBLW *+                     ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                     ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
; INT28 VECTOR
    TBLW *+                     ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                     ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
; INT29 VECTOR
    TBLW *+                     ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                     ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
; INT30 VECTOR
    TBLW *+                     ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                     ; WRITE VECTOR (0000H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
; INT31 VECTOR
    TBLW *+                     ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                     ; WRITE VECTOR (0000H) TO TABLE
; NOW THAT THE TABLE IS FULL, PUNCH IN THE INDIVIDUAL VECTORS USED IN
; THIS PROGRAM
    LACC #8006H                 ; ACCL = LOCATION IN SECONDARY TABLE
    SPLK #TIMERISR,AUX + 1      ; PUT VECTOR IN DATA SPACE
    TBLW *+                     ; WRITE BRANCH OPCODE (7980H) TO TABLE
    ADD #1                      ; INCREMENT ADDRESS IN ACCUMULATOR
    TBLW *-                     ; WRITE VECTOR (TIMERISR) TO TABLE
    RET                         ; RETURN TO CALLER

;*************************************************************************************************
; DECLARAÇÃO DE TODOS OS COEFICIENTES DO MODELO DO SENSOR - SÃO CONSTANTES EM MEMÓRIA DE PROGRAMA:

; CENTROS VETORIAIS DOS CLUSTERS:
; CENTRO VETORIAL - CLUSTER 1:
Q3_N1_C1_SIGN    .WORD 0H
Q3_N1_C1_EXP     .WORD -1
```

```
Q3_N1_C1_HI         .WORD 67A1H
Q3_N1_C1_LO         .WORD 0D7EAH      ; 0 Q3(n-1) = 0,4048132844331

Q3_N2_C1_SIGN       .WORD 0H
Q3_N2_C1_EXP        .WORD -1
Q3_N2_C1_HI         .WORD 6768H
Q3_N2_C1_LO         .WORD 4806H       ; 1 Q3(n-2) = 0,4039349568989

Q3_N3_C1_SIGN       .WORD 0H
Q3_N3_C1_EXP        .WORD -1
Q3_N3_C1_HI         .WORD 6751H
Q3_N3_C1_LO         .WORD 5D86H       ; 2 Q3(n-3) = 0,4035852862376

Q3_N4_C1_SIGN       .WORD 0H
Q3_N4_C1_EXP        .WORD -1
Q3_N4_C1_HI         .WORD 679CH
Q3_N4_C1_LO         .WORD 83C6H       ; 3 Q3(n-4) = 0,4047319753516

PH_N1_C1_SIGN       .WORD 0H
PH_N1_C1_EXP        .WORD 3
PH_N1_C1_HI         .WORD 651AH
PH_N1_C1_LO         .WORD 5E88H       ; 4 pH(n-1) = 6,3189378071355

PH_N2_C1_SIGN       .WORD 0H
PH_N2_C1_EXP        .WORD 3
PH_N2_C1_HI         .WORD 651EH
PH_N2_C1_LO        .WORD 1583H        ; 5 pH(n-2) = 6,3198447348479

; CENTRO VETORIAL - CLUSTER 2:
Q3_N1_C2_SIGN       .WORD 0H
Q3_N1_C2_EXP        .WORD 0
Q3_N1_C2_HI         .WORD 444FH
Q3_N1_C2_LO         .WORD 0C334H      ; 6 Q3(n-1) = 0,5336841585834

Q3_N2_C2_SIGN       .WORD 0H
Q3_N2_C2_EXP        .WORD 0
Q3_N2_C2_HI         .WORD 4446H
Q3_N2_C2_LO         .WORD 55BEH       ; 7 Q3(n-2) = 0,5333964516508

Q3_N3_C2_SIGN       .WORD 0H
Q3_N3_C2_EXP        .WORD 0
Q3_N3_C2_HI         .WORD 444EH
Q3_N3_C2_LO         .WORD 0ACC2H    ; 8 Q3(n-3) = 0,5336509654531

Q3_N4_C2_SIGN       .WORD 0H
Q3_N4_C2_EXP        .WORD 0
Q3_N4_C2_HI         .WORD 4457H
Q3_N4_C2_LO         .WORD 7855H       ; 9 Q3(n-4) = 0,5339193741920

PH_N1_C2_SIGN       .WORD 0H
PH_N1_C2_EXP        .WORD 4
PH_N1_C2_HI         .WORD 4C9EH
PH_N1_C2_LO         .WORD 5EF2H       ; 10 pH(n-1) = 9,5773295276932

PH_N2_C2_SIGN       .WORD 0H
PH_N2_C2_EXP        .WORD 4
PH_N2_C2_HI         .WORD 4C9EH
PH_N2_C2_LO         .WORD 4F34H       ; 11 pH(n-2) = 9,5772995076161

; CENTRO VETORIAL - CLUSTER 3:
Q3_N1_C3_SIGN       .WORD 0H
Q3_N1_C3_EXP        .WORD -1
Q3_N1_C3_HI         .WORD 75C2H
Q3_N1_C3_LO         .WORD 4DCEH       ; 12 Q3(n-1) = 0,4599960926463

Q3_N2_C3_SIGN       .WORD 0H
Q3_N2_C3_EXP        .WORD -1
```

```
Q3_N2_C3_HI         .WORD 761DH
Q3_N2_C3_LO         .WORD 0CC11H      ; 13 Q3(n-2) = 0,4613921680873

Q3_N3_C3_SIGN       .WORD 0H
Q3_N3_C3_EXP        .WORD -1
Q3_N3_C3_HI         .WORD 7644H
Q3_N3_C3_LO         .WORD 0E5BH       ; 14 Q3(n-3) = 0,4619759532356

Q3_N4_C3_SIGN       .WORD 0H
Q3_N4_C3_EXP        .WORD -1
Q3_N4_C3_HI         .WORD 7672H
Q3_N4_C3_LO         .WORD 6F8BH       ; 15 Q3(n-4) = 0,4626836504270

PH_N1_C3_SIGN       .WORD 0H
PH_N1_C3_EXP        .WORD 3
PH_N1_C3_HI         .WORD 6EDCH
PH_N1_C3_LO         .WORD 6F78H       ; 16 pH(n-1) = 6,9288172409225

PH_N2_C3_SIGN       .WORD 0H
PH_N2_C3_EXP        .WORD 3
PH_N2_C3_HI         .WORD 6EDEH
PH_N2_C3_LO         .WORD 4CA4H       ; 17 pH(n-2) = 6,9292723098276

; CENTRO VETORIAL - CLUSTER 4:
Q3_N1_C4_SIGN       .WORD 0H
Q3_N1_C4_EXP        .WORD -1
Q3_N1_C4_HI         .WORD 7D7CH
Q3_N1_C4_LO         .WORD 0BE06H      ; 18 Q3(n-1) = 0,4901846661710

Q3_N2_C4_SIGN       .WORD 0H
Q3_N2_C4_EXP        .WORD -1
Q3_N2_C4_HI         .WORD 7D1DH
Q3_N2_C4_LO         .WORD 4676H       ; 19 Q3(n-2) = 0,4887279546435

Q3_N3_C4_SIGN       .WORD 0H
Q3_N3_C4_EXP        .WORD -1
Q3_N3_C4_HI         .WORD 7D27H
Q3_N3_C4_LO         .WORD 2BBDH       ; 20 Q3(n-3) = 0,4888789497517

Q3_N4_C4_SIGN       .WORD 0H
Q3_N4_C4_EXP        .WORD -1
Q3_N4_C4_HI         .WORD 7DA8H
Q3_N4_C4_LO         .WORD 05C9H       ; 21 Q3(n-4) = 0,4908450712974

PH_N1_C4_SIGN       .WORD 0H
PH_N1_C4_EXP        .WORD 3
PH_N1_C4_HI         .WORD 7801H
PH_N1_C4_LO         .WORD 77E3H       ; 22 pH(n-1) = 7,5003584722086

PH_N2_C4_SIGN       .WORD 0H
PH_N2_C4_EXP        .WORD 3
PH_N2_C4_HI         .WORD 7802H
PH_N2_C4_LO         .WORD 0AE90H      ; 23 pH(n-2) = 7,5006547564447

; CENTRO VETORIAL - CLUSTER 5:
Q3_N1_C5_SIGN       .WORD 0H
Q3_N1_C5_EXP        .WORD -1
Q3_N1_C5_HI         .WORD 5B4FH
Q3_N1_C5_LO         .WORD 0D0E5H      ; 24 Q3(n-1) = 0,3566866454540

Q3_N2_C5_SIGN       .WORD 0H
Q3_N2_C5_EXP        .WORD -1
Q3_N2_C5_HI         .WORD 5B5BH
Q3_N2_C5_LO         .WORD 0A9BCH      ; 25 Q3(n-2) = 0,3568674167060

Q3_N3_C5_SIGN       .WORD 0H
Q3_N3_C5_EXP        .WORD -1
```

```
Q3_N3_C5_HI         .WORD 5B48H
Q3_N3_C5_LO         .WORD 1F10H      ; 26 Q3(n-3) = 0,3565692341750

Q3_N4_C5_SIGN       .WORD 0H
Q3_N4_C5_EXP        .WORD -1
Q3_N4_C5_HI         .WORD 5B54H
Q3_N4_C5_LO         .WORD 2F0AH      ; 27 Q3(n-4) = 0,3567532919332

PH_N1_C5_SIGN       .WORD 0H
PH_N1_C5_EXP        .WORD 3
PH_N1_C5_HI         .WORD 5B65H
PH_N1_C5_LO         .WORD 6F60H      ; 28 pH(n-1) = 5,7122644176846

PH_N2_C5_SIGN       .WORD 0H
PH_N2_C5_EXP        .WORD 3
PH_N2_C5_HI         .WORD 5B68H
PH_N2_C5_LO         .WORD 5D3FH      ; 29 pH(n-2) = 5,7129795522048

; CENTRO VETORIAL - CLUSTER 6:
Q3_N1_C6_SIGN       .WORD 0H
Q3_N1_C6_EXP        .WORD -1
Q3_N1_C6_HI         .WORD 61C4H
Q3_N1_C6_LO         .WORD 409EH      ; 30 Q3(n-1) = 0,3819008240793

Q3_N2_C6_SIGN       .WORD 0H
Q3_N2_C6_EXP        .WORD -1
Q3_N2_C6_HI         .WORD 61CFH
Q3_N2_C6_LO         .WORD 9F07H      ; 31 Q3(n-2) = 0,3820742982023

Q3_N3_C6_SIGN       .WORD 0H
Q3_N3_C6_EXP        .WORD -1
Q3_N3_C6_HI         .WORD 6210H
Q3_N3_C6_LO         .WORD 49F8H      ; 32 Q3(n-3) = 0,3830610495351

Q3_N4_C6_SIGN       .WORD 0H
Q3_N4_C6_EXP        .WORD -1
Q3_N4_C6_HI         .WORD 6209H
Q3_N4_C6_LO         .WORD 0CCDH      ; 33 Q3(n-4) = 0,3829505921809

PH_N1_C6_SIGN       .WORD 0H
PH_N1_C6_EXP        .WORD 3
PH_N1_C6_HI         .WORD 60BBH
PH_N1_C6_LO         .WORD 24D6H      ; 34 pH(n-1) = 6,0456894253529

PH_N2_C6_SIGN       .WORD 0H
PH_N2_C6_EXP        .WORD 3
PH_N2_C6_HI         .WORD 60BCH
PH_N2_C6_LO         .WORD 0F785H     ; 35 pH(n-2) = 6,0461344906026

; CENTRO VETORIAL - CLUSTER 7:
Q3_N1_C7_SIGN       .WORD 0H
Q3_N1_C7_EXP        .WORD -1
Q3_N1_C7_HI         .WORD 53FFH
Q3_N1_C7_LO         .WORD 6DFEH      ; 36 Q3(n-1) = 0,3281162971704

Q3_N2_C7_SIGN       .WORD 0H
Q3_N2_C7_EXP        .WORD -1
Q3_N2_C7_HI         .WORD 5371H
Q3_N2_C7_LO         .WORD 0E9FFH     ; 37 Q3(n-2) = 0,3259569403759

Q3_N3_C7_SIGN       .WORD 0H
Q3_N3_C7_EXP        .WORD -1
Q3_N3_C7_HI         .WORD 5277H
Q3_N3_C7_LO         .WORD 7A76H      ; 38 Q3(n-3) = 0,3221355950559

Q3_N4_C7_SIGN       .WORD 0H
Q3_N4_C7_EXP        .WORD -1
```

```
Q3_N4_C7_HI         .WORD 52DBH
Q3_N4_C7_LO         .WORD 57D3H      ; 39 Q3(n-4) = 0,3236594094867

PH_N1_C7_SIGN       .WORD 0H
PH_N1_C7_EXP        .WORD 3
PH_N1_C7_HI         .WORD 44AEH
PH_N1_C7_LO         .WORD 6A17H      ; 40 pH(n-1) = 4,2925816448634

PH_N2_C7_SIGN       .WORD 0H
PH_N2_C7_EXP        .WORD 3
PH_N2_C7_HI         .WORD 44B4H
PH_N2_C7_LO         .WORD 0EFD1H      ; 41 pH(n-2) = 4,2941740204742

; CENTRO VETORIAL - CLUSTER 8:
Q3_N1_C8_SIGN       .WORD 0H
Q3_N1_C8_EXP        .WORD -1
Q3_N1_C8_HI         .WORD 70B7H
Q3_N1_C8_LO         .WORD 0C28CH       ; 42 Q3(n-1) = 0,4403039543875

Q3_N2_C8_SIGN       .WORD 0H
Q3_N2_C8_EXP        .WORD -1
Q3_N2_C8_HI         .WORD 7096H
Q3_N2_C8_LO         .WORD 8797H      ; 43 Q3(n-2) = 0,4397969002058

Q3_N3_C8_SIGN       .WORD 0H
Q3_N3_C8_EXP        .WORD -1
Q3_N3_C8_HI         .WORD 706CH
Q3_N3_C8_LO         .WORD 4CDEH      ; 44 Q3(n-3) = 0,4391525309527

Q3_N4_C8_SIGN       .WORD 0H
Q3_N4_C8_EXP        .WORD -1
Q3_N4_C8_HI         .WORD 7063H
Q3_N4_C8_LO         .WORD 0F99BH       ; 45 Q3(n-4) = 0,4390254977584

PH_N1_C8_SIGN       .WORD 0H
PH_N1_C8_EXP        .WORD 3
PH_N1_C8_HI         .WORD 6A9EH
PH_N1_C8_LO         .WORD 0EEDAH       ; 46 pH(n-1) = 6,6638020064570

PH_N2_C8_SIGN       .WORD 0H
PH_N2_C8_EXP        .WORD 3
PH_N2_C8_HI         .WORD 6AA1H
PH_N2_C8_LO         .WORD 274CH      ; 47 pH(n-2) = 6,6643441159787

; CENTRO VETORIAL - CLUSTER 9:
Q3_N1_C9_SIGN       .WORD 0H
Q3_N1_C9_EXP        .WORD -1
Q3_N1_C9_HI         .WORD 7F7EH
Q3_N1_C9_LO         .WORD 9900H      ; 48 Q3(n-1) = 0,4980254769030

Q3_N2_C9_SIGN       .WORD 0H
Q3_N2_C9_EXP        .WORD -1
Q3_N2_C9_HI         .WORD 7EFDH
Q3_N2_C9_LO         .WORD 0E808H       ; 49 Q3(n-2) = 0,4960618037863

Q3_N3_C9_SIGN       .WORD 0H
Q3_N3_C9_EXP        .WORD -1
Q3_N3_C9_HI         .WORD 7E04H
Q3_N3_C9_LO         .WORD 0DA0CH       ; 50 Q3(n-3) = 0,4922615316590

Q3_N4_C9_SIGN       .WORD 0H
Q3_N4_C9_EXP        .WORD -1
Q3_N4_C9_HI         .WORD 7E54H
Q3_N4_C9_LO         .WORD 93D1H      ; 51 Q3(n-4) = 0,4934780488683

PH_N1_C9_SIGN       .WORD 0H
PH_N1_C9_EXP        .WORD 3
```

```
PH_N1_C9_HI         .WORD 7DB1H
PH_N1_C9_LO         .WORD 0C8C3H      ; 52 pH(n-1) = 7,8559043505173

PH_N2_C9_SIGN       .WORD 0H
PH_N2_C9_EXP        .WORD 3
PH_N2_C9_HI         .WORD 7DB2H
PH_N2_C9_LO         .WORD 0C9A7H      ; 53 pH(n-2) = 7,8561493435046

; CENTRO VETORIAL - CLUSTER 10:
Q3_N1_C10_SIGN      .WORD 0H
Q3_N1_C10_EXP       .WORD 0H
Q3_N1_C10_HI        .WORD 424FH
Q3_N1_C10_LO        .WORD 13F7H       ; 54 Q3(n-1) = 0,5180382684389

Q3_N2_C10_SIGN      .WORD 0H
Q3_N2_C10_EXP       .WORD 0H
Q3_N2_C10_HI        .WORD 429AH
Q3_N2_C10_LO        .WORD 52AAH       ; 55 Q3(n-2) = 0,5203345612864

Q3_N3_C10_SIGN      .WORD 0H
Q3_N3_C10_EXP       .WORD 0H
Q3_N3_C10_HI        .WORD 42C4H
Q3_N3_C10_LO        .WORD 0F1C8H      ; 56 Q3(n-3) = 0,5216352677688

Q3_N4_C10_SIGN      .WORD 0H
Q3_N4_C10_EXP       .WORD 0H
Q3_N4_C10_HI        .WORD 42EBH
Q3_N4_C10_LO        .WORD 6485H       ; 57 Q3(n-4) = 0,5228086134957

PH_N1_C10_SIGN      .WORD 0H
PH_N1_C10_EXP       .WORD 4
PH_N1_C10_HI        .WORD 4A73H
PH_N1_C10_LO        .WORD 0A327H      ; 58 pH(n-1) = 9,3064635338741

PH_N2_C10_SIGN      .WORD 0H
PH_N2_C10_EXP       .WORD 4
PH_N2_C10_HI        .WORD 4A72H
PH_N2_C10_LO        .WORD 7613H       ; 59 pH(n-2) = 9,3058892686903

; CENTRO VETORIAL - CLUSTER 11:
Q3_N1_C11_SIGN      .WORD 0H
Q3_N1_C11_EXP       .WORD -1
Q3_N1_C11_HI        .WORD 57B4H
Q3_N1_C11_LO        .WORD 25B0H       ; 60 Q3(n-1) = 0,3425925782744

Q3_N2_C11_SIGN      .WORD 0H
Q3_N2_C11_EXP       .WORD -1
Q3_N2_C11_HI        .WORD 580DH
Q3_N2_C11_LO        .WORD 0D5D7H      ; 61 Q3(n-2) = 0,3439611100136

Q3_N3_C11_SIGN      .WORD 0H
Q3_N3_C11_EXP       .WORD -1
Q3_N3_C11_HI        .WORD 580EH
Q3_N3_C11_LO        .WORD 0CF75H      ; 62 Q3(n-3) = 0,3439759883835

Q3_N4_C11_SIGN      .WORD 0H
Q3_N4_C11_EXP       .WORD -1
Q3_N4_C11_HI        .WORD 57D5H
Q3_N4_C11_LO        .WORD 10BBH       ; 63 Q3(n-4) = 0,3430948693593

PH_N1_C11_SIGN      .WORD 0H
PH_N1_C11_EXP       .WORD 3
PH_N1_C11_HI        .WORD 5664H
PH_N1_C11_LO        .WORD 4416H       ; 64 pH(n-1) = 5,3994789928736

PH_N2_C11_SIGN      .WORD 0H
PH_N2_C11_EXP       .WORD 3
```

```
PH_N2_C11_HI        .WORD 5666H
PH_N2_C11_LO        .WORD 73A8H       ; 65 pH(n-2) = 5,4000126431111

; CENTRO VETORIAL - CLUSTER 12:
Q3_N1_C12_SIGN      .WORD 0H
Q3_N1_C12_EXP       .WORD -1
Q3_N1_C12_HI        .WORD 52F4H
Q3_N1_C12_LO        .WORD 7354H       ; 66 Q3(n-1) = 0,3240425186326

Q3_N2_C12_SIGN      .WORD 0H
Q3_N2_C12_EXP       .WORD -1
Q3_N2_C12_HI        .WORD 51BFH
Q3_N2_C12_LO        .WORD 67BFH       ; 67 Q3(n-2) = 0,3193268624523

Q3_N3_C12_SIGN      .WORD 0H
Q3_N3_C12_EXP       .WORD -1
Q3_N3_C12_HI        .WORD 504BH
Q3_N3_C12_LO        .WORD 1998H       ; 68 Q3(n-3) = 0,3136459347252

Q3_N4_C12_SIGN      .WORD 0H
Q3_N4_C12_EXP       .WORD -1
Q3_N4_C12_HI        .WORD 503EH
Q3_N4_C12_LO        .WORD 0F1AEH      ; 69 Q3(n-4) = 0,3134604501874

PH_N1_C12_SIGN      .WORD 0H
PH_N1_C12_EXP       .WORD 2
PH_N1_C12_HI        .WORD 7CDAH
PH_N1_C12_LO        .WORD 0AA53H      ; 70 pH(n-1) = 3,9016925445694

PH_N2_C12_SIGN      .WORD 0H
PH_N2_C12_EXP       .WORD 2
PH_N2_C12_HI        .WORD 7CE7H
PH_N2_C12_LO        .WORD 0B8FH       ; 71 pH(n-2) = 3,9032037543296

; CENTRO VETORIAL - CLUSTER 13:
Q3_N1_C13_SIGN      .WORD 0H
Q3_N1_C13_EXP       .WORD 0H
Q3_N1_C13_HI        .WORD 4041H
Q3_N1_C13_LO        .WORD 0ED79H      ; 72 Q3(n-1) = 0,5020119514484

Q3_N2_C13_SIGN      .WORD 0H
Q3_N2_C13_EXP       .WORD 0H
Q3_N2_C13_HI        .WORD 4097H
Q3_N2_C13_LO        .WORD 9968H       ; 73 Q3(n-2) = 0,5046264417883

Q3_N3_C13_SIGN      .WORD 0H
Q3_N3_C13_EXP       .WORD 0H
Q3_N3_C13_HI        .WORD 4131H
Q3_N3_C13_LO        .WORD 8FA1H       ; 74 Q3(n-3) = 0,5093249831945

Q3_N4_C13_SIGN      .WORD 0H
Q3_N4_C13_EXP       .WORD 0H
Q3_N4_C13_HI        .WORD 4104H
Q3_N4_C13_LO        .WORD 0AFFH       ; 75 Q3(n-4) = 0,5079358813337

PH_N1_C13_SIGN      .WORD 0H
PH_N1_C13_EXP       .WORD 4
PH_N1_C13_HI        .WORD 4544H
PH_N1_C13_LO        .WORD 0B645H      ; 76 pH(n-1) = 8,6585507744678

PH_N2_C13_SIGN      .WORD 0H
PH_N2_C13_EXP       .WORD 4
PH_N2_C13_HI        .WORD 4541H
PH_N2_C13_LO        .WORD 71C3H       ; 77 pH(n-2) = 8,6569552668014

; CENTRO VETORIAL - CLUSTER 14:
Q3_N1_C14_SIGN      .WORD 0H
```

```
Q3_N1_C14_EXP       .WORD -1
Q3_N1_C14_HI        .WORD 7FADH
Q3_N1_C14_LO        .WORD 99DDH       ; 78 Q3(n-1) = 0,4987426915229

Q3_N2_C14_SIGN      .WORD 0H
Q3_N2_C14_EXP       .WORD 0H
Q3_N2_C14_HI        .WORD 4027H
Q3_N2_C14_LO        .WORD 49F5H       ; 79 Q3(n-2) = 0,5011990018907

Q3_N3_C14_SIGN      .WORD 0H
Q3_N3_C14_EXP       .WORD 0H
Q3_N3_C14_HI        .WORD 405FH
Q3_N3_C14_LO        .WORD 3A45H       ; 80 Q3(n-3) = 0,5029061161490

Q3_N4_C14_SIGN      .WORD 0H
Q3_N4_C14_EXP       .WORD 0H
Q3_N4_C14_HI        .WORD 4040H
Q3_N4_C14_LO        .WORD 0D0D0H      ; 81 Q3(n-4) = 0,5019780172762

PH_N1_C14_SIGN      .WORD 0H
PH_N1_C14_EXP       .WORD 4
PH_N1_C14_HI        .WORD 4367H
PH_N1_C14_LO        .WORD 3B1EH       ; 82 pH(n-1) = 8,4254057293466

PH_N2_C14_SIGN      .WORD 0H
PH_N2_C14_EXP       .WORD 4
PH_N2_C14_HI        .WORD 4366H
PH_N2_C14_LO        .WORD 0E3C8H      ; 83 pH(n-2) = 8,4252391425036

; CENTRO VETORIAL - CLUSTER 15:
Q3_N1_C15_SIGN      .WORD 0H
Q3_N1_C15_EXP       .WORD -1
Q3_N1_C15_HI        .WORD 5419H
Q3_N1_C15_LO        .WORD 8961H       ; 84 Q3(n-1) = 0,3285146581476

Q3_N2_C15_SIGN      .WORD 0H
Q3_N2_C15_EXP       .WORD -1
Q3_N2_C15_HI        .WORD 5404H
Q3_N2_C15_LO        .WORD 612AH       ; 85 Q3(n-2) = 0,3281918266144

Q3_N3_C15_SIGN      .WORD 0H
Q3_N3_C15_EXP       .WORD -1
Q3_N3_C15_HI        .WORD 5471H
Q3_N3_C15_LO        .WORD 62F6H       ; 86 Q3(n-3) = 0,3298551417844

Q3_N4_C15_SIGN      .WORD 0H
Q3_N4_C15_EXP       .WORD -1
Q3_N4_C15_HI        .WORD 53C1H
Q3_N4_C15_LO        .WORD 8D00H       ; 87 Q3(n-4) = 0,3271721004546

PH_N1_C15_SIGN      .WORD 0H
PH_N1_C15_EXP       .WORD 3
PH_N1_C15_HI        .WORD 49AEH
PH_N1_C15_LO        .WORD 0EB10H      ; 88 pH(n-1) = 4,6052046424311

PH_N2_C15_SIGN      .WORD 0H
PH_N2_C15_EXP       .WORD 3
PH_N2_C15_HI        .WORD 49B6H
PH_N2_C15_LO        .WORD 0DBBEH      ; 89 pH(n-2) = 4,6071431574452

; CENTRO VETORIAL - CLUSTER 16:
Q3_N1_C16_SIGN      .WORD 0H
Q3_N1_C16_EXP       .WORD 0H
Q3_N1_C16_HI        .WORD 4027H
Q3_N1_C16_LO        .WORD 7EAFH       ; 90 Q3(n-1) = 0,5012052876145

Q3_N2_C16_SIGN      .WORD 0H
```

```
Q3_N2_C16_EXP       .WORD 0H
Q3_N2_C16_HI        .WORD 4013H
Q3_N2_C16_LO        .WORD 0CF2H       ; 91 Q3(n-2) = 0,50058137721 51

Q3_N3_C16_SIGN      .WORD 0H
Q3_N3_C16_EXP       .WORD -1
Q3_N3_C16_HI        .WORD 7F00H
Q3_N3_C16_LO        .WORD 8C3DH       ; 92 Q3(n-3) = 0,4961021087920

Q3_N4_C16_SIGN      .WORD 0H
Q3_N4_C16_EXP       .WORD -1
Q3_N4_C16_HI        .WORD 7EC7H
Q3_N4_C16_LO        .WORD 4296H       ; 93 Q3(n-4) = 0,4952279679703

PH_N1_C16_SIGN      .WORD 0H
PH_N1_C16_EXP       .WORD 4
PH_N1_C16_HI        .WORD 411BH
PH_N1_C16_LO        .WORD 4777H       ; 94 pH(n-1) = 8,1383199033806

PH_N2_C16_SIGN      .WORD 0H
PH_N2_C16_EXP       .WORD 4
PH_N2_C16_HI        .WORD 4116H
PH_N2_C16_LO        .WORD 0F630H      ; 95 pH(n-2) = 8,1362117535406

; CENTRO VETORIAL - CLUSTER 17:
Q3_N1_C17_SIGN      .WORD 0H
Q3_N1_C17_EXP       .WORD -1
Q3_N1_C17_HI        .WORD 5571H
Q3_N1_C17_LO        .WORD 0E144H      ; 96 Q3(n-1) = 0,3337689200753

Q3_N2_C17_SIGN      .WORD 0H
Q3_N2_C17_EXP       .WORD -1
Q3_N2_C17_HI        .WORD 55EDH
Q3_N2_C17_LO        .WORD 0183H       ; 97 Q3(n-2) = 0,3356476732124

Q3_N3_C17_SIGN      .WORD 0H
Q3_N3_C17_EXP       .WORD -1
Q3_N3_C17_HI        .WORD 5698H
Q3_N3_C17_LO        .WORD 3A57H       ; 98 Q3(n-3) = 0,3382603133038

Q3_N4_C17_SIGN      .WORD 0H
Q3_N4_C17_EXP       .WORD -1
Q3_N4_C17_HI        .WORD 5675H
Q3_N4_C17_LO        .WORD 71E4H       ; 99 Q3(n-4) = 0,3377295666720

PH_N1_C17_SIGN      .WORD 0H
PH_N1_C17_EXP       .WORD 3
PH_N1_C17_HI        .WORD 4FF5H
PH_N1_C17_LO        .WORD 99FCH       ; 100 pH(n-1) = 4,9974613051111

PH_N2_C17_SIGN      .WORD 0H
PH_N2_C17_EXP       .WORD 3
PH_N2_C17_HI        .WORD 4FF3H
PH_N2_C17_LO        .WORD 703EH       ; 101 pH(n-2) = 4,9969332159856

; CENTRO VETORIAL - CLUSTER 18:
Q3_N1_C18_SIGN      .WORD 0H
Q3_N1_C18_EXP       .WORD -1
Q3_N1_C18_HI        .WORD 7AAEH
Q3_N1_C18_LO        .WORD 3654H       ; 102 Q3(n-1) = 0,4792207675257

Q3_N2_C18_SIGN      .WORD 0H
Q3_N2_C18_EXP       .WORD -1
Q3_N2_C18_HI        .WORD 7A91H
Q3_N2_C18_LO        .WORD 0D569H      ; 103 Q3(n-2) = 0,4787877446943

Q3_N3_C18_SIGN      .WORD 0H
```

```
Q3_N3_C18_EXP       .WORD -1
Q3_N3_C18_HI        .WORD 7A7DH
Q3_N3_C18_LO        .WORD 8AD4H       ; 104 Q3(n-3) = 0,4784781233562

Q3_N4_C18_SIGN      .WORD 0H
Q3_N4_C18_EXP       .WORD -1
Q3_N4_C18_HI        .WORD 7A93H
Q3_N4_C18_LO        .WORD 26E6H       ; 105 Q3(n-4) = 0,4788078604883

PH_N1_C18_SIGN      .WORD 0H
PH_N1_C18_EXP       .WORD 3
PH_N1_C18_HI        .WORD 73A5H
PH_N1_C18_LO        .WORD 0EF88H      ; 106 pH(n-1) = 7,2280116385779

PH_N2_C18_SIGN      .WORD 0H
PH_N2_C18_EXP       .WORD 3
PH_N2_C18_HI        .WORD 73A7H
PH_N2_C18_LO        .WORD 0ACBAH      ; 107 pH(n-2) = 7,2284362084554

; CENTRO VETORIAL - CLUSTER 19:
Q3_N1_C19_SIGN      .WORD 0H
Q3_N1_C19_EXP       .WORD 0H
Q3_N1_C19_HI        .WORD 40A8H
Q3_N1_C19_LO        .WORD 0B0BCH      ; 108 Q3(n-1) = 0,5051480216005

Q3_N2_C19_SIGN      .WORD 0H
Q3_N2_C19_EXP       .WORD 0H
Q3_N2_C19_HI        .WORD 4106H
Q3_N2_C19_LO        .WORD 0FDE1H      ; 109 Q3(n-2) = 0,5080258703678

Q3_N3_C19_SIGN      .WORD 0H
Q3_N3_C19_EXP       .WORD 0H
Q3_N3_C19_HI        .WORD 4189H
Q3_N3_C19_LO        .WORD 4822H       ; 110 Q3(n-3) = 0,5120020072451

Q3_N4_C19_SIGN      .WORD 0H
Q3_N4_C19_EXP       .WORD 0H
Q3_N4_C19_HI        .WORD 416DH
Q3_N4_C19_LO        .WORD 0CB20H      ; 111 Q3(n-4) = 0,5111631304899

PH_N1_C19_SIGN      .WORD 0H
PH_N1_C19_EXP       .WORD 4
PH_N1_C19_HI        .WORD 47B3H
PH_N1_C19_LO        .WORD 81EBH       ; 112 pH(n-1) = 8,9626501414504

PH_N2_C19_SIGN      .WORD 0H
PH_N2_C19_EXP       .WORD 4
PH_N2_C19_HI        .WORD 47B3H
PH_N2_C19_LO        .WORD 0AE52H      ; 113 pH(n-2) = 8,9627348368923

; CENTRO VETORIAL - CLUSTER 20:
Q3_N1_C20_SIGN      .WORD 0H
Q3_N1_C20_EXP       .WORD 0H
Q3_N1_C20_HI        .WORD 4697H
Q3_N1_C20_LO        .WORD 0F56BH      ; 114 Q3(n-1) = 0,5515124106041

Q3_N2_C20_SIGN      .WORD 0H
Q3_N2_C20_EXP       .WORD 0H
Q3_N2_C20_HI        .WORD 46EAH
Q3_N2_C20_LO        .WORD 19EEH       ; 115 Q3(n-2) = 0,5540192045701

Q3_N3_C20_SIGN      .WORD 0H
Q3_N3_C20_EXP       .WORD 0H
Q3_N3_C20_HI        .WORD 472FH
Q3_N3_C20_LO        .WORD 0E6C3H      ; 116 Q3(n-3) = 0,5561493349603

Q3_N4_C20_SIGN      .WORD 0H
```

```
Q3_N4_C20_EXP       .WORD 0H
Q3_N4_C20_HI        .WORD 4721H
Q3_N4_C20_LO        .WORD 0E2C4H      ; 117 Q3(n-4) = 0,5557216124448

PH_N1_C20_SIGN      .WORD 0H
PH_N1_C20_EXP       .WORD 4
PH_N1_C20_HI        .WORD 4E7EH
PH_N1_C20_LO        .WORD 0A594H      ; 118 pH(n-1) = 9,8118392494484

PH_N2_C20_SIGN      .WORD 0H
PH_N2_C20_EXP       .WORD 4
PH_N2_C20_HI        .WORD 4E7EH
PH_N2_C20_LO        .WORD 0208H       ; 119 pH(n-2) = 9,8115273145509

;*********************************************************************************************
; SUBMODELOS LINEARES - SAÍDAS PARCIAIS:
; SUBMODELO 1:
B10_SIGN      .WORD 0H
B10_EXP       .WORD -1
B10_HI        .WORD 54D0H
B10_LO        .WORD 0E682H      ; 120 b10 = 0,33131256751638

B11_SIGN      .WORD 0H
B11_EXP       .WORD -4
B11_HI        .WORD 64A1H
B11_LO        .WORD 0CD99H      ; 121 b11 = 0,04913673996249

B12_SIGN      .WORD 0H
B12_EXP       .WORD -5
B12_HI        .WORD 707DH
B12_LO        .WORD 5F53H       ; 122 b12 = 0,02746331439865

B13_SIGN      .WORD 0FFFFH
B13_EXP       .WORD -5
B13_HI        .WORD 4682H
B13_LO        .WORD 8137H       ; 123 b13 = -0,01721430276881

B14_SIGN      .WORD 0FFFFH
B14_EXP       .WORD -5
B14_HI        .WORD 4332H
B14_LO        .WORD 3F7AH       ; 124 b14 = -0,01640534205397

B15_SIGN      .WORD 0H
B15_EXP       .WORD -1
B15_HI        .WORD 67C8H
B15_LO        .WORD 0E62FH      ; 125 b15 = 0,40540922782838

B16_SIGN      .WORD 0H
B16_EXP       .WORD 0H
B16_HI        .WORD 44E6H
B16_LO        .WORD 5A0EH       ; 126 b16 = 0,53827977826054

; SUBMODELO 2:
B20_SIGN      .WORD 0H
B20_EXP       .WORD 0H
B20_HI        .WORD 62A9H
B20_LO        .WORD 0AAF8H      ; 127 b20 = 0,77080285157864

B21_SIGN      .WORD 0H
B21_EXP       .WORD -2
B21_HI        .WORD 4793H
B21_LO        .WORD 3CD1H       ; 128 b21 = 0,13979520845857

B22_SIGN      .WORD 0H
B22_EXP       .WORD -3
B22_HI        .WORD 5BF7H
B22_LO        .WORD 5D3BH       ; 129 b22 = 0,08981080694931
```

```
B23_SIGN    .WORD 0H
B23_EXP     .WORD -8
B23_HI      .WORD 72E9H
B23_LO      .WORD 0B5BEH     ; 130 b23 = 0,00350686430142

B24_SIGN    .WORD 0FFFFH
B24_EXP     .WORD -9
B24_HI      .WORD 51C6H
B24_LO      .WORD 0699H      ; 131 b24 = -0,00124776516968

B25_SIGN    .WORD 0H
B25_EXP     .WORD -1
B25_HI      .WORD 5D7AH
B25_LO      .WORD 71BAH      ; 132 b25 = 0,36514960079241

B26_SIGN    .WORD 0H
B26_EXP     .WORD 0H
B26_HI      .WORD 455EH
B26_LO      .WORD 9E86H      ; 133 b26 = 0,54195004977586

; SUBMODELO 3:
B30_SIGN    .WORD 0H
B30_EXP     .WORD 0H
B30_HI      .WORD 4B77H
B30_LO      .WORD 826AH      ; 134 b30 = 0,58958463833480

B31_SIGN    .WORD 0H
B31_EXP     .WORD -3
B31_HI      .WORD 677FH
B31_LO      .WORD 72DFH      ; 135 b31 = 0,10107211577160

B32_SIGN    .WORD 0H
B32_EXP     .WORD -4
B32_HI      .WORD 7166H
B32_LO      .WORD 0E509H     ; 136 b32 = 0,05537203725797

B33_SIGN    .WORD 0FFFFH
B33_EXP     .WORD -6
B33_HI      .WORD 6D5DH
B33_LO      .WORD 4882H      ; 137 b33 = -0,01335014497121

B34_SIGN    .WORD 0FFFFH
B34_EXP     .WORD -6
B34_HI      .WORD 4007H
B34_LO      .WORD 5C0AH      ; 138 b34 = -0,00781600929271

B35_SIGN    .WORD 0H
B35_EXP     .WORD -1
B35_HI      .WORD 6EB2H
B35_LO      .WORD 6AF9H      ; 139 b35 = 0,43240994053683

B36_SIGN    .WORD 0H
B36_EXP     .WORD -1
B36_HI      .WORD 7904H
B36_LO      .WORD 0F2A5H     ; 140 b36 = 0,47273174781889

; SUBMODELO 4:
B40_SIGN    .WORD 0H
B40_EXP     .WORD -1
B40_HI      .WORD 66A7H
B40_LO      .WORD 5542H      ; 141 b40 = 0,40099079958627

B41_SIGN    .WORD 0H
B41_EXP     .WORD -1
B41_HI      .WORD 7475H
B41_LO      .WORD 7A2EH      ; 142 b41 = 0,45491756070336
```

```
B42_SIGN    .WORD 0H
B42_EXP     .WORD -2
B42_HI      .WORD 679EH
B42_LO      .WORD 07B1H     ; 143 b42 = 0,20237754859945

B43_SIGN    .WORD 0FFFFH
B43_EXP     .WORD -7
B43_HI      .WORD 6DF2H
B43_LO      .WORD 0C854H    ; 144 b43 = -0,00671071589752

B44_SIGN    .WORD 0FFFFH
B44_EXP     .WORD -10
B44_HI      .WORD 45C1H
B44_LO      .WORD 3ADAH     ; 145 b44 = -0,00053218692202

B45_SIGN    .WORD 0H
B45_EXP     .WORD 0H
B45_HI      .WORD 42E4H
B45_LO      .WORD 9D7FH     ; 146 b45 = 0,52260178262772

B46_SIGN    .WORD 0H
B46_EXP     .WORD -1
B46_HI      .WORD 61A0H
B46_LO      .WORD 377BH     ; 147 b46 = 0,38135096318396

; SUBMODELO 5:
B50_SIGN    .WORD 0H
B50_EXP     .WORD -3
B50_HI      .WORD 7903H
B50_LO      .WORD 0F555H    ; 148 b50 = 0,11817916229610

B51_SIGN    .WORD 0H
B51_EXP     .WORD -2
B51_HI      .WORD 40B0H
B51_LO      .WORD 0A3D4H    ; 149 b51 = 0,12634765585788

B52_SIGN    .WORD 0H
B52_EXP     .WORD -3
B52_HI      .WORD 4302H
B52_LO      .WORD 0C9BDH    ; 150 b52 = 0,06544032304786

B53_SIGN    .WORD 0FFFFH
B53_EXP     .WORD -7
B53_HI      .WORD 7E41H
B53_LO      .WORD 5467H     ; 151 b53 = -0,00770600549927

B54_SIGN    .WORD 0FFFFH
B54_EXP     .WORD -6
B54_HI      .WORD 4954H
B54_LO      .WORD 0CA76H    ; 152 b54 = -0,00895156424525

B55_SIGN    .WORD 0H
B55_EXP     .WORD -1
B55_HI      .WORD 7BDBH
B55_LO      .WORD 1004H     ; 153 b55 = 0,48381137938910

B56_SIGN    .WORD 0H
B56_EXP     .WORD -1
B56_HI      .WORD 7BD2H
B56_LO      .WORD 959EH     ; 154 b56 = 0,48368201363852

; SUBMODELO 6:
B60_SIGN    .WORD 0H
B60_EXP     .WORD -2
B60_HI      .WORD 4181H
B60_LO      .WORD 0ECB7H    ; 155 b60 = 0,12794437159084
```

```
B61_SIGN    .WORD 0H
B61_EXP     .WORD -3
B61_HI      .WORD 57E9H
B61_LO      .WORD 0EEE1H     ; 156 b61 = 0,08585332154972

B62_SIGN    .WORD 0H
B62_EXP     .WORD -4
B62_HI      .WORD 5F9AH
B62_LO      .WORD 0B231H     ; 157 b62 = 0,04668177807147

B63_SIGN    .WORD 0FFFFH
B63_EXP     .WORD -7
B63_HI      .WORD 487CH
B63_LO      .WORD 0F629H     ; 158 b63 = -0,00442432440977

B64_SIGN    .WORD 0FFFFH
B64_EXP     .WORD -7
B64_HI      .WORD 6336H
B64_LO      .WORD 9C05H      ; 159 b64 = -0,00605550037643

B65_SIGN    .WORD 0H
B65_EXP     .WORD -1
B65_HI      .WORD 708AH
B65_LO      .WORD 6774H      ; 160 b65 = 0,43961187928072

B66_SIGN    .WORD 0H
B66_EXP     .WORD 0H
B66_HI      .WORD 43EEH
B66_LO      .WORD 0C1FBH     ; 161 b66 = 0,53072380798331

; SUBMODELO 7:
B70_SIGN    .WORD 0FFFFH
B70_EXP     .WORD -2
B70_HI      .WORD 4FC1H
B70_LO      .WORD 0D7FCH     ; 162 b70 = -0,15577578499069

B71_SIGN    .WORD 0H
B71_EXP     .WORD 1
B71_HI      .WORD 41B0H
B71_LO      .WORD 0BBC6H     ; 163 b71 = 1,02641195614584

B72_SIGN    .WORD 0H
B72_EXP     .WORD -1
B72_HI      .WORD 7355H
B72_LO      .WORD 0A87AH     ; 164 b72 = 0,45052578908584

B73_SIGN    .WORD 0FFFFH
B73_EXP     .WORD -7
B73_HI      .WORD 7846H
B73_LO      .WORD 8285H      ; 165 b73 = -0,00734102960593

B74_SIGN    .WORD 0H
B74_EXP     .WORD -7
B74_HI      .WORD 5487H
B74_LO      .WORD 0FAA4H     ; 166 b74 = 0,00515937306095

B75_SIGN    .WORD 0H
B75_EXP     .WORD 0H
B75_HI      .WORD 4325H
B75_LO      .WORD 0F565H     ; 167 b75 = 0,52459590361591

B76_SIGN    .WORD 0H
B76_EXP     .WORD -1
B76_HI      .WORD 65F2H
B76_LO      .WORD 0B28EH     ; 168 b76 = 0,39823451966739
```

```
; SUBMODELO 8:
B80_SIGN    .WORD 0H
B80_EXP     .WORD -1
B80_HI      .WORD 5C5AH
B80_LO      .WORD 0BF69H     ; 169 b80 = 0,36075969996947

B81_SIGN    .WORD 0H
B81_EXP     .WORD -3
B81_HI      .WORD 5058H
B81_LO      .WORD 0B3C4H     ; 170 b81 = 0,07846337207682

B82_SIGN    .WORD 0H
B82_EXP     .WORD -4
B82_HI      .WORD 54FBH
B82_LO      .WORD 6747H      ; 171 b82 = 0,04149513898083

B83_SIGN    .WORD 0FFFFH
B83_EXP     .WORD -6
B83_HI      .WORD 7167H
B83_LO      .WORD 0B04CH     ; 172 b83 = -0,01384338791707

B84_SIGN    .WORD 0FFFFH
B84_EXP     .WORD -6
B84_HI      .WORD 7A85H
B84_LO      .WORD 4E23H      ; 173 b84 = -0,01495614300904

B85_SIGN    .WORD 0H
B85_EXP     .WORD -1
B85_HI      .WORD 696CH
B85_LO      .WORD 4450H      ; 174 b85 = 0,41180827104631

B86_SIGN    .WORD 0H
B86_EXP     .WORD 0H
B86_HI      .WORD 4376H
B86_LO      .WORD 0FE37H     ; 175 b86 = 0,52706887891981

; SUBMODELO 9:
B90_SIGN    .WORD 0H
B90_EXP     .WORD -1
B90_HI      .WORD 474FH
B90_LO      .WORD 727BH      ; 176 b90 = 0,27855601800848

B91_SIGN    .WORD 0H
B91_EXP     .WORD 0H
B91_HI      .WORD 76C4H
B91_LO      .WORD 0FA9CH     ; 177 b91 = 0,92788632043301

B92_SIGN    .WORD 0H
B92_EXP     .WORD -1
B92_HI      .WORD 5CE8H
B92_LO      .WORD 7200H      ; 178 b92 = 0,36292183405436

B93_SIGN    .WORD 0FFFFH
B93_EXP     .WORD -7
B93_HI      .WORD 40A9H
B93_LO      .WORD 6709H      ; 179 b93 = -0,00394663869755

B94_SIGN    .WORD 0FFFFH
B94_EXP     .WORD -8
B94_HI      .WORD 68CFH
B94_LO      .WORD 3E39H      ; 180 b94 = -0,00319853342211

B95_SIGN    .WORD 0H
B95_EXP     .WORD 0H
B95_HI      .WORD 46DFH
B95_LO      .WORD 0AAFEH     ; 181 b95 = 0,55370080400581
```

```
B96_SIGN      .WORD 0H
B96_EXP       .WORD -1
B96_HI        .WORD 5460H
B96_LO        .WORD 0EC5CH      ; 182 b96 = 0,32960393194160

; SUBMODELO 10:
B100_SIGN     .WORD 0H
B100_EXP      .WORD 0H
B100_HI       .WORD 5963H
B100_LO       .WORD 5039H       ; 183 b100 = 0,69834330365103

B101_SIGN     .WORD 0H
B101_EXP      .WORD -2
B101_HI       .WORD 7E10H
B101_LO       .WORD 0CFE5H      ; 184 b101 = 0,24622201605785

B102_SIGN     .WORD 0H
B102_EXP      .WORD -2
B102_HI       .WORD 5133H
B102_LO       .WORD 0B046H      ; 185 b102 = 0,15859747753301

B103_SIGN     .WORD 0H
B103_EXP      .WORD -10
B103_HI       .WORD 6EEAH
B103_LO       .WORD 9AD0H       ; 186 b103 = 0,00084622516432

B104_SIGN     .WORD 0FFFFH
B104_EXP      .WORD -12
B104_HI       .WORD 66D1H
B104_LO       .WORD 0FFB5H      ; 187 b104 = -0,00019611417391

B105_SIGN     .WORD 0H
B105_EXP      .WORD -1
B105_HI       .WORD 5ABFH
B105_LO       .WORD 5F2DH       ; 188 b105 = 0,35448260165190

B106_SIGN     .WORD 0H
B106_EXP      .WORD 0H
B106_HI       .WORD 4631H
B106_LO       .WORD 9958H       ; 189 b106 = 0,54838864112603

; SUBMODELO 11:
B110_SIGN     .WORD 0H
B110_EXP      .WORD -4
B110_HI       .WORD 528CH
B110_LO       .WORD 0ACB6H      ; 190 b110 = 0,04030737810288

B111_SIGN     .WORD 0H
B111_EXP      .WORD -1
B111_HI       .WORD 44F0H
B111_LO       .WORD 6E4DH       ; 191 b111 = 0,26929368372633

B112_SIGN     .WORD 0H
B112_EXP      .WORD -2
B112_HI       .WORD 486EH
B112_LO       .WORD 0F30H       ; 192 b112 = 0,14146468596695

B113_SIGN     .WORD 0FFFFH
B113_EXP      .WORD -8
B113_HI       .WORD 4C94H
B113_LO       .WORD 7F12H       ; 193 b113 = -0,00233703808484

B114_SIGN     .WORD 0H
B114_EXP      .WORD -8
B114_HI       .WORD 45A0H
B114_LO       .WORD 2237H       ; 194 b114 = 0,00212480230999
```

```
B115_SIGN      .WORD 0H
B115_EXP       .WORD -1
B115_HI        .WORD 7D5DH
B115_LO        .WORD 0D89EH      ; 195 b115 = 0,48971322877235

B116_SIGN      .WORD 0H
B116_EXP       .WORD -1
B116_HI        .WORD 79D8H
B116_LO        .WORD 29DBH       ; 196 b116 = 0,47595464323213

; SUBMODELO 12:
B120_SIGN      .WORD 0FFFFH
B120_EXP       .WORD -5
B120_HI        .WORD 462DH
B120_LO        .WORD 810CH       ; 197 b120 = -0,01713323983716

B121_SIGN      .WORD 0H
B121_EXP       .WORD -1
B121_HI        .WORD 76F7H
B121_LO        .WORD 0BE33H      ; 198 b121 = 0,46471775770316

B122_SIGN      .WORD 0H
B122_EXP       .WORD -2
B122_HI        .WORD 78B3H
B122_LO        .WORD 0C83EH      ; 199 b122 = 0,23574662931642

B123_SIGN      .WORD 0H
B123_EXP       .WORD -4
B123_HI        .WORD 425CH
B123_LO        .WORD 9AAFH       ; 200 b123 = 0,03240319105494

B124_SIGN      .WORD 0H
B124_EXP       .WORD -6
B124_HI        .WORD 77DDH
B124_LO        .WORD 9D8DH       ; 201 b124 = 0,01463204166036

B125_SIGN      .WORD 0H
B125_EXP       .WORD -1
B125_HI        .WORD 7EB0H
B125_LO        .WORD 3BB5H       ; 202 b125 = 0,49487660564389

B126_SIGN      .WORD 0H
B126_EXP       .WORD -1
B126_HI        .WORD 7211H
B126_LO        .WORD 0B08H       ; 203 b126 = 0,44557255700795

; SUBMODELO 13:
B130_SIGN      .WORD 0H
B130_EXP       .WORD -1
B130_HI        .WORD 4724H
B130_LO        .WORD 17C1H       ; 204 b130 = 0,27789448213612

B131_SIGN      .WORD 0H
B131_EXP       .WORD 0H
B131_HI        .WORD 7F4CH
B131_LO        .WORD 21FDH       ; 205 b131 = 0,99451088767321

B132_SIGN      .WORD 0H
B132_EXP       .WORD -1
B132_HI        .WORD 6C00H
B132_LO        .WORD 295BH       ; 206 b132 = 0,42187746493254

B133_SIGN      .WORD 0H
B133_EXP       .WORD -5
B133_HI        .WORD 4471H
B133_LO        .WORD 0D30FH      ; 207 b133 = 0,01671011394922
```

```
B134_SIGN       .WORD 0H
B134_EXP        .WORD -7H
B134_HI         .WORD 5E00H
B134_LO         .WORD 9AAEH       ; 208 b134 = 0,00573744874301

B135_SIGN       .WORD 0H
B135_EXP        .WORD 0H
B135_HI         .WORD 4537H
B135_LO         .WORD 9186H       ; 209 b135 = 0,54075831448599

B136_SIGN       .WORD 0H
B136_EXP        .WORD -1
B136_HI         .WORD 5810H
B136_LO         .WORD 6848H       ; 210 b136 = 0,34400035620880

; SUBMODELO 14:
B140_SIGN       .WORD 0H
B140_EXP        .WORD -1
B140_HI         .WORD 7A39H
B140_LO         .WORD 321EH       ; 211 b140 = 0,47743523829900

B141_SIGN       .WORD 0H
B141_EXP        .WORD 1
B141_HI         .WORD 5C8FH
B141_LO         .WORD 0E59EH      ; 212 b141 = 1,44628277278551

B142_SIGN       .WORD 0H
B142_EXP        .WORD 0H
B142_HI         .WORD 4DD9H
B142_LO         .WORD 6EDAH       ; 213 b142 = 0,60819802890475

B143_SIGN       .WORD 0FFFFH
B143_EXP        .WORD -6
B143_HI         .WORD 433EH
B143_LO         .WORD 72FBH       ; 214 b143 = -0,00820848900937

B144_SIGN       .WORD 0H
B144_EXP        .WORD -13
B144_HI         .WORD 464DH
B144_LO         .WORD 6C41H       ; 215 b144 = 0,00006704562481

B145_SIGN       .WORD 0H
B145_EXP        .WORD 0H
B145_HI         .WORD 4029H
B145_LO         .WORD 64ECH       ; 216 b145 = 0,50126325146228

B146_SIGN       .WORD 0H
B146_EXP        .WORD -1
B146_HI         .WORD 51FFH
B146_LO         .WORD 0C567H      ; 217 b146 = 0,32030900725251

; SUBMODELO 15:
B150_SIGN       .WORD 0FFFFH
B150_EXP        .WORD -3
B150_HI         .WORD 7C52H
B150_LO         .WORD 0C105H      ; 218 b150 = -0,12140943136318

B151_SIGN       .WORD 0H
B151_EXP        .WORD 1
B151_HI         .WORD 4ACEH
B151_LO         .WORD 3D10H       ; 219 b151 = 1,16883780034167

B152_SIGN       .WORD 0H
B152_EXP        .WORD 0H
B152_HI         .WORD 44D6H
B152_LO         .WORD 0AED9H      ; 220 b152 = 0,53780160497000
```

```
B153_SIGN       .WORD 0H
B153_EXP        .WORD -7
B153_HI         .WORD 4F1DH
B153_LO         .WORD 0C4FFH       ; 221 b153 = 0,00482887495103

B154_SIGN       .WORD 0FFFFH
B154_EXP        .WORD -9
B154_HI         .WORD 6802H
B154_LO         .WORD 4952H        ; 222 b154 = -0,00158705034328

B155_SIGN       .WORD 0H
B155_EXP        .WORD -1
B155_HI         .WORD 7B69H
B155_LO         .WORD 681EH        ; 223 b155 = 0,48207712863516

B156_SIGN       .WORD 0H
B156_EXP        .WORD -1
B156_HI         .WORD 6BA6H
B156_LO         .WORD 0F15BH       ; 224 b156 = 0,42051609497319

; SUBMODELO 16:
B160_SIGN       .WORD 0H
B160_EXP        .WORD -1
B160_HI         .WORD 5898H
B160_LO         .WORD 982EH        ; 225 b160 = 0,34607840663568

B161_SIGN       .WORD 0H
B161_EXP        .WORD 1
B161_HI         .WORD 599FH
B161_LO         .WORD 006BH        ; 226 b161 = 1,40032968988121

B162_SIGN       .WORD 0H
B162_EXP        .WORD 0H
B162_HI         .WORD 464EH
B162_LO         .WORD 2260H        ; 227 b162 = 0,54925946914117

B163_SIGN       .WORD 0FFFFH
B163_EXP        .WORD -7
B163_HI         .WORD 5F4AH
B163_LO         .WORD 1721H        ; 228 b163 = -0,00581600435907

B164_SIGN       .WORD 0FFFFH
B164_EXP        .WORD -6
B164_HI         .WORD 786DH
B164_LO         .WORD 0DA1H        ; 229 b164 = -0,01470043813714

B165_SIGN       .WORD 0H
B165_EXP        .WORD 0H
B165_HI         .WORD 4500H
B165_LO         .WORD 0C5AH        ; 230 b165 = 0,53906397244732

B166_SIGN       .WORD 0H
B166_EXP        .WORD -1
B166_HI         .WORD 4CD6H
B166_LO         .WORD 3CF6H        ; 231 b166 = 0,30014401434356

; SUBMODELO 17:
B170_SIGN       .WORD 0FFFFH
B170_EXP        .WORD -4
B170_HI         .WORD 70FFH
B170_LO         .WORD 76F3H        ; 232 b170 = -0,05517476013385

B171_SIGN       .WORD 0H
B171_EXP        .WORD 0H
B171_HI         .WORD 542BH
B171_LO         .WORD 2AA8H        ; 233 b171 = 0,65756734070885
```

```
B172_SIGN     .WORD 0H
B172_EXP      .WORD -1
B172_HI       .WORD 40EEH
B172_LO       .WORD 53A8H      ; 234 b172 = 0,25363657820562

B173_SIGN     .WORD 0H
B173_EXP      .WORD -6
B173_HI       .WORD 6541H
B173_LO       .WORD 679EH      ; 235 b173 = 0,01236028897692

B174_SIGN     .WORD 0H
B174_EXP      .WORD -7
B174_HI       .WORD 6D1DH
B174_LO       .WORD 4D76H      ; 236 b174 = 0,00665981831288

B175_SIGN     .WORD 0H
B175_EXP      .WORD 0H
B175_HI       .WORD 4D85H
B175_LO       .WORD 722AH      ; 237 b175 = 0,60563494725797

B176_SIGN     .WORD 0H
B176_EXP      .WORD -1
B176_HI       .WORD 579CH
B176_LO       .WORD 8471H      ; 238 b176 = 0,34223201530239

; SUBMODELO 18:
B180_SIGN     .WORD 0H
B180_EXP      .WORD 0H
B180_HI       .WORD 4075H
B180_LO       .WORD 2C6CH      ; 239 b180 = 0,50357585207922

B181_SIGN     .WORD 0H
B181_EXP      .WORD -2
B181_HI       .WORD 741EH
B181_LO       .WORD 0CC63H     ; 240 b181 = 0,22679747303939

B182_SIGN     .WORD 0H
B182_EXP      .WORD -3
B182_HI       .WORD 7357H
B182_LO       .WORD 0AE82H     ; 241 b182 = 0,11263916654329

B183_SIGN     .WORD 0H
B183_EXP      .WORD -9
B183_HI       .WORD 563CH
B183_LO       .WORD 7910H      ; 242 b183 = 0,00131586032516

B184_SIGN     .WORD 0FFFFH
B184_EXP      .WORD -10
B184_HI       .WORD 697CH
B184_LO       .WORD 42D3H      ; 243 b184 = -0,00080478969330

B185_SIGN     .WORD 0H
B185_EXP      .WORD -1
B185_HI       .WORD 7968H
B185_LO       .WORD 8B36H      ; 244 b185 = 0,47425146160793

B186_SIGN     .WORD 0H
B186_EXP      .WORD -1
B186_HI       .WORD 6EDBH
B186_LO       .WORD 24CFH      ; 245 b186 = 0,43303136884840

; SUBMODELO 19:
B190_SIGN     .WORD 0H
B190_EXP      .WORD -1
B190_HI       .WORD 5B39H
B190_LO       .WORD 1FB3H      ; 246 b190 = 0,35634039038177
```

```
B191_SIGN       .WORD 0H
B191_EXP        .WORD 0H
B191_HI         .WORD 438BH
B191_LO         .WORD 4CF2H       ; 247 b191 = 0,52768861583061

B192_SIGN       .WORD 0H
B192_EXP        .WORD -1
B192_HI         .WORD 4970H
B192_LO         .WORD 0C963H      ; 248 b192 = 0,28687723790925

B193_SIGN       .WORD 0H
B193_EXP        .WORD -9
B193_HI         .WORD 410BH
B193_LO         .WORD 41C2H       ; 249 b193 = 0,00099249225053

B194_SIGN       .WORD 0H
B194_EXP        .WORD -7
B194_HI         .WORD 5B20H
B194_LO         .WORD 0C3AH       ; 250 b194 = 0,00556183999848

B195_SIGN       .WORD 0H
B195_EXP        .WORD -1
B195_HI         .WORD 7013H
B195_LO         .WORD 0B6ABH      ; 251 b195 = 0,43780080495019

B196_SIGN       .WORD 0H
B196_EXP        .WORD -1
B196_HI         .WORD 79F0H
B196_LO         .WORD 0CD51H      ; 252 b196 = 0,47633059708347

; SUBMODELO 20:
B200_SIGN       .WORD 0H
B200_EXP        .WORD 0H
B200_HI         .WORD 53A4H
B200_LO         .WORD 0E00FH      ; 253 b200 = 0,65346909262616

B201_SIGN       .WORD 0H
B201_EXP        .WORD -3
B201_HI         .WORD 5AB2H
B201_LO         .WORD 0D16CH      ; 254 b201 = 0,08857276172970

B202_SIGN       .WORD 0H
B202_EXP        .WORD -4
B202_HI         .WORD 750FH
B202_LO         .WORD 0E64CH      ; 255 b202 = 0,05715923232111

B203_SIGN       .WORD 0H
B203_EXP        .WORD -8
B203_HI         .WORD 5C51H
B203_LO         .WORD 6E14H       ; 256 b203 = 0,00281732439972

B204_SIGN       .WORD 0H
B204_EXP        .WORD -8
B204_HI         .WORD 6BFCH
B204_LO         .WORD 0E9BH       ; 257 b204 = 0,00329542840190

B205_SIGN       .WORD 0H
B205_EXP        .WORD -1
B205_HI         .WORD 627DH
B205_LO         .WORD 3B9FH       ; 258 b205 = 0,38472340236550

B206_SIGN       .WORD 0H
B206_EXP        .WORD 0H
B206_HI         .WORD 4549H
B206_LO         .WORD 0D17AH      ; 259 b206 = 0,54131525457211

.END                              ; FIM DO PROGRAMA
```

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * FINAL DO PROGRAMA *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

ANEXO A - "Benchmark" em Matlab / Simulink do processo de neutralização de pH

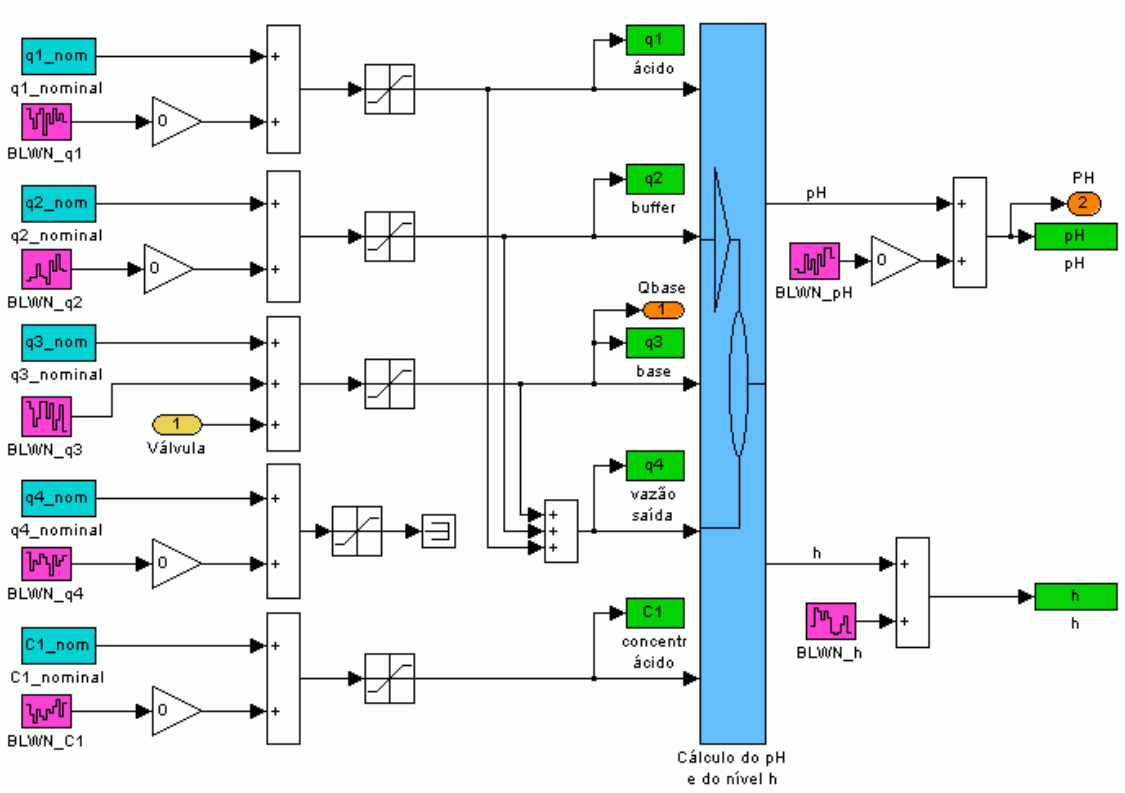


Figura A-1 - Diagrama geral em Simulink - processo de neutralização de pH

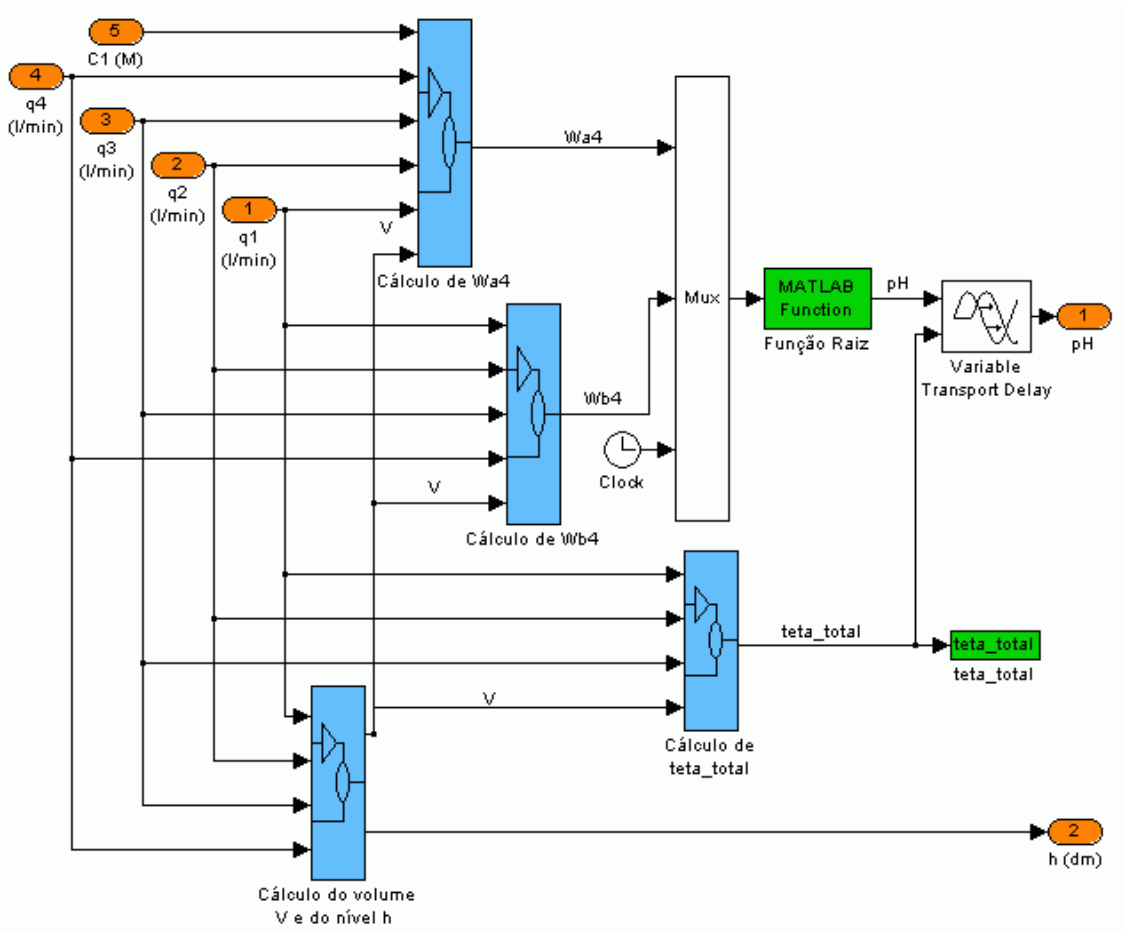


Figura A-2 - Subsistema em Simulink para o cálculo do pH e do nível