UNIVERSIDADE SANTA CECÍLIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA
MESTRADO EM ENGENHARIA MECÂNICA

CLÁUDIO LUÍS MAGALHÃES FERNANDES

LÓGICA PARACONSISTENTE APLICADA EM SISTEMAS DE AUTOMAÇÃO E CONTROLE

SANTOS/SP
2012

CLÁUDIO LUÍS MAGALHÃES FERNANDES

LÓGICA PARACONSISTENTE APLICADA EM SISTEMAS DE AUTOMAÇÃO E CONTROLE

Dissertação apresentada à Universidade Santa Cecília como parte dos requisitos para obtenção de titulo de mestre na área de concentração Automação e Controle de Processos, do curso de Mestrado Profissionalizante em Engenharia Mecânica, sob orientação de:

Orientador: Prof. Dr. João Inácio da Silva Filho

Co-orientador: Prof. Dr. Maurício Conceição Mario.

SANTOS/SP

2012

Ficha catalográfica elaborada pela Biblioteca da
Universidade Santa Cecilia

FERNANDES, Cláudio Luís Magalhães
Lógica Paraconsistente aplicada em Sistemas de Automação e Controle / Cláudio Luís Magalhães Fernandes. Santos/SP, 2012.
109 p. il.

Inclui bibliografia.

Orientador: João Inácio da Silva Filho. Dissertação de Mestrado (Pós-Graduação). Universidade Santa Cecília.

1. Automação industrial 2. Controlador programável 3. IEC 61131-3 4. Lógica Paraconsistente. 5. Lógica Paraconsistente Anotada. I. Título. II. Universidade Santa Cecília.

CDD 006.3

Dedico este trabalho ao meu filho que teve a paciência para suportar minha ausência nas brincadeiras do dia a dia.

AGRADECIMENTOS

Um trabalho científico não é desenvolvido somente pelo seu autor. Nesta dissertação não poderia ser diferente. Dentro deste conceito, diversas pessoas tiveram participação importante na concepção desta dissertação, seja através de contribuições técnicas para a confecção dos experimentos, seja através da palavra de conforto e amiga. Portanto, neste momento, não poderia deixar de agradecer a algumas destas pessoas.

Aos meus orientadores, professores João Inácio da Silva Filho e Maurício Conceição Mario e ao professor Heraldo Silveira Barbuy, pela dedicação e orientações relevantes para o desenvolvimento e conclusão desta dissertação.

Aos colegas da turma de mestrandos da UNISANTA, que, sempre através de um convívio baseado na amizade e cooperação mutua de uma forma ou de outra, cooperaram para o desenvolvimento deste trabalho.

A todo corpo docente do programa de Pós-Graduação em Engenharia Mecânica, com citação especial aos professores Aldo Ramos Santos, Deovaldo de Moraes Junior, José Carlos Morilla e Maria Cristina Pereira Matos pelo carinho dispensado a todos os alunos ao longo destes dois anos.

Aos colegas professores do SENAI-SP, e em especial ao amigo Sérgio Luiz da Conceição Matos, que foi de fundamental importância na conclusão dos experimentos que deram origem a esta dissertação.

A todos os meus familiares que me apoiaram e incentivaram no transcorrer destes dois anos, com uma menção especial ao meu querido pai, que mesmo não estando mais em nosso convívio foi e continuará sendo o meu melhor amigo.

E finalmente agradeço a DEUS, pois sem ele nada é possível.

RESUMO

Nesta dissertação são apresentados os resultados de uma aplicação da Lógica Paraconsistente Anotada com anotação de dois valores (LPA2v) em Sistemas de automação e controle através de CP - Controlador Programável seguindo a Norma IEC 61131 em seu item 7 que preconiza o estudo de lógicas não-clássicas. A LPA2v é uma lógica não clássica que apresenta como principal propriedade a capacidade de tratar contradição. Quando interpretada em seu Reticulado Associado pode-se deduzir equações dos fundamentos da LPA2v, das quais são originados algoritmos para tratamento de sinais contraditórios. Utilizando uma das linguagens da Norma em seu item 3, denominada linguagem de Texto Estruturado, foi criado um bloco funcional baseado em um Algoritmo da LPA2v denominado de Para-Analisador. O Para-Analisador tem sido aplicado com êxito em *hardware* de sistemas de controle de robôs móveis autônomos e em controle de processos contínuos de temperatura, no entanto todos os desenvolvimentos foram dedicados, não possuindo modularidade e nem reutilização de código. A implantação do BF_Paracon permite que todas as propriedades da Lógica Paraconsistente Anotada que estavam presentes no Para-Analisador sejam utilizadas por um Controlador Programável em controle de processos contínuos ou de manufatura dentro da Norma IEC 61131. Para demonstrar a aplicação da LPA2v dentro destas condições foi implementado um controle de temperatura em uma malha fechada onde o Para-Analisador funciona como o bloco principal de controle da malha. Foi feito um estudo comportamental da malha controlada pelo algoritmo da LPA2v com um controle convencional PID mostrando afinidades entre os dois tipos de controle. A relevância nos resultados desta pesquisa se destaca na implementação inédita do Algoritmo *Para-Analisador* funcionando através de um Controlador programável. A contribuição nas pesquisas de automação e controle fica destacada pela criação deste bloco funcional (BF_Paracon) que engloba características de uma lógica não-clássica, onde é demonstrado que a sua modularidade facilita a implementação de sistemas de controle de automação baseados em fundamentos da Lógica Paraconsistente, bem como permite a união de blocos que atuam em conjunto com lógicas clássicas e sistemas baseados em outras linguagens pertencentes a Norma (*Ladder*). Os resultados obtidos nesta pesquisa abrem um amplo campo de desenvolvimento para utilização de lógicas não-clássicas inseridas na Norma IEC 61131 -7 aplicadas à Sistemas de Controle e Automação com a utilização de Controladores Programáveis.

Palavra-Chave*:* 1. Automação industrial 2. Controlador programável 3. IEC 61131-3 4. Lógica Paraconsistente. 5. Lógica Paraconsistente Anotada

ABSTRACT

This dissertation presents the results of an application of Paraconsistent Annotated Logic with annotation of two values (PAL2v) in Systems of automation and control through PC - Programmable Controller according to IEC 61131 in item 7 which calls for the study of non-classical logics. The PAL2v is a non-classical logic which shows as main property the ability to handle conflict. When interpreted in its associated Lattice can be deduced equations from PAL2v concepts, which are derived algorithms for processing of contradictory signals. Using one of the languages of IEC 61131 in the item 3, called Structured Text language, was created a functional block (FB_Paracon) based on an algorithm of PAL2v called Para-Analyzer. The Para-Analyzer has been successfully applied in hardware of control systems for autonomous mobile robots and in continuous process control of temperature, however all developments were dedicated not having modularity nor code reusing. The implementation of FB_Paracon allows all properties of Paraconsistent Annotated Logic that were present in Para-Analyzer are used by a programmable controller in continuous process control or manufacturing in the standard IEC 61131. To demonstrate the application of PAL2v under these conditions was implemented in a temperature control in a closed loop, where the analyzer acts as the main block in the control loop. It was done a study of the controlled loop by the algorithm from LPA2V with a normal PID showing the affinities between both kinds of control. The importance of the results of this research wins the unprecedent implementation of the Para-Analyzer Algorithm running through a Programmable Controller. As contribution to research in automation and control highlights the creation of this functional block (FB_Paracon) that includes features of a non-classical logic, where it is shown that modularity facilitates the implementation of automation control systems based on the foundations of Paraconsistent logic as well it allows the union of blocks that work together with classical logics and systems based in other languages belonging to the Standard (Ladder). The results obtained in this study open a wide field of development for use of non-classical logics included in IEC 61131 -7 applied to the Control System and automation with the use of Programmable Controllers .

Keywords: 1. Industrial Automation 2. Programmable controller. 3. IEC 61131-3 4. Paraconsistent Logic. 5. Paraconsistent Annotated Logic.

LISTA DE ILUSTRAÇÔES

LISTA DE TABELAS

LISTA DE SIGLAS

CP	Controlador Programável
IEC	*International Electrotechnical Commission*
LP	Lógica Paraconsistente
LPA	Lógica Paraconsistente Anotada
LPA2v	Lógica Paraconsistente Anotada com anotação de dois valores
QUPC	Quadrado Unitário do Plano Cartesiano
BF_Paracon	Bloco Funcional Paraconsistente
POU	Unidade de Organização de Programa
LI	Lista de Instruções
LD	Ladder
ST	Texto estruturado
BF	Bloco Funcional
FBD	Diagrama de blocos Funcionais
SFC	Sequenciamento Gráfico de Funções
CFC	Gráfico Contínuo de Funções
CLP	Controlador Lógico programável
SDCD	Sistema Digital de Controle Distribuído
IHM	Interface Homem Máquina
SCR	*Silicon-Controlled Rectifier*

LISTA DE SÍMBOLOS

$\mu 1$ Grau de crença ou grau de evidência favorável.

$\mu 2$ Grau de descrença ou grau de evidência desfavorável.

T Inconsistente

$\perp$ Paracompleto ou indeterminado

$\perp \rightarrow f$ Indeterminado, tendendo ao Falso;

$\perp \rightarrow v$ Indeterminado, tendendo ao Verdadeiro;

$T \rightarrow f$ Inconsistente, tendendo ao Falso;

$Qv \rightarrow T$ Quase Verdadeiro, tendendo ao Inconsistente;

$Qf \rightarrow T$ Quase Falso, tendendo ao Inconsistente;

$Qf \rightarrow \perp$ Quase Falso, tendendo ao Indeterminado;

$Qv \rightarrow \perp$ Quase Verdadeiro, tendendo ao Indeterminado.

G_{it} Grau de Inconsistência;

G_{id} Grau de Indeterminação;

G_{v} Grau de Verdade;

G_{f} Grau de Falsidade;

G_{ct} Grau de Contradição;

G_{c} Grau de Certeza;

V_{scc} Variável superior de controle de certeza (C_1);

V_{icc} Variável inferior de controle de certeza (C_2);

V_{scct} Variável superior de controle de contradição (C_3);

V_{icct} Variável inferior de controle de contradição (C_4).

SUMÁRIO

1 INTRODUÇÃO

Atualmente a economia no mundo moderno está sustentada pela produtividade das indústrias que dependem da eficiência apresentada pelos seus equipamentos e máquinas instaladas. A eficiência, por sua vez é obtida através de planos de implantação de sistemas de automação e controle que tragam garantias de elevação no índice de qualidade do produto manufaturado final.

Segundo Duarte (1999), a evolução nos meios produtivos está atrelada a inovações tecnológicas, sejam elas a simples utilização de equipamentos mais modernos, ou até mesmo o emprego de novas técnicas de produção.

Dentro desta ação de inovação tecnológica, iniciou-se em meados do século passado uma crescente utilização de computadores para atuar nos processos de automação e controle das máquinas de produção com o objetivo de otimizar os processos e dar maior flexibilidade à produção industrial.

Os computadores dedicados aos processos de automação são denominados de controladores industriais e no inicio surgiram dois tipos de controladores, o Controlador Lógico Programável (CLP), que é mais dedicado ao sistema de manufatura, e o Sistema Digital de Controle Distribuído (SDCD), sendo este último mais utilizado na indústria de processos contínuos. A rápida evolução da informática permitiu que logo houvesse uma convergência das duas tecnologias onde um mesmo computador pode ser configurado para reunir as mesmas funcionalidades, e este passou a ser denominado de CP – Controlador Programável.

No intuito de garantir uma operação segura e economicamente viável aos sistemas de automação e controle em Processos Contínuos e da Manufatura foram instalados Controladores Programáveis (CPs) em, praticamente, toda planta industrial.

O surgimento destes CPs originou para a engenharia industrial a necessidade de gerar conhecimento aos trabalhadores envolvidos nos processos de produção, capacitando-os para utilizar as novas técnicas. Com essa nova formalização a implantação de processos de automação e controle nas máquinas de produção ficou intimamente ligada à área da informática e ao desenvolvimento de projetos nos quais os Controladores Programáveis estariam inseridos.

Na implantação destas novas tecnologias verificou-se que havia a necessidade de encontrar respostas eficientes a um controle de processos, onde os

intervalos entre os limites que expressam a qualidade do produto ficavam, a cada dia, menores. Dessa forma, considerou-se a implementação nos sistemas operacionais dos Controladores Programáveis de algoritmos que pudessem apresentar resultados mais eficientes, e com baixo custo computacional. Foi na busca da resolução deste problema em particular, que alguns CPs foram oferecidos com condições de serem configurados com algoritmos fundamentados em lógicas diferentes da lógica binária, também conhecida como lógica clássica.

Segundo Abe (1992), as lógicas denominadas de não-clássicas são aquelas que divergem em seus fundamentos de alguma das leis binárias que sustentam a lógica clássica. Atualmente os CPs modernos, trazem condições nas quais os algoritmos podem ser configurados para tratar sinais com base em Lógica Nebulosa, que é uma lógica não-clássica também conhecida como Lógica *Fuzzy* (Kaufmann & Gupta , 1985).

Vindo ao encontro desta evolução, liderada pela instalação de controladores industriais, onde a informática, os modos de configuração e as técnicas lógicas de desenvolvimento na implantação tornaram-se de suma importância para elevar o índice de eficiência e qualidade dos processos, houve a necessidade de se buscar uma padronização entre os vários fabricantes de CPs. Este processo evolutivo originou a criação da norma para as configurações de CPs conhecida como IEC 61131, cujos fundamentos principais serão expostos a seguir.

1.1 Noções Básicas da Norma IEC 61131

A norma IEC 61131 foi elaborada pela *International Electrotechnical Commission* com o objetivo de padronizar diversos aspectos relacionados aos Controladores Programáveis (CP), assim como aplicar modernas técnicas e linguagens de programação para o desenvolvimento de software para estes sistemas (Torres et al, 2001).

O item três (3) da norma trata da estrutura do software, execução do programa e principalmente da padronização de utilização dos cinco (5) tipos de linguagens de programação. Estes tipos que serão detalhados posteriormente são: Sequenciamento Gráfico de Funções (SFC), Lista de Instruções (IL), Diagrama *Ladder* (LD), Diagrama de Blocos (FBD) e Texto Estruturado (ST) (Casado Lima, 2003). Como é visto em Guimarães (2005), a Norma preconiza a estruturação

destas formas de programação, facilitando a modularização e a reutilização de código

Este conceito implica na criação de blocos funcionais e funções padrões as quais possibilitam a criação de novas estruturas baseadas nas existentes, que por sua vez acrescentam a característica de recursividade e o desenvolvimento de bibliotecas pelo próprio programador.

Verifica-se que a norma em seu item sete (7) leva também em consideração as imprecisões e incertezas que ocorrem em um processo industrial bem como as suas implicações na teoria de controle moderno. Para essa finalidade é aberto um campo de pesquisa voltada às lógicas não-clássicas, em especial a Programação utilizando Lógica Nebulosa (*Fuzzy*).

Dessa forma abre-se uma possibilidade de considerar as condições implícitas no referido item sete (7) para outros tipos de lógicas não-clássicas diferentes da Fuzzy, tais como, a Lógica Paraconsistente (PL) conforme será visto a seguir (Bottura Filho, 2007).

1.2 Objetivo do Trabalho

O principal objetivo deste trabalho e mostrar uma forma de aplicação da Lógica Paraconsistente Anotada com anotação de dois valores (LPA2v) em Sistemas de Automação e Controle através de CP - Controlador Programável seguindo a norma IEC 61131 que, em seu item 7, preconiza o estudo de lógicas não-clássicas.

Os objetivos secundários são:

a) Mostrar uma exploração conjunta dos itens 3 e 7 da Norma IEC 61131 considerando as técnicas envolvidas na utilização da lógica não-clássica denominada de Lógica Paraconsistente (LP) ao lado da Lógica *Fuzzy* que já está consolidada na norma.

b) Através das considerações que envolvem a Lógica Paraconsistente, utilizar um CP em atendimento a Norma com aplicação de um Bloco Funcional do Algoritmo extraído da *LP* em um processo de automação para controle de temperatura.

1.3 Organização da Dissertação

Esta apresentação da Dissertação está organizada da seguinte forma:

Neste capítulo 1, é apresentada uma visão geral do projeto. O capítulo 2 apresenta uma introdução da Lógica Paraconsistente (*LP*) e seus principais fundamentos necessários à compreensão da pesquisa. Também neste capítulo são apresentadas as extensões da Lógica Paraconsistente, a Lógica Paraconsistente Anotada (LPA) e Lógica Paraconsistente Anotada com anotação de dois valores (LPA2v) finalizando com o algoritmo Para-Analisador, citando algumas aplicações já desenvolvidas. O capítulo 3 mostra características dos Controladores Programáveis e alguns itens importantes referentes à norma IEC 61131, mais especificamente o item 3 que trata das linguagens de programação de CPs e o item 7 que trata das lógicas não-clássicas. O capítulo 4 apresenta os métodos utilizados nas configurações e construção dos blocos lógicos utilizados no tratamento de sinais. Ainda neste capítulo 4 é mostrada a metodologia aplicada para a criação do bloco funcional (BF_Paracon) que teve sua construção baseada no Algoritmo Para-Analisador da LPA2v. No capítulo 5 é apresentado o desenvolvimento do projeto de processo de automação para controle de temperatura aplicando um CP com o bloco funcional (BF_Paracon). Ainda neste capítulo 5 são mostrados os resultados obtidos no desenvolvimento e discussões sobre os valores resultantes da implantação. É feito um estudo comparativo do comportamento de uma malha fechada de controle de temperatura utilizando o bloco funcional (BF_Paracon) e uma ação PID. No final deste capítulo são apresentadas sugestões e os procedimentos necessários para projetos futuros utilizando o bloco funcional (BF_Paracon). No capítulo 6 são elaboradas as conclusões e destacadas as principais contribuições desta pesquisa envolvendo CP e Lógica Paraconsistente (*LP*) para a área de automação e controle.

2 LÓGICA PARACONSISTENTE

Os processos tecnológicos de automação e controle utilizam computadores para tratamento de dados e interpretação de resultados oferecendo altos níveis de eficiência às máquinas. Basicamente os algoritmos que sustentam a programação computacional são construídos baseados em fundamentos da lógica clássica que possui características binárias. A alta produção de produtos manufaturados faz com que o controle sobre a qualidade, dimensões e desperdício de material seja monitorado com maior rigor. Isto significa que os limites que definem a faixa para a produção fiquem mais próximos, impondo assim maiores responsabilidades aos sistemas de aquisição e interpretação de dados provenientes de informações reais sobre as condições do processo. Esse quadro que exige modelos mais próximos da realidade gera, em muitas situações, informações incompletas, difusas e contraditórias, o que torna a lógica clássica incapaz de oferecer algoritmos com respostas eficientes e em um tempo ideal para tomadas de decisão. Para atenuar esses problemas os grandes centros de pesquisa partiram para um relevante esforço em desenvolver técnicas de utilização de lógicas que pudessem funcionar diferentemente da condição binária da lógica clássica.

De maneira geral pode-se considerar uma lógica não-clássica como sendo toda aquela que, de alguma forma, apresenta fundamentos que possam contrariar ou ignorar algum princípio da Lógica clássica, incluindo-se àquelas que a complementam. Recentemente, com a ocorrência de novas necessidades para a automação de processos os sistemas computacionais trazem condições de serem configurados com algoritmos baseados em lógicas não-clássicas onde, a que obteve maior sucesso, tem sido a lógica nebulosa ou *Fuzzy*.

Entre as lógicas denominadas não-clássicas a Lógica Paraconsistente (*LP*) é caracterizada pela sua mais importante propriedade, que é a de considerar contradição sem o perigo de trivialização em seus fundamentos.

Segundo Abe (1992) a Lógica Paraconsistente (*LP*) pode ser entendida conforme a seguir: - É dito que uma teoria é consistente quando entre seus teoremas não houver contradição, em caso contrário, ela é denominada inconsistente. Uma teoria é denominada como trivial se todas as sentenças (ou fórmulas) de sua linguagem forem teoremas; se o contrário ocorrer, ela é

classificada como sendo não-trivial. Uma lógica é denominada Paraconsistente se puder tratar de inconsistências e de forma não trivial.

Por esse mesmo raciocínio, uma lógica é chamada paracompleta se pode funcionar como a lógica subjacente de teorias na qual há fórmulas tais que estas fórmulas e suas negações são ambas falsas. Como exemplo, suponha que temos duas proposições sendo, uma de crença, indicando em relação à presença de um objeto, que este não existe, portanto, apresentando uma falsidade e, outra proposição de descrença, onde esta também indica falsidade que, neste caso, indicaria a presença de um objeto. Essa lógica seria denominada paracompleta pelas duas falsidades apresentadas. Uma teoria é chamada paracompleta se sua lógica subjacente é uma lógica paracompleta.

Apresentam-se a seguir um resumo da Lógica Paraconsistente e a sua extensão denominada de Lógica Paraconsistente Anotada com anotação de dois valores LPA2v.

2.1 Lógica Paraconsistente Anotada

As Lógicas Paraconsistentes nasceram da necessidade de se encontrar meios de dar tratamento adequado às situações contraditórias. De forma simplificada pode-se considerar que a Lógica Paraconsistente tem a finalidade de ser aplicada em casos onde a Lógica Clássica, que só trabalha com condições verdadeiras ou falsas, não pode ser empregada. Em situações como, por exemplo, em sistemas de controle aplicados à automação, aparece a necessidade do emprego de uma lógica não-clássica que por apresentar fundamentações não ligadas as rígidas leis binárias da Lógica Clássica poderá dar origem a análises mais eficientes com aproximações de sinais contraditórios que retratam a realidade.

Em muitos estudos, entre eles os encontrados em (Abe 1997), (Da Costa 1996) e (Da Costa & Subrahmanian 1989), apresentaram resultados que possibilitam considerar as inconsistências em sua estrutura de um modo não trivial e por isso, se mostram mais propícias no enquadramento de problemas ocasionados por situações de contradições que aparecem quando lidamos com o mundo real.

Uma Lógica Paraconsistente Anotada (LPA) é uma classe de Lógica Paraconsistente que possui um reticulado associado que desempenha um papel sumamente importante na sua representação.

Na LPA as fórmulas proposicionais vêm acompanhadas de anotações. Cada anotação (μ) pertencente a um reticulado finito τ atribui valores à sua correspondente fórmula proposicional ρ. Considerando uma lógica evidencial as anotações vêm representadas por Graus de evidência ou crença.

A LPA pode ser estudada através de um Quadrado unitário no Plano Cartesiano - QUPC (figura 1) onde são inseridos os graus de evidência favorável (crença) μ no eixo *x* e graus de evidência desfavorável (descrença) λ no eixo *y*. Verifica-se que a interpolação de valores máximos permite que se possam obter representações de quatro situações possíveis:

1-Inconsistente; no ponto (1, 1) = T

2-Verdadeira; no ponto (1, 0) =V

3- Falsa; no ponto (0, 1) = F

4- Indeterminada ou paracompleta; no ponto (0, 0) = $\perp$

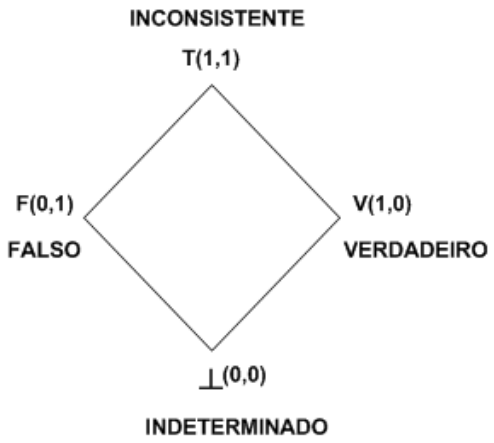


Figura 1. Reticulado Representativo De Hasse
Fonte: Da Silva Filho & Abe, 2001.

Quando as proposições vêm acompanhadas de anotações, ou grau de evidência é possível a aplicação real da Lógica Paraconsistente em Sistemas de Análises e tomadas de decisão.

2.2 Lógica Paraconsistente Anotada com Anotação de 2 Valores LPA2V

A Lógica Paraconsistente Anotada de Anotação com dois valores (LPA2v) utiliza uma anotação composta por dois sinais de informação, ou seja, cada

proposição utiliza dois valores de graus de evidência denominados: grau de crença (evidência favorável) e grau de descrença (evidência desfavorável).

O primeiro valor da anotação representa a evidência favorável à proposição *p*, ou seja, representa o grau de crença dessa proposição (denominado μ_1). O segundo valor da anotação representa a evidência contrária à proposição *p*, ou seja, representa o grau de descrença dessa proposição (denominado μ_2).

Em várias situações reais onde é utilizada a LPA2v, os graus de evidência favorável (crença) e evidência desfavorável (descrença) são considerados como informações de entrada do sistema (exemplo: sinais de entrada de corrente, tensão, etc.) e os estados lógicos representados nos vértices e nas regiões internas do reticulado são as saídas resultantes correspondente da análise paraconsistente. Na figura 2 é apresentado o diagrama:

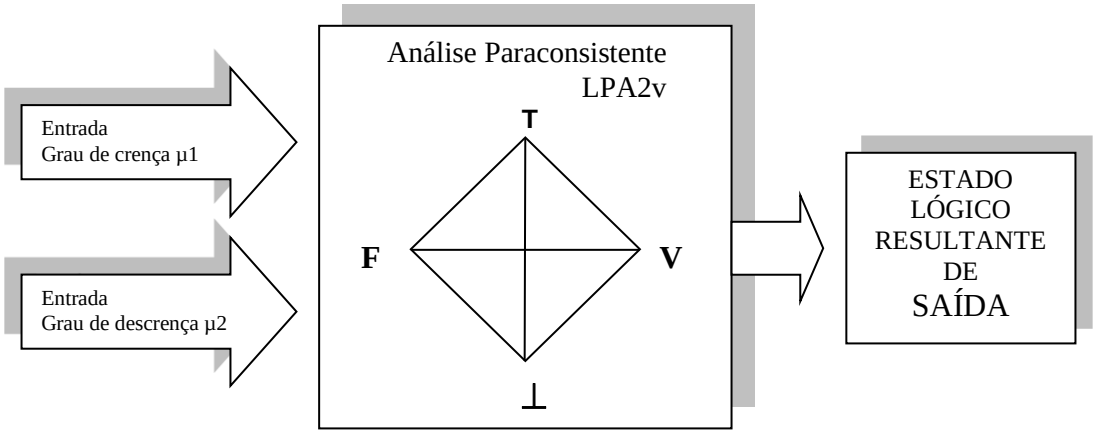


Figura 2. Representação da análise Paraconsistente LPA2v
Fonte: Da Silva Filho & Abe, 2001, p 35

2.3 Representação no Quadrado Unitário do Plano Cartesiano – QUPC

A LPA pode ser estudada em um Quadrado unitário no Plano Cartesiano - QUPC (figura 3) onde são inseridos os graus de evidência favorável (crença) μ e graus de evidência desfavorável (descrença) λ.

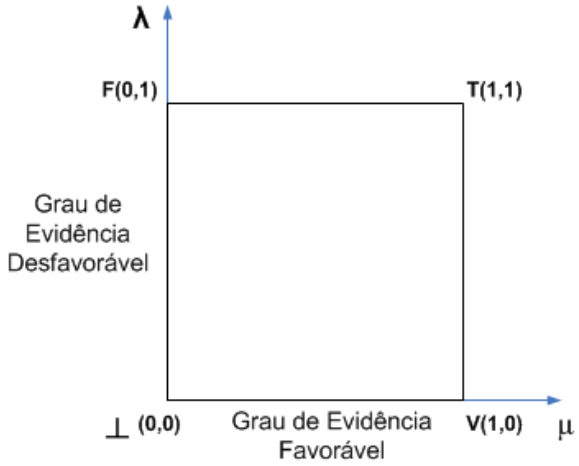


Figura 3. Graus de evidência no Quadrado Unitário do Plano Cartesiano.
Fonte: Da Silva Filho & Abe, 2001.

.

.

2.3.1 O Grau de Contradição

Conforme visto em (Da Silva Filho 2008) a partir do Quadrado unitário são elaboradas transformações nas quais se podem obter os valores dos graus de contradição (Gct) e dos graus de certeza (Gc) referentes aos cálculos que envolvem os dois graus de evidência.

Inicialmente considera-se a análise no Quadrado Unitário no Plano Cartesiano (QUPC) onde os graus de crença estão dispostos no eixo *x* e os graus de descrença estão dispostos no eixo *y*, como mostrado na figura 4

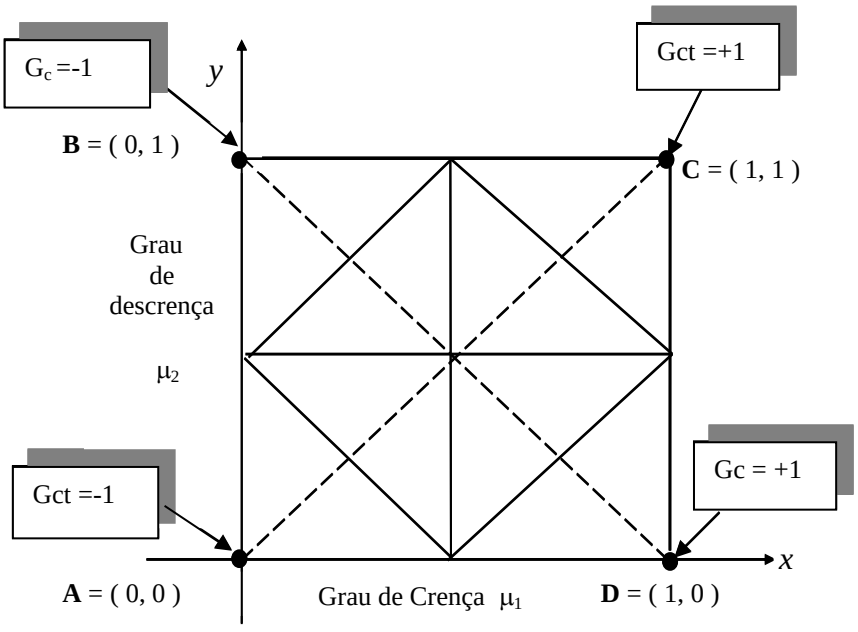


Figura 4. Quadrado unitário no Plano Cartesiano
Fonte: Da Silva Filho & Abe, 2001. p. 3

O grau de contradição (Gct) é definido como sendo o valor que representa a interpolação entre os graus de evidência favorável e de evidência desfavorável no quadrado unitário no plano cartesiano, representados pelos pontos B e D, onde B = (0, 1) – Falso e D = (1,0) - Verdadeiro. Encontra-se o valor de Gct no intervalo real fechado [-1,1]. Portanto, o grau de contradição (Gct) é calculado através da equação:

$$Gct = \mu_1 + \mu_2 - 1 \quad (1)$$

Sendo:

$0 \leq \mu_1 \leq 1$

$0 \leq \mu_2 \leq 1$

Neste contexto Da Silva Filho & ABE, 2001 no livro, “Fundamentos das redes neurais artificiais”, afirma que:

> “O valor $Gct_{=}$-1 é estabelecido no ponto A = (0,0) representando contradição máxima negativa e o valor $Gct_{=}$+1 é estabelecido no ponto C = (1,1) significando que temos uma contradição máxima positiva. De acordo com os sensores utilizados em situações reais de uso, quando chegam valores que resultam nesses graus de contradição, podemos afirmar que essas informações são completamente contraditórias. Em um Sistema de Análise Paraconsistente, quanto mais a interpolação entre os graus de crença e de descrença se aproximarem do segmento de reta BD, representado no QUPC, mais o resultado da soma dos graus de crença e de descrença ($\mu_1 + \mu_2$) se aproxima de 1, diminuindo, assim, o valor de Gct. Esta diminuição de Gct representa uma menor contradição entre as informações na entrada. Quando a soma dos graus de crença e de descrença ($\mu_1 + \mu_2$) for igual a 1, o grau de Contradição é zero e o ponto de interpolação estará sobre a reta BD. Neste caso, Gct = 0 e não há contradição entre os sinais. (2001, p. 36)”.

2.3.2 O Grau de Certeza

O grau de certeza (Gc) é definido como sendo o valor que representa a intersecção entre os graus de evidência favorável (crença) e de evidência desfavorável (descrença) no quadrado unitário no plano cartesiano, representados pelos pontos A e C da figura 4, onde, A = (0, 0) - Indeterminado ao ponto C = (1, 1) Inconsistente. Encontra-se no intervalo real fechado [-1, 1].

Portanto, o grau de certeza (Gc) é calculado através da equação:

$$Gc = \mu_1 - \mu_2 \quad (2)$$

Sendo:

$0 \leq \mu_1 \leq 1$

$0 \leq \mu_2 \leq 1$

Segundo (Da Silva Filho 1999) no livro, “Métodos de Aplicações da Lógica Paraconsistente Anotada com anotação de dois valores- LPA2v com a construção de Algoritmo e Implementação de Circuitos Eletrônicos”, diz que:

> “O valor $Gc_{=}$-1 é estabelecido no ponto B = (0,1) representando certeza máxima na negação da proposição e o valor $Gc_{=}$+1 é estabelecido no ponto D = (1,0) significando que temos uma certeza máxima na afirmação da proposição. De acordo com os sensores utilizados em situações reais de uso, quando chegam valores que resultam nesses graus de contradição, podemos afirmar que essas informações são consistentes. Quanto mais a interpolação entre os graus de crença e de descrença se aproximarem do segmento de reta AC, representado no QUPC, mais o resultado da subtração dos graus de crença pelo grau de descrença ($\mu_1 - \mu_2$) se aproxima de 0, diminuindo, assim, o valor de Gc. Esta diminuição de Gc representa uma menor certeza entre as informações na entrada porque significa uma maior coincidência entre os Graus de crença e de descrença. Quando os graus de crença e de descrença forem de valores iguais ($\mu_1 = \mu_2$), o grau de certeza é zero e o ponto de interpolação estará sobre a reta AC. (2001, p.37)”.

Com o procedimento básico de interpretar os valores através de um Quadrado Unitário descrito no plano cartesiano podem-se obter valores diversos de Gct e Gc a partir dos graus evidência recebidos normalizados e, portanto valorados na faixa entre 0 e 1 inclusive. Os valores dos graus de Certeza e de contradição podem ser dispostos em dois eixos: “Eixo dos Graus de Contradição (Gct)” e “Eixo dos Graus de certeza (Gc)” que, sobrepostos e cruzando-os no ponto de Indefinição, onde os graus de evidências valem 0,5 representam o reticulado da LPA com valores. Para qualquer anotação, composta por valores de Graus de evidência favorável (crença) e Graus de evidência desfavorável (descrença), pode-se obter um ponto de Interpolação entre valores de Graus de Certeza e Graus de Contradição, que é pertencente ao reticulado da LPA2v. Dessa forma, para qualquer anotação, com valores de graus de evidência favorável (crença) e de evidência desfavorável (descrença), são obtidos os valores do Grau de Certeza (Gc) e de Contradição (Gct),

relacionado a determinada proposição *P*. Fazendo a interpolação dos dois valores pode-se visualizar no Reticulado (figura 5) onde está localizado o ponto resultante considerado como o estado lógico Paraconsistente (ε_T).

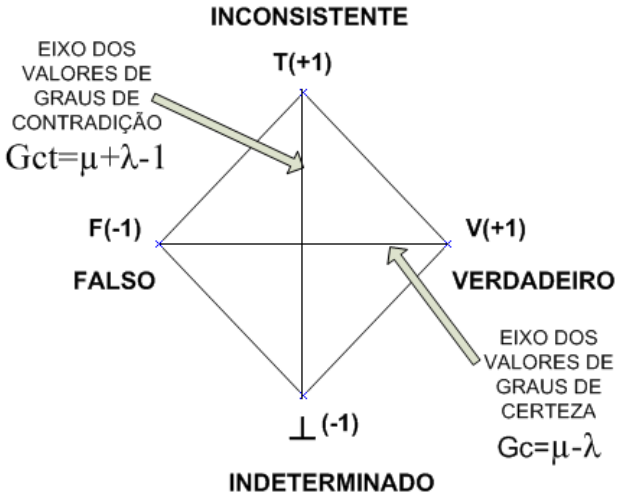


Figura 5. Eixos dos Graus de Contradição e de Certeza.
Fonte: DA SILVA FILHO, 1999.

Na prática a tomada de decisão pode ser feita relacionando o resultado com proximidade, ou não, de algum dos estados lógicos extremos localizados nos vértices. Por exemplo, intuitivamente, em tal reticulado, (1.0, 0.0) indica 'evidência total', (0.0, 1.0) indica 'não evidência total', (1.0, 1.0) indica 'evidência totalmente inconsistente' e (0.0, 0.0) indica 'evidência totalmente indeterminada'.

As representações da proposição com a anotação composta por dois Graus de Evidência são descritas como se segue:

$p_T = p_{(1,\ 1)} \Rightarrow$ A anotação composta pelos graus de evidência favorável (crença) = 1 e de evidência desfavorável (descrença)= 1 atribui à proposição *p* uma leitura intuitiva de que p é inconsistente.

$p_1 = p_{(1,\ 0)} \Rightarrow$ A anotação composta pelos graus de evidência favorável (crença) = 1 e de evidência desfavorável (descrença)= 0 atribui à proposição *p* uma leitura intuitiva de que p é verdadeira.

$p_0 = p_{(0,\ 1)} \Rightarrow$ A anotação composta pelos graus evidência favorável (crença) = 0 e de evidência desfavorável

(descrença)= 1 atribui à proposição *p* uma leitura intuitiva de que p é falsa.

$p_{\perp} = p_{(0,\ 0)} \Rightarrow$ A anotação composta pelos graus de evidência favorável (crença)= 0 e de evidência desfavorável (descrença) = 0 atribui à proposição *p* uma leitura intuitiva de que *p* é indeterminada. (Da Silva Filho & Abe 2001).

2.4 O Reticulado da LPA com Regiões Delimitadas

Conforme foi visto no item anterior, considerando os valores encontrados no quadrado unitário do plano cartesiano pode-se estender a análise em uma representação de 2 eixos: um com os valores do grau de contradição e outro com os valores do grau de certeza. Estes dois eixos são sobrepostos de tal forma a serem comparados com o reticulado do LPA2v, onde pode-se delimitar regiões as quais serão comparadas a determinados estados lógicos paraconsistentes.

Conforme é visto na figura 6, para as delimitações das regiões os dois limites externos e arbitrários; *Valor superior de controle de Certeza* (Vscc) e *Valor inferior de controle de Certeza* (Vicc) determinam quando o grau de Certeza resultante é alto o suficiente para que a proposição analisada seja considerada totalmente verdadeira ou totalmente falsa. Da mesma forma os dois limites externos e arbitrários *Valor superior de controle de Contradição* (Vscct) e *Valor inferior de controle de Contradição* (Vicct) determinam quando o grau de Contradição resultante é alto o suficiente para que a proposição analisada seja considerada totalmente consistente, ou totalmente inconsistente, como é possível de ser observado na figura 6.

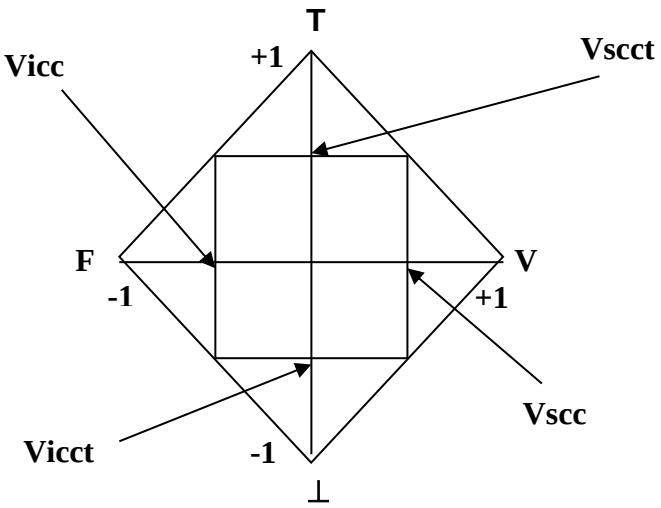


Figura 6. Representação do Reticulado LPA2v com valores extremos
Fonte: Da Silva Filho, 1999.

O reticulado pode ser delimitado conforme a relação abaixo:

$\mathbf{V_{scc}}$: Valor superior de controle de Certeza (variando entre 0 e +1);

$\mathbf{V_{icc}}$: Valor inferior de controle de Certeza (variando entre 0 e –1);

$\mathbf{V_{scct}}$: Valor superior de controle de Contradição (variando entre 0 e +1);

$\mathbf{V_{icct}}$: Valor inferior de controle de Contradição (variando entre 0 e –1).

2.5 O Algoritmo Para-Analisador

Através das limitações superiores e inferiores dos graus de certeza e contradição o reticulado é repartido em 12 regiões, o que significa uma divisão em 12 estados lógicos resultantes. Conforme é visto na figura 7, a partir das regiões delimitadas do reticulado pode-se relacionar estados lógicos resultantes obtidos através das interpolações dos graus de certeza e de contradição, que por sua vez são dependentes dos valores dos Graus de Evidência obtidos de medições no meio físico.

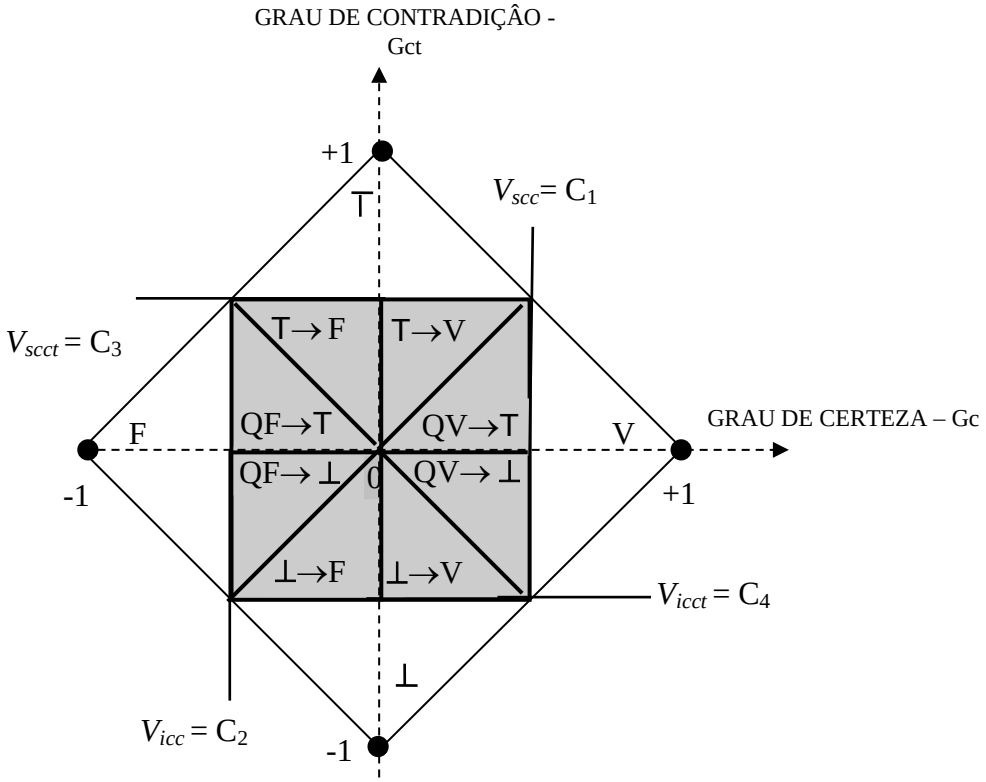


Figura 7. Reticulado da LPA2v repartido em 12 estados lógicos
Fonte: Da Silva Filho, 1999.

Os estados Lógicos Extremos são representados pelas regiões que ocupam os vértices do reticulado: Verdadeiro, Falso, Indeterminado e Inconsistente.

As regiões internas no reticulado são representantes dos estados lógicos de saída, denominados de estados Lógicos não-Extremos, todos nomeados conforme a proximidade com os estados Lógicos Extremos. Os estados lógicos não-extremos são:

⊥ →	**F**	Indeterminado (paracompleto) tendendo a Falso
⊥ →	**V**	Indeterminado (paracompleto) tendendo a Verdadeiro
⊤ →	**F**	Inconsistente tendendo a Falso
⊤ →	**V**	Inconsistente tendendo a Verdadeiro
QV →	⊤	Quase – verdadeiro tendendo a Inconsistente
QF →	⊤	Quase – falso tendendo a Inconsistente
QF →	⊥	Quase – falso tendendo a Indeterminado
QV →	⊥	Quase – verdadeiro tendendo a Indeterminado

As variáveis de controle para recursos de otimização são:

V_{scc} – Valor superior de controle de certeza

V_{scct} – Valor superior de controle de contradição

V_{icc} – Valor inferior de controle de certeza

V_{icct} – Valor inferior de controle de contradição

Com estas considerações pode-se descrever o Algoritmo Para-Analisador conforme se segue:

O algoritmo “Para-Analisador”

/Definições dos valores/

V_{scc} = C_1*/ Definição do valor superior de controle de certeza*/

V_{icc} = C_2 */ Definição do valor inferior de controle de certeza*/

V_{scct} = C_3*/ Definição do valor superior de controle de contradição*/

V_{icct} = C_4 */ Definição do valor inferior de controle de contradição

/Variáveis de entrada/

μ_1

μ_2

/Variáveis de saída

Saída discreta = S_1

Saída analógica = S_{2a}

Saída analógica = S_{2b}

*/Expressões matemáticas */

sendo : $0 \leq \mu_1 \leq 1$ e $0 \leq \mu_2 \leq 1$

$Gct = \mu_1 + \mu_2 - 1$

$Gc = \mu_1 - \mu_2$

*/determinação dos estados lógicos extremos */

Se $Gc \geq C_1$ então $S_1 = V$

Se $Gc \leq C_2$ então $S_1 = F$

Se $Gct \geq C_3$ então $S_1 = T$

Se $Gct \leq C_4$ então $S_1 = \perp$

/determinação dos estados lógicos não-extremos/

Para $0 \leq Gc < C_1$ e $0 \leq Gct < C_3$

se $Gc \geq Gct$ então $S_1 = Qv \rightarrow T$

se $Gc < Gct$ então $S_1 = T \rightarrow v$

Para $0 \leq Gc < C_1$ e $C_4 < Gct \leq 0$

se $Gc \geq |Gct|$ então $S_1 = Qv \rightarrow \perp$

se Gc < | Gct | então $S_1 = \perp \rightarrow v$

Para $C_2 < Gc \leq 0$ e $C_4 < Gct \leq 0$

se |Gc | ≥ | Gct | então $S_1 = Qf \rightarrow \perp$

se |Gc | < | Gct | então $S_1 = \perp \rightarrow F$

Para $C_2 < Gc \leq 0$ e $0 \leq Gct < C_3$

se |Gc | ≥ Gct então $S_1 = Qf \rightarrow T$

se |Gc | < Gct então $S_1 = T \rightarrow F$

$Gct = S_{2a}$

$Gc = S_{2b}$

/ FIM/

Portanto, o algoritmo Para-Analisador traduz a análise paraconsistente, através das informações recebidas na forma de grau de evidência favorável (crença) e grau de evidência desfavorável (descrença), resultando nos valores de grau de certeza e grau de contradição, pelos quais o sistema determinará um estado lógico para tomada de decisão.

Os valores de grau de certeza e de contradição também poderão ser considerados como sinais de saída, os quais podem ser utilizados, por exemplo, em realimentação para um sistema de controle contínuo.

2.6 Implementação do Algoritmo Para-Analisador em Hardware

Diversos trabalhos vêm sendo desenvolvidos com a utilização do algoritmo Para-Analisador onde as principais aplicações foram voltadas à robótica móvel autônoma (Torres C. R. 2010) (Maciel et al, 2011). Nestes trabalhos verificou-se que uma linguagem de programação é escolhida e utilizada sem, no entanto, existir a preocupação com a modularidade, reutilização e replicação de códigos. A primeira aplicação do Algoritmo Para-Analisador foi em (Da Silva Filho, 1999) implementado em um controlador lógico paraconsistente empregado na construção do robô Emmy visto na figura 8:

Figura 8. Robô Emmy
Fonte: http://paralogike.com.br/roboemmy.htm

2.6.1 A Estrutura Básica do Robô EMMY

O robô Emmy foi construído em módulos, onde cada qual tem uma função distinta atuando no sistema de controle. A figura 9 destaca os módulos que compõem todo o sistema de controle do Robô Emmy. Neste projeto, dois sensores de ultrassom captam os sinais identificando a distância entre o robô e os objetos (obstáculos). Utiliza-se para tal processo de captura de sinais um sistema de ultrassom onde os valores de crença e descrença - que podem variar entre 0 e 5V - são proporcionais ao tempo entre a emissão de um trem de pulso gerado por um microprocessador e o recebimento do eco pelos sensores.

O Controlador Lógico Paraconsistente denominado *Para-Control,* faz a análise destes sinais considerando sempre os valores de μ_1 e μ_2 como entradas e calcula as respostas na forma dos Graus de certeza e de Contradição através das equações (1) e (2), reproduzidas abaixo:

$Gct = \mu_1 + \mu_2 - 1$

$Gc = \mu_1 - \mu_2$

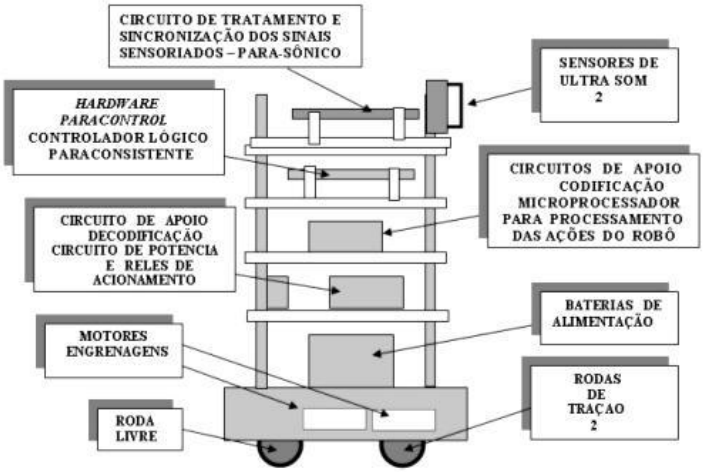


Figura 9. Estrutura básica do robô Emmy
Fonte: http://paralogike.com.br/roboemmy.htm

2.6.2 O Controlador do Robô EMMY

O Controlador do robô Emmy denominado de *Para-Control* utiliza o algoritmo Paranalisador e foi confeccionado em duas etapas sendo uma delas através de um circuito analógico com a utilização de amplificadores operacionais (Da Silva Filho,1999). Esse circuito fornece os quatro estados lógicos extremos; verdadeiro, falso, inconsistente e indeterminado, além dos graus de contradição e de certeza. São também fornecidos quatro sinais denominados A, B, C, D que são utilizados como entradas do circuito gerador dos oito estados lógicos não-extremos. Com os valores dos graus de certeza e de contradição o Controlador seleciona como saída um dos estados lógicos entre os 12 possíveis do reticulado repartido em regiões. A figura 10 mostra a análise paraconsistente efetuada pelo Controlador Lógico Paraconsistente - *Para-Control*.

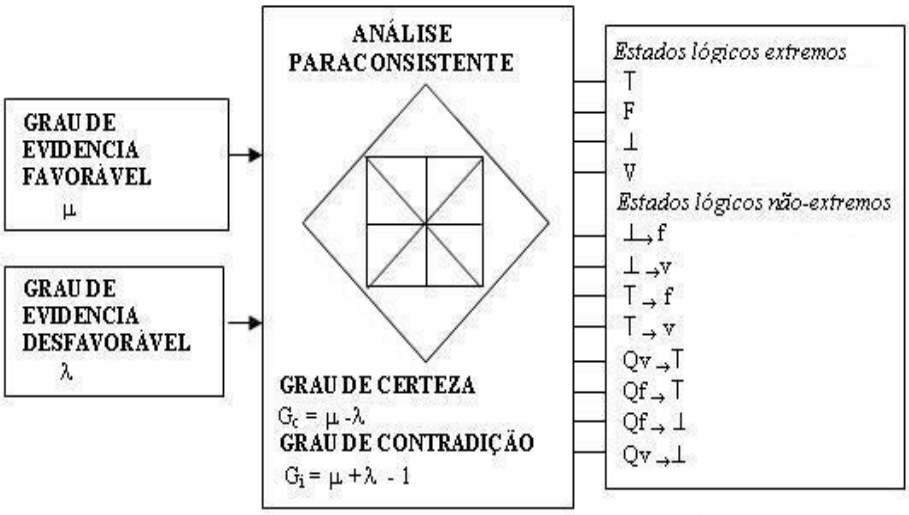


Figura 10. Análise paraconsistente feita pelo controlador
Fonte: http://paralogike.com.br/roboemmy.htm

Em relação a aplicação do Algoritmo Para-Analisador e como exemplo de implementação de *hardware* com a utilização de *software*, é possível citar alguns trabalhos. Entre estes tem-se o apresentado em (Maciel, 2011), onde o algoritmo paraconsistente utilizado no controle foi implementado no robô *Lego* utilizando a programação elaborada na plataforma *LabView*.

Outro exemplo que pode ser citado é a construção do robô Emmy II, trabalho desenvolvido em (Torres, 2010) onde a programação do algoritmo Para-Analisador foi desenvolvida em *Assembly* e implementada no microcontrolador 89C52.

No próximo capítulo será realizada a implementação do algoritmo Paranalisador em um CP conforme a Norma IEC 61131-3.

3 CONTROLADOR PROGRAMÁVEL E A NORMA IEC 61131-3

Segundo OLIVEIRA et al. (2007), o primeiro Controlador Lógico Programável (CLP) nasceu em 1968, e foi concebido diante da dificuldade encontrada pelas indústrias automobilísticas em alterar suas linhas de montagem. Verificava-se nas industrias da época que qualquer ampliação ou modernização em seu pátio fabril revertia em um aumento de custo, devido à necessidade de substituição ou troca da estrutura física interna dos Painéis de Comando, responsáveis até então pelo controle dos processos de manufatura industrial. Como resultado os custos com o tempo de parada, mão de obra e troca de equipamentos, inviabilizava o desenvolvimento e atualização dos produtos, além de prejudicar a competitividade destas empresas. Ao que tudo indica, iniciou-se a utilização de CLPs quando a equipe técnica da empresa *General Motors* ofereceu uma especificação de um módulo a relês, que veio, na época, ao encontro da necessidade da indústria manufatureira como um todo. Desde então, diante das necessidades mercadológicas os CLPs evoluíram com a inclusão de blocos lógicos digitais, aumento da velocidade de processamento, criação de entradas e saídas analógicas para controle de Processos Contínuos e, principalmente modernizações na *interface* com o usuário.

A figura 11 mostra o aspecto físico de um CP compacto:

Figura 11. Aspecto físico de um Controlador Programável CP351 - ALTUS[1].

Fonte: Catalogo ALTUS CP DU351

[1] ALTUS Empresa fabricante de CPs, inversores de frequência, supervisórios, etc. Localizada em São Leopoldo - RS com filiais espalhadas pelo Brasil..

Conforme apresentado em OLIVEIRA et al. (2007), os CLPs podem ser divididos historicamente de acordo com o sistema de programação utilizada.

Os CLPs de “Primeira Geração” segundo HUGHES 1989, e OLIVEIRA et al. 2007, eram programados através da linguagem Assembly, cuja programação dependia diretamente do processador envolvido no projeto. Ou seja, eram equipamentos cuja utilização dependia do conhecimento do usuário sobre o hardware envolvido na confecção do sistema. A evolução para os CLPs de “Segunda Geração” se deu com o aparecimento das primeiras linguagens de programação, que seriam então compiladas e gravadas em programadores de memória EPROM. Depois de programadas estas memórias, assim como nos de primeira geração, eram recolocadas no CLP para a execução do programa.

Na “Terceira Geração” de CLPs os mesmos passam a ter uma interface de programação com o usuário, via teclado ou programador portátil. Esta melhoria no sistema permitiu que o programa fosse alterado, apagado e efetivamente testado através de uma função denominada *Debug*. Acentuou-se também a tendência à utilização de CLPs com bastidores em formato de *Rack*, com módulos de fonte, CPU, entradas e saídas digitais, entradas e saídas analógica entre outros.

A “Quarta Geração” de CLPs aparece com o advento e popularização do computador pessoal (PC). Com a inserção de uma porta serial de comunicação entre o CLP e o PC, passou a ser possível a utilização de diversas linguagens de programação, além de se oferecer a possibilidade - dependendo do *software* de *interface* - de serem realizados os respectivos testes e simulações do programa desenvolvido pelo usuário.

Atualmente, nos sistemas de Automação Industrial, em uma mesma planta de processos estão interligados com a função de gerenciamento e controle de um processo industrial, diversos CLPs de diferentes fabricantes. Estes apresentam a necessidade de estabelecer uma comunicação entre si e com o respectivo “Supervisório de Controle”, o que vem ao encontro da criação de uma padronização entre todos os fabricantes destes equipamentos. Essa padronização levou ao que conhecemos hoje como “Controlador Programável” (CP), termo este recebido pelo desempenho de funções mais complexas que o antigo CLP, tais como; controle de malhas analógicas com a utilização de blocos mais complexos e o aumento da capacidade aritmética.

3.1 Histórico da IEC 61131

A *International Electrotechnical Comission* (IEC) é uma organização internacional, não governamental, sem fins lucrativos, formada por representantes de fabricantes, fornecedores, distribuidores e usuários de CPs pertencentes aos Comitês Nacionais dos países membros de todo mundo. Com sua sede desde 1948 em Genebra a IEC possui diversos centros regionais, dentre eles, um na América Latina, na cidade de São Paulo, no Brasil.

Segundo Karl-Heinz (2001), para oficializar normas relacionadas a utilização de CPs a IEC constituiu em 1992 um grupo formado por representantes de usuários e fabricantes de Controladores Programáveis denominando esta comissão de SC65B WG7. Segundo o mesmo pesquisador, dessa comissão surgiu a norma IEC 61131 que representa a combinação contínua de diferentes normas, fazendo menção a outras dez normas IEC50, IEC 559, IEC617-12, IEC617-13, IEC 848, ISO/AFNOR ISO/IEC 646, ISO 8601, ISO 7185, ISO 7498.

Este grupo teve como objetivo analisar o *hardware*, processo de instalação, testes, documentação, programação e comunicação dos Controladores Programáveis (CPs) buscando definir uma padronização entre os diversos fabricantes. Uma subdivisão em diversos grupos de trabalho ocorreu levando a formação do grupo de trabalho três (3) que recebeu como incumbência o desenvolvimento da padronização de um novo padrão de linguagens para CPs, que passou a se denominar como a parte três na norma 1131. A partir de 1996, a IEC 1131 passou a ser chamada de 61131 devido ao fato da existência do código 1131 presente em diversos países.

A norma IEC 61131-3 foi, portanto, desenvolvida buscando a padronização e visando o atendimento às demandas da comunidade industrial tendo como principais aspectos a adoção de Linguagens de Programação, recursos Multitarefa e Reutilização de *Software*.

Na figura 12 é possível visualizar a evolução da norma ao longo das últimas décadas.

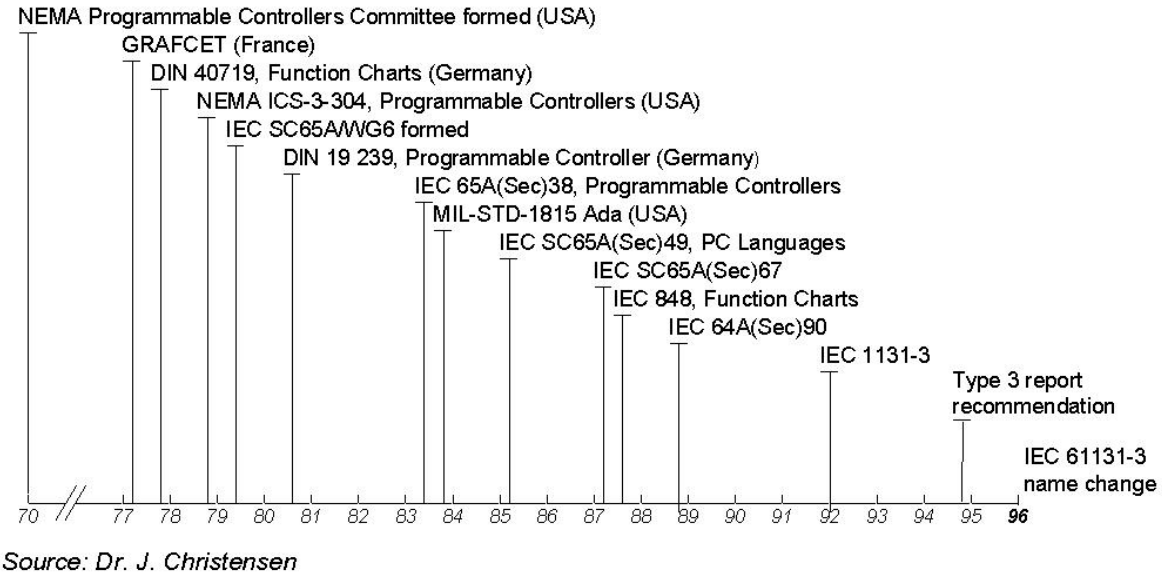


Figura 12. Evolução da Norma IEC 61131-3

Fonte: Guimarães (2005)

Esta norma tem como objetivo gerar uma portabilidade de software entre os diferentes tipos de fabricantes. Encontramos as partes da norma na tabela 1 e discutiremos posteriormente as características especificas do item três.

Tabela 1. Itens da norma IEC 61131-3

Parte	Titulo	Conteúdo	Publicação
Parte 1	***General Information***	Definição da terminologia e conceitos.	**2003 (2ª Ed.)**
Parte 2	***Equipment requirements and tests***	Requisitos de teste de verificação e fabricação eletrônica e Mecânica (hardware).	**2003 (2ª Ed.)**
Parte 3	***Programmable Languages***	**Estrutura do Software do CP, linguagens e execução de programas**	**2003 (2ª Ed.)**
Parte 4	***User guidelines***	Guia de usuário. Orientações para seleção, instalação e manutenção de CP's.	**2004 (2ª Ed.)**
Parte 5	***Communications***	Funcionalidades para comunicação com outros dispositivos.	**2000 (1ª Ed.)**
Parte 6	***Reservada***	Atualmente com estudos sendo desenvolvidos relativo a Comunicação via Fieldbus	
Parte 7	***Fuzzy Control Programming***	Funcionalidades de software, incluindo blocos funcionais padrões para tratamento de lógica nebulosa dentro de CP's.	**2000 (1ª Ed.)**
Parte 8	***Guidelines for the Application and Implementation of Programming Languages***	Guia para aplicação e implementação das linguagens de programação. Orientações para implementação das linguagens **IEC 61131-3.**	**2003 (2ª Ed.)**

3.2 A norma IEC 61131 e a PLCOpen

A PLCOpen é uma associação mundial independente, de produtos e fabricantes, que foi fundada em 1992 tendo sua sede na Holanda. A missão da PLCOpen é liderar a implementação de soluções em assuntos relacionados ao aperfeiçoamento da programação de controladores industriais seguindo as normas vigentes e visando a melhoria da eficiência nos sistemas de controle.

Apesar da norma IEC61131-3 ter padronizado diversos aspectos relativos à programação dos Controladores Programáveis, a mesma deixou abertura aos fabricantes quanto às definições das formas de sua implementação. Isto ocasionou problemas como, por exemplo, a definição da base de tempo dos temporizadores que apresentavam variação de acordo com o fabricante. Logo se verificou que estas variações provocavam falhas e incompatibilidades na migração de programas de usuário entre diversos equipamentos e, que este problema só seria resolvido alterando-se a lógica de programação, o que tornaria inviável o seu reaproveitamento.

Uma das atividades da PLCopen está diretamente ligada à norma IEC 61131-3, que é hoje a única norma mundial para programação de controladores industriais. Como já foi visto, a norma padroniza a forma como os usuários projetam, operam os controladores industriais e principalmente definem sua interface de programação.

A idéia é alcançar uma interface de programação padronizada que permita aos usuários de diferentes formações (automação, computação, etc.) a possibilidade de criar diferentes elementos de um programa em estágios do ciclo de vida do *software* como; especificação, projeto, implementação, teste, implantação e manutenção.

Para isso a norma inclui a definição da linguagem Sequenciamento Gráfico de Funções (SFC), usada para estruturar a organização interna do programa, e quatro linguagens de programação interoperáveis: Lista de Instruções (IL), Diagrama Ladder (LD), Diagrama de Blocos (FBD) e Texto Estruturado (ST). Foi introduzido, portanto, o conceito de decomposição em elementos lógicos, modularização e técnicas de *software* como programação orientada a objeto, onde permite que cada programa seja estruturado de forma a melhorar a sua reutilização, reduzindo erros e melhorando a programação e eficiência para o usuário.

São atividades desenvolvidas pela PLCOpen:

a- Especificação de elementos mandatórios para certificação de produtos em diferentes níveis de conformidade com a norma IEC 61131-3;

b- Desenvolvimento de um software de teste baseado nas especificações de certificação;

c- Realização de testes e certificação de produtos conforme a norma IEC 61131-3;

d- Divulgação da norma e realização de eventos para estimular a adoção

da mesma;

e- Elaboração de bibliotecas de funções e blocos funcionais padronizados para atendimento às necessidades dos usuários da norma;

f- Atuação junto às entidades normalizadoras e usuários para definição de melhorias e sugestões para revisão da norma junto ao IEC;

g- Complementação dos aspectos não cobertos pela norma IEC 61131-3.

Os produtos testados e aprovados pela PLCOpen recebem um selo de acordo com o certificado obtido, como pode ser observado a seguir.

Define as características mínimas a serem obedecidas para que os produtos sejam considerados como aderentes à norma IEC 61131-3 e que utilizam a sintaxe padrão;

Define os requisitos para que alguns elementos de *Software* (blocos) possam ser portados entre diferentes produtos compatíveis com este nível

Define a conformidade quanto à quantidade de tipos de dados declarados baseado na IEC 61131-3;

Define os requisitos para que todos os elementos de software, incluindo toda uma configuração, sejam conformes e compatíveis entre diferentes fabricantes de produtos certificados neste nível.

3.3 Linguagens de Programação segundo a IEC 61131-3

A norma IEC 61131 estabelece em seu item três (3) a padronização de cinco (5) tipos de linguagens de programação. Estes tipos, que estarão sendo descritos a seguir com as respectivas particularidades, são: Sequenciamento Gráfico de Funções (SFC), Lista de Instruções (IL), Diagrama *Ladder* (LD), Diagrama de Blocos (FBD) e Texto Estruturado (ST). Na figura 13 pode ser observado que são consideradas como linguagens textuais IL e STD, e linguagens gráficas SFT, FBD e LD.

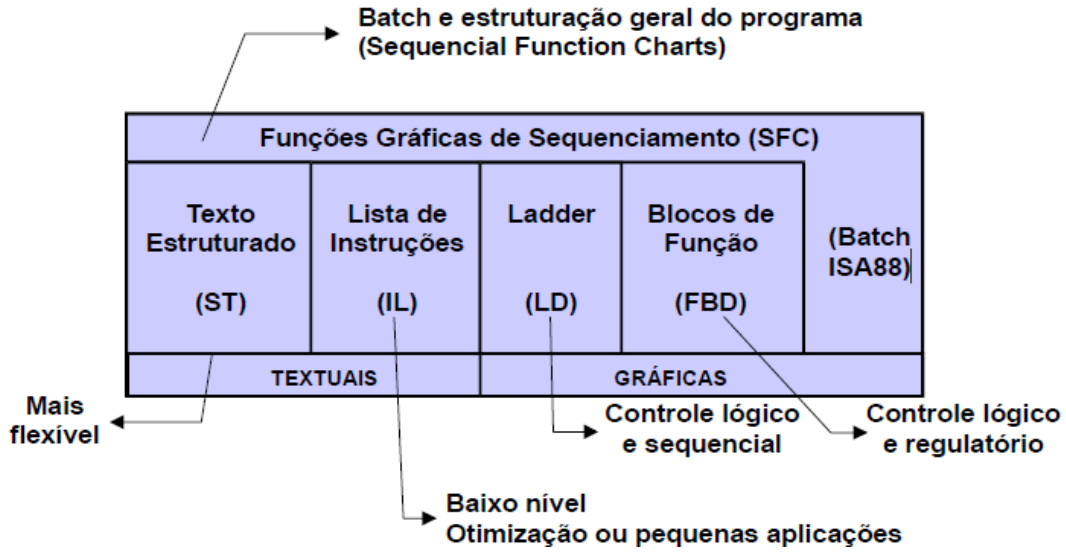


Figura 13. Divisão das linguagens em Textuais e Gráficas

Fonte: Catálogo Altus MasterTool IEC

3.3.1 Diagrama *Ladder*

A linguagem de programação *Ladder Diagram* (LD) é, devido ao fato de apresentar um aspecto gráfico baseado nos diagramas de comandos elétricos, o tipo de linguagem mais utilizada em ambientes industriais, como pode ser observado na figura 14.

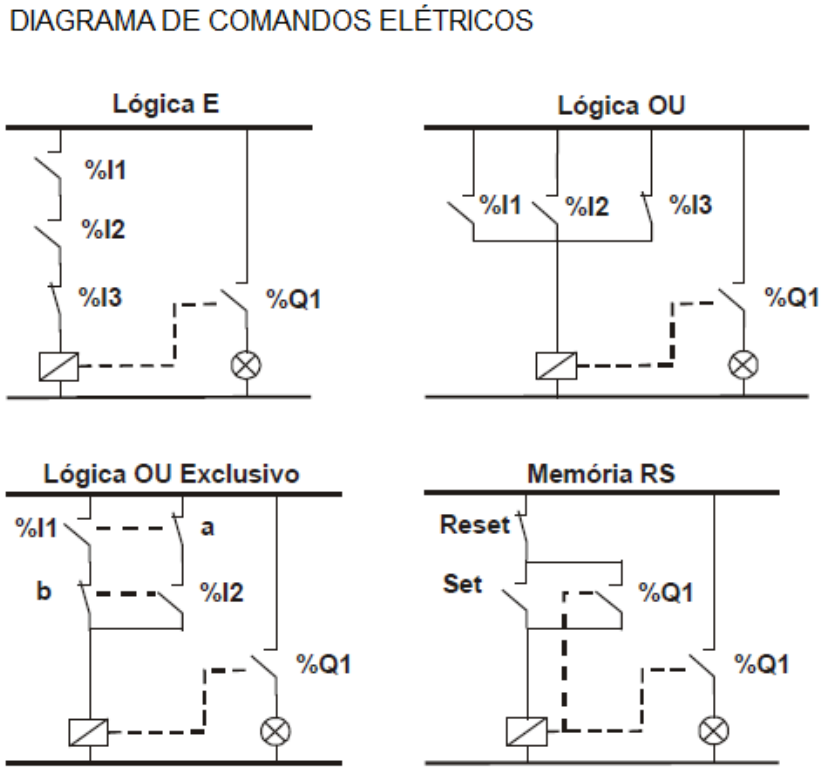


Figura 14. Diagrama de comandos elétricos

Fonte: Catálogo Altus MasterTool IEC

Esta aparência gráfica faz uso da representação de contatos, e bobinas interconectados, destacando o fluxo de energização entre os elementos. É comumente utilizada para descrever o comportamento de blocos funcionais, funções, além de passos, ações e transições na linguagem SFC.

Seus elementos fundamentais de programação são contatos normalmente abertos e fechados como entradas de sinal e bobinas de relés utilizadas como saídas, como pode ser observado na figura 15.

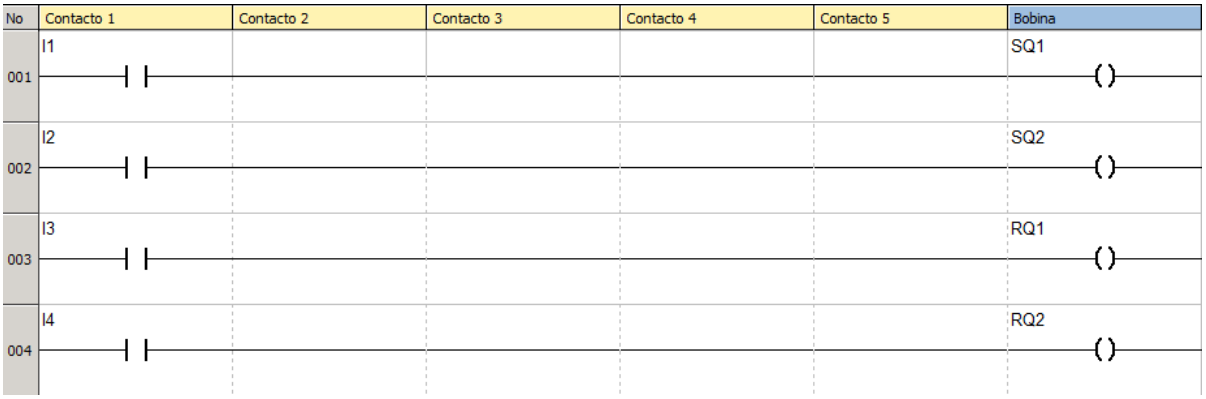


Figura 15. Diagrama *Ladder* (contatos)

Os elementos devem ser ligados entre uma barra vertical à esquerda, que representa um barramento energizado e, a barra da direita que representa o comum do circuito.

A denominação *Ladder*, é proveniente do inglês que significa “escada”.

3.3.2 Texto Estruturado

Outra linguagem definida na IEC 61131-3 é a de Texto Estruturado (ST) *Structured Text*. A ST é uma linguagem de alto nível com sintaxe similar a programação Pascal da norma ISO 7185. Sua principal aplicação é no desenvolvimento de programas computacionais em plantas de controle industrial e possui a capacidade de descrever o comportamento de programas, blocos funcionais, funções, além de passos, ações e transições na linguagem SFC.

Esta linguagem de alto nível *ST* apresenta a característica de flexibilidade, no entanto necessita de especialistas em desenvolvimento de *softwares* para a construção de aplicações voltadas à indústria.

O ST possui comandos comuns em linguagens estruturadas, tais como:

- Comandos da Linguagem Texto Estruturado
- Cálculos Aritméticos
- Comando condicional: *IF THEN ELSE*
- Comando condicional: *CASE*
- Comando de repetição: *FOR ... DO*
- Comando de repetição: *WHILE ... DO*
- Comando de repetição: *REPEAT ... UNTIL*
- *EXIT*
- *RETURN*

3.3.3 Diagrama de Blocos Funcionais

A linguagem de Diagrama de Blocos Funcionais, *Function Block Diagram* (FBD), é outra linguagem gráfica definida pela norma IEC 61131. Esta linguagem possui sua base de programação baseada em diagramas de blocos interconectados,

onde deve ser considerado o fluxo de sinais entre os elementos. É utilizada para descrever o comportamento de programas, blocos funcionais, funções, além de passos, ações e transições na linguagem SFC.

É possível conceituar blocos como sendo um elemento que possui entradas, faz um processamento específico e depois escreve o resultado das operações nas saídas, como podemos observar na figura 16.

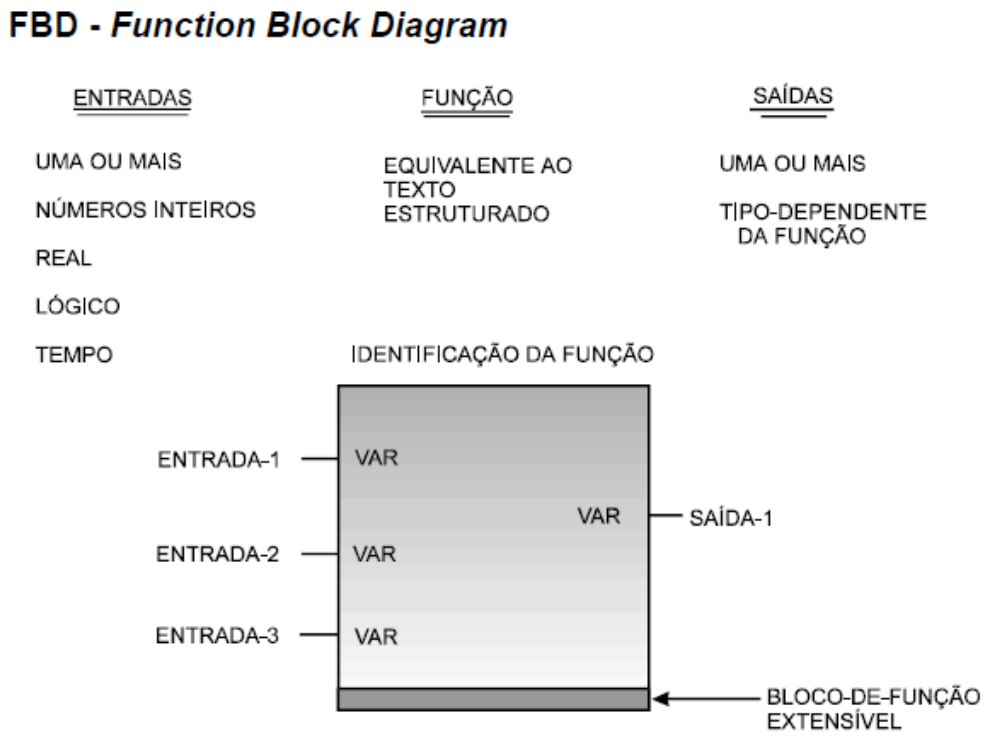


Figura 16. Diagrama de Blocos funcionais

Os blocos são “Classes” que podem ser de dois tipos, o primeiro é denominado de “Bloco Funcional” e o segundo de “Função”. O Bloco Funcional possui a característica de apresentar persistência de dados, ou seja, após ser instanciado, pode ser executado em diversos ciclos de execução. Já o outro tipo de classe denominada “Função” executa a sua funcionalidade e, após a execução, não persiste informação, tendo a característica de escrever o resultado na saída. Na figura 17 são apresentados alguns blocos básicos comuns aos CPs e na figura 18 uma programação de um FDB a partir de um programa desenvolvido em ST. Como característica esta linguagem apresenta seu processamento sendo executado da esquerda para a direta e de cima para baixo no diagrama.

Fonte: Catálogo Altus MasterTool IEC

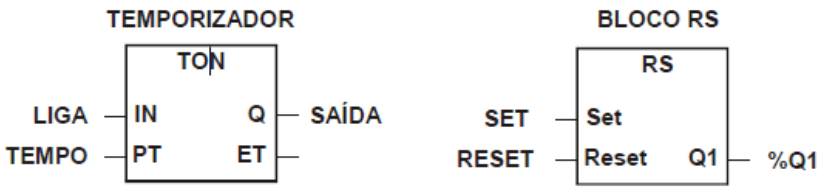


Figura 17. Blocos funcionais básicos

Fonte: Catálogo Altus MasterTool IEC

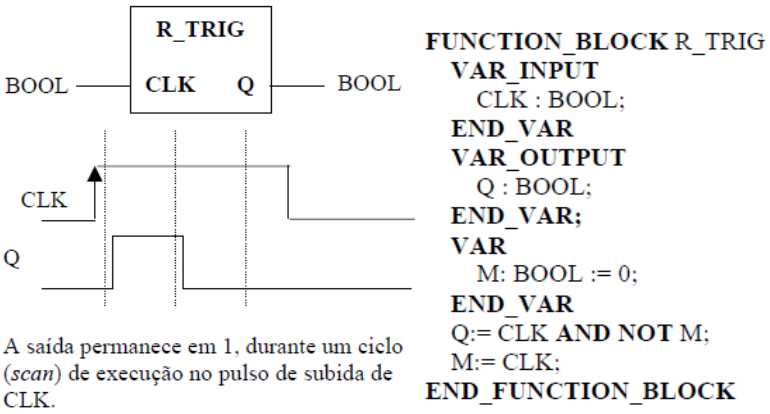


Figura 18. Criação de Bloco Funcional A Partir de Um Programa em ST

Fonte: Guimarães (2005)

Os blocos funcionais, assim como as funções, são comumente utilizados em conjunto com o Diagrama *Ladder* puro, proporcionando, portanto a interação entre as linguagens de programação como preconiza a norma.

3.3.4 Lista de Instruções

A linguagem Lista de Instruções, *Instruction List* (IL), é um tipo de linguagem textual de baixo nível que tem a característica de apresentar uma estrutura muito similar à linguagem *Assembly*. Sua grande vantagem é na velocidade de processamento, devido ao seu baixo tempo de compilação para linguagem de máquina utilizada pelo processador. É indicada sua utilização na otimização de código onde o tempo de execução seja importante, como por exemplo, na leitura de entradas de sinais proveniente de sensores.

A estrutura das instruções possui; o operador, um operando e um modificador opcional. São comumente utilizados marcadores (*label*) no intuito de indicar as instruções de salto. Todas as instruções e operações são direcionadas a um registrador de acumulação que será o destino ou origem dos dados, de acordo com a instrução a ser executada. Na figura 19 é apresenta uma tabela de operadores e operandos com suas respectivas funções.

Operador	Modificador	Operando	Comentários
LD	N	Qualquer	Carrega operando no acumulador
ST	N	Qualquer	Armazena acumulador no operando
ST		BOOL	Reset operando para TRUE
R		BOOL	Reset operando para FALSE
AND	N,(	BOOL	E booleno
&	N,(	BOOL	equivalente a E
OR	N,(	BOOL	OU booleano
XOR	N,(	BOOL	OU exclusivo
ADD	(	Qualquer	Adição
SUB	(	Qualquer	Subtração
MUL	(	Qualquer	Multiplicação
DIV	(	Qualquer	Divisão

Figura 19. Tabela de operadores e operandos

Fonte: Guimarães (2005)

Na figura 20 é possível verificar a apresentação de alguns blocos funcionais utilizados em linguagem *Ladder*.

As suas respectivas programações em LI podem ser observadas na figura 21.

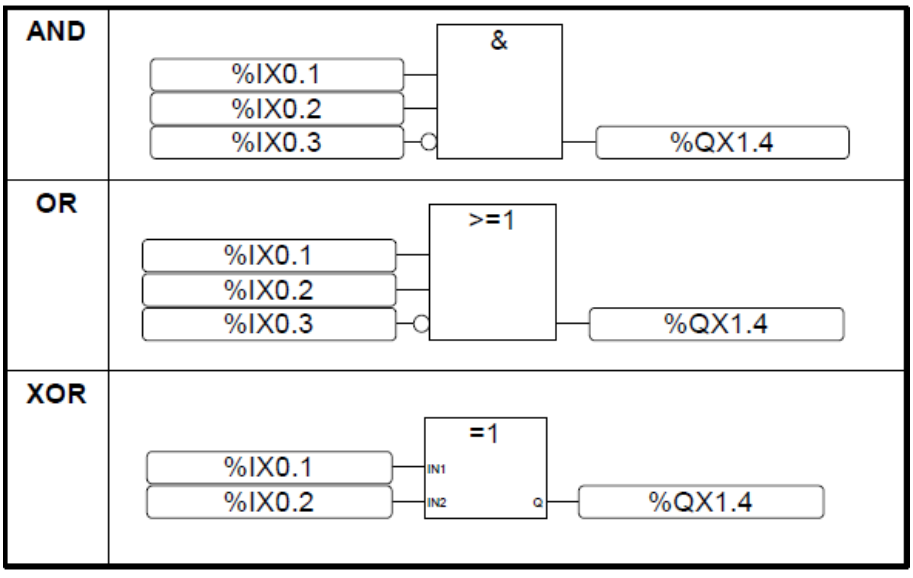


Figura 20. Blocos funcionais utilizados em ladder

Fonte: Catálogo Altus MasterTool IEC

AND	OR	XOR
LD %IX0.1 AND %IX0.2 ANDN %IX0.3 ST %QX1.1	LD %IX0.1 OR %IX0.2 ORN %IX0.3 ST %QX1.1	LD %IX0.1 XOR %IX0.2 ST %QX1.1

Figura 21. Programação em LI de Blocos Funcionais

Fonte: Catálogo Altus MasterTool IEC

Pode ser observado com os exemplos das figuras 18, 20 e 21 o atendimento a norma no que se refere à reutilização de software com o reaproveitamento de código, além da disponibilização de recursos Multitarefa. Na figura 22 é apresentado um novo exemplo de código que pode ser escrito em LI ou ST.

• ST

```
IF saida = 4 THEN
        Q00 := TRUE;

ELSE
        Q00 := FALSE;
END_IF
```

IL

```
        LD      saida1
        EQ      4
        NOT
        JMPC    else4_0
        LD      TRUE
        ST      Q00
        JMP     end4_0
else4_0:
        LD      FALSE
        ST      Q00
end4_0:
```

Figura 22. Programação em LI e ST

Fonte: Catálogo Altus MasterTool IEC

3.3.5 Sequenciamento Gráfico de Funções

A quinta e última linguagem de programação definida pela IEC 61131-3 é a de Sequenciamento Gráfico de Funções - *Sequential Function Charts* (SFC). O estudo desta norma é baseado em técnicas de comportamento seqüencial. Como referência é usado o padrão europeu conforme descrito na norma IEC 848 que, por sua vez, foi baseado em Redes de Petri, sendo a linguagem de programação denominada de *Grafcet*. A norma IEC 61131 introduziu modificações na IEC 848 de forma a adequar o SFC as outras linguagens da norma. A norma ISA SP 88 define o SFC como uma linguagem própria para ser utilizada em sistemas de controle de bateladas.

O fluxo de leitura de um programa em SFC é de cima para baixo, havendo em muitas oportunidades ramo de retorno. Cada passo executa uma ação de forma

constante ou orientada a eventos, tais como, entrada e saída do estado, como pode ser observado na figura 23.

A linguagem SFC é composta por vários passos conectados por linhas verticais, sendo que cada passo representa um estado onde o programa permanece enquanto a condição de transição descrita na linha de conexão entre os passos não é satisfeita, como pode ser observado na figura 24.

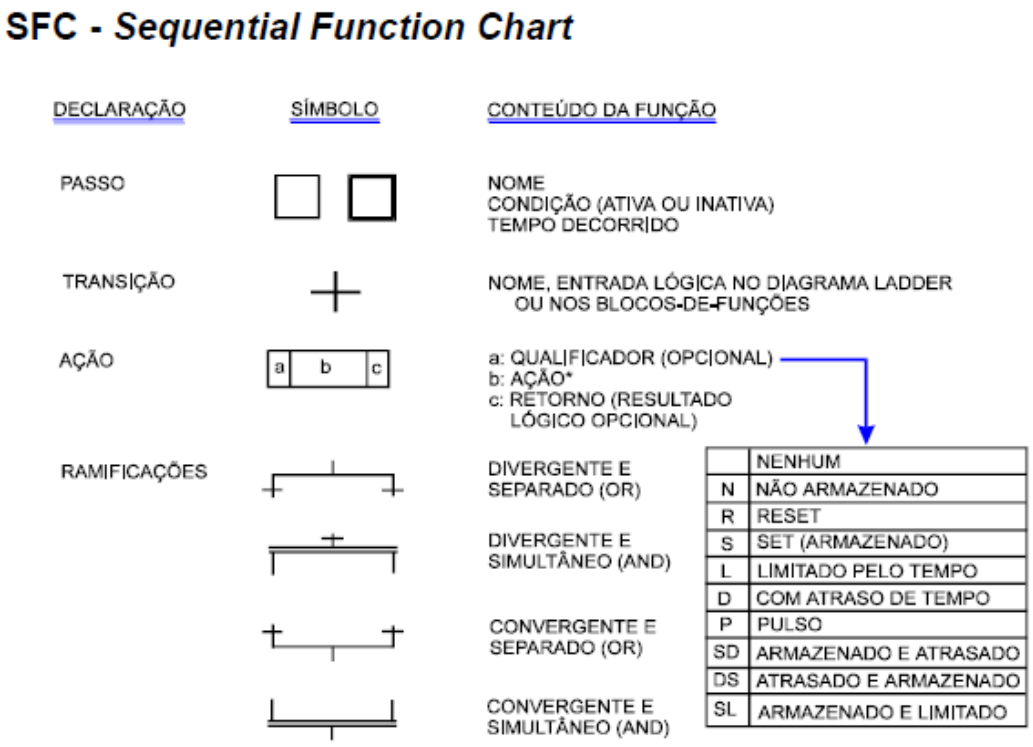


Figura 23. Simbologia da linguagem SFC

Fonte: Catálogo Altus MasterTool IEC

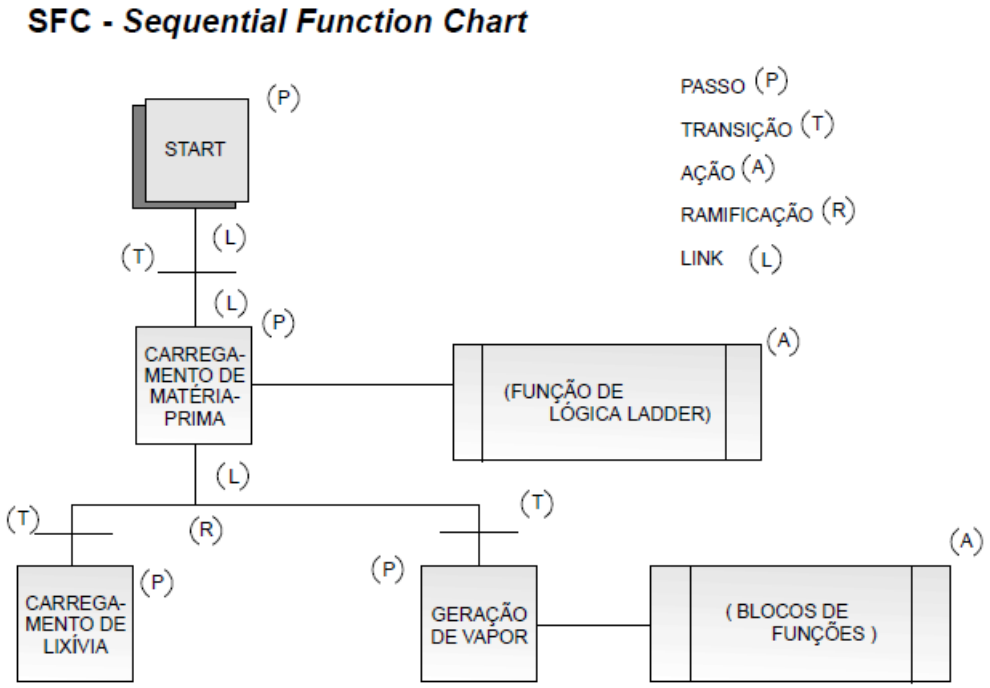


Figura 24. Exemplo de programa em SFC

Fonte: Catálogo Altus MasterTool IEC

A descrição do SFC pode ser feita utilizando duas representações gráficas. A primeira é a descrita na norma IEC 6113-3 e a segunda uma alternativa mais amigável, denominada “Gráfico Contínuo de Função”, conforme será vista no próximo item.

3.3.6 Gráfico Contínuo de Funções

A linguagem de programação denominada Gráfico Contínuo de Funções, (*Continuous Function Charts* (CFC)) é uma linguagem gráfica não descrita pela IEC 61131, no entanto esta linguagem vem surgindo no mercado como um complemento às indicadas pela norma. A CFC é considerada como uma forma de aperfeiçoamento da norma, e, apesar de guardar semelhanças com a linguagem FBD, faz uso de uma numeração nos blocos, que é muito útil na indicação da seqüência de execução do diagrama. De forma geral, este método é um facilitador no desenvolvimento e na compreensão do diagrama.

3.3.6.1 Exemplo de aplicação

Como forma de aplicação e entendimento da norma com a linguagem CFC, é apresentado um exemplo de um processo simples para tomadas de decisão futura. Neste exemplo é feito a utilização de todas as linguagens, no qual é considerada para a análise uma função aplicada ao calculo do peso de um determinado lote com o desconto do peso fixo da tara.

Os diversos tipos de linguagens desse processo considerado como exemplo são mostradas na figura 25.

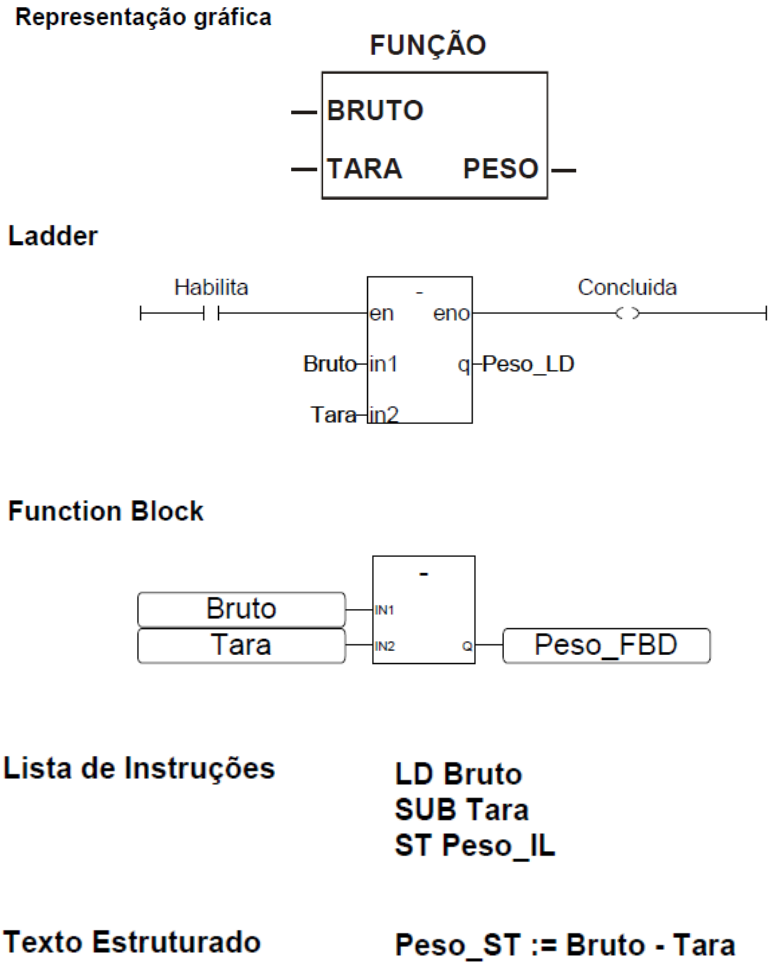


Figura 25. Exemplo de programa nas linguagens da IEC 61131-3.

Fonte: Catálogo Altus *MasterTool* IEC

4 DESENVOLVIMENTO DO BLOCO FUNCIONAL (BF_PARACON)

Neste capítulo é mostrada a metodologia aplicada para a criação do bloco funcional (BF_Paracon) que compõem Algoritmo Para-Analisador da LPA2v utilizando as técnicas de programação estruturadas na norma IEC 61131-3.

Conforme visto em Da Silva Filho, 1999, e em Lima Jr 2003, o algoritmo Para-Analisador tem sido aplicado com êxito em *hardware* e *software* de sistemas de controle de robôs móveis autônomos e em controle de processos contínuos de temperatura. No entanto, verifica-se que todos os desenvolvimentos foram dedicados, nos quais os sistemas de controle construídos não apresentaram modularidade e nem condições para reutilização de código.

Seguindo os objetivos deste trabalho de pesquisa, o algoritmo proveniente da LPA2V foi implementado em linguagem de Texto Estruturado com a criação de uma função que deu origem a um bloco funcional denominado BF_Paracon na linguagem FBD. Desta forma, com a criação do bloco, foi possível sua inserção para os devidos testes em uma linguagem de programação *Ladder*, ou seja, foi criado um módulo (ou biblioteca) para ser utilizado em tempo de execução.

O bloco funcional (BF_Paracon) engloba características de uma lógica não-clássica. Pode ser demonstrado que a sua modularidade facilita a implementação de sistemas de controle de automação baseados em fundamentos da Lógica Paraconsistente. Além disso, a modularidade deste Bloco Funcional permite a união de blocos que atuam em conjunto com lógicas clássicas e em sistemas baseados em outras linguagens pertencentes a Norma (*Ladder*).

Como ferramenta de programação no desenvolvimento desse trabalho foi utilizado o *software* programador *MasterTool* IEC, que será detalhado a seguir.

4.1 O Software de Programação *Mastertool* IEC do CP DU351 da Altus

O software programador *MasterTool* IEC, que foi utilizado como ferramenta de programação para elaboração do BF_Paracon, possibilita a utilização das cinco linguagens de programação descritas pela norma IEC 61131-3. Essa ferramenta de programação permite que todas as cinco linguagens de programação possam ser

inseridas simultaneamente em um mesmo projeto. Sendo assim, o bloco BF_Paracon foi desenvolvido com as cinco linguagens onde soma-se ainda a uma linguagem denominada de Gráfico Contínuo de Funções (CFC). Sua programação foi elaborada atendendo as diretrizes da norma e, facilitando a reutilização de código. Para isso utiliza-se o processo de criação de blocos de programação denominados de “Unidades de Organização de Programas” (POUs). No *MasterTool* IEC são utilizadas três os tipos de POUs:

• Funções;

• Blocos funcionais;

• Programas.

Estas POUs são basicamente as formas de implementação de programas no CP, através da associação de variáveis e instruções, podendo uma POU ser parte integrante de outra POU. Segundo a norma IEC 61131-3 deve ser estabelecida uma hierarquia entre as POUs seguindo os seguintes princípios:

- Funções somente podem conter Funções;
- Blocos Funcionais podem conter outros Blocos Funcionais e Funções;
- Programas podem conter qualquer tipo de POU.

O conceito de função, segundo a norma IEC 61131-3 deve apresentar em sua forma organizacional as seguintes características fundamentais:

- Podem ser utilizados como elementos comuns para a definição de novas POUs.
- São elementos reutilizáveis de software.
- Produzem um único dado como resultado, que é próprio nome da função.
- Só existem em tempo de execução.
- Deve ser feita a declaração do tipo antes do uso.
- Não possui memória de estados, ou seja, se houver o chamamento de uma função com as mesmas entradas seu resultado deve ser o mesmo.
- As respectivas chamadas em linguagens textuais devem ser feitas pelo nome e em linguagens gráficas pelo bloco.

O conceito aplicado à linguagem de Blocos Funcionais, é baseado no item da norma que se refere à reutilização de código e modularidade. Dentre os seus fundamentos tem como características principais:

- Ser um dos principais elementos para estruturação de programas do modelo *top-down botton-up*.
- Ser um elemento encapsulado de software que deve apresentar a possibilidade de reutilização.
- Somente tipos de dados definidos podem ser utilizados por interfaces externas.
- Valores das variáveis são mantidos entre uma execução e outra.
- Um Bloco Funcional é representado por um retângulo com pinos de entrada do lado esquerdo e pinos de saída do lado direito (conforme pode ser observado no bloco apresentado na figura 26).
- Deve apresentar variáveis de entrada e saída além de poderem apresentar em seu código a utilização de variáveis internas.
- Para cada instância declarada é reservada uma área de memória.
- Devem ser declarados o tipo e as instâncias do Bloco Funcional.
- Um BF pode produzir uma ou mais variáveis de saída como observado no bloco da figura 26.
- As variáveis internas não são acessíveis a não ser pelo algoritmo interno do bloco.
- Uma instância de um Bloco Funcional pode ser utilizada na declaração de outro Bloco Funcional ou Programa.
- A declaração de um Bloco Funcional é feita utilizando a estrutura inicial FUNCTION_BLOCK e terminação END_FUNCTION_BLOCK.
- Não é permitida a declaração de variáveis globais dentro de um Bloco Funcional.

Os Programas são considerados a maior forma de construção de uma POU, podendo nele estarem contidos agrupamentos de Funções e Blocos Funcionais. Um Programa segundo a norma, pode ser o elemento responsável por executar uma função de controle, ou ser declarado a nível de recurso para ser utilizado por um outro programa principal, como por exemplo, a criação de uma biblioteca que será manipulada por um programa com a função de controle de uma variável de processo.

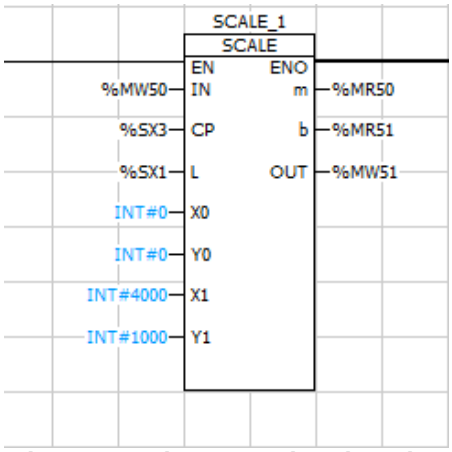


Figura 26. Bloco Funcional Scale

4.2 Inicialização do Mastertool IEC

Para criação do Bloco funcional BF_Paracon é necessário inicialmente realizar as configurações básicas do *software*. O *MasterTool* IEC possui, de acordo com o modelo do CP utilizado, algumas configurações básicas fornecidas pelo fabricante. Nestas configurações, estão inseridas as inicializações para funcionamento da IHM, display, etc. Para a versão de *software* compatível com o CP DU 351 utilizado no projeto, foi escolhido o "Modelo_DU350_DU351_v110", conforme mostrado nas figuras 27 e 28.

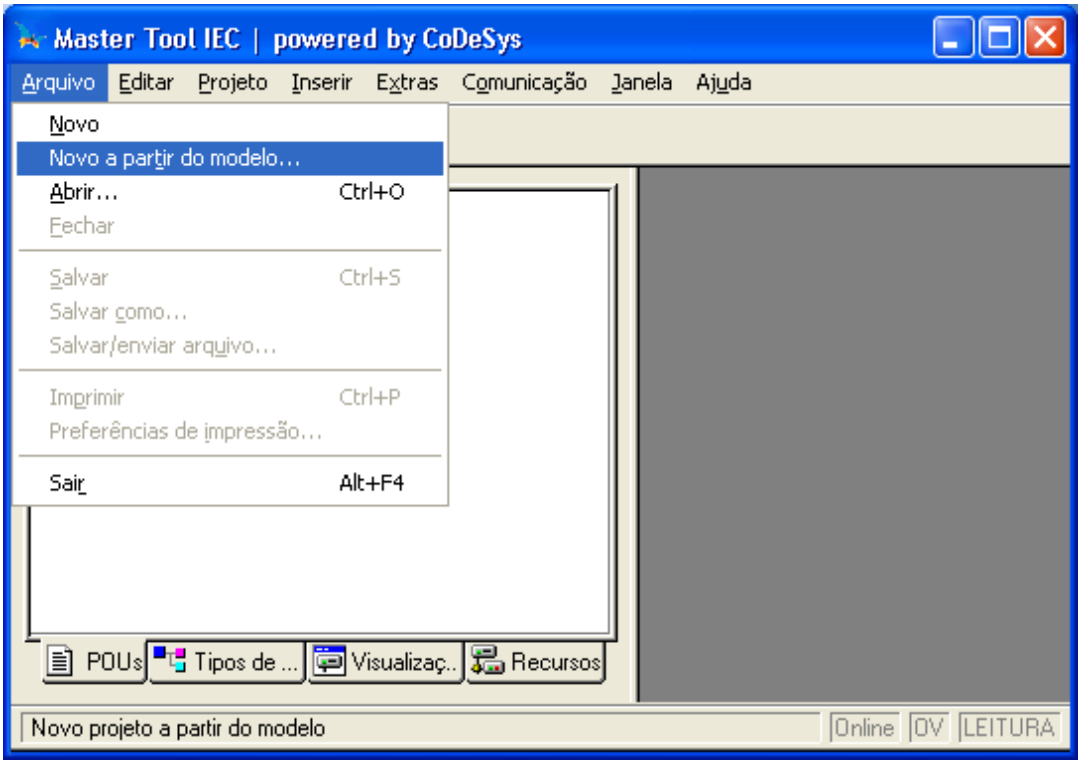


Figura 27. Inicialização do Projeto do BF_Paracon

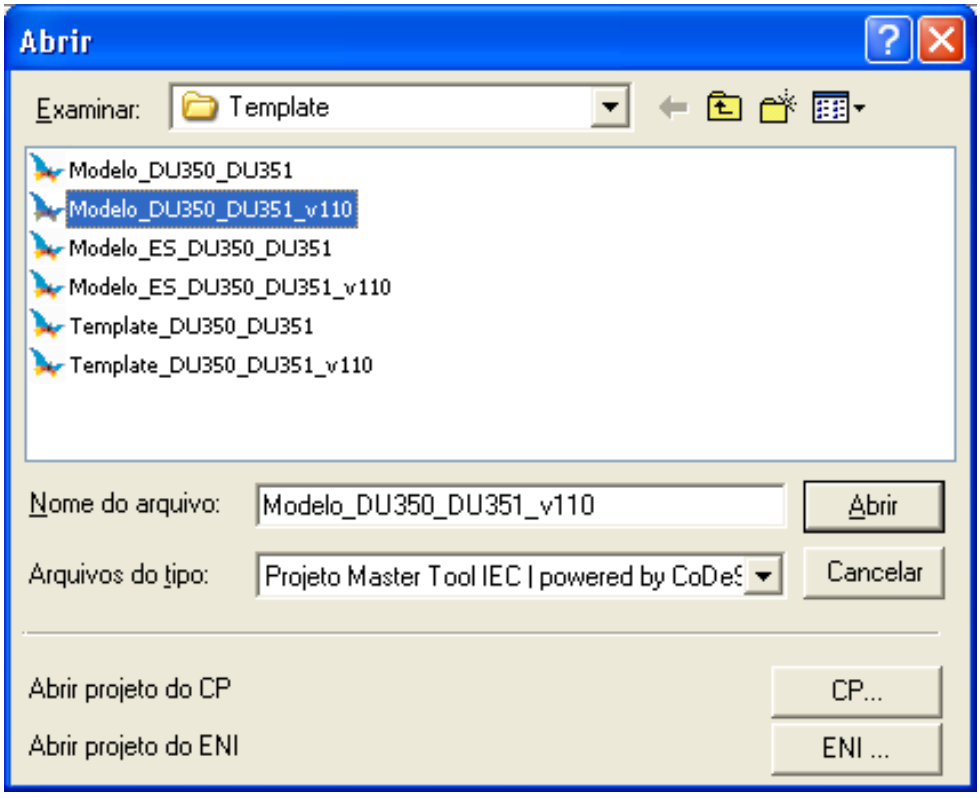


Figura 28. Escolha do Modelo de acordo com o CP

Como já discutido no item 4.1 e seguindo indicação da norma, uma POU, pode ser um programa, uma função ou um bloco funcional escrito em qualquer uma das linguagens da IEC 61131.

O modelo utilizado possui uma “POU” incorporada ao produto denominada “NAVEGA”. Na prática, o NAVEGA é um programa criado automaticamente pelo Modelo_DU350_DU351_v110 escolhido, no qual o programador pode declarar o nome das telas habilitando a navegação entre elas usando o próprio *software* ou as setas do teclado do CP DU 351.

Na figura 29 pode ser observada a tela com a criação do programa NAVEGA e, na figura 30, a sua respectiva programação criada pelo modelo.

É possível observar neste programa que é criada uma matriz de 31 elementos (31 POUs), sendo cada uma possível de ser descrita por uma *string* de dez caracteres.

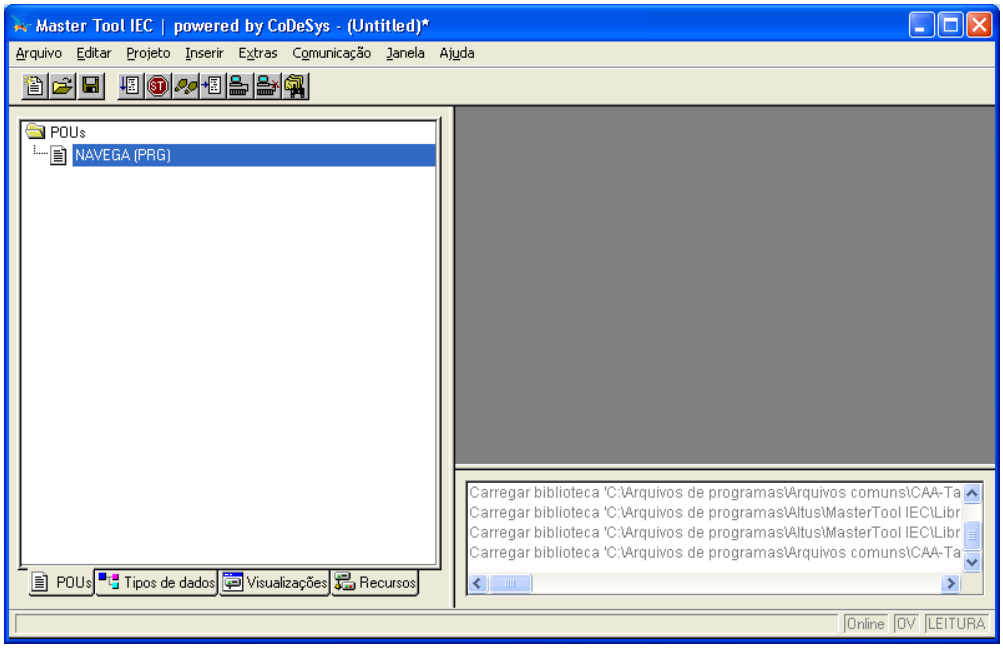


Figura 29. Criação pelo Modelo do programa NAVEGA

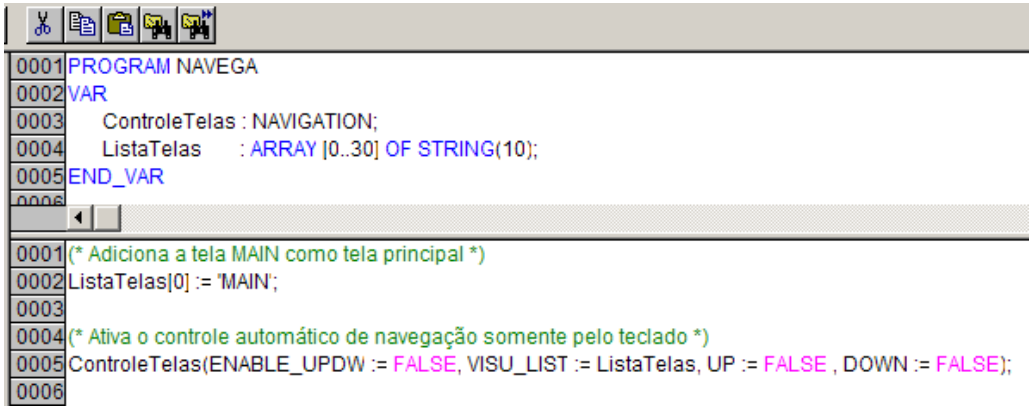


Figura 30. Programação do MODELO para o programa NAVEGA

4.3 Construção do Bloco Funcional BF_Paracon

Após a escolha do modelo, que por sua vez é o responsável pela inicialização do programa NAVEGA, é então possível a criação de novas POUs. Segundo o fabricante a primeira POU do projeto a ser criada deve receber o nome de PLC_PRG. Neste projeto foi escolhida a opção de programa em linguagem *Ladder*, com a qual é possivel comprovar a modularidade do Bloco Funcional BF_Paracon no controle de uma planta de temperatura.

Na figura 31 pode ser observada a inicialização de uma nova POU e, na figura 32, a criação obrigatória do programa PLC_PRG em linguagem *Ladder*.

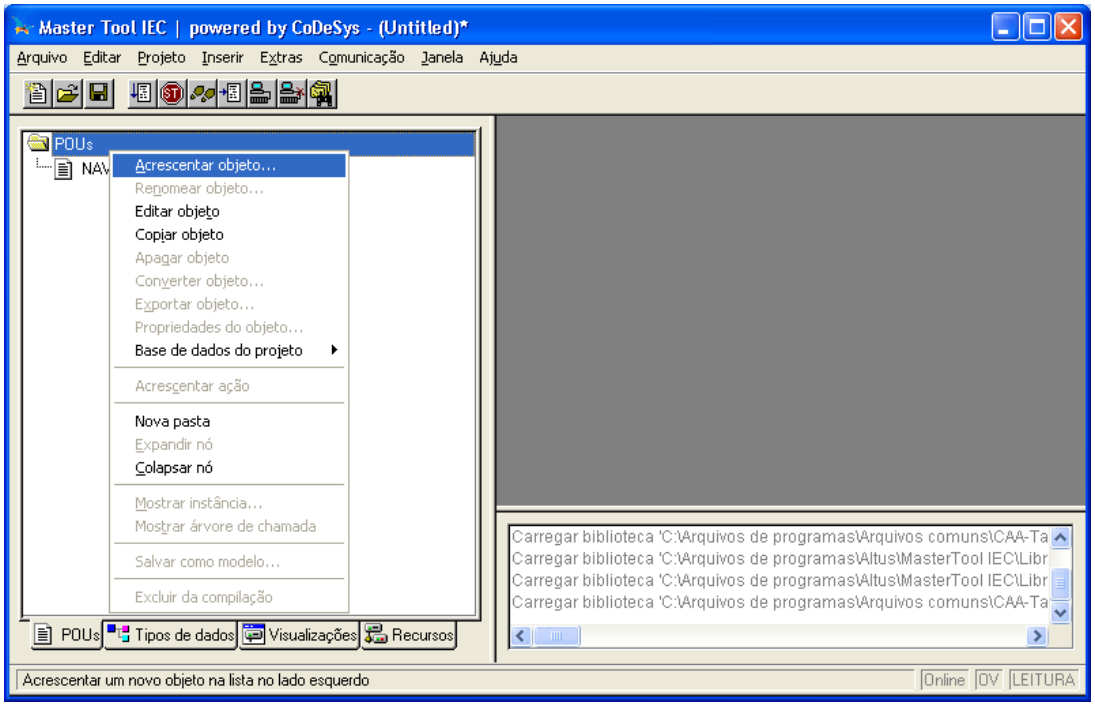


Figura 31. Criação de novas POUs

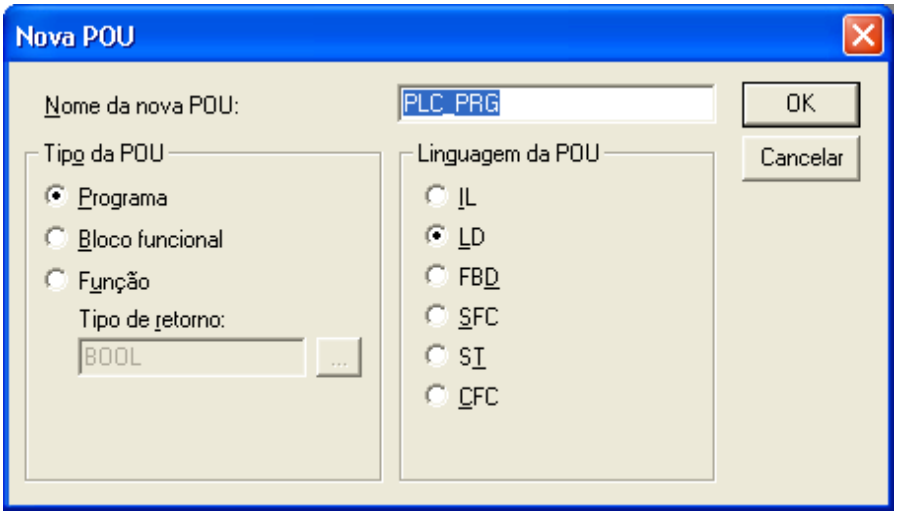


Figura 32. Criação do programa PLC_PRG em Ladder.

Após a criação da POU com o programa PLC_PRG, foi possível a criação de novas POUs que constituem o Bloco do Algoritmo Para-Analisador. Inicialmente, para a construção do Bloco Funcional BF_Paracon, foram seguidos os mesmos passos das figuras 31 e 32, sendo escolhida a opção de Função na qual foi implementado o algoritmo com o nome *PARACON* na linguagem de Texto Estruturado “ST”, como pode ser observado na figura 33.

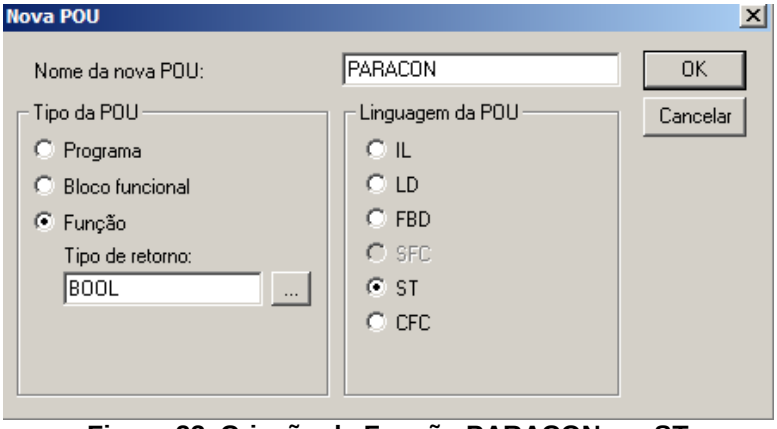


Figura 33. Criação da Função PARACON em ST

4.4 Adequação de valores para o Bloco Funcional BF_Paracon

A programação da Função PARACON foi baseada no algoritmo Para-Analisador da LPA2V. Para adequação dos valores de engenharia envolvidos na simulação os valores calculados dos Graus de Certeza e Graus de Contradição foram utilizados entre o intervalo de -10000 e +10000 representados através de valores inteiros, como pode ser observado na figura 34.

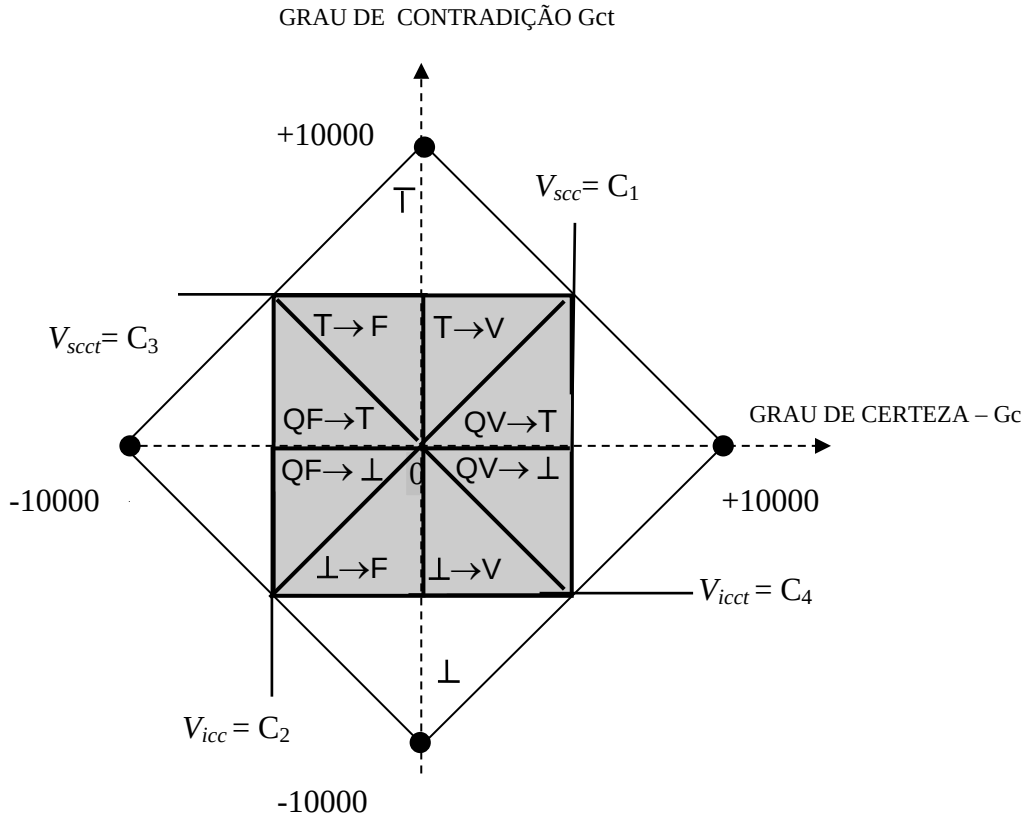


Figura 34. Reticulado da LPA2v repartido em 12 estados lógicos com a adequação dos valores de Graus de Certeza e de Contradição.

Levando em consideração que a implementação do algoritmo original possui uma variação no intervalo de -1 e +1 foi elaborada uma comparação entre os valores estabelecidos na adequação. Na tabela 2 é apresentada a comparação dos valores com os correspondentes fatores de aproximação utilizados.

Tabela 2. Tabela de conversão adotada

Valor representativo em real	Valor representativo em inteiro
0,0000	0
0,0001	1
0,001	10
0,01	100
0,1	1 000
1,0	10 000

4.5 Ajustes com base nos resultados iniciais do Bloco Funcional BF_Paracon

Apesar de o *software* utilizado apresentar a possibilidade de trabalhar com números reais, segundo testes iniciais realizados, as sucessivas conversões em um número de casas decimais definidos ocasionaram um acumulo de erros na leitura que poderiam influenciar de modo significativo na tomada de decisão do Bloco Funcional construído. Devido a esta constatação foi feita a opção pelo desenvolvimento do bloco a partir da utilização de números inteiros, conforme apresentado na tabela 2.

4.6 Programação básica do Bloco Funcional BF_Paracon

A seguir é apresentada a programação com a coleta de dados iniciais da Função PARACON, a obtenção dos índices inferiores e superiores limítrofes que terão seus valores recebidos a partir da criação do bloco. Toda a programação é realizada na linguagem de Texto Estruturado e também contém as operações matemáticas que serão utilizadas na sequência do programa.

```
(*Inicializando Programação*)
(*Indices Superiores e inferiores para definição*)
Vscc;
Vicc;
Vscct;
Vicct;


(*Variaveis do programa*)
u1;
u2;
S2a;
S2b;
Gct;
Gc;

(*Operações Matemáticas*)
Gct:=(u1+u2)-10000;
Gc:=u1-u2;

modulo_Gc:=0-Gc;
modulo_Gct:=-Gct;
```

A partir das operações matemáticas realizadas é possível a determinação dos estados extremos e não-extremos como descrito a seguir.

```
(*Determinação dos casos lógicos extremos*)
IF(Gc>=Vscc)THEN
        verdadeiro:=1; (*Verdadeiro*)
        falso:=0;
        inconsistente:=0;
        indeterminado:=0;
        Q_verdadeiro_inconsistente:=0;
        inconsistente_verdadeiro:=0;
        Q_verdadeiro_indeterminado:=0;
        indeterminado_verdadeiro:=0;
        Q_falso_indeterminado:=0;
        indeterminado_falso:=0;
        Q_falso_inconsistente:=0;
        inconsistente_falso:=0;

ELSIF(Gc<=Vicc)THEN
        verdadeiro:=0;
        falso:=1; (*falso*)
        inconsistente:=0;
        indeterminado:=0;
        Q_verdadeiro_inconsistente:=0;
        inconsistente_verdadeiro:=0;
```

```
        Q_verdadeiro_indeterminado:=0;
        indeterminado_verdadeiro:=0;
        Q_falso_indeterminado:=0;
        indeterminado_falso:=0;
        Q_falso_inconsistente:=0;
        inconsistente_falso:=0;


ELSIF(Gct>=Vscct)THEN
        verdadeiro:=0;
        falso:=0;
        inconsistente:=1;(*Inconsistente*)
        indeterminado:=0;
        Q_verdadeiro_inconsistente:=0;
        inconsistente_verdadeiro:=0;
        Q_verdadeiro_indeterminado:=0;
        indeterminado_verdadeiro:=0;
        Q_falso_indeterminado:=0;
        indeterminado_falso:=0;
        Q_falso_inconsistente:=0;
        inconsistente_falso:=0;

ELSIF(Gct<=Vicct)THEN
        verdadeiro:=0;
        falso:=0;
        inconsistente:=0;
        indeterminado:=1; (*Indeterminado*)
        Q_verdadeiro_inconsistente:=0;
        inconsistente_verdadeiro:=0;
        Q_verdadeiro_indeterminado:=0;
        indeterminado_verdadeiro:=0;
        Q_falso_indeterminado:=0;
        indeterminado_falso:=0;
        Q_falso_inconsistente:=0;
        inconsistente_falso:=0;

END_IF


(*Determinação dos estados Lógicos não-Extremos*)
(*primeira fase*)


IF(Gc<Vscc AND Gc>=0 AND Gct<Vscct AND Gct>=0) THEN
        IF(Gc>=Gct)THEN
                verdadeiro:=0;
                falso:=0;
                inconsistente:=0;
                indeterminado:=0;
                Q_verdadeiro_inconsistente:=1; (*Quase verdadeiro tendendo a
inconsistente*)
```

```
            inconsistente_verdadeiro:=0;
            Q_verdadeiro_indeterminado:=0;
            indeterminado_verdadeiro:=0;
            Q_falso_indeterminado:=0;
            indeterminado_falso:=0;
            Q_falso_inconsistente:=0;
            inconsistente_falso:=0;
        ELSE
            verdadeiro:=0;
            falso:=0;
            inconsistente:=0;
            indeterminado:=0;
            Q_verdadeiro_inconsistente:=0;
            inconsistente_verdadeiro:=1; (*Inconsistente tendendo a verdadeiro*)
            Q_verdadeiro_indeterminado:=0;
            indeterminado_verdadeiro:=0;
            Q_falso_indeterminado:=0;
            indeterminado_falso:=0;
            Q_falso_inconsistente:=0;
            inconsistente_falso:=0;
        END_IF

(*segunda fase _pesquisar modulo*)

ELSIF(Gc<Vscc AND Gc>=0 AND Gct<=0 AND Gct>Vicct) THEN
        IF(Gc>=modulo_Gct)THEN (*necessita modulo de Gct*)
            verdadeiro:=0;
            falso:=0;
            inconsistente:=0;
            indeterminado:=0;
            Q_verdadeiro_inconsistente:=0;
            inconsistente_verdadeiro:=0;
            Q_verdadeiro_indeterminado:=1; (*Quase verdadeiro tendendo a
indeterminado*)
            indeterminado_verdadeiro:=0;
            Q_falso_indeterminado:=0;
            indeterminado_falso:=0;
            Q_falso_inconsistente:=0;
            inconsistente_falso:=0;
        ELSE
            verdadeiro:=0;
            falso:=0;
            inconsistente:=0;
            indeterminado:=0;
            Q_verdadeiro_inconsistente:=0;
            inconsistente_verdadeiro:=0;
            Q_verdadeiro_indeterminado:=0;
            indeterminado_verdadeiro:=1; (*Indeterminado tendendo a verdadeiro*)
            Q_falso_indeterminado:=0;
            indeterminado_falso:=0;
```

```
            Q_falso_inconsistente:=0;
            inconsistente_falso:=0;
        END_IF



(*terceira fase _pesquisar modulo*)

ELSIF(Gc>Vicc AND Gc<=0 AND Gct<=0 AND Gct>Vicct) THEN
        IF(modulo_Gc>=modulo_Gct)THEN (*necessita modulo de Gc e Gct*)
            verdadeiro:=0;
            falso:=0;
            inconsistente:=0;
            indeterminado:=0;
            Q_verdadeiro_inconsistente:=0;
            inconsistente_verdadeiro:=0;
            Q_verdadeiro_indeterminado:=0;
            indeterminado_verdadeiro:=0;
            Q_falso_indeterminado:=1; (*Quase falso tendendo a indeterminado*)
            indeterminado_falso:=0;
            Q_falso_inconsistente:=0;
            inconsistente_falso:=0;
        ELSE
            verdadeiro:=0;
            falso:=0;
            inconsistente:=0;
            indeterminado:=0;
            Q_verdadeiro_inconsistente:=0;
            inconsistente_verdadeiro:=0;
            Q_verdadeiro_indeterminado:=0;
            indeterminado_verdadeiro:=0;
            Q_falso_indeterminado:=0;
            indeterminado_falso:=1; (*Indeterminado tendendo a falso*)
            Q_falso_inconsistente:=0;
            inconsistente_falso:=0;
        END_IF

(*quarta fase _pesquisar modulo*)

ELSIF(Gc>Vicc AND Gc<=0 AND Gct>=0 AND Gct<Vscct) THEN
        IF(modulo_Gc>=Gct)THEN (*necessita modulo de Gc e Gct*)
            verdadeiro:=0;
            falso:=0;
            inconsistente:=0;
            indeterminado:=0;
            Q_verdadeiro_inconsistente:=0;
            inconsistente_verdadeiro:=0;
            Q_verdadeiro_indeterminado:=0;
```

```
            indeterminado_verdadeiro:=0;
            Q_falso_indeterminado:=0;
            indeterminado_falso:=0;
            Q_falso_inconsistente:=1; (*Quase falso tendendo a inconsistente*)
            inconsistente_falso:=0;
        ELSE
            verdadeiro:=0;
            falso:=0;
            inconsistente:=0;
            indeterminado:=0;
            Q_verdadeiro_inconsistente:=0;
            inconsistente_verdadeiro:=0;
            Q_verdadeiro_indeterminado:=0;
            indeterminado_verdadeiro:=0;
            Q_falso_indeterminado:=0;
            indeterminado_falso:=0;
            Q_falso_inconsistente:=0;
            inconsistente_falso:=1; (*Inconsistente tendendo a falso*)
        END_IF
END_IF
S2a:=Gct;
S2b:=Gc;
```

A partir da Função PARACON, foi possível a criação do bloco BF_Paracon acrescentando-se uma nova POU na linguagem FBD (Bloco Funcional) e a respectiva importação da função. Todo o bloco funcional, como já descrito, possui suas entradas situadas à esquerda do bloco e suas saídas localizadas a direita.

Na figura 35, é possível verificar o programa do BF_Paracon com as atribuições das variáveis de entradas (VAR_INPUT) dos graus de crença e descrença denominadas de u1 e u2 e as respectivas variáveis de ajuste Vscc, Vicc, Vscct e Vicct.

Como variáveis de saída (VAR_OUTPUT), são utilizadas as variáveis inteiras S2a e S2b, e as variáveis booleanas que indicam os doze (12) estados do quadrado unitário do plano cartesiano.

As variáveis Gct, Gc, modulo_Gc e modulo_Gct , são variáveis locais presentes apenas em tempo de execução da função.

Com a determinação das variáveis de entrada e saída do bloco, e com a programação da função PARACON devidamente inserida, a POU do tipo Bloco Funcional denominado BF_Paracon esta pronta para ser utilizada em qualquer tipo de programa.

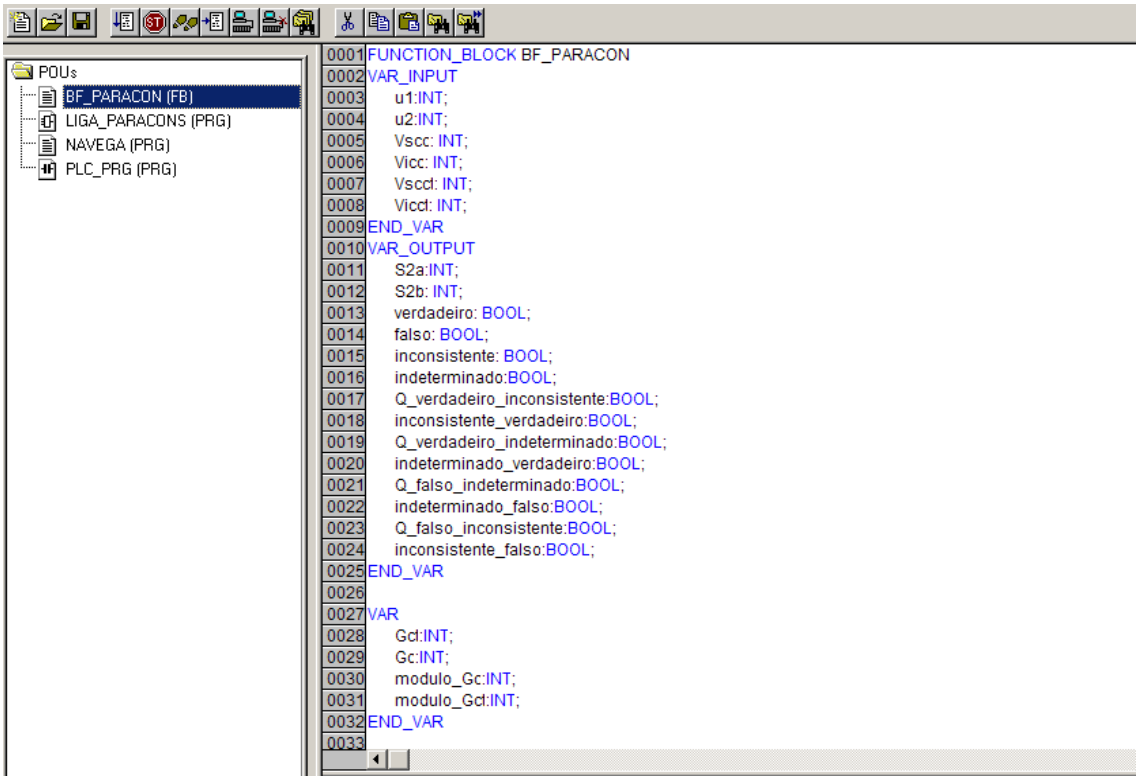


Figura 35. Determinação de entradas e saídas do BF_PARACON

Na figura 36, é possível visualizar o bloco completo, cuja construção seguiu a norma IEC 61131 em seus itens 3 e 7 e, que pode ser aplicado na automação de plantas industriais que trabalhem com Processos Contínuos ou em Batelada.

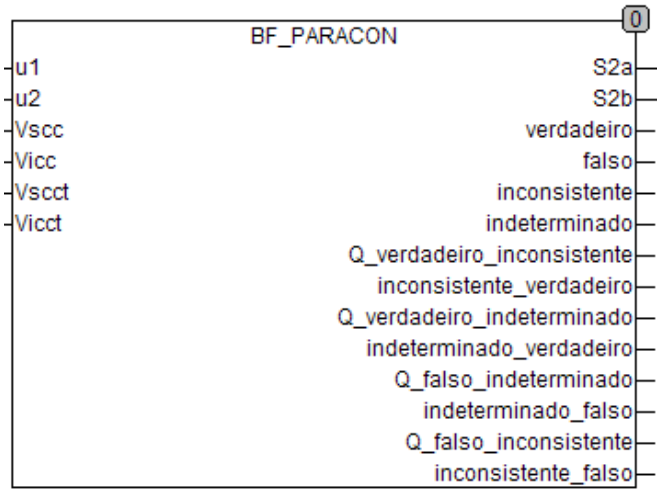


Figura 36. Bloco Funcional BF_Paracon

5 PROJETO DE UM PROCESSO DE AUTOMAÇÃO PARA CONTROLE PARACONSISTENTE DE TEMPERATURA

Neste capítulo será apresentada uma aplicação da Lógica Paraconsistente, na forma de seu algoritmo Para-Analisador programado no Bloco BF_Paracon, em um projeto de malha fechada para controle de temperatura.

A planta de temperatura utilizada neste trabalho de pesquisa foi adequada às características de um sistema de controle em malha fechada. Neste tipo de sistema, toda tomada de decisão e, consequentemente a ação de controle, vão depender diretamente de um sinal monitorado por um elemento sensor na planta do processo.

O sinal proveniente da diferença entre o sinal de saída e um sinal de referência, também conhecido no meio industrial como *Set Point* (SP), é denominado como sinal de erro (OGATA, K 1990).

O conceito de malha fechada é definida através de uma realimentação originada do sensoriamento entre o sinal de saída e a sua devida comparação com o valor de referência (SP). Na figura 37 pode ser observado um modelo deste tipo de sistema.

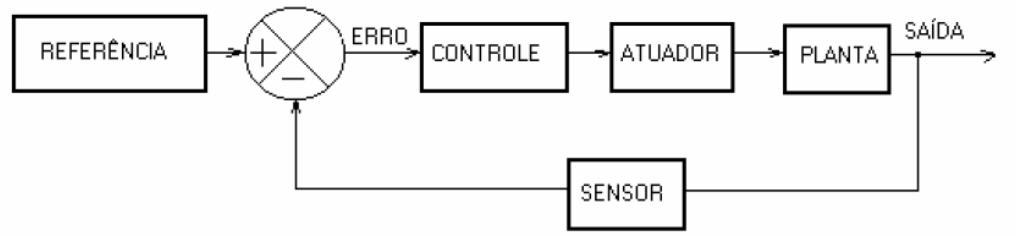


Figura 37. Sistema de controle em malha fechada

Fonte: OGATA, K 1990

Entre as principais vantagens na utilização dos sistemas em malha fechada podem ser citadas:

- Aumento de precisão.
- Menor sensibilidade a ruídos devido a compensação efetuada pelo sistema aos possíveis desvios originados.
- Aumento da largura de faixa de resposta de frequências de entrada na qual o sistema responde satisfatoriamente.

Como desvantagem, o sistema apresenta tendências a instabilidades ou oscilações. Na figura 38 esta sendo apresentado o diagrama de blocos da planta de temperatura utilizada como teste na aplicação de controle do Bloco Funcional BF_Paracon.

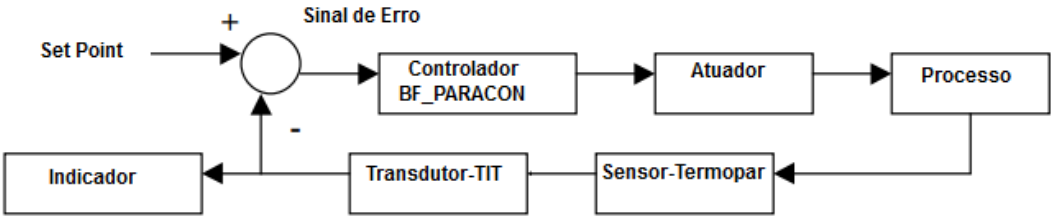


Figura 38. Diagrama de blocos da planta de temperatura utilizada na aplicação do bloco BF_Paracon

Na sequência deste trabalho serão apresentados os equipamentos pertencentes à planta de controle de temperatura utilizada como aplicação do bloco BF_Paracon construído.

5.1 Elemento Sensor - Termopar

Em 1821, o físico alemão J. T. SEEBECK (Thomas D, 1988) através de experiências demonstrou a teoria que num circuito fechado, formado por dois fios de metais diferentes, ao serem inseridos os dois pontos de junção à temperaturas diferentes, é gerada uma corrente elétrica cuja intensidade depende da natureza dos dois metais utilizados e da diferença de temperatura existente entre as duas junções. Este processo conhecido como efeito Seebeck fundamenta os conceitos de termopares

Atualmente os termopares são os sensores mais utilizados na medição de temperatura em ambientes indústrias e, devido a este fato, foi o sensor utilizado neste trabalho de pesquisa. Sua instalação é baseada na ligação da junção de medida ao ponto de medição. O outro extremo do termopar é aberto e normalmente conectado a um transmissor de temperatura. Este ponto é chamado junção de referência ou junta fria, como pode ser observado na figura 39.

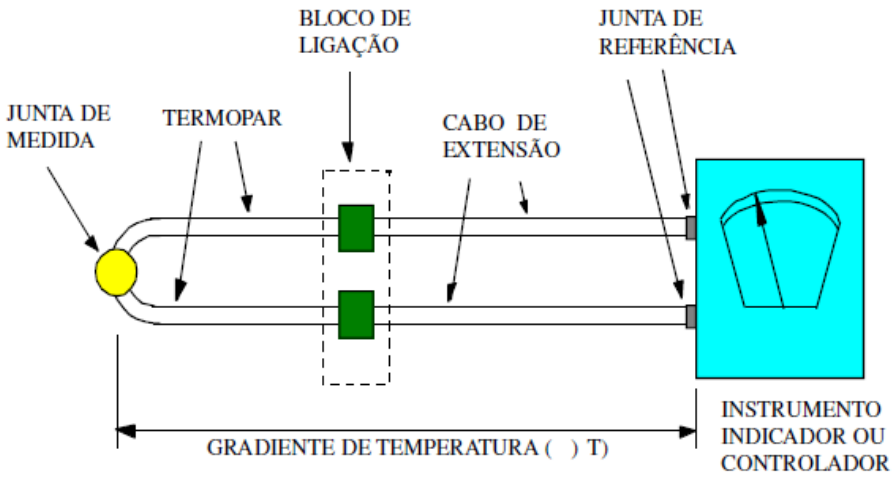


Figura 39. Ligação de termopar
Fonte: Catalogo ICEL

Os cabos de extensão que são mostrados na figura 26 são condutores formados com as mesmas ligas dos termopares a que se destinam, apresentando a mesma curva de milivoltagem por temperatura. Estes fios são utilizados para ligação do termopar até os terminais do transmissor, onde é medida a temperatura da junta de referência sem ocasionar erros de leitura. Uma tabela de composição destes materiais com as cores de padronização dependendo da norma utilizada pode ser verificada na tabela 3.

Tabela 3 Tipos de Termopar com seus cabos de extensão e cores.

Tipo de Termopar	Tipo do Cabo	Material dos Condutores		Norma Americana ANSI MC 96.1			Norma Alemã DIN 43714			Norma Japonesa JISC 1610/81		
		+	-	cabo	+	-	cabo	+	-	cabo	+	-
T	extensão	cobre	cobre-níquel	azul	azul	vermelho	marrom	vermelho	marrom	marrom	vermelho	branco
J	extensão	ferro	cobre-níquel	preto	branco	vermelho	azul	vermelho	azul	amarelo	vermelho	branco
E	extensão	níquel - cromo	cobre-níquel	roxo	roxo	vermelho	preto	vermelho	preto	roxo	vermelho	branco
K	extensão	níquel-cromo	níquel-alumínio	amarelo	amarelo	vermelho	verde	vermelho	verde	azul	vermelho	branco
K	compensação	ferro	níquel-cobre	-	-	-	verde	vermelho	verde	azul	vermelho	branco
S	compensação	cobre	cobre-níquel	verde	preto	vermelho	branco	vermelho	branco	preto	vermelho	branco
R	compensação	cobre	cobre-níquel	verde	preto	vermelho	branco	vermelho	branco	preto	vermelho	branco
B	cabo comum	cobre	cobre	cinza	cinza	vermelho	cinza	vermelho	cinza	cinza	vermelho	branco
N	extensão	níquel-cromo-silício	níquel-silício	laranja	laranja	vermelho	-	-	-	-	-	-

Fonte: Catalogo ICEL

Um termopar, portanto, gera uma tensão que varia em função da diferença de temperatura entre a junção de medição e à junção de junta fria. Atualmente existem vários tipos de termopares disponíveis no mercado, cuja aplicação depende da faixa de trabalho e característica do local de medição. Na figura 40 é possível ser observado um gráfico das curvas características de diversos termopares.

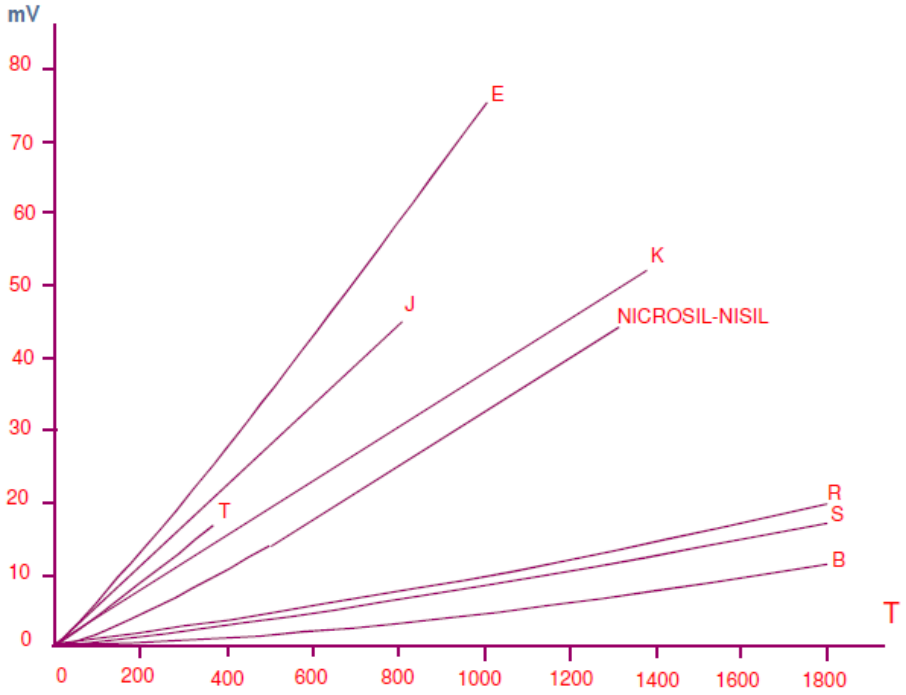


Figura 40. Curva característica de termopares
Fonte: Catalogo ICEL

Para este trabalho foi escolhido o termopar tipo “J” do fabricante ECIL[2] devido as suas características, dentre as quais é possível citar:

a- Baixo custo devido a sua construção Ferro-Constantan (liga de Cobre-Niquel);

b- Não indicado em utilização em temperaturas abaixo de 0ºC principalmente quando houver condições propicias a condensação, o que pode provocar a oxidação do ferro e consequentemente sua fratura.

c- Temperaturas de até 750ºC com milivoltagem em torno de 50mV.

d- Termopar juntamente com o tipo K mais utilizado na indústria.

[2] ECIL empresa fabricante de sensores de temperatura e equipamentos e projetos na subdivisão de energia. Localizada em São Paulo-SP.

Outro ponto importante a ser observado na utilização do termopar como elemento sensor, é que, para ser encontrada a temperatura medida, a temperatura na junção de referência deve ser adicionada à temperatura da junta fria. Isto é chamado de compensação de junta fria. Esta compensação, denominada de compensação automática da junta de referência ou temperatura ambiente, é realizada por instrumentos como, por exemplo, o Transmissor de temperatura descrito a seguir.

5.2 Transmissor de Temperatura

Neste trabalho, foi utilizado como elemento responsável pelo recebimento dos sinais provenientes dos sensores bem como o respectivo envio destes sinais ao elemento de controle, o Transmissor Indicador de Temperatura TIT301 da marca SMAR, que realiza automaticamente a compensação automática da junta fria.

Na memória do transmissor TIT301, estão inseridas as tabelas dos termopares padrões mais utilizados em ambientes industriais (termopares B, E, J, K, N, R, S e T). Na figura 41, pode ser observada uma foto deste equipamento.

Figura 41. TIT301 SMAR[3]
Fonte: Catalogo SMAR

O transmissor de temperatura TIT301 recebe em sua entrada o sinal de milivoltagem enviado pelo termopar e fornece em sua saída um sinal de 4 à 20mA que será utilizado pelo controlador. Na tabela 4 podem ser observadas as características do equipamento indicadas pelo fabricante.

[3] SMAR Empresa de Automação, fabricante de sensores, transmissores e equipamentos para redes industriais.

Tabela 4 Características do TIT301 da SMAR

Sinal de Saída	4-20 mA dc a dois fios, de acordo com a NAMUR NE43, com comunicação digital sobreposta (Protocolo HART).
Alimentação	12-45 Vdc
Entrada	Uma única unidade aceita sinais de: o Termopares, RTD's e RTD's Diferenciais o Sinais de mV de pirômetros de radiação, células de cargar, etc o Sinais Ohm de indicadores de posição, etc
Exatidão	±0.02 % exatidão básica
Configuração	Via comunicação HART (PC ou programador de mão) ou via ajuste local.
Calibração	Com referência e sem referência.
Proteção de Acesso	Sim, via senhas e via hardware.
Diagnósticos	Sim, via LCD.
Bloco de Controle	Sim, com PID incorporado.
Bloco Gerador de Setpoint	Sim, ideal para processos em batelada.
Indicador	Indicador de cristal líquido de 4½-dígitos numéricos e 5 caracteres alfanuméricos.Permite montagem em várias posições facilitando a visualização.
Limites de Temperatura	Ambiente: -40 a 85 ºC (-40 a 185 ºF)
Limites de Umidade	10 a 100% RH
Certificação em Área Classificada	À prova de explosão, à prova de tempo e intrinsecamente seguro, padrão CENELEC, FM e CEPEL.
Peso	Sem display e braçadeira de montagem: 0,80 kg Somar para o display: 0,13 kg Somar para a braçadeira de montagem: 0,60 kg.

Fonte: Catálogo SMAR

5.3 Elemento de Controle (CP)

Na malha de testes de controle de temperatura utilizada neste trabalho foi utilizado como elemento de controle do processo o CP da marca ALTUS DU351. Este controlador programável atende a norma 61131-3 que preconiza a reutilização de código e modularidade.

O CP da marca ALTUS DU351 apresenta condições para a utilização das cinco linguagens de programação, que são:

- Sequenciamento Gráfico de Funções (SFC);
- Lista de Instruções (IL);
- Diagrama *Ladder* (LD);

- Diagrama de Blocos (FBD);
- Texto Estruturado (ST).

Também permite uma linguagem extra, conhecida como:

- Gráfico Contínuo de Funções (CFC).

Uma foto do equipamento é mostrada na figura 42.

Figura 42. CP DU351 marca ALTUS©

Fonte: Catalogo ALTUS – CP DU351

Este controlador possui vinte (20) entradas digitais, sendo estas subdivididas em três (3) grupos de isolação, como pode ser observado na tabela 5. Quatro destas entradas apresentam a característica de “Entradas Rápidas” com tempo de resposta de 10µs comparativamente as entradas comuns, que apresentam um tempo de resposta de 500µs.

O CP DU 351 utilizado no projeto possui quatorze (14) saídas digitais a relê, subdivididas em dois grupos de isolação, que trata das características das saídas digitais deste controlador programável, como pode ser observado na tabela 6.

Tabela 5 Características das entradas digitais do CP DU351

	DU350, DU351
Número de entradas	20 entradas digitais divididas em 3 grupos de isolação: I00..I08 - 9 entradas – Grupo 0 I10..I18 - 9 entradas – Grupo 1 I20..I21 - 2 entradas – Grupo 2
Tensão de entrada	14 a 30 Vdc em relação ao comum para estado 1 0 a 5 Vdc em relação ao comum para estado 0
Corrente de entrada	5 mA (24 Vdc em relação ao comum) – Entradas comuns 15 mA (24 Vdc em relação ao comum) – Entradas rápidas
Tipo de entrada	"sink" tipo 1
Impedância de entrada	4,3 KΩ - Entradas comuns 1,5KΩ - Entradas rápidas
Isolação	2000 Vac por um minuto entre cada grupo de entrada 2000 Vac por um minuto entre grupo de entrada e circuito lógico
Configuração do borne	As entradas digitais estão divididas em 3 conectores (grupos de isolação) isolados entre si e isolados do circuito lógico. Cada conector é constituído de um borne para cada entrada e um borne para a referência de tensão. I00 a I08 – entrada 0 a 8 do grupo de isolação 0. I10 a I18 – entrada 0 a 8 do grupo de isolação 1. I20 a I21 – entrada 0 a 1 do grupo de isolação 2. C0 – comum do grupo de isolação 0. C1 – comum do grupo de isolação 1 C2 – comum do grupo de isolação 2 As entradas I00 a I02 e I10 a I12, são entradas rápidas, as entradas rápidas I00 a I02 pertencem ao Bloco 0 de entradas rápidas e as entradas rápidas I10 a I12 pertencem ao Bloco 1 de entradas rápidas. As entradas rápidas podem ser utilizadas como entradas comuns.
Tempo de resposta	0,5 ms – Entradas comuns 10 us - Entradas rápidas
Indicação de estado	Pode ser visualizado nas telas padrões do produto

Fonte: Catalogo ALTUS – CP DU351

Tabela 6 Características das saídas digitais do CP DU351

	DU351
Número de saídas	14 saídas digitais a relé divididas em 2 grupos de isolação: Q02 a Q07 – 6 saídas – Grupo 0 Q10 a Q17 – 8 saídas – Grupo 1
Corrente máxima por ponto	1 A
Tipo de saída	Relé normalmente aberto
Carga mínima	5 mA
Vida útil esperada	$10x10^4$ operações com carga nominal
Tempo máximo de comutação	10 ms
Freqüência máxima de chaveamento	0,5 Hz máximo com carga nominal
Indicação de estado	Pode ser visualizado nas telas padrões do produto
Tensão máxima(C6,C8)	30 Vdc grupo de isolação 0 30 Vdc grupo de isolação 1 240 Vac grupo de isolação 1
Isolação	2000 Vac por um minuto entre cada grupo de saída 2000 Vac por um minuto entre grupo de saída e circuito lógico
Resistência de contato	< 250 mΩ
Configuração do borne	As saídas digitais a relé estão divididas em 2 conectores(grupos de isolação). Cada conector é constituído de um borne para cada saída, um borne para o contato comum a todos os relés do mesmo conector e um borne de 0V (somente utilizado em saídas a transistor). Q02 a Q07 – saída a relé 2 a 7 do grupo de isolação 0 Q10 a Q17 – saída a relé 0 a 7 do grupo de isolação 1 C5 – não utilizado para as saídas a relé C6 – comum de todos relés do grupo de isolação 0, e utilizado para alimentar as saídas rápidas. Na opção de saída tipo sink (0 Vdc no pino C6) as saídas Q00 e Q01 não poderão ser utilizadas. Os relés do grupo de isolação 0 não poderão acionar cargas AC A utilizações de tensão alternada no pino C6 causará danos irreversíveis ao produto C7 – Pino não utilizado para as saídas a relé C8 – Pino ligado ao comum de todos relés do grupo de isolação 1

Fonte: Catalogo ALTUS – CP DU351

As Interfaces de entradas analógicas permitem que o CP possa receber e manipular os sinais analógicos que, neste trabalho, serão enviados pelo transmissor de temperatura TIT301 com sinais de corrente de 0 a 20 mA , 4 a 20 mA e tensão de 0 à 10 VCC, como pode ser observado na tabela 7.

Tanto a conversão das entradas, como a conversão das saídas, são realizadas pelo CP com uma resolução de doze (12) bits.

Na tabela 8 estão as principais características das saídas analógicas do CP DU351 fornecidas pelo fabricante.

Tabela 7 Características das entradas analógicas do CP DU351

	DU350, DU351
Número de entradas	4 entradas analógicas não isoladas do circuito lógico
Tipo de entrada	Tensão: 0 a 10 Vdc Corrente: 0 a 20 mA, 4 a 20 mA
Resolução do conversor	12 bits
Configuração do borne	AV0 – entrada de tensão canal 0 AI0 – entrada de corrente canal 0 C9 – comum para entradas 0 e 1 AV1 – entrada de tensão canal 1 AI1 – entrada de corrente canal 1 AV2 – entrada de tensão canal 2 AI2 – entrada de corrente canal 2 C10 – comum para entradas 2 e 3 AV3 – entrada de tensão canal 3 AI3 – entrada de corrente canal 3
Parâmetros configuráveis	Tipo da entradas para cada ponto, tensão ou corrente Fundo de escala para cada canal, máximo 30000 Filtro de primeira ordem com constantes de tempo pré-definidas
Proteções	Diodo TVS em todas as entradas analógicas
Tempo de atualização	60 ms

Fonte: Catalogo ALTUS – CP DU351

Tabela 8 Características das saídas analógicas do CP DU351

	DU350, DU351
Número de saídas	2 saídas analógicas não isoladas do circuito lógico
Tipo de saída	-Tensão: 0 a 10 Vdc -Corrente: 0 a 20 mA
Resolução do conversor	12 bits
Configuração do borne	C3 – comum para a saída AO0. AO0 – saída analógica 0. (Configurável por software como tensão ou corrente), C4 – comum para a saída AO1 AO1 – saída analógica 1 (Configurável por software como tensão ou corrente)
Proteções	Diodo TVS em todas as saídas analógicas.
Parâmetros configuráveis	Tipo de sinal em cada canal (tensão ou corrente) Fundo de escala para cada canal, máximo 30000

Fonte: Catalogo ALTUS – CP DU351

A saída utilizada no projeto, que será enviada ao elemento final de controle, terá uma corrente com valores variando de 0 a 20 mA.

O DU351 possui ainda uma IHM que suporta textos e gráficos, além de um teclado alfanumérico com sete (7) teclas de função.

5.4 Elemento Final de Controle (Conversor I/P)

Na planta de controle de temperatura utilizada nos testes foi utilizado o conversor de potência TH 6200A da marca THERMA como elemento final de controle.

De modo convencional este conversor (I/P) atua no controle da potência através de ajustes na intensidade de corrente elétrica aplicada a um banco de resistências. Uma foto do conversor (I/P) utilizado na simulação pode ser observada na figura 43.

Figura 43. TH 6200A da THERMA[4]
Fonte: Catalogo THERMA – TH 6200A

No controle lógico paraconsistente aplicado a malha de teste o conversor (I/P) TH 6200A recebe um sinal de 0 a 20mA proveniente do CP DU351.

No funcionamento do controle este sinal recebido pelo TH 6200A é um valor dependente do resultado da análise discreta efetuada pelo Bloco Funcional BF_Paracon, que foi construído baseado no algoritmo do *Para-Analisador* da LPA2V. De acordo com o sinal de corrente obtido na analise paraconsistente e

[4] THERMA - Instrumentos de Medição Automação e Projetos Ltda. Fabricante de conversores de potência. Localizada em São Paulo-SP

aplicado em sua entrada, o conversor (I/P) executa uma ação que transforma o valor resultante em um percentual de potência.

A tabela 9 apresenta uma exemplificação desta ação que faz a correspondência entre os valores percentuais envolvidos nos testes.

Tabela 9 Exemplo de correspondência Corrente/Potência

0 mA	0%
5 mA	25%
10 mA	50%
15 mA	75%
20 mA	100%

O conversor TH 6200A pode ser operado com diversas faixas de atuação, tais como sinais de controle ajustados nas faixas de 0 a 20mA, de 4 a 20mA, de 0 a 10V, de 0 a 5V, de 1 a 5V, etc.

Outra característica importante é que o conversor TH 6200A permite fazer uma conversão de grandezas do tipo Corrente/frequencia (I/f). Portanto, este equipamento pode ser considerado um conversor de I/F.

Neste projeto o conversor TH 6200A foi ajustado para receber um sinal de controle de 0 a 20 mA como sinal de entrada e, através de um comando eletrônico de disparo irá transformar esta variação de corrente em uma variação de frequência. Com a variação de frequência obtida através do comando eletrônico, é feito o controle do disparo do módulo de tiristores que, neste modelo, é constituído por tiristores (SCRs) (diodo controlado de silício).

Ainda a respeito deste módulo de potencia, as principais características construtivas e elétricas são:

- O módulo de tiristores possui dissipadores de alumínio o que mantem a temperatura dos SCRs abaixo da máxima permitida.
- O conjunto possui um termostato de temperatura de proteção que desativa o conversor se a temperatura dos dissipadores for excedida.
- A Tensão de rede; 110V, 220V, 380V, 440V, 460V e 480V.

O Módulo de potencia trabalha com trem de impulsos onde o conversor controla a potência entregue a carga (banco de resistências elétricas) através do

envio de uma onda senoidal com intervalos. O formato das ondas e intervalos, com o correspondente valor de intensidade de corrente, podem ser observados na figura 44.

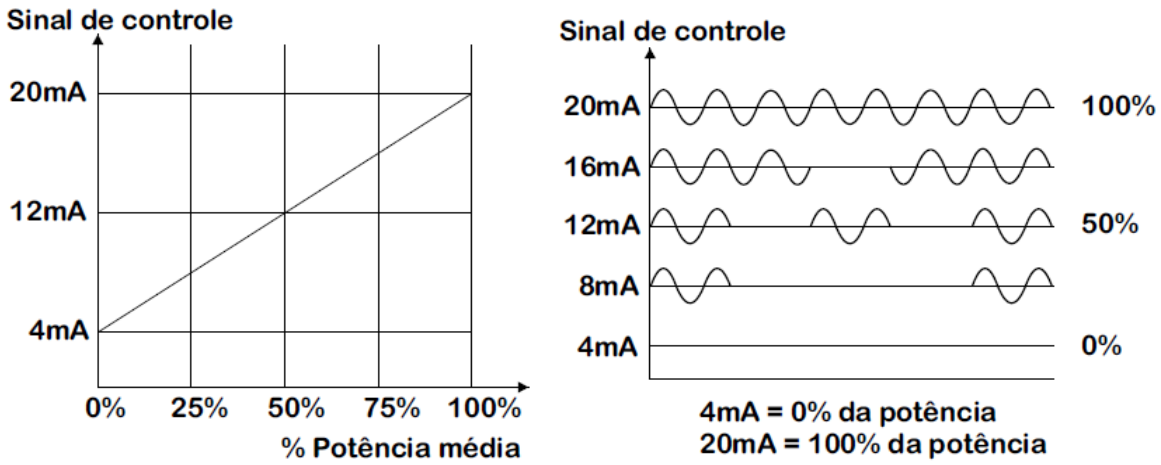


Figura 44. Modulação de potência para um sinal de controle de 4 à 20ma

Fonte: Catalogo THERMA TH 6200A

Segundo o fabricante, o conversor TH 6200A apresenta um fator de potência de aproximadamente “1”.

Na figura 45 é demonstrado o esquema de ligação do conversor ao banco de resistências utilizado na planta de controle de temperatura preparada para os testes do BF_Paracon.

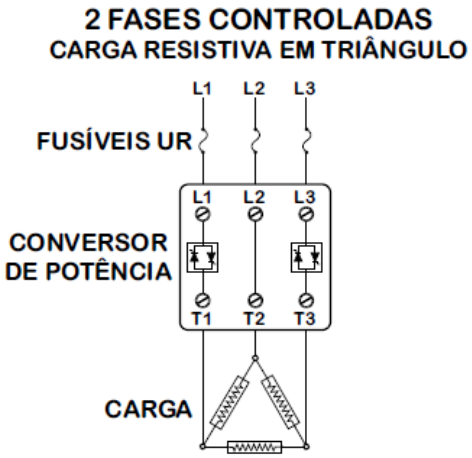


Figura 45 Ligação do TH 6200A ao banco de resistências elétricas

Fonte: Catalogo THERMA TH 6200ª

5.5 Sistema de Controle com o Bloco Funcional (BF_Paracon)

Para verificação de funcionamento e testes de aplicação do bloco BF_Paracon foi montada uma planta piloto de controle de temperatura aplicada em um tanque de ar comprimido que faz parte de um processo de secamento de ar em um sistema pneumático. Portanto, o controle da variável de processo (VP) será realizado no tanque onde estão inseridas as resistências elétricas, componentes do banco controlado pelo elemento final da malha.

A figura 46 apresenta um diagrama em blocos resumido da planta de controle de temperatura na qual está sendo feita a aplicação do bloco BF_Paracon.

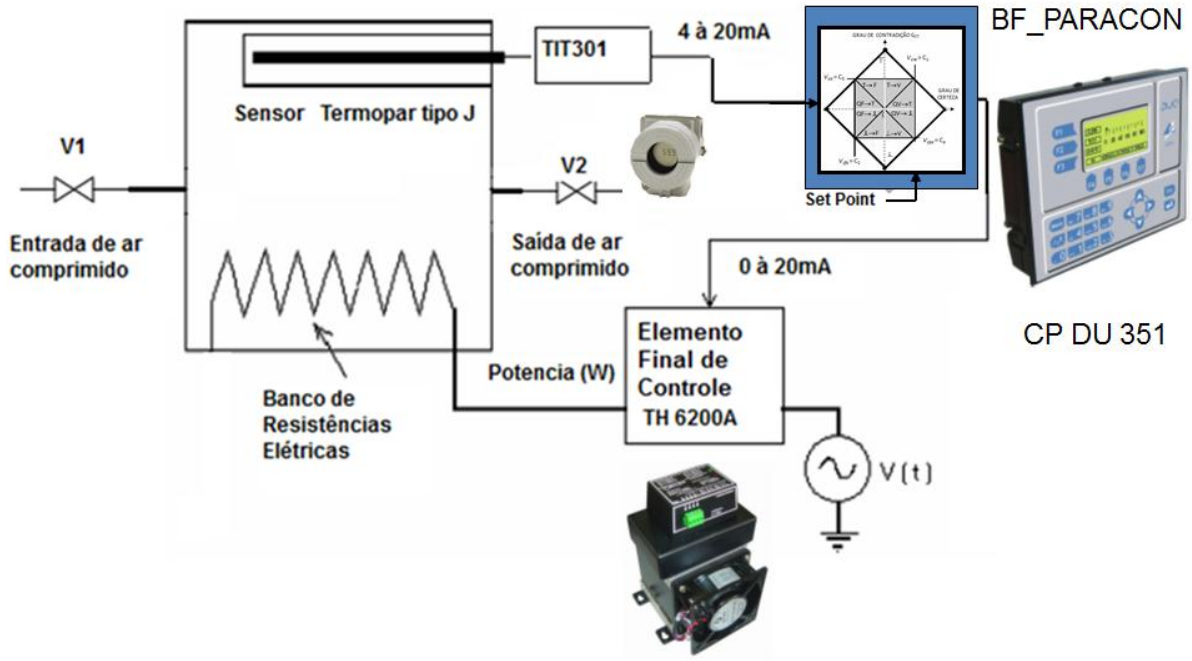


Figura 46. Planta de Temperatura

Com a planta de temperatura baseada no controle em malha fechada, o CP DU351 é o elemento de controle do sistema. Este controle é efetuado pelo Bloco Funcional BF_Paracon construído segundo os padrões estabelecidos pela norma 61131-3 e, que teve sua construção baseada no algoritmo Para-Analisador pertencente a lógica não-clássica denominada de Lógica Paraconsistente Anotada com Anotação de dois Valores (LPA2V).

A figura 47 apresenta uma foto da planta onde foram feitos os testes de funcionalidade do bloco BF_Paracon.

Figura 47 Planta da malha de controle de temperatura utilizada para testes.

5.5.1 Modelagem de sinais e descrição geral da Planta piloto para teste

A planta de controle de temperatura utilizada no projeto trabalha em um range de 0 ºC a 100ºC. Nela, um elemento sensor Termopar do tipo J (marca ICEL), esta inserido em um tanque cilíndrico de ar comprimido que possui uma válvula de entrada de ar V1 e uma válvula de saída V2 conforme diagrama da figura 46. Dentro do mesmo tanque encontra-se instalado um banco de resistências elétricas.

O termopar efetua a medição da temperatura interna do tanque, fornecendo uma saída em milivoltagem (mV), que é recebida pelo Transmissor Indicador de Temperatura TIT301 do fabricante SMAR. Este transmissor fornece em sua saída um valor em corrente cujo range varia entre 4 e 20mA, que é estabelecido em função do valor recebido em mV em sua entrada.

O sinal de corrente de saída enviado pelo TIT301 é coletado por uma das entradas analógicas do CP DU351 da marca ALTUS. Este sinal é então convertido para um número inteiro entre 0 e 10000 para que possa ser analisado e manipulado pelo bloco BF_Paracon.

A tabela 10 apresenta um exemplo da conversão efetuada.

.

Tabela 10. Conversão da Entrada Analógica do CP de Corrente para um número Inteiro

4 mA	0
6 mA	1250
8 mA	2500
10 mA	3750
12 mA	5000
14 mA	6250
16 mA	7500
18 mA	8750
20 mA	10000

Na configuração da entrada analógica é necessário fazer seleção do padrão de funcionamento do BF_Paracon realizando os seguintes ajustes:

- Tipo do Canal:

  Selecionar a escala desejada. No caso o valor escolhido foi de 4 a 20 mA.

- Filtro:

  Tempo para o Filtro. No caso foi selecionado o Filtro ‘0’ que corresponde \ intervalos de 2 *ms*.

- Fundo de Escala:

  Valor máximo desejado. No caso o valor ajustado foi 10000.

Após as configurações realizadas, os valores correspondentes aos recebidos pela entrada analógica selecionada serão entregues ao bloco previamente ajustado automaticamente no range de 0 à 10000.

Seguindo os conceitos de modularidade e reaproveitamento de código da norma IEC61131-3 o bloco BF_Paracon foi inserido no programa principal PLC_PRG que, por sua vez, já havia sido previamente configurado para trabalhar em linguagem *Ladder*.

Na figura 48 é mostrada uma tela onde o programa PLC_PRG recebe o bloco BF_Paracon. Nessa fase, de acordo com a programação, somente é necessário definir as memórias e os pinos que estarão ligados às entradas e saídas do bloco, como observado na figura 49.

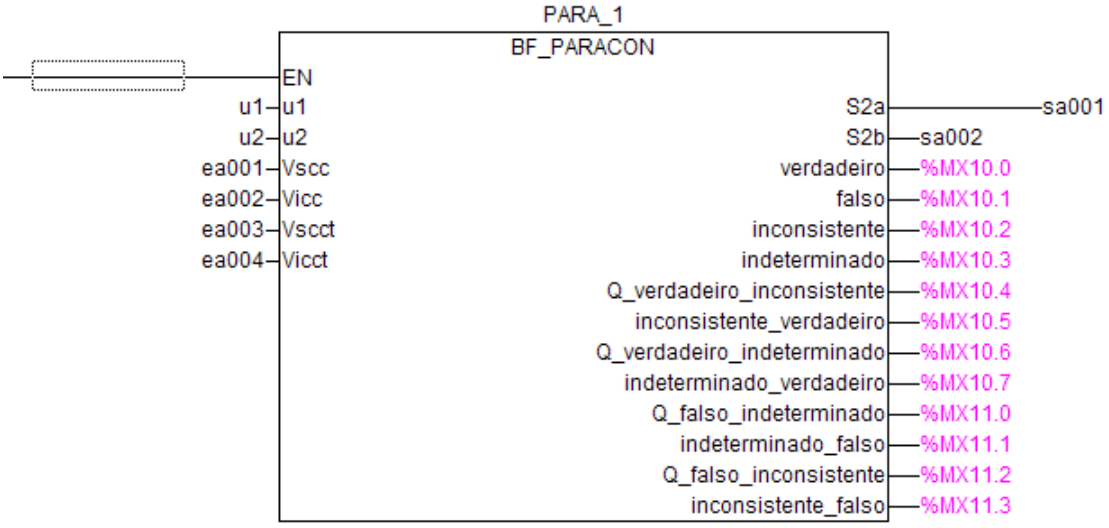


Figura 48. Programa PLC_PRG em Ladder com o BF_Paracon

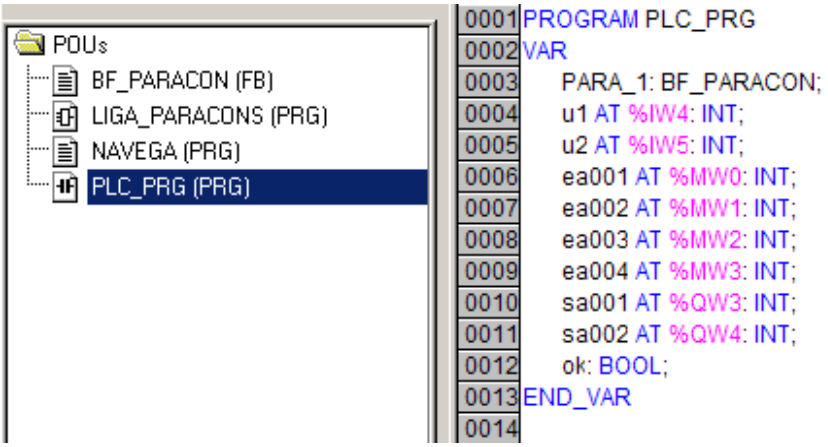


Figura 49. Determinação das ligações dos pinos de entrada e saída do BF_Paracon no programa PLC_PRG

5.5.2 Sistema de Controle com o Bloco Funcional (BF_Paracon)

O bloco BF_Paracon inserido no programa PLC_PRG efetua a análise dos sinais de entrada, fornecendo em sua saída um valor também em numero inteiro dentro do mesmo range, ou seja, de 0 a 10000. Nesta análise, o algoritmo Para-Analisador presente no BF_Paracon, traduz a análise paraconsistente através das informações recebidas na forma de grau de evidência favorável (crença) e evidência desfavorável (descrença), resultando nos valores de grau de certeza (Gc) e contradição (Gct), pelos quais o Controlador seleciona como saída um dos estados lógicos entre os 12 do reticulado.

Com este estado encontrado, é necessária definir a atuação do CP DU351 no elemento final de controle e, para isso, foi seguida uma tabela em porcentagem de potência que o Conversor TH 6200A deve entregar ao banco de resistências elétricas utilizados no experimento, como pode ser verificado na tabela 11.

A tabela 11 foi construída tomando como procedimento base a condição de que; quanto mais o estado se aproxima do “Verdadeiro” no QUPC, menor será a potência entregue a carga. Portanto, como forma de exemplificação, a condição verdadeira (V) entrega a carga um valor de manutenção de 5% e a condição falsa (F) por sua vez, deve fornecer a carga um valor de 95% de potência máxima. Este valor entregue a carga é devido a diferença de temperatura identificada entre a Referência (SP), representada como Grau de Evidência desfavorável, e a Variável de Processo (VP), representada como Grau de Evidência favorável.

Tabela 11. Relação entre a Área de sintonia do Reticulado e a Potência entregue ao banco de resistências

Verdadeiro	5%
Quase verdadeiro tendendo a Inconsistente	22%
Quase verdadeiro tendendo a Indeterminado	22%
Inconsistente tendendo a Verdadeiro	45%
Indeterminado tendendo a Verdadeiro	45%
Inconsistente tendendo a Falso	75%
Indeterminado tendendo a Falso	75%
Quase Falso tendendo a Inconsistente	90%
Quase Falso tendendo a Indeterminado	90%
Falso	95%
Inconsistente	50%
Indeterminado	50%

A partir dos valores de potência entregue ao banco de resistências, são gerados os valores correspondentes em números inteiros, que, por sua vez são entregues a saída analógica do CP DU351, como pode ser visto na tabela 12.

Tabela 12. Relação entre a Área do Reticulado, a Potência entregue ao banco de resistências e o número inteiro entregue pelo bloco a saída analógica do CP DU351

Verdadeiro	5%	500
Quase verdadeiro tendendo a Inconsistente	22%	2200
Quase verdadeiro tendendo a Indeterminado	22%	2200
Inconsistente tendendo a Verdadeiro	45%	4500
Indeterminado tendendo a Verdadeiro	45%	4500
Inconsistente tendendo a Falso	75%	7500
Indeterminado tendendo a Falso	75%	7500
Quase Falso tendendo a Inconsistente	90%	9000
Quase Falso tendendo a Indeterminado	90%	9000
Falso	95%	9500
Inconsistente	50%	5000
Indeterminado	50%	5000

A saída analógica escolhida, tal qual o procedimento efetuado para configuração de sua entrada, teve o tipo de canal definido como 0 a 20mA e o seu fundo de escala com o valor de 10000. Após a configuração realizada, o próprio software se encarrega de realizar a conversão para a faixa de saída de 0 a 20mA.

5.5.3 Exemplo numérico do Bloco Funcional (BF_Paracon)

Como forma de exemplificação e apresentação dos procedimentos no tratamento de sinais com a Lógica Paraconsistente são apresentados dois exemplos numéricos do controle efetuado pelo bloco BF_Paracon, conforme pode ser observado a seguir.

Condições:

- O sinal μ1 esta entregando ao BF_Paracon o sinal coletado pela entrada analógica junto a Variável de Processo (VP).
- O valor μ2 se refere ao “*Set Point* “ (SP - valor desejado pelo processo).

Exemplo 1: μ1= 10000; μ2= 2000;

Da equação (1) determina-se o Grau de Contradição:

Gct=(u1+u2) -10000

Gct=(10000+2000) -10000

Gct= 2000

Da equação (2) determina-se o Grau de Certeza:

Gc=u1-u2

Gc= 10000 – 2000

Gc= 8000

De acordo com os valores obtidos de Gc e Gct é possível identificar qual a área pertencente ao reticulado da LPA2v corresponde ao estado lógico resultante. Portanto, utilizando os valores numéricos do exemplo o algoritmo interno do bloco BF_Paracon seleciona entre os 12 estados lógicos possíveis o estado “Verdadeiro”, como pode ser verificado na figura 50.

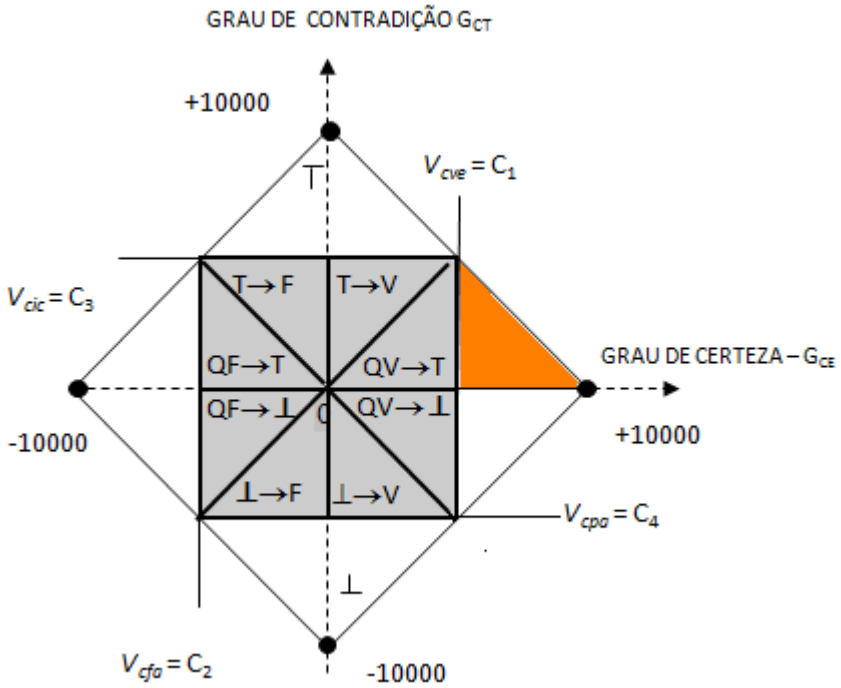


Figura 50. Determinação do estado verdadeiro referente ao Exemplo 1

Analisando o exemplo numérico na malha de controle verifica-se que o bloco BF_Paracon entrega à saída do CP o valor inteiro quinhentos (500) (tabela 12), que será convertido para 1mA, correspondente a 5% da faixa de 0 a 20 mA. Este valor de corrente será recebido pelo Conversor TH 6200A, que aplicará a sua saída uma potência de 170W, o que corresponde a 5% da faixa de 0 a 3400W.

Exemplo 2: µ1= 6000; µ2= 2000;

Da equação (1) determina-se o Grau de Contradição:

Gct=(u1+u2) -10000

Gct=(6000+2000) -10000

Gct= - 2000

Da equação (2) determina-se o Grau de Certeza:

Gc=u1-u2

Gc= 6000 – 2000

Gc= 4000

Através dos valores de Gc e Gct verifica-se que a posição da área do reticulado identificada pelo algoritmo corresponde ao estado lógico “Quase Verdadeiro tendendo a Indeterminado”, como pode ser verificado na figura 51.

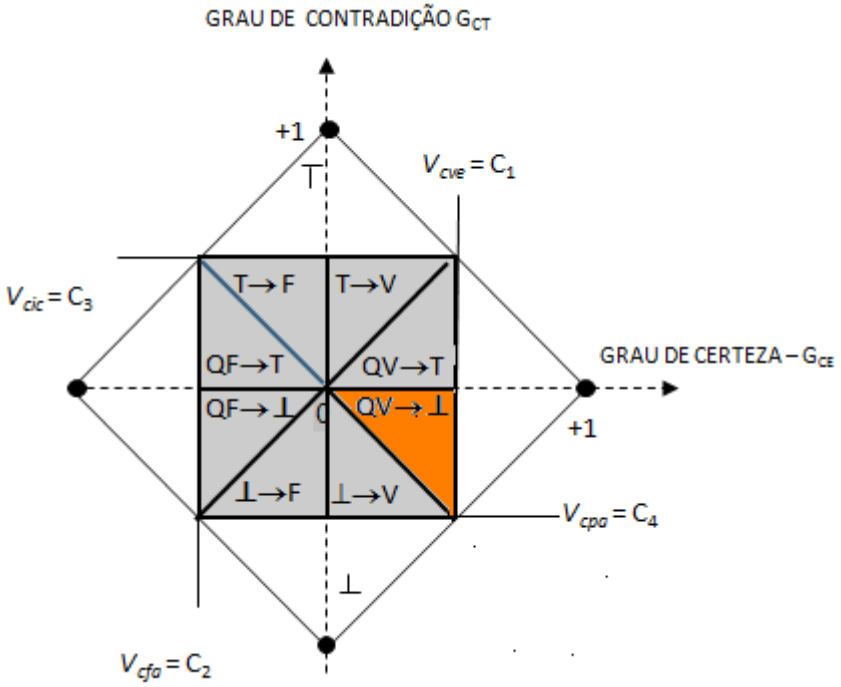


Figura 51. Determinação do estado Quase verdadeiro tendendo a Indeterminado referente ao Exemplo 2

Nestas condições o bloco BF_Paracon entrega à saída do CP o valor inteiro dois mil e duzentos (2200) (tabela 12), que será convertido para 4,4mA, correspondente a 22% da faixa de 0 a 20 mA. Este valor de corrente será recebido

pelo Conversor TH 6200A que aplicará a sua saída uma potência de 748W, também correspondente a 22% da faixa de 0 a 3400W.

Através dos exemplos citados é possível verificar que o sinal proveniente da análise Paraconsistente efetuado pelo bloco BF_Paracon é enviado em forma de número inteiro e posteriormente convertido em uma faixa de 0 a 20mA pela saída analógica do CP DU351.

Este sinal de corrente é entregue ao conversor de potência TH 6200A, que a partir do valor coletado em sua entrada efetua o controle de potência do banco de resistências elétricas instaladas no interior do tanque de ar comprimido através da modificação do ângulo de disparo de seu módulo de tiristores.

Diversos testes foram realizados com a planta da figura 46, onde a ação do Bloco Funcional BF_Paracon inserido no programa principal denominado PLC_PRG, realizou a análise dos sinais recebidos e, entregou de acordo com a tabela 12, uma resposta para a saída analógica do CP DU351. Esta saída foi ligada ao conversor TH 6200A atuando no controle de potência do banco de Resistências Elétricas. A seguir são apresentados os resultados obtidos no experimento.

5.6 Resultados Obtidos no Desenvolvimento

Com o objetivo de investigar o comportamento do Bloco Funcional, foi realizado um estudo comparativo com uma ação de um bloco PID, configurado no mesmo Controlador Programável (ALTUS DU351) e aplicado à mesma planta de controle de temperatura.

Neste trabalho foi utilizado o mesmo método apresentado em Barbuy et al (2001) e Barbuy et al (2003), seguindo os seguintes procedimentos baseados nas equações a seguir:

$$vc(t) = Kp[e(t) + \frac{1}{Ti}\int e(t)dt + Td\ \frac{de(t)}{dt}] \quad (3)$$

Sendo:

Kp é o ganho proporcional;

Ti *é* o tempo integral

Td *é* o tempo do derivativo

Sendo:

$$\frac{1}{Ti} = Ki\ (ISA) \tag{4}$$

PID na convenção de Parâmetros Independentes

$$vc(t) = Kp[e(t) + Ki_{ind} \int e(t)dt + Kd_{ind}\ \frac{de(t)}{dt}] \tag{5}$$

$$Kd_{ind} = Kp * Kd_{isa}\ ou\ Td_{isa} \tag{6}$$

$$Ki_{ind} = Kp * \frac{1}{Ti} \tag{7}$$

PI – ISA

$$PID\ com\ Td = Kd = 0 \tag{8}$$

$$vc(t) = Kp[e(t) + Ki \int e(t)dt] \tag{9}$$

Aplicando LAPLACE na equação (7)

$$Vc(s) = Kp[\ E(s) + Ki\ \frac{1}{s} + E(s)] \tag{10}$$

$$Vc(s) = Kp * \ E(s)[1 + \frac{Ki}{s}] \tag{11}$$

$$Vc(s) = Kp * \ E(s)[\frac{s\ +\ Ki}{s}] \tag{12}$$

Considerando:

$s = \ -ki$ => ”Zero” da Função de Transferência

$$Gc(s) = \frac{Vc(s)}{E(s)} = Kp\ \left(\frac{s\ + Ki}{s}\right) \tag{13}$$

Adotando:

$$Ki_{isa} = \frac{1}{\tau_{atuador}} = Polo\ do\ atuador \tag{14}$$

$$G(s)_{atuador} = \frac{Kp}{s\tau + 1} \tag{15}$$

$$G(s)_{atuador} = \frac{\frac{Kp}{\tau}}{s + \frac{1}{\tau}} \quad => sendo\ \frac{1}{\tau}\ o\ Polo\ do\ atuador \tag{16}$$

Para encontrar os parâmetros da malha de controle de temperatura inicialmente foram efetuados testes na planta para se determinar o tempo de resposta do sistema, ou seja, sua constante de tempo (τ_{atuador}) cujo valor foi utilizado para cálculo da constante Integral do Bloco PID, com o zero do controlador PI cancelando o pólo do atuador.

Na tomada de temperatura, as válvulas V1 e V2 foram fechadas, mantendo o ar no tanque confinado. Em seguida no intuito de se determinar a constante de tempo da planta, foi gerado um degrau na saída do controlador.

Com esta configuração, na figura 52 pode ser observado um valor de 30ºC, que foi utilizado como variável de processo, para um *Set Point* de 40ºC, valor requerido pelo sistema. Os TAGs “Tanque 2” e “*Set Point* 2”, que aparecem com marcação em amarelo, podem ser observados na tela do supervisório utilizado no controle do CP DU 351.

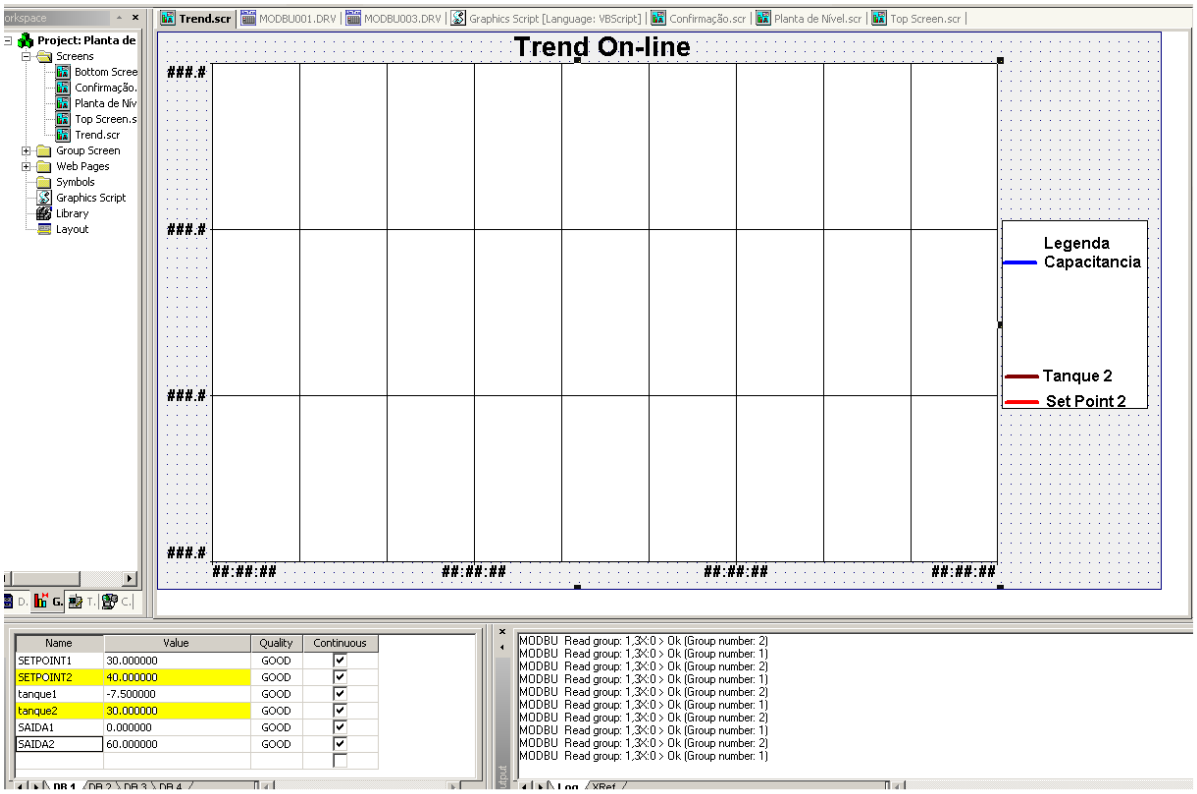


Figura 52. Estado inicial da planta de temperatura

A partir da diferença das temperaturas, em que 30ºC é reconhecida como a variável de processo, e sendo a temperatura interna do tanque de 40ºC denominado valor desejado de Set Point, foi possível se determinar o tempo de resposta (τ) da planta, conforme pode ser verificado na figura 53.

Através de um Span de 10ºC - diferença algébrica entre os valores superior e inferior da faixa de operação - o valor de 36,3ºC, equivalente a 63,3% da variação deste Span foi alcançado em um tempo de 1 minuto e 21 segundos, em cinco diferentes testes efetuados. Logo: aplicando a equação (14):

$$Ki_{isa} = \frac{1}{\tau_{atuador}} = \frac{1}{81} = 0{,}012$$

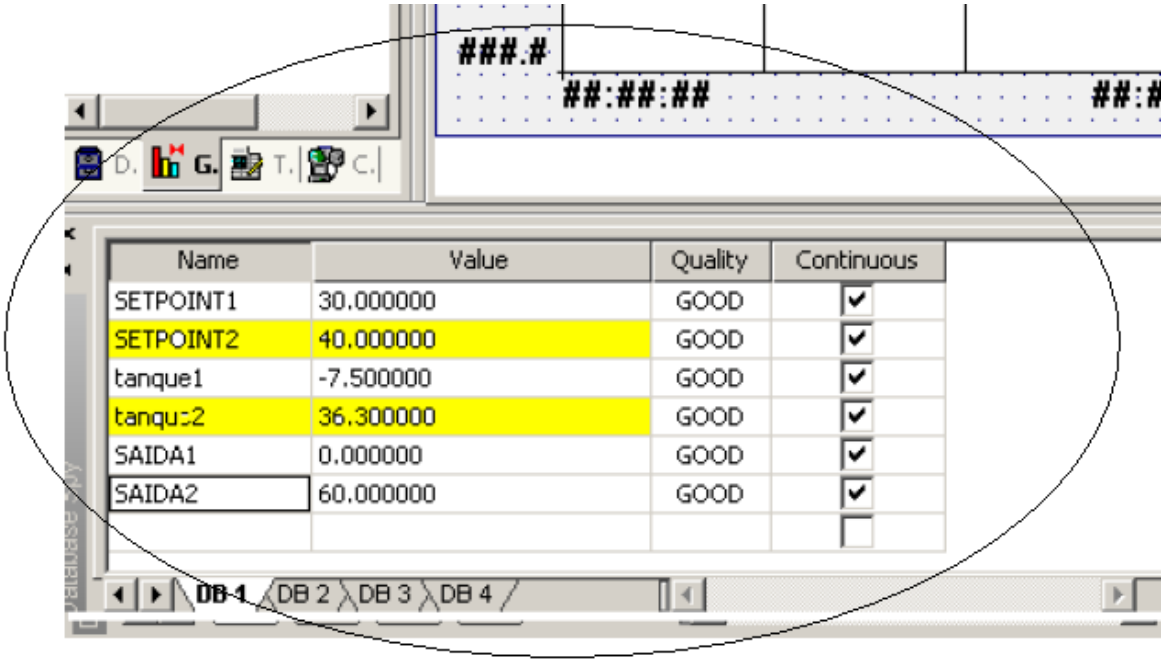


Figura 53. Determinação da constante de tempo da planta de controle de temperatura em estudo.

No procedimento da determinação do ganho Proporcional (*Kp*) foram efetuados oito (8) testes, aumentando-se gradativamente o valor da parcela da ação Proporcional até o valor máximo, sem a temperatura ultrapassar o valor desejado. Isto originou um *KP* correspondente ao amortecimento crítico, cujo valor encontrado foi *KP*= 0,8.

Com os valores de *Ki* e *Kp* definidos, estes foram inseridos no bloco PID do DU 351, sendo possível, então, ser verificada a resposta do sistema.

Foi verificado que, após ser transcorrido o equivalente a cinco constantes de tempo (5τ) a planta se estabilizou, mantendo a temperatura da Variável de Processo dentro do valor estabelecido pelo *Set Point,* ou seja em 40ºC, como pode ser observado na figura 54.

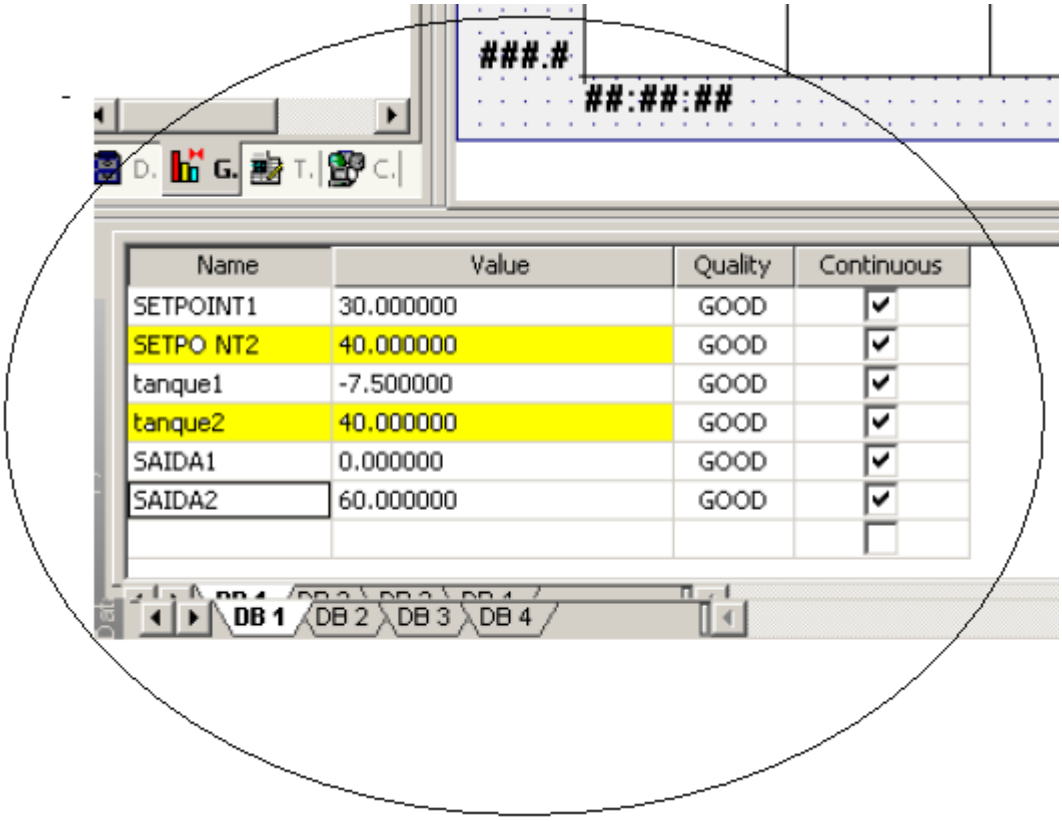


Figura 54. Estabilização do sistema (SP=VP)

5.7 Estudo comparativo comportamental entre os métodos convencional e Paraconsistente

Nesta pesquisa o estudo comparativo foi feito com o objetivo de investigar o comportamento do Bloco BF_Paracon, que apresenta saídas discretas, em relação a um procedimento de ação Proporcional e Integral de um bloco PID na mesma malha e em condições idênticas, ou mais próximas possível.

Como as características diferenciadas dos blocos não se permitem uma comparação de precisão de respostas, foi feita neste trabalho uma comparação comportamental frente aos dois tipos de controle do processo.

O gráfico da curva de resposta do Bloco PI do CP DU 351, com os respectivos valores de *Ki* e *Kp* utilizados como parâmetros, são apresentados na figura 55. Verifica-se que com o controle PID o sistema se mostrou estável a partir da atribuição dos valores dos ganhos proporcional e integral em todos os cinco (5) testes realizados.

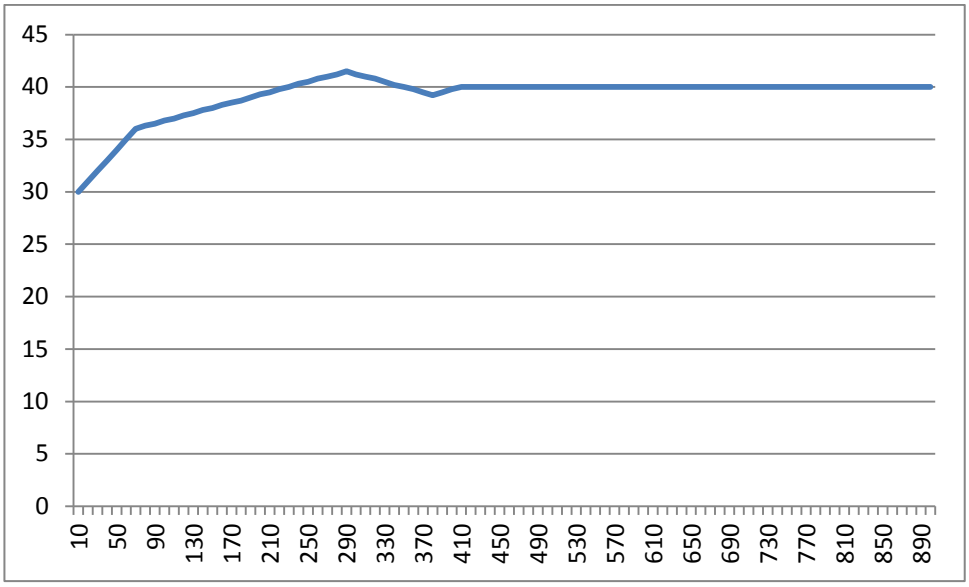


Figura 55. Resposta do bloco PID na planta de temperatura

O bloco BF_Paracon por sua vez, trabalha com doze níveis de potência para temperaturas diferentes, tendo em vista os doze estados pertencentes ao reticulado. Este método que utiliza a LPA2v apresenta, portanto um comportamento “Discreto” onde se espera que a temperatura se apresente com uma oscilação em torno do valor de *Set Point* desejado.

Foram efetuados testes onde, a partir de uma temperatura inicial de 30ºC, portanto o mesmo valor utilizado como exemplo do PID, o bloco construído a partir da lógica LPA2V, corrigiu o sistema como previsto, como pode ser observado no gráfico da figura 56.

Verifica-se que o gráfico apresenta a saída oscilando em torno dos 40ºC, que foi a temperatura definida como *Set Point*.

Da mesma forma como foram efetuados os testes no controlador PID, o teste com o bloco BF_Paracon foi repetido cinco (5) vezes, obtendo-se os mesmos valores referenciais, o que comprova a estabilidade do sistema mesmo nestas condições especiais de controle discreto.

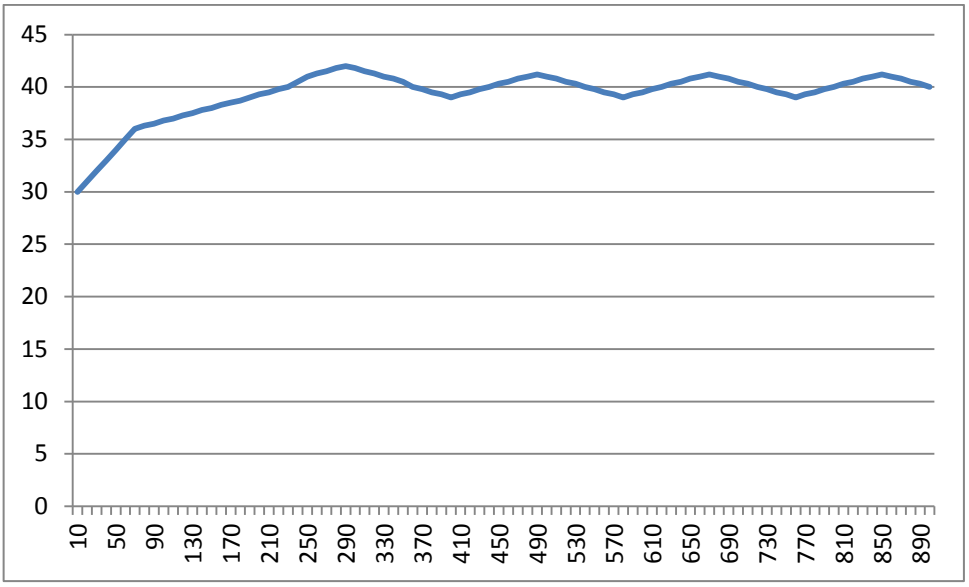


Figura 56. Resposta do bloco BF_PARACON na planta de temperatura (*Set Point* = 40ºC)

Utilizando novamente como variável de processo inicial a temperatura de 30ºC, para um *Set Point* de 50ºC, o bloco BF_Paracon manteve o mesmo comportamento, apresentando sua saída oscilando em torno do valor desejado, como pode ser verificado na figura 57.

No intuito de comprovar a estabilização do sistema foram repetidas cinco (5) vezes o mesmo teste, mantendo o mesmo padrão de resposta.

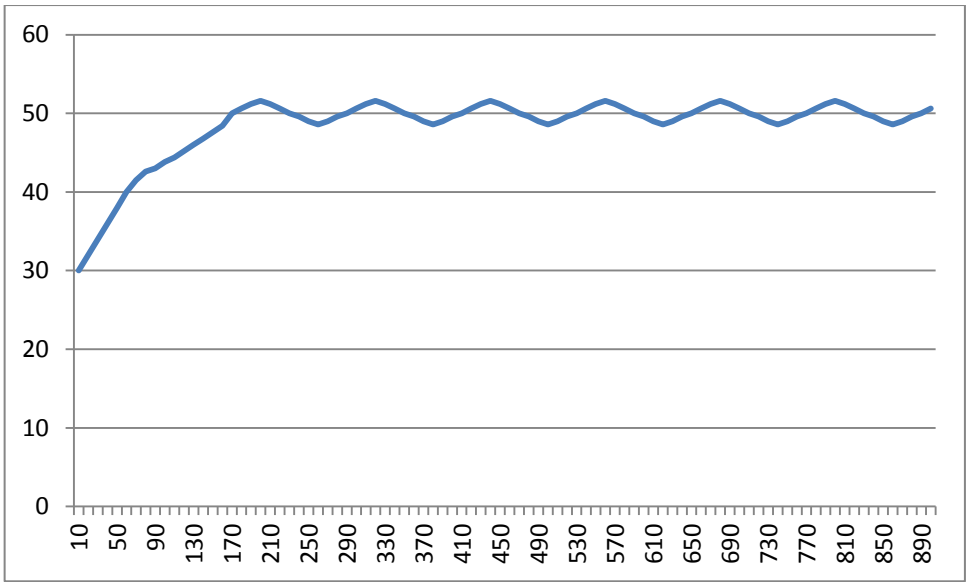


Figura 57. Resposta do bloco BF_Paracon na planta de temperatura (*Set Point* = 50ºC)

5.7 Discussões e futuros trabalhos

Este trabalho mostra que a adequação de Sistemas de Controle lógicos paraconsistentes às normas IEC 61131-3 oferece grandes possibilidades para que futuros estudos utilizando os fundamentos da lógica Paraconsistente sejam intensificados em aplicações voltadas à Automação Industrial no controle de processos contínuos e de manufatura. Como exemplo de aplicações para futuras pesquisas neste campo de aplicação voltada à LPA2v, é apresentado na figura 58, o desenvolvimento inicial de um programa denominado LIGA_PARACONS, cuja linguagem de programação definida é a de Gráfico Contínuo de Funções (CFC).

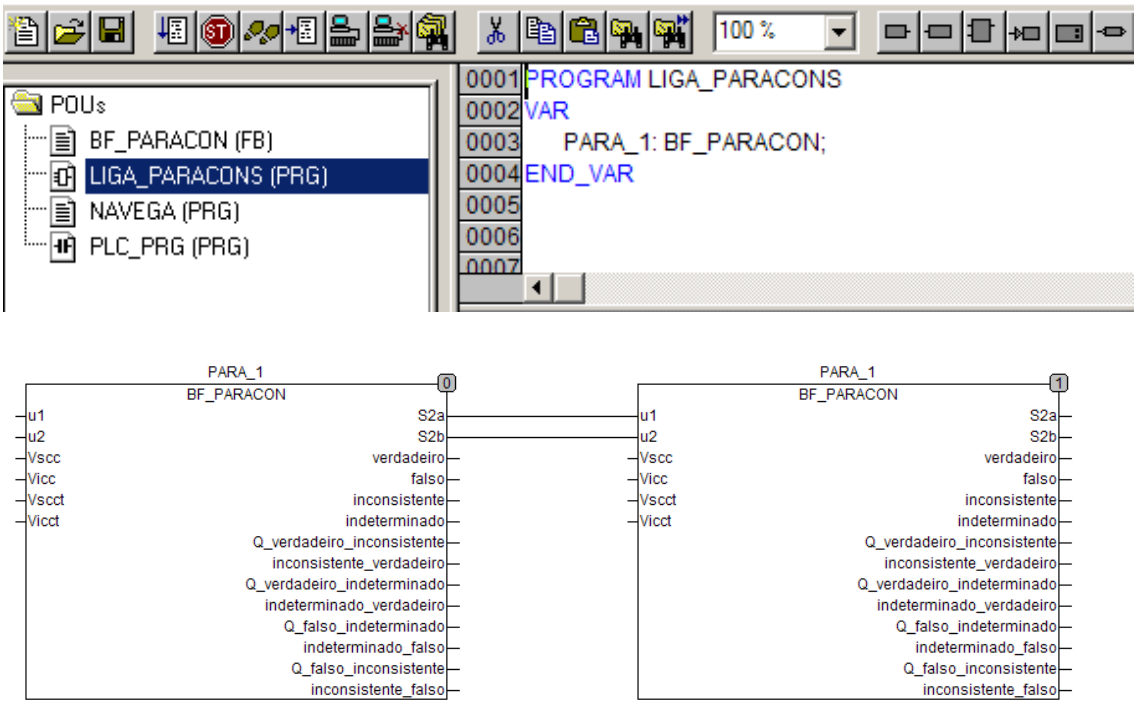


Figura 58. Programação em linguagem CFC com interligação de dois blocos do tipo BF_Paracon

Conforme foi visto no capitulo três (3), a linguagem Gráfico Contínuo de Funções (CFC) faz parte do conjunto de linguagens estabelecido pelas normas IEC e baseia-se na interligação de blocos funcionais. Na figura 58, pode ser verificada uma aplicação da linguagem Gráfico Contínuo de Funções (CFC) em que os dois (2) blocos do tipo BF_Paracon são interligados entre si.

Verifica-se que as saídas S2a e S2b do bloco “0” estão realimentando os graus de evidencia favorável (crença) u1 e evidência desfavorável u2 (descrença) do bloco “1”.

Nesta configuração, há apenas o bloco BF_Paracon como declaração de variáveis, ao qual está sendo atribuída a variável do tipo Bloco Funcional, denominada de PARA_1.

No futuro novas configurações poderão ser feitas interligando blocos do tipo BF_Paracon funcionando em redes de tratamento de sinais. Estas redes serão configuradas em Sistemas de Controle robustos, com alto grau de precisão, atuando na automação de malhas em processos industriais complexos.

6 CONCLUSÕES

Nesta pesquisa verificou-se o desenvolvimento de um método paraconsistente de controle com a implementação do primeiro Bloco Funcional baseado na LPA2V em um Controlador Programável (CP) com programação fundamentada na Norma IEC 61131-3. Essa implementação veio a atender a norma em seu item 7, cujo teor trata de incentivo as pesquisas com lógicas não-clássicas.

Os resultados obtidos abrem um amplo campo de desenvolvimento para utilização de lógicas não-clássicas aplicadas a Sistemas de Controle e Automação com a utilização de CPs.

A pesquisa desenvolvida até aqui, além de proporcionar uma ampla visão sobre as inovações na área de Projetos de Automação e Controle utilizando CPs, mostrou que a implementação do bloco funcional BF_Paracon oferece uma resposta satisfatória a norma IEC 61131 em seus itens 3 e 7.

Os resultados obtidos dos estudos comparativos do comportamento da malha de temperatura, frente ao um controle utilizando o método convencional (PID) e a análise paraconsistente (LPA2v), vem demonstrar que o bloco BF_Paracon foi aplicado com sucesso em uma planta de processos contínuos. Pode-se afirmar, portanto que a Lógica Paraconsistente Anotada com Anotação de 2 Valores é seguramente uma das lógicas não-clássicas que apresenta grandes potencialidades de aplicações conjuntas com as técnicas usuais de controle aplicadas nos Controladores Programáveis (CPs).

Como principal contribuição nas pesquisas de automação e controle pode-se destacar a criação do bloco funcional (BF_Paracon), que engloba características de uma lógica não-clássica. Fica demonstrado que a sua modularidade facilita a implementação de sistemas de controle de automação baseados em fundamentos da Lógica Paraconsistente e oferece condições que permitem a união de blocos atuando em conjunto com lógicas clássicas e sistemas baseados em outras linguagens pertencentes a Norma IEC 61131-3.

Pode-se destacar outra contribuição no fato de que este trabalho utilizou o Algoritmo Para-Analisador, que proporciona um controle “discreto” em doze níveis, os quais são ativados de acordo com a área do reticulado da LPA2V. No entanto, outros algoritmos da LPA2v que respondem com valores analógicos poderão ser implementados e proporcionar ajuste contínuo da variável de processo. Esta é uma

linha de pesquisa que pode ser explorada em futuros estudos baseada nos fundamentos de aplicação da Lógica Paraconsistente aqui apresentados.

REFERÊNCIAS BIBLIOGRÁFICAS

ABE, J. M. **Fundamentos da Lógica Anotada**, (Foundations of Annotated Logic) (in Portuguese), Ph. D. Thesis, Universidade de São Paulo, São Paulo, 1992.

ABE, J. M. **Paraconsistent Artificial Neural Networks: an Introduction, Lecture Notes In Artificial Intelligence 3214**, Springer, Eds. J.G. Carbonell & J. Siekmann, ISSN 0302-9743, ISBN 3-540-23206-0, pp. 942-948, 2004.

BARBUY, H. S.; GOLDEMBERG, C. **Regulador De Tensão De Gerador**. Boletim Técnico Da Escola Politécnica Da Usp. Bt/Pea, São Paulo, V. 621, N.Bt/Pea, P. 3121-3133, ISSN 1413-2214, 2001.

BARBUY, H. S.; ROCCO, A. ; FERNANDES, L. A. P.; GOLDEMBERG, C.. **Rectifier Choices For Synchronous Generator Excitation**. Anais Do 7º Congresso Brasileiro De Eletrônica De Potência, Fortaleza/ Ceará/ Brasil, P. 160-166, 2003.

BOTTURA FILHO, J. A. **Um estudo de caso de organização de programas de automação industrial baseada na Norma IEC61131-3.** Monografia de Conclusão de Curso. PUC. São Paulo, 2007.

CARBOGIM, D. V. **Programação em Lógica Anotada: Teoria e Aplicações**. Dissertação de Mestrado. Instituto de Matemática e Estatística – USP. São Paulo, 1996.

DE CARVALHO, F. R.; BRUNSTEIN, Israel; ABE, J. M. **Um Estudo de Tomada de Decisão Baseado em Lógica Paraconsistente Anotada: Avaliação do Projeto de uma Fábrica.** Revista Pesquisa e Desenvolvimento Engenharia de Produção n.1, p. 47-62, dez. 2003.

DA COSTA, N. C. A, ABE, J.M. e SUBRAHMANIAN V. S. **Remarks on Annotated Logic**. Instituto de Estudos Avançados da USP, 2000.

DA COSTA, N. C. A; HENSCHEN, L. J.; LU, J. J. et al. **Automatic Theorem Proving in Paraconsistent Logics: Teory and Implementation**. Instituto de Estudos Avançados da USP - 1990.

DA COSTA, N. C. A & SUBRAHMANIAN, V. S.. **Paraconsistent Logics as a Formalism for Reasoning About Inconsistent Knowledge Bases**. Instituto de Estudos Avançados da USP - 1989.

DA COSTA, N. C. A; SUBRAHMANIAN, V. S. e VAGO, C. **The Paraconsistent Logics Pτ**. Instituto de Estudos Avançados da USP - 1989.

DA SILVA FILHO, J. I. **Métodos de Aplicações da Lógica Paraconsistente Anotada de anotação com dois valores-LPA2v com construção de Algoritmo e Implementação de Circuitos Eletrônicos.** Tese (Doutorado) – Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais. São Paulo, 1999.

DA SILVA FILHO, J. I. & ABE, J. M., **Fundamentos das Redes Neurais Paraconsistentes – Destacando Aplicações em Neurocomputação,** (in Portuguese) Editora Arte & Ciência, ISBN 85-7473-045-9, 247 pp., 2001.

DA SILVA FILHO, **Conheça O Robo Emmy – O Primeiro Robô Que Funciona com a Lógica Paraconsistente**. Disponível em: http;//paralogike.com.br/roboemmy.htm. Acesso em 02 de março de 2012.

FALQUETE, V. L. M. **Utilização de Lógica Paraconsistente para Tratamento de Inconsistências em Sistemas de Raciocínio Baseado em Casos**. Dissertação de Mestrado. Pontifícia Universidade Católica do Paraná, Curitiba, 2004.

GOMIDE, F. C. e GUDWIN, R. R. **Modelagem, Controle, Sistemas e Lógica Fuzzy**. Departamento de Engenharia de Computação e Automação Industrial (DCA).Faculdade de Engenharia Elétrica (FEE).Universidade Estadual de Campinas (UNICAMP). SBA Controle & Automação/Vol.4 n°3/setembro-outubro, pp. 97-115, 1994.

GUIMARÃES H. C. F., **Norma IEC 61131-3 para Programação de Controladores Programáveis: Estudo e Aplicação.** Monografia de conclusão de curso Universidade Federal do Espírito Santo, 2005.

HASEGAWA, F. M. **Uma Abordagem Baseada em Lógica Paraconsistente para Avaliação de Ofertas em Negociações entre Organizações Artificiais**. Dissertação de Mestrado da Pontifícia Universidade Católica do Paraná, Curitiba, 2004.

HUGHES, T. A., **Programable Controllers. - Resources Formeasurements and Control Series - ISA.** North Caroline, InstrumentSociety of America., 1989, 252p.

INTERNATIONAL ELECTROTECHNICAL COMISSION – **IEC. IEC61131-3 Programmable Controllers - Part 3. Programming Languages,** IEC, 1993.

LEMES NETO, M. C. e VENSON N. **Lógica Paraconsistente**. Departamento de Informática e Estatística – INE, Universidade Federal de Santa Catarina (UFSC), Brasil, 2002.

LIMA JR L. H. C. de. **Lógica Paraconsistente Anotada: Aplicação na Aquisição de Dados de Sensores de Temperatura.** Dissertação de mestrado em Engenharia Elétrica. Universidade Federal de Campina Grande. 2003.

MACIEL, A. C.; FRANCISCO, B. S. U.; FARIAS, I, S. et al, **Aplicação Em Robótica Do Algoritmo Para- Analisador Utilizando A Tecnologia Lego Mindstorms Nxt.** Seleção Documental - GLPA N.21 Ano 6 Ed. Paralogike - Santos - SP Brasil, pp. 5-10, 2011.

MÁRIO, M. C. **Proposta de aplicação das redes neurais artificiais paraconsistentes como classificador de sinais utilizando aproximação funciona**. Dissertação (Mestrado) - Universidade Federal de Uberlândia. 129 p. Uberlândia- MG, 2003.

MIYAGI, P. E. **Controle Programável**. Editora Edgard Blücher Ltda. – 1996.

MOLLENKAMP, R. A. **Controle Automático de Processos**. EBRAS Editora Brasileira – SMAR. 1988.

OGATA, K. **Engenharia de Controle Moderno**, 2ª ed. – Prentice Hall, 1990.

OLIVEIRA, M.; SEIXAS, C.; BOTTURA FILHO, J., **Aplicando a norma IEC 61131 na Automação de Processos**, 2007.

SIGHIERI, L. & NISHINARI, A. **Controle Automático de Processos Industriais – Instrumentação**. Editora EDGARD BLÜCHER LTDA. 2ª Edição – 1988.

RIBEIRO, A. M. **Instrumentação**, 8ª ed. Tek Treinamento & Consultoria Ltda, 1999.

SUCOLOTTI, A. A.; CENTENARO, A. C. e DRUZIANI, C. F. M. **Recuperação de Informação em Bases Textuais: Uma Abordagem Paraconsistente**. Universidade Paranaense (UNIPAR), 2007.

THOMAS D. McGee. **Principles and Methods of Temperature Measurement** – Wiley-Interscience Publication, 1988.

TORRES et al, **Otimização de Estratégias de Controle em Sistemas Multivariáveis, Congresso Internacional de Automação, Sistemas e Instrumentação – ISA Show Brasil**, São Paulo, outubro, 2001.

TORRES, C. R., **Sistema Inteligente Baseado na Lógica Paraconsistente Anotada Evidencial Et para Controle e Navegação de Robôs Móveis Autônomos em um Ambiente não Estruturado,** Tese de doutorado em Ciências em Energia Elétrica – Universidade Federal de Itajubá. 193 p. Itajubá- MG, agosto de 2010.

ANEXO A – Programação Completa do Bloco Funcional BF_Paracon

```
(*Inicializando Programação*)
(*Indices Superiores e inferiores para definição*)
Vscc;
Vicc;
Vscct;
Vicct;

(*Variaveis do programa*)
u1;
u2;
S2a;
S2b;
Gct;
Gc;

(*Operações Matemáticas*)
Gct:=(u1+u2)-10000;
Gc:=u1-u2;

modulo_Gc:=0-Gc;
modulo_Gct:=-Gct;


(*Determinação dos casos lógicos extremos*)
IF(Gc>=Vscc)THEN
        verdadeiro:=1; (*Verdadeiro*)
        falso:=0;
        inconsistente:=0;
        indeterminado:=0;
        Q_verdadeiro_inconsistente:=0;
        inconsistente_verdadeiro:=0;
        Q_verdadeiro_indeterminado:=0;
        indeterminado_verdadeiro:=0;
        Q_falso_indeterminado:=0;
        indeterminado_falso:=0;
        Q_falso_inconsistente:=0;
        inconsistente_falso:=0;

ELSIF(Gc<=Vicc)THEN
        verdadeiro:=0;
        falso:=1; (*falso*)
        inconsistente:=0;
        indeterminado:=0;
        Q_verdadeiro_inconsistente:=0;
        inconsistente_verdadeiro:=0;
        Q_verdadeiro_indeterminado:=0;
```

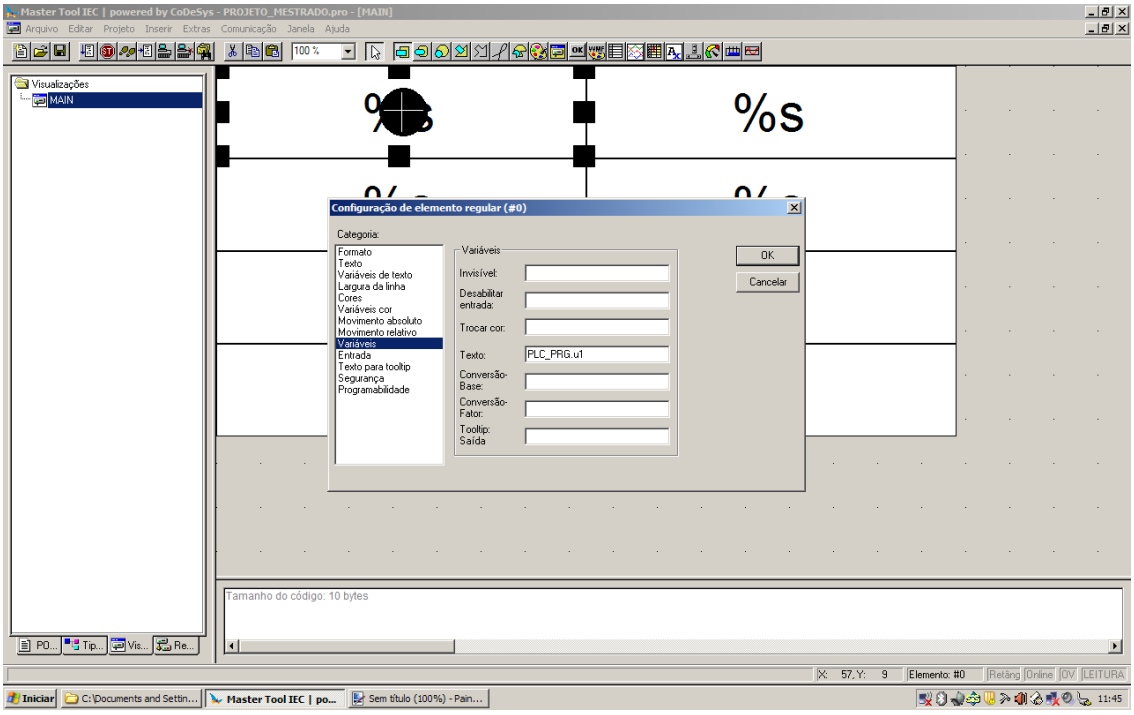
```
        indeterminado_verdadeiro:=0;
        Q_falso_indeterminado:=0;
        indeterminado_falso:=0;
        Q_falso_inconsistente:=0;
        inconsistente_falso:=0;

ELSIF(Gct>=Vscct)THEN
        verdadeiro:=0;
        falso:=0;
        inconsistente:=1;(*Inconsistente*)
        indeterminado:=0;
        Q_verdadeiro_inconsistente:=0;
        inconsistente_verdadeiro:=0;
        Q_verdadeiro_indeterminado:=0;
        indeterminado_verdadeiro:=0;
        Q_falso_indeterminado:=0;
        indeterminado_falso:=0;
        Q_falso_inconsistente:=0;
        inconsistente_falso:=0;

ELSIF(Gct<=Vicct)THEN
        verdadeiro:=0;
        falso:=0;
        inconsistente:=0;
        indeterminado:=1; (*Indeterminado*)
        Q_verdadeiro_inconsistente:=0;
        inconsistente_verdadeiro:=0;
        Q_verdadeiro_indeterminado:=0;
        indeterminado_verdadeiro:=0;
        Q_falso_indeterminado:=0;
        indeterminado_falso:=0;
        Q_falso_inconsistente:=0;
        inconsistente_falso:=0;

END_IF


(*Determinação dos estados Lógicos não Extremos*)
(*primeira fase*)

IF(Gc<Vscc AND Gc>=0 AND Gct<Vscct AND Gct>=0) THEN
        IF(Gc>=Gct)THEN
                verdadeiro:=0;
                falso:=0;
                inconsistente:=0;
                indeterminado:=0;
                Q_verdadeiro_inconsistente:=1; (*Quase verdadeiro tendendo a
inconsistente*)
                inconsistente_verdadeiro:=0;
                Q_verdadeiro_indeterminado:=0;
                indeterminado_verdadeiro:=0;
```

```
            Q_falso_indeterminado:=0;
            indeterminado_falso:=0;
            Q_falso_inconsistente:=0;
            inconsistente_falso:=0;
        ELSE
            verdadeiro:=0;
            falso:=0;
            inconsistente:=0;
            indeterminado:=0;
            Q_verdadeiro_inconsistente:=0;
            inconsistente_verdadeiro:=1; (*Inconsistente tendendo a verdadeiro*)
            Q_verdadeiro_indeterminado:=0;
            indeterminado_verdadeiro:=0;
            Q_falso_indeterminado:=0;
            indeterminado_falso:=0;
            Q_falso_inconsistente:=0;
            inconsistente_falso:=0;
        END_IF

(*segunda fase _pesquisar modulo*)

ELSIF(Gc<Vscc AND Gc>=0 AND Gct<=0 AND Gct>Vicct) THEN
        IF(Gc>=modulo_Gct)THEN (*necessita modulo de Gct*)
            verdadeiro:=0;
            falso:=0;
            inconsistente:=0;
            indeterminado:=0;
            Q_verdadeiro_inconsistente:=0;
            inconsistente_verdadeiro:=0;
            Q_verdadeiro_indeterminado:=1; (*Quase verdadeiro tendendo a
indeterminado*)
            indeterminado_verdadeiro:=0;
            Q_falso_indeterminado:=0;
            indeterminado_falso:=0;
            Q_falso_inconsistente:=0;
            inconsistente_falso:=0;
        ELSE
            verdadeiro:=0;
            falso:=0;
            inconsistente:=0;
            indeterminado:=0;
            Q_verdadeiro_inconsistente:=0;
            inconsistente_verdadeiro:=0;
            Q_verdadeiro_indeterminado:=0;
            indeterminado_verdadeiro:=1; (*Indeterminado tendendo a verdadeiro*)
            Q_falso_indeterminado:=0;
            indeterminado_falso:=0;
            Q_falso_inconsistente:=0;
            inconsistente_falso:=0;
        END_IF
```

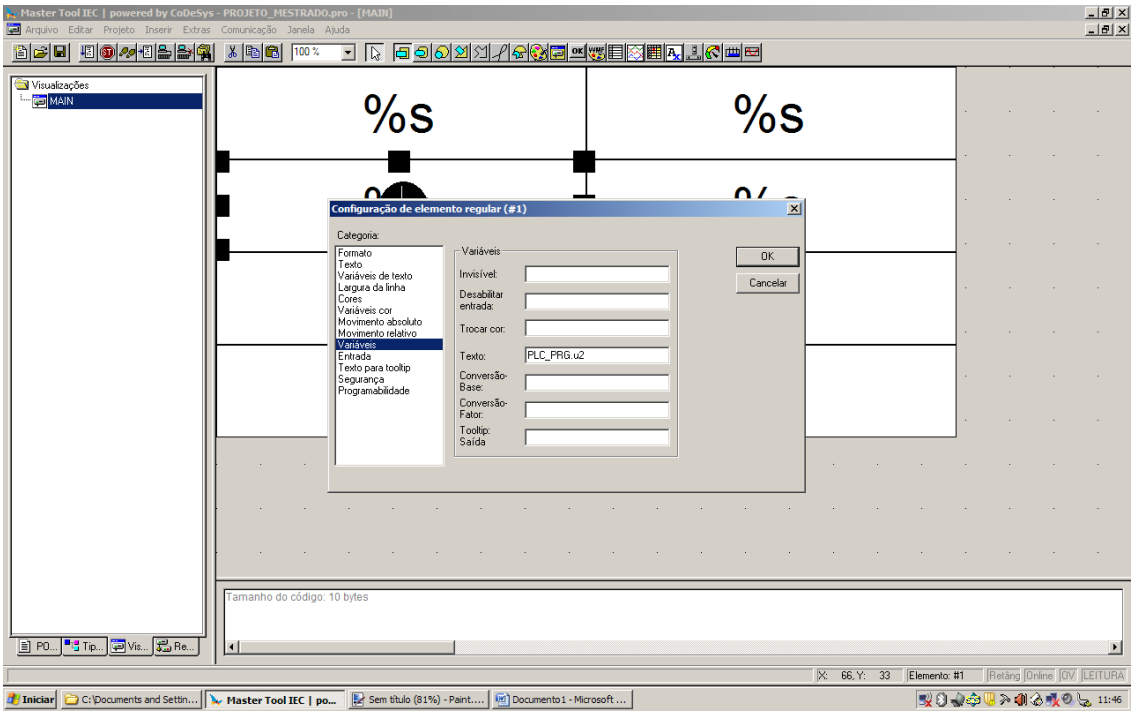
```
(*terceira fase _pesquisar modulo*)

ELSIF(Gc>Vicc AND Gc<=0 AND Gct<=0 AND Gct>Vicct) THEN
        IF(modulo_Gc>=modulo_Gct)THEN (*necessita modulo de Gc e Gct*)
                verdadeiro:=0;
                falso:=0;
                inconsistente:=0;
                indeterminado:=0;
                Q_verdadeiro_inconsistente:=0;
                inconsistente_verdadeiro:=0;
                Q_verdadeiro_indeterminado:=0;
                indeterminado_verdadeiro:=0;
                Q_falso_indeterminado:=1; (*Quase falso tendendo a indeterminado*)
                indeterminado_falso:=0;
                Q_falso_inconsistente:=0;
                inconsistente_falso:=0;
        ELSE
                verdadeiro:=0;
                falso:=0;
                inconsistente:=0;
                indeterminado:=0;
                Q_verdadeiro_inconsistente:=0;
                inconsistente_verdadeiro:=0;
                Q_verdadeiro_indeterminado:=0;
                indeterminado_verdadeiro:=0;
                Q_falso_indeterminado:=0;
                indeterminado_falso:=1; (*Indeterminado tendendo a falso*)
                Q_falso_inconsistente:=0;
                inconsistente_falso:=0;
        END_IF

(*quarta fase _pesquisar modulo*)

ELSIF(Gc>Vicc AND Gc<=0 AND Gct>=0 AND Gct<Vscct) THEN
        IF(modulo_Gc>=Gct)THEN (*necessita modulo de Gc e Gct*)
                verdadeiro:=0;
                falso:=0;
                inconsistente:=0;
                indeterminado:=0;
                Q_verdadeiro_inconsistente:=0;
                inconsistente_verdadeiro:=0;
                Q_verdadeiro_indeterminado:=0;
                indeterminado_verdadeiro:=0;
                Q_falso_indeterminado:=0;
                indeterminado_falso:=0;
                Q_falso_inconsistente:=1; (*Quase falso tendendo a inconsistente*)
                inconsistente_falso:=0;
        ELSE
                verdadeiro:=0;
                falso:=0;
```

```
                inconsistente:=0;
                indeterminado:=0;
                Q_verdadeiro_inconsistente:=0;
                inconsistente_verdadeiro:=0;
                Q_verdadeiro_indeterminado:=0;
                indeterminado_verdadeiro:=0;
                Q_falso_indeterminado:=0;
                indeterminado_falso:=0;
                Q_falso_inconsistente:=0;
                inconsistente_falso:=1; (*Inconsistente tendendo a falso*)
        END_IF
END_IF
S2a:=Gct;
S2b:=Gc;
```

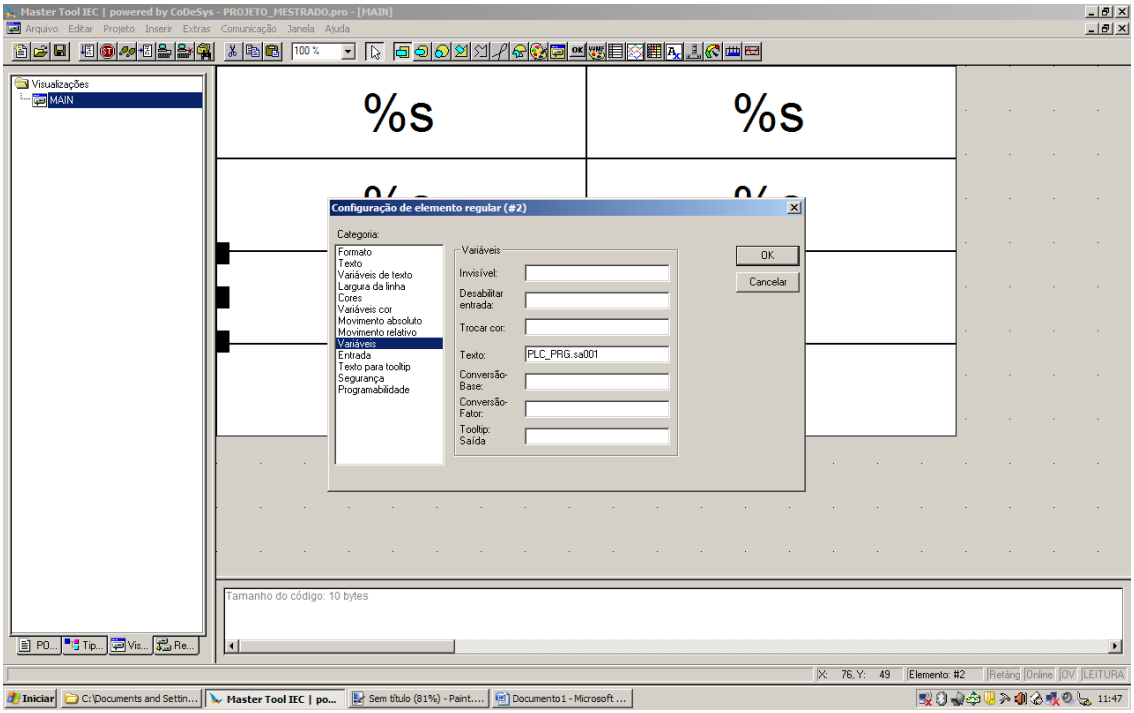
ANEXO B – Configuração da IHM – Entradas e Saídas do BF_Paracon



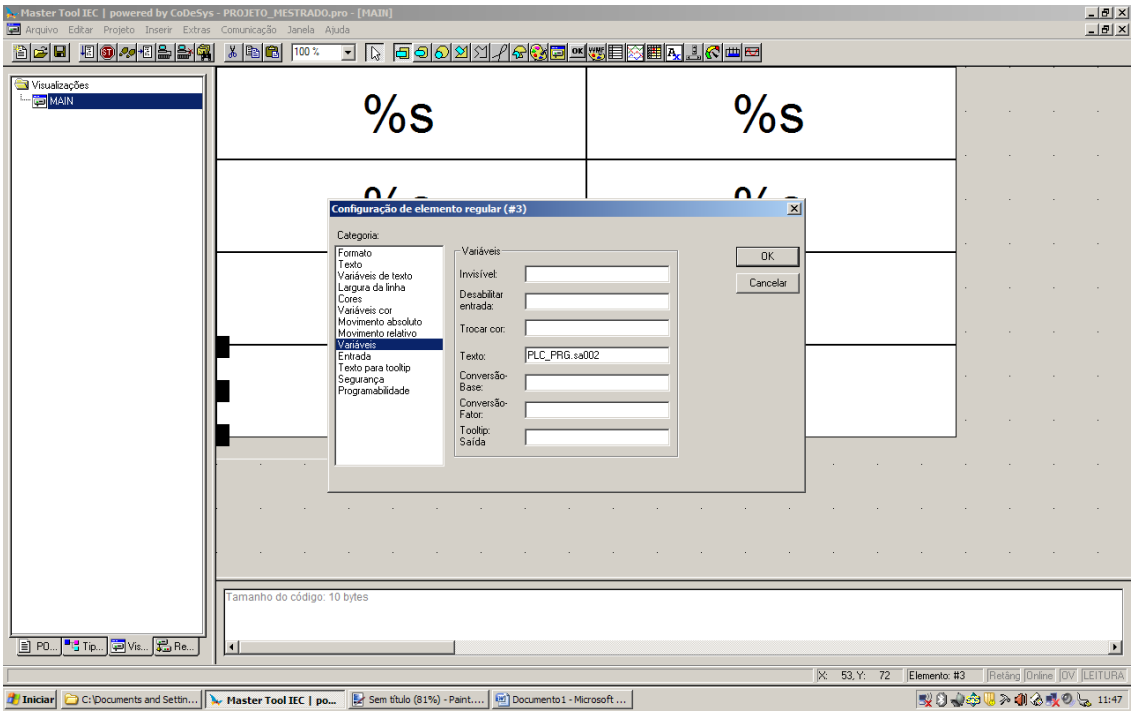
Criação da variavel µ1 na IHM do CP - DU 351



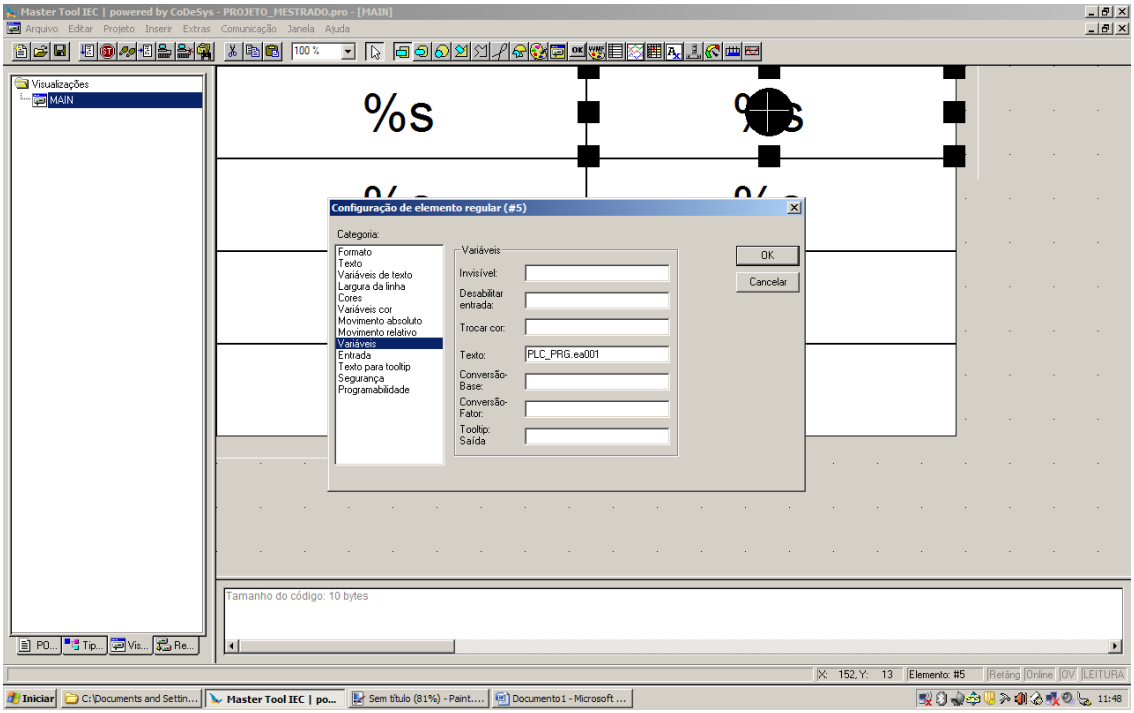
Criação da variavel µ2 na IHM do CP - DU 351



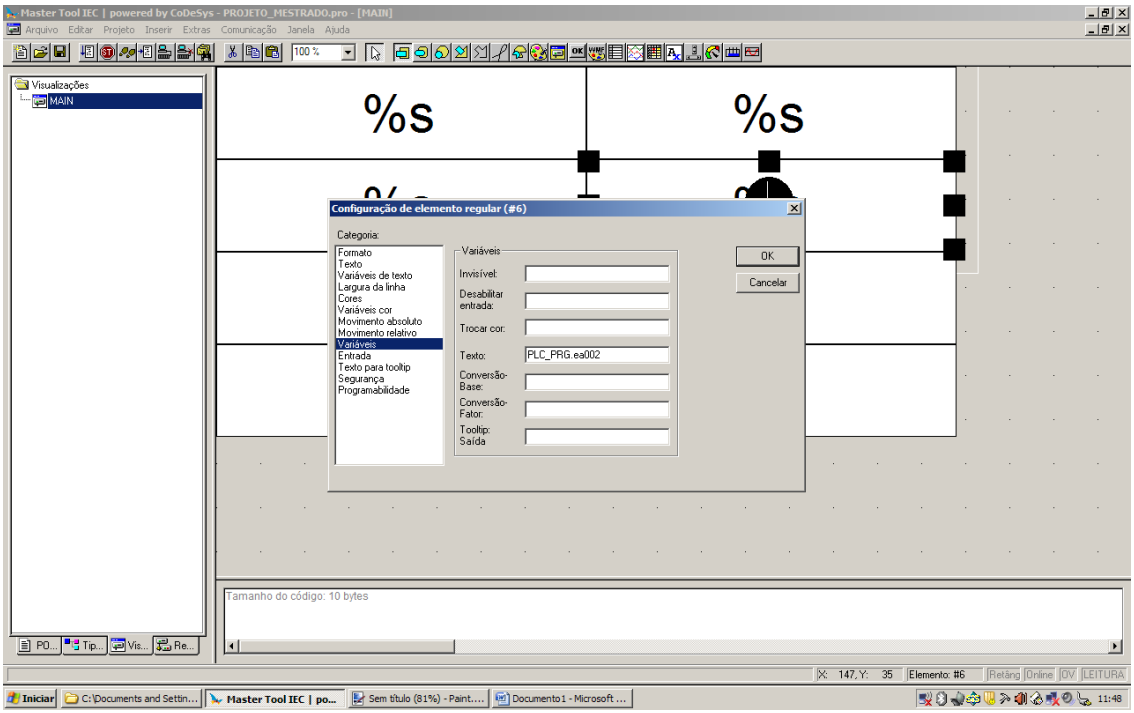
Criação da variavel de saída sa001 na IHM do CP - DU 351



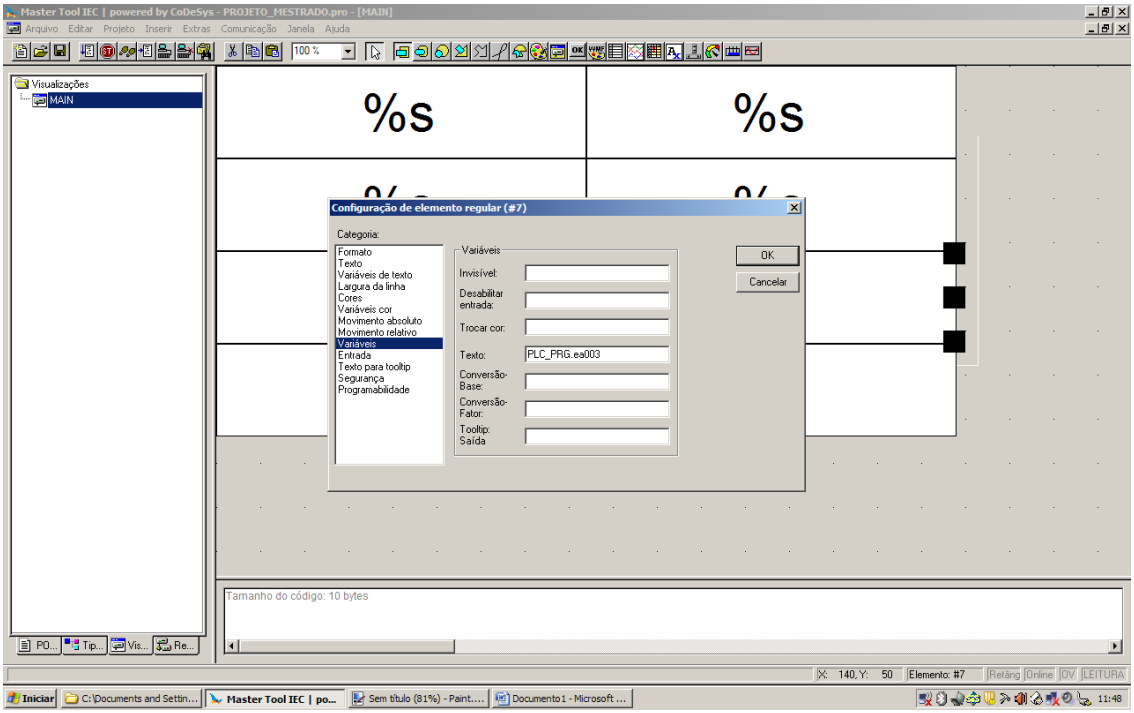
Criação da variavel de saída sa002 na IHM do CP - DU 351



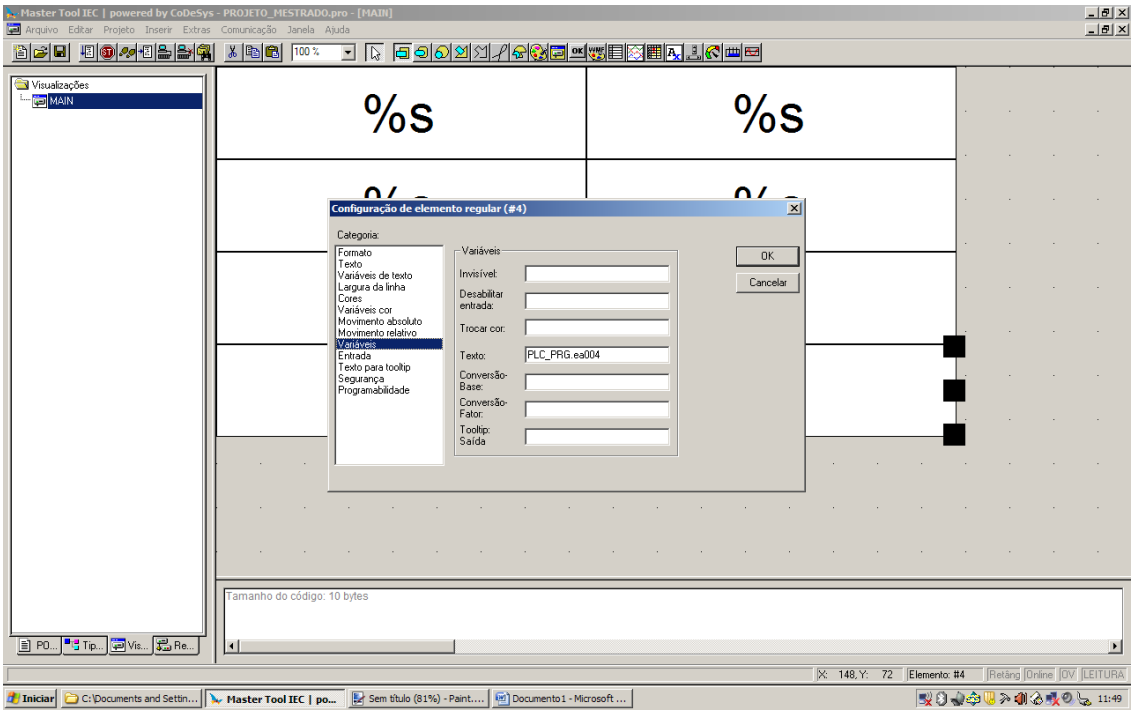
Criação da variavel de controle de Limite Superior de Certeza na IHM do CP - DU 351



Criação da variavel de controle de Limite Inferior de Certeza na IHM do CP - DU 351



Criação da variavel de controle de Limite Superior de Contradição na IHM do CP - DU 351



Criação da variavel de controle de Limite Inferior de Contradição na IHM do CP - DU 351