**UNIVERSIDADE FEDERAL DO RIO DE JANEIRO**

**ESCOLA POLITÉCNICA**

**DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

PROJETO DE GRADUAÇÃO

# PROJETO DE UM SISTEMA DE AUTOMAÇÃO DE UMA CÉLULA DE MANUFATURA UTILIZANDO CLP SIEMENS S7-1200

**DANIEL DOS SANTOS BOTELHO**

RIO DE JANEIRO, RJ - BRASIL

JUNHO DE 2011

**PROJETO DE UM SISTEMA DE AUTOMAÇÃO DE UMA CÉLULA DE MANUFATURA UTILIZANDO CLP SIEMENS S7-1200**

Daniel dos Santos Botelho

PROJETO SUBMETIDO AO CORPO DOCENTE DO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRICISTA.

Aprovado por:

_____

Prof. Marcos Vicente de Brito Moreira, D.Sc. (Orientador)

_____

Prof. Sergio Sami Hazan, Ph.D.

_____

Luiz Otávio Mazoni Andrade, B.Sc.

RIO DE JANEIRO, RJ - BRASIL

JUNHO DE 2011

**AGRADECIMENTOS**

A Deus, pois a Ele e para Ele são e serão toda a honra e toda a glória, referentes à todas as conquistas da minha vida.

À minha mãe, Janete Russowsky, por me dar amor incondicional e orientação em todos os momentos da minha vida. Você é a maior guerreira que eu já conheci. Com você eu aprendi a lutar pelos meus sonhos e essa vitória eu dedico a você.

Aos meus irmãos: Thiago Botelho, Marcos Vinicius Silva, e Matheus Botelho. Vocês são os meus melhores amigos e me motivam a ser uma pessoa melhor.

À minha família: Miguel Russowsky, Janete Russowsky, Thiago Botelho, e Marcos Vinicius Silva. Vocês são a base das minhas conquistas.

À minha avó, Wilma Pereira dos Santos. Pois o seu amor e as suas orações sempre me ajudaram a suportar os momentos mais difíceis durante esta caminhada. Obrigado pela sua presença.

À minha avó, Jandira Botelho. Pois eu sei que as suas orações a Deus me ajudarão por toda a minha vida. *(In Memorian)*

À família do meu pai: Paulo Elias Lima Botelho, Vivian Botelho, e Matheus Botelho. Obrigado por também fazerem parte desta conquista.

Aos meus amigos da UFRJ: Ricardo Vianna, Vinícius Carvalho, Leonardo Baptista, Wallace Tayson dos Santos, Cesar Cedrola, Guilherme Vasconcellos, Rafael Peixoto, Raphael Baptista, Lucas Velloso, Felipe Ribeiro, Peterson Nogueira, Beatriz Levy, Carolina Peleteiro, Frederico Marques, Monique Alves, e Tiago Moraes. Sem vocês essa trajetória teria sido muito menos prazerosa.

Ao meu professor, orientador, e amigo: Marcos Vicente de Brito Moreira. Pela excelente orientação ao longo da universidade e por ter acreditado no meu potencial.

A todos os meus amigos e aqueles que contribuíram direta ou indiretamente para que este sonho fosse realizado.

*“(...) em todas as coisas somos mais que*

*vencedores por meio daquele que nos amou (...)”*

*Romanos 8:37*

## CONTEÚDO

## LISTA DE FIGURAS

## LISTA DE TABELAS

# 1 INTRODUÇÃO

O avanço da indústria mundial e a grande influência dos processos industriais na economia global impulsionaram altos investimentos tecnológicos nas áreas de controle e automação. Para que a indústria cresça é necessário otimizá-la, e para isso é necessário ter o controle total das informações sobre o processo que ocorre na planta industrial. Muitas dessas informações são bastante específicas e de difícil acesso físico, o que demanda a aplicação de uma tecnologia própria.

Para atender ao seu próprio crescimento a indústria inseriu em seu processo os controladores lógico-programáveis (CLPs), que permitiram a automatização das linhas de produção aumentando a eficiência e o acesso a determinados parâmetros com alta exatidão. Com a modernização da indústria, os CLPs passaram a definir todos os parâmetros das linhas de produção, proporcionando ao operador total controle da lógica de processo.

Desta forma, tornou-se também necessário criar sistemas computacionais avançados capazes de programar estes controladores e de viabilizar a sua comunicação com os equipamentos presentes na indústria.

Neste trabalho é apresentada a automação de uma célula de manufatura composta por duas máquinas, dois braços robóticos, e um buffer, que funciona como um espaço de trabalho de capacidade limitada.

A automação dessa célula de manufatura é feita através do CLP Siemens S7-1200, que se comunica com uma IHM, isto é, uma interface homem-máquina, modelo KTP600, também Siemens, via Ethernet com protocolo profibus. A automação do sistema é feita desde a modelagem da lógica do sistema utilizando-se redes de Petri, até a sua simulação e implementação.

A implementação do sistema de automação foi feita utilizando-se os equipamentos presentes na bancada do Laboratório de Controle e Automação do Departamento de Engenharia Elétrica da Universidade Federal do Rio de Janeiro. A programação do controlador e da IHM, assim como a configuração da rede de profibus, foi feita através do software de configuração e programação *Totally Integrated Automation Portal V10* da Siemens. A utilização do software de automação Siemens é apresentada em formato de tutorial ao longo do trabalho.

Este trabalho está estruturado da seguinte maneira: no capítulo 2 são apresentados os conceitos básicos de redes de Petri e redes de Petri interpretadas para controle. No capítulo 3 são apresentadas as instruções básicas para a programação do CLP Siemens S7-1200 e o método de conversão de redes de Petri interpretadas para controle em diagrama Ladder. No capítulo 4 a configuração da rede profibus e a programação do CLP e da IHM são apresentadas em forma de tutorial. No capítulo 5 são apresentadas a modelagem e integração do sistema de automação da célula de manufatura e, finalmente, no capítulo 6 são apresentadas as conclusões e as sugestões para trabalhos futuros.

## 2 FUNDAMENTOS TEÓRICOS DE REDES DE PETRI

No capítulo são apresentadas as principais características e funcionalidades das Redes de Petri. Uma rede de Petri é um grafo orientado bipartido [1], ou seja, possuindo dois tipos de vértices que nunca são ligados entre si. Em outras palavras, uma rede de Petri é um dispositivo que manipula eventos respeitando regras bem definidas [2].

### 2.1. Grafo de uma rede de Petri

O processo de definição de uma rede de Petri pode ser dividido em duas partes. A primeira, na qual definimos o grafo da Rede de Petri, também chamado de estrutura da rede de Petri; e a segunda, na qual adicionamos ao grafo um estado inicial, um conjunto de estados marcados e uma função de transição, o que resulta no modelo completo da rede de Petri.

***Definição 1:*** *Um grafo de uma rede de Petri é uma quíntupla:*

$$(P, T, Pre, Pos, \omega),$$

*em que P, definido por* $P=\{p_1, p_2,...,p_n\}$ *onde* $n \in N^*$*, é um conjunto finito, não vazio, de objetos chamados de lugares; T, definido por* $T=\{t_1, t_2,...,t_m\}$ *onde* $m \in N^*$*, um conjunto finito, não vazio, de objetos chamados de transições; Pre, definido por* $Pre \subseteq P \times T$*, é um conjunto de relações de entrada que ligam os lugares (P) às transições (T); Pos, definido por* $Pos \subseteq T \times P$*, é um conjunto de relações de saída que ligam as transições (T) aos lugares (P); e ω, definido por* $\omega:(P \times T) \cup (T \times P) \rightarrow N$*, é uma função que atribui uma peso maior do que zero a cada arco da rede de Petri, e zero para* $(p_i, t_j) \notin Pre$ *e* $(t_j, p_i) \notin Pos$.

Os arcos, que ligam o lugar $p_i$ à transição $t_j$ representam os elementos do conjunto *Pre*. Assim, se existe um arco saindo do lugar $p_i$ em direção à transição $t_j$, temos que

$$(p_i, t_j) \in Pre.$$

Os arcos, que ligam a transição $t_j$ ao lugar $p_i$ representam os elementos do conjunto *Pos*. Assim, se existe um arco saindo da transição $t_j$ para o lugar $p_i$, temos que

$$(t_j, p_i) \in Pos.$$

Se todos os arcos possuírem peso igual a 1 (um) a rede de Petri é chamada ordinária. Analogamente, se existir ao menos um arco cujo peso for maior que 1 (um), a rede é chamada de rede de Petri generalizada.

Por convenção, uma rede de Petri possui uma representação gráfica, de forma que os lugares são representados por círculos, as transições por barras, e as relações por arcos orientados com peso dado pela função ω.

### 2.2. Marcação das redes de Petri

***Definição 2:*** *Uma rede de Petri é uma sêxtupla N=(P, T, Pre, Pos, ω, $\underline{\underline{x}}$), em que (P, T, Pre, Pos, ω) é um grafo de uma rede de Petri, e $\underline{\underline{x}} = [\, x(p_1)\ x(p_2)\ x(p_3) \ldots\ x(p_n)]^T$, onde n é o número de lugares existentes na rede de Petri, é um vetor coluna que indica a marcação do conjunto de lugares de N. Em relação ao vetor coluna $\underline{\underline{x}}$, cada elemento $x(p_i)$ representa o número de fichas presentes no lugar $p_i$.*

A marcação de uma Rede de Petri é uma função $x: P \rightarrow N$. Representa-se graficamente esta função $x$ inserindo elementos chamados de fichas dentro dos lugares da rede de Petri. Assim, um lugar pode ser marcado (quando este possui uma ou mais fichas) ou não marcado (quando este está vazio).

Na rede de Petri da figura 1 podemos visualizar um exemplo de marcação, de forma que nesta rede os lugares 1 e 2 são marcados por duas fichas e uma ficha, respectivamente, enquanto o lugar 3 se encontra vazio.

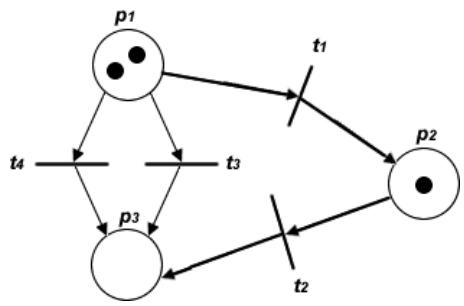



**Figura 1 – Rede de Petri: os lugares $p_1$ e $p_2$ possuem fichas, e o lugar p3 encontra-se vazio.**

A marcação inicial de uma rede de Petri pode ser indicada por um vetor coluna $\underline{x}_0$ . Para a rede de Petri da figura 1, considerando-se que o estado atual da mesma represente seu estado inicial, seu vetor $\underline{x}_0$ pode ser representado por:

$$\underline{x}_0 = [2\ 1\ 0]^T.$$

### 2.3. Habilitação de uma transição

Uma transição $t_j$ é dita habilitada se e somente se todos os lugares de entrada (lugares $p_i$ tal que $(p_i, t_j) \in Pre$) possuírem no mínimo um número de fichas maior ou igual ao peso dos arcos que os ligam a esta transição.

Na rede de Petri presente na figura 2, podemos visualizar que de acordo com a sua marcação, encontram-se habilitadas as transições $t_1$ e $t_2$. Na rede de Petri da figura 3, por sua vez, podemos visualizar que de acordo com a sua marcação, a transição $t_1$ encontra-se habilitada enquanto $t_2$ encontra-se não-habilitada.

### 2.4. Disparo de uma transição

Uma transição $t_j$ pode ser disparada se e somente se esta estiver habilitada. As consequências do disparo de uma transição (sempre considerado instantâneo) podem ser destacadas da seguinte forma:

- Deve ser retirado dos lugares de entrada de $t_j$, um número de fichas igual ao peso respectivo dos arcos que ligam esses à transição $t_j$.
- Deve ser depositado um número de fichas igual ao peso respectivo dos arcos que ligam a transição $t_j$ aos seus lugares de saída.

Como pode ser observado através das consequências listadas acima, quando uma transição *tj* é disparada pode ocorrer a mudança da marcação da rede de Petri. Considerando-se que a rede de Petri representada na figura 2 possui marcação inicial $\underline{x}_0 = [3\ 1\ 1\ 0\ 0]^{\mathrm{T}}$, então as transições *t1* e *t2* estão inicialmente habilitadas e o disparo de uma dessas transições atribui à rede de Petri uma nova

marcação $\underline{x}$'. Nas figuras 4 e 5, as consequências dos disparos das transições *t1* e *t2*, respectivamente, podem ser visualizadas, assim como a mudança da marcação da rede de Petri.

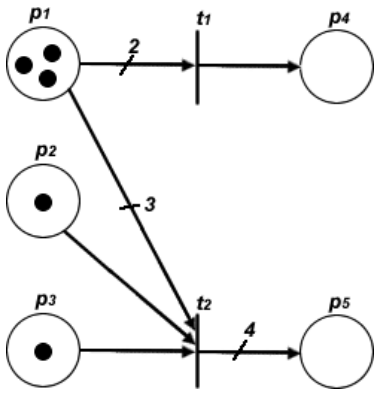



**Figura 2 – Rede de Petri: as transições *t1* e *t2* estão habilitadas.**

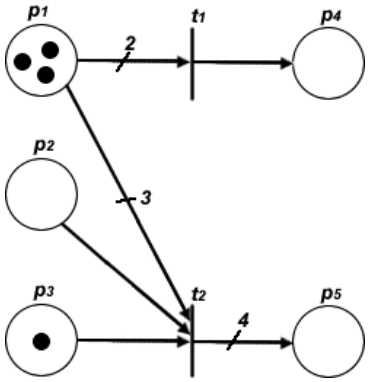



**Figura 3 - Rede de Petri: a transição *t1* encontra-se habilitada e a transição *t2* não-habilitada.**

### 2.5. Equação de Estado e Matriz de Incidência

O estado atual de uma rede de Petri é representado pela marcação da mesma. Assim, uma mudança na marcação de uma rede de Petri representa uma mudança em seu estado. Na seção 2.4 foi apresentado um exemplo de rede de Petri cuja marcação foi modificada mediante o disparo de uma determinada transição.

A matriz de incidência *A* de uma rede de Petri é composta de *n* linhas e *m* colunas, que representam, respectivamente, o número de lugares e o número de transições de uma rede de Petri. Assim, podemos definir cada elemento desta matriz *A*, da seguinte forma:

$$a_{i,j} = w(t_j, p_i) - w(p_i, t_j),$$

em que $w(t_j, p_i)$ representa o peso do arco que liga a *j*-ésima transição ao *i*-ésimo lugar, e $w(p_i, t_j)$ representa o peso do arco que liga o *i*-ésimo lugar à *j*-ésima transição.

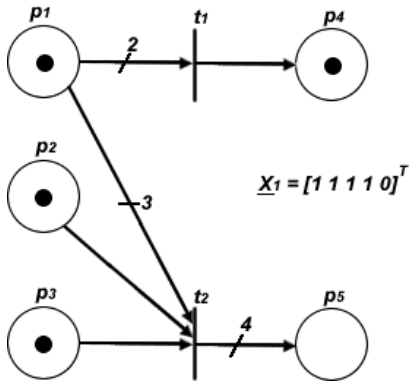



**Figura 4 - Rede de Petri da figura 2 após o disparo da transição *t1*.**

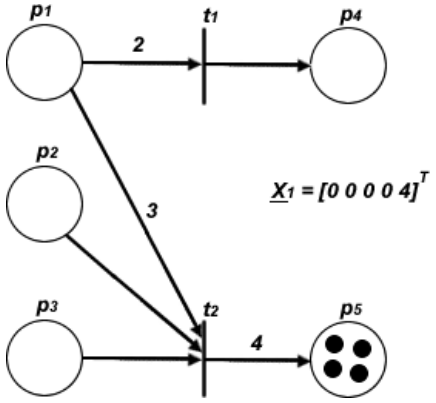



**Figura 5 - Rede de Petri da figura 2 após o disparo da transição *t2*.**

A partir da figura 1 temos que a matriz de incidência da rede de Petri é representada da seguinte forma:

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

A matriz de incidência de uma rede de Petri também pode ser apresentada como o resultado de uma operação matemática entre a matriz de incidência de entrada, $A_{in}$, e a matriz de incidência de saída, $A_{out}$. Assim, podemos dizer que a matriz de incidência *A*, *n x m,* é dada por:

$$A = A_{out} - A_{in},$$

em que $A_{in} = [a_{ij}^{in}]$, com $a_{ij}^{in} = w(p_i, t_j)$, é a matriz de incidência de entrada, e $A_{out} = [a_{ij}^{out}]$, com $a_{ij}^{out} = w(t_j, p_i)$, é a matriz de incidência de saída.

A equação de estado define, de forma algébrica, a marcação que uma rede de Petri vai possuir após o disparo de uma transição. Essa definição deve ser feita a partir do estado atual da rede de Petri $\underline{x}$ e da sua matriz de incidência *A*. A equação de estado é representada na equação 2.1.

$$\underline{x}' = \underline{x} + A\,\underline{u}, \tag{2.1}$$

sendo $\underline{x}'$ o estado alcançado a partir de $\underline{x}$ após o disparo de uma transição $t_j$ e $\underline{u} = [0 \ldots 0\ 1\ 0 \ldots 0]^T$, um vetor coluna de dimensão *m*, sendo *m* o número de transições da rede de Petri, e o único 1 aparecendo na *j*-ésima posição, para indicar que a *j*-ésima transição está sendo disparada.

Considere, agora, o estado inicial da rede de Petri da figura 1, $\underline{x}_0 = [2\ 1\ 0]^T$. Caso se queira saber o estado seguinte dessa rede de Petri mediante o disparo da transição $t_1$, devemos compor o vetor $\underline{u}$ de modo a representar na equação de estado o disparo da transição $t_1$:

$$\underline{u} = [1\ 0\ 0\ 0]^T.$$

Substituindo-se os valores atribuídos às variáveis $\underline{x}$, $\underline{u}$ e *A*, na equação de estado, temos que:

$$\underline{x}' = [1\ 2\ 0]^T.$$

O estado alcançado, representado pelo vetor coluna $\underline{x}'$, indica que a partir de um estado definido pela marcação inicial $\underline{x}_0 = [2\ 1\ 0]^T$, a rede de Petri representada pela matriz de incidência *A*, mediante o disparo da transição $t_1$, apresentará a marcação $\underline{x}' = [1\ 2\ 0]^T$.

## 2.6. Redes de Petri Interpretadas para Controle (RPIC)

Até agora, neste trabalho, foram apresentadas as redes de Petri ordinárias. Essas redes são bastante empregadas na modelagem de sistemas a eventos discretos, porém possuem limitações quando se deseja modelar os controladores desses mesmos sistemas.

As redes de Petri Interpretadas para Controle (RPIC) possuem uma estrutura que permite a modelagem do controlador de sistemas industriais. Assim, a RPIC considera algumas condições impostas por eventos externos à rede, além das condições normais de habilitação de uma dada transição que foram apresentadas na subseção 2.3.

***Definição 3:*** *Uma rede de Petri Interpretada para controle é uma undécupla:*

$$N = (P, T, Pre, Pos, w, x, E, C, D, O, Q)$$

*em que (P, T, Pre, Pos, ω, x) é uma rede de Petri segura, ou seja, todos os lugares devem ter no máximo uma única ficha para todos os estados alcançáveis da rede de Petri; T= $T_0$ ∪̇ $T_D$ , em que $T_0$ é o conjunto de transições não temporizadas e $T_D$ é o conjunto de transições temporizadas; E= {$e_j$ : $t_j$ ∈$T_0$}, e C={$c_j$ : $t_j$ ∈ $T_0$} são os conjuntos de eventos e de condições, respectivamente, associados às transições não temporizadas de $T_0$; D = {$d_j$ ∈ $\boldsymbol{R}$: $t_j$ ∈$T_D$} é o conjunto de atrasos de disparo associados às transições temporizadas de $T_D$; e O e Q são, respectivamente, os conjuntos das operações e ações booleanas ou impulsionais associados aos lugares.*

Note que na *Definição 3* supomos que toda transição não-temporizada possui uma condição associada à mesma. Se a condição $c_j$ associada a $t_j$ não é explicitamente especificada, $c_j$ = 1, isto é, a condição lógica $c_j$ é verdadeira. Além disso, se o evento $e_j$ não é explicitamente especificado, $e_j$ = $e$, o evento que sempre ocorre. Assim, de acordo com a *Definição 3*, todas as transições de $T_0$ possuem expressões booleanas associadas a estas. Essas expressões podem ser escritas como $R_j = c_j \cdot e_j$ para $j=\{1,...,m\}$, em que $R_j$ é chamada de receptividade da transição $t_j$.

É importante ressaltar que algumas RPIC na literatura utilizam três tipos diferentes de arcos: arcos simples, arcos habilitadores, e arcos inibidores. Arcos simples são definidos de lugares para

transições e de transições para lugares, enquanto os arcos habilitadores e os arcos inibidores são definidos somente de lugares para transições.

Os arcos habilitadores possuem as mesmas características de um laço entre o lugar e a transição, isto é, ele habilita a transição quando o número de fichas no lugar de entrada é maior do que o peso dos arcos habilitadores, porém o lugar de entrada não perde fichas mediante o disparo da transição.

Os arcos inibidores, por outro lado, possuem uma aplicação diferente dos arcos habilitadores. Quando o número de fichas no lugar de entrada é maior que o peso do arco inibidor, a transição não é habilitada. Embora arcos inibidores não sejam definidos pelo controlador da rede de Petri descrita neste trabalho, o procedimento de conversão de uma rede de Petri para a linguagem Ladder, que será descrito no capítulo 3, pode ser utilizado nestes casos apenas com uma pequena modificação na matriz de incidência de entrada da Rede de Petri.

As redes de Petri Interpretadas para Controle possuem duas partes distintas chamadas de parte de processamento de dados e parte de controle. Na figura 6 podemos visualizar o fluxo de dados em uma rede de Petri Interpretada para Controle, bem como as suas partes discriminadas.

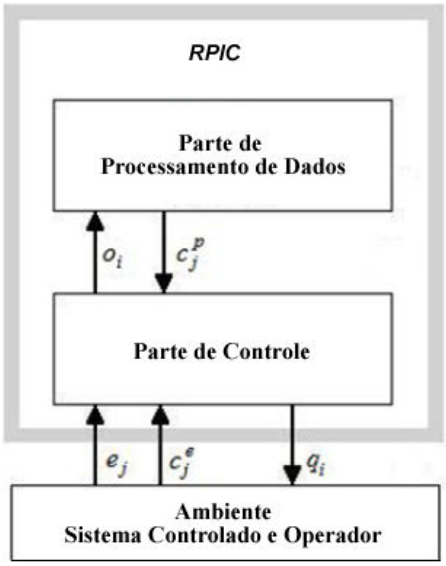



**Figura 6 - Ilustração do Fluxo de Dados em uma Rede de Petri Interpretada para Controle.**

Na parte de processamento de dados são feitas operações através das ordens de operação $o_i$ recebidas do controlador a eventos discretos, em seguida esta envia informações ao controlador relacionadas às condições internas associadas aos dados processados $c_j^p$. A RPIC recebe as condições $c_j^e$ do ambiente, e envia ações $q_i$ para o mesmo.

Na figura 7 uma representação de um lugar e uma transição genérica de uma RPIC é apresentada. A transição $t_j$ irá disparar caso esteja habilitada e a condição $c_j$ seja verdadeira, quando o evento $e_j$ ocorrer.

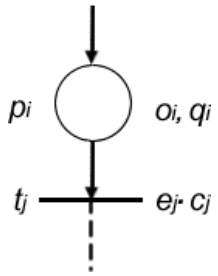



**Figura 7 - Rede de Petri Interpretada para Controle apresentada de forma genérica.**

Ainda na figura 7, quando uma ficha é depositada no lugar $p_i$, uma ação booleana $O_i$ é iniciada e uma operação $Q_i$ é realizada. Vale a pena ressaltar que a operação booleana é continuada caso uma ficha permaneça no lugar $p_i$.

## 2.7. Conclusão

Neste capítulo vimos as principais definições teóricas sobre redes de Petri. Essas definições nos orientam a trabalhar com esta forma de modelagem e conhecer as suas principais características.

Através das equações de estado vimos que podemos determinar o estado futuro de um sistema modelado por uma rede de Petri mediante o disparo de uma determinada transição. Este estado é definido pelo vetor de estados, que caracteriza o estado futuro do sistema, a partir do vetor de estados atual e da matriz de incidência da rede de Petri.

Vimos também uma nova definição de redes de Petri, chamadas de redes de Petri Interpretadas para Controle. Esse tipo de rede de Petri é apropriado para descrever o funcionamento de sistemas de controle que recebem sinais associados a eventos externos, que é o caso de sistemas industriais controlados por CLPs [3].

# 3 MÉTODO DE CONVERSÃO DE RPIC PARA LADDER

Os controladores lógico-programáveis (CLPs) são equipamentos utilizados na indústria e que trocam dados com os demais equipamentos presentes na planta industrial, utilizando sinais elétricos específicos para redes de automação. A frequência com que esses controladores acessam os equipamentos de campo e utilizam ou editam as suas informações são especificadas através da lógica de controle.

Neste capítulo será apresentada uma linguagem de programação de CLPs chamada de diagrama Ladder, bem como seus componentes principais e, em seguida, será apresentado um método de conversão de redes de Petri Interpretadas para Controle para diagramas Ladder. O método de conversão aqui apresentado é baseado no método de Moreira, Botelho & Basilio (2009) [4].

## 3.1. Diagrama Ladder

A norma técnica internacional IEC 61131-3 define que os fabricantes de controladores industriais devem disponibilizar, no mínimo, uma das 5 linguagens permitidas na norma para que os usuários possam programá-los. São elas: Diagrama Ladder (LD), Blocos de Funções (FB), Texto Estruturado (ST), Sequencial Function Chart (SFC), e Lista de Instruções (IL). Entre as 5 linguagens apresentadas, a mais comum é o diagrama Ladder.

### 3.1.1.Componentes Básicos de um diagrama Ladder

O diagrama Ladder é uma linguagem simbólica que utiliza contatos, bobinas, temporizadores, contadores, instruções de comparação, instruções de cálculos matemáticos elementares, e instruções de cálculos matemáticos complexos. Neste trabalho iremos nos limitar à utilização de somente alguns dentre os componentes citados.

### 3.1.2.Contatos

Um dos principais componentes de um diagrama Ladder são os contatos. Os contatos representam variáveis digitais e são divididos em dois tipos: contatos normalmente abertos e contatos normalmente fechados.

Os contatos normalmente abertos são fechados quando a variável booleana associada a este possui valor lógico verdadeiro, ou igual a 1 (um). De forma análoga, esses contatos são abertos quando a variável booleana associada possui valor lógico falso, ou igual a 0 (zero). Na figura 8 podemos visualizar um contato normalmente aberto associado à variável booleana I0.0.

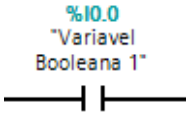



**Figura 8 – Contato normalmente aberto associado a uma variável booleana.**

Os contatos normalmente fechados trabalham de forma contrária aos contatos normalmente abertos. Os contatos normalmente fechados são fechados quando a variável booleana associada a este possui valor lógico falso, ou igual a 0 (zero). De forma análoga, esses contatos são abertos quando a variável booleana associada possui valor lógico verdadeiro, ou igual a 1 (um). Na figura 9 podemos visualizar um contato normalmente fechado associado à variável booleana I0.0.

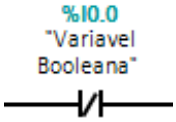



**Figura 9 – Contato normalmente fechado associado a uma variável booleana.**

### 3.1.3. Contatos “Scan Positive Signal Edge and Operand” (tipo P)

O contato “Scan Positive Signal Edge and Operand” (ou contato tipo P) é um tipo especial de contato, utilizado para detectar a subida de uma determinada variável booleana. Este contato funciona de maneira análoga a um contato normalmente aberto, só que ao invés de fechar quando a variável associada apresenta valor lógico igual a 1, esse contato fecha somente durante o ciclo de varredura imediatamente após a subida da variável booleana associada, ou seja, da sua passagem de 0 para 1. Durante o segundo ciclo de varredura após a subida da variável booleana associada, o contato abre, pois nenhuma mudança positiva de estado lógico é detectada.

Os contatos tipo P em controladores Siemens possuem duas variáveis associadas. A primeira variável, posicionada acima do contato, é a variável a qual se deseja detectar a mudança positiva de estado lógico, e a segunda variável, posicionada abaixo do contato, é a variável que armazena o valor da primeira variável durante o último ciclo de varredura do programa do controlador. O contato tipo P detecta a subida da primeira variável associada comparando-a ao valor da segunda variável associada, ou seja, o contato compara seu valor atual com o seu valor durante a última varredura. A partir dessa comparação, caso a subida do valor lógico seja detectada, o contato fecha, caso contrário, o contato abre.

Para ilustrar o funcionamento do contato “Scan Positive Signal Edge”, na figura 10, podemos visualizar um contato tipo P que “seta” a variável booleana 3 mediante a subida da variável associada I0.0. Note ainda que o valor da variável booleana 1, durante o último ciclo de varredura, é armazenado na variável booleana 2.


**Figura 10 – Contato tipo P associado à variável booleana I0.0.**

### 3.1.4.Contatos “Scan Negative Signal Edge and Operand” (tipo N)

O contato “Scan Negative Signal Edge and Operand” (ou contato tipo N) é um tipo especial de contato, utilizado para detectar a descida de uma determinada variável booleana. Este contato funciona de maneira análoga ao contato tipo P, só que ao invés de fechar na subida lógica da variável associada, esse contato fecha na descida da variável booleana associada, ou seja, da sua passagem de 1 para 0.

Para ilustrar o funcionamento do contato “Scan Negative Signal Edge”, na figura 11, podemos visualizar um contato tipo N que energiza a bobina SET associada à variável booleana 3 mediante a descida da variável associada I0.0.


**Figura 11 – Contato tipo N associado à variável booleana I0.0.**

### 3.1.5.Bobinas

As bobinas são componentes fundamentais de um diagrama Ladder. Esses componentes possuem a função de atualizar informações de saída, isto é, dependendo da lógica que antecede as bobinas no diagrama Ladder, estas modificam o estado lógico de variáveis booleanas de acordo com o seu tipo. As bobinas podem ser de três tipos diferentes: bobina simples, bobina SET, bobina de RESET.

As bobinas simples modificam para verdadeiro, instantaneamente, o estado lógico da variável booleana à qual encontra-se associada, caso a lógica que a antecede possua valor lógico verdadeiro. Caso a lógica que antecede esta bobina torne-se falsa, isto é, assuma valor lógico igual a zero, esta é desenergizada e a variável booleana à qual encontra-se associada assume imediatamente valor lógico falso, ou seja, 0 (zero). Na figura 12 podemos visualizar uma bobina simples associada à variável booleana I0.0.

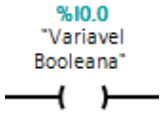



**Figura 12 – Bobina simples associada a uma variável booleana.**

A bobina SET atualiza e retém o valor lógico da variável booleana associada a ela em verdadeiro, ou seja 1. O valor da variável booleana associada à bobina SET recebe o valor 1 quando a lógica que antecede a bobina torna-se verdadeira. Caso a lógica que antecede a bobina SET modifique o seu valor lógico de verdadeiro para falso, a variável de saída, à qual esta encontra-se

associada, permanece com seu valor lógico igual a 1. Isso ocorre pois as bobinas SET são retentores de informação lógica de saída. Após ser *"setado"* em verdadeito, o valor lógico da variável de saída só se tornará falso caso, em alguma outra linha do diagrama Ladder, exista uma bobina de RESET associada à mesma variável booleana sendo energizada. Na figura 13 podemos visualizar uma bobina SET associada à variável booleana I0.0.

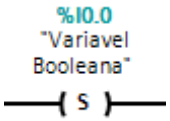



**Figura 13 – Bobina SET associada a uma variável booleana.**

A bobina RESET atualiza e retém o valor lógico da variável booleana associada a ela em falso, ou igual a 0. Isso ocorre quando a lógica que antecede a bobina RESET possui valor lógico verdadeiro. Caso a lógica que antecede a bobina RESET modifique o seu valor lógico de verdadeiro para falso, a variável de saída, à qual esta encontra-se associada, permanece com seu valor lógico igual a 0. Isso ocorre, pois as bobinas RESET são retentores de informação lógica de saída. Após ser *"ressetado"* em falso, o valor lógico da variável de saída só se tornará verdadeiro caso, em alguma outra linha do diagrama Ladder, exista uma bobina de SET associada à mesma variável booleana sendo energizada. Na figura 14 podemos visualizar uma bobina RESET associada a uma variável booleana.

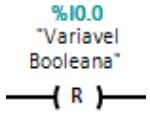



**Figura 14 – Bobina RESET associada a uma variável booleana.**

### 3.1.6.Temporizador TON (Time On Delay)

Os temporizadores são blocos de funções muito utilizados em automação de processos. Existem diversos tipos de temporizadores, que podem atuar de variadas formas na lógica de processo. Neste trabalho, utilizaremos somente o temporizador do tipo TON (Time On Delay).

A quantidade de variáveis envolvidas na lógica de um temporizador do tipo TON, bem como as suas funções e nomenclaturas, varia de acordo com o fabricante do CLP no qual será programada a lógica do processo. Neste trabalho, descrevemos um sistema de manufatura controlado por equipamentos SIEMENS, e por motivos didáticos, descreveremos as variáveis associadas ao temporizador do tipo TON de controladores Siemens, embora sua lógica geral de funcionamento seja similar à lógica dos demais fabricantes.

Um bloco temporizador TON (Time on Delay), como o próprio nome em inglês nos indica, é um bloco que adiciona um atraso. O bloco TON adiciona um atraso em milissegundos igual ao valor pré-configurado dentro do bloco, à condição lógica que o antecede. No caso do bloco temporizador de controladores SIEMENS, o valor deste atraso, chamado de *PRESET,* encontra-se armazenado na variável *PT*.

Quando o temporizador verifica em sua entrada a borda de subida de um sinal, o tempo de atraso inicia sua contagem a partir de zero, e quando este tempo chega ao valor armazenado na variável *PT*, a saída do temporizador passa a possuir valor lógico verdadeiro, ou seja 1. Na figura 15, é possível visualizar um bloco temporizador TON de um controlador SIEMENS.

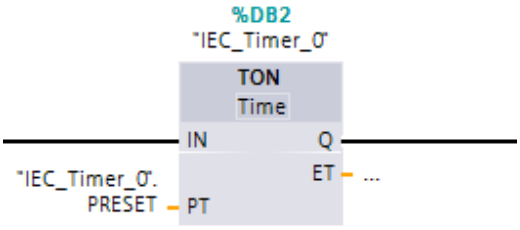



**Figura 15 – Bloco temporizador de um controlador SIEMENS.**

As variáveis associadas aos parâmetros de um bloco temporizador de um controlador SIEMENS, são *PT*, *ET*, *IN*, e *Q*. As descrições de cada um desses parâmetros podem ser visualizadas na tabela 1.

Caso o valor de *IN* torne-se falso antes que o tempo percorrido *ET* alcance o valor armazenado *PT*, o valor de *ET* é zerado e uma nova contagem de tempo é iniciada quando o valor de *IN* tornar-se novamente igual a 1. Isso ocorre, pois o temporizador do tipo TON não é um bloco

temporizador retentivo, ou seja, ele não retém o valor do parâmetro *ET* mediante a descida do seu sinal lógico de entrada *IN*.

**Tabela 1 – Parâmetros associados a um bloco temporizador do tipo TON de um controlador SIEMENS.**

| Parâmetros | Tipo de Variável | Descrição |
|---|---|---|
| IN | Booleana | Sinal Lógico na entrada do bloco temporizador TON |
| PT | Inteira | Tempo de atraso |
| Q | Booleana | Saída atrasada pelo bloco TON por um tempo igual a PT |
| ET | Inteira | Tempo "percorrido" |

### 3.1.7.Blocos comparadores (CMP)

Os blocos comparadores possuem a função de comparar dois valores de variáveis analógicas e podem ser dos seguintes tipos: comparador de igualdade (==), comparador de desigualdade (<>), comparador maior que (>), comparador menor que (<), comparador maior ou igual (>=), e comparador menor ou igual (<=).

Como estamos utilizando uma plataforma SIEMENS, apresentaremos, o bloco comparador maior ou igual, de um controlador SIEMENS, bem como todas as variáveis associadas ao bloco. Este bloco compara a variável declarada acima do bloco com a variável abaixo do bloco, nesta mesma ordem. Caso a saída desta operação lógica seja verdadeira, o bloco habilita a sua saída, funcionando como um contato fechado, caso contrário, o bloco desabilita a sua saída funcionando como um contato aberto.

Na figura 16 podemos visualizar um bloco comparador maior ou igual, que compara duas variáveis inteiras. Caso a operação lógica (Variável Inteira 1 >= Variável Inteira 2) possua valor lógico verdadeiro, a saída do bloco é habilitada.

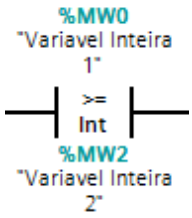



**Figura 16 – Bloco comparador maior ou igual (>=) comparando duas variáveis inteiras.**

O bloco comparador menor ou igual (<=) funciona de forma análoga ao bloco comparador maior ou igual. Esse bloco também compara a variável declarada acima do bloco com a variável abaixo do bloco, nesta mesma ordem. Caso a saída desta operação lógica seja verdadeira, o bloco habilita a sua saída, funcionando como um contato fechado, caso contrário o bloco desabilita a sua saída, funcionando como um contato aberto.

Na figura 17 podemos visualizar um bloco comparador menor ou igual, que compara duas variáveis inteiras. Caso a operação lógica (Variável Inteira 1 <= Variável Inteira 2) possua valor lógico verdadeiro, a saída do bloco é habilitada.

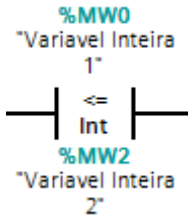



**Figura 17 - Bloco comparador menor ou igual (<=) comparando duas variáveis inteiras.**

### 3.1.8.Blocos de operações matemáticas

Os blocos de operações matemáticas possuem a função de executar operações matemáticas utilizando variáveis analógicas do controlador, e de armazenar o resultado em uma variável também analógica. O CLP SIEMENS possui diversos blocos de operações matemáticas, entretanto, neste trabalho, somente dois blocos são apresentados: bloco de adição (ADD) e o bloco de subtração (SUB).

O bloco de adição (ADD) tem a função de adicionar duas variáveis analógicas associadas a este e de armazenar o resultado da adição em uma terceira variável. Assim, quando a entrada EN do bloco possui valor lógico verdadeiro, o bloco executa a operação de adição, caso contrário, a operação não acontece. Na figura 18 podemos visualizar o bloco de adição (ADD) associado a três variáveis inteiras. O cálculo do bloco da figura 18 pode ser representado pela equação *(Variável Inteira 3 = Variável Inteira 1 + Variável Inteira 2)*.

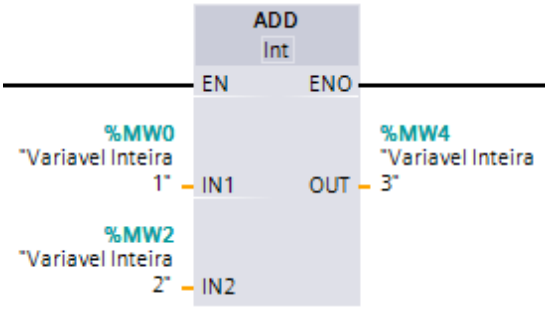



**Figura 18 – Bloco de adição (ADD) que adiciona duas variáveis inteiras e armazena o resultado em uma terceira variável.**

O bloco de subtração (SUB) tem a função de subtrair duas variáveis e armazenar o resultado da subtração em uma terceira variável. Assim, quando a entrada EN do bloco possui valor lógico verdadeiro, o bloco executa a operação de subtração, caso contrário, a operação não acontece. Na figura 19 podemos visualizar o bloco de subtração (SUB) associado a três variáveis inteiras. O cálculo do bloco da figura 19 pode ser representado pela equação *(Variável Inteira 3 = Variável Inteira 1 - Variável Inteira 2)*.

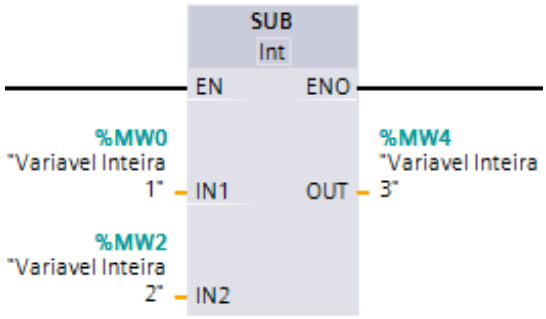



**Figura 19 – Bloco de subtração (SUB) que subtrai duas variáveis inteiras e armazena o resultado em uma terceira variável.**

### 3.1.9.Varredura de um diagrama Ladder

O diagrama Ladder é uma linguagem de programação, e como toda linguagem de programação, ele possui um critério de varredura, isto é, uma forma com que o código será lido pelo processador e executado utilizando os recursos físicos e de memória do controlador.

A varredura do diagrama Ladder ocorre de cima para baixo e da esquerda para a direita. O diagrama é estruturado por linhas, chamadas também de *rungs*, que são percorridas por “correntes elétricas” imaginárias no sentido e direção da varredura do programa. O fluxo da corrente imaginária pode ser interrompido pelo estado lógico dos contatos e blocos existentes ao longo do diagrama, de modo que se o estado lógico do contato for verdadeiro, a corrente passa, e quando o estado lógico é falso a corrente é interrompida.

Na figura 20 a varredura do diagrama Ladder é ilustrada, através de setas, em uma linha de programa do controlador SIEMENS.

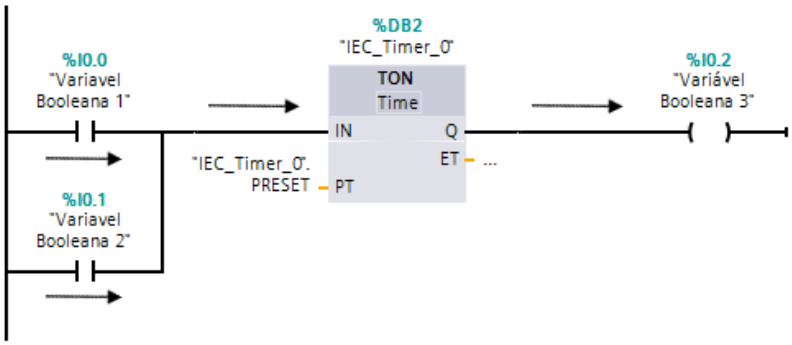



**Figura 20 – Varredura de um diagrama Ladder de um controlador SIEMENS ilustrada por setas.**

## 3.2. Método de Conversão

Após apresentar os principais componentes de um diagrama Ladder na subseção 3.1, e as principais características de redes de Petri Interpretadas para Controle no capítulo 2, podemos entender o método de conversão de RPIC para Ladder.

O método de conversão de RPIC para Ladder proposto em Moreira, Botelho & Basilio (2009), é dividido em 5 módulos: módulo de inicialização, módulo de eventos, módulo de condições para o disparo das transições, módulo da dinâmica da rede de Petri, e módulo das ações.

Para entendermos o método de conversão, utilizaremos um exemplo simples e didático de rede de Petri interpretada para controle. A RPIC do exemplo apresentado na figura 21 representa o funcionamento de um sistema formado por um botão, uma chave automática, e uma lâmpada.

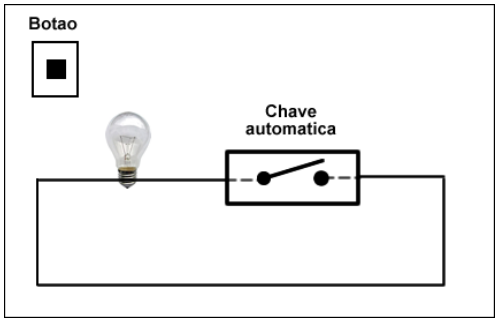



**Figura 21 – Sistema de acionamento de uma lâmpada através de um botão.**

O objetivo do sistema é controlar o estado de uma lâmpada através do acionamento de um botão. Caso a lâmpada esteja apagada, quando o botão for pressionado a lâmpada é ligada, e caso a lâmpada esteja acesa, quando o botão for pressionado esta é apagada. O controlador do sistema percebe o acionamento do botão e envia um comando para que a chave feche ou abra. As variáveis do sistema podem ser visualizadas na tabela 2.

A chave é pré-programada para fechar ou abrir o circuito de acordo com o comando que receber. Caso a chave receba o comando para fechar, ela fecha, e caso a chave receba o comando para abrir, ela abre. As duas variáveis de comando são variáveis booleanas.

O botão é representado por uma variável booleana, que possui valor lógico verdadeiro quando este é pressionado e falso quando não está sendo pressionado. A lâmpada, assim como o botão, possui um sensor de luminosidade que indica se a mesma está ligada ou não.

**Tabela 2 – Tabela de variáveis do sistema de acionamento de uma lâmpada através de um botão.**

| VARIÁVEL | DESCRIÇÃO |
|---|---|
| cmd_fechar | Comando para fechar a chave |
| cmd_abrir | Comando para abrir a chave |
| botao | Sensor de status do botão |
| lampada | Sensor de luminosidade da lâmpada |

A rede de Petri interpretada para controle do sistema de acionamento de uma lâmpada através de um botão pode ser visualizada na figura 22.

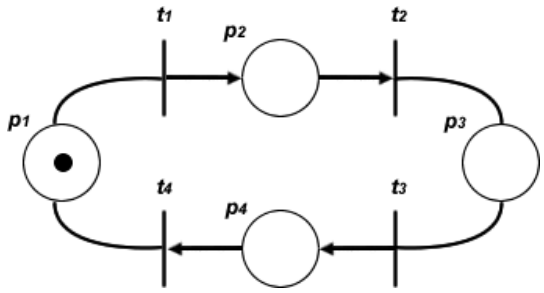



**Figura 22 – RPIC do sistema de acionamento de uma lâmpada através de um botão.**

As receptividades associadas às transições e as ações e estados associados aos lugares da RPIC da figura 22 podem ser vistas nas tabelas 3 e 4, respectivamente.

**Tabela 3 – Receptividades associadas às transições.**

| TRANSIÇÃO | RECEPTIVIDADE | CONDIÇÃO | EVENTO |
|---|---|---|---|
| t1 | R1 | 1 | [botao]↑ |
| t2 | R2 | 1 | [lâmpada]↑ |
| t3 | R3 | 1 | [botao] ↑ |
| t4 | R4 | 1 | [lâmpada]↓ |

**Tabela 4 – Ações e estados associados aos lugares.**

| LUGAR | ESTADO | AÇÃO |
|---|---|---|
| p1 | Lâmpada desligada | ---- |
| p2 | ---- | Comando de ligar lâmpada |
| p3 | Lâmpada ligada | ---- |
| p4 | ---- | Comando de desligar lâmpada |

### 3.2.1.Módulo de Inicialização

O módulo de inicialização está associado ao estado inicial da rede de Petri, isto é, ele define a marcação inicial da rede. O módulo de inicialização é representado somente por uma linha (*rung)* no diagrama Ladder. Essa linha possui um contato normalmente fechado associado a uma variável binária interna, que no primeiro ciclo de varredura energiza logicamente bobinas associadas a lugares marcados. Após o primeiro ciclo de varredura o contato normalmente fechado é aberto, não permitindo que esta linha seja executada novamente.

O módulo de inicialização do sistema descrito na figura 21 é apresentado na figura 23.

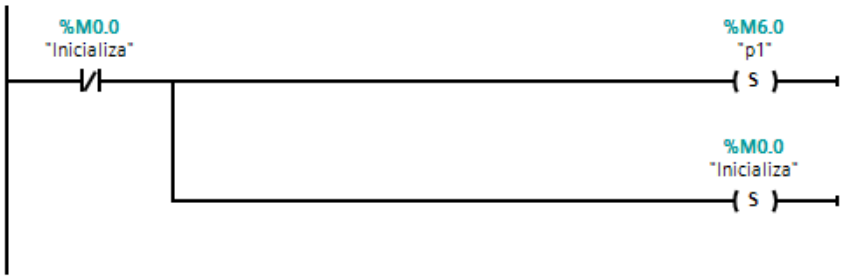



**Figura 23 – Módulo de Inicialização – sistema de acionamento de uma lâmpada.**

### 3.2.2.Módulo de eventos

O módulo de eventos consiste na interpretação de eventos externos como eventos internos, de forma a gerar reações correspondentes por parte do controlador. Os eventos externos são associados a contatos, e interpretados internamente através de bobinas. Cada contato associado a um evento externo é representado por um contato tipo P, que representa a subida da variável binária associada. Assim, a presença do contato tipo P evita que ações associadas a eventos com duração maior que um ciclo de varredura do programa sejam executadas mais de uma vez.

Note que o evento 3 é associado a um contato tipo N. Este contato é utilizado pois o evento 3 está associado à descida da variável “lâmpada”. Note ainda, que o evento 4 não possui nenhum contato associado. Esta configuração é adotada pois o evento 4 é o evento que sempre ocorre *e*.

O módulo de eventos do sistema descrito na figura 21 pode ser visualizado na figura 24.

### 3.2.3.Módulo de condições para o disparo das transições

O módulo de condições para o disparo das transições descreve as condições de habilitação presentes na matriz de incidência de entrada, $A_{in}$, e as receptividades associadas. Esse módulo possui $n$ linhas, sendo que cada linha corresponde à marcação do lugar $p_i$ mediante o disparo da transição $t_j$. Abaixo é possível visualizar a matriz de incidência de entrada, $A_{in}$, do sistema cuja representação em rede de Petri é apresentada na figura 22.

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

A idéia básica é descrever a regra de disparo para uma transição, lembrando que uma transição $t_j$ é habilitada quando todos os lugares de entrada possuem fichas e a receptividade $R_j$ associada a ela possui valor lógico verdadeiro. Para uma determinada transição $t_j$, por exemplo, as condições associadas no diagrama Ladder são expressas por uma associação de contatos.

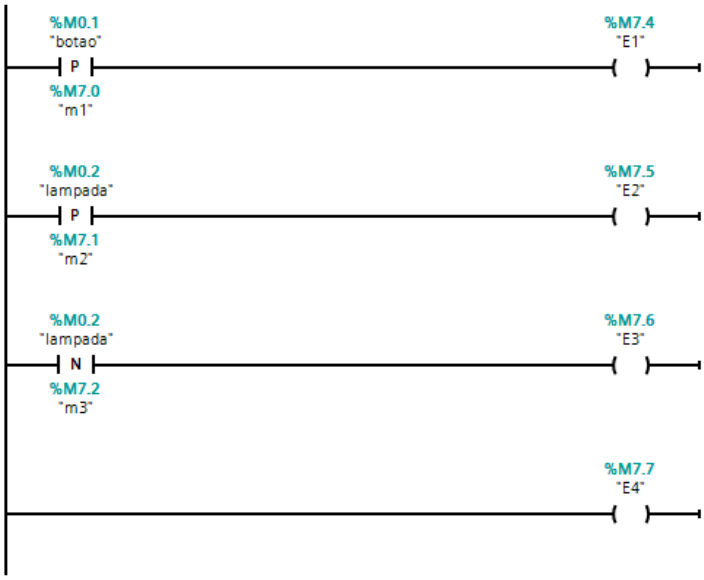



**Figura 24 – Módulo de Eventos – sistema de acionamento de uma lâmpada.**

Para representar o disparo da transição $t_j$, a expressão booleana para a receptividade $R_j$ (implementada, geralmente, com uma simples associação de contatos normalmente abertos e normalmente fechados) é conectada em série com as condições dos lugares de entrada de $t_j$.

O módulo de condições para o disparo das transições do sistema descrito na figura 21 pode ser visualizado na figura 25.

### 3.2.4.Módulo da dinâmica

Após o disparo de uma transição $t_j$, o número de fichas da rede de Petri deve ser atualizado. Este processo é descrito pela equação de estado da rede de Petri, capaz de atualizar a marcação da rede mediante o disparo de uma determinada transição, como visto na subseção 2.5.


**Figura 25 – Módulo de condições para o disparo das transições – sistema de acionamento de uma lâmpada.**

Para a construção do módulo responsável pela dinâmica da rede de Petri devemos utilizar a matriz de incidência *A*, cuja notação também fora apresentada na subseção 2.5. Este módulo possui *n* linhas, sendo que cada linha está associada a um lugar e expressa as mudanças na marcação dos lugares após o disparo da transição associada. Abaixo é apresentada a matriz de incidência, *A*, do sistema cuja representação em rede de Petri é apresentada na figura 22.

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

O módulo da dinâmica do sistema descrito na figura 21 é apresentado na figura 26.

### 3.2.5.Módulo das ações

O módulo das ações associa uma bobina de saída ao lugar que possui uma ação instantânea. Desta forma, mediante o disparo de uma determinada transição, as ações associadas aos lugares de saída são executadas por meio de associações com bobinas. O tamanho do módulo das ações depende do número de lugares que possuem ações associadas.

O módulo das ações do sistema descrito na figura 21 pode ser visualizado na figura 27. Note ainda que na figura 27, $p_2$ e $p_4$, pois são os únicos lugares que possuem ações associadas.

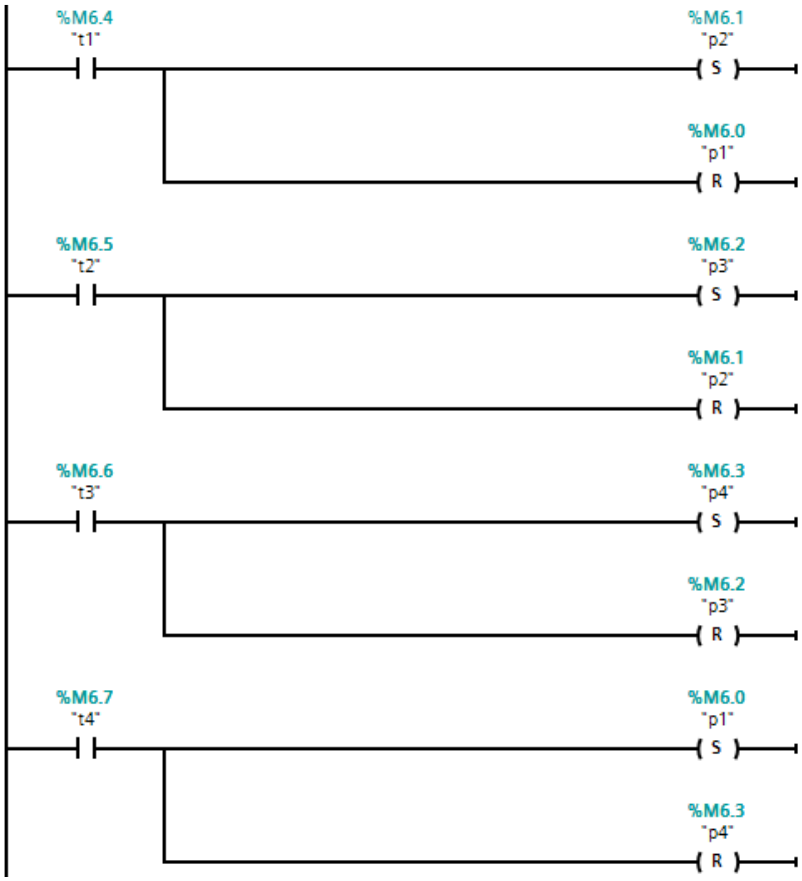



**Figura 26 – Módulo da dinâmica – sistema de acionamento de uma lâmpada.**

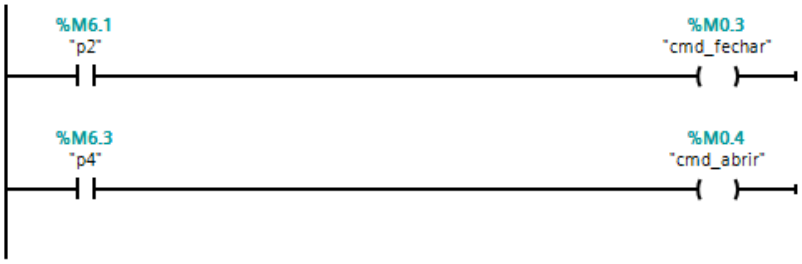



**Figura 27 – Módulo das ações – sistema de acionamento de uma lâmpada.**

### 3.2.6. Observações sobre o método de conversão

O método de conversão proposto em Moreira, Botelho & Basílio (2009) possui características especiais, que otimizam o diagrama Ladder traduzido. Estas características especiais encontram-se destacadas abaixo:

- A disposição das linhas (*rungs*) propostas pelo método evita o efeito avalanche [5] e [6], uma vez que cada lugar na rede de Petri Interpretada para Controle mantém sua marcação durante, no mínimo, um ciclo de varredura do programa Ladder.
- O número de linhas do programa Ladder obtido pelo método possui um valor máximo de (*2m+n+1)*, em que *m* é o número de transições, e *n* o número de lugares. Embora este número de linhas possa ser menor, o diagrama Ladder traduzido mostra claramente a estrutura da rede de Petri, uma vez que $A_{in}$ e *A* podem ser obtidas a partir do diagrama Ladder. Além disso, as ações e as receptividades são facilmente relacionadas com seus lugares e transições correspondentes. Desta forma, a RPIC pode ser obtida diretamente a partir do diagrama Ladder, e vice-versa.
- Caso a RPIC possua um arco inibidor, então uma modificação deverá ser feita na matriz de incidência de entrada para que o método possa ser utilizado [7]. Neste caso, essa matriz de incidência de entrada modificada não deverá ser utilizada no cálculo da matriz de incidência da RPIC, uma vez que o disparo de uma transição associada a um arco inibidor não modifica o número de fichas no lugar de entrada. Entretanto, ela pode ser utilizada na obtenção das condições no módulo de condições de disparo das transições.

### 3.3. Conclusão

Neste capítulo foi apresentada a linguagem Ladder, que segundo a norma internacional IEC 61131-3, pode ser utilizada em todos os controladores industriais. Vimos os principais componentes de um diagrama Ladder e a forma com que este diagrama é lido pelo processador do controlador industrial.

Além da apresentação do diagrama Ladder, foi proposto, baseado em Moreira, Botelho & Basílio (2009), um método de conversão de redes de Petri Interpretadas para Controle para diagrama Ladder. Este método foi introduzido com um exemplo didático e será utilizado ao longo deste trabalho para a conversão da RPIC do sistema de manufatura em diagrama Ladder [8].

# 4 CONFIGURAÇÃO DO SOFTWARE - TUTORIAL

Neste capítulo é apresentado um tutorial para a configuração do Software *Totally Integrated Automation Portal*, responsável pela programação e configuração do CLP e da IHM SIEMENS. O tutorial pode ser utilizado para a realização de projetos de Automação que utilizem a plataforma SIEMENS no Laboratório de Controle e Automação da Universidade Federal do Rio de Janeiro, bem como na elaboração da automação do sistema de manufatura apresentado neste trabalho.

## 4.1. Apresentação do Hardware

No Laboratório de Controle e Automação da UFRJ são utilizados quatro equipamentos básicos de automação fabricados pela SIEMENS. São eles: um controlador SIEMENS CPU 1214 C AC/DC/Rly (6ES7 214-1BE30-X0B0), um módulo externo de saídas analógicas de 12bits (6ES7 232-4HA30-0XB0) compatível com o controlador, uma IHM SIEMENS de 6 polegadas modelo KTP600 PN, e um hub Ethernet SIEMENS. Na figura 28 é possível visualizar os equipamentos do Laboratório de Controle e Automação da UFRJ.


**Figura 28 – Equipamentos SIEMENS do Laboratório de Controle e Automação da UFRJ.**

## 4.2. Apresentação do Software

Para integrar a estrutura de hardware apresentada na seção anterior, precisamos de um sistema computacional compatível com esta estrutura, integrado com os equipamentos por meio de um computador. Essa integração pode ser feita quando os dispositivos são conectados de forma que

exista um sincronismo de protocolo utilizado entre eles. Na figura 29 pode ser visualizada a arquitetura dos equipamentos dispostos em rede.

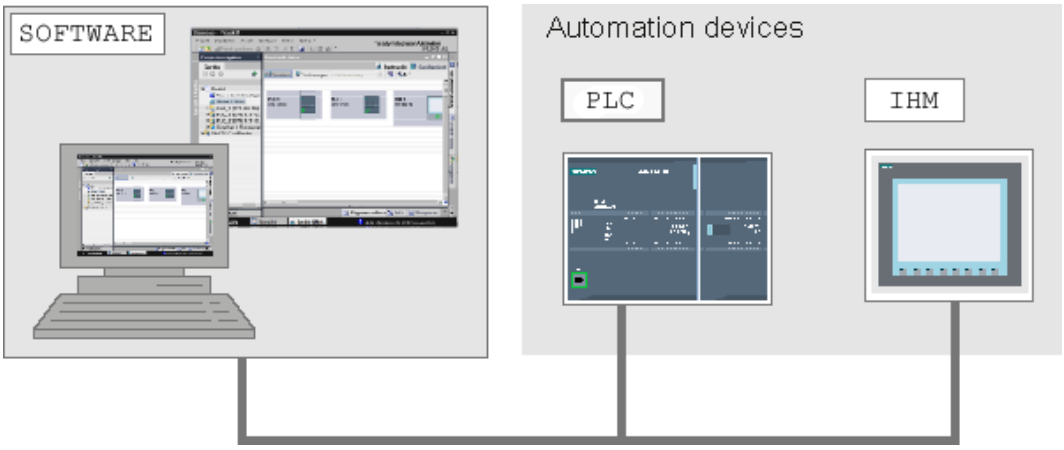



**Figura 29 – Arquitetura dos equipamentos de automação dispostos em rede.**

## 4.3. Tutorial

O processo de definição da arquitetura é o primeiro passo descrito neste tutorial. A primeira fase consiste em identificar os componentes e seus atributos para configurá-los de maneira correta.

### 4.3.1. Iniciando um projeto

Para iniciar um projeto, basta clicar no ícone correspondente ao software *Totally Integrated Automation Portal V10* (WinCC e STEP7 integrados em uma só plataforma) no desktop do computador. Assim que abrir o software, ao clicar em “New Project” é possível ver a tela presente na figura 30.

### 4.3.2. Adicionando equipamentos (devices)

Após iniciarmos o projeto, devemos descrever para o software todos os equipamentos a serem configurados. Nesta etapa, devemos saber exatamente cada tipo de equipamento que será utilizado no laboratório, bem como seus nomes técnicos.

- CPU 1214 C AC/DC/Rly (6ES7 214-1BE30-X0B0);
- ANALOG OUTPUTS MODULE - módulo externo de saídas analógicas de 12bits (6ES7 232-4HA30-0XB0);

- HMI SIEMENS 6’’ KTP600 PN.

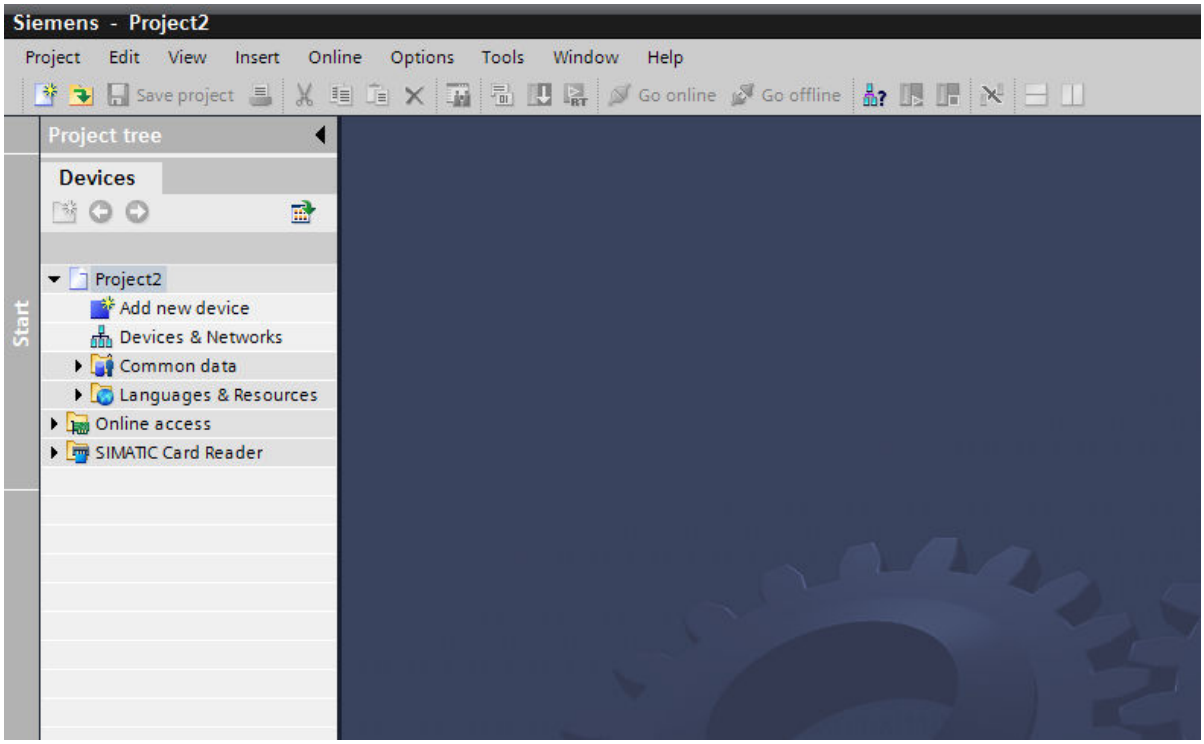



**Figura 30 – Tela de Inicialização de Projeto – *Totally Integrated Automation Portal V10* SIEMENS.**

Os equipamentos descritos acima estão integrados na mesma plataforma, ou seja, através do mesmo software eles podem ser programados, configurados, e dispostos de forma que possam compartilhar informações (variáveis de processo) via protocolo de comunicação profibus e meio físico Ethernet.

Ao clicarmos em “*add device*”, no menu “*Devices*” em “*Project tree*”, a tela apresentada na figura 31 poderá ser visualizada.

#### 4.3.2.1. Adicionando IHM (HMI device)

Para adicionar a IHM devemos selecionar o ícone *SIMATIC HMI* destacado na figura 31. Em seguida, devemos selecionar o modelo KTP600 PN, de 6 polegadas, e pressionar o botão “OK”.

A tela e as opções que deverão ser selecionadas são apresentadas na figura 32.

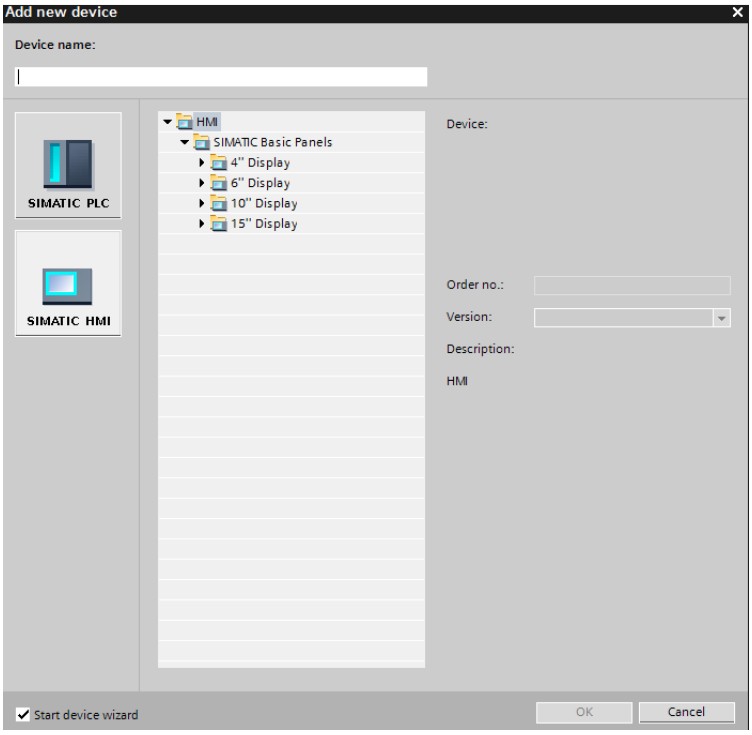



**Figura 31 – Tela para adicionar novos Equipamentos de automação.**

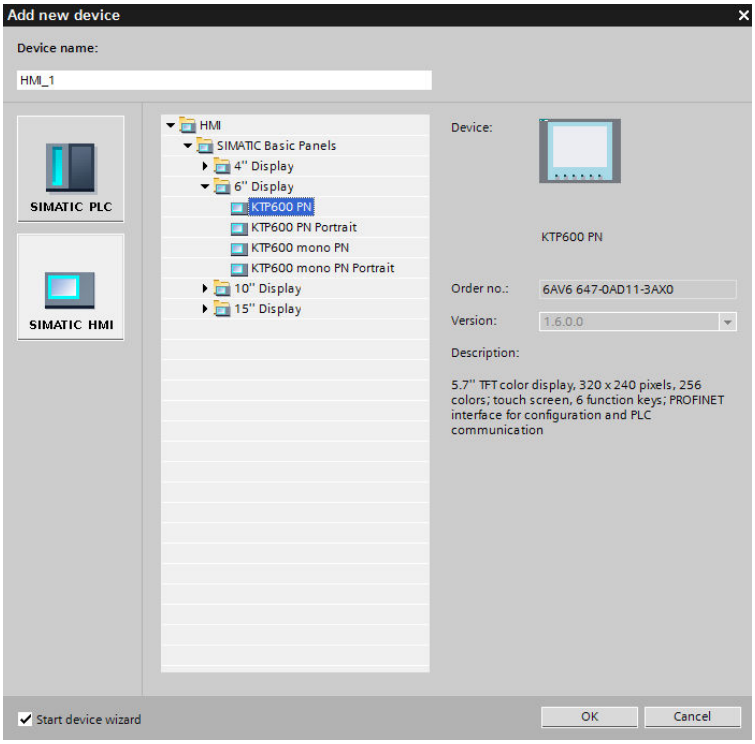



**Figura 32 – Adicionando nova IHM ao sistema de Automação SIEMENS.**

#### 4.3.2.2. Adicionando CLP (PLC device)

Para adicionar o CLP devemos, na tela "*Add new device*", selecionar o ícone *SIMATIC PLC* destacado na figura 31. Em seguida, devemos selecionar o modelo de CPU 1214 C AC/DC/Rly (6ES7 214-1BE30-X0B0), e pressionar o botão "OK".

A tela e as opções que deverão ser selecionadas são apresentadas na figura 33.

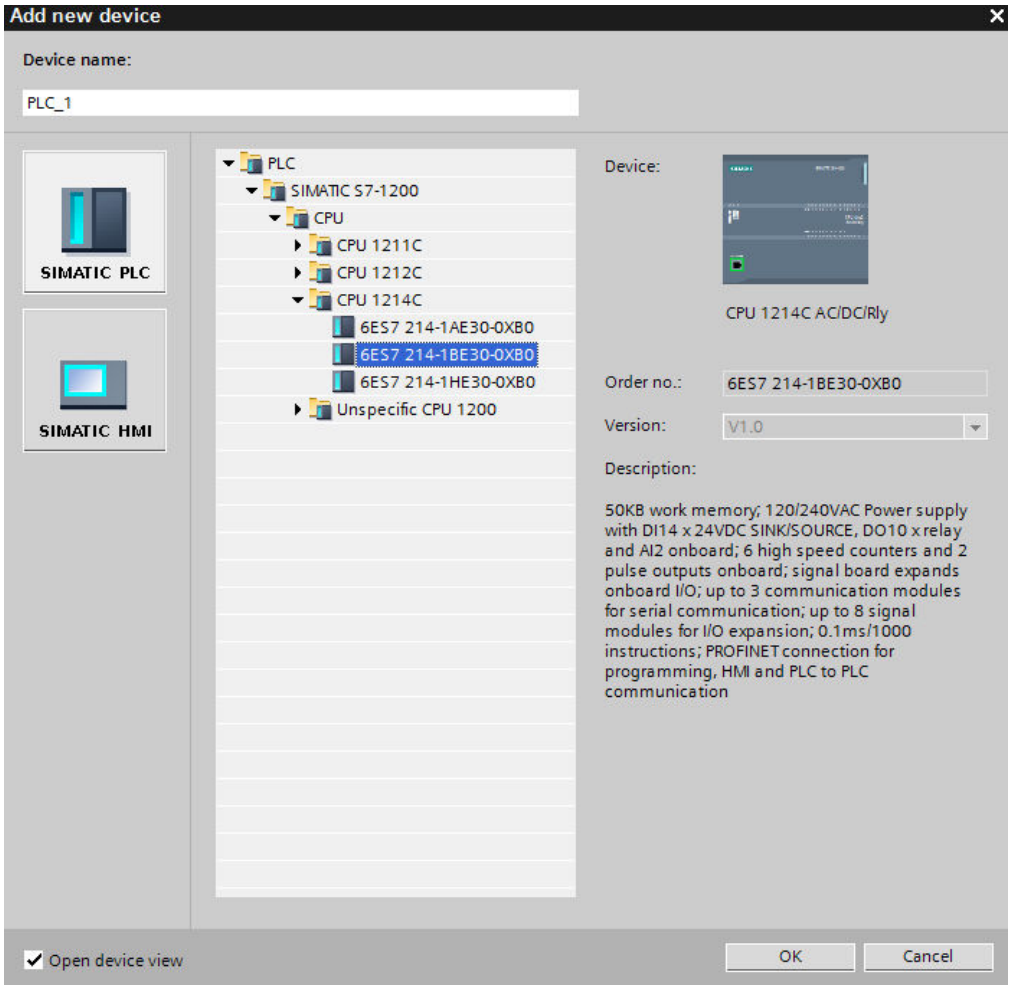



**Figura 33 – Adicionando novo CLP ao sistema de Automação SIEMENS.**

#### 4.3.2.3. Adicionando módulo externo de saídas analógicas (AO1x12bits)

Para configurar o módulo externo de saídas analógicas devemos alterar a configuração padrão do controlador. A partir da tela inicial da figura 30 acessamos o menu *"Devices"* em *"Project Tree"*, à esquerda da janela do software. No menu *"Devices"* podemos visualizar o ícone do controlador, com o nome de *"PLC_1"*. Ao clicarmos no ícone *"PLC_1"*, o controlador S7-1200

aparecerá na janela do software. Na parte central do controlador visualizado devemos clicar com o botão direito do mouse no local onde se conecta fisicamente o módulo externo, e em seguida na opção *"Display catalog"*. A tela indicativa que pode ser visualizada na figura 34 se abrirá.

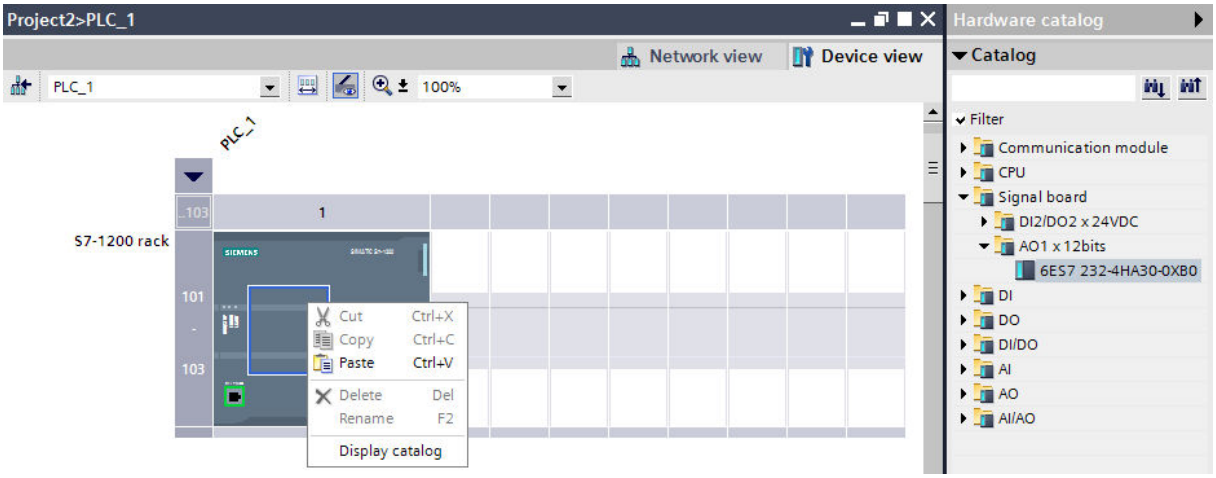



**Figura 34 – Adicionando Módulo Externo de Saídas Analógicas ao CLP SIEMENS.**

Em seguida, devemos selecionar exatamente o cartão de saída analógica correspondente ao que possuímos. Para isso, selecionamos no menu à direita a opção que corresponde ao cartão 6ES7 232-4HA30-0XB0, isto é, Signal Board > AO1x12bits > 6ES7 232-4HA30-0XB0.

A tela e as opções que deverão ser selecionadas podem ser visualizadas na figura 35.

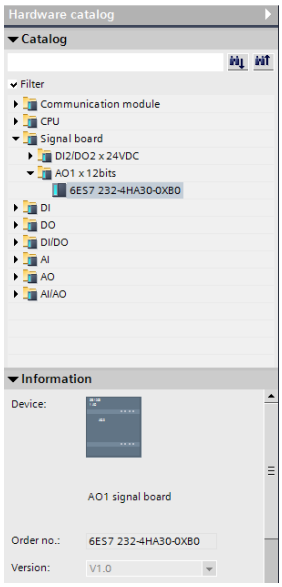



**Figura 35 - Opções de seleção do módulo externo de saídas analógicas.**

### 4.3.3.Configurando conexões da IHM (devices)

Após adicionarmos a IHM, um utilitário do programa é aberto para iniciarmos um processo de configuração da interface touch screen. Esse utilitário é chamado "*HMI Device Wizard*".

Na primeira tela do utilitário de configuração devemos selecionar o device "PLC_1", através do botão *Browse*, para que possamos criar o link de conexão entre a IHM e o CLP. Feito isso, esses dispositivos estarão prontos para compartilharem variáveis entre si. Esse procedimento pode ser visualizado na figura 36.

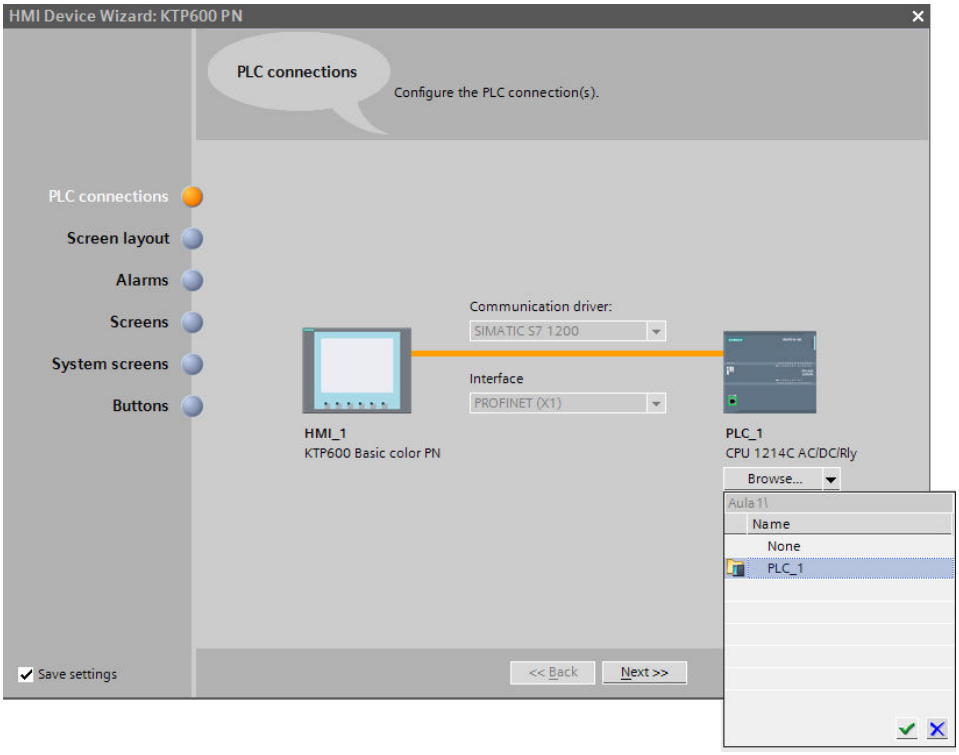



**Figura 36 – Configuração de rede entre a IHM e o CLP.**

### 4.3.4.Configurando Network (Ethernet Configuration)

Até agora possuímos um sistema de automação formado por 2 equipamentos e um módulo externo de saída analógica de 12 bits. O próximo passo é configurar o IP dos equipamentos para que estes, após serem conectados em uma rede profibus, possam compartilhar dados entre si.

Para alterarmos as configurações de rede de cada equipamento devemos entrar nas configurações de cada um deles e modificar as opções de Ethernet.

Alterar as configurações de rede de cada equipamento significa atribuir para cada um deles um endereço IP, de modo que estes possam ser endereçados e identificados na rede profibus.

Os endereços IP são constituídos por uma seqüência numérica que identifica cada ponto, ou nó, em uma rede de dispositivos. Assim, cada dispositivo do laboratório de automação deve possuir um endereço IP único, que não poderá ser atribuído a nenhum outro dispositivo dentro da mesma rede.

A atribuição de endereços IP deve obedecer a estrutura determinada pela máscara de sub-rede. Essa máscara limita a faixa de endereços lógicos que devem ser utilizados para formar os endereços IP e especifica a classe de rede utilizada.

No caso do sistema de manufatura especificado neste trabalho, adotaremos uma faixa de IPs que corresponde à números que são convencionalmente utilizados em redes privadas ou particulares, e possuem máscara de sub-rede igual a "*255.255.255.0*".

A escolha da faixa de IPs não influencia no funcionamento dos equipamentos, desde que estes possuam endereços IP diferentes entre si e que obedeçam à máscara de sub-rede especificada.

Uma vez inserido na rede profibus, o computador responsável pela programação dos equipamentos também deverá sofrer alterações no que diz respeito às suas configurações de rede. O primeiro passo é acessar as propriedades da conexão de rede utilizada através do painel de controle (caso o SO Windows esteja sendo utilizado), selecionar o item de conexão *TCP/IPv4*, e clicar em "*Properties*". Este procedimento pode ser visto na figura 37.

O segundo e último passo, é definir o endereço IP, a máscara de sub-rede, e o Gateway Padrão do computador. A janela que se abre após ser realizado o primeiro passo deve ser preenchida conforme indicado na figura 38.

#### 4.3.4.1. IHM Network

Para configurar o endereço IP da IHM, devemos acessar a configuração de Ethernet deste equipamento. Para isso devemos acessar, no menu *"Project Tree"*, o item *Devices & Networks*. Em

seguida, ao selecionar a IHM, podemos visualizar suas configurações gerais. Nesta mesma janela, para acessarmos as configurações de rede da IHM, devemos clicar na opção *"Ethernet Addresses"*.

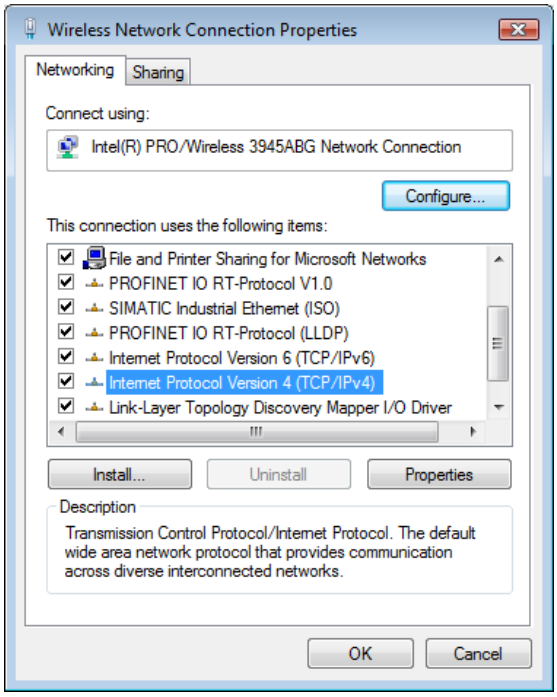



**Figura 37 – Procedimento de Configuração da placa de rede do computador.**

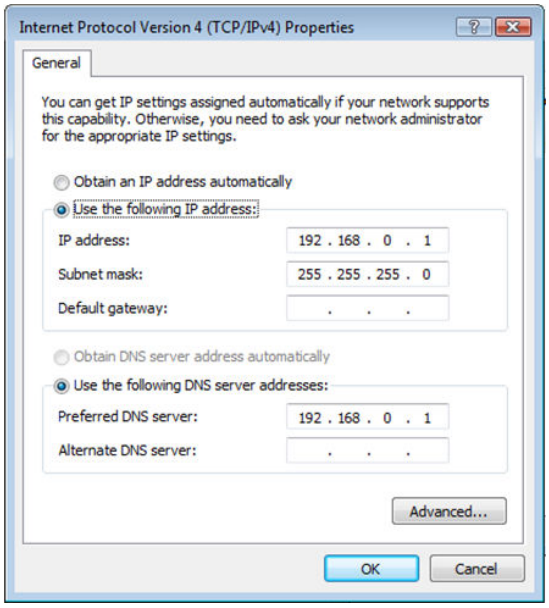



**Figura 38 – Configuração do endereço IP e da máscara de sub-rede do computador.**

Na opção *IP PROTOCOL* selecionamos as especificações de IP e a máscara de sub-rede. Para padronizar os endereços IP da rede profibus, escolheremos:

- IP address: 192.168.0.2
- Subnet mask: 255.255.255.0

A tela para configuração das especificações de rede do sistema pode ser visualizada na figura 39.

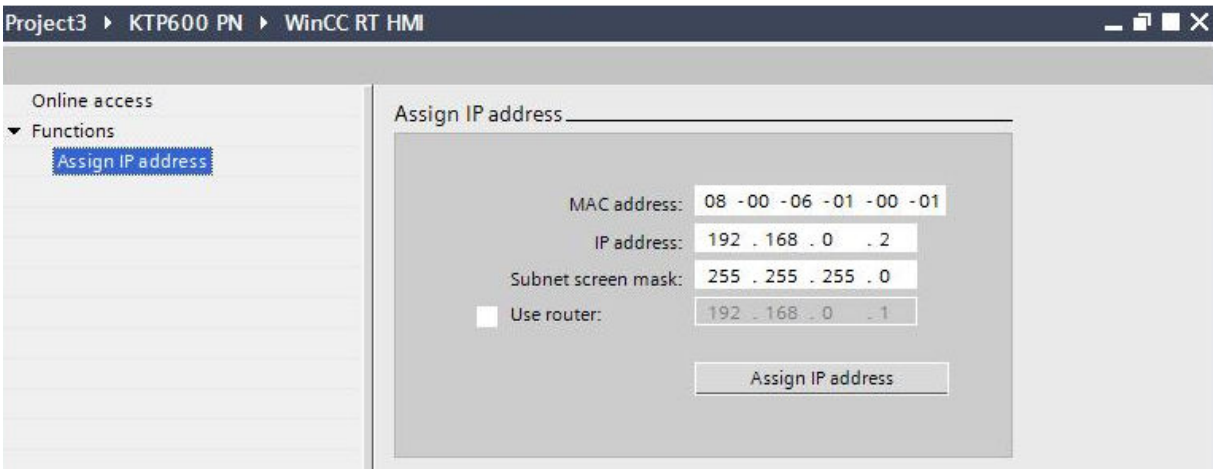



**Figura 39 – Configuração do IP e da máscara de sub-rede da IHM.**

#### 4.3.4.2. PLC Network

Analogamente à IHM, para configurar o endereço IP do CLP, devemos acessar a configuração de Ethernet deste equipamento. Para isso devemos acessar, no menu *"Project Tree"*, o item "*Devices & Networks"*. Em seguida, ao selecionar o CLP, podemos visualizar as suas configurações gerais. Nesta mesma janela, devemos clicar na opção *"Ethernet Addresses"*.

Na opção *IP PROTOCOL* selecionamos as especificações de IP e a máscara de sub-rede. Para padronizar os endereços IP da rede Profinet, escolheremos:

- IP address: 192.168.0.3
- Subnet mask: 255.255.255.0

A tela para configuração das especificações de rede do sistema pode ser visualizada na figura 40.

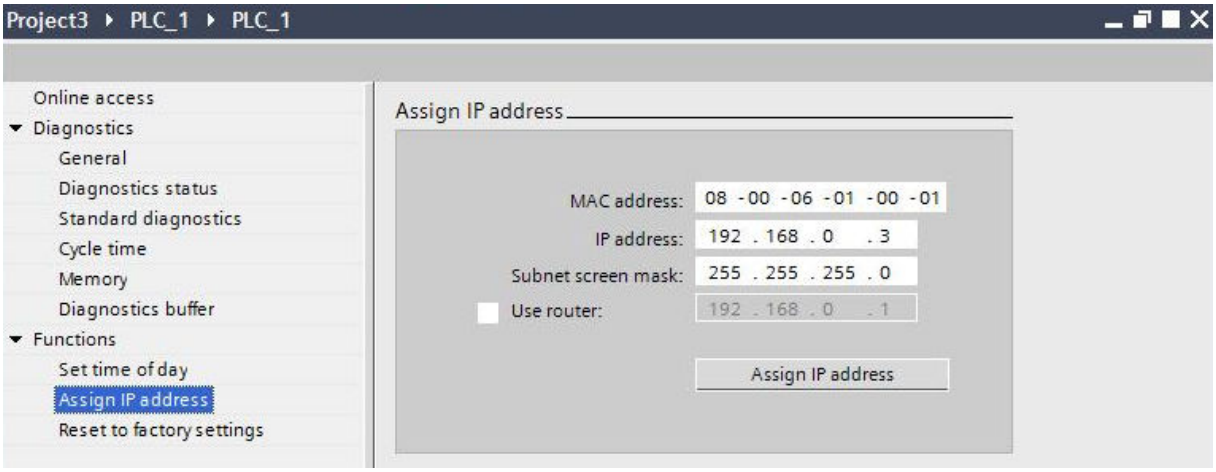



**Figura 40 – Configuração do IP e da máscara de sub-rede do CLP.**

### 4.3.5.TAGs de sistema

Sistemas de automação utilizam uma estrutura própria de variáveis chamadas de TAGs. TAGs podem ser interpretadas como variáveis de um sistema de automação, ou seja, espaços alocados na memória do controlador ou das interfaces homem-máquina (IHMs).

Para compreendermos exatamente o papel das variáveis do sistema de automação e sabermos como os dispositivos trocam dados entre si, devemos entender primeiro a forma como esses equipamentos estão dispostos na rede de automação. Para facilitar o entendimento podemos fazer analogia a um sistema computacional.

Em uma rede doméstica convencional, possuímos os computadores que fazem parte da rede e um hub, através do qual os computadores estão conectados entre si. Cada computador possui um endereço IP, capaz de identificá-los individualmente, de forma que os computadores possuam uma identidade dentro da rede. Neste caso o meio físico utilizado é o Ethernet, e o protocolo TCP/IP. No caso da célula de manufatura apresentada neste trabalho, os dispositivos também estão dispostos em forma de rede, conectados entre si através de um hub, onde cada equipamento também possui um endereço IP capaz de identificá-los individualmente. Neste caso o meio físico utilizado é Ethernet, e o protocolo utilizado é chamado de profibus.

Uma vez identificados, os dispositivos de automação podem trocar dados, isto é, as variáveis utilizadas em um determinado dispositivo podem ser visualizadas pelos outros. No item 4.3.3 determinamos as conexões da rede de automação, onde a IHM estaria conectada ao CLP através de uma sub-rede. Esta disposição nos permite compartilhar variáveis presentes nestes dois equipamentos. Assim, uma variável utilizada no controlador, por exemplo, poderá ser visualizada e alterada pela IHM. O mesmo acontece com as variáveis presentes na IHM, que podem ser visualizadas e alteradas pelo controlador.

#### 4.3.5.1. Tipos de TAGs

Analogamente a um sistema computacional, cada TAG deve ser declarada de acordo com a estrutura da variável de processo que esta representa. Para explicarmos os tipos de TAGs de forma clara utilizaremos um exemplo simples de processo que utiliza uma variável analógica e uma variável digital.

Considere um motor cuja velocidade de rotação desejada é definida através de um teclado virtual pelo usuário e suponha que exista um botão que possa ligar ou desligar o sistema em qualquer momento. Esse sistema pode ser visualizado na figura 41.

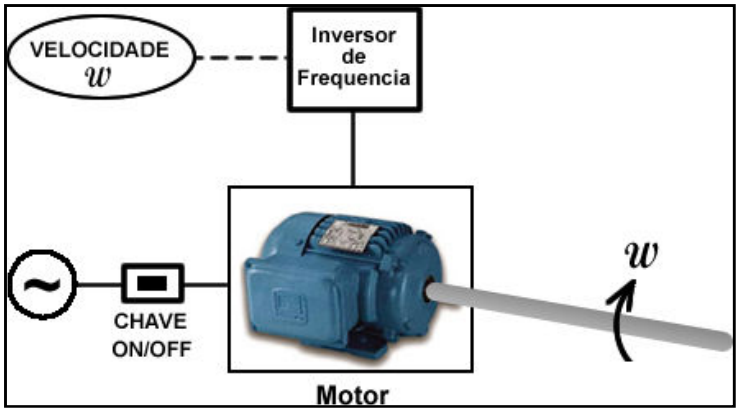



**Figura 41 – Exemplo de um sistema com duas variáveis de entrada.**

O sistema descrito no exemplo acima utiliza uma variável digital, cujo estado está associado à chave que liga ou desliga o motor, e uma variável analógica, necessária para indicarmos ao inversor de frequência a velocidade do rotor do motor. Para este exemplo estamos focando somente na estrutura de variáveis e a forma como estas são tratadas no sistema integrado de automação

SIEMENS, não havendo a necessidade, por exemplo, de nos preocuparmos com a forma com que estas serão utilizadas na lógica de funcionamento do exemplo.

Para associarmos cada variável a um tipo de estrutura computacional devemos primeiro conhecer os tipos de variáveis que o controlador SIEMENS consegue interpretar. Na tabela 5 podemos visualizar os tipos de variáveis que podem ser lidas pelo sistema de automação SIEMENS.

**Tabela 5 – Tipos de Variáveis de sistemas de Automação SIEMENS.**

| **Tipo de Variável** | **Comprimento (bits)** | **Formato Padrão** | **Faixa de Valores** | **Exemplo de Valor de Entrada** |
|---|---|---|---|---|
| BOOL | 1 | Boolean | TRUE/FALSE | TRUE |
| BYTE | 8 | Hexadecimal number | 16#0 a 16#FF | 16#F0 |
| WORD | 16 | Hexadecimal number | 16#0 a 16#FFFF | 16#F0F0 |
| DWORD | 32 | Hexadecimal number | 16#0000_0000 a 16#FFFF_FFFF | 16#F0F0_F0F0 |
| SINT | 8 | Signed integers | -128 a 127 | (+)120 |
| USINT | 8 | Unsigned integers | 0 a 255 | 50 |
| INT | 16 | Signed integer | -32768 a 32767 | (+)1 |
| UINT | 16 | Unsigned integers | 0 a 65535 | 300 |
| DINT | 32 | Signed integers | - 2 147 483 648 a + 2 147 483 647 | (+)2131754992 |
| UDINT | 32 | Unsigned integers | 0 a 4294967295 | 4042322160 |
| REAL | 32 | Floating-point numbers | -3.402823e+38 a -1.175 495e-38<br><br>±0<br><br>+1.175 495e-38 a +3.402823e+38 | 1.234567e+13 |
| TIME | 32 | Time period with sign: | T# -24d20h31m23s648ms<br><br>a<br><br>T#+24d20h31m23s647ms | T#10d20h30m20s630ms |
| CHAR | 8 | ASCII characters | ASCII character set | 'I' |

No caso do exemplo descrito na figura 41, utilizando a tabela 5 para nos orientar na escolha das variáveis utilizadas, devemos declarar a variável digital como BOOL (booleana), e a variável analógica como INT. Na verdade, para a variável analógica, pode-se declará-la como diversos tipos especificados na tabela 5, mas por convenção e também por motivos didáticos, utilizaremos um tipo mais simples de estrutura computacional.

#### 4.3.5.2. Endereços Mnemônicos

Cada tipo de variável escolhida, por sua vez, possui um símbolo correspondente na memória do controlador e da interface homem-máquina. Esses símbolos ou representações são chamados de endereços mnemônicos. O entendimento de mnemônicos é a base da programação de controladores

**Tabela 6 – Estrutura de Endereços Mnemônicos dos sistemas de Automação SIEMENS.**

| Mnemônicos ingleses | Mnemônicos alemães | Definição | Tipo de Variável | Áreas de Endereço |
|---|---|---|---|---|
| I | E | Input bit | BOOL | 0.0..1023.7 |
| IB | EB | Input byte | BYTE, CHAR, SINT, USINT | 0..1023 |
| IW | EW | Input word | WORD, INT, UINT | 0..1022 |
| ID | ED | Input double word | DWORD, DINT, UDINT, REAL, TIME | 0..1020 |
| Q | A | Output bit | BOOL | 0.0..1023.7 |
| QB | AB | Output byte | BYTE, CHAR, SINT, USINT | 0..1023 |
| QW | AW | Output word | WORD, INT, UINT | 0..1022 |
| QD | AD | Output double word | DWORD, DINT, UDINT, REAL, TIME | 0..1020 |
| M | M | Memory bit | BOOL | 0.0..8191.7 |
| MB | MB | Memory byte | BYTE, CHAR, SINT, USINT | 0..8191 |
| MW | MW | Memory word | WORD, INT, UINT | 0..8190 |
| MD | MD | Memory double word | DWORD, DINT, UDINT, REAL, TIME | 0..8188 |

industriais, uma vez que cada fabricante possui a sua própria estrutura de símbolos mnemônicos. Na tabela 6 podemos visualizar a estrutura de mnemônicos utilizada em controladores e sistemas de automação SIEMENS.

#### 4.3.5.3. Endereçamento de variáveis (TAGs) na memória do controlador

As redes de automação trabalham com um sistema de endereçamento de variáveis que define a posição em que estas se encontram alocadas nas memórias do controlador e da IHM. Neste caso, a forma com que as variáveis são nomeadas durante a sua declaração influencia no local exato da memória do controlador onde estas serão alocadas.

A forma com que as variáveis são alocadas na memória do controlador, por sua vez, depende também do tipo de variável escolhida. Isto acontece, pois cada tipo de variável ocupa um determinado espaço na memória. Por exemplo, uma variável do tipo BOOL ocupa somente 1 bit de memória, enquanto uma variável do tipo WORD ocupa 16 bits de memória.

O parâmetro que define quanto de espaço na memória cada tipo de variável pode ocupar é atribuído pelo “Range de endereços”. Desta forma, as variáveis do controlador devem ser endereçadas e numeradas de acordo com a tabela 7.

Na tabela 7 podemos identificar também que cada área na memória do controlador compreende um conjunto de definições de variáveis, que são representadas pelos seus endereços mnemônicos e alocadas em um determinado range de endereços de memória do controlador.

#### 4.3.5.4. Relacionando TAGs aos Endereços Mnemônicos

A escolha dos endereços mnemônicos é estritamente relacionada à escolha do tipo de variável que iremos trabalhar. No caso do exemplo descrito na figura 41, utilizando a tabela 5 para nos orientar na escolha do tipo de variável a ser utilizada, e a tabela 6 para relacionarmos o tipo de variável ao seu endereço mnemônico, temos:

- Chave ON/OFF → BOOL (variável digital) → Q0.1 (mnemônico de saída digital)
- Velocidade w → INT (variável inteira) → QW1 (mnemônico de saída analógica)

Note que a forma com que o endereçamento das variáveis foi feito obedece às especificações feitas no item 4.3.5.3, detalhadamente relacionadas na tabela 7.

### 4.3.6.Criação de TAGs

Nesta subseção indicaremos como criar as variáveis de sistema, também chamadas de TAGs, tanto para o controlador quanto para a interface homem-máquina.

**Tabela 7 – Endereçamento e numeração das variáveis do sistema de Automação SIEMENS.**

| Área de Memória | Definição da Área | Mnemônico SIEMENS | Range de Endereços |
|---|---|---|---|
| Entrada | Input bit | I | 0.0 a 65,535.7 |
| | Input byte | IB | 0 a 65,535 |
| | Input word | IW | 0 a 65,534 |
| | Input double word | ID | 0 a 65,532 |
| Saída | Output bit | Q | 0.0 a 65,535.7 |
| | Output byte | QB | 0 a 65,535 |
| | Output word | QW | 0 a 65,534 |
| | Output double word | QD | 0 a 65,532 |
| Bit de memória interna | Memory bit | M | 0.0 a 255.7 |
| | Memory byte | MB | 0 a 255 |
| | Memory word | MW | 0 a 254 |
| | Memory double word | MD | 0 a 252 |
| I/O Periférico: Entrada externa | Peripheral input byte | PIB | 0 a 65,535 |
| | Peripheral input word | PIW | 0 a 65,534 |
| | Peripheral Inp. Doub. word | PID | 0 a 65,532 |
| I/O Periférico: Saída Externa | Peripheral output byte | PQB | 0 a 65,535 |
| | Peripheral output word | PQW | 0 a 65,534 |
| | Peripheral output double word | PQ | 0 a 65,53 |
| Temporizador | Timer (T) | T | 0 a 25 |
| Contador | Counter (C) | C | 0 a 25 |
| Bloco de Dados | Bloco de dados aberas com a instrução DB – (OPN) | | |
| | Data bit | DBX | 0.0 a 65,535.7 |
| | Data byte | DBB | 0 a 65,535 |
| | Data word | DBW | 0 a 65, 534 |
| | Data double word | DBD | 0 a 65,532 |
| | Bloco de dados aberas com a instrução DI – (OPN) | | |
| | Data bit | DIX | 0.0 a 65,535.7 |
| | Data byte | DIB | 0 a 65,535 |
| | Data word | DIW | 0 a 65, 534 |
| | Data double word | DID | 0 a 65,532 |
| Dados Locais | Temporary local data bit | L | 0.0 a 65,535.7 |
| | Temporary local data byte | LB | 0 a 65,535 |
| | Temporary local data word | LW | 0 a 65, 534 |
| | Temporary local data double word | L | 0 a 65,532 |

#### 4.3.6.1. Criando TAGs no controlador (PLC Tags)

A criação de TAGs no controlador possibilita a programação da lógica de processo. Esta lógica é implementada no controlador na forma de linguagem LADDER. Cada variável utilizada na lógica LADDER deve ser, obrigatoriamente, associada a um TAG na memória deste controlador.

Caso uma variável que o programador quiser utilizar não tiver sido criada, esta não poderá ser utilizada no código LADDER. Isto ocorre pois todas as vezes que o usuário iniciar o processo de transferência da lógica programada no software SIEMENS para o controlador, o software executa um procedimento de compilação. Este procedimento de compilação verifica se cada variável do LADDER está associada a um TAG do controlador, e caso isto não ocorra, a transferência do código LADDER para o CLP é interrompida.

Uma das considerações importantes para a declaração de TAGs é saber a quantidade de endereços físicos e de endereços de memória que podem ser utilizados. Este entendimento informa ao usuário o número máximo de variáveis que este poderá utilizar em sua lógica LADDER.

A quantidade de endereços de memória varia de acordo com o modelo da CPU utilizada. A CPU 1214C, utilizada neste trabalho, possui 50KB de memória interna alocada que permite a criação de um determinado número de variáveis de memória. O total de variáveis de memória utilizado independe do tipo de variável, estando somente limitado à memória interna disponível na CPU.

A quantidade de endereços físicos varia de acordo com o modelo da CPU e com a quantidade de módulos externos utilizados. A CPU 1214C, apresentada neste trabalho, possui 14 Entradas Digitais de 24VDC (corrente contínua), 10 saídas tipo relé (corrente contínua ou alternada, de acordo com a configuração dos canais), e 2 canais de entradas analógicas variando em uma faixa de 0 e 10V. Temos ainda um módulo externo com 2 saídas analógicas variando em uma faixa de 0 a 10V, ou de 0 a 20mA, dependendo da forma com que o módulo for configurado no software.

Ao contrário dos endereços de memória, os tipos de variáveis correspondentes a endereços físicos são limitados pelo número de saídas e entradas, digitais e analógicas, ligadas direta ou indiretamente ao controlador. Os tipos de variáveis correspondentes a endereços físicos, bem como seus endereços mnemônicos, podem ser visualizados quando acessamos as configurações do controlador através do software SIEMENS.

Para visualizar estas informações, devemos acessar a opção "*Device Configuration"*, no menu lateral esquerdo, abaixo do equipamento *PLC_1*. Esta operação pode ser visualizada na figura 42.

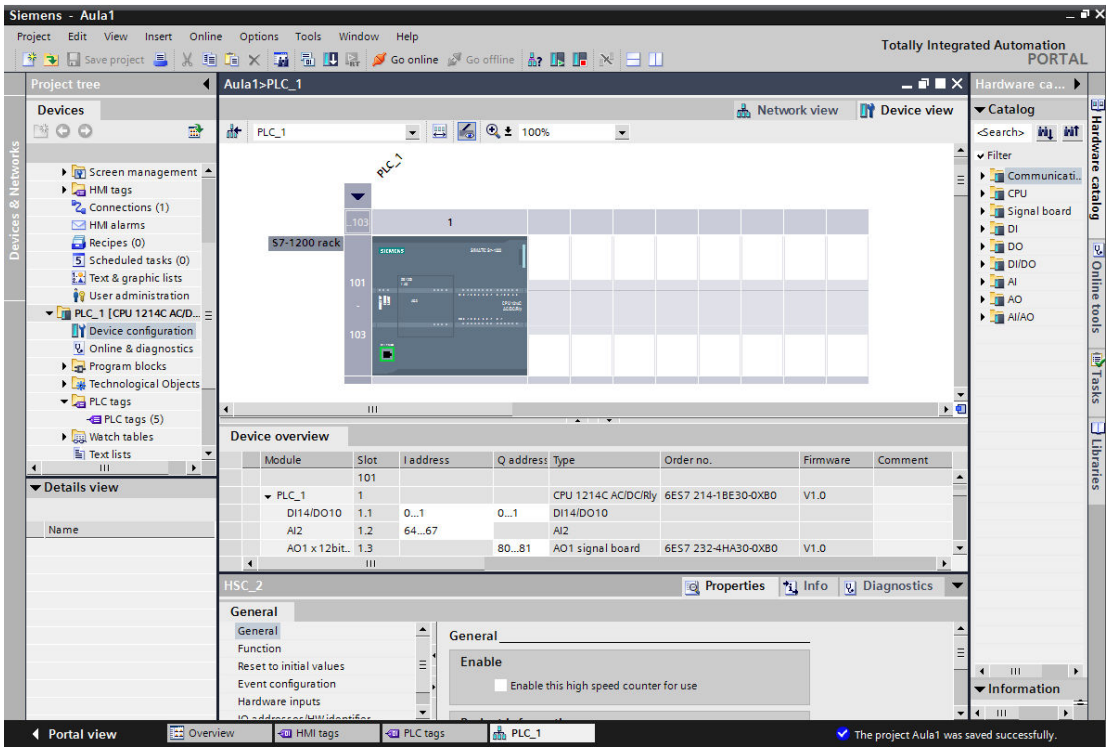



**Figura 42 – Tela de Configuração de Equipamento do controlador SIEMENS.**

Note que na aba *Device overview,* temos a faixa de endereços de entrada (*I address)* e a faixa de endereços de saída *(Q address)* para cada entrada e saída, digital e analógica, configuradas no controlador.

Para verificar mais detalhadamente o tipo de endereçamento mnemônico utilizado para cada módulo de entrada ou saída, devemos clicar no módulo que se deseja visualizar e, em seguida, poderemos verificar abaixo o endereço mnemônico que deverá ser utilizado para cada canal de cada módulo. Este procedimento pode ser visualizado na figura 43, onde verificamos que o endereço mnemônico utilizado para o canal 0 (Channel 0) do módulo interno de entradas digitais do controlador corresponde a *"I0.0"*.

O mesmo procedimento pode ser utilizado de forma análoga, para identificar os endereços mnemônicos utilizados para os demais módulos de entrada e saída do controlador.

Após identificarmos os endereços físicos e de memória, bem como seus endereços mnemônicos correspondentes que podem ser utilizados na programação da lógica LADDER, podemos iniciar o processo de criação de TAGs do controlador.

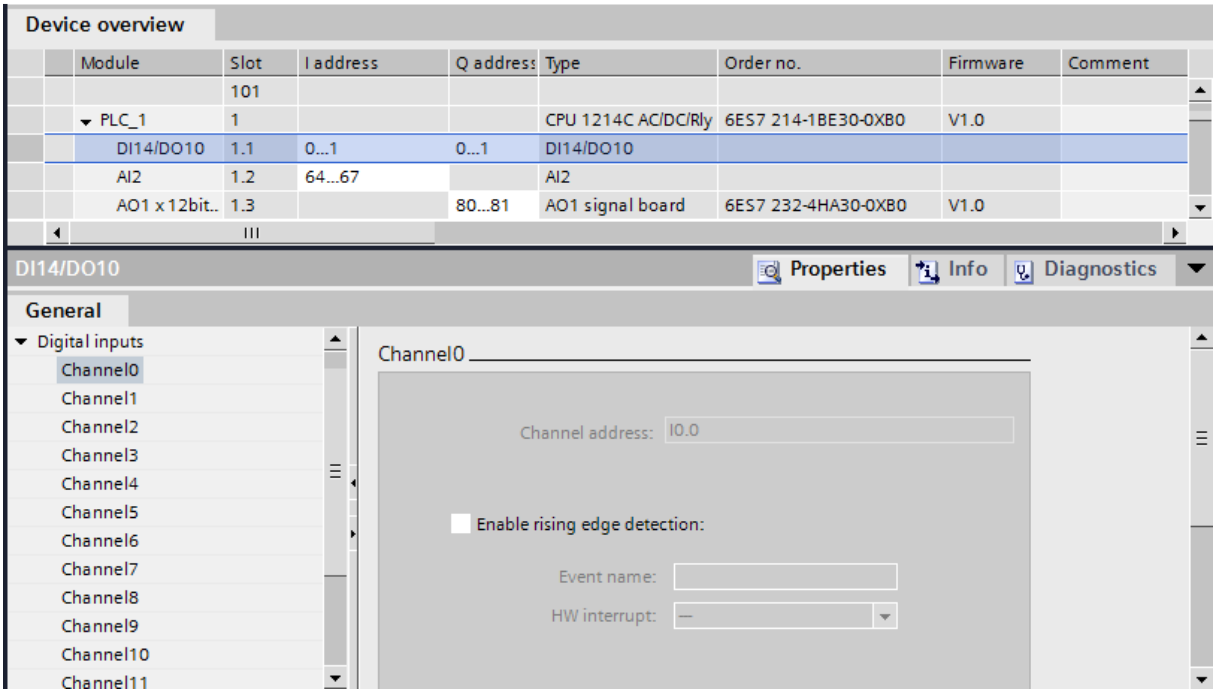



**Figura 43 – Visualização das características dos canais do módulo interno de entradas e saídas analógicas do CLP SIEMENS.**

Para criar um TAG do controlador, devemos acessar a opção *PLC Tags*, na aba do controlador *PLC_1* e, em seguida, clicar sobre o subgrupo *PLC Tags*. O nome do subgrupo pode ser alterado, assim como novos subgrupos de variáveis podem ser adicionados.

Nesta nova janela cada linha corresponde a um TAG (ou variável). Assim, devemos preencher as colunas de acordo com o tipo de variável que queremos criar, especificando o seu tipo (*Data Type)*, o seu endereço mnemônico *(Address)*, e adicionando um comentário. O campo *RETAIN* corresponde a variáveis cujos valores deverão ser armazenados mediante um desligamento repentino da fonte de alimentação da CPU, ou da mudança de estado da CPU de *STOP* para *RUN*.

Este procedimento de criação de TAGs, bem como o preenchimento da tabela de variáveis, pode ser visualizado na figura 44.

Na figura 44, podemos ainda ver outros tipos de variáveis que foram criadas para ilustrar a dinâmica da declaração de variáveis e a atribuição de seus endereços mnemônicos.

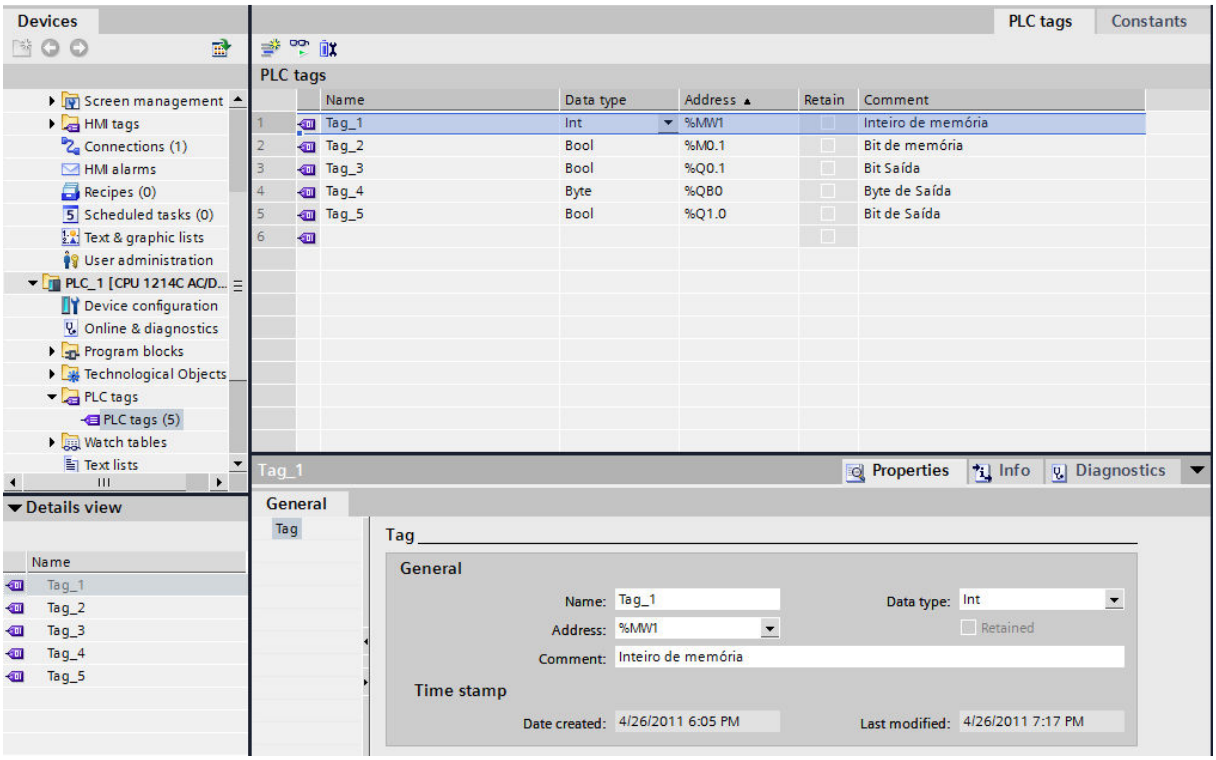



**Figura 44 – Tabela de TAGs do CLP Siemens.**

#### 4.3.6.2. Criando TAGs na IHM (HMI Tags)

A criação de TAGs na IHM nada mais é do que a criação das variáveis que serão utilizadas nas telas da IHM. Essas variáveis deverão ser criadas de forma análoga à criação de TAGs no controlador.

A forma com que os tipos de TAG são associados a endereços de memória é a mesma aplicada na criação de TAGs no controlador. Entretanto, a associação dos endereços físicos aos TAGs deverá ser feita de forma diferente, uma vez que a IHM não possui módulos de entrada e saída de dados.

Caso seja necessária a utilização de endereços físicos do controlador nas telas da IHM, os TAGs criados na IHM devem ser associados aos TAGs que correspondem aos endereços físicos do CLP. Esta associação deve ser feita no momento da criação do TAG na IHM.

Para criar um TAG da IHM devemos acessar a opção *HMI Tags*, na aba da IHM *HMI_1* e, em seguida, clicar sobre o subgrupo *HMI Tags*. O nome do subgrupo pode ser alterado, assim como novos subgrupos de variáveis podem ser adicionados.

Nesta nova janela cada linha corresponde a um TAG (ou variável). Assim, devemos preencher as colunas de acordo com o tipo de variável que queremos criar, especificando o seu nome *(Name)*, conexão *(Connection)*, tipo (*Data Type)*, associação à um TAG do CLP *(PLC Tag)*, o seu endereço mnemônico *(Address)*, o seu número de array *(Array Element)*, o seu ciclo de aquisição *(Acquisition cycle)*, e adicionando um comentário a respeito da variável (*Comment)*.

As especificações de cada variável (colunas), bem como o procedimento de criação de TAGs na IHM, podem ser vistas na figura 45.

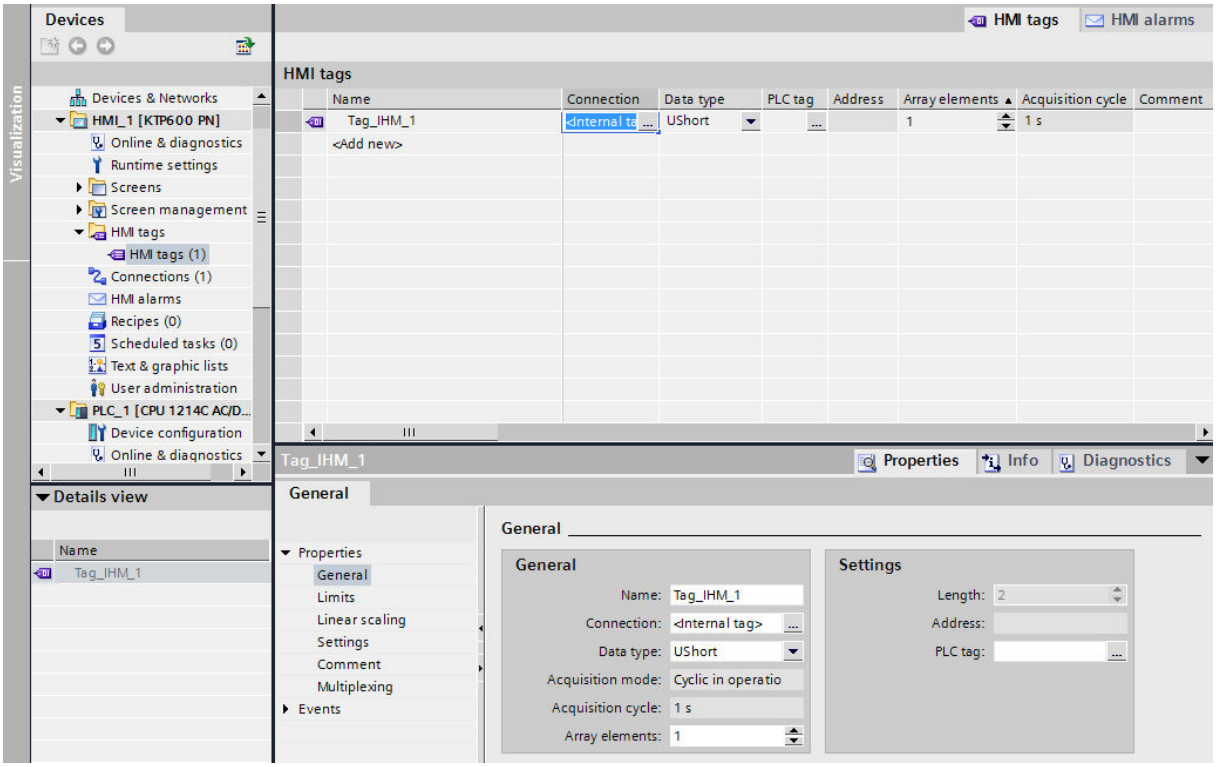



**Figura 45 – Tabela de TAGs da IHM SIEMENS.**

A coluna *"Connection"* permite a associação do TAG criado na IHM à um TAG criado no controlador. Ao acessar as opções desta coluna, caso a associação seja feita para este TAG, a conexão *HMI_connection_1* deverá ser selecionada. Caso o usuário não queira associar o TAG

criado a nenhum TAG do controlador, a opção *<internal tag>* deverá ser selecionada. Esta configuração pode ser visualizada na figura 46.

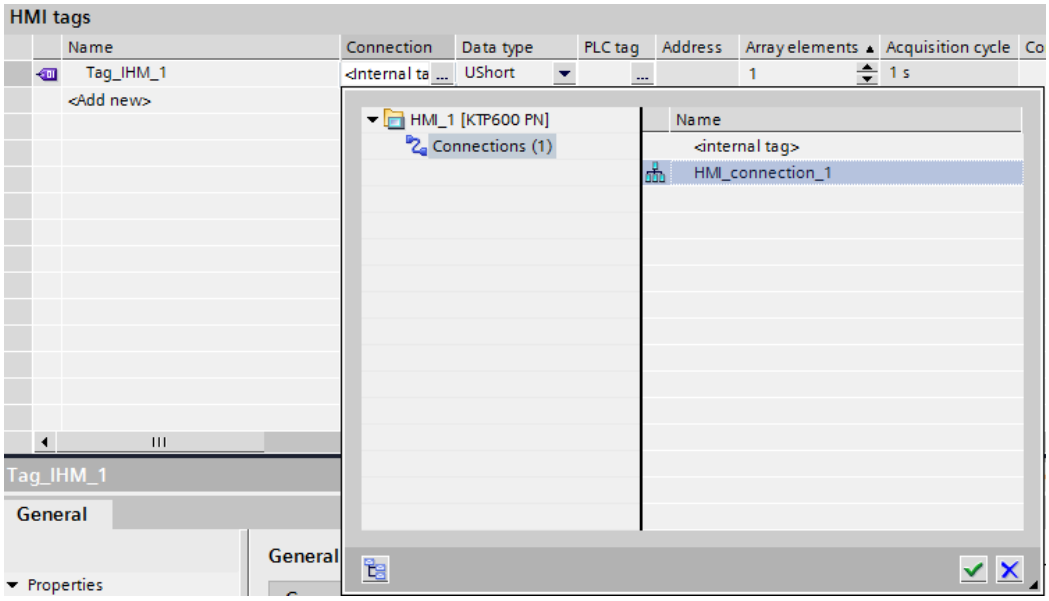



**Figura 46 – Associação de um TAG do controlador a um TAG da IHM.**

Caso a conexão *HMI_connection_1* seja selecionada na coluna *Connection*, o TAG criado na IHM deverá ser associado a um TAG do controlador através das opções da coluna *PLC tag.* Assim, ao clicarmos nas opções da coluna *PLC tags*, devemos selecionar o TAG do controlador, que desejamos associar à variável da IHM. Este procedimento pode ser visualizado na figura 47.

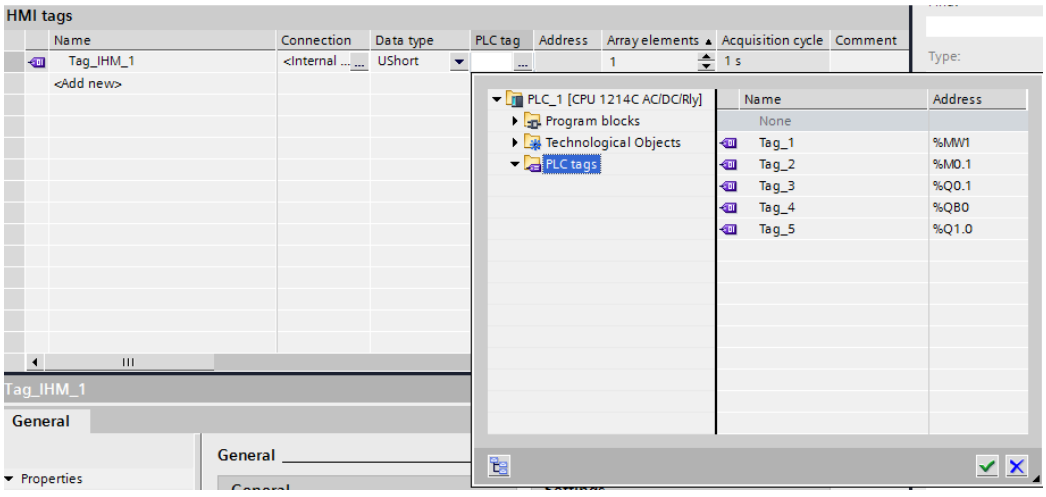



**Figura 47 - Selecionando o TAG do CLP ao qual o TAG da IHM será associado.**

A coluna “*Array Elements*” corresponde à dimensão do vetor de variáveis do mesmo nome e tipo, associado ao TAG. Caso esta coluna seja preenchida com o número 1 somente um TAG será criado.

A coluna “*Acquisition Cycle*” corresponde ao tempo de atualização da variável criada. Isto significa que o tempo atribuído a esta coluna será o tempo correspondente ao ciclo de atualização do valor desta variável na tabela de variáveis. Esta opção é interessante para sistemas de automação cujas variáveis possuam tempos de atualização diferentes.

### 4.3.7.Iniciando Programação do CLP

O CLP S7-1200 SIEMENS pode ser programado nas 5 linguagens-padrão utilizadas por controladores lógicos programáveis. Neste trabalho a linguagem utilizada será o diagrama Ladder.

Para programarmos o CLP utilizamos a mesma plataforma de software utilizada para programarmos a IHM e suas telas. Como indicado na figura 48, para iniciarmos a programação do CLP devemos primeiro selecionar o bloco de programação, para isso clicamos em *“PLC_1”* no menu situado à esquerda do software e, em seguida, clicamos em *“Program Blocks”*.

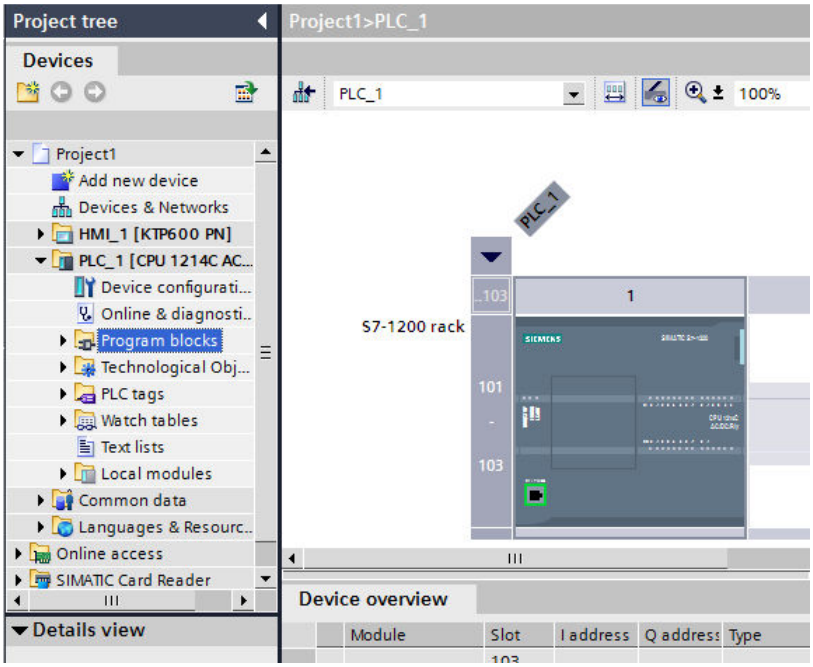



**Figura 48 - Opção *Program Blocks*, utilizada para selecionar o bloco de programação do CLP.**

Ao clicarmos na opção *"Program Blocks"* devemos selecionar "Main Block", o bloco de programação principal, ou podemos ainda clicar em *"Add new block"* para criarmos um novo bloco. As opções de blocos de programação podem ser visualizadas na figura 49.

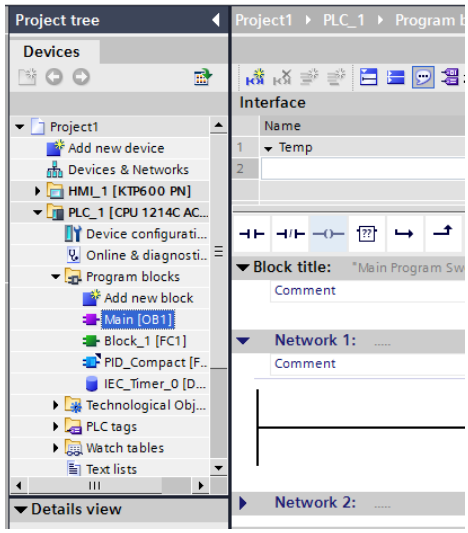



**Figura 49 – Blocos de programação dentro de *Program Blocks.***

Para programarmos o CLP devemos acessar as linhas de código, chamadas de *rungs* situadas dentro dos blocos de programação. No caso da figura 50, podemos visualizar uma *rung* situada dentro do bloco de programação *Main*.

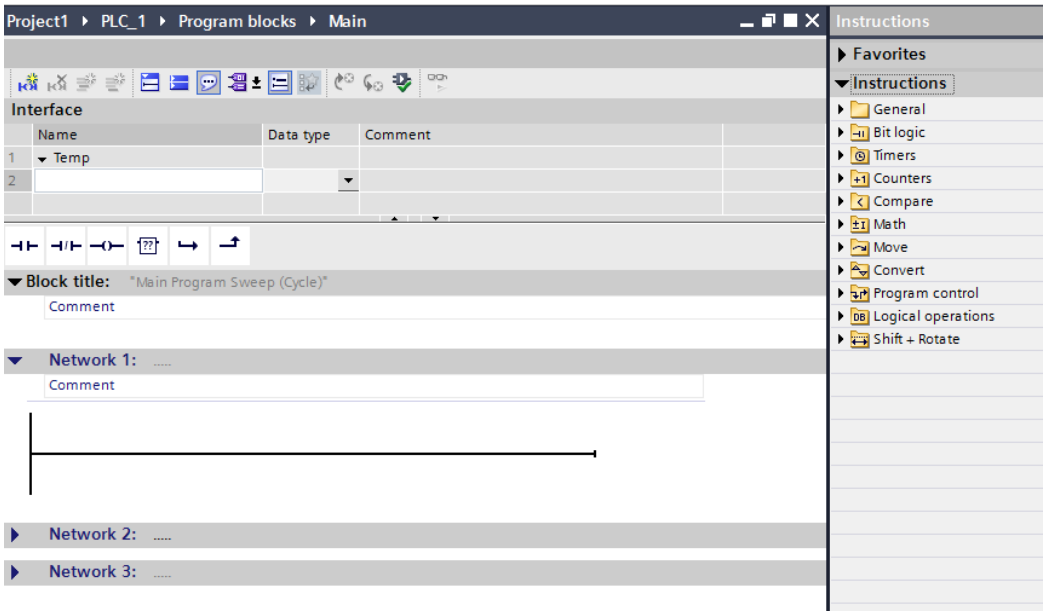



**Figura 50 – *Rung* de programação dentro do bloco *Main.***

O diagrama Ladder, como apresentado na subseção 3.2, é constituído de um conjunto de instruções através das quais a lógica do sistema é representada. As *rungs* de um bloco de programação podem ser editadas, e nestas são inseridas as instruções que combinadas definirão a lógica do controlador.

O CLP S7-1200 da SIEMENS possui instruções dos tipos: gerais de programação Ladder, variáveis binárias, temporizadores, contadores, blocos comparadores, bloco de operações matemáticas, blocos de mudança de dados, blocos de conversão de dados, blocos de controle do programa, blocos de operações lógicas, e blocos shift. Ao abrirmos uma *rung,* as instruções podem ser visualizadas à direita da janela do software de programação.

Na figura 51 são apresentadas as instruções de operações lógicas com bits. Ao clicarmos duas vezes em uma das instruções esta é inserida automaticamente na *rung* selecionada.

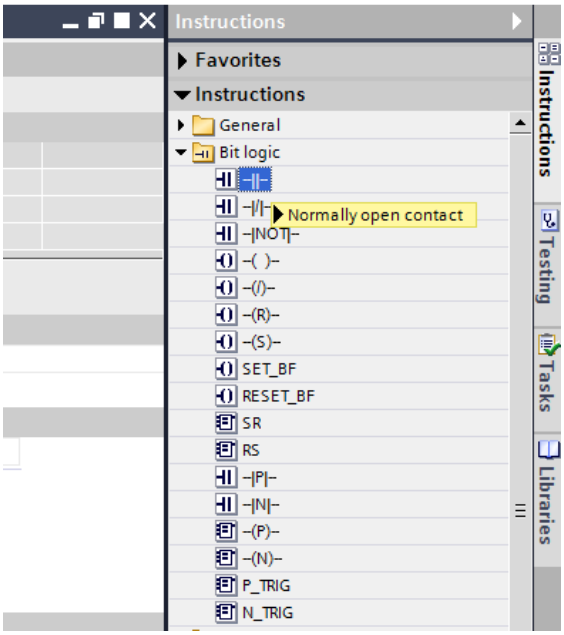



**Figura 51 – Conjunto de Instruções de operações lógicas com bits.**

### 4.3.8.Iniciando Programação da IHM

A IHM KTP600 também deve ser programada através do software de programação utilizado para a programação do CLP. A IHM funciona como um sistema operacional, desempenhando o papel da interface entre as variáveis do controlador e o operador, apresentando ao operador o processo de

forma amigável. A IHM possui acesso às variáveis do CLP através da rede Profibus, cuja configuração foi apresentada na subseção 4.3.3.

A IHM é programada através da criação de telas, onde o programador pode criar objetos e associá-los às variáveis de processo. Assim, quando alguma modificação ocorrer nas variáveis de processo do controlador, esta poderá ser visualizada na tela da IHM. Para iniciar a programação de uma tela da IHM, devemos primeiramente criá-la e, em seguida, editá-la. Uma tela pode ser criada ao clicarmos no ícone HMI_1 na esquerda do software de programação e, em seguida, em "*Add new screen*". Na figura 52 podemos visualizar o menu de criação de telas no software de programação, bem como a conexão profibus, que permite o compartilhamento de variáveis entre a IHM e o CLP.

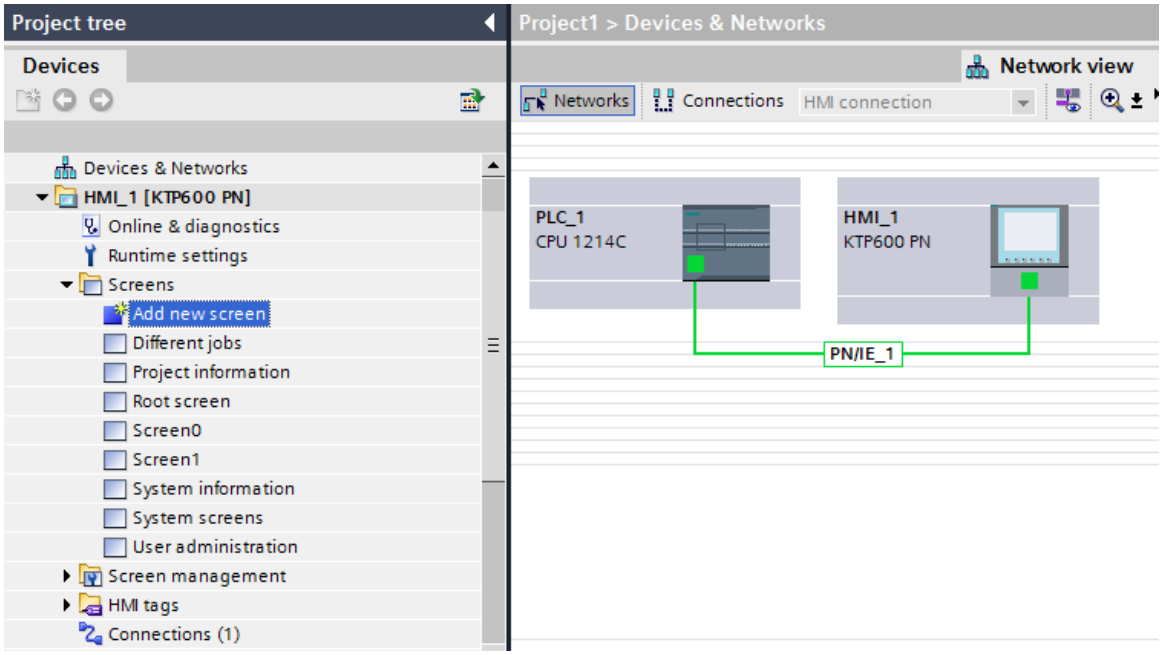



**Figura 52 - Menu responsável pela criação de telas da IHM e conexão da rede profibus.**

Após a criação de uma tela da IHM, devemos programá-la. A programação da IHM consiste na criação de objetos que podem ser associados às variáveis do CLP. Os objetos são estruturas computacionais gráficas que representarão o processo, logo podem ser desde simples figuras geométricas até gráficos de variáveis.

Após criarmos uma nova tela, um menu de objetos aparece à direita da janela do software de programação. Nessa janela encontramos desde objetos básicos até outros mais avançados. Para

adicionarmos esses objetos nas telas devemos clicar no objeto desejado e em seguida inseri-los na tela. Na figura 53 podemos visualizar o menu de objetos à direita da janela do software.

Para ilustrar a inserção de um objeto na tela, a partir da tela observada na figura 53, na aba "*Elements*", podemos visualizar a inserção de um objeto gráfico que representa uma figura do sistema. Foi selecionado um objeto gráfico qualquer, representando uma figura do sistema, e em seguida este foi inserido na tela. Como este objeto representa uma figura do sistema, podemos associá-lo a qualquer figura presente no computador onde o software de programação está instalado.

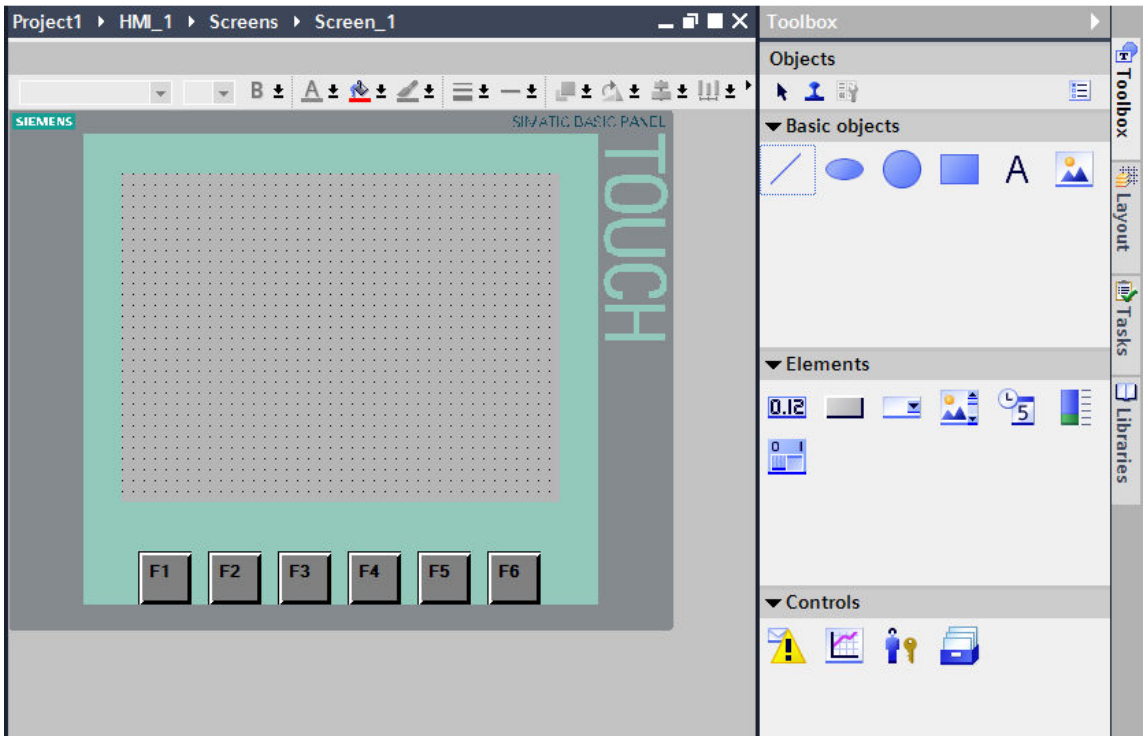



**Figura 53 - Menu de Objetos à direita do software de programação.**

Além de objetos representando variáveis do controlador, para que o operador navegue entre as telas da IHM, é preciso que sejam criados botões de navegação. Esses objetos são encontrados na aba "*Elements*" no menu à direita da janela do software. Na figura 55 podemos visualizar o ícone de criação de botões na aba *"Elements"*.

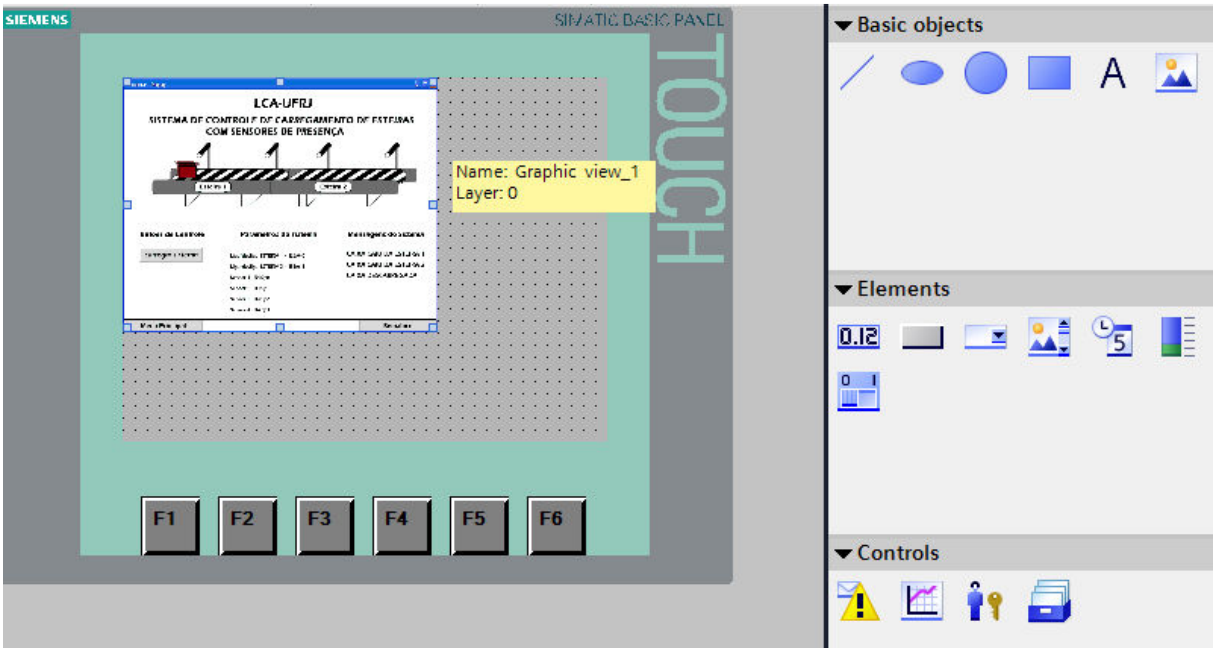



**Figura 54 - Inserção de um objeto associado à uma figura do sistema.**

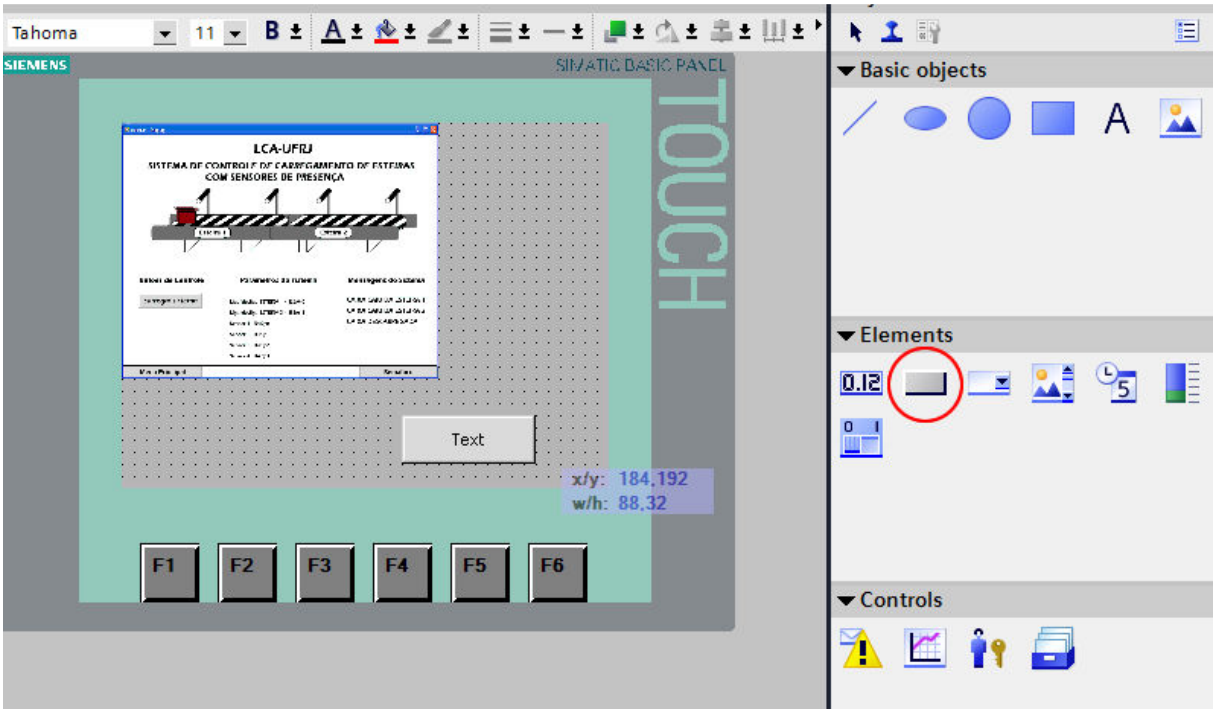



**Figura 55 - Inserção de um botão de navegação na tela da IHM.**

A cada botão é associado um evento. São exemplos de eventos: o clicar do botão direito do mouse, o clicar do botão esquerdo do mouse, o soltar do botão de scroll, etc. Ao evento associado ao

botão também deve ser associada uma ação. Esta ação acontece cada vez que o evento ocorre. Assim, podemos associar, por exemplo, a mudança de tela a um evento de clicar do botão esquerdo do mouse.

Na figura 56 podemos visualizar a atribuição da ação *ativar tela* a um evento associado ao botão criado. Ao clicarmos no botão criado, seremos transferidos para a tela correspondente da IHM.

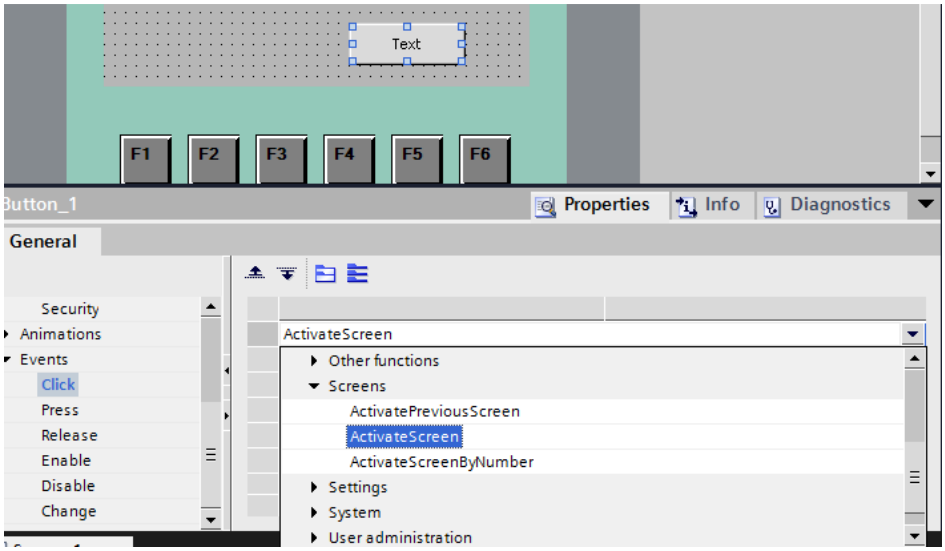



**Figura 56 – Atribuição da ação *Activate Screen* ao evento *Clicar* do mouse.**

### 4.3.9. Download do programa do CLP e da IHM

Nas duas últimas subseções, 4.3.7 e 4.3.8, a programação do CLP e da IHM foi introduzida. A partir das idéias apresentadas em cada uma dessas subseções, o programador pode iniciar a programação dos equipamentos, podendo aprofundar a complexidade da programação caso seja necessário para o processo.

Após terminar a programação do CLP e da IHM através do software de programação, é necessário enviar estas informações para a memória dos respectivos equipamentos. Como em automação o objeto principal do controle do processo é o controlador, o procedimento de envio da programação é chamado de *download*. Essa terminologia é utilizada também para a IHM, por se tratar da interface de controle entre o operador e o processo.

Para enviarmos as configurações do CLP e da IHM, bem como a programação de cada um desses, devemos acessar o menu principal do programa clicando em *"Online"* e, em seguida selecionar *"Download to device"*. Em seguida o download será iniciado e toda a lógica de configuração e de programação será transferida para a memória do CLP e da IHM.

Na figura 57 podemos visualizar o acesso ao menu responsável pela inicialização do download. O download pode ainda ser iniciado ao pressionarmos as teclas de atalho CTRL+L.

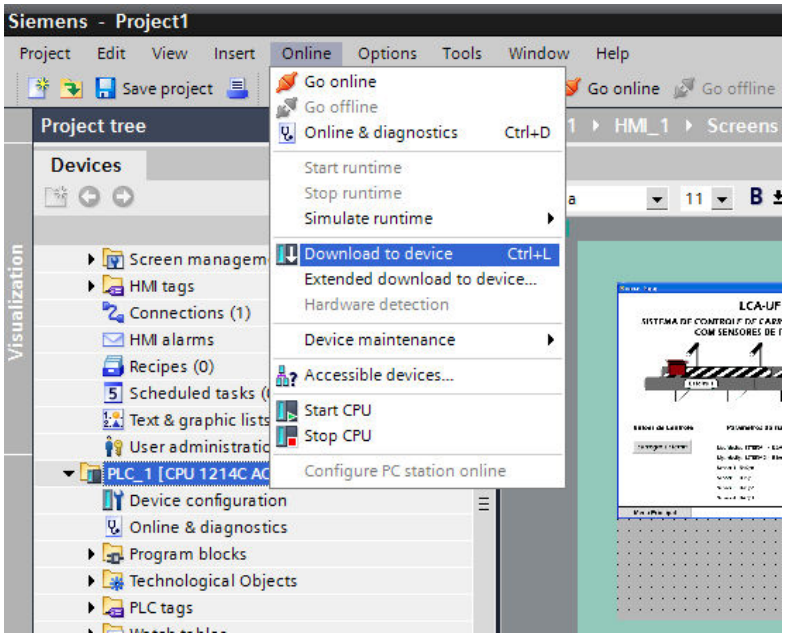



**Figura 57 – Menu responsável pelo download da configuração e da lógica do CLP e da IHM.**

## 4.4. Conclusão

Neste capítulo, em formato de tutorial, foi apresentado o software *Totally Integrated Automation Portal*, ferramenta de programação e configuração do CLP S7-1200 e da IHM KTP600. Esses equipamentos estão presentes no Laboratório de Controle e Automação da UFRJ.

Vimos também que através da correta configuração e programação dos equipamentos de automação podemos integrar sistemas de processos industriais de forma a controlar o seu funcionamento e criar uma interface entre o sistema e o operador.

Também foram apresentadas a filosofia de integração de sistemas de automação da plataforma SIEMENS, a configuração de uma rede profibus utilizando meio físico Ethernet, e a programação dos equipamentos de automação interconectados através dessa rede.

Esta ferramenta será utilizada na implementação do sistema integrado de automação da célula de manufatura apresentada no capítulo 5. Esta célula de manufatura é controlada pelo CLP S7-1200 integrado com a IHM KTP600 através de uma rede profibus.

# 5 SISTEMA DE MANUFATURA

Neste capítulo é apresentado um sistema de manufatura cuja automação é implementada através do CLP S7-1200 e da IHM KTP600 PN, ligados através de uma rede de automação profibus.

## 5.1. Definições do sistema

O sistema consiste em uma linha de manufatura de um determinado produto, cuja especificação não é o objetivo deste trabalho. O sistema de manufatura é composto por cinco componentes principais: duas máquinas de manufatura, dois braços robóticos, e um espaço de trabalho comum. O sistema, bem como os seus componentes, é apresentado na figura 58.

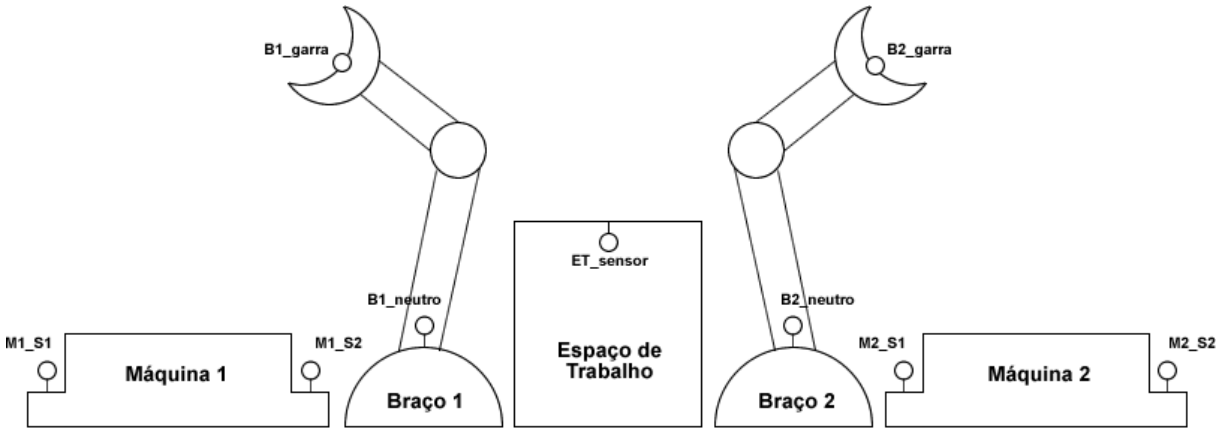



**Figura 58 – Sistema de Manufatura.**

As duas máquinas de manufatura são idênticas quanto à lógica de funcionamento, ou seja, possuem o mesmo número de variáveis associadas, porém, podem apresentar diferentes tempos de processamento segundo a sua programação interna. Quanto à lógica de funcionamento, sabe-se que as máquinas iniciam o processamento da peça mediante a recepção de um comando para iniciar o processamento. Finalizado o processamento, a máquina libera automaticamente a peça processada na sua saída e não inicia um novo ciclo de processamento até que a peça liberada seja retirada.

As máquinas possuem um sensor na entrada para indicar a presença de uma peça, um sensor na saída também para indicar a presença de uma peça, e emite um sinal que indica que o processamento foi finalizado. Além disso, as máquinas também recebem uma variável externa de

comando para iniciar o processamento. As variáveis associadas às máquinas podem ser visualizadas na tabela 8.

**Tabela 8 – Variáveis associadas às Máquinas do sistema de manufatura.**

| VARIÁVEL | MÁQUINA | DESCRIÇÃO |
|---|---|---|
| M1_f | 1 | Processamento finalizado na máquina 1 |
| M1_S1 | 1 | Peça presente na entrada da máquina 1 |
| M1_S2 | 1 | Peça presente na saída da máquina 1 |
| M1_cmd_iniciar | 1 | Comando para iniciar processamento da máquina 1 |
| M2_f | 2 | Processamento finalizado na máquina 2 |
| M2_S1 | 2 | Peça presente na entrada da máquina 2 |
| M2_S2 | 2 | Peça presente na saída da máquina 2 |
| M2_cmd_iniciar | 2 | Comando para iniciar processamento da máquina 2 |

Os braços robóticos são idênticos e trabalham com base em um conjunto de variáveis de entrada (comando) e saída (sensores), porém, também podem apresentar diferentes tempos de resposta para os comandos segundo as suas programações internas. Os braços robóticos possuem três posições possíveis: pegando ou soltando peça na máquina, neutra, e pegando ou soltando peça no espaço de trabalho. O braços possuem ainda um sensor que indica a presença de uma peça em sua garra e um sensor que indica se o mesmo está na posição neutra.

Os braços também recebem comandos para pegar peça e de soltar peça. Sabe-se ainda, que no sistema de manufatura da figura 58, o braço 1 é pré-programado internamente para que o comando de pegar peça o faça pegar uma peça na saída da máquina 1, e para que o comando de soltar peça o faça soltar uma peça no espaço de trabalho. O braço 2, por sua vez, é pré-programado internamente para que o comando de pegar peça o faça pegar uma peça no espaço de trabalho, e para que o comando de soltar peça o faça soltar uma peça na entrada da máquina 2. As variáveis associadas aos braços robóticos podem ser visualizadas na tabela 9.

**Tabela 9 – Variáveis associadas aos braços robóticos do sistema de manufatura.**

| VARIÁVEL | TIPO | DESCRIÇÃO |
|---|---|---|
| B1_neutro | booleana | Braço 1 em posição neutra |
| B1_garra | booleana | Braço 1 com peça |
| B1_cmd_pegar | booleana | Comando para Braço 1 pegar peça |
| B1_cmd_soltar | booleana | Comando para Braço 1 soltar peça |
| B2_neutro | booleana | Braço 2 em posição neutra |
| B2_garra | booleana | Braço 2 com peça |
| B2_cmd_pegar | booleana | Comando para Braço 2 pegar peça |
| B2_cmd_soltar | booleana | Comando para Braço 2 soltar peça |

O espaço de trabalho desempenha o papel de uma caixa com capacidade limitada em 3 peças, e pode ser ocupado por somente um braço robótico por vez, isto é, caso haja um braço robótico pegando ou soltando uma peça em seu interior o outro braço deverá aguardar até que este deixe o espaço de trabalho. O CLP armazena ainda, o número de peças presentes no espaço de trabalho na variável inteira “ET_qtde”.

O espaço de trabalho possui um sensor que indica a presença de um braço em seu interior. A única variável associada ao espaço de trabalho pode ser visualizada na tabela 10.

**Tabela 10 – Variáveis associadas ao espaço de trabalho.**

| VARIÁVEL | DESCRIÇÃO |
|---|---|
| ET_sensor | Braço presente no interior do espaço de trabalho |

## 5.2. Funcionamento do Sistema

A lógica de funcionamento do sistema de manufatura apresentado na figura 58 é feita com base nas variáveis de entrada e saída (comandos e sensores) de cada um dos seus componentes apresentados na subseção 5.1.

Quando o controlador percebe, através do sensor de presença na entrada da máquina 1, a presença de uma peça, este envia para a máquina 1 o comando de iniciar processamento. O processamento então é iniciado, e quando é finalizado, a máquina 1 libera a peça em sua saída.

Caso seja detectada uma peça pelo sensor na saída da máquina 1, e o braço 1 esteja na posição neutra e sem peça, o controlador envia para o braço 1 o comando para pegar peça. Ao pegar a peça na saída da máquina 1, o braço 1 volta com a peça para a posição neutra e aguarda o comando para soltar a peça no espaço de trabalho.

Caso o espaço de trabalho não tenha atingido o limite máximo de peças em seu interior, o controlador envia para o braço 1 o comando de soltar peça. Caso contrário, o braço 1 não recebe comando para soltar a peça até que o número de peças no interior do espaço de trabalho seja menor que o seu limite máximo.

O espaço de trabalho suporta um limite máximo de 3 peças, e comporta somente um braço robótico por vez em seu interior. Logo, um braço robótico só pode acessar o espaço de trabalho caso

este não possua nenhum braço em seu interior. A presença de um braço no espaço de trabalho é detectada por um sensor de presença.

No sistema de manufatura apresentado, os braços possuem diferentes prioridades de acesso ao espaço de trabalho. Caso o espaço de trabalho esteja livre e ambos os braços possam acessá-lo, o braço 1 tem prioridade sobre o braço 2.

O braço 2 funciona de maneira análoga ao braço 1, este porém, tem a função de pegar peças no espaço de trabalho e transportá-las para a entrada da máquina 2. Supondo que o braço 2 esteja na posição neutra e que o espaço de trabalho esteja livre e com peças em seu interior, o controlador envia para o braço 2 o comando de pegar peça. Ao pegar uma peça no espaço de trabalho, o braço 2 volta com a peça para a posição neutra e aguarda o comando para soltar a peça na entrada da máquina 2.

Ao receber o comando para soltar a peça na entrada da máquina 2, o braço 2 realiza a operação e retorna para a posição neutra sem peça. Após o recebimento do comando, o braço 2 só deposita a peça na máquina 2 caso a mesma esteja vazia. Caso a máquina 2 não esteja realizando nenhum processamento nem possua nenhuma peça em sua saída, esta inicia automaticamente o processamento da peça depositada em sua entrada, e ao finalizar, a peça é liberada na sua saída. Até que a peça na saída da máquina 2 seja retirada, esta não inicia outro ciclo de processamento.

### 5.3. Modelo em RPIC do sistema de manufatura

Antes de iniciar a programação da lógica de controle em diagrama Ladder no controlador lógico-programável, foi criado um modelo em rede de Petri que nos permite estudar o correto funcionamento do sistema de manufatura.

A RPIC segue o funcionamento do sistema descrito na subseção 5.2, e obedece as definições para redes de Petri interpretadas para controle da subseção 2.7. A rede de Petri interpretada para controle do sistema de manufatura é apresentada na figura 59.

Através da figura 59, é possível visualizar que a RIPC do sistema possui 17 lugares e 14 transições. As receptividades associadas às transições e as ações e estados associados aos lugares podem ser vistas nas tabelas 11 e 12, respectivamente.

**Tabela 11 – Receptividades associadas às transições.**

| TRANSIÇÃO | RECEPTIVIDADE | CONDIÇÃO | EVENTO |
|---|---|---|---|
| t1 | R1 | [not[(B1_neutro=1)*(B1_garra=1)]* (ET_qtde<3)*(ET_qtde>0)] + [Et_qtde=3] | e |
| t2 | R2 | [ET_qtde<3] | e |
| t3 | R3 | 1 | [B2_neutro]↑ |
| t4 | R4 | 1 | [B1_neutro]↑ |
| t5 | R5 | 1 | e |
| t6 | R6 | 1 | e |
| t7 | R7 | 1 | [B2_neutro]↑ |
| t8 | R8 | 1 | [B1_neutro]↑ |
| t9 | R9 | 1 | [M2_S2]↑ |
| t10 | R10 | 1 | [M1_S1]↑ |
| t11 | R11 | 1 | e |
| t12 | R12 | 1 | [M1_S2]↑ |
| t13 | R13 | 1 | e |
| t14 | R14 | 1 | e |

**Tabela 12 – Ações e estados associados aos lugares.**

| LUGAR | ESTADO | AÇÃO | OPERAÇÃO |
|---|---|---|---|
| p1 | Espaço de trabalho livre | ---- | ---- |
| p2 | Braço 1 - posição neutra com peça | ---- | ---- |
| p3 | Braço 2 - posição neutra sem peça | ---- | ---- |
| p4 | ---- | Comando braço 1 soltar peça | ---- |
| p5 | ---- | Comando braço 2 pegar peça | ---- |
| p6 | Braço 1 - posição neutra sem peça | ---- | ---- |
| p7 | Braço 2 - posição neutra com peça | ---- | ---- |
| p8 | ---- | Comando braço 1 pegar peça | ---- |
| p9 | ---- | Comando braço 2 soltar peça | ---- |
| p10 | Máquina 1 livre | ---- | ---- |
| p11 | ---- | Comando iniciar processamento máquina 2 | ---- |
| p12 | ---- | Comando iniciar processamento máquina 1 | ---- |
| p13 | Máquina 2 - processamento finalizado | ---- | ---- |
| p14 | Máquina 1 - processamento finalizado | ---- | ---- |
| p15 | Máquina 2 livre | ---- | ---- |
| p16 | ---- | ---- | ET_qtde=ET_qtde+1 |
| p17 | ---- | ---- | ET_qtde=ET_qtde-1 |

Ainda na RPIC do sistema de manufatura da figura 59, podemos encontrar um conflito efetivo entre as transições $t_1$ e $t_2$. Este conflito é resolvido através de condições mutuamente excludentes, que tornam $t_2$ prioritário em relação à $t_1$. O conflito poderia ainda ser resolvido utilizando-se RPIC com prioridade entre transições.

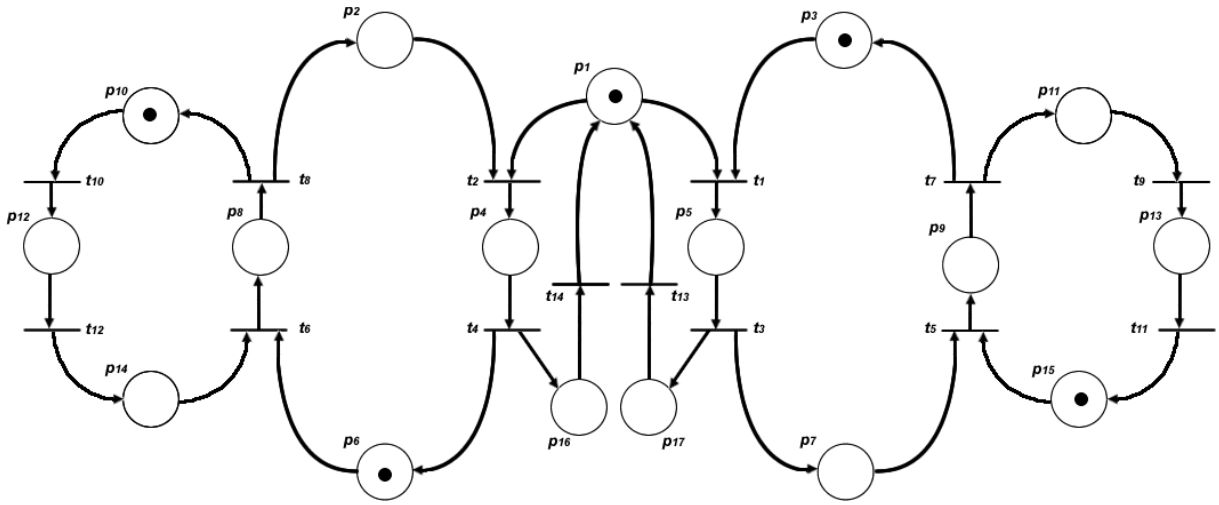



**Figura 59 – Rede de Petri interpretada para controle do sistema de manufatura.**

A rede de Petri da figura 59 foi simulada no software HPSim 1.1, e o resultado da simulação atendeu às expectativas para o funcionamento do projeto. Uma ilustração da simulação pode ser visualizada na figura 60.

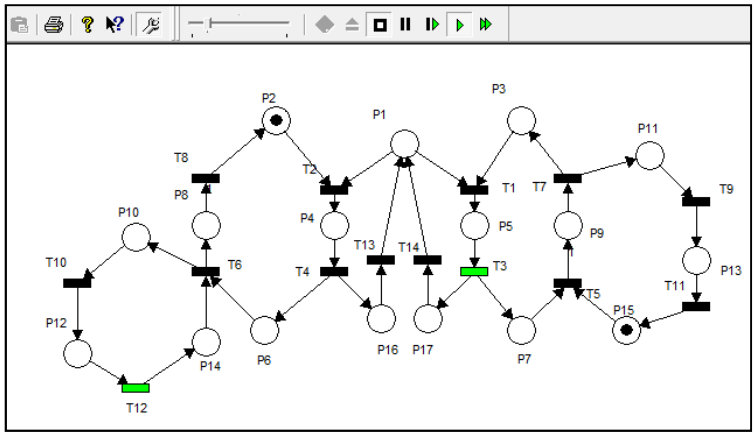



**Figura 60 – Rede de Petri interpretada para controle do sistema de manufatura simulada no programa HPSIM.**

### 5.4. Matrizes de Incidência

A rede de Petri do sistema de manufatura, assim como apresentado na subseção 2.5, pode ser representada através da sua matriz de incidência. Consequentemente, como a matriz de

incidência é formada por uma operação matemática entre as matrizes de incidência de entrada e incidência de saída, estas também podem ser representadas.

$$A = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & +1 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ +1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix}$$

$$A_{in} = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix}$$

$$A_{out} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & +1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

A matriz de incidência, a matriz de incidência de entrada, e a matriz de incidência de saída, do sistema representado na figura 59, podem ser visualizadas acima.

### 5.5. Conversão da RPIC do sistema de manufatura em diagrama Ladder

Para convertermos a RPIC do sistema em diagrama Ladder, será utilizado o método apresentado na subseção 3.2. Uma vez que o método de conversão proposto em Moreira, Botelho & Basilio (2009), a conversão da rede de Petri do sistema em diagrama Ladder consiste na criação dos 5 módulos do diagrama Ladder propostos.

O diagrama Ladder criado a partir da RPIC do sistema de manufatura encontra-se no apêndice A deste trabalho separado em módulo de inicialização, módulo de eventos, módulo de condições para o disparo das transições, módulo de dinâmica, e módulo das ações.

O diagrama Ladder do sistema de manufatura foi programado no CLP S7-1200 da SIEMENS, utilizando as instruções apresentadas na subseção 3.1. O programa foi testado e validado no CLP presente no Laboratório de Controle e Automação da Universidade Federal do Rio de Janeiro.

### 5.6. Programação do sistema na IHM KTP600 PN

O sistema de manufatura também foi programado na IHM KTP600 PN, apresentada na subseção 4.1, através do software *Totally Integrated Automation Portal V10 SIEMENS*.

A programação da IHM da célula de manufatura apresentada consiste na criação de um sistema operacional, representado por um conjunto de telas que representam a rede de Petri interpretada para controle do sistema. Através dessas telas o operador é capaz de saber o status atual do sistema através da visualização da sua RPIC.

A partir da tela inicial visualizada na figura 61, é possível acessar o menu de telas, onde podemos encontrar um link para as diferentes telas do sistema. O menu de telas pode ser visualizado na figura 62.

O botão REDE DE PETRI acessa a tela onde se encontra a rede de Petri do sistema de manufatura. Esta rede de Petri é dinâmica, no sentido que a marcação na tela acompanha online, através da rede profibus, a atual marcação do sistema.

A visibilidade das fichas presentes nos lugares depende das variáveis associadas a esses no controlador. Através da rede profibus, a IHM acessa a variável correspondente ao lugar na memória do CLP, e a associa ao objeto que representa a ficha na tela. Desta forma, caso a variável referente ao lugar possua valor lógico verdadeiro, a ficha torna-se visível no lugar correspondente, caso contrário a ficha desaparece.

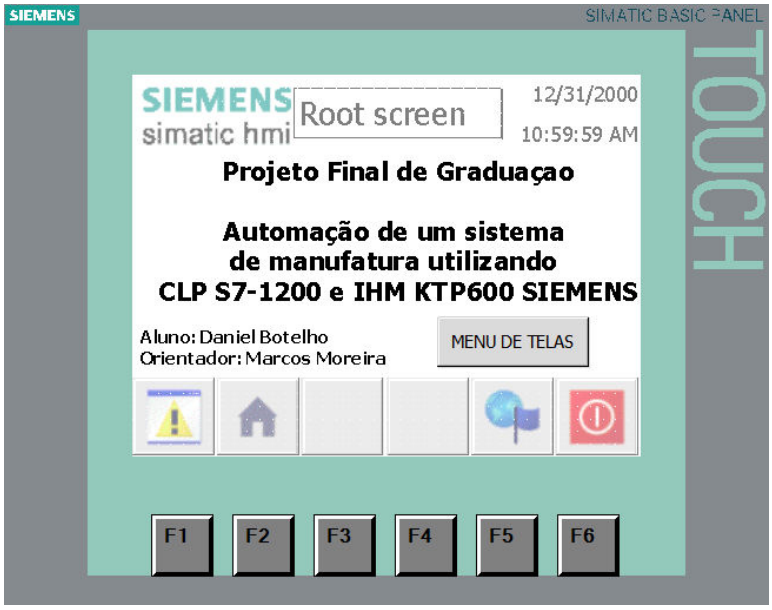



**Figura 61 – Tela inicial – Programa do sistema de manufatura.**

A tela da IHM onde é possível acessar a rede de Petri do sistema de manufatura, pode ser visualizada na figura 63.

Como a tela da IHM possui 6 polegadas e a descrição de todos os estados na própria tela da RPIC deixaria a mesma saturada, duas novas telas com as descrições de cada lugar foram criadas. Através dessas duas telas o operador pode saber a descrição do estado atual do sistema. O estado atual do sistema expresso nos lugares, representa a visão que o controlador possui do sistema, uma vez que a modelagem foi feita através de uma rede de Petri interpretada para controle.

As telas das descrições dos lugares podem ser visualizadas nas figuras 64 e 65, e a tela de acompanhamento do status do buffer pode ser visualizada na figura 66. Também podemos acessar a tela de informações, apresentada na figura 67, através do menu, onde são encontradas informações sobre o projeto de automação da célula de manufatura.

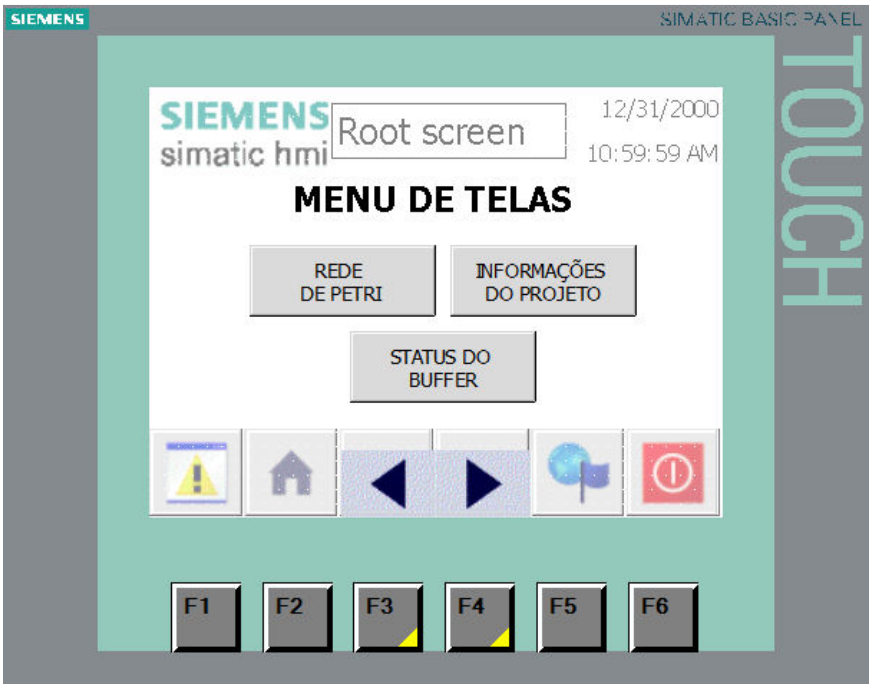



**Figura 62 – Menu de Telas – Programa do sistema de manufatura.**

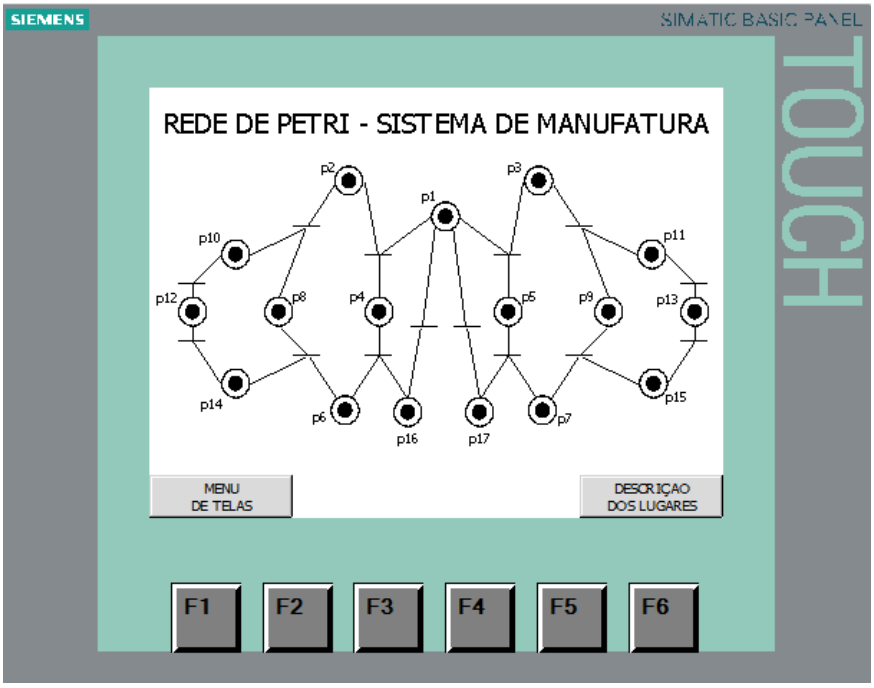



**Figura 63 - Tela da rede de Petri do sistema de manufatura.**

## 5.7. Conclusão

Neste capítulo foram apresentadas a modelagem e a integração do sistema de automação da célula de manufatura. A modelagem foi feita através de uma rede de Petri interpretada para controle, e a integração do sistema de automação através de um CLP Siemens S7-1200 e uma IHM KTP600, ambos apresentados na subseção 4.1. Os equipamentos são conectados utilizando uma rede Ethernet com protocolo de comunicação profibus.

A RPIC do sistema de manufatura foi traduzida para diagrama Ladder utilizando o método proposto neste trabalho, apresentado na subseção 3.2. Posteriormente, o diagrama Ladder foi implementado no CLP através do software *Totally Integrated Automation Portal*.

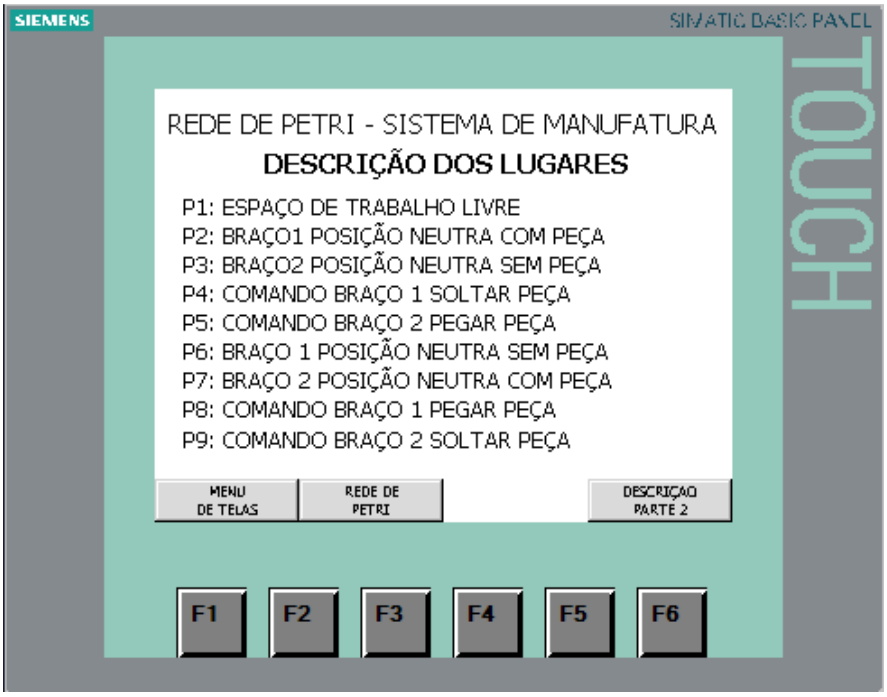



**Figura 64 – Tela de descrição dos lugares – parte 1**

A IHM do sistema também foi programada através do *software Totally Integrated Automation Portal*. A IHM acessa as variáveis do CLP via Ethernet e as utiliza para tornar as telas do processo de manufatura mais dinâmicas para o operador.

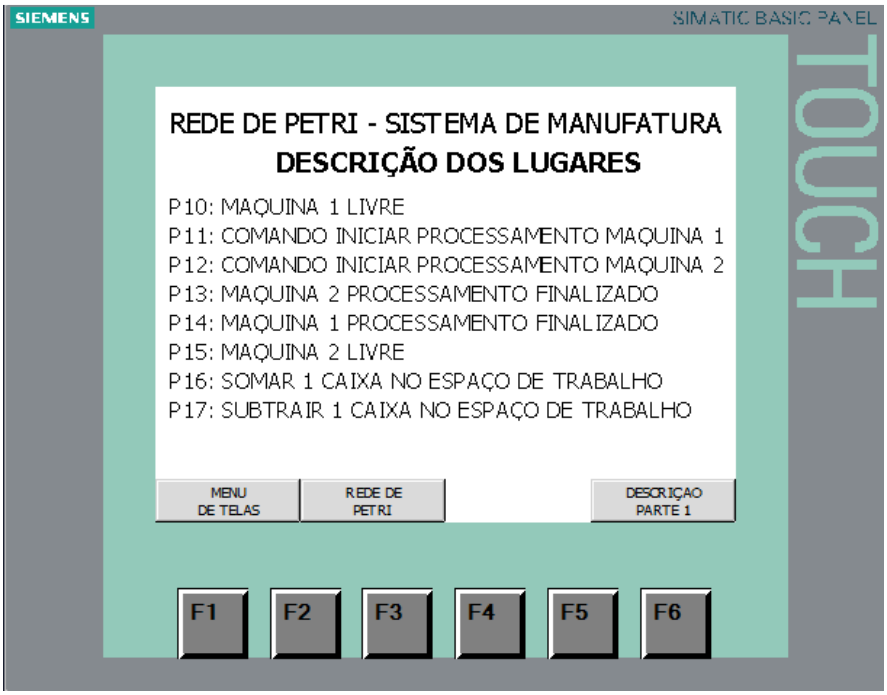



**Figura 65 – Tela de descrição dos lugares – parte 2.**


**Figura 66 - Tela de status do buffer.**

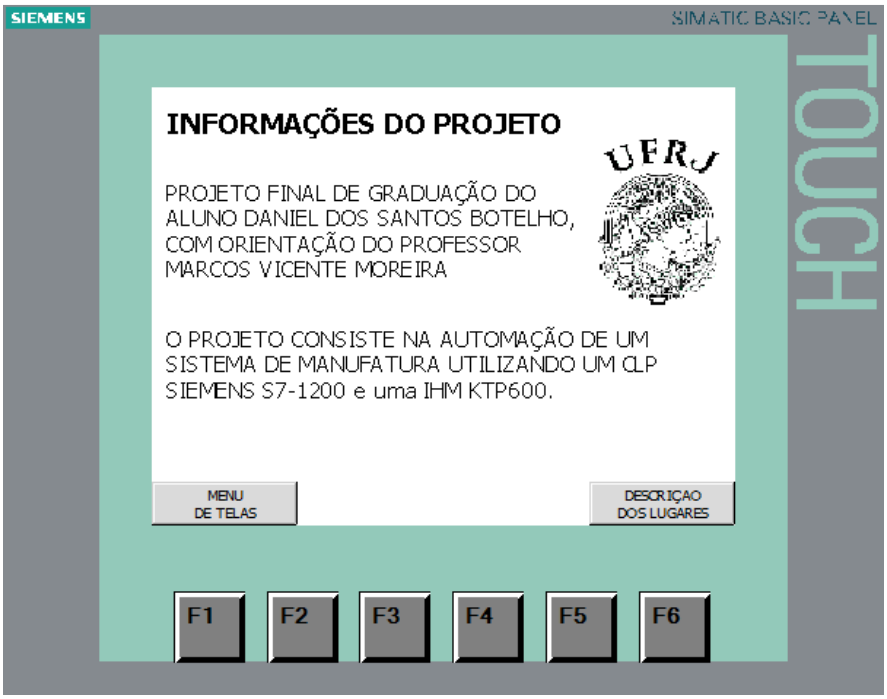



**Figura 67 - Tela de informações do projeto.**

Para analisar o correto funcionamento do programa do CLP e da IHM, foi feito, em um primeiro momento, o download dos programas para os equipamentos. Em seguida, com o computador conectado na rede profibus, as variáveis de processo contidas na memória do CLP e da IHM foram monitoradas em tempo real. As variáveis foram modificadas intencionalmente de forma a simular os eventos externos, e a mudança do estado do sistema foi acompanhada através da RPIC na tela da IHM.

O sistema integrado de automação da célula de manufatura foi implementado e simulado no Laboratório de Controle e Automação da Universidade Federal do Rio de Janeiro, e atendeu a todas as expectativas de funcionamento do projeto.

# 6 CONCLUSÃO

Este trabalho teve como objetivo a modelagem e a integração de um sistema de automação de uma célula de manufatura composta por duas máquinas, dois braços mecânicos, e um buffer com capacidade limitada, utilizando CLP Siemens S7-1200 e uma IHM KTP600. Os equipamentos se comunicam através de uma rede Ethernet, com protocolo de comunicação profibus.

Para a modelagem da lógica de controle do sistema de manufatura, foi utilizada a rede de Petri interpretada para controle. Foi feita também uma breve introdução aos conceitos básicos de redes de Petri, e às particularidades das redes de Petri interpretadas para controle.

Para traduzirmos o modelo em rede de Petri interpretada para controle em linguagem Ladder, para uma futura implementação no controlador lógico-programável, foi desenvolvido um método de conversão baseado em Moreira, Botelho & Basílio (2009).

A configuração da rede de automação utilizando meio físico Ethernet e protocolo profibus, a programação da IHM, e a programação do diagrama Ladder, foram feitas utilizando o software de integração *Totally Integrated Automation System*, da Siemens. O software foi apresentado, e a configuração da rede de automação, a programação do controlador, e a programação do aplicativo da IHM, foram demonstradas em formato de tutorial.

Finalmente, foi apresentada a modelagem em RPIC do controle da célula de manufatura, a sua tradução em diagrama Ladder através do método proposto, e o aplicativo da IHM. A modelagem em redes de Petri interpretadas para controle, a programação em Ladder, e a integração do CLP Siemens com a IHM, formam o sistema integrado de automação para o controle da célula de manufatura apresentada. O sistema foi simulado e mostrou ter o desempenho esperado.

## 6.1. Sugestões e perspectivas futuras

O sistema de automação pode ser integrado para que o mesmo seja operado local ou remotamente. Podem ser sugeridos como trabalhos futuros: (i) integração de um sistema de automação que possa ser operado remotamente; (ii) integração de um sistema de automação utilizando outro tipo de protocolo de comunicação; (iii) integração de dois diferentes controladores em um mesmo sistema de manufatura; (iv) diagnóstico de falhas do sistema de manufatura apresentado.

Assim, espera-se que este trabalho motive o desenvolvimento de novas tecnologias voltadas para a integração de sistemas de automação industrial, novas formas de aplicar a modelagem através de redes de Petri interpretadas para controle, e de novos projetos e soluções de automação para processos industriais.

## 7 BIBLIOGRAFIA

[1] Thomasset, D.; Les Systèmes à événements discrets: Approche par Réseaux de Petri et Grafcet, Institut National des Sciences Appliquées de Lyon, 2009/2010.

[2] Cassandras, C. G. e Lafortune, S. Introduction to Discrete Event System , 2a edição, Springer, 2008.

[3] Moreira, M. V. ; Botelho, D. S. ; Hazan, S. S. . Implementação de sistemas de automação descritos por Redes de Petri Interpretadas para Controle. In: 37 COBENGE, Recife. 37 Congresso Brasileiro de Educação em Engenharia, 2009.

[4] Moreira, M. V., Botelho, D. S e Basílio, J. C.; Ladder diagram implementation of Control Interpreted Petri Nets: a state equation approach 4th IFAC Workshop on Discrete-Event System Design, Valencia, Espanha, pp. 85 - 90 , 2009.

[5] Fabian, M. & Hellgren, A.; PLC-based Implementation of Supervisory Control for Discrete Event Systems; Proceedings of the 37th IEEE, Conference on Decision & Control, 1998

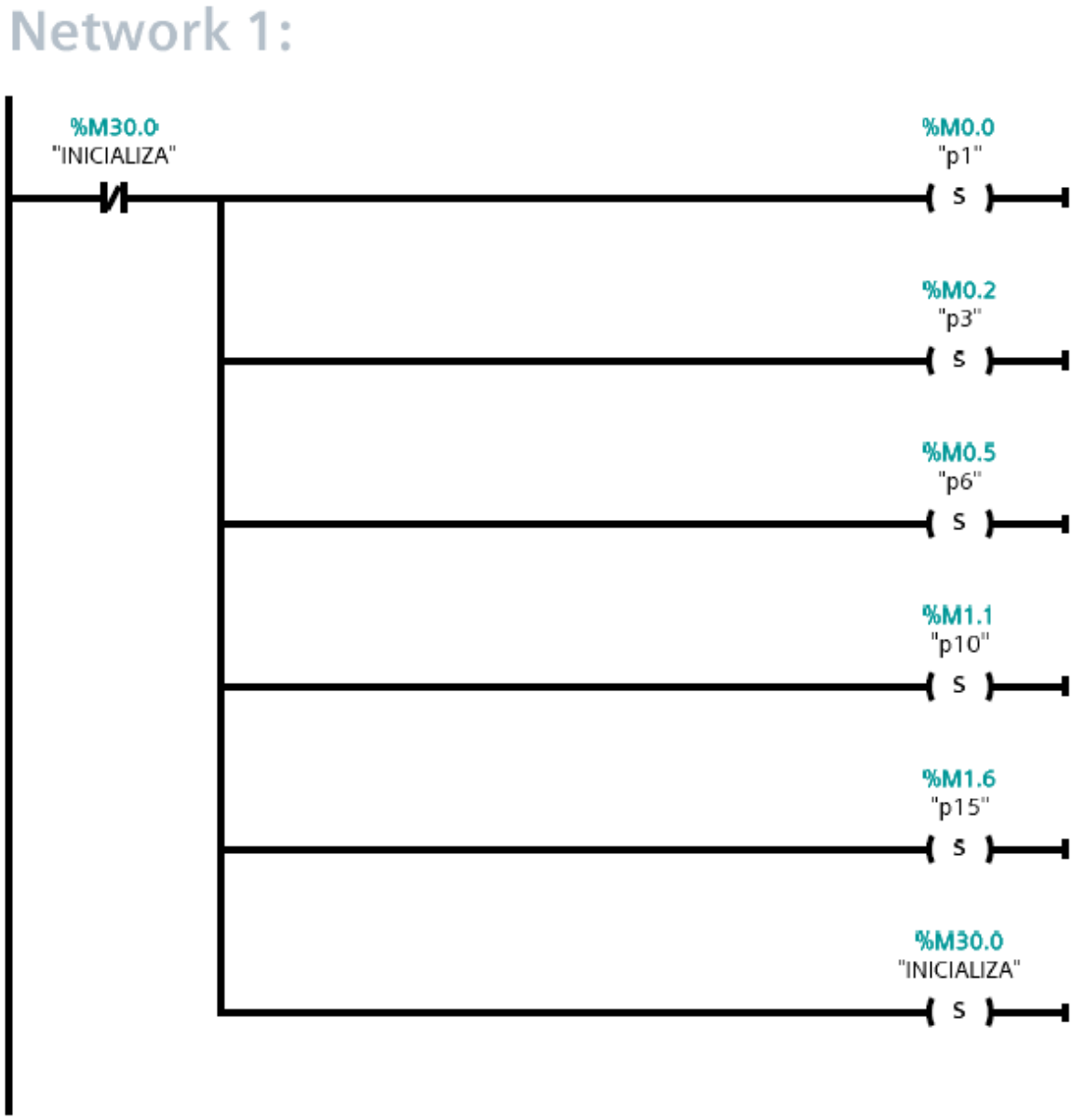
[6] Hellgren, A., Fabian, M., Lennartson, B,; On the execution of sequential function charts; Control Engineering Practice, 2005

[7] Silvestre, R. P.,; “Implementação em Ladder de Sistemas de Automação descritos por redes de Petri interpretadas para Controle”, Projeto final de graduação, UFRJ, Escola Politécnica, 2010.

[8] Fessler, D. P.; “Sistema de Automação de uma máquina injetora de plástico” Projeto final de graduação, UFRJ, Escola Politécnica, 2010.

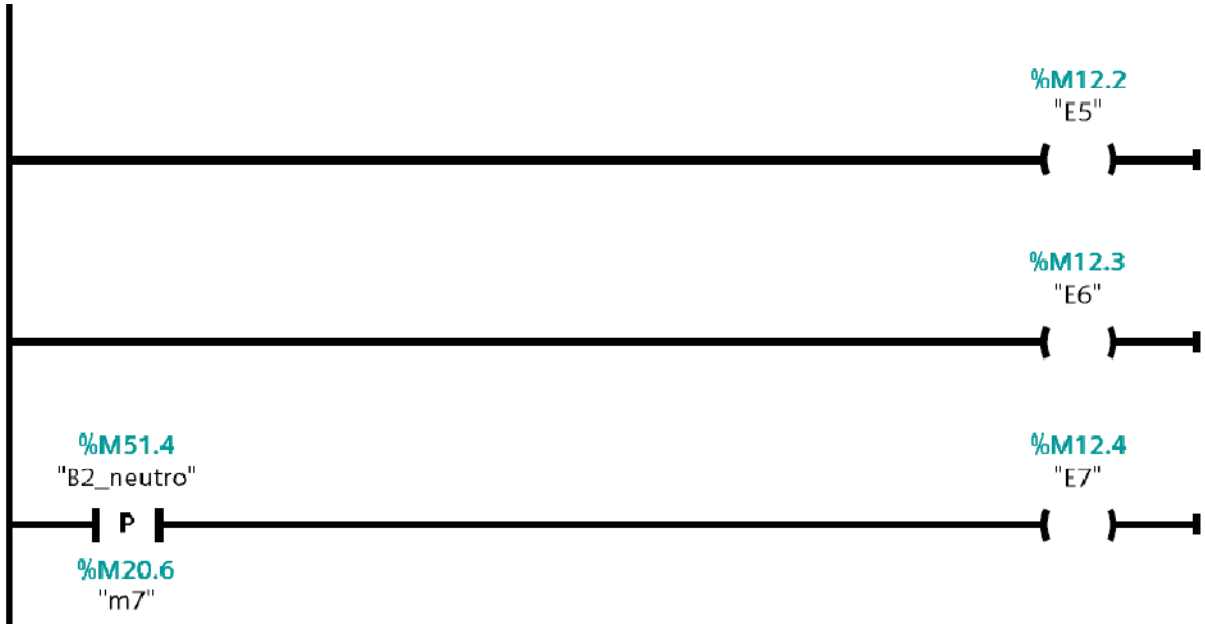
**APÊNDICE A – DIAGRAMA LADDER**

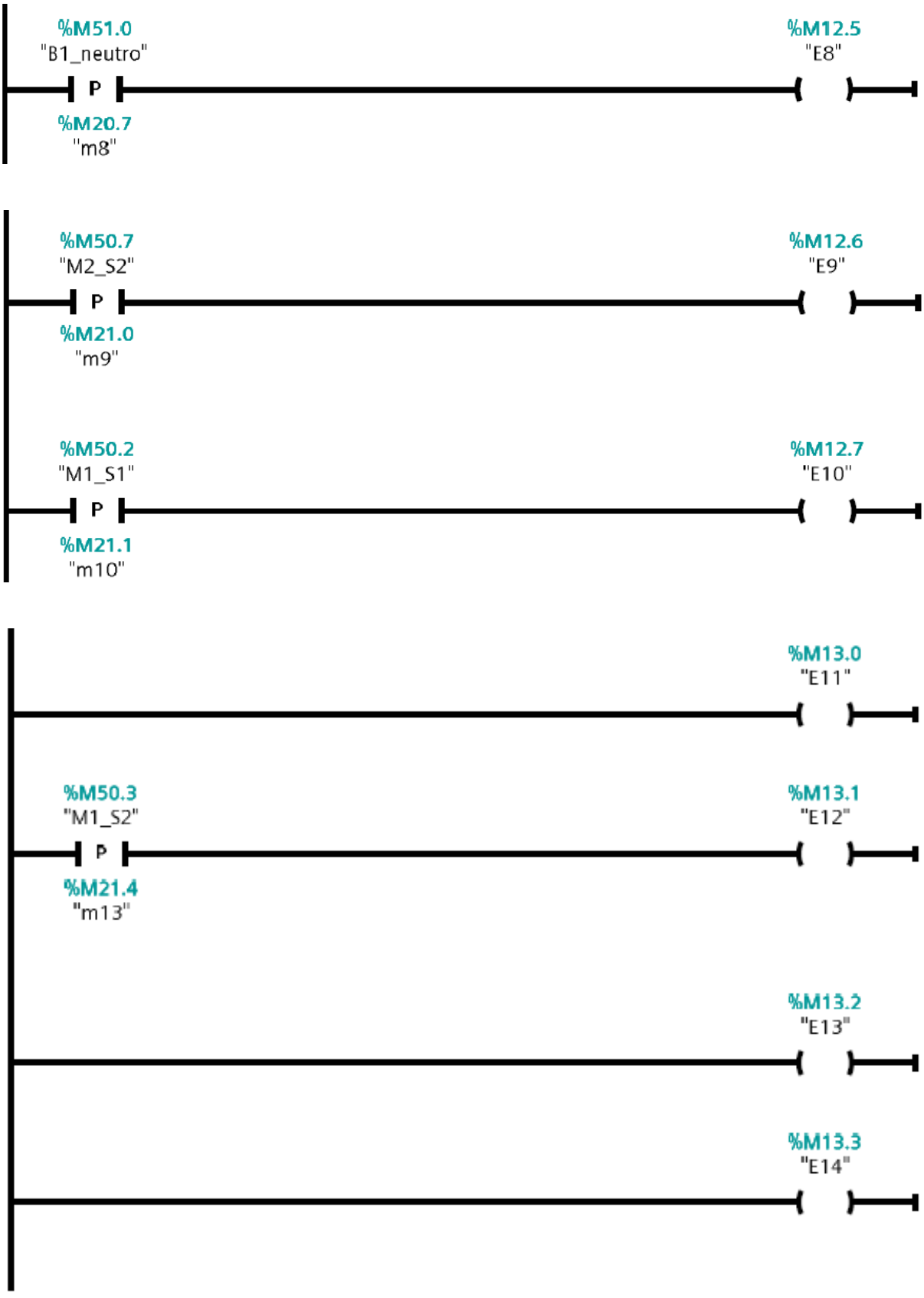
MÓDULO DE INICIALIZAÇÃO



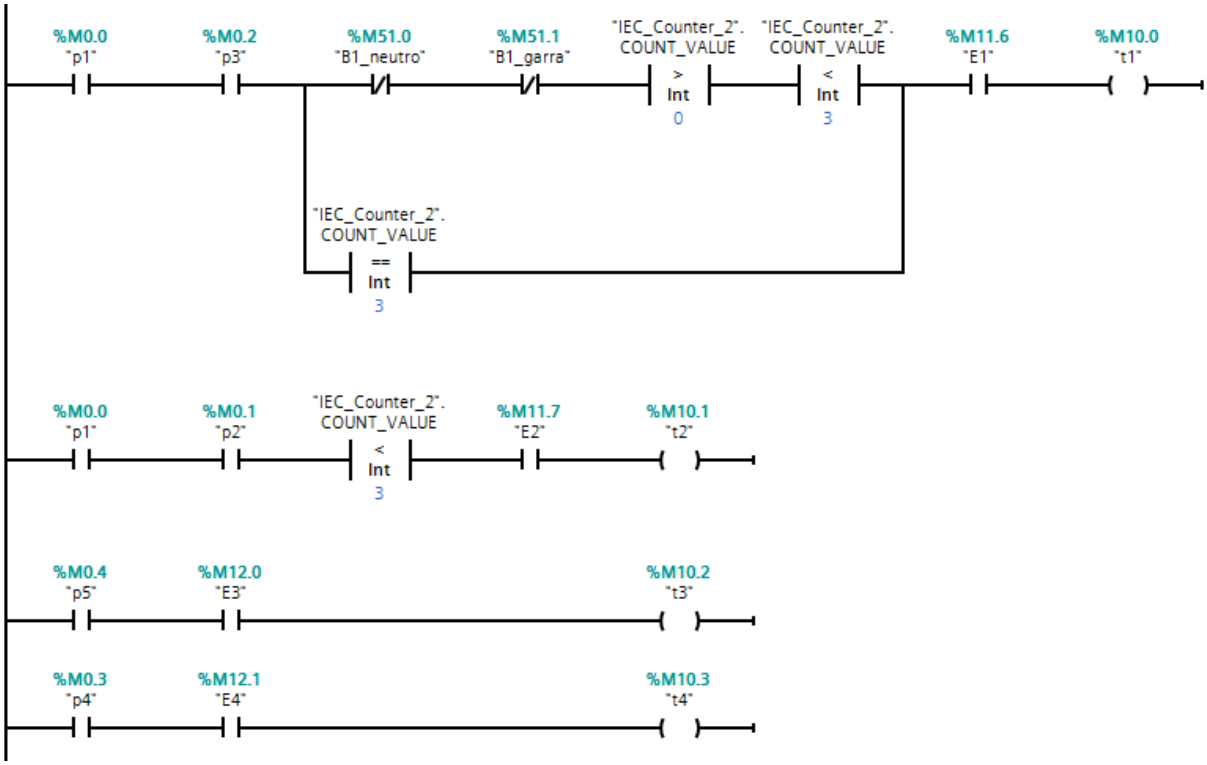
## Network 2:

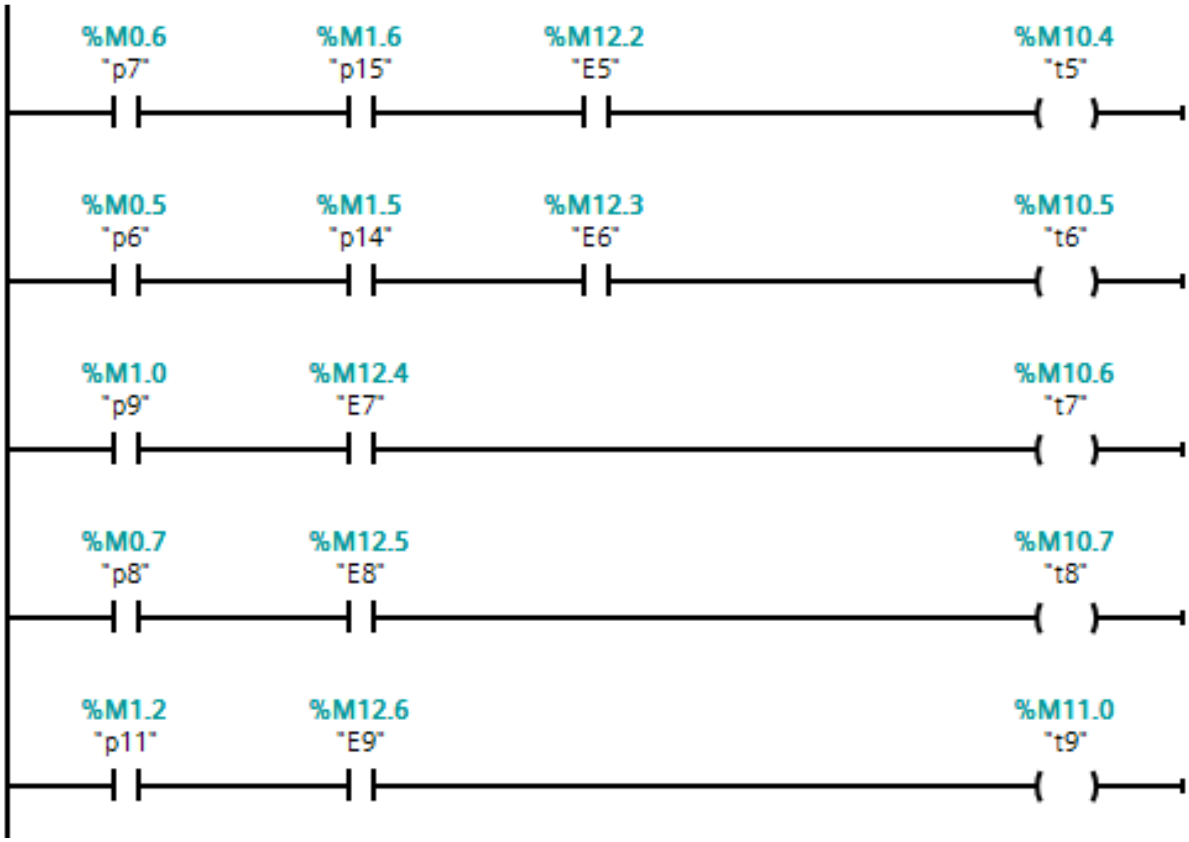

%M11.6
"E1"
%M11.7
"E2"
%M51.4
"B2_neutro"
P
%M20.0
"m1"
%M12.0
"E3"
%M51.0
"B1_neutro"
P
%M20.1
"m2"
%M12.1
"E4"

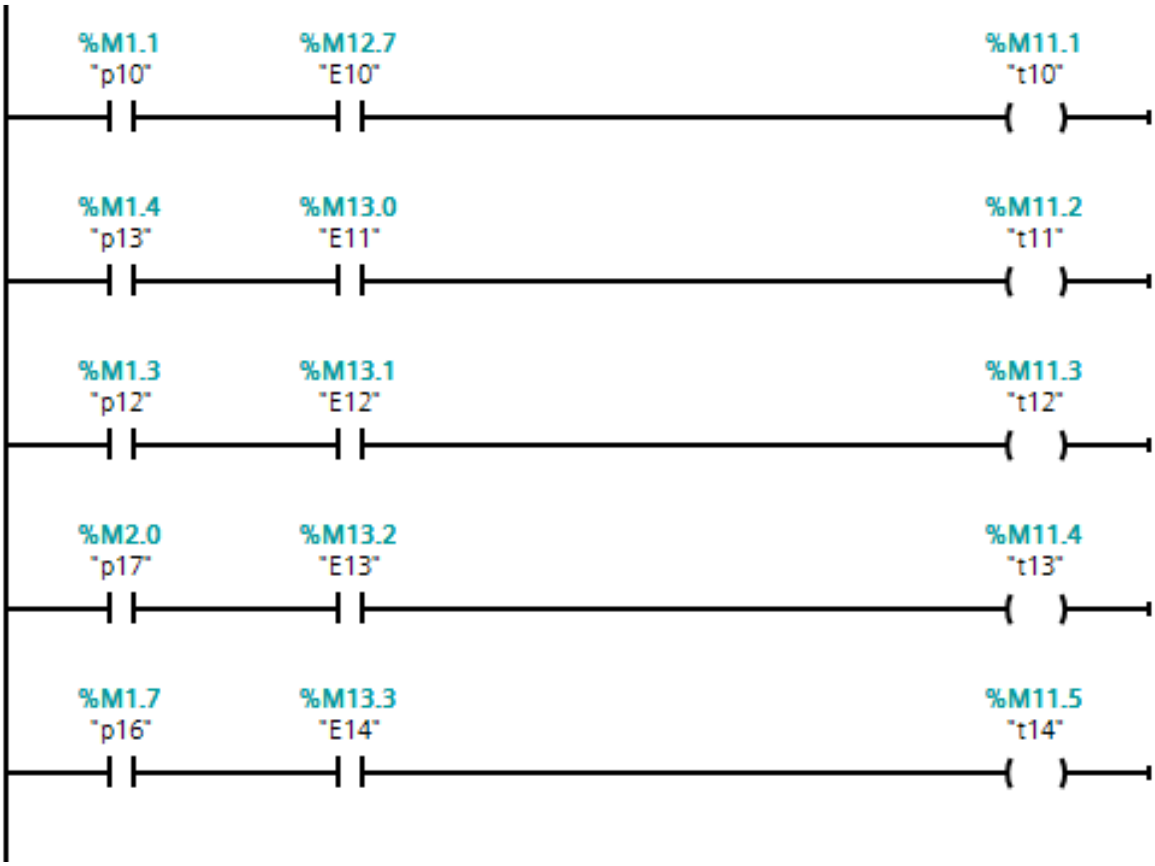

%M12.2
"E5"
%M12.3
"E6"
%M51.4
"B2_neutro"
P
%M20.6
"m7"
%M12.4
"E7"


%M51.0
"B1_neutro"
P
%M20.7
"m8"
%M12.5
"E8"
%M50.7
"M2_S2"
P
%M21.0
"m9"
%M12.6
"E9"
%M50.2
"M1_S1"
P
%M21.1
"m10"
%M12.7
"E10"
%M13.0
"E11"
%M50.3
"M1_S2"
P
%M21.4
"m13"
%M13.1
"E12"
%M13.2
"E13"
%M13.3
"E14"

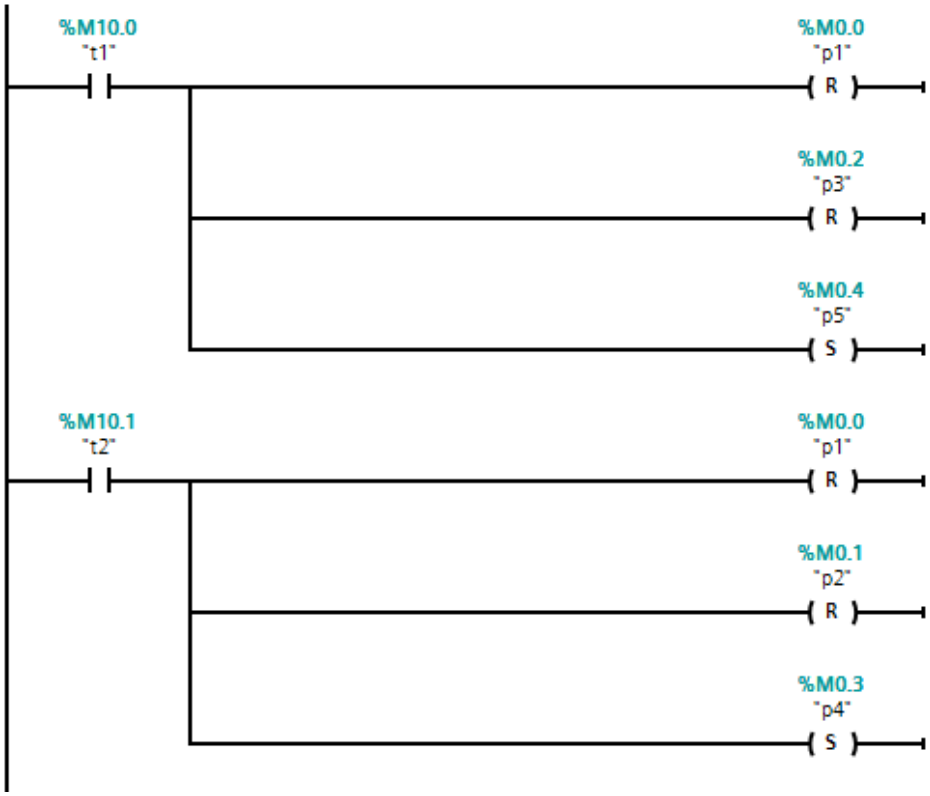
MÓDULO DE CONDIÇÕES PARA O DISPARO DAS TRANSIÇÕES

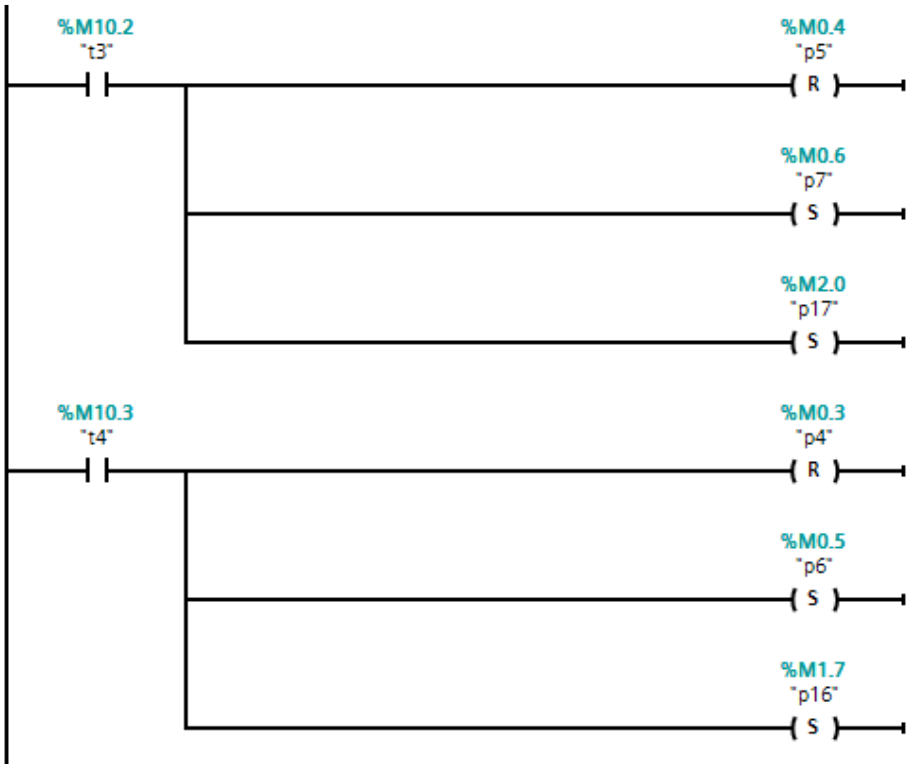


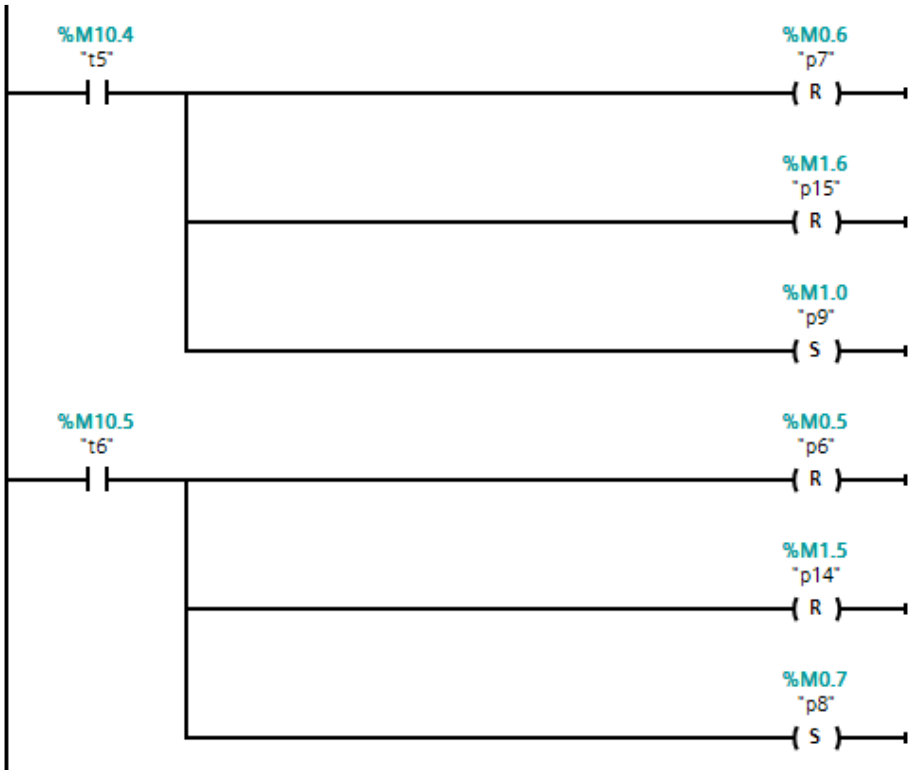



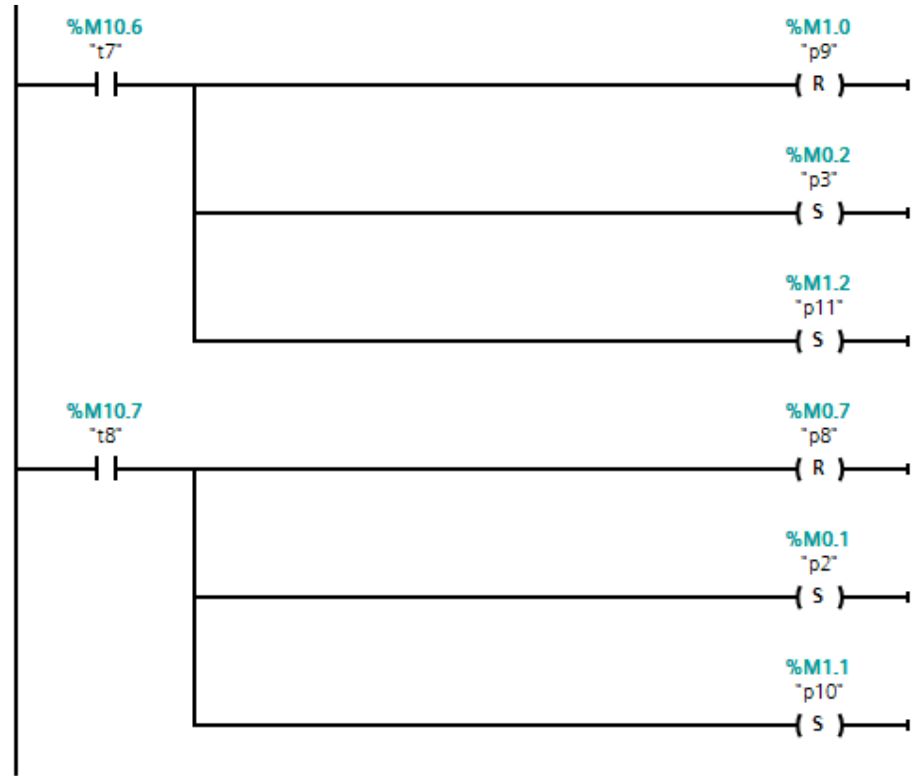


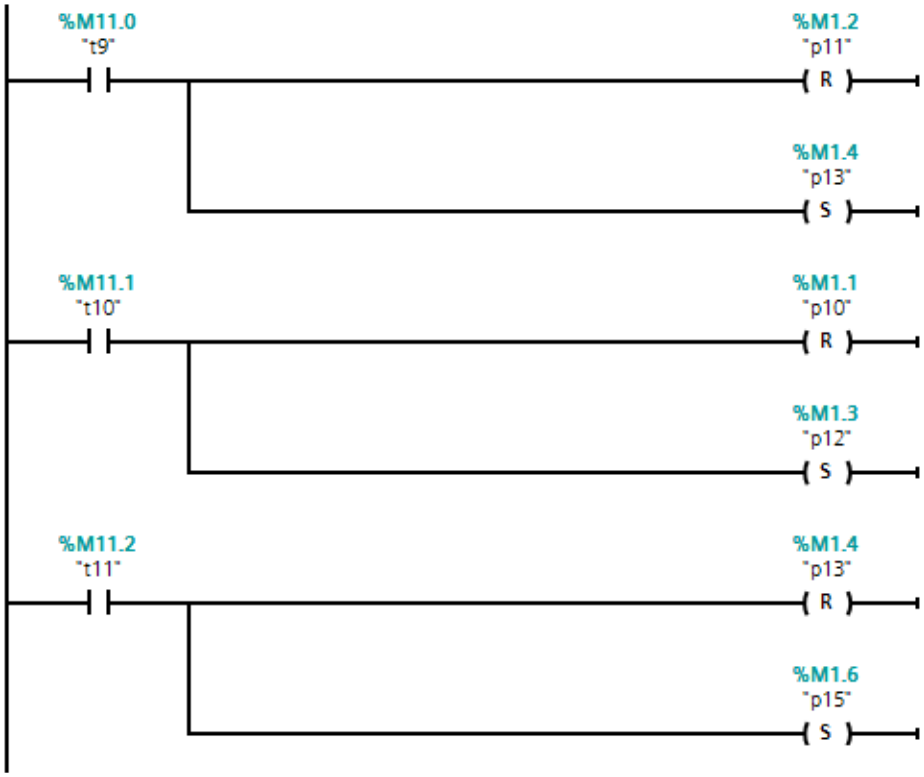

MÓDULO DE DINÂMICA

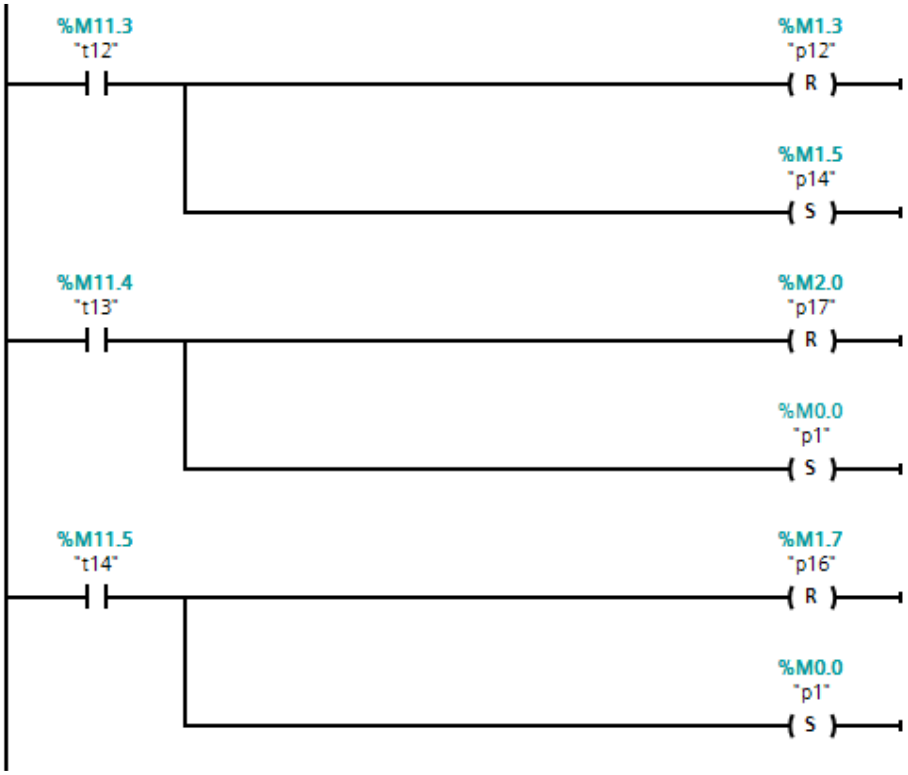



%M10.2
"t3"
%M0.4
"p5"
R
%M0.6
"p7"
S
%M2.0
"p17"
S
%M10.3
"t4"
%M0.3
"p4"
R
%M0.5
"p6"
S
%M1.7
"p16"
S

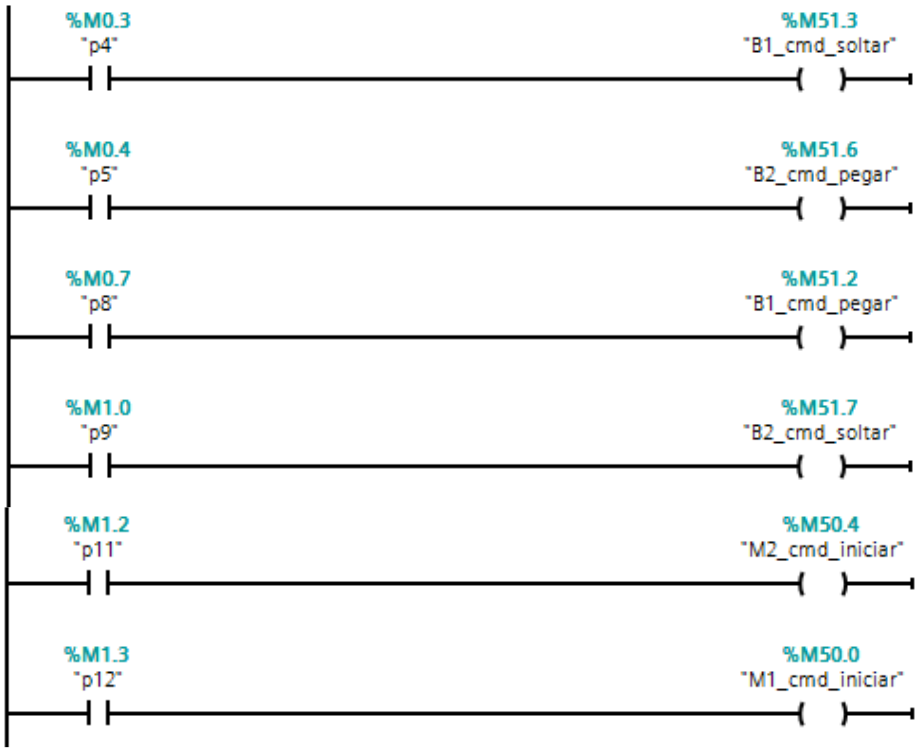

%M10.4
"t5"
%M0.6
"p7"
R
%M1.6
"p15"
R
%M1.0
"p9"
S
%M10.5
"t6"
%M0.5
"p6"
R
%M1.5
"p14"
R
%M0.7
"p8"
S

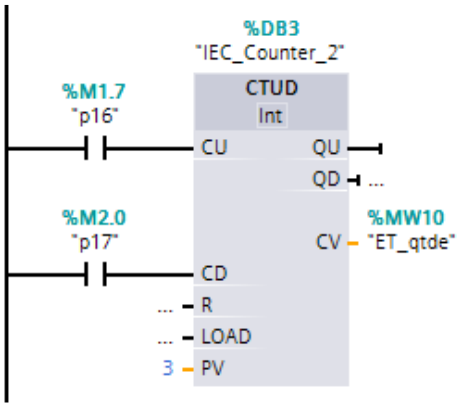

%M10.6
"t7"
%M1.0
"p9"
R
%M0.2
"p3"
S
%M1.2
"p11"
S
%M10.7
"t8"
%M0.7
"p8"
R
%M0.1
"p2"
S
%M1.1
"p10"
S


%M11.0
"t9"
%M1.2
"p11"
R
%M1.4
"p13"
S
%M11.1
"t10"
%M1.1
"p10"
R
%M1.3
"p12"
S
%M11.2
"t11"
%M1.4
"p13"
R
%M1.6
"p15"
S




MÓDULO DAS AÇÕES




%DB3
"IEC_Counter_2"
CTUD
Int
%M1.7
"p16"
CU
QU
QD
...
%M2.0
"p17"
%MW10
"ET_qtde"
CV
CD
...
R
...
LOAD
3
PV