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

Formação: Mestrado em Engenharia Elétrica

DISSERTAÇÃO DE MESTRADO OBTIDA POR

Jeferson Luiz Curzel

SÍNTESE E IMPLEMENTAÇÃO DE CONTROLE SUPERVISÓRIO EM UMA CÉLULA FLEXÍVEL DE MANUFATURA DIDÁTICA

Apresentada em 06/06/2008 Perante a Banca Examinadora:

Dr. André Bittencourt Leal – Presidente (CCT/UDESC)
Dr. Eduardo Alves Portela Santos (PUC/PR)
Dra. Tatiana Renata Garcia (IST/SOCIESC)
Dr. Marcelo Teixeira dos Santos (BRASILMATICS)
Dr. Marcelo da Silva Hounsell (CCT/UDESC)

UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC

CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT

DEPARTAMENTO DE ENGENHARIA ELÉTRICA – DEE

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA - PPGEEL

DISSERTAÇÃO DE MESTRADO

Mestrando: JEFERSON LUIZ CURZEL – Engenheiro Eletricista

Orientador: Prof. Dr. ANDRÉ BITTENCOURT LEAL

CCT/UDESC - JOINVILLE

SÍNTESE E IMPLEMENTAÇÃO DE CONTROLE SUPERVISÓRIO EM UMA CÉLULA FLEXÍVEL DE MANUFATURA DIDÁTICA

DISSERTAÇÃO APRESENTADA PARA OBTENÇÃO DO TÍTULO DE MESTRE EM ENGENHARIA ELÉTRICA DA UNIVERSIDADE DO ESTADO DE SANTA CATARINA, CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT, ORIENTADA PELO PROF. DR. ANDRÉ BITTENCOURT LEAL.

Joinville

2008

JEFERSON LUIZ CURZEL

SÍNTESE E IMPLEMENTAÇÃO DE CONTROLE SUPERVISÓRIO EM UMA CÉLULA FLEXÍVEL DE MANUFATURA DIDÁTICA

'Esta dissertação foi julgada adequada para obtenção do Título de Mestre em Engenharia Elétrica – Automação Industrial e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade do Estado de Santa Catarina'.

Banca Examinadora:

Orientador:

_____
Prof. Dr. André Bittencourt Leal
Universidade do Estado de Santa Catarina – UDESC

Membros:

_____
Prof. Dr. Eduardo Alves Portela Santos
Pontifícia Universidade Católica do Paraná – PUCPR

_____
Prof[a]. Dr[a]. Tatiana Renata Garcia
IST - SOCIESC

_____
Prof. Dr. Marcelo Teixeira dos Santos
Empresa Brasilmatics – Joinville - SC

_____
Prof. Dr. Marcelo da Silva Hounsell
Universidade do Estado de Santa Catarina – UDESC

Joinville, SC, Junho de 2008

FICHA CATALOGRÁFICA

NOME: CURZEL, Jeferson Luiz
DATA DEFESA: 06/06/2008
LOCAL: Joinville, CCT/UDESC
NÍVEL: Mestrado Número de ordem: **05** – CCT/UDESC
FORMAÇÃO: Engenharia Elétrica **ÁREA DE CONCENTRAÇÃO**: Automação de Sistemas
TÍTULO: "Síntese e Implementação de Controle Supervisório em uma Célula Flexível de Manufatura Didática" **PALAVRAS – CHAVE**: Sistemas a Eventos Discretos, Célula Flexível de Manufatura Didática, Controlador Lógico Programável. **NÚMERO DE PÁGINAS**: xiv, 106p.
CENTRO/UNIVERSIDADE: Centro de Ciências Tecnológicas da UDESC
PROGRAMA: Pós-graduação em Engenharia Elétrica – PPGEEL
Nº CAPES: 41002016012P-0
ORIENTADOR: Dr. André Bittencourt Leal
PRESIDENTE DA BANCA: Dr. André Bittencourt Leal **MEMBROS DA BANCA**: Dr. Eduardo Alves Portela Santos, Drª. Tatiana Renata Garcia, Dr. Marcelo Teixeira dos Santos, Dr. Marcelo da Silva Hounsell.

À minha família.

AGRADECIMENTOS

Agradeço a Deus pelo dom da vida, pela saúde e por ter me guiado por mais esta etapa da minha vida.

À minha Esposa, Janaine, pelo amor e carinho incondicionais.

Ao meu Orientador, Prof. André Bittencourt Leal, pela atenção, dedicação e amizade.

Ao Prof. Silas do Amaral, por ter permitido o uso do Laboratório de Robótica para a integração dos equipamentos.

Ao Prof. Márcio Rubens Baumer, por ter cedido o Controlador Lógico Programável para a implementação prática.

Aos colegas Fabiano e Jean, bolsistas do PET, pelo auxílio no desenvolvimento do software.

Aos colegas do CEFET – Centro Federal de Educação Tecnológica – Unidade Joinville, pelo apoio e compreensão dispensados sempre que solicitados.

A todos os amigos, por abdicarem de tão importante tempo da nossa amizade para a realização deste sonho.

RESUMO

Este trabalho apresenta a integração dos equipamentos existentes no Laboratório de Robótica da UDESC/CCT, de forma a compor uma célula flexível de manufatura didática para a qual foi resolvido um problema de controle aplicando-se a Teoria de Controle Supervisório (TCS). A utilização deste formalismo (TCS) para o desenvolvimento do projeto de controladores visa sistematizar o processo, de forma a disseminar o uso da metodologia junto às indústrias. A partir da TCS foram desenvolvidos os modelos dos equipamentos da célula utilizando-se a representação por autômatos e após a síntese de projeto obteve-se um supervisor monolítico minimamente restritivo, o qual define o comportamento do sistema (planta) de acordo com um conjunto de especificações impostas pelo projetista. Uma extensão à teoria clássica, denominada controle modular local, também é aplicada de forma a explorar a modularidade da planta, onde são obtidos os supervisores locais cuja atuação conjunta coordena o sistema global. O controle da célula é feito por um Controlador Lógico Programável (CLP), o qual coordena o funcionamento dos equipamentos com base nas rotinas de supervisão monolítica e modular local. Os supervisores foram implementados no CLP em linguagem LADDER e a programação foi estruturada em blocos, visando uma melhor organização do programa. Para agilizar a implementação do supervisor monolítico no CLP foi criado um *software* de geração automática de código, o qual converte o código do supervisor monolítico diretamente para a linguagem LADDER do CLP.

Palavras-chave: Teoria de Controle Supervisório, Sistemas a Eventos Discretos, Célula Flexível de Manufatura Didática, Controlador Lógico Programável.

ABSTRACT

This work presents the integration of existing equipment in the Laboratory of Robotics of the UDESC/CCT, to compose a didactic flexible manufacturing cell for which it was resolved a problem of control by applying the Theory of Supervisory Control. The use of this formalism for the development of the project aims to systematize the process controllers in order to spread the use of the methodology in the industries. From the Theory of Supervisory Control were developed models of the equipment of the cell using the representation by automata and after the synthesis of project returned to a supervisor monolithic minimally restrictive, which defines the system (plant) in accordance with a set of specifications imposed by the designer. An extension to the classic theory, called local modular control is also applied in order to exploit the modularity of the plant, where they obtained the local supervisors whose joint action coordinates the global system. The control of the cell is made by a Programmable Logic Controller (PLC), which coordinates the operation of equipment based on the routines of supervision monolithic and local modular. The supervisors were implemented in the PLC in LADDER language and the programming has been structured in blocks, better organization of the program. To improve the implementation of monolithic supervisor in the PLC was developed a software for automatic generation of code, which converts the code of monolithic supervisor directly to the LADDER language of the PLC.

Keywords: Theory of Supervisory Control, Discrete Events Systems, Flexible Manufacturing Cell Didactic, Programmable Logic Controller.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

O aumento na complexidade dos sistemas automatizados em função da globalização tem exigido uma evolução constante dos dispositivos de controle, objetivando uma maior qualidade e custos reduzidos de produção. Com isso, funcionalidades têm sido agregadas aos dispositivos de controle, tais como, maior capacidade de processamento, mais memória para o armazenamento de informações e possibilidade de conexão a outros dispositivos. Contudo, esta evolução não é notada na utilização de ferramentas formais para o desenvolvimento do projeto do controle destes sistemas automatizados, principalmente em ambientes industriais.

A utilização de procedimentos sistemáticos para o desenvolvimento de modelos formais para análise, projeto (síntese) e implementação de sistemas de controle é abordada por Cassandras e Lafortune (1999), dentre os quais se destacam: Redes de Petri, Cadeias de Markov, Teoria das Filas, Simulação, Álgebra de Processos e Max-Plus, Lógica Temporal, Autômatos e Linguagens.

Enquanto a maior parte destes modelos baseia-se na inspiração e experiência do projetista, limitando-se à análise de soluções de controle propostas por estes, a Teoria de Controle Supervisório proposta por Ramadge e Wonham (1989) é uma das mais adequadas para o desenvolvimento dos modelos formais de sistemas de controle, pois é dotada de procedimentos de síntese de controladores. No modelo de Ramadge e Wonham (1989) é realizado um processo automático de síntese do controle, ao invés dos procedimentos tradicionais (empíricos).

Conforme Hellgren *et al*. *apud* Vieira (2007, p. 5), “apesar da grande aceitação da Teoria de Controle Supervisório pelo meio acadêmico, havendo diversas extensões à mesma e um número muito grande de publicações com foco em aspectos teóricos, são raras as aplicações industriais. A razão principal para isto é o problema da implementação física”. Segundo os autores há poucas referências de como implementar os supervisores obtidos através da aplicação desta teoria e, no caso da implementação em Controladores Lógico Programáveis (CLPs), a distância entre o mundo dos autômatos baseados em eventos e o mundo do CLP baseado em sinais tem que ser superada. Vieira (2007) cita ainda diversas abordagens desenvolvidas no intuito de superar a distância entre os dois mundos, dentre as quais algumas foram utilizadas como referências também para este trabalho, destacando-se: (Fabian e Helgren, 1998), (Dias, 2005), (Costa, 2005), (Moraes e Leal, 2006), (Teixeira *et al*., 2006), (Carvalho, 2007), (Queiroz, 2004), (Queiroz e Cury, 2002b), dentre outras obras, (Vieira, 2001), (Bouzon *et al.*, 2004), (Santos *et al.*, 2006).

Neste trabalho serão apresentados os resultados relativos ao estudo e aplicação da Teoria de Controle Supervisório, conforme apresentados e discutidos previamente em (Curzel e Leal, 2006) e (Curzel *et al.*, 2006).

1.1 Justificativa

A Teoria de Controle Supervisório (TCS) (Ramadge e Wonham, 1989) possibilita que problemas de controle sejam solucionados de forma sistemática, garantindo assim que o supervisor obtido atenderá as especificações impostas pelo projetista. Além disto, o seu uso facilita alterações no programa do controlador lógico programável (CLP), alterações estas que são necessárias sempre que é feita alguma modificação (inclusão/exclusão de equipamentos ou alteração no *layout*) na célula flexível de manufatura. Entretanto, a teoria supracitada não é conhecida e dominada no âmbito industrial, de forma que, em geral, a resolução de problemas de controle supervisório nas indústrias é feita sem a utilização de um procedimento formal.

Desta forma, tendo em vista a necessidade de um laboratório para a realização de testes práticos relativos à síntese e implementação da TCS na resolução de problemas e a disponibilidade de equipamentos existentes no laboratório de robótica da UDESC/CCT, vislumbrou-se a possibilidade de integração destes equipamentos para compor uma célula flexível de manufatura didática. Esta célula possibilitará que os resultados do estudo e aplicação da TCS nas disciplinas de graduação e pós-graduação do curso de engenharia elétrica possam ser implementados na prática. Com isso, além de estimular os alunos ao trabalho em equipe, será possível contribuir para a difusão da TCS entre os alunos, diminuindo a distância entre a teoria vista em sala de aula com a prática realizada em laboratório, proporcionando aos estudantes a solução de problemas que, em escala reduzida, são fiéis à realidade encontrada nas indústrias.

1.2 Objetivos

Os objetivos deste trabalho são:

- Integrar os equipamentos do Laboratório de Robótica da UDESC/CCT para compor uma célula flexível de manufatura didática;
- Aplicar a Teoria de Controle Supervisório para a resolução de um problema motivador sob a ótica de duas abordagens distintas, a abordagem monolítica e a abordagem modular local, introduzindo um procedimento formal para a solução de problemas de controle supervisório para a célula em questão;

- Fazer a implementação em CLP dos supervisores obtidos nas duas abordagens utilizando-se a linguagem LADDER (IEC, 2003), por ser esta uma linguagem mais difundida no meio industrial.
- Desenvolver um *software* de conversão de código, com a função de converter o código do supervisor monolítico diretamente para a linguagem do CLP. Com isto objetiva-se agilizar o processo de programação do CLP, tornando mais rápidas as alterações no programa do CLP sempre que houver alguma alteração no projeto do supervisor.

1.3 Organização do trabalho

Esta dissertação está estruturada da seguinte forma: no Capítulo 2 é apresentada uma revisão sobre a Teoria de Controle Supervisório proposta por Ramadge e Wonham (1989). Inicialmente são apresentados os conceitos de Sistemas a Eventos Discretos (SEDs) e os formalismos de Linguagens e Autômatos utilizados para a representação de SEDs. Na seqüência, são apresentadas as abordagens Monolítica e Modular Local para o Controle Supervisório de SEDs.

No Capítulo 3 é apresentada a célula flexível de manufatura que foi integrada e utilizada na realização deste trabalho. São descritos os equipamentos que fazem parte da célula e como é feita a integração dos sinais elétricos destes com o CLP, através de um módulo de interface desenvolvido para essa finalidade. Ainda neste capítulo são descritas as rotinas de programação dos robôs didáticos utilizados na célula.

No Capítulo 4 é mostrada a síntese de supervisores da célula de manufatura utilizada, através da modelagem por autômatos, e apresentada a resolução de um problema-exemplo, utilizando-se as abordagens monolítica e modular local vistas no Capítulo 2.

No Capítulo 5 é apresentada primeiramente a proposta de implementação do supervisor monolítico em CLP Siemens Step7-300, em linguagem LADDER, utilizando-se uma estrutura dividida em blocos de programa, bem como o *software* desenvolvido para a geração automática de código para o CLP utilizado. Na seqüência do Capítulo 5 é mostrada a implementação dos supervisores modulares locais para o mesmo CLP, utilizando-se também uma estrutura dividida em blocos de programa, o que torna em ambos os casos, mais fácil o entendimento e alterações no programa.

No último capítulo são comentados os resultados das duas técnicas de implementação do controle supervisório em CLP utilizadas, são apresentadas as contribuições

do trabalho e as propostas para futuros trabalhos, tendo-se como base a célula flexível de manufatura concebida.

Demais informações relativas à utilização do *software* desenvolvido para a geração automática de código para o CLP encontram-se no APÊNDICE A.

2 TEORIA DE CONTROLE SUPERVISÓRIO

Com o crescente avanço da tecnologia e aumento da complexidade dos sistemas automatizados, estes passaram a necessitar de um formalismo específico para a otimização do seu controle. Exemplos de aplicações onde estes sistemas estão presentes são: a automação da manufatura, a robótica, a supervisão de tráfego, a logística, redes de comunicação e de computadores, sistemas operacionais, concepção de *software*, gerenciamento de base de dados, otimização de processos distribuídos, entre outros.

A semelhança entre estes sistemas é dada em função da natureza da ocorrência de mudanças no ambiente a sua volta, o que se percebe pela recepção de estímulos instantâneos e discretos no tempo, os quais são denominados eventos. Aos sistemas com estas características dá-se a denominação de Sistemas a Eventos Discretos, ou simplesmente, SEDs.

A modelagem e controle de SEDs é uma área de pesquisa de grande interesse nas universidades e no meio industrial, sendo estimulada pela diversidade de aplicações, como é o caso dos modernos sistemas de manufatura. Destaca-se ainda a aplicabilidade da teoria de controle de sistemas a eventos discretos proposta por Ramadge e Wonham (1989) pelo seu potencial para desenvolvimento de ferramentas teóricas (Cury, 2001).

O presente capítulo tem como objetivo apresentar os fundamentos da teoria de controle supervisório, introduzida por Ramadge e Wonham (1989). Primeiramente é abordada a classe de sistemas do qual trata este trabalho, denominada Sistemas a Eventos Discretos (SEDs). Em seguida, são apresentados os formalismos para a representação de SEDs, a abordagem proposta por Ramadge e Wonham (1989) e uma extensão desta teoria, especificamente a abordagem proposta por Queiroz e Cury (2000). A partir da Seção 2.2 deste capítulo são apresentados os formalismos utilizados nas abordagens mencionadas.

2.1 Sistemas a Eventos Discretos

Um sistema a eventos discretos (SED) é um sistema de estados discretos, dirigido a eventos, isto é, sua evolução de estado depende da ocorrência de eventos discretos assíncronos no tempo (Cassandras e Lafortune, 1999).

Segundo Cury *apud* Attié (1998, p. 54),

> Quando o espaço de estados de um sistema é naturalmente descrito por um conjunto discreto, e as transições de estado são observadas somente em pontos discretos do tempo, associam-se estas transições a eventos. O conceito de evento é um desses conceitos primitivos, cuja compreensão deve ser deixada à intuição, mais do que a uma exata definição. Não se pode, porém, deixar de enfatizar que

> um evento deve ser pensado como de ocorrência instantânea e como causador de uma transição no valor (discreto) do estado do sistema.

Desta definição são extraídas duas propriedades principais dos SEDs:

- O espaço de estados é discreto;
- A transição de estados é dirigida por eventos.

Um evento pode ser identificado como uma ação específica, como por exemplo, o acionamento de um botão, a parada de uma máquina, ou pode ainda, ser o resultado de várias condições que ocorrem em um dado instante. O mesmo evento pode ter efeitos diferentes (pode levar a estados distintos), dependendo do estado em que ocorre. Ramadge e Wonham (1989) classificam ainda os eventos em dois tipos: os eventos controláveis, cuja ocorrência pode ser desabilitada pela ação de controle, como por exemplo, o início de uma atividade ou a parada de uma esteira, e os eventos não controláveis, cuja ocorrência não pode ser desabilitada pela ação de controle, como por exemplo, a ativação de um sensor. Esta classificação é importante para a modelagem dos subsistemas físicos que serão apresentados no Capítulo 4.

2.2 Formalismos para representação de SEDs

Nesta seção serão apresentados os conceitos básicos da teoria de linguagens e autômatos utilizados como formalismos para a representação de SEDs. Para um maior detalhamento dos assuntos apresentados poderão ser consultadas as seguintes referências: (Cassandras e Lafortune, 1999), (Cury, 2001) e (Wonham, 2004).

2.2.1 Linguagens

O comportamento lógico de um SED pode ser modelado a partir de linguagens, que pode ser visto como um modelo comportamental externo do sistema, pois se baseia na descrição de uma seqüência de eventos. A palavra linguagem vem do fato que se pode pensar em um conjunto de eventos como um *alfabeto* e seqüências de eventos como *palavras* (Cassandras e Lafortune, 1999).

Define-se que uma linguagem L definida sobre um alfabeto Σ, é um conjunto de cadeias ou palavras formadas por símbolos pertencentes a esse alfabeto Σ. Define-se ainda Σ^* como o conjunto de todas as cadeias finitas de elementos do conjunto Σ, incluindo a cadeia vazia ε (Cury, 2001).

Algumas operações podem ser executadas sobre linguagens, tais como as operações sobre conjuntos.

Deve-se considerar para as operações sobre linguagens que se *tuv = s*, com *t, u, v* ∈ Σ*, então:

- *t* é chamado *prefixo* de *s*
- *u* é chamada uma *subcadeia* de *s*
- *v* é chamado *sufixo* de *s*

São três as operações consideradas sobre linguagens:

1. *Concatenação:*

Considerando as linguagens L1 e L2 definidas como L1,L2 ⊆ Σ*, a concatenação de uma cadeia de L1 com uma cadeia de L2, denotada por L1L2, é definida por:

$$L1L2 = \{s \in \Sigma^* : (s = s1s2)\ ,\ (s1 \in L1)\ e\ (s2 \in L2)\} \tag{2.1}$$

Desta forma, uma cadeia está em L1L2 se ela pode ser escrita como a concatenação de uma cadeia de L1 com uma cadeia de L2.

2. *Prefixo-Fechamento:*

Seja uma linguagem L ∈ Σ* , então o prefixo-fechamento de L, denotado por $\overline{L}$, é definido por:

$$\overline{L} = \{s \in \Sigma^* : \exists t \in \Sigma^* (st \in L)\} \tag{2.2}$$

Uma linguagem L é prefixo-fechada se qualquer prefixo de qualquer cadeia de L é também cadeia de L.

3. *Fechamento-Kleene:*

Seja uma linguagem L ⊆ Σ*, o fechamento-Kleene de L, denotado por L* é definido por:

$$L^* = \{\varepsilon\} \cup L \cup LL \cup LLL \cup .... \tag{2.3}$$

Uma cadeia de L* é formada pela concatenação de um número finito de cadeias de L, incluindo a cadeia vazia ε.

Como exemplo para as operações mostradas acima, considera-se o alfabeto Σ = {*a, b, c*}, e as linguagens L1 = {*ε, a, abb*} e L2 = {*c*} definidas sobre Σ. L1 e L2 não são prefixo-fechadas, pois nota-se que *ab* ∉ L1 e que ε ∉ L2. Assim:

- $L1L2 = \{c, ac, abbc\}$
- $\overline{L1} = \{\varepsilon, a, ab, abb\}$
- $\overline{L2} = \{\varepsilon, c\}$

- $L2^{*} = \{\varepsilon, c, cc, ccc, ...\}$

2.2.2 Representação de SEDs por linguagens

O comportamento de um sistema a eventos discretos pode ser descrito através de um par de linguagens. A evolução seqüencial do SED, ou seu comportamento lógico, pode ser modelado através de *D = (L, Lm)*.

No modelo *D*, $L \subseteq \Sigma^{*}$ é a linguagem prefixo fechada que descreve o comportamento gerado pelo sistema, ou seja, o conjunto de todas as cadeias de eventos fisicamente possíveis de ocorrerem no sistema. Após partir do estado inicial e percorrer uma determinada trajetória em seu espaço de estados, um SED poderá completar uma ou mais tarefas. As seqüências de eventos que levam às tarefas completas formam também uma linguagem. Assim, no modelo *D*, $Lm \subseteq L$ é a linguagem que descreve o comportamento marcado do sistema, ou seja, o conjunto de cadeias em *L* que correspondem a tarefas completas que o sistema pode realizar.

Duas propriedades das linguagens *L e Lm* são sintetizadas para representar um SED:

1. $L \supseteq Lm$, ou seja, o comportamento gerado contém o comportamento marcado de um SED;
2. $L = \overline{L}$, ou seja, o comportamento gerado de um SED é prefixo-fechado.

Como exemplo, podemos considerar um sistema representado por uma esteira, onde o conjunto de eventos associado é representado por Σ = {a,b}, onde ‘a’ representa o início de operação da esteira e ‘b’ representa o fim de operação da mesma. Neste caso, a linguagem *L* que corresponde ao comportamento gerado pela esteira consiste de todas as seqüências de eventos que iniciam com ‘a’ e terminam com ‘b’. É importante observar que um evento ‘b’ não ocorre se não houver acontecido um evento ‘a’ anteriormente e que $\varepsilon \in L$ corresponde à situação da esteira no seu estado inicial, onde nenhum evento ocorreu ainda. Assim, consideram-se tarefas completadas da esteira todas as cadeias que levam ao estado de fim de operação ou repouso. Desta forma, pode-se afirmar que *Lm* consiste de todas as cadeias *L* que terminam com ‘b’ e também a cadeia ε. Assim:

$$L = \{\varepsilon, a, ab, abab, ababa, abababab, ...\} \tag{2.4}$$

e

$$Lm = \{\varepsilon, ab, abab, abababab, ...\} \tag{2.5}$$

Uma linguagem pode ser uma das maneiras formais de descrição comportamental para um SED, podendo representar todas as possíveis seqüências de eventos (L) ou as tarefas que o sistema pode completar (Lm).

Contudo, a descrição de uma linguagem natural como vista anteriormente pode ser uma tarefa pouco prática, pois pode especificar todas as possíveis seqüências de eventos que um SED pode gerar. Uma alternativa é a utilização de expressões regulares e autômatos para o formalismo de SEDs.

As expressões regulares fornecem um meio de descrição de linguagens, obtidas pela aplicação de um conjunto de regras de composição. Assim, para um dado alfabeto Σ, define-se recursivamente uma expressão regular da seguinte forma:

a) $\varnothing$ é uma expressão regular que representa a linguagem vazia;
b) ε é uma expressão regular que denota a linguagem $\{\varepsilon\}$;
c) σ é uma expressão regular representando $\{\sigma\}$ $\forall\sigma \in \Sigma$; (para todo $\sigma \subset \Sigma$)
d) Se *r* e *s* são expressões regulares então *rs*, *r**, *s**, *(r+s)** são expressões regulares;
e) Toda expressão regular é obtida pela aplicação das regras acima um número finito de vezes.

Retornado ao exemplo da esteira visto anteriormente, *L* e *Lm* podem ser escritos da seguinte forma:

$$L = (ab)^*(\varepsilon + a) \tag{2.6}$$

e

$$Lm = (ab)^* \tag{2.7}$$

Entretanto, embora a teoria de linguagens seja atrativa para apresentar aspectos da modelagem e para discutir propriedades de SEDs, ela não é conveniente para realizar a verificação de propriedades ou a síntese do controlador (Cassandras e Lafortune, 1999). Além disso, a representação através de linguagens e expressões regulares é limitada computacionalmente. Para resolver este problema utiliza-se a representação de SEDs através de autômatos.

2.2.3 Autômatos para representação de SEDs

Nesta seção serão apresentados os principais conceitos relacionados aos autômatos de estados finitos e as relações entre os autômatos e as linguagens. Também será apresentada a representação de um SED utilizando-se autômatos.

Um autômato determinístico de estados finitos é uma quíntupla $G = (X, \Sigma, f, x_o, X_m)$, onde:

- X é o conjunto finito de estados do autômato;
- Σ é o conjunto de eventos que definem o alfabeto;
- $f : X \times \Sigma \rightarrow X$ é a função de transição de estados e em geral é parcial;
- x_o é o estado inicial do autômato;
- X_m é o conjunto de estados finais (marcados) onde $X_m \subseteq X$.

Um autômato pode ser representado graficamente como um grafo dirigido, onde os círculos representam os estados e os arcos entre os estados indicam as transições ou eventos. O estado inicial é identificado através de uma seta apontando para ele e os estados finais (marcados) são representados com círculos duplos. Arcos seccionados por uma pequena linha transversal indicam que o evento é controlável (Cury, 2001).

Na Figura 2.1 é mostrado um autômato determinístico, chamado de G, cuja descrição formal é dada da seguinte forma:

- X = { A, B, C}
- Σ = {x, y, z}
- A função parcial de transição de estados é: f(A,x) = A, f(A,z) = C, f(B,y) = B, f(B,x) = A, f(C,y) = C, f(C,x) = f(C,z) = B
- x_o = {A}
- X_m = {A,B}

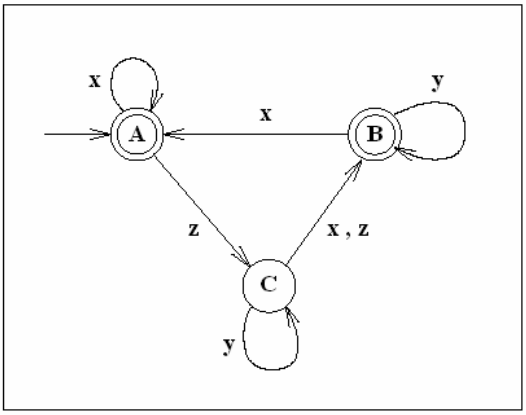


Figura 2.1 – Autômato de estados finitos

O autômato G da Figura 2.1 pode ser interpretado como um dispositivo que inicia a partir do estado inicial x_o e permanece nesse estado até a ocorrência de um evento *z* e este processo continua baseado nas transições definidas em *f*. Este autômato está associado a duas linguagens, a linguagem gerada L(G) que representa todas as cadeias que podem ser seguidas no autômato, a partir do estado inicial, e a linguagem marcada Lm(G), que considera todas as

cadeias que partindo do estado inicial chegam a um estado marcado, incluindo, neste caso, o próprio estado inicial que é marcado.

Assim, um SED pode ser modelado por um autômato G, onde L(G) é o comportamento gerado pelo sistema e Lm(G) é o conjunto de tarefas completas do sistema (CURY, 2001).

De forma geral, um autômato de estados finitos pode ter estados inacessíveis, que jamais podem ser alcançados a partir do estado inicial. Formalmente, um estado $x \in X$ é dito ser acessível se $x = f(x_o, u)$ para algum $u \in \Sigma^*$. G é dito ser acessível se x é acessível para todo $x \in X$. A componente acessível Gac de um autômato G é obtida pela eliminação de seus estados não acessíveis e das transições associadas a tais estados.

De outra forma, G é dito ser co-acessível ou não bloqueante, se cada cadeia $u \in L(G)$ pode ser completada por algum $w \in \Sigma^*$ tal que $uw \in Lm(G)$, ou seja, se cada cadeia $u \in L(G)$ for um prefixo de uma cadeia em Lm(G). Em palavras, um autômato é co-acessível se a partir de qualquer de seus estados, existir ao menos um caminho que leve a um estado marcado.

É possível descrever a condição de co-acessibilidade de um autômato pela equação:

$$L(G) = \overline{Lm(G)} \tag{2.8}$$

A equação 2.1 permite definir a idéia de ausência de bloqueio em um SED. Um SED é dito não bloqueante se, e somente se, satisfaz as condições da equação apresentada. Caso contrário, o SED é dito bloqueante (Cury, 2001). A condição de bloqueio corresponde a cadeias de eventos geradas pelo sistema a partir das quais não se pode completar alguma tarefa.

O autômato apresentado na Figura 2.2 é co-acessível (não bloqueante), pois o estado *B* representa uma tarefa completada do sistema, o que caracteriza o não bloqueio.

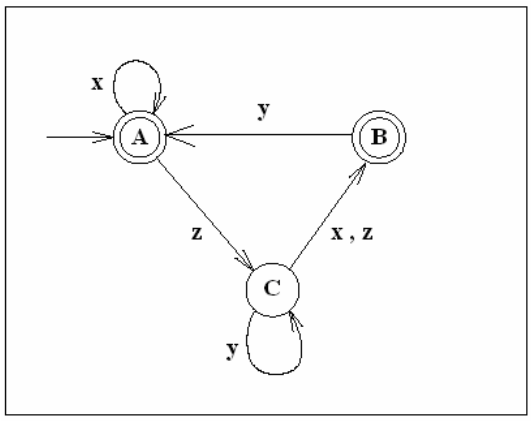


Figura 2.2 – Autômato não bloqueante

Já na Figura 2.3 é mostrado um autômato onde ocorre o bloqueio do sistema no estado *C*, ainda que seja possível a ocorrência do evento *y*.

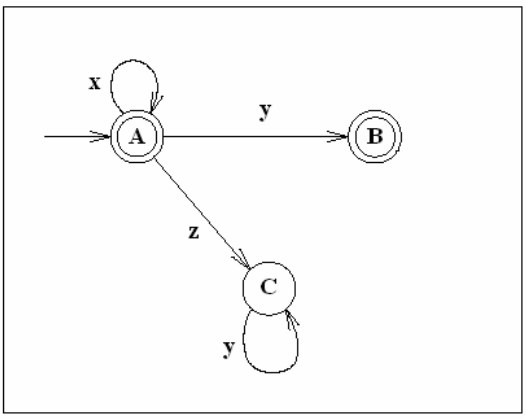


Figura 2.3 – Autômato bloqueante

2.2.4 Composição de autômatos

Um SED pode ser visto como a composição de subsistemas sendo regidos por um controlador que restringe o comportamento do sistema de acordo com as especificações impostas no projeto. Assim, a composição dos subsistemas é uma operação que permite a representação de um autômato com todas as seqüências possíveis e tarefas que podem ser completadas pelo sistema. Cabe ressaltar que para sistemas de grande porte a composição dos subsistemas pode ser de grande complexidade, e que qualquer inclusão ou alteração destes no projeto acarreta na reconstrução do modelo como um todo. Cury (2001) define essa metodologia de modelagem como sendo uma abordagem global.

Para a composição de autômatos, é utilizada a composição paralela, freqüentemente chamada de composição síncrona (Cury, 2001). Assim, dados dois autômatos $G1 = (X1, \Sigma 1, f1, x_{o1}, X_{m1})$ e $G2 = (X2, \Sigma 2, f2, x_{o2}, X_{m2})$, a composição síncrona G1||G2 é definida por:

$$\text{G1}||\text{G2} = Ac(X_1 \times X_2,\ \Sigma_1 \cup \Sigma_2,\ f_{1||2},\ (x_{o1}, x_{o2}),\ X_{m1} \times X_{m2}) \tag{2.9}$$

onde *Ac* é a componente acessível e

$$f_{1||2} : (X_1 \times X_2) \times (\Sigma_1 \cup \Sigma_2) \rightarrow (X_1 \times X_2) \tag{2.10}$$

ou seja:

$$f_{1\|2}((x1, x2), \sigma) = \left\{ \begin{array}{c} (f1(x1, \sigma),\ f2(x2, \sigma)) \quad se \quad \sigma \in \sum 1 \cap \sum 2 \quad e \quad \sigma \in \sum 1(x1) \cap \sum 2(x2) \\ (f1(x1, \sigma), x2) \quad se \quad \sigma \in \sum 1 \quad e \quad \sigma \notin \sum 2 \quad e \quad \sigma \in \sum 1(x1) \\ (x1,\ f2(x2, \sigma)) \quad se \quad \sigma \in \sum 2 \quad e \quad \sigma \notin \sum 1 \quad e \quad \sigma \in \sum 2(x2) \\ indefinida \qquad caso\ contrário \end{array} \right\} \tag{2.11}$$

A seguir apresenta-se um exemplo no intuito de ilustrar a operação de composição síncrona. Sejam os autômatos G1 e G2 mostrados na Figura 2.4.

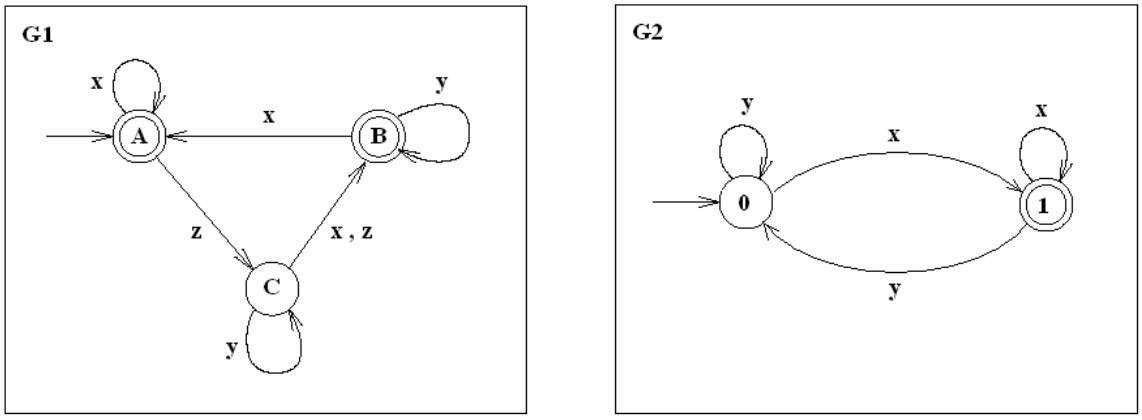


Figura 2.4 – Autômatos G1 e G2

Assim, de acordo com o exposto por Cury (2001), um evento comum a Σ1 e Σ2 só pode ser executado sincronamente nos dois autômatos. Os demais eventos ocorrem assincronamente, ou seja, de modo independente em cada autômato. Assim, pode-se interpretar G1||G2 como a ação coordenada dos dois autômatos. O autômato resultante desta sincronização é mostrado na Figura 2.5.

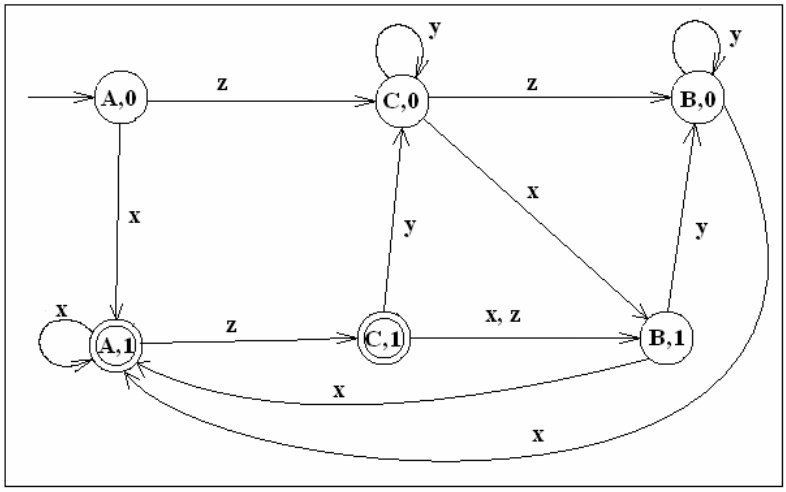


Figura 2.5 – Composição síncrona G1||G2

Como já ressaltado anteriormente, a composição síncrona pode ser uma tarefa difícil, mas existem ferramentas computacionais desenvolvidas para estas atividades. Neste trabalho utilizou-se o *software* Grail (Reiser *et al.*, 2006) e informações relativas a sua aplicação são

encontradas em (Cury, 2001) e (Garcia, 2006). O *software* CTCT (Wonham, 2004) também é utilizado como ferramenta para autômatos, porém com menor ênfase neste trabalho.

Concluída a apresentação dos formalismos utilizados para a representação de SEDs, apresenta-se a seguir as metodologias utilizadas para a síntese do controle de SEDs.

2.3 Controle Supervisório de SEDs

A Teoria de Controle Supervisório proposta por Ramadge e Wonham (1989) é uma das mais adequadas para o desenvolvimento dos modelos formais de sistemas de controle, pois é dotada de procedimentos de síntese automática de controladores para SEDs. Nesta teoria é feita uma distinção entre o sistema a ser controlado (*planta*), que corresponde, em geral, a um conjunto de equipamentos (*subsistemas*), e o controle do sistema, que é chamado de *supervisor*. O papel do supervisor é exercer uma ação de controle minimamente restritiva sobre os subsistemas, de modo que se comportem de acordo com um conjunto de *especificações*, coordenando assim o funcionamento do sistema como um todo. A função de uma especificação é, então, limitar o comportamento do sistema em termos de seqüências de eventos indesejados, ou seja, certos estados do sistema que não são desejados devem ser evitados. Estes estados podem ser causadores de bloqueio ou então são fisicamente inadmissíveis, como por exemplo, a colisão de um robô com um veículo auto guiado ou a tentativa de colocar uma peça em um armazém cheio (Cury, 2001). As especificações, assim como os subsistemas, também são representadas por autômatos de estados finitos.

A abordagem de Ramadge e Wonham para o controle supervisório de SEDs é também denominada Abordagem Monolítica, pois tem como objetivo projetar um único controlador para o sistema. Porém, como já exposto na Seção 2.2.4, esta tarefa pode acarretar em problemas como um elevado número de estados da planta e/ou do supervisor (explosão de estados) e o consumo de uma memória muito grande por parte do dispositivo de controle, por exemplo, um CLP.

Como alternativa, foi desenvolvida uma extensão à abordagem monolítica, chamada de Controle Modular Local (Queiroz e Cury, 2000), na qual é explorada a modularidade das especificações de controle e a própria modularidade da planta. Assim, são definidos os supervisores locais, onde a atuação conjunta dos supervisores coordena o comportamento do sistema como um todo.

Nas próximas seções apresentam-se as duas abordagens utilizadas como base conceitual para o desenvolvimento do Capítulo 4.

2.4 Abordagem Monolítica

Nesta abordagem, a composição dos comportamentos de cada subsistema isolado é modelado por um autômato de estados finitos, que pode ser chamado de *planta*. Este autômato é denominado G, sendo representado por uma linguagem gerada L(G) e por uma linguagem marcada Lm(G). Assim, por representar todas as possibilidades para a planta, G pode conter cadeias de eventos indesejáveis. O autômato G, portanto, modela o comportamento do SED sem nenhuma ação de controle.

Para a realização do controle monolítico, é necessário introduzir um agente de controle no sistema, que atue de acordo com um conjunto de restrições. Estas restrições são chamadas de *especificações de controle* e têm uma natureza permissiva em relação à ação de controle, ou seja, de acordo com as especificações impostas pelo projetista do SED, somente aquelas ações que levam a um estado proibido ou a uma seqüência de eventos indesejada são proibidas.

A ação de controle tem, então, o objetivo de habilitar e desabilitar certos eventos, conforme a seqüência de eventos observados na planta. A essa ação de controle dá-se o nome de *supervisor*, denominado S.

Desta forma, o supervisor S interage com a planta G observando os eventos ocorridos e define, de acordo com o estado atual da planta, quais eventos fisicamente possíveis são habilitados. A Figura 2.6 ilustra a estrutura de controle em malha fechada de uma planta G sob ação do supervisor S.

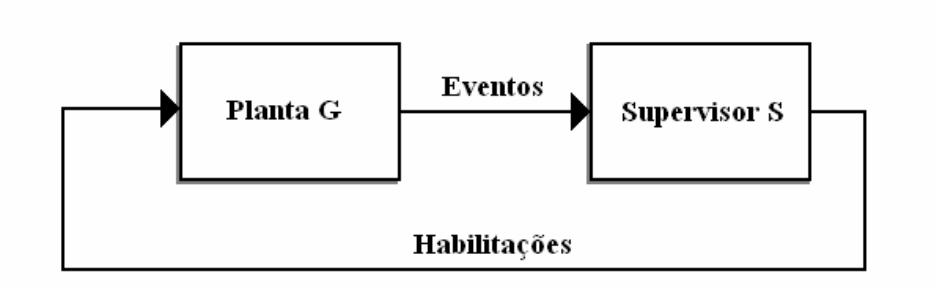


Figura 2.6 – Controle de SED em malha fechada

Tomando-se a planta G, inicialmente deve-se particionar o conjunto de eventos Σ em eventos controláveis Σc, cuja ocorrência pode ser desabilitada pela ação de controle, e eventos não controláveis Σu, cuja ocorrência não pode ser desabilitada pela ação de controle. Sobre estes eventos particionados, define-se, para uma entrada de controle $\gamma \in \Gamma$, uma estrutura de controle Γ para G, tal que:

$$\Gamma = \{\gamma \in 2^{\Sigma} : \gamma \supseteq \Sigma u\} \qquad (2.12)$$

onde a condição $\gamma \supseteq \Sigma u$ indica que os eventos não controláveis não podem ser desabilitados.

Formalmente, um supervisor S é um mapeamento S: L $\rightarrow \Gamma$ que especifica um conjunto de eventos habilitados para cada cadeia possível de eventos gerados $w \in$ L. O funcionamento do sistema controlado pode ser descrito pelo autômato resultante da composição síncrona S||G. Assim, somente as transições permitidas tanto no sistema controlado G, como no supervisor S são permitidas.

O comportamento do sistema em malha fechada é então dado por:

$$L(S/G) = L(S||G) \quad e \quad Lm(S/G) = Lm(S||G) \tag{2.13}$$

Diz-se que um supervisor S é não bloqueante para G se garantir o não bloqueio do sistema em malha fechada, isto é, se $L(S/G) = \overline{Lm(S/G)}$.

Assim, pode-se representar um supervisor por um autômato S, definido sobre o mesmo alfabeto Σ da planta G, cujas mudanças de estado são regidas pela ocorrência de eventos na planta G. A ação de controle de S definida para cada estado do autômato, consiste em desabilitar em G os eventos que não possam ocorrer em S após uma cadeia de eventos observada.

Cury (2001) ressalta ainda que o conceito de controlabilidade de linguagens é essencial para a solução do controle supervisório. Assim, dada uma planta G, com comportamento (L(G) e Lm(G)) e estrutura de controle Γ, definidos sobre o conjunto de eventos Σ e a linguagem K $\subseteq$ L(G), K é dita ser controlável se:

$$\overline{K}\Sigma u \cap L(G) \subseteq \overline{K} \tag{2.14}$$

No caso em que a linguagem K que especifica o comportamento desejado não é controlável, é possível projetar uma aproximação de K, que é chamada de máxima linguagem controlável contida em K, denotada por supC(K). O supervisor que implementa esta linguagem é chamado supervisor não bloqueante e minimamente restritivo ou supervisor ótimo, tal que Lm(S/G) = supC(K), (Ramadge e Wonham, 1989).

Considerando o exposto, as etapas de desenvolvimento do projeto são as seguintes:

1. Identificação do conjunto de subsistemas envolvidos no problema;
2. Construção de modelos de autômatos que representem os subsistemas envolvidos;
3. Mapeamento dos eventos controláveis e não controláveis dos subsistemas;
4. Construção de modelos de autômatos que representem as especificações de controle para cada subsistema envolvido;

5. Obtenção do modelo da planta G, fazendo-se a composição síncrona dos autômatos dos subsistemas;
6. Obtenção do modelo da especificação E, fazendo-se a composição síncrona dos autômatos das especificações individuais;
7. Cálculo da linguagem K, ou seja, a linguagem da planta que satisfaz a especificação, através da composição síncrona de G e E;
8. Cálculo da máxima linguagem controlável (supC) contida na linguagem K;

No capítulo 4 será apresentado a síntese para o supervisor e estas etapas serão utilizadas para o desenvolvimento de um caso prático. No entanto, quando um grande número de subsistemas compõe a planta G e há igualmente um grande número de especificações, ocorre um crescimento exponencial no número de estados do autômato final devido à composição síncrona da planta G e da especificação E. Neste caso, a abordagem monolítica pode ter um desempenho computacional desfavorável, acarretando na explosão de estados (Queiroz, 2004).

Na seqüência será apresentada a abordagem modular local.

2.5 Abordagem Modular Local

Em (Queiroz, 2000) e (Queiroz e Cury, 2000a, 2000b, 2000c e 2002a) é apresentada uma extensão à teoria proposta por Ramadge e Wonham (1989), que permite explorar a modularidade das especificações e da planta, de forma a diminuir a complexidade computacional da síntese de supervisores. Esta extensão à teoria clássica foi denominada abordagem de controle modular local (Queiroz, 2004).

Segundo os autores, o sistema físico deve ser decomposto em subsistemas. Estes subsistemas devem ser modelados por autômatos que representem seu comportamento, por exemplo, o autômato deve possuir informação de início de operação e fim de operação do subsistema.

Assim, de acordo com cada especificação imposta pelo projetista, teremos uma *planta local*, que consiste na composição síncrona dos subsistemas cujos eventos são comuns a uma dada especificação (ou a mais de uma).

Nesta abordagem, o diferencial está no tratamento das especificações, onde se tem a chamada *especificação local*, que consiste no sincronismo de uma especificação com a sua *planta local*, ou seja, os subsistemas que têm eventos comuns com a especificação. Na Figura 2.7 é mostrada uma ilustração da metodologia usada para obtenção dos supervisores locais:

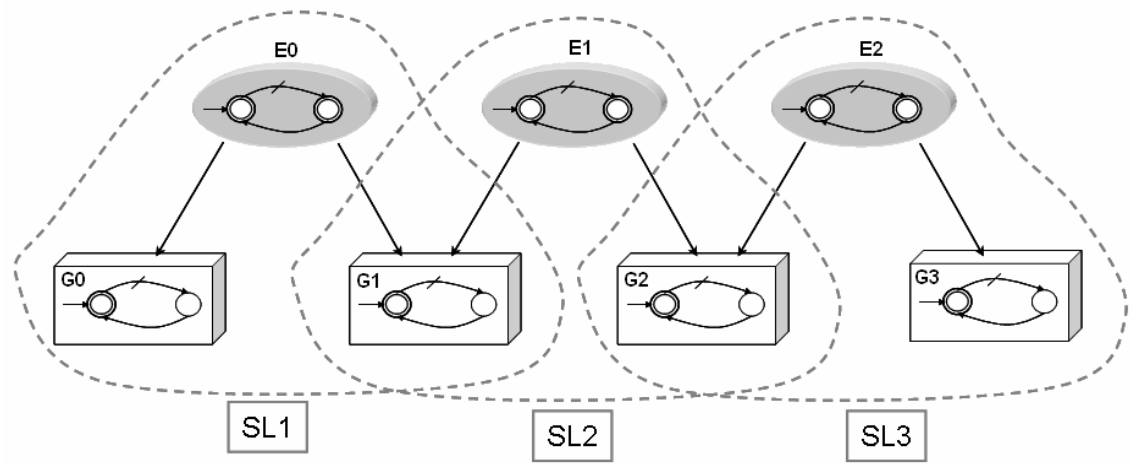


Figura 2.7 – Ilustração da metodologia para obtenção dos Supervisores Locais

A ação de controle nesta abordagem é distribuída entre vários supervisores, aqui chamados de *supervisores locais,* onde cada um deles representa a máxima linguagem controlável para cada planta local. Desta forma, cada supervisor local coordena uma parte do sistema global. A Figura 2.8 ilustra a estrutura de controle dos supervisores modulares locais.

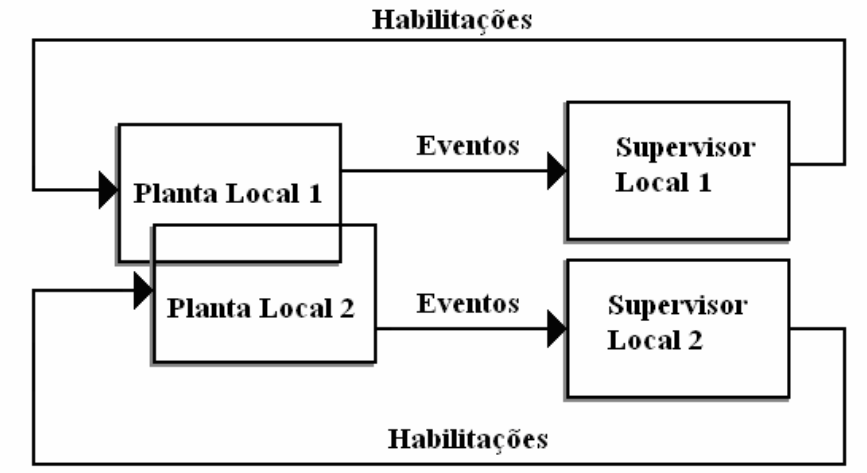


Figura 2.8 – Estrutura de controle modular local

Conforme exposto, cada subsistema será representado por um autômato de estrutura Gi = (Xi, Σi, fi, x_{oi}, X_{mi}), onde i ∈ I = {1,....,n}, e n é o número de subsistemas.

Cada especificação é representada por um autômato Ei e definida, respectivamente, em subconjuntos de eventos $\Sigma i \subseteq \Sigma$ para i ∈ I = {1,....,m}, onde m é o número de especificações.

Desta forma, a composição síncrona dos subsistemas cujo comportamento é restringido por uma determinada especificação resulta em uma planta chamada de *planta local*, denotada por G_{localj}, e j = 1...m. Assim, o comportamento desejado pode ser expresso

por uma linguagem alvo denotado por $K_{localj} = Ei||Lm(G_{localj})$. A partir disso, calculam-se as máximas linguagens controláveis locais $SupC(K_{localj}, G_{localj})$, e j = 1...m. Com este procedimento é possível realizar a síntese de um supervisor local para cada uma das especificações definidas no projeto. A representação destes supervisores locais é dada por S_{Li}, onde $i \in I = \{1,....,m\}$ e m é o número de supervisores locais.

Conforme demonstrado por Queiroz e Cury (2000a, 2000b) em seu Teorema 1, se o conjunto $\{SupC(K_{local,j}, G_{local,j}), \quad j = 1....m\}$ for localmente modular, então, $SupC(K_{global}, G) = ||_{j=1}^{m} SupC(K_{local,j}, G_{loc,j})$, ou seja, caso seja verificada a modularidade local do conjunto de supervisores locais, é assegurado que a ação conjunta de todos os supervisores é não bloqueante e que este procedimento não resulta em perda de desempenho em relação ao controle monolítico.

De acordo com Queiroz (2000), a sistematização da aplicação do controle modular local é implementada nas seguintes etapas:

1. Identificação do conjunto de subsistemas envolvidos no problema;

2. Construção de autômatos que representem os subsistemas envolvidos;

3. Obtenção da composição síncrona dos subsistemas que apresentem eventos comuns;

4. Construção do modelo de cada especificação isoladamente, considerando apenas os eventos relevantes;

5. Obtenção da planta local para cada especificação compondo-se os subsistemas que possuem eventos em comum com a especificação em questão;

6. Cálculo da linguagem alvo para cada planta local, através do produto síncrono da planta local com sua respectiva especificação;

7. Cálculo da máxima linguagem controlável contida em cada linguagem alvo;

8. Verificar a modularidade local (não conflito) das linguagens resultantes;

9. Se forem modulares, implementar um supervisor local para cada linguagem controlável;

10. Se não forem modulares, resolver o conflito. Algumas alternativas de solução são propostas detalhadamente por (Queiroz, 2004).

O teste da modularidade (não conflito) pode ser feito através da composição síncrona de todos os autômatos que representam as máximas linguagens controláveis (supervisores modulares locais) obtidas no passo 7. Caso o autômato resultante desta composição síncrona

seja Trim[1], os supervisores são modulares entre si. Porém, conforme destacado por Queiroz (2000), a complexidade do teste da modularidade local, por exigir a composição de todos os supervisores locais, acaba crescendo exponencialmente com o número de especificações e subsistemas envolvidos, visto que o problema de conflito é um problema global que ocorre pela interação de todos os supervisores com toda a planta. O autor ainda aponta para a necessidade do desenvolvimento de métodos mais eficientes para o teste, tais como a verificação de condições da estrutura da planta que garantam a modularidade.

2.6 Conclusões

Neste capítulo foram apresentados os formalismos matemáticos necessários para o entendimento da TCS, abordando a classe de Sistemas a Eventos Discretos e suas formas de representação por linguagens e autômatos. Foram apresentadas as duas abordagens utilizadas neste trabalho, a abordagem Monolítica proposta por Ramadge e Wonham (1989) e a extensão desta teoria, a abordagem Modular Local, proposta por Queiroz e Cury (2000). Também foram vistas as etapas para o cálculo do supervisor monolítico e modular local.

No Capítulo 4 as etapas descritas anteriormente serão utilizadas para realização da síntese monolítica e modular local de supervisores para uma célula flexível de manufatura didática, a qual será descrita no Capítulo 3.

[1] O autômato é dito Trim quando tem somente componentes acessíveis e co-acessíveis.

3 A INTEGRAÇÃO DOS EQUIPAMENTOS DA CÉLULA FLEXÍVEL DE MANUFATURA DIDÁTICA

O presente capítulo tem por objetivo apresentar os equipamentos utilizados para a integração da célula flexível de manufatura didática apresentada neste trabalho. São descritos os equipamentos que fazem parte da célula, suas características técnicas e como é feita a adequação dos sinais elétricos dos controladores dos robôs com o CLP, através de um módulo de interface desenvolvido especificamente para essa finalidade. Neste capítulo são descritas também as rotinas de programação dos robôs didáticos utilizados na célula, destinados a fazer o transporte das peças entre os dispositivos que compõem a célula de manufatura didática.

3.1 Descrição dos equipamentos

A célula flexível de manufatura didática apresentada neste trabalho é composta pelos seguintes equipamentos (ver Figura 3.1):

- 2 Robôs Eshed Robotech Scorbot ER4PC;
- 1 Mesa giratória Intelitek (Rotary Table);
- 1 Esteira Intelitek (Conveyor ASSV);
- 1 Mesa de experimentos Intelitek (Experiment Table);
- 1 Estação de teste (Sensor fotoelétrico Sense tipo Dark on / Dark light).

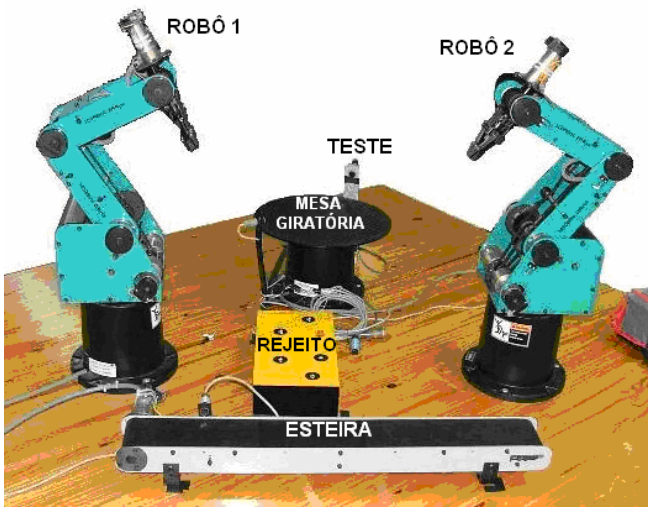


Figura 3.1 - Equipamentos que compõem a célula flexível de manufatura.

A seguir apresenta-se uma descrição dos dispositivos utilizados e das adaptações feitas para a sua integração na célula a ser controlada através de um Controlador Lógico Programável (CLP). Maiores informações sobre o *hardware* e a programação do CLP podem

ser obtidas consultando-se (Castrucci e Moraes, 2007), (Silveira e Santos, 1998) e (Siemens, 2001, 2002a, 2002b, 2002c).

Os Robôs Scorbot ER4pc são robôs didáticos de 5 graus de liberdade (Robotec, 1982a). São robôs desenvolvidos com as mesmas características de um robô industrial, porém sua estrutura permite a observação das engrenagens e mecanismos de movimentação. O controle de cada robô é feito por um controlador dedicado e sua programação é feita utilizando-se instruções padronizadas na linguagem de programação do equipamento. Cada controlador Scorbot ER4pc possui 8 entradas digitais e 4 entradas analógicas em tensão de 0 a 10V e resolução de 8 bits. O controlador possui também 8 saídas digitais, sendo 4 saídas a relés, 4 saídas transistorizadas e mais duas saídas analógicas em tensão de 0 a 10V e resolução de 8 bits. Na Figura 3.2 tem-se o Robô e seu controlador.

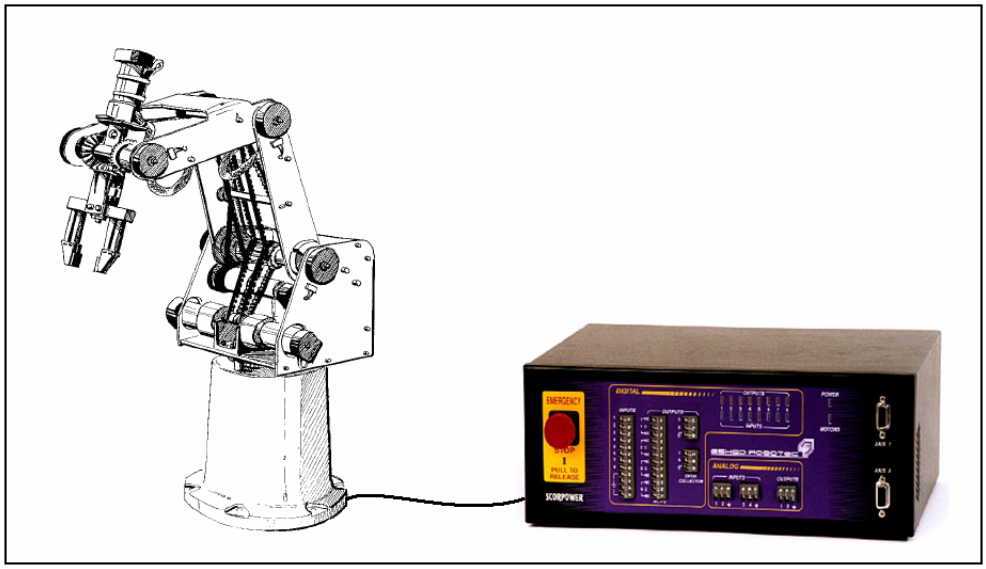


Figura 3.2 – Robô e seu controlador

A mesa giratória Intelitek (Figura 3.3) tem a capacidade de girar nos sentidos horário e anti-horário e o motor que provoca seu giro tem alimentação de 24 V. Assim, a velocidade do giro é controlada pelo valor da tensão aplicada ao motor e o sentido de giro pela polaridade da tensão aplicada. Muito embora originalmente o ângulo de giro da mesa fosse controlado por intermédio do controlador do robô com auxílio de um *encoder*, optou-se por fazer o seu controle através do CLP. Assim, no intuito de definir um ângulo de giro para a mesa, fixaram-se quatro marcadores de metal (arruelas) sob a sua base de forma a indicar as quatro posições nas quais a mesa deve parar o giro. Um sensor indutivo (alcance de 5mm) conectado a uma entrada do CLP é usado para detectar (sem contato direto) a passagem dos marcadores.

Figura 3.3 – Mesa Giratória

De forma semelhante, a velocidade e o sentido de giro da esteira Intelitek (alimentação de 10 a 30 V) pode ser modificada de acordo com o valor e a polaridade da tensão aplicada ao motor responsável pelo seu funcionamento. Um sensor fotoelétrico fixado na lateral da esteira é conectado ao CLP para indicar a presença de peça sobre a mesma, conforme a Figura 3.4.

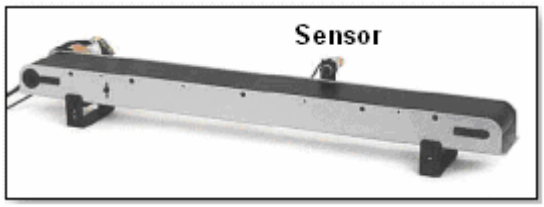


Figura 3.4 – Esteira e Sensor

A mesa de experimentos Intelitek (Figura 3.5) possui 4 posições nas quais existem chaves tipo fim de curso que indicam a presença de peças. Assim, esta mesa pode ser usada como um armazém de peças, por exemplo. A lâmpada e a sirene podem ser utilizadas como elementos de sinalização.

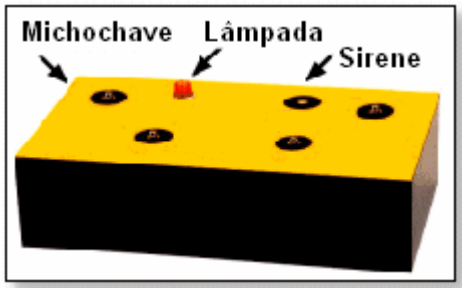


Figura 3.5 – Mesa de Experimentos

A estação de teste (Figura 3.6) consiste em um sensor fotoelétrico tipo *Dark on / Dark light*, que pode identificar a presença (ou ausência) de luz. Este sensor é utilizado para simular o teste de qualidade das peças, conforme explicado mais adiante.

Figura 3.6 – Sensor de teste

Como elemento de controle da célula é utilizado um CLP Siemens da família S7-300, CPU modelo 312 IFM, cujas características básicas são: memória de trabalho de 6kB, 10 entradas digitais e 6 saídas digitais integradas e comunicação com o computador via conversor RS232/RS485 padrão MPI (protocolo de comunicação Siemens). Maiores detalhes sobre este equipamento e suas características técnicas encontram-se no manual do fabricante (Siemens, 2001, 2002b). Na Figura 3.7 é mostrado o CLP utilizado no trabalho.

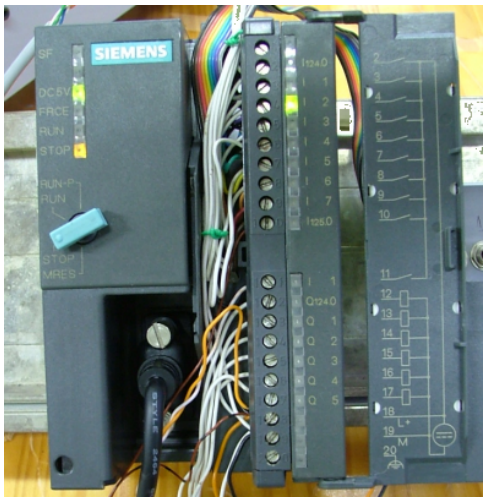


Figura 3.7 – CLP Siemens S7-300

Além dos equipamentos descritos anteriormente, os seguintes dispositivos foram utilizados na célula: uma interface para adequação dos sinais dos controladores dos robôs e do CLP, que será detalhada na Seção 3.2, 2 computadores Pentium II com 32MB de memória RAM, que são usados para realizar a execução do *software* de programação dos controladores dos robôs, além do *software* Simatic S7 para a programação do CLP. Maiores detalhes sobre o *software* de programação deste CLP encontram-se no manual do fabricante (Siemens, 2002a, 2002c).

3.2 Módulo de interface e Integração dos equipamentos

Os sinais dos sensores e atuadores dos equipamentos da célula são interligados, respectivamente, ao módulo de entradas e saídas digitais do CLP, por meio do qual é feito o controle da seqüência de operação de cada dispositivo. Entretanto, as rotinas de execução das tarefas dos robôs são executadas em controladores individuais, dedicados para cada robô. Assim, para que os robôs sejam efetivamente integrados à célula de manufatura, é desejável que se possa comandá-los através do elemento de controle da célula, ou seja, através do CLP. Desta forma, através do CLP é possível comandar (disparar) a execução de tarefas pré-programadas dos robôs, mas o controle efetivo de cada junta do robô fica a cargo do controlador dedicado.

Uma vez que os controladores dos robôs trabalham com lógica negativa (NPN) e o CLP trabalha com lógica positiva (PNP), foi necessário desenvolver uma interface para realizar a conexão dos controladores dos robôs com o CLP. A interface confeccionada em placa de circuito impresso e é composta de 6 relés (24Vcc / contato reversível de 1A) e uma fonte DC 12V/2A (Figura 3.8).

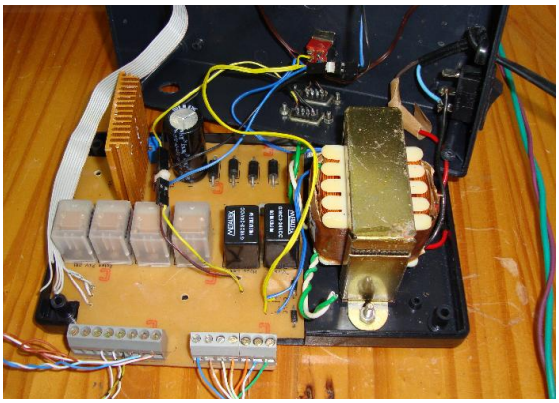


Figura 3.8 – Interface de sinais

O diagrama de blocos apresentado na Figura 3.9 ilustra o uso da interface entre o CLP e os controladores dos robôs. Na Figura 3.10 detalha-se a interligação do CLP com o controlador do robô. Três sinais de saída do CLP (24V) são interligados a relés da interface de sinais. O contato normalmente aberto desses relés realimenta a referência de sinal (0V) dos controladores dos robôs, ativando assim a rotina correspondente na programação de cada robô.

Uma vez que a saída do CLP tem uma capacidade de corrente de apenas 500mA, utilizou-se uma fonte auxiliar de 12V (cuja corrente de saída é de 2A) para realizar o acionamento dos motores da esteira e da mesa giratória.

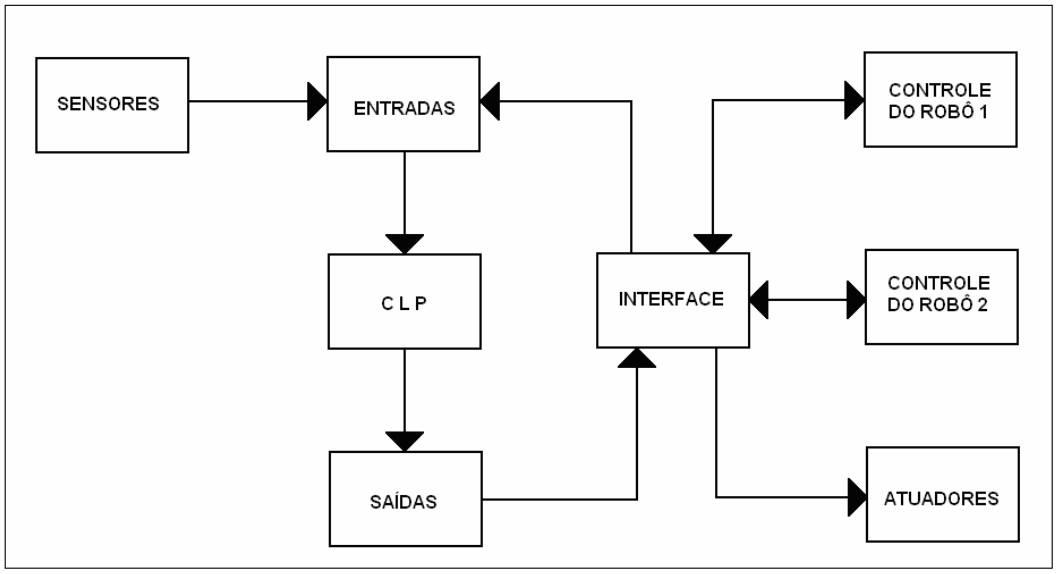


Figura 3.9 - Interface CLP – Entradas e Saídas da Célula

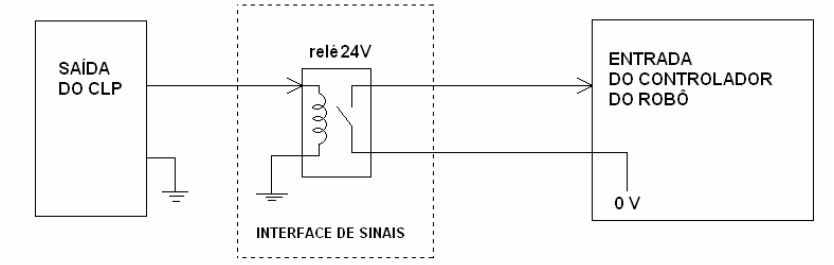


Figura 3.10 - Acionamento do controlador do robô pelo CLP

Desta forma, conforme mostrado na Figura 3.11, o sinal de saída do CLP (24V) responsável pela ativação do motor da esteira é interligado ao relé na interface de sinais, cujo contato normalmente aberto alimenta o motor da esteira com a tensão da fonte auxiliar (12V/2A).

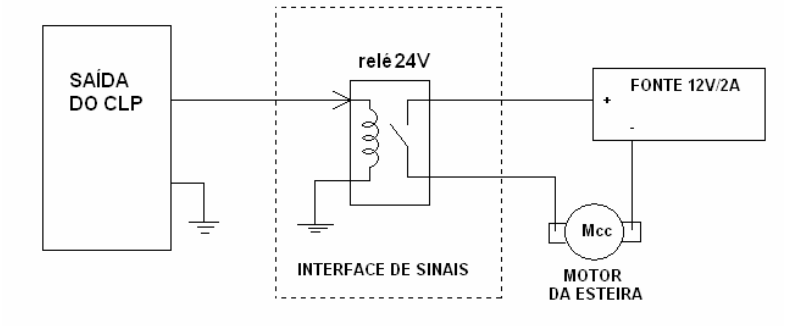


Figura 3.11 - Acionamento do motor da esteira pelo CLP

No intuito de possibilitar que o CLP receba a sinalização de final de operação dos robôs, foi necessário ainda interligar os sinais de saída dos controladores dos robôs ao CLP. Para isso, a alimentação do CLP (24V) foi ligada diretamente ao contato do relé de saída do controlador do robô, conforme a Figura 3.12.

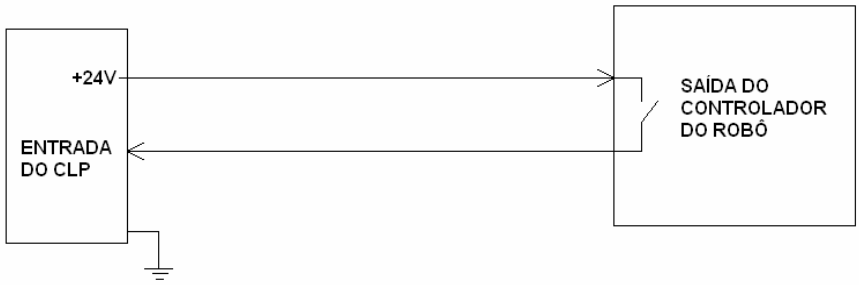


Figura 3.12 - Ligação da saída do controlador do robô com o CLP

No total foram utilizados seis relés na interface de sinais, dois para os acionamentos do motor da esteira e da mesa giratória, e quatro para a conexão dos sinais de saída do CLP às entradas dos controladores. Na Figura 3.13 mostra-se a interface completa e todas as conexões.

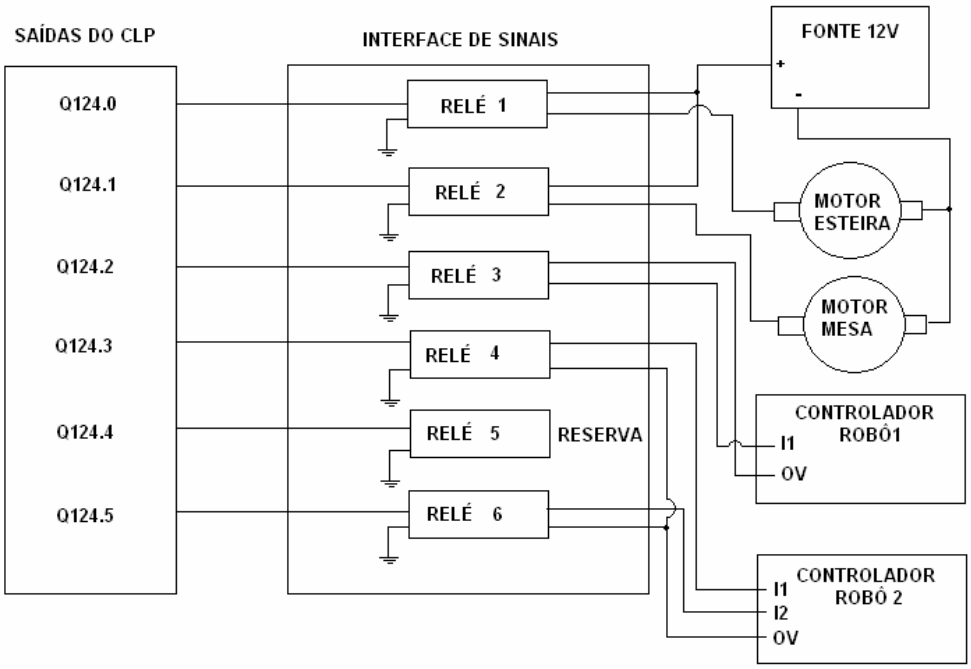


Figura 3.13 - Esquema de ligações da interface de sinais

3.3 Programação das Rotinas dos Robôs

A programação dos robôs é feita no *software* SCORBASEpro (Robotec, 1982b), através do qual são definidas as rotinas de execução de movimentos de acordo com o modelamento feito no projeto. Na Seção 4.1 será detalhada toda a seqüência de funcionamento dos robôs na célula. Assim, cada robô terá tarefas a executar, como por exemplo, o transporte de uma peça da esteira de entrada até a mesa giratória. Para que essa rotina de transporte de peça tenha início, é necessário um comando do controlador (CLP), que ativa a rotina de execução. Além da programação da trajetória do robô, é preciso que o mesmo envie um sinal para o controlador informando que finalizou a tarefa (retorno para a posição inicial). Estes comandos e definição de movimentos serão explicados utilizando-se os programa desenvolvidos para os Robôs 1 e 2 da célula, onde cada comando está comentado com duas barras, por exemplo, "*// Início da rotina*". Assim, conforme a seqüência de programação do Robô 1, quando o controlador recebe um sinal do CLP, esse processamento é feito na linha 2 do programa:

```
1: START:                          // Início da rotina
2: If Input 1 on jump to ESTEIRA   // Se I1=1, executa rotina ESTEIRA
3: Jump to START                   // Se I1=0, retorna para rotina START
4: ESTEIRA:                        // Seqüência de movimentação
5: Go to Position 2 fast           // Vai para Posição 2
6: Open Gripper                    // Abre a garra
7: Go to Position 3 fast           // Vai para Posição 3
8: Close Gripper                   // Fecha a garra
9: Go to Position 2 fast           // Vai para Posição 3
10: Go to Position 1 fast          // Vai para Posição 1
11: Go to Position 5 fast          // Vai para Posição 5
12: Go to Position 6 fast          // Vai para Posição 6
13: Open Gripper                   // Abre a garra
14: Go to Position 5 fast          // Vai para Posição 5
15: Close Gripper                  // Fecha a garra
16: Go to Position 1 speed 7       // Vai para Posição 1
17: Turn on output 1               // Ativa a saída 1 do controlador
18: Wait 10 (10 ths of seconds)    // Aguarda 1 segundo
19: Turn off output 1              // Desliga a saída 1 do controlador
20: Jump to START                  // Retorna para a linha 1 do programa
```

Na linha 17 do programa é possível observar como o controlador do robô se comunica com o CLP ao final da rotina de execução, desta forma, um sinal é enviado ao CLP através do acionamento da saída 1 do controlador do robô, informando que a tarefa foi concluída.

Na linha 20 do programa ocorre o retorno do programa para a linha 1, onde é feita novamente a verificação do sinal de entrada do controlador para que a rotina seja executada se um novo sinal do CLP for recebido pelo controlador.

O fluxograma que mostra a seqüência de funcionamento da rotina do robô 1 pode ser visto na Figura 3.14:

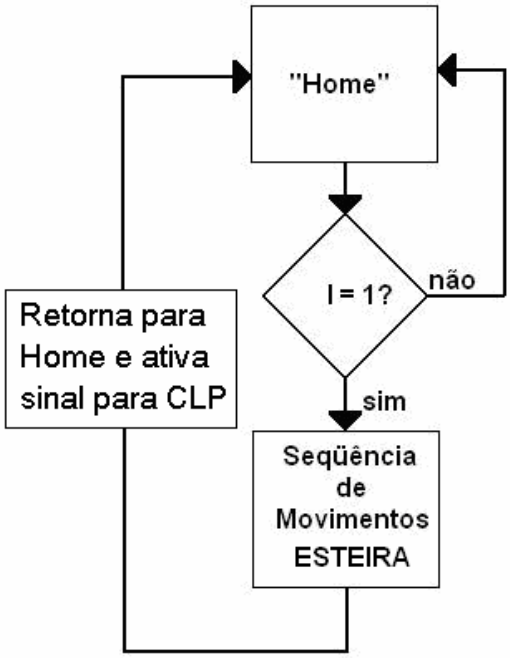


Figura 3.14 - Fluxograma da rotina do robô 1

Este fluxograma indica que a partir da posição inicial do robô, também chamada *home*, é possível a seqüência de movimentos, desde que a entrada 1 (I1) seja ativada. Conforme já descrito, o CLP é responsável pela ativação desta entrada. Ao final da seqüência programada o robô retorna para a posição *home* e envia um sinal para o CLP.

Já para a programação do Robô 2 foi preciso definir duas rotinas de movimentação, chamadas de REJEITO e SAÍDA. Estas duas rotinas são responsáveis por transportar a peça para uma área de retrabalho ou para uma área de retirada da peça da célula.

A programação do Robô 2 é apresentada na seqüência. Observa-se que para este robô são definidas duas entradas, cada uma responsável por iniciar uma das rotinas programadas:

```
1: START:                          // Início da rotina
2: If Input 1 on jump to REJEITO   // Se I1=1, executa rotina REJEITO
3: If Input 2 on jump to SAÍDA     // Se I2=1, executa rotina SAÍDA
4: Jump to START                   // Senão, retorna para rotina START
5: REJEITO:                        // Seqüência de REJEITO
6: Go to Position 1 fast
7: Go to Position 2 fast
8: Open Gripper
9: Go to Position 3 fast
10: Close Gripper
11: Go to Position 2 fast
12: Go to Position 1 fast
13: Go to Position 7 fast
14: Go to Position 8 fast
15: Open Gripper
16: Go to Position 7 fast
17: Close Gripper
18: Go to Position 1 fast
19: Turn on output 1
20: WaiT 10 (10 ths of seconds)
21: Turn off output 1
22: Jump to START
23: SAÍDA:                         // Seqüência de SAÍDA
24: Go to Position 1 fast
25: Go to Position 2 fast
26: Open Gripper
27: Go to Position 3 fast
28: Close Gripper
29: Go to Position 2 fast
30: Go to Position 1 fast
31: Go to Position 9 fast
32: Go to Position 10 fast
33: Open Gripper
34: Go to Position 9 fast
35: Close Gripper
36: Go to Position 1 fast
37: Turn on output 1
38: WaiT 10 (10 ths of seconds)
39: Turn off output  1
40: Jump to START                  // Retorna para a linha 1 do programa
```

O fluxograma que mostra a seqüência de funcionamento da rotina do robô 1 pode ser visto na Figura 3.15 - Fluxograma da rotina do robô Figura 3.15

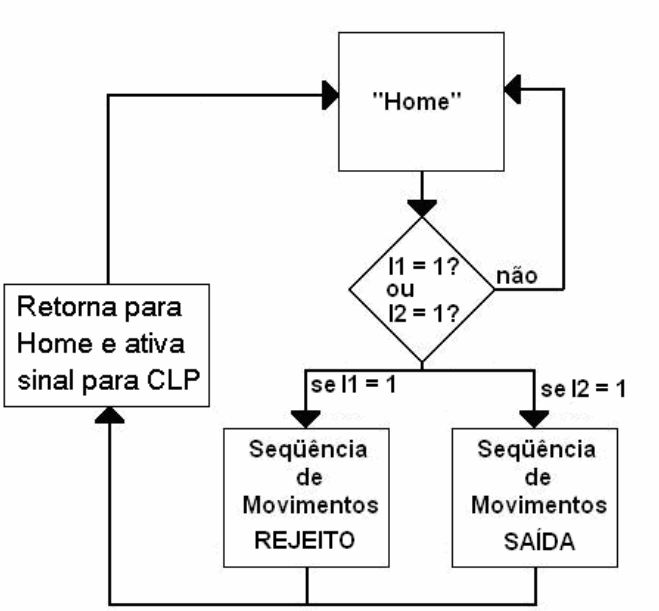


Figura 3.15 - Fluxograma da rotina do robô 2

Este fluxograma indica que a partir da posição inicial do robô 2 (*home*), é possível a seqüência de movimentos REJEITO ou SAÍDA, desde que as entradas I1 ou I2 sejam ativadas. Fisicamente não é possível ativar as duas entradas ao mesmo tempo, e conforme já descrito, o CLP é responsável pela ativação destas entradas. Desta forma, o robô 2 precisa finalizar uma rotina para iniciar a outra, se for o caso. Ao final de qualquer uma das seqüências programadas, o robô retorna para a posição *home* e envia um sinal para o CLP. Este sinal independe de qual rotina foi executada, pois a posição *home* é a mesma para qualquer uma das seqüências programadas.

3.4 Conclusões

Neste capítulo foram apresentados os equipamentos utilizados para a integração da célula flexível de manufatura didática utilizada neste trabalho e suas características técnicas. Foi visto ainda como é feita a adequação dos sinais elétricos do CLP com os controladores dos robôs, utilizando-se o módulo de interface. Neste capítulo são descritas também as rotinas de programação dos robôs didáticos utilizados na célula.

No capítulo seguinte será visto como realizar a síntese monolítica e modular local de supervisores para a célula flexível de manufatura didática apresentada anteriormente.

4 SÍNTESE DE SUPERVISORES PARA A CÉLULA DE MANUFATURA DIDÁTICA

A modelagem da célula e a resolução de problemas de controle são feitos seguindo a Teoria de Controle Supervisório de Sistemas a Eventos Discretos (Ramadge e Wonham, 1989). Nesta abordagem o controle é feito por um autômato denominado de supervisor, o qual restringe o comportamento do sistema físico, satisfazendo a um conjunto de especificações. O alfabeto de eventos que compõe os modelos individuais de cada dispositivo é dividido em dois tipos de eventos: os eventos controláveis, cuja ocorrência pode ser modificada pela ação de controle, como por exemplo, o início de uma atividade ou a parada de uma esteira, e os eventos não controláveis, cuja ocorrência não pode ser desabilitada pela ação de controle, como por exemplo, a ativação de um sensor (Cury, 2001). A metodologia básica para a síntese de um supervisor ótimo (não bloqueante e minimamente restritivo) que resolve o problema foi inicialmente proposta por Ramadge e Wonham (1989), e se baseia em três passos principais. Estes passos serão desenvolvidos na seqüência deste capítulo, com base nos conceitos apresentados no Capítulo 2. São eles:

1. Modelagem da planta;
2. Modelagem das especificações;
3. Síntese do(s) supervisor(es).

Cabe ressaltar que na obtenção do modelo para a planta foram introduzidas algumas restrições físicas, conforme proposto por Moraes e Leal (2006), que foram modeladas devido aos modelos individuais não possuírem informações importantes do sistema global. Estas restrições serão detalhadas na Seção 4.3.

Neste capítulo (Seção 4.6) é apresentada ainda uma extensão à Teoria de Controle Supervisório, chamada de Abordagem Modular Local (Queiroz e Cury, 2000), na qual o problema de modelagem e controle é abordado de forma modular, ou seja, a planta é ‘dividida’ em função da modularidade das especificações de controle.

4.1 Um Problema Motivador

Como simplificação para a resolução do problema proposto neste trabalho, adota-se apenas duas posições para a mesa giratória, onde na primeira posição tem-se a entrada da peça, e na segunda posição tem-se o teste e a retirada da peça da mesa giratória. Conforme a Figura 4.1, a seqüência de funcionamento consiste em:

- Transporte de peça pela esteira;
- Parada da peça em frente ao sensor (no final da esteira)
- Robô 1 transporta a peça para a posição P1 da mesa giratória;
- A peça é movimentada para a posição P2 na mesa giratória;
- O sensor realiza o teste (T);
- Robô 2 retira a peça de acordo com o resultado do teste:
- Peça boa é retirada para ‘área de depósito de peças prontas’;
- Peça ruim é retirada para a mesa de rejeito;

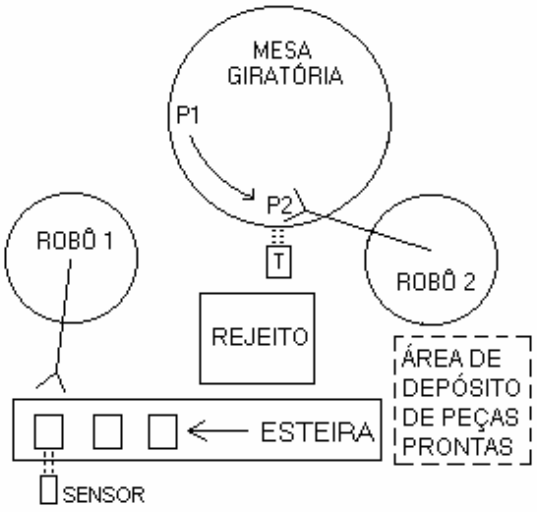


Figura 4.1 - Planta baixa da célula.

A Tabela 4-1 indica o evento associado a cada dispositivo da célula, o endereço físico do sinal no CLP (entradas e saídas) e a descrição do evento. Desta forma, os eventos controláveis são associados às saídas do CLP, que serão ativadas ou desativadas conforme a seqüência definida pelo supervisor (ação de controle). Já os eventos não controláveis são associados às entradas do CLP, uma vez que dependem unicamente do estado dos dispositivos da célula, não podendo ser impedidos de ocorrer pelo supervisor.

Tabela 4-1 - Eventos dos dispositivos da célula.

DISPOSITIVO	EVENTO	TIPO DE SINAL	DESCRIÇÃO
Esteira	E_liga	Saída Q124.0	Início de operação da esteira.
	E_desl	Saída Q124.0	Fim de operação da esteira.
Sensor da esteira	S_liga	Entrada I124.0	Chegada de peça no final da esteira.
	S_desl	Entrada I124.0	Retirada de peça no final da esteira.
Mesa giratória	I_giro	Saída Q124.1	Início de operação da mesa giratória.
	F_giro	Entrada I 124.2	Fim de operação da mesa giratória.

Robô 1	T_M	Saída Q124.2	Início de operação do robô 1.
	F_rb1	Entrada I125.0	Fim de operação do robô 1.
Estação de teste	I_teste	Saída Q124.4	Início de operação da estação de teste.
	T_OK	Entrada I124.4	Resultado do teste para peça boa.
	T_NOK	Entrada I124.4	Resultado do teste para peça ruim.
Robô 2	T_R	Saída Q124.3	Início de operação do robô 2, transporte de peça ruim.
	T_S	Saída Q124.5	Início de operação do robô 2, transporte de peça boa.
	F_rb2	Entrada I125.1	Fim de operação do robô 2.

4.2 Modelos dos dispositivos da planta

Para a nomenclatura dos eventos na modelagem dos autômatos de cada dispositivo da célula, foi utilizada a Tabela 4-1 como referência para os eventos controláveis e não controláveis. Nos modelos dos autômatos apresentados, os círculos representam os estados do dispositivo e os círculos duplos representam estados onde a tarefa é completada. Os arcos entre os estados indicam as transições ou eventos. Arcos seccionados por uma pequena linha transversal indicam que o evento é controlável (Cury, 2001).

O funcionamento da esteira pode ser modelado pelo autômato G1, mostrado na Figura 4.2, onde o estado inicial “0” representa a esteira desligada e o estado “1” representa a esteira ligada. Nota-se aqui que os eventos E_liga e E_desl são eventos controláveis, ou seja, são gerados pela ação de controle do supervisor.

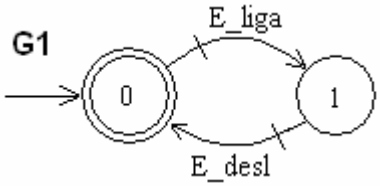


Figura 4.2 - Autômato para a esteira.

A Figura 4.3 ilustra o autômato G2, que modela o sensor colocado no final da esteira. Os eventos S_liga e S_desl correspondem a ativação e desativação do sensor provocados pela presença e ausência de peça respectivamente. Sendo assim, estes eventos não são controlados pelo supervisor.

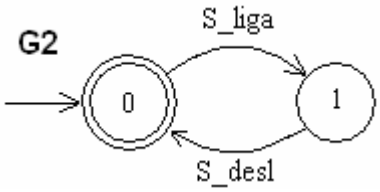


Figura 4.3 - Autômato para o sensor.

O comportamento do robô 1 é modelado pelo autômato G3 da Figura 4.4. O evento T_M inicia a rotina programada no controlador do robô que faz o transporte da peça da esteira para a posição P1 da mesa giratória. O evento F_rb1 é gerado quando o CLP recebe a informação de que o robô 1 finalizou a tarefa e está na posição inicial novamente.

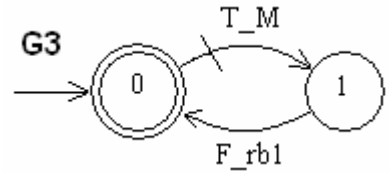


Figura 4.4 - Autômato para o Robô 1.

A Figura 4.5 ilustra o autômato G4, que modela o funcionamento da mesa giratória. O evento I_giro é gerado pela ação de controle, iniciando o movimento da mesa giratória, que movimenta a peça da posição P1 para a posição P2. O evento F_giro corresponde a leitura do sensor ativado pelos marcadores de metal posicionados na mesa giratória.

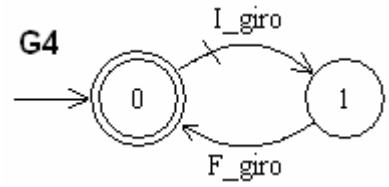


Figura 4.5 - Autômato para a mesa giratória.

O funcionamento da estação de teste pode ser modelado pelo autômato G5 da Figura 4.6, onde o evento I_teste habilita a estação de teste (neste caso, um sensor fotoelétrico) para realizar o teste da peça. Os eventos T_OK e T_NOK indicam, respectivamente, se a peça foi aprovada ou não, através da ativação ou não da saída deste sensor.

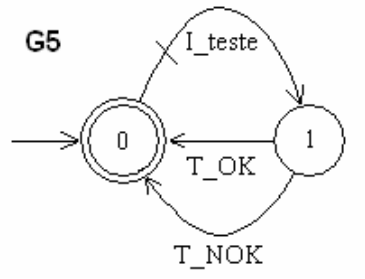


Figura 4.6 - Autômato para a estação de teste.

A Figura 4.7 ilustra o autômato G6, que modela o comportamento do robô 2. Os eventos T_R e T_S são gerados em função do resultado do teste. Assim, T_R corresponde ao transporte da peça pelo robô 2 para a estação de rejeito (no caso do T_NOK) e T_S corresponde ao transporte da peça para a área de saída (no caso do T_OK). O evento F_rb2 é

gerado quando o CLP recebe a informação de que o robô 2 finalizou a tarefa e está na posição inicial novamente.

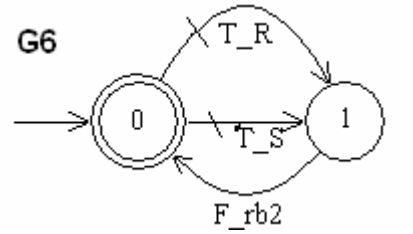


Figura 4.7 - Autômato para o Robô 2.

4.3 Modelo das restrições físicas da planta

Conforme já exposto no início deste capítulo, Moraes e Leal (2006) propõem a modelagem de restrições físicas que aproximam o modelo da seqüência real, pois, ainda que os modelos individuais descrevam o comportamento preciso de cada subsistema, estes modelos não possuem algumas informações importantes sobre o aspecto construtivo do sistema global. Por exemplo, se a esteira não estiver acionada, não ocorrerá o acionamento do sensor posicionado ao final desta, não havendo desta forma a geração do evento correspondente. É importante garantir que não haverá interferências do ser humano, tais como passar a mão em frente ao sensor ou colocar/retirar uma peça manualmente. Assim, a restrição física R1 mostrada na Figura 4.8, indica que o sensor da esteira só pode ser ativado quando a esteira está ligada.

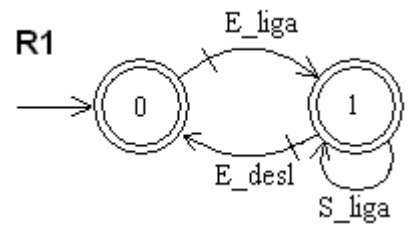


Figura 4.8 - Autômato para a restrição R1.

A restrição física R2, na Figura 4.9, informa que uma vez ativado o sensor da esteira, existe tempo hábil para (e é desejado) que a esteira seja desligada antes do sensor ser desativado, caso contrário, a peça não pararia na frente do sensor.

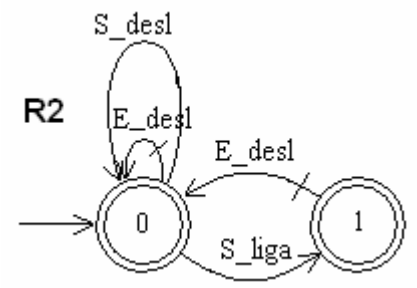


Figura 4.9 - Autômato para a restrição R2.

A restrição R3 (Figura 4.10) considera o fato de que a esteira será desligada na ativação do sensor (R2) e de que assim a desativação deste sensor depende da ação do robô 1. Desta forma, a desativação de tal sensor se dará após o início e antes do final de operação do robô 1.

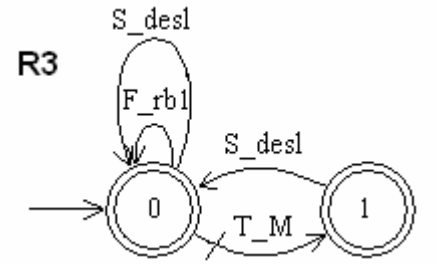


Figura 4.10 - Autômato para a restrição R3

A restrição física R4 (Figura 4.11) informa que o sensor da esteira não pode ser desativado se a esteira estiver desligada e o robô 1 estiver parado.

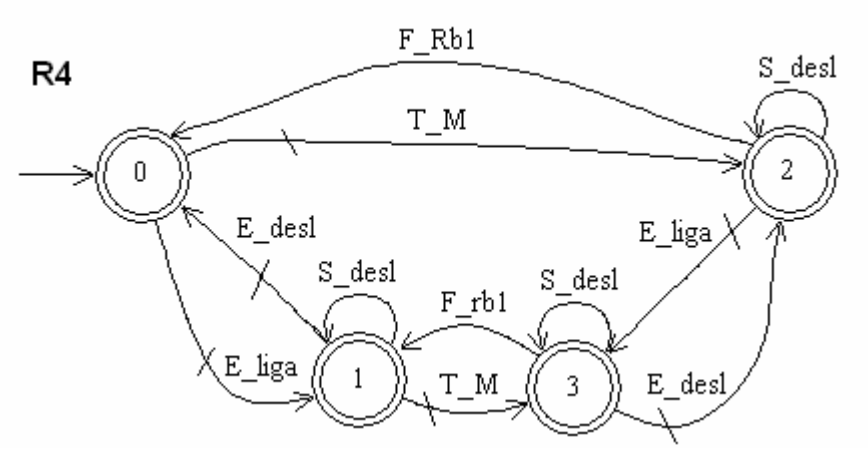


Figura 4.11 - Autômato para a restrição R4.

4.4 Modelo das especificações

A Teoria proposta por Ramadge e Wonham (1989) prevê que a partir de uma especificação que traduza a condição desejada para a operação do sistema, se possa chegar ao supervisor para controlar o seu funcionamento, de acordo com a seqüência de eventos desejada. Nesta seção são apresentadas as especificações que determinam como deve ser a operação coordenada dos dispositivos para o problema proposto no início deste capítulo.

O autômato mostrado na Figura 4.12 representa a especificação 1 e indica que a esteira pode ser ligada somente quando o sensor está desativado e pode ser desligada apenas quando o sensor está ativado.

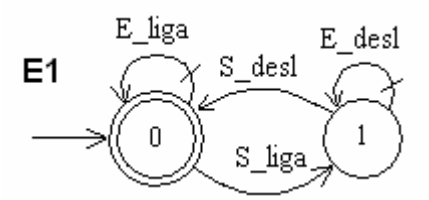


Figura 4.12 - Autômato para a Especificação 1.

O autômato mostrado na Figura 4.13 representa a especificação 2 e indica que o robô 2 executa o transporte da peça para o rejeito ou para a saída em função do resultado da estação de teste.

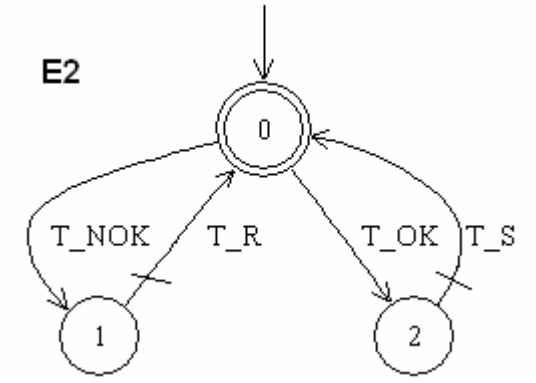


Figura 4.13 - Autômato para a Especificação 2.

O autômato mostrado na Figura 4.14 representa a especificação 3 e impõe que a mesa giratória inicie seu giro somente após o fim de operação do robô 1.

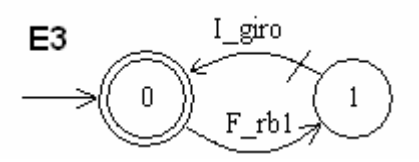


Figura 4.14 - Autômato para a Especificação 3

O autômato mostrado na Figura 4.15 representa a especificação 4 e indica que o robô 2 deve iniciar sua operação somente após o fim de giro da mesa. Além disso, a mesa só pode girar novamente após o fim de operação do robô 2.

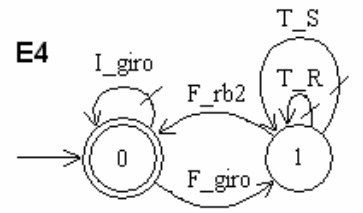


Figura 4.15 - Autômato para a Especificação 4.

O autômato mostrado na Figura 4.16 representa a especificação 5 e indica que o robô 1 só pode finalizar sua operação com a mesa em repouso. Esta especificação garante que o robô 1 só colocará peças sobre a mesa quando esta estiver parada.

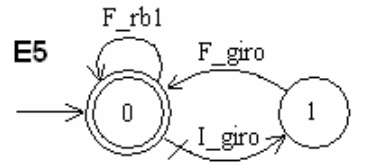


Figura 4.16 - Autômato para a Especificação 5.

O autômato mostrado na Figura 4.17 representa a especificação 6 e indica que a estação de teste deve iniciar sua operação somente após o fim de giro da mesa. Além disso, um novo giro na mesa não deve ser iniciado enquanto o teste não for finalizado.

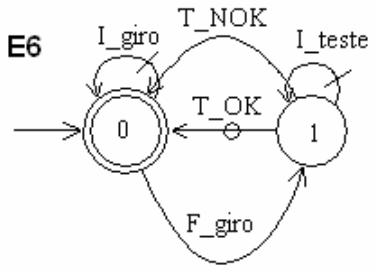


Figura 4.17 - Autômato para a Especificação 6.

O autômato mostrado na Figura 4.18 representa a especificação 7 e indica que a operação do robô 1 deve ser iniciada somente quando o sensor estiver ativado (quando houver peça na frente do sensor).

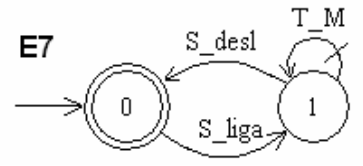


Figura 4.18 - Autômato para a Especificação 7.

O autômato mostrado na Figura 4.19 representa a especificação 8 e indica que o robô 1 só deve iniciar sua operação se a esteira estiver desligada.

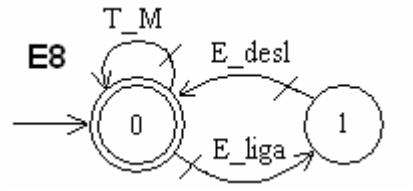


Figura 4.19 - Autômato para a Especificação 8.

O autômato mostrado na Figura 4.20 representa a especificação 9 e prioriza o início de teste em detrimento ao início de transporte do robô 1.

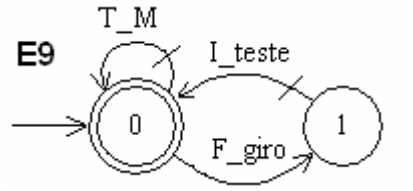


Figura 4.20 - Autômato para a Especificação 9.

4.5 Síntese do Supervisor Monolítico

Concluídos os modelos dos dispositivos que compõem a planta, das restrições físicas e das especificações de funcionamento, é preciso fazer a composição síncrona de todos estes modelos (Cury, 2001). Para isso, foi utilizado o *software* Grail (Reiser *et al.*, 2006). A obtenção de um autômato que modela o comportamento global da planta livre é feita através da composição síncrona dos autômatos dos dispositivos e das restrições físicas do sistema, ou seja, $G = G_1 \| .... \| G_6 \| R_1 \| .... \| R_4$. Este autômato possui 128 estados e 728 transições e representa o comportamento da planta em malha aberta (sem ação de controle). A especificação global E é obtida fazendo-se a composição síncrona das especificações individuais Ei, isto é, $E = E_1 \| .... \| E_9$. Determinou-se então, a linguagem alvo K, que é obtida a partir da sincronização de G e E, fazendo-se $K = G \| E$. Por fim, calculou-se a máxima linguagem controlável, denotada SupC(G,K). Assim, o autômato que reconhece SupC(G,K) possui 73 estados e 158 transições e consiste no supervisor não bloqueante e minimamente restritivo a ser implementado no CLP de forma a garantir o cumprimento das especificações de controle. Na Figura 4.21 é mostrado o autômato do supervisor monolítico de 73 estados e 158 transições. Na Seção 5.1 é apresentada a implementação deste supervisor monolítico.

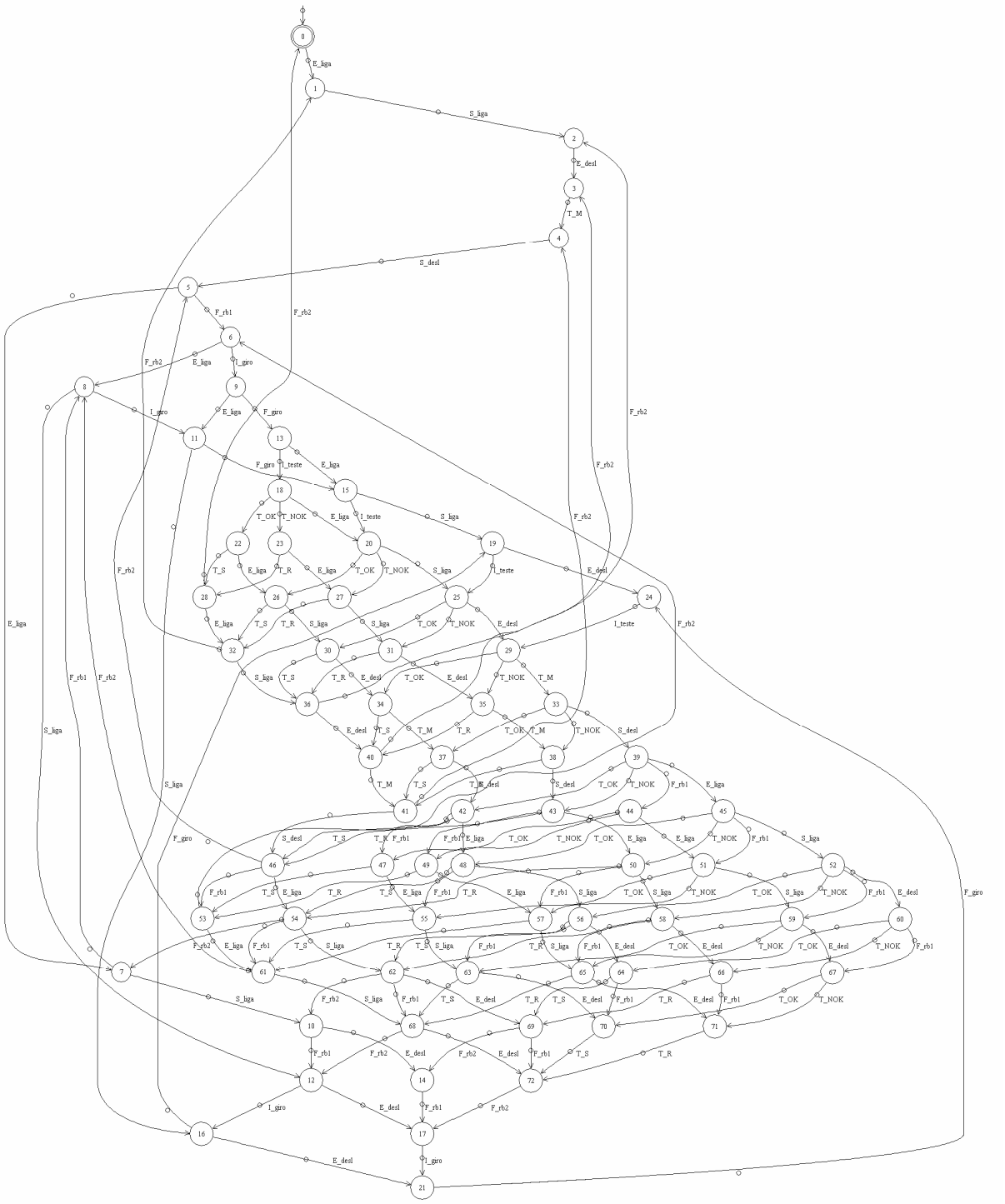


Figura 4.21 – Supervisor Monolítico de 73 estados e 158 transições

4.6 Síntese dos Supervisores Modulares Locais

Conforme apresentado no Capítulo 2, nesta abordagem a ação de controle é distribuída entre vários autômatos denominados *supervisores locais*, onde cada um deles

representa a máxima linguagem controlável para cada planta local. Desta forma, cada supervisor local coordena uma parte do sistema global.

De acordo com o exposto, na abordagem monolítica o sistema é analisado como um todo e procura-se um autômato que represente todas as possíveis seqüências de eventos que ele pode gerar e tarefas que pode completar. Como conseqüência, qualquer alteração/inclusão de dispositivos ou especificação de controle requer que o supervisor seja gerado novamente.

Na extensão à TCS proposta por Queiroz e Cury (2000), o problema de modelagem e controle é abordado de forma modular, ou seja, a planta é 'dividida' em função da modularidade das especificações de controle. Esta abordagem permite que alterações somente exigirão mudanças no modelo específico correspondente, e não no sistema como um todo.

A partir dos modelos dos dispositivos que compõem a planta e das especificações de funcionamento vistos nas Seções 4.2 e 4.4, é preciso fazer a composição síncrona entre estes modelos, conforme Queiroz e Cury (2000). Primeiramente, a composição síncrona foi feita entre os autômatos dos dispositivos, de forma a se obter as plantas locais. Para a obtenção das plantas locais, observou-se a ocorrência de eventos com o mesmo nome nos modelos individuais dos dispositivos, que ocorressem também em alguma especificação. Conforme a Figura 4.22, é possível observar que os eventos da especificação E1 aparecem nos subsistemas G1 (Esteira) e G2 (Sensor). Desta forma, é realizada a composição síncrona entre os autômatos dos subsistemas, dando origem à planta local GL1 e posteriormente é feita a composição síncrona entre a planta local GL1 e a especificação E1, dando origem à especificação local EL1. Este procedimento é feito para cada especificação do projeto que tenha eventos comuns com os subsistemas correspondentes.

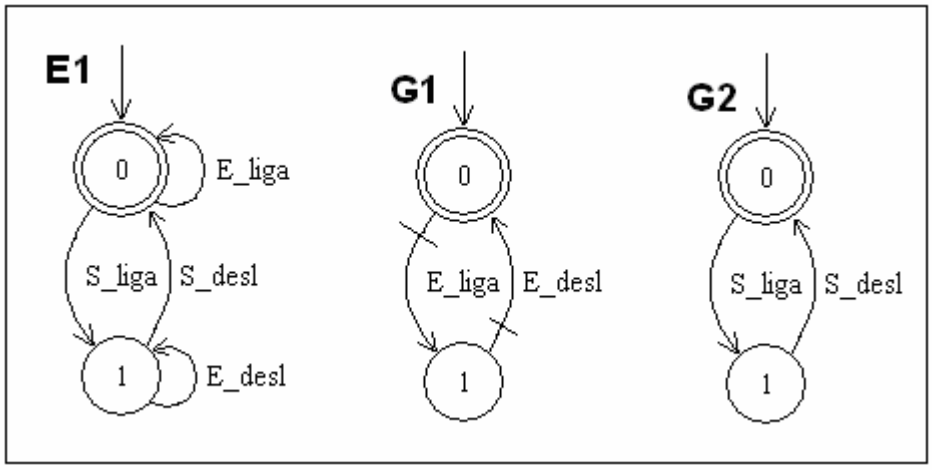


Figura 4.22 – Eventos comuns aos autômatos

Na Tabela 4-2 são apresentados todos os eventos comuns entre as especificações e os subsistemas. Pode-se observar que as especificações E3 e E5 têm eventos comuns aos autômatos G3 e G4, por esta razão, é feita primeiramente a composição síncrona entre as duas especificações (E3||E5), chegando-se à especificação E35 e posteriormente é feita a composição síncrona entre os subsistemas (G3||G4), o que resulta na planta local GL3. Este fato é observado também para as especificações 7 e 8, que têm eventos comuns aos modelos G1, G2 e G3. Assim, é feita também a composição síncrona entre as especificações (E7||E8), que resulta na especificação E78 e entre os subsistemas (G1||G2||G3), chegando-se a planta local GL6.

Tabela 4-2 – Eventos comuns entre modelos e especificações

	G_1		G_2		G_3		G_4		G_5			G_6			G_L
	E_liga	S_liga	S_liga	E_desl	T_M	F_rb1	I_giro	F_giro	I_teste	T_OK	T_NOK	T_R	T_S	F_rb2	
E_1	X	X	X	X											G_{L1}
E_2										X	X	X	X		G_{L2}
E_3						X	X								G_{L3}
E_5						X	X	X							G_{L3}
E_4							X	X				X	X	X	G_{L4}
E_6							X	X	X	X	X				G_{L5}
E_7		X	X		X										G_{L6}
E_8	X			X	X										G_{L6}
E_9					X			X	X						G_{L7}

As composições síncronas foram realizadas utilizado o *software* Grail (Reiser *et al.*, 2006), da seguinte forma:

$G_{L1} = G_1 \| G_2$
$G_{L2} = G_5 \| G_6$
$G_{L3} = G_3 \| G_4$
$G_{L4} = G_4 \| G_6$
$G_{L5} = G_4 \| G_5$
$G_{L6} = G_1 \| G_2 \| G_3$
$G_{L7} = G_3 \| G_4 \| G_5$

Na seqüência foi feita a composição síncrona das plantas locais com as especificações da planta que continham eventos comuns, gerando as especificações locais, da seguinte forma:

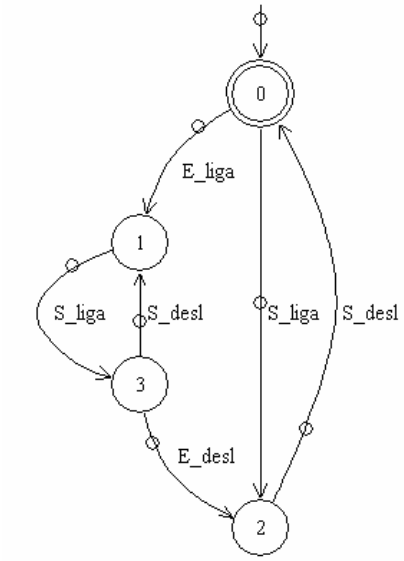
$E_{L1} = G_{L1} \| E_1$
$E_{L2} = G_{L2} \| E_2$
$E_{L3} = G_{L3} \| E_{35}$ onde $E_{35} = E_3 \| E_5$
$E_{L4} = G_{L4} \| E_4$
$E_{L6} = G_{L5} \| E_6$
$E_{L7} = G_{L6} \| E_{78}$ onde $E_{78} = E_7 \| E_8$
$E_{L9} = G_{L7} \| E_9$

Por fim, calculou-se a máxima linguagem controlável para cada supervisor, denotada $SupC(G_{Li},E_{Li})$. Assim, chega-se aos supervisores locais não reduzidos que serão implementados no CLP de forma a garantir o cumprimento das especificações de controle.

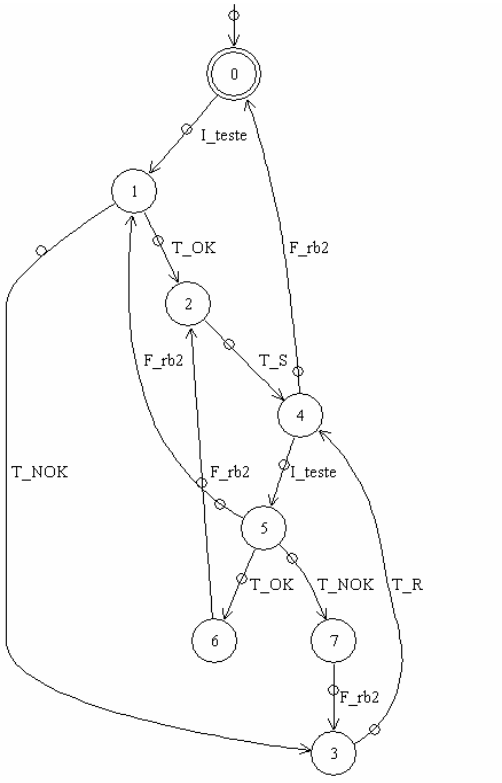
Cabe ressaltar que para o cálculo dos supervisores modulares locais não foram utilizadas as restrições físicas, devido ao fato de que estas agregam eventos de vários subsistemas físicos, o que acaba forçando a composição de praticamente todos os subsistemas e comprometendo a modularidade da planta, inviabilizando, desta forma, o uso da abordagem modular local.

Na Figura 4.23 são mostrados os supervisores modulares locais. Os números indicados entre parêntesis são, respectivamente, os números de estados e de transições de cada supervisor. Assim, S_{L1} (4,6), S_{L2} (8,12), S_{L35} (4,4), S_{L4} (4,5), S_{L6} (4,5), S_{L78} (8,21) e S_{L9} (12,32).

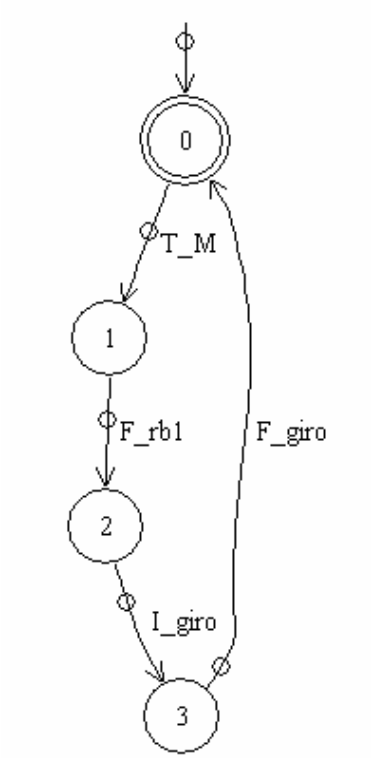
SL1: ESTEIRA e SENSOR



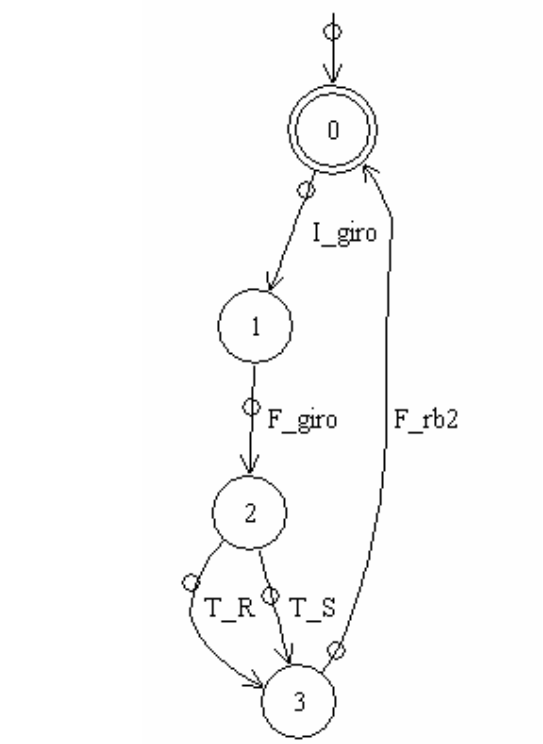
SL2: ROBÔ 2 e TESTE



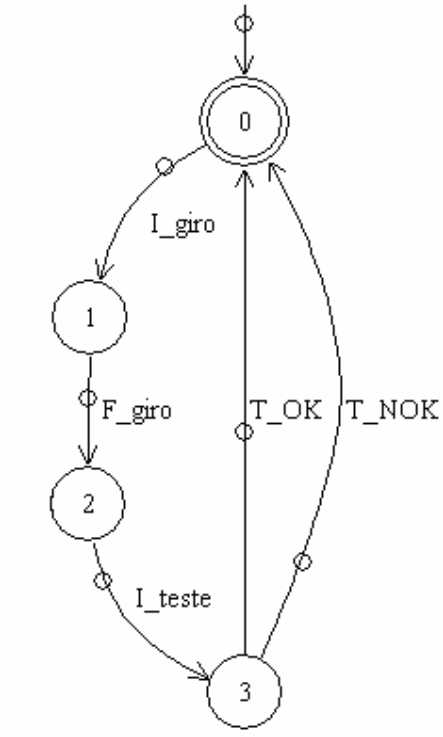
SL35: ROBÔ 1 e MESA



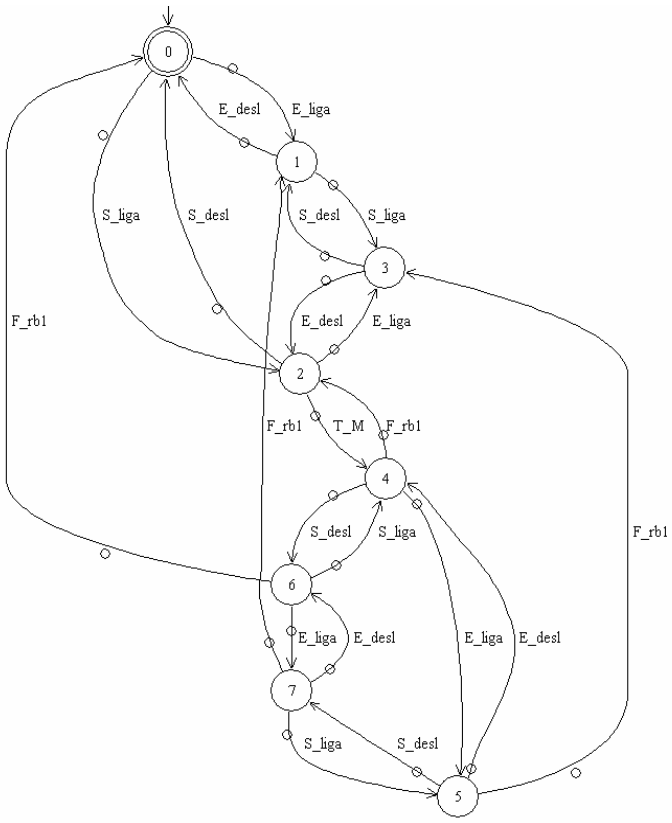
SL4: MESA E ROBÔ 2



SL6: MESA e TESTE



SL78: ESTEIRA e SENSOR e ROBÔ 1



SL9: MESA e TESTE E ROBÔ 1

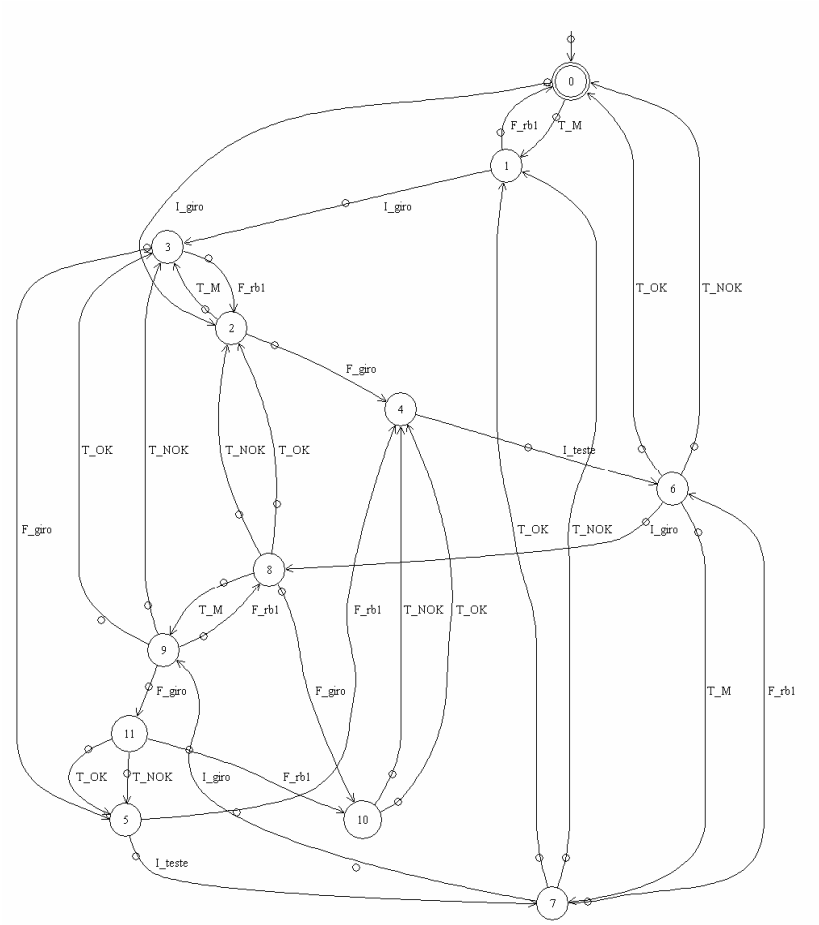


Figura 4.23 – Supervisores Modulares

Na seção 5.3 é apresentada a implementação dos supervisores modulares locais não reduzidos para o problema apresentado no início deste capítulo.

Porém, antes da implementação, é necessário assegurar a modularidade local do conjunto de supervisores locais, garantindo que a ação conjunta de todos os supervisores é não bloqueante, conforme demonstrado em (Queiroz e Cury, 2000). A verificação da modularidade local consiste em realizar a composição síncrona de todos os supervisores locais da seguinte forma:

$$S = S_{L1}||S_{L2}||S_{L35}||S_{L4}||S_{L6}||S_{L78}||S_{L9}$$

Após a composição síncrona verificou-se que o autômato resultante desta composição é Trim, isto é, não tem estados bloqueantes. Assim, pode-se afirmar que os supervisores locais são modulares entre si.

4.7 Conclusões

Neste capítulo foi apresentada a síntese de supervisores tendo por base um problema motivador para a célula flexível de manufatura didática objeto deste trabalho. No processo de síntese, foram seguidos os passos propostos por Ramadge e Wonham (1989). Foi adotada uma seqüência de funcionamento e foram definidos os eventos que fazem parte do problema e sua classe (controláveis e não controláveis). Na seqüência foi feito o modelamento dos dispositivos, das especificações e das restrições físicas utilizando-se a teoria de autômatos.

Na etapa de síntese foram aplicadas as duas abordagens apresentadas no Capítulo 2. Como resultado da síntese a partir da abordagem monolítica, obteve-se um supervisor não bloqueante e minimamente restritivo, o qual coordena o sistema todo. A partir da aplicação da abordagem modular local chegou-se a seis supervisores modulares locais, onde cada um coordena uma parte do sistema. Para esta segunda abordagem, é necessário verificar se a ação coordenada de todos os supervisores não gera bloqueio no sistema, para isso é feito o teste da modularidade.

No capítulo seguinte é apresentada a implementação em CLP para o supervisor monolítico e para os supervisores modulares locais obtidos anteriormente.

5 IMPLEMENTAÇÃO DOS SUPERVISORES NO CLP

Neste capítulo é apresentada primeiramente a proposta de implementação do supervisor monolítico no CLP Siemens Step7-300, em linguagem LADDER, utilizando-se uma estrutura dividida em blocos de programa. É mostrado também o *software* desenvolvido para a geração automática de código do supervisor monolítico para o CLP utilizado. Na seqüência do capítulo é apresentada a implementação dos supervisores modulares locais para o mesmo CLP, utilizando-se também uma estrutura dividida em blocos de programa, o que facilita, em ambos os casos, a programação, o entendimento e eventuais alterações no programa do CLP.

5.1 Implementação do Supervisor Monolítico

A partir do processo de síntese do supervisor monolítico, apresentado na Seção 4.5, é realizada a implementação em CLP. O CLP utilizado para a programação foi o Siemens da família Step7_300, e a CPU 312IFM. Esta CPU possui 10 entradas e 06 saídas digitais, as quais são interligadas à planta para a monitoração dos estados dos sensores e acionamento dos dispositivos. A linguagem utilizada na programação do CLP foi a Diagrama de Contatos (LADDER), por ser mais conhecida no meio industrial e de fácil edição e entendimento. Na implementação aqui apresentada, quando da ocorrência de eventos não controláveis, variáveis internas do CLP (*flags)* armazenam as informações até que elas sejam processadas. Desta forma é assegurado que a informação não será perdida sem que tenha sido processada pelo supervisor. A metodologia aqui utilizada difere daquela proposta por Fabian e Hellgren (1998), pois estes utilizam a implementação com base na detecção de flancos positivos para os eventos não controláveis. No trabalho de Queiroz *et al.* (2001), a implementação também difere, pois naquele é utilizada a implementação modular local, onde existe a habilitação de desabilitação de eventos em cada supervisor local.

Neste trabalho, propõe-se que a implementação do supervisor seja feita em quatro blocos de programa, de forma a facilitar a organização:

- bloco de gerenciamento (OB1)
- bloco do supervisor (FC1)
- bloco dos eventos (FC2)
- bloco das saídas (FC3)

5.1.1 Programação do bloco OB1

O bloco OB1 é necessário no CLP utilizado, pois ele gerencia o ciclo do programa. Em sua rotina de programação são definidos quais blocos de função (FC) serão executados, conforme a Figura 5.1.

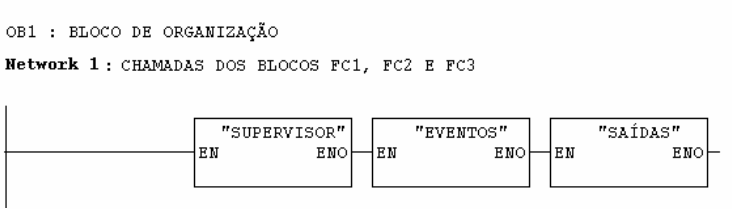


Figura 5.1 - Programação do bloco OB1 em LADDER.

5.1.2 Programação do bloco FC1- Supervisor

Na programação do bloco FC1 foi feita a implementação do supervisor monolítico. Desta forma, cada estado do supervisor corresponde a uma memória interna do CLP, que é ativada ou desativada de acordo com a seqüência de eventos que ocorre na planta e que é reconhecida pelo supervisor. O supervisor é mostrado em parte na Figura 5.2, onde cada estado foi nomeado no bloco de programação com o sufixo “S”. Assim, o estado “0” do supervisor passou a ser simbolizado por “S0” na programação do CLP. Para que o supervisor transite no autômato de “S0” para “S1”, é necessário a ocorrência do evento “E_liga”. Da mesma forma, para transitar de “S1” para “S2”, é necessário a ocorrência do evento “S_liga”.

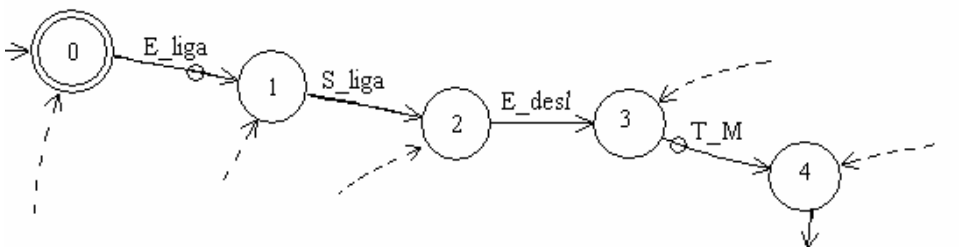


Figura 5.2 - Parte inicial do supervisor obtido.

Para uma melhor estruturação do programa do CLP, a geração de eventos controláveis e não controláveis é feita no bloco FC2 e o acionamento das saídas é feito no bloco FC3. Na seqüência descreve-se como deve ser realizada a programação do bloco FC1 em função do tipo dos eventos.

Conforme ilustrado na Figura 5.3, a partir do comando “INÍCIO”, o supervisor ativa o estado “S0” (*Network* 1). Neste estado, o supervisor fica aguardando a ocorrência do evento

E_liga (*Network* 2), que por ser um evento controlável, deve ser disparado pelo próprio programa do CLP. Para tanto, é necessário que o motor da esteira tenha sido efetivamente ligado, o que é feito no bloco FC3 (Figura 5.4 – *Network* 1). Assim, após a ativação da saída do CLP que corresponde ao motor da esteira, a geração do evento "E_liga" é feita no bloco FC2 (Figura 5.5 – *Network* 1). Desta forma, com a geração do evento "E_liga", o supervisor transita no autômato para o estado "S1" e fica aguardando a ocorrência do próximo evento da seqüência definida pela lógica de controle.

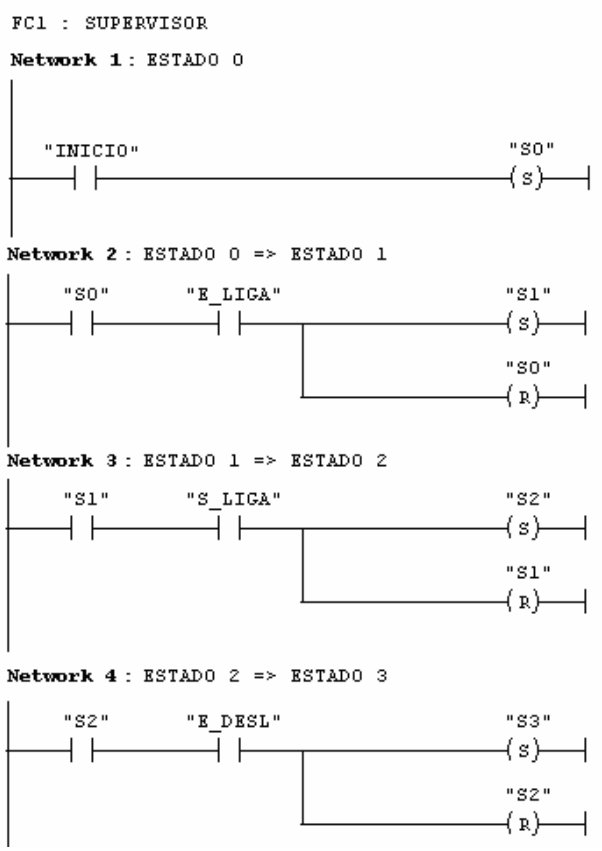


Figura 5.3 - Programação do bloco FC1 em LADDER.

5.1.3 Programação do bloco FC3- Saídas

Neste bloco são definidas as ações do supervisor, de acordo com o estado definido no bloco FC1. Assim, conforme visto, quando o supervisor estiver no estado "S0", a saída que ativa o motor da esteira será acionada, conforme a Figura 5.4 (Network 1). Este comando envia um sinal para a saída física do CLP, conforme a Tabela 4-1, vista anteriormente.

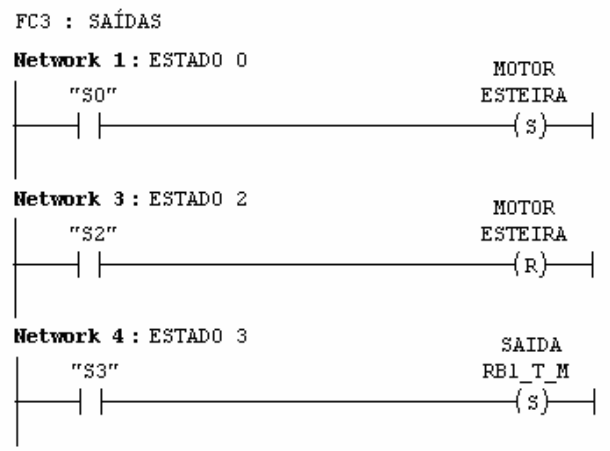


Figura 5.4 - Programação do bloco FC3 em LADDER.

5.1.4 Programação do bloco FC2- Eventos

Neste bloco são ativados ou desativados os eventos de acordo com sinais de entrada e saída do CLP. Desta forma, ao ser ligado o motor da esteira, o evento E_liga será ativado, indicando para o supervisor (bloco FC1) que o motor da esteira foi ativado. Ao mesmo tempo, o evento que informava ao supervisor que o motor da esteira estava desligado é desativado, conforme a Figura 5.5.

Uma particularização na implementação deste problema foi adotada quando da ocorrência de dois eventos controláveis em um estado qualquer do supervisor. Na Figura 5.6 representamos uma seqüência iniciada no estado 6, que transita direto para o estado 11, pois os dois eventos (E_liga e I_giro) são disparados. A programação do CLP quando da ocorrência dos dois eventos controláveis a partir do estado 6 é feita conforme vemos na Figura 5.7 e na Figura 5.8.

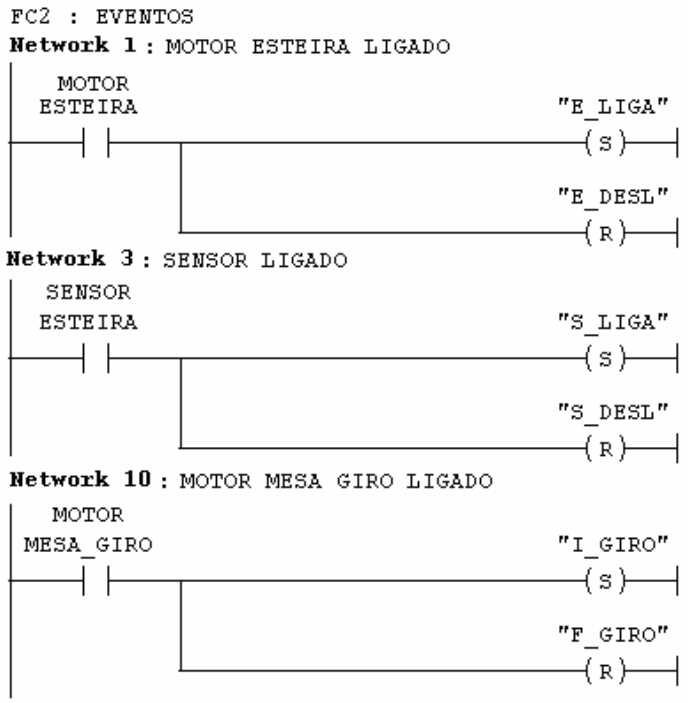


Figura 5.5 - Programação do bloco FC2 em LADDER.

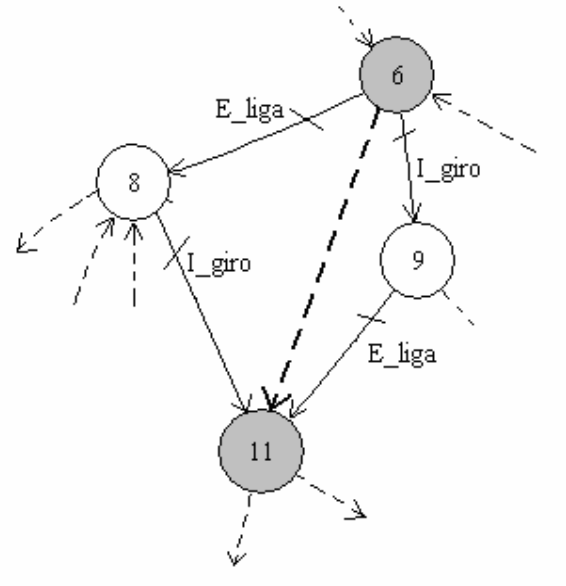


Figura 5.6 - Disparo simultâneo de eventos controláveis.

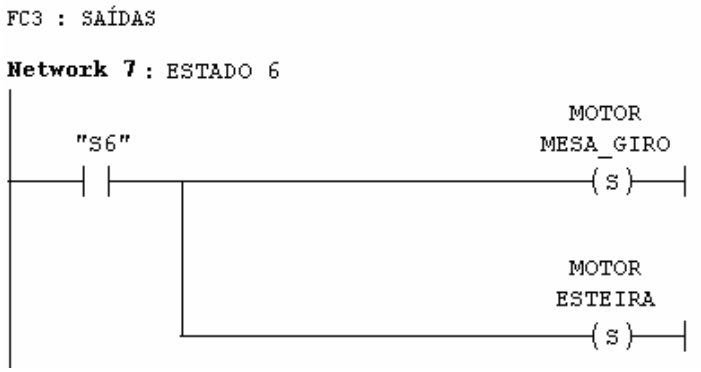


Figura 5.7 - Disparo simultâneo de eventos no bloco FC3.

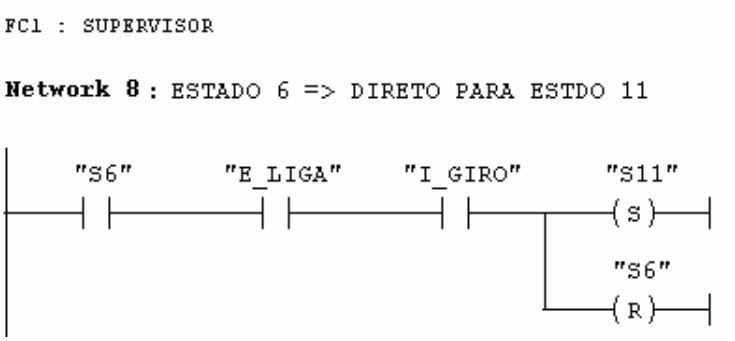


Figura 5.8 - Disparo simultâneo de eventos no bloco FC1.

Generalizando, caso ocorram dois ou mais eventos controláveis simultâneos em um estado qualquer, o autômato transita direto para o estado no qual tais eventos convergem.

Nos estados do supervisor em que existem eventos controláveis e não controláveis, dá-se prioridade no tratamento dos não controláveis, pois estes são espontaneamente gerados em função da dinâmica da planta e devem ser tratados imediatamente após a sua ocorrência. Na programação do CLP, essa prioridade é dada de acordo com a disposição das linhas do programa, ou seja, como a execução do programa é cíclica, os eventos não controláveis são executados nas linhas anteriores aos eventos controláveis. Por exemplo, no estado 29, os eventos não controláveis T_OK e T_NOK têm prioridade de execução em relação ao evento controlável T_M (Figura 5.9).

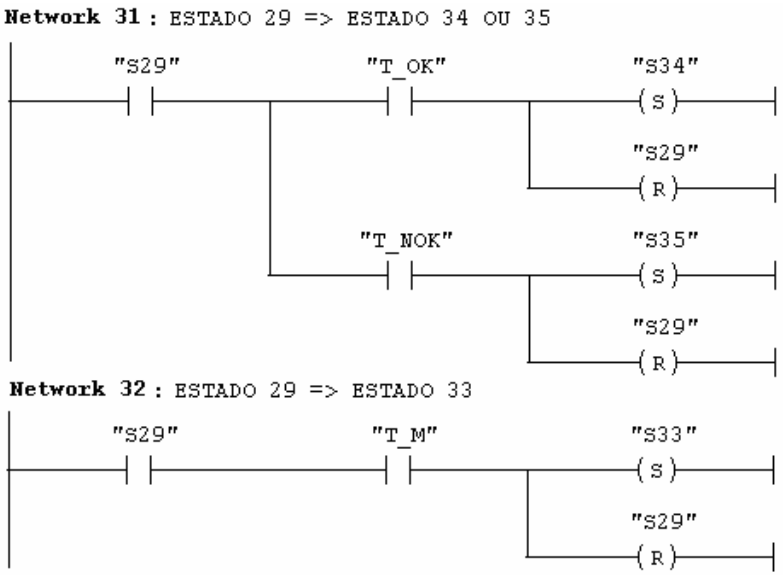


Figura 5.9 - Programação de eventos não controláveis (bloco FC1).

5.2 Ferramenta para geração de código monolítico (GPACLP)

Com o intuito de auxiliar na tarefa de transcrição do autômato obtido para o supervisor no código de programação do CLP, foi desenvolvida uma ferramenta de geração automática de código para o CLP. Esta ferramenta foi desenvolvida com o auxílio de dois bolsistas do Programa de Educação Tutorial – PET Engenharia Elétrica.

O GPACLP é um programa (ou tradutor) no qual, a partir de um arquivo de formato conhecido da Teoria de Controle Supervisório oriundo dos *softwares* Grail ou TCT, é possível gerar o programa em STL (*Statement List*) para ser implementado diretamente em CLPs das famílias S7-300/S7-400 da Siemens.

O programa gerado para o CLP é formado por um bloco de organização (OB1), três blocos de função: Supervisor (FC1), Saídas (FC3), Eventos (FC2), conforme já descrito na seção 5.1. Além disso, é gerada uma tabela de símbolos na qual são associados automaticamente os nomes dos eventos às entradas/saídas do CLP e os estados do supervisor às memórias internas. Na Figura 5.10 é apresentado um diagrama que elucida como é o procedimento para a geração automática de código para o CLP.

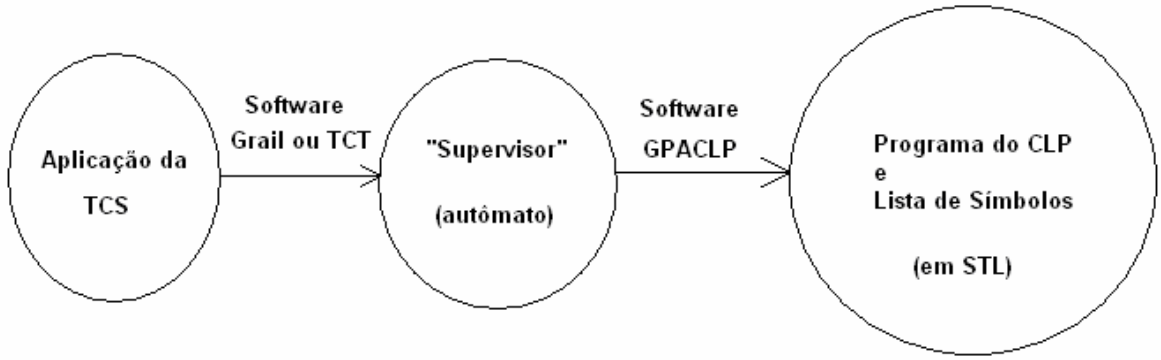


Figura 5.10 – Geração automática de código para o CLP

Na seqüência apresenta-se um exemplo do uso do programa GPACLP para uma parte do supervisor monolítico da célula em questão, que envolve apenas o sensor e a esteira de entrada. Na Figura 5.11 é mostrado o autômato que representa o supervisor gerado a partir do *software* Grail e a seu arquivo texto correspondente.

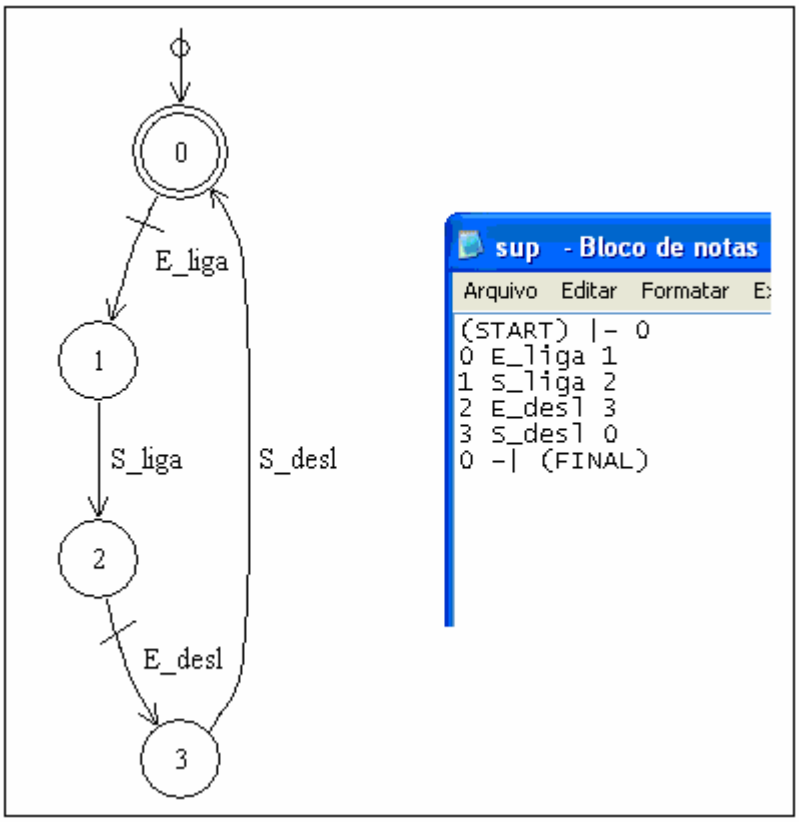


Figura 5.11 – Autômato do Supervisor e seu arquivo texto

A descrição do uso do GPACLP é feita com maiores detalhes no APÊNDICE A, o qual consiste em um tutorial de utilização deste *software*.

O primeiro passo consiste em abrir o arquivo referente ao supervisor (arquivo no formato do Grail ou do TCT). Ao abrir este arquivo a ferramenta mostra a listagem de todos os eventos do supervisor no campo “Eventos”. Deve-se então selecionar o evento desejado e configurá-lo de acordo com a sua natureza. Assim, na Figura 5.12 o evento ‘E_liga’ é configurado como um evento controlável (ver Tabela 4-1) e no campo Evento no CLP é informado qual será o símbolo gerado na tabela de símbolos do programa do CLP. No exemplo, o símbolo ‘Q_Est’ será associado automaticamente à primeira saída do CLP, nesse caso, a saída Q124.0. Além disso, o evento ‘E-liga’ refere-se a ligar a esteira e, portanto, deve-se marcar o campo SET[2] (instrução *SET* do CLP).

Ainda na Figura 5.12, o evento ‘E_desl’ é configurado também como um evento controlável (ver Tabela 4-1) e no campo “Evento no CLP” é informado o mesmo símbolo ‘Q_Est’. Porém, o evento ‘E-desl’ refere-se a desligar a esteira, portanto, deve-se marcar o campo RESET[3] (instrução *RESET* do CLP).

Após essa configuração, é necessário ‘Salvar’ as alterações.

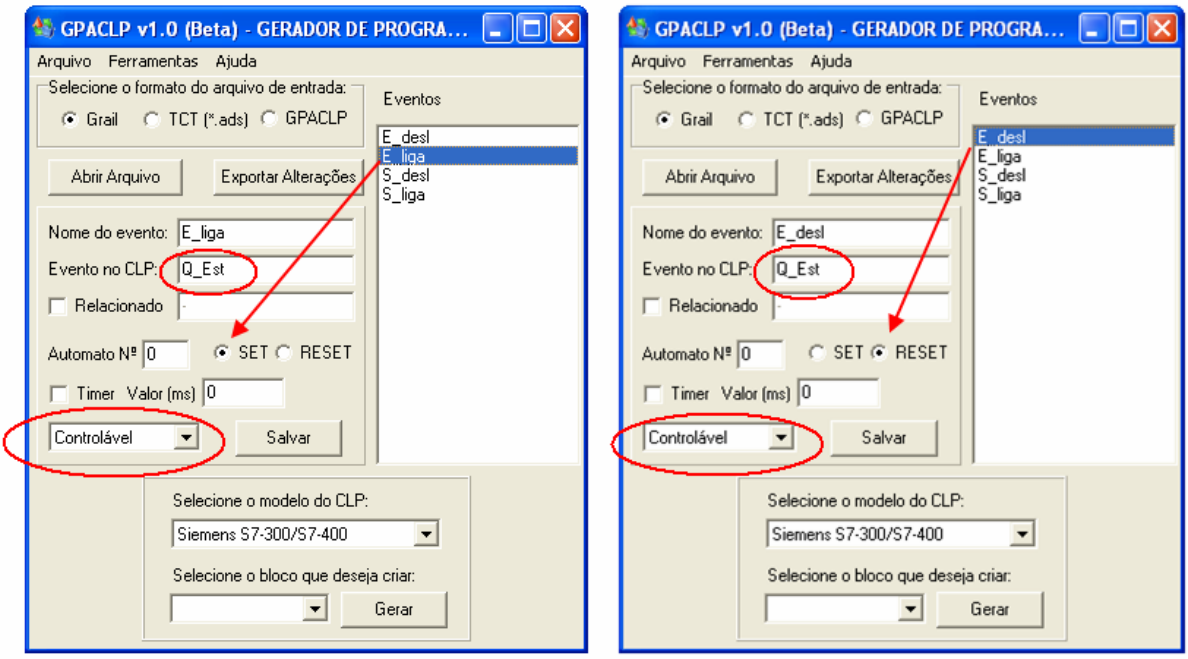


Figura 5.12 – Configuração dos Eventos Controláveis

O mesmo procedimento é adotado para os eventos não controláveis, porém, o símbolo informado no campo “Evento no CLP” será associado a uma entrada do CLP. Para

[2] A instrução SET ativa a saída correspondente do CLP.

[3] A instrução RESET desativa a saída correspondente do CLP.

estes eventos (Figura 5.13) também é informado, de acordo com a Tabela 4-1, se o evento será ativado *(SET)* ou desativado *(RESET)*.

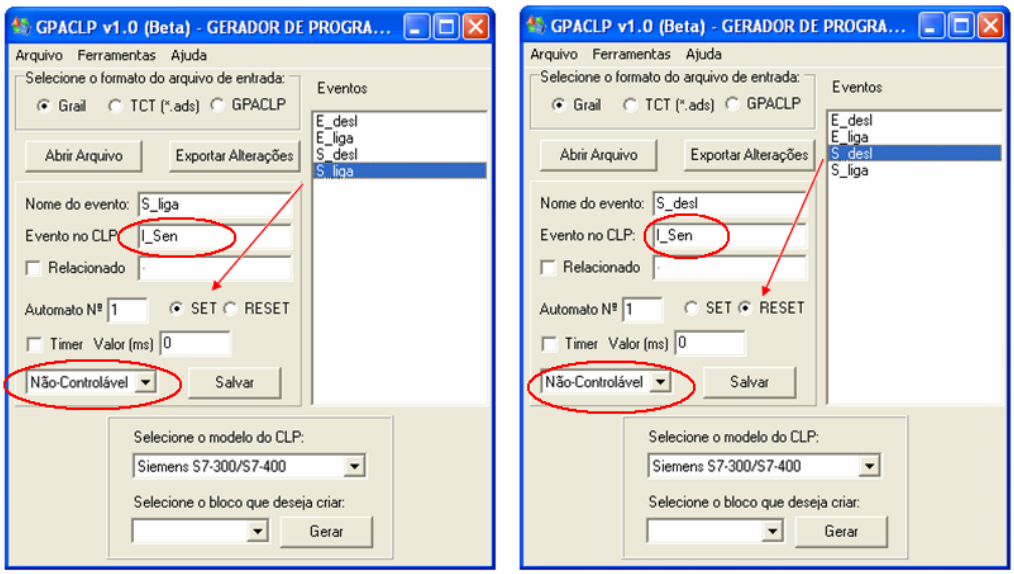


Figura 5.13 – Configuração dos Eventos não controláveis

Após terem sido configurados os eventos, é iniciado o procedimento para a geração do programa do CLP. Deve-se selecionar então o "modelo do CLP" e qual o bloco a ser gerado (ver Figura 5.14). Esta opção permite que no caso de haver alguma alteração das especificações do projeto, apenas o bloco do supervisor seja gerado novamente, agilizando o processo.

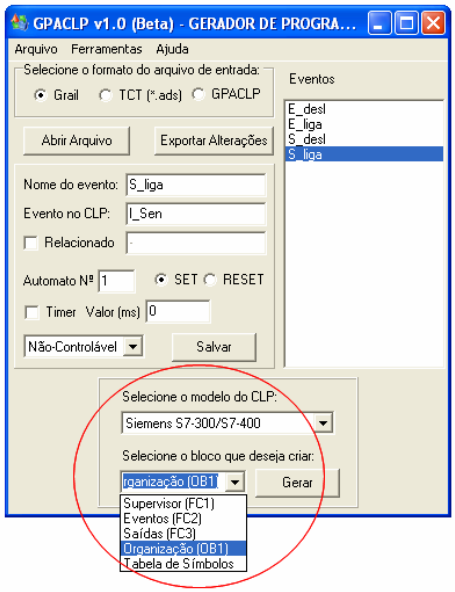


Figura 5.14 – Geração dos Blocos de Programa

Na Figura 5.15 são mostrados os blocos gerados e gravados na pasta correspondente ao projeto. É importante que essa pasta seja criada para facilitar a importação desses arquivos pelo *software* Simatic Manager, que é usado para a programação do CLP.

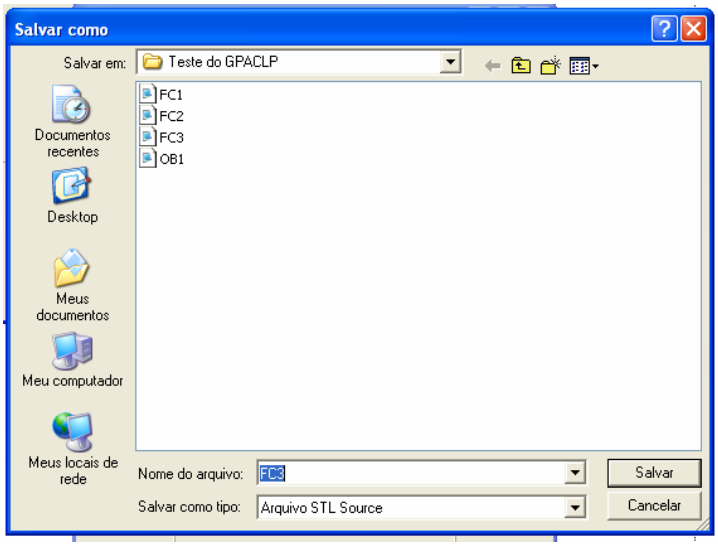


Figura 5.15 – Pasta de armazenamento dos blocos de programa

Por fim é gerada a tabela de símbolos do programa (Figura 5.16), a qual deverá ser editada posteriormente para ajustar as entradas e saídas de acordo com os equipamentos ligados ao CLP, pois o programa gera os endereços das entradas e saídas automaticamente a partir a entrada I124.0 e da saída Q124.0. Como referência para esta edição usa-se a Tabela 4-1.

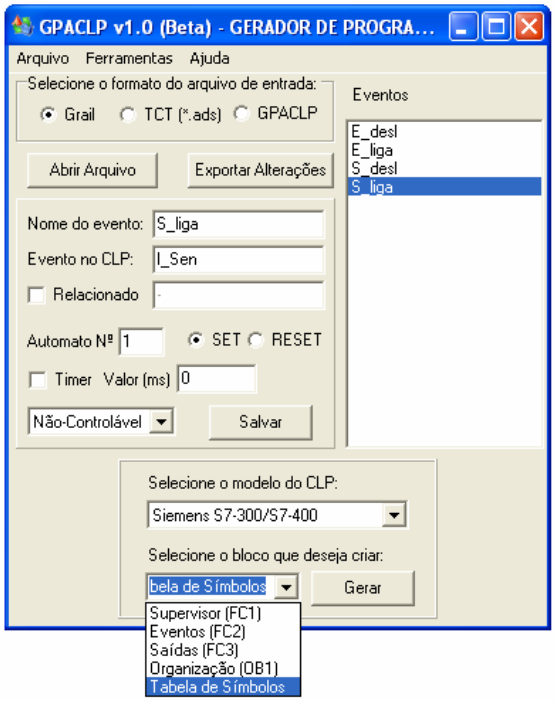


Figura 5.16 – Geração da tabela de símbolos

Na Figura 5.17 é mostrada a tabela de símbolos gerada e gravada na pasta correspondente ao projeto.

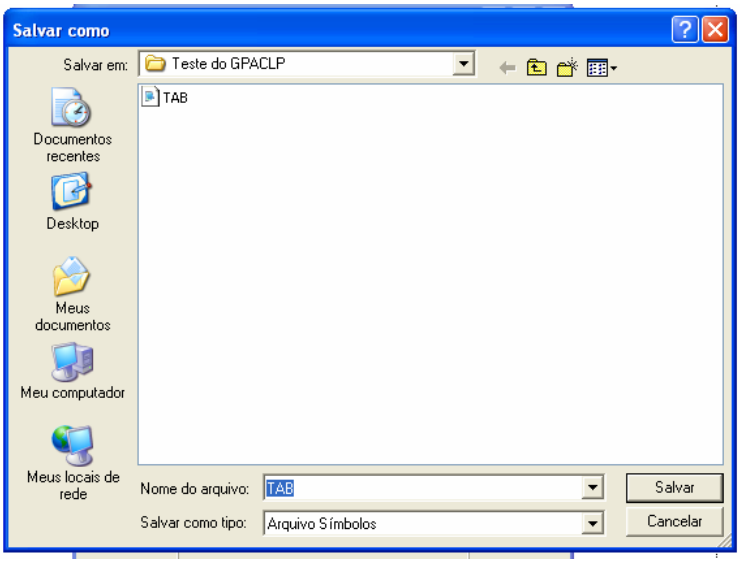


Figura 5.17 – Pasta de armazenamento da tabela de símbolos

Após estes procedimentos, os arquivos gerados estão prontos para serem compilados pelo programa do CLP. Na Figura 5.18 são mostrados os blocos gerados (Arquivos *AWL*) e a tabela de símbolos (Arquivo *SEQ*). *AWL* é a extensão da linguagem do CLP e *SEQ* é a extensão dos arquivos contendo os símbolos.

Figura 5.18 – Arquivos gerados pelo *software* GPACLP

Na seqüência, deve-se abrir o Simatic Manager (Siemens 2002a, 2002c) e importar primeiramente a tabela de símbolos gerada (Figura 5.19):

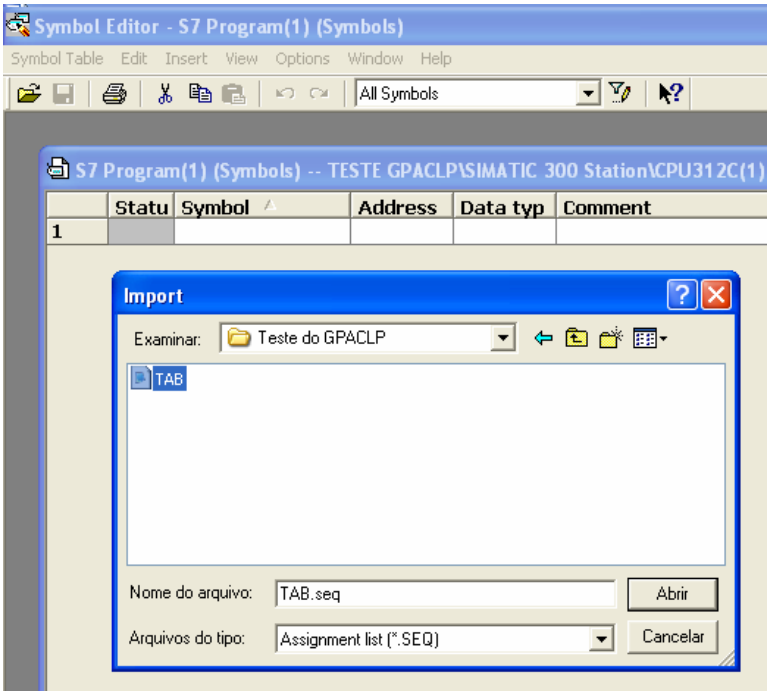


Figura 5.19 – Tabela de símbolos gerada

Na Figura 5.20 é mostrada a tabela de símbolos importada para o Simatic Manager. É importante lembrar que é necessário ajustar as entradas e saídas de acordo com os equipamentos da planta (vide Tabela 4-1), ou seja, é preciso verificar se fisicamente a saída Q124.0 do CLP está ligada ao motor da esteira e também se na entrada I124.0 está conectado o sinal do sensor da esteira. Conforme já mencionado, o *software* atribui automaticamente as entradas e saídas a partir do primeiro endereço e os ajustes têm que ser feitos posteriormente.

S7 Program(1) (Symbols) -- TESTE GPACLP\SIMATIC 300 Station\CPU312C(1)

	Statu	Symbol	Address	Data typ	Comment
1		SUPERVISOR	FC 1	FC 1	
2		EVENTOS	FC 2	FC 2	
3		SAÍDAS	FC 3	FC 3	
4		Q_Est	Q 124.0	BOOL	
5		I_Sen	I 124.0	BOOL	
6		E_desl	M 1.0	BOOL	
7		E_liga	M 1.1	BOOL	
8		S_desl	M 1.2	BOOL	
9		S_liga	M 1.3	BOOL	
10		S0	M 1.4	BOOL	
11		S1	M 1.5	BOOL	
12		S2	M 1.6	BOOL	
13		S3	M 1.7	BOOL	
14					

Figura 5.20 – Tabela de símbolos importada para o Simatic Manager

Após a importação dos símbolos, procede-se à criação dos blocos de programa, o que é feito em duas etapas. Primeiramente são importados do programa de conversão os arquivos gravados na pasta do projeto. Estes arquivos são gravados como ‘arquivos fonte’ no Simatic Manager. Em uma segunda etapa, é necessário compilar estas informações dos arquivos fonte para que o programa do CLP interprete o código gerado para a linguagem de programação STL. Como a linguagem de implementação adotada neste trabalho foi o LADDER, após a compilação deve-se proceder à mudança na representação de STL para LADDER, o que será explicado mais adiante.

O procedimento para a criação do arquivo fonte a partir da pasta onde foram armazenadas as informações do *software* de conversão GPACLP é ilustrado através das figuras 5.21 e 5.22.

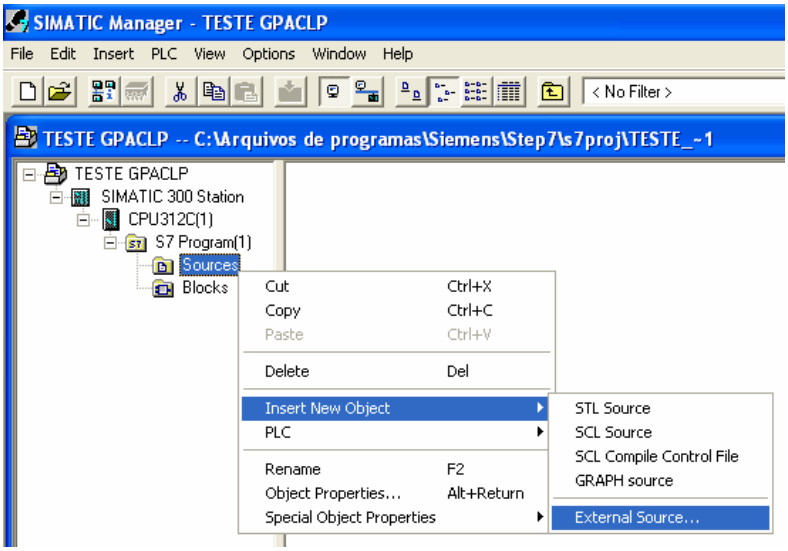


Figura 5.21 – Geração dos arquivos fonte

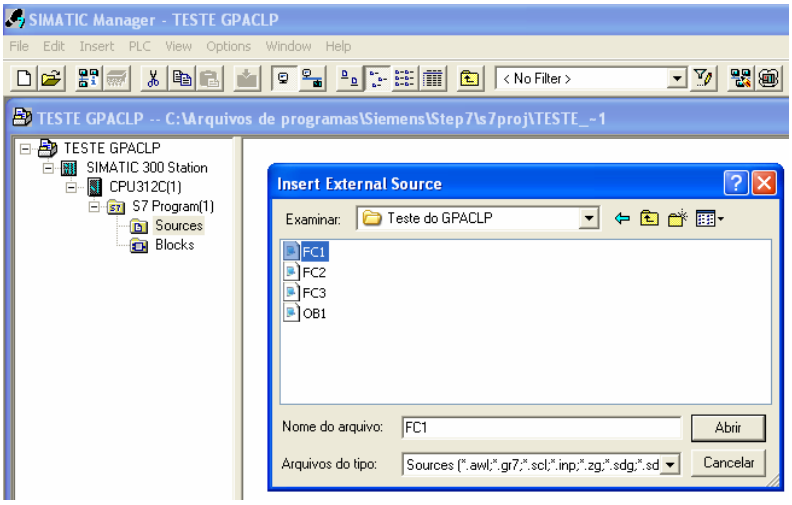


Figura 5.22 – Pasta de armazenamento do GPACLP

Na Figura 5.23 é mostrado o arquivo fonte FC1 após sua importação do programa GPACLP.

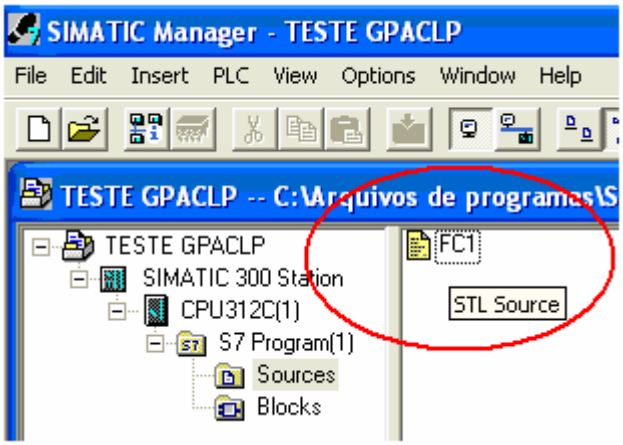


Figura 5.23 – Arquivo fonte FC1 importado da Pasta do GPACLP

Deve-se repetir o processo para todos os blocos. Na Figura 5.24 são mostrados todos os arquivos fonte importados. Estes arquivos fonte devem ser agora abertos e compilados.

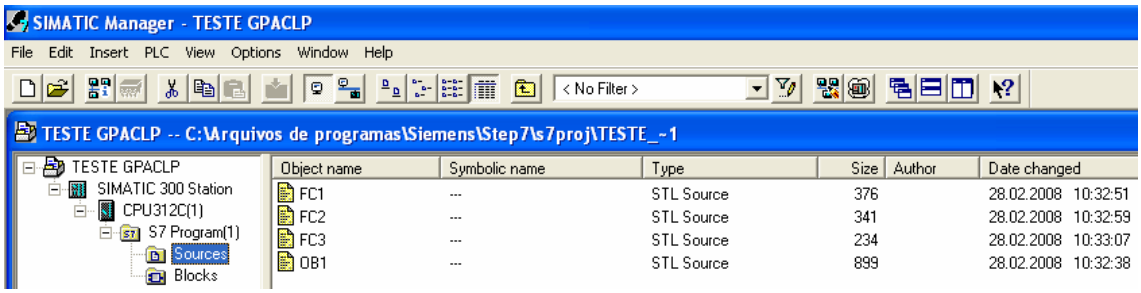


Figura 5.24 – Arquivos fonte

Para o procedimento de compilação, deve-se abrir cada bloco e clicar no ícone indicado pela seta da Figura 5.25. Como exemplo é mostrada a compilação do bloco FC1.

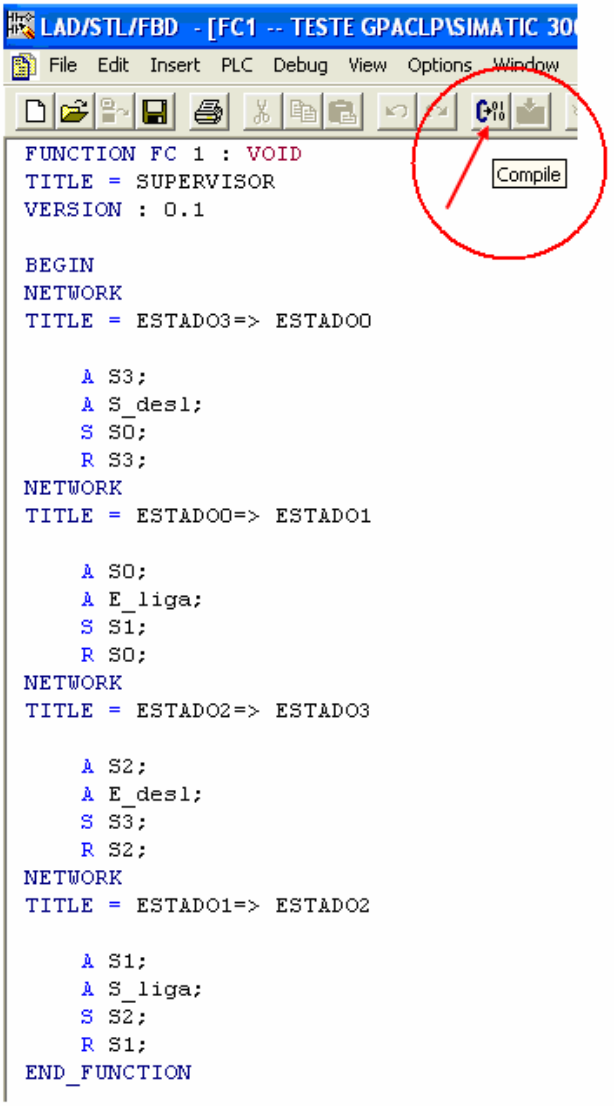


Figura 5.25 – Compilando o bloco FC1

Deve-se repetir o processo para os outros blocos e o último bloco a ser compilado deverá ser o OB1, caso contrário ocorrerá erros em função da chamada dos outros blocos. Na Figura 5.26 são mostrados os quatro blocos compilados.

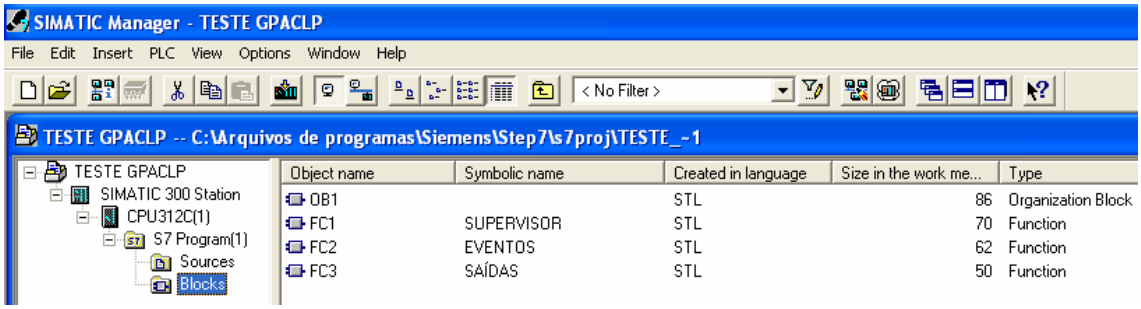


Figura 5.26 – Blocos após a compilação

Para a visualização do programa que será implementado no CLP, é necessário abrir cada bloco e alterar o modo de visualização de STL para LADDER (LAD). Conforme indicado na Figura 5.27, deve-se selecionar a opção LAD no *menu View*.

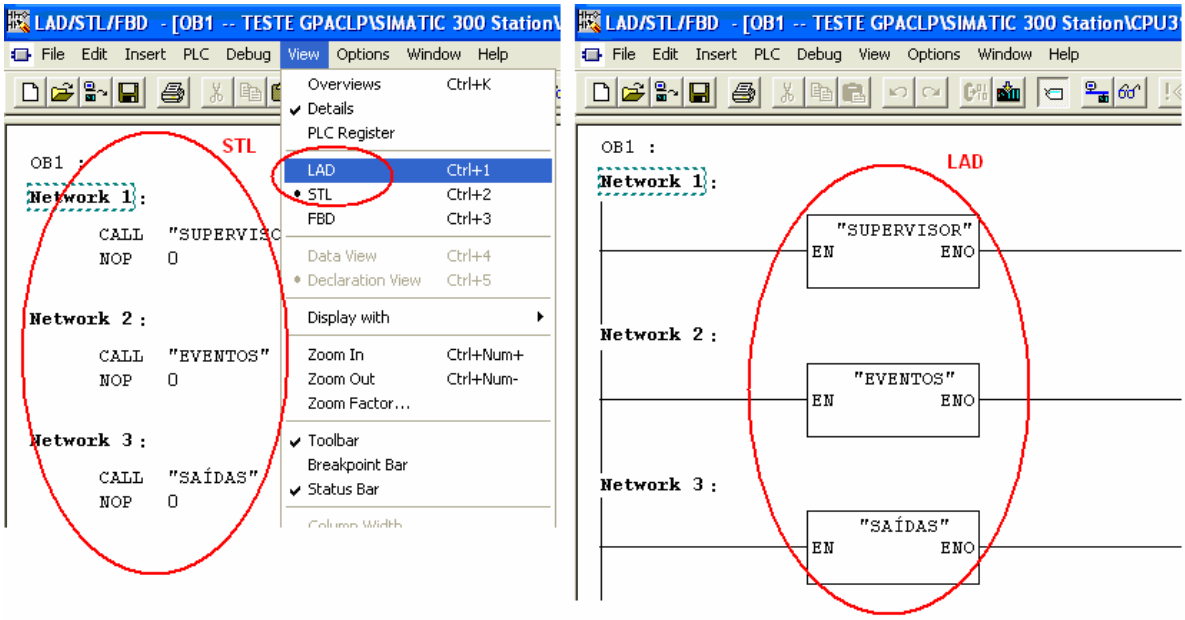


Figura 5.27 – Alteração do modo de visualização de STL para LAD

O mesmo procedimento deve ser adotado para os outros blocos. Na Figura 5.28 são mostrados os FC1, FC2 e FC3 em LADDER.

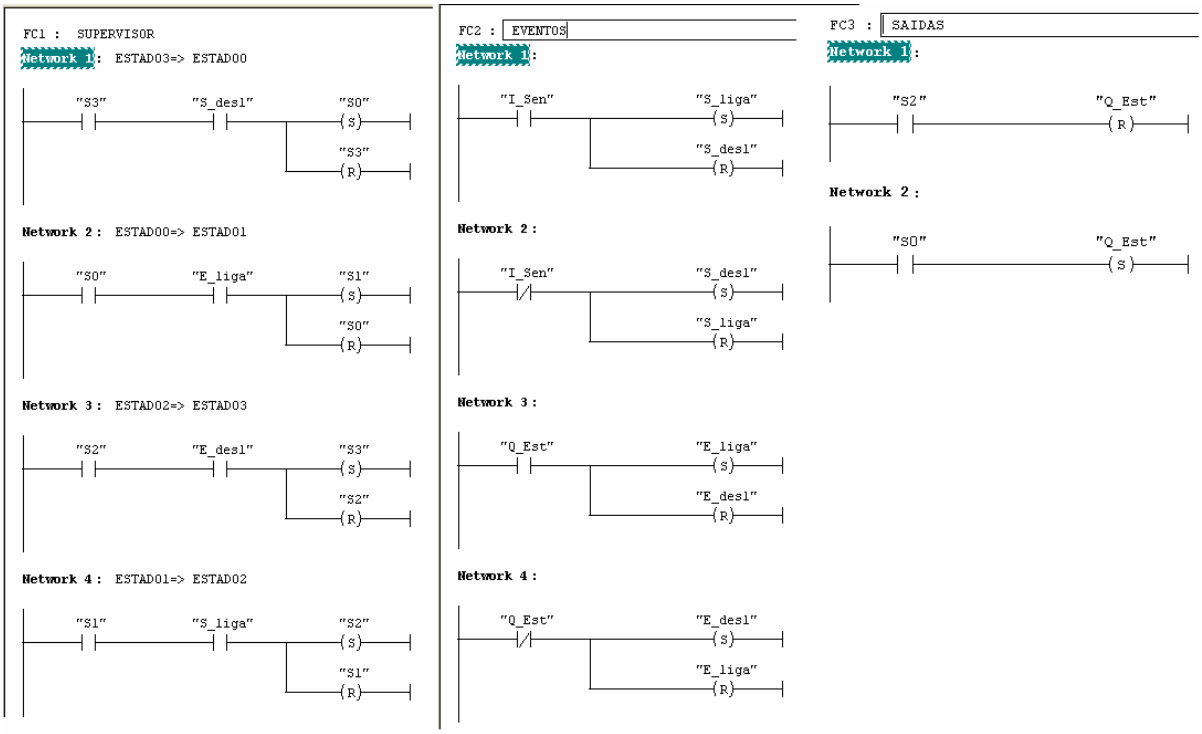


Figura 5.28 – Blocos FC1, FC2 e FC3 em LADDER

Por fim, o programa pode ser transferido para o CLP e testado o seu funcionamento na célula. Para isso é necessário fazer o *download* dos blocos do programa (Siemens, 2002a, 2002c).

5.3 Implementação dos Supervisores Modulares Locais

A partir do processo de síntese dos supervisores modulares locais, apresentado na seção 4.6, é realizada a implementação em CLP.

A metodologia aqui empregada é proposta no trabalho de Queiroz *et al.* (2001), onde é abordada a implementação modular local. Porém, na proposta original os autores implementam os supervisores modulares reduzidos, razão pela qual existe o bloco chamado Sistema Produto, conforme ilustrado na Figura 5.29. Na implementação aqui proposta, não foi utilizado o Sistema Produto, pois foram implementados os supervisores modulares não reduzidos, não sendo necessário o bloco de Sistema Produto. A linha tracejada da Figura 5.29 mostra os blocos em questão.

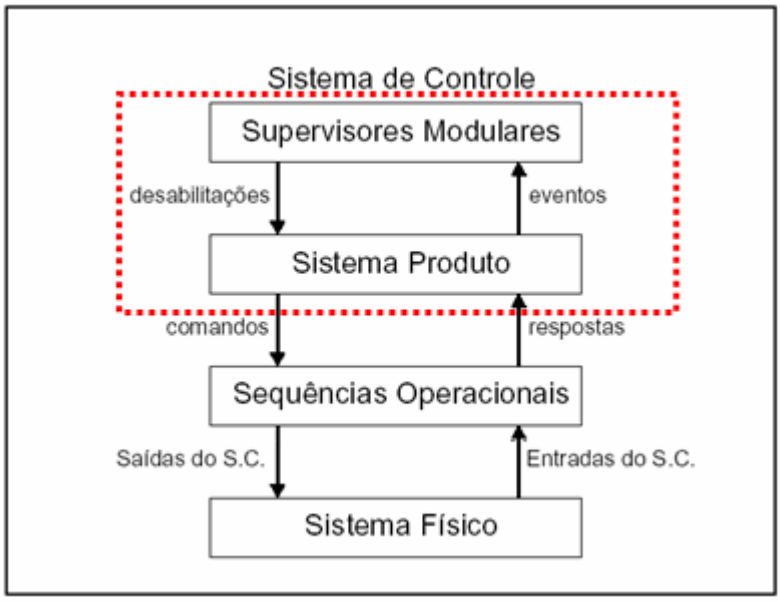


Figura 5.29 – Modelo de implementação adaptado de Queiroz et al. (2001)

O CLP utilizado para a programação foi o mesmo descrito na seção 5.1 e a linguagem de programação do CLP também foi a Diagrama de Contatos (LADDER). Para a implementação, foram criados blocos de programa de forma a facilitar a organização:

- bloco de gerenciamento
- blocos dos supervisores locais
- blocos dos subsistemas físicos
- bloco das desabilitações

Esta estrutura de programação, dividida em blocos, é também conhecida como programação estruturada (Siemens, 2002a, 2002c). O bloco de gerenciamento, chamado de OB (*Organization Block*) é responsável pela execução dos blocos de função. Os blocos de função, chamados de FC (*Function*), são responsáveis pela execução das rotinas de programação. Na seqüência será explicada em detalhes a programação de cada bloco.

5.3.1 Programação do bloco OB1

O bloco OB1 é necessário no CLP utilizado, pois ele gerencia o ciclo do programa. Em sua rotina de programação são definidos quais blocos de função (FCs) serão executados, conforme a Figura 5.30. A ordem de execução destes blocos não afeta a coordenação do sistema devido ao ciclo de varredura do CLP.

Observa-se na *Network 1* da Figura 5.30 um comando chamado *FIRST CHECK*, cuja função é inicializar todos os supervisores locais e os subsistemas no estado inicial, garantido assim a sincronização da planta com os supervisores. Este comando é gerado a partir da detecção de um flanco positivo da memória M127.7. Ao inicializar o CLP esta memória está desligada e sua associação a um contato fechado na linha de programa (lógica inversora), provoca uma borda de subida na memória M127.6 durante o primeiro ciclo de varredura do CLP. A instrução —(P)— é responsável por detectar a mudança de estado da memória M127.7.

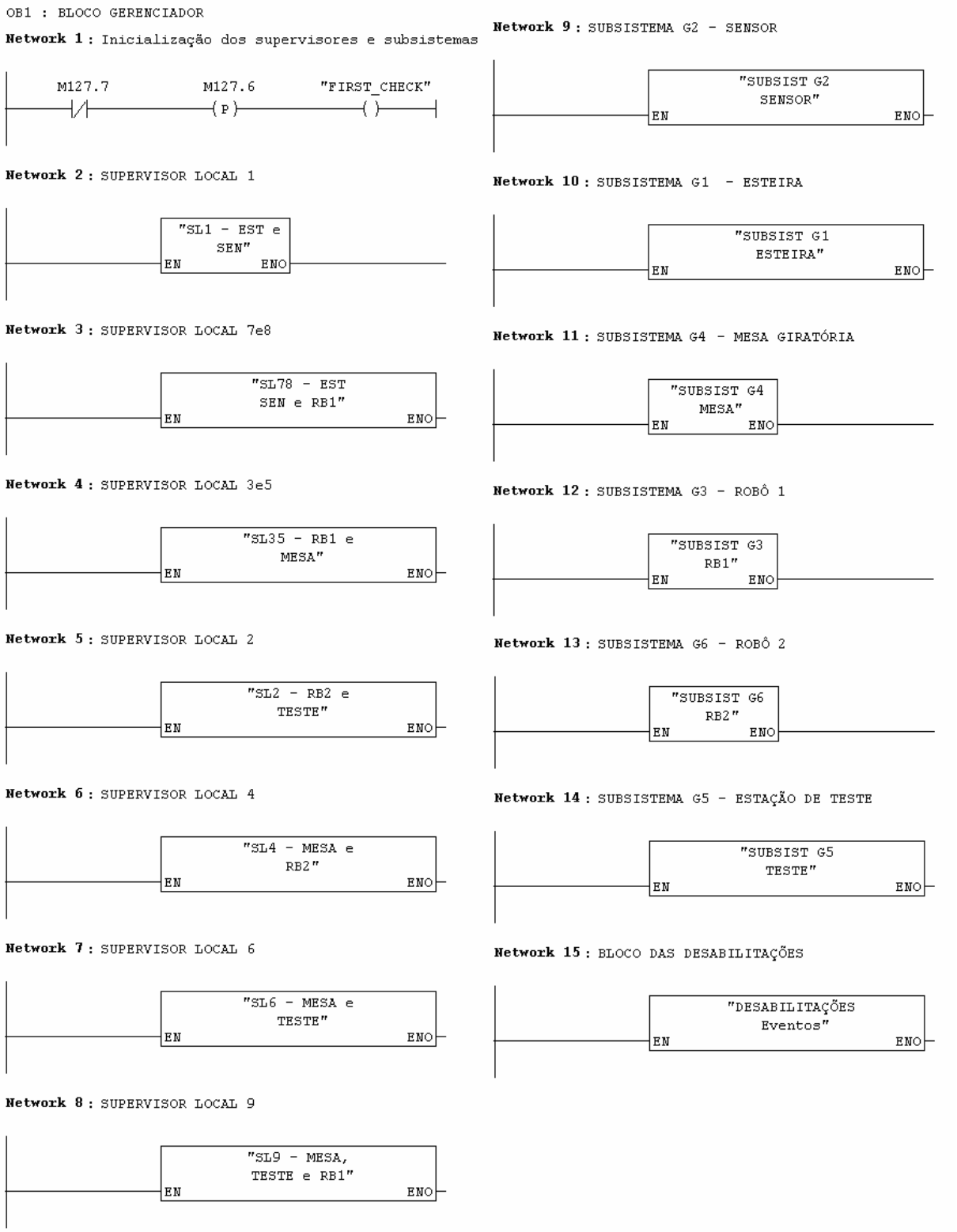


Figura 5.30 - Programação do bloco OB1 em LADDER.

5.3.2 Programação dos blocos FCs - Supervisores locais

Na programação dos blocos dos supervisores modulares locais, cada estado do supervisor corresponde a uma memória interna do CLP, que é ativada ou desativada de acordo com a seqüência de eventos. Para o supervisor local 1, S_{L1}, mostrado na Figura 5.31, onde

cada estado foi nomeado no bloco de programação com o prefixo “S1” seguido do número do estado. Assim, o estado “0” do supervisor passou a ser simbolizado por “S1,0” na programação do CLP (Figura 5.32).

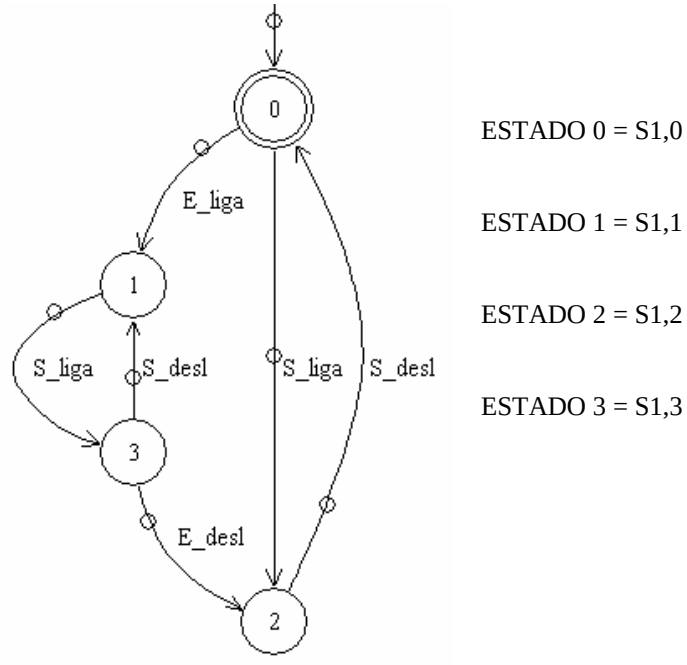


Figura 5.31 - Supervisor local 1 – SL1.

Conforme ilustrado na *Network 1*da Figura 5.32, a partir do comando *FIRST CHECK*, o supervisor local 1 ativa o estado “S1,0”. Neste estado o supervisor fica aguardando a ocorrência do evento E_liga (*Network* 2) para mudar de estado (transitar no autômato). O evento E_liga, que será ativado no subsistema G1 (Bloco FC7) é controlável, ou seja, uma ação de controle (desabilitação) é associada a este evento. No bloco FC7 também é gerada a ação de controle para acionar a saída que liga o motor da esteira. Quando essa saída é acionada, ela ativa o evento E_liga, conforme visto na Figura 5.34, o que permite que o supervisor 1 transite do estado “S1,0” para o “S1,1”.

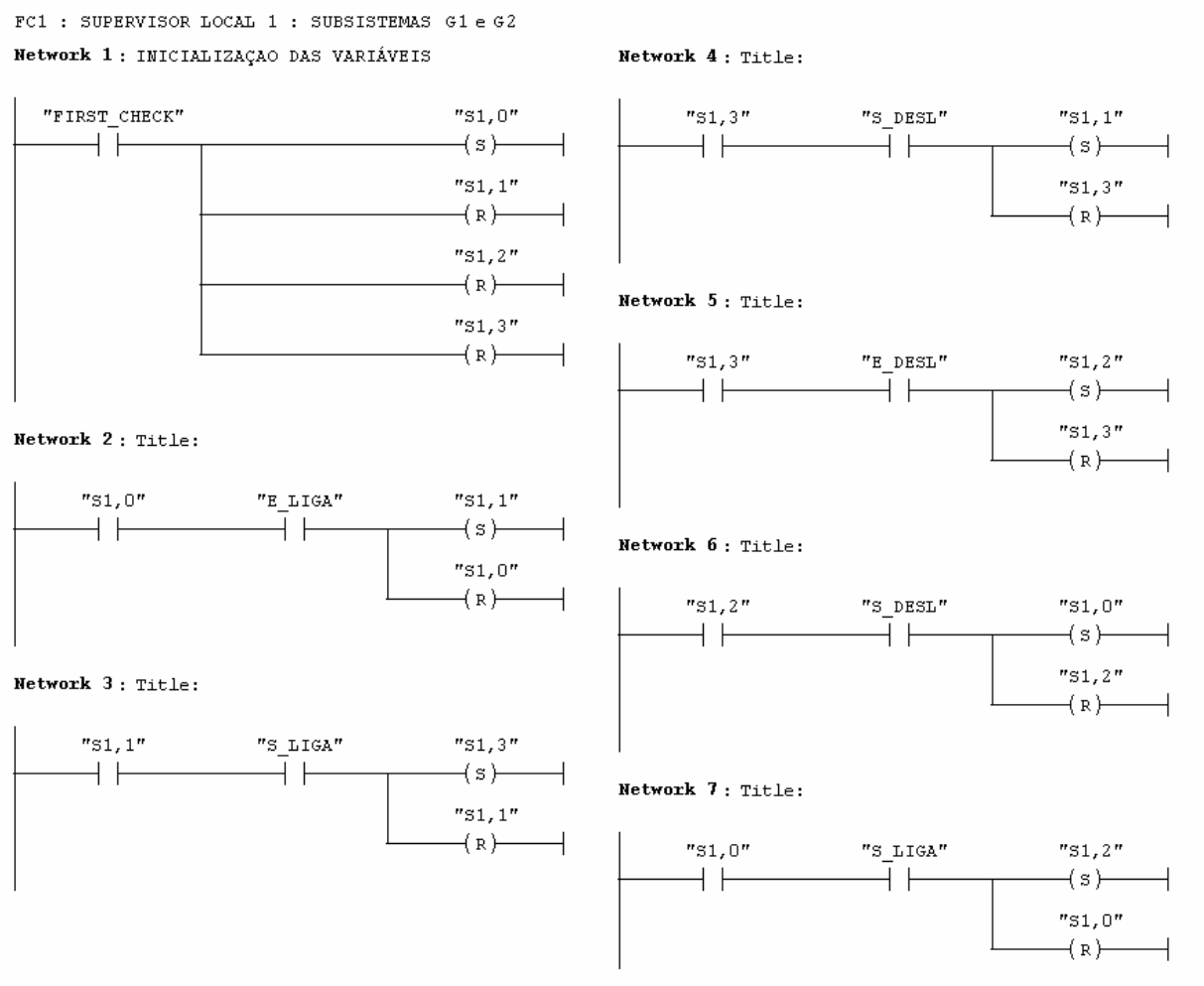
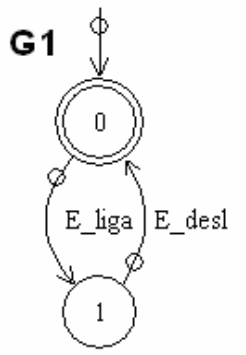


Figura 5.32 - Programação do SL1 em LADDER.

5.3.3 Programação do blocos FCs - Subsistemas

Na programação dos blocos dos subsistemas, cada estado também corresponde a uma memória interna do CLP, que é ativada ou desativada de acordo com a seqüência de eventos. Para o subsistema G1, mostrado na Figura 5.33, cada estado foi nomeado no bloco de programação com o sufixo “G1” seguido do número do estado. Assim, o estado “0” do subsistema passou a ser simbolizado por “G1,0” na programação do CLP.



ESTADO 0 = G1,0

ESTADO 1 = G1,1

Figura 5.33 – Subsistema G1 – Esteira

Assim, conforme a Figura 5.34, para que o subsistema G1 evolua de estado e execute a seqüência operacional para ligar a esteira (o que é realizado pelo comando *SET* para a saída correspondente ao motor da esteira) é necessário que o evento correspondente à esta ação de controle esteja habilitado, o que é processado no bloco das desabilitações (FC20). Na Figura 5.35 (*Network 1*), observa-se que quando o supervisor local 1 estiver no estado "S1,0", a desabilitação correspondente é o evento d_E_desl. Ainda na Figura 5.35, na *Network 2*, observa-se que se o evento d_E_liga não é desabilitado, permitindo então que a esteira seja ligada.

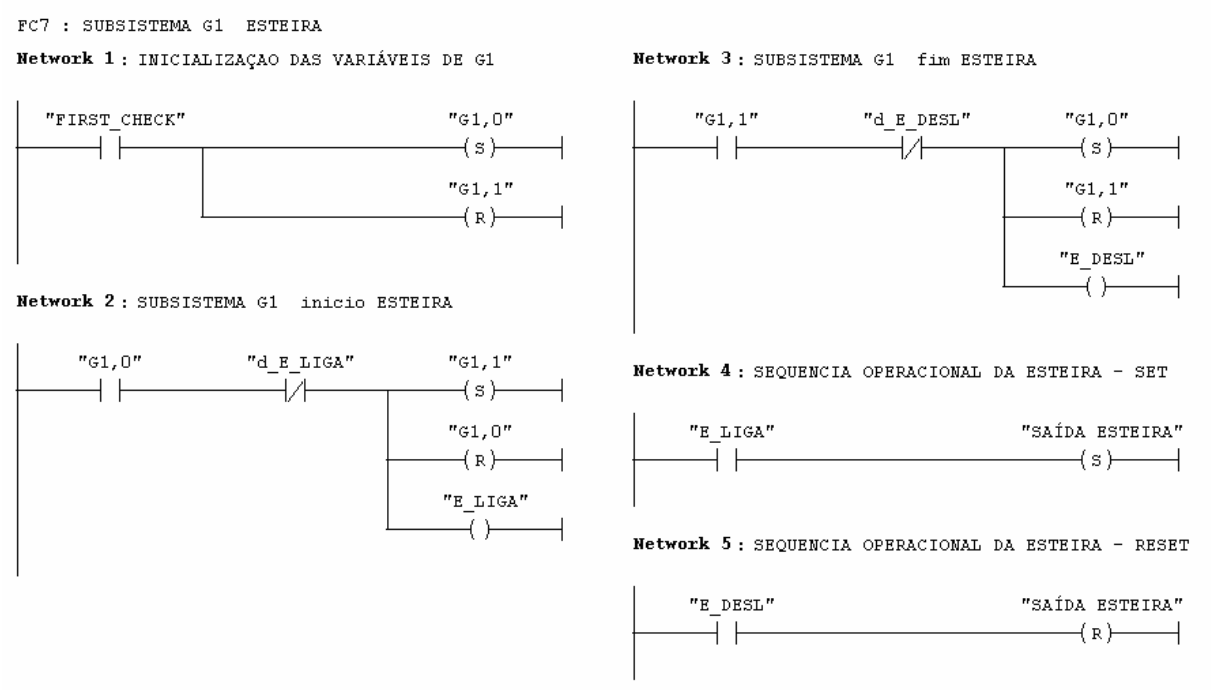


Figura 5.34 - Programação do bloco FC7 em LADDER.

5.3.4 Programação do bloco de Desabilitações (FC20)

Neste bloco são desabilitados os eventos controláveis, de acordo com o estado dos supervisores locais. Desta forma, a desabilitação do evento E_desl será ativada nos estados dos supervisores S_{L1} E S_{L78}, conforme a Figura 5.35.

No caso do subsistema G2, que modela o comportamento do sensor, seu processamento é feito unicamente com base na geração dos eventos não controláveis a partir do sinal do sensor. Desta forma, como não existe uma ação de controle do CLP sobre este subsistema, não é feita nenhuma desabilitação para seus eventos. A programação do subsistema G2 é feita no bloco FC8, conforme mostrado na Figura 5.36.

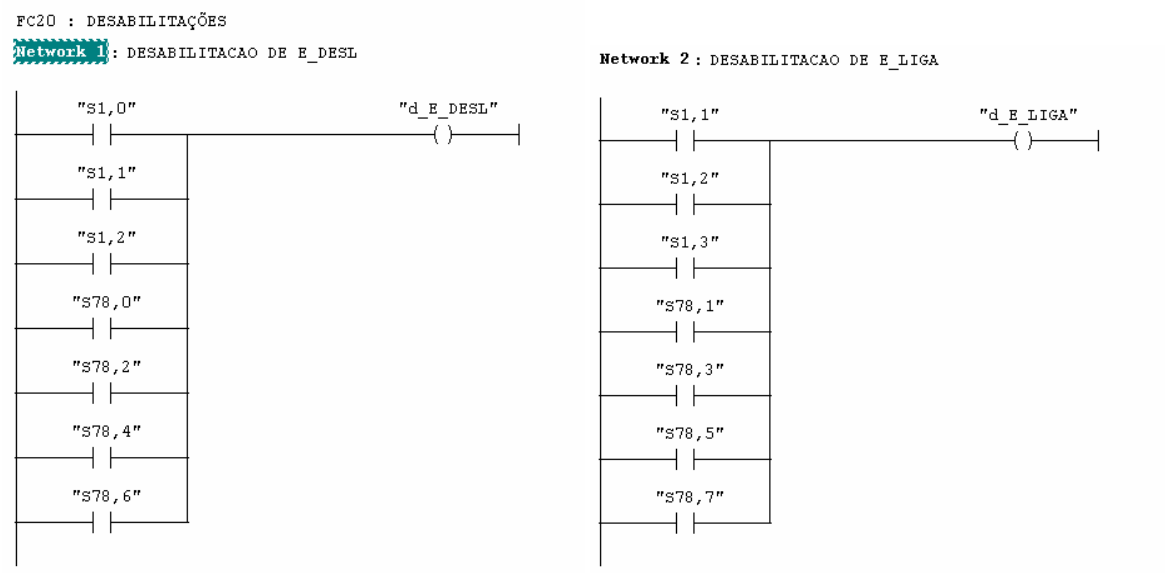


Figura 5.35 - Programação do bloco FC20 em LADDER.

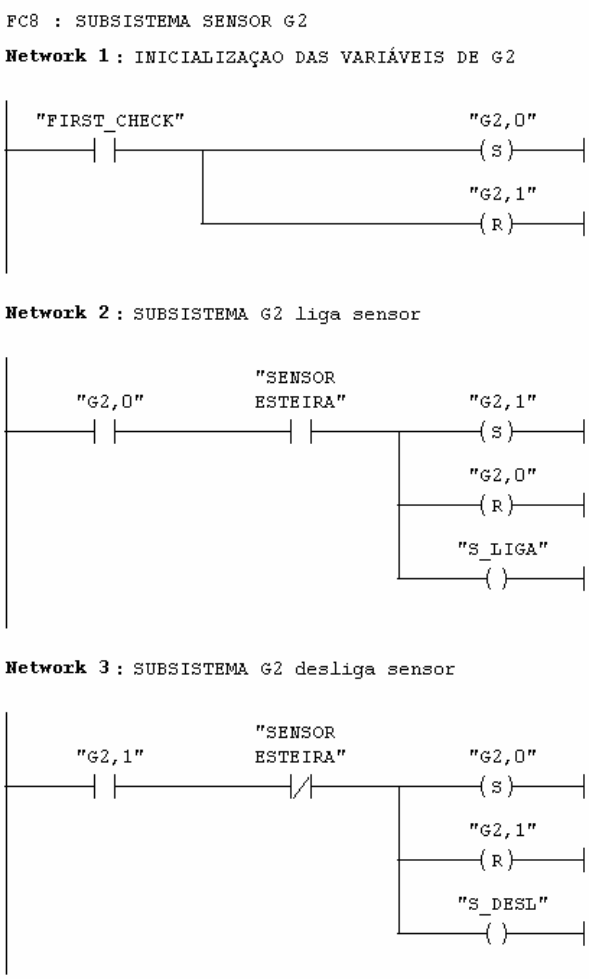


Figura 5.36 – Programação do Bloco FC8 em LADDER

Após a finalização da programação dos blocos, é necessário transferir o programa para o CLP (Siemens, 2002a, 2002c).

5.4 Conclusões

Neste capítulo foram apresentadas as duas propostas de implementação para o CLP Siemens Step7-300, a implementação monolítica e a implementação modular local. Ambas foram desenvolvidas em linguagem LADDER, utilizando-se uma estrutura dividida em blocos de programa, o que facilita a programação.

Foi apresentado também o *software* desenvolvido para a geração automática de código para o CLP utilizado. Este *software* ainda apresenta apenas a funcionalidade para o supervisor monolítico.

Destaca-se ainda que a metodologia empregada neste trabalho para a implementação dos supervisores modulares locais é proposta por Queiroz *et al.* (2001), tendo sido feita uma adaptação do procedimento originalmente adotado pelos autores, por se tratar neste trabalho de supervisores modulares locais não reduzidos.

6 CONCLUSÕES

O emprego da Teoria de Controle Supervisório e suas abordagens Monolítica e Modular Local para a célula flexível de manufatura didática modelada a partir de autômatos e sua implementação em linguagem de CLP mostrou-se bastante viável, uma vez que na indústria a solução de problemas é feita com base na experiência do projetista, e geralmente, não é utilizado um procedimento formal. Esse exemplo real serve de modelo para aplicação e solução de novos problemas, visto que apresenta um procedimento sistemático, baseado na TCS.

A organização do programa em blocos de funções permite que o programa seja melhor visualizado, tornando mais simples a tarefa de reprogramação e/ou solução de problemas durante o funcionamento da célula, uma vez que é possível identificar em que bloco ocorreu o problema e solucioná-lo sem a necessidade de percorrer todo o programa do CLP.

Na implementação do supervisor monolítico verificou-se a necessidade de mais memória para o CLP, questão que havia sido discutida ao longo do trabalho, uma vez que ocorre a explosão de estados do supervisor. Apesar de o sistema modelado ser relativamente pequeno, na abordagem monolítica percebeu-se que a composição de restrições físicas agregava mais estados ao autômato final, ficando este mais próximo da seqüência real.

Outro aspecto relevante na implementação monolítica é quando da inclusão ou alteração de subsistemas ou especificações no projeto do supervisor, pois há a necessidade de refazer todo o projeto. Por esta razão, um dos objetivos alcançados foi o desenvolvimento do *software* de geração automática de código, o qual torna a tarefa de implementação monolítica bem mais rápida. Desta forma, para a solução do problema da célula de manufatura didática, a metodologia monolítica de síntese e implementação apresentada neste trabalho mostrou-se apropriada.

Como alternativa para a implementação de supervisores em CLP, foram implementados os supervisores modulares locais, cuja abordagem foi apresentada ao longo do trabalho. A abordagem modular local permite explorar a modularidade dos subsistemas da planta em função das especificações impostas pelo projetista. Dessa forma, cada supervisor tem ação em uma parte da planta, não sendo necessário refazer todo o projeto quando for alterada alguma especificação ou incluído um novo subsistema. Aplica-se o método de síntese e gera-se um novo supervisor local, o qual será implementado no CLP juntamente com os que já existiam. O uso das restrições físicas foi abolido na abordagem modular local, por afetar a

modularidade dos subsistemas, visto que as restrições físicas agregam eventos de várias especificações diferentes.

Na abordagem modular local é preciso fazer o teste para a verificação da modularidade dos supervisores, ou seja, a verificação de não bloqueio para a ação coordenada de todos os supervisores. Os autores apontam o teste da modularidade e a existência de bloqueio como os principais desafios para a aplicação da técnica para sistemas de grande porte, visto que o teste tem complexidade computacional que cresce exponencialmente com o número de subsistemas e especificações.

Quando comparada com a implementação do supervisor monolítico, a implementação modular local economizou, no exemplo visto, em torno de 10% da capacidade de memória do CLP, o que é significativo para sistemas de grande porte.

Com os resultados obtidos neste trabalho, verifica-se que a proposta inicial de se utilizar uma metodologia formal para desenvolver de forma sistemática o programa de controle a partir do modelo da planta e das especificações e sua posterior implementação em CLP foi atingida de forma satisfatória. Com base nestes resultados, pretende-se disseminar o uso da teoria de controle supervisório em células flexíveis de manufatura industriais.

Outro aspecto relevante a ser considerado com relação à célula flexível de manufatura apresentada neste trabalho é a possibilidade de ser usada como uma importante ferramenta de ensino em diversas áreas das engenharias. Pode-se utilizá-la para discutir desde aspectos técnicos específicos de determinadas áreas, como a programação de robôs e programação de CLPs, até aspectos que envolvam conteúdos de diversas disciplinas.

Além disso, o uso da célula possibilita que as disciplinas sejam desenvolvidas na forma de projetos, nos quais os alunos devem trabalhar em equipe de forma a resolver problemas com os quais possivelmente se defrontarão na sua futura vida profissional. Através desta metodologia, pode-se facilitar o desenvolvimento de habilidades importantes para a formação de um bom profissional, tais como: capacidade de trabalho em equipe, habilidades em gerenciamento de projetos e visão crítica.

Neste sentido, também o ensino de realidade virtual (Hounsell e Pimentel, 2003) pode ser feito com auxílio da célula de manufatura didática proposta. Com a inclusão de dispositivos de comunicação em rede (Ethernet, Profibus) será possível ainda a utilização da célula para ensino e pesquisa na área de Manufatura Remota (Controle via Internet), Sistemas Integrados de Manufatura, CAD/CAPP/CAM, viabilizando sua programação e supervisão remotas (Álvares e Ferreira, 2003), além das disciplinas de Mecatrônica, Robótica,

Informática Industrial, Automação da Manufatura (automação de sistemas ou automação industrial) e Sistemas Flexíveis de Manufatura, por exemplo.

6.1 Contribuições

Destacam-se como contribuições do presente trabalho:

- Integração dos equipamentos da célula de manufatura didática do Laboratório de Robótica da UDESC – Joinville;
- Implementação de uma interface para os sinais dos equipamentos e do CLP;
- Proposição e resolução de um problema prático sobre a célula em questão;
- Implementação da TCS no CLP S7_300 da Siemens, utilizando uma estrutura particionada em blocos de programa;
- Criação de um programa para conversão de código de autômatos para CLP;
- Estudos preliminares de integração da célula real com a RV;
- Publicação de trabalhos científicos divulgando a implementação de supervisores monolíticos em CLP:
    - CBA 2006 - Congresso Brasileiro de Automática (Curzel e Leal, 2006);
    - COBENGE 2006 - Congresso Brasileiro de Ensino de Engenharia (Curzel *et al.*, 2006);
- Publicação de trabalhos científicos divulgando a integração da célula de manufatura didática com a Realidade Virtual:
    - SIECI 2007 - Simposium Iberoamericano de Educación, Cibernética e Informática (Curzel *et al*. 2007a);
    - ICECE 2007 – International Conference on Engineering and Computer Education (Curzel *et al*. 2007b);
- Disseminação da aplicação da TCS junto à comunidade acadêmica;
- Elaboração de material tutorial sobre a implementação de supervisores em CLP e sobre a geração automática de código;
- Submissão de artigos divulgando a implementação de supervisores modulares locais em CLP:
    - CBA 2008 - Congresso Brasileiro de Automática;
    - INDUSCON 2008 – Conferência Internacional de Aplicações Industriais.

6.2 Trabalhos Futuros

Possibilidades de desenvolvimento para futuros trabalhos utilizando a célula de manufatura didática:

- Integração de novos dispositivos (subsistemas) na célula, tais como: sensores para identificação de cores, sistema de visão artificial, dispositivos atuadores na mesa giratória e a inclusão do Robô ABB existente no Laboratório de Robótica da UDESC – Joinville;
- Expansão da quantidade de Entradas e Saídas digitais no CLP, de forma a permitir que os novos dispositivos sejam conectados;
- Expansão da memória do CLP, permitindo a implementação de supervisores com maior número de estados;
- Verificação da necessidade de criação de novas interfaces para sinais do CLP;
- Implementação da estrutura de controle distribuída (múltiplos CLPs), onde um novo CLP será responsável pela comunicação no nível supervisório e outros CLPs serão responsáveis pelo controle da célula (Vieira, 2007);
- Implementação de uma estrutura de rede de comunicação industrial (Profibus ou Ethernet);
- Implementação de supervisores modulares locais reduzidos (Queiroz, 2004);
- Integração com a Realidade Virtual permitindo a operação virtual da célula, de forma que a programação possa ser transferida para a célula virtual, agilizando o processo de testes e verificação;
- Uso da célula para ensino e treinamento com o auxílio da Realidade Virtual, onde problemas como colisão e bloqueio podem ser identificados e corrigidos sem danificar os dispositivos reais;
- Utilização de outros *softwares* de controle, tais como *LabView*, com placas acopladas para entrada e saída dos sinais da célula e robôs;
- Utilização de *software* Supervisório para a monitoração e controle da célula, como o Elipse E3, por exemplo;
- Implementação de especificações de segurança (modo manual e automático) para a célula (Vieira, 2007);
- Implementação do(s) supervisor(es) em Microcontroladores;

REFERÊNCIAS

ÁLVARES, A.J.; FERREIRA, J.C.E., **Metodologia para Implantação de Laboratórios Remotos Via Internet na Área de Automação da Manufatura**, Anais do 2o Congresso Brasileiro de Engenharia de Fabricação (COBEF), Uberlândia, MG, 18 a 21 de maio, 2003.

ATTIÉ, S.S.; **Automação Hidráulica e Pneumática Empregando a Teoria de Sistemas a Eventos Discretos.** Dissertação (Mestrado em Engenharia Mecânica) – Centro Tecnológico, Universidade Federal de Santa Catarina. Florianópolis. 1998.

BOUZON, G.; OLIVEIRA, M. L.; VALLIM, M. B.; LACOMBE, J. P.; FREITAS, G. M.; CURY, J.E.R.; FARINES, J.M.; **CEBE: uma plataforma para experimentação real aplicada ao ensino de sistemas a eventos discretos.** Anais do Congresso Brasileiro de Automática CBA.v. 1. p. 724-729, 2004.

CARVALHO, J. R.; **Contribuições a Implementação da Estrutura de Controle Modular Local.** Mestrado (Dissertação em Engenharia de Produção e Sistemas) – Pontifícia Universidade Católica do Paraná. Curitiba, 2007.

CASSANDRAS, C.G.; LAFORTUNE S.; **Introduction to Discrete Event Systems.** Kluwer Academic Publishers. USA, 1999.

CASTRUCCI, L.C.; MORAES, C.C.; **Engenharia de Automação Industrial,** Editora LTC, Brasil, 2007.

COSTA, G. O.; **Uma Plataforma Computacional de Suporte ao Ciclo de Desenvolvimento de Sistemas Automatizados de Manufatura.** Mestrado (Dissertação em Engenharia de Produção e Sistemas) – Pontifícia Universidade Católica do Paraná. Curitiba, 2005.

CURY, J.E.R; **Teoria de Controle Supervisório de Sistemas a Eventos Discretos.** V Simpósio Brasileiro de Automação Inteligente. Canela-RS, 2001.

CURZEL, J.L.; LEAL, A.B.; **Implementação de controle supervisório em linguagem Ladder para uma célula flexível de manufatura didática**. In: XVI Congresso Brasileiro de Automática. Bahia, Brasil, p.2700-2705. 2006.

CURZEL, J.L.; SILVA, F.T. da; AMARAL, S. do e LEAL, A.B.; **Concepção de uma célula flexível de manufatura didática para o ensino de engenharia**. In: Anais do XXXIV Congresso Brasileiro de Ensino de Engenharia. Passo Fundo-RS, p.1916-1926. 2006.

CURZEL, J. L.; HOUNSELL, M. da S. ; LEAL, A. B. . **Uso da Realidade Virtual para Ensino de Automação da Manufatura.** In: ICECE 2007 International Conference on Engineering and Computer Education, 2007, Monguaguá / Santos. Anais da International Conference on Engineering and Computer Education, v. 1. p. 773-777. 2007a.

CURZEL, J. L.; LEAL, A. B.; HOUNSELL, M. da S.**Realidade Virtual como ferramenta de simulação aplicada no ensino de automação da manufatura.** In: 4º Simpósio Iberoamericano de Educación, Cibernética e Informática (SIECI 2007), Orlando. 2007b.

CURZEL, J.L.; LEAL, A.B.; **Implementação de controle supervisório modular local para uma célula flexível de manufatura didática**. In: VIII Conferência Internacional de Aplicações Industriais. MG, Brasil. 2008. (artigo aceito para publicação em 12/06/08).

DIAS, J. R. S.; **Um Laboratório para um Curso de Automação Industrial Utilizando a Teoria de Sistemas a Eventos Discretos.** Mestrado (Dissertação em Engenharia Elétrica) – Universidade Federal do Rio de Janeiro. Rio de Janeiro, 2005.

FABIAN, M.; HELLGREN, A. **PLC-based implementation of supervisory control for discrete event systems.** In: 37th IEEE Conference on Decision and Control, v. 3, p. 3305-3310. 1998

GARCIA, T. R.; CURY, J. E. R.; **Grail para Controle Supervisório de Sistemas a Eventos Discretos**. Centro Tecnológico, Universidade Federal de Santa Catarina, Departamento de Automação e Sistemas, Florianópolis, 2006.

HELLGREN, A.; FABIAN, M.; LENNARTSON, B.; **On the execution of sequential function charts.** Control Engineering Practice, v. 13 p. 1283-1293. 2005.

HOUNSELL, M. S.; PIMENTEL, A.; **On The Use of Virtual Reality to Teach Robotics**. In: ICECE - International Conference on Engineering and Computer Education, 2003, Santos-SP. International Conference on Engineering and Computer Education (IEEE Education Society), v. 01. p. 1-5. 2003.

IEC; **International Standard IEC 61131-3,** Programmable Logic Controllers – Part 3: Programming Languages. 2003.

MORAES, W. R. de; LEAL, A. B.; **Controle supervisório do transportador de entrada de um sistema flexível de manufatura.** Anais do VI Induscon. Joinville, 2006.

QUEIROZ, M.H. de; **Controle Supervisório Modular de Sistemas de Grande Porte.** Mestrado (Dissertação em Engenharia Elétrica) – Centro Tecnológico, Universidade Federal de Santa Catarina. Florianópolis, 2000.

_____; **Controle Supervisório Modular e Multitarefa de Sistemas Compostos.** Tese (Doutorado em Engenharia Elétrica) – Centro Tecnológico, Universidade Federal de Santa Catarina. Florianópolis, 2004.

QUEIROZ, M.H. de; CURY, J.E.R.; **Modular supervisory control of large scale discrete event systems.** In: Proceedings of the 5th International Workshop on Discrete Event Systems: Analysis and Control. Ghent, Belgium: Kluwer Academic Publishers, p. 103-110. 2000a.

_____; **Modular control of composed systems.** In: Proceedings of the American Control Conference. Chicago, USA. 2000b.

_____; **Controle supervisório modular de sistemas de manufatura.** Revista controle & automação, v. 13, n. 2, p. 115-125. 2002a.

_____; **Synthesis and implementation of local modular supervisory control for a manufacturing cell.** In: 6th International Workshop in Discrete Event Systems. Zaragoza, Spain: Kluwer Academic Publishers, p. 377-382. 2002b.

QUEIROZ, M.H. de, SANTOS, E.A.P. e CURY, J.E.R.; **Síntese modular do controle supervisório em diagrama escada para uma célula de manufatura**. Anais do V Simpósio Brasileiro de Automação Inteligente, Gramado RS. 2001.

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

REISER, C; CUNHA, A. E. C. da ; CURY, J. E. R. . **The Environment Grail for Supervisory Control of Discrete Event Systems.** In: 8th Workshop on Discrete Event Systems (WODES 2006), Proceedings of the 8th Workshop on Discrete Event Systems, v. 1. p. 390-391. Ann Arbor. 2006.

ROBOTEC, E.; **Scorbot-ER 4pc – User's Manual**, Catalog #100118 Rev A. 1982a.

ROBOTEC, E.; **Scorbase para Windows – Manual do Usuário**, Catálogo N^{o} 100280 Rev A. 1982b.

SANTOS, E.A.P.; VIEIRA, A.D.; BUSETTI, M.A.; **Controle de um sistema integrado de manufatura baseado na Teoria de Controle Supervisório.** In: Congresso Brasileiro de Automática, Bahia, Brasil, p. 1181-1186. 2006.

SILVEIRA, P. R.; SANTOS, W. E.; **Automação e Controle Discreto,** Editora Érica, São Paulo, 231 p. 1998

SIEMENS, S.; **SIMATIC S7-300 Programmable Controller Hardware and Installation – Manual,** Edition 10/2001 A5E00105492-01. Siemens AG, Nuemberg, 2001.

_____; **SIMATIC Ladder Logic (LAD) for S7-300 and S7-400 Programming – Reference Manual,** Edition 11/2002 A5E00171231-01. Siemens AG, Nuemberg, 2002a.

_____; **SIMATIC Configuring Hardware and Communication Connections STEP7 V5.2 – Manual,** Edition 12/2002 A5E00171229-01. Siemens AG, Nuemberg, 2002b.

_____; **SIMATIC Programming with STEP7 V5.2 – Manual,** Edition 12/2002 A5E00171230-01. Siemens AG, Nuemberg, 2002c.

TEIXEIRA, C.A.; LEAL, A.B.; SOUZA, A.H.; **Implementação de supervisores em microcontroladores: Uma abordagem baseada na teoria de controle de sistemas a eventos discretos.** In: XVI Congresso Brasileiro de Automática. Bahia, Brasil, p. 2772-2777. 2006.

VIEIRA, A. D.; **Método de Implementação do Controle de Sistemas a Eventos Discretos com Aplicação da Teoria de Controle Supervisório.** Tese (Doutorado em Engenharia Elétrica) – Centro Tecnológico, Universidade Federal de Santa Catarina. Florianópolis, 2007.

_____; **Modelagem e implementação de sistemas seqüenciais utilizando o método passo a passo.** Pontifícia Universidade Católica do Paraná. Curitiba, 2001.

WONHAM, W. M.; **Supervisory Control of Discrete-Event Systems.** Dept. of Electrical and Computer Engineering, University of Toronto, Canada. 2004.

APÊNDICE A – Tutorial do Gerador de Código para o CLP

O Gerador de Programa Automático para CLP (GPACLP) é uma ferramenta de geração automática de código para o CLP, com o intuito de auxiliar na tarefa de transcrição do autômato obtido para o supervisor no código de programação do CLP. Esta ferramenta foi desenvolvida com o auxílio de dois bolsistas do Programa de Educação Tutorial – PET Engenharia Elétrica.

O GPACLP é um programa (ou compilador) capaz de, a partir de um autômato obtido para o supervisor monolítico, gerar o código para o CLP referente à lógica de controle. O GPACLP tem como arquivos de entrada oriundos dos *softwares* Grail, TCT ou do próprio GPACLP, e a partir deles é possível gerar o programa em STL (*Statement List*) para ser implementado diretamente em CLPs das famílias S7-300/S7-400 da Siemens.

O programa suporta a conversão com origem em três tipos de arquivos (Grail, TCT e GPACLP):

GRAIL = Esse tipo arquivo está relacionado aos arquivos gerados pelo programa Grail.

TCT = Esse tipo arquivo está relacionado aos arquivos gerados pelo programa CTCT.

GPACLP = Corresponde aos arquivos salvos pelo próprio programa (depois de realizar alguma modificação nas variáveis, pode-se gerar um novo arquivo para o qual se adotou o nome de GPACLP).

O programa gerado para o CLP, é formado por três blocos de função: Supervisor, Saídas e Eventos, e um bloco de organização: OB1, conforme descrito na Seção 5.1. Além disso, pode-se gerar uma tabela de símbolos que associa automaticamente os nomes dos eventos escolhidos com as entradas/saídas e memórias do CLP.

>> Ao abrir o programa (executável) surgirá à tela conforme a Figura A.1:

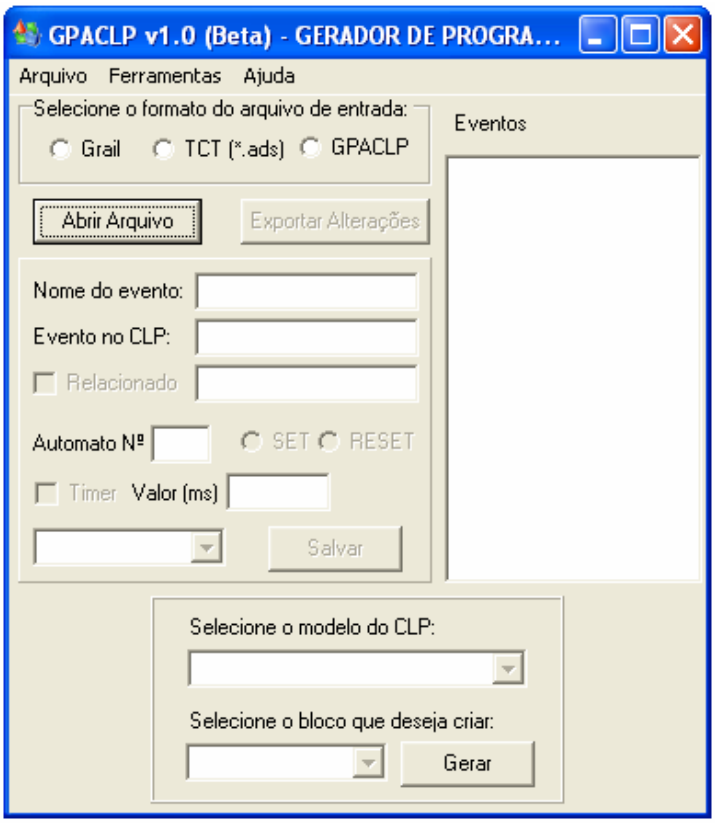


Figura A.1

>> Primeiramente deve-se abrir o arquivo do qual se deseja realizar a conversão para o formato dos CLPs Siemens.

>> Selecione o tipo de arquivo que deseja abrir e clique em Abrir Arquivo (Figura A.2).

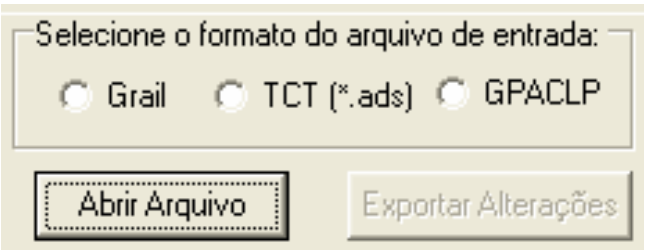


Figura A.2

>> Em seguida, o arquivo será carregado pelo programa e na lista de eventos à direita aparecerão todos os eventos (sem repetição) que estiverem presentes no arquivo do supervisor que foi aberto (Figura A.3).

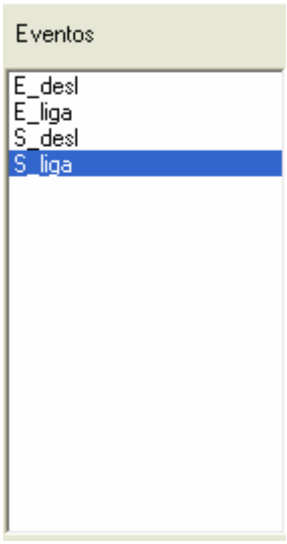


Figura A.3

>> Ao clicar em qualquer um dos eventos contidos na lista, os parâmetros desse evento poderão ser alterados (se necessário) através dos campos à esquerda (Figura A.4):

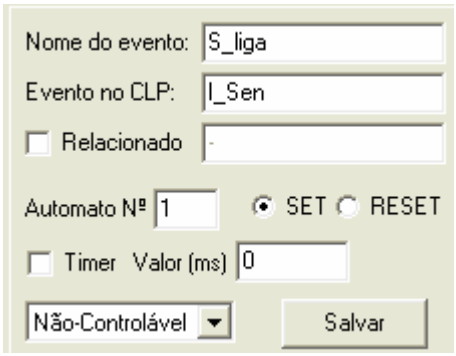


Figura A.4

Nome do Evento = Coloque (se desejar alterar) um nome qualquer para o evento no supervisor (arquivos TCT são, por padrão, números associados ao evento e não nomes, e podem permanecer assim se for conveniente). Ex: S_Liga.

Evento no CLP = Coloque um nome para um evento no CLP, este corresponderá a um "símbolo" da saída/entrada física do CLP. Ex: LIGA_MOTOR_ESTEIRA (é comum que o nome/símbolo usado como evento no CLP tenha relação com a característica física que o mesmo executa).

Relacionado = Caso tenha sido feito o modelamento de dois equipamentos por um único autômato, ou de dois eventos relacionados (Ex: Esteira+Sensor), deve-se atribuir um símbolo no CLP para o evento relacionado.

Autômato Nº = Indicar o número do autômato ao qual pertence o evento em questão. Por exemplo, os eventos S_liga e S_desl pertencem ao autômato 1 (modelo para o sensor)

SET / RESET = Deve-se atribuir o SET caso seja um evento destinado a ligar (habilitar), e um RESET se o evento for destinado a desligar (desabilitar).

Timer / Valor = Alguns eventos necessitam de um tempo de atraso para que o processamento do evento pelo CLP possa ser executado. Nestes casos, pode ser feita a adição de um **Timer** e colocado seu **Valor** em milissegundos.

Controlável / Não-Controlável = Indique se o evento selecionado é Controlável ou Não-Controlável.

Salvar = Clique no botão depois de alterar qualquer um dos parâmetros acima, caso contrário, a(s) alteração(ões) não será(ão) salva(s).

Depois de realizadas todas as modificações necessárias, pode-se passar para o processo de geração dos blocos de programa.

>> Selecione primeiramente o modelo do CLP e em seguida o bloco que deseja criar (conforme a Figura A.5). Então, basta clicar no botão "Gerar" e escolher o local onde será salvo o arquivo.

Na versão atual do GPACLP são aceitos somente os CLPs da Siemens S7_300 e 400.

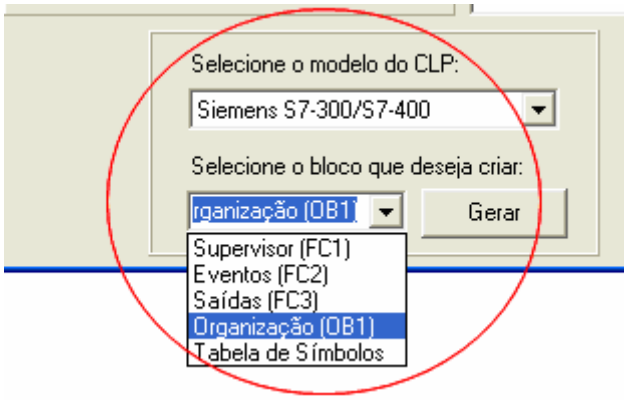


Figura A.5

Ao selecionar a criação da tabela de símbolos, será gerada uma lista com os símbolos definidos anteriormente (EVENTO NO CLP) relacionando-os aleatoriamente com as saídas físicas do CLP a partir do primeiro endereço do CLP. Uma vez que esta lista está

relacionada diretamente as ligações elétricas do CLP, ela deve ser alterada posteriormente conforme a Tabela 4-1 - Eventos dos dispositivos da célula., *que define como estão fisicamente distribuídas as entradas e saídas.*

*Ao selecionar os blocos Organização, Supervisor, Eventos ou Saídas, será gerado um arquivo no formato *.awl, que poderá ser importado diretamente no programa do CLP da Siemens (SIMATIC STEP7) conforme será mostrado posteriormente.*

IMPLEMENTAÇÃO NO CLP SIEMENS (SIMATIC Step7 V5.1)

Após feita a instalação do SIMATIC STEP7, clique no atalho do SIMATIC MANAGER. Caso não tenha aparecido nada na tela, localize no canto superior esquerdo da tela e clique em File (Arquivo) e em seguida em 'New Project' Wizard (conforme a Figura A.6).

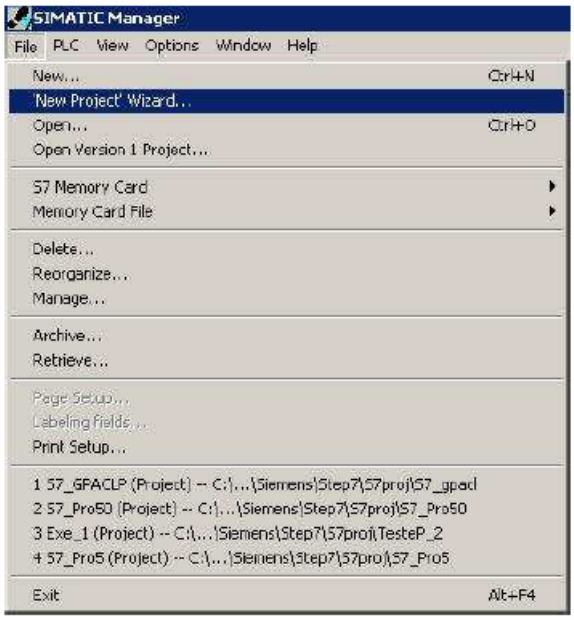


Figura A.6

Uma janela então se abrirá (Figura A.7), clique em "Next (Avançar)".

Figura A.7

Selecione na próxima janela (Figura A.8) a CPU que seu CLP SIEMENS possui e clique em "Next (Avançar)".

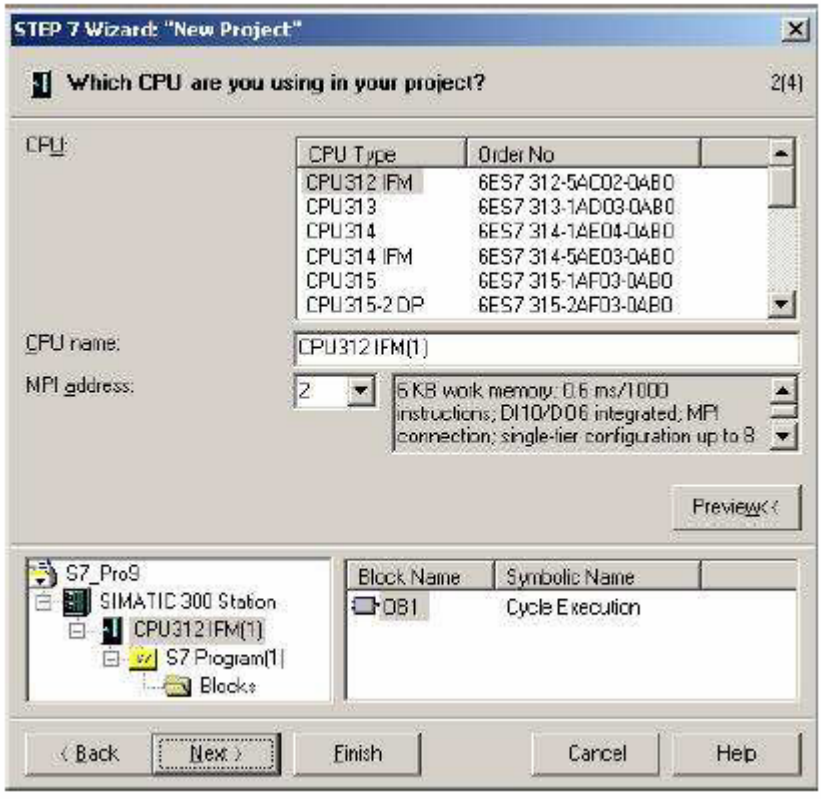


Figura A.8

Desmarque o bloco OB1 (assim como todos os outros) e clique em Next (Figura A.9).

Figura A.9

Dê um nome para seu projeto e clique em Finish (Figura A.10).

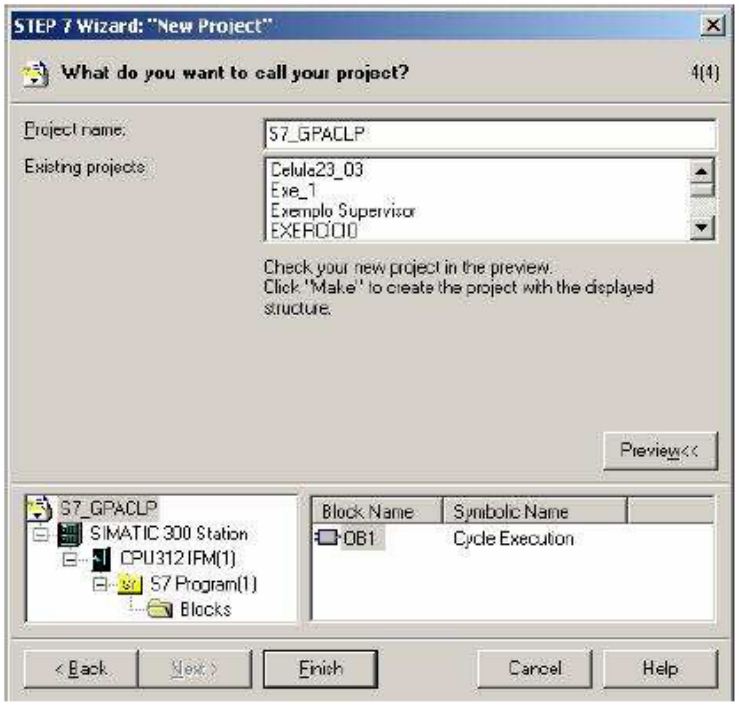


Figura A.10

O primeiro passo após a criação do projeto, é fazer a importação da tabela de símbolos gerada pelo GPACLP.

Abra o Symbol Editor (Editor de Símbolos) e a janela mostrada na Figura A.11 irá se abrir.

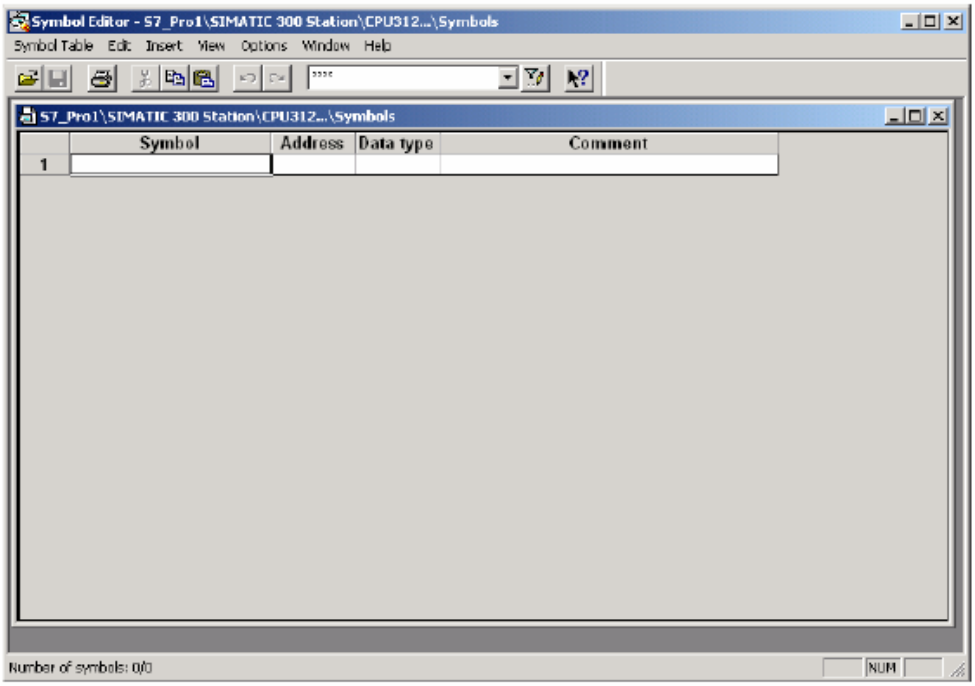


Figura A.11

Em seguida vá ao menu superior e clique em Symbol Table e Import (como mostra a Figura A.12).

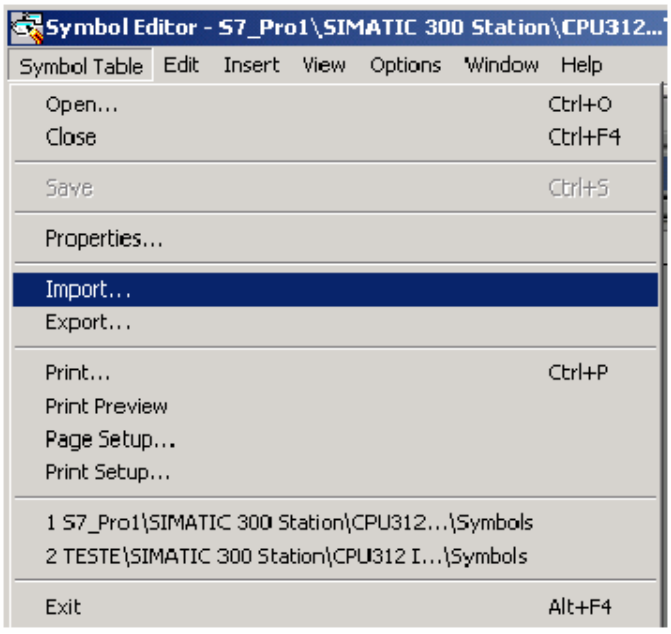


Figura A.12

Ao abrir a caixa de diálogo, selecione a extensão do arquivo para *.SEQ e procure a pasta onde foi salvo o arquivo gerado pelo GPACLP. Em seguida clique em abrir *(Figura A.13).*

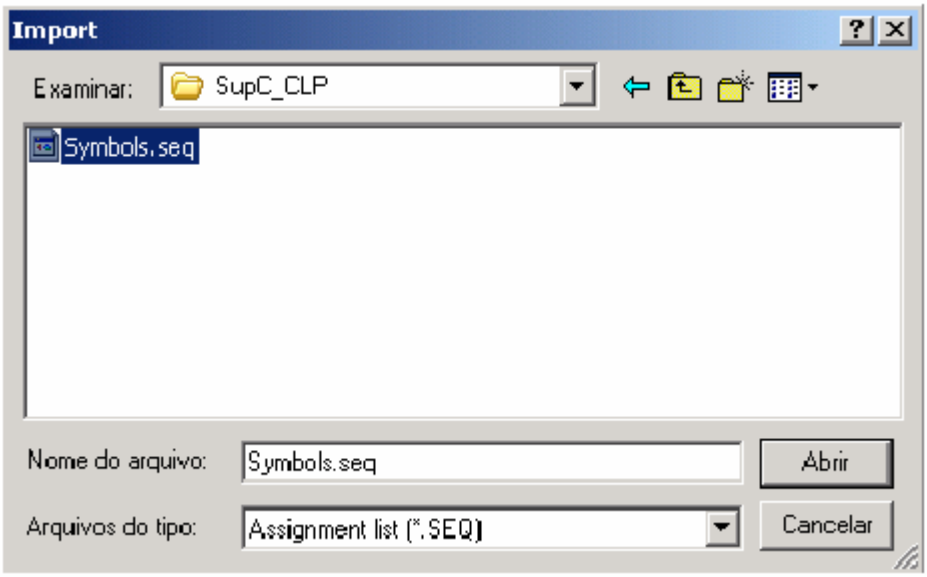


Figura A.13

Caso apareçam as caixas de diálogos mostradas abaixo *(Figura A.14)* clique em SIM para sobrepor a tabela de símbolos já existente (uma vez que esta está vazia) e clique em NÃO para abrir o arquivo de protocolo.

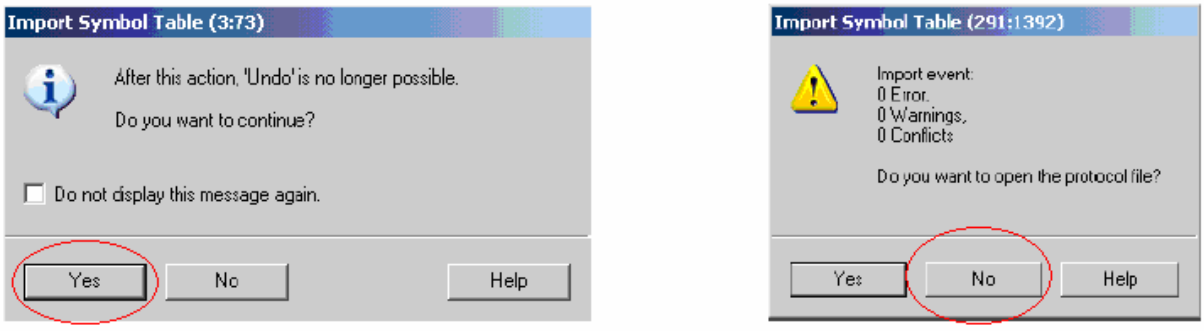


Figura A.14

Depois de feita a importação da tabela de símbolos com sucesso, pode-se fechar a janela do Symbol Editor.

Voltando ao SIMATIC MANAGER, o próximo passo consiste em fazer a compilação de todos os blocos.

Clique então em Sources no menu à esquerda, em seguida clique com o botão direito do *mouse* na parte da direita da janela depois Insert New Object e External Source (conforme a Figura A.15).

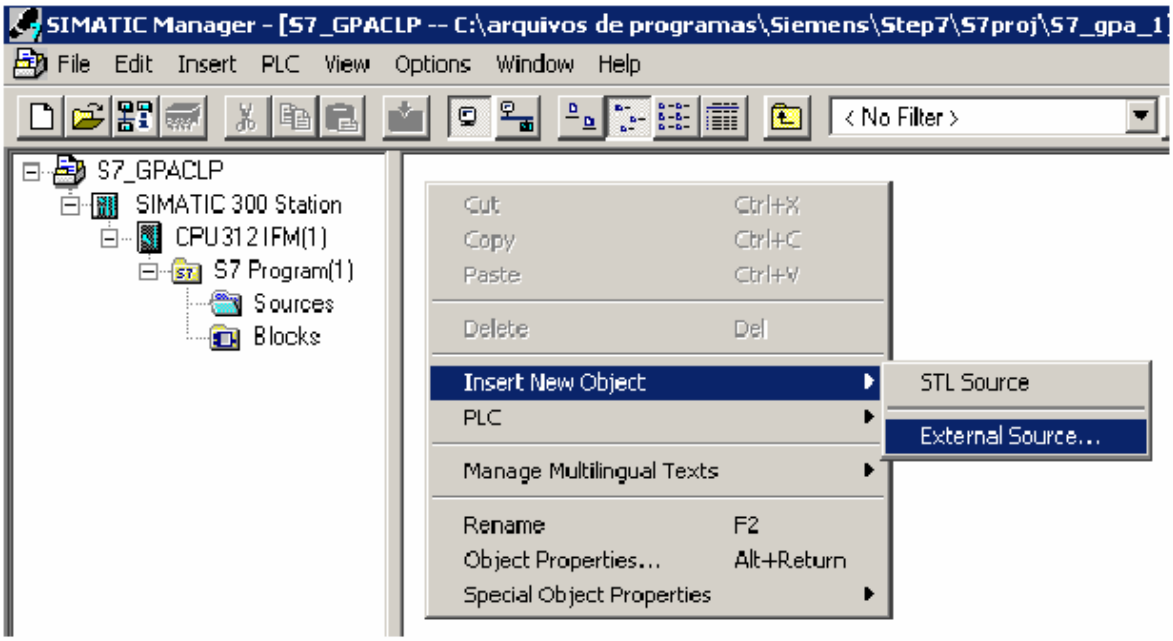


Figura A.15

Ao abrir a caixa de diálogo, procure a pasta onde foi salvo o arquivo gerado pelo GPACLP. Em seguida clique em abrir *(Figura A.16).*

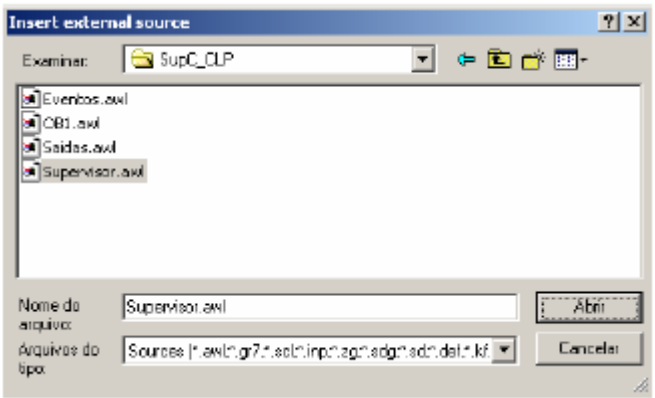


Figura A.16

Deve-se então compilar o arquivo. Para tanto, procure o símbolo indicado na Figura A.17 e clique nele.

Figura A.17

Caso o arquivo tenha sido compilado com sucesso, na barra se status na parte inferior da janela surgirá a mensagem mostrada na Figura A.18.

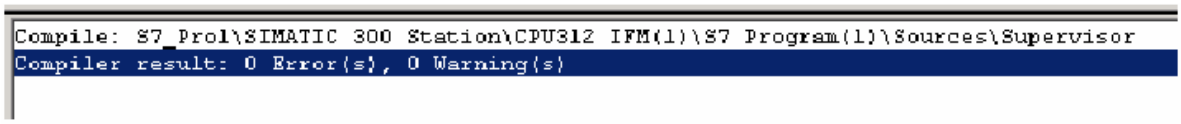


Figura A.18

Caso tenha sido encontrado algum erro, este será mostrado nessa mesma barra para posteriores correções.

O arquivo compilado é mostrado na Figura A.19:

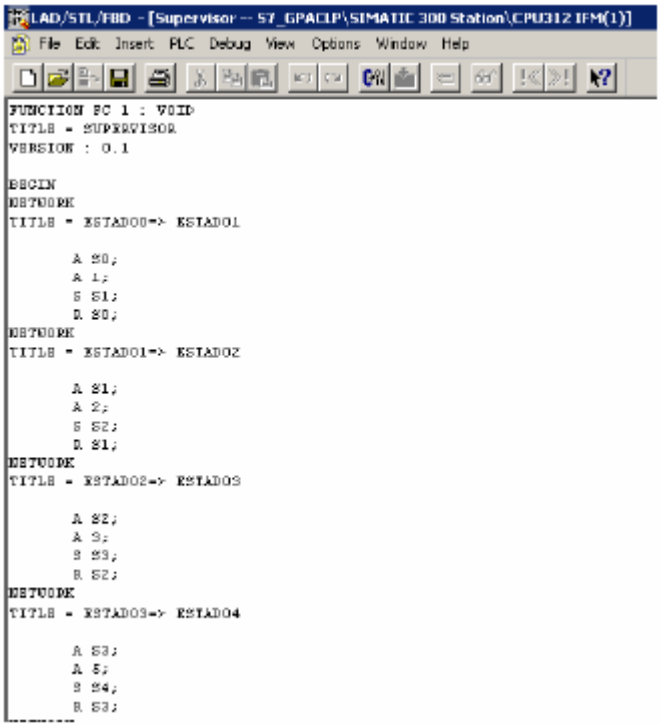


Figura A.19

Repita o processo de compilação para os demais blocos na seguinte ordem:

1º) Saídas.awl

2º) Eventos.awl

3º) OB1.awl

Após todos os blocos compilados com sucesso, o programa está pronto para ser implementado no CLP. Isso pode ser feito basicamente selecionando os blocos na aba Blocks do menu à esquerda e em seguida clicando no botão Download (conforme a Figura A.20).

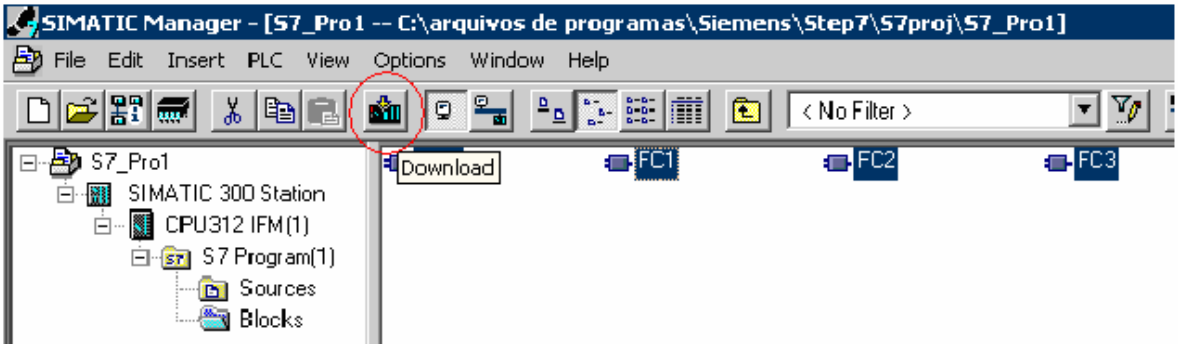


Figura A.20

Maiores detalhes sobre o uso do software do CLP e características técnicas do hardware utilizado encontram-se no manual (Siemens, 2001, 2002b).