DISSERTAÇÃO DE MESTRADO Nº 746

ESTUDO E IMPLEMENTAÇÃO DE TÉCNICAS DE CONTROLE DE SISTEMAS A EVENTOS DISCRETOS EM CLP: APLICAÇÃO EM UM SISTEMA FLEXÍVEL DE MANUFATURA DIDÁTICO

Jonatham Silva Rezende

DATA DA DEFESA: 02/08/2012

Universidade Federal de Minas Gerais

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

Estudo e Implementação de Técnicas de Controle de Sistemas a Eventos Discretos em CLP: Aplicação em um Sistema Flexível de Manufatura Didático

Jonatham Silva Rezende

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Carlos Andrey Maia
Co-Orientadora: Profª. Drª. Patrícia Nascimento Pena

Belo Horizonte - MG
Agosto de 2012

Agradecimentos

Agradeço a Deus por sempre me proteger e iluminar.

Agradeço a minha família e irmãs, principalmente aos meus pais, Eustáquio e Rose, pelo amor, apoio e incentivo sempre dado.

Agradeço a minha esposa Paola pelo amor, carinho e paciência.

Agradeço aos professores Carlos Andrey Maia e Patrícia Nascimento Pena pela imensa paciência, disposição, compreensão e orientação.

Agradeço aos colegas Regiane Silva, Marcelino Mendes, Hugo Jerzy, o Grupo de Análise e Controle de Sistemas a Eventos Discretos (GACSED), aos colegas do LVAS, da Pós-Graduação da UFMG e ao professor Ivan pela troca de experiência e que de alguma forma contribuíram nesta longa etapa.

Agradeço a tia Climenê (*sempre presente*), a minha avó Maria de Lourdes e a meus parentes pelas conversas e incentivos.

Agradeço a UFMG, ao CNPQ e as empresas onde trabalhei pela oportunidade.

Meus sinceros agradecimentos.

Resumo

Os sistemas de produção buscam eficiência e redução dos custos para melhorar a competitividade no mercado. Desta forma, a automação exerce papel fundamental nos processos por meio da coordenação dos subsistemas, a fim de que as operações individuais e o funcionamento global do sistema sejam garantidos. Um conjunto de equipamentos que exerce diversas atividades e transforma matéria-prima em produto pode ser chamado de sistema de manufatura. Considera-se como um sistema flexível de manufatura (SFM), quando apresenta flexibilidade de produtos, rotas de produção e a capacidade de uma máquina em executar trabalhos diferentes. O SFM demanda rapidez no desenvolvimento e alteração na lógica de controle. Usualmente, a lógica de controle é implementada em um controlador lógico programável (CLP) baseada na experiência do programador e de forma empírica. Entretanto, existem métodos formais para implementar a lógica de controle em CLP como a Teoria de Controle Supervisório (TCS) baseada nos Autômatos e nas Redes de Petri via Invariantes de Lugar. A estrutura da TCS possui a planta que reflete o comportamento fisicamente possível do sistema e o supervisor que exerce uma ação de controle restritiva sobre a mesma para confinar seu comportamento àquele que corresponde a uma dada especificação. A Teoria de Linguagens e Autômatos é a base para a modelagem da planta e das especificações de controle para a síntese dos supervisores que são obtidos pelo Controle Modular Local. A Rede de Petri (RP) é a base para os Invariantes de Lugar que sintetizam um supervisor capaz de restringir as operações da planta modelada como uma RP de acordo com as restrições de segurança. Este trabalho propõe o estudo e avaliação de três metodologias de implementação em CLP da TCS baseada nos Autômatos e uma metodologia de implementação em CLP da TCS baseada nas Redes de Petri via Invariantes de Lugar com o objetivo de implementação prática e automação de um sistema flexível de manufatura didático construído no Laboratório de Análise e Controle de Sistemas a Eventos Discretos (LACSED) da UFMG. Uma análise comparativa entre as quatro metodologias é apresentada e não tem como objetivo indicar a melhor metodologia, mas estabelecer as vantagens e desvantagens das metodologias de implementação em CLP para promoverem o conhecimento e a disseminação da aplicação dessas metodologias.

Palavras-chave: controlador lógico programável, controle supervisório, autômatos, redes de petri, sistemas a eventos discretos.

Abstract

The production systems seek efficiency and reduction of costs to improve competitiveness in the market. In this way, automation exerts a fundamental role in the processes through coordination of the subsystems, in order that the individual operation and the overall operation of the system is guaranteed. A set of equipment that perform various activities and transform raw materials into products can be called a manufacturing system. A manufacturing system considered a flexible manufacturing system (FMS) when it presents product flexibility, production routes and capacity of a machine to perform different jobs. The FMS demands speed in development and change in control logic. Usually, the control logic is implemented in a programmable logic controller (PLC) based on the experience of the programmer and empirically. However, there are formal methods to implement control logic in a PLC as Supervisory Control Theory (SCT) based on Automata and Petri Nets via Place Invariants. The structure of SCT has a plant that reflects the behavior physically possible of the system and a supervisor that performs a restrictive control action on the plant in a way to confine its behavior to that corresponding a given specification. The Languages and Automata Theory is the basis for modeling the plant and control specifications for the synthesis of supervisors that are obtained by Local Modular Control. The Petri Net (PN) is the basis for the Place Invariants that synthesize a supervisor able to restrict the operations of the plant modeled as a PN according to security restrictions. This work proposes the study and evaluation of three methods of implementation in a PLC of SCT based on Automata and one implementation methodology in a PLC of SCT based on Petri Nets via Place Invariants with the objective of practical implementation and automation of a flexible manufacturing system didactic built in the Laboratório de Análise e Controle de Sistemas a Eventos Discretos (LACSED) at UFMG. A comparative analysis between the four methods is presented and it is not intended to indicate the best methodology, but establish the advantages and disadvantages of each method of implementation in the PLC to promote knowledge and dissemination of the application of these methodologies.

Keywords: *programmable logical controller, supervisory control, automata, petri nets, discrete-event systems.*

Sumário

Lista de Abreviaturas

CLP	Controlador Lógico Programável
CML	Controle Modular Local
IL	Invariantes de Lugar
LACSED	Laboratório de Análise e Controle de Sistemas a Eventos Discretos
RP	Redes de Petri
RSP	Representação por Sistema Produto
RW	Ramadge e Wonham
SED	Sistema a Eventos Discretos
SFM	Sistema Flexível de Manufatura
TCS	Teoria de Controle Supervisório
TCT	Toy Control Theory

Lista de Símbolos

Σ	Conjunto finito de eventos (alfabeto)
Σ_c	Conjunto de eventos controláveis
Σ_u	Conjunto de eventos não-controláveis
σ	Evento
Σ^*	Conjunto de todas as cadeias finitas de elementos de Σ
ϵ	Cadeia vazia
L	Linguagem
G	Autômato determinístico
δ	Função de transição de estados
Q	Conjunto de estados do autômato
q_0	Estado inicial do autômato
Q_m	Conjunto de estados marcados de Q
$\mathcal{L}(G)$	Linguagem gerada por G
$\mathcal{L}_m(G)$	Linguagem marcada por G
P_i	Projeção natural
P_i^{-1}	Projeção inversa
$\|$	Composição síncrona
Γ	Função de eventos ativos
γ	Entrada de controle
S	Supervisor
S_{red}	Supervisor reduzido
S/G	Sistema controlado
K	Linguagem desejada
$SupC$	Supervisor que implementa a máxima sublinguagem controlável
E	Especificação de segurança
MP	Máquina de Pintura
MM	Máquina de Montagem
C_1, C_2, C_3	Correias transportadoras 1, 2 e 3
$B_1, ..., B_8$	Depósitos Unitários

P	Conjunto finito de lugares
T	Conjunto finito de transições
A	Conjunto de arcos
w	Função peso associada aos arcos
$\mathtt{x}$	Estado atual da RP
$\mathtt{x}'$	Próximo estado da RP
$\mathtt{u}$	Vetor de disparo da transição da RP
$\mathtt{D}$	Matriz de incidência
$\mathtt{D}^+$	Matriz pesos dos arcos de entrada nos lugares
$\mathtt{D}^-$	Matriz pesos dos arcos de saída nos lugares

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Os processos industriais estão cada vez mais automatizados e os métodos de produção são aprimorados e otimizados.

Os sistemas flexíveis de manufatura são conjuntos de equipamentos integrados que exercem diversas atividades, transformam matéria-prima em produtos e apresentam flexibilidade de produtos, volumes de produção e rotas [Santos, 2007]. Estes sistemas devem produzir com qualidade e de acordo com as necessidades do cliente para uma solução adequada ao atendimento do mercado, são sistemas de produção altamente automatizados, capazes de produzir diversos produtos utilizando o mesmo equipamento e o mesmo sistema de controle.

Os sistemas de controle exercem um papel fundamental nos processos industriais por meio da coordenação e integração entre os subsistemas e/ou equipamentos por meio das lógicas de controle, a fim de que as operações individuais e o funcionamento global do sistema sejam garantidos.

Cassandras e Lafortune (1999) listam ferramentas formais como a Álgebra de Processos, Álgebra Max-Plus, Cadeias de Markov, Lógica Temporal, Teoria das Filas, Redes de Petri, Teoria de Linguagens e Autômatos que permitem o desenvolvimento de análise, síntese e implementação de sistemas de controle.

Conforme Queiroz (2004), a maior parte dos modelos limita-se à análise de soluções de controle propostas, que são geralmente desenvolvidas com base na experiência do programador e de forma empírica. As ferramentas formais podem auxiliar o processo de verificação da lógica de controle proposta ao permitir a verificação de propriedades dos modelos. Entretanto, a Teoria de Controle Supervisório (TCS) proposta por Ramadge e Wonham (1989), permite a síntese de supervisores ótimos a partir da planta e das especificações, o que garante maior confiabilidade nos sistemas de controle.

A TCS faz uma distinção clara entre o sistema a ser controlado denominado planta e a entidade a qual o controla, recebendo o nome de supervisor. A planta é um modelo que reflete o comportamento fisicamente possível do sistema. O papel do supervisor é o de exercer uma ação de controle restritiva sobre a planta para confinar seu comportamento àquele que corresponde a uma dada especificação. Uma vantagem desse modelo é a de

restringir o comportamento da planta apenas o necessário para evitar que realize ações proibidas [Cury, 2001].

A Teoria de Linguagens e Autômatos permite o desenvolvimento dos modelos formais dos sistemas de controle. Esta teoria modela a planta e as especificações de controle para a síntese dos supervisores que neste caso são obtidos pelo Controle Modular Local (CML).

Conforme Lima II (2002) e Yamalidou et al. (1995) é descrito um controlador de Rede de Petri para um SED modelado como Rede de Petri. O controlador é computado transformando o conjunto de restrições de segurança em invariantes de lugar do sistema controlado e, baseado neste conceito, supervisores para SED, como na Teoria de Controle Supervisório, podem ser gerados com o objetivo de restringirem as operações da planta de acordo com as restrições de segurança. As restrições de segurança são representadas na forma de desigualdades lineares.

Existem metodologias para a implementação em CLP dos resultados obtidos com a aplicação da TCS baseada nos Autômatos segundo Queiroz e Cury (2002), Vieira (2007) e Leal et al. (2009) e baseada nas Redes de Petri via Invariantes de Lugar segundo Lima II (2002).

Este trabalho propõe o estudo e avaliação de três metodologias de implementação em CLP da TCS baseada nos Autômatos [Queiroz e Cury, 2002], [Leal et al., 2009] e [Vieira, 2007] e uma metodologia de implementação em CLP da TCS baseada nas Redes de Petri via Invariantes de Lugar [Lima II, 2002] com o objetivo de implementação prática e automação de um sistema flexível de manufatura didático. Aspectos do desempenho do sistema e complexidade do programa do CLP são discutidos.

As metodologias de implementação em CLP discutidas utilizam Ladder Diagram (LD) e Sequencial Function Chart (SFC), ambas linguagens da norma IEC 61131-3 que favorecem a facilidade de entendimento e manutenção do sistema de controle.

Uma análise comparativa entre as quatro metodologias é apresentada e não tem como objetivo indicar a melhor metodologia, mas estabelecer as vantagens e desvantagens das metodologias de implementação em CLP.

É importante que hajam métodos de implementação em CLP para as diferentes soluções de controle, a TCS baseada em Autômatos e nas Redes de Petri via Invariantes de Lugar. Desta forma, o projetista tem a liberdade de escolher qual a metodologia a ser utilizada no sistema de controle. Uma vez escolhida a utilização da TCS baseada em Autômatos, há diversas opções e, neste trabalho, três delas são estudadas. Se o projetista for mais familiar com a linguagem de programação SFC que é uma linguagem de alto nível, a metodologia III é a indicada, pois tem a vantagem de ser modular, trata vários eventos por ciclo de atualização do CLP e implementa diversos modos de operação do sistema. Caso o projetista tenha preferência pela linguagem Ladder Diagram, a metodologia II tem a vantagem sobre a metodologia I, pois é modular, trata vários eventos por ciclo de atualização do CLP e possui a lógica para escolha de eventos. Uma vez escolhida a utilização

da TCS baseada nas Redes de Petri via Invariantes de Lugar, a metodologia IV utiliza a linguagem de programação Ladder Diagram e implementa a lógica para escolha de eventos.

1.1 Objetivo Geral

Aplicar e disseminar abordagem formal de modelagem, síntese e implementação de controle de Sistemas a Eventos Discretos empregando diferentes formalismos matemáticos e métodos de implementação.

1.2 Objetivos Específicos

- Revisar os principais conceitos do controle de Sistemas a Eventos Discretos permitindo a comparação entre os métodos formais;
- Automatizar a planta didática representativa de um Sistema Flexível de Manufatura;
- Modelar o comportamento dos subsistemas da planta utilizando o método formal das Redes de Petri;
- Estabelecer as restrições de segurança da planta modelada como uma Rede Petri;;
- Sintetizar os supervisores para controle da planta modelada como uma Rede de Petri;
- Implementar Arquitetura de Controle em Controlador Lógico Programável (CLP) conforme Queiroz e Cury (2002);
- Implementar Arquitetura de Controle em CLP conforme Leal et al. (2009);
- Implementar Arquitetura de Controle em CLP conforme Vieira (2007);
- Implementar Arquitetura de Controle em CLP conforme Lima II (2002);
- Analisar e comparar quatro implementações em CLP;
- Aplicar métodos formais para controle de Sistemas a Eventos Discretos;
- Disseminar o uso de metodologias de implementação em Controlador Lógico Programável (CLP) na automação dos sistemas de produção.

1.3 Organização do Texto

Esta dissertação está organizada da seguinte forma: no Capítulo 2, são apresentados os conceitos básicos dos Sistemas a Eventos Discretos e são definidos os formalismos para a modelagem destes sistemas baseados na Teoria de Linguagens e Autômatos e nas Redes de Petri.

No Capítulo 3, é exposta a teoria de controle para os Sistemas a Eventos Discretos e são definidos os formalismos para fazer o controle por intermédio da Teoria do Controle Supervisório baseada nos Autômatos e na Teoria do Controle Supervisório fundamentada nas Redes de Petri via Invariantes de Lugar.

O Capítulo 4 apresenta o controle de um sistema flexível de manufatura didático sob a TCS fundamentada nos Autômatos e nas Redes de Petri via Invariantes de Lugar.

O Capítulo 5 mostra os problemas na implementação do controle, a arquitetura de controle, as metodologias de implementação do controle no controlador lógico programável aplicados no sistema flexível de manufatura e uma análise comparativa das metodologias de implementação em CLP.

No Capítulo 6, são apresentadas algumas conclusões.

Capítulo 2

Modelagem de Sistemas a Eventos Discretos

Na seção inicial deste capítulo, são definidos os Sistemas a Eventos Discretos (SED). Nas próximas seções, são definidos os formalismos para a modelagem de SED, a Teoria de Linguagens e Autômatos na seção 2.2 e as Redes de Petri na seção 2.3.

2.1 Sistema a Eventos Discretos

Sistema a Eventos Discretos é um sistema dinâmico que evolui com a ocorrência de eventos em instantes indeterminados de tempo. Em Cassandras e Lafortune (1999) um SED é definido como um sistema cujo espaço de estados é descrito por um conjunto discreto e as transições de estados ocorrem por meio de eventos instantâneos no tempo. Por exemplo, um evento em um sistema flexível de manufatura pode ser a chegada de matéria-prima e a conclusão da fabricação de um produto.

O SED é representado por modelos formais que fornecem informações estruturais sobre o sistema.

Os principais modelos utilizados são:

- Álgebra Max-Plus;
- Cadeias de Markov;
- Redes de Petri;
- Teoria das Filas;
- Teoria de Linguagens e Autômatos.

As Redes de Petri e a Teoria de Linguagens e Autômatos são utilizadas para modelagem do sistema deste trabalho.

2.2 Linguagens e Autômatos

A Teoria de Linguagens e Autômatos é capaz de representar o SED e fornecer o arcabouço matemático para estudar o comportamento deste tipo de sistema.

2.2.1 Definições Básicas

Inicialmente o conceito de alfabeto é definido como sendo um conjunto de eventos representado por Σ.

Exemplos de alfabeto:

- $\Sigma_1 = \{\alpha, \beta, \gamma, ...\}$, conjunto das letras gregas minúsculas;
- $\Sigma_2 = \{1, 2, 3, 4, ...\}$, conjunto dos números reais.

A cadeia é uma sequência finita de eventos do alfabeto. A cadeia ϵ é a cadeia vazia que não possui qualquer evento. O comprimento da cadeia é definido como o número de eventos nela contidos. A representação do comprimento de uma cadeia é $|s|$. O comprimento da cadeia vazia ϵ é zero.

Exemplos de cadeia:

- A cadeia α do alfabeto Σ_1 possui comprimento 1 (um), $|\alpha| = 1$;
- A cadeia 3 5 4 1 do alfabeto Σ_2 possui comprimento 4 (quatro), $|3\ 5\ 4\ 1| = 4$.

O conjunto de todas as cadeias possíveis de comprimento k com eventos de um alfabeto é definido como potências de um alfabeto e sua representação é dada por Σ^k.

Exemplos de potências:

- Seja o alfabeto Σ_1. $\Sigma_1^0 = \epsilon$;
- Seja o alfabeto Σ_2. $\Sigma_2^2 = \{11, 12, 13, 14, 21, 22, ...\}$.

O conjunto de todas as cadeias possíveis de serem formadas com eventos de um alfabeto é representado por Σ^*. Σ^* inclui ϵ.

- $\Sigma_2^* = \{\epsilon, 1, 2, 3, 4, 11, 12, 13, 14, 21, 22, 23, 24, 31, 32, 33, 34, ...\}$.

O conjunto de todas as cadeias possíveis de serem formadas com eventos de um alfabeto, exceto ϵ, é representado por Σ^+.

A concatenação de cadeias é definida como uma cadeia seguida de outra cadeia.

Exemplos de concatenação:
Sejam as cadeias x = $\alpha\beta\chi$ e y = γ pertencentes ao alfabeto Σ_1.

- A concatenação xy é: xy = $\alpha\beta\chi\gamma$;
- A concatenação yx é: yx = $\gamma\alpha\beta\chi$.

As seguintes terminologias são aplicadas nas cadeias. Dada uma cadeia qualquer, $tuv = s$, com $t, u, v \in \Sigma^*$:

- t é prefixo de s;
- u é subcadeia de s;
- v é sufixo de s.

2.2.2 Linguagens

Uma linguagem definida sobre um alfabeto Σ é um subconjunto de cadeias de comprimento finito composto dos eventos em Σ. Esta linguagem não precisa necessariamente incluir cadeias com todos os eventos do alfabeto Σ.

Exemplos de linguagens:
Suponha $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Pode-se definir as linguagens:

- L_1 = {1, 3, 5, 7, 9 }, todas as possíveis cadeias de comprimento unitário com os números ímpares;
- L_2 = {todas as possíveis cadeias de tamanho 2 que contenham pelo menos a ocorrência do evento 1};
- L_3 = {todas as possíveis cadeias de tamanho 3 que terminam com o evento 8}.

São linguagens também $\emptyset$, Σ e Σ^*. Qualquer linguagem sobre Σ é um subconjunto de Σ^*.

2.2.3 Operações sobre Linguagens

As linguagens são conjuntos. Aplicam-se operações como a união, interseção, diferença e complemento. Outras operações também podem ser aplicadas como:

- Concatenação: Seja $L_1, L_2 \subseteq \Sigma^*$:

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

- Prefixo-fechamento: Seja $L \subseteq \Sigma^*$:

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

- Fechamento-Kleene: Seja $L \subseteq \Sigma^*$:

$$L^* := \{\epsilon\} \cup L \cup LL \cup LLL \cup \ldots .$$

Exemplos das outras operações:
Seja $\Sigma = \{\alpha, \beta, \gamma\}$ e considere as linguagens $L_1 = \{\epsilon, \alpha, \alpha\beta\gamma\}$ e $L_2 = \{\gamma\}$. Nem L_1 e nem L_2 são prefixo-fechadas, uma vez que $\alpha\beta \notin L_1$, $\epsilon \notin L_2$.

- Concatenação: $L_1 L_2 = \{\gamma, \alpha\gamma, \alpha\beta\gamma\gamma\}$;

- Prefixo-fechamento: $\overline{L_1} = \{\epsilon, \alpha, \alpha\beta, \alpha\beta\gamma\}$;

- Fechamento-Kleene:

  $L_1^* = \{\epsilon, \alpha, \alpha\beta\gamma, \alpha\alpha, \alpha\alpha\beta\gamma, \alpha\beta\gamma\alpha, \alpha\beta\gamma\alpha\beta\gamma, \alpha\alpha\alpha, \alpha\alpha\alpha\beta\gamma, \alpha\alpha\alpha\alpha, ...\}$.

2.2.4 Autômatos

Um autômato é um modelo matemático capaz de representar um par linguagens de acordo com regras bem definidas. A representação do autômato pode ser por meio de um grafo direcionado ou um diagrama de transição de estados.

Um autômato determinístico denotado por G é uma sextupla

$$G = (Q, \Sigma, \delta, \Gamma, q_0, Q_m)$$

na qual:

- Q é o conjunto de estados;

- Σ é o conjunto finito de eventos (alfabeto) associados a G;
- $\delta : Q \times \Sigma \to Q$ é a função de transição de estados, possivelmente parcial, ou seja, definida apenas para alguns elementos de $Q \times \Sigma$;
- $\Gamma : Q \to 2^{\Sigma}$ é a função de eventos factíveis; $\Gamma(x)$ é o conjunto de todos os eventos e tais que a função $\delta(x,\ e)$ é definida;
- q_0 é o estado inicial;
- $Q_m \subseteq Q$ é o conjunto de estados marcados.

A função de transição pode ser estendida para cadeia s de eventos como $\delta : Q \times \Sigma^* \to Q$, então $\delta(q, s\sigma) := \delta(\delta(q, s)\sigma)$ para algum $s \in \Sigma^*$, $\sigma \in \Sigma$ e $q \in Q$.

Na definição do autômato determinístico, o estado inicial é um estado único, todas as transições tem eventos rotulados $\sigma \in \Sigma$ e a função de transição é determinística no sentido que se o evento $\sigma \in \Gamma(x)$, então σ causa a transição de x para um único estado $y = \delta(x, \sigma)$.

O autômato possui o estado inicial q_0 e os estados marcados em Q_m. O estado inicial é reconhecido por uma seta que não possui origem em nenhum estado e aponta para ele. Os estados marcados são reconhecidos por círculos duplos, estes estados representam uma tarefa completa.

Como exemplo, no diagrama de transição de estados da Figura 2.1, os nós representam os estados e os arcos rotulados representam os eventos (transições entre os estados).

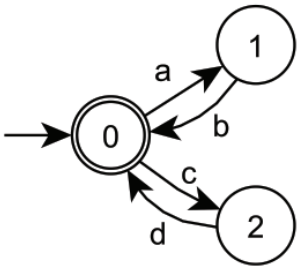

Figura 2.1: Autômato (três estados e quatro eventos)

O conjunto de estados do autômato é $Q = \{0, 1, 2\}$. O conjunto de eventos do autômato é $\Sigma = \{a, b, c, d\}$. O estado inicial é $q_0 = \{0\}$. O estado marcado é $Q_m = \{0\}$. A função de transição do autômato é representada graficamente por intermédio dos arcos no grafo e é expressa por $\delta : Q \times \Sigma \to Q$:
$\delta(0, a) = 1$;
$\delta(0, c) = 2$;
$\delta(1, b) = 0$;
$\delta(2, d) = 0$.

A notação $\delta(0, a) = 1$ expressa: se o estado ativo do autômato é o estado '0' então a ocorrência do evento 'a' resulta instantaneamente na ativação do estado '1' e na desativação do estado '0'.

2.2.5 Linguagens Representadas por Autômatos

Ao autômato G podem ser associadas duas linguagens: a linguagem gerada $\mathcal{L}(G)$ e a linguagem marcada $\mathcal{L}_m(G)$.

A linguagem gerada por G é:

$$\mathcal{L}(G) := \{s \in \Sigma^* : \delta(q_0, s) \text{ } \textit{é definido}\}.$$

A linguagem $\mathcal{L}(G)$ representa todas as cadeias formadas no autômato a partir do estado inicial. Uma cadeia s está na linguagem $\mathcal{L}(G)$ se, e somente se, corresponde a uma sequência factível no autômato G. A $\mathcal{L}(G)$ é prefixo-fechada por definição.

A linguagem marcada por G é:

$$\mathcal{L}_m(G) := \{s \in \mathcal{L}(G) : \delta(q_0, s) \in Q_m\}.$$

O conjunto de cadeias, que começa no estado inicial e termina em estados marcados, é representado pela linguagem marcada $\mathcal{L}_m(G)$, o qual é um subconjunto da linguagem $\mathcal{L}(G)$ do autômato. A linguagem marcada por G não precisa ser prefixo-fechada, uma vez que nem todos os estados são marcados.

Um SED pode ser modelado por um autômato G, onde $\mathcal{L}(G)$ é o comportamento gerado e a $\mathcal{L}_m(G)$ é o comportamento marcado ou conjunto de tarefas completas do sistema.

2.2.6 Operações sobre Autômatos

Um autômato G pode apresentar estados acessíveis e coacessíveis. Um estado q é acessível considerando $q \in Q$ e para $s \in \Sigma^*$ tal que $\delta(q_0, s) = q$. Um autômato é acessível se todo estado $q \in Q$ é acessível.

Um estado $q \in Q$ é coacessível, se $\exists s \in \Sigma^*$ tal que $\delta(q, s) \in Q_m$, ou seja, caso exista uma cadeia s que, partindo do estado q, leve a um estado marcado. Um autômato é coacessível, se todos os seus estados forem coacessíveis.

Um autômato é aparado (*trim*), desde que ele seja acessível e coacessível. A operação *Trim(G)* elimina todos os estados não acessíveis ou não coacessíveis, resultando em um autômato aparado.

Um autômato é não bloqueante, se e somente se:

$$\overline{\mathcal{L}_m(G)} = \mathcal{L}(G).$$

O sistema, no estado de bloqueio, não é capaz de atingir os estados marcados.

Um estado $q \in Q$ tal que $\Gamma(q) = \emptyset$ e $q \notin Q_m$ é um estado de bloqueio morto (deadlock).

Na Figura 2.2 o estado '1' não é acessível, mas é coacessível. Os estados '2', '4' e '5' não são coacessíveis, mas são acessíveis. Os estados '0' e '3' são acessíveis e coacessíveis. O estado '5' é um estado deadlock. O estado '2' é um estado de bloqueio vivo (livelock), ou seja, pode ocorrer uma cadeia de eventos, mas o sistema não é capaz de completar uma tarefa, não atingindo, portanto, o estado marcado.

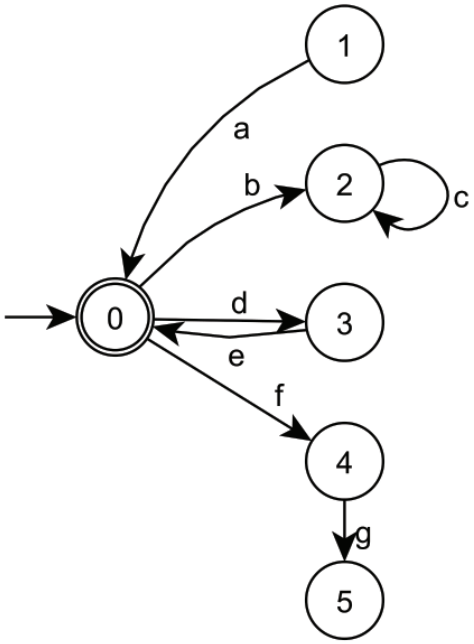

Figura 2.2: Autômato (estado não acessível e não coacessível)

Na Figura 2.3 é mostrado o autômato aparado.

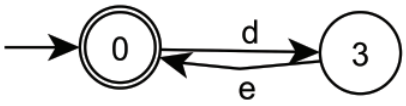

Figura 2.3: Autômato Aparado

Projeção Natural

Dados Σ e Σ_i, com $\Sigma_i \subseteq \Sigma$. A projeção natural, $P_i : \Sigma^* \to \Sigma_i^*$, de Σ^* para Σ_i^* é definida por:

$$P_i(\epsilon) = \epsilon;$$

$$P_i(\sigma) = \begin{cases} \epsilon \text{ se } \sigma \notin \Sigma_i; \\ \sigma \text{ se } \sigma \in \Sigma_i; \end{cases}$$

$$P_i(s\sigma) = P_i(s)P_i(\sigma) \text{ com } s \in \Sigma^*, \sigma \in \Sigma.$$

A projeção natural apaga os eventos de uma cadeia que não pertence a Σ_i. Aplicado a um autômato, a projeção natural gera um autômato possivelmente não determinístico o qual gera e marca as projeções das linguagens do autômato original. A diferença do autômato não determinístico em relação ao autômato determinístico é a função de transição que retorna um conjunto de estados. A extensão da operação de projeção natural para a linguagem é dada por:

$$P_i(L) = \{u_i \in \Sigma_i^* | u_i = P_i(u) \text{ para algum } u \in L\}.$$

E a projeção inversa é:

$$P_i^{-1}(L_i) = \{u \in \Sigma^* | P_i(u) \in L_i\}.$$

A projeção inversa gera uma linguagem que contém a original, mas não recupera a linguagem antes da projeção. Para o autômato, a projeção inversa é obtida adicionando-se auto-laços para todos os eventos de Σ que não estão em Σ_i do autômato G.

Composição Síncrona

A composição síncrona de $G_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$ e $G_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, Q_{m2})$, representada por $G_1||G_2$, é dada por:

$$G = Ac(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta, \Gamma_{1||2}, (q_{01}, q_{02}), Q_{m1} \times Q_{m2})$$

onde:

$$\delta((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) & \text{se } \delta_1(q_1, \sigma)! \text{ e } \delta_2(q_2, \sigma)! \\ (\delta_1(q_1, \sigma), q_2) & \text{se } \delta_1(q_1, \sigma)! \text{ e } \sigma \notin \Sigma_2 \\ (q_1, \delta_2(q_2, \sigma)) & \text{se } \delta_2(q_2, \sigma)! \text{ e } \sigma \notin \Sigma_1 \\ \text{indefinida} & \text{senão} \end{cases}.$$

A função de eventos ativos é

$$\begin{aligned} \Gamma_{1||2}((q_1, q_2)) &= [\Gamma_1(q_1) \cup (\Sigma_2 - \Sigma_1)] \cap [\Gamma_2(q_2) \cup (\Sigma_1 - \Sigma_2)] \\ &= [\Gamma_1(q_1) \cap \Gamma_2(q_2)] \cup [\Gamma_1(q_1) - \Sigma_2] \cup [\Gamma_2(q_2) - \Sigma_1]. \end{aligned}$$

A composição síncrona é o resultado da evolução dos eventos dos autômatos G_1 e G_2 em paralelo, ou seja, um evento local de cada autômato pode ser executado, quando estiver habilitado e o evento comum aos dois autômatos somente é executado simultaneamente nos dois autômatos.

A Figura 2.4 ilustra essa operação.

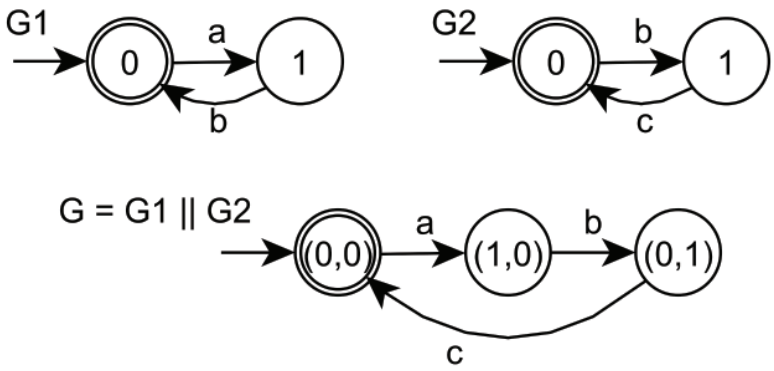

Figura 2.4: Composição Síncrona entre Autômatos

2.3 Redes de Petri

Um outro formalismo matemático para estudar o comportamento de um SED é baseado nas Redes de Petri (RP). Na RP, os eventos são associados às transições e o lugar possui informação da condição da transição. A RP é um grafo que possui transições e lugares conectados por um arco. O modelo da RP é análogo ao diagrama de transição de estados de um autômato.

2.3.1 Definições Básicas

Cassandras e Lafortune (1999) define uma Rede de Petri como uma quádrupla

$$(P, T, A, w)$$

na qual:

- $|P|$ = n é o conjunto finito de lugares;
- $|T|$ = m é o conjunto finito de transições;
- A é o conjunto de arcos, sub-conjunto do conjunto $(P \times T) \cup (T \times P)$;
- $w : A \rightarrow 1, 2, 3, \ldots$ é a função peso associada aos arcos.

É conveniente usar $I(t_j)$ para representar o conjunto dos lugares de entrada para a transição t_j e $O(t_j)$ para representar o conjunto dos lugares de saída da transição t_j, então:

$$I(t_j) = \{p_i \in P : (p_i, t_j) \in A\}, \quad O(t_j) = \{p_i \in P : (t_j, p_i) \in A\}.$$

Assume-se que (P, T, A, w) não possui lugares nem transições isoladas e a condição de conjuntos finitos pode ser relaxada, admitindo-se conjuntos contáveis.

O lugar é representado por círculo e pode ser interpretado como uma condição, um estado parcial ou um procedimento. O lugar tem um predicado associado. Como exemplos:

máquina livre, peça em espera, etc. A transição é representada por uma barra que é associada a um evento o qual ocorre no sistema. Como exemplos: o evento iniciar operação, término de operação, etc. O arco conecta lugar a transição ou transição ao lugar.

Exemplo de uma Rede de Petri:

A peça é um bloco bruto que é colocado na esteira C1 e possui um tempo de entrada o qual é a transição $T0$. O lugar $P0$ representa a peça na esteira C1. A peça é transportada na esteira C1 e possui um tempo de transporte que é a transição $T1$. A peça é então depositada no buffer B1 que é representado pelo lugar $P1$. A transição $T2$ representa o tempo para retirar a peça do buffer B1. O lugar $P2$ representa o depósito final e o lugar $P3$ limita o buffer B1 para a capacidade de uma peça.

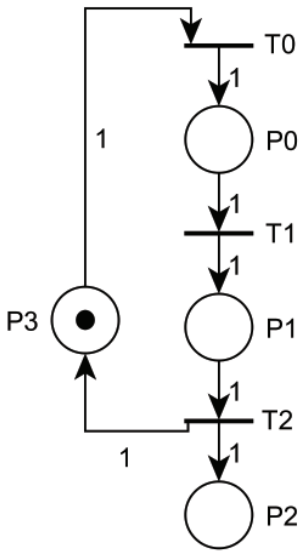

Figura 2.5: Rede de Petri (quatro lugares e três transições)

A Rede de Petri da Figura 2.5 possui:
Os lugares $P = \{P0, P1, P2, P3\}$;
As transições $T = \{T0, T1, T2\}$;
Os arcos $A = \{(T0, P0), (P0, T1), (T1, P1), (P1, T2), (T2, P2), (T2, P3), (P3, T0)\}$;
A função peso é
$w(T0, P0) = 1$;
$w(P0, T1) = 1$;
$w(T1, P1) = 1$;
$w(P1, T2) = 1$;
$w(T2, P2) = 1$;
$w(T2, P3) = 1$;
$w(P3, T0) = 1$.

Uma forma de indicar que uma condição é satisfeita em uma RP é associar ficha a um lugar. A ficha é um indicador significando que a condição associada ao lugar é verificada.

Por exemplo: uma ficha no lugar que representa máquina livre indica que a máquina está livre, ou seja, o predicado associado ao lugar é verdadeiro. Caso não exista ficha neste lugar, o predicado é falso e, por conseguinte, a máquina não está livre. O número de fichas em um lugar determina a marcação da RP.

Uma marcação x de uma RP é uma função

$$x : P \to \mathbb{N}$$

onde P é o conjunto de lugares e $\mathbb{N}$ é o conjunto dos números naturais. Uma marcação é representada por um vetor $\mathtt{x} = [x(p_1), ..., x(p_n)]^T$ e indica o estado da RP.

Uma Rede de Petri marcada é uma quíntupla

$$(P, T, A, w, x)$$

onde (P, T, A, w) é uma RP e x é a marcação do conjunto de lugares.

Na Figura 2.5, a marcação da RP é o vetor:

$$\mathtt{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

2.3.2 Dinâmica da Rede de Petri

A dinâmica da RP acontece pelo movimento de fichas e para entender esta evolução é necessário conhecer o conceito da transição habilitada.

Uma transição $t_j \in T$ em uma Rede de Petri marcada é dita habilitada se:

$$x(p_i) \geq w(p_i, t_j), \forall p_i \in I(t_j).$$

Assim, a transição t_j está habilitada, quando o número de fichas em p_i for maior ou igual ao peso do arco que conecta p_i a t_j. Para todos os lugares, p_i que são entradas da transição t_j, assim, uma transição está habilitada se todas as condições necessárias para a sua ocorrência são satisfeitas.

A função de transição de estados $f : \mathbb{N}^n \times T \to \mathbb{N}^n$ de uma Rede de Petri (P, T, A, w, x) é definida para uma transição $t_j \in T$ se, e somente se, esta transição está habilitada, ou seja, caso:

$$x(p_i) \geq w(p_i, t_j), \forall p_i \in I(t_j).$$

Esta condição garante que a função de transição só seja definida para transições habilitadas.

Se $f(\mathtt{x},t_j)$ é definida, diz-se que $\mathtt{x}' = f(\mathtt{x}, t_j)$, onde:

$$x'(p_i) = x(p_i) - w(p_i, t_j) + w(t_j, p_i)$$

para i = 1, . . . , n. Nesta condição, o próximo estado depende explicitamente dos lugares de entrada e saída da transição e dos pesos dos arcos que conectam estes lugares às transições.

Por exemplo, na Figura 2.5 a transição T0 da RP está habilitada e as transições T1 e T2 não estão habilitadas.

Algumas observações podem ser apresentadas sobre as Redes de Petri:

- Pode ocorrer que o número total de fichas distribuídos ao longo dos lugares não se conserve;
- A sequência de disparo das transições não é especificada na Rede de Petri;
- Nem todos os estados em $\mathbb{N}^n$ precisam ser alcançáveis.

A dinâmica de uma RP pode ser representada da seguinte forma:

- $\mathtt{x} = [x(p_1), x(p_2), ..., x(p_n)]^T$ o estado atual, com dimensão $n \times 1$;
- $\mathtt{x}' = [x'(p_1), x'(p_2), ..., x'(p_n)]^T$ o próximo estado, após o disparo da j-ésima transição, com dimensão $n \times 1$;
- $\mathtt{u} = [0, ..., 0, 1, 0, ..., 0]^T$ o vetor de disparo, com dimensão $m \times 1$, onde o 1 aparece na posição j e corresponde ao disparo da j-ésima transição, $j \in 1, ..., m$;
- $\mathtt{A} = [a_{ji}]$ matriz de incidência, de dimensão $m \times n$, tal que cada posição a_{ji} é definida como $a_{ji} = w(t_j, p_i) - w(p_i, t_j)$.

A dinâmica de uma Rede de Petri pode ser representada por:

$$\mathtt{x}' = \mathtt{x} + \mathtt{A}^T \times \mathtt{u}$$

onde + e × realizam operações matriciais de soma e multiplicação, respectivamente. A equação acima apresenta o processo de transição de estado gerado pela entrada u e que representa o disparo da j-ésima transição. O j-ésimo elemento do vetor u é diferente de zero.

A matriz de incidência transposta, $\mathtt{A}^T$, também pode ser definida como

$$\mathtt{A}^T = \mathtt{D} = \mathtt{D}^+ - \mathtt{D}^-$$

onde $-$ realiza operação matricial de subtração, $\mathtt{D}^+ \in \mathbb{Z}^{n \times m}$ com $\mathtt{D}^+ \geq 0$ e $\mathtt{D}^- \in \mathbb{Z}^{n \times m}$ com $\mathtt{D}^- \geq 0$ sendo n o número de lugares e m o número de transições da rede. A matriz $\mathtt{D}^+$ representa os pesos dos arcos de entrada nos lugares e a matriz $\mathtt{D}^-$ representa os pesos dos arcos de saída nos lugares.

Um vetor de disparo $\mathtt{u}$ para uma marcação $\mathtt{x}$ é válido se, e somente se,

$$\mathtt{x} - \mathtt{D}^- \times \mathtt{u} \geq 0.$$

Na Figura 2.5, a dinâmica da RP pode ocorrer da seguinte forma. A matriz de incidência é:

$$\mathtt{D} = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}.$$

A equação correspondente para o disparo da transição $T0$ na sequência de disparo $T0\,T1\,T2$ a partir do estado $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$ é:

$$\mathtt{x}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

A equação correspondente para o disparo da transição $T1$ na sequência de disparo $T0\,T1\,T2$ a partir do estado $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$ é:

$$\mathtt{x}_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

A equação correspondente para o disparo da transição $T2$ na sequência de disparo $T0\,T1\,T2$ a partir do estado $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$ é:

$$\mathtt{x}_3 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}.$$

2.3.3 Análise da Rede de Petri

Na análise da Rede de Petri, algumas definições são aplicadas para assegurar eficiência, segurança, uso adequado dos recursos e o não bloqueio do sistema. Elas buscam caracterizar propriedades desejáveis ou indesejáveis nos sistemas.

A *limitação* é a propriedade de um lugar manter o número de fichas, de modo a nunca exceder um número inteiro positivo.

Um lugar $p_i \in P$ em uma Rede de Petri com uma marcação inicial $\mathtt{x}_0$ é dito k-limitado ou k-seguro, se $x(p_i) \leq k$ para qualquer estado em qualquer trajetória possível. Um lugar 1-seguro é dito seguro. Um lugar k-limitado é dito limitado. Em uma RP, se todos os lugares são limitados, então a rede é dita limitada.

A *conservação* é a propriedade da rede sempre manter constante a soma ponderada do número de fichas distribuídas ao longo dos lugares. Uma RP com um dado estado inicial $\mathtt{x}_0$ é dita ser conservativa em relação a um vetor $\boldsymbol{\gamma} = [\gamma_1, \gamma_2, ..., \gamma_n]$, se

$$\sum_{i=1}^{n} \gamma_i \times x(p_i) = constante$$

para qualquer estado em qualquer trajetória possível. O vetor $\boldsymbol{\gamma}$ relaciona pesos aos lugares $P1, P2..., Pn$ definindo conservação relativamente a estes pesos.

A *vivacidade* está ligada às possíveis transições que podem disparar na RP. Uma RP com um dado estado inicial $\mathtt{x}_0$ é dita viva se, a partir de qualquer estado alcançado a partir de $\mathtt{x}_0$, existir pelo menos uma transição habilitada.

- Uma transição é dita L0-viva ou morta, se a transição nunca disparar a partir do estado inicial;
- Uma transição é dita L1-viva, caso exista alguma sequência de disparos, tal que a transição possa disparar pelo menos uma vez;
- Uma transição é dita L2-viva, desde que a transição dispare pelo menos k-vezes para algum número positivo k;
- Uma transição é dita L3-viva, se existe alguma sequência infinita de disparos na qual a transição aparece infinitas vezes;
- Uma transição é dita L4-viva ou viva, se a transição for L1-viva para qualquer estado alcançado a partir do estado inicial.

A *alcançabilidade* é uma propriedade relacionada aos estados que uma rede pode alcançar a partir de um estado inicial. Um estado $\mathtt{x}$ numa RP é dito alcançável a partir de um estado $\mathtt{x}_0$, se existir uma sequência de disparo de transições iniciando-se em $\mathtt{x}_0$ e tal que o estado possa se tornar $\mathtt{x}$.

A *cobertura* de um estado é uma generalização do conceito de alcançabilidade. Dada uma RP com estado inicial $\mathtt{x}_0$, um estado $\mathtt{y}$ pode ser coberto, caso exista uma sequência de transições iniciando-se em $\mathtt{x}_0$ e tal que o estado possa se tornar $\mathtt{x}$ e

$$x(p_i) \geq y(p_i), \forall i = 1, ..., n.$$

Se $\mathtt{y}$ é o estado que garante minimamente o disparo de uma transição t_j então, desde que $\mathtt{y}$ não possa ser coberto a partir do estado corrente, poder-se-á afirmar que t_j está morta.

A *persistência* é uma propriedade de RP que garante que uma transição não é desabilitada pelo disparo de outra transição. Uma RP é dita persistente, se para quaisquer duas transições habilitadas, o disparo de uma não desabilita o disparo da outra.

A *árvore de cobertura* é uma técnica de análise que permite verificar algumas características das Redes de Petri.

Capítulo 3

Controle dos Sistemas a Eventos Discretos

Este capítulo trata da Teoria de Controle Supervisório para os Sistemas a Eventos Discretos e nas próximas seções dois métodos para realizar o controle do SED são apresentados. A Teoria do Controle Supervisório fundamentada nos Autômatos na seção 3.2 e a Teoria do Controle Supervisório Baseada nas Redes de Petri via Invariantes de Lugar na seção 3.3.

3.1 Introdução

A Teoria de Controle Supervisório permite o funcionamento seguro dos sistemas e subsistemas dinâmicos. Para a TCS baseada em Autômatos, Ramadge e Wonham (1989) propuseram uma estrutura dividida em planta e supervisor. O supervisor pode ser sintetizado utilizando a abordagem do Controle Monolítico [Ramadge e Wonham, 1989] e como alternativas, as extensões Modular [Wonham e Ramadge, 1988] e Modular Local [de Queiroz e Cury, 2000]. Estas estratégias serão abordadas nas seções 3.2.1, 3.2.2 e 3.2.3, respectivamente.

Um supervisor de Rede de Petri para um SED modelado como RP é descrito em [Yamalidou et al., 1995], [Moody e Antsaklis, 1998] e [Jones, 1998]. Lima II (2002) apresenta o supervisor computado na TCS baseada nas Redes de Petri via Invariantes de Lugar. Esta estratégia será abordada na seção 3.3.

3.2 Controle Supervisório Baseado em Autômatos

Esta seção apresenta os conceitos do Controle Monolítico [Ramadge e Wonham, 1989], Modular [Wonham e Ramadge, 1988] e Modular Mocal [de Queiroz e Cury, 2000] e possui como referência os trabalhos dos referidos autores.

3.2.1 Controle Monolítico

A planta é o conjunto de subsistemas (equipamentos) do sistema a ser controlado, representa fisicamente as tarefas dos equipamentos e gera os eventos de forma espontânea

e assíncrona no tempo.

Na fase de modelagem da planta é feito o particionamento de Σ em

$$\Sigma = \Sigma_c \cup \Sigma_u$$

onde:

- Σ_c é o conjunto de eventos controláveis que podem ser inibidos de ocorrer no sistema;
- Σ_u é o conjunto de eventos não-controláveis que não podem ser inibidos de ocorrer no sistema.

Um evento controlável, por exemplo, pode ser ligar uma máquina ou o início de um processo; e um evento não controlável pode ser o desligar de uma máquina ou a sinalização do término de uma operação utilizando sensores.

As especificações exprimem as condições de segurança do sistema como, por exemplo, evitar o overflow ou underflow de um depósito, evitar o bloqueio de um equipamento ou sistema, e podem estabelecer interligações entre os subsistemas, garantindo prioridade de tarefas, segurança e vivacidade para o sistema.

O supervisor é o controlador do sistema, restringe o mínimo possível as tarefas dos equipamentos (ação de controle permissível) por meio de um conjunto de especificações e mantém a planta dentro do comportamento desejado.

No Controle Supervisório Monolítico, um único supervisor S é projetado de forma a habilitar ou desabilitar os eventos controláveis, tendo a referência do estado atual da planta G. Então o sistema em malha fechada obedece às especificações de segurança.

Na Figura 3.1 é mostrada a estrutura de controle em malha fechada da planta sob ação do supervisor.

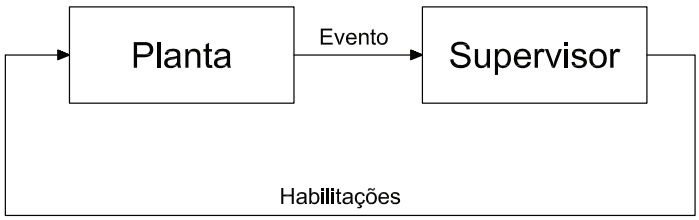

Figura 3.1: Estrutura de Controle em Malha Fechada da Planta

A estrutura de controle é o conjunto de todas as entradas de controle válidas associadas à planta. A entrada de controle é o conjunto de eventos habilitados pelo supervisor em um determinado instante, de acordo com a sequência de eventos observada e de forma que a planta permaneça no comportamento desejado.

O sistema controlado pode ser descrito pelo autômato resultante da composição síncrona $(S\|G)$, sendo S autômato representando um supervisor e G autômato representando

a planta. Para cada estado da planta, o supervisor executa uma ação de controle desabilitadora, de forma a desabilitar em G os eventos que não possam ocorrer em S após uma cadeia de eventos observada. O comportamento em malha fechada é

$$\mathcal{L}(S/G) = \mathcal{L}(S||G) \; e \; \mathcal{L}_m(S/G) = \mathcal{L}_m(S||G).$$

O supervisor S garante o não bloqueio do sistema em malha fechada, ou seja, $\overline{\mathcal{L}_m(S/G)} = \mathcal{L}(S/G)$.

Para a solução do controle supervisório, a controlabilidade da linguagem K é uma condição necessária e suficiente para a existência de um supervisor S que considera uma especificação ($K \subseteq \mathcal{L}(G)$) e ($\mathcal{L}_m(S/G) = K$).

Uma linguagem $K \subseteq \mathcal{L}(G)$ é uma sublinguagem controlável em relação a $\mathcal{L}(G) \subset \Sigma^*$, se:

$$\overline{K}\Sigma_u \cap \mathcal{L}(G) \subseteq \overline{K},$$

ou seja, K é controlável em relação a $\mathcal{L}(G)$, se e somente, se nenhuma cadeia de $\mathcal{L}(G)$ que esteja no prefixo de K, quando seguida de um evento não controlável em G, deixa de ser prefixo de K.

Caso o comportamento desejado, representado pela linguagem K, não seja controlável, é possível projetar um supervisor não-bloqueante que atenda às especificações de forma minimamente restritiva, ou seja, é possível projetar uma aproximação de K chamada de suprema linguagem controlável. Neste caso, o controle monolítico sintetiza um supervisor S a partir de uma linguagem especificada $K \subseteq \mathcal{L}(G)$, tal que $\mathcal{L}_m(S/G) = SupC(K, G)$.

O procedimento para a síntese monolítica consiste da:

1. Modelagem das plantas G_i, para $i \in I$, tal que I é o conjunto de índices que identificam os subsistemas;

2. Obtenção da planta global G pela composição de G_i, para $i \in I$;

3. Consecução das especificações genéricas E_j, para $j \in J$, tal que J é o conjunto de índices que identificam as especificações;

4. Obtenção da especificação global E pela composição de E_j, para $j \in J$;

5. Consecução da linguagem desejada K, $K = E||L(G)$ e obtenção da componente aparada;

6. Obtenção do supervisor que implementa a suprema sublinguagem controlável em relação a G contida em K, $S = SupC(K, G)$.

Na Figura 3.2 o resultado da composição das plantas G_1, G_2, G_3 e G_4 é a planta global G e o resultado da composição das especificações locais E_1, E_2 e E_3 é a especificação global E.

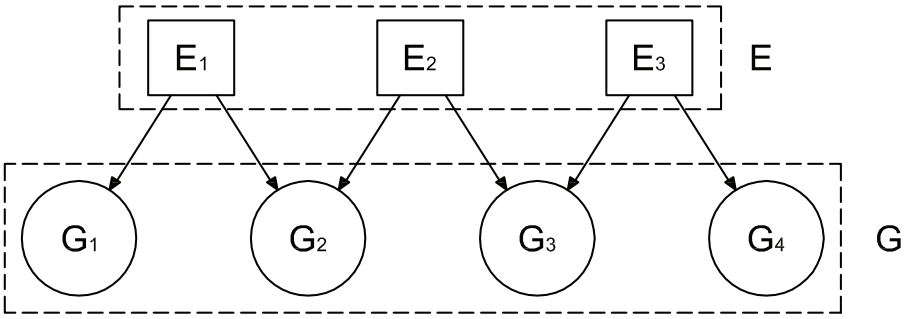

Figura 3.2: Composição Síncrona entre as Plantas Locais e Especificações Locais

3.2.2 Controle Modular

No Controle Modular, é possível sintetizar um supervisor para cada especificação, de forma que, atuando em conjunto, satisfaçam a especificação global do sistema. Na Figura 3.3, a ação de controle conjunta dos supervisores modulares desabilita um evento controlável da planta, sempre que este evento seja desabilitado por pelo menos um dos supervisores do sistema.

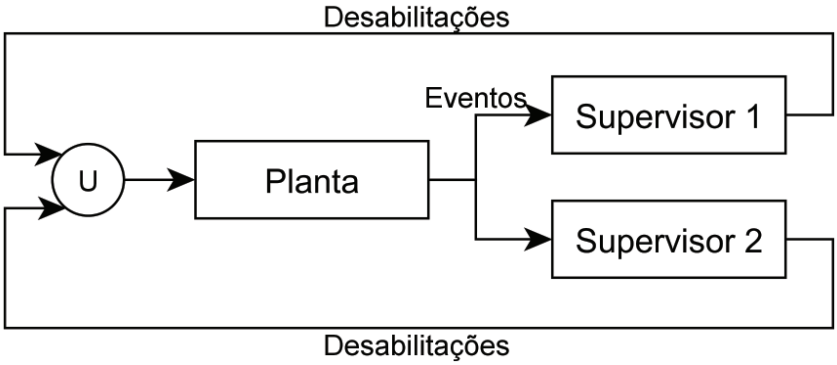

Figura 3.3: Estrutura de Controle Modular

O Controle Modular obtém módulos mais simples a partir da separação do sistema global, desta forma atua com uma visão parcial do estado de funcionamento do sistema global e pode gerar conflitos na ação de controle. Para que o Controle Modular seja equivalente ao Monolítico, deve ser verificada a modularidade das linguagens marcadas pelas ações dos supervisores.

Sejam as linguagens $L_i \subseteq \Sigma^*$, $i = 1, ..., n$. O conjunto de linguagens $(L_i = 1, ..., n)$ é não-conflitante (modular), se:

$$\cap_{i=1}^{n} \overline{L_i} = \overline{\cap_{i=1}^{n} L_i}.$$

Desta forma, sempre que um prefixo for aceito por todo o conjunto de linguagens, todo o conjunto deve aceitar uma palavra contendo esse prefixo. Neste caso, as linguagens não

geram conflito [Queiroz, 2002].

O procedimento para a síntese modular consiste de:

1. Modelagem das plantas G_i, para $i \in I$;
2. Obtenção da planta global G pela composição de G_i, para $i \in I$;
3. Consecução das especificações genéricas E_j, para $j \in J$;
4. Obtenção das linguagens desejadas K_j, $K_j = E_j||L(G)$ e obtenção das componentes aparadas para todo $j \in J$;
5. Consecução dos supervisores que implementam as máximas sublinguagens controláveis em relação a G contidas em K_j, $S_j = SupC(K_j, G)$, para todo $j \in J$;
6. Teste de não-conflito entre os supervisores obtidos, ou seja, $\cap_{i=1}^{n} \overline{S_i} \stackrel{?}{=} \overline{\cap_{i=1}^{n} S_i}$.

Na Figura 3.4 o resultado da composição das plantas locais G_1, G_2, G_3 e G_4 é a planta global G e as especificações genéricas E_1, E_2 e E_3 são usadas separadamente para síntese de um supervisor para cada especificação de tal forma que, quando combinadas, atenda-se à especificação global no sistema resultante em malha fechada, o que só vai ocorrer, se os supervisores forem não-conflitantes.

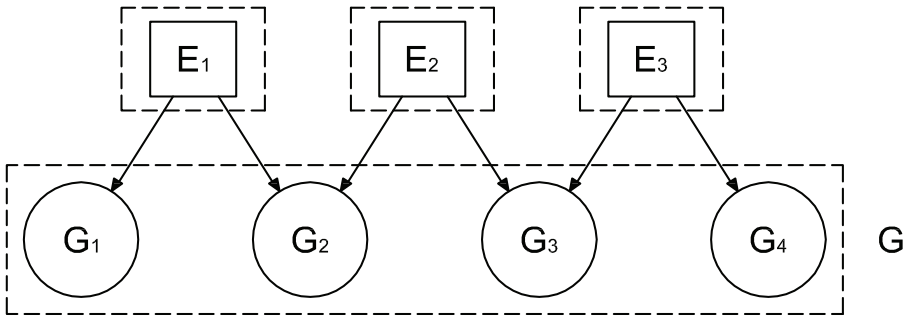

Figura 3.4: Composição Síncrona entre as Plantas Locais e Especificações Locais

3.2.3 Controle Modular Local

Considerando cada especificação individualmente a planta local correspondente, a planta local consiste na composição síncrona dos subsistemas que compartilham algum evento com tal especificação. A especificação local consiste no sincronismo de uma especificação com a sua planta local. O supervisor local coordena cada planta local que é uma parte do sistema global.

Os supervisores modulares controlam apenas os subsistemas associados a uma dada especificação, exploram a modularidade da planta e das especificações, e em conjunto

satisfazem a especificação global do sistema, desde que a propriedade de não-conflito seja verificada. Um supervisor modular desabilita apenas os eventos controláveis da planta local correspondente. A estrutura do Controle Modular Local (CML) pode ser vista na Figura 3.5.

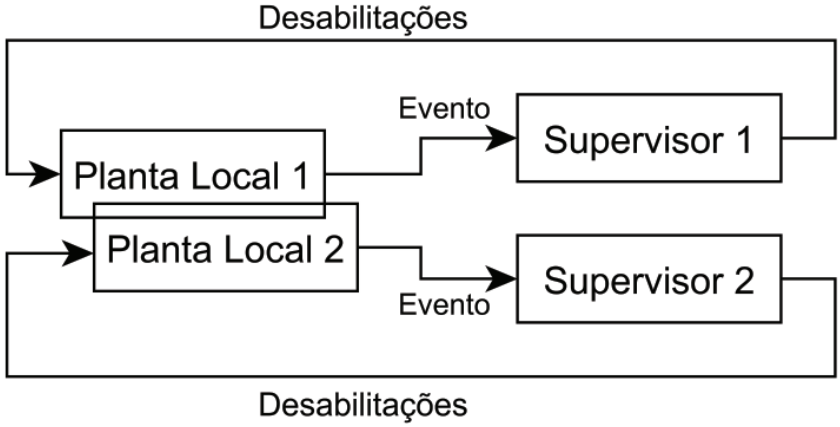

Figura 3.5: Estrutura do Controle Modular Local

Sejam as linguagens $S_i \subseteq \Sigma^*$, $i = 1, ..., n$ implementadas pelos supervisores. O conjunto de linguagens $(S_i, = 1, ..., n)$ é localmente modular, se:

$$\|_{i=1}^{n} \overline{S_i} = \overline{\|_{i=1}^{n} S_i}.$$

A suprema linguagem controlável de várias especificações é obtida a partir das especificações locais sem perda de desempenho em relação à solução monolítica e à solução modular, desde que a modularidade local seja verificada.

Quando a modularidade dos supervisores não é verificada, é possível utilizar técnicas para a resolução dos conflitos.

O procedimento para síntese do controle modular local consiste da:

1. Modelagem dos subsistemas G_i', para $i \in I$;

2. Cálculo da mais refinada representação por sistema produto (RSP), ou seja, obtenção da composição síncrona dos subsistemas que apresentem eventos comuns;

3. Obtenção das especificações genéricas E_j, para $j \in J$, considerando apenas os eventos relevantes;

4. Consecução da planta G_j para cada especificação, $j \in J$, compondo-se os subsistemas que possuem eventos em comum com a especificação em questão;

5. Obtenção das linguagens desejadas K_j, $K_j = E_j||\mathcal{L}_m(G_j)$ e obtenção das componentes aparadas, para todo $j \in J$;

6. Consecução dos supervisores que implementam as supremas sublinguagens controláveis em relação a G_j contidas em K_j, $S_j = SupC(K_j, G_j)$, para todo $j \in J$;

7. Verificação da propriedade de modularidade local (não-conflito) dos supervisores obtidos, $\|\overline{S_j} \stackrel{?}{=} \overline{\|S_j}$.

   (a) Se forem modulares, implementar um supervisor local para cada linguagem controlável;

   (b) Se não forem modulares, resolver o conflito.

Alguns trabalhos propõem métodos para resolver o conflito dos supervisores modulares, assim como [Queiroz, 2004] e [Pena et al., 2010].

A Figura 3.6 representa a síntese do Controle Modular Local onde são projetados supervisores para cada especificação em separado a partir da visão local da planta e de modo a atender a especificação global no sistema resultante em malha fechada.

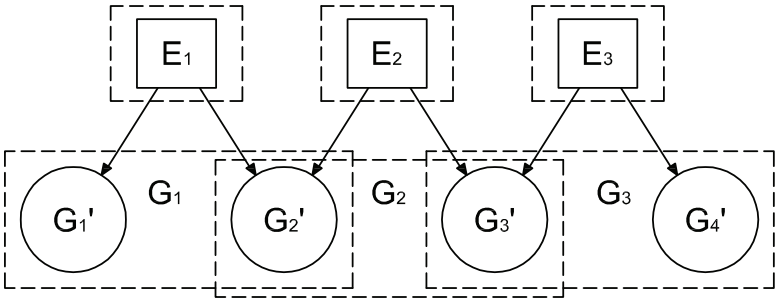

Figura 3.6: Composição Síncrona entre as Plantas Locais e Especificações Locais

3.3 Controle Supervisório Baseado nas Redes de Petri via Invariantes de Lugar

Esta seção apresenta os conceitos do Controle Supervisório fundamentado nas Redes de Petri via Invariantes de Lugar [Lima II, 2002], [Yamalidou et al., 1995], [Moody e Antsaklis, 1998] e [Jones, 1998] e possui como referência os trabalhos dos referidos autores.

3.3.1 Invariantes de Lugar

Os invariantes de lugar são conjuntos de lugares, cuja soma ponderada de marcas permanece constante para toda a marcação possível da rede a partir de $\mathtt{x}_0$ e são representados por vetores de n inteiros, onde n é o número de lugares da Rede de Petri e cujas entradas com os valores diferentes de zero indicam os lugares pertencentes ao invariante.

Um invariante de lugar é definido como todo o vetor inteiro $\boldsymbol{\gamma} \in \mathbb{Z}^n$, com dimensão $n \times 1$ tal que

$$\boldsymbol{\gamma}^T \times \mathtt{x} = \boldsymbol{\gamma}^T \times \mathtt{x}_0$$

sendo $\mathtt{x}_0$ um vetor coluna, $n \times 1$, que representa a marcação inicial da RP e $\mathtt{x}$ um vetor coluna, $n \times 1$, que representa a marcação.

Os invariantes de lugar de uma RP são encontrados a partir do resultado das soluções inteiras da equação

$$\boldsymbol{\gamma}^T \times \mathtt{D} = 0$$

onde $\mathtt{D}$ é a matriz de incidência $n \times m$ da Rede de Petri, sendo n o número de lugares e m o número de transições da rede.

3.3.2 Controle Supervisório Baseado nas Redes de Petri via Invariantes de Lugar

O sistema a ser modelado por uma Rede de Petri com n lugares e m transições é chamado de planta e a matriz de incidência da planta é chamada de $\mathtt{D}_p$.

É assumido que todas as transições habilitadas podem disparar, desta forma, é possível que a planta viole certas restrições impostas sobre o seu comportamento. Existe, portanto, uma necessidade de controle.

A matriz de incidência do controle é chamada de $\mathtt{D}_c$ e contém arcos que conectam os lugares de controle às transições da RP da planta. O número de lugares de controle n_c é igual ao número de restrições impostas na planta. Todo lugar usado para controlar a planta adiciona uma linha na matriz de incidência $\mathtt{D}$ do sistema controlado, formando um sistema em malha fechada. O sistema controlado é uma RP composta pela planta e os lugares de controle adicionados na planta.

A matriz de incidência $\hat{\mathtt{D}}$, com dimensão $\mathbb{Z}^{(n+n_c) \times m}$ é

$$\hat{\mathtt{D}} = \begin{bmatrix} \mathtt{D}_p \\ \mathtt{D}_c \end{bmatrix}.$$

Os arcos que conectam os lugares controladores para a RP da planta original do sistema serão computados na solução da equação dos invariantes de lugar.

O objetivo do controle supervisório é forçar a planta a obedecer as restrições da forma

$$\mathtt{L} \times \mathtt{x} \leq \mathtt{b}$$

sendo $\mathtt{x}$ um vetor $n \times 1$ que representa a marcação da rede sem controle, $\mathtt{L}$ uma matriz de ponderação de dimensão $n_c \times n$ e $\mathtt{b}$ um vetor de restrição dimensão $n_c \times 1$.

Caso seja necessário restringir a marcação da rede a mais de uma restrição, então, todas as restrições devem ser satisfeitas ao mesmo tempo e a cada restrição corresponde um lugar do controlador.

A desigualdade acima é transformada em equação, adicionando variáveis de folga

$$\mathtt{L} \times \mathtt{x} + \mathtt{x}_c = \mathtt{b}$$

sendo $\mathtt{x}_c$ um vetor $n_c \times 1$ que representa a marcação dos lugares do controlador. A equação acima define um invariante de lugar na Rede de Petri para o sistema controlado. Então, satisfazendo a equação acima tem-se:

$$\hat{\boldsymbol{\gamma}}^T \times \hat{\mathtt{D}} = \begin{bmatrix} \mathtt{L} & \mathtt{I} \end{bmatrix} \times \begin{bmatrix} \mathtt{D}_p \\ \mathtt{D}_c \end{bmatrix} = 0$$

sendo $\hat{\boldsymbol{\gamma}}$ uma matriz que representa os n_c invariantes e $\mathtt{I}$ uma matrix identidade $n_c \times n_c$.

A matriz de incidência do controlador é:

$$\mathtt{D}_c = -\mathtt{L} \times \mathtt{D}_p.$$

A marcação inicial do controlador $\mathtt{x}_{c0}$ é:

$$\mathtt{x}_{c0} = \mathtt{b} - \mathtt{L} \times \mathtt{x}_0$$

sendo $\mathtt{x}_0$ marcação inicial da planta, ou seja, da RP sem controle.

No cálculo do controlador, ou seja, supervisor, foi utilizada apenas a matriz $\mathtt{D}_p$ e é necessário que a rede não possua auto-laços. O supervisor calculado por esse método é maximamente permissivo [Moody e Antsaklis, 1998]. O supervisor é um "controlador de fichas" que asssegura que as restrições sejam respeitadas controlando a marcação da rede de Petri que modela a planta.

Lima II (2002) diz que a condição de habilitação do disparo das transições da Rede de Petri, $\mathtt{x} - \mathtt{D}^- \times \mathtt{u} \geq 0$, inibe uma transição, somente se seu disparo levar algum de seus lugares de entrada a uma marcação negativa ($\mathtt{x} \ngeq 0$). Então, o lugar do controlador apenas inibe uma transição, se seu disparo provocar $\mathtt{x}_c < 0$.

O controlador inibe uma transição somente em casos em que seu disparo cause uma violação da restrição. Nos sistemas reais, algumas transições podem não aceitar o controle, como exemplo, término de uma tarefa. Desta forma, são definidos dois subconjuntos das transições da rede, sendo:

- T_c o conjunto de transições controláveis que podem ser habilitadas ou desabilitadas por agentes externos;
- T_{uc} o conjunto de transições não controláveis que não podem ser desabilitadas por agentes externos.

Com a divisão, é gerada uma submatriz, $\mathtt{D}_{uc}$, composta pelas colunas de $\mathtt{D}$, correspondentes às transições não controláveis. Os lugares do controlador não podem inibir os disparos de transições não controláveis, assim, Moody e Antsaklis (1998) propuseram a desigualdade

$$\mathtt{L} \times \mathtt{D}_{uc} \leq 0.$$

Caso a desigualdade acima seja satisfeita, diz-se que a restrição é admissível e se não for satisfeita, diz-se que a restrição não é admissível e nesse caso é ainda possível controlar o sistema, determinando outra restrição, esta admissível, tal que para toda marcação alcançável da rede, a restrição original seja satisfeita.

Nos trabalhos de Moody e Antsaklis (1998) e Lima II (2002) são propostos algoritmos capazes de encontrar as restrições que serão transformadas em admissíveis e que tornem o controlador o mais permissivo possível.

Como exemplo, um sistema possui duas esteiras que não podem funcionar ao mesmo tempo. Desta forma, é necessário fazer um controle que tem o objetivo de impedir o funcionamento ao mesmo tempo das duas esteiras. A Figura 3.7 abaixo representa as duas esteiras modeladas por intermédio da Rede de Petri.

Figura 3.7: Rede de Petri Representando Duas Esteiras

A transição controlável $T1$ representa o comando ligar e a transição não controlável $T2$ retrata o comando desligar da esteira 1. O lugar $P1$ indica a esteira 1 funcionando e o lugar $P2$ aponta a esteira 2 funcionando. A transição controlável $T3$ representa o comando ligar e a transição não controlável $T4$ retrata o comando desligar da esteira 2.

A matriz de incidência da planta é:

$$\mathtt{D}_p = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}.$$

A matriz $\mathtt{D}_{uc}$ das transições não controláveis é:

$$\mathtt{D}_{uc} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}.$$

O objetivo de controle é evitar que $P1$ e $P2$ estejam com fichas ao mesmo tempo. Assim, as duas esteiras não operam concomitantemente. A restrição a ser imposta é:

$$\begin{bmatrix} 1 & 1 \end{bmatrix} \times \begin{bmatrix} x_{(P1)} \\ x_{(P2)} \end{bmatrix} \leq 1.$$

Desta forma, $\mathtt{L} = \begin{bmatrix} 1 & 1 \end{bmatrix}$ e $\mathtt{b} = [1]$.

Para que a restrição seja admissível a desigualdade $\mathtt{L} \times \mathtt{D}_{uc} \leq 0$ deve ser satisfeita:

$$\begin{bmatrix} 1 & 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} -1 & -1 \end{bmatrix}.$$

A restrição é admissível, pois o resultado $\mathtt{L} \times \mathtt{D}_{uc}$ é menor que zero.

A matriz de incidência do controlador é:

$$\mathtt{D}_c = -\ \mathtt{L} \times \mathtt{D}_p = -\begin{bmatrix} 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 1 & -1 & 1 \end{bmatrix}.$$

A marcação inicial para o controlador é:

$$\mathtt{x}_{c0} = \mathtt{b} - \mathtt{L} \times \mathtt{x}_0 = [1] - \begin{bmatrix} 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 1.$$

Na Figura 3.8, o sistema controlado é representado.

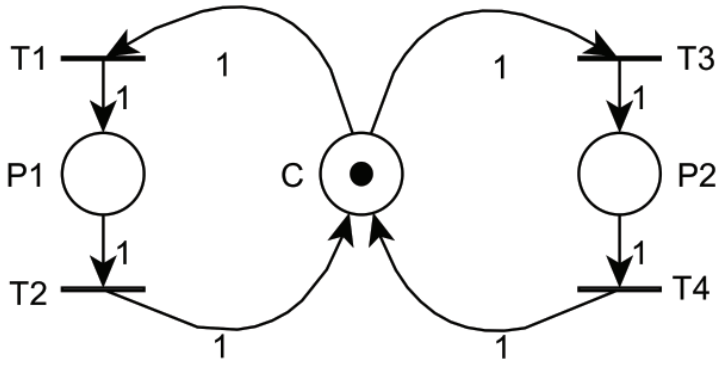

Figura 3.8: Rede de Petri Representando Duas Esteiras Sob Controle do Supervisor

Capítulo 4

Sistema Flexível de Manufatura Didático

Os sistemas de produção buscam eficiência e redução dos custos para melhorar a competitividade no mercado. Com este objetivo, a automação exerce um papel fundamental nos processos por intermédio da coordenação, integração e controle entre os subsistemas ou equipamentos de forma que as operações individuais e conjuntas sejam atendidas e que o funcionamento global do sistema seja garantido.

Um conjunto de equipamentos integrados que exercem diversas atividades e transformam matéria-prima em produtos pode ser chamado de sistema de manufatura. O sistema de manufatura quando apresenta flexibilidade de produtos, rotas, volume de produção e a capacidade de uma máquina em executar trabalhos diferentes pode ser considerado como um sistema flexível de manufatura (SFM), [Santos, 2007].

O SFM possui características como: a redução dos equipamentos e do espaço físico nos sistemas de produção, assim como o aumento na utilização dos equipamentos e melhor resposta às mudanças.

Este capítulo apresenta a descrição do problema na seção 4.1. A descrição do sistema físico utilizado no trabalho na seção 4.2 e para o controle do sistema flexível de manufatura didático abordado, a aplicação da TCS baseada nos Autômatos na seção 4.3 e a TCS baseada nas Redes de Petri via Invariantes de Lugar na seção 4.4.

4.1 Descrição do Problema

O Sistema Flexível de Manufatura tratado neste trabalho foi abordado na tese [Queiroz, 2004].

O SFM produz dois tipos de produtos: um produto composto por uma base, tendo um pino cônico no topo chamado de produto A; outro produto composto também da mesma base e um pino cilíndrico pintado no topo chamado de produto B. O SFM possui oito equipamentos: três esteiras (C_1, C_2 e C_3), um robô, uma fresa, um torno, uma máquina de pintura (MP), uma máquina de montagem (MM) e oito depósitos unitários ($B_1, ..., B_8$) que consistem na interligação entre os equipamentos.

Na Figura 4.1 é apresentado o sistema.

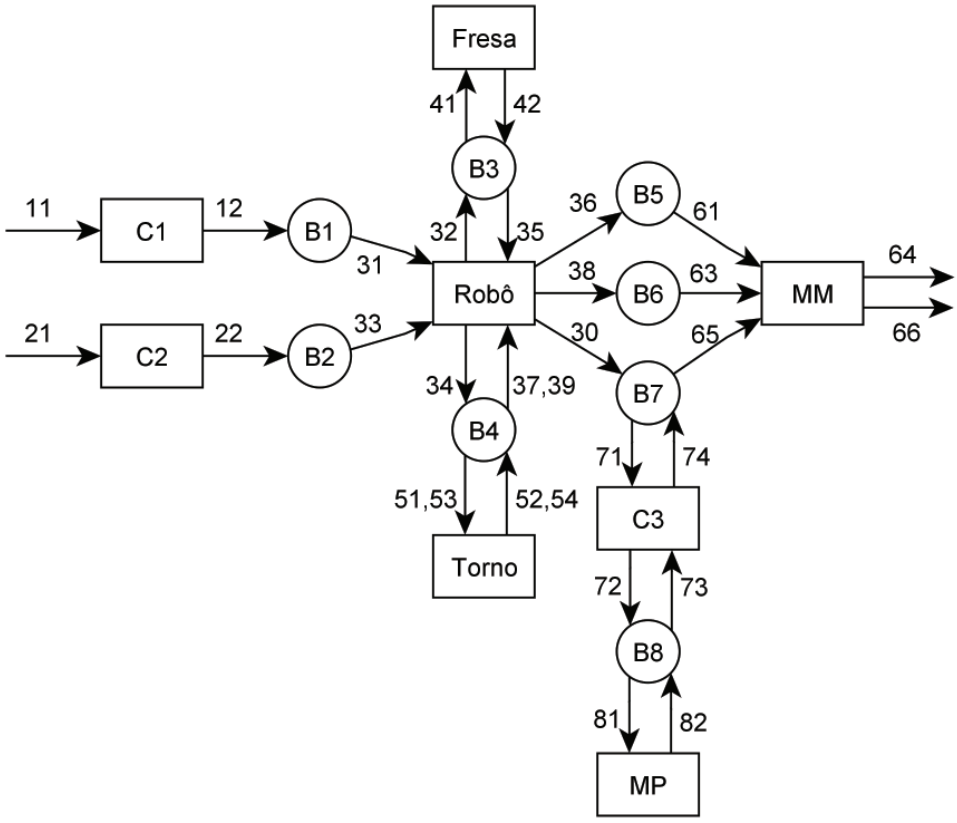

Figura 4.1: Sistema Flexível de Manufatura

As setas indicam os fluxos de peças e as etapas do processo de fabricação dos produtos. Blocos brutos são colocados na esteira C_1 pela execução do evento 11 e chegam ao depósito B_1 com a ocorrência do evento 12. Tarugos brutos são colocados na esteira C_2 por meio do evento 21 e chegam ao depósito B_2 com a ocorrência do evento 22. O robô pega e move um bloco bruto de B_1 para B_3 (31, 32) ou pega e move um tarugo bruto de B_2 para B_4 (33, 34). Do depósito B_3 o bloco bruto é colocado na fresa (41) e, após processado, retorna um bloco acabado (base acabada) a B_3 (42). Do depósito B_4, o tarugo bruto é colocado no torno e, após torneado, retorna ao B_4 um pino cônico (51, 52) ou um pino cilíndrico (53, 54). O robô então pega a base acabada de B_3 e leva a B_5 (35, 36) e se apropria de um pino cônico de B_4 e conduz a B_6 (37, 38) ou pega um pino cilíndrico de B_4 e transporta a B_7 (39, 30). O pino cilíndrico é pintado e passado de B_7 para a esteira C_3 (71), da esteira C_3 ao depósito B_8 (72) e de B_8 à máquina de pintura (81). O pino cilíndrico após ser pintado retorna ao depósito B_8 (82), é colocado na esteira C_3 (73) e retorna ao depósito B_7 (74). A etapa final é a montagem do produto e a base acabada deve chegar primeiro até a máquina de montagem MM, ou seja, passar do depósito B_5 para a MM (61). Em seguida, do B_6 sai um pino cônico que é colocado sobre a base acabada na máquina de montagem (63) e é fabricado o Produto A ou do B_7 sai um pino cilíndrico pintado que é colocado sobre a base acabada na máquina de montagem (65) e é fabricado o Produto B. O produto já montado é liberado pelo sistema por intermédio do

evento 64, seja o Produto A ou por meio do evento 66, seja o Produto B.

Para o controle geral do sistema flexível de manufatura, deve-se sintetizar uma lógica de controle que atenda às seguintes especificações.

- Não permitir sobreposição (overflow) ou falta (underflow) de peças nos depósitos;
- Permitir a operação simultânea da fresa e do torno no sistema;
- Garantir a manufatura de dois tipos de produtos, A e B.

4.2 Descrição do Sistema Físico

O sistema flexível de manufatura didático está localizado no Laboratório de Análise e Controle de Sistemas a Eventos Discretos (LACSED) da UFMG e foi construído utilizando kits FischerTechnik. Estes kits possuem motores, sensores fotoelétricos, chaves fim de curso, contadores de passo relacionados em [Neto, 2011] e outras peças que permitiram a construção do SFM didático.

O sistema físico do SFM didático simula a produção de dois tipos de produtos: um produto composto por uma base e um pino cônico no topo chamado de produto A, outro produto composto também da mesma base e um pino cilíndrico pintado no topo chamado de produto B. Na Figura 4.2 é apresentada a montagem completa do SFM didático e na Figura 4.3 é mostrada a vista superior dos equipamentos.

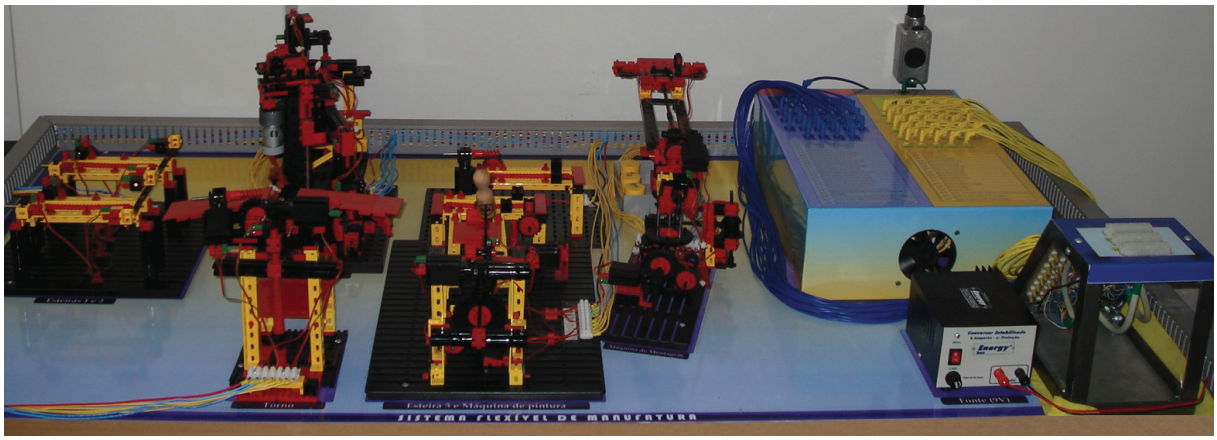

Figura 4.2: Montagem Completa SFM Didático

O SFM didático construído é interligado a um CLP modelo 1769-L32E Compact-Logix da marca Allen-Bradley Rockwell Automation e o software de programação utilizado para este CLP é o RSLogix5000. Uma interface entre o sistema físico do SFM didático e CLP foi construída e é apresentada na Figura 4.4. Esta interface permite a interligação nos pontos de entradas e saídas digitais do CLP.

Figura 4.3: Vista Superior dos Equipamentos

Figura 4.4: Interface do Sistema Físico

4.3 Controle Supervisório Baseado nos Autômatos

Nesta seção, é aplicado o conceito do Controle Supervisório Abordagem Modular Local discutido no Capítulo 3 para o controle do sistema flexível de manufatura.

Como visto na Seção 3.2.3, o CML possui as plantas locais que consistem na composição síncrona dos subsistemas, cujos eventos são comuns a uma dada especificação. A especificação local consiste no sincronismo de uma especificação genérica com a sua planta local e em conjunto satisfazem a especificação global do sistema. Os supervisores localmente modulares exploram a modularidade da planta e desabilitam apenas os eventos controláveis da planta local.

A solução utilizando CML foi apresentada em [Queiroz, 2004] e [Pena et al., 2010] e

os principais aspectos do projeto são discutidos a seguir.

a) Modelagem dos subsistemas G_i, $i = 1, ..., 8$;

A modelagem das plantas em malha aberta do SFM é composta por oito autômatos assíncronos, Figura 4.5, onde os números ímpares são os eventos controláveis e os números pares são os eventos não controláveis.

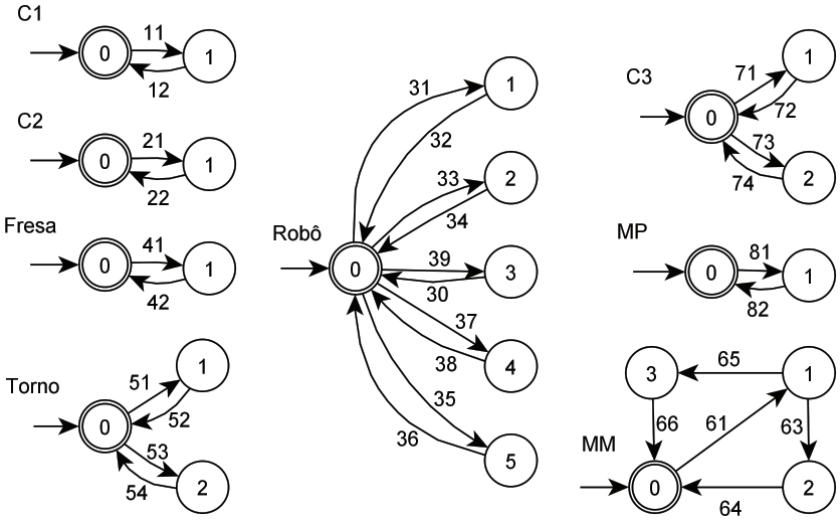

Figura 4.5: Autômatos Representando as Plantas

b) Esta já é a mais refinada representação por sistema produto (RSP).

c) Obtenção das especificações genéricas E_j, $j = 1, ..., 8$;

As especificações, Figura 4.6, são geradas de acordo com a Seção 4.1.

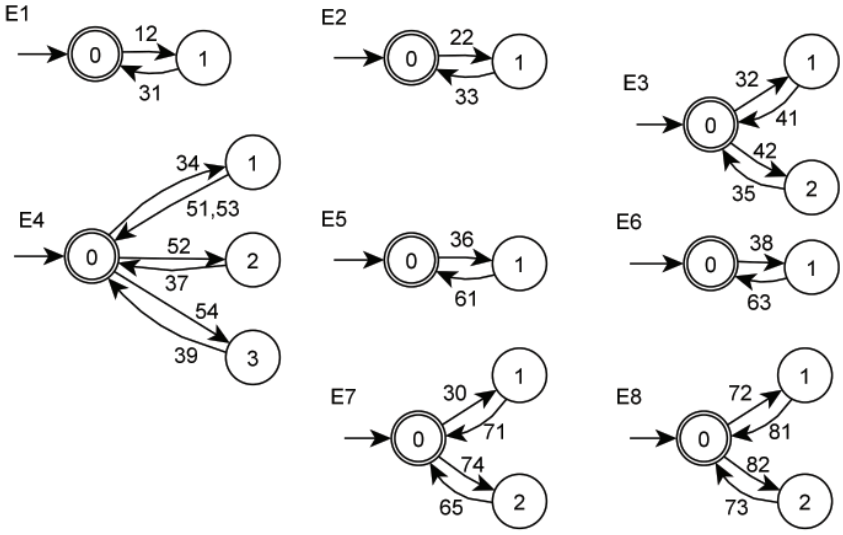

Figura 4.6: Autômatos Representando as Especificações

d) Consecução da planta local para cada especificação, compondo as plantas locais que possuem eventos em comum com a especificação em questão;

$G_1 = C_1$ || Robô;
$G_2 = C_2$ || Robô;
G_3 = Fresa || Robô;
G_4 = Robô || Torno;
G_5 = Robô || MM;
G_6 = Robô || MM = G_5;
G_7 = Robô || MM || $C3$;
G_8 = MP || C_3.

e) Obtenção das linguagens desejadas K_j, $K_j = E_j||\mathcal{L}_m(G_j)$, $\forall j = 1, ..., 8$ e obtenção das componentes aparadas.

f) Consecução dos supervisores que implementam as máximas sublinguagens controláveis em relação a G_j contidas em K_j, S_j = SupC(K_j, $L(G_j)$, $\forall j = 1, ..., 8$.

g) Teste de não-conflito entre os supervisores obtidos.

Os supervisores modulares são conflitantes e Pena et al. (2010) torna o sistema não-bloqueante através da resolução dos conflitos entre as plantas do Robô, da esteira C_3, da máquina de pintura (MP) e da máquina de montagem (MM). Para isto, é feita a composição síncrona entre as especificações locais E_7 e E_8 resultando em uma só especificação local E_{78} relacionada com uma planta local G_{78} agora composta pelos subsistemas afetados, G_7 e G_8. Assim, é obtida a linguagem desejada K_{78} por meio de $K_{78} = E_{78}||\mathcal{L}_m(G_{78})$ e o supervisor S_{78} por meio de S_{78} = SupC(K_{78}, $L(G_{78})$.

É aplicado o teste de não-conflito entre os supervisores modulares locais por $\overline{S_1}\|\overline{S_2}\|\overline{S_3}\|\overline{S_4}\|\overline{S_5}\|\overline{S_6}\|\overline{S_{78}} = \overline{S_1\|S_2\|S_3\|S_4\|S_5\|S_6\|S_{78}}$ e foi verificado que o conflito foi eliminado, pois não houve diferença entre o número de estados e transições de $\overline{S_1}\|...\|\overline{S_{78}}$ e de $\overline{S_1\|...\|S_{78}}$.

h) Redução de supervisores.

Neste item, é executada a redução dos supervisores encontrados por meio do TCT, [Feng e Wonham, 2006], pois as informações redundantes da estrutura da planta podem ser eliminadas dos supervisores. No TCT, a rotina utilizada para a redução dos supervisores é baseada no trabalho [Su e Wonham, 2004]. A versão do TCT para o Windows XP é o software XPTCT [XPTCT, 2009]. Este software também é capaz de construir os autômatos.

Na Figura 4.7, são apresentados os supervisores reduzidos e as desabilitações de eventos controláveis em cada estado dos supervisores.

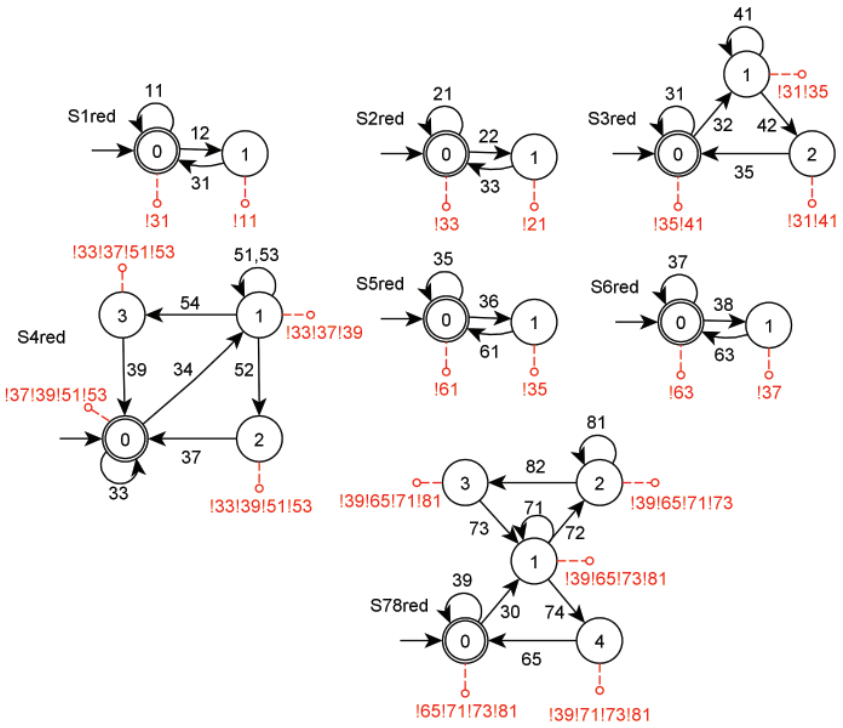

Figura 4.7: Autômatos Representando os Supervisores Reduzidos

Esta solução do Controle Modular Local é utilizada para implementação em um controlador lógico programável.

4.4 Controle Supervisório Baseado nas Redes de Petri via Invariantes de Lugar

Nesta seção são aplicados os conceitos do Controle Supervisório Baseado nas Redes de Petri via Invariantes de Lugar discutido no Capítulo 3 para o controle do sistema flexível de manufatura.

Como visto na seção 3.3.2, o Controle Supervisório via Invariantes de Lugar tem o objetivo de forçar a planta a obedecer às restrições de segurança. A modelagem do SFM utiliza a Rede de Petri (RP) que é apresentada na Figura 4.8. Esta RP modela o fluxo de fabricação dos produtos A e B. Para a modelagem da RP do SFM foi utilizado o software HPSim [HPSim, 2001].

Na Figura 4.8, os lugares são $P0$, ..., $P27$. As transições $T0$, $T2$, $T4$, $T6$, $T8$, $T10$, $T12$, $T14$, $T16$, $T17$, $T19$, $T21$, $T23$, $T25$ e $T27$ são controláveis e as transições $T1$, $T3$, $T5$, $T7$, $T9$, $T11$, $T13$, $T15$, $T18$, $T20$, $T22$, $T24$ e $T26$ são não controláveis. As transições relacionam-se com os eventos que ocorrem no SFM por meio dos números entre parênteses. Por exemplo, a transição $T2$ representa a ação do robô em pegar um bloco bruto de B_1 que corresponde ao evento 31 do SFM (Figura 4.1). Na Tabela 4.1 é relacionado o significado de cada lugar e na Tabela 4.2, o significado de cada transição.

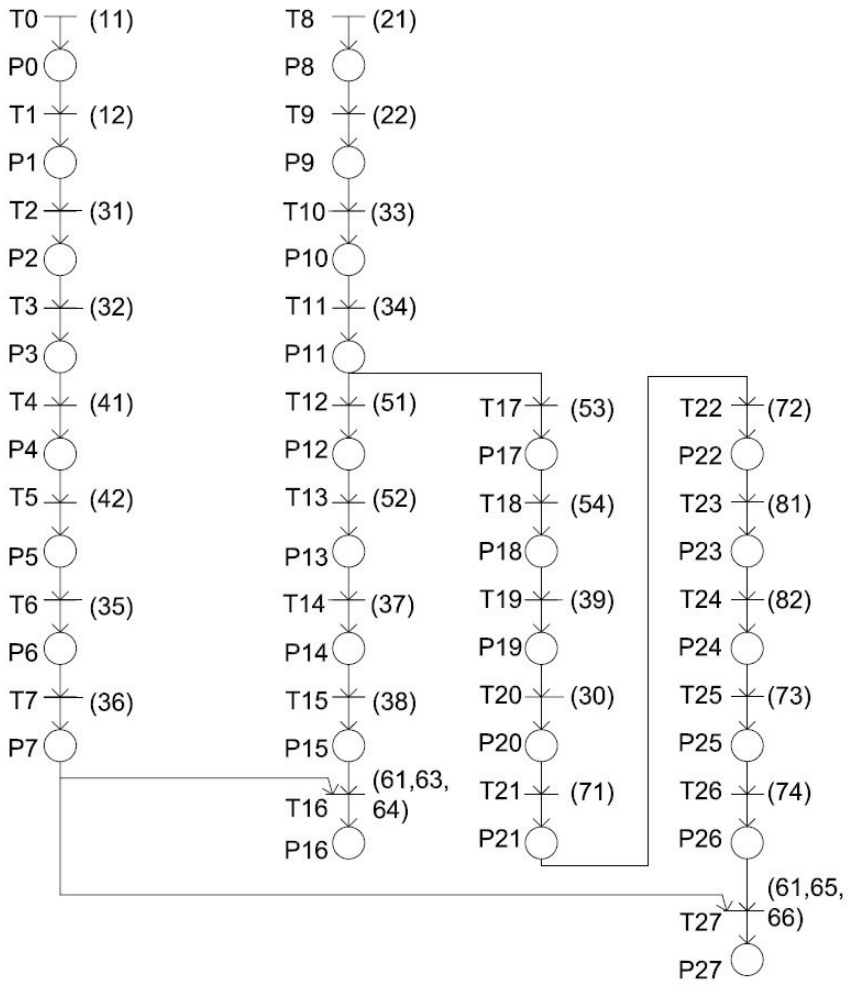

Figura 4.8: Rede de Petri do Sistema Flexível de Manufatura - Sem Controle

A marcação inicial da planta é zero ($\mathtt{x}_0 = 0$) e a rede não possui auto-laços.
As seguintes propriedades são verificadas na Rede de Petri do SFM:

- A rede é segura, pois para qualquer estado em qualquer trajetória possível $x(p_i) \leq 1$ $\forall i = 0, 1, 2, ..., 27$, desta forma, é garantido que o sistema é estável. Após a síntese do supervisor, nenhuma transição será adicionada e somente lugares serão adicionados na rede; com isto, não existe a possibilidade da rede deixar de ser segura;

- A rede não é conservativa, pois o número de fichas se altera com o disparo das transições;

- A rede é viva, pois para qualquer estado alcançado a partir do estado inicial $\mathtt{x}_0$, existe alguma trajetória na qual uma transição qualquer possa disparar;

- A rede não é bloqueante;

- A rede não é persistente, pois para quaisquer duas transições habilitadas, o disparo de uma desabilita o disparo da outra.

Tabela 4.1: Significado dos Lugares

LUGAR	DESCRIÇÃO
P0	Bloco bruto na esteira C1
P1	Bloco bruto no buffer B1
P2	Bloco bruto no robô
P3	Bloco bruto no buffer B3
P4	Bloco bruto na fresa
P5	Bloco fresado no buffer B3
P6	Bloco bruto no robô
P7	Bloco fresado no buffer B5
P8	Tarugo bruto na esteira C2
P9	Tarugo bruto no buffer B2
P10	Tarugo bruto no robô
P11	Tarugo bruto no buffer B4
P12	Tarugo bruto no torno
P13	Pino cônico no buffer B4
P14	Pino cônico no robô
P15	Pino cônico no buffer B6
P16	Produto A
P17	Tarugo bruto no torno
P18	Pino cilíndrico no buffer B4
P19	Pino cilíndrico no robô
P20	Pino cilíndrico no buffer B7
P21	Pino cilíndrico na esteira C3
P22	Pino cilíndrico no buffer B8
P23	Pino cilíndrico na máquina de pintura
P24	Pino cilíndrico pintado no buffer B8
P25	Pino cilíndrico pintado na esteira C3
P26	Pino cilíndrico pintado no buffer B7
P27	Produto B

A matriz de incidência da planta é chamada de $\mathtt{D}_p$ e está mostrada a seguir.

Tabela 4.2: Significado das Transições

TRANSIÇÃO	DESCRIÇÃO
T0	Bloco bruto entra esteira C1 (Tempo de entrada).
T1	Bloco bruto na esteira C1 (Tempo de transporte).
T2	Robô pega bloco bruto do buffer B1. Tempo deslocamento até buffer B1 + Tempo de pegar peça.
T3	Robô coloca bloco bruto no buffer B3. Tempo deslocamento até buffer B3 + Tempo de soltar peça.
T4	Bloco bruto entra na fresa. Início do processo da fresa
T5	Bloco bruto na fresa (Tempo de fresa). Fresa retorna bloco fresado no buffer B3.
T6	Robô pega bloco fresado do buffer B3. Tempo deslocamento até buffer B3 + Tempo de pegar peça.
T7	Robô coloca bloco fresado no buffer B5. Tempo deslocamento até buffer B5 + Tempo de soltar peça.
T8	Tarugo bruto entra esteira C2 (Tempo de entrada).
T9	Tarugo bruto na esteira C2 (Tempo de transporte).
T10	Robô pega tarugo bruto do buffer B2. Tempo deslocamento até buffer B2 + Tempo de pegar peça.
T11	Robô coloca tarugo bruto no buffer B4. Tempo deslocamento até buffer B4 + Tempo de soltar peça.
T12	Tarugo bruto entra no torno. Início processo torno pino cônico.
T13	Tarugo bruto no torno (Tempo de tornear pino cônico). Torno retorna pino cônico no buffer B4.
T14	Robô pega pino cônico do buffer B4. Tempo deslocamento até buffer B4 + Tempo de pegar peça.
T15	Robô coloca pino cônico no buffer B6. Tempo deslocamento até buffer B6 + Tempo de soltar peça.
T16	Máquina Montagem pega bloco fresado e coloca sobre ele pino cônico (Tempo de montagem).
T17	Tarugo bruto entra no torno. Início processo torno pino cilíndrico.
T18	Tarugo bruto no torno (Tempo de tornear pino cilíndrico). Torno retorna pino cilíndrico no buffer B4.
T19	Robô pega pino cilíndrico do buffer B4. Tempo deslocamento até buffer B4 + Tempo de pegar peça.
T20	Robô coloca pino cilíndrico no buffer B7. Tempo deslocamento até buffer B7 + Tempo de soltar peça.
T21	Pino cilíndrico entra na esteira C3.
T22	Pino cilíndrico na esteira C3 (Tempo de transporte).
T23	Pino cilíndrico entra na máquina de pintura. Ínicio processo de pintura.
T24	Máquina Pintura retorna pino cilíndrico pintado no B8 (Tempo de pintura).
T25	Pino cilíndrico pintado entra na esteira C3.
T26	Pino cilíndrico pintado na esteira C3 (Tempo de transporte).
T27	Máquina Montagem pega bloco fresado e coloca sobre ele pino cilíndrico pintado (Tempo de montagem).

$$\mathtt{D}_p = \begin{bmatrix}
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

A matriz das transições não controláveis, $\mathtt{D}_{uc}$, está mostrada a seguir.

$$\mathtt{D}_{uc} = \begin{bmatrix}
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

As restrições de segurança associadas a cada um dos depósitos são listadas abaixo (de 1 a 8). Por exemplo, a primeira restrição implementa overflow ou underflow de peças no depósito B1.

1. $x(P0) + x(P1) \leq 1$
2. $x(P2) + x(P3) + x(P4) + x(P5) \leq 1$
3. $x(P6) + x(P7) \leq 1$
4. $x(P8) + x(P9) \leq 1$
5. $x(P10) + x(P11) + x(P12) + x(P13) + x(P17) + x(P18) \leq 1$
6. $x(P14) + x(P15) \leq 1$
7. $x(P19) + x(P20) + x(P21) + x(P22) + x(P23) + x(P24) + x(P25) + x(P26) \leq 1$
8. $x(P2) + x(P6) + x(P10) + x(P14) + x(P19) \leq 1.$

As oito restrições obedecem à forma $\mathtt{L} \times \mathtt{x} \leq \mathtt{b}$ e a matriz $\mathtt{L}$ é mostrada abaixo.

$$\mathtt{L} = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

O supervisor é calculado por $\mathtt{D}_c = -\ \mathtt{L} \times \mathtt{D}_p$, então, é apresentado abaixo o resultado.

$$\mathtt{D}_c = \begin{bmatrix}
-1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & -1 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

A marcação inicial do supervisor é calculada por $\mathtt{x}_{c0} = \mathtt{b} - \mathtt{L} \times \mathtt{x}_0$ e o resultado é mostrado a seguir. Desta forma, cada lugar de controle possui uma ficha na marcação inicial.

$$\mathtt{x}_{c0} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

O sistema controlado é a Rede de Petri composta pela planta e o supervisor. Na Figura 4.9, apresenta o SFM sob controle do supervisor, ou seja, após a inclusão de todos os lugares de controle $P28$, $P29$, $P30$, $P31$, $P32$, $P33$, $P34$ e $P35$ na RP gerados pelo controle. O supervisor é um "controlador de fichas" que asssegura que as restrições sejam respeitadas controlando a marcação da Rede de Petri que modela a planta. Esta solução de controle é utilizada para implementação em um controlador lógico programável.

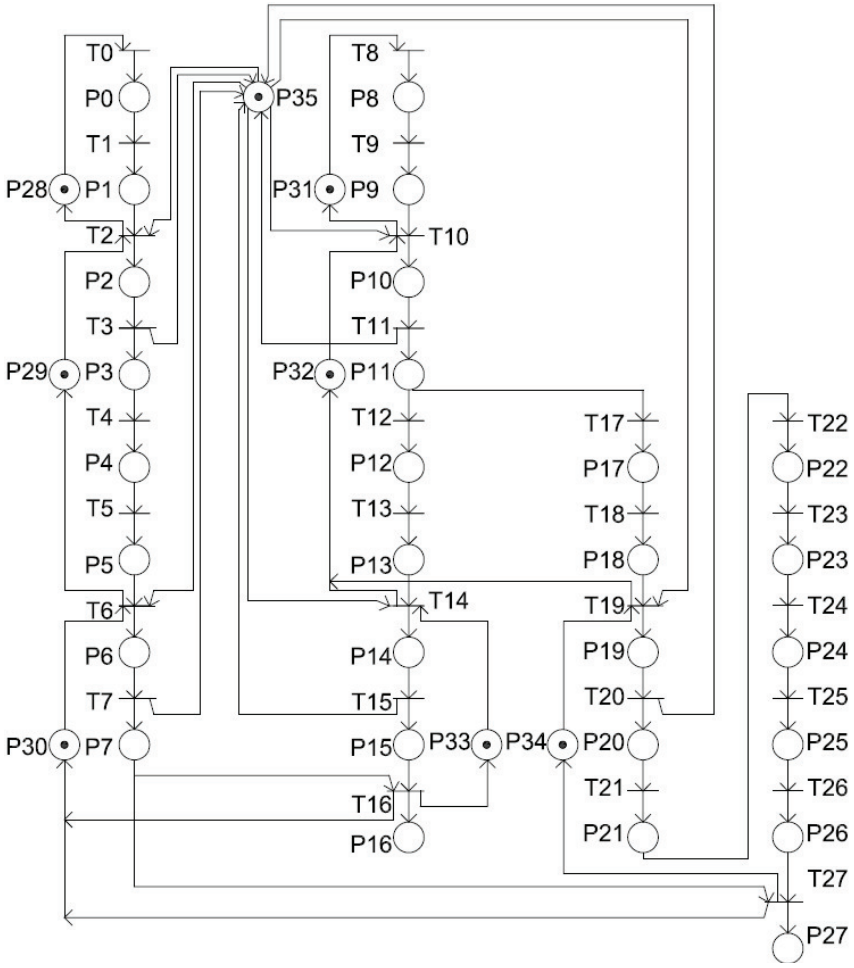

Figura 4.9: Rede de Petri do Sistema Flexível de Manufatura - Com Controle

Tabela 4.3: Significado dos Lugares de Controle

LUGAR DE CONTROLE	DESCRIÇÃO
P28	Buffer B1 - Capacidade 1 bloco bruto
P29	Buffer B3 - Capacidade 1 bloco bruto
P30	Buffer B5 - Capacidade 1 bloco fresado
P31	Buffer B2 - Capacidade 1 tarugo bruto
P32	Buffer B4 - Capacidade 1 tarugo bruto
P33	Buffer B6 - Capacidade 1 pino cônico
P34	Buffer B7 - Capacidade 1 pino cilíndrico
P35	Robô - Controle

Na Tabela 4.3, é relacionado o significado de cada lugar adicionado pelo Controle Supervisório baseado nas Redes de Petri via Invariantes de Lugar para o sistema flexível de manufatura.

Os lugares de controle não inibem os disparos das transições não controláveis, pois satisfazem a desigualdade $\mathtt{L} \times \mathtt{D}_{uc} \leq 0$ e o resultado é apresentado. As restrições de segurança são admissíveis.

$$\mathtt{L} \times \mathtt{D}_{uc} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \end{bmatrix}$$

Capítulo 5

Implementações da Teoria de Controle Supervisório em Controlador Lógico Programável

Nos capítulos 1 e 2, foram apresentados os conceitos, os formalismos para a modelagem e as teorias para o desenvolvimento do controle dos sistemas a eventos discretos. O sistema flexível de manufatura e as estratégias de controle foram apresentados no capítulo 3. Este capítulo apresenta na seção inicial os problemas na implementação do controle. Na seção 5.2, é apresentada a arquitetura de controle. Na seção 5.3, são apresentadas as metodologias de implementação em CLP e na seção 5.4 são apresentados os resultados e feitas as comparações entre as metodologias de implementação em CLP.

5.1 Problemas na Implementação do Controle

A estrutura dividida, em planta e supervisor, proposta por [Ramadge e Wonham, 1989] para a teoria de controle supervisório, permite um funcionamento adequado do sistema dinâmico, no entanto, a implementação desta estrutura não é trivial.

Fabian e Hellgren (1998) listam problemas que surgem quando da implementação da planta e dos supervisores em CLP's através das linguagens de programação.

Sinais e Eventos: a TCS considera que os eventos são assíncronos e ocorrem em instantes de tempo discretos e que as ocorrências determinam as transições dos estados ativos dos supervisores. O CLP processa as variáveis e seus valores são atualizados em cada scan. Dois fatores podem ocorrer: o efeito avalanche e a incapacidade de reconhecer a ordem de eventos. O efeito avalanche é a ocorrência de um evento que resulta na transição indevida de mais de um estado do supervisor e que pode resultar na quebra da sincronia entre a planta física e o supervisor implementado no CLP. Outra questão que se coloca é que durante um scan do CLP é feita a atualização das variáveis de entrada e caso ocorra a transição do sinal de duas variáveis de entrada não será possível saber a ordem de ativação. Neste caso, se as variáveis de entrada são os eventos não controláveis, então o sistema será incapaz de reconhecer a ordem destes eventos não controláveis.

Causalidade: A Teoria de Controle Supervisório assume que todos os eventos são gerados espontaneamente pela planta e que os supervisores devem enviar os sinais de desabilitação dos eventos para a planta conforme apresentado na Figura 5.1.

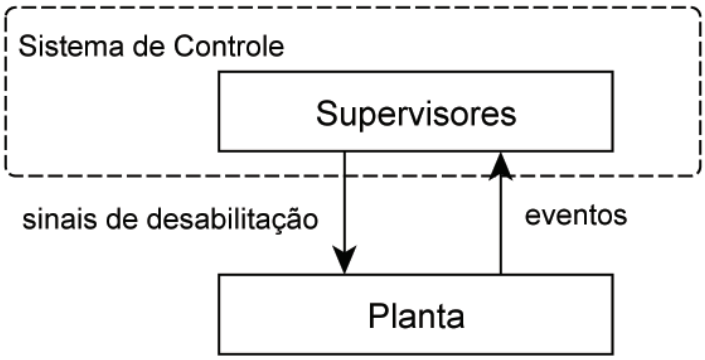

Figura 5.1: Arquitetura de Controle

Entretanto, nas aplicações práticas, os eventos controláveis não são gerados pela planta física, mas são gerados a partir das respostas aos comandos dados pelos supervisores. Assim, o problema da causalidade é gerado devido ao seguinte questionamento: quem é responsável por gerar os eventos controláveis, o supervisor ou a planta?

Escolha de Eventos: a TCS garante à planta um comportamento com maior liberdade por meio do supervisor minimamente restritivo. Depois da ocorrência da sequência de eventos, podem existir várias opções para o comportamento da planta prosseguir. Assim, quando o supervisor é responsável pela geração de uma parcela de eventos, ele pode ter que escolher entre as opções em um único estado.

Sincronização Inexata: durante a execução do programa, a mudança de um sinal de entrada pode ocorrer e esta alteração será reconhecida no início do próximo scan do CLP. Neste caso, a comunicação entre a planta e o CLP está sujeita a atrasos. Então, a sincronização inexata é um problema que ocorre, quando uma mudança no sinal de entrada no CLP invalida a ação de controle que corresponde à geração de um evento controlável segundo [Fabian e Hellgren, 1998].

Além destes problemas citados, Queiroz e Cury (2000) citam outro problema que é a necessidade de representar o sistema com uma abstração que considere somente os eventos necessários às tarefas de coordenação, a fim de evitar a explosão do espaço de estados e permitir a síntese dos supervisores. Durante a fase de modelagem é necessário ter ideia das atividades referentes aos modelos e considerar os eventos relevantes necessários às tarefas de coordenação entre os subsistemas.

5.2 Arquitetura de Controle

Para resolver os problemas citados na implementação da Teoria de Controle Supervisório em CLP's, Queiroz e Cury (2002) propuseram uma Arquitetura de Controle. Esta

arquitetura é a base para os métodos da implementação do Controle Supervisório no controlador lógico programável e é composta por quatro níveis: Supervisores Modulares, Sistema-Produto, Sequências Operacionais e Sistema Real apresentados na Figura 5.2.

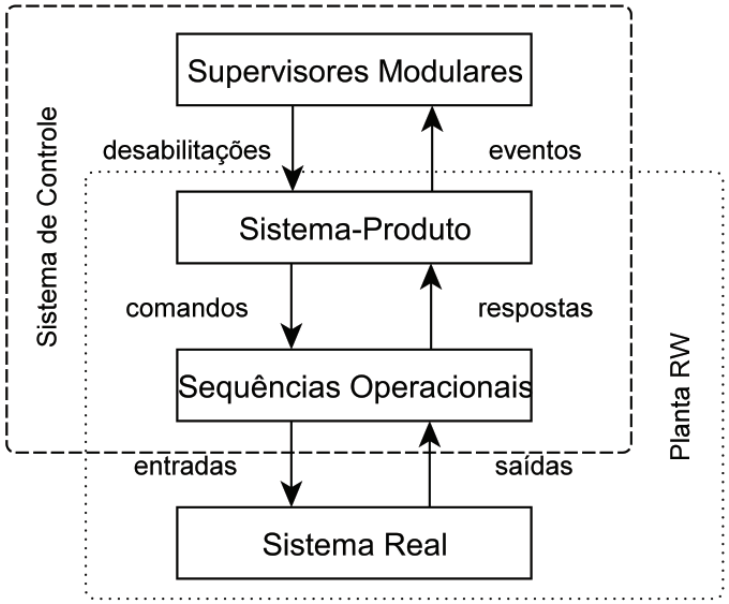

Figura 5.2: Arquitetura de Controle (Queiroz e Cury, 2002)

No nível **Supervisores Modulares**, os supervisores obtidos pelo CML apresentados na Figura 4.7 são implementados e os estados ativos dos supervisores são atualizados com a sinalização de eventos provenientes do nível do Sistema-Produto. Este nível associa aos estados ativos um conjunto de sinais de desabilitação dos eventos controláveis que controlam o nível do Sistema-Produto.

No nível **Sistema-Produto**, os modelos dos equipamentos (Figura 4.5) são implementados e os eventos controláveis são executados sempre que não forem desabilitados no nível dos Supervisores Modulares. Neste nível, também são ativados os comandos para o nível das Sequências Operacionais. Os eventos não controláveis são relacionados às respostas do nível das Sequências Operacionais.

No nível **Sequências Operacionais**, é feita a interface entre a solução de controle proposta e a planta física. Este nível recebe os comandos do nível do Sistema-Produto, fornece os sinais para as entradas da planta física e observa os sinais das saídas da planta física, fornecendo-os para o nível do Sistema-Produto como respostas.

No nível **Sistema Real** está a planta física.

5.3 Metodologias de Implementação em CLP

As metodologias de implementação em CLP utilizadas foram aquelas abordadas nos trabalhos de Queiroz e Cury (2002), Leal et al. (2009), Vieira (2007) e Lima II (2002). Estas metodologias são programadas no CLP CompactLogix modelo 1769-L32E, da Allen-

Bradley Rockwell Automation por meio do software RSLogix 5000.

5.3.1 Implementação I (Queiroz e Cury, 2002)

A metodologia de implementação em CLP utiliza a arquitetura de controle proposta na Figura 5.2.

Os supervisores modulares apresentados na Figura 4.7 e o sistema-produto composto pelos equipamentos apresentados na Figura 4.5 são implementados em CLP em uma lógica de programação utilizando a linguagem Ladder Diagram. As sequências operacionais são implementadas em CLP em uma lógica de programação utilizando a linguagem SFC.

Cada estado é uma variável do tipo booleana. Para cada transição do autômato é implementada uma linha de programação. Quando a instrução representando o estado anterior e a instrução representando um evento forem ativadas, o próximo estado recebe o valor 1, o estado anterior e o evento executado recebem valor 0.

Na implementação de um supervisor reduzido, o código em ladder deve refletir além das mudanças de estado decorrentes da execução de um evento, as desabilitações. Por exemplo, o autômato S_{1red} é apresentado na Figura 5.3.

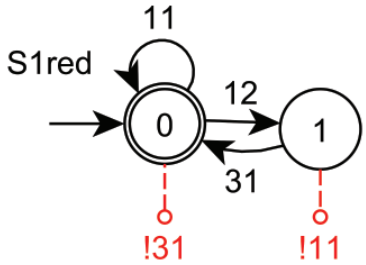

Figura 5.3: Supervisor S1red

A Figura 5.4 apresenta a lógica que implementa o autômato S_{1red}.

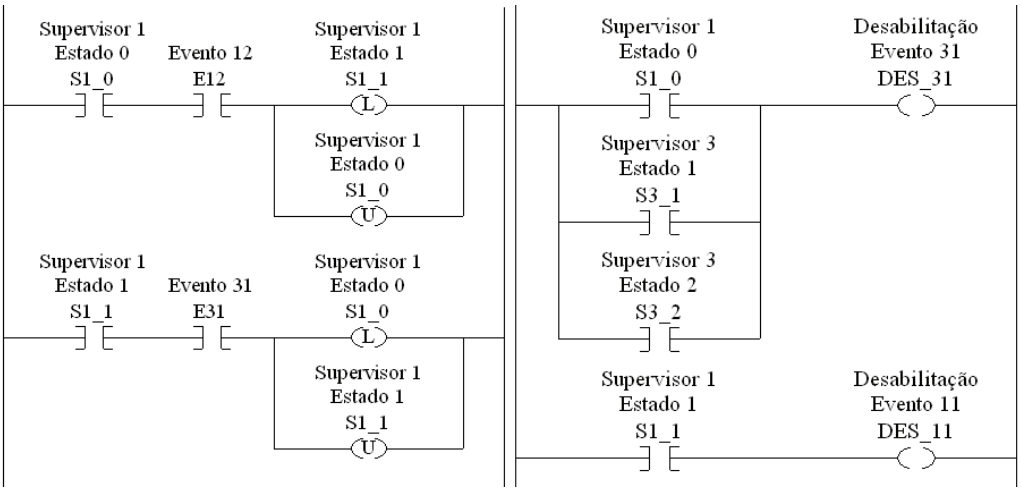

Figura 5.4: Lógica - S_{1red}

As variáveis S1_0 e S1_1 representam os estados. As variáveis E12 e E31 representam os eventos e as variáveis DES_31 e DES_11 representam os sinais de desabilitação de eventos controláveis associadas aos estados de S_{1red}. Quando as variáveis S1_0 e E12 estiverem com o valor 1, a variável S1_1 recebe o valor 1 e S1_0 recebe o valor 0, indicando que o supervisor está no estado 1. Quando as variáveis S1_1 e E31 estiverem com o valor 1, a variável S1_0 recebe o valor 1 e S1_1 recebe o valor 0, indicando que o supervisor está no estado 0. Quando a variável S1_0 estiver com o valor 1, a variável DES_31 recebe o valor 1, indicando que no estado 0 o supervisor desabilita o evento 31. As variáveis S3_1 e S3_2 representam os estados do autômato S_{3red} que também ativam a variável DES_31. Quando a variável S1_1 estiver com o valor 1, a variável DES_11 recebe o valor 1, indicando que no estado 1 o supervisor desabilita o evento 11.

No caso da implementação do Sistema-Produto, apenas estados e transições precisam ser implementados. Nesta implementação, não ocorrem duas transições seguidas no Sistema-Produto sem que os Supervisores Modulares tenham sido atualizados. Desta forma, é feito um pulo no final de cada transição para a primeira linha de programação dos supervisores. Por exemplo, a esteira C1 é representada pelo autômato mostrado na Figura 5.5.

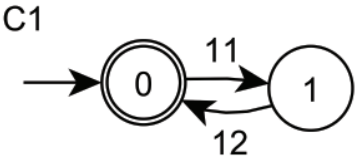

Figura 5.5: Autômato Representando Esteira C1

A Figura 5.6 apresenta a lógica que implementa o autômato da esteira C1. A primeira linha de programação é relacionada com o evento não controlável (12) que é tratado antes do evento controlável (11) na segunda linha de programação. As variáveis C1_0 e C1_1 representam os estados. As variáveis E11 e E12 representam os eventos. A variável $START\_ESTEIRA1$ inicia a sequência operacional da esteira. A variável $END\_ESTEIRA1$ sinaliza o fim da sequência operacional da esteira. A variável SUP é relacionada com a chamada da lógica do supervisor e a variável DES_11 representa a desabilitação associada ao estado 1 de S_{1red}.

Na primeira linha de programação, quando a esteira estiver em operação, a variável C1_1 possui o valor 1 e ao final da operação da esteira, a variável $END\_ESTEIRA1$ recebe o valor 1. Nesta condição, o evento não controlável é ativado (E12 recebe o valor 1) indicando a ocorrência de transição no autômato. A variável C1_0 recebe o valor 1 indicando que o autômato está no estado 0. A variável C1_1 recebe o valor 0 indicando que foi resetada. A variável $END\_ESTEIRA1$ recebe o valor 0 indicando que foi resetada e a variável SUP é ativada indicando um salto desta lógica para a lógica dos supervisores.

Na segunda linha de programação, quando a esteira não está em operação, a variável

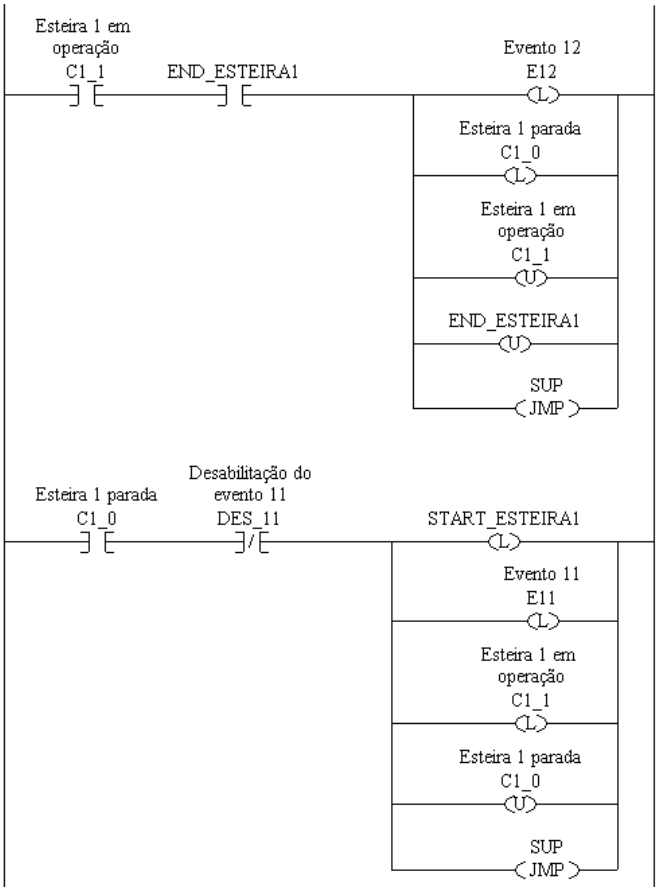

Figura 5.6: Lógica Esteira C1

C1_0 possui o valor 1 e o evento controlável (11) não está desabilitado pelo supervisor (a variável DES_11 está com o valor 0). A sequência operacional da esteira é iniciada, quando $START\_ESTEIRA1$ recebe valor 1. O evento controlável é ativado (E11 recebe o valor 1) indicando a ocorrência de transição no autômato. A variável C1_1 recebe o valor 1 indicando que o autômato está no estado 1. A variável C1_0 recebe o valor 0 indicando que foi resetada e a variável SUP é ativada indicando um salto desta lógica para a lógica dos supervisores.

Observa-se que a instrução JMP com o tag SUP está no final de cada linha de programação das transições não controláveis e controláveis devido aos supervisores necessitarem de estar atualizados antes da ocorrência de um novo evento controlável. Assim, um único evento é tratado em cada scan do programa no CLP.

A esteira C1 possui um motor, um sensor fotoelétrico localizado no início da esteira, indicando a presença de peça no início e outro sensor fotoelétrico localizado no final da esteira, indicando a presença de peça no final da esteira. A sequência operacional da esteira C1 é apresentada na Figura 5.7.

Quando a sequência operacional é iniciada, a variável $START\_ESTEIRA1$ está com o valor 1 e é detectada a presença de peça no início da esteira (variável $C1\_IC$ está com o valor 1), então o motor da esteira é energizado pela saída $Local : 3 : O.Data.0$ do CLP por

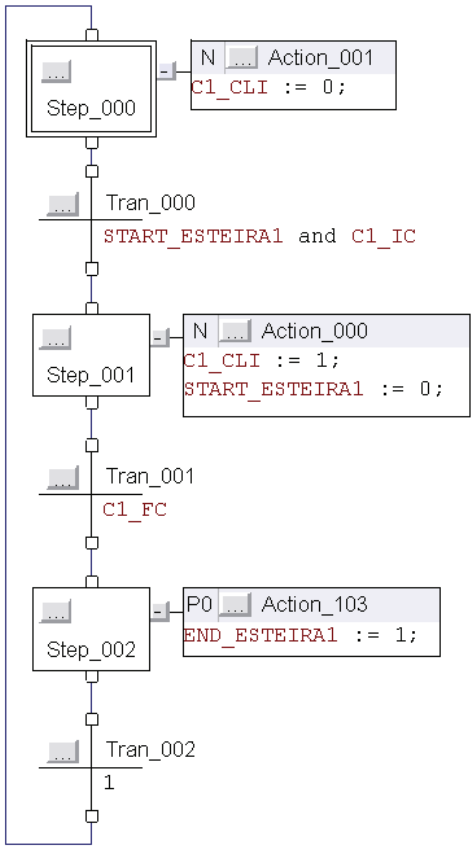

Figura 5.7: Sequência Operacional Esteira C1

meio da variável $C1\_CLI$ e a variável $START\_ESTEIRA1$ recebe o valor 0, indicando que foi resetada. Quando o sensor fotoelétrico indicar a presença da peça na posição final da esteira, a variável $C1\_FC$ recebe o valor 1. O motor é desenergizado e a sequência operacional da esteira é finalizada.

As lógicas de programação desta metodologia de implementação estão em uma única rotina principal, assim, a estrutura do programa não é modular.

No apêndice A, como exemplo, são apresentadas as lógicas de programação do supervisor S_{1red}, autômato da esteira C1 e do Robô, sequência operacional da esteira C1 e do Robô e parte da lógica principal deste método de implementação em CLP aplicado ao sistema flexível de manufatura.

5.3.2 Implementação II (Leal et al., 2009)

A implementação II também utiliza a arquitetura de controle proposta por Queiroz e Cury (2002), porém, as rotinas dos níveis Sistema-Produto e Supervisores Modulares são separadas de acordo com a controlabilidade dos eventos, conforme apresentado na Figura 5.8. Primeiramente são implementadas todas as transições com os eventos não controláveis e, posteriormente, todas as transições com os eventos controláveis, assim, consegue-se tratar em um único ciclo de scan os vários eventos gerados pela planta física

que são identificados na leitura das entradas do CLP. Em seguida, a atualização dos estados do Sistema-Produto e dos Supervisores Modulares é feita.

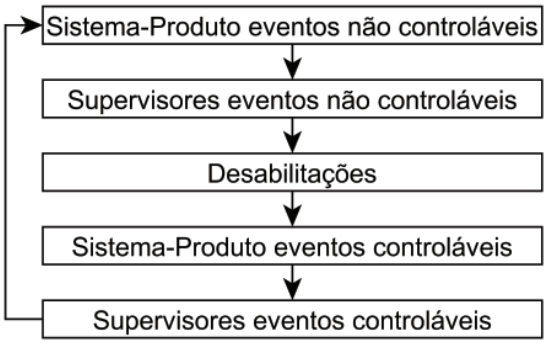

Figura 5.8: Fluxograma Parcial

A Figura 5.9 apresenta o fluxograma completo da implementação em CLP proposta por Leal et al. (2009).

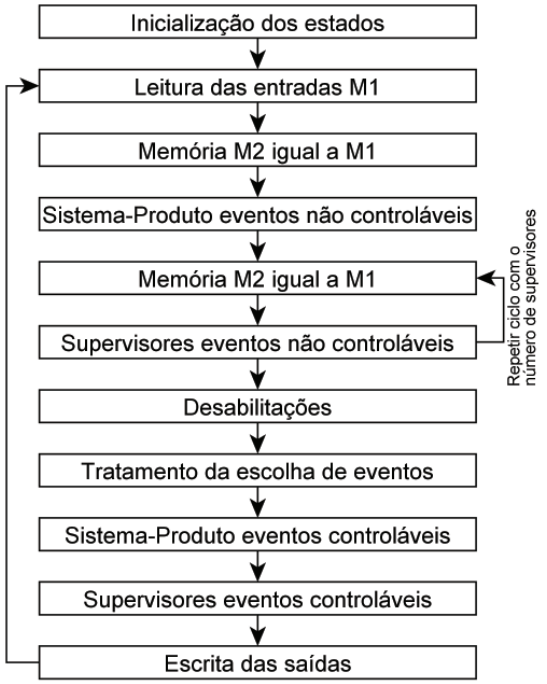

Figura 5.9: Fluxograma Completo

Os supervisores modulares apresentados na Figura 4.7 e o sistema-produto composto pelos equipamentos apresentados na Figura 4.5 são implementados em CLP em uma lógica de programação utilizando a linguagem Ladder Diagram.

Na subrotina *Inicialização dos estados*, as instruções que representam os estados iniciais dos supervisores e dos equipamentos que compõem o sistema-produto recebem o valor 1 (nível lógico 1). Considera-se que este procedimento deve ser realizado apenas uma única vez no primeiro ciclo de scan do CLP. Na inicialização, é implementada a lógica que representa os estados iniciais dos supervisores e do sistema-produto, Figura 5.10. Por

exemplo, a variável $S1\_0$ recebe o valor 1 e indica que o autômato do supervisor S_{1red} está no estado 0.

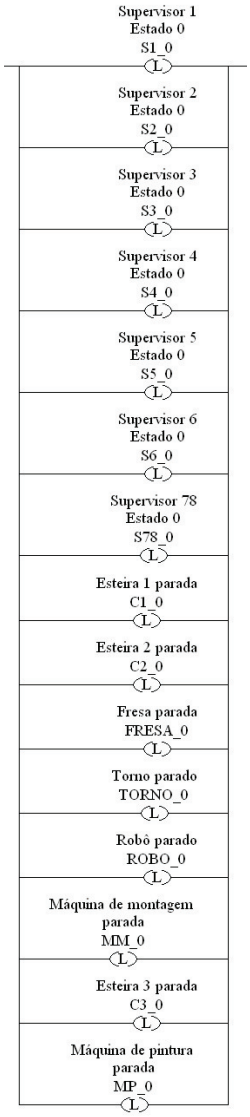

Figura 5.10: Inicialização

Na subrotina *Leitura das entradas M1*, são identificados os eventos gerados pela planta física e que correspondem aos eventos não controláveis. Estes eventos são armazenados em uma memória M1 do CLP e após as transições de estados no sistema-produto, estes eventos são apagados.

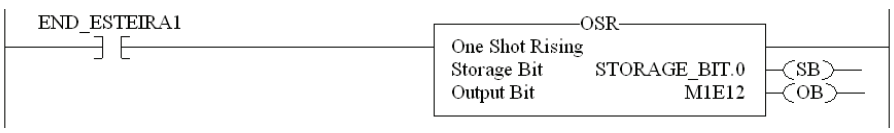

Figura 5.11: Leitura das Entradas

Na Figura 5.11, o tag $END\_ESTEIRA1$ representa o sinal proveniente da planta física e a instrução OSR identifica a alteração no valor deste sinal e armazena temporariamente na mémoria interna M1.

A subrotina *Memória M2 igual a M1* mantém os eventos não controláveis armazenados para permitir as transições de estados nos supervisores. A lógica é apresentada na Figura 5.12 .

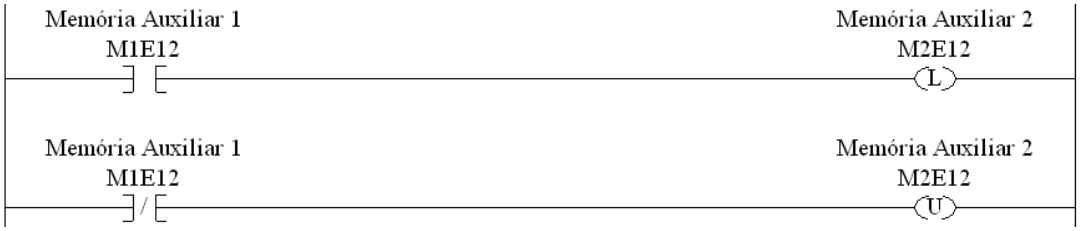

Figura 5.12: Memória M2 igual a M1

Na subrotina *Sistema-Produto eventos não controláveis*, são implementadas apenas as transições de estados com os eventos não controláveis e após as transições, estes eventos são resetados. A lógica é apresentada na Figura 5.13.

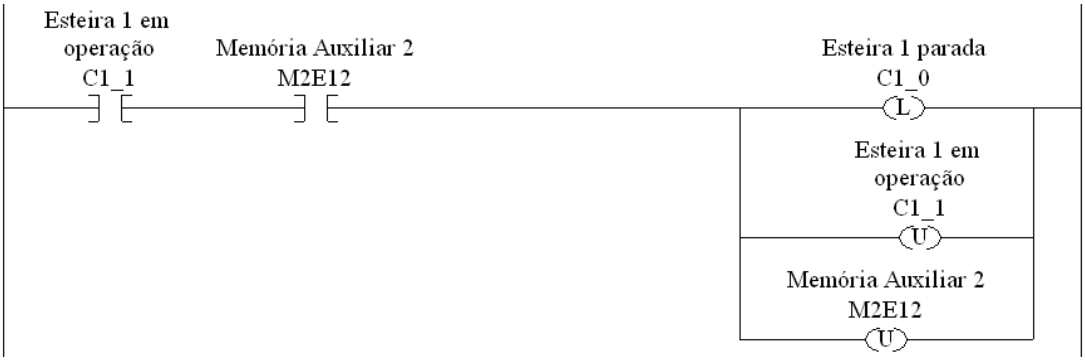

Figura 5.13: Sistema-Produto Evento Não Controlável

A subrotina *Supervisores eventos não controláveis* realiza apenas as transições de estados com os eventos não controláveis, Figura 5.14, desta forma, é necessário resgatar estes eventos por meio da memória M2 e associar aos estados ativos um conjunto de sinais de desabilitação dos eventos controláveis na subrotina *Desabilitações*.

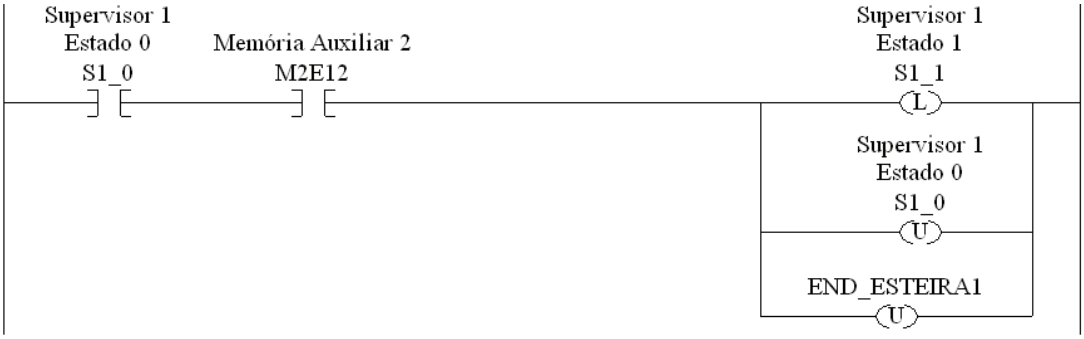

Figura 5.14: Supervisor Evento Não Controlável

As desabilitações são implementadas de acordo com o que foi proposto por Queiroz e Cury (2002), ou seja, as desabilitações dos eventos são implementadas considerando os estados dos supervisores. Por exemplo, na Figura 5.15 a variável $S1\_0$ com valor 1 ativa a variável DES_11, ou seja, o estado 1 do supervisor S_{1red} desabilita o evento 11.

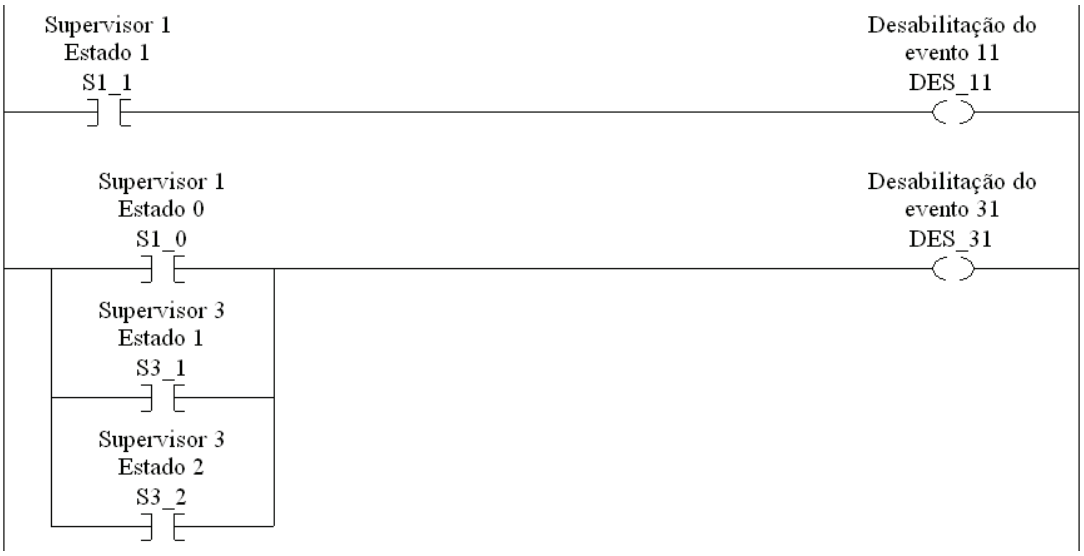

Figura 5.15: Desabilitação

O problema da Escolha deve ser considerado sempre que há mais do que um evento controlável habilitado, para tanto não basta olhar apenas para cada supervisor isoladamente, mas sim para o conjunto de desabilitações. A subrotina *Tratamento da escolha de eventos* é executada quando dois ou mais eventos pertencentes ao mesmo estado estão ativos, desta forma esta rotina escolhe um evento a ser desabilitado. Dois ou mais eventos estão ativos se não forem desabilitados por qualquer supervisor e a ocorrência dos mesmos é prevista no modelo que implementa o Sistema-Produto.

No caso do sistema flexível de manufatura, o supervisor modular S_{r4} possui dois eventos controláveis factíveis, 51 e 53, no mesmo estado. A implementação da lógica para este problema consiste na alternância entre eles, conforme Leal et al. (2009) (Figura 5.16).

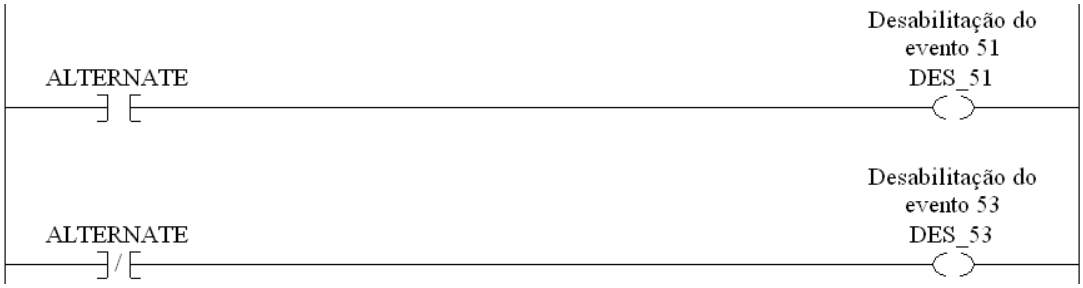

Figura 5.16: Escolha entre 51 ou 53

A subrotina *Sistema-Produto eventos controláveis* corresponde à geração do evento controlável pelo sistema-produto. Nesta subrotina, é implementada a lógica correspondente à geração dos eventos controláveis que são possíveis de ocorrer na planta física, Figura 5.17. Nesta figura, as variáveis C1_0 e C1_1 representam os estados. A variável E11 representa o evento controlável. A variável $START\_ESTEIRA1$ inicia a sequência operacional da esteira e a variável DES_11 representa a desabilitação associada ao estado 1 de S_{1red}. Quando a esteira não está em operação, a variável C1_0 possui o valor 1 e o evento controlável (11) não está desabilitado pelo supervisor (variável DES_11 está com o valor 0). A sequência operacional da esteira é iniciada quando $START\_ESTEIRA1$ recebe valor 1. O evento controlável é ativado (E11 recebe o valor 1) indicando a ocorrência de transição no autômato. A variável C1_1 recebe o valor 1 indicando que o autômato está no estado 1 e a variável C1_0 recebe o valor 0 indicando que foi resetada.

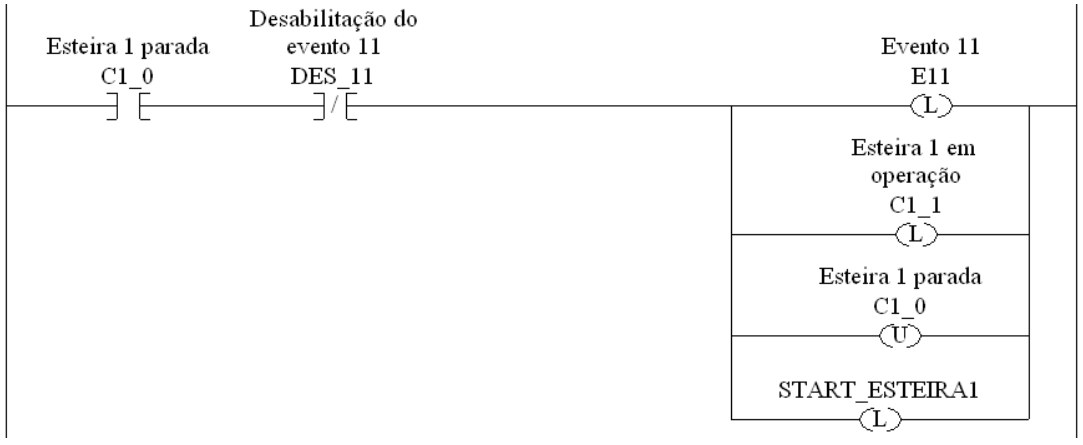

Figura 5.17: Sistema-Produto Evento Controlável

Na subrotina *Supervisores eventos controláveis*, é feita a atualização dos estados dos supervisores com os eventos controláveis gerados. Por exemplo, a lógica é apresentada na Figura 5.18. As variáveis S1_0 e S1_1 representam os estados. A variável E31 representa o evento controlável. Quando as variáveis S1_1 e E31 estiverem com o valor 1, a variável E31 é resetada. A variável S1_0 recebe o valor 1 e S1_1 recebe o valor 0, indicando que o supervisor está no estado 0.

Figura 5.18: Supervisor Evento Controlável

A subrotina *Escrita das saídas* promove os acionamentos das saídas do CLP com os eventos controláveis para a planta física. Por exemplo, a lógica de escrita da esteira é apresentada na Figura 5.19. A esteira C1 possui um motor, um sensor fotoelétrico localizado no início da esteira indicando a presença de peça no início e outro sensor fotoelétrico localizado no final da esteira indicando a presença de peça no final da esteira. A peça é detectada pelo sensor fotoelétrico por meio do tag $C1\_IC$ (entrada Local:6:I.Data.0), logo o motor da esteira é energizado por meio do tag $C1\_CLI$ (saída Local:3:O.Data.0) e quando o sensor fotoelétrico por meio do tag $C1\_FC$ (entrada Local:6:I.Data.1) indicar a presença da peça na posição final da esteira, o motor é desenergizado.

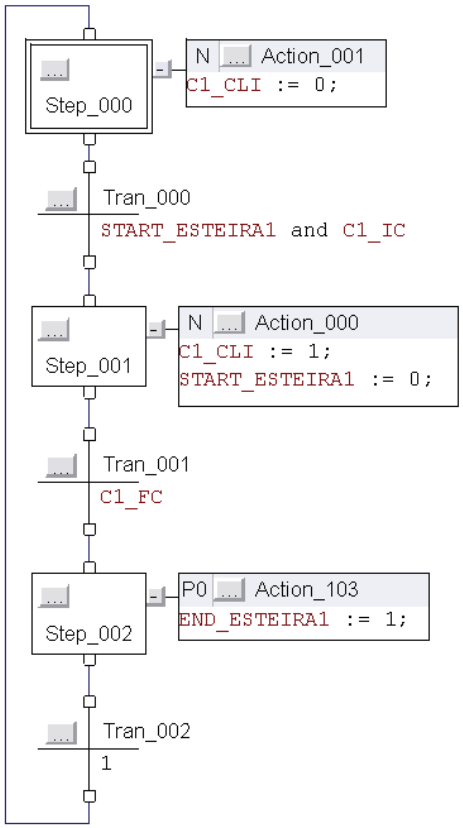

Figura 5.19: Escrita das Saídas

Ao final, um novo ciclo é iniciado.

As lógicas de programação desta metodologia de implementação estão conectadas na rotina principal, assim, a estrutura do programa é modular.

No apêndice B, como exemplo, são apresentadas as lógicas de programação do supervisor S_{1red}, autômato da esteira C1 e do Robô, sequência operacional da esteira C1 e do Robô e parte da lógica principal deste método de implementação em CLP aplicado ao sistema flexível de manufatura.

5.3.3 Implementação III (Vieira, 2007)

A metodologia de implementação III também utiliza a arquitetura de controle proposta por Queiroz e Cury (2002).

Vieira (2007) considera para a metodologia de implementação que cada supervisor, representação por sistema-produto e procedimento operacional será representado por meio da linguagem SFC na lógica do CLP.

O método de implementação define procedimentos sistemáticos que permitem converter os autômatos dos supervisores modulares apresentados na Figura 4.7 e do sistema-produto composto pelos equipamentos apresentados na Figura 4.5 nos códigos SFC's correspondentes. Algumas orientações são dadas para a implementação do procedimento operacional em SFC.

Na metodologia de implementação, é feito um programa principal, SFC Main, que estabelece seis modos de operação do sistema (Figura 5.20).

A ação *action_SI* do modo **Software Initialization (SI)** é executada apenas uma vez e todos os SFC's nos conjuntos dos supervisores $\{s_j|j \in J\}$, das plantas $\{g_i|i \in I\}$ e dos procedimentos operacionais $\{o|\sigma \in \Sigma\}$ são inicializados e às outras variáveis envolvidas no programa são atribuídos os valores pré-determinados.

Após a inicialização do software, a transição PSinit é ativada e o programa executa a ação *action_PSI* do modo **Physical System Initialization (PSI)**. A ativação faz com que o sistema a ser controlado vá para o estado inicial.

Após a inicialização do sistema físico, a transição PSready é ativada e o modo **PSIted** é executado e desta forma o sistema de controle entra no modo Idle.

Enquanto o sistema está no modo Idle, existem duas possibilidades para controlar o sistema físico, sendo uma por meio da transição Manual e a outra por meio da transição Superv. As transições podem ser ativadas pelo sistema de supervisão ou botoeiras.

Caso a transição Manual ocorra, a ação *action_Man* do modo **Manual (Man)** é executada e neste modo, o operador coordena a evolução do sistema. Isto é possível por intermédio da desativação seletiva dos sinais de desabilitação associados aos eventos controláveis.

Caso a transição Superv ocorra, a ação *action_Sup* do modo **Supervisionado (Sup)**

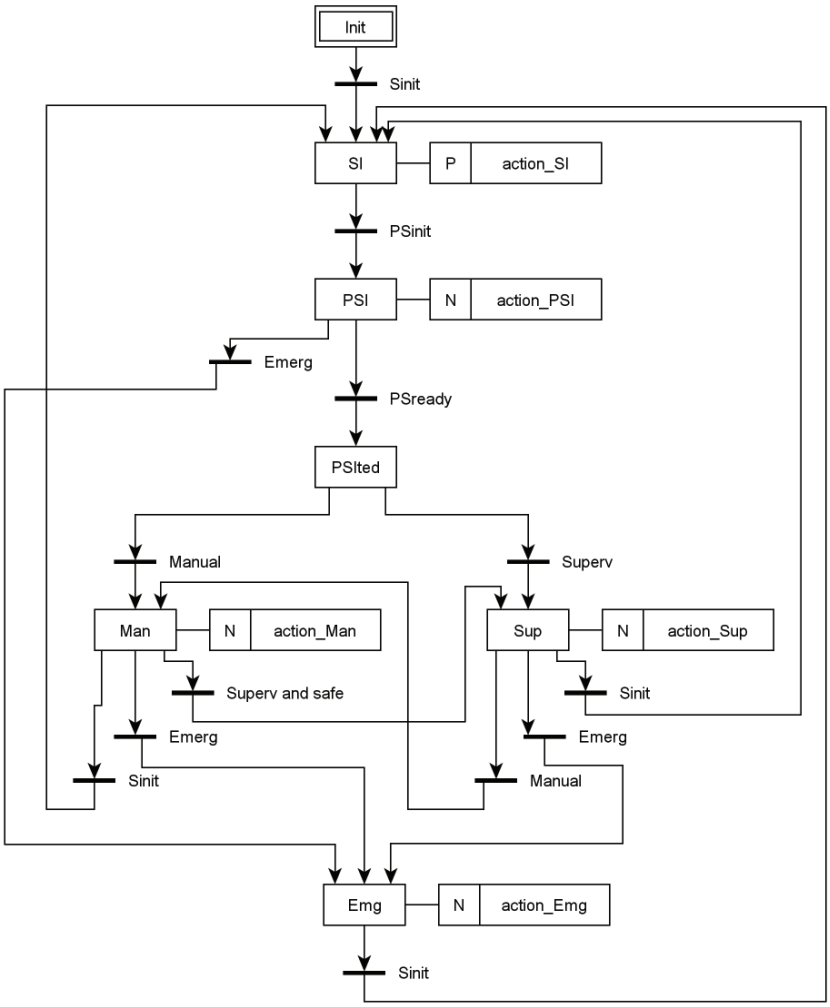

Figura 5.20: SFC Main

é executada e a coordenação das ações no sistema físico ocorre de acordo com o estado ativo do conjunto de supervisores. A ação *action_Sup* é executada uma vez a cada ciclo de atualização do CLP e executa as seguintes sequências de atividades, conforme [Vieira, 2007]:

1. Atualização do estado ativo dos supervisores com os eventos tratados no ciclo de atualização anterior, bem como, atualização da ação de controle dos supervisores. Isto é realizado através da chamada do Function Block Supervisor Modular (FB MS);

2. Desativação condicional das variáveis em $\{g_i evt | i \in I\}$, $g_i evt$ sinaliza que foi tratado algum evento na planta correspondente. À cada variável em $\{g_i evt | i \in I\}$ deve ser estabelecido o valor lógico FALSO sempre que toda variável no conjunto do comando do evento controlável C_i correspondente tiver valor lógico FALSO;

3. Suspensão do tratamento de todo evento controlável para priorizar o tratamento de eventos não controláveis. Isto é realizado através da ativação da variável denominada

Tabela 5.1: Functions Blocks

Nome	Tipo	Comentário
s_j $j \in J$	FB	Representa o supervisor S_j
MS	FB	Estrutura lógica que executa a chamada ordenada de todos os FB's em $\{s_j \mid j \in J\}$
g_i $i \in I$	FB	Representa o módulo do sistema-produto associado ao subsistema G_i
dg_i $i \in I$	FB	Estrutura lógica que suspende ou autoriza o tratamento de eventos em Σ^{G_i}
PS	FB	Estrutura lógica que executa a chamada ordenada de todos os FB's em $\{g_i \mid i \in I\}$ e $\{dg_i \mid i \in I\}$
o_σ $\sigma \in \Sigma$	FB	Representa o procedimento operacional o_σ
OP	FB	Estrutura lógica que executa a chamada ordenada de todos os FB's em $\{o_\sigma \mid \sigma \in \Sigma\}$
Main	Program	Coordena a operação da arquitetura

CED (Desativação de Evento Controlável);

4. Tratamento de eventos não controláveis. Se ocorreu algum evento não controlável e a resposta correspondente armazena valor maior do que zero, então a ocorrência do evento será sinalizada. Isto é realizado através da chamada do Function Block Sistema-Produto (FB PS);

5. Autorização do tratamento de eventos controláveis. Isto é realizado através da desativação da variável mencionada no terceiro ítem;

6. Tratamento de eventos controláveis, realizado através de nova chamada do Function Block Sistema-Produto (FB PS);

7. Atualização de todos os procedimentos operacionais em $\{o_\sigma \mid \sigma \in \Sigma\}$, o_σ é um procedimento operacional vinculado a um evento pertencente ao alfabeto. Isto é realizado através da chamada do Function Block Procedimento Operacional (FB OP).

O modo **Supervisionado** faz chamada dos functions blocks que são rotinas implementadas para executar as ações dos supervisores, sistema-produto e procedimentos operacionais. A Tabela 5.1 [Vieira, 2007] apresenta os functions blocks.

A transição emerg é ativada, quando os procedimentos de emergência são necessários e devem ser executados, assim, a ação *action_Emg* do modo **Emergência (Emg)** suspende imediatamente todas as atividades que até então estavam sendo realizadas, consequentemente, os procedimentos de emergência são executados.

Os modos Manual e Emergência desta metodologia não foram implementados, pois as implementações deste trabalho consideram apenas a operação supervisionada da planta.

O Function Block $\{s_j|j \in J\}$ representa os supervisores modulares locais reduzidos S_j obtidos no processo de síntese dos supervisores. O Function Block MS executa a chamada ordenada de cada FB em $\{s_j|j \in J\}$, desta forma, a atualização do estado ativo do conjunto de $\{s_j|j \in J\}$ em um único ciclo de atualização do CLP é feita. O FB MS também estabelece o estado dos sinais de desabilitação, ou seja, o estado das variáveis nos conjuntos $\{dg_i|i \in I\}$. Os functions blocks MS e $\{s_j|j \in J\}$ constituem o nível dos Supervisores Modulares da Arquitetura de Controle Supervisório apresentada em [Queiroz e Cury, 2002].

O Function Block $\{g_i|i \in I\}$ representa o módulo do sistema-produto. O Function Block em $\{dg_i|i \in I\}$ permite a autorização ou suspensão do tratamento de eventos de todo subsistema que compartilha alguma célula de controle com o subsistema em questão. O FB PS realiza a chamada ordenada dos FB's $\{g_i|i \in I\}$ e $\{dg_i|i \in I\}$, sendo que a chamada de cada FB em $\{dg_i|i \in I\}$ é executada imediatamente antes de se realizar a chamada do correspondente FB em $\{g_i|i \in I\}$. O FB PS deve chamar o FB g_i somente se o tratamento de eventos em G_i não foi suspenso pelo FB dg_i, isto é, somente se a variável g_id apresentar valor lógico FALSO. Os functions blocks $\{g_i|i \in I\}$ e $\{dg_i|i \in I\}$ e PS constituem o nível Sistema-Produto (PS) da Arquitetura de Controle Supervisório apresentada em [Queiroz e Cury, 2002].

O Function Block $\{o_\sigma|\sigma \in \Sigma\}$ representa os procedimentos operacionais associados aos eventos $\sigma \in \Sigma$. O FB OP executa a chamada ordenada de todos os FB's em $\{o_\sigma|\sigma \in \Sigma\}$. Os functions blocks $\{o_\sigma|\sigma \in \Sigma\}$ e OP constituem o nível dos Procedimentos Operacionais (OP) da Arquitetura de Controle Supervisório apresentada em [Queiroz e Cury, 2002].

A seguir, serão descritos os procedimentos para a obtenção dos supervisores modulares, dos módulos do sistema-produto e os procedimentos operacionais.

SFC's e FB's no conjunto $\{s_j|j \in J\}$

Esta seção apresenta o procedimento sistemático para a conversão de cada supervisor no conjunto $\{s_j|j \in J\}$ no correspondente SFC que será implementado no CLP. Primeiramente, é definida a base de dados correspondentes às variáveis do CLP e posteriormente, os FB's correspondentes aos SFC's.

Definição das Variáveis do CLP

Na definição das variáveis, considere o conjunto dos supervisores $S_j = (S_j, \Phi_j), j \in J$, com $S_j = (\Sigma^{S_j}, Q^{S_j}, \delta^{S_j}, q_0^{S_j}, Q_m^{S_j})$ e $\Phi_j : Q^{S_j} \rightarrow 2^{\Sigma_c}$. Os eventos sujeitos à ação desabilitadora do supervisor S_j são aqueles, conforme [Vieira, 2007], definidos em $\Sigma_p^{S_j} = \cup_{\forall q \in Q^{S_j}} \Phi_j(q)$.

O evento $\sigma \in \Sigma_p^{S_j}$ que será desabilitado pelo conjunto de supervisores é relacionado com uma variável booleana do CLP $\sigma ds_j \in P_j, j \in J$. O conjunto de variáveis do CLP

P_j é tal que $|P_j| = |\Sigma_p^{S_j}|$. A variável σds_j é ativada, quando o supervisor S_j estabelece a desabilitação do evento associado a tal variável e assim, as variáveis σds_j, ativam as variáveis $\sigma d \in D_i, i \in D$ do CLP que são relacionadas às desabilitações dos eventos e que pertencem ao conjuntos $\{D_i | i \in D\}$.

Exemplo: De acordo com os supervisores, Seção 4.7, a função de desabilitação dos eventos controláveis referentes aos eventos E11d e E31d está sujeita a uma ação de controle dos supervisores S1 e S3. Desta forma, é feita uma relação dos referidos eventos com os respectivos supervisores e as variáveis do CLP:

- o Supervisor S1: $(11, E11DS1)$;
- o Supervisor S1: $(31, E31DS1)$;
- o Supervisor S3: $(31, E31DS3)$.

A Figura 5.21 apresenta a desabilitação do evento controlável.

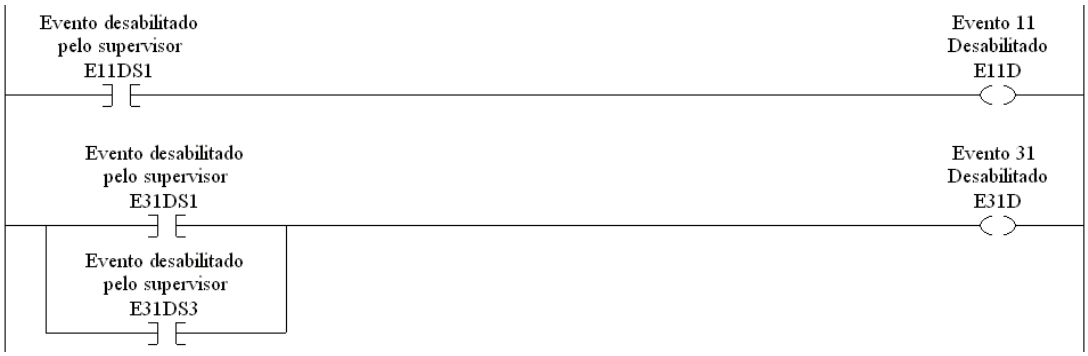

Figura 5.21: Desabilitação Evento Controlável

Obtenção do SFC $S_j = \left(X^{S_j}, T^{S_j}, X_0^{S_j}\right), j \in J$

Considerando o conjunto de supervisores S_j, o conjunto de eventos cuja ocorrência proporciona a transição do estado q para um estado distinto q' do supervisor S_j é definido como $\Sigma_{q,q'}^{S_j} = \left\{\sigma \in \Sigma | \left(q' = \delta^{S_j}(q, \sigma)\right) \wedge (q \neq q')\right\}$

O SFC $S_j, j \in J$, é obtido realizando os itens abaixo conforme [Vieira, 2007]:

1. Um passo $xq \in X^{S_j}$ é associado a cada estado $q \in Q^{S_j}$, de forma que $|X^{S_j}| = |Q^{S_j}|$;
2. $x_0^{S_j}$ é o passo inicial associado ao estado inicial $q_0^{S_j}$;
3. Uma transição $(xq, xq') \in T^{S_j}$ é associada a cada par ordenado $(q, q') \in \left\{Q^{S_j}\ x\ Q^{S_j}\right\}$, quando $\Sigma_{q,q'}^{S_j} \neq \emptyset$, onde $xq, xq' \in X^{S_j}$ estão respectivamente associados à q, q'. Esta associação tem de ser feita de modo que $|T^{S_j}|$ seja igual ao número de par ordenado com $\Sigma_{q,q'}^{S_j} \neq \emptyset$.

A condição de transição associada à cada transição em $T^{S_j}, j \in J$, é a expressão Booleana (σ_1 *or* σ_2 *or* ... *or* σ_n) que diz que o conjunto de eventos pode causar a transição do estado q para o estado q' do autômato S_j é $\Sigma_{q,q'}^{S_j} = \{\sigma_1, \sigma_2, \sigma_n\}$ onde (q, q') é o par ordenado que originou tal transição e que os eventos $\sigma_1, \sigma_2, \ldots, \sigma_n$ estão relacionados às variáveis do CLP $\sigma_1, \sigma_2, \ldots, \sigma_n$, respectivamente.

A ação associada $xq \in X^{S_j}$ é baseada na suposição que:

1. $xq \in X^{S_j}$ é associado com $q \in Q^{S_j}$;
2. $\Phi_j(q) = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$;
3. Os eventos $\sigma_1, \sigma_2, \ldots, \sigma_n$ estão respectivamente associados às variáveis do CLP $\sigma 1ds_j, \sigma 2ds_j, \ldots, \sigma ndsj$ onde $\{\sigma 1ds_j, \sigma 2ds_j, \ldots, \sigma ndsj\} \subseteq P_j$.

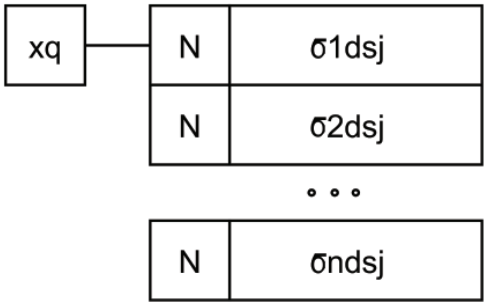

Figura 5.22: Ações associadas ao passo xq do SFC

Considere o supervisor S_1 apresentado na Figura 4.7. A função de desabilitação de eventos deste supervisor é $\phi_{S_1}(0) = \{31\}$, $\phi_{S_1}(1) = \{11\}$. Baseada nesta função, é definido $\Sigma_p^{S_1} = \phi_{S_1}(0) \cup \phi_{S_1}(1) = \{11, 31\}$ como o conjunto de eventos sujeitos à desabilitação pelo supervisor S_1. Conforme a etapa inicial, é feita a relação "evento x variável do CLP", onde $(11, E11ds1)$, $(31, E31ds1)$ e o resultado no conjunto de variáveis $P_{S_1} = \{E11ds1, E31ds1\}$. Baseado no autômato S_1 e conforme descrito na segunda etapa, define-se $\Sigma_{0,1}^{S_1} = \{12\}$ e $\Sigma_{1,0}^{S_1} = \{31\}$.

Na Figura 5.23, é apresentado o SFC S1 que foi obtido relacionando o "estado x passo" apresentado como $(0, x0)$ e $(1, x1)$.

No SFC S_1, Figura 5.23, o passo 13 (Step_013) representa o estado 0 do autômato S_1, a variável E31DS1 recebe o valor 1 e representa o sinal de desabilitação associado ao evento 31 desabilitado pelo supervisor S_1, a variável E11DS1 recebe o valor 0 e representa o evento 11 não desabilitado pelo supervisor S_1 e a transição 16 (Tran_016) representa o evento 12 do SFM.

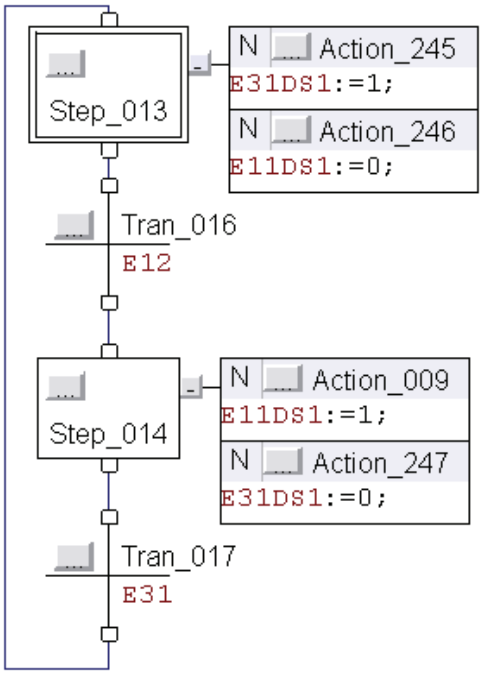

Figura 5.23: Lógica S_1

SFC's e FB's no conjunto $(g_i|i \in I)$

Esta seção apresenta o procedimento sistemático para a conversão de cada autômato em $\{G_i|i \in I\}$ no correspondente SFC que será implementado no CLP. Primeiramente, é definida a base de dados correspondentes às variáveis do CLP. Na segunda etapa, são apresentadas as propriedades para a conversão dos autômatos G_i nos autômatos H_i e na terceira etapa, os FB's dos SFC's são obtidos com base nos autômatos H_i.

Definição das variáveis do CLP

De acordo com Vieira (2007), é necessária a definição das variáveis que serão utilizadas nos SFCs e FBs do conjunto $g_i|i \in I$. Assim, uma variável booleana do CLP CED, Desabilitação de Evento Controlável, é utilizada para suspender o tratamento dos eventos controláveis, quando é ativada e permite priorizar o tratamento de eventos não controláveis em detrimento do tratamento de eventos controláveis.

A cada subsistema em $G_i|i \in I$ são associadas duas variáveis Booleanas, "$g_i evt''$ e "$g_i d''$, sendo que a variável "$g_i evt''$ sinaliza que foi tratado algum evento em Σ^{G_i}, quando ativada no FB g_i e a variável $g_i d$ do function block dg_i autoriza ou não o tratamento de eventos em Σ^{G_i}.

Uma variável booleana $\sigma \in E_i$ é associada a cada evento $\sigma \in \Sigma^{G_i}, i \in I$ e sinaliza o tratamento do evento $\sigma \in \Sigma^{G_i}$. Uma variável booleana $\sigma_d \in D_i$ ativa o sinal de desabilitação. Uma variável booleana $cmd\sigma \in C_i$ é o comando associado ao evento controlável $\sigma \in \Sigma_c^{G_i}$ e uma variável inteira $rsp\sigma \in R_i$ é associada a cada evento não-controlável $\sigma \in \Sigma_{uc}^{G_i}, i \in I$, sendo que o valor desta variável indica quantas ocorrências do evento estão pendentes

para tratamento se o evento não controlável $\sigma \in \Sigma_{uc}^{G_i}$ está pendente para tratamento ou não. Assim, são obtidos os conjuntos de variáveis $|E_i| = |\Sigma^{G_i}|, |D_i| = |C_i| = |\Sigma_c^{G_i}|$ e $|R_i| = |\Sigma_{uc}^{G_i}|$.

Conversão dos Autômatos G_i em Autômatos $H_i, i \in I$

O objetivo da conversão é obter um autômato H_i correspondente ao autômato G_i, tal que, cada estado do autômato H_i seja relacionado a um evento distinto em Σ^{G_i} e desta forma, não deve haver auto-laços em H_i para assegurar que, para toda transição do SFC g_i, o passo sucessor seja diferente do passo antecessor da transição em questão. De acordo com Vieira (2007), um autômato do sistema produto, $G_i = \left(\Sigma^{G_i}, Q^{G_i}, \delta^{G_i}, q0^{G_i}, Qm^{G_i}\right)$ é convertido no autômato $H_i = \left(\Sigma^{H_i}, Q^{H_i}, \delta^{H_i}, q0^{H_i}, Qm^{H_i}\right)$ onde: Σ^{H_i} - alfabeto de eventos, $\Sigma^{H_i} = \Sigma^{G_i}$; $q0^{H_i}$ - estado inicial; Q^{H_i} - conjunto de estados, com $Q^{H_i} \subseteq \left\{Q^{G_i} \times \Sigma^{G_i}\right\} \cup \left\{q0^{G_i}, \sigma_{dummy}\right\}$; δ^{H_i} - função de transição de estados, $\delta^{H_i} : Q^{H_i} \times \Sigma^{H_i} \rightarrow Q^{H_i}$; Qm^{H_i} - conjunto de estados marcados.

As propriedades a seguir sendo satisfeitas, realizam a conversão do autômato G_i, no autômato H_i.

1. $L\left(H_i\right) = L\left(G_i\right)$;
2. $\left[\forall \left(q1, q2 \in Q^{H_i}\right), \forall \left(\sigma_a, \sigma_b \in \Sigma^{H_i}\right)\right] \quad \delta^{H_i}\left(q1, \sigma_a\right) = \delta^{H_i}\left(q2, \sigma_b\right)$ *somente se* $\sigma_a = \sigma_b$;
3. $\left[\forall \left(q \in Q^{H_i}\right), \forall \left(\sigma \in \Sigma^{H_i}\right)\right] \qquad \delta^{H_i}\left(q, \sigma\right) \neq q0^{H_i}$;
4. $\left[\forall \left(q \in Q^{H_i}\right), \forall \left(\sigma \in \Sigma^{H_i}\right)\right] \qquad \delta^{H_i}\left(q, \sigma\right) \neq q$.

A propriedade (1) estabelece que a linguagem gerada pelo autômato H_i é exatamente a mesma que pode ser gerada pelo autômato G_i e desta forma os autômatos G_i e H_i representam o mesmo comportamento livre dos subsistemas. Esta propriedade também traduz a capacidade do autômato H_i não reconhecer tarefas completas. A propriedade (2) define que cada estado de H_i será alcançado através da ocorrência do mesmo evento. A propriedade (3) indica que um evento $\sigma \in \Sigma^{H_i}$ não conduzirá o autômato H_i ao seu estado inicial $q0^{H_i}$. A propriedade (4) indica que o autômato H_i não possui auto-laços.

O procedimento de conversão é baseado no procedimento proposto em [Carroll e Long, 1989] para converter um autômato de Mealy em um autômato de Moore, assim, o autômato H_i é obtido tomando a componente acessível do autômato, sendo:

1. $q0 = \left(q0^{G_i}, \sigma_{dummy}\right)$, tal que $\sigma_{dummy} \notin \Sigma$;
2. $\left[\forall \left(q \in Q^{G_i}\right), \forall \left(\sigma_1 \in \Sigma^{G_i}\right), \forall \left(\sigma_2 \in \Sigma^{G_i} \cup \{\sigma_{dummy}\}\right)\right] \quad \delta\left(\left(q1, \sigma_2\right), \sigma_1\right) = \left(\delta^{G_i}\left(q, \sigma_1\right), \sigma_1\right)$;
3. $Q_m = \emptyset$.

Obtenção dos SFCs g_i correspondentes aos Autômatos H_i

Esta etapa é baseada no procedimento apresentado em [Cassandras e Lafortune, 1999] para conversão dos Autômatos em Redes de Petri.

A obtenção do SFC $g_i = (X^{g_i}, T^{g_i}, x_0^{g_i}), i \in I$, correspondente aos módulos do sistema-produto implementados no CLP, são descritos:

1. Um passo $xq \in X^{g_i}$ é associado a cada estado $q \in Q^{H_i}$, de tal modo que $|X^{g_i}| = |Q^{H_i}|$;

2. $x0^{g_i}$ é o passo associado $q0^{H_i}$;

3. Uma transição $(xq, xq') \in T^{g_i}$ é associada a cada tripla ordenada (q, σ, q') de H_i, onde $\delta^{H_i}(q, \sigma) = q'$ e os passos $xq, xq' \in X^{g_i}$ são associados, respectivamente, aos estados $q, q' \in Q^{H_i}$. Esta associação tem de ser feita de modo que $|T^{g_i}|$ seja igual ao número da tripla ordenada.

Como a transição $(xq, xq') \in T^{g_i}, i \in I$ é associada ao evento $\sigma \in \Sigma^{G_i}$ por intermédio da tripla ordenada (q, σ, q'), as condições abaixo são relacionadas e o evento σ é associado a tal transição bem como às variáveis do CLP σ_d ou $rsp\sigma$.

1. $NOT\sigma_d \; AND \; NOTCED$

2. $rsp\sigma > 0$

A definição (1) é utilizada quando o evento associado é controlável com $\sigma \in \Sigma_c^{G_i}$ e a definição (2) é associada a um evento não controlável $\sigma \in \Sigma_{uc}^{G_i}$. Nenhuma ação é associada ao passo inicial $x0^{G_i}$. Na Figura 5.24, é apresentada na letra 'a' uma passo com uma ação associada a um evento controlável e na letra 'b' uma passo com uma ação associada a um evento não controlável.

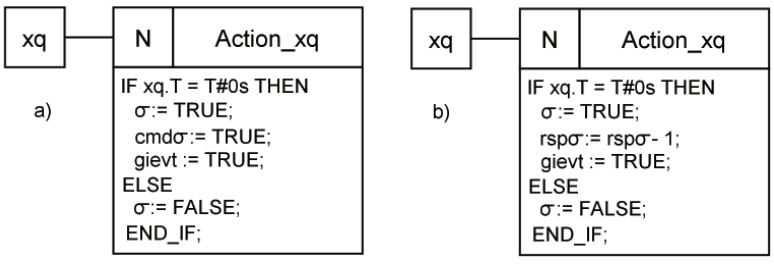

Figura 5.24: Ações associadas ao passo xq

Exemplo: Considere a esteira $C1$ apresentada na Figura 4.5 autômato G_1. Os $\Sigma_c^{G1} = \{11\}$ e $\Sigma_{uc}^{G1} = \{12\}$. De acordo com a etapa inicial, as variáveis $g1evt$ e $g1d$ devem ser associadas ao subsistema $G1$, sendo a associação com $(G1, g1evt)$ e $(G1, g1d)$ e é necessário relacionar o "evento x variável do CLP" com $(11, E11)$, $(11, CMDE11)$, $(11, E11d)$, $(12, E12)$,

$(12, RSPE12)$, tal que $E1 = \{E11, E12\}$, $D1 = \{E11D\}$, $C1 = \{CMDE11\}$ *e* $R1 = \{RSPE12\}$. O autômato H_1 é obtido na segunda etapa, sendo $Q^{G1} = \{0, 1\}$ e $\Sigma^{G1} = \{11, 12\}$, tem-se que $Q^{H1} \subseteq Q^{G1} \times \Sigma^{G1} = \{(0, 11), (0, 12), (1, 11), (1, 12)\} \cup \{0, \sigma_{dummy}\}$. A função de transição de estados $\delta^{H_1} : Q^{H_1} \times \Sigma^{H_1} \to Q^{H_1}$ é obtida por:

$\delta^{H_1}((0, \sigma_{dummy}), 11) = \left(\delta^{G_1}(0, 11), 11\right) = (1, 11)$;
$\delta^{H_1}((0, \sigma_{dummy}), 12) = \left(\delta^{G_1}(0, 12), 12\right) =$ a função não é definida;
$\delta^{H_1}((0, 11), 11) = \left(\delta^{G_1}(0, 11), 11\right) = (1, 11)$;
$\delta^{H_1}((0, 11), 12) = \left(\delta^{G_1}(0, 11), 12\right) = (1, 12)$;
$\delta^{H_1}((0, 12), 11) = \left(\delta^{G_1}(0, 11), 11\right) = (1, 11)$;
$\delta^{H_1}((0, 12), 12) = \left(\delta^{G_1}(0, 12), 11\right) =$ a função não é definida;
$\delta^{H_1}((1, 11), 11) = \left(\delta^{G_1}(1, 11), 11\right) =$ a função não é definida;
$\delta^{H_1}((1, 11), 12) = \left(\delta^{G_1}(1, 12), 12\right) = (0, 12)$;
$\delta^{H_1}((1, 12), 11) = \left(\delta^{G_1}(1, 12), 11\right) = (0, 11)$;
$\delta^{H_1}((1, 12), 12) = \left(\delta^{G_1}(1, 12), 12\right) = (0, 12)$.

A Figura 5.25 apresenta o autômato H_1.

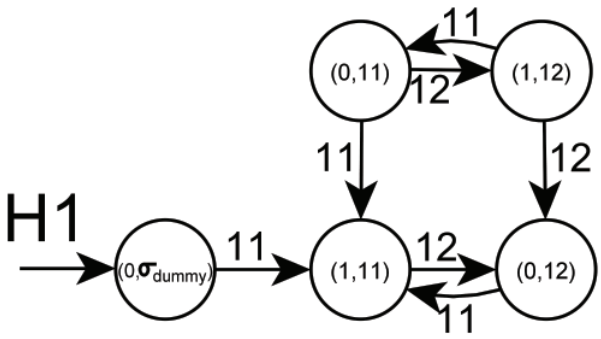

Figura 5.25: Autômato H1

Na Figura 5.26 é apresentada a componente acessível do autômato H_1.

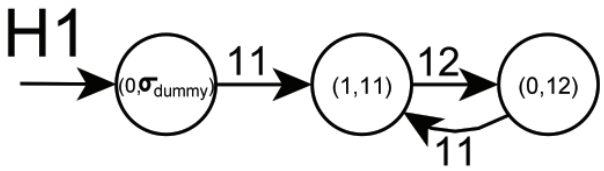

Figura 5.26: Autômato H1 - Trim

Na etapa 3, deve-se realizar a associação dos estados do autômato H_1 com os passos do SFC g_1 . Assim, o resultado é $((0, \sigma_{dummy}), x0), ((1, 11), x1)$ *e* $((0, 12), x2)$. As triplas ordenadas associadas ao autômato H_1 são:

$((0, \sigma_{dummy}), 11, (1, 11))$;
$((1, 11), 12, (0, 12))$;
$((0, 12), 11, (1, 11))$.

As triplas ordenadas associadas ao SFC g_1 são: $(x0, 11, x1), (x1, 12, x2)\, e\, (x2, 11, x1)$. As transições associadas aos eventos são: $(11, NOTE11D \;\; AND \;\; NOTCED), (12, RSPE12 > 0)$. O SFC g_1 resultante é apresentado na Figura 5.27.

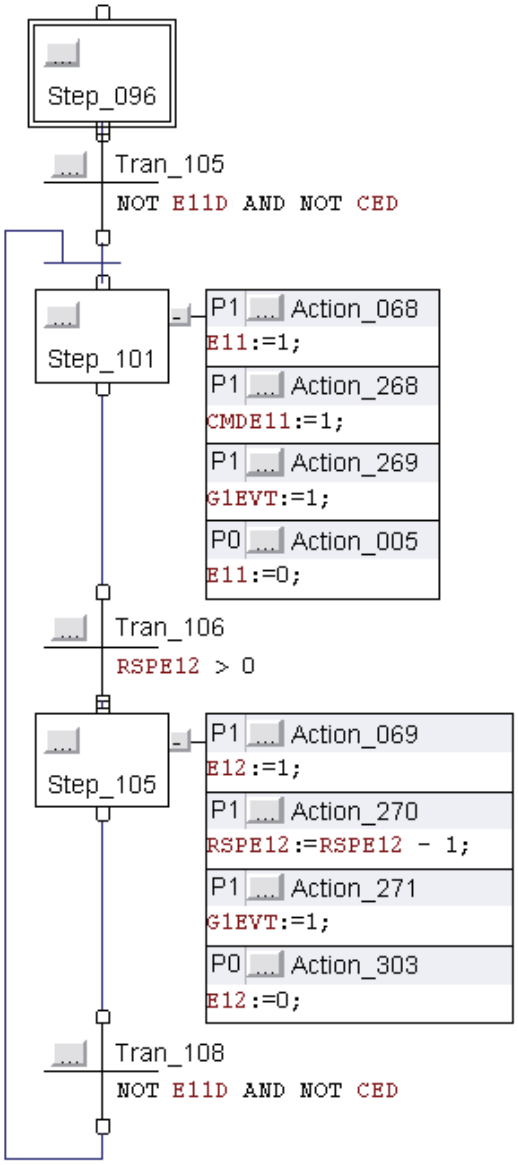

Figura 5.27: SFC g1

Funções lógicas e FB's no conjunto $\{dg_i | i \in I\}$

O objetivo da lógica $dg_i, i \in I$ é evitar o tratamento simultâneo de eventos, em subsistemas que compartilham uma mesma célula de controle.

Dado o subsistema do sistema produto, $\{g_i | i \in I\}$. A representação $G_k | k \in K_i, K_i \subseteq I$ é o conjunto formado por todos os subsistemas que compartilham alguma célula de controle com o subsistema g_i considerado e K_i é o conjunto de índices que identificam estes subsistemas.

O tratamento de eventos em Σ_{g_i} deve ser suspenso sempre que algum evento em qualquer $\Sigma_{G_k}, k \in K_i$, tiver sido tratado. O tratamento de evento em $\Sigma_{G_w}|w \notin K_i$ não requer a suspensão do tratamento de eventos em Σ_{g_i}, ou seja, o tratamento de eventos de subsistemas que não compartilham qualquer célula de controle com g_i não requer a suspensão.

A suspensão no tratamento de eventos deve ser mantida até a ocorrência da atualização do estado ativo de todos os supervisores que compartilham alguma célula de controle com o subsistema que teve o evento tratado.

A lógica do FB $dgi, i \in I$, estabelece o valor lógico VERDADEIRO à variável gid se pelo menos uma variável em $gkevt|k \in K_i$ apresentar o valor lógico VERDADEIRO e caso contrário, FALSO.

Como exemplo, considere o subsistema G1 da Figura 4.5, o conjunto de subsistemas que compartilham alguma célula de controle com o subsistema $G1$ é $\{G1, G5\}$, portanto, $K2 = \{1, 5\}$. O FB para dg1 assume a relação "subsistema x variável do CLP", então, $(G1, g1evt), (G5, g5evt)$, então, o FB $dg1$ é apresentado na Figura 5.28.

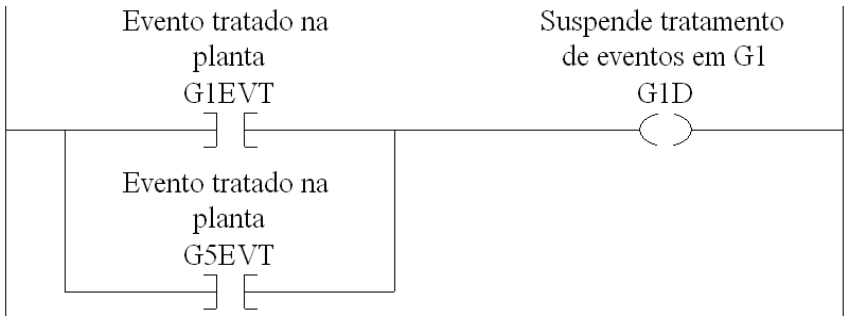

Figura 5.28: FB dg1

Procedimentos operacionais no conjunto $\{o_\sigma|\sigma \in \Sigma\}$

A lógica de controle que dirige a execução das atividades pelo sistema a ser controlado é implementado por cada FB no conjunto $\{o_\sigma|\sigma \in \Sigma\}$.

Algumas dicas, conforme [Vieira, 2007], podem ser seguidas para a obtenção do SFC para o procedimento operacional:

- Se o procedimento operacional está associado a um evento controlável, e portanto a um comando, tal comando deve ser empregado para compor as expressões Booleanas que representam a condição de transição associada a toda transição cujo passo predecessor é o passo inicial do SFC. Além disto, este comando deve ser desativado no passo sucessor à transição cuja condição de transição considera o referido comando. A desativação do comando confirma que o mesmo já foi processado pelo procedimento operacional correspondente;

- Se o procedimento operacional é o responsável por identificar a ocorrência de um determinado evento não controlável, então, na ativação do passo do SFC em que é identificada a ocorrência de tal evento, a resposta associada ao evento em questão deve ser incrementada de uma unidade;
- A evolução do SFC que implementa o procedimento operacional resulta, em algum momento, na ativação do passo inicial.

O conjunto de evento controlável é $\Sigma_c = \{11\}$ e o conjunto de evento não controlável é $\Sigma_{uc} = \{12\}$. O comando que corresponde à variável do CLP e está relacionado com o evento controlável é $C = \{cmdE11\}$ e com o evento não controlável é $R = \{RSPE12\}$. O procedimento operacional associado à tais eventos é $O = \{SO\_ESTEIRA\_C1\}$ e o SFC correspondente é apresentado na Figura 5.29.

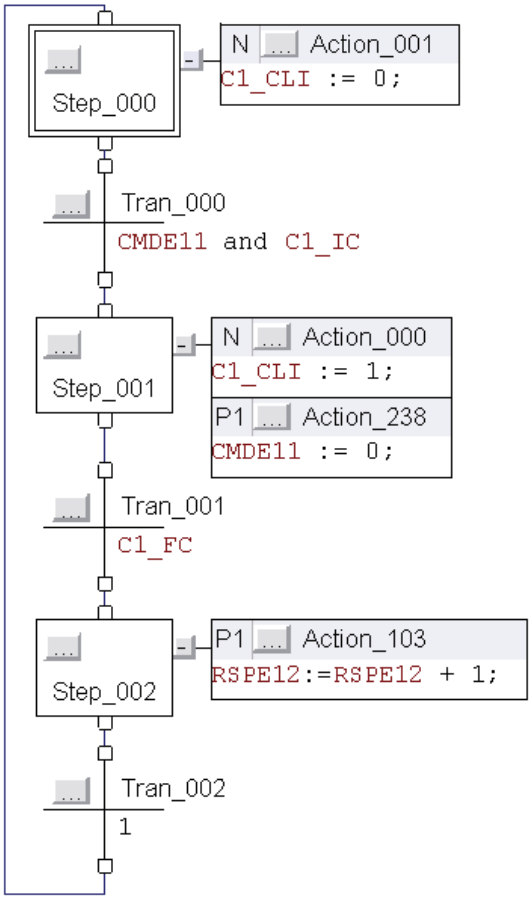

Figura 5.29: SFC Esteira C1

Os sensores $C1\_IC$ e $C1\_FC$ detectam a presença de peça no ínicio e fim e são condições para ligar e desligar a esteira, respectivamente. O motor da esteira é ligado pelo tag $C1\_CLI$ que é uma saída do cartão de saída digital do CLP, desta forma, a peça é transportada do ínicio até o fim.

As lógicas de programação desta metodologia de implementação estão conectadas na rotina principal, assim a estrutura do programa é modular.

No apêndice C, como exemplo, são apresentadas as lógicas de programação do supervisor S_{1red}, autômato da esteira C1 e do Robô, sequência operacional da esteira C1 e do Robô e parte da lógica principal deste método de implementação em CLP aplicado ao sistema flexível de manufatura.

5.3.4 Implementação IV (Lima II, 2002)

Lima II, (2002) propôs uma metodologia de implementação em CLP que utiliza uma arquitetura de controle dividida em 4 níveis. Esta arquitetura é inspirada na arquitetura de Queiroz e Cury (2002) e pode ser vista na Figura 5.30.

Figura 5.30: Arquitetura de Controle (Lima II, 2002)

O nível **Modelo Supervisionado** observa os eventos da planta supervisionada que é a Rede de Petri do sistema adicionado dos lugares do supervisor, atualiza sua marcação e desabilita transições proibidas de acordo com essa marcação.

O nível **Executor** seleciona uma das transições habilitadas e é responsável por comandar o início das sequências operacionais.

O nível **Sequências Operacionais** comanda os sinais de saída de modo a controlar as sub-operações do sistema e ao final de uma operação envia uma informação ao nível do modelo supervisionado por meio de uma transição não controlável e a marcação da rede é atualizada.

O nível **Sistema Real** representa o sistema físico, seus sensores e atuadores.

Na estrutura de Queiroz e Cury (2002) utilizando Autômatos, existem duas estruturas distintas: o modelo da planta e o supervisor. Na estrutura de Lima II (2002) utilizando Redes de Petri, o supervisor é incorporado no próprio modelo do sistema (planta). O

modelo da planta em [Queiroz e Cury, 2002] é representado pelo nível Sistema-Produto e em Lima II (2002) o modelo da planta faz parte do nível Modelo Supervisionado.

Os níveis a serem implementados no CLP são o modelo supervisionado, o executor e as sequências operacionais. Esses níveis são implementados em CLP utilizando a linguagem Ladder Diagram.

Modelo Supervisionado

No nível Modelo Supervisionado, as transições controláveis são habilitadas ou desabilitadas de acordo com a marcação da rede e são transmitidas ao executor. Já as transições não controláveis são observadas pelo Modelo Supervisionado e se algum disparo ocorrer, a marcação da rede é atualizada. Neste nível, é implementada a rotina JOG.

A rotina JOG observa os eventos ocorridos na planta e atualiza a marcação da rede, por $\mathtt{x}' = \mathtt{x} + \mathtt{D} \times \mathtt{u}$ e também determina as transições desabilitadas pela Rede de Petri por meio de $\mathtt{x} < \mathtt{D}^{-} \times \mathtt{u}$, em que $\mathtt{u}$ é um vetor de dimensão m, com $\mathtt{u}_j = 1$ para $i = j$ e $\mathtt{u}_j = 0$ para $i \neq j$.

A rotina é constituída por duas partes, sendo que a primeira atualiza a marcação da rede e a segunda verifica as transições controláveis desabilitadas. Na primeira, existe uma linha no diagrama ladder para cada transição da Rede de Petri e para o caso da transição j seja disparada ($Qj = 1$), são atualizadas as marcas de todos os lugares conectados à transição j, somando ou subtraindo os pesos dos arcos.

Exemplo: Na Figura 5.31, é mostrada uma parte de uma rede com três transições (T0, T1, T2), sendo T0 e T2 transições controláveis e T1 transição não controlável e quatro lugares (P0, P1, P2, P3). O disparo de T0 retira uma ficha do lugar P3 e coloca uma ficha em P0. O disparo de T1 retira uma ficha do lugar P0 e coloca uma ficha em P1. O disparo de T2 retira uma ficha de P1 e coloca em P2 e P3. A variável Q0 representa o disparo de T0. A variável Q1 representa o disparo de T1 e Q2 o disparo de T2. M0, M1, M2 e M3 armazenam o número de fichas nos lugares P0, P1, P2 e P3 respectivamente.

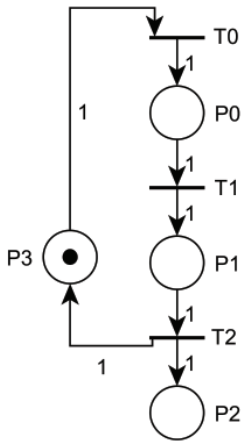

Figura 5.31: Exemplo RP

Na Figura 5.32, é mostrado o trecho do programa em ladder para T2. Quando a variável Q2 recebe o nível lógico 1 indicando o disparo da transição T2, ocorre a subtração de uma unidade na variável M1 que indica a retirada da ficha no lugar P1, também ocorre a adição de uma unidade nas variáveis M2 e M3, indicando que os lugares P2 e P3 receberam uma ficha cada.

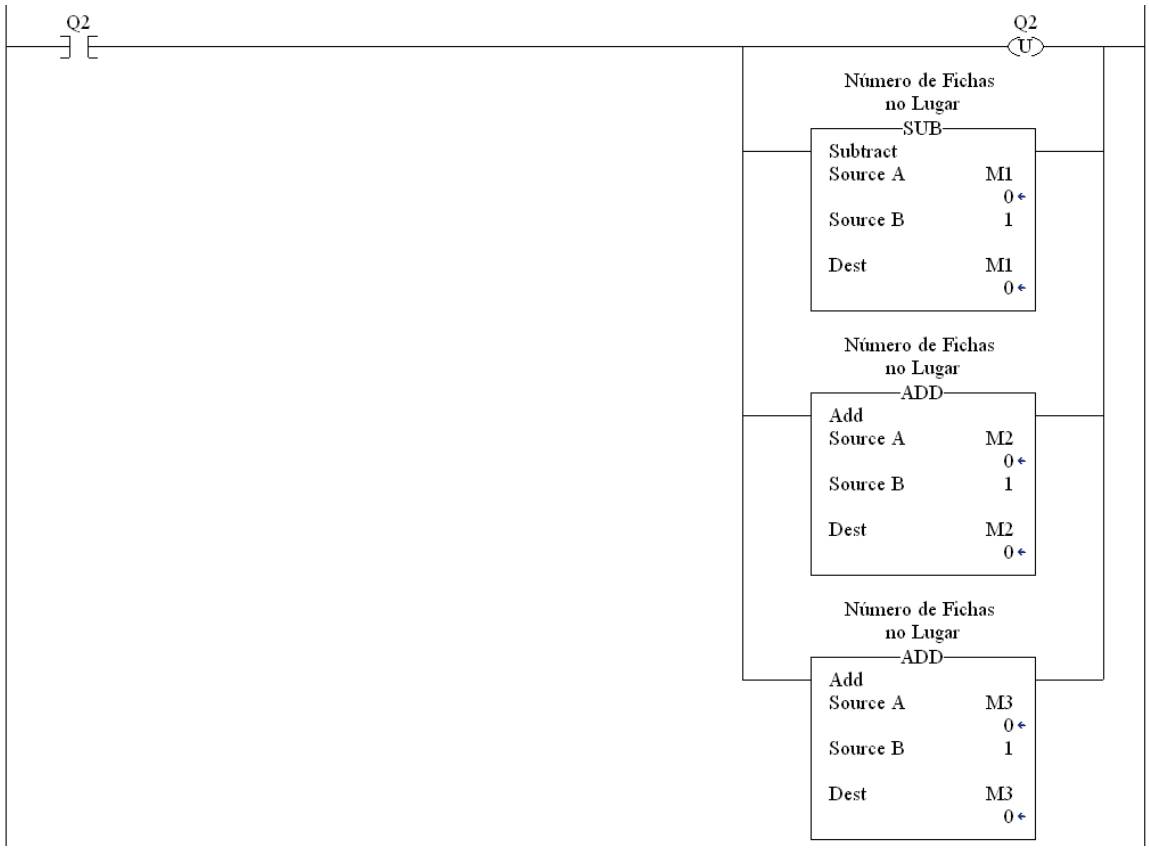

Figura 5.32: Lógica Exemplo RP

Na segunda parte, são determinadas as transições controláveis desabilitadas, e para isto é criada uma linha que define o valor do elemento correspondente à transição no vetor D no programa Ladder para cada transição controlável. A transição está habilitada se $\mathtt{x} \geq \mathtt{D}^- \times \mathtt{u}$, ou seja, se a marca de cada lugar que precede o disparo da transição é maior que o peso do arco que o liga à mesma.

Exemplo: Para a Figura 5.33, é mostrado o trecho do programa em Ladder Diagram para a inibição da transição T0 e T2. Quando o número de fichas for maior ou igual a um na variável M3, a variável DES_Q0 recebe o valor 1 indicando que a transição T0 não foi desabilitada, o que equivale dizer que o evento 11 não foi desabilitado. Quando o número de fichas for maior ou igual a um na variável M1, a variável DES_Q2 recebe o valor 1 indicando que a transição T2 não foi desabilitada, o que equivale dizer que o evento 31 não foi desabilitado.

Figura 5.33: Lógica de Habilitação da transição

Vale observar que é utilizada a matriz $\mathtt{D} = \mathtt{D}^+ - \mathtt{D}^-$ na primeira parte da rotina para definir os valores dos arcos e na segunda parte utiliza-se a matriz $\mathtt{D}^-$, o que permite implementar a rotina JOG para redes que possuam auto-laços.

Executor

O nível Executor ordena o início da operação cuja transição não esteja desabilitada pelo modelo supervisionado. Neste nível é implementada a rotina EXE.

A rotina EXE verifica as transições controláveis habilitadas, ou transições controláveis que não estão desabilitadas ($d_c(T) = falso$) e faz $q_c = verdadeiro$ para uma delas e salta para o final da rotina. A ordem de prioridade foi a própria ordem de numeração das transições. Apenas uma transição controlável é disparada por vez para evitar problemas que poderiam surgir, quando duas transições em conflito estivessem habilitadas. Caso duas transições controláveis pudessem disparar simultaneamente, a rotina prioriza o disparo de uma delas, pois o tempo de scan do CLP é bem menor que as constantes de tempo do sistema, assim, o disparo de duas transições em sequência tem o mesmo efeito de um disparo simultâneo.

Exemplo: Na Figura 5.34 é mostrado o trecho do programa em Ladder Diagram para os comandos de disparo das transições T0 e T2 da rede da Figura 5.31. A variável DES_Q0 com o valor 1 indica que a transição T0 não foi desabilitada e ativa a variável QC0 que dispara a transição controlável T0.

Sequências Operacionais

O nível Sequências Operacionais comanda os sinais de saída de modo a controlar as sub-operações do sistema e ao final de uma operação envia uma informação ao nível do modelo supervisionado por intermédio de uma transição não controlável e a marcação da rede é atualizada. Neste nível, é implementada a rotina SEQOP.

Na rotina SEQOP, o vetor de disparo de transições $\mathtt{u}$ é $\mathtt{u}(k) = q_c(k) + q_{uc}(k)$, sendo

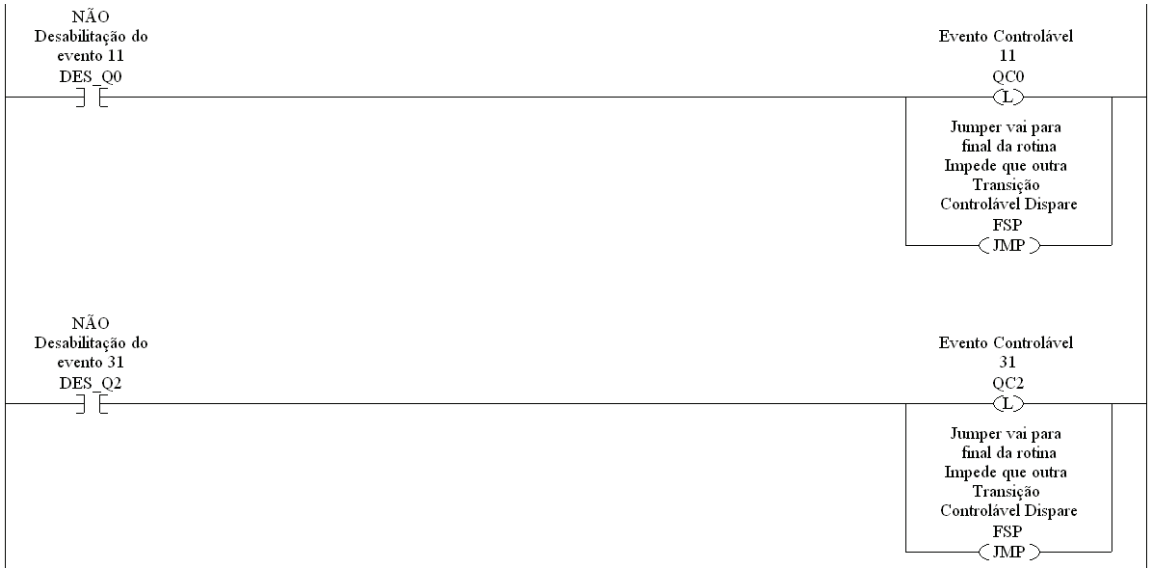

Figura 5.34: Lógica Disparo das Transições Controláveis

q_c o vetor de comando dos disparos das transições controláveis, gerado na rotina EXE a partir do vetor de desabilitações fornecido pelo modelo supervisionado e q_{uc} o vetor que indica a ocorrência dos eventos não controláveis lidos pelo CLP e gerado pela rotina das sequências operacionais. A representação de u no programa do CLP é Q que é definido como um conjunto de variáveis booleanas com m posições, sendo m o número de transições da rede, indicando se a transição foi disparada ou não. A representação de q_c no programa do CLP é QC que é um conjunto de variáveis booleanas com m posições. Os disparos das transições não controláveis (q_{uc}) são somados diretamente em Q. A representação de d_c no programa do CLP é DES que é um conjunto de variáveis booleanas, indicando as desabilitações e só faz sentido para as transições controláveis. No CLP, a representação da marcação da RP é M que é o conjunto de variáveis inteiras (DINT) com n posições, sendo n o número de lugares da rede.

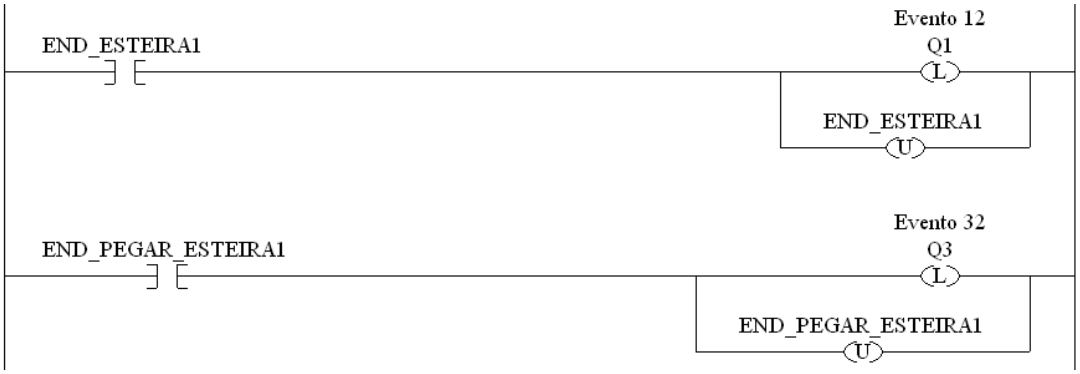

Figura 5.35: Lógica Disparo das Transições Não Controláveis

Por exemplo, a rotina da sequência operacional para a esteira C1 utiliza a linguagem de programação em SFC, Figura 5.36. A esteira C1 possui um motor, um sensor fotoelétrico

localizado no início da esteira indicando a presença de peça no início e outro sensor fotoelétrico localizado no final da esteira indicando a presença de peça no final da esteira. A peça é detectada pelo sensor fotoelétrico por meio do tag $C1\_IC$ (entrada Local:6:I.Data.0), logo o motor da esteira é energizado por meio do tag $C1\_CLI$ (saída Local:3:O.Data.0) e quando o sensor fotoelétrico por meio do tag $C1\_FC$ (entrada Local:6:I.Data.1) indicar a presença da peça na posição final da esteira, o motor é desenergizado.

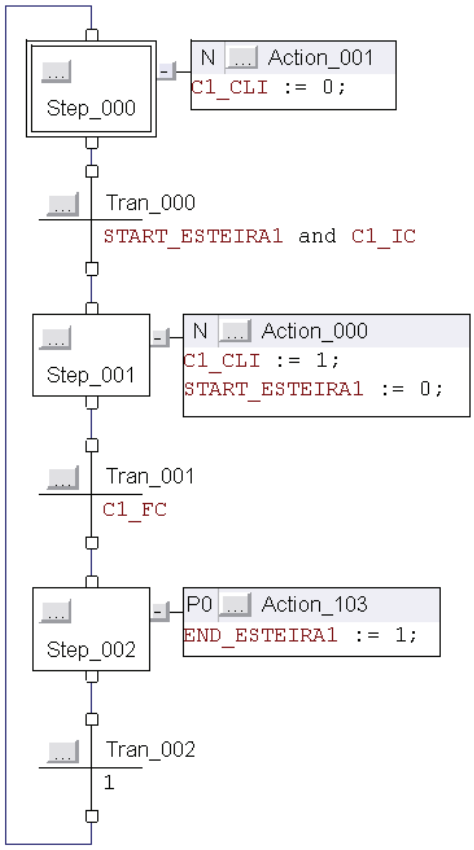

Figura 5.36: Lógica Sequência Operacional C1

Considerações sobre os Vetores de Disparo

Uma transição controlável é disparada no jogador de marcas da Rede de Petri (ativando a posição correspondente em Q), quando essa transição provocar a mudança de estado de alguma sequência operacional.

A rotina SEQOP deve anular o disparo de transições QC, mesmo que algum comando de disparo não possa ser realizado.

Exemplo: Na Figura 5.37, é mostrado o trecho do programa em Ladder Diagram que desativa em QC a posição relativa à transição controlável T0 e T2 do exemplo 5.31.

As transições controláveis são disparadas no jogador, quando são disparadas nas sequências operacionais, porém, quando estas transições não fazem parte de nenhuma sequência operacional, é necessário dispará-las no jogador (Q) sempre que o executor

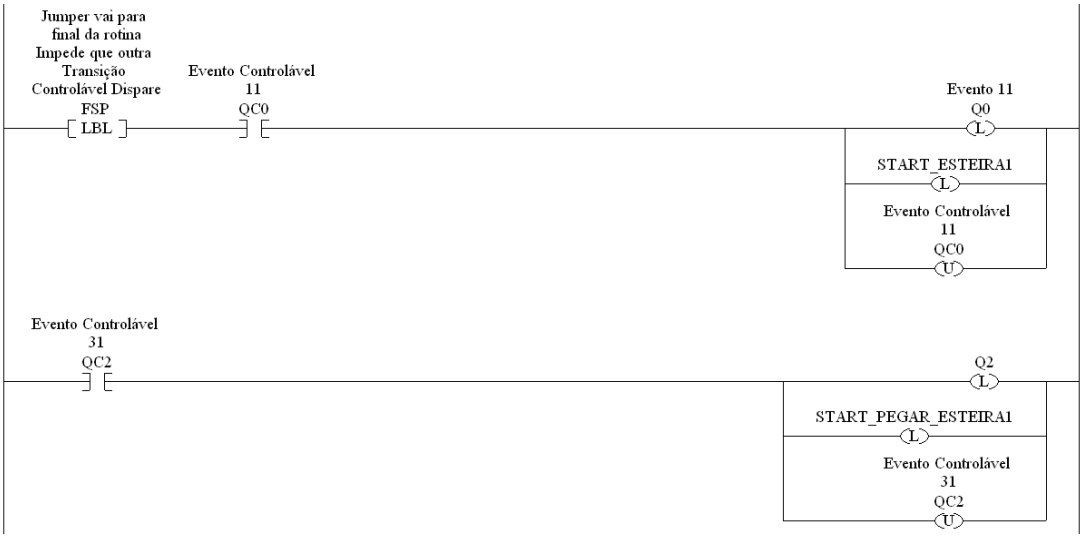

Figura 5.37: Desativação em QC

comandar (QC) e esta condição pode ser implementada na mesma linha em que QC é anulado.

As lógicas de programação desta metodologia de implementação estão em uma única rotina principal, assim a estrutura do programa não é modular.

No apêndice D, como exemplo, são apresentadas as lógicas de programação do supervisor S_{1red}, autômato da esteira C1 e do Robô, sequência operacional da esteira C1 e do Robô e parte da lógica principal deste método de implementação em CLP aplicado ao sistema flexível de manufatura.

5.4 Resultados

Alguns critérios tratados em [Ferigollo et al., 2011] e outros observados são utilizados para a análise comparativa das metodologias de implementação em CLP. Os resultados são sintetizados na Tabela 5.2 e são discutidos a seguir.

A primeira diferença a ser destacada consiste no método de projeto dos supervisores que nas metodologias de implementação I, II e III utiliza a Teoria de Controle Supervisório e na metodologia IV utiliza os Invariantes de Lugar. Outra diferença a ser destacada é a modelagem do sistema baseada em Autômatos nas metodologias de implementação I, II e III e as Redes de Petri na metodologia IV. Embora existam diferenças entre as teorias de controle e modelagem, é por meio dessas teorias que os métodos formais são implementados em CLP e os aspectos do sistema podem ser verificados como, por exemplo, bloqueio, alcançabilidade de estados, entre outros.

As linguagens de programação em CLP das metodologias estão de acordo com a norma IEC 61131-3, desta forma, as metodologias podem ser implementadas em CLP's de diversos fabricantes. As metodologias I, II e IV utilizam a linguagem de programação Ladder

Tabela 5.2: Comparações das Metodologias de Implementação em CLP

CRITÉRIOS	I	II	III	IV
Teoria de controle do sistema	TCS	TCS	TCS	IL
Modelagem do sistema	Autômatos	Autômatos	Autômatos	Redes de Petri
Linguagem de programação	Ladder	Ladder	SFC	Ladder
Tempo de produção 1A (s)	233	234	228	232
Tempo de produção 1B (s)	274	274	277	281
Tempo médio de scan (s)	0.00193	0.00218	0.00443	0.00195
Memória utilizada ($Bytes$)	241.272	260.776	384.116	245.236
Estrutura do programa	Principal	Modular	Modular	Principal
Modo de operação do sistema	1	1	6	1
N^o eventos/scan	1	Vários	Vários	1
Lógica para a escolha de eventos	Não	Sim	Não	Sim

Diagram para a implementação em CLP que é uma linguagem amplamente conhecida entre os programadores e no meio industrial. A metodologia III utiliza a linguagem de programação SFC que é de alto nível para a implementação em CLP e esta linguagem facilita o modo de visualização da implementação dos modelos obtidos a partir da TCS.

Os tempos de produção de um produto A e um produto B foram considerados. A metodologia III obteve o menor tempo, 228 segundos, para a produção de um produto A e as metodologias I e II obtiveram o menor tempo, 274 segundos, para a produção de um produto B. De modo geral, os tempos de produção de um produto A entre as metodologias I, II, III e IV não apresentaram diferenças tão significativas, o mesmo ocorre para os tempos de produção de um produto B.

O tempo médio de scan no programa do CLP das metodologias de implementação I, II e IV não apresenta diferença relevante e o tempo médio de scan da metodologia III é maior devido ao tempo necessário para o controlador executar um número maior de instruções do programa obtidos conforme o procedimento para conversão dos autômatos nas lógicas de programação utilizando a linguagem SFC. De modo geral, o tempo médio de scan do programa do CLP na ordem de microsegundos não é tão significativa e não é um fator que limita a operação do sistema, pois os tempos de execução dos sistemas mecânicos do SFM didático são na ordem de segundos.

A memória utilizada no programa do CLP das metodologias de implementação I, II e IV não apresenta diferença significativa e a memória utilizada no programa do CLP da metodologia III é maior devido à quantidade e aos tipos de dados obtidos, conforme o procedimento para conversão dos autômatos nas lógicas de programação utilizando a linguagem SFC. Existe um procedimento sistemático que permite representar um SFC em Ladder Diagram, com isto a alocação de memória é reduzida.

A estrutura do programa das metodologias de implementação I e IV é constituída por uma rotina principal contendo todas as lógicas de programação incluídas e a estrutura do programa das metodologias II e III possui rotinas separadas e conectadas em uma rotina

principal. A estrutura do programa modular possui uma melhor organização das lógicas de programação, maior facilidade de interpretação e modificação.

O modo de operação do sistema das metodologias de implementação I, II e IV é único, ou seja, supervisionado e o modo de operação do sistema da metodologia III possui seis modos: a inicialização do software, inicialização física do sistema, stand-by, operação manual, supervisionada e emergência, sendo assim, esta metodologia facilita a operação do sistema e tem a vantagem de permitir ao operador de conduzir o sistema em situações que não foram previstas nas metodologias I, II e IV.

O número de eventos por scan tratados nas metodologias de implementação I e IV é somente um evento e na metodologia II e III são tratados vários eventos. O programa do CLP que trata um evento por scan gasta "n" ciclos de scan para tratar "n" eventos, desta forma, em sistemas complexos, atrasos de comunicação entre o CLP e o sistema a ser controlado podem ocorrer. O programa do CLP que trata vários eventos por scan gasta um ciclo para tratar "n" eventos, minimizando atrasos de comunicação entre o CLP e o sistema a ser controlado.

A lógica para escolha de eventos é implementada nas metodologias II e IV por meio da escolha aleatória do evento a ser desabilitado. As metodologias I e III não possuem esta lógica, no entanto, a escolha é pela ordem de programação dos eventos que determina o primeiro a ser desabilitado.

Os resultados apresentados se referem às comparações das quatro metodologias implementadas no controlador lógico programável da Teoria de Controle Supervisório neste capítulo. Este capítulo também mostra que as quatro metodologias implementadas foram bem sucedidas. Desta forma, a planta didática do sistema flexível de manufatura é automatizada. O funcionamento do SFM didático e a facilidade de interpretação dos programas do CLP indicam a viabilidade das metodologias utilizadas.

Capítulo 6

Conclusão

Nesta dissertação, são estudadas e avaliadas quatro metodologias de implementação em controlador lógico programável (CLP) dos supervisores responsáveis pela operação de um SFM didático obtidos pela Teoria do Controle Supervisório, baseada nos Autômatos e nas Redes de Petri via Invariantes de Lugar.

Os formalismos matemáticos, bem como as teorias de controle para os Sistemas a Eventos Discretos são definidos. A Teoria do Controle Supervisório (TCS), baseada nos Autômatos e nas Redes de Petri via Invariantes de Lugar, tem como objetivo garantir um funcionamento seguro dos sistemas e subsistemas do SFM didático pela coordenação dos equipamentos. A solução utilizada para a TCS, baseada nos Autômatos, proporcionou a aplicação de três entre as quatro metodologias de implementação em CLP. Já a solução da TCS, baseada nas Redes de Petri via Invariantes de Lugar, foi obtida neste trabalho e permitiu a aplicação de uma metodologia de implementação em CLP. No total, quatro metodologias foram implementadas com sucesso no CLP modelo 1769-L32E CompactLogix da marca Allen-Bradley Rockwell Automation e aplicadas na automação de um sistema flexível de manufatura didático construído no Laboratório de Análise e Controle de Sistemas a Eventos Discretos (LACSED) da UFMG. O funcionamento do SFM didático e a facilidade de interpretação dos programas do CLP indicam a viabilidade das metodologias utilizadas na automação de processos industriais.

A partir dos critérios utilizados, foi apresentada uma análise comparativa entre as quatro metodologias que não tem como objetivo indicar a melhor metodologia, mas estabelecer as vantagens e desvantagens.

É importante que hajam métodos de implementação em CLP para as diferentes soluções de controle, a TCS baseada nos Autômatos e nas Redes de Petri via Invariantes de Lugar. Desta forma, o projetista tem a liberdade de escolher qual a metodologia a ser utilizada no sistema de controle. Uma vez escolhida a utilização da TCS baseada nos Autômatos, há diversas opções e, neste trabalho, três delas foram estudadas. Se o projetista for mais familiar com a linguagem de programação SFC que é uma linguagem de alto nível, a metodologia III é a indicada, pois tem a vantagem de ser modular, trata vários eventos por ciclo de atualização do CLP e implementa diversos modos de operação

do sistema. Caso o projetista tenha preferência pela linguagem Ladder Diagram, a metodologia II tem a vantagem sobre a metodologia I, pois é modular, trata vários eventos por ciclo de atualização do CLP e possui a lógica para escolha de eventos. Uma vez escolhida a utilização da TCS baseada nas Redes de Petri via Invariantes de Lugar, a metodologia IV utiliza a linguagem de programação Ladder Diagram e implementa a lógica para escolha de eventos.

6.1 Contribuições

As contribuições deste trabalho são:

- Revisão dos principais conceitos do controle de Sistemas a Eventos Discretos permitindo a comparação entre os métodos formais;
- Automação da planta didática representativa de um Sistema Flexível de Manufatura;
- Modelagem do comportamento dos subsistemas da planta utilizando o método formal das Redes de Petri;
- Estabelecimento das restrições de segurança da planta modelada como uma Rede Petri;
- Obtenção dos supervisores para controle da planta modelada como uma Rede de Petri;
- Implementação da Arquitetura de Controle em Controlador Lógico Programável (CLP) conforme Queiroz e Cury (2002);
- Implementação da Arquitetura de Controle em CLP conforme Leal et al. (2009);
- Implementação da Arquitetura de Controle em CLP conforme Vieira (2007);
- Implementação da Arquitetura de Controle em CLP conforme Lima II (2002);
- Analise e comparação das quatro implementações em CLP;
- Aplicação dos métodos formais para controle de Sistemas a Eventos Discretos;
- Disseminação do uso de metodologias de implementação em Controlador Lógico Programável (CLP) na automação dos sistemas de produção.

6.2 Sugestões para Trabalhos Futuros

Como sugestões para continuidade do trabalho, têm-se:

- Aplicar outras teorias de controle para controlar o sistema flexível de manufatura didático;
- Utilizar outras metodologias de implementação em CLP;
- Implementar outros modos de operação do sistema nas metodologias I, II e IV;
- Implementar um algoritmo capaz de otimizar a escolha de eventos para vários eventos controláveis.

Apêndice A

Implementação I

Para elucidar, neste apêndice é apresentada a metodologia de implementação I [Queiroz e Cury, 2002] em CLP utilizada na automação do sistema flexível de manufatura didático que contém a estrutura do programa no CLP, a lógica de programação principal e a lógica que envolve a esteira C1 e o robô do SFM didático. É apresentada a lógica do autômato supervisor S_{1red} (nível Supervisores Modulares), as lógicas dos autômatos esteira C1 e robô (nível Sistema-Produto) e as instruções que fazem parte da lógica principal desta metodologia. As lógicas das sequências operacionais esteira C1 e robô (nível Sequência Operacional) e a lógica de programação dos movimentos do robô são também apresentadas.

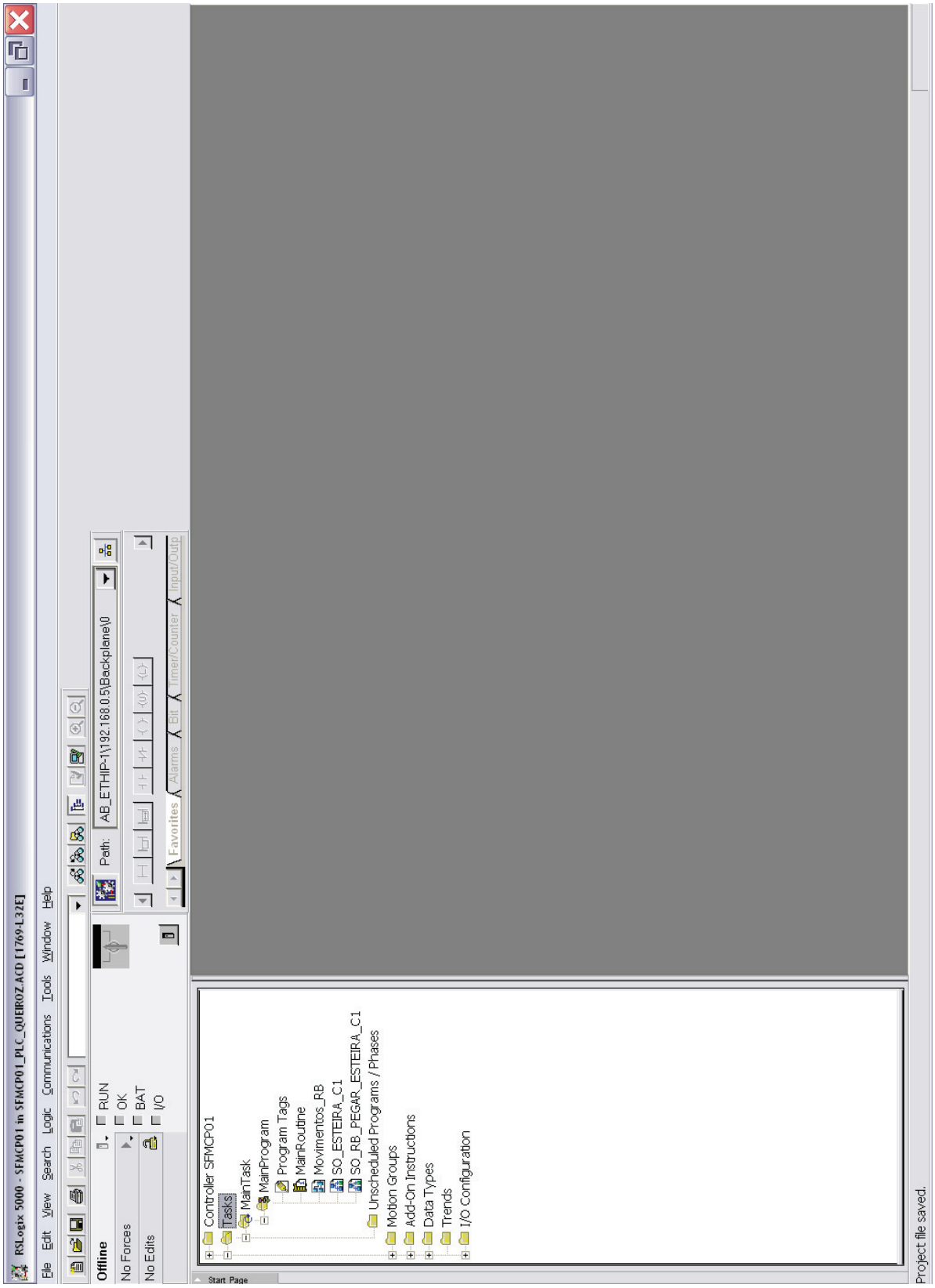

Figura A.1: Estrutura Programa CLP

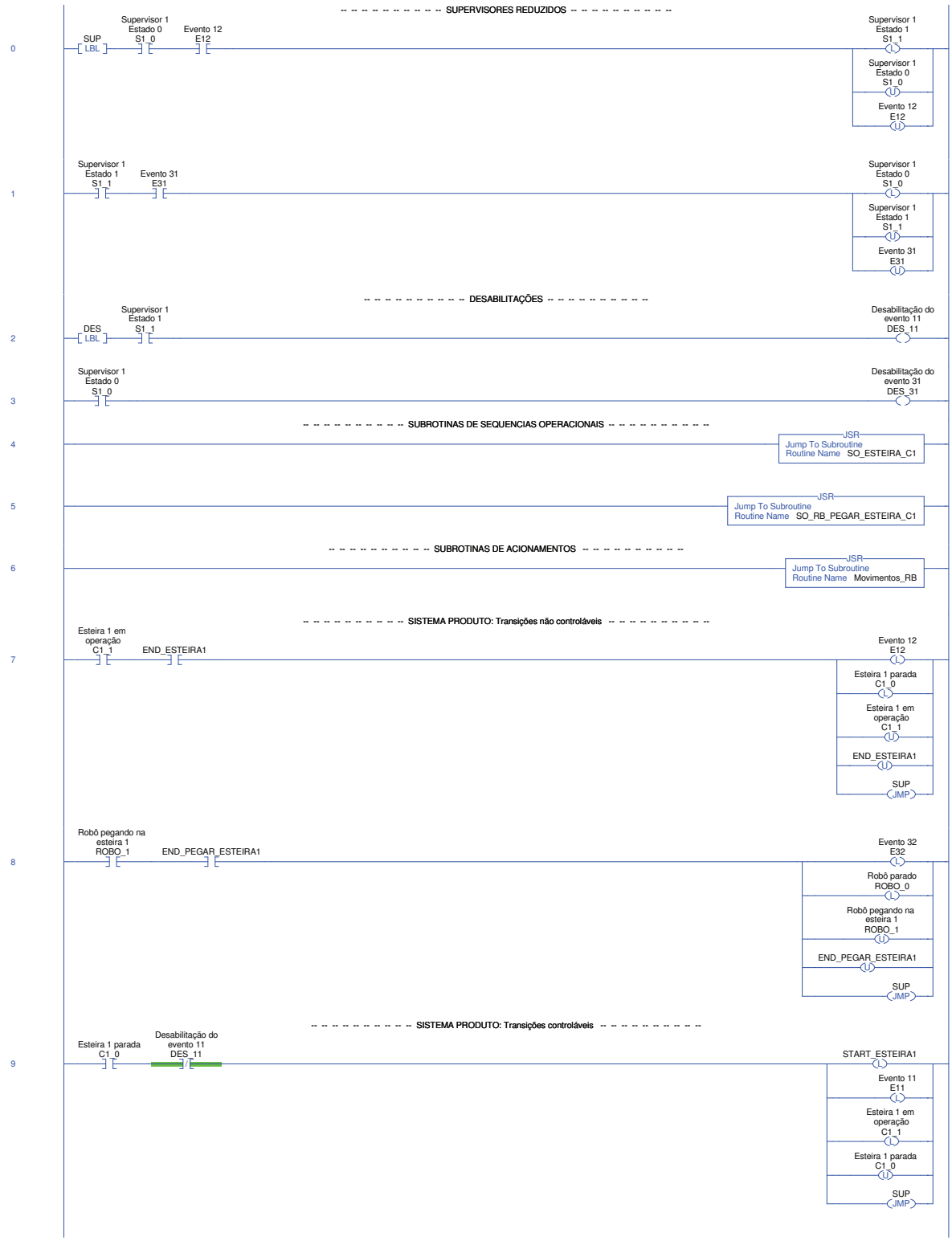

Figura A.2: Lógica Principal - Implementação I

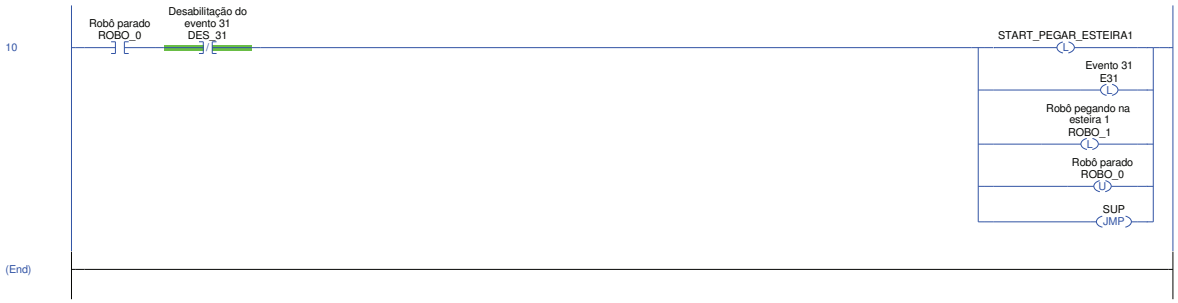

Figura A.3: Lógica Principal - Implementação I (Continuação)

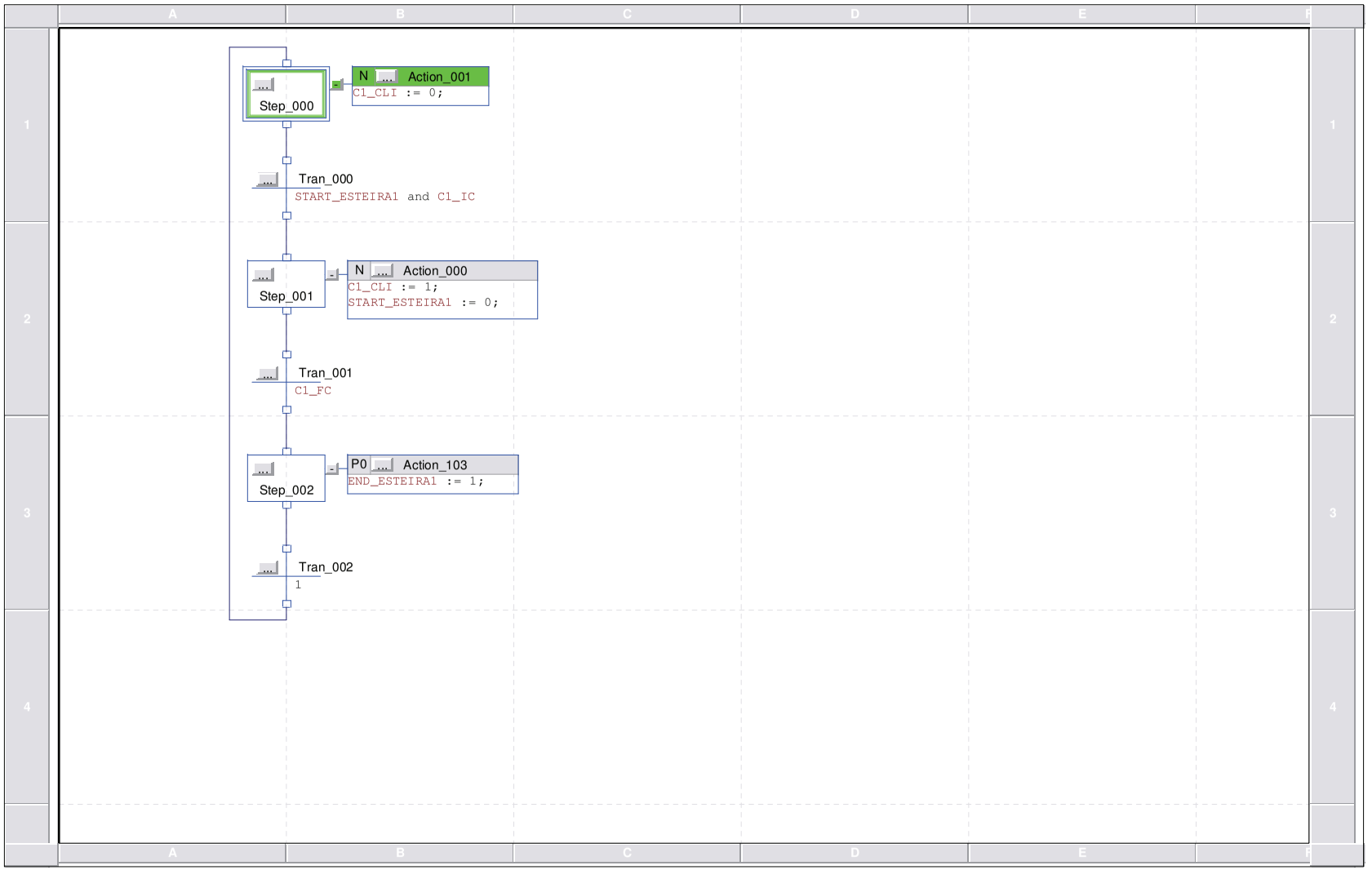

Figura A.4: Lógica Sequência Operacional Esteira C1

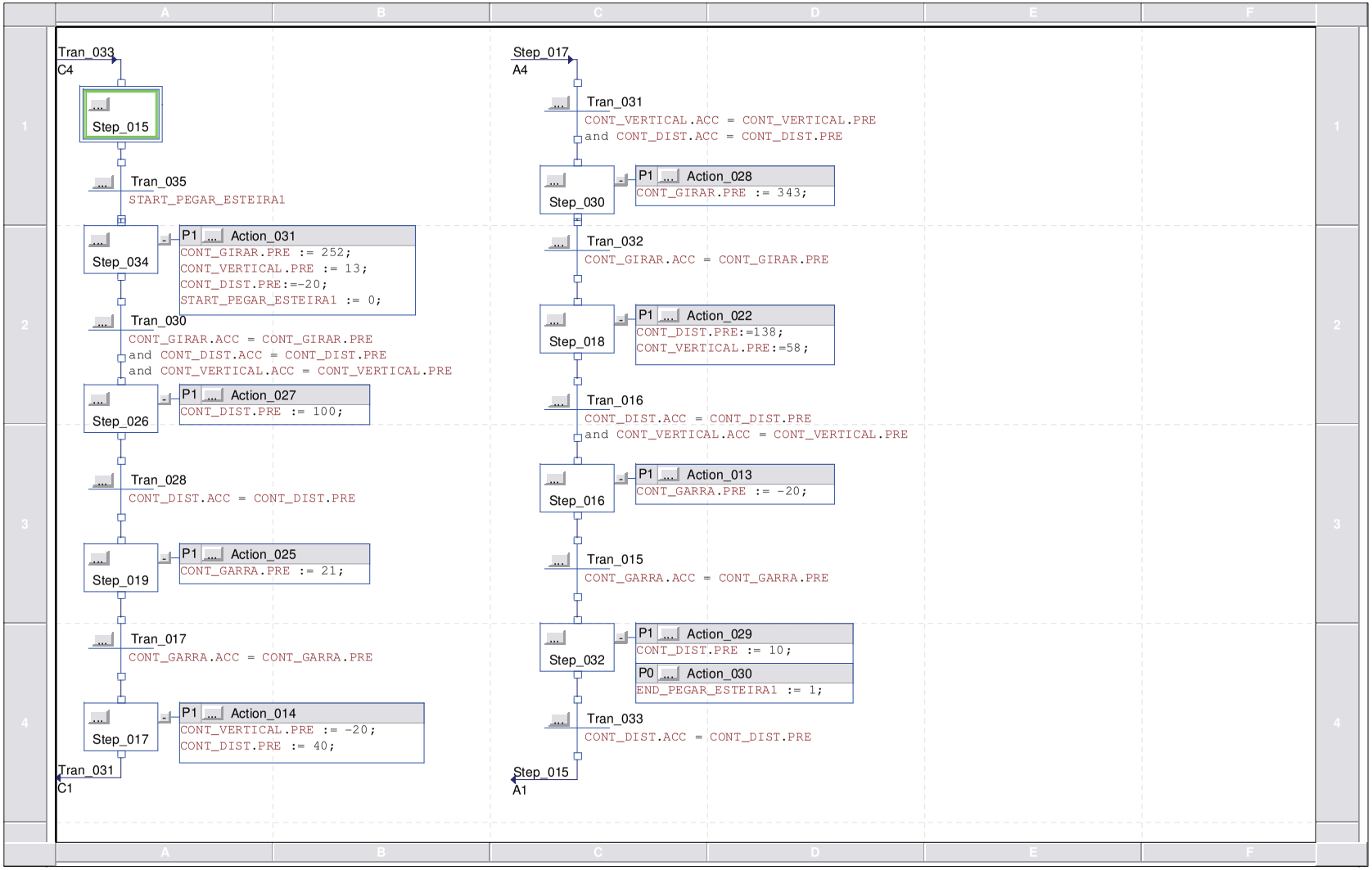

Figura A.5: Lógica Sequência Operacional Robô Pega Peça na Esteira C1

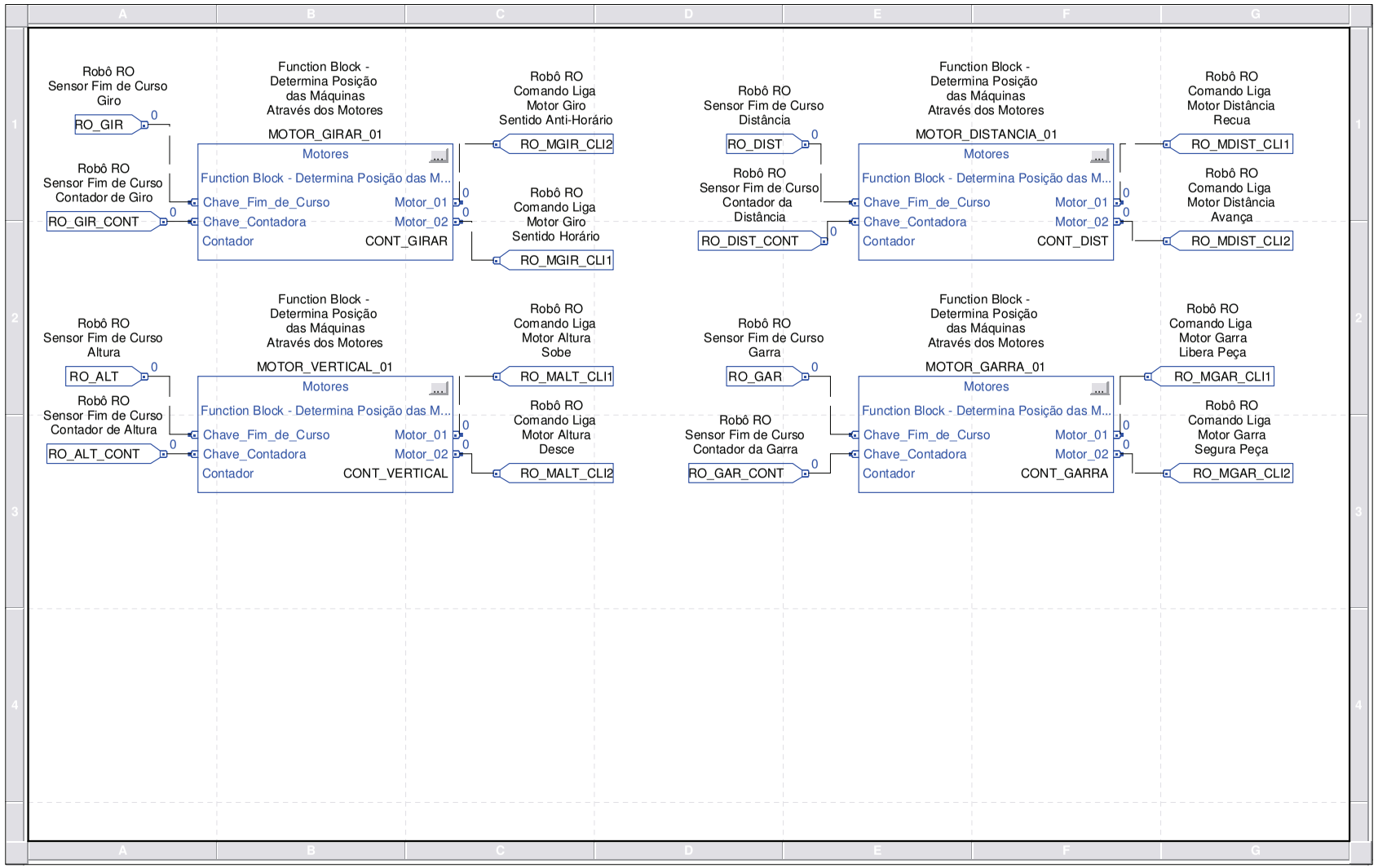

Figura A.6: Lógica Movimentos Robô

Apêndice B

Implementação II

Para elucidar, neste apêndice é apresentada a metodologia de implementação II [Leal et al., 2009] em CLP utilizada na automação do sistema flexível de manufatura didático que contém a estrutura do programa no CLP, a lógica de programação principal e a lógica que envolve a esteira C1 e o robô do SFM didático. É apresentada a lógica do autômato supervisor S_{1red} (nível Supervisores Modulares), as lógicas dos autômatos esteira C1 e robô (nível Sistema-Produto) e as instruções que fazem parte da lógica principal desta metodologia. As lógicas das sequências operacionais esteira C1 e robô (nível Sequência Operacional) e a lógica de programação dos movimentos do robô são também apresentadas.

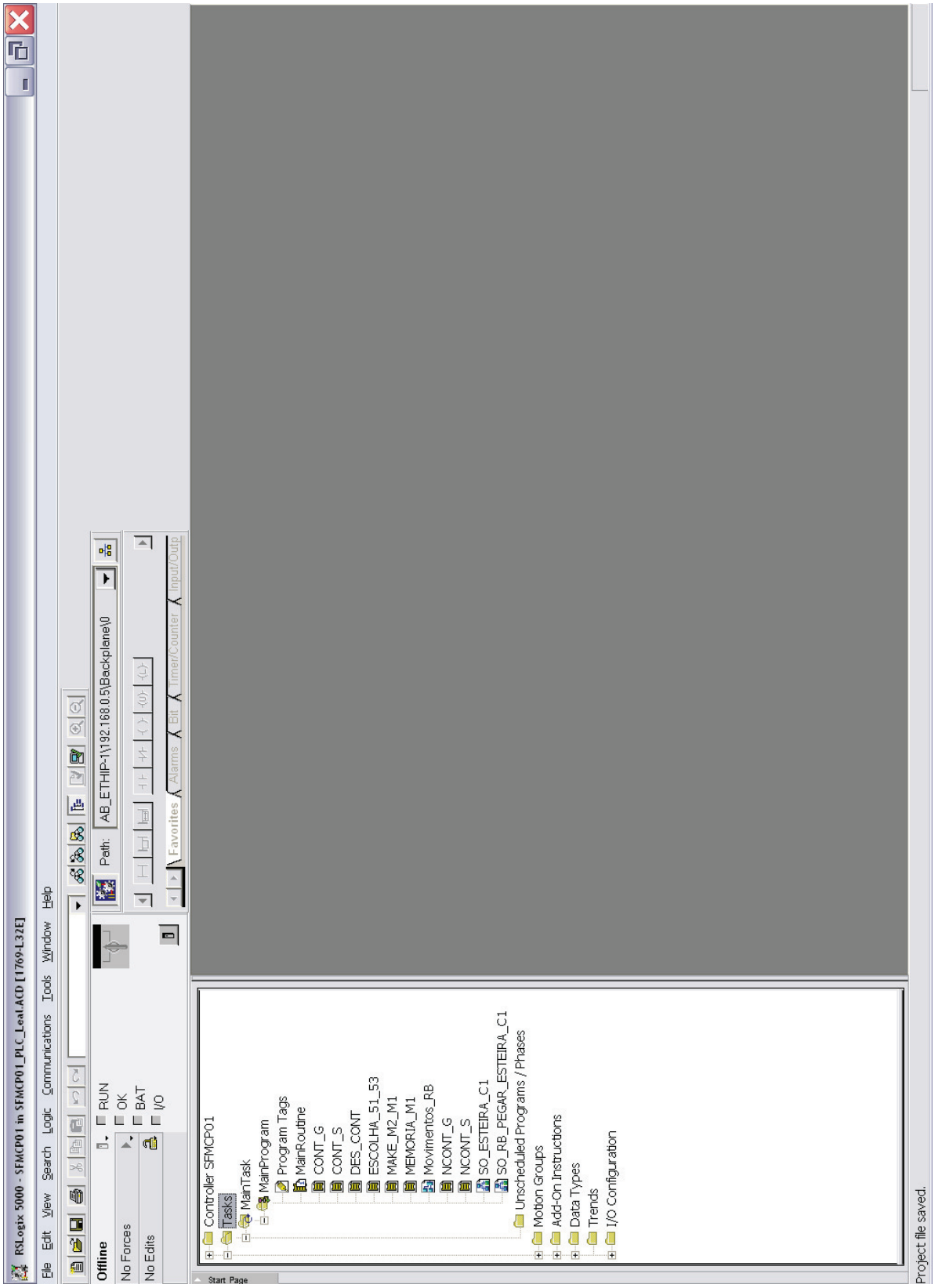

Figura B.1: Estrutura Programa CLP

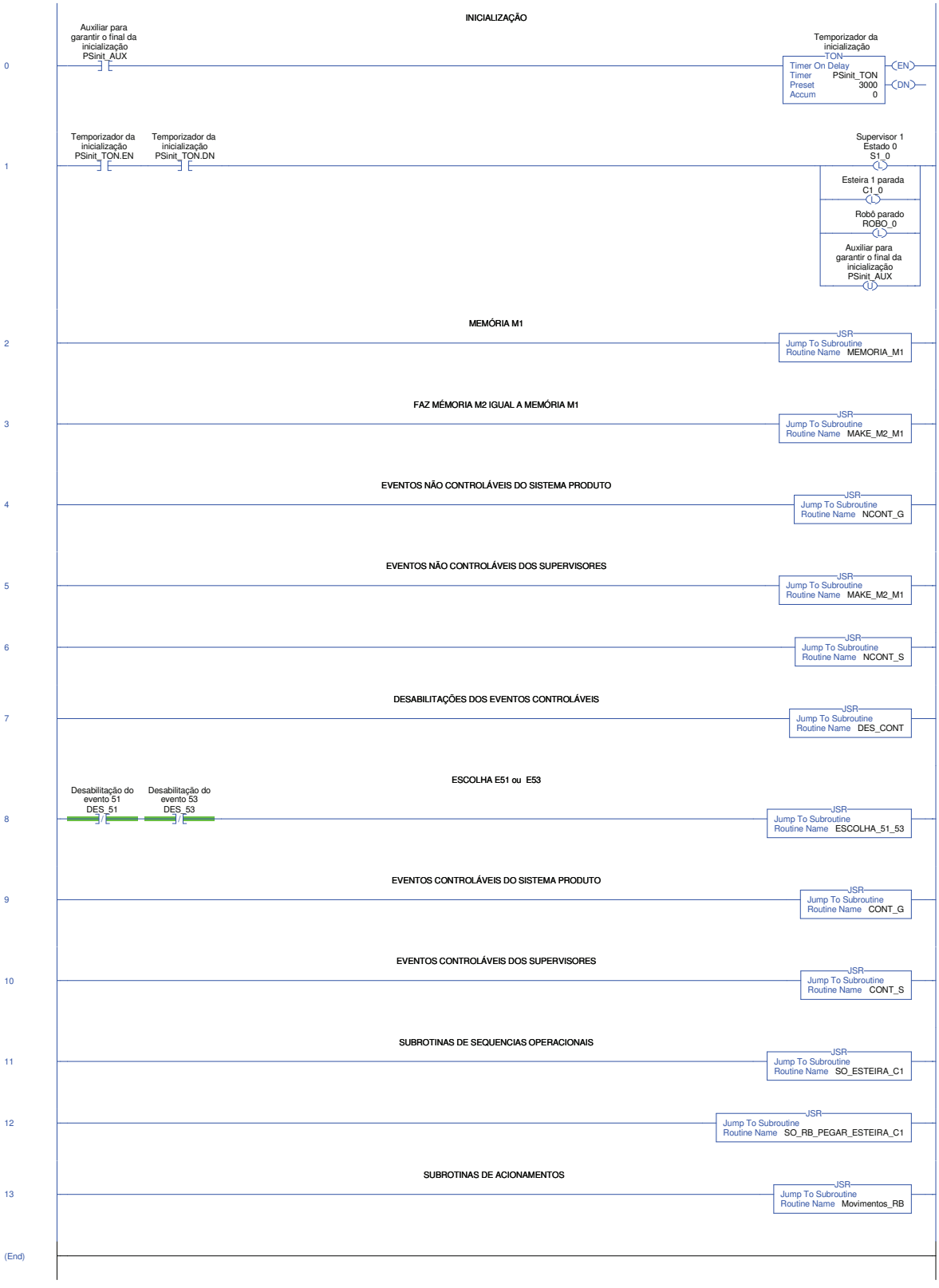

Figura B.2: Lógica Rotina Principal

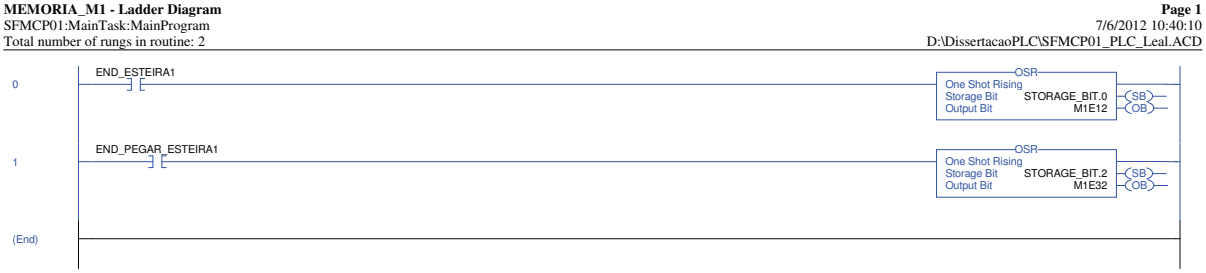

Figura B.3: Lógica Memória M1

Figura B.4: Lógica Memória M2 Igual a M1

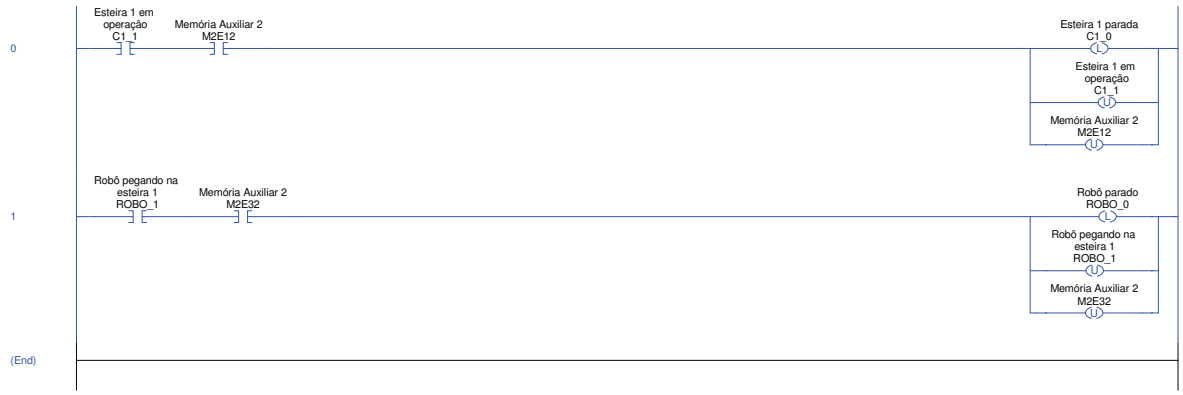

Figura B.5: Lógica Eventos Não Controláveis Sistema Produto

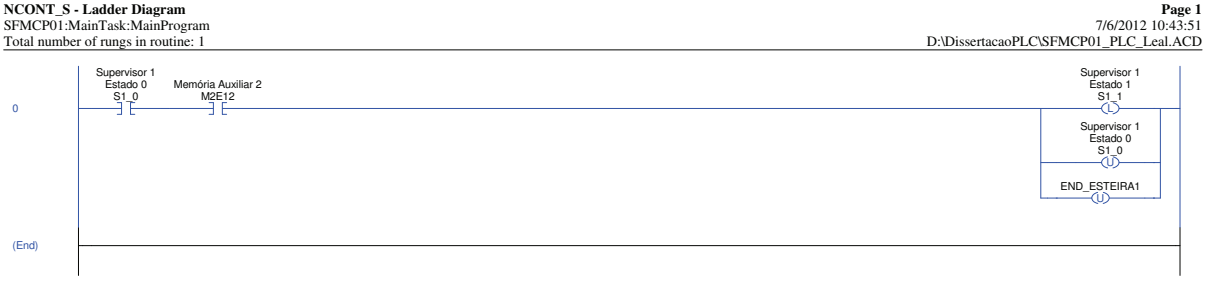

Figura B.6: Lógica Eventos Não Controláveis Supervisores

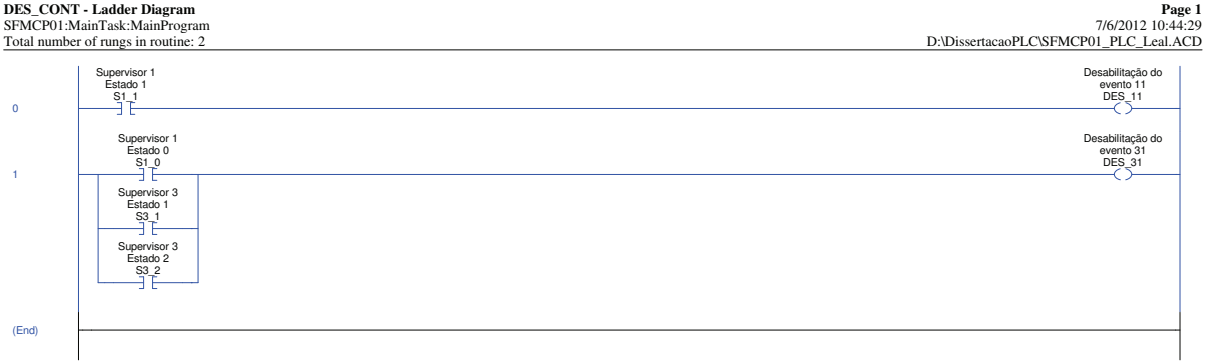

Figura B.7: Lógica Desabilitações Eventos Controláveis

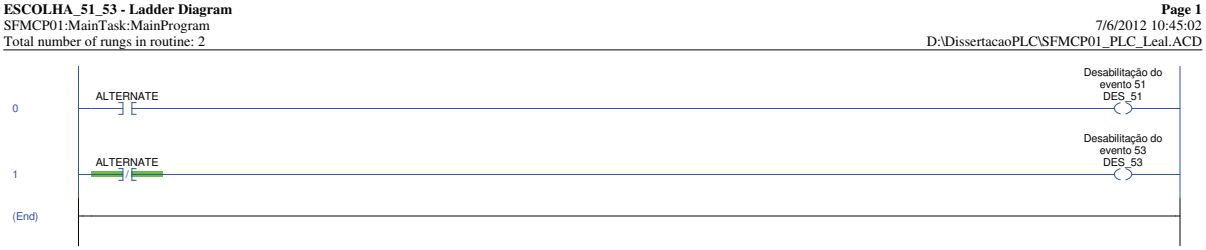

Figura B.8: Lógica Escolha Eventos 51 ou 53

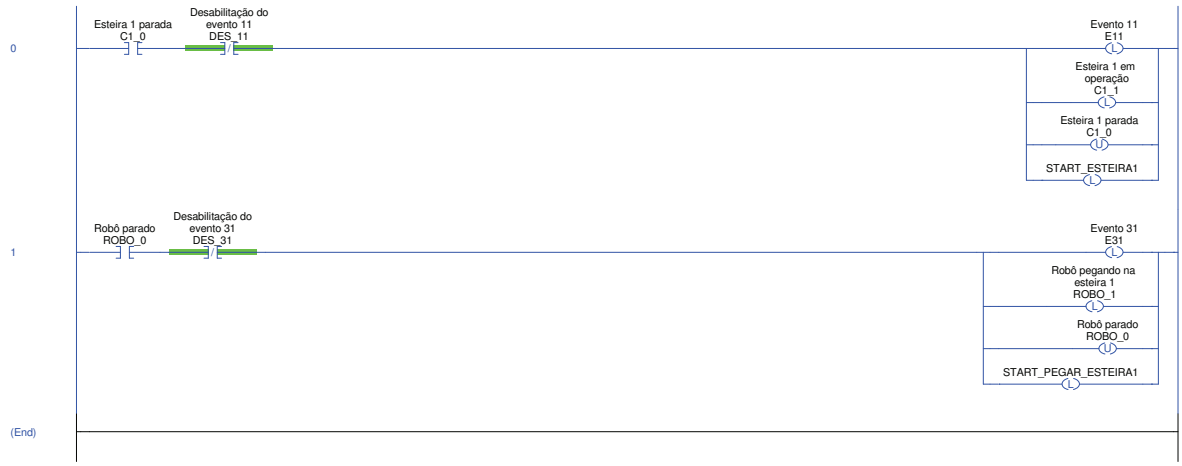

Figura B.9: Lógica Eventos Controláveis Sistema Produto

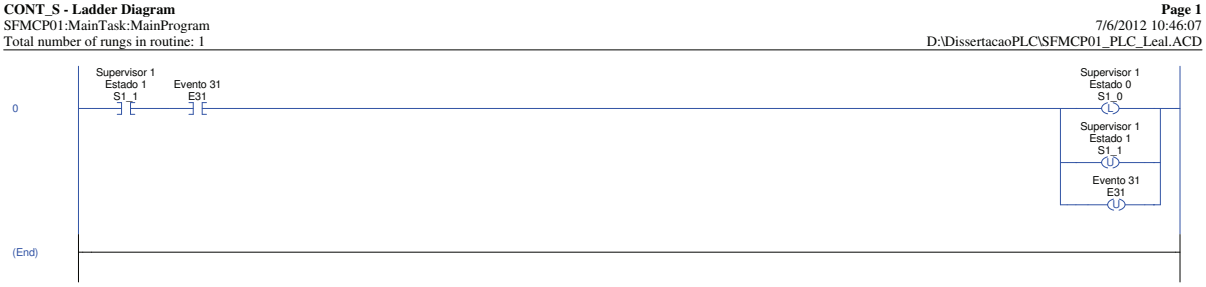

Figura B.10: Lógica Eventos Controláveis Supervisores

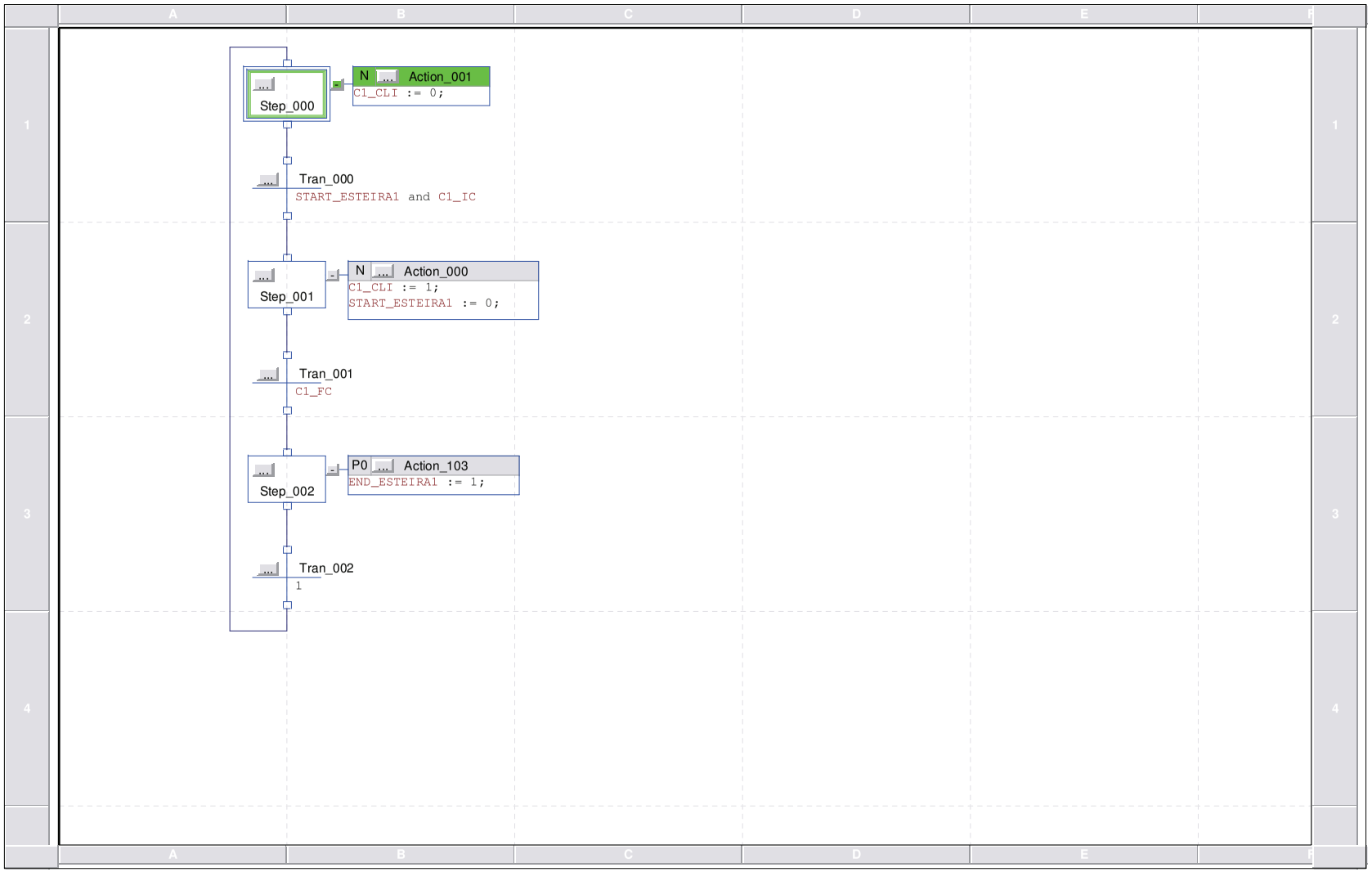

Figura B.11: Lógica Sequência Operacional Esteira C1

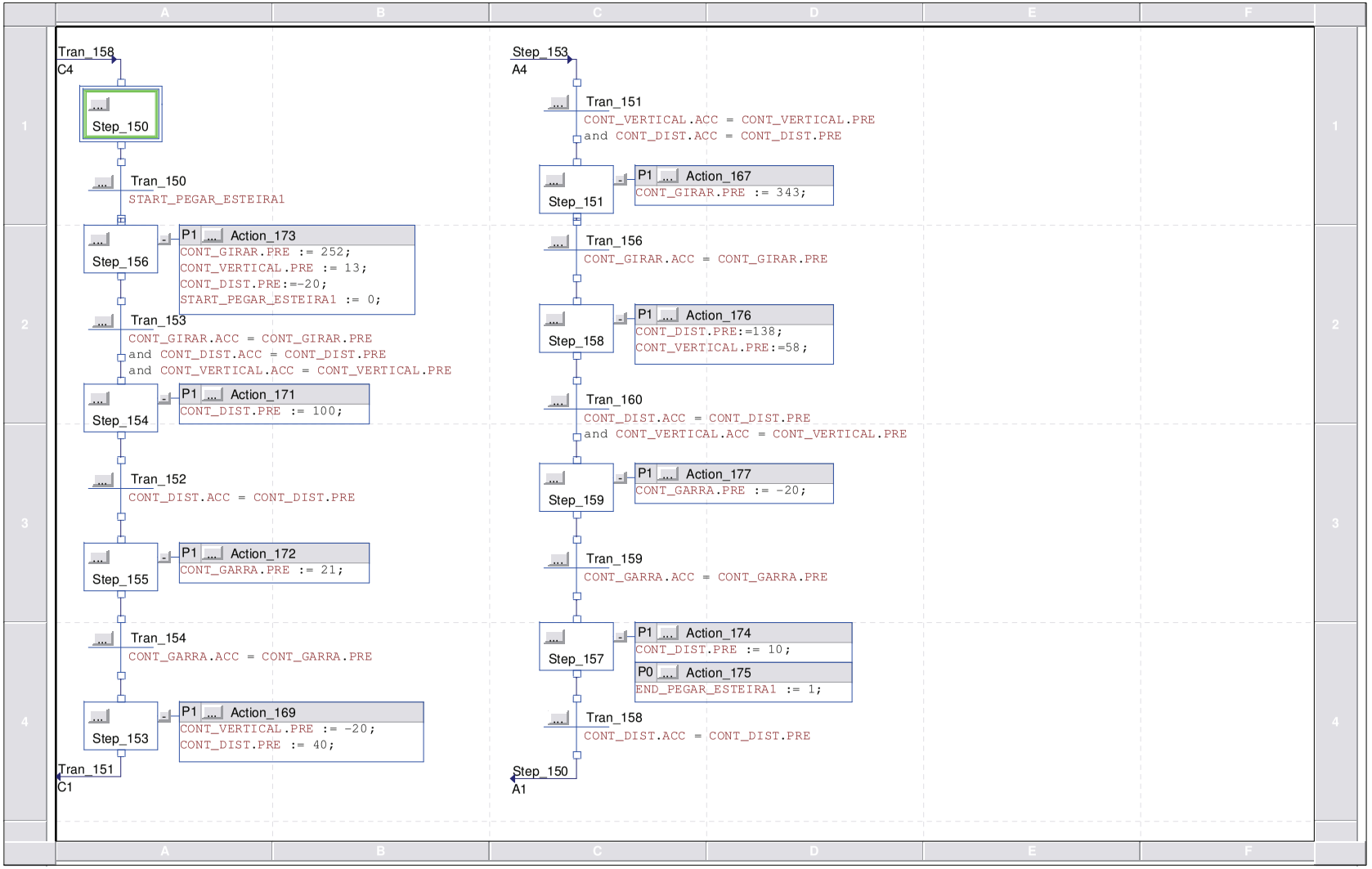

Figura B.12: Lógica Sequência Operacional Robô Pega Peça na Esteira C1

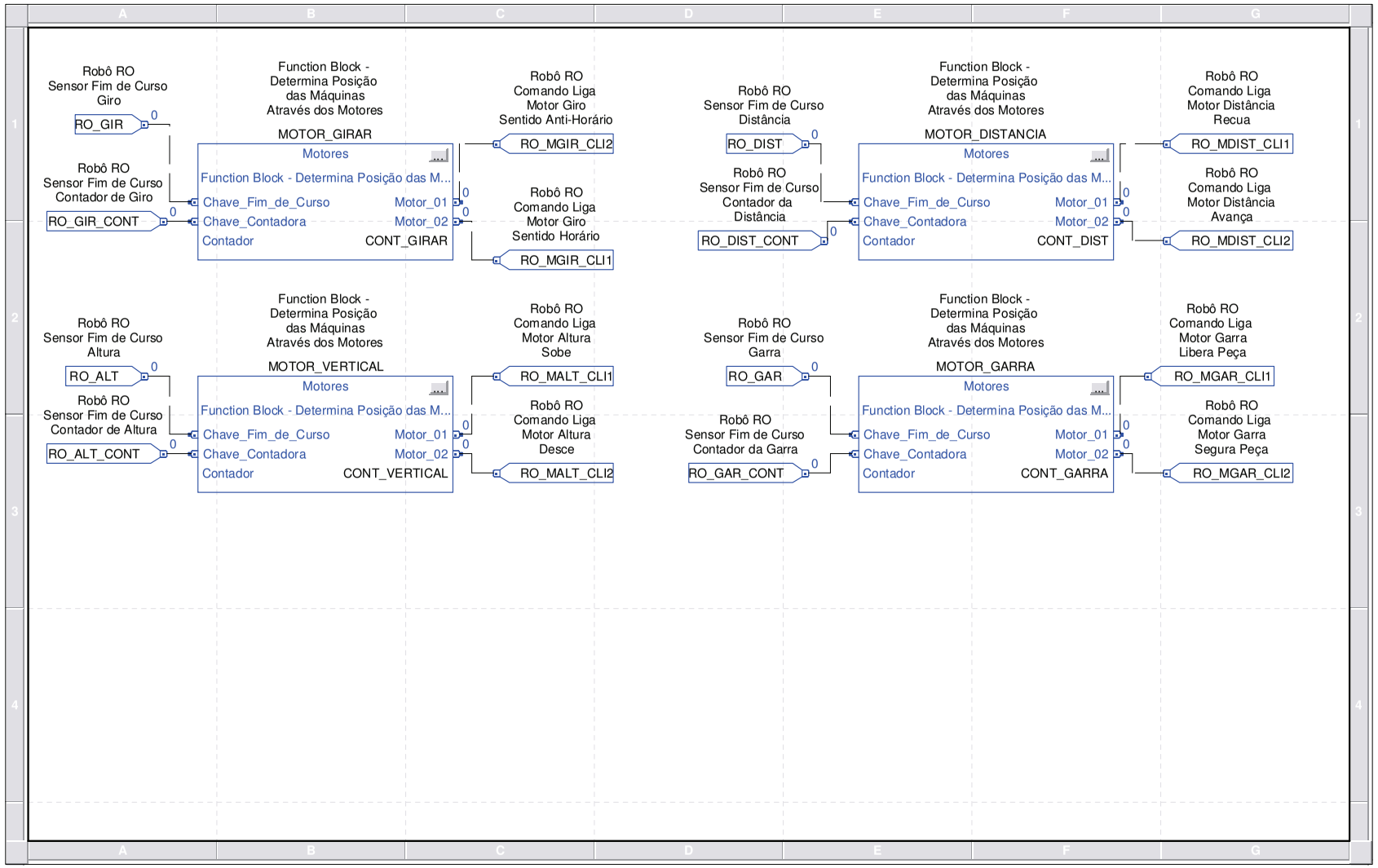

Figura B.13: Lógica Movimentos Robô

Apêndice C

Implementação III

Para elucidar, neste apêndice é apresentada a metodologia de implementação III [Vieira, 2007] em CLP utilizada na automação do sistema flexível de manufatura didático que contém a estrutura do programa no CLP, a lógica de programação principal e a lógica que envolve a esteira C1 e o robô do SFM didático. É apresentada a lógica do autômato supervisor S_{1red} (nível Supervisores Modulares), as lógicas dos autômatos esteira C1 e robô (nível Sistema-Produto) e as instruções que fazem parte da lógica principal desta metodologia. As lógicas das sequências operacionais esteira C1 e robô (nível Sequência Operacional) e a lógica de programação dos movimentos do robô são também apresentadas.

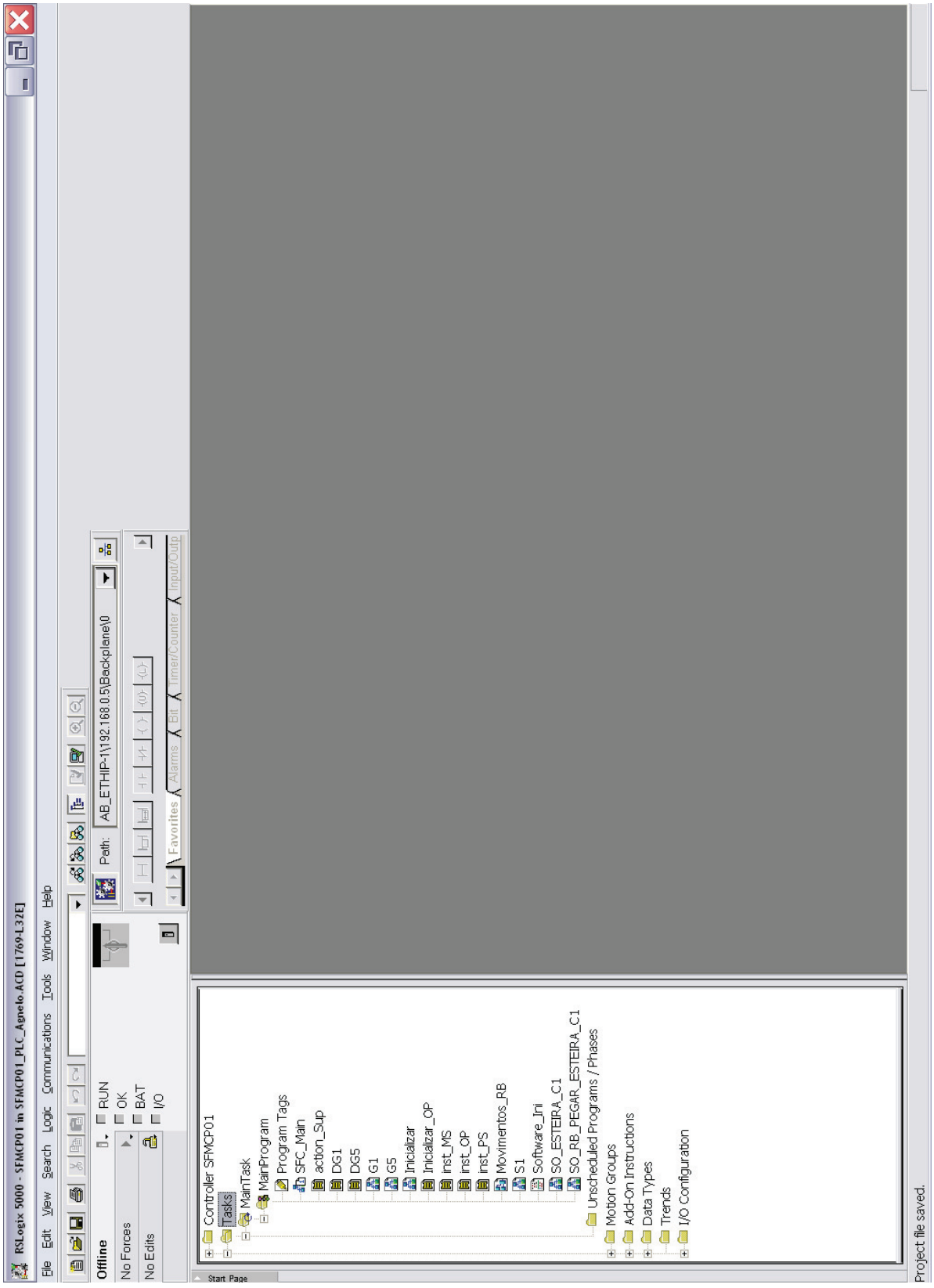

Figura C.1: Estrutura Programa CLP

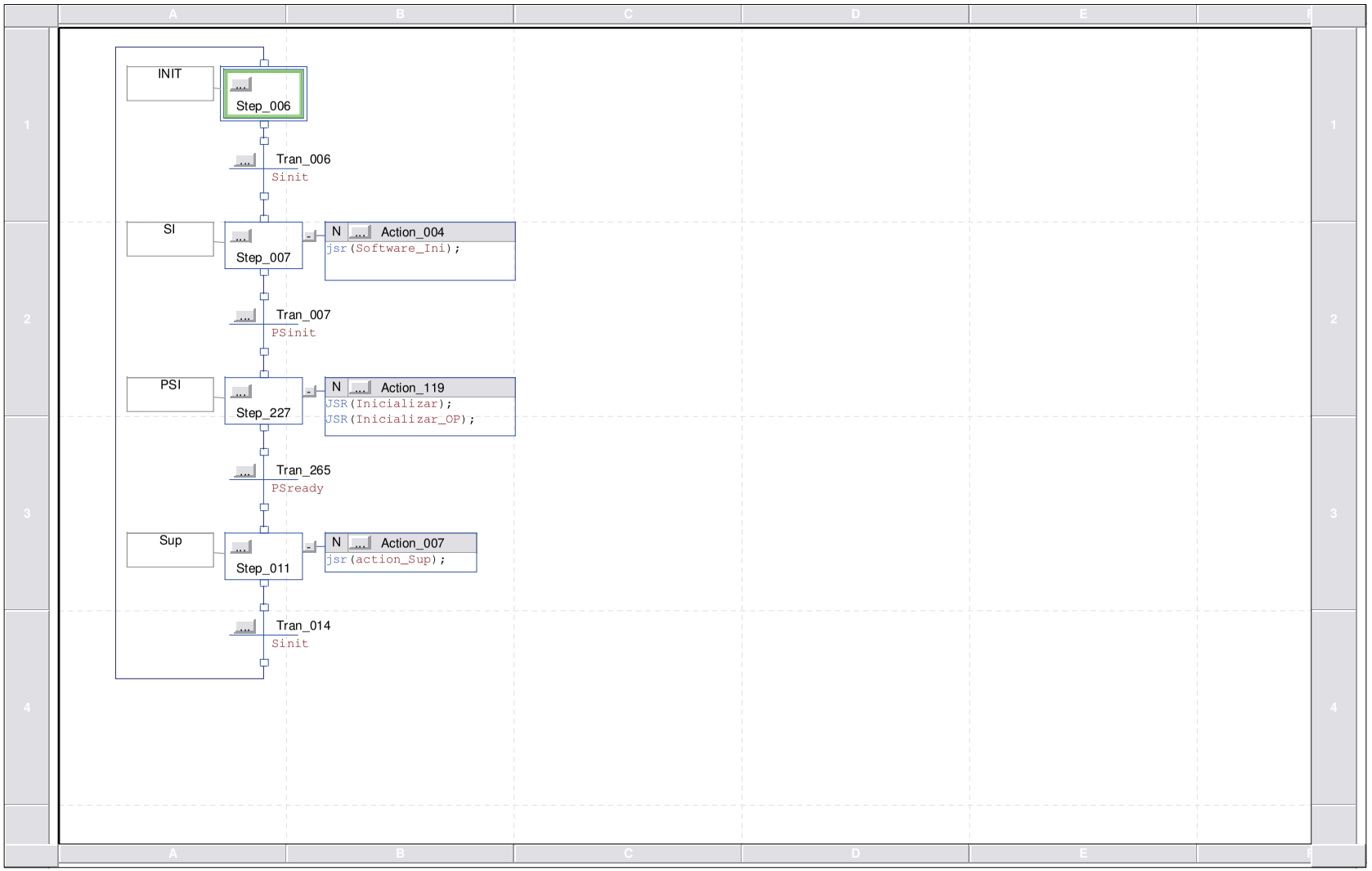

Figura C.2: Lógica SFC Principal

action_Sup - Ladder Diagram **Page 1**
SFMCP01:MainTask:MainProgram 7/6/2012 19:22:07
Total number of rungs in routine: 8 D:\DissertacaoPLC\SFMCP01_PLC_AGNELO.ACD

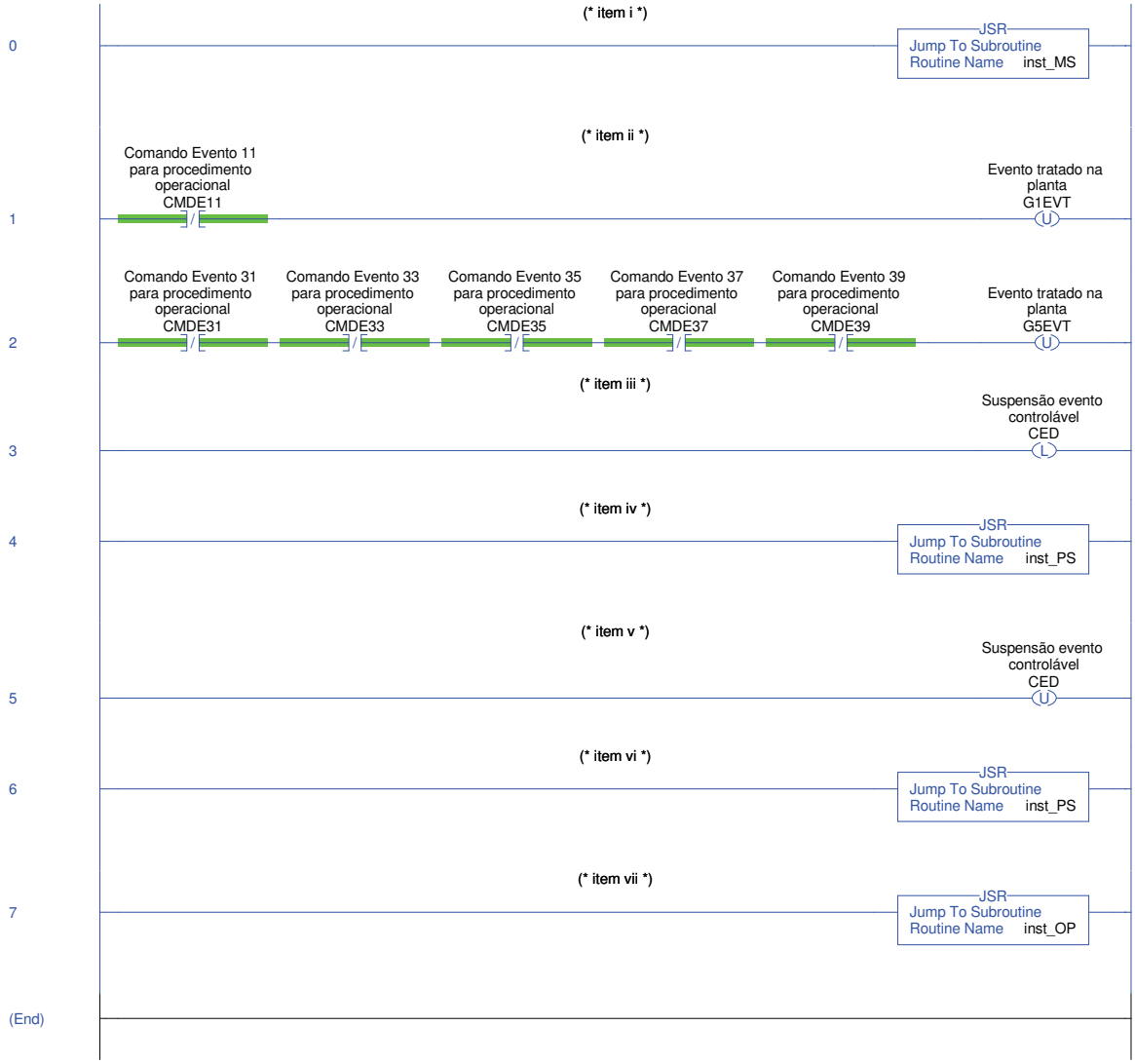

Figura C.3: Lógica Modo Supervisionado (action SUP)

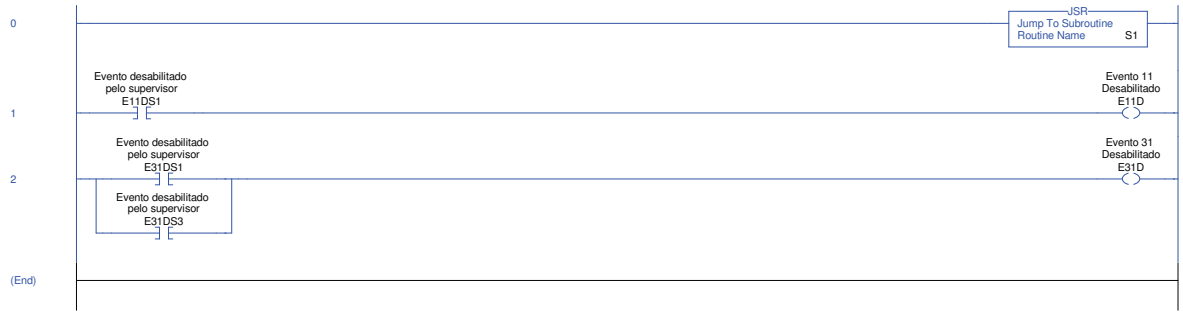

Figura C.4: Lógica Chamada dos Supervisores Modulares (inst MS)

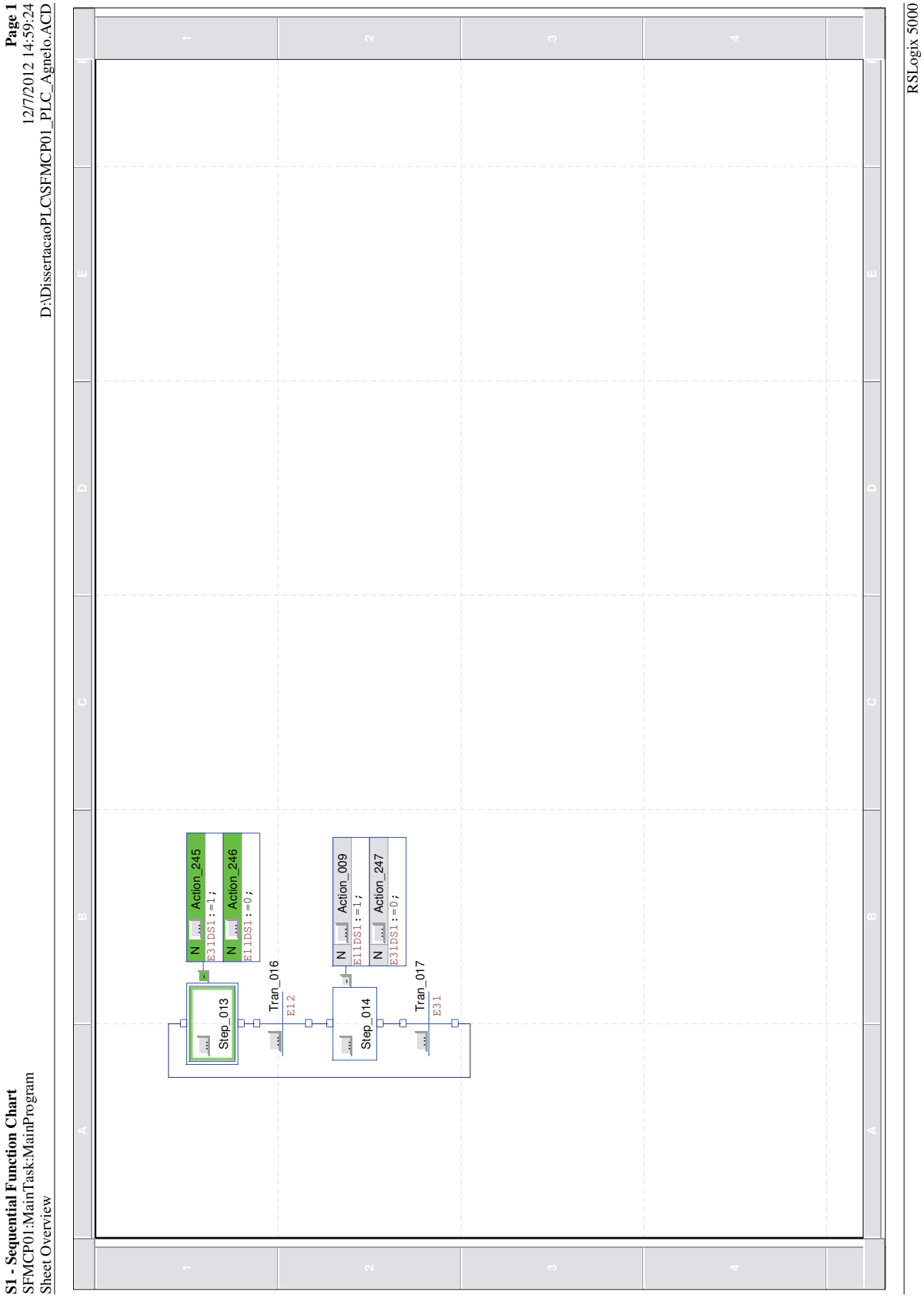

Figura C.5: Lógica Supervisor S_{1red}

Figura C.6: Lógica Chamada do Sistema-Produto (inst PS)

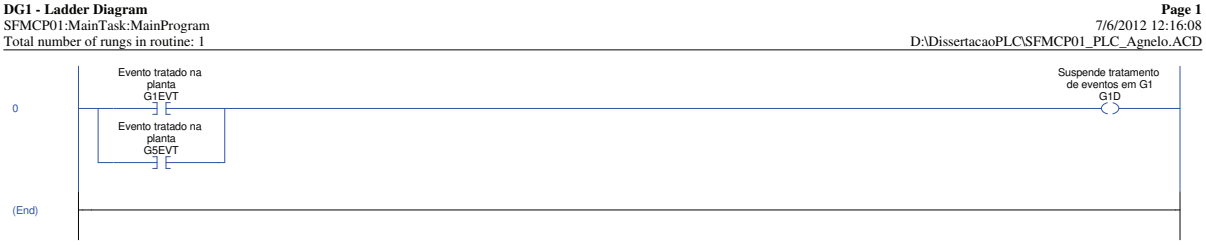

Figura C.7: Lógica DG1

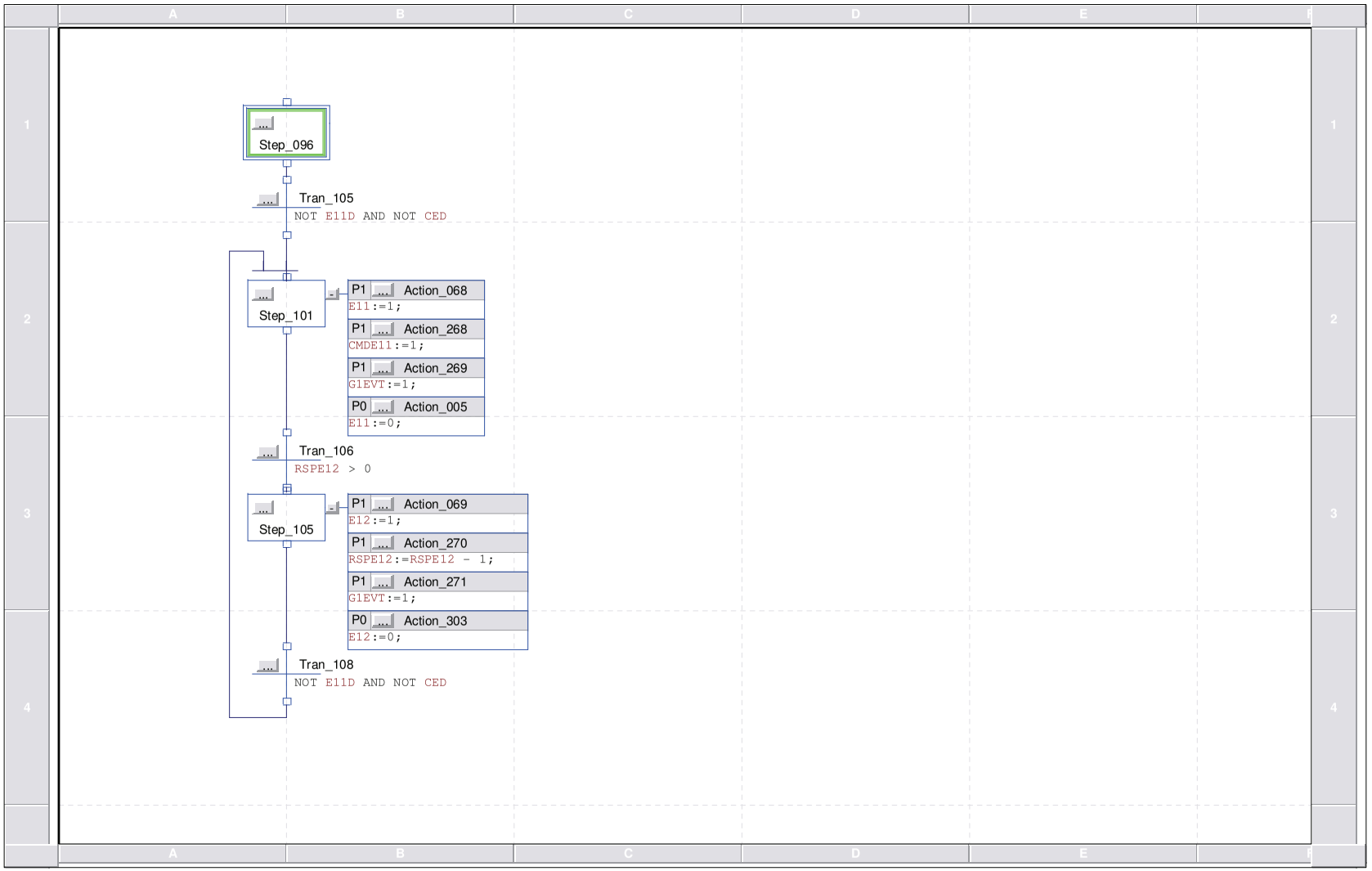

Figura C.8: Lógica G1 - Esteira C1

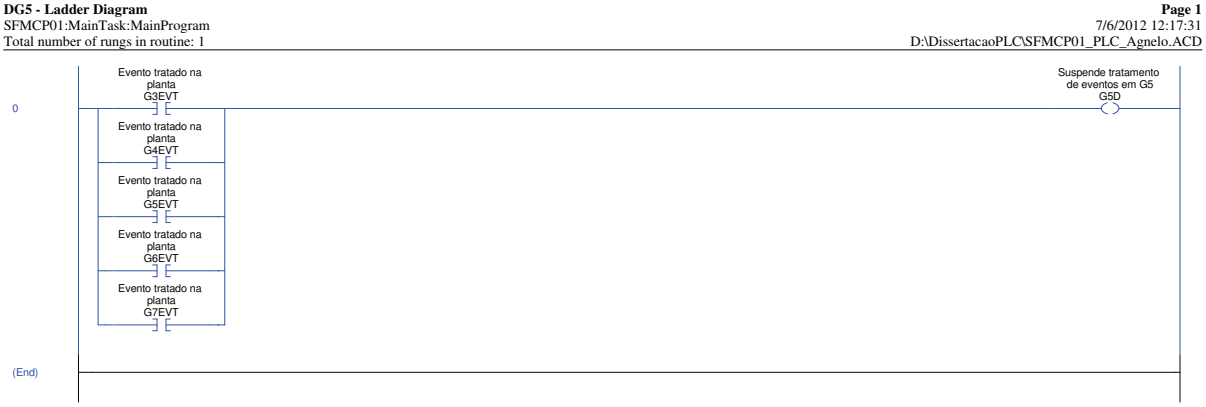

Figura C.9: Lógica DG5

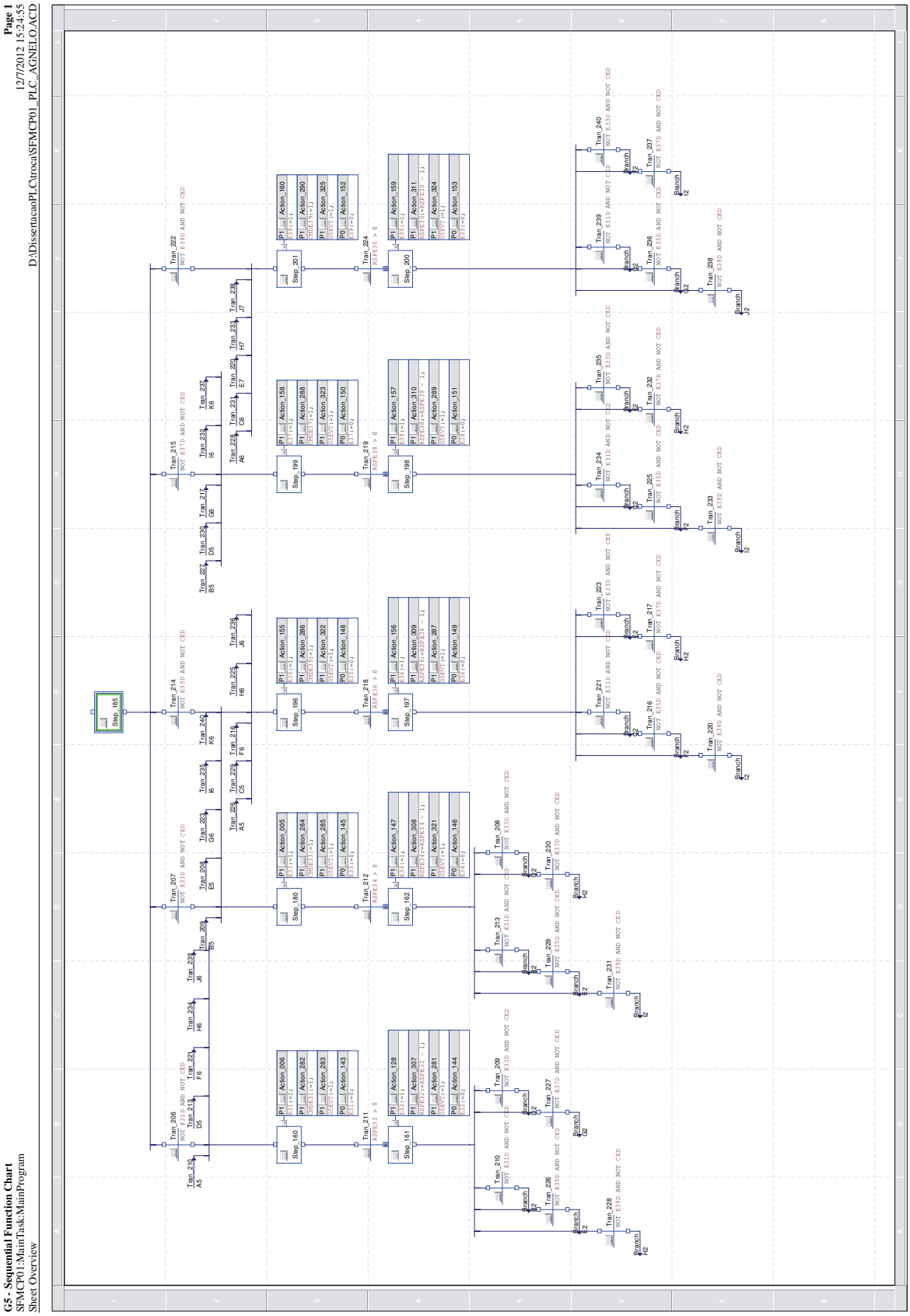

Figura C.10: Lógica G5 - Robô

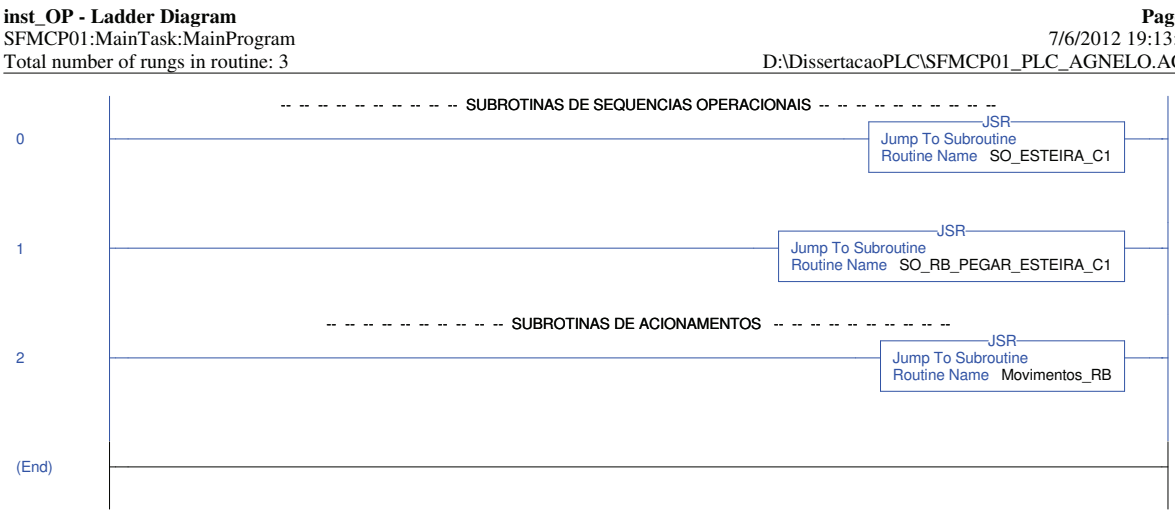

Figura C.11: Lógica Chamada das Sequências Operacionais (inst OP)

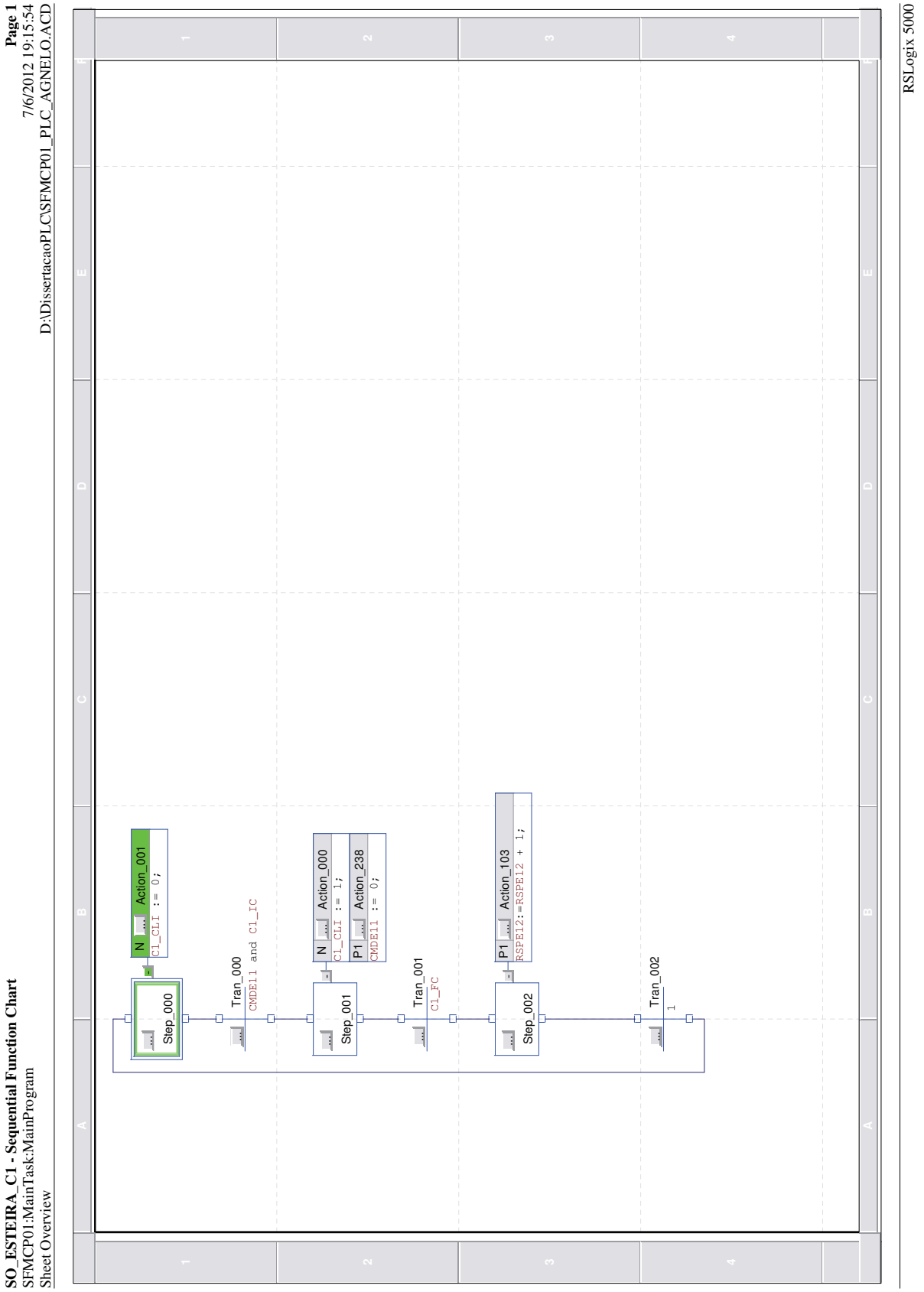

Figura C.12: Lógica Sequência Operacional Esteira C1

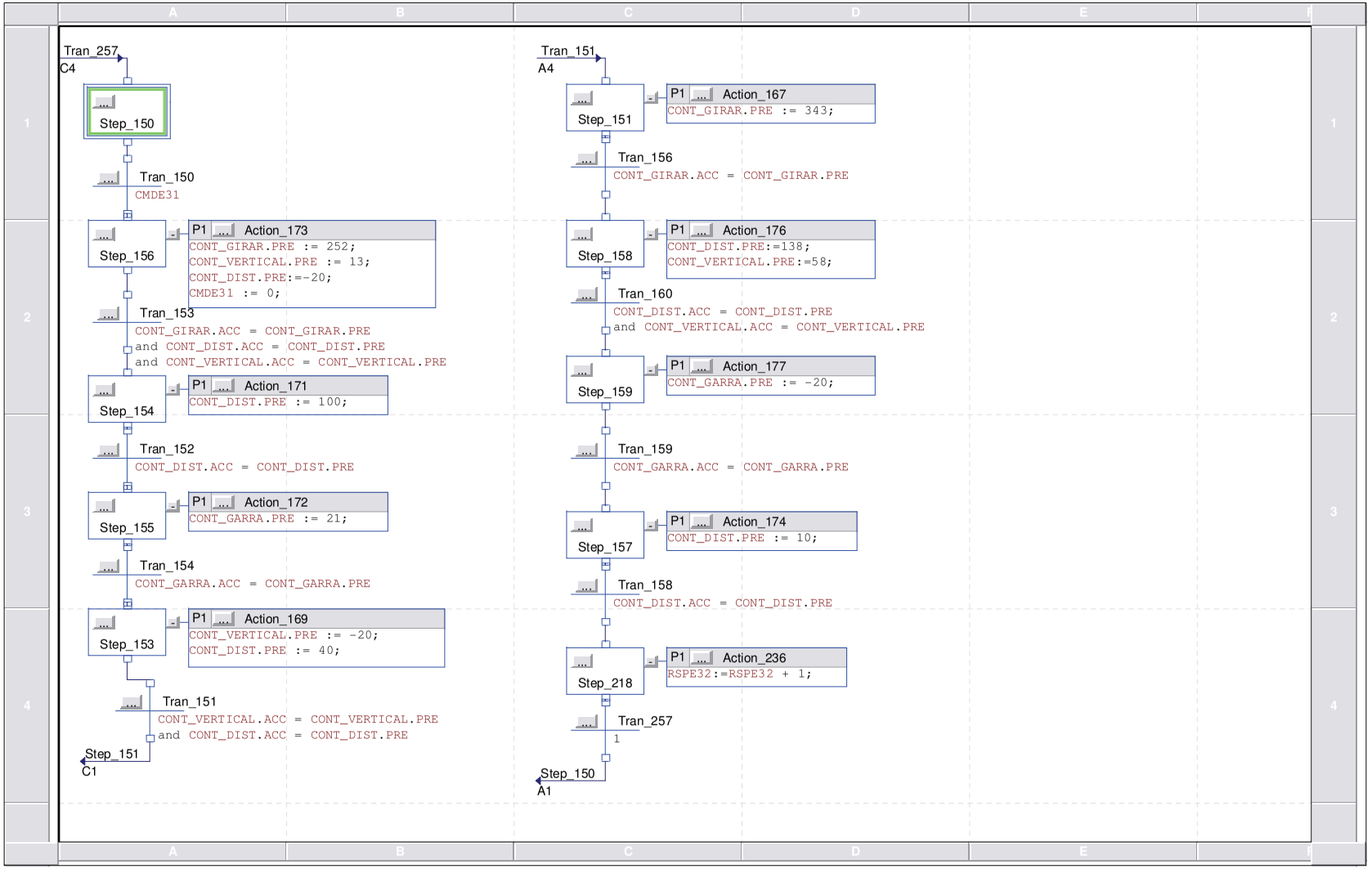

Figura C.13: Lógica Sequência Operacional Robô Pega Peça na Esteira C1

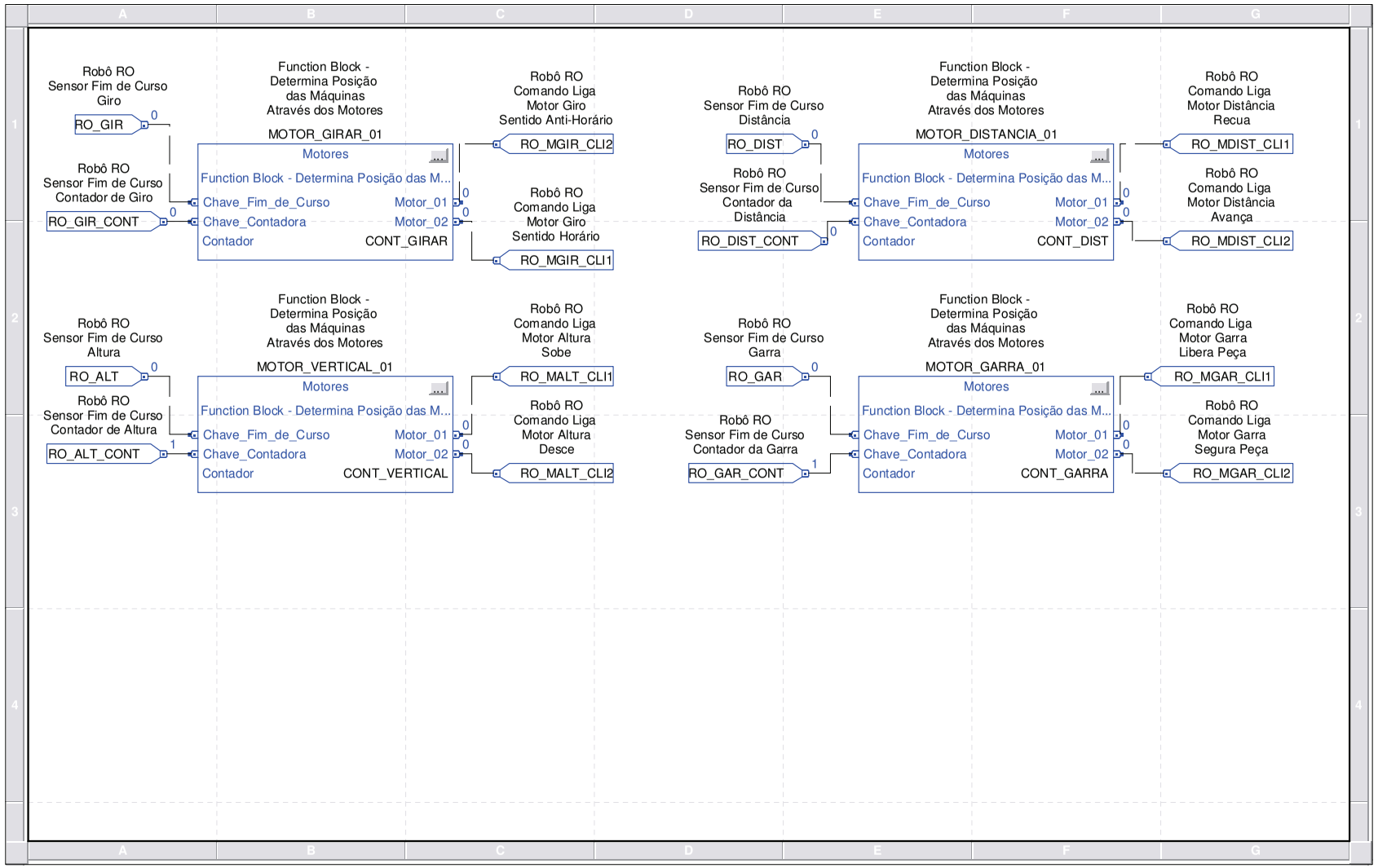

Figura C.14: Lógica Movimentos Robô

Apêndice D

Implementação IV

Para elucidar, neste apêndice é apresentada a metodologia de implementação IV [Lima II, 2002] em CLP utilizada na automação do sistema flexível de manufatura didático que contém a estrutura do programa no CLP, a lógica de programação principal e a lógica que envolve a esteira C1 e o robô do SFM didático. É apresentada a lógica do Modelo Supervisionado, Executor e as instruções que fazem parte da lógica principal desta metodologia. As lógicas das sequências operacionais esteira C1 e robô (nível Sequência Operacional) e a lógica de programação dos movimentos do robô são também apresentadas.

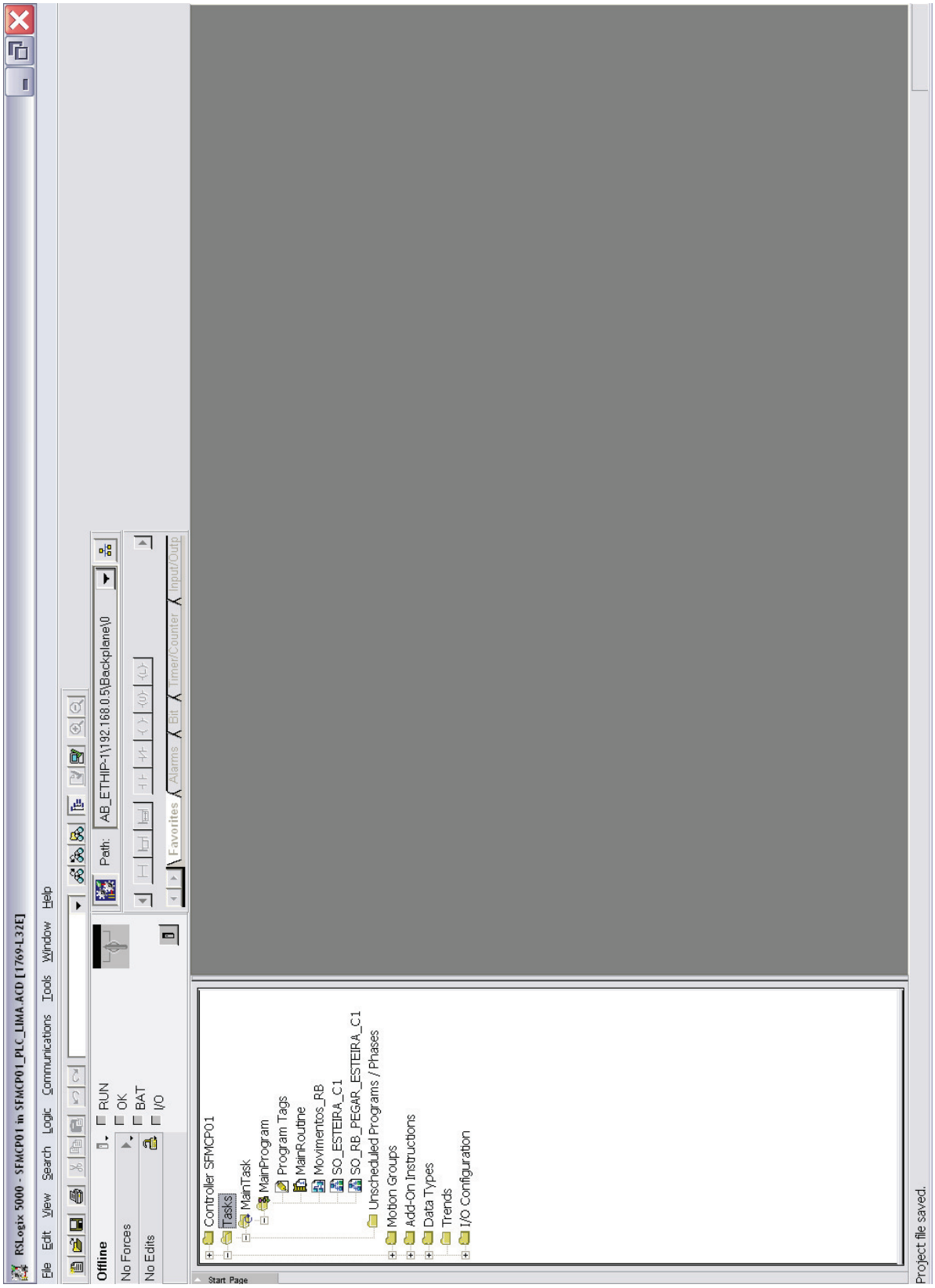

Figura D.1: Estrutura Programa CLP

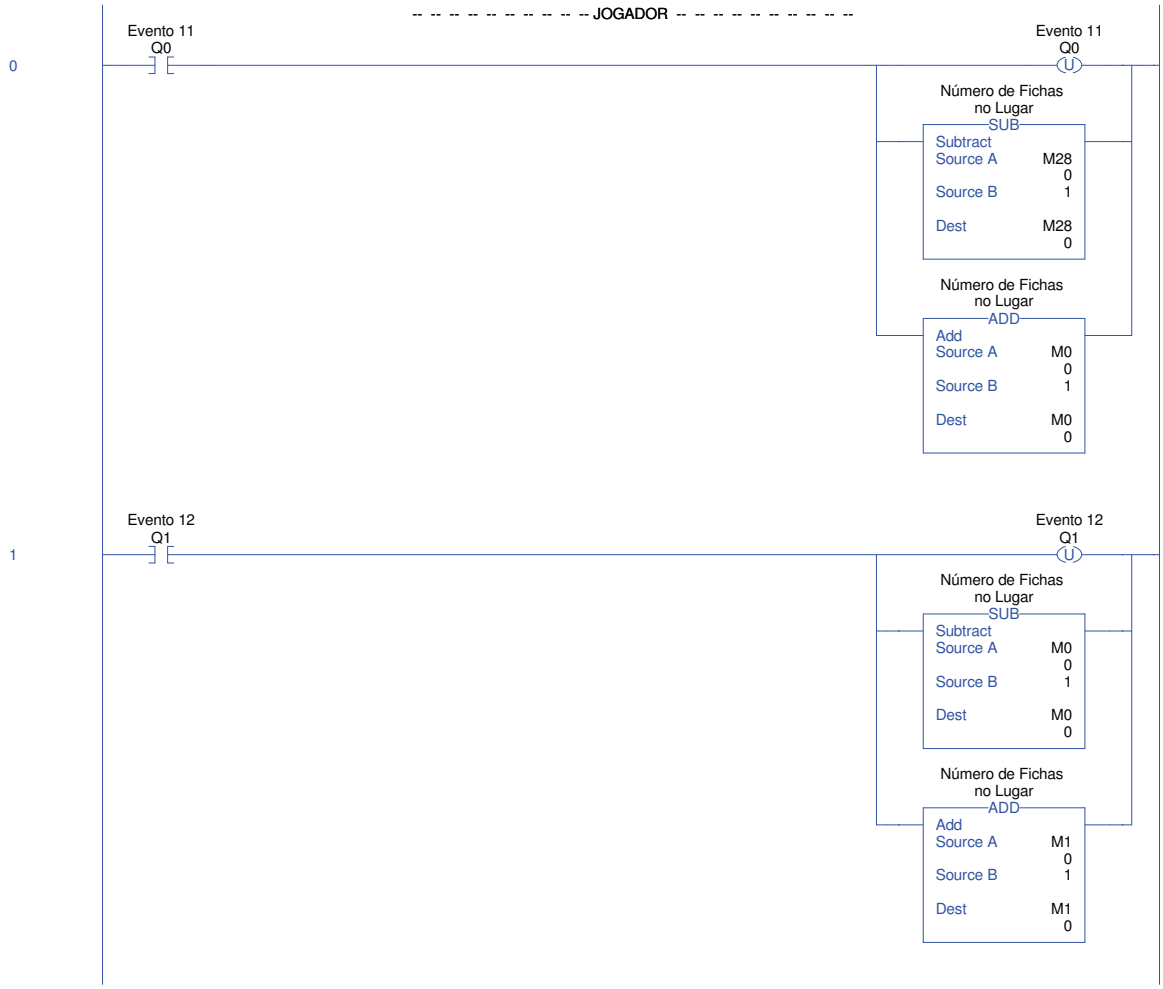

Figura D.2: Lógica Principal - Implementação IV

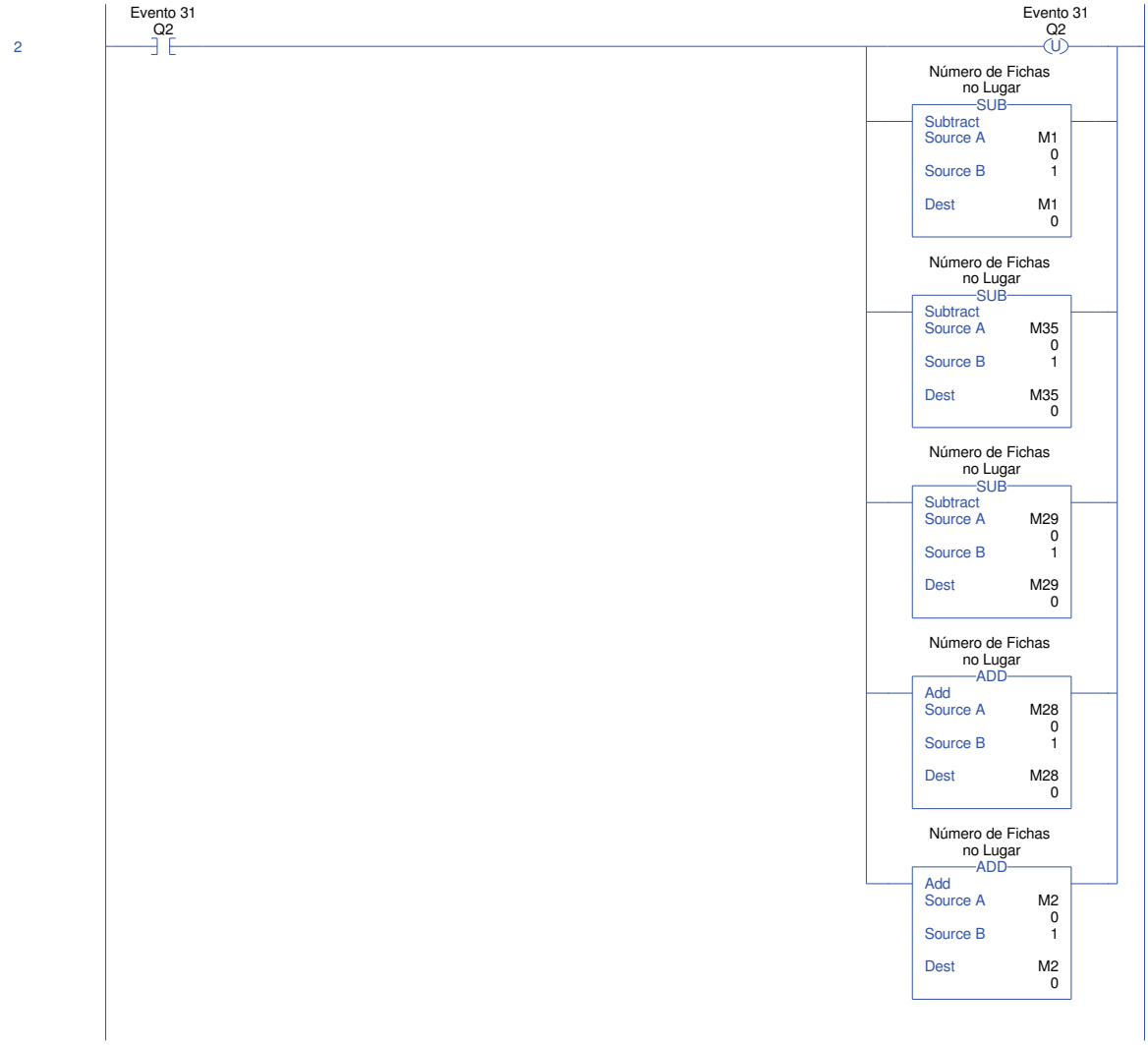

Figura D.3: Lógica Principal - Implementação IV (Continuação)

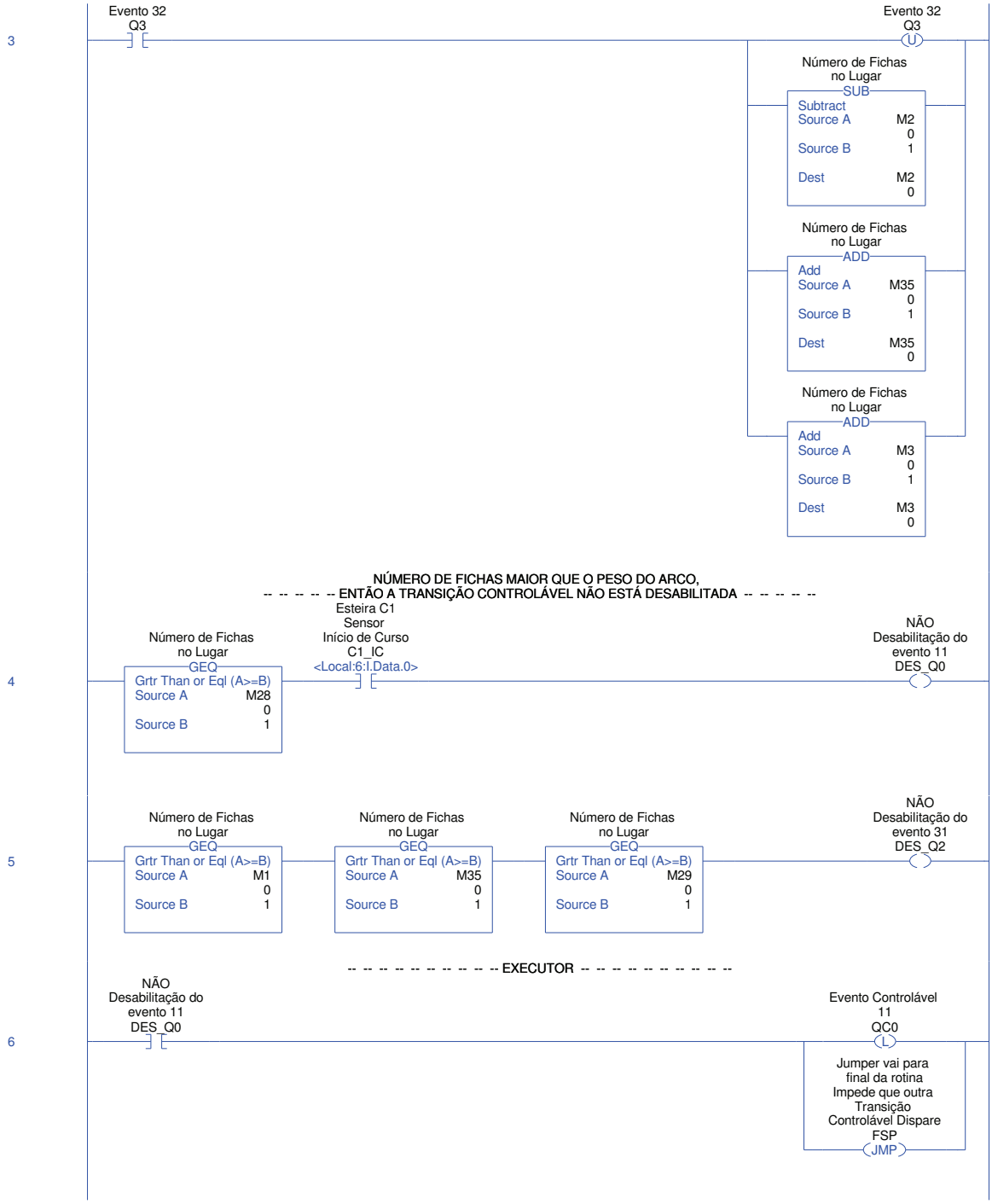

Figura D.4: Lógica Principal - Implementação IV (Continuação)

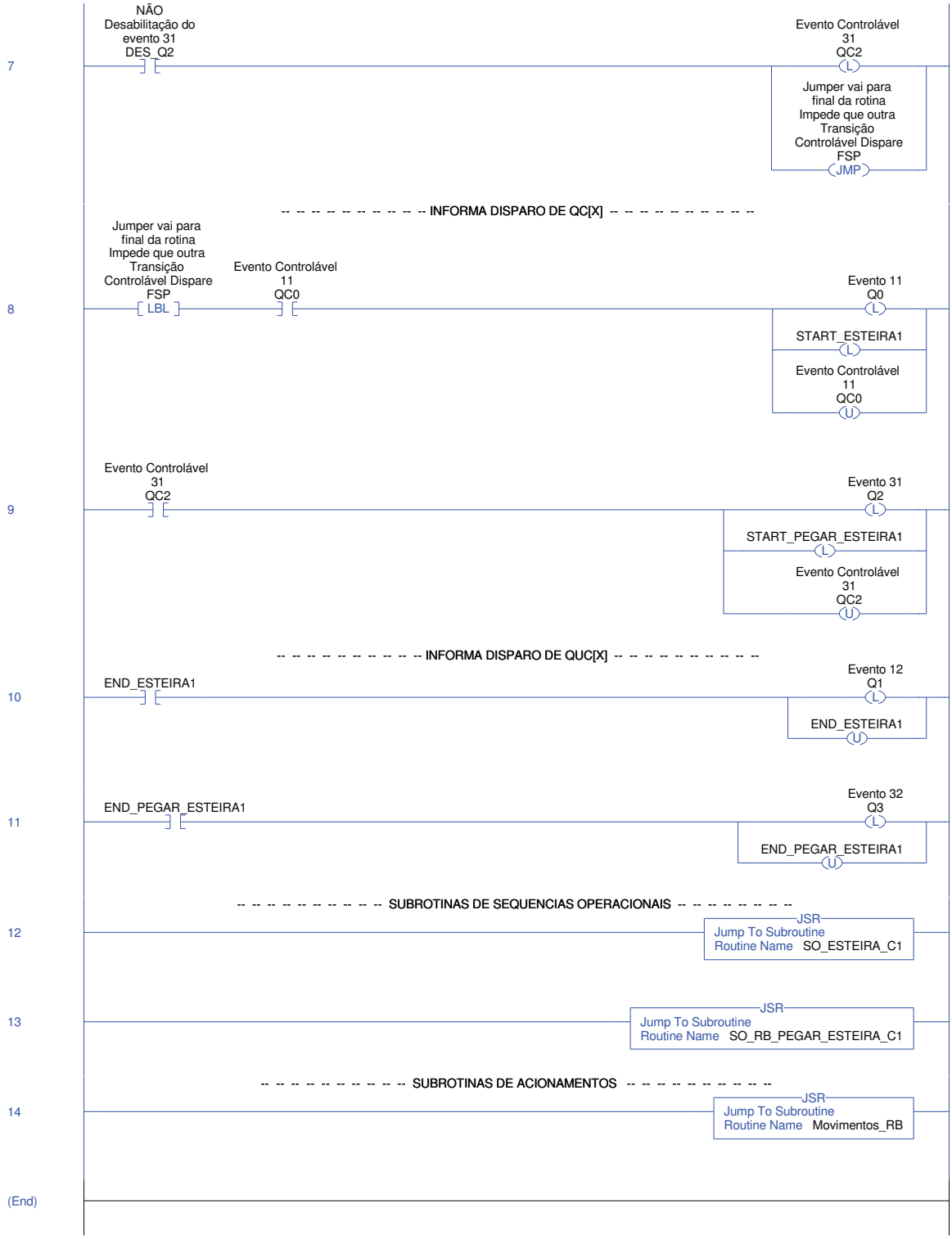

Figura D.5: Lógica Principal - Implementação IV (Continuação)

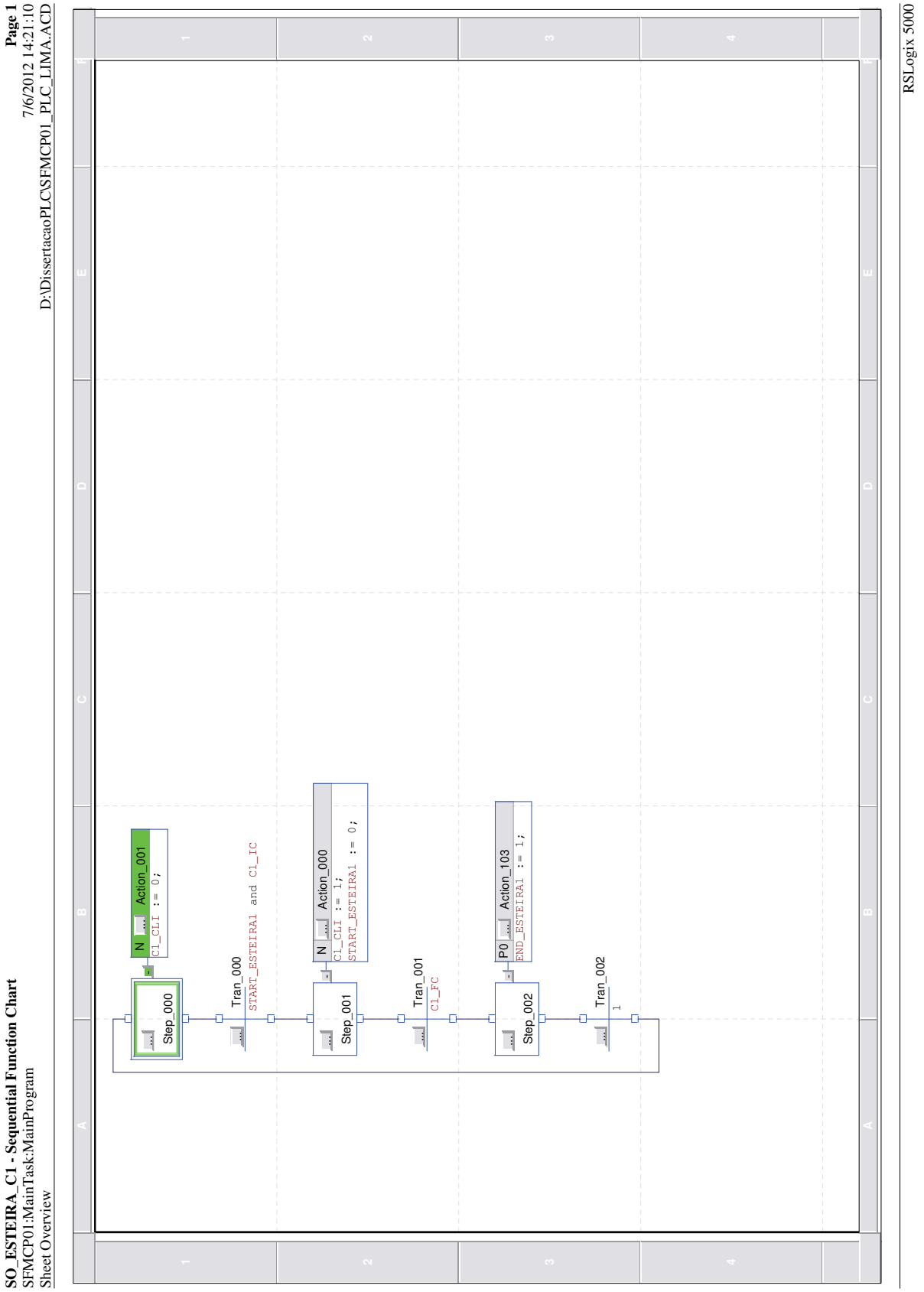

Figura D.6: Lógica Sequência Operacional Esteira C1

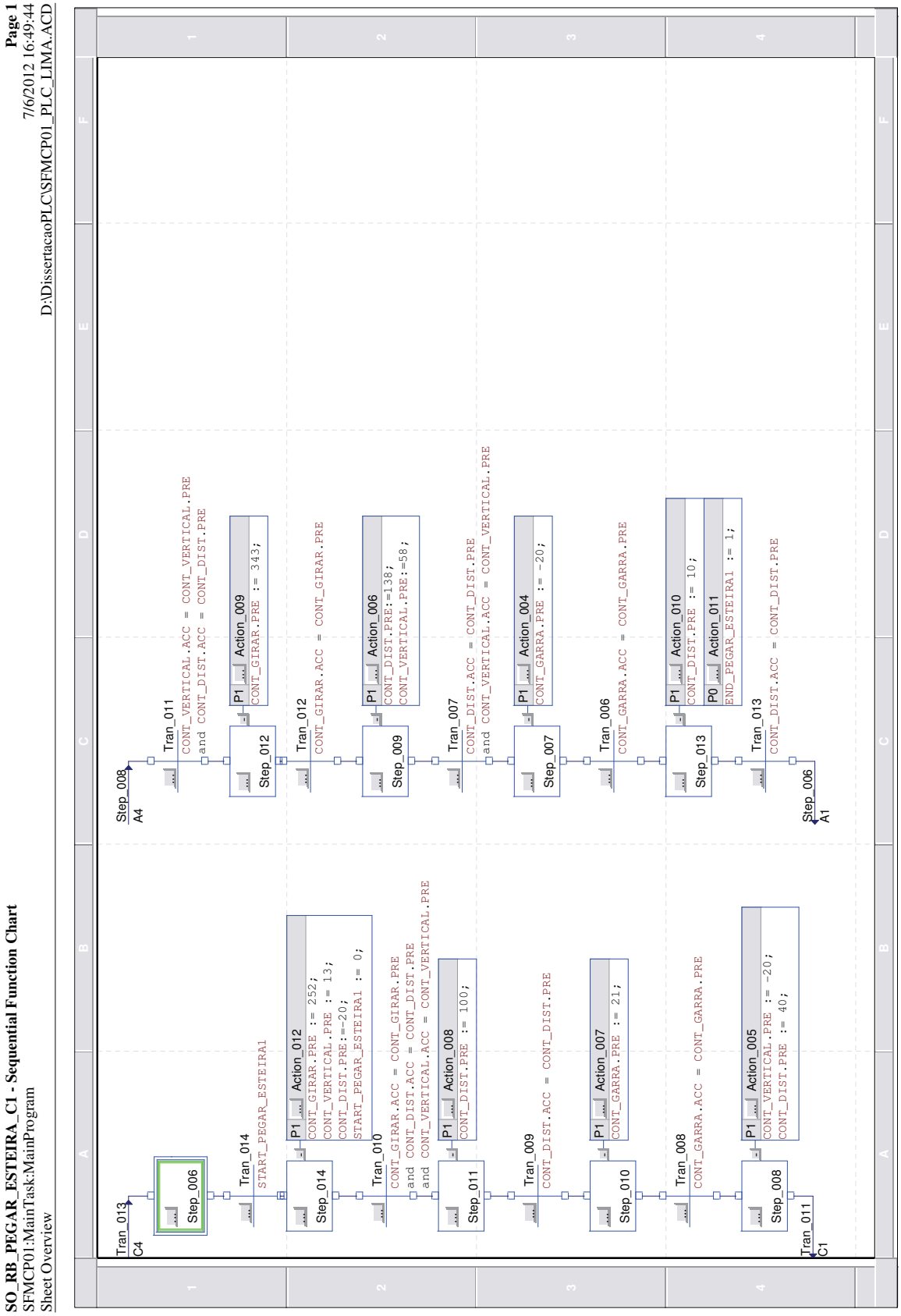

Figura D.7: Lógica Sequência Operacional Robô Pega Peça na Esteira C1

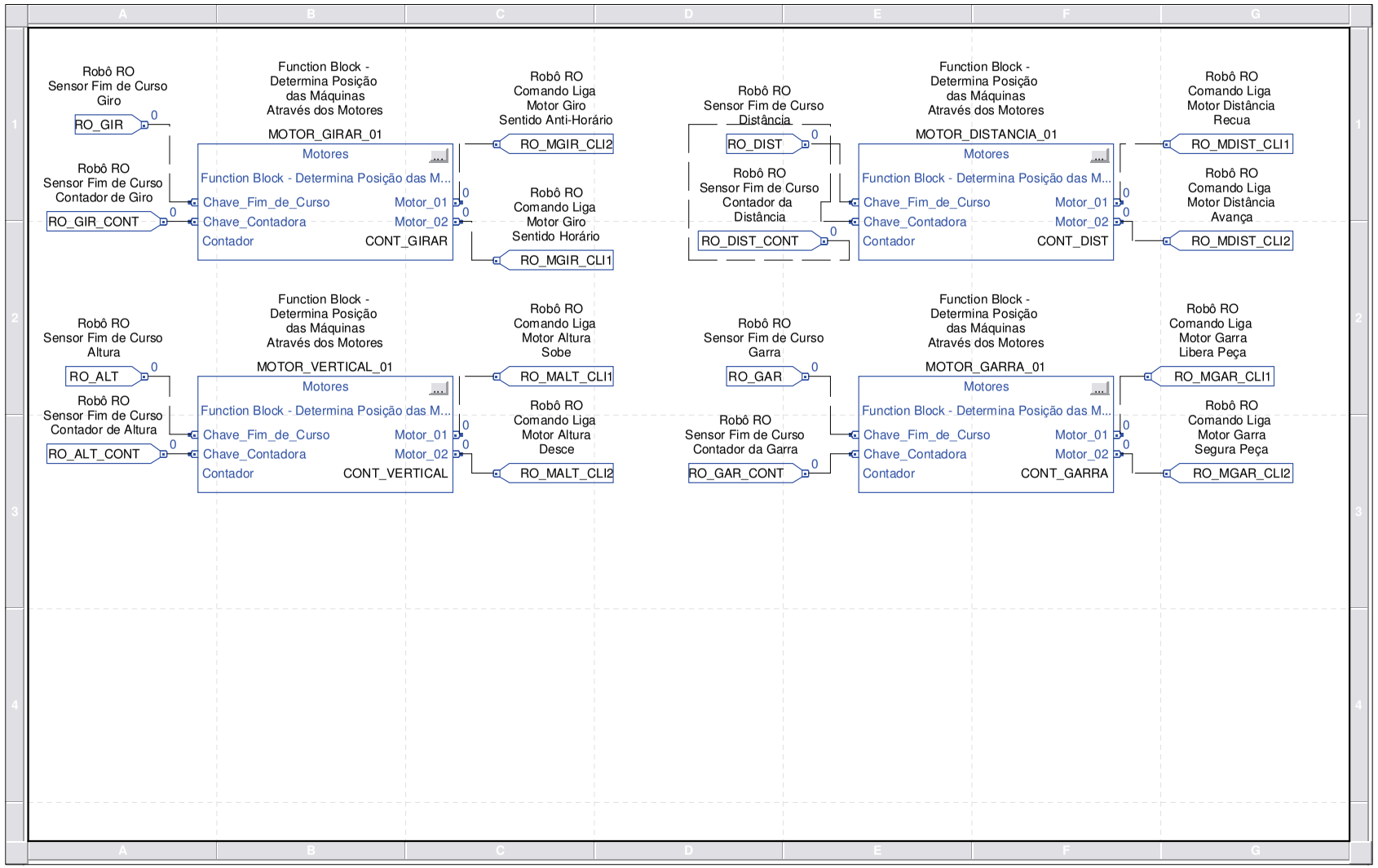

Figura D.8: Lógica Movimentos Robô

Referências Bibliográficas

[Carroll e Long, 1989] Carroll, J. e Long, D. (1989). *Theory of Finite Automata*. Prentice-Hall.

[Cassandras e Lafortune, 1999] Cassandras, C. G. e Lafortune, S. (1999). *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, segunda edição.

[Cury, 2001] Cury, J. E. R. (2001). Teoria de Controle Supervisório de Sistemas a Eventos Discretos. Minicurso do V Simpósio Brasileiro de Automação Inteligente.

[de Queiroz e Cury, 2000] de Queiroz, M. H. e Cury, J. E. R. (2000). Modular Supervisory Control of Large Scale Discrete Event Systems. *In Proceedings of the 5th Workshop on Discrete Event Systems, WODES'00*, 1:103–110.

[Fabian e Hellgren, 1998] Fabian, M. e Hellgren, A. (1998). Plc-based implementation of supervisory control for discrete event systems. *37th IEEE Conference on Decision and Control*, 3:3305–3310.

[Feng e Wonham, 2006] Feng, L. e Wonham, W. M. (2006). TCT: A Computation Tool for Supervisory Control Synthesis. *Proceedings of the 8th International Workshop on Discrete Event Systems, WODES'06*, 1:388–389.

[Ferigollo et al., 2011] Ferigollo, C., Torrico, C. R. C., e Leal, A. B. (2011). Análise de Desempenho das Abordagens Monolítica e Modular Local da Teoria de Controle Supervisório Implementada em Microcontroladores. *10º Simpósio Brasileiro de Automação Inteligente*, 10:1238–1243.

[HPSim, 2001] HPSim (2001). Ferramenta Computacional para Modelagem e Simulação de Redes de Petri. Disponível em: www.winpesim.de/3.html.

[Jones, 1998] Jones, A.H. e Uzam, M. (1998). A formal technique for the synthesis of petri net supervisors for discrete events systems. *UKACC International Conference on Control*, 455:845–852.

[Leal et al., 2009] Leal, A. B., Cruz, D. L. L., e Hounsell, M. S. (2009). Supervisory Control Implementation into Programmable Logic Controllers.

[Lima II, 2002] Lima II, E. J. (2002). *Uma Metodologia para a Implementação através de CLPS de Controle Supervisório de Células de Manufatura Utilizando Redes de Petri*. Dissertação de Mestrado, Universidade Federal da Bahia.

[Moody e Antsaklis, 1998] Moody, J. O. e Antsaklis, P. J. (1998). *Supervisory Control of Discrete Event Systems Using Petri Nets*. Kluwer Academic Publishers.

[Neto, 2011] Neto, M. M. A. (2011). Implementação das Sequências Operacionais no Sistema Flexível de Manufatura. Relatório técnico, Universidade Federal de Minas Gerais.

[Pena et al., 2010] Pena, P. N., Cury, J. E. R., Cunha, A. E. C., e Lafortune, S. (2010). Metodologia e Ferramenta de Apoio ao Teste de Não-conflito no Controle Modular de Sistemas a Eventos Discretos. *Revista Controle & Automação*, 21:58–68.

[Queiroz, 2002] Queiroz, M. H. e Cury, J. E. R. (2002). Controle Supervisório Modular de Sistemas de Manufatura. *Revista Controle & Automacão*, 13:123–133.

[Queiroz, 2004] Queiroz, M. H. (2004). *Controle Supervisório Modular e Multitarefa de Sistemas Compostos*. Tese de Doutorado, Universidade Federal de Santa Catarina.

[Queiroz e Cury, 2002] Queiroz, M. H. e Cury, J. E. R. (2002). Synthesis and Implementation of Local Modular Supervisory Control for a Manufacturing Cell.

[Ramadge e Wonham, 1989] Ramadge, P. J. G. e Wonham, W. M. (1989). The Control of Discrete Event Systems. *Proceedings of the IEEE, Special Issue on Discrete Event Dynamic Systems*, 77:81–98.

[Santos, 2007] Santos, H. G. (2007). *Desenvolvimento de um Supervisório Modular para uma Célula Flexível de Manufatura*. Dissertação de Mestrado, Universidade Federal de Santa Catarina.

[Su e Wonham, 2004] Su, R. e Wonham, W. (2004). Supervisor Reduction for Discrete-Event Systems. *Discrete Event Dynamic Systems: Theory and Applications*, 14:31–53.

[Vieira, 2007] Vieira, A. D. (2007). *Método de Implementação do Controle de Sistemas a Eventos Discretos com Aplicação da Teoria de Controle Supervisório*. Tese de Doutorado, Universidade Federal de Santa Catarina.

[Wonham e Ramadge, 1988] Wonham, W. e Ramadge, P. (1988). Modular Supervisory Control of Discrete Event Systems. *Control, Signals Systems*, 1:13–30.

[XPTCT, 2009] XPTCT (2009). Ferramenta Computacional para Síntese de Controle Supervisório para Sistemas a Eventos Discretos. Disponível em: www.control.utoronto.ca/people/profs/wonham/wonham.html.

[Yamalidou et al., 1995] Yamalidou, K., Moody, J. O., Lemmon, M., e Antsaklis, P. J. (1995). Feedback Control of Petri Nets Based on Place Invariants. *Automática*, 32:15–28.