UNIVERSIDADE DO ESTADO DE SANTA CATARINA - UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
PROGRAMA DE PÓS GRADUAÇÃO EM ENGENHARIA ELÉTRICA

RAMON AUGUSTO KÜHL

METODOLOGIA DE IMPLEMENTAÇÃO DO CONTROLE SUPERVISÓRIO MODULAR LOCAL ADERENTE À NORMA IEC 61499

JOINVILLE

2018

RAMON AUGUSTO KÜHL

METODOLOGIA DE IMPLEMENTAÇÃO DO CONTROLE SUPERVISÓRIO MODULAR LOCAL ADERENTE À NORMA IEC 61499

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, para a obtenção do grau de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Roberto Silvio Ubertino Rosso Junior
Coorientador: Prof. Dr. André Bittencourt Leal

JOINVILLE

2018

Kühl, Ramon Augusto
METODOLOGIA DE IMPLEMENTAÇÃO DO CONTROLE SUPERVISÓRIO MODULAR LOCAL ADERENTE À NORMA IEC 61499 / Ramon Augusto Kühl. - Joinville , 2018.
115 p.

Orientador: Roberto Silvio Ubertino Rosso Junior
Co-orientador: André Bittencourt Leal
Dissertação (Mestrado) - Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação Profissional em Engenharia Elétrica, Joinville, 2018.

1. Automação distribuída. 2. Controle modular local. 3. IEC 61499. 4. Sistemas a eventos discretos. 5. Teoria de controle supervisório. I. Rosso Junior, Roberto Silvio Ubertino . II. Leal, André Bittencourt . , .III. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação Profissional em Engenharia Elétrica. IV. Título.

Metodologia de Implementação do Controle Supervisório Modular Local Aderente à Norma IEC61499

por

Ramon Augusto Kühl

Esta dissertação foi julgada adequada para obtenção do título de

MESTRE EM ENGENHARIA ELÉTRICA

Área de concentração em "Automação de Sistemas"
e aprovada em sua forma final pelo

CURSO DE MESTRADO PROFISSIONAL EM ENGENHARIA ELÉTRICA
DO CENTRO DE CIÊNCIAS TECNOLÓGICAS DA
UNIVERSIDADE DO ESTADO DE SANTA CATARINA.

Prof. Dr. Roberto Silvio Ubertino Rosso Junior
CCT/UDESC (Orientador/Presidente)
Membro da Banca Examinadora

Prof. Dr. Douglas Wildgrube Bertol
CCT/UDESC
Membro da Banca Examinadora

Prof. Dr. André Bittencourt Leal
CCT/UDESC (Coorientador)

Prof. Dr. Marco Aurélio Wehrmeister
UTFPR
Membro da Banca Examinadora

Joinville,SC, 20 de setembro de 2018.

Dedico este trabalho à minha linda esposa Kamila que não mediu esforços em me apoiar para que esse trabalho pudesse ser concluído da melhor forma possível.

AGRADECIMENTOS

Gostaria de agradecer sinceramente a todos aqueles que direta ou indiretamente contribuíram para a elaboração desta dissertação de mestrado.

Agradeço a Deus por ter me dado forças e sabedoria para concluir esse trabalho e superar mais um desafio em minha vida.

Aos meus professores Roberto Silvio Ubertino Rosso Junior e André Bittencourt Leal pelas orientações recebidas, pela amizade e pelo tempo dedicado para me manter sempre no rumo certo.

Ao doutorando Leandro Israel Pinto pelo auxílio nos momentos de dúvida, pela amizade e pelo apoio no desenvolvimento desse trabalho.

Aos meus pais, que sempre me apoiaram com conselhos sábios para que eu nunca desanimasse, mas sempre me mantivesse firme nessa jornada, e a todos os outros familiares.

Ao Renan Gustavo Kühl, ao Michel Kazmierski e a todo o pessoal da Erzinger e da TAB energia que deram seu melhor nos momentos que eu não pude estar presente profissionalmente.

À Universidade do Estado de Santa Catarina pela estrutura fornecida para que eu pudesse alcançar mais uma meta em minha vida.

E a pessoa mais importante da minha vida, que nunca mediu esforços para me compreender e entender em todos os momentos dessa longa e desafiadora jornada, me enchendo todos os dias com seu infinito amor e me mostrando que sou capaz de fazer coisas maravilhosas que nem eu mesmo sabia. Obrigado por tudo minha linda esposa e eterno amor Kamila Silva de Morais Kühl.

“Quando faltam máquinas, você as pode comprar; se não tiver dinheiro, pode pegar emprestado; mas homens você não pode comprar ou pedir emprestado, e homens motivados são a base do êxito.”

Eggon João da Silva

RESUMO

Esse trabalho contribui com as pesquisas em torno da norma IEC 61499 e da teoria de controle supervisório de sistemas a eventos discretos através de uma metodologia de implementação da estrutura de controle modular local nos blocos de função definidos pela norma IEC 61499. A teoria de controle supervisório modular local é utilizada para obtenção de uma lógica de controle ótima e minimamente restritiva, a qual geralmente é implementada através das linguagens de programação definidas pela norma IEC 61131 na indústria moderna. No entanto, esta norma possui limitações no que se refere a implementação de controle de sistemas de automação distribuídos, como, por exemplo, a não exigência de interoperabilidade, o que dificulta a utilização de dispositivos inteligentes de mais de um fabricante num mesmo sistema. Logo, percebe-se uma carência em metodologias que implementem a estrutura de controle supervisório modular local de acordo com os blocos de função da norma IEC 61499, a qual foi originada para trazer soluções e benefícios para o controle de sistemas de automação distribuída. Este trabalho aborda uma metodologia baseada em duas etapas que criam um passo a passo para que sistemas a eventos discretos, modelados pela teoria de controle supervisório modular local, possam ser implementados de acordo com as definições da norma IEC 61499. Um estudo de caso de um sistema flexível de manufatura, o qual teve sua lógica de controle obtida pela abordagem modular local e implementada pelos blocos de função da IEC 61499, demonstra o procedimento conforme metodologia desenvolvida neste trabalho.

Palavras-chaves: Automação Distribuída, Controle Modular Local, IEC 61499, Sistemas a Eventos Discretos, Teoria de Controle Supervisório.

ABSTRACT

This work contributes with the research on the IEC 61499 standard and supervisory control theory of discrete event systems through a methodology based on the local modular structure of control implemented using the function blocks defined in the IEC 61499 standard. The modular local supervisory control theory is used to obtain an optimal and minimally restrictive control logic, which is often implemented through programming languages according to the IEC 61131 standard in the modern industry. However, this standard has limitations regarding the control implementation in distributed automation systems, such as not requiring interoperability, which makes it difficult to use intelligent devices from more than one manufacturer in the same system. Therefore, there is a lack of methodologies that implement the local modular supervisory control structure using the function blocks compliant with the IEC 61499 standard, which originated to bring solutions and benefits to the control of distributed automation systems. This work addresses a two-step methodology that creates a step-by-step approach for discrete event systems, modeled by local modular supervisory control theory, to be implemented according to the IEC 61499 standard. Validation was performed through a case study of a flexible manufacturing system. The system had its control logic obtained by the local modular approach and implemented using IEC 61499 compliant function blocks, according to the methodology developed in this work.

Key-words: Distributed Automation, Local Modular Approach, IEC 61499, Discrete Events Systems, Supervisory Control Theory.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

CLP	Controlador Lógico Programável
SED	Sistema a Eventos Discretos
FB	Function Block
TCS	Teoria de Controle Supervisório
ADEF	Autômato Determinístico de Estados Finitos
IEC	*International Electrotechnical Commission*
DCS	Sistema de Controle Distribuído
IHM	Interface Homem Máquina
TLA	Teoria de Linguagens e Autômatos
ECC	*Execution Chart Control*
FMS	Sistema Flexível de Manufatura
XML	*eXtensible Markup Language*
Lm	Linguagem marcada
RSP	Representação por Sistema Produto
POU	*Program Organization Unit*
FUN	Função
PROG	Programa
LD	*Ladder*
FBD	*Function Block Diagram*
SFC	*Sequencial Function Chart*
ST	*Structured Text*
IL	*Instrunction List*
SIFB	*Service Interface Function Block*
CFB	*Composite Function Block*
SCADA	*Supervisory Control and Data Acquisition*

LISTA DE SÍMBOLOS

L	Linguagem
σ	Evento
Σ	Alfabeto de símbolos finitos e não vazio
G	Autômato que representa os subsistemas da planta
E	Autômato que representa as especificações de controle
K	Linguagem alvo
R	Autômato que representa a linguagem alvo
S	Autômato que representa a máxima linguagem controlável
Q	Conjunto de estados de um autômato
Q_m	Conjunto de estados marcados ou de aceitação de um autômato
q_0	Estado inicial de um autômato
δ	Função de transição de um autômato
ε	Palavra vazia
$L = \emptyset$	Linguagem vazia
PS	Sistema Produto
RS	Supervisor modular local reduzido

SUMÁRIO

1 INTRODUÇÃO

Uma série de mudanças revolucionárias na configuração de sistemas de manufatura ocorreram desde o sistema de produção artesanal até os sistemas utilizados atualmente (LOPES, 2012). A primeira revolução industrial é geralmente relacionada à máquina a vapor, iniciando a era industrial. A segunda revolução industrial é geralmente vista como a aplicação de eletricidade para criar produção em massa, especialmente na nova indústria automotiva. A terceira revolução industrial está geralmente ligada ao uso extensivo de eletrônicos e tecnologia da informação para automatizar a produção. Hoje já se fala muito na quarta revolução industrial (BASSI, 2017).

Em grande parte, hoje os sistemas de automação são projetados utilizando controladores lógicos programáveis (CLPs) aderentes a norma IEC 61131 (CRUZ, 2011). Sistemas de grande porte possuem um considerável número de CLPs, podendo ser caracterizados como sistemas distribuídos. Esses CLPs se comunicam via uma ou mais redes e controlam vários sensores e atuadores, o que normalmente ocasiona dificuldades para futuras modificações e extensões.

A dificuldade inerente à implementação de sistemas distribuídos, assim como a incompatibilidade entre dispositivos e ferramentas de diferentes fabricantes, podem dificultar uma adaptação ágil da linha de produção a novos produtos, geralmente tornando a empresa totalmente dependente de um único fabricante, o que é extremamente ruim para a sua competitividade (VYATKIN, 2007).

A norma IEC 61499 (LEWIS, 2008) fornece uma alternativa de implementação para esse tipo de sistema, propondo um mecanismo capaz de diminuir a complexidade da implementação de sistemas distribuídos e garantir três características importantes: a capacidade de se editar e compilar o mesmo código e bibliotecas em diferentes ambientes de programação existentes (portabilidade), a capacidade de se configurar vários dispositivos de diferentes fabricantes com a mesma ferramenta de software, com mínima ou nenhuma modificação (configurabilidade) e a capacidade de sistemas e dispositivos de diferentes fabricantes trocarem informações e as utilizarem em conjunto (interoperabilidade) (PINTO, 2014).

A norma também propõe mecanismos que facilitam a alteração em tempo de execução, a chamada reconfiguração dinâmica, utilizando uma abordagem orientada a componentes (blocos de funções), com a qual é possível descrever um sistema inteiro, independente da plataforma de execução (PINTO; LEAL; ROSSO, 2017).

Embora várias pesquisas têm sido desenvolvidas a respeito da IEC 61499, algumas barreiras ainda têm impedido uma aplicação maior da norma na indústria

moderna, como posição conservadora, questões técnicas de implementação, difusão do conhecimento teórico e prático, maioria das ferramentas com características voltadas ao meio acadêmico, entre outras (DAI; VYATKIN, 2011).

Com sistemas cada vez mais complexos que envolvem um grande número de subsistemas relacionados por intermédio de complexas lógicas de controle, as soluções para este tipo de problema não podem mais ser baseadas única e exclusivamente na experiência do projetista e no uso de métodos baseados na tentativa e erro. Nesse sentido, a adoção de métodos formais para a solução de problemas complexos de automação é fundamental para que as indústrias se tornem cada vez mais competitivas, deixando de lado problemas como, por exemplo, travamento de linhas de produção, ineficiência por lógicas de programação restritivas, entre outros (LEAL; CRUZ; HOUNSELL, 2012). Dentre os diversos métodos formais existentes na literatura, a Teoria de Controle Supervisório de Sistemas a Eventos Discretos tem sido usada com sucesso para a solução de problemas que envolvem aplicações nas áreas de automação da manufatura e controle de sistemas embarcados.

No trabalho de Ramadge e Wonham (1989) foi introduzida uma abordagem formal que permite a síntese de um controlador (chamado de supervisor), que garante o adequado funcionamento da planta em malha fechada. Neste, tanto o comportamento da planta quanto das especificações de controle são modelados por intermédio de autômatos de estados finitos, a partir dos quais se obtém um modelo que representa o supervisor monolítico para controle da planta. Tal abordagem apresenta um problema de crescimento exponencial do número de estados da planta em função do aumento de subsistemas. Além disso, a metodologia resulta em um único supervisor, o qual normalmente é implementado de forma centralizada, num único elemento de controle, como num CLP, por exemplo.

Para evitar esses problemas, Queiroz (2004) propôs uma abordagem de síntese modular local de supervisores que permite explorar tanto a modularidade da planta como das especificações de controle. Com o uso dessa abordagem, obtêm-se supervisores modulares locais que garantem que o comportamento do sistema em malha fechada será não bloqueante, minimamente restritivo, e obedecerá a todas as especificações de controle. Além disso, tais supervisores podem ser implementados de forma distribuída, em um ou mais elementos de controle.

Muito embora isso seja possível, a grande maioria dos trabalhos encontrados na literatura não abordam conjuntamente estratégias de implementação distribuída, estrutura de controle modular e a norma IEC 61499.

O trabalho de Vlad et al. (2009), por exemplo, aborda a ideia de implementar o código de controle aderente à IEC 61499 a partir de um sistema de automação previ-

amente modelado por redes de Petri. Os autores utilizam uma abordagem monolítica para implementar o código de controle através dos blocos de função da IEC 61499, o que representa um grande problema para sistemas mais complexos, nos quais os controladores monolíticos ficam com incontáveis estados e transições. Outros trabalhos que também utilizam redes de petri para modelagem da estrutura de controle e os blocos de função da IEC 61499 para implementação do código de controle são (BASILE; CHIACCHIO; GERBASIO, 2013), (ABRISHAMBAF et al., 2015), (HAGGE, 2007), (HAGGE; WAGNER, 2007) e (DROZDOV; DUBININ; VYATKIN, 2016).

O método apresentado no trabalho de Cengic et al. (2005) aplica a abordagem monolítica para implementar os blocos de função da IEC 61499. Segundo os autores, aplicar a teoria de controle supervisório para implementação do código de controle de um sistema pode reduzir significativamente o tempo de desenvolvimento de uma solução, porém existem certas dificuldades de implementação do controle em sistemas distribuídos.

O artigo escrito por Flordal et al. (2007) apresenta um método que utiliza informações em um ambiente de simulação de robôs para extrair automaticamente modelos de estados finitos. Esses modelos podem então ser usados para gerar um supervisor monolítico para garantir que situações indesejadas sejam evitadas. A partir de tal modelo de supervisor, é gerado automaticamente um código de controle baseado na IEC 61131 para CLPs. Toda a informação também é estruturada e pode ser exportada no formato XML, importante para padronização e interoperabilidade definida na IEC 61499.

Outros trabalhos já abordam o uso da norma IEC 61131 para implementação da teoria de controle supervisório, como no caso de Petin, Gouyon e Morel (2007), Afzalian, Noorbakhsh e Wonham (2015), Gelen e Uzam (2014), Leal, Cruz e Hounsell (2012), Moniruzzaman e Gohari (2007) e Moreira e Bassilio (2014).

O trabalho desenvolvido por Pinto, Leal e Rosso (2017) apresenta uma metodologia para a execução da reconfiguração dinâmica em sistemas cuja dinâmica é impulsionada pela ocorrência de eventos discretos. Aplicando a teoria de controle supervisório, foi desenvolvida uma metodologia para permitir mudança na lógica de controle do sistema em tempo de execução, garantindo a consistência local da aplicação, de acordo com a norma IEC 61499. Esse trabalho apresenta um estudo de caso baseado numa célula de manufatura serial com o intuito de comprovar a eficácia da metodologia proposta. Através da aplicação da teoria de controle supervisório, chega-se num supervisor monolítico para controle em malha fechada do sistema de automação. A partir desse supervisor monolítico, aplica-se a metodologia proposta e se chega em outro supervisor monolítico final, que permite a reconfiguração dinâmica, ou seja, que ocorra alterações de hardware em tempo de execução sem prejudicar o

funcionamento do sistema ou a segurança de sua operação.

Os trabalhos de Prenzel e Provost (2018), Vieira et al. (2017) e Vieira (2007) se diferem dos demais pelo fato de proporem uma metodologia que permite que a implementação seja concentrada em um único controlador ou distribuída em um conjunto de controladores. Entretanto, tal metodologia está baseada na norma IEC 61131-3, uma desvantagem frente aos benefícios que a norma IEC 61499 apresenta, como, por exemplo, ágil adaptação da linha de produção, independência no uso de dispositivos de qualquer fabricante, reconfiguração dinâmica, ambiente de programação único, menos gastos com licenças, padronização de programação e execução distribuída da lógica de controle em diversos dispositivos limitados em processamento e memória.

O trabalho de Pinto et al. (2015) e Pinto et al. (2016) apresenta o $ICARU-FB$, um ambiente para execução de sistemas aderente à norma IEC 61499 que pode ser executado em hardware de baixa capacidade de memória e processamento, como, por exemplo, o Arduino $ATmega2560$, um microcontrolador de 8-bits (NAYYAR; PURI, 2016). Esse trabalho traz importantes contribuições aos estudos da IEC 61499, pois garante que a norma possa sim ser implementada em hardwares distribuídos com menor poder de processamento e armazenamento de memória, ao contrário da norma IEC 61131 executada nos CLPs utilizados em grande parte das aplicações industriais modernas.

É preciso preencher essa lacuna que existe entre a síntese do modelo de controle através da teoria de controle supervisório modular local com a implementação do código baseada na arquitetura orientada a eventos da norma IEC 61499 (ZAYTOON; RIERA, 2017), pois desta forma as indústrias modernas poderão se beneficiar das vantagens da união dessas duas abordagens.

Esta dissertação de mestrado apresenta uma metodologia inovadora que permite que a lógica de controle para um sistema de automação, obtida por intermédio da abordagem modular local, possa ser implementada por blocos de função de acordo com as definições da norma IEC 61499, gerando um padrão funcional e replicável para qualquer tipo de sistema a eventos discretos.

1.1 OBJETIVO GERAL

O objetivo geral deste trabalho consiste em desenvolver uma metodologia de implementação da estrutura modular local que seja aderente à norma IEC 61499.

1.2 OBJETIVOS ESPECÍFICOS

No intuito de alcançar o objetivo geral, são definidos os seguintes objetivos específicos:

1. Estudar as características da norma IEC 61499, a fim de justificar a sua utilização para implementação do código de controle em sistemas distribuídos de automação industrial;

2. Definir formas de mapear a estrutura de controle modular local nos blocos de função definidos pela norma IEC 61499;

3. Definir formas de interligações entre os eventos e variáveis de dados dos blocos de função mapeados, a fim de que os mesmos gerem o resultado de controle esperado;

4. Validar a metodologia desenvolvida através de um estudo de caso.

1.3 ORGANIZAÇÃO DO TRABALHO

Os demais capítulos dessa dissertação estão organizados da seguinte forma: no capítulo 2 é apresentada uma fundamentação teórica expondo os principais conceitos sobre as normas IEC 61131 e IEC 61499, a Teoria de Controle Supervisório e alguns trabalhos relacionados. No capítulo 3 apresenta-se a metodologia desenvolvida no contexto desta tese para a implementação da estrutura de controle supervisório modular local aderente à IEC 61499. No capítulo 4 é apresentado um estudo de caso para validação da metodologia desenvolvida. Por fim, no capítulo 5 se têm as conclusões e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

A norma IEC 61131 (IEC, 2003) padroniza um conjunto de linguagens de programação para CLPs. Esse padrão é implementado pela maioria dos fabricantes de CLPs e tem grande aceitação na automação industrial moderna. No entanto, essa norma não garante a compatibilidade entre diferentes fabricantes, pois apesar de utilizarem a mesma linguagem de programação, os ambientes de desenvolvimentos geralmente não são compatíveis, sendo uma limitação dessa norma.

A norma IEC 61499 (IEC, 2013) define um novo modelo de bloco de funções, o qual é capaz de encapsular o código e possui uma execução orientada a eventos, trazendo vantagens aos usuários finais, como, por exemplo, reuso de soluções padronizadas e maior flexibilidade através da capacidade de adicionar componentes de forma *plug-and-play*.

Quando o espaço de estados de um sistema é naturalmente descrito por um conjunto discreto como $\{0, 1, 2, \ldots\}$ e as transições de estado são observadas apenas em pontos discretos no tempo, associam-se essas transições de estados a “eventos” e chama-se tal sistema de “sistema a eventos discretos” (CASSANDRAS; LAFORTUNE, 2007).

Na estrutura de controle monolítica todos os modelos que representam o comportamento da planta são compostos a fim de obter um modelo para o seu comportamento global e, a partir desse e de um modelo para a especificação global, é feita a síntese de um supervisor que é responsável por garantir o adequado funcionamento do sistema em malha fechada (CURY, 2001).

Na estrutura de controle modular local é explorada tanto a modularidade da planta como das especificações de controle. Com o uso dessa abordagem, obtêm-se supervisores modulares locais que garantem que o comportamento do sistema em malha fechada será não bloqueante, minimamente restritivo, e obedecerá a todas as especificações de controle. Além disso, tais supervisores podem ser implementados de forma distribuída, em um ou mais elementos de controle (QUEIROZ, 2004).

2.1 A NORMA IEC 61131

A norma IEC 61131 foi criada para uniformizar a programação de controladores lógicos programáveis (CLP) na indústria. A organização de um programa é feita através de POUs (*Program Organization Unit*), que são as menores unidades do programa. Existem três tipos de POUs (TIEGELKAMP; JOHN, 2010):

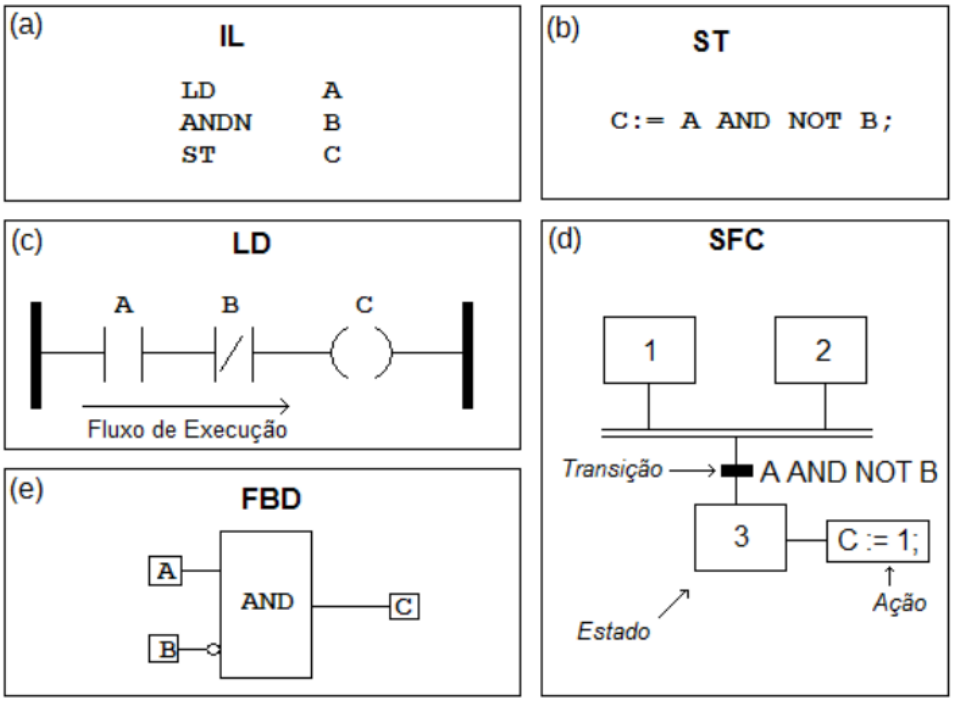
1. Function (FUN): funcão que sempre produz o mesmo resultado quando chamada com a mesma entrada (parâmetros), ela não tem nenhum mecanismo que recorde a execução anterior. A FUN pode fazer chamadas somente a outras FUNs;

2. Function Block (FB): A principal diferença de FB para FUN é que o FB não perde os dados da execução anterior, ele pode produzir saídas diferentes mesmo recebendo os mesmos parâmetros;

3. Program (PROG): Representa o maior nível na organização do programa. Além de ser capaz de chamar FBs e FUNs, o PROG tem acesso as entradas e saídas do dispositivo e pode conceder esse acesso as demais POUs.

As cinco linguagens definidas pela norma IEC 61131 são ilustradas pela figura 1 e estão descritas a seguir:

Figura 1 – As cinco linguagens da norma IEC 61131



fonte: adaptado de (TIEGELKAMP; JOHN, 2010)

1. Ladder Diagram (LD): se assemelha a um diagrama elétrico no qual uma entrada é representada por um símbolo de contato e uma saída é representada por um símbolo de bobina. Nessa linguagem, considera-se um fluxo de execução da esquerda para a direita;

2. Function Block Diagram (FBD): a construção do algoritmo é semelhante a construção de um diagrama elétrico utilizando componentes. Os componentes são na

verdade FBs ou FUNs que são representados em blocos, a esquerda do bloco são colocadas as entradas e a direita as saídas;

3. Sequential Function Chart (SFC): é semelhante a uma rede de Petri, na qual são utilizados estados e transições. Em cada estado pode-se ter uma ou mais ações a serem executadas e a mudança de estado se dá por transições que ocorrem mediante uma condições booleanas;

4. Instruction List (IL): se assemelha a uma linguagem de montagem, como, por exemplo, *Assembly*. Ela pode operar somente sobre as variáveis definidas dentro da POU, assim como todas as demais linguagens;

5. Structured Text (ST): é uma linguagem de texto. Segundo Wenger e Zoitl (2012), devido a ampla adoção industrial da IEC 61131-3, existe uma grande quantidade de bibliotecas em ST. Reutilizá-las para a IEC 61499 pode reduzir bastante o custo de mudança para o novo padrão e permite alavancar os investimentos existentes.

2.2 A NORMA IEC 61499

A norma IEC 61499 é considerada como o aprimoramento da próxima geração da engenharia de sistemas com CLPs (DAI; VYATKIN, 2009).

Essa norma é baseada numa estrutura dirigida a eventos, os quais são invocados apenas quando um evento chega a uma de suas entradas de evento, atualizando em conjunto algumas entradas de dados. Durante o restante do tempo, o bloco de função permanece ocioso, proporcionando menos consumo de energia ao sistema e menor necessidade de processamento. Além disso, utilizando os blocos de função da IEC 61499 é possível ter uma visão geral dos dispositivos e *layouts* de comunicação, e os projetos podem facilmente ser reutilizados para outras aplicações, diminuindo perdas de tempo por retrabalho.

A interoperabilidade é uma das características mais importantes que a norma IEC 61499 traz como vantagem em relação a sua antecessora IEC 61331. Em sistemas com automação distribuída, ter interoperabilidade significa sincronização de comportamentos, padronização de protocolos de comunicação e de formatos de dados (PATIL et al., 2013).

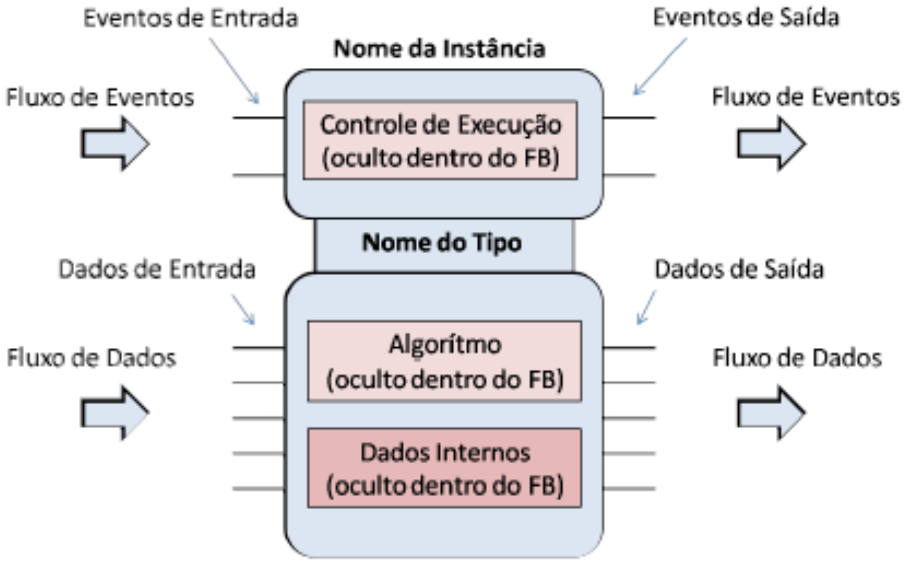
A máquina de estados acoplada aos blocos de função básicos da IEC 61499 fornece uma condição de fluxo de controle que remove a necessidade de implementar manualmente máquinas de estados mais complexas, como muitas vezes acontece na IEC 61131 (SHAW; ROOP; SALCIC, 2010).

Três principais características são definidas pela norma IEC 61499 para sistemas distribuídos: a capacidade de se editar e compilar o mesmo código e bibliotecas em diferentes ambientes de programação existentes (portabilidade), a capacidade de se configurar vários dispositivos de diferentes fabricantes com a mesma ferramenta de *software*, com mínima ou nenhuma modificação (configurabilidade) e a capacidade de sistemas e dispositivos de diferentes fabricantes trocarem informações e as utilizarem em conjunto (interoperabilidade) (PINTO, 2014). Para que tudo isso seja possível, a norma IEC 61499 estipula o uso da linguagem XML (eXtensible Markup Language) para representação dos FBs (LEWIS, 2008).

2.2.1 O Bloco de Função Básico

A base de toda a arquitetura da norma IEC 61499 consiste no modelo de bloco de função básico ilustrado na figura 2. Cada FB possui (VYATKIN, 2007):

Figura 2 – Características gerais de um Bloco de Função



fonte: (HARBS, 2012)

1. um nome do tipo e um nome de instância;

2. um conjunto de eventos de entrada;

3. um conjunto de eventos de saída;

4. um conjunto de entradas de dados;

5. um conjunto de saídas de dados;

6. um conjunto de variáveis internas;

7. um gráfico de execução de controle, chamado de Execution Chart Control (ECC), o qual possui o modelo de uma máquina de estados e é responsável por definir

quais algoritmos serão acionados com a chegada de eventos e quando os eventos de saída serão disparados. O ECC é muito importante na dinâmica de funcionamento de um bloco de função (LINDGREN et al., 2015).

A execução de um FB segue basicamente cinco passos (LEWIS, 2008), conforme descritos a seguir:

1. as variáveis de entrada de dados relevantes para o evento de entrada se tornam disponíveis;

2. o evento de entrada ocorre e o ECC do FB gera uma transição;

3. um ou mais algoritmos associados a um estado do ECC são executados;

4. o algoritmo gera as variáveis de saída de dados associadas com o evento de saída;

5. o ECC gera um ou mais eventos de saída.

A IEC 61499 define o bloco de função básico como a unidade básica de programação. Eles foram desenvolvidos a partir das funções da IEC 61131, mantendo os dados de entrada e saída assim como a implementação dos algoritmos internos de controle através das 5 linguagens de programação da IEC 61131, porém, adquirindo como novidade os eventos de entrada e saída controlados pelo ECC (HARBS, 2012).

FBs do tipo básico são estruturas de *software* projetadas para implementar funções básicas de controle, como cálculos simples, tratamento de eventos, entre outros (VYATKIN, 2007). O seu comportamento é definido em termos dos algoritmos que são invocados em resposta a eventos de entrada. Após a execução do algoritmo, eventos de saída são disparados.

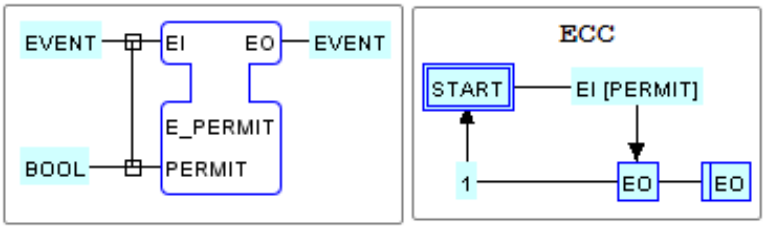
2.2.1.1 FB básico do tipo PERMIT

O FB básico do tipo PERMIT (*permissive event propagation function block*) definido na norma IEC 61499 (IEC, 2013) está ilustrado na figura 3.

De forma resumida, sua principal função é replicar o evento de entrada EI na saída de evento EO desde que o mesmo esteja associado a uma variável de dado $PERMIT$ com valor booleano $TRUE$.

Esse bloco de função será importante na implementação da metodologia proposta no capítulo três desse trabalho, pois há momentos nos quais serão necessários transformar variáveis de dados em eventos, como será explicado posteriormente.

Figura 3 – Bloco de Função Permit

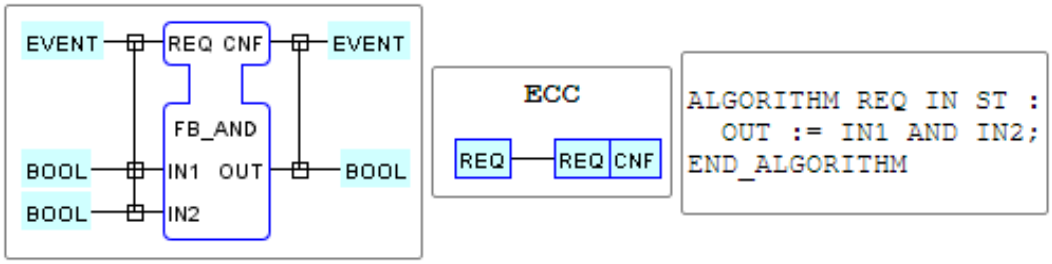


fonte: (HOLOBLOC, 2018)

2.2.1.2 FB básico do tipo AND

O FB do tipo AND é um bloco de função básico que tem como principal objetivo executar uma operação Booleana AND entre duas variáveis de dados de entrada, conforme ilustrado na figura 4.

Figura 4 – Bloco de função do tipo AND



fonte: (HOLOBLOC, 2018)

De forma resumida, seu funcionamento é gerar um valor booleano $TRUE$ na saída de dado OUT desde que as entradas de dados $IN1$ e $IN2$ também possuam valor booleano $TRUE$ simultaneamente e no mesmo instante que um evento de entrada REQ for recebido. Caso contrário, a saída de dado OUT permanecerá com o valor booleano $FALSE$.

Esse bloco de função será importante na implementação da metodologia proposta no capítulo três desse trabalho, pois será necessário em certos momentos que se faça uma operação booleana AND para todos os eventos controláveis σ que pertençam ao alfabeto de mais de um supervisor modular local, como será explicado posteriormente.

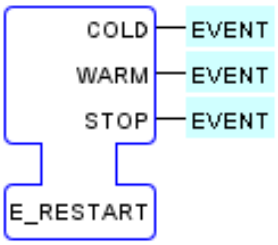
2.2.1.3 FB básico do tipo E_RESTART

O FB de inicialização é um bloco de função básico definido na norma IEC 61499 (IEC, 2013), o qual é comumente denotado pelo nome $E\_RESTART$. A sua função é gerar um evento de inicialização que passará por todos os eventos $INIT$ e $INITO$ da aplicação, executando os algoritmos internos de inicialização. Seu modelo está ilustrado na figura 5.

De forma resumida, ao iniciar a aplicação, o bloco de função $E\_RESTART$ irá, através de usa saída de eventos $COLD$, por padrão da norma, gerar o evento de inicialização.

Esse bloco de função será importante na implementação da metodologia proposta no capítulo três desse trabalho, pois é o responsável por inicializar qualquer aplicação implementada.

Figura 5 – Bloco de função de inicialização



fonte: (HOLOBLOC, 2018)

2.2.2 O Bloco de Função Composto

A funcionalidade do bloco de função composto (CFB) é determinada por uma rede de blocos interconectados. Um FB composto tem sua estrutura ilustrada na figura 6.

Ao contrário dos FBs básicos, o CFB não possui função de controle de execução (ECC). A lógica particular de chamada dos FBs na rede de blocos dentro do CFB pode ser implementada usando um bloco de função externo do tipo básico, o qual poderá emitir eventos para ativar os blocos da rede.

O CFB se destina a ser o principal instrumento de um desenvolvedor de aplicativos (VYATKIN, 2007).

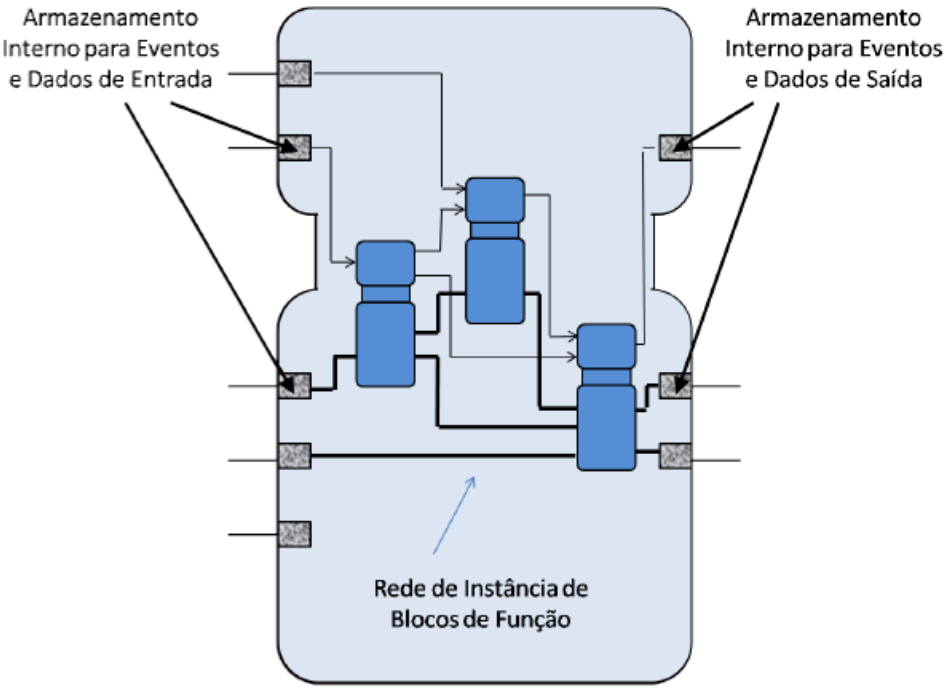
2.2.3 O Bloco de Função de Interface de Serviço

O bloco de função de interface de serviço (SIFB) permite a abstração de serviços específicos para uma plataforma, tais como leitura de entradas, escrita nas saídas ou serviços de comunicação.

Para que haja comunicação dos blocos de função dentro de um recurso e o mundo externo ou entre recursos há a exigência de um bloco de função de interface de serviço (LEWIS, 2008).

O SIFB serve como uma função que empacota as dependências do *hardware* e permite ao desenvolvedor focar na lógica da aplicação. Sua implementação requer conhecimento de baixo nível do *hardware* que será utilizado (PINTO, 2014).

Figura 6 – Bloco de função composto

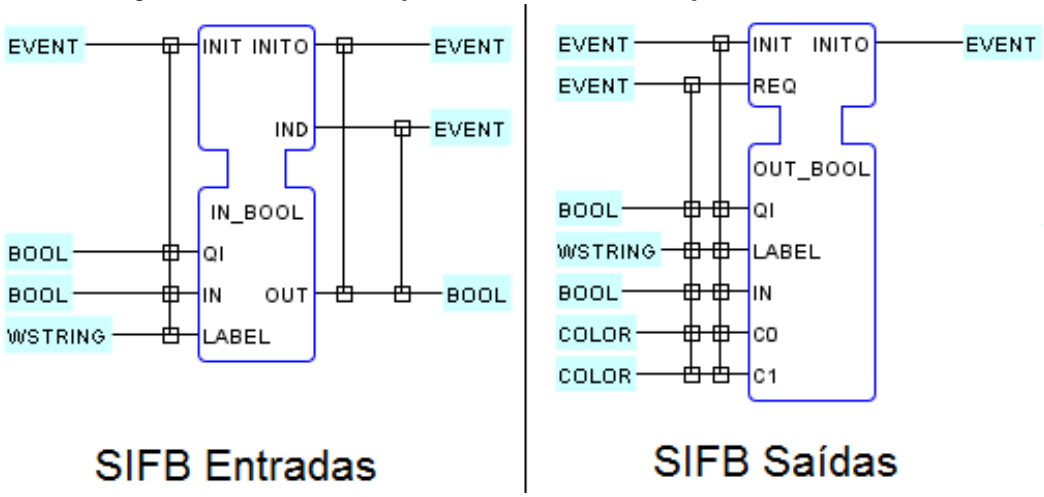


fonte: (HARBS, 2012)

Assim, os SIFBs não devem ser desenvolvidos pelos mesmos desenvolvedores da aplicação, mas sim fornecidos pelos próprios fabricantes do equipamento (VYATKIN, 2007).

A figura 7 exemplifica dois modelos de blocos de função de interface de serviço, um para leitura de entradas e outro para escrita nas saídas. Os eventos e variáveis de dados estão descritos a seguir:

Figura 7 – Blocos de função de interface de serviço de entrada e saída



fonte: do próprio autor

1. Evento de entrada INIT: inicializa o serviço do bloco de função ao receber

um evento;

2. Evento de saída INITO: indica que a inicialização do bloco de função foi completada;

3. Evento de entrada REQ: solicita um evento para executar o serviço fornecido pelo SIFB;

4. Evento de saída IND: esse evento indica que uma operação de leitura foi concluída;

5. Dado de entrada QI: se essa entrada é verdadeira durante a ocorrência do evento INIT, a inicialização do serviço é requisitada. Caso contrário, a finalização do serviço é requisitada;

6. Dado de entrada IN: dado de entrada referente à uma entrada física do *hardware*;

7. Dado de entrada LABEL: nomeia a variável de entrada na interface IHM;

8. Dado de entrada C0: define uma cor para a variável na interface IHM quando a mesma possuir valor booleano TRUE;

9. Dado de entrada C1: define uma cor para a variável na interface IHM quando a mesma possuir valor booleano FALSE;

10. Dado de saída OUT: envia um dado booleano resultante da execução de um algoritmo interno para outro bloco de função.

2.2.4 Modelo de Aplicação

Uma aplicação é uma rede de FBs interconectados por eventos e variáveis de dados (VYATKIN, 2007) que define completamente a funcionalidade desejada de um sistema, como, por exemplo, o código de controle implementado numa célula de manufatura que precisa produzir uma peça.

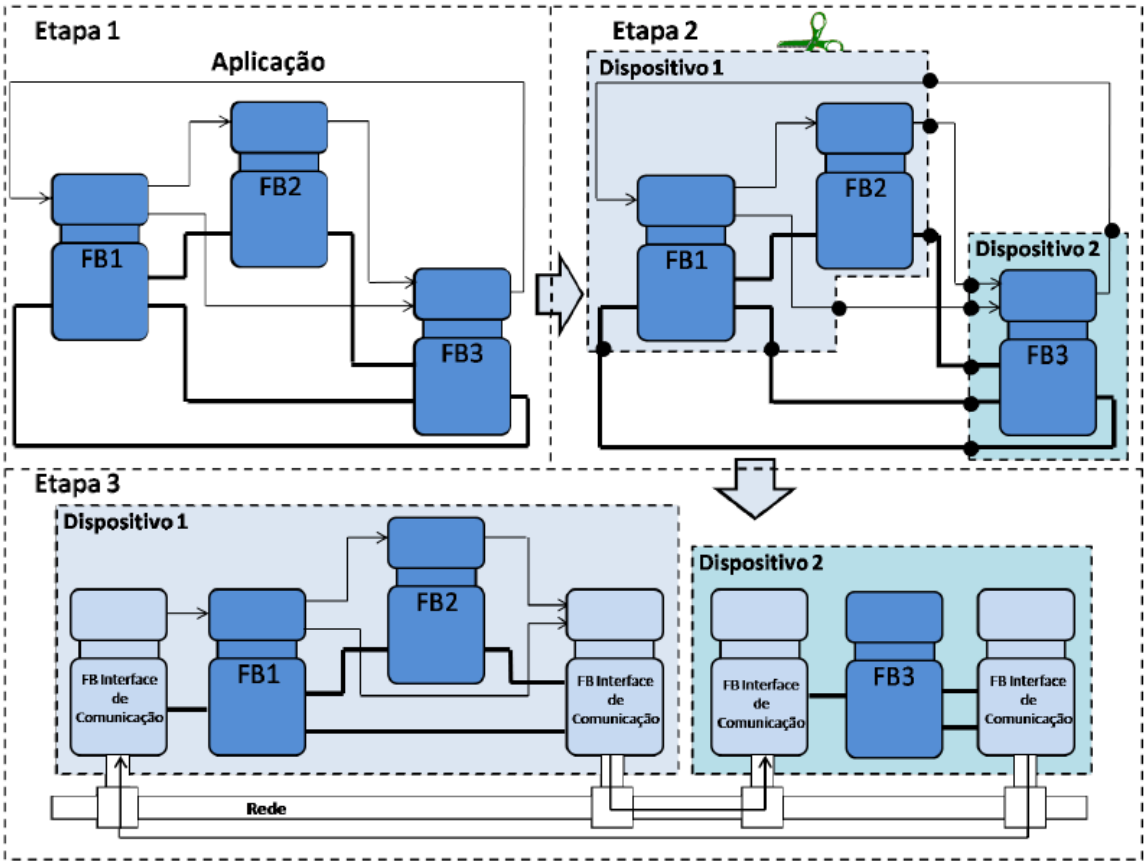
Uma aplicação pode existir em um único dispositivo ou possuir funcionalidades distribuídas em diversos dispositivos.

Na figura 8 é ilustrado um exemplo de aplicação.

2.2.5 Modelo de Recurso

Recurso é uma unidade funcional contida dentro de um dispositivo que provê vários serviços para as aplicações, como, por exemplo, sequenciamento e execução de algoritmos (LEWIS, 2008).

Figura 8 – Modelo de Aplicação



fonte: (HARBS, 2012)

Cada recurso possui controle independente de suas operações. Isto significa que um recurso pode ser criado, configurado, parametrizado, inicializado, apagado, etc., sem afetar outros recursos dentro do mesmo dispositivo (VYATKIN, 2007).

As principais características de um modelo de recurso são mostradas na figura 9.

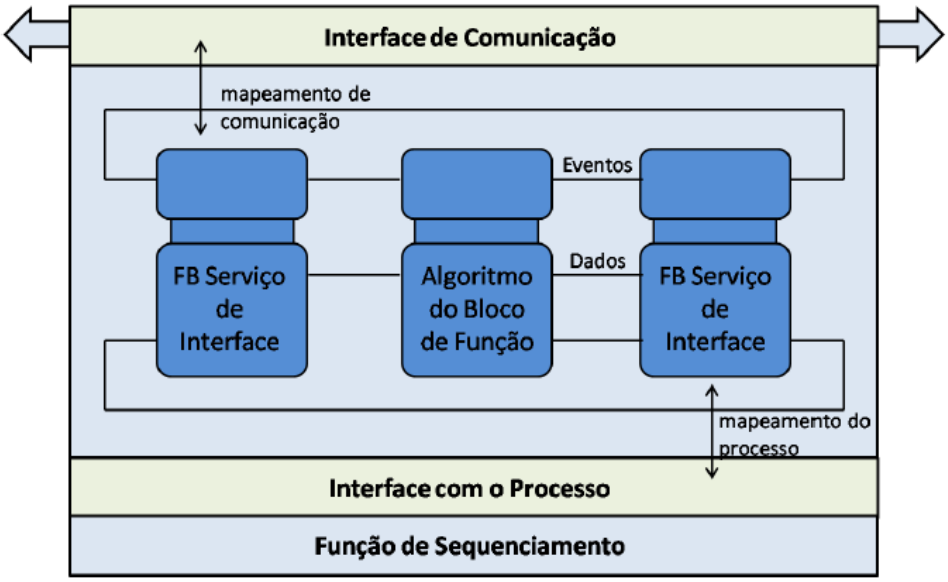
2.2.6 Modelo do Dispositivo

Um dispositivo representa uma unidade de controle, o qual pode possuir zero ou mais recursos, juntos com interfaces de processos e interfaces de comunicação (VYATKIN, 2007).

A interface de processo de um dispositivo provê serviços que permitem que recursos troquem dados com entradas e saídas físicas do dispositivo e a interface de comunicação provê serviços que permitem que recursos troquem dados com outros dispositivos (LEWIS, 2008).

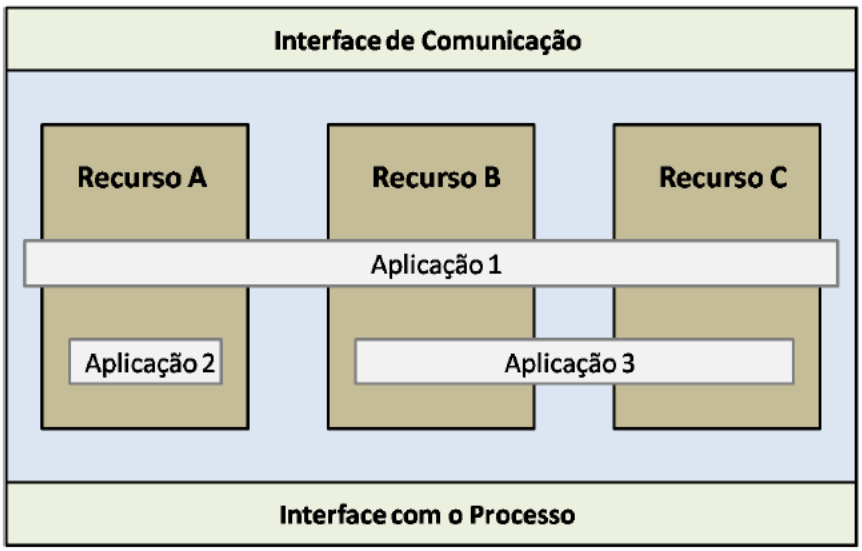
A figura 10 ilustra um modelo de dispositivo.

Figura 9 – Modelo de Recurso



fonte: (HARBS, 2012)

Figura 10 – Modelo de Dispositivo



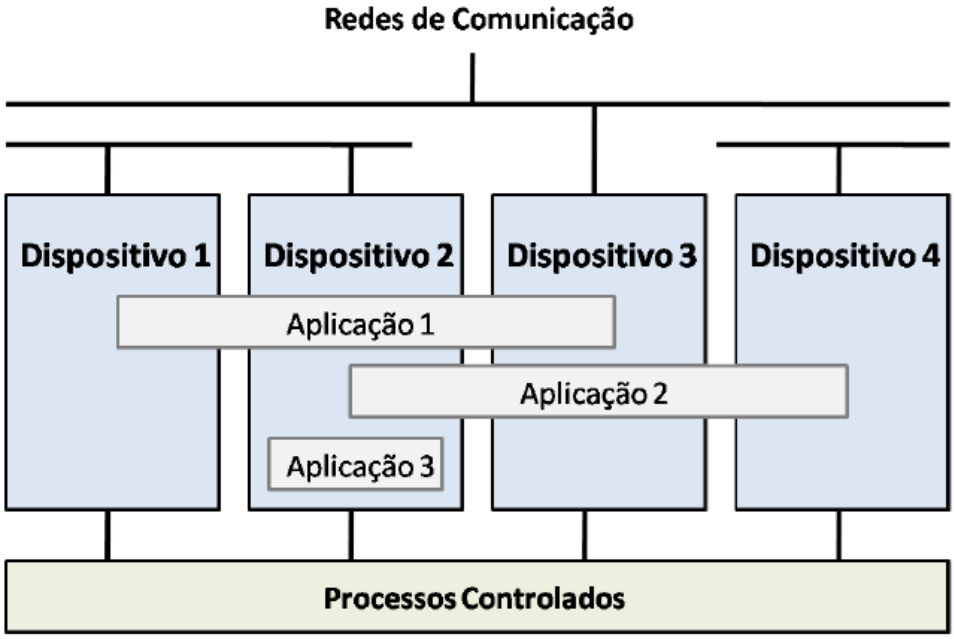
fonte: (HARBS, 2012)

2.2.7 Modelo de Sistema

O sistema representa a estrutura mais complexa da arquitetura da IEC 61499 e é definida como um conjunto de dispositivos, nos quais os FBs das aplicações são alocados (LEWIS, 2008).

A figura 11 ilustra o conceito geral de um sistema distribuído genérico da IEC 61499, em que os dispositivos podem comunicar-se uns com os outros sobre um ou mais links de comunicação e possíveis interfaces com máquinas e processos a controlar.

Figura 11 – Modelo de Sistema



fonte: adaptado de (HARBS, 2012)

2.2.8 Ambientes de programação aderentes à IEC 61499

Todas as operações relatadas anteriormente podem ser executadas com o auxílio de alguma ferramenta computacional para implementação do código baseado na norma IEC 61499. As principais ferramentas são listadas a seguir:

1. FBE: essa ferramenta fornece uma interface gráfica que permite criar, visualizar e editar um sistema implementado através de blocos de funções. O sistema é armazenado como um arquivo XML, conforme definido pela norma IEC 61499. Para implementar ou alterar os algoritmos associados aos blocos de funções é aberta uma janela para edição de textos. A interface do FBE foi implementada usando a linguagem Lua. Nesse editor também é possível simular a execução dos blocos de funções para depuração. Essa ferramenta é gratuita e foi desenvolvida pelo Grupo de Automação de Sistemas e Robótica do Centro de Ciências Tecnológicas da Universidade do Estado

de Santa Catarina, porém atualmente se encontra desatualizado em relação a última versão da norma IEC 61499 (PINTO et al., 2015);

2. 4DIAC: esse ambiente de desenvolvimento integrado é um ambiente de engenharia extensível, compatível com a IEC 61499 para aplicações de controle distribuído. As aplicações modeladas podem ser baixadas para dispositivos de campo distribuídos de acordo com os meios definidos pela norma IEC 61499 (STRASSER; ROOKER; EBENHOFER, 2008).

O ambiente de execução 4DIAC-RTE é uma pequena implementação portátil de um ambiente de execução IEC 61499 voltado para pequenos dispositivos de controle embarcados (16/32 Bit), implementados em C++. Ele suporta a reconfiguração online de suas aplicações e a execução com capacidade em tempo real de todos os tipos de blocos de funções fornecidos pelo padrão IEC 61499 (PANG; PATIL; YANG, 2014). Essa ferramenta é de código aberto;

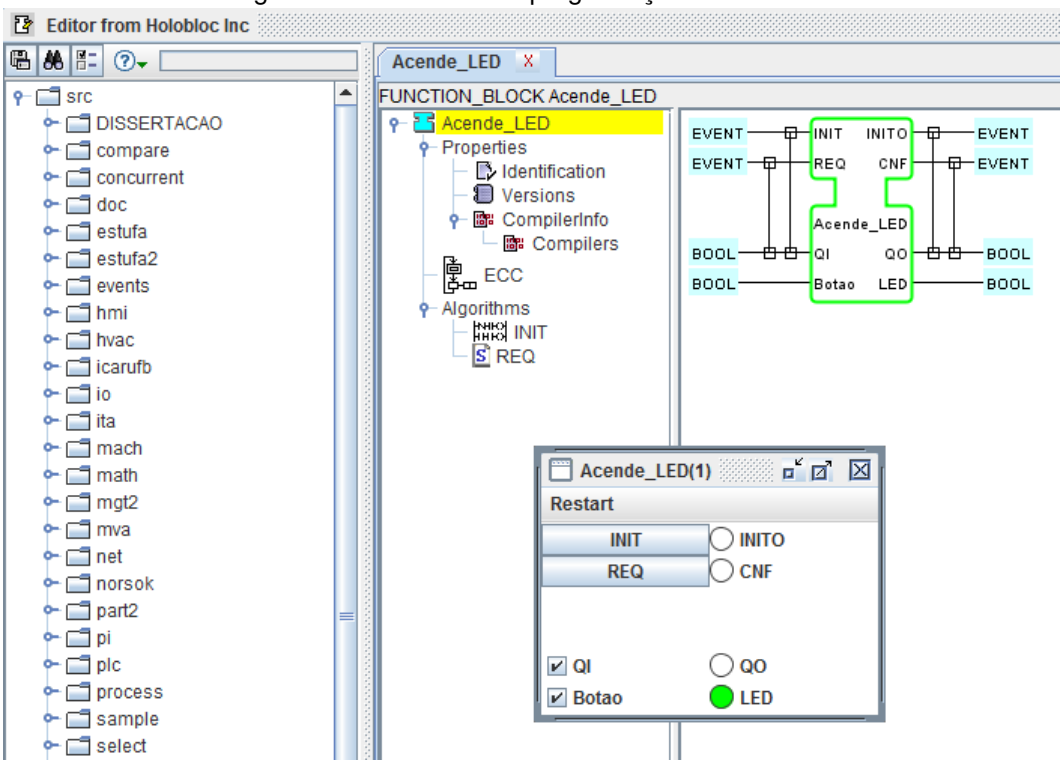
3. ISaGRAF: a *Rockwell Automation* lançou em 2005 a versão 5 de seu amplamente conhecido e utilizado ISaGRAF Workbench para programação IEC 61131 com suporte para IEC 61499, tornando-se assim a primeira ferramenta de *software* comercial com suporte à IEC 61499. Atualmente, a ISaGRAF está trabalhando na versão 6 de seu *workbench*, construído no *Microsoft Visual Studio*, que permite extensões de avaliação por fornecedores de *hardware* e máquinas (VYATKIN; CHOUINARD, 2008);

4. nxtSTUDIO: este ambiente de desenvolvimento IEC 61499, comercialmente suportado, foi introduzido em 2008 pela empresa austríaca nxtControl. Atualmente na versão 1.4, é usado por um grande número de fornecedores de dispositivos e máquinas da automação predial e de manufatura. Uma característica interessante é o uso de *Compound Automation Types (CATs)*, que incluem engenharia de controle via IEC 61499, visualização HMI/ SCADA, incluindo símbolos, diálogos operacionais, interconexão de entradas/saídas específicas de *hardware* e documentação (CHRISTENSEN et al., 2012). O nxtSTUDIO possui um ambiente gráfico para simulação bem desenvolvido;

5. FBDK/FBRT: o *"Function Block Development Kit"* é uma ferramenta de *software* amplamente usada em projetos relacionados com a IEC 61499. A ferramenta permite o desenvolvimento gráfico de blocos de função e de sistemas baseados em blocos de função. O FBDK compila os tipos de blocos de função desenvolvidos em linguagem de programação Java. A natureza orientada a objetos desta linguagem permite a implementação consistente de blocos de função. Além disso, a independência de plataforma da linguagem Java leva à portabilidade do controlador (PANG; PATIL; YANG, 2014).

O uso de FBDK pode ser combinado com outras ferramentas de desenvolvi-

Figura 12 – Ambiente de programação FBDK/FBRT



fonte: do próprio autor

mento Java, por exemplo, ambiente de desenvolvimento integrado Eclipse (FOUNDATION, 2018), especialmente quando o código Java é sofisticado, como por exemplo na visualização em 3D ou interface de dispositivos periféricos que precisam ser encapsulados em blocos de função. As aplicações do bloco de funções, projetados usando FBDK, são executados com a máquina virtual Java e o ambiente de execução FBRT (*Function Block Run Time*) dos elementos da biblioteca (PANG; PATIL; YANG, 2014).

O FBDK foi desenvolvido pela HOLOBLOC Inc, uma corporação com fins lucrativos do estado de Ohio, EUA, que começou a operar em abril de 2005, para fornecer suporte, consultoria e treinamento para a norma IEC 61499 (HOLOBLOC, 2018).

A ferramenta FBDK/FBRT foi importante nesse trabalho, pois todas as simulações com os blocos de função da IEC 61499 foram executadas nele. Apesar dele possuir uma interface simples com algumas limitações, atendeu de forma funcional. A figura 12 ilustra o ambiente de programação do FBDK juntamente com a interface gráfica de simulação FBRT. O FBRT nada mais é que uma interface IHM desenvolvida para simular as aplicações desenvolvidas no FBDK através de elementos gráficos como LEDs, esteiras, caixas de texto, botões, etc, conforme exemplo $Acede_LED(1)$ ilustrado na figura 12.

2.3 SISTEMAS A EVENTOS DISCRETOS

Em (CASSANDRAS; LAFORTUNE, 2007) é realizada a classificação dos sistemas em diversas categorias. Os sistemas denominados “sistemas contínuos” se caracterizam basicamente por dois fatores:

1. o espaço de estados é contínuo, isto é, as variáveis do sistema podem assumir qualquer valor dentro de um determinado intervalo de variação contínuo;

2. o comportamento das variáveis do sistema é regido pelo tempo.

De forma geral, o formalismo matemático para estudo dos sistemas contínuos se baseia em equações diferencias.

Em contraposição aos “sistemas contínuos”, os sistemas denominados “sistemas a eventos discretos” (SEDs) apresentam as seguintes características:

1. o espaço de estados é discreto, ou seja, as variáveis do sistema podem assumir valores preestabelecidos pertencentes a um conjunto discreto;

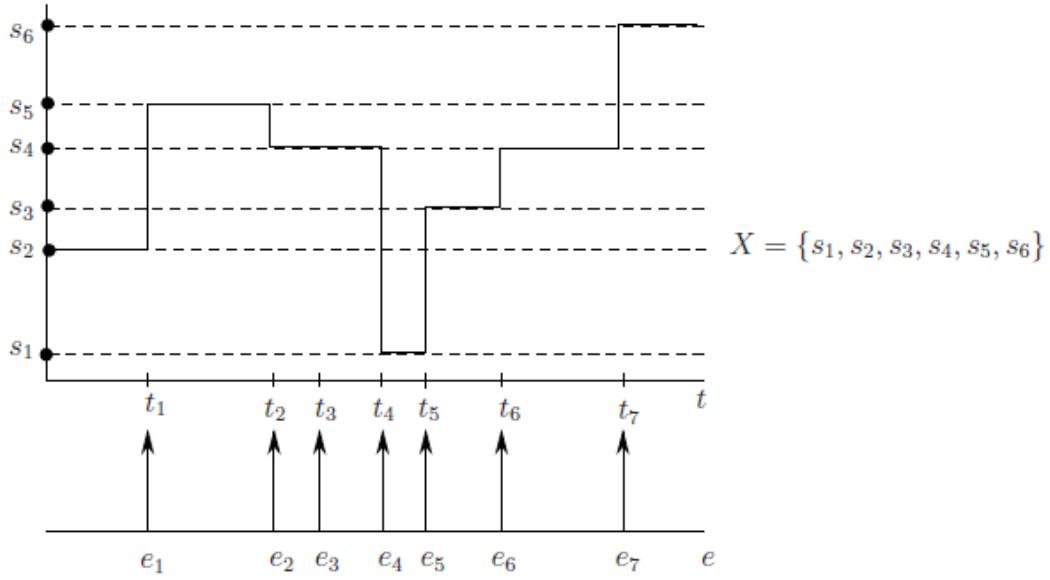
2. o comportamento das variáveis independe do tempo e é dirigido por eventos.

A figura 13 ilustra um exemplo da dinâmica de um sistema a eventos discretos. Nesta trajetória ocorrem eventos representados por e_{i}, os quais fazem o sistema ir para um estado específico representado por s_{i}. Pode-se notar que os eventos ocorrem sempre instantaneamente num instante representando por t_{i} (CASSANDRAS; LAFORTUNE, 2007).

De acordo com Cury (2001) há vários modelos desenvolvidos para SEDs, mas nenhum deles é tido como um modelo universal. Isso se deve ao fato de que esses modelos refletem diferentes objetivos de modelagem (análise, controle, otimização, etc.). Os principais modelos são Redes de Petri controladas com e sem temporização e a Teoria de Linguagens e Autômatos, as quais são capazes de realizar a síntese de controladores.

Neste trabalho é de particular interesse a modelagem de SEDs pela Teoria de Linguagens e Autômatos (RAMADGE; WONHAM, 1989), pois a mesma se mostra adequada para a síntese de controladores para SEDs.

Figura 13 – Trajetória de estados de um SED



fonte: (CASSANDRAS; LAFORTUNE, 2007)

2.4 TEORIA DE LINGUAGENS

Um alfabeto é um conjunto finito Σ de símbolos distintos. O conjunto Σ^* é formado por todas as cadeias finitas de símbolos, também chamadas de sequências ou palavras, da forma $\sigma_1\sigma_2...\sigma_k$, com $\sigma_i \in \Sigma$, onde $i \in 1, 2, ...$, que podem ser formados a partir de Σ (PINOTTI, 2012).

Sendo uma linguagem um conjunto de palavras, as propriedades matemáticas e operações da teoria de conjuntos são aplicáveis (CRUZ, 2011), como as descritas a seguir:

1. Concatenação: sejam duas linguagens L_1, $L_2 \subseteq \Sigma^*$, então a concatenação de L_1 e L_2, denotado L_1L_2, é definida por (CURY, 2001):

$$L_1L_2 := \{s \in \Sigma^* : (s = s_1s_2) \text{ e } (s_1 \in L_1) \text{ e } (s_2 \in L_2)\}$$

2. Prefixo-Fechamento: Seja uma linguagem $L \in \Sigma^*$, então, o prefixo-fechamento de L, denotado por $\overline{L}$, é definido por (CURY, 2001):

$$\overline{L} := \{s \in \Sigma^* : \exists t \in \Sigma^*(st \in L)\}$$

2.5 REPRESENTAÇÃO DE SEDS POR LINGUAGENS

A linguagem $L \subseteq \Sigma^*$, que recebe o nome de linguagem gerada, é uma linguagem prefixo-fechada que descreve o conjunto de eventos fisicamente possíveis de ocorrer na planta. Já a linguagem $L_m \subseteq L$, que recebe o nome de linguagem marcada, representa o conjunto de eventos que levam o sistema a completar tarefa.

Duas propriedades das linguagens são utilizadas de forma a representar um SED (CURY, 2001):

1. $L \subset L_m$, ou seja, o comportamento gerado contém o comportamento marcado de um SED;

2. $L = \overline{L}$, ou seja, o comportamento gerado de um SED é prefixo-fechado.

2.6 AUTÔMATOS COMO MODELOS PARA SISTEMAS A EVENTOS DISCRETOS

Um autômato determinístico de estados finitos (ADEF) é uma quíntupla G = (X, Σ, δ, $x0$, Xm), onde (CASSANDRAS; LAFORTUNE, 2007):

1. X é o conjunto finito de estados do autômato;

2. Σ é o conjunto de eventos que definem o alfabeto;

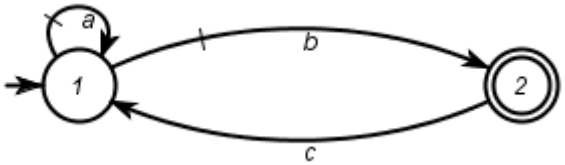
3. $\delta : X\ x\ \Sigma -> X$ é a função de transição;

4. $x0$ é o estado inicial do autômato;

5. Xm é o conjunto de estados marcados.

A figura 14 representa graficamente um autômato, na qual os círculos representam os estados e os arcos etiquetados representam as transições. O estado inicial é identificado através de uma seta apontando para ele e os estados finais são representados com círculos duplos. O autômato da figura 14 pode ser descrito da seguinte forma:

Figura 14 – Exemplo gráfico de um autômato



fonte: do próprio autor

1. X = {1,2};
2. Σ = {a,b,c};
3. A função de transição é $f(1,a) = 1$, $f(1,b) = 2$ e $f(2,c) = 1$;
4. $x0$ = {1};
5. Xm = {2}.

2.7 OPERAÇÕES COM AUTÔMATOS

A composição síncrona entre autômatos é uma operação importante para cálculo e obtenção dos supervisores. Dados dois autômatos G_1 = (X_1,Σ_1,f_1,x_{0_1},X_{m_1}) e G_2 = (X_2,Σ_2,f_2,x_{0_2},X_{m_2}), define-se a composição síncrona $G1 \parallel G2$ como (CURY, 2001):

$$G_1 \parallel G_2 = (X_1 \text{ x } X_2, \Sigma_1 \cup \Sigma_2, f_{1\parallel 2}, (x_{0_1}, x_{0_2}), X_{m_1} \text{ x } X_{m_2})$$

onde

$$f_{1\parallel 2} : (X_1 \text{ x } X_2) \text{ x } (\Sigma_1 \cup \Sigma_2) \text{ -> } (X_1 \text{ x } X_2)$$

2.8 TEORIA DE CONTROLE SUPERVISÓRIO DE SISTEMAS A EVENTOS DISCRETOS

Um sistema a eventos discretos é um conjunto de subsistemas, que, em conjunto, definem as ações de uma planta industrial, denotada por aqui por G.

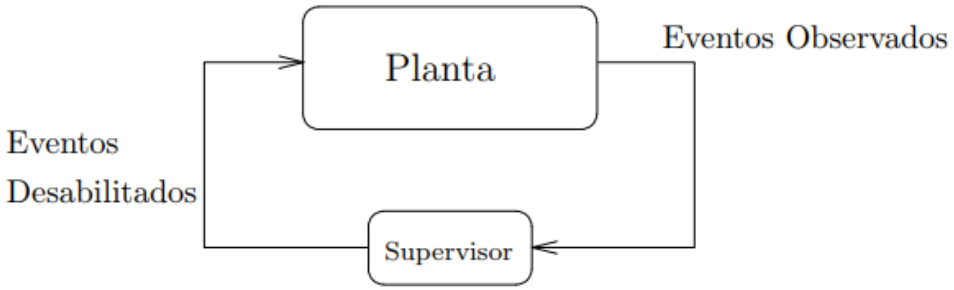
Para que a planta não opere de forma aleatória ou gere situações de perigo, um conjunto de especificações de controle E é definido. Por exemplo, é necessário que um torno não comece a operar sem que tenha uma peça nele.

Para que cada subsistema atue de forma correta, é calculado um agente de controle S, denominado supervisor, o qual atuará sobre a planta G numa estrutura de malha fechada, definido quais eventos são permissíveis de acontecer, conforme ilustrado na figura 15.

2.8.1 A Abordagem Modular Local

A abordagem modular local traz algumas vantagens sobre a abordagem monolítica apresentada anteriormente, como menor necessidade de poder de processamento computacional e maior facilidade para implementação do código de controle.

Figura 15 – Ação de controle de um Supervisor Monolítico



Fonte: (CURY, 2001)

Para isso, é calculado um supervisor modular local para cada planta modular local do sistema a ser implementado.

As etapas para obtenção dos supervisores modulares locais podem ser descritas pelos seguintes procedimentos (QUEIROZ, 2004):

1. obter o autômato para cada subsistema do sistema produto;

2. fazer a composição síncrona dos subsistemas síncronos;

3. modelar cada especificação isoladamente;

4. obter a planta local para cada especificação compondo-se os subsistemas que tenham eventos em comum com as respectivas especificações;

5. calcular as linguagens através do composição síncrona de cada planta local com sua respectiva especificação;

6. calcular a máxima linguagem controlável contida em cada especificação local;

7. realizar o teste de não conflito;

8. implementar um supervisor para cada linguagem controlável.

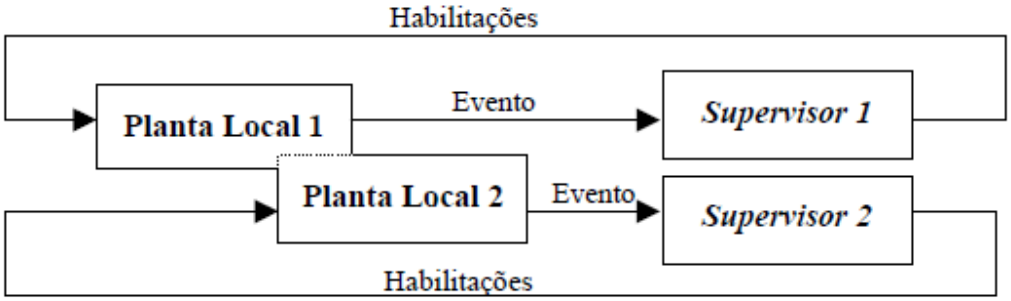
A figura 16 ilustra a estrutura de controle modular local para dois supervisores.

2.8.2 Ferramentas Computacionais para Síntese de Supervisores

Todas as operações relatadas anteriormente podem ser executadas com o auxílio de alguma ferramenta computacional para modelagem de SEDs e síntese de supervisores. Dentre as várias ferramentas existentes, podem ser citadas:

1. Nadzoru: essa ferramenta foi desenvolvida na Universidade do Estado de Santa Catarina e possui uma interface gráfica amigável para a criação de modelos e,

Figura 16 – Abordagem modular local



fonte: (QUEIROZ, 2004)

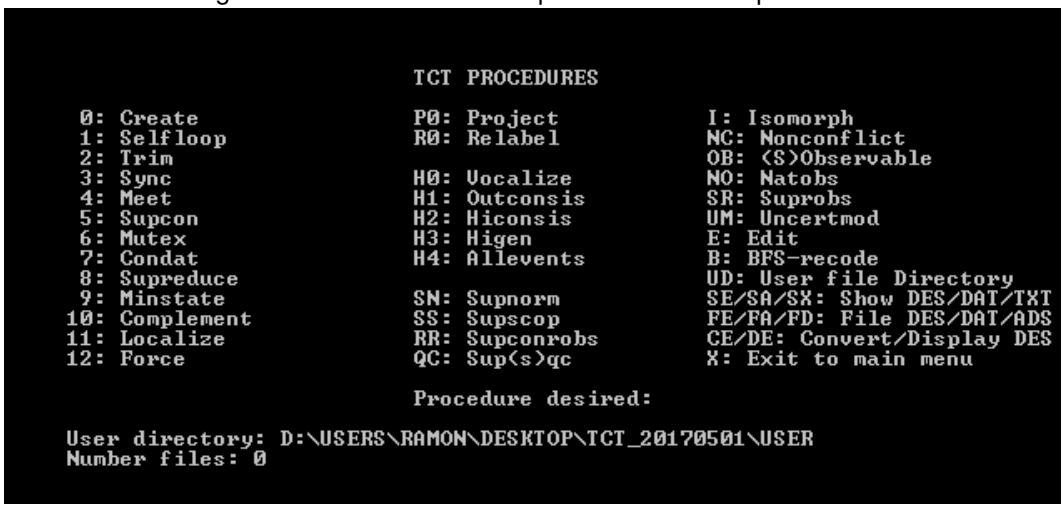
entre outras funcionalidades, permite a simulação de autômatos, análise de problemas relacionados à implementação da lógica de controle e geração de código para vários dispositivos. Atualmente a ferramenta é capaz de gerar código para alguns microcontroladores e controladores lógico programáveis. Outra característica da ferramenta é a capacidade de ser facilmente estendida com novos algoritmos e módulos (PINHEIRO et al., 2015);

2. Supremica: para facilitar o desenvolvimento de modelos grandes, um tipo de autômato foi introduzido nessa ferramenta. Este novo tipo de autômato é chamado de autômato finito expandido, o qual possui fórmulas de guarda e ação associadas as transições. O Supremica possui uma interface de usuário que é um ambiente de desenvolvimento integrado completo, o qual contém um editor gráfico de autômatos, além de uma interface fácil para análise, síntese e simulação dos supervisores obtidos. Também é possível gerar código que implemente o comportamento do modelo usando tanto o padrão IEC 61131 quanto o padrão IEC 61499 (AKESSON et al., 2006);

3. UltraDES: essa ferramenta é uma biblioteca de funções e estruturas de dados para análise e controle de Sistemas a Eventos Discretos baseados em .NET Framework. O objetivo principal é criar um ambiente para a implementação de algoritmos para Sistemas a Eventos Discretos, bem como a integração destes algoritmos com códigos das áreas de TI (Tecnologia da Informação) e AT (Tecnologia de Automação). Essa ferramenta foi desenvolvida no Departamento de Engenharia e Eletrônica da Universidade Federal de Minas Gerais e possui seu código aberto (ALVES; MARTINS; PENA, 2017).

4. TCT: o precursor de ferramentas para SEDs, foi desenvolvido pela equipe do professor Wonham no Departamento de Engenharia Elétrica e de Computação da Universidade de Toronto. Tem uma interface textual básica, porém possui um dos conjuntos mais completos e eficientes de algoritmos para operações sobre SEDs. Inclui operações para autômatos e síntese de supervisores. A última versão do TCT foi lançada em 2017 (WONHAM, 2018).

Figura 17 – Ferramenta TCT para síntese de supervisores

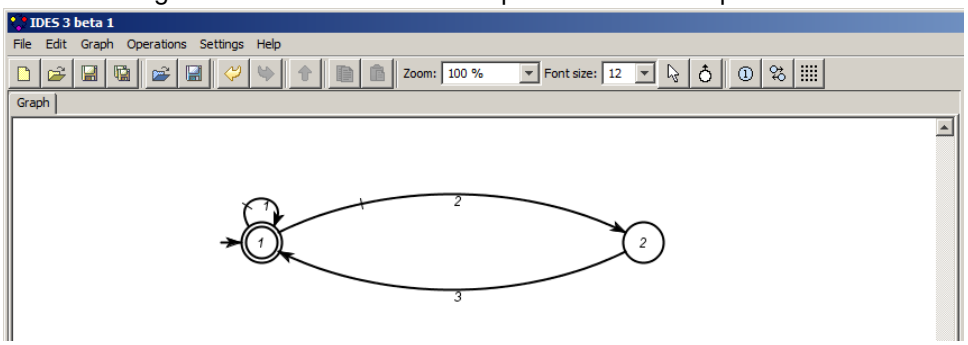


fonte: do próprio autor

Essa ferramenta foi importante nessa dissertação de mestrado por possuir um conjunto funcional de algoritmos para redução dos supervisores modulares locais, conforme será explicado posteriormente ainda nesse capítulo. A figura 17 ilustra a interface textual do TCT. Pode-se notar que todos os comandos são textuais, como, por exemplo, o comando $3 : Sync$, o qual é responsável por realizar a operação de sincronização entre autômatos;

5. IDES: desenvolvido na Queen's University pela equipe de Karen Rudie, tem uma interface muito bem concebida, especialmente para a entrada gráfica de autômatos. Além disso, inclui um conjunto de operações para autômatos e síntese de supervisores. No entanto, não é uma ferramenta de código aberto, o que é um obstáculo para a comunidade acadêmica poder adicionar novos algoritmos, por exemplo, para síntese de supervisores reduzidos (RUDIE, 2006).

Figura 18 – Ferramenta IDES3 para síntese de supervisores



fonte: do próprio autor

Essa ferramenta foi importante nessa dissertação de mestrado por possuir uma interface amigável para síntese dos supervisores e para obtenção de imagens aqui ilustradas. A figura 18 ilustra a interface gráfica do IDES3. Pode-se notar no

cabeçalho da ferramenta os vários comandos para criação de modelos gráficos de autômatos e o exemplo de um autômato modelado com dois estados e três eventos.

2.8.3 Redução dos Supervisores Modulares Locais

Na teoria de controle supervisório, dependendo da aplicação, os supervisores podem possuir um grande número de estados e transições, o que pode dificultar na implementação do código de controle e prejudicar o desempenho do sistema devido a uma maior necessidade de processamento e memória. Su e Wonham (2004) apresentam um algoritmo para redução significativa no número de estados e transições de supervisores, preservando todas as ações de controle.

Pode-se considerar que dois supervisores são equivalentes se os dois produzem os mesmos efeitos sobre uma planta G. Considerando dois supervisores S1 e S2, seria o equivalente a dizer que $L(S_1 \parallel G) = L(S_2 \parallel G)$ e $L_m(S_1 \parallel G) = L(S_2 \parallel G)$. Logo, S_2 é um supervisor reduzido para S_1 se ambos forem equivalentes e se o número de estados de S_1 for menor ou igual a S_2.

Os supervisores reduzidos podem ser obtidos através de duas operações resultantes na ferramenta TCT, conforme ilustrado na figura 17. Usa-se primeiro o comando 7-Condat com as seguintes informações: $DES1 = Gloc\_j$ (coloca-se o nome da planta local relacionada com o supervisor a ser reduzido), $DES2 = S\_j$ (coloca-se o nome do supervisor a ser reduzido) e $DES3 = Data\_j$ (variável com os dados que serão usados no próximo comando). Após o comando anterior, agora utiliza-se o comando 8-Supreduce com os seguintes dados: $DES1 = Gloc\_j$, $DES2 = S\_j$, $DAT2 = Data\_j$ (nome do arquivo escolhido no comando anterior) e $DES3 = RS\_j$ (nome do supervisor reduzido).

2.8.4 Um exemplo ilustrativo

De forma a exemplificar a abordagem modular local e também para auxiliar na explicação da metodologia proposta no capítulo três desta dissertação, utilizou-se um exemplo ilustrativo de uma linha de transferência industrial com três máquinas e dois buffers (QUEIROZ, 2000), conforme ilustrado na Figura 19.

Considera-se que o comportamento básico de cada máquina, sem considerar a possibilidade de quebra, é o seguinte: ao receber um sinal de comando a_c, a máquina M_i carrega uma peça e realiza uma operação sobre a mesma. Ao finalizar a operação (evento b_u) a máquina descarrega automaticamente a peça. Também é considerado que cada máquina ou buffer tem capacidade para apenas um única peça, não havendo possibilidade de sobrecarga (*overflow*) ou operação sem peça (*underflow*).

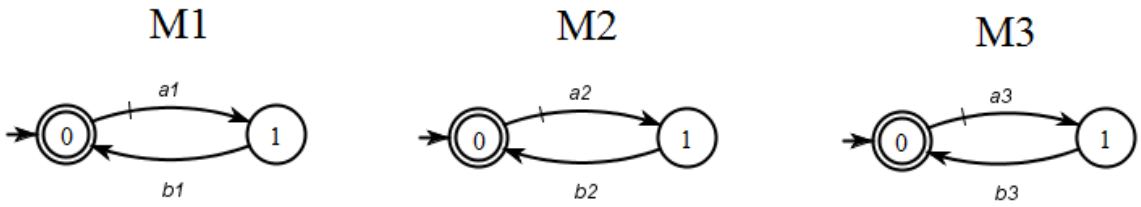
Através da teoria de controle supervisório modular local foram obtidos os mo-

Figura 19 – Linha de transferência industrial

fonte: do próprio autor

delos dos subsistemas, ou seja, das máquinas M_1, M_2 e M_3, conforme apresentados na figura 20.

Figura 20 – Modelos dos subsistemas



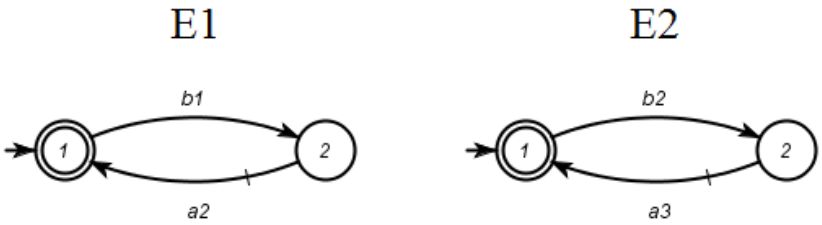
fonte: do próprio autor

Pode-se notar que o modelo do autômato é o mesmo para as três máquinas. O estado 0 é tanto o estado inicial como um estado marcado, ou seja, que representa que uma tarefa foi completada, nesse caso, a operação de uma peça. Do estado 0 para o estado 1 existe o evento controlável a_c, o qual indica início de operação da máquina M_i. O modelo do autômato indica que no estado 1 a máquina M_i está em operação. Do estado 1 para o estado 0 existe o evento não controlável b_u, o qual indica fim de operação da máquina M_i.

Os modelos das especificações de controle, as quais são responsáveis por não permitir que aconteçam situações proibitivas, nesse caso *overflow* ou *underflow*, foram modelados e estão ilustrados na figura 21.

Pode-se notar que o modelo do autômato é o mesmo para as duas especificações de controle. Do estado 1 para o estado 2 existe o evento não controlável b_u. Do estado 2 para o estado 1 existe o evento controlável a_{c+1}. No estado 1, esse modelo de autômato indica que está proibida a ocorrência do evento controlável a_{c+1}, que indica início da máquina M_{i+1}, sem que antes tenha ocorrido o evento não controlável b_u, que indica fim da máquina M_i, não permitindo que aconteça underflow da máquina M_{i+1}.

Figura 21 – Modelos das especificações de controle

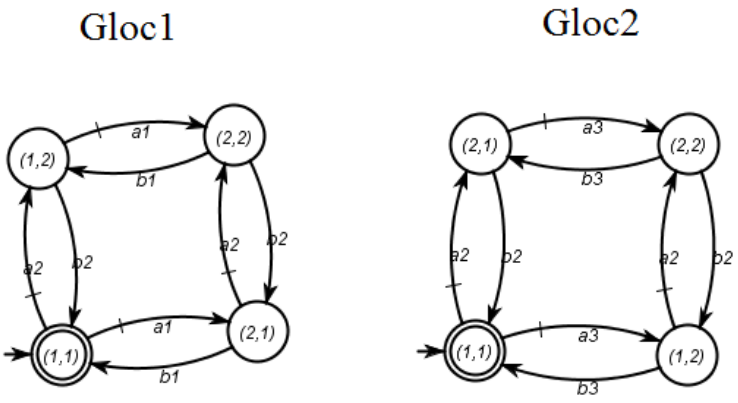


fonte: do próprio autor

Da mesma forma, no estado 2 esse modelo de autômato indica que está proibida a ocorrência do evento não controlável b_u, que indica fim da máquina M_i, sem que antes tenha ocorrido o evento controlável a_{c+1}, que indica início da máquina M_{i+1}, não permitindo que aconteça overflow da máquina M_i.

Assim, após serem modelados os subsistemas e as especificações de controle, foram obtidas as plantas modulares locais, através do composição síncrona $Gloc1 = M1 \parallel M2$ e $Gloc2 = M2 \parallel M3$, respectivamente. O resultado está ilustrado na figura 22.

Figura 22 – Modelos das plantas modulares locais



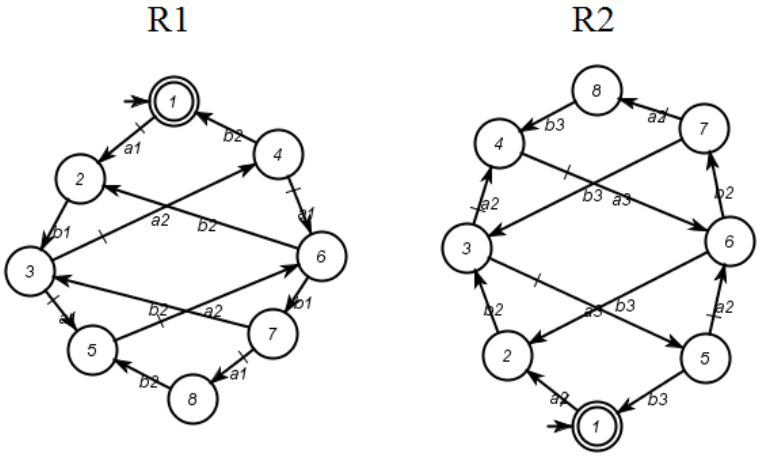
fonte: do próprio autor

Através do composição síncrona das plantas modulares locais com as especificações modulares locais foram obtidas as linguagens locais $R1 = Gloc1 \parallel E1$ e $R2 = Gloc2 \parallel E2$, conforme ilustrado na figura 23.

Na sequência, foram calculadas as máximas linguagens controláveis, obtendo-se assim os modelos dos supervisores locais $S1 = SupC(R1, Gloc1)$ e $S2 = SupC(R2, Gloc2)$, conforme ilustrados na Figura 24.

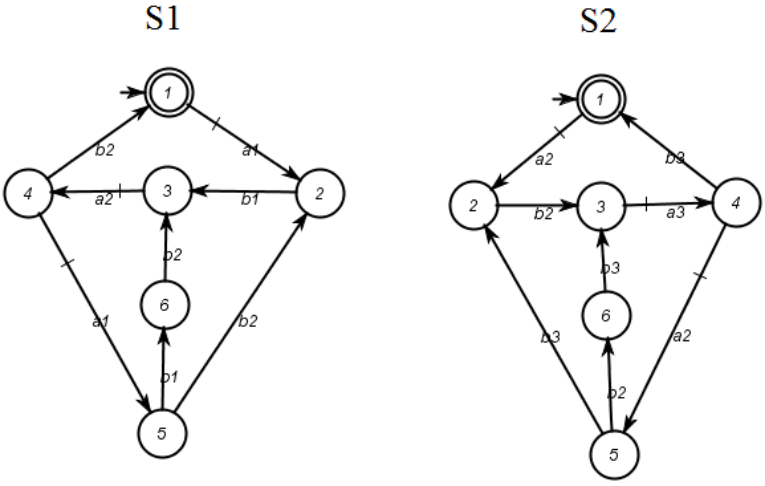
Por fim, obteve-se os supervisores locais reduzidos através dos comandos 7-Condat e 8-Supreduce da ferramenta TCT, conforme ilustrado na figura 25. Todos os outros cálculos foram executados na ferramenta IDES3.

Figura 23 – Modelos das linguagens locais



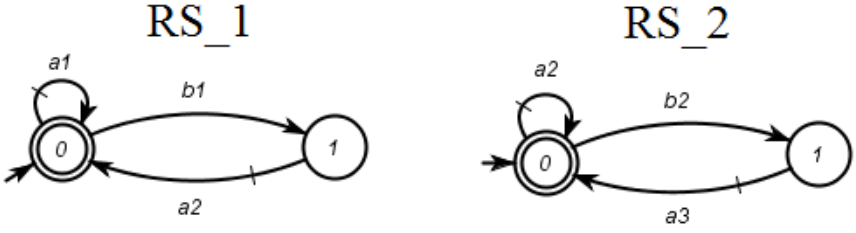
fonte: do próprio autor

Figura 24 – Modelos dos supervisores locais



fonte: do próprio autor

Figura 25 – Modelos dos supervisores locais reduzidos



fonte: do próprio autor

2.9 RESUMO DO CAPÍTULO

Neste capítulo foi apresentada inicialmente uma fundamentação teórica sobre a norma IEC 61131 e seus padrões para implementações de código de controle em CLPs. As cinco linguagens definidas são: *ladder diagram*, *function block diagram*, *sequential function chart*, *structured text* e *instrunction list*.

Na sequência foram apresentados os principais conceitos da norma IEC 61499, assim como os blocos de função definidos pela mesma. Ela fornece um mecanismo funcional para implementar sistemas de automação distribuídos através do uso de blocos de funções.

Posteriormente, introduziu-se os conceitos dos sistemas a eventos discretos, teoria das linguagens, autômatos e a Teoria de Controle Supervisório. A Teoria de Controle Supervisório se mostra uma solução funcional para modelagem de sistemas a eventos discretos, gerando um controle ótimo e minimamente restritivo.

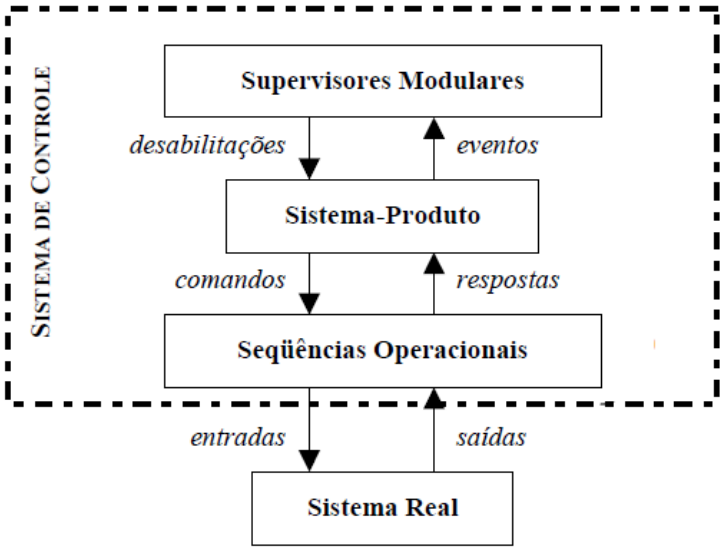
A partir da união das duas frentes de pesquisa, tanto do controle supervisório modular local como da implementação de sistemas aderentes a norma IEC 61499, foi possível avançar para o terceiro capítulo, no qual será apresentada uma metodologia que permite que um sistema modelado pela teoria de controle supervisório modular local possa ter o controle implementado por blocos de função de acordo com as definições da norma IEC 61499, gerando um padrão funcional e replicável para qualquer tipo de sistema a eventos discretos.

3 METODOLOGIA DE IMPLEMENTAÇÃO DO CONTROLE SUPERVISÓRIO MODULAR LOCAL ADERENTE À NORMA IEC 61499

A metodologia apresentada neste capítulo permite que a lógica de controle obtida para um sistema a eventos discretos, por intermédio da síntese de supervisores modulares locais (QUEIROZ, 2004), possa ser implementada por blocos de função de acordo com as definições da norma IEC 61499.

A metodologia desenvolvida se baseia na estrutura de implementação de controle modular local proposta por Queiroz (2004), a qual é ilustrada na figura 26.

Figura 26 – Estrutura de implementação do sistema de controle



fonte: (QUEIROZ, 2004)

O nível dos supervisores modulares é composto por um conjunto de supervisores modulares locais reduzidos. O nível do sistema produto é composto pelos modelos dos subsistemas assíncronos que modelam o comportamento global da planta em malha aberta. O nível de sequências operacionais constitui uma interface entre o sistema de controle e o sistema físico. Cada evento controlável associa-se exclusivamente a um comando e cada evento não controlável é unicamente associado a uma resposta. As entradas e saídas do sistema são estabelecidas com base nos sensores e atuadores dos equipamentos.

A metodologia proposta possui duas etapas principais, as quais serão explicadas detalhadamente no decorrer do trabalho. Cada etapa possui alguns passos que devem ser seguidos, conforme descrito a seguir:

Etapa 1: mapeamento da estrutura de controle supervisório modular local nos

blocos de função da IEC 61499.

a. mapeamento dos supervisores modulares locais reduzidos e dos mapas de desabilitações;

b. mapeamento dos subsistemas do sistema produto;

c. mapeamento das sequências operacionais;

d. mapeamento da leitura das entradas;

e. mapeamento da escrita nas saídas;

f. mapeamento de FBs básicos auxiliares.

Etapa 2: Interligações entre os blocos de função criados na primeira etapa.

a. inicializações dos blocos de função;

b. interligações entre SIFBs de leitura de entrada e FBs do tipo PERMIT P1;

c. interligações entre FBs do tipo PERMIT P1 e FBs do sistema produto;

d. interligações entre FBs do sistema produto e FBs do tipo PERMIT P2;

e. interligações entre FBs do tipo PERMIT P2 e FBs dos supervisores modulares locais reduzidos;

f. interligações entre FBs dos supervisores modulares locais reduzidos com FBs do tipo AND e/ou FBs do tipo PERMIT P1;

g. interligações entre FBs do sistema produto com SIFBs de escrita nas saídas.

Ao final das etapas 1 e 2, o modelo proposto será composto por oito colunas e distribuído conforme ilustrado na figura 27. O modelo proposto possui oito colunas, pois a metodologia define basicamente oito tipos de blocos de função, e as colunas são distribuídas nessa sequência para facilitar o fluxo das interligações entre eventos e variáveis de dado. Cada coluna está descrita a seguir:

1. Coluna 1: composta pelo FB básico de inicialização $E\_RESTART$;

2. Coluna 2: composta pelos SIFBs de leitura de entrada IN_σ_u;

3. Coluna 3: composta pelos FBs do tipo PERMIT $P1\_\sigma$;

4. Coluna 4: composta pelos FBs do sistema produto PS_Gi;

5. Coluna 5: composta pelos FBs do tipo PERMIT $P2\_\sigma$;

6. Coluna 6: composta pelos FBs dos supervisores modulares locais reduzidos RS_j;

7. Coluna 7: composta pelos FBs do tipo AND AND_σ_c;

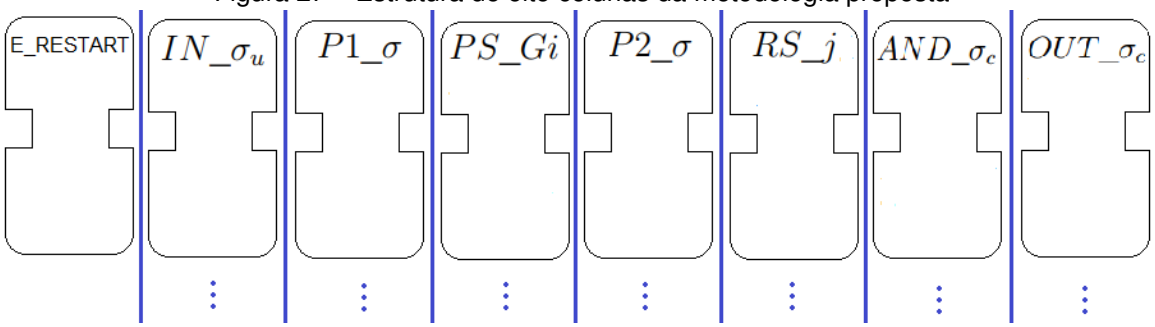
8. Coluna 8: composta pelos SIFBs de escrita nas saídas OUT_σ_c.

Cada coluna contém um conjunto limitado de blocos de função de acordo com a aplicação a ser implementada.

O fluxo de eventos controláveis funcionará da seguinte forma: no momento que o mapa de habilitações de um supervisor reduzido da coluna seis habilitar alguma variável de dado referente a algum evento controlável, a mesma será direcionada para a coluna sete, onde acontecerá uma operação booleana do tipo AND para verificar se não há proibição dessa variável por algum outro supervisor reduzido. Caso o resultado da operação booleana do tipo AND seja $TRUE$, essa variável de dado passará por um bloco do tipo $PERMIT$ na coluna três para se transformar num evento e será direcionado para o subsistema respectivo na coluna quatro. Esse evento gerará uma transição no ECC do bloco de função do subsistema, o qual gerará o evento controlável respectivo.

O fluxo de eventos não controláveis funcionará da seguinte forma: no momento que uma leitura de entrada acontecer, será gerada uma variável de dado num bloco de função de interface de serviço da coluna dois. Essa variável de dado passará por um bloco do tipo $PERMIT$ na coluna três para se transformar num evento e será direcionado para o subsistema respectivo na coluna quatro. Esse evento gerará uma transição no ECC do bloco de função do subsistema, o qual gerará o evento não controlável respectivo.

Figura 27 – Estrutura de oito colunas da metodologia proposta



fonte: do próprio autor

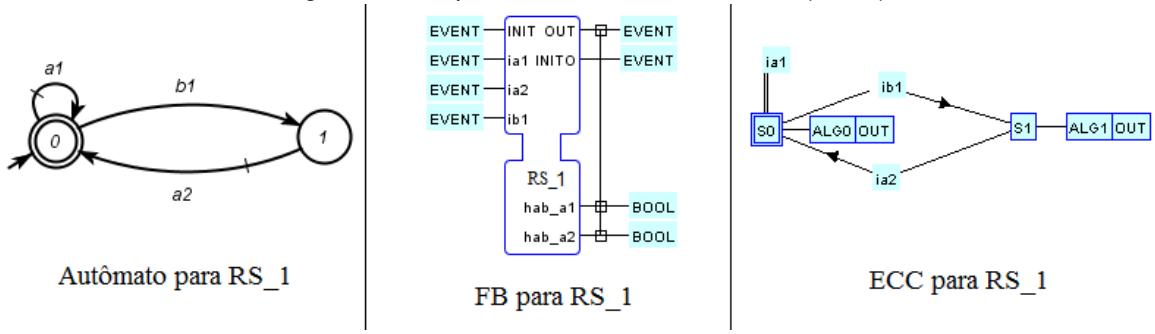
3.1 MAPEAMENTO DA ESTRUTURA DE CONTROLE MODULAR LOCAL NOS BLOCOS DE FUNÇÃO DA IEC 61499

A seguir serão descritos todos os passos referentes a etapa 1 descrita anteriormente da metodologia proposta para implementação do controle supervisório modular local aderente à norma IEC 61499.

3.1.1 Mapeamento dos supervisores modulares locais reduzidos e dos mapas de desabilitações

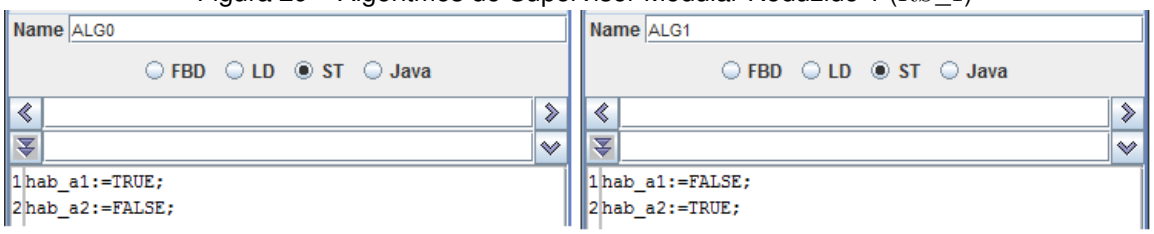
Para cada modelo de supervisor modular reduzido se deve criar um bloco de função básico respectivo, denotado por RS_j. Por exemplo, para o modelo do Supervisor Reduzido 1 da linha de transferência industrial, deve-se criar o FB RS_1. Para a configuração desse bloco de função, deve-se seguir os seguintes passos:

Figura 28 – Supervisor Modular Reduzido 1 (RS_1)



fonte: do próprio autor

Figura 29 – Algoritmos do Supervisor Modular Reduzido 1 (RS_1)



fonte: do próprio autor

1. Criar um evento de entrada σ para cada evento σ do alfabeto do supervisor modular local reduzido. Assim, no bloco de função de RS_1, devem ser criados os eventos de entrada ia_1, ia_2 e ib_1, conforme ilustrado na figura 28.

2. Para cada evento controlável σ_c do alfabeto do supervisor local reduzido, deve-se criar um dado de saída hab_σ_c no bloco de função RS_j, o qual será utilizado para definir o mapa de desabilitações em cada estado do ECC. Assim, no FB RS_1

devem ser criadas as saídas de dados hab_a1 e hab_a2 para os eventos controláveis $a1$ e $a2$, respectivamente, conforme ilustrado na figura 28.

3. Para cada estado j do autômato do supervisor modular local reduzido, deve-se criar no ECC do bloco de função RS_j um estado S_k respectivo, e para cada evento σ do autômato do supervisor modular local reduzido, deve-se criar no ECC do bloco de função RS_j uma transição $i\sigma$ respectiva, conforme exemplificado pela figura 28. No exemplo em análise, os eventos a_1, a_2 e b_1 do autômato RS_1 são mapeados como as transições ia_1, ia_2 e ib_1 do ECC e os estados 0 e 1 do autômato são mapeados como os estados S_0 e S_1 do ECC do bloco de função RS_1.

4. Criar também no bloco de função RS_j os eventos de saída OUT e $INITO$ e um evento de entrada $INIT$. Estes eventos têm por objetivos manter a continuidade da aplicação e inicializar o bloco de função, respectivamente, conforme ilustrado na figura 28.

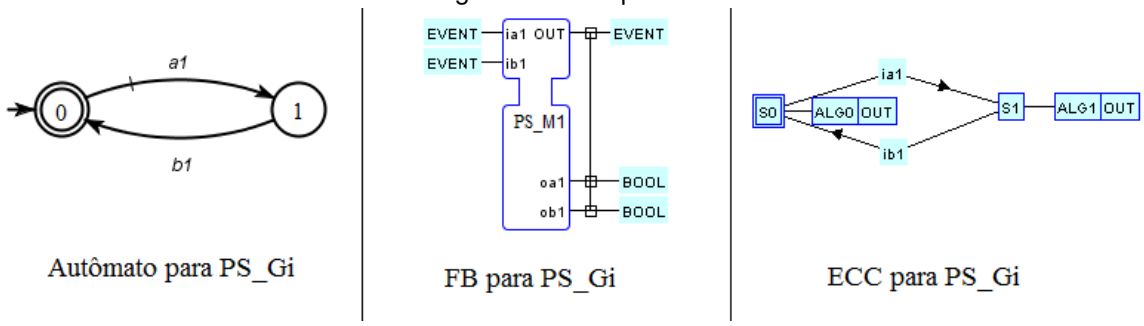
5. Criar um algoritmo ALG_k para cada um dos estados S_k do ECC, associando os mesmos ao evento de saída OUT, que será gerado após a execução do algoritmo, (ver figura 28). Pode-se notar que no ECC de RS_j o estado S_k possui o algoritmo ALG_k e o dado de saída OUT associados a ele. Como no estado 0 do autômato o evento controlável a_1 é permitido, no ALG_0 a variável de saída hab_a1 respectiva a ele recebe o valor booleano $TRUE$, já o evento controlável a_2 é proibido, logo nesse estado a variável de saída hab_a2 respectiva a ele recebe o valor booleano $FALSE$,conforme ilustrado na figura 29. No estado 1 do RS_1 o evento controlável a_1 é proibido, logo a variável de saída hab_a1 no ALG_1 recebe o valor booleano $FALSE$, já o evento controlável a_2 é permitido, logo a variável de saída hab_a2 respectiva a ele recebe o valor booleano $TRUE$, conforme ilustrado na figura 29.

3.1.2 Mapeamento dos subsistemas do sistema produto

Para cada modelo de subsistema G_i do sistema produto, deve-se criar um bloco de função básico respectivo denotado por PS_Gi. Por exemplo, para o modelo da máquina M1 da linha de transferência industrial, deve-se criar um bloco denominado PS_M1, o qual deve ser configurado de acordo com os seguintes passos:

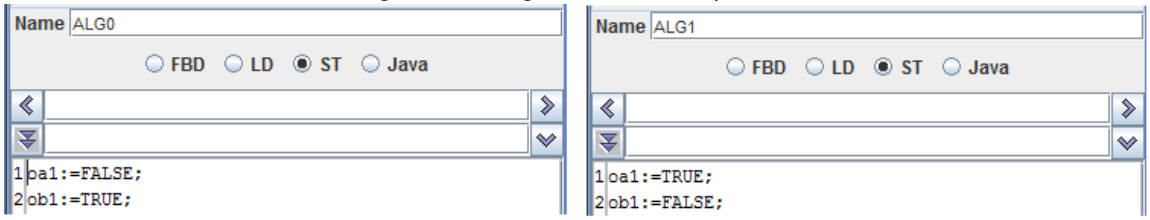
1. Criar um evento de entrada σ para cada evento σ do alfabeto do subsistema G_i. Assim, no bloco de função de PS_M1, devem ser criados os eventos de entrada

Figura 30 – Máquina M1



fonte: do próprio autor

Figura 31 – Algoritmos da Máquina M1



fonte: do próprio autor

ia_1 e ib_1, que correspondem respectivamente aos eventos a_1 e b_1 do subsistema G_i, conforme ilustrado na figura 30.

2. Para cada evento σ do alfabeto do subsistema G_i, deve-se criar um dado de saída $o\sigma$ no bloco de função PS_Gi, o qual será responsável pela geração dos eventos controláveis e não controláveis. Assim, no FB PS_M1 devem ser criadas as saídas de dados oa_1 e ob_1, correspondentes aos eventos a_1 e b_1 do subsistema G_i conforme ilustrado na figura 30.

3. Para cada estado k do autômato do subsistema G_i deve-se criar no ECC do bloco de função PS_Gi um estado S_k respectivo, e para cada evento σ, deve-se criar no ECC do bloco de função PS_Gi uma transição $i\sigma$ respectiva, conforme exemplificado pela figura 30. No exemplo em análise, os eventos a_1 e b_1 do autômato PS_M1 são mapeados como as transições ia_1 e ib_1 do ECC e os estados 0 e 1 do autômato são mapeados como os estados S_0 e S_1 do ECC do bloco de função PS_M1.

4. Criar no bloco de função PS_Gi o evento de saída OUT, cujo objetivo é manter a continuidade da aplicação, (ver figura 30).

5. Criar um algoritmo ALG_k para cada um dos estados S_k do ECC, associando este com o evento de saída OUT, que será gerado após a execução do algoritmo, (ver

Figura 30). Pode-se notar que no ECC de PS_M1 o estado S_0 possui o algoritmo ALG_0 e o dado de saída OUT associados a ele e o estado S_1 possui o algoritmo ALG_1 e o dado de saída OUT associados a ele, conforme ilustrado na figura 31. No estado S_1 a variável de saída hab_a1 recebe o valor booleano $TRUE$, já que houve uma transição a_1, indicando dessa forma a geração de um evento controlável. No estado S_0 a variável de saída hab_b1 recebe o valor booleano $TRUE$, já que houve uma transição b_1, indicando assim que houve a geração de um evento não controlável, conforme ilustrado na figura 31.

3.1.3 Mapeamento das sequências operacionais

As Sequências Operacionais são equivalentes a uma interface entre o Sistema Produto, onde se encontram os blocos de função dos subsistemas, e o Sistema Real, onde se encontram as entradas e saídas do *hardware* (QUEIROZ, 2004).

Esses procedimentos são implementados em baixo nível e definem processos importantes para o bom funcionamento do sistema, mas que não são modelados pela teoria do controle supervisório.

Por exemplo, todo o código implementado entre um evento ”faça furo” e outro evento ”fim de operação” de uma furadeira, e que não é modelado pela TCS, mas que é importante para que a operação de furo aconteça, inclusive a comunicação da furadeira com o CLP, são as sequências operacionais. Esse código depende das características de cada dispositivo físico, de modo que como nesse trabalho não são usados equipamentos reais, não se faz necessário usar sequências operacionais.

3.1.4 Mapeamento da leitura das entradas

Para cada evento não controlável σ_u do sistema implementado se deve criar um SIFB respectivo para leitura de entrada do dispositivo, denotado por IN_σ_u.

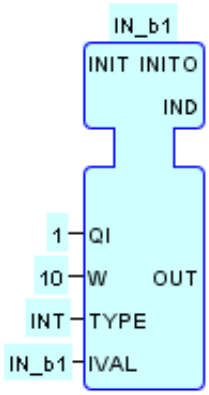
Por exemplo, para o evento não controlável b_1 da linha de transferência industrial foi criado o SIFB de leitura de entrada IN_b1, conforme ilustrado na figura 32. Esse bloco possui as entradas e saídas descritas a seguir:

1. Evento de entrada $INIT$: inicializa o bloco de função ao receber um evento de inicialização;

2. Evento de saída $INITO$: indica que a inicialização do bloco de função foi completada e envia um evento de inicialização para um próximo bloco de função;

3. Evento de saída IND: esse evento indica que a operação de leitura foi

Figura 32 – SIFB para leitura de entrada



fonte: do próprio autor

concluída e envia um evento de continuidade para um próximo bloco de função;

4. Dado de entrada QI: essa entrada de dado permanece com o valor booleano $TRUE$ para que a inicialização do serviço seja requisitada;

5. Dado de entrada $IVAL$: endereço da entrada física do *hardware* referente ao evento não controlável b_1. Nesse caso da simulação, o endereço de entrada se refere á um dado forçado pela IHM;

6. Dado de entrada $TYPE$: define o tipo de variável de entrada que será lida. Nesse caso da simulação, o tipo de variável é INT e se refere a uma variável forçada pela IHM;

7. Dado de entrada W: define o tamanho da variável de entrada;

8. Dado de saída OUT: envia o dado booleano $TRUE$ resultante da leitura da entrada física para um próximo bloco de função.

3.1.5 Mapeamento da escrita nas saídas

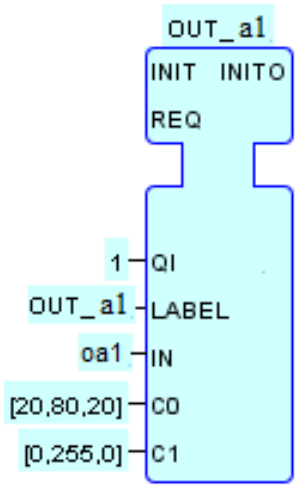
Para cada evento controlável σ_c do sistema implementado se deve criar um SIFB respectivo para escrita na saída do dispositivo, denotado por OUT_σ_c.

Por exemplo, para o evento controlável a_1 da linha de transferência industrial foi criado o SIFB de escrita na saída IN_a1, conforme ilustrado na figura 33. Esse bloco possui as entradas e saídas descritas a seguir:

1. Evento de entrada $INIT$: inicializa o bloco de função ao receber um evento de inicialização;

2. Evento de saída $INITO$: indica que a inicialização do bloco de função foi completada e envia um evento de inicialização para um próximo bloco de função;

Figura 33 – SIFB para escrita na saida



fonte: do próprio autor

3. Evento de entrada REQ: esse entrada solicita um evento para poder executar o serviço do bloco de função;

4. Dado de entrada QI: essa entrada de dado permanece com o valor booleano $TRUE$ para que a inicialização do serviço seja requisitada;

5. Dado de entrada IN: dado de entrada referente ao evento controlável gerado. Nesse caso, o dado de saída oa_1 do subsistema PS_M1;

6. Dado de entrada $LABEL$: define o nome do LED referente ao evento controlável na IHM da simulação;

7. Dado de entrada $C0$: define a cor vermelha para o LED da IHM quando IN tiver valor booleano $FALSE$;

8. Dado de entrada $C1$: define a cor verde para o LED da IHM quando IN tiver valor booleano $TRUE$.

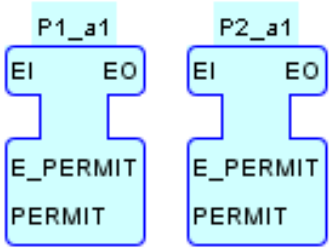
3.1.6 Mapeamento de FBs básicos auxiliares

Para implementar a metodologia proposta nesse capítulo da dissertação, será necessário utilizar também outros três modelos de FBs básicos, conforme estão descritos a seguir.

3.1.6.1 Bloco de função básico do tipo PERMIT

Para cada evento controlável σ_c do sistema implementado, deve-se criar dois FBs do tipo PERMIT respectivos. Por exemplo, para o evento controlável a_1 da linha de transferência industrial, deve-se criar os FBs do tipo PERMIT $P1\_a1$ e $P2\_a1$, conforme ilustrado na figura 34.

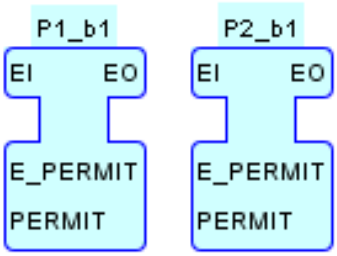
Figura 34 – Blocos de função do tipo permit $P1\_a1$ e $P2\_a1$



fonte: do próprio autor

Para cada evento não controlável σ_u do sistema implementado se deve criar dois FBs do tipo PERMIT respectivos. Por exemplo, para o evento não controlável b_1 da linha de transferência industrial, deve-se criar os FBs do tipo PERMIT $P1\_b1$ e $P2\_b1$, conforme ilustrado na figura 35.

Figura 35 – Blocos de função do tipo permit $P1\_b1$ e $P2\_b1$

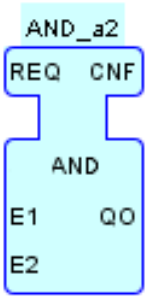


fonte: do próprio autor

3.1.6.2 Bloco de função básico do tipo AND

Deve-se criar um bloco de função do tipo AND, denotado por AND_σ_c, para cada evento controlável σ_c que pertencer ao alfabeto Σ de mais de um supervisor modular local. Por exemplo, na linha de transmissão industrial, o evento controlável a_2 pertence ao alfabeto Σ dos supervisores RS_1 e RS_2, logo, foi criado o FB AND_a2, conforme ilustrado na figura 36.

Figura 36 – Bloco de função AND_a2



fonte: do próprio autor

3.1.6.3 Bloco de função básico de inicialização

Para todo sistema desenvolvido, deve-se utilizar um FB básico do tipo E_RESTART definido na norma, que terá por objetivo inicializar a aplicação.

O modelo do FB básico do tipo E_RESTART, definida no ambiente de programação FBDK, está ilustrado na figura 37.

Figura 37 – Bloco de função do tipo E_RESTART

fonte: do próprio autor

3.1.7 Resultado da aplicação da Etapa 1 à linha de transferência industrial

Após a aplicação das etapas um e dois da metodologia de implementação da teoria de controle supervisório modular local aderente à IEC 61499 no sistema de transferência industrial, obteve-se o resultado ilustrado na figura 38, de acordo com o modelo proposto de oito colunas definido no início desse capítulo e ilustrado na figura 27.

3.2 INTERLIGAÇÕES ENTRE OS BLOCOS DE FUNÇÃO CRIADOS NA PRIMEIRA ETAPA

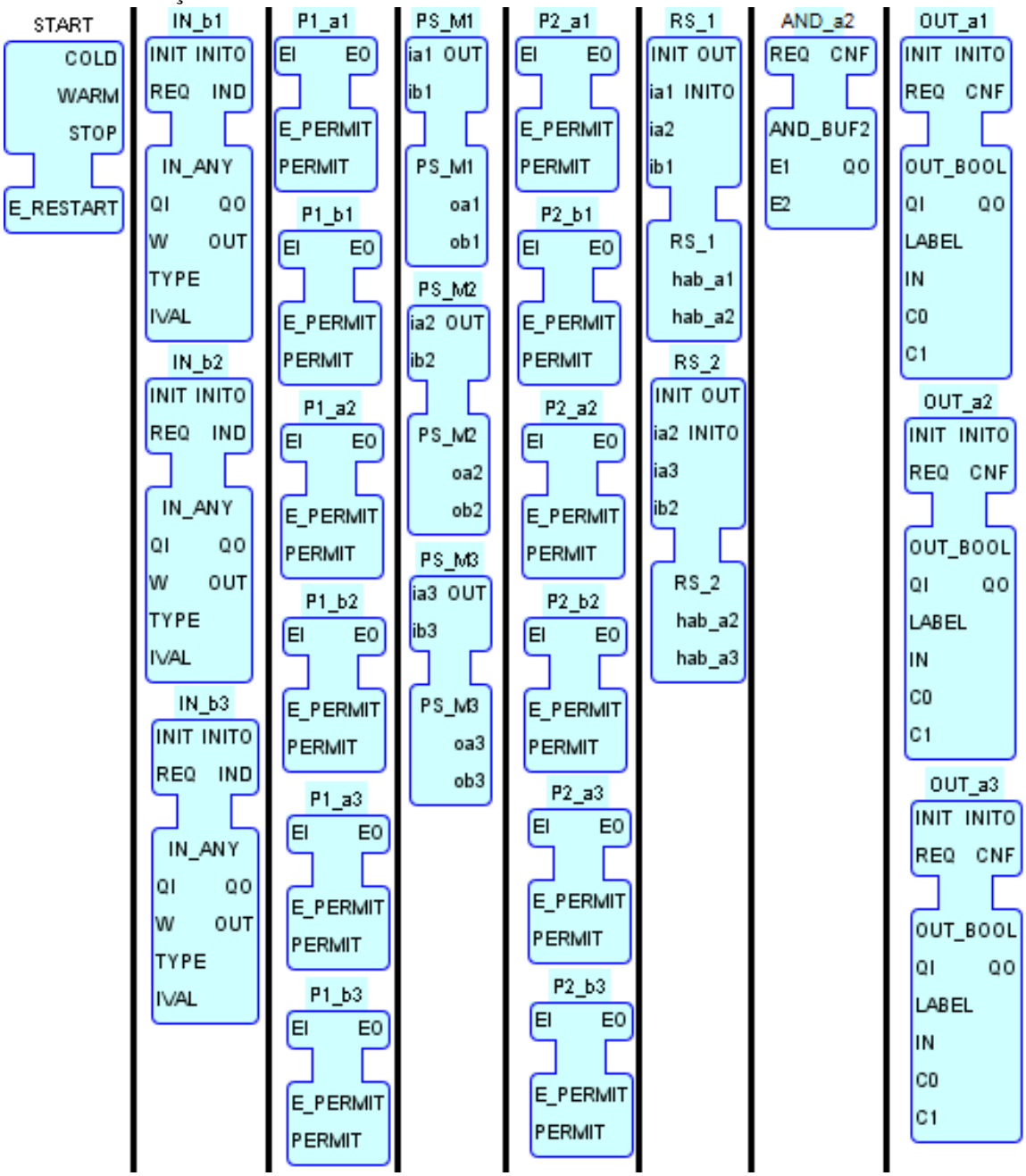
A seguir serão descritos todos os passos referentes a etapa 2 descrita anteriormente da metodologia proposta para implementação do controle supervisório modular local aderente à norma IEC 61499.

3.2.1 Inicializações dos blocos de função

Para que a aplicação comece a executar corretamente é preciso primeiramente que o sistema seja inicializado. Para que isso aconteça, o FB $E\_RESTART$ irá gerar um evento de inicialização através de sua saída de evento $COLD$ que passará por todas as entradas e saídas de eventos $INIT$ e $INITO$ dos blocos de função IN_σ_u, OUT_σ_c e RS_j.

No caso da linha de transferência industrial, a parte implementada referente a inicialização do sistema está ilustrada na figura 39. Pode-se verificar que o evento $COLD$ do FB básico $E\_RESTART$ é conectado sequencialmente a todos as entradas

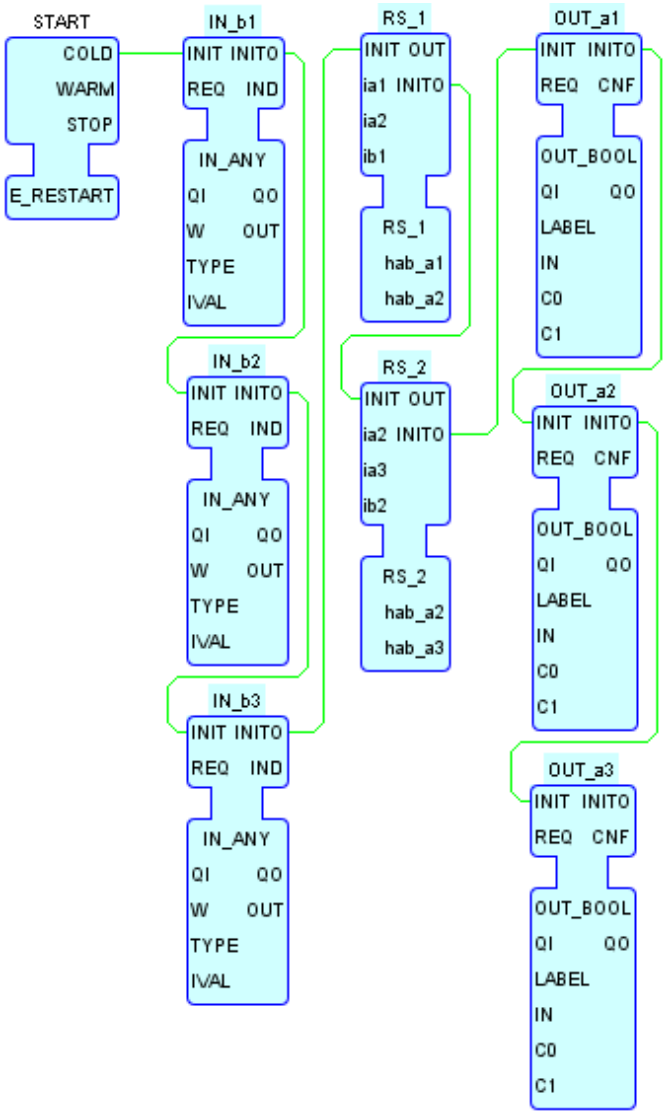
Figura 38 – Mapeamento da abordagem modular local da linha de transferência industrial nos blocos de função da IEC 61499



fonte: do próprio autor

e saídas de eventos $INIT$ e $INITO$ dos SIFBs de leitura de entrada IN_σ_u, dos FBs dos supervisores modulares locais reduzidos RS_j e dos SIFBs de escrita nas saídas OUT_σ_c. Por exemplo, a saída de evento $INITO$ do SIFB IN_b_1 está conectada a entrada de evento $INIT$ do SIFB IN_b_2.

Figura 39 – Inicialização dos SIFBs de leitura de entrada, dos FBs dos supervisores modulares locais reduzidos e dos SIFBs de escrita nas saídas



fonte: do próprio autor

3.2.2 Interligações entre SIFBs de leitura de entrada e FBs do tipo PERMIT P1

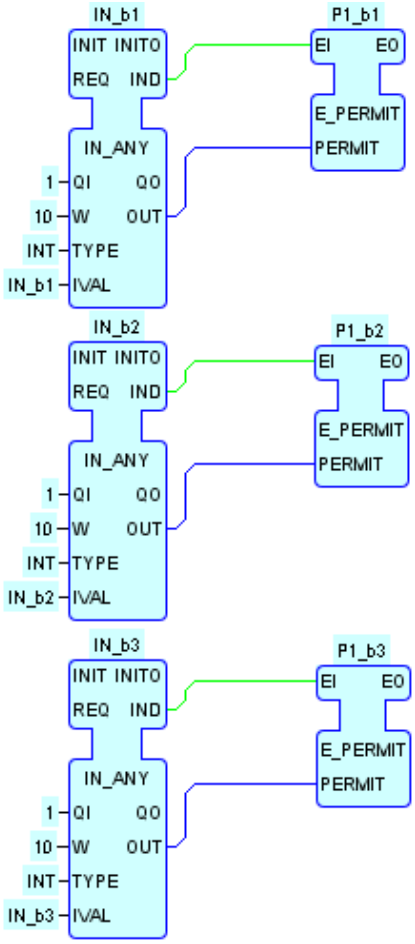
Cada SIFB IN_σ_u irá enviar um evento IND e um dado OUT para a entrada de evento EI e a entrada de dado $PERMIT$ do FB do tipo PERMIT $P1\_\sigma_u$ respectivo.

Os SIFBs também serão parametrizados nessa etapa para poderem ser simulados na IHM. As entradas de dados QI, W, $TYPE$ e $IVAL$ receberão, respectivamente, os valores 1, 10, INT e IN_σ_u, com as funções respectivas de habilitar o

execução do serviço, definir o tamanho da variável escolhida, definir o tipo da variável de entrada e nomear a variável de entrada na IHM durante a simulação.

No caso da linha de transferência industrial, a parte implementada referente a essa etapa está ilustrada na figura 40. Pode-se verificar que o evento de saída IND e o dado de saída OUT de cada SIFB de leitura de entrada IN_σ_u estão conectados as suas respectivas entradas EI e $PERMIT$ do seu respectivo FB básico do tipo PERMIT $P1\_\sigma_u$. Por exemplo, a saída de evento IND do SIFB IN_b_1 está conectada a entrada de evento EI do FB do tipo PERMIT $P1\_b_1$.

Figura 40 – Interligação entre os SIFBs de leitura de entrada e os FBs do tipo PERMIT



fonte: do próprio autor

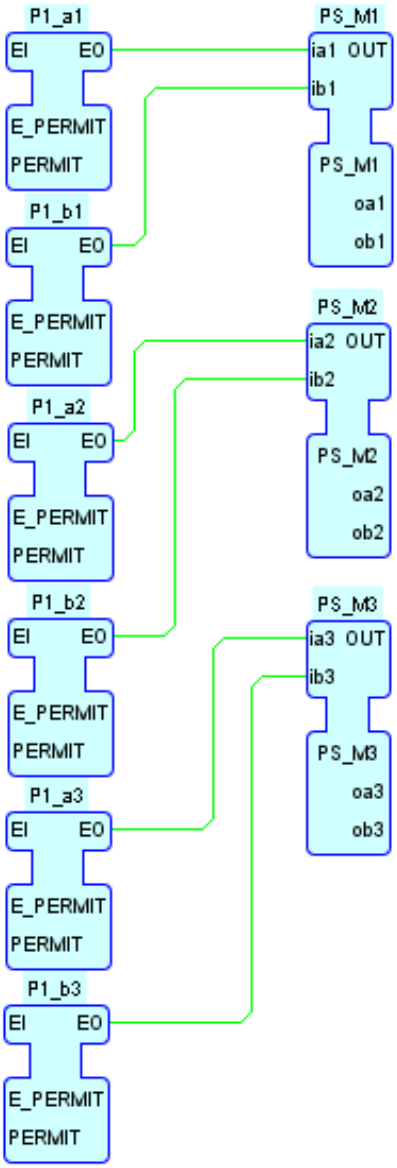
3.2.3 Interligações entre FBs do tipo PERMIT P1 e FBs do sistema produto

Nessa etapa, os FBs do tipo PERMIT $P1\_\sigma$ irão enviar um evento EO para as entradas de eventos $i\sigma$ do sistema produto PS_G_i respectivo caso as entradas de dados $PERMIT$ possuam o valor booleano $TRUE$.

No caso da linha de transferência industrial, a parte implementada referente a essa etapa está ilustrada na figura 41. Pode-se verificar que a saída de evento $E0$ dos FBs do tipo PERMIT $P1\_\sigma$ estão conectadas as suas respectivas entradas de eventos

$i\sigma$ dos seus respectivos FBs PS_G_i. Por exemplo, a saída de evento $E0$ do FB do tipo PERMIT $P1\_a_1$ está conectada a entrada de evento ia_1 do FB PS_M_1.

Figura 41 – Interligação entre os FBs do tipo PERMIT e os FBs do Sistema Produto



fonte: do próprio autor

3.2.4 Interligações entre FBs do sistema produto e FBs do tipo PERMIT P2

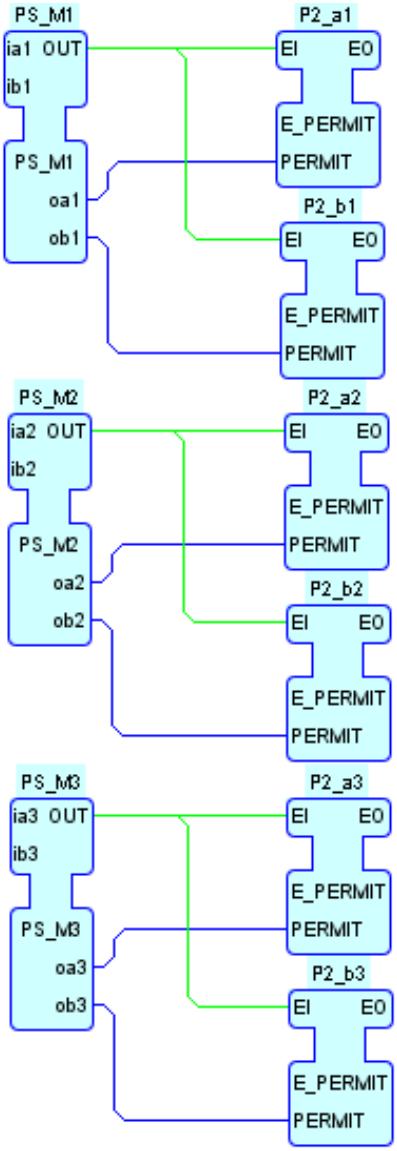
A partir do momento que uma variável de dado de saída $o\sigma$ de um subsistema PS_Gi, referente a um evento controlável ou não controlável σ, receber o valor booleano $TRUE$, esse mesmo dado será enviado para um FB do tipo PERMIT $P2\_\sigma$ associado a um evento de saída OUT do subsistema.

É no sistema produto que efetivamente acontece a geração tanto dos eventos controláveis como dos eventos não controláveis. No momento que um algoritmo interno ALG_k é executado e um valor booleano $TRUE$ é associado a uma das variáveis

de dados de saída $o\sigma$, acontece a geração do seu respectivo evento controlável ou não controlável, representado no autômato por σ.

No caso da linha de transferência industrial, a parte implementada referente a essa etapa está ilustrada na figura 42. Pode-se verificar que o evento de saída OUT e as variáveis de dados $o\sigma$ estão conectados as suas respectivas entradas de evento EI e entrada de dados $PERMIT$ dos seus respectivos FBs do tipo PERMIT $P2\_\sigma$. Por exemplo, a saída de evento OUT e a saída de dado oa_1 do FB PS_M1 estão conectados a entrada de evento EI e a entrada de dado PERMIT do FB $P2\_a_1$.

Figura 42 – Interligação entre os FBs do Sistema Produto com os FBs do tipo PERMIT



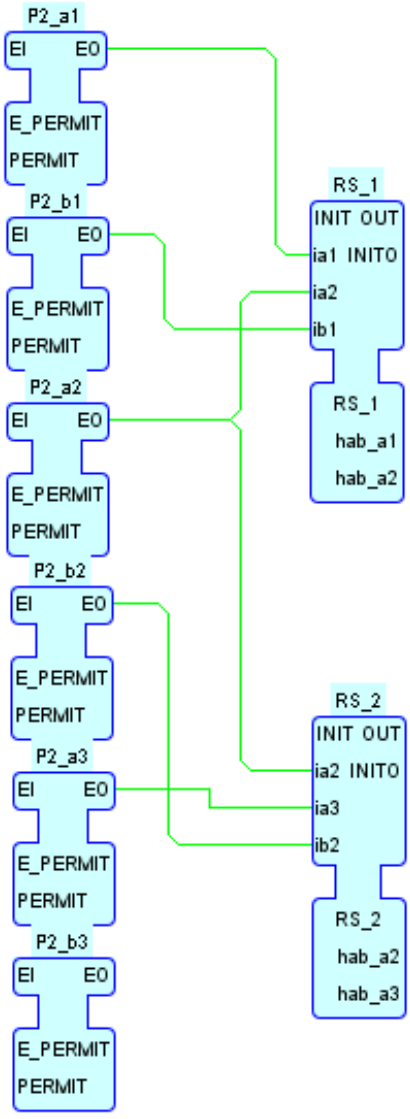
fonte: do próprio autor

3.2.5 Interligações entre FBs do tipo PERMIT P2 e FBs dos supervisores modulares locais reduzidos

Nessa etapa, os FBs do tipo PERMIT $P2\_\sigma$ irão enviar um evento EO para as entradas de eventos respectivas $i\sigma$ dos supervisores modulares locais RS_j, caso as entradas de dados $PERMIT$ possuam o valor booleano $TRUE$.

No caso da linha de transferência industrial, a parte implementada referente a essa etapa está ilustrada na figura 43. Pode-se verificar que a saída de evento $E0$ dos FBs do tipo PERMIT $P2\_\sigma$ estão conectados as suas respectivas entradas de eventos $i\sigma$ dos seus respectivos FBs RS_σ_j. Por exemplo, a saída de evento $E0$ do FB do tipo PERMIT $P2\_a_1$ está conectada a entrada de evento ia_1 do FB RS_1.

Figura 43 – Interligação entre os FBs do tipo PERMIT com os FBs dos supervisores modulares locais



fonte: do próprio autor

3.2.6 Interligações entre FBs dos supervisores modulares locais reduzidos com FBs do tipo AND e/ou FBs do tipo PERMIT P1

A partir do momento que uma variável de dado de saída hab_σ_c, referente ao mapa de habilitações de eventos controláveis, de um supervisor modular local RS_j, receber o valor booleano $TRUE$, esse mesmo dado será enviado para um FB do tipo PERMIT $P1\_\sigma$. Porém, caso o evento controlável pertença ao alfabeto Σ de mais de um supervisor modular local, o mesmo deverá primeiro ser direcionado para um FB do tipo AND AND_σ, pra depois ser direcionado para um FB do tipo PERMIT $P1\_\sigma$. Além da variável de dado de saída, sempre será também enviado um evento de saída OUT associado.

No caso da linha de transferência industrial, a parte implementada referente a essa etapa está ilustrada na figura 44. Pode-se verificar que a saída de evento OUT e as saídas de dados hab_σ_c dos FBs RS_j estão conectadas a entrada de evento EI e a entrada de dado $PERMIT$ de seus respectivos FBs do tipo PERMIT $P1\_\sigma$. Se a saída de dado hab_σ_c for compartilhada em mais de um supervisor, nesse caso a mesma será direcionada primeiramente a entrada de dado E do seus respectivo FB do tipo AND AND_σ. Por exemplo, a saída de evento OUT e a saída de dado hab_a_2 do FB RS_1 estão conectados a entrada de evento REQ e a entrada de dado $E1$ do FB AND_a_2.

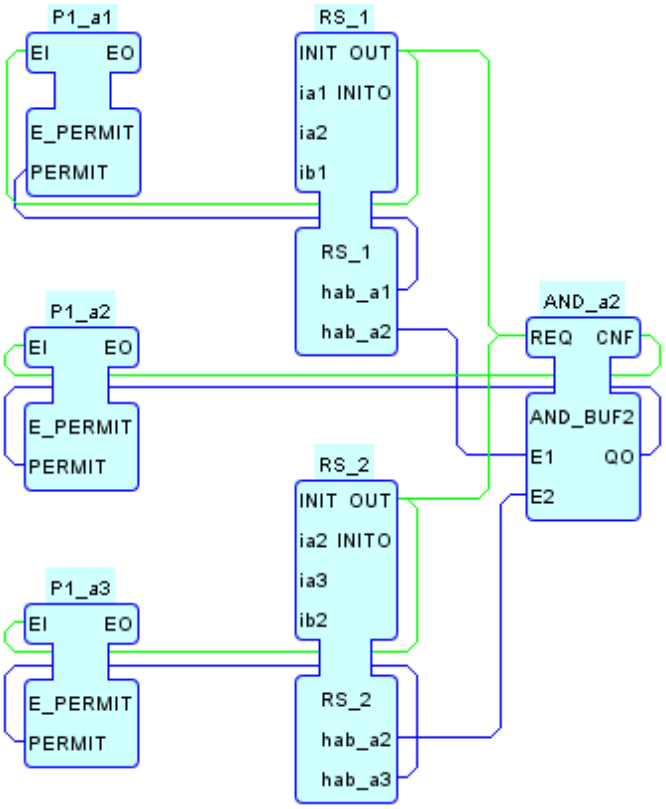
3.2.7 Interligações entre FBs do sistema produto com SIFBs de escrita nas saídas

A partir do momento que uma variável de dado de saída $o\sigma$ de um subsistema PS_Gi, referente a um evento controlável σ_c, receber o valor booleano $TRUE$, esse mesmo dado será enviado para um SIFB de escrita na saída OUT_σ_c associado com um evento OUT, de forma que aconteça o acionamento físico de um atuador conectado a alguma das saídas do dispositivo de controle.

Os SIFBs também serão parametrizados nessa etapa para poderem ser simulados na IHM. As entradas de dados QI, $LABEL$, $C0$ e $C1$ receberão, respectivamente, os valores 1, OUT_σ_c, $[20, 80, 20]$ e $[0, 255, 0]$, com as funções respectivas de habilitar o execução do serviço, nomear o LED na simulação da IHM, definir a cor do LED na simulação IHM para o caso da variável IN receber o valor booleano $FALSE$ e definir a cor do LED na simulação IHM para o caso da variável IN receber o valor booleano $TRUE$.

No caso da linha de transferência industrial, a parte implementada referente a essa etapa está ilustrada na figura 45. Pode-se verificar que a saída de evento OUT e as saídas de dados $o\sigma_c$ referente aos eventos controláveis dos FBs PS_Gi

Figura 44 – Interligação entre os FBs dos supervisores modulares locais com os FBs do tipo AND ou com os FBs do tipo PERMIT



fonte: do próprio autor

estão conectados ao evento de entrada REQ e aos dados de entrada IN de seus respectivos SIFBs OUT_σ_c. Por exemplo, a saída de evento OUT e a saída de dado oa_1 do FB PS_M1 estão conectados a entrada de evento REQ e a entrada de dado IN do FB OUT_a_1.

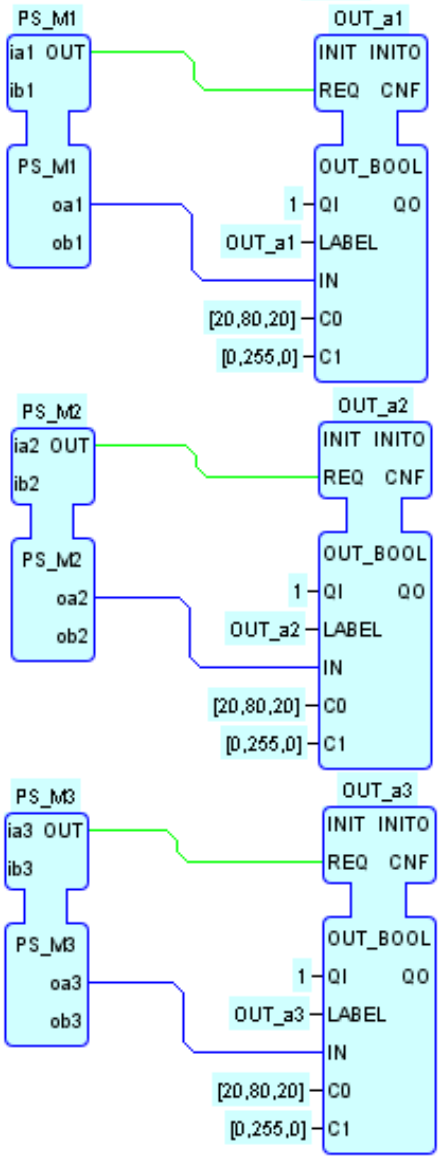
3.2.8 Resultados obtidos das simulações da linha de transferência industrial no FBDK

O resultado da aplicação das etapas 1 e 2 ao sistema de transferência industrial pode ser verificado no apêndice A desse trabalho.

Após a implementação da linha de transferência industrial na ferramenta FBDK/F-BRT, foram simuladas diferentes etapas para verificar se a metodologia desenvolvida gerou resultados funcionais, conforme ilustrado na figura 46.

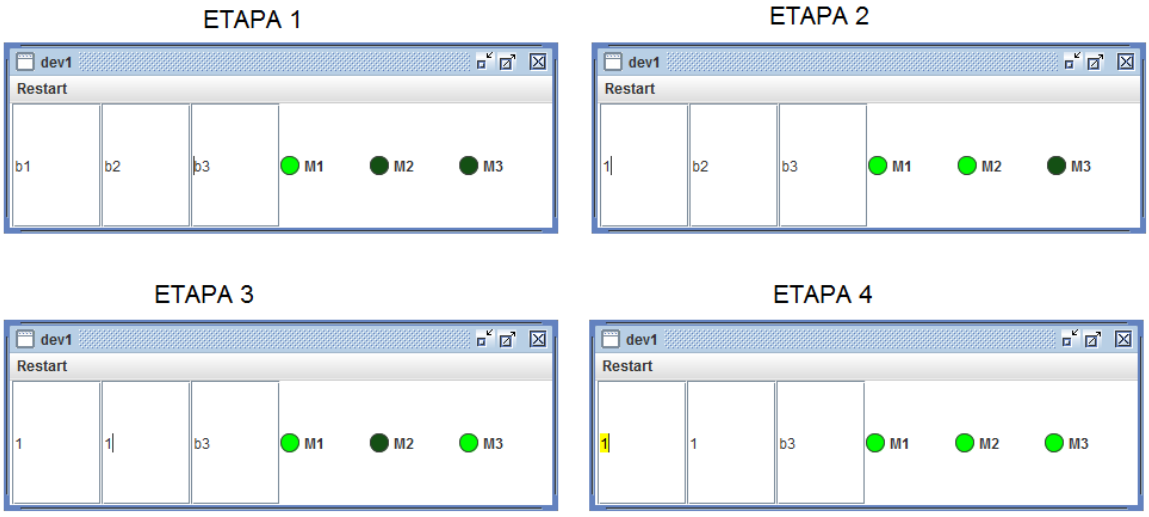
As caixas de textos a esquerda da imagem são responsáveis pela geração manual dos eventos não controláveis σ_u na hora da simulação, e os LEDs indicativos a direita da imagem representam o estado atual de cada máquina, ou seja, se a mesma está operando uma peça (verde claro) ou está vazia (verde escuro). As etapas simuladas são descritas a seguir:

Figura 45 – Interligação entre os FBs do Sistema Produto com os SIFBs de escrita na saída



fonte: do próprio autor

Figura 46 – Resultados obtidos da simulação



fonte: do próprio autor

1. Etapa 1: inicialização do sistema, ou seja, no momento que se ligou o sistema, o evento controlável a_1 foi automaticamente executado, visto que o mesmo estava habilitado pelo mapa de habilitações dos supervisores modulares locais. O LED com valor booleano $TRUE$ resultante dessa operação é M_1, que indica operação de máquina. Já os LEDs M_2 e M_3 estão apagados, indicando máquinas ociosas;

2. Etapa 2: após a etapa anterior, forçou-se um evento não controlável b_1 (fim da máquina M_1). Logo, a peça que estava sendo trabalhada em M_1 foi colocada no buffer B_1 e, como a máquina M_2 estava ociosa, rapidamente o evento controlável a_2 (inicia máquina M_2) foi automaticamente gerado, já que o mesmo estava habilitado pelo mapa de habilitações dos supervisores modulares locais. Nesse mesmo instante, como a máquina M_1 se tornou ociosa, o evento controlável a_1 (inicia máquina M_1) foi automaticamente gerado, já que o mesmo estava habilitado pelo mapa de habilitações dos supervisores modulares locais. Os LEDs com valor booleano $TRUE$ resultantes dessa operação são M_1 e M_2, que indica operação das máquinas. Já o LED M_3 continua apagado, indicando máquina ociosa;

3. Etapa 3: após a etapa anterior, forçou-se um evento não controlável b_2 (fim da máquina M_2). Logo, a peça que estava sendo trabalhada em M_2 foi colocada no buffer B_2 e, como a máquina M_3 estava ociosa, rapidamente o evento controlável a_3 (inicia máquina M_3) foi automaticamente gerado, já que o mesmo estava habilitado pelo mapa de habilitações dos supervisores modulares locais. Os LEDs com valor booleano $TRUE$ resultantes dessa operação são M_1 e M_3, que indica operação das máquinas. Já o LED M_2 está apagado, indicando máquina ociosa;

4. Etapa 4: após a etapa anterior, forçou-se um evento não controlável b_1 (fim da máquina M_1). Logo, a peça que estava sendo trabalhada em M_1 foi colocada no buffer B_1 e, como a máquina M_2 estava ociosa, rapidamente o evento controlável a_2 (inicia máquina M_2) foi automaticamente gerado, já que o mesmo estava habilitado pelo mapa de habilitações dos supervisores modulares locais. Nesse mesmo instante, como a máquina M_1 se tornou ociosa, rapidamente o evento controlável a_1 (inicia máquina M_1) foi automaticamente gerado, já que o mesmo estava habilitado pelo mapa de habilitações dos supervisores modulares locais. Os LEDs com valor booleano $TRUE$ resultantes dessa operação são M_1, M_2 e M_3, que indica operação das máquinas.

Após a etapa 4, o sistema se encontra em seu estado de produtividade máxima, ou seja, com as três máquinas operando ao mesmo tempo. Os supervisores modulares locais não permitem que mais peças entrem no sistema para não acontecer travamentos ou problemas, como *overflow* ou *underflow*.

3.3 RESUMO DO CAPÍTULO

Nesse capítulo foi apresentada uma metodologia que permite que um sistema modelado pela estrutura de controle modular local possa ser implementado por blocos de função de acordo com as definições da norma IEC 61499.

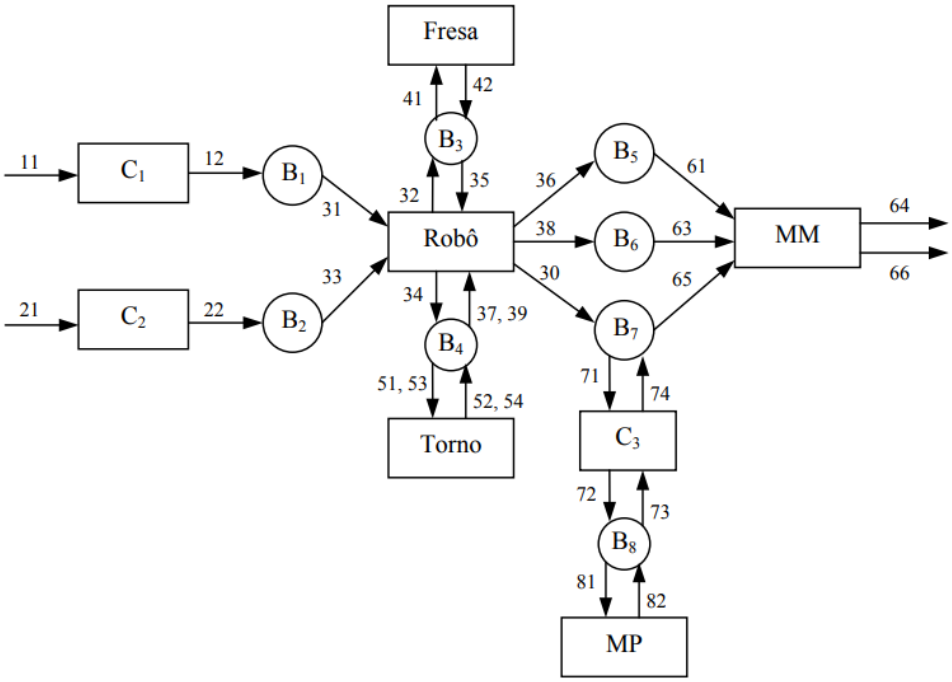
A partir do momento que são obtidos os modelos de autômato dos subsistemas do sistema produto e dos supervisores modulares reduzidos de um sistema qualquer a eventos discretos, seguem-se duas etapas. A primeira consiste no mapeamento dos modelos de autômatos dos sub sistemas e dos supervisores reduzidos nos blocos de função da IEC 61499. A segunda consiste nas interligações entre os eventos e variáveis de dado dos blocos de função.

Ao final do capítulo foi apresentada a simulação sendo executada corretamente na ferramenta FBDK/FBRT, demonstrando a eficácia da metodologia aqui desenvolvida.

4 ESTUDO DE CASO

Como estudo de caso, de forma a validar a funcionalidade da metodologia desenvolvida no capítulo 3 dessa dissertação, a mesma foi implementada em um sistema flexível de manufatura (SFM), ilustrado na figura 47. O trabalho de Pena et al. (2016) utilizou a abordagem modular local para calcular os supervisores desse sistema.

Figura 47 – Sistema Flexível de Manufatura



fonte: (PENA et al., 2016)

Esse sistema flexível de manufatura (QUEIROZ, 2004) transforma blocos brutos e tarugos brutos em dois tipos de produtos: um bloco com um pino cônico no topo (Produto A) e um bloco com um pino cilíndrico pintado (Produto B).

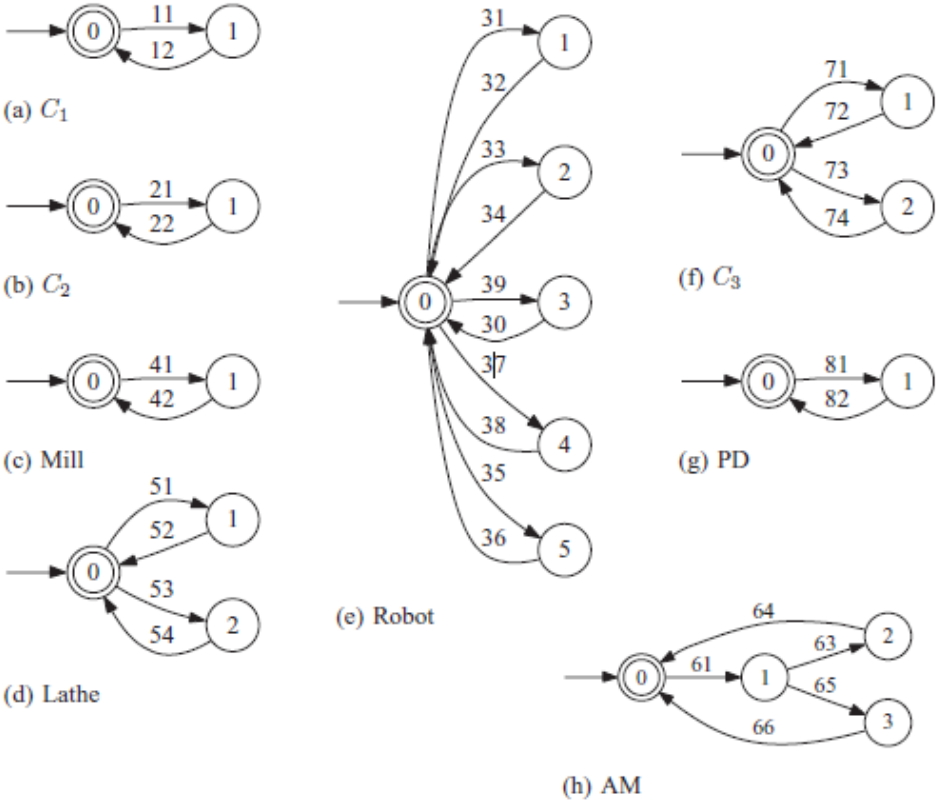
O SFM é composto de oito equipamentos: três esteiras (C_1, C_2 e C_3), uma fresa ($Fresa$), um torno ($Torno$), um robôc($Robô$), uma máquina de pintura (MP) e uma máquina de montagem (MM). Os equipamentos são conectados através de depósitos unitários, chamados de buffers (B_1, B_2, B_3, B_4, B_5, B_6, B_7 e B_8), os quais têm capacidade para armazenar apenas uma peça por vez.

As setas na figura 47 indicam o fluxo de peças inacabadas pelo SFM. Blocos brutos entram na esteira C1 (evento 11) e alcançam B1 (evento 12). Tarugos brutos entram na esteira C2 (evento 21) e chegam em B2 (evento 22), O robô pega um bloco bruto de B1 (31) e o coloca em B3 (32) ou move um tarugo bruto de B2 (33) para B4 (34). A fresadora começa a processar um bloco de B3 pelo evento 41 e retorna uma

peça com forma geométrica e um buraco no topo pelo evento 42. O torno pode fazer dois tipos de pinos com os tarugos de B4: um pino cônico (eventos 51 e 52) ou pino cilíndrico (eventos 53 e 54). Na sequência, o robô move um bloco acabado de B3 para B5 (eventos 35 e 36), move um pino cônico de B4 para B6 (eventos 37 e 38) ou move um pino cilíndrico de B4 para B7 (eventos 39 e 30). A esteira C3 transporta o pino de B7 para B8 (eventos 71 e 72), onde o pino é pintado (eventos 81 e 82), e o retorna para B7 (eventos 73 e 74). Finalmente, a máquina de montagem pega um bloco de B5 (evento 61) e põe sobre ele um pino cônico de B6 (evento 63), gerando um Produto A (evento 64), ou a máquina de montagem insere um pino cilíndrico pintado de B7 no topo do bloco, retornando um Produto B (evento 66).

Os modelos dos autômatos dos subsistemas estão ilustrados na figura 48 e os modelos dos autômatos dos supervisores modulares locais reduzidos estão ilustrados na figura 49

Figura 48 – Modelos dos autômatos dos subsistemas do sistema flexível de manufatura



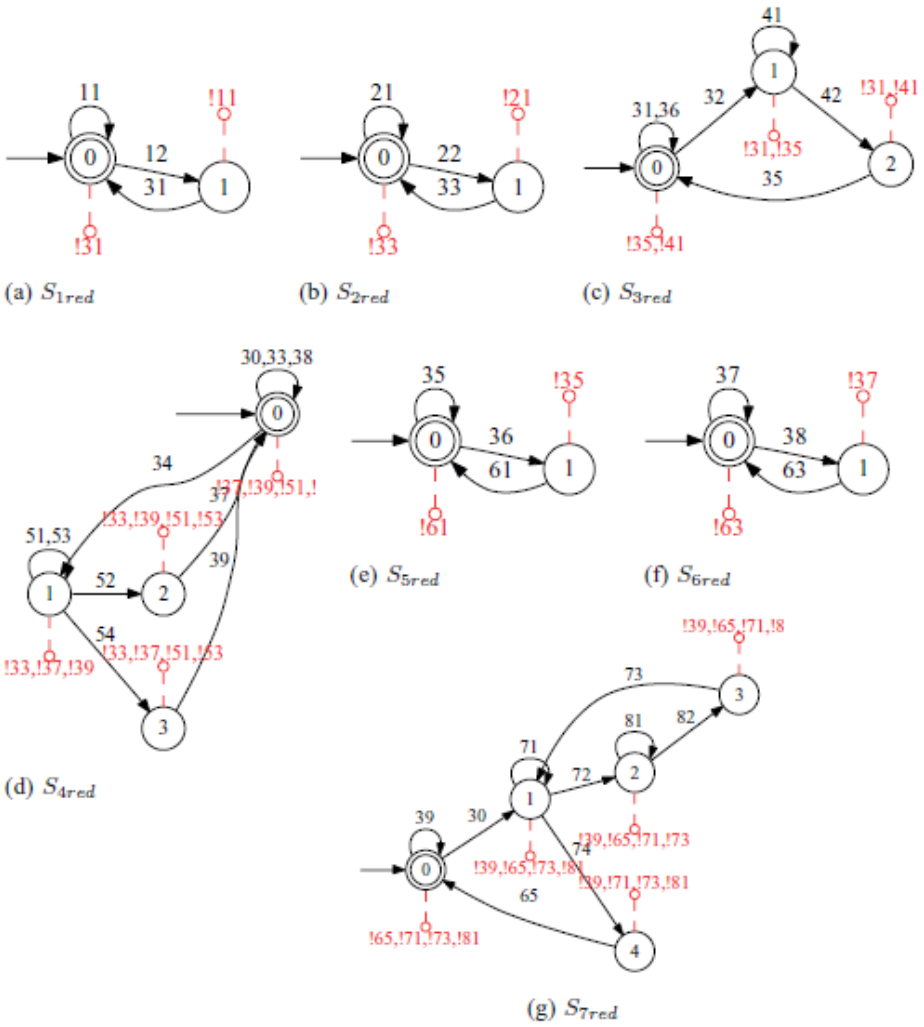
fonte: (PENA et al., 2016)

Demais informações a respeito do sistema implementado podem ser obtidas a partir da leitura de (PENA et al., 2016) e (QUEIROZ, 2004).

Toda a implementação do código e simulações foram desenvolvidas no ambiente de programação FBDK/FBRT.

A seguir serão explicados a implementação de cada passo conforme as eta-

Figura 49 – Modelos dos autômatos dos supervisores modulares locais reduzidos do sistema flexível de manufatura



fonte: (PENA et al., 2016)

pas 1 e 2 da metodologia de implementação do controle supervisório modular local aderente à norma IE 61499 desenvolvida no capítulo 3 dessa dissertação.

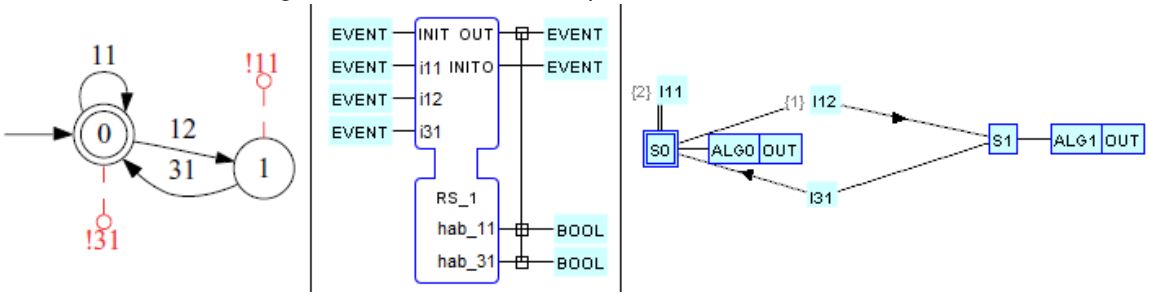
4.1 MAPEAMENTO DA ESTRUTURA DE CONTROLE SUPERVISÓRIO MODULAR LOCAL NOS BLOCOS DE FUNÇÃO DA IEC 61499

Nas próximas subseções será descrito o processo de mapeamento de todos os autômatos da estrutura de controle supervisório modular local nos blocos de função da IEC 61499.

4.1.1 Mapeamento dos supervisores modulares locais reduzidos e dos mapas de desabilitações

A figura 50 ilustra o mapeamento do autômato do Supervisor Reduzido 1 RS_1. Pode-se verificar as seguintes características referentes a este autômato:

Figura 50 – Autômato do Supervisor Reduzido 1 RS_1



fonte: do próprio autor

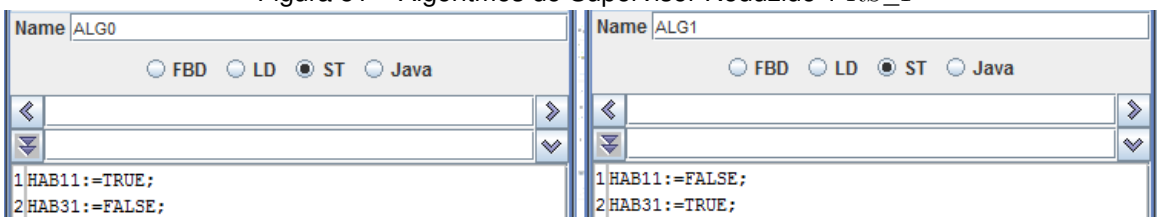
a. no bloco de função básico RS_1 foram criadas os eventos de entrada $i11$, $i12$ e $i31$ e as saídas de dado hab_11 e hab_31, referentes aos eventos 11, 12 e 31 do alfabeto Σ autômato;

b. os estados 0 e 1 foram mapeados como os estados S_0 e S_1 do ECC do bloco de função RS_1;

c. os eventos 11, 12 e 31 do autômato foram mapeados como as transições $i11$, $i12$ e $i31$ do ECC do bloco de função RS_1;

d. cada estado do ECC possui o evento de saída OUT e um algoritmo associado a ele, ALG_0 no caso do estado S_0 e ALG_1 no caso do estado S_1. Os algoritmos estão ilustrados na figura 51;

Figura 51 – Algoritmos do Supervisor Reduzido 1 RS_1

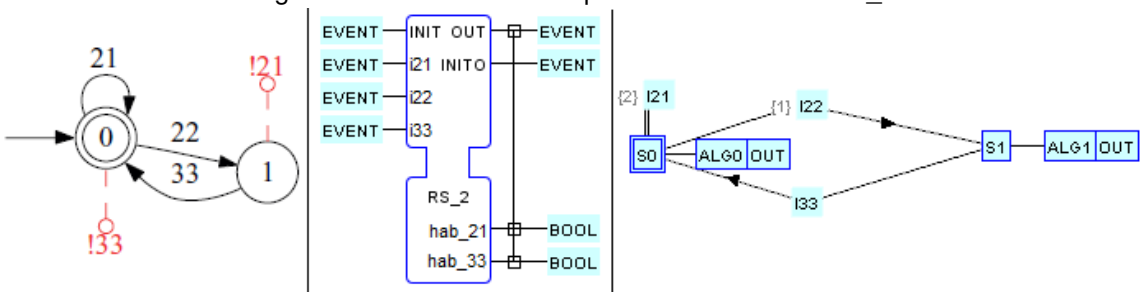


fonte: do próprio autor

A figura 52 ilustra o mapeamento do autômato do Supervisor Reduzido 2 RS_2. Pode-se verificar as seguintes características referentes a este autômato:

a. no bloco de função básico RS_2 foram criadas os eventos de entrada $i21$, $i22$ e $i33$ e as saídas de dado hab_21 e hab_33, referentes aos eventos 21, 22 e 33 do alfabeto Σ do autômato;

Figura 52 – Autômato do Supervisor Reduzido 2 RS_2



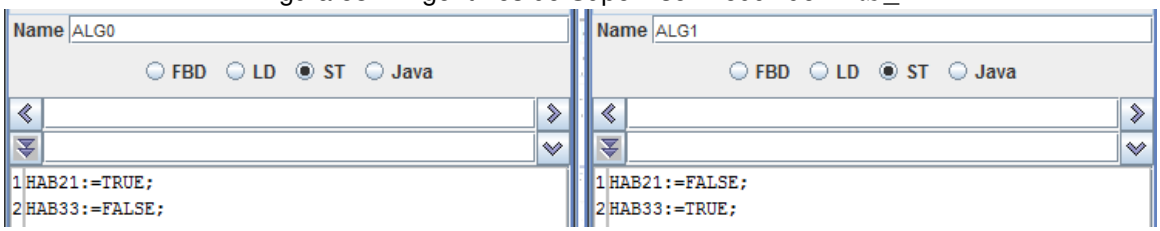
fonte: do próprio autor

b. os estados 0 e 1 foram mapeados como os estados S_0 e S_1 do ECC do bloco de função RS_2;

c. os eventos 11, 12 e 31 do autômato foram mapeados como as transições $i11$, $i12$ e $i31$ do ECC do bloco de função RS_2;

d. cada estado do ECC possui o evento de saída OUT e um algoritmo associado a ele, ALG_0 no caso do estado S_0 e ALG_1 no caso do estado S_1. Os algoritmos estão ilustrados na figura 53;

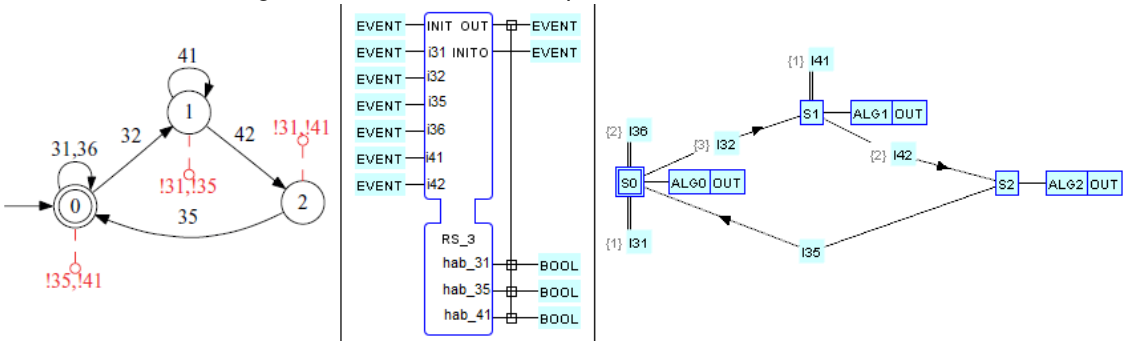
Figura 53 – Algoritmos do Supervisor Reduzido 2 RS_2



fonte: do próprio autor

A figura 54 ilustra o mapeamento do autômato do Supervisor Reduzido 3 RS_3. Pode-se verificar as seguintes características referentes a este autômato:

Figura 54 – Autômato do Supervisor Reduzido 3 RS_3



fonte: do próprio autor

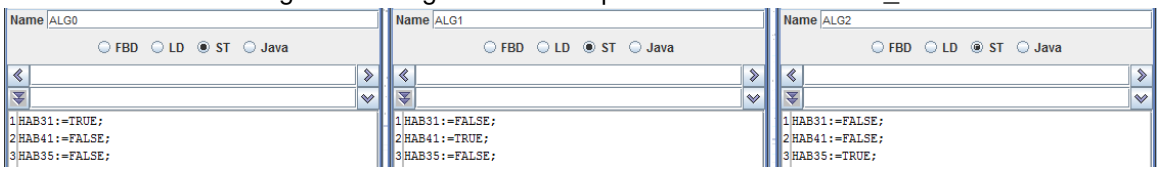
a. no bloco de função básico RS_3 foram criadas os eventos de entrada $i31$, $i32$, $i35$, $i36$, $i41$ e $i42$ e as saídas de dado hab_31, hab_35 e hab_41, referentes aos eventos 31, 32, 35, 36, 41 e 42 do alfabeto Σ do autômato;

b. os estados 0, 1 e 2 foram mapeados como os estados S_0, S_1 e S_2 do ECC do bloco de função RS_3;

c. os eventos 31, 32, 35, 36, 41 e 42 do autômato foram mapeados como as transições $i31$, $i32$, $i35$, $i36$, $i41$ e $i42$ do ECC do bloco de função RS_3;

d. cada estado do ECC possui o evento de saída OUT e um algoritmo associado a ele, ALG_0 no caso do estado S_0, ALG_1 no caso do estado S_1 e ALG_2 no caso do estado S_2. Os algoritmos estão ilustrados na figura 55;

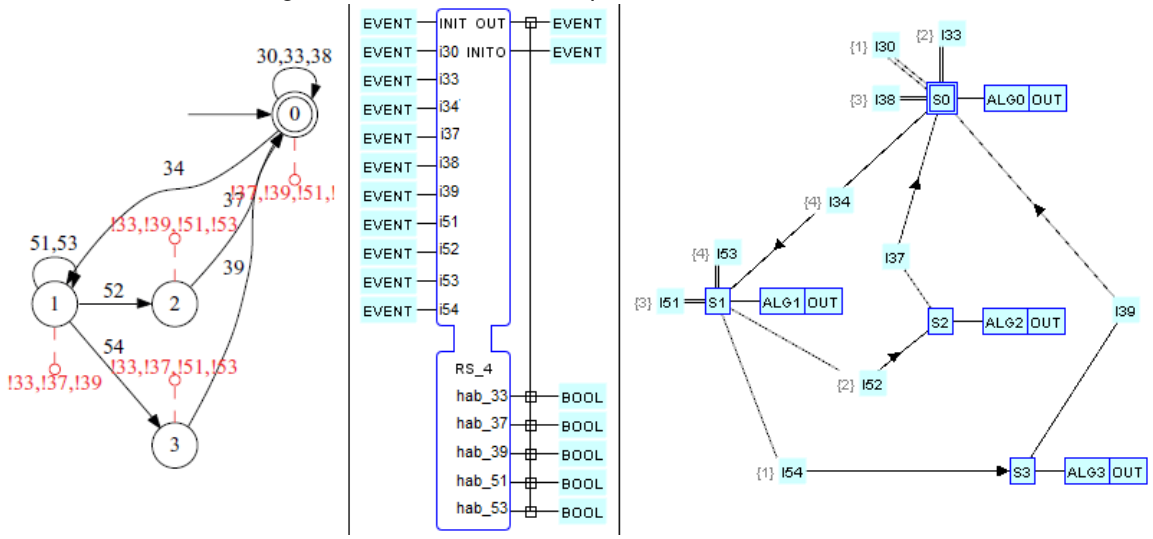
Figura 55 – Algoritmos do Supervisor Reduzido 3 RS_3



fonte: do próprio autor

A figura 56 ilustra o mapeamento do autômato do Supervisor Reduzido 4 RS_4. Pode-se verificar as seguintes características referentes a este autômato:

Figura 56 – Autômato do Supervisor Reduzido 4 RS_4



fonte: do próprio autor

a. no bloco de função básico RS_4 foram criadas os eventos de entrada $i30$, $i33$, $i34$, $i37$, $i38$, $i39$, $i51$, $i52$, $i53$ e $i54$ e as saídas de dado hab_33, hab_37, hab_39,

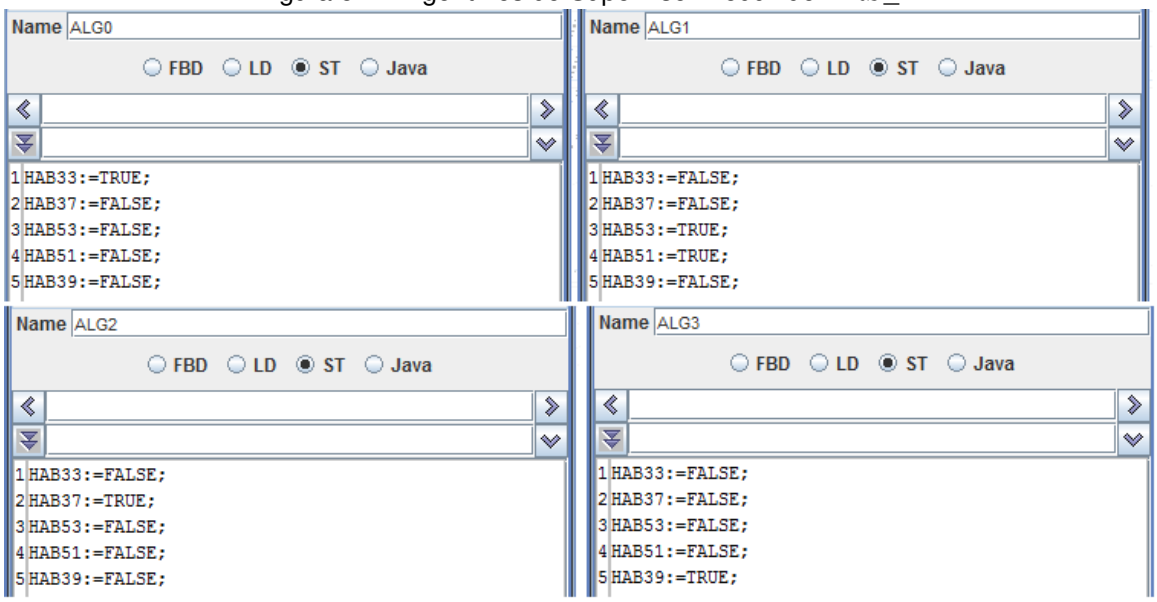
hab_51 e hab_53, referentes aos eventos 30, 33, 34, 37, 38, 39, 51, 52, 53 e 54 do alfabeto Σ do autômato;

b. os estados 0, 1, 2 e 3 foram mapeados como os estados S_0, S_1, S_2 e S_3 do ECC do bloco de função RS_4;

c. os eventos 30, 33, 34, 37, 38, 39, 51, 52, 53 e 54 do autômato foram mapeados como as transições $i30$, $i33$, $i34$, $i37$, $i38$, $i39$, $i51$, $i52$, $i53$ e $i54$ do ECC do bloco de função RS_3;

d. cada estado do ECC possui o evento de saída OUT e um algoritmo associado a ele, ALG_0 no caso do estado S_0, ALG_1 no caso do estado S_1, ALG_2 no caso do estado S_2 e ALG_3 no caso do estado S_3. Os algoritmos estão ilustrados na figura 57;

Figura 57 – Algoritmos do Supervisor Reduzido 4 RS_4



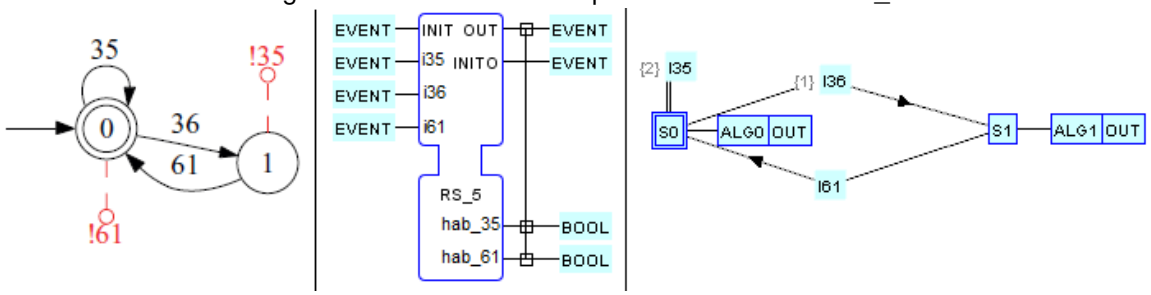
fonte: do próprio autor

A figura 58 ilustra o mapeamento do autômato do Supervisor Reduzido 5 RS_5. Pode-se verificar as seguintes características referentes a este autômato:

a. no bloco de função básico RS_5 foram criadas os eventos de entrada $i35$, $i36$ e $i61$ e as saídas de dado hab_35 e hab_61, referentes aos eventos 35, 36 e 61 do alfabeto Σ do autômato;

b. os estados 0 e 1 foram mapeados como os estados S_0 e S_1 do ECC do bloco de função RS_5;

Figura 58 – Autômato do Supervisor Reduzido 5 RS_5

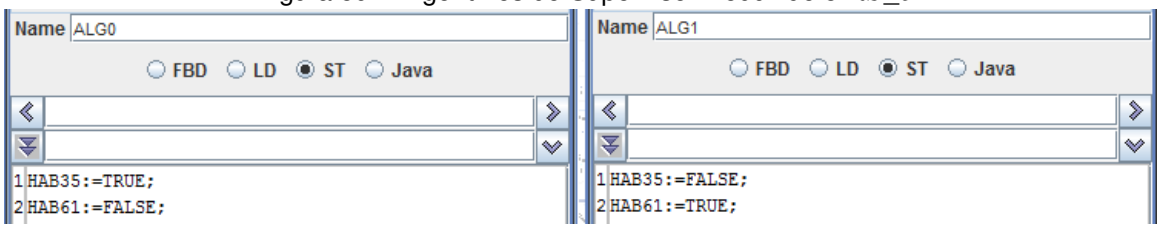


fonte: do próprio autor

c. os eventos 35, 36 e 61 do autômato foram mapeados como as transições $i35$, $i36$ e $i61$ do ECC do bloco de função RS_5;

d. cada estado do ECC possui o evento de saída OUT e um algoritmo associado a ele, ALG_0 no caso do estado S_0 e ALG_1 no caso do estado S_1. Os algoritmos estão ilustrados na figura 59;

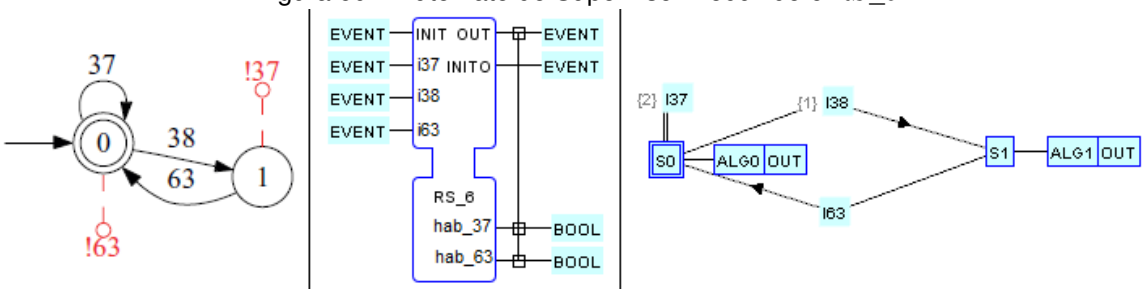
Figura 59 – Algoritmos do Supervisor Reduzido 5 RS_5



fonte: do próprio autor

A figura 60 ilustra o mapeamento do autômato do Supervisor Reduzido 6 RS_6. Pode-se verificar as seguintes características referentes a este autômato:

Figura 60 – Autômato do Supervisor Reduzido 6 RS_6



fonte: do próprio autor

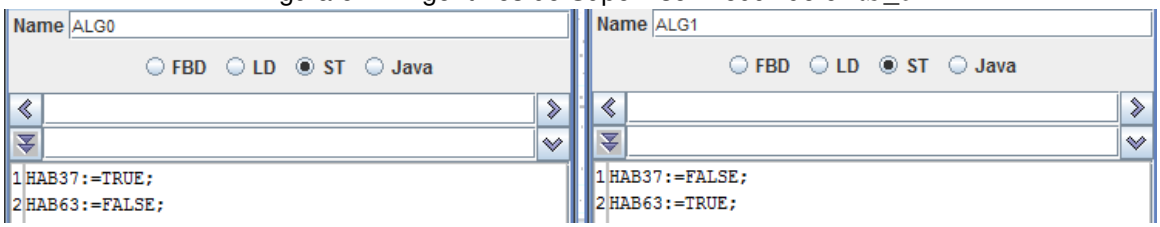
a. no bloco de função básico RS_6 foram criadas os eventos de entrada $i37$, $i38$ e $i63$ e as saídas de dado hab_37 e hab_63, referentes aos eventos 37, 38 e 63 do alfabeto Σ do autômato;

b. os estados 0 e 1 foram mapeados como os estados S_0 e S_1 do ECC do bloco de função RS_6;

c. os eventos 37, 38 e 63 do autômato foram mapeados como as transições $i37$, $i38$ e $i63$ do ECC do bloco de função RS_6;

d. cada estado do ECC possui o evento de saída OUT e um algoritmo associado a ele, ALG_0 no caso do estado S_0 e ALG_1 no caso do estado S_1. Os algoritmos estão ilustrados na figura 61;

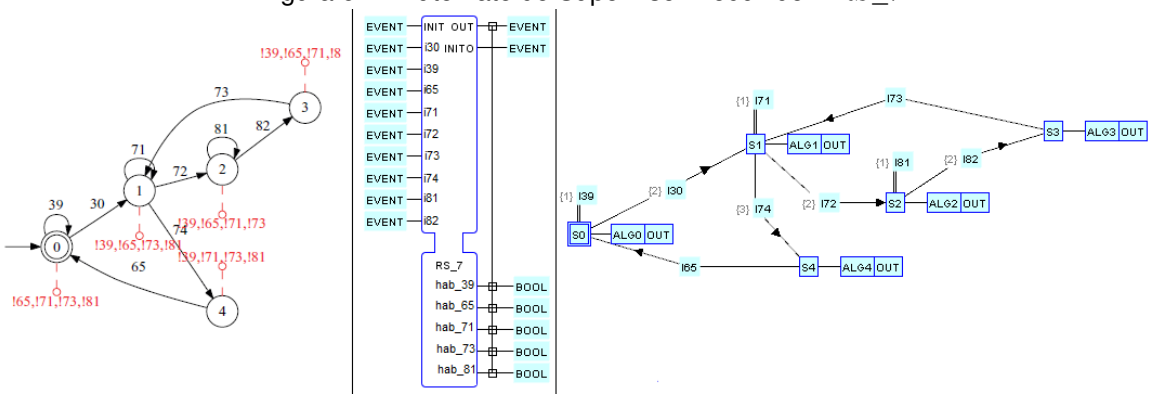
Figura 61 – Algoritmos do Supervisor Reduzido 6 RS_6



fonte: do próprio autor

A figura 62 ilustra o mapeamento do autômato do Supervisor Reduzido 6 RS_7. Pode-se verificar as seguintes características referentes a este autômato:

Figura 62 – Autômato do Supervisor Reduzido 7 RS_7



fonte: do próprio autor

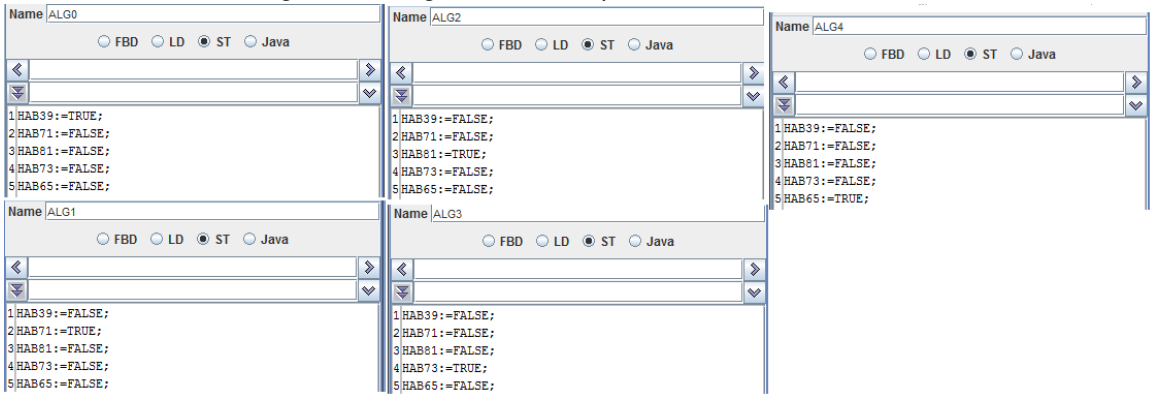
a. no bloco de função básico RS_7 foram criadas os eventos de entrada $i30$, $i39$, $i65$, $i71$, $i72$, $i73$, $i74$, $i81$ e $i82$ e as saídas de dado hab_39, hab_65, hab_71, hab_73 e hab_81, referentes aos eventos 30, 39, 65, 71, 72, 73, 74, 81 e 82 do alfabeto Σ do autômato;

b. os estados 0, 1, 2, 3 e 4 foram mapeados como os estados S_0, S_1, S_2, S_3 e S_4 do ECC do bloco de função RS_7;

c. os eventos 30, 39, 65, 71, 72, 73, 74, 81 e 82 do autômato foram mapeados como as transições $i30$, $i39$, $i65$, $i71$, $i72$, $i73$, $i74$, $i81$ e $i82$ do ECC do bloco de função RS_7;

d. cada estado do ECC possui o evento de saída OUT e um algoritmo associado a ele, ALG_0 no caso do estado S_0, ALG_1 no caso do estado S_1, ALG_2 no caso do estado S_2, ALG_3 no caso do estado S_3 e ALG_4 no caso do estado S_4. Os algoritmos estão ilustrados na figura 63;

Figura 63 – Algoritmos do Supervisor Reduzido 7 RS_7

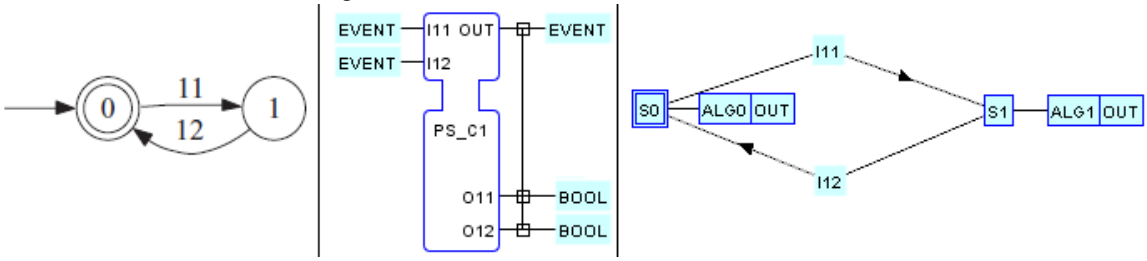


fonte: do próprio autor

4.1.2 Mapeamento dos subsistemas do sistema produto

A figura 64 ilustra o mapeamento do autômato do subsistema da esteira C1 PS_C1. Pode-se verificar as seguintes características referentes a este autômato:

Figura 64 – Autômato da Esteira C1 PS_C1



fonte: do próprio autor

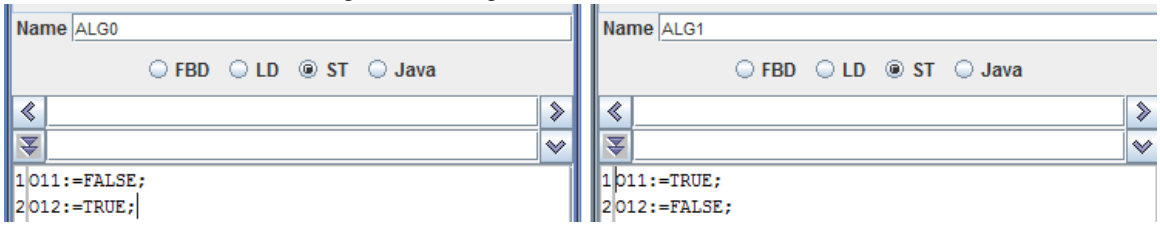
a. no bloco de função básico PS_C1 foram criados os eventos de entrada $i11$ e $i12$ e as saídas de dado $o11$ e $o12$, referentes aos eventos 11 e 12 do alfabeto Σ do autômato;

b. os estados 0 e 1 foram mapeados como os estados S_0 e S_1 do ECC do bloco de função PS_C1;

c. os eventos 11 e 12 foram mapeados como as transições $i11$ e $i12$ do ECC do bloco de função PS_C1;

d. cada estado do ECC possui o evento de saída OUT e um algoritmo associado a ele, ALG_0 no caso do estado S_0 e ALG_1 no caso do estado S_1. Os algoritmos estão ilustrados na figura 65;

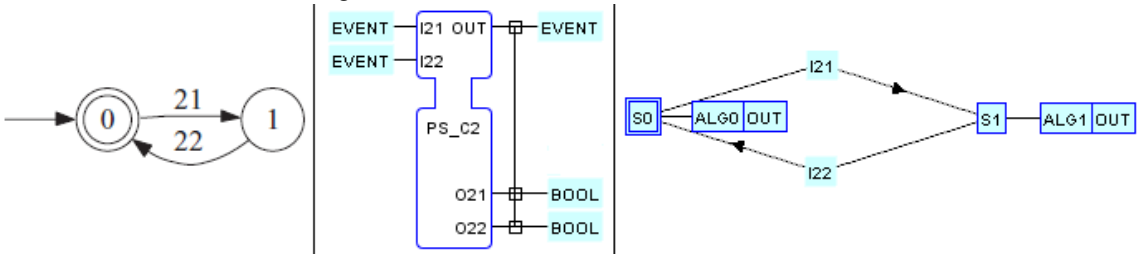
Figura 65 – Algoritmos da Esteira C1 PS_C1



fonte: do próprio autor

A figura 66 ilustra o mapeamento do autômato do subsistema da esteira C2 PS_C2. Pode-se verificar as seguintes características referentes a este autômato:

Figura 66 – Autômato da Esteira C2 PS_C2



fonte: do próprio autor

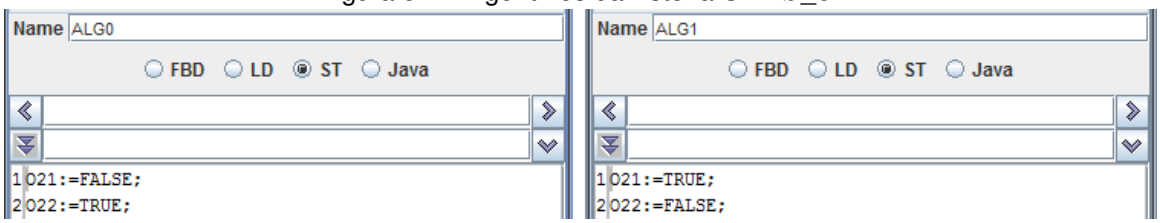
a. no bloco de função básico PS_C2 foram criados os eventos de entrada $i21$ e $i22$ e as saídas de dado $o21$ e $o22$, referentes aos eventos 21 e 22 do alfabeto Σ do autômato;

b. os estados 0 e 1 foram mapeados como os estados S_0 e S_1 do ECC do bloco de função PS_C2;

c. os eventos 21 e 22 foram mapeados como as transições $i21$ e $i22$ do ECC do bloco de função PS_C2;

d. cada estado do ECC possui o evento de saída OUT e um algoritmo associado a ele, ALG_0 no caso do estado S_0 e ALG_1 no caso do estado S_1. Os algoritmos estão ilustrados na figura 67;

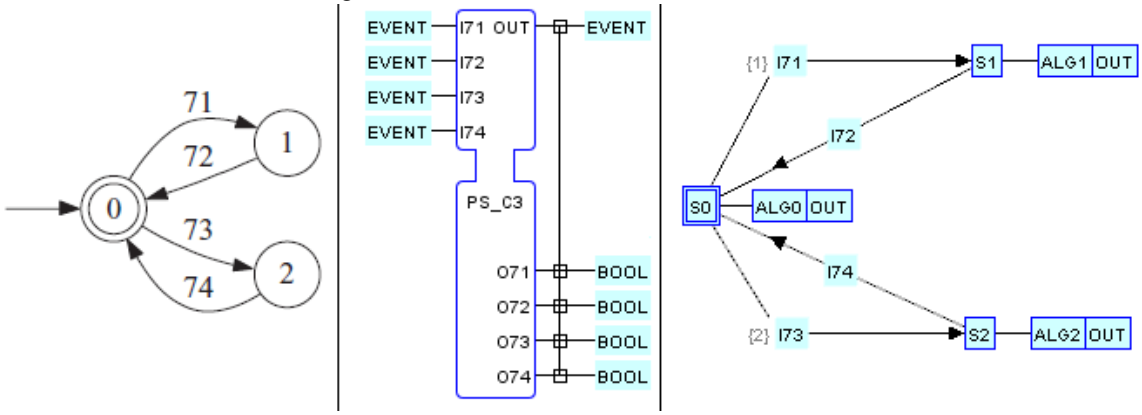
Figura 67 – Algoritmos da Esteira C2 PS_C2



fonte: do próprio autor

A figura 68 ilustra o mapeamento do autômato do subsistema da esteira C3 PS_C3. Pode-se verificar as seguintes características referentes a este autômato:

Figura 68 – Autômato da Esteira C3 PS_C3



fonte: do próprio autor

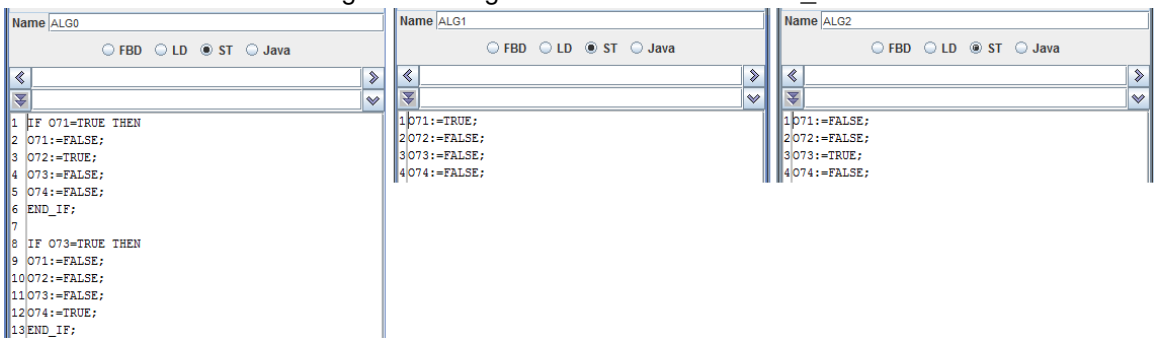
a. no bloco de função básico PS_C3 foram criados os eventos de entrada $i71$, $i72$, $i73$ e $i74$ e as saídas de dado $o71$, $o72$, $o73$ e $o74$, referentes aos eventos 71, 72, 73 e 74 do alfabeto Σ do autômato;

b. os estados 0, 1 e 2 foram mapeados como os estados S_0, S_1 e S_2 do ECC do bloco de função PS_C3;

c. os eventos 71, 72, 73 e 74 foram mapeados como as transições $i71$, $i72$, $i73$ e $i74$ do ECC do bloco de função PS_C3;

d. cada estado do ECC possui o evento de saída OUT e um algoritmo associado a ele, ALG_0 no caso do estado S_0, ALG_1 no caso do estado S_1 e ALG_2 no caso do estado S_2. Os algoritmos estão ilustrados na figura 69;

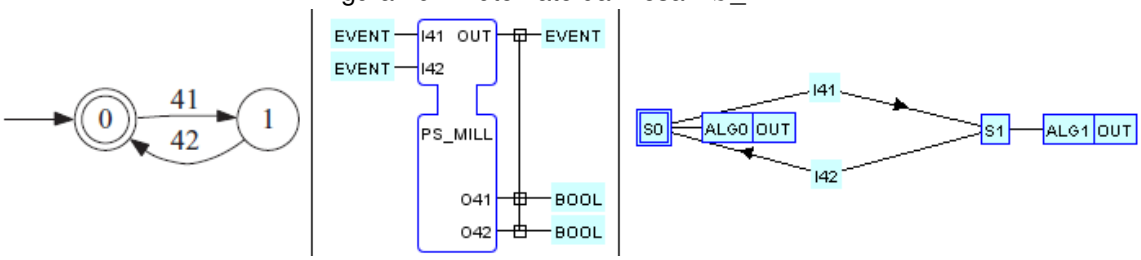
Figura 69 – Algoritmos da Esteira C3 PS_C3



fonte: do próprio autor

A figura 70 ilustra o mapeamento do autômato do subsistema da fresa PS_MILL. Pode-se verificar as seguintes características referentes a este autômato:

Figura 70 – Autômato da Fresa PS_MILL



fonte: do próprio autor

a. no bloco de função básico PS_MILL foram criados os eventos de entrada $i41$ e $i42$ e as saídas de dado $o41$ e $o42$, referentes aos eventos 41 e 42 do alfabeto Σ do autômato;

b. os estados 0 e 1 foram mapeados como os estados S_0 e S_1 do ECC do bloco de função PS_MILL;

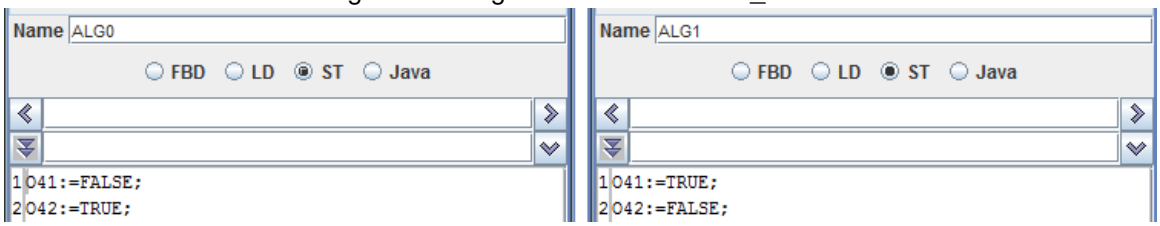
c. os eventos 41 e 42 foram mapeados como as transições $i41$ e $i42$ do ECC do bloco de função PS_MILL;

d. cada estado do ECC possui o evento de saída OUT e um algoritmo associado a ele, ALG_0 no caso do estado S_0 e ALG_1 no caso do estado S_1. Os algoritmos estão ilustrados na figura 71;

A figura 72 ilustra o mapeamento do autômato do subsistema do torno PS_LATHE. Pode-se verificar as seguintes características referentes a este autômato:

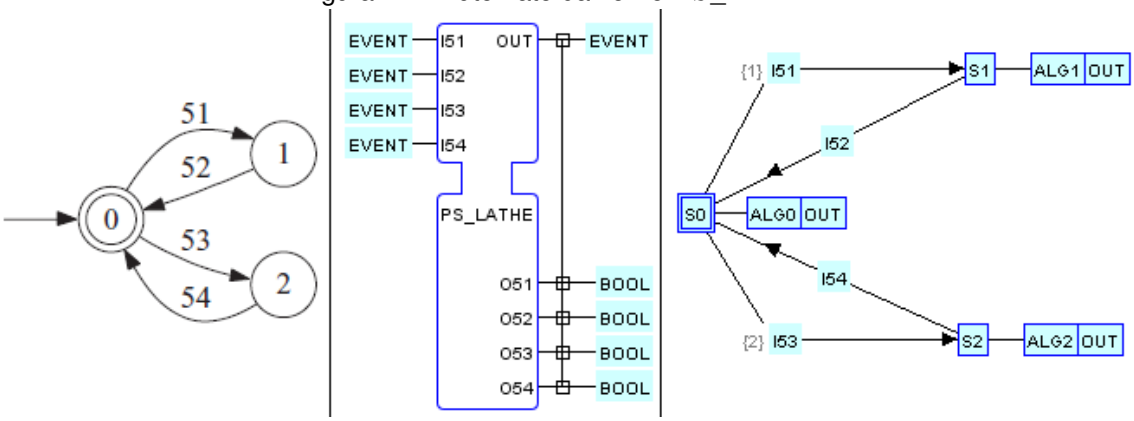
a. no bloco de função básico PS_LATHE foram criados os eventos de entrada $i51$, $i52$, $i53$ e $i54$ e as saídas de dado $o51$, $o52$, $o53$ e $o54$, referentes aos eventos

Figura 71 – Algoritmos da Fresa PS_MILL



fonte: do próprio autor

Figura 72 – Autômato da Torno PS_LATHE



fonte: do próprio autor

51, 52, 53 e 54 do alfabeto Σ do autômato;

b. os estados 0, 1 e 2 foram mapeados como os estados S_0, S_1 e S_2 do ECC do bloco de função PS_LATHE;

c. os eventos 51, 52, 53 e 54 foram mapeados como as transições $i51$, $i52$, $i53$ e $i54$ do ECC do bloco de função PS_LATHE;

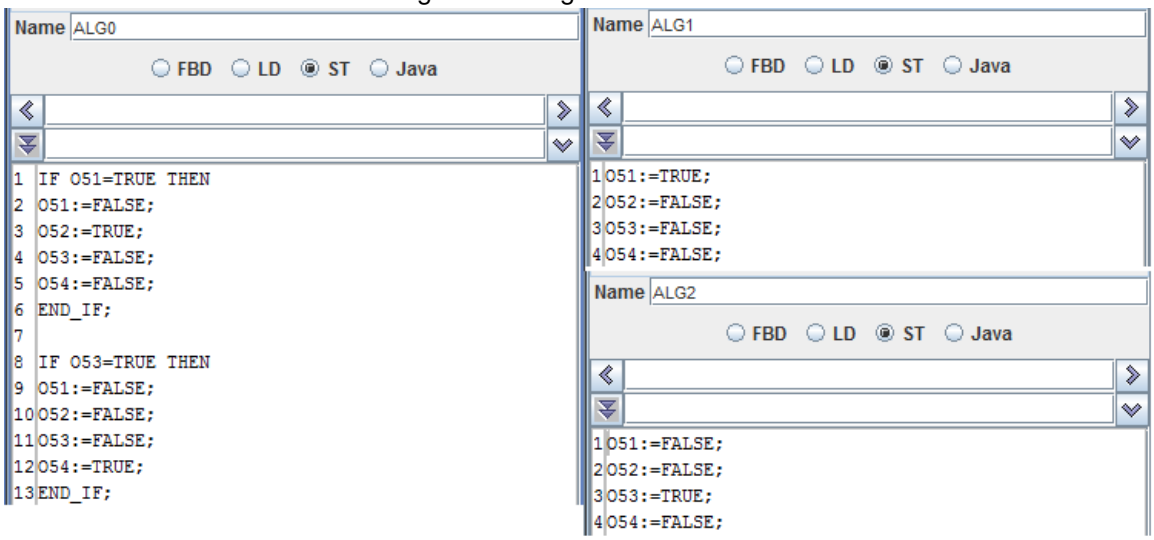
d. cada estado do ECC possui o evento de saída OUT e um algoritmo associado a ele, ALG_0 no caso do estado S_0, ALG_1 no caso do estado S_1 e ALG_2 no caso do estado S_2. Os algoritmos estão ilustrados na figura 73;

A figura 74 ilustra o mapeamento do autômato do subsistema do robô PS_ROBOT. Pode-se verificar as seguintes características referentes a este autômato:

a. no bloco de função básico PS_ROBOT foram criados os eventos de entrada $i31$, $i32$, $i33$, $i34$, $i35$, $i36$, $i37$, $i38$, $i39$ e $i30$ e as saídas de dado $o31$, $o32$, $o33$, $o34$, $o35$, $o36$, $o37$, $o38$, $o39$ e $o30$, referentes aos eventos 31, 32, 33, 34, 35, 36, 37, 38, 39 e 30 do alfabeto Σ do autômato;

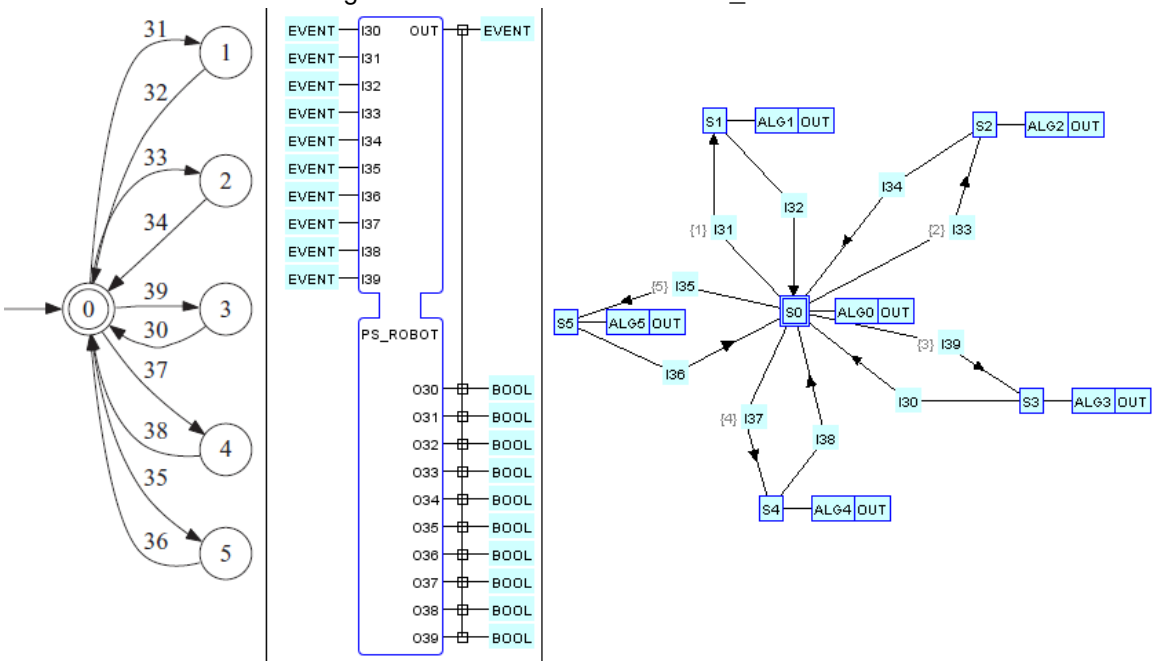
b. os estados 0, 1, 2, 3, 4 e 5 foram mapeados como os estados S_0, S_1, S_2, S_3,

Figura 73 – Algoritmos da Torno



fonte: do próprio autor

Figura 74 – Autômato da Robô PS_ROBOT



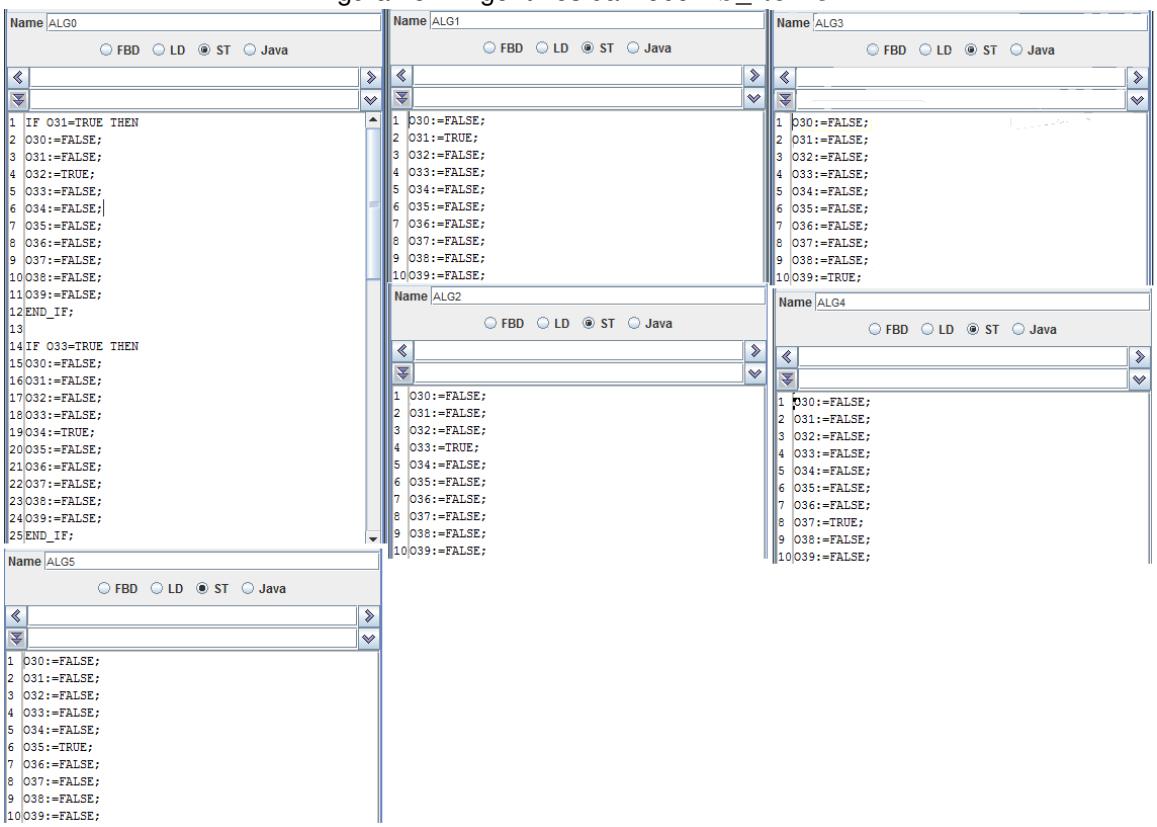
fonte: do próprio autor

S_4 e S_5 do ECC do bloco de função PS_ROBOT;

c. os eventos 31, 32, 33, 34, 35, 36, 37, 38, 39 e 30 foram mapeados como as transições $i31$, $i32$, $i33$, $i34$, $i35$, $i36$, $i37$, $i38$, $i39$ e $i30$ do ECC do bloco de função PS_ROBOT;

d. cada estado do ECC possui o evento de saída OUT e um algoritmo associado a ele, ALG_0 no caso do estado S_0, ALG_1 no caso do estado S_1, ALG_2 no caso do estado S_2, ALG_3 no caso do estado S_3, ALG_4 no caso do estado S_4 e ALG_5 no caso do estado S_5. Os algoritmos estão ilustrados na figura 75;

Figura 75 – Algoritmos da Robô PS_ROBOT



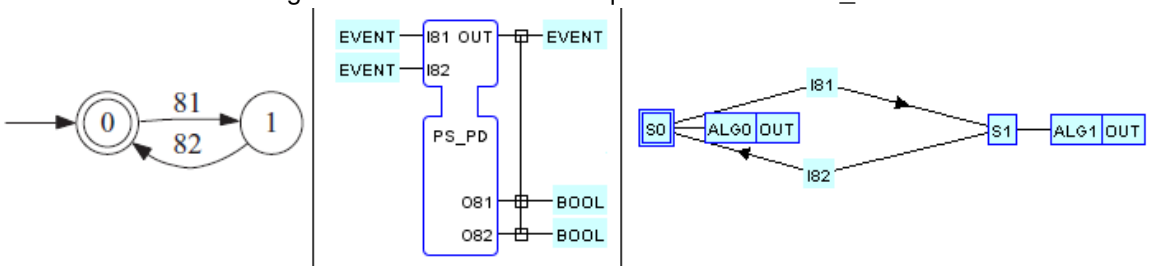
fonte: do próprio autor

A figura 76 ilustra o mapeamento do autômato do subsistema da máquina de pintura PS_PD. Pode-se verificar as seguintes características referentes a este autômato:

a. no bloco de função básico PS_PD foram criados os eventos de entrada $i81$ e $i82$ e as saídas de dado $o81$ e $o82$, referentes aos eventos 81 e 82 do alfabeto Σ do autômato;

b. os estados 0 e 1 foram mapeados como os estados S_0 e S_1 do ECC do bloco

Figura 76 – Autômato da Máquina de Pintura PS_PD



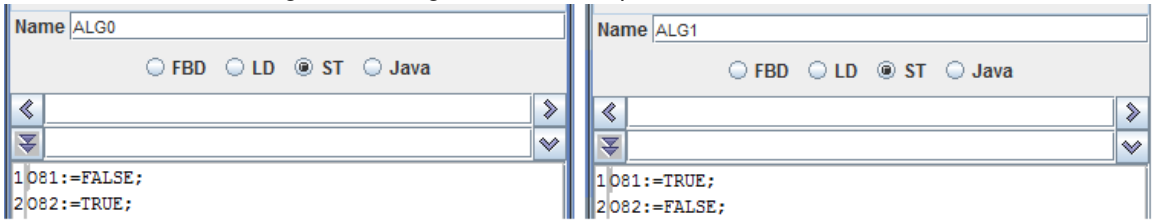
fonte: do próprio autor

de função PS_PD;

c. os eventos 81 e 82 foram mapeados como as transições $i81$ e $i82$ do ECC do bloco de função PS_PD;

d. cada estado do ECC possui o evento de saída OUT e um algoritmo associado a ele, ALG_0 no caso do estado S_0 e ALG_1 no caso do estado S_1. Os algoritmos estão ilustrados na figura 77;

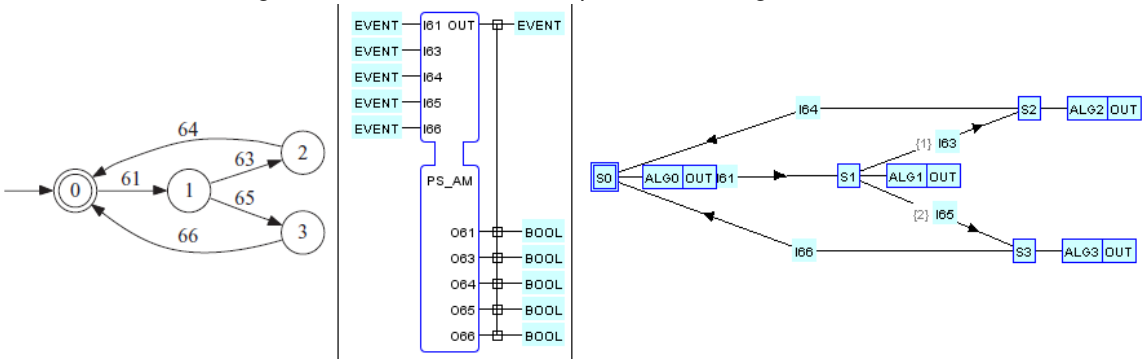
Figura 77 – Algoritmos da Máquina de Pintura PS_PD



fonte: do próprio autor

A figura 78 ilustra o mapeamento do autômato do subsistema da máquina de montagem PS_AM. Pode-se verificar as seguintes características referentes a este autômato:

Figura 78 – Autômato da Máquina de Montagem PS_AM



fonte: do próprio autor

a. no bloco de função básico PS_AM foram criados os eventos de entrada $i61$, $i63$, $i64$, $i65$ e $i66$ e as saídas de dado $o61$, $o63$, $o64$, $o65$ e $o66$, referentes aos

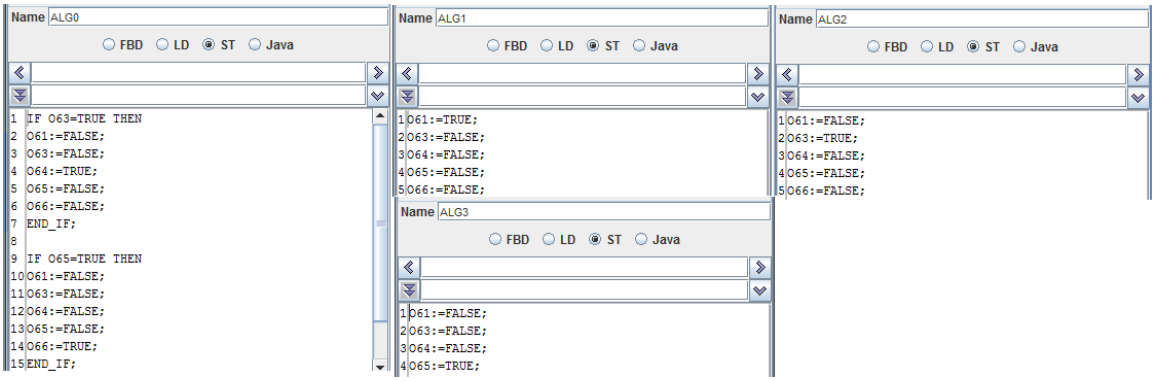
eventos 61, 63, 64, 65 e 66 do alfabeto Σ do autômato;

b. os estados 0, 1, 2 e 3 foram mapeados como os estados S_0, S_1, S_2 e S_3 do ECC do bloco de função PS_AM;

c. os eventos 61, 63, 64, 65 e 66 foram mapeados como as transições $i61$, $i63$, $i64$, $i65$ e $i66$ do ECC do bloco de função PS_AM;

d. cada estado do ECC possui o evento de saída OUT e um algoritmo associado a ele, ALG_0 no caso do estado S_0, ALG_1 no caso do estado S_1, ALG_2 no caso do estado S_2 e ALG_3 no caso do estado S_3. Os algoritmos estão ilustrados na figura 79;

Figura 79 – Algoritmos da Máquina de Montagem PS_AM



fonte: do próprio autor

4.1.3 Mapeamento das sequências operacionais

Na implementação do sistema flexível de manufatura, como somente há comandos simples de operação no nível do sistema produto, como por exemplo, início de operação do torno e fim de operação do robô, não houve necessidade de implementar qualquer algoritmo no nível das sequências operacionais.

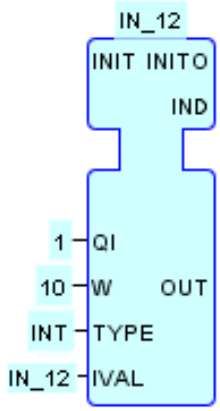
4.1.4 Mapeamento da leitura das entradas

A figura 80 ilustra o SIFB para leitura da entrada digital referente ao evento não controlável 12. Pode-se verificar as seguintes características:

a. o SIFB de leitura de entrada possui o evento de entrada $INIT$ e os eventos de saída $INITO$ e IND;

b. o SIFB de leitura de entrada possui o dado de saída OUT;

Figura 80 – SIFB para leitura da entrada digital referente ao evento não controlável 12



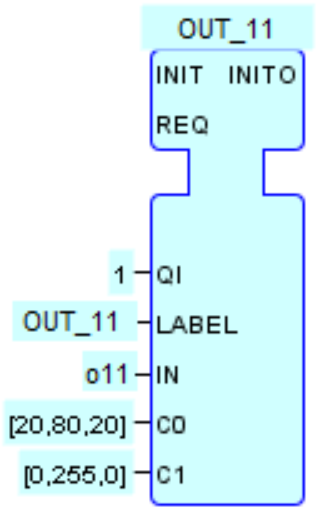
fonte: do próprio autor

c. as variáveis de dados de entrada QI, W, $TYPE$ e $IVAL$ possuem, respectivamente, os seguintes parâmetros: 1, 10, INT e IN_12;

4.1.5 Mapeamento da escrita nas saídas

A figura 81 ilustra o SIFB para escrita na saída digital referente ao evento controlável 11. Pode-se verificar as seguintes características:

Figura 81 – bloco de função de interface de serviço para escrita na saída digital referente aos eventos controláveis do Robô



fonte: do próprio autor

a. o SIFB de escrita na saída possui os eventos de entrada $INIT$ e REQ e o evento de saída $INITO$;

b. as variáveis de dados de entrada QI, $LABEL$, IN, $C0$ e $C1$ possuem, respectivamente, os seguintes parâmetros: 1, OUT_11, o_11, $[20, 80, 20]$ e $[0, 255, 0]$;

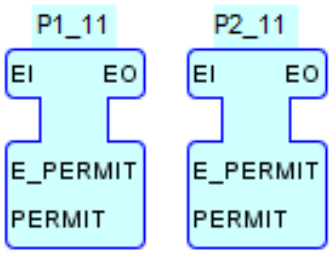
4.1.6 Mapeamento de FBs básicos auxiliares

Para implementar o estudo de caso foi necessário utilizar os três modelos de FBs básicos, conforme estão descritos a seguir.

4.1.6.1 Bloco de função básico do tipo PERMIT

Para cada evento σ_i do sistema flexível de manufatura foram criados dois FBs do tipo PERMIT respectivos. Por exemplo, para o evento controlável 11, criou-se os FBs do tipo PERMIT $P1\_11$ e $P2\_11$, conforme ilustrado na figura 82.

Figura 82 – Blocos de função do tipo permit $P1\_11$ e $P2\_11$

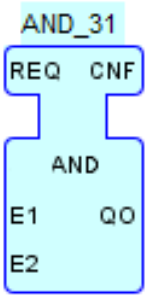


fonte: do próprio autor

4.1.6.2 Bloco de função básico do tipo AND

Para cada evento controlável σ_c que pertence ao alfabeto Σ de mais de um supervisor modular local reduzido criou-se um FB básico do tipo AND. Por exemplo, no sistema de manufatura flexível, os eventos controláveis 31, 33, 35, 37 e 39 pertencem ao alfabeto Σ de mais de um supervisor modular local reduzido, logo foram criados os FBs AND_31, AND_33, AND_35, AND_37 e AND_39, conforme ilustrado na figura 83.

Figura 83 – Bloco de função AND_31



fonte: do próprio autor

4.1.6.3 Bloco de função básico de inicialização

Para o sistema flexível de manufatura, utilizou-se um FB básico do tipo E_RESTART com o objetivo de inicializar a aplicação, conforme ilustrado na figura 84.

Figura 84 – Bloco de função do tipo E_RESTART

fonte: do próprio autor

4.2 INTERLIGAÇÕES ENTRE OS BLOCOS DE FUNÇÃO CRIADOS NA PRIMEIRA ETAPA

Nas próximas subseções será descrito o processo de conexão das variáveis de dados e dos eventos entre os blocos de função.

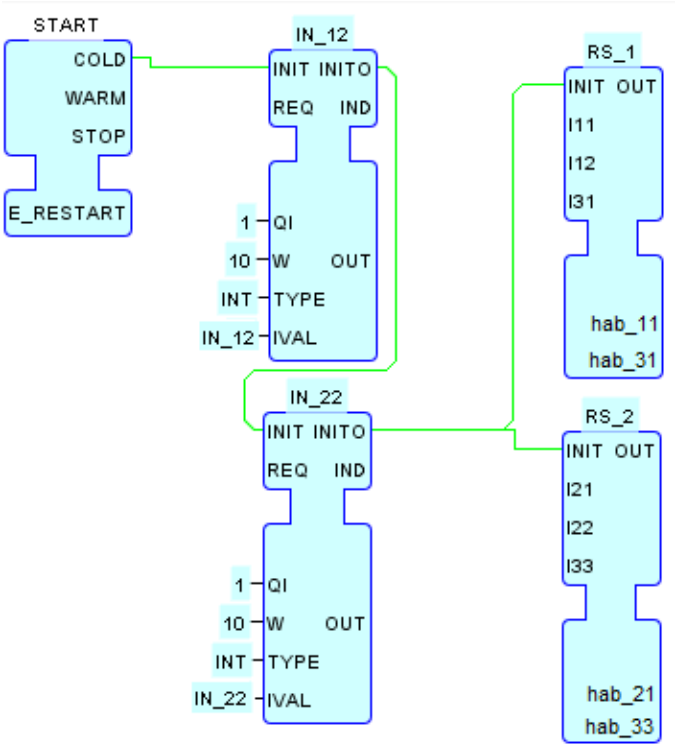
4.2.1 Inicialização dos blocos de função

No sistema flexível de manufatura, a parte implementada referente a esse passo consiste em conectar o evento $COLD$ do FB básico $E\_RESTART$ de forma sequencialmente a todos as entradas e saídas de eventos $INIT$ e $INITO$ dos SIFBs de leitura de entrada IN_σ_u, dos FBs dos supervisores modulares locais reduzidos RS_j e dos SIFBs de escrita nas saídas OUT_σ_c. Como a simulação ficou com um tamanho que não é possível representá-la em sua totalidade aqui (verificar Apêndice C), colocou-se parte da implementação desse passo, conforme ilustrado na figura 85. Por exemplo, a saída de evento $INITO$ do SIFB IN_12 está conectada a entrada de evento $INIT$ do SIFB IN_12.

4.2.2 Interligações entre SIFBs de leitura de entrada e FBs do tipo PERMIT P1

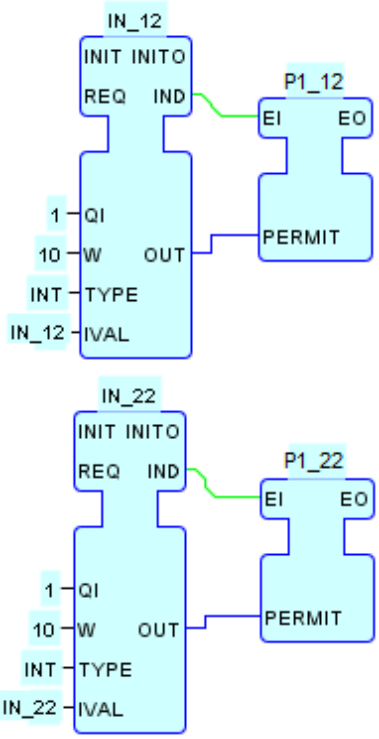
No sistema flexível de manufatura, a parte implementada referente a esse passo consiste em conectar o evento de saída IND e o dado de saída OUT de cada SIFB de leitura de entrada IN_σ_u as respectivas entradas EI e $PERMIT$ do seu respectivo FB básico do tipo PERMIT $P1\_\sigma$. Como a simulação ficou com um tamanho que não é possível representá-la em sua totalidade aqui, colocou-se parte da implementação desse passo, conforme ilustrado na figura 86. Por exemplo, a saída de evento IND do SIFB IN_12 está conectada a entrada de evento EI do FB do tipo PERMIT $P1\_12$.

Figura 85 – inicialização dos blocos de função



fonte: do próprio autor

Figura 86 – interligações entre SIFBs de leitura de entrada e FBs do tipo PERMIT P1

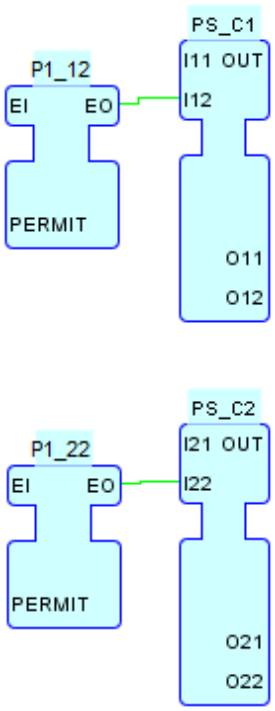


fonte: do próprio autor

4.2.3 Interligações entre FBs do tipo PERMIT P1 e FBs do sistema produto

No sistema flexível de manufatura, a parte implementada referente a esse passo consiste em conectar a saída de evento $E0$ dos FBs do tipo PERMIT $P1\_\sigma$ as respectivas entradas de eventos $i\sigma$ dos seus respectivos FBs PS_Gi. Como a simulação ficou com um tamanho que não é possível representá-la em sua totalidade aqui, colocou-se parte da implementação desse passo, conforme ilustrado na figura 87. Por exemplo, a saída de evento $E0$ do FB do tipo PERMIT $P1\_11$ está conectada a entrada de evento $i11$ do FB PS_C1.

Figura 87 – interligações entre FBs do tipo PERMIT P1 e FBs do sistema produto

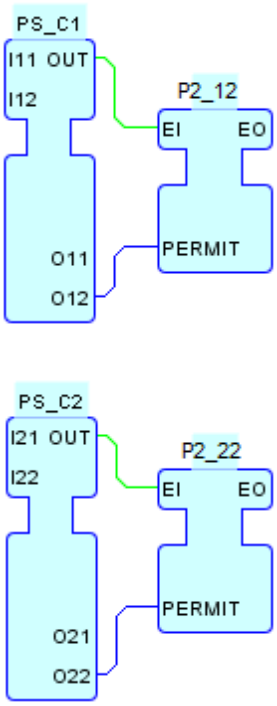


fonte: do próprio autor

4.2.4 Interligações entre FBs do sistema produto e FBs do tipo PERMIT P2

No sistema flexível de manufatura, a parte implementada referente a esse passo consiste em conectar o evento de saída OUT e as variáveis de dados $o\sigma$ as respectivas entradas de evento EI e entrada de dados $PERMIT$ dos seus respectivos FBs do tipo PERMIT $P2\_\sigma$. Como a simulação ficou com um tamanho que não é possível representá-la em sua totalidade aqui, colocou-se parte da implementação desse passo, conforme ilustrado na figura 88. Por exemplo, a saída de evento OUT e a saída de dado o_{12} do FB PS_C1 estão conectados a entrada de evento EI e a entrada de dado PERMIT do FB $P2\_12$.

Figura 88 – interligações entre FBs do sistema produto e FBs do tipo PERMIT P2



fonte: do próprio autor

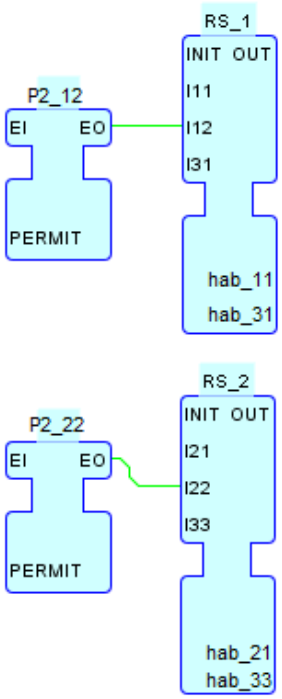
4.2.5 Interligações entre FBs do tipo PERMIT P2 e FBs dos supervisores modulares locais reduzidos

No sistema flexível de manufatura, a parte implementada referente a esse passo consiste em conectar a saída de evento $E0$ dos FBs do tipo PERMIT $P2\_\sigma$ as respectivas entradas de eventos $i\sigma$ dos seus respectivos FBs RS_σ_j. Como a simulação ficou com um tamanho que não é possível representá-la em sua totalidade aqui, colocou-se parte da implementação desse passo, conforme ilustrado na figura 89. Por exemplo, a saída de evento $E0$ do FB do tipo PERMIT $P2\_12$ está conectada a entrada de evento i_{12} do FB RS_1.

4.2.6 Interligações entre FBs dos supervisores modulares locais reduzidos com FBs do tipo AND e/ou FBs do tipo PERMIT P1

No sistema flexível de manufatura, a parte implementada referente a esse passo consiste em conectar a saída de evento OUT e as saídas de dados hab_σ_c dos FBs RS_j a entrada de evento EI e a entrada de dado $PERMIT$ de seus respectivos FBs do tipo PERMIT $P1\_\sigma$. Se a saída de dado hab_σ_c for compartilhada em mais de um supervisor, nesse caso a mesma será direcionada primeiramente a entrada de dado E do seu respectivo FB do tipo AND AND_σ. Como a simulação ficou com um tamanho que não é possível representá-la em sua totalidade aqui (verificar Figura 102 no apêndice C), colocou-se parte da implementação desse passo, conforme ilustrado na figura 90. Por exemplo, a saída de evento OUT e a saída de dado hab_31 do FB

Figura 89 – interligações entre FBs do tipo PERMIT P2 e FBs dos supervisores modulares locais reduzidos



fonte: do próprio autor

RS_1 estão conectados a entrada de evento REQ e a entrada de dado $E1$ do FB AND_31.

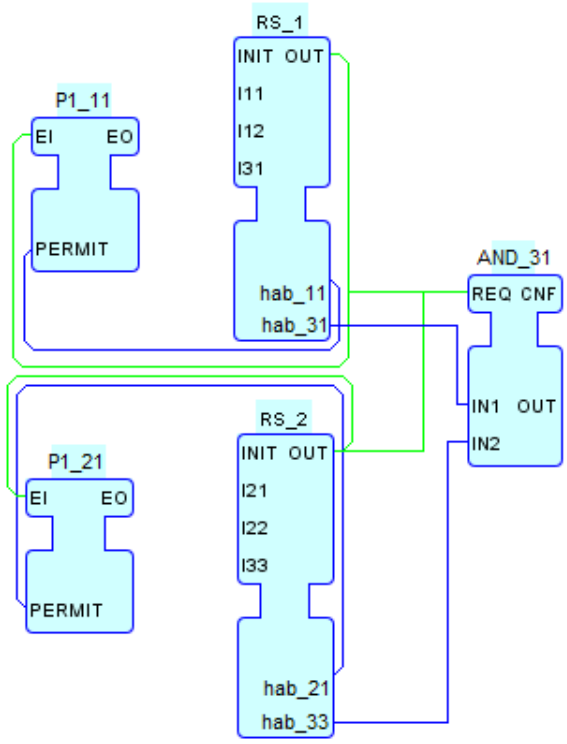
4.2.7 Interligações entre FBs do sistema produto com SIFBs de escrita nas saídas

No sistema flexível de manufatura, a parte implementada referente a esse passo consiste em conectar a saída de evento OUT e a saída de dado $o\sigma_c$ referente aos eventos controláveis dos FBs PS_Gi ao evento de entrada REQ e ao dado de entrada IN de seus respectivos SIFBs OUT_σ_c. Como a simulação ficou com um tamanho que não é possível representá-la em sua totalidade aqui, colocou-se parte da implementação desse passo, conforme ilustrado na figura 91. Por exemplo, a saída de evento OUT e a saída de dado $o11$ do FB PS_C1 estão conectados a entrada de evento REQ e a entrada de dado IN do FB OUT_11.

4.3 SIMULAÇÕES

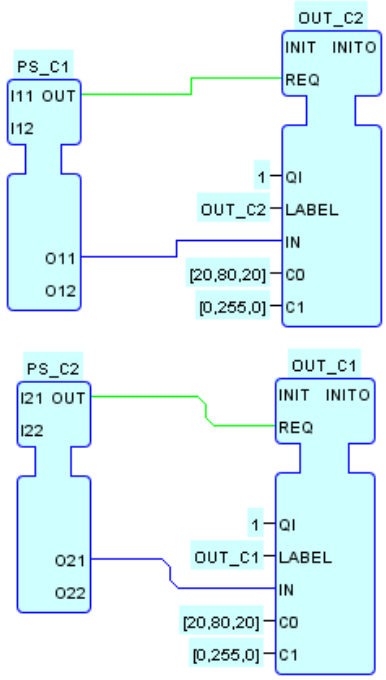
Após a aplicação das etapas um e dois da metodologia de implementação da teoria de controle supervisório modular local aderente à IEC 61499, desenvolvida nesse trabalho, ao sistema flexível de manufatura, obtiveram-se os blocos de função conforme imagens ilustradas no apêndice B. As imagens estão separadas de acordo

Figura 90 – interligações entre FBs dos supervisores modulares locais reduzidos com FBs do tipo AND e/ou FBs do tipo PERMIT P1



fonte: do próprio autor

Figura 91 – interligações entre FBs do sistema produto com SIFBs de escrita nas saídas



fonte: do próprio autor

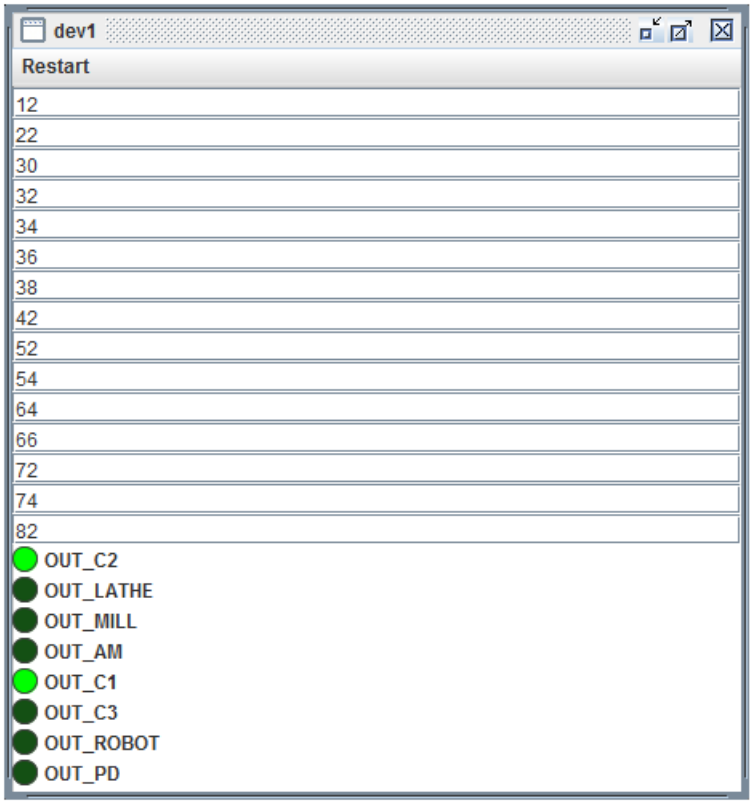
com as colunas definidas pelo modelo proposta no capítulo três, conforme ilustrado na figura 27.

O sistema flexível de manufatura foi simulado no FBDK/FBRT, seguindo a metodologia proposta no capítulo 3 desse trabalho.

Na figura 92 se pode verificar uma imagem do sistema sendo executado. Nessa imagem, as caixas de textos a cima são responsáveis pela geração manual dos eventos não controláveis na hora da simulação, e os LEDs indicativos a baixo representam se cada máquina está operando ou não.

Como o sistema está em estado inicial, pode-se notar que os LEDs das esteiras C1 e C2 estão acesos, indicando que as peças estão sendo transportadas.

Figura 92 – Simulação do sistema flexível de manufatura



fonte: do próprio autor

Após a inicialização do sistema, foram simuladas diferentes etapas para verificar se a metodologia desenvolvida realmente funcionava. Os mesmos foram descritos a seguir, de acordo com a figura 93:

1. Etapa A: etapa de inicialização do sistema, ou seja, no momento que se ligou o sistema, os eventos controláveis 11 e 21 foram automaticamente executados, pois estavam habilitados pelos supervisores. Os LEDs com valor booleano $TRUE$ resultantes dessa operação são OUT_C1 e OUT_C2, indicando que uma peça começou

a ser transportada tanto na esteira C1 como na esteira C2;

2. Etapa B: após a inicialização do sistema, forçou-se um evento não controlável 12 (fim da esteira C1). Logo, a peça que estava sendo transportada em C1 foi colocada no buffer B1 e, como o robô estava em *stand by*, rapidamente o evento controlável 31 (robô pega peça em B1) foi automaticamente executado. Os LEDS com valor booleano $TRUE$ resultantes dessa operação são OUT_C1 (uma peça continua se movendo na esteira C1), OUT_C2 (outra peça começou a se mover na esteira C2) e OUT_ROBOT (o robô está agora ocupado com uma peça que chegou da esteira C2);

3. Etapa C: após a etapa anterior, forçou-se então o evento não controlável 32 (robô coloca peça no buffer B3). Já na sequência, automaticamente aconteceu o evento controlável 41 (Fresa pega peça em B3), pois o mesmo estava habilitado pelos supervisores. Os LEDS com valor booleano $TRUE$ resultantes dessa operação são OUT_C1 (uma peça continua se movendo na esteira C1), OUT_C2 (outra peça continua a se mover na esteira C2) e OUT_MILL (a fresa está agora ocupada com uma peça que chegou do buffer B3);

4. Etapa D: após a etapa anterior, forçou-se então o evento não controlável 42 (fim de operação da fresa), resultando numa peça novamente no buffer B3. Já na sequência, automaticamente aconteceu o evento controlável 35 (robô pega peça de B3), pois o mesmo estava habilitado pelos supervisores. Os LEDS com valor booleano $TRUE$ resultantes dessa operação são OUT_C1 (uma peça continua se movendo na esteira C1), OUT_C2 (outra peça continua a se mover na esteira C2) e OUT_ROBOT (o robô está novamente ocupado com uma peça que chegou do buffer B3);

5. Etapa E: após a etapa anterior, forçou-se então o evento não controlável 36 (fim do robô), resultando numa peça no buffer B5. Já na sequência, automaticamente aconteceu o evento controlável 61 (máquina de montagem pega peça do buffer B5), pois o mesmo estava habilitado pelos supervisores. Os LEDS com valor booleano $TRUE$ resultantes dessa operação são OUT_C1 (uma peça continua se movendo na esteira C1), OUT_C2 (outra peça continua a se mover na esteira C2) e OUT_AM (a máquina de montagem está ocupada agora com uma peça que chegou do buffer B5);

6. Etapa F: após a etapa anterior, forçou-se então o evento não controlável 22 (fim da esteira C2), resultando numa peça no buffer B2. Já na sequência, auto-

maticamente aconteceu o evento controlável 33 (robô pega peça do buffer B2), pois o mesmo estava habilitado pelos supervisores. Os LEDS com valor booleano $TRUE$ resultantes dessa operação são OUT_C1 (uma peça nova começa a se mover na esteira C1), OUT_C2 (outra peça continua a se mover na esteira C2), OUT_AM (a máquina de montagem continua ocupada com uma peça que chegou do buffer B5) e OUT_ROBOT (o robô está agora ocupado com uma peça que chegou da esteira C2);

7. Etapa G: após a etapa anterior, forçou-se então o evento não controlável 34 (fim do robô), resultando numa peça no buffer B4. Já na sequência, automaticamente aconteceu o evento controlável 51 (liga o Torno), pois o mesmo estava habilitado pelos supervisores. Os LEDS com valor booleano $TRUE$ resultantes dessa operação são OUT_C1 (uma peça continua a se mover na esteira C1), OUT_C2 (outra peça continua a se mover na esteira C2), OUT_AM (a máquina de montagem continua ocupada com uma peça que chegou do buffer B5) e OUT_LATHE (o Torno está agora ocupado com uma peça que chegou do buffer B4);

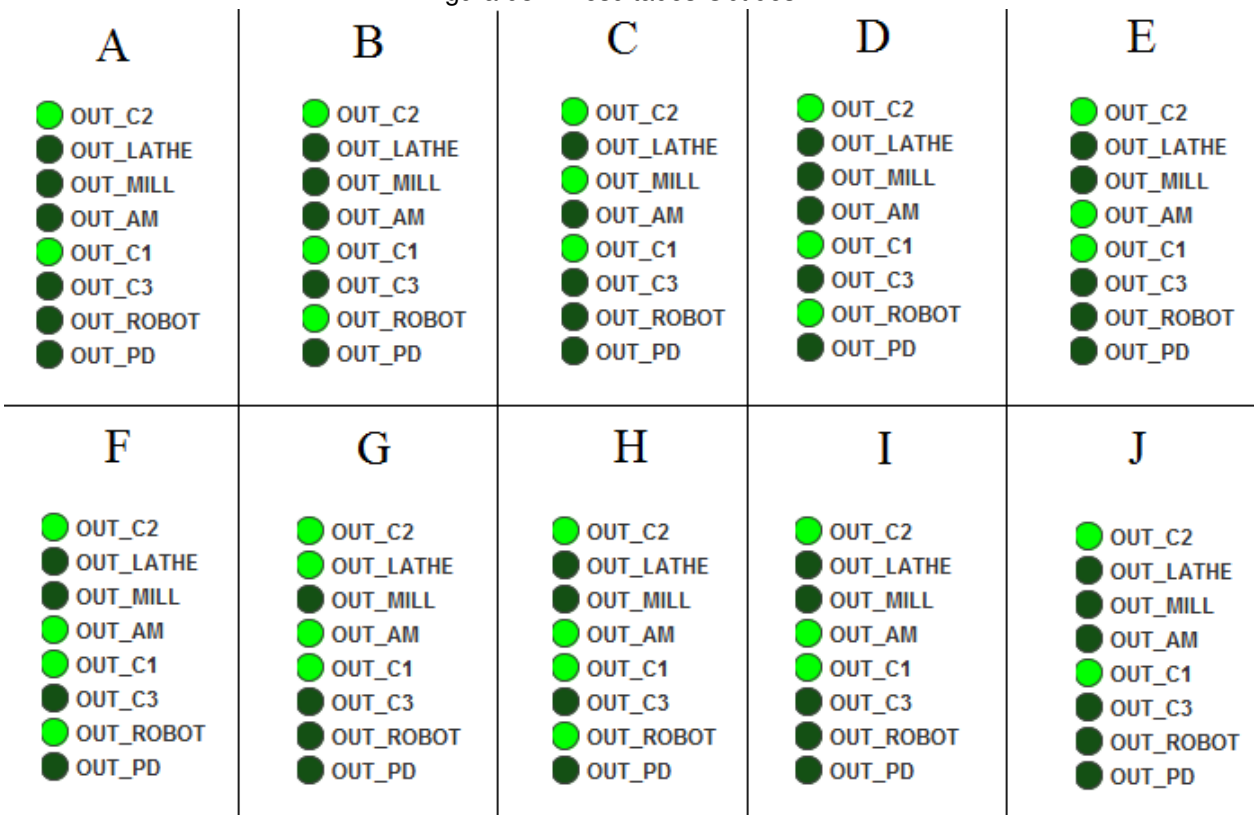
8. Etapa H: após a etapa anterior, forçou-se então o evento não controlável 52 (fim do Torno), resultando numa peça novamente no buffer B4. Já na sequência, automaticamente aconteceu o evento controlável 37 (robô pega peça do buffer B4), pois o mesmo estava habilitado pelos supervisores. Os LEDS com valor booleano $TRUE$ resultantes dessa operação são OUT_C1 (uma peça continua a se mover na esteira C1), OUT_C2 (outra peça continua a se mover na esteira C2), OUT_AM (a máquina de montagem continua ocupada com uma peça que chegou do buffer B5) e OUT_ROBOT (o robô está agora ocupado com uma peça que chegou do buffer B4);

9. Etapa I: após a etapa anterior, forçou-se então o evento não controlável 38 (fim do Robô), resultando numa peça no buffer B6. Já na sequência, automaticamente aconteceu o evento controlável 63 (máquina de montagem pega peça do buffer B6), pois o mesmo estava habilitado pelos supervisores. Os LEDS com valor booleano $TRUE$ resultantes dessa operação são OUT_C1 (uma peça continua a se mover na esteira C1), OUT_C2 (outra peça continua a se mover na esteira C2 e OUT_AM (a máquina de montagem agora está ocupada com uma peça que chegou do buffer B5 e outra que chegou do buffer B6);

10. Etapa J: após a etapa anterior, forçou-se então o evento não controlável 64 (fim de operação da máquina de montagem), resultando num produto do tipo A. Os LEDS com valor booleano $TRUE$ resultantes dessa operação são OUT_C1 (uma peça continua a se mover na esteira C1) e OUT_C2 (outra peça continua a se mover

na esteira C2. Esse cenário se assemelha com a etapa A, ou seja, um novo ciclo automaticamente se iniciou, o qual irá resultar novamente num produto do tipo A ou num produto do tipo B;

Figura 93 – Resultados Obtidos



fonte: do próprio autor

Os resultados obtidos nas simulações desse estudo de caso foram importantes para demonstrar a funcionalidade da metodologia de controle supervisório modular local aderente à IEC 61499 desenvolvida nesse trabalho, porém, como não foram realizadas também simulações com outros exemplos de SEDs, não é possível comprovar que a metodologia é totalmente funcional e não apresenta falhas.

Outro ponto importante a ser comentado é que para SEDs com outras características específicas, como os que possuem retroalimentação ou distinguidores, por exemplo, não é possível tirar conclusões, visto que não foram feito estudos referentes a tais sistemas.

4.4 RESUMO DO CAPÍTULO

O estudo de caso apresentado nesse capítulo foi importante para demonstrar a funcionalidade da metodologia desenvolvida para mapeamento e implementação

do código de controle de um sistema a eventos discretos modelado pela Teoria de Controle Supervisório Modular Local na IEC 61499.

Inicialmente foi detalhado o processo de mapeamento dos autômatos do sistema produto e dos supervisores modulares nos blocos de função da IEC 61499.

Na sequência, foi detalhado o processo de conexões das variáveis de dados e dos eventos entre os blocos de função, resultando assim num controle em malha fechada do sistema.

Ao final do capítulo foram apresentados os resultados das simulações.

5 CONCLUSÕES

Neste trabalho foi apresentada uma metodologia de implementação do controle supervisório modular local aderente à norma IEC 61499. O método é baseado em duas etapas principais, a primeira consiste no mapeamento da estrutura de controle modular local nos blocos de função da IEC 61499, e a segunda consiste nas interligações entre os blocos de função criados na primeira etapa.

Foram apresentados alguns trabalhos relacionados que abordam principalmente a implementação do código de controle através dos blocos de função da IEC 61499, mas somente para uma abordagem de controle monolítico, na qual se trabalha com um único supervisor, que para sistemas maiores traz sérios problemas, como, por exemplo, dificuldades para implementação do código de controle, demora de processamento, falta de memória, dificuldade de entendimento da lógica de controle, etc. Outros trabalhos abordaram a implementação da estrutura modular local, porém apenas considerando a norma IEC 61131. Nesse cenário, observou-se a carência de um método de implementação da teoria de controle supervisório modular local aderente à norma IEC 61499. Através do trabalho desenvolvido messa dissertação, criou-se uma metodologia funcional que implementa a estrutura modular local da Teoria de Controle Supervisório de SEDs nos FBs da norma IEC 61499, o que abre novas possibilidades para pesquisas envolvendo as duas áreas.

A metodologia aplicada ao sistema de transferência industrial foi simulada no ambiente de programação e simulação FBDK/FBRT. Os resultados obtidos foram todos funcionais, ou seja, de acordo com o modelo de controle ótimo obtido através da aplicação da abordagem de controle modular local, sem que houvesse travamento, redundância ou qualquer estado restritivo na lógica de controle, demonstrando que a metodologia implementa o código de forma minimamente restritiva, assim como proposta pela teoria de controle supervisório.

Um estudo de caso de maior complexidade, considerando número e tamanho dos autômatos, foi implementado no capítulo quatro para verificar a replicabilidade da metodologia de implementação do controle supervisório modular local aderente à norma IEC 61499 desenvolvida nesse trabalho. As etapas um e dois da metodologia foram aplicadas e, novamente, simulou-se o sistema no FBDK/FBRT. Os resultados obtidos novamente foram funcionais e conforme o esperado pela aplicação da abordagem modular local. No entanto, como não foram realizadas também simulações com outros exemplos de SEDs, não é possível comprovar que a metodologia é totalmente funcional e não apresenta falhas.

Outro ponto importante a ser comentado é que para SEDs com outras características específicas, como os que possuem retroalimentação ou distinguidores, por exemplo, não é possível tirar conclusões, visto que não foram feito estudos referentes a tais sistemas.

Conclui-se também que todo o processo de implementação da metodologia pode ser automatizado, desde a geração dos blocos de função até as interligações entre os mesmos, pois os passos para implementação são sempre os mesmos, segue-se uma regra. Isso é importante para gerar processos automatizados na indústria para que a mesma possa se tornar mais eficiente em seus processos.

Como ponto negativo da metodologia, pode-se verificar que para sistemas com grande número de autômatos, estados e eventos, o número de blocos de função se torna demasiadamente grande, dificultando a implementação. Como solução, é sugerido como estudo futuro a utilização de blocos de função compostos com o objetivo de diminuir a quantidade de FBs do tipo básico, facilitando assim o entendimento e a implementação da metodologia.

5.1 TRABALHOS FUTUROS

A seguir são apresentadas as recomendações para futuros trabalhos no tema, que foram identificadas durante a realização deste trabalho:

a. Esse trabalho serve como base para avanços nas pesquisas referentes à reconfiguração dinâmica que usam a teoria de controle supervisório, já que a maior parte dos trabalhos sobre esse tema utilizam ainda a abordagem monolítica da teoria de controle supervisório, indo contra a distribuição do código, proposto pela norma IEC 61499. Dessa forma, através da metodologia aqui proposta, avançar para o estudo de estados seguros na abordagem de controle modular local;

b. Estudos futuros também podem ser feitos para resolverem problemas já conhecidos para implementação da TCS em CLPs, como, por exemplo: causalidade, efeito avalanche, simultaneidade, sincronização inexata e problema de escolha citados no trabalho de (CRUZ, 2011), mas agora com o foco em achar soluções através da implementação pelos blocos de funções da IEC 61499;

c. Estudos futuros para implementação prática da metodologia do controle supervisório modular local aderente à norma IEC 61499 desenvolvida nesse trabalho em microcontroladores e outros dispositivos de menor capacidade de processamento e memória, dessa forma contribuindo com a ideia de distribuição do código de controle

em vários dispositivos e interoperabilidade definidos pela norma IEC 61499;

d. Definição de blocos de função padrões para implementação de modelos de subsistemas aplicados na indústria, como, por exemplo, para tornos, fresadoras, robôs, esteiras, CNCs, AGVs, etc., de forma a simplificar a implementação do código de controle por parte dos engenheiros de automação e controle, que não terão que dedicar tempo para modelar dispositivos que são padrões para diversas aplicações;

e. Desenvolvimento de uma ferramenta para mapeamento automático dos modelos dos subsistemas e dos supervisores modulares locais reduzidos nos blocos de função da IEC 61499 e também das interligações entre os mesmos conforme a metodologia de implementação do controle supervisório modular local aderente à norma IEC 61499 desenvolvida nesse trabalho. Dessa forma, através de entradas de dados referentes a TCS, o software gerará automaticamente os blocos de função e suas interligações, automatizando e agilizando a obtenção do código de controle para sistemas industriais;

f. Expansão dos experimentos: neste trabalho dois casos teóricos foram apresentados, logo, a aplicação em uma planta industrial real se torna um importante estudo de caso de forma a se obter resultados práticos para entender melhor e suprir as demandas da indústria moderna.

g. Apesar da metodologia apresentada nesse trabalho ser funcional, para sistemas de manufatura mais complexos é evidente que o número de blocos de função se torna demasiadamente grande, como visto no estudo de caso do capítulo 4, dificultando a implementação e o entendimento da lógica de controle da solução final. Um estudo futuro sugerido seria criar formas de otimizar a metodologia através do uso de blocos de função compostos, dessa forma minimizando assim o número de blocos de função na solução final, também contribuindo para que os sistemas sejam mais rápidos e necessitem de menos memória para serem implementados.

REFERÊNCIAS

ABRISHAMBAF, R. et al. An energy aware design flow of distributed industrial wireless sensor and actuator networks. **IEEE International Conference on Industrial Technology (ICIT)**, v. 1, n. 1, p. 2166–2171, 2015.

AFZALIAN, A.; NOORBAKHSH, M.; WONHAM, W. Supervisory control for under-load tap-changing transformers using discrete-event systems. **Control Engineering Practice**, v. 1, n. 1, p. 285–310, 2015.

AKESSON, K. et al. Supremica - an integrated environment for verification, synthesis and simulation of discrete event systems. **International Workshop on Discrete Event Systems**, v. 1, n. 8, p. 384–385, 2006.

ALVES, L. V. R.; MARTINS, L. R. R.; PENA, P. N. Ultrades - a library for modeling, analysis and control of discrete event systems. **International Federation of Automatic Control**, v. 1, n. 1, p. 5831–5836, 2017.

BASILE, F.; CHIACCHIO, P.; GERBASIO, D. On the implementation of industrial automation systems based on PLC. **IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING**, v. 10, n. 4, p. 990 – 1003, 2013.

BASSI, L. Industry 4.0: Hope, hype or revolution? **International Forum on Research and Technologies for Society and Industry (RTSI)**, v. 1, n. 3, p. 1–6, 2017.

CASSANDRAS, C. G.; LAFORTUNE, S. **Introduction to Discrete Event Systems**. USA: Kluwer Academic Publishers, 2007.

CENGIC, G. et al. Implementation of full synchronous composition using iec 61499 function blocks. **IEEE International Conference on Automation Science and Engineering**, v. 1, n. 1, p. 267–272, 2005.

CHRISTENSEN, J. H. et al. The IEC 61499 function block standard: Software tools and runtime platforms. **ISA Automation Week**, 2012.

CRUZ, D. L. L. **Metodologia para implementação de controle supervisório modular local em controladores lógicos programáveis**. Dissertação (Dissertação de Mestrado) — Centro de Ciências Tecnológicas, Universidade do Estado de Santa Catarina, 2011.

CURY, J. E. R. Teoria de controle supervisório de sistemas a eventos discretos. In: **V Simpósio Brasileiro de Automação Inteligente**. [S.l.: s.n.], 2001.

DAI, W.; VYATKIN, V. Ontology model for migration from IEC 61131-3 PLC to IEC 61499 function block. **IEEE International Symposium on Electronic Design, Test and Application**, v. 1, n. 6, p. 172–175, 2011.

DAI, W. W.; VYATKIN, V. A case study on migration from iec 61131 plc to iec 61499 function block control. **IEEE International Conference on Industrial Informatics**, v. 1, n. 7, p. 79–84, 2009.

DROZDOV, D.; DUBININ, V.; VYATKIN, V. Speculative computation in iec 61499 function blocks execution — modeling and simulation. **IEEE International Conference on Industrial Informatics (INDIN)**, v. 1, n. 14, p. 748–755, 2016.

FLORDAL, H. et al. Automatic model generation and plc-code implementation for interlocking policies in industrial robot cells. **Control Engineering Practice**, v. 15, n. 1, p. 1416–1426, 2007.

FOUNDATION, E. The platform for open innovation and collaboration. **https://www.eclipse.org/**, 2018.

GELEN, G.; UZAM, M. The synthesis and plc implementation of hybrid modular supervisors for real time control of an experimental manufacturing system. **Journal of Manufacturing Systems**, v. 33, n. 4, p. 535–550, 2014.

HAGGE, N. Integrating cnet and iec 61499 function blocks. **IEEE Conference on Emerging Technologies and Factory Automation (EFTA)**, v. 1, n. 1, p. 506–509, 2007.

HAGGE, N.; WAGNER, B. Modeling and clarifying the execution of iec 61499 function blocks using xnet. **IEEE International Conference on Industrial Informatics**, v. 1, n. 5, p. 1177–1182, 2007.

HARBS, E. **CNC-C2: Um controlador aderente as normas ISO 14649 E IEC 61499**. Dissertação (Dissertação de Mestrado) — Centro de Ciências Tecnológicas, Universidade do Estado de Santa Catarina, 2012.

HOLOBLOC, I. Function block development kit (FBDK). **https://www.holobloc.com/**, 2018.

IEC. **Inernational Standard IEC 61131-3: Programmable Logic Controllers - Part 3: Programming languages**. [S.l.], 2003.

IEC. **Inernational Standard IEC 61499: Function Blocks**. [S.l.], 2013.

LEAL, A. B.; CRUZ, D. L. L.; HOUNSELL, M. S. Plc-based implementation of local modular supervisory control for manufacturing systems. **IntechOpen**, v. 1, n. 1, p. 159–182, 2012.

LEWIS, R. **Modeling Control Systems Using IEC 61499: Applying function blocks to distributed systems**. London, England: Institute of Engineering and Technology, 2008.

LINDGREN, P. et al. A formal perspective on IEC 61499 execution control chart semantics. **IEEE Trustcom/BigDataSE/ISPA**, v. 1, n. 1, p. 293–300, 2015.

LOPES, Y. K. **Integração dos Níveis MES, SCADA e Controle da Planta de Manufatura com Base na Teoria de Linguagens e Autômatos**. Dissertação (Dissertação de Mestrado) — Centro de Ciências Tecnológicas, Universidade do Estado de Santa Catarina, 2012.

MONIRUZZAMAN, M.; GOHARI, P. Implementing supervisory control maps with plc. **American Control Conference**, v. 1, n. 1, p. 3594–3599, 2007.

MOREIRA, M. V.; BASSILIO, J. C. Bridging the gap between design and implementation of discrete-event controllers. **IEEE Transactions of Automation Science and Engineering**, v. 1, n. 11, p. 48–65, 2014.

NAYYAR, A.; PURI, V. A review of arduino board's, lilypad's e arduino shields. **International Conference on Computing for Sustainable Global Development (INDIACom)**, v. 1, n. 3, p. 1485–1492, 2016.

PANG, C.; PATIL, S.; YANG, C. A portability study of IEC 61499: Semantics and tools. **IEEE International Conference on Industrial Informatics (INDIN)**, n. 12, p. 440–445, 2014.

PATIL, S. et al. On composition of mechatronic components enabled by interoperability and portability provisions of IEC 61499: A case study. **Conference on Emerging Technologies e Factory Automation (ETFA)**, v. 1, n. 18, p. 1–4, 2013.

PENA, P. et al. Control of flexible manufacturing systems under model uncertainty using supervisory control theory and evolutionary computation schedule synthesis. **Information Sciences**, p. 491–502, 2016.

PETIN, J.; GOUYON, D.; MOREL, G. Supervisory synthesis for product-driven automation and its application to a flexible assembly cell. **Control Engineering Practice**, v. 1, n. 5, p. 595–614, 2007.

PINHEIRO, L. P. et al. Nadzoru: A software tool for supervisory control of discrete event systems. **IFAC International Workshop on Dependable Control and Discrete Systems**, v. 48, n. 7, p. 182–187, 2015.

PINOTTI, A. J. **Um método para projeto de sistemas embarcados baseado no controle supervisório modular local**. Dissertação (Dissertação de Mestrado) — Centro de Ciências Tecnológicas, Universidade do Estado de Santa Catarina, 2012.

PINTO, L. I. **ICARU-FB: Uma Infraestrutura de Software Aderente à Norma IEC 61499**. Dissertação (Dissertação de Mestrado) — Centro de Ciências Tecnológicas, Universidade do Estado de Santa Catarina, 2014.

PINTO, L. I.; LEAL, A. B.; ROSSO, R. S. U. Safe dynamic reconfiguration through supervisory control in IEC 61499 compliant systems. **IEEE International Conference on Industrial Informatics (INDIN)**, n. 15, p. 753 – 758, 2017.

PINTO, L. I. et al. ICARU-FB e FBE: Um ambiente de desenvolvimento aderente à norma IEC 61499. **Congresso Brasileiro de Software: Teoria e Prática (CBSoft)**, 2015.

PINTO, L. I. et al. ICARU-FB: An IEC 61499 compliant multiplatform software infrastructure. **IEEE Transactions on Industrial Informatics**, v. 12, n. 3, p. 1074–1083, 2016.

PRENZEL, L.; PROVOST, J. Plc implementation of symbolic, modular supervisory controllers. **IFAC-PapersOnLine**, v. 51, n. 1, p. 304–309, 2018.

QUEIROZ, M. H. **Controle Supervisório Modular de Sistemas de Grande Porte**. Dissertação (Dissertação de Mestrado) — Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina, 2000.

QUEIROZ, M. H. **Controle Supervisório Modular Local e Multitarefa de Sistemas Compostos**. Tese (Doutorado) — Centro Tecnológico, Universidade Federal de Santa Catarina, SC, Brasil, 2004.

RAMADGE, P. J.; WONHAM, W. M. The control of discrete event systems. **Proceedings of IEEE, Special Issue on Discrete Event Dynamic Systems**, v. 77, n. 1, p. 81 – 98, 1989.

RUDIE, K. The integrated discrete-event systems tool. **IEEE International Workshop**, n. 8, p. 394–395, 2006.

SHAW, G. D.; ROOP, P. S.; SALCIC, Z. Reengineering of iec 61131 into iec 61499 function blocks. **IEEE International Conference on Industrial Informatics**, v. 1, n. 8, p. 1148–1153, 2010.

STRASSER, T.; ROOKER, M.; EBENHOFER, G. Framework for distributed industrial automation (4DIAC). **IEEE International Conference on Industrial Informatics**, n. 6, p. 283–288, 2008.

SU, R.; WONHAM, W. M. Supervisor reduction for discrete-event systems. **Discrete Event Dynamic Systems: Theory and Applications**, v. 1, n. 1, p. 31–53, 2004.

TIEGELKAMP, M.; JOHN, K. **IEC 61131-3: Programming Industrial Automation Systems**. Berlim, Germany: Springer, 2010.

VIEIRA, A. D. **Método de implementação do controle de sistemas a eventos discretos com aplicação da teoria de controle supervisório**. Tese (Doutorado) — Centro Tecnológico, Universidade Federal de Santa Catarina, SC, Brasil, 2007.

VIEIRA, A. D. et al. A method for PLC implementation of supervisory control of discrete event systems. **IEEE Transactions on control systems technology**, v. 25, n. 1, p. 175–191, 2017.

VLAD, V. et al. An example of modeling manufacturing systems using petri nets and the IEC 61499 standard. **Proceedings of the 13th WSEAS International Conference on SYSTEMS**, p. 357–363, 2009.

VYATKIN, V. IEC 61499 function blocks for embedded and distributed control systems design. In: **O3neida and Instrumentation Society of America (ISA)**. [S.l.: s.n.], 2007.

VYATKIN, V.; CHOUINARD, J. On comparisons of the ISaGRAF implementation of IEC 61499 with FBDK and other implementations. **IEEE International Conference on Industrial Informatics**, n. 6, p. 289–294, 2008.

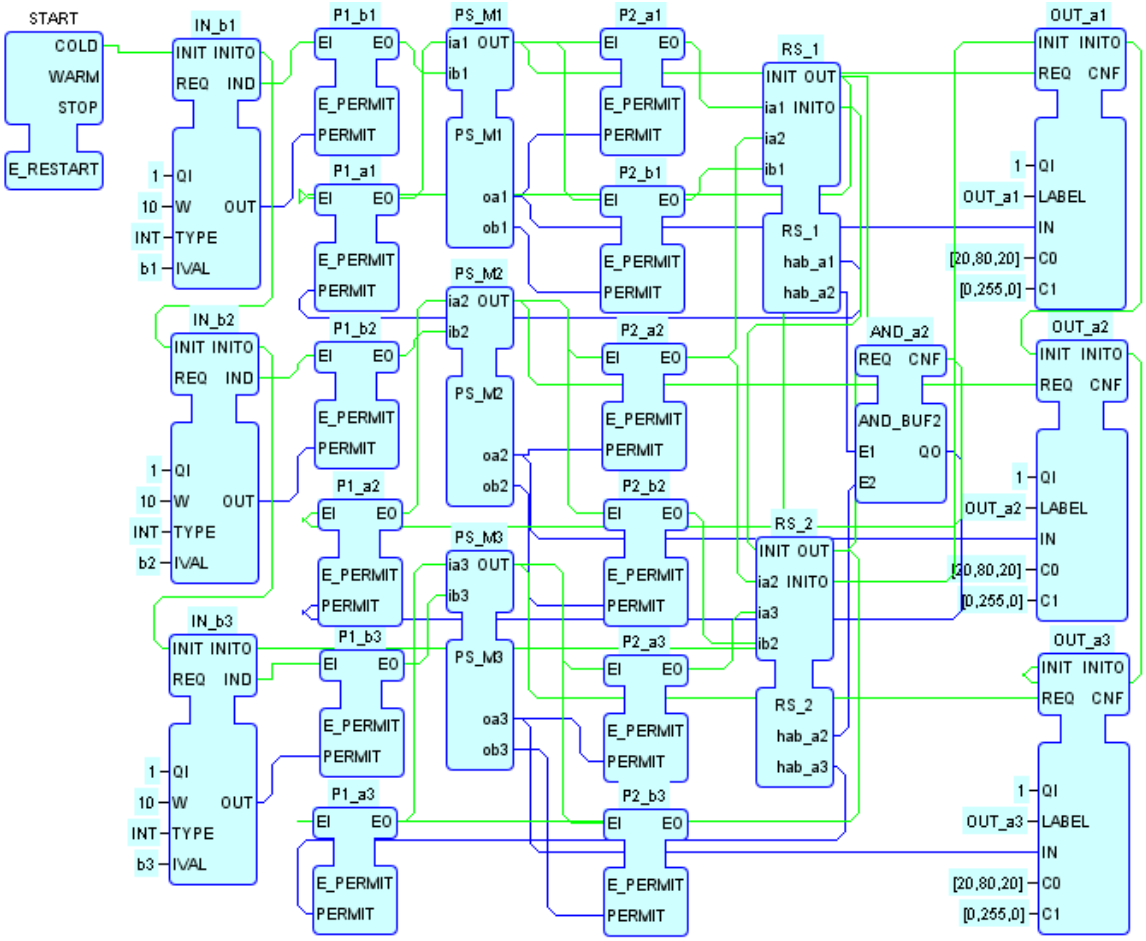
WENGER, M.; ZOITL, A. Re-use of IEC 61131-3 structured text for IEC 61499. **IEEE International Conference on Industrial Technology**, v. 1, n. 1, p. 78–83, 2012.

WONHAM, W. M. W. m. wonham's homepage. **https://www.control.utoronto.ca/DES/**, 2018.

ZAYTOON, J.; RIERA, B. Synthesis and implementation of logic controllers – a review. **Annual Reviews in Control**, v. 1, n. 1, p. 152–168, 2017.

APÊNDICE A – SIMULAÇÕES DO SISTEMA DE TRANSFERÊNCIA INDUSTRIAL

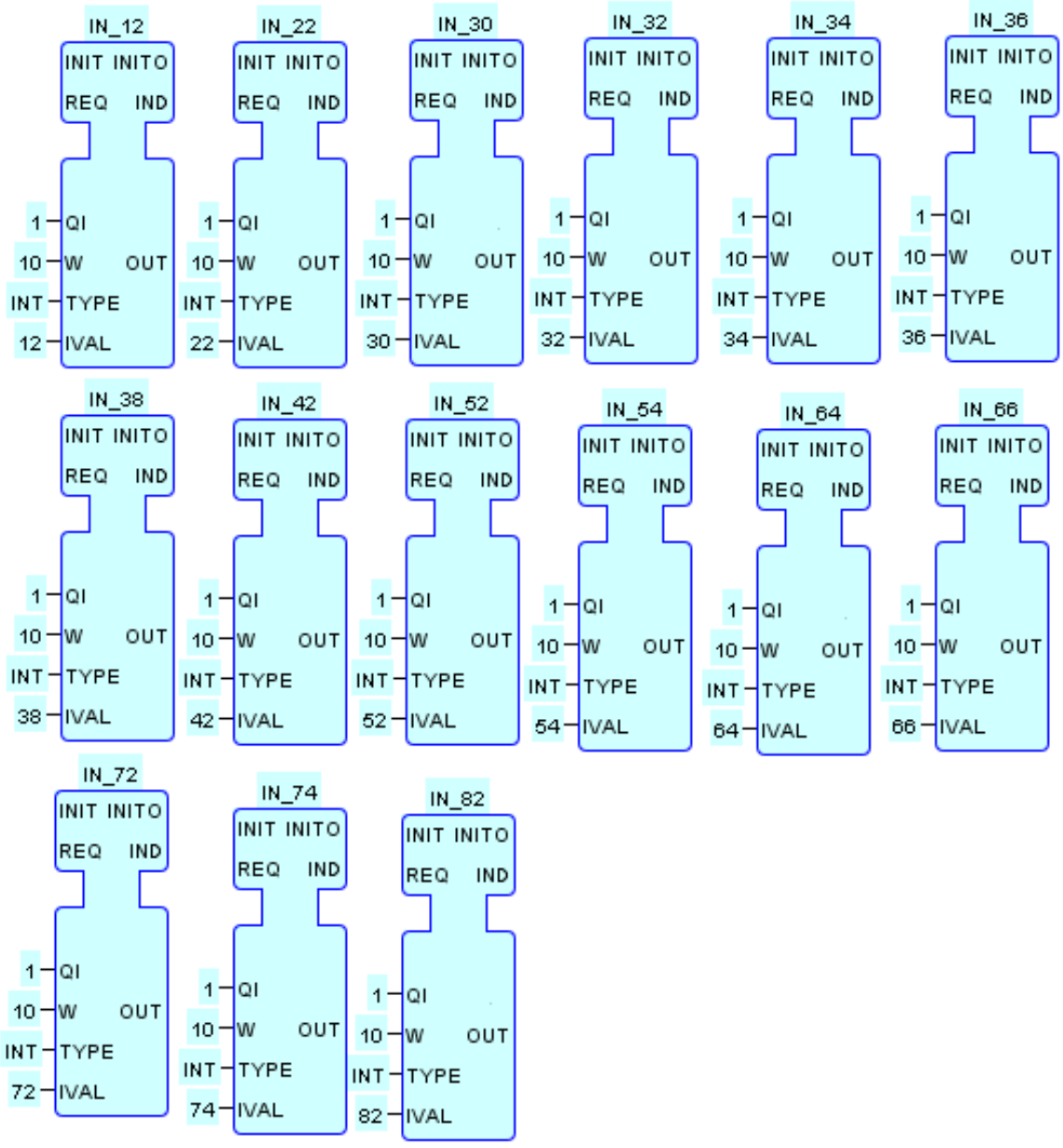
Figura 94 – Simulações do sistema de transferência industrial



fonte: do próprio autor

APÊNDICE B – SIMULAÇÕES DO SISTEMA FLEXÍVEL DE MANUFATURA

Figura 95 – Coluna 2 do modelo proposto

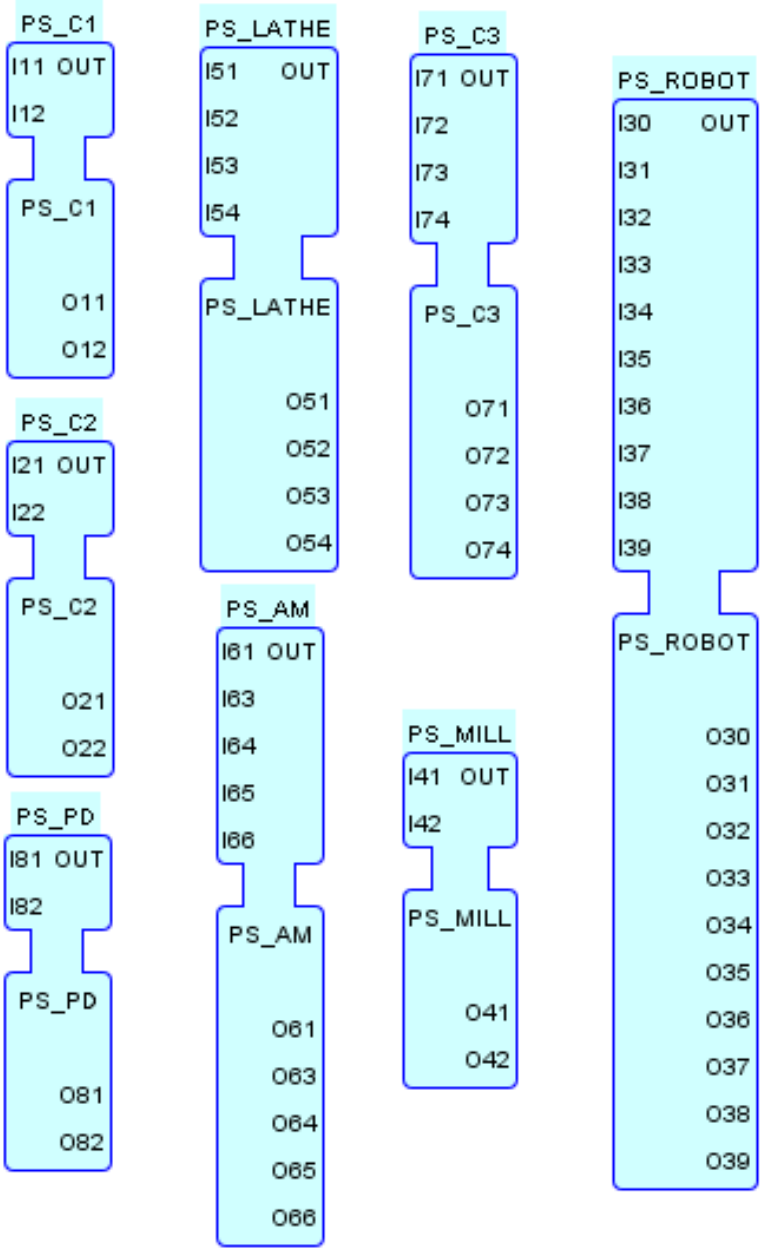


fonte: do próprio autor

Figura 96 – Coluna 3 do modelo proposto

fonte: do próprio autor

Figura 97 – Coluna 4 do modelo proposto

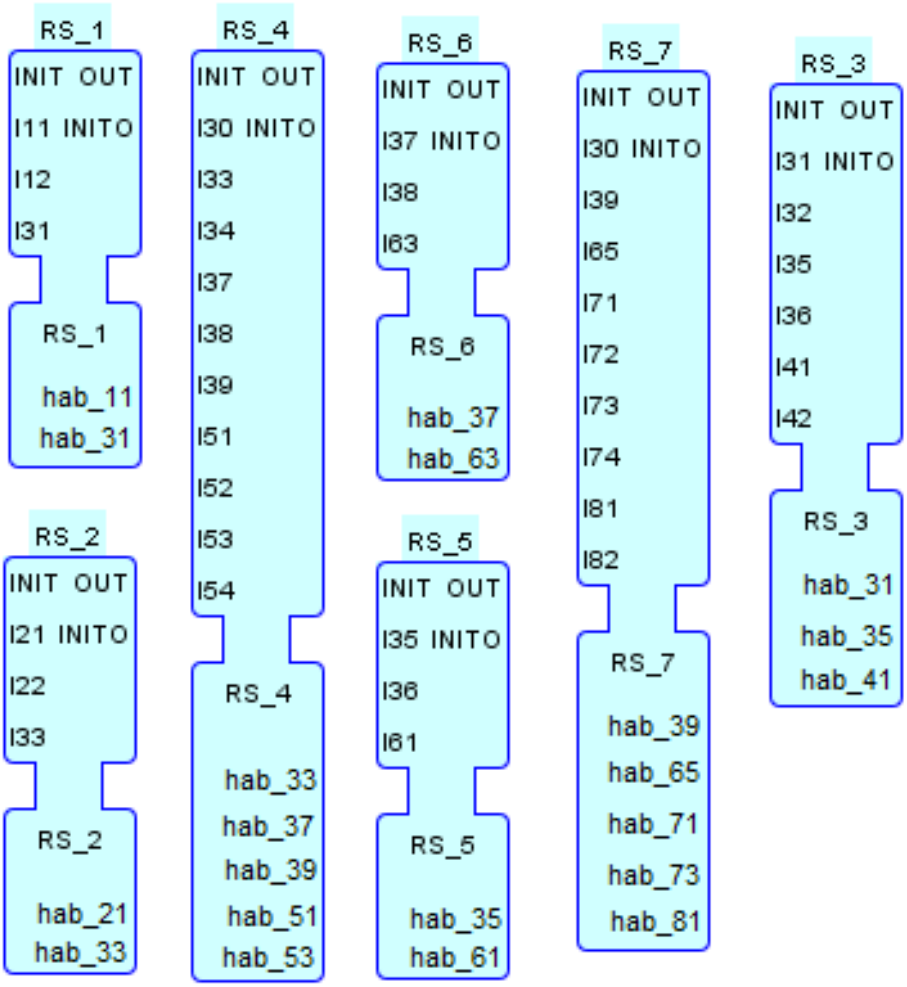


fonte: do próprio autor

Figura 98 – Coluna 5 do modelo proposto

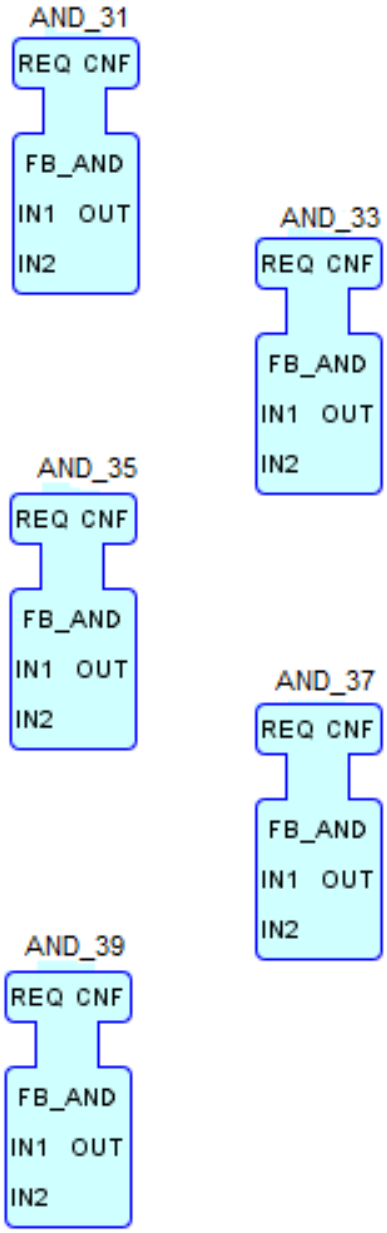
fonte: do próprio autor

Figura 99 – Coluna 6 do modelo proposto



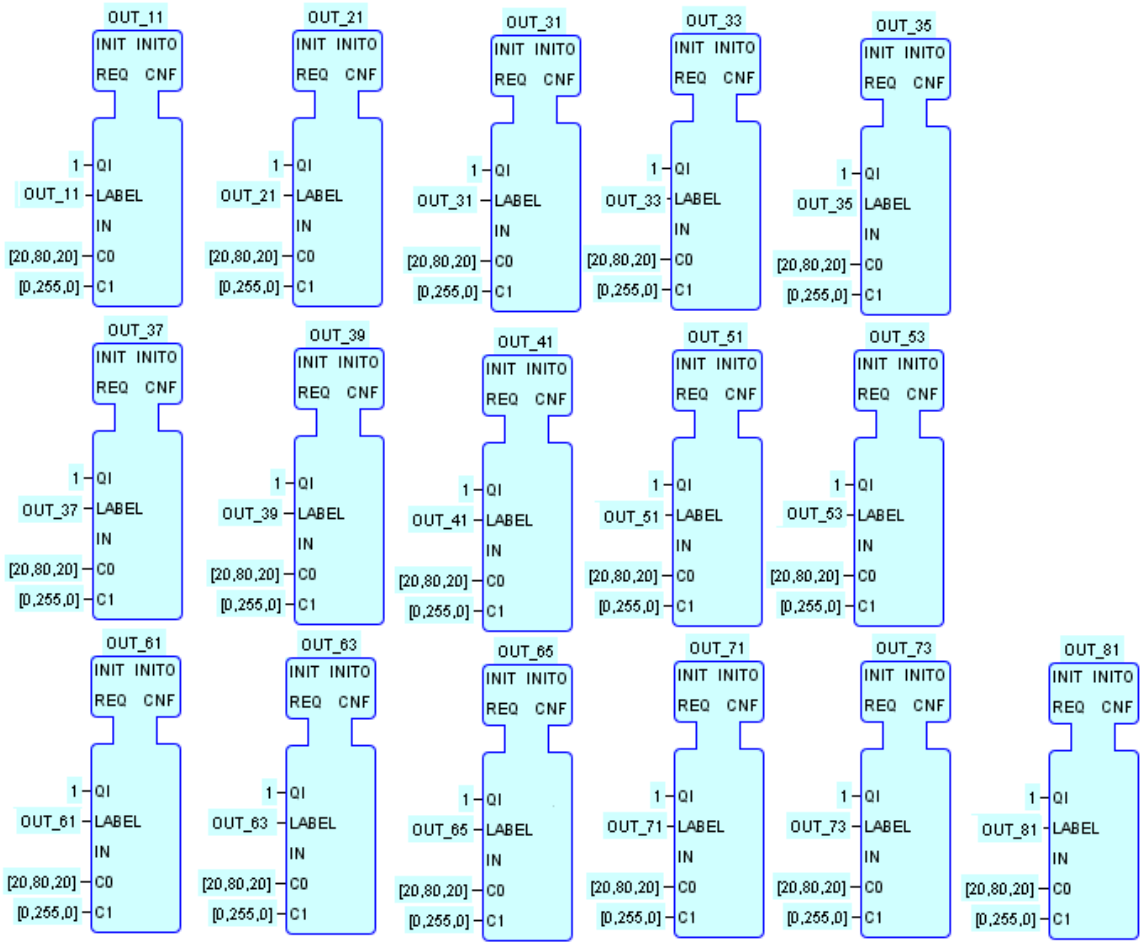
fonte: do próprio autor

Figura 100 – Coluna 7 do modelo proposto



fonte: do próprio autor

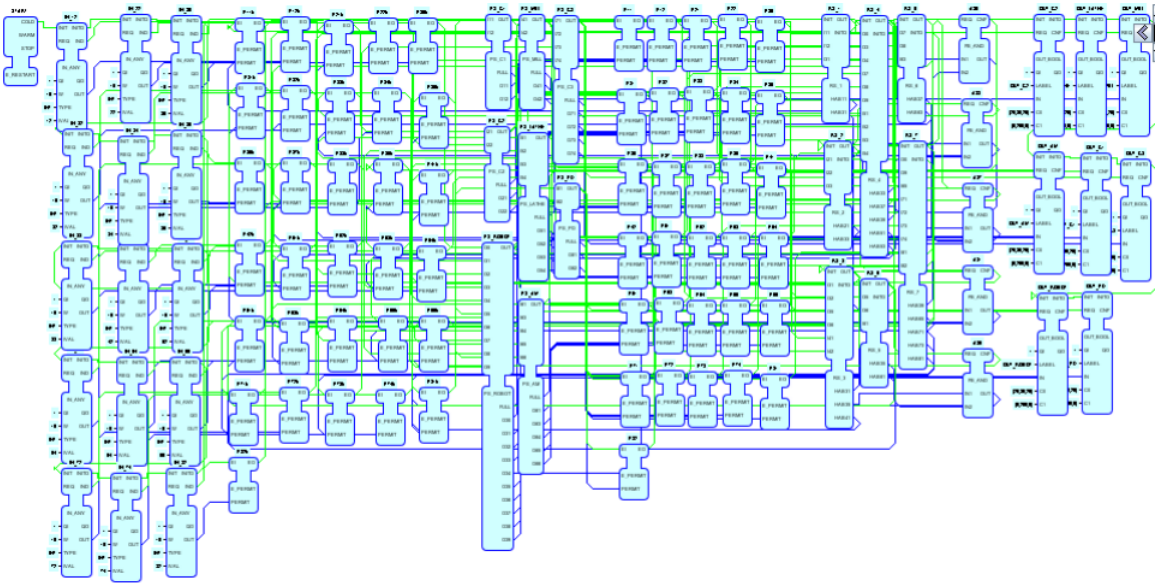
Figura 101 – Coluna 8 do modelo proposto



fonte: do próprio autor

APÊNDICE C – REPRESENTAÇÃO FINAL DA SIMULAÇÃO DO SISTEMA FLEXÍVEL DE MANUFATURA

Figura 102 – Representação final da simulação do sistema flexível de manufatura



fonte: do próprio autor