

# LABORATÓRIO DE AUTOMAÇÃO DE SISTEMAS ELÉTRICOS



2012

## 1. Familiarização com o Controlador Programável

*Prof. Eduardo Cesar Senger*

*PEA/EPUSP*

Esta experiência visa familiarizar o aluno com o controlador programável CompactLogix e com o software de programação RSLogix 5000 fornecidos pela Rockwell Automation. Esta familiarização se dará através do desenvolvimento e implementação de dois programas exemplos baseados nas linguagens de programação previstas na norma IEC 61131-3.

# Familiarização com o Controlador Programável

## 1. FAMILIARIZAÇÃO COM O HARDWARE DO COMPACTLOGIX

A família CompactLogix de Controladores Programáveis (CP) fornecidos pela Rockwell Automation é uma solução modular de hardware para automação industrial que pode ser classificada como controladores de pequeno porte, mas que apresentam sofisticados recursos e características que antes só podiam ser encontradas em CP de grande porte.

Um sistema baseado nos controladores da família Compactlogix é constituído pela associação de três tipos de componentes: a) módulo de processamento e comunicação; b) fonte de alimentação; c) módulos de entradas e saídas. O hardware do CP utilizado no laboratório possui a seguinte configuração:

- módulo de processamento e comunicação modelo 1769-L32E, equipado com duas portas de comunicação (1 serial + 1 Ethernet).
- 01 cartão 1769-IQ16 com 16 entradas digitais 24 V<sub>DC</sub>,
- 01 cartão 1769-OB16 com 16 saídas digitais 24 V<sub>DC</sub>,
- 01 cartão 1769-IF4XOF2 com 4 entradas analógicas + 2 saídas analógicas.

A Figura 1 e a Tabela 1 mostram os principais componentes do CP disponível no laboratório. Vale observar que, para esse modelo de controlador, pode-se utilizar até 16 módulos distintos de entradas e saídas conforme a necessidade de pontos a serem monitorados e dispositivos a serem acionados para cada particular aplicação em campo.

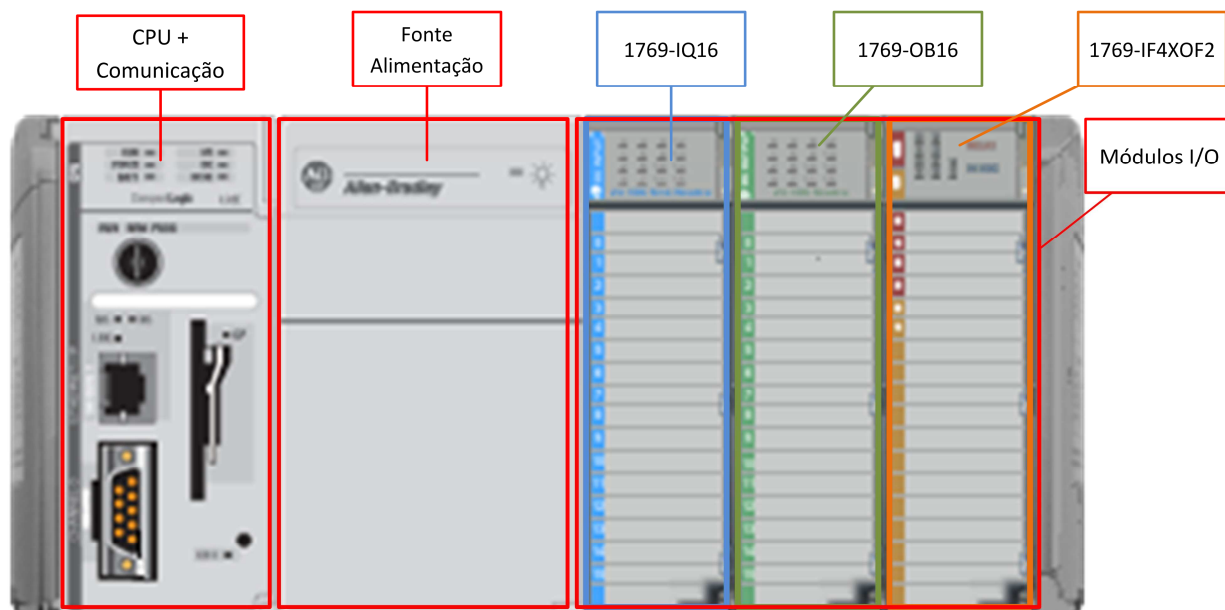


Figura 1 – Controlador Programável CompactLogix 1769-L32E

Tabela 1 - Identificação dos módulos do CP

Módulo	Código	Observações
CPU	1769-L32E	750KB de memória de instruções e controle
Entradas Digitais	1769-IQ16	16 entradas digitais (24V DC)
Saídas Digitais	1769-OB16	16 saídas digitais (24V DC)
Entradas/Saídas Analógicas	1769-IF4XOF2	4 entradas analógicas (04: 0-10V ou 0-20mA) 2 saídas analógicas (01: 0-10V + 01: 0-20mA)

A Tabela 2 indica os possíveis estados dos LEDs de sinalização frontal do módulo processador do Compactlogix e seus respectivos significados.

Tabela 2 - Indicação dos LEDs do módulo processador

LED	Quando	Indica que
RUN	Aceso verde constante	O controlador está no modo RUN
	Apagado	O controlador está em outro modo que não RUN
FORCE	Piscando âmbar	Entradas ou saídas foram forçadas para ON ou OFF sem que isto tenha sido habilitado
	Aceso âmbar constante	Pontos forçados foram habilitados
	Apagado	Pontos forçados inexistentes
BATT	Aceso vermelho constante	Bateria baixa, ou jumper inexistente, ou bateria e jumper não conectados.
	Apagado	Bateria OK ou jumper presente
I/O	Aceso verde constante	A configuração dos módulos de I/O corresponde à existente no programa e os mesmos estão comunicando corretamente
	Piscando Verde	Existe ao menos um módulo configurado no programa que não está comunicando com a CPU.
	Piscando Vermelho	Todos os módulos configurados no programa não estão comunicando com a CPU. Há uma falha no controlador
	Apagado	Não foram definidos módulos de entradas e saídas no programa do controlador Não há programa no controlador
OK	Aceso verde constante	Controlador está OK
	Piscando Verde	Controlador está lendo ou escrevendo na memória não volátil
	Aceso vermelho constante	Controlador apresentou uma falha grave (Reinicialize do CP)
	Piscando Vermelho	Controlador está em falha
	Apagado	Sem alimentação

Existem três modos possíveis de operação do CP, selecionados através da posição de uma chave seletora localizada na frontal do módulo da CPU.

**Posição RUN:** Esta posição habilita o CP ao modo de Operação (Run). Nesta condição o controlador executa as instruções do programa aplicativo, monitora dispositivos de entrada, energiza dispositivos de saída e ativa pontos forçados de I/O habilitados. O modo do CP pode ser alterado somente por meio da chave seletora. Não é possível desenvolver a edição do programa on-line.

**Posição PROG:** Esta posição habilita o CP ao modo de Programação (Program). O controlador não executa as instruções do aplicativo e as saídas são desligadas. É possível

desenvolver a edição do programa on-line. O modo pode ser alterado somente por meio da chave seletora.

**Posição REM:** Esta posição habilita o CP ao modo Remoto (Remote): modos REMote Run, REMote Program ou REMote Test. O modo do controlador pode ser alterado por meio da posição da chave seletora ou da interface de programação/operação. É possível desenvolver a edição de programa on-line nessa posição.

As figuras 3 a 6 mostram os esquemas elétricos para conexão com as portas dos três cartões de entradas e saídas digitais e analógicas utilizadas no hardware do controlador disponível no laboratório.

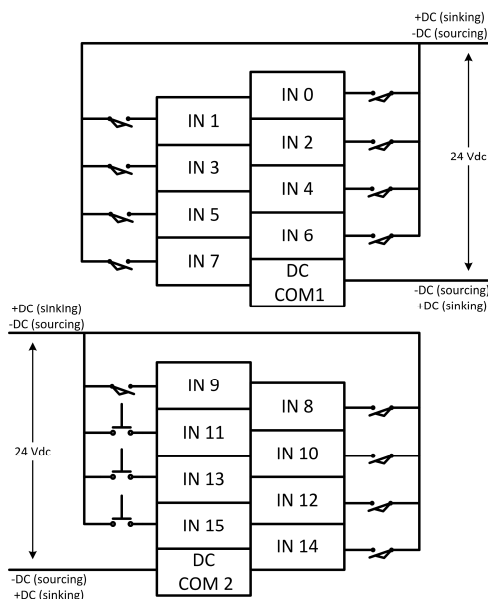


Figura 3. Cartão de entradas digitais

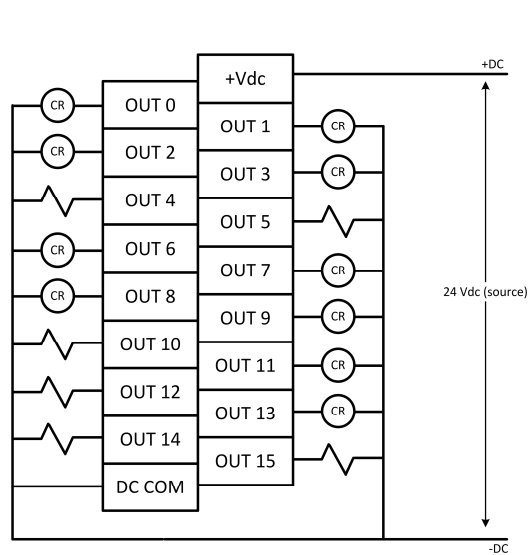
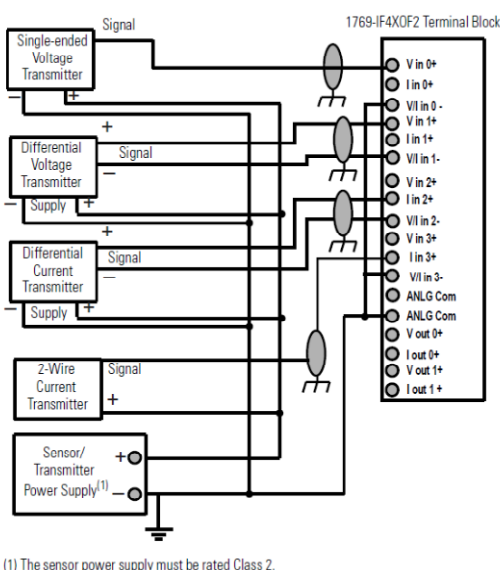


Figura 4. Cartão de saídas digitais



(1) The sensor power supply must be rated Class 2.

Figura 5. Entradas analógicas

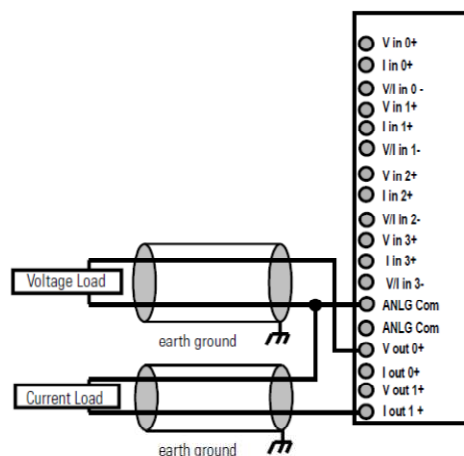


Figura 6. Saídas analógicas

## 1. Familiarização com o Controlador Programável

Para facilitar a montagem das experiências foi desenvolvido um kit didático contendo o Controlador Programável Compactlogix, borneiras de acesso às entradas e saídas dos cartões, e dispositivos para simulação e sinalização das entradas e saídas. O kit inclui também uma fonte interna de 24 Vcc e um switch 10/100 Mbps cujas portas podem ser acessadas pela lateral traseira. A Figura 7 indica os principais componentes desse kit didático.

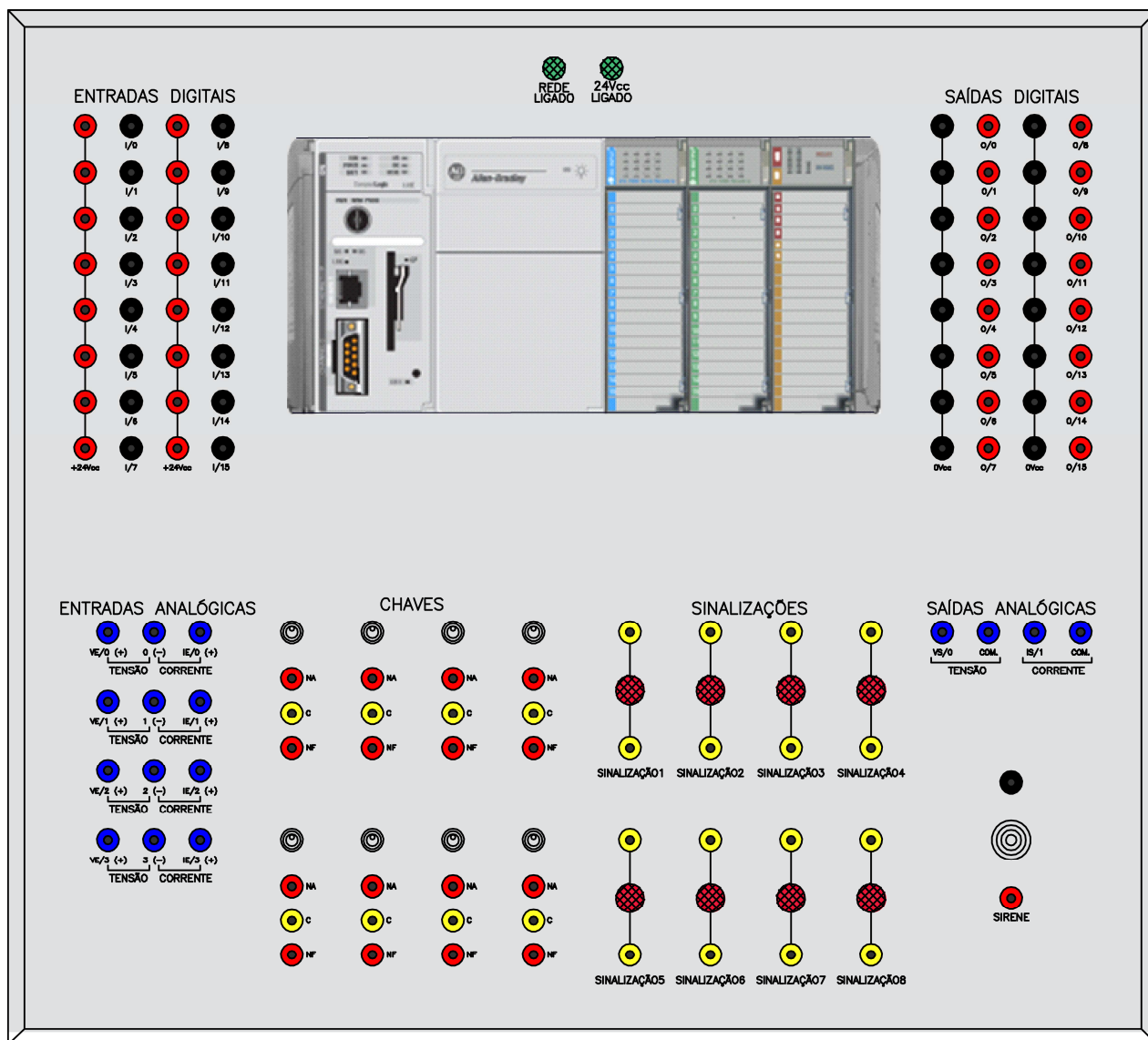


Figura 7. Kit didático disponível no laboratório

### 1.1.1 Energização e Inicialização do Kit didático

Siga o procedimento indicado a seguir para iniciar a utilização do kit didático:

- Observe atentamente cada módulo e pinagem existente no kit didático. Somente após estar absolutamente seguro de ter entendido todos os detalhes, energize o CP. **Em caso de dúvida chame o professor. Trata-se de equipamento eletrônico sensível e qualquer erro pode provocar a queima do equipamento.**

- b) Conecte o cabo de força do kit na tomada e ligue o disjuntor existente na lateral traseira (fonte do CP) e a chave na lateral esquerda (fonte de 24 Vcc). Verifique se os leds “Rede Ligado” e “24Vcc Ligado”, localizados na frontal do kit encontram-se acesos.
- c) Conecte os cabos de rede do CP e do computador no switch e verifique as configurações da rede (Ethernet);
- d) O Windows 7 instalado no computador possui dois usuários: user: **Administrator** com senha: **admin4all** e user: **Operador** com senha: **operador**. Utilize o segundo usuário. Os softwares da Rockwell encontram-se instalados em uma máquina virtual (para tanto, execute o software **VMware Player**). A máquina virtual é **Windows Server 2008 SP2 x86**, com user: **Administrator** e senha: **1234567890**. Armazene os arquivos criados na pasta **Arquivo dos Alunos**.
- e) Abra o programa RSLinx Classic localizado em Menu Iniciar/Todos os programas/Rockwell Software/**RSLinx**
- f) Na tela do RSLinx Classic Lite, clique em Communications/RSWho.
- g) Verifique a existência da conexão AB\_ETHIP-1, Ethernet. Se ela existir, passe para o item l).
- h) Não existindo a conexão acima, é necessário criá-la para verificar a conexão entre o CP e o PC. Na tela do RS Linx Classic Lite clique em Communications/Configure Drivers.
- i) Na tela Configure Drivers em Available Driver Types, selecione a opção EtherNet/IP Driver, clique em Add new e, na tela seguinte, em OK.
- j) A tela Configure driver: AB\_ETHIP-1 deve abrir automaticamente.
- l) Clique em OK
- m) Clique em Close
- n) Verifique se o CP está visível e comunicando (sem um ‘X’ vermelho sobre ele).

Para criar um novo programa aplicativo siga o procedimento abaixo para que o software RSLogix500 reconheça o CP e seus cartões de expansão:

- a) Abra o programa RSLogix5000 localizado em Menu Iniciar/Todos os programas/Rockwell Software/RS Logix 5000 Enterprise Series/ **RSLogix5000**
- b) Clique em file/new
- c) Abrirá a tela New Controller. Dê um nome para o seu programa em Name, selecione o CP a ser utilizado; no caso, o 1769-L32E CompactLogix5332E Controller e a revisão do Firmware (Revision: 19).
- d) Clique em OK. A tela do seu programa irá abrir.
- e) Clique com o botão direito sobre CompactBus Local localizado à esquerda na tela
- f) Clique em New Module..., abrirá a tela Select Module.
- g) Clique sobre a categoria Digital e em seguida sobre o módulo 1769-IQ16.
- h) Confirme a inclusão desse módulo clicando OK
- i) A tela de propriedade do módulo deverá abrir, configure o Slot como 1 e clique OK

- j) Repita os passos de e) a i) para adicionar os seguintes módulos:
  - Saídas Digitais: 1769-OB16 (Slot 2)
  - I/O Analógico: 1769IF4XOF2 (Slot 3)
- k) Na tela de propriedades de cada módulo é possível adicionar um nome e uma descrição para os mesmos, além, é claro, de configurar algumas particularidades do módulo. Verifique as propriedades de cada módulo que você acabou de adicionar.
- l) Clique em file/save e salve seu programa

Após concluir o desenvolvimento de seu aplicativo no software RSLogix5000, siga o procedimento abaixo para carregar o aplicativo do PC para o CP:

- a) Clique em Communications/Who Active a tela Who Active abrirá.
- b) Localize o CP de sua bancada (AB\_ETHIP-1,Ethernet/"nº do IP do CP...") e clique em Download.

## 2. FAMILIARIZAÇÃO COM O SOFTWARE RSLOGIX 500

### 2.1 Exemplo 1: Sistema de Partida de Pórtico com Alarme

Este tipo de sistema é normalmente utilizado na partida de equipamentos móveis como, por exemplo, pórticos, esteiras transportadoras, etc. Quando o push-button de partida é acionado um alarme sonoro e luminoso é ligado por certo intervalo de tempo (10 seg). Isto permite que pessoas nas proximidades do equipamento sejam alertadas do funcionamento iminente do mesmo. Depois de transcorrido o tempo de alarme, o equipamento é colocado em movimento.

O sistema de controle a ser desenvolvido deverá atender aos seguintes requisitos:

- O sistema pode ser operado em modo *Automático* ou modo *Manual*. A escolha do modo de operação é definida pela posição de uma chave seletora de duas posições (Manual/Automático).
- No modo Automático a partida e parada do processo serão realizadas através de uma botoeira com dois botões sem retenção (*push-button*). O primeiro é do tipo NA e é utilizado para partir o processo, o qual inicialmente fará o acionamento do alarme durante 10 s e, na sequência, energizará o motor do equipamento. O acionamento do segundo *push-button*, do tipo NF, provoca a parada imediata do equipamento. No modo *Manual*, a função desses dois botões é desabilitada.

- Para implementar o alarme sonoro e luminoso deverão ser utilizados uma sirene e um LED, disponíveis no kit didático, conectados às portas 01 e 02 do cartão de saídas digitais. Durante o período de alarme, de forma a gerar um sinal de alerta conveniente, as portas 01 e 02 deverão alternar entre os estados *on/off* com frequência de 1 Hz (sinal sonoro) e 0.5 Hz (sinal luminoso), respectivamente.
- No modo *Manual*, deverá ser habilitada uma segunda botoeira, semelhante à descrita acima, com a função de controle manual do equipamento. O acionamento dessa botoeira irá partir ou parar o motor imediatamente, sem o período de alerta de partida. O modo *Manual* é utilizado somente para fins de manutenção do equipamento. No modo *Automático*, a função do *push-button* de partida manual é desabilitado, enquanto que o de parada deve permanecer ativo, funcionando como um botão de emergência prioritário.
- O método de partida do motor é por partida direta através do acionamento de um contator trifásico. Falhas no processo de partida (devido à quebra da fiação de controle, emperramento ou bloqueio do contator, etc) serão detectadas através do monitoramento de um contato auxiliar do contator (Obs: caso o motor fosse acionado por um inversor de frequência conectado ao CP através de rede Ethernet, a ocorrência de falha no acionamento do motor poderia ser informada ao CP através da rede de comunicação).

O sistema de controle do motor é constituído por dois circuitos eletricamente independentes: a) circuito de controle; b) circuito de potência. A seguir são mostrados os detalhes de cada um desses dois circuitos.

### - Circuito de Controle

Na configuração do CP disponível no laboratório o cartão de entradas digitais opera com tensão de 24 Vcc e, dessa forma, essa é a tensão utilizada no circuito de controle que conecta as diversas botoeiras, chaves e contatos com esse cartão. A figura 8 apresenta o diagrama elétrico para esse circuito.



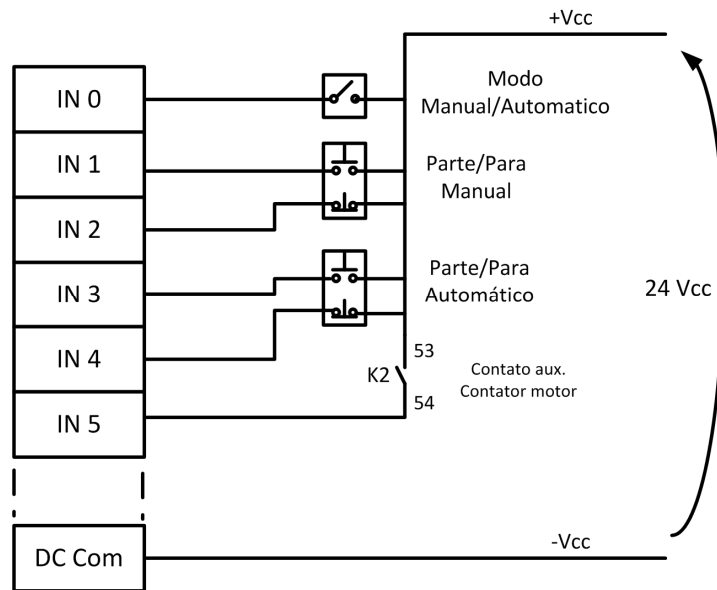
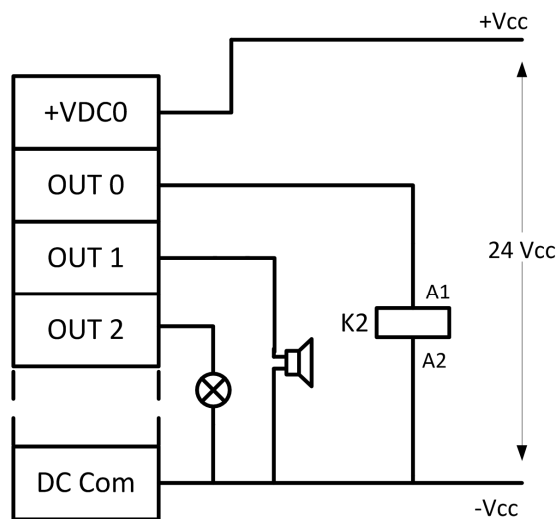


Fig 8. Circuito de Controle – Entradas digitais

Na figura 9 é mostrado o circuito de controle que conecta a sirene, o led e a bobina do contator que energiza o motor com o cartão de entradas digitais. O contator utilizado possui bobina com tensão nominal de 24 Vcc.



K2 – bobina do contator de acionamento do motor

Figura 9. Circuito de Controle – saídas digitais

- Circuito de Potência

Neste exemplo, considera-se o motor, acionado pelo contator K2, como sendo trifásico, 220 Vac, partida direta, com os enrolamento conectados em delta. O circuito de potência utilizado para energização desse motor é mostrado na figura 10.

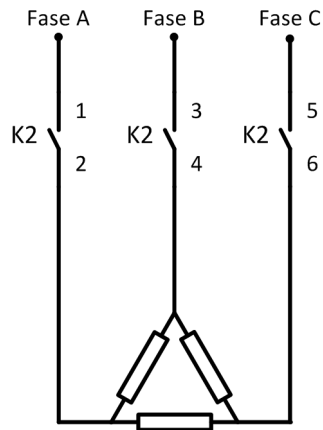


Figura 10. Circuito de potência responsável pela energização do motor

Como mostrado na tabela 3 e na figura 11, a estrutura do software a ser implementado no CP para controle do sistema descrito é constituído por uma única *Tarefa* (*Tarefa Principal*) do tipo *contínua*, constituída por dois programas: **Aciona\_MOTOR** e **Processo\_Sequencia**. O primeiro programa possui duas rotinas (**Principal**, **M\_01**) e o segundo três rotinas (**Principal**, **Anunciador**, **Sequencia**).

Tabela 3. Estrutura do Software para controle do Pórtico

Tasks	Programas	Rotinas	Linguagem
<i>Tarefa Principal</i>	<i>Aciona_MOTOR</i>	<i>Principal</i>	<i>Ladder</i>
		<i>M_01</i>	<i>Ladder</i>
	<i>Processo_Sequencia</i>	<i>Principal</i>	<i>Ladder</i>
		<i>Anunciador</i>	<i>Ladder</i>
		<i>Sequencia</i>	<i>SFC</i>

A base de dados com as variáveis utilizadas no software são mostradas na figura 12. As cinco primeiras variáveis são do tipo booleanas, enquanto que a última, denominada *M\_01*, é uma estrutura cujos campos são descritos na Tabela 4. Essa estrutura armazena toda a informação, tanto de status como de comando, associada com o motor 01. Caso fosse utilizado mais de um motor, essa estrutura poderia ser replicada para cada motor adicional utilizado.

Como uma boa prática de programação, recomenda-se sempre utilizar lógica direta para as variáveis booleanas, ou seja, nível lógico alto (1) para indicar comando ou status ativo (por ex., *M\_01.STS.AUTO = 1* indica que o modo *AUTOMATICO* encontra-se ativado).

# 1. Familiarização com o Controlador Programável

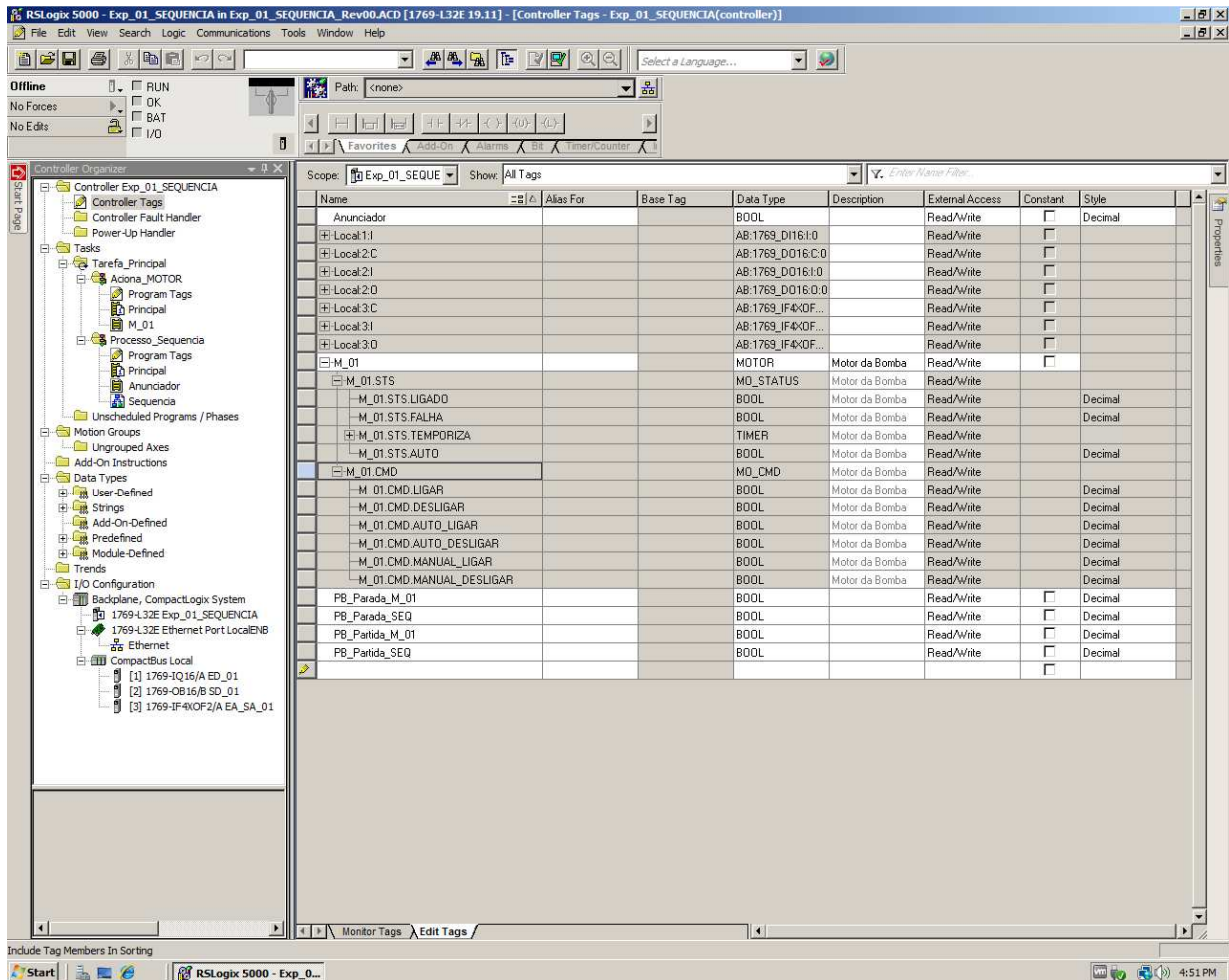


Figura 11. Estrutura e Base de Dados do Software de Controle do Pórtico.

Tabela 4. Descrição da estrutura M\_01

Tipo de informação	Campo	Descrição
status	M_01.STS.AUTO	Variável interna associada à posição da chave <i>Auto/Manual</i> (entrada digital 0)
	M_01.STS.LIGADO	Variável interna associada ao contato auxiliar do contator (entrada digital 5)
	M_01.STS.FALHA	Variável que indica falha na partida do motor (comando de ligar foi enviado e contator não ligou)
	M_01.STS.TEMPORIZA	Variáveis do temporizador utilizado para detectar falha na partida do motor
comando	M_01.CMD.MANUAL_LIGAR	Variável interna associada ao acionamento do botão <i>Parte Motor (Manual)</i> (entrada digital 1)
	M_01.CMD.MANUAL_DESLIGAR	Variável interna associada ao acionamento do botão <i>Para Motor (Manual)</i> (entrada digital 1)
	M_01.CMD.AUTO_LIGAR	Comando automático para ligar contator
	M_01.CMD.AUTO_DESLIGAR	Comando automático para desligar contator
	M_01.CMD.LIGAR	Variável interna associada à saída digital 0 que aciona o contator que energiza o motor

As características principais de cada um dos dois programas indicados na Tabela 3 são comentadas a seguir.

### A)- Programa: *Aciona\_MOTOR*

O programa *Aciona\_MOTOR* é constituído por duas rotinas:

#### A1)- Rotina 1: *Principal*

Cada programa sempre conterá uma rotina Principal, escrita em linguagem Ladder, que fará a chamada das diversas rotinas que constituem o programa. Para o programa *Aciona\_MOTOR* o código em linguagem Ladder é mostrado na figura 12.

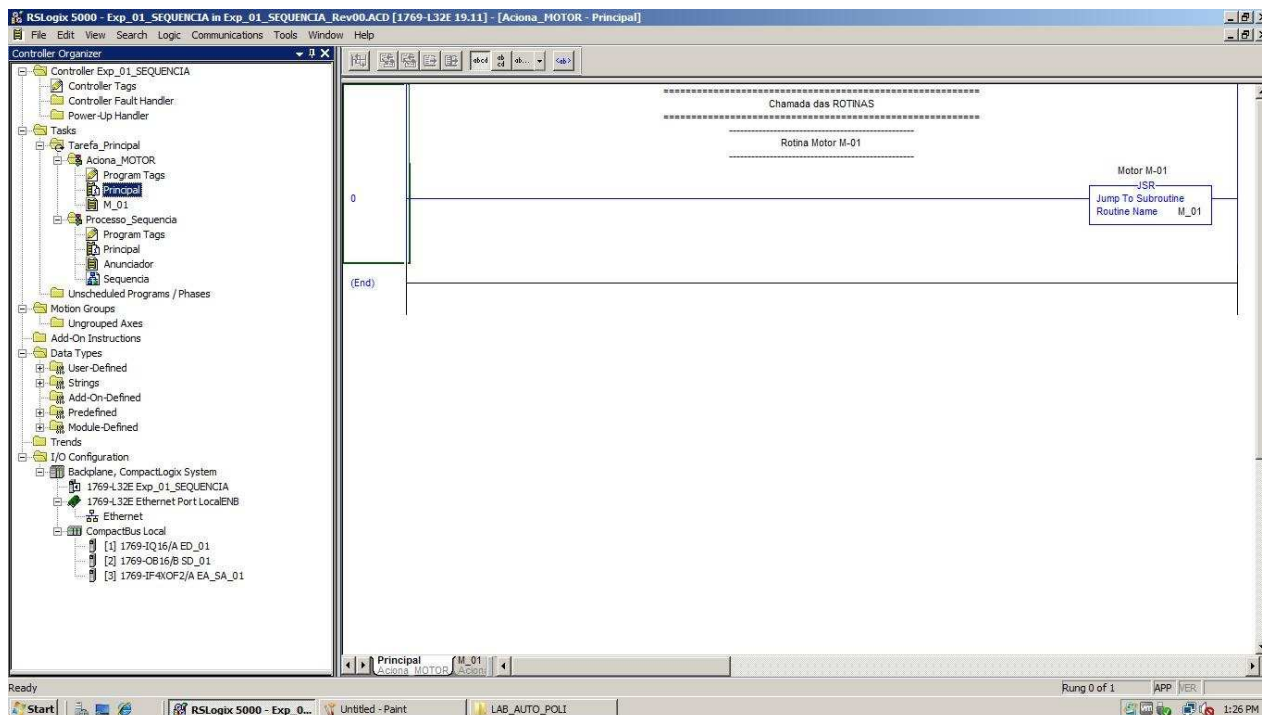


Figura 12. Rotina Principal do programa *Aciona\_Motor*

#### A2)- Rotina 2: *M\_01*

A figura 13 mostra o código da rotina *M\_01*, escrito em linguagem Ladder, para controle de partida/parada do motor. Como pode ser observado, as instruções que constituem essa rotina são agrupadas em três blocos: *Leitura das Entradas*; *Lógica do Motor*; *Escrita das Saídas*.

O bloco *Leitura das Entradas* agrupa as instruções que fazem a leitura das diversas entradas digitais utilizadas e atribuem os valores lidos às correspondentes variáveis internas do programa. Essas variáveis internas são em seguida utilizadas nas instruções que constituem o bloco *Lógica do Motor*. No bloco *Escrita das Saídas*, algumas das variáveis internas geradas pelo bloco anterior

são atribuídas às portas correspondentes do cartão de saídas digitais do CP. Esse arranjo é recomendado como uma boa prática de programação pelas seguintes razões:

- O status das variáveis internas do programa associadas às entradas digitais são atualizadas somente no início do scan. Dessa forma, variações das entradas digitais no meio do scan não serão consideradas até o início do próximo scan, o que evita possíveis efeitos indesejáveis. De forma análoga, as saídas digitais somente serão atualizadas no fim do scan.
- Esse arranjo permite sempre empregar, para as variáveis utilizadas ao longo do programa, o nível lógico alto para indicar comando ou status ativo (lógica direta), independente do tipo de saída ou entrada utilizada pelos equipamentos físicos existentes no campo. No momento da implementação em campo, caso algum dos sinais de entrada ou saída não obedeça a essa lógica, basta alterar a linha corresponde no bloco de *Leitura das Entradas* ou de *Escrita das Saídas*. Por exemplo, a variável interna `PB_Parada_M_01` deverá assumir o valor alto quando o operador acionar o *push-button* de parada manual do motor. Como foi admitido que esse botão é do tipo NF (entrada 2 do cartão de entradas digitais), na segunda linha do código mostrado na figura 13, utilizou-se uma instrução XIO. Caso, no momento da implementação em campo, se verifique que o botão é na realidade do tipo NA, bastaria somente alterar essa instrução para XIC, sem a necessidade de alterações em diversos pontos ao longo do código.

No bloco Lógica do Motor, a linha 5 gera a variável `M_01.CMD.LIGAR` que irá ser atribuída à saída 0 do cartão de saídas digitais para acionamento da bobina do contator que energiza o motor. Na linha 6 é gerado um sinal de falha na partida do motor (variável `M_01.STS.FALHA`) caso, após o comando de ligar o motor (variável `M_01.CMD.LIGAR` igual a 1) não ocorra o fechamento do contato auxiliar do contator (variável `M_01.STS.LIGADO` indo para 1) em até 5 segundos. A ocorrência dessa condição desativa o comando de ligar o motor (linha 5).

Caso existisse mais de um motor no processo a ser controlado, poderia ser criada uma nova instância dessa rotina para cada um dos motores existentes.

# 1. Familiarização com o Controlador Programável

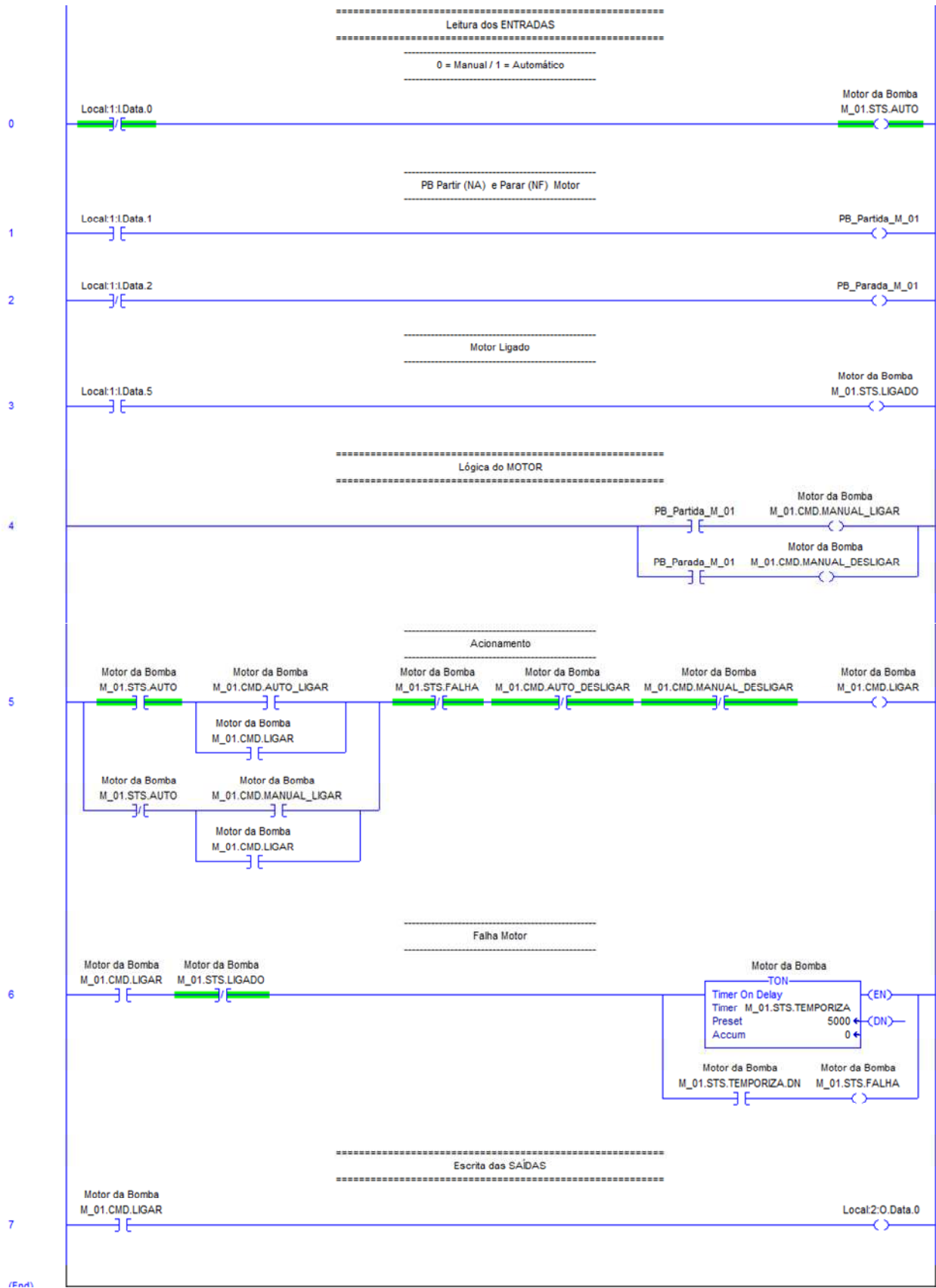


Figura 13. Rotina M01

**B)- Programa: Processo\_Sequencia**

O programa *Processo\_Sequencia*, por sua vez, é constituído por três rotinas:

**B1)- Rotina 1: Principal**

Como mostrado na figura 14, esta rotina, escrita em linguagem Ladder, nas duas primeiras linhas (bloco *Leitura das ENTRADAS*) faz as leituras das entradas digitais associadas com o controle automático do processo, isto é, dos botões *Parte/Para Processo (Autom)*. Em seguida, faz a chamada da sub-rotina *Sequencia*.

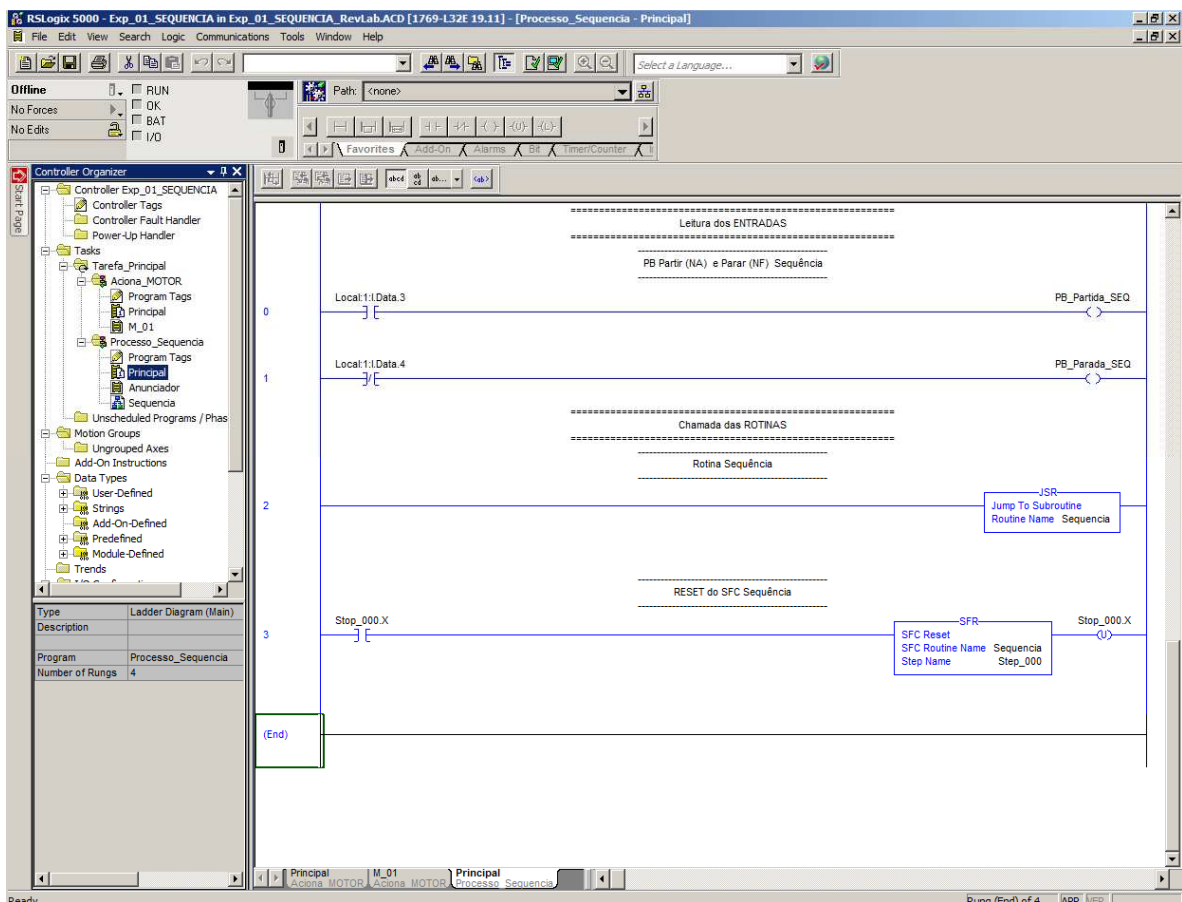


Figura 14. Processo\_Rotina\_Principal

**B2)- Rotina 2: Sequencia**

O código da rotina *Sequencia*, escrita em linguagem SFC (Sequential Function Chart), é mostrado na figura 15. Como pode ser observado, esse SFC é constituído por 3 passos. O passo *Step\_000* inicializa, isto é, zera as variáveis *M\_01.CMD.AUTO\_LIGAR* e *M\_01.CM.AUTO\_DESLIGAR*. A transição para o segundo passo (*Step\_001*) ocorre quando a variável *PB\_Partida\_SEQ* torna-se ativa. Essa passo permanece ativo até que o *push-button* de

parada é acionado e a variável *PB\_Parada\_SEQ* torna-se verdadeira. Esse passo contém três ações:

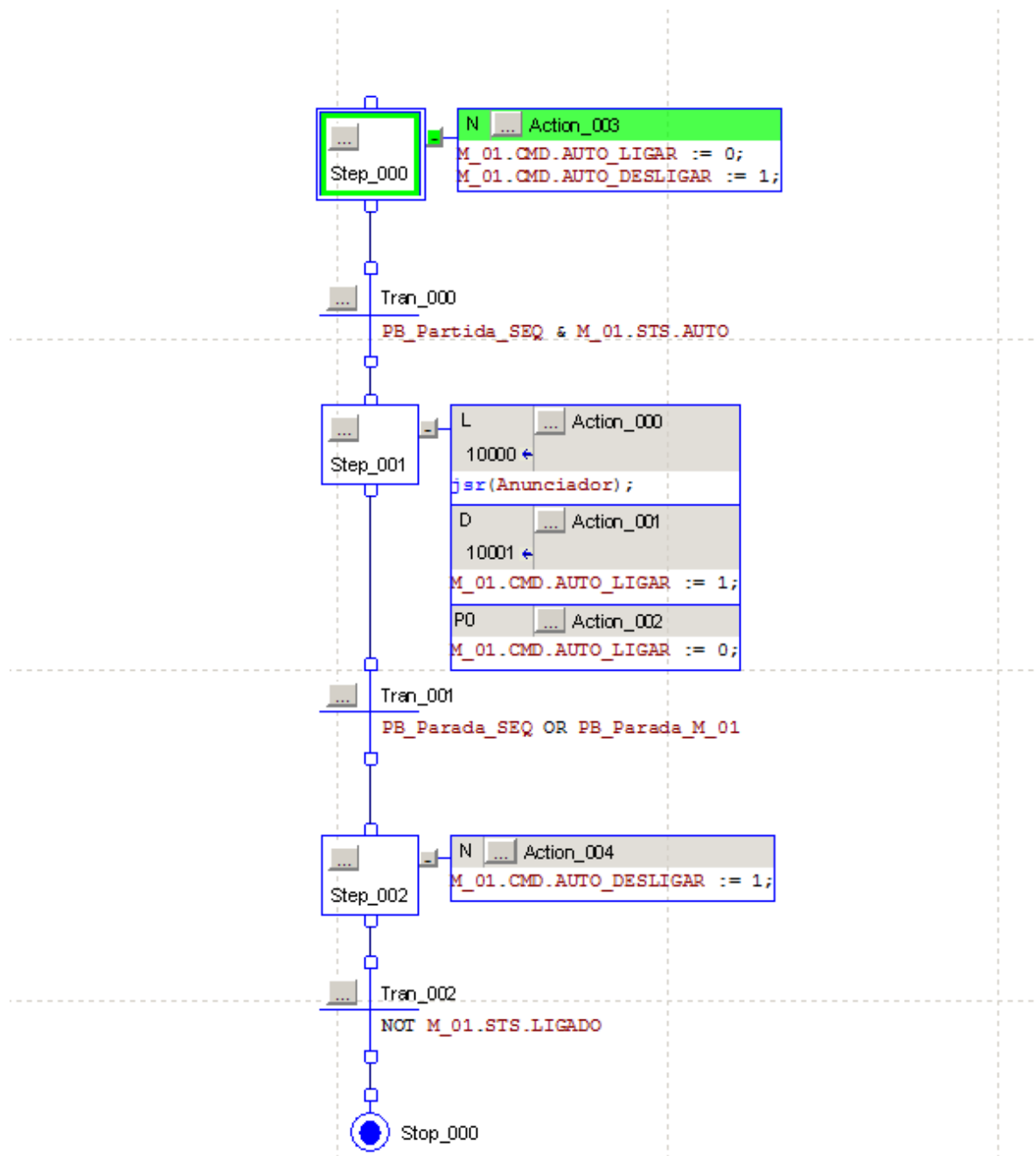


Figura 15. Rotina\_Sequencia

- Action\_000 (ação do tipo L – limitada no tempo): executa a rotina *Anunciador* (gera o sinal luminoso e sonoro com frequência de 1Hz) durante os primeiros 10000 mseg;
- Action\_001 (ação do tipo D – atrasada no tempo): esta ação, iniciada no instante 10001 mseg e encerrada quando o passo é desativado, faz `M_01.CMD.AUTO_LIGAR = 1`.



- Action\_002 (ação do tipo P0 – executada somente uma vez quando o passo é desativado): faz  $M_{01}.CMD.AUTO\_LIGAR = 0$ .

Após o terceiro passo, quando o motor é desligado, a transição *Tran\_002* torna-se verdadeira e o SFC vai para Stop. Nesse último estado, a variável *Stop\_000.X* torna-se verdadeira, o que produz o reset do SFC através da instrução SFR na última linha da rotina principal (vide figura 14).

### B3)- Rotina 3: Anunciador

O código da rotina *Anunciador*, escrita em Ladder, é mostrado na figura 16. Essa rotina gera o sinal de 1 Hz na porta 01 e de 0.5 Hz na porta 02 do cartão de saídas digitais, que é utilizado para produzir a sinalização sonora e luminosa de alerta.

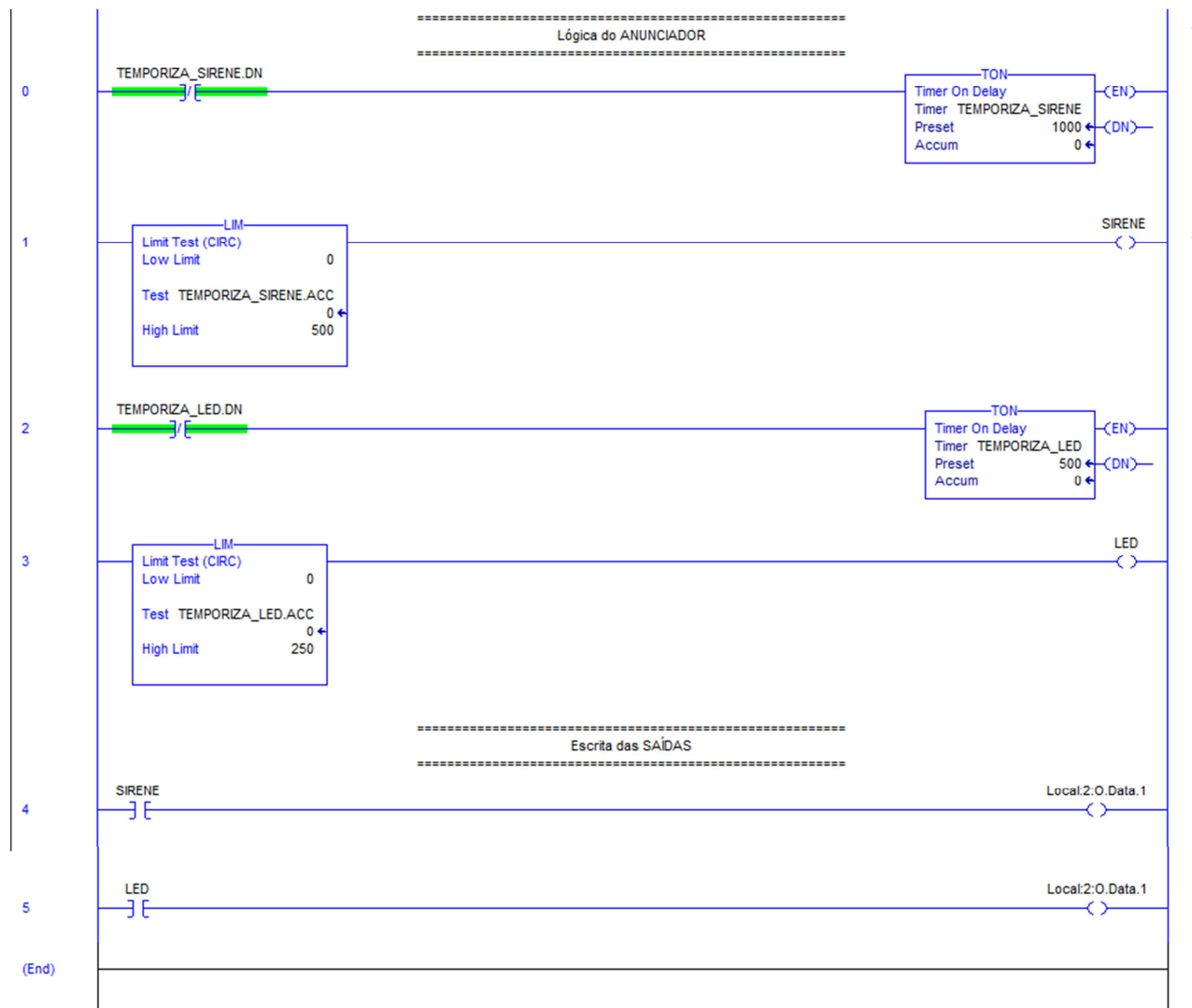


Figura 16. Rotina Anunciador

## 2.2 Exemplo 2: Controle de Semáforo

Neste segundo exemplo será desenvolvido um sistema de controle para um semáforo de trânsito baseado em rotinas escritas nas linguagens Ladder, Texto Estruturado e SFC. Para o desenvolvimento deste exemplo os seguintes pontos devem ser considerados:

- O CP deverá controlar dois semáforos (A e B) utilizados para a sinalização de um cruzamento simples de duas vias, podendo, portanto, ser representado por quatro estados como mostrado na tabela 5.

Tabela 5. Descrição dos estados dos semáforos

Semáforo	Estado dos semáforos			
	1	2	3	4
A	Verde	Amarelo	Vermelho	Vermelho
B	Vermelho	Vermelho	Verde	Amarelo

- Deverá ser possível alterar os tempos de permanência em cada estado, como mostrado na tabela 6, de forma a alterar a prioridade de tráfego para cada uma das vias. Para alterar essa prioridade serão utilizadas duas chaves existentes no kit didático. No aplicativo para controle dos semáforos CP será definida uma variável inteira, denominada *Prioridade*, que irá assumir valores entre 0 e 2 para cada uma das quatro possíveis combinações de posições das chaves, conforme indicado na tabela 6. Para *Prioridade* igual a 0 (chaves nas posições 00 e 11) os dois semáforos possuem a mesma prioridade. Já para *Prioridade* igual a 1 (chaves em 01) a preferência será do semáforo A, enquanto que para o valor 2 a preferência será do B.

Tabela 6. Tempos de permanência em cada estado

Chaves	Prioridade	Tempos (seg) de cada estado			
		1	2	3	4
00	0	15	5	15	5
01	1	20	5	10	5
10	2	10	5	20	5
11	0	15	5	15	5

A figura 17 mostra a ligação dos cartões de entradas e saídas digitais para o controle do semáforo de acordo com a especificação proposta.

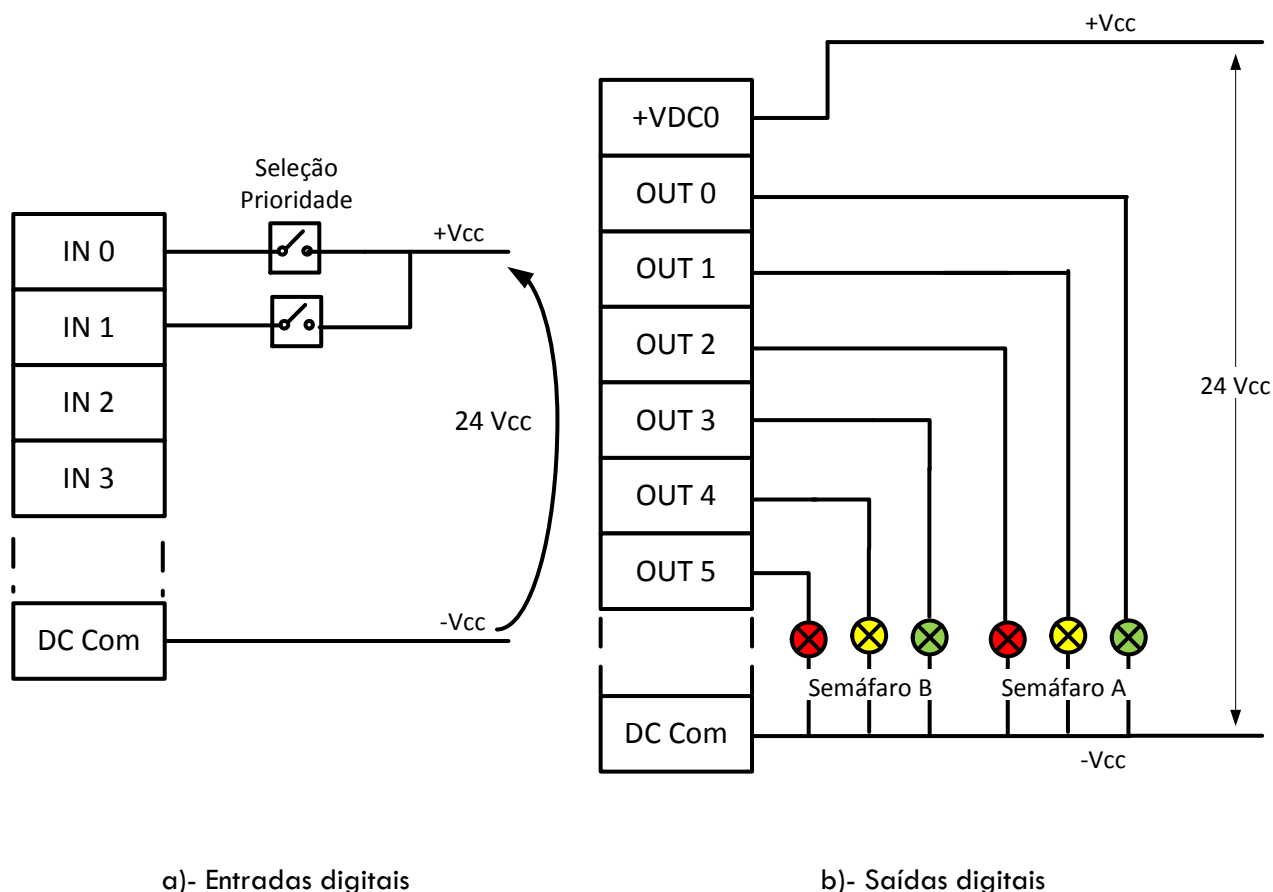


Figura 17- Conexão das entradas e saídas digitais

A figura 18 mostra a Estrutura e a Base de dados do aplicativo desenvolvido. Observe que existe uma única tarefa (*Tarefa*), do tipo contínua, com um único programa (*Processo\_Semaforo*) constituído por 3 rotinas: *Principal*; *Ajuste\_Temp\_Prioridade*; *Semaforo*. A seguir são apresentadas cada uma dessas rotinas.

#### a) Rotina *Principal*:

De forma análoga ao procedimento utilizado no exemplo 1, a rotina *Principal*, escrita em linguagem Ladder, possui três blocos *Leitura das Entradas*, *Chamada das Rotinas*, *Escrita das Saídas* (vide figura 19). No primeiro bloco são lidas as duas chaves que definem a prioridade e atribuído, através da instrução *MOV*, o valor conveniente para a variável *Prioridade*. No segundo bloco são chamadas as outras duas rotinas, enquanto que no terceiro as variáveis internas correspondentes ao estado de cada uma das seis lâmpadas dos semáforos, geradas na rotina *Semaforo*, são escritas nas saídas digitais que energizam essas lâmpadas.

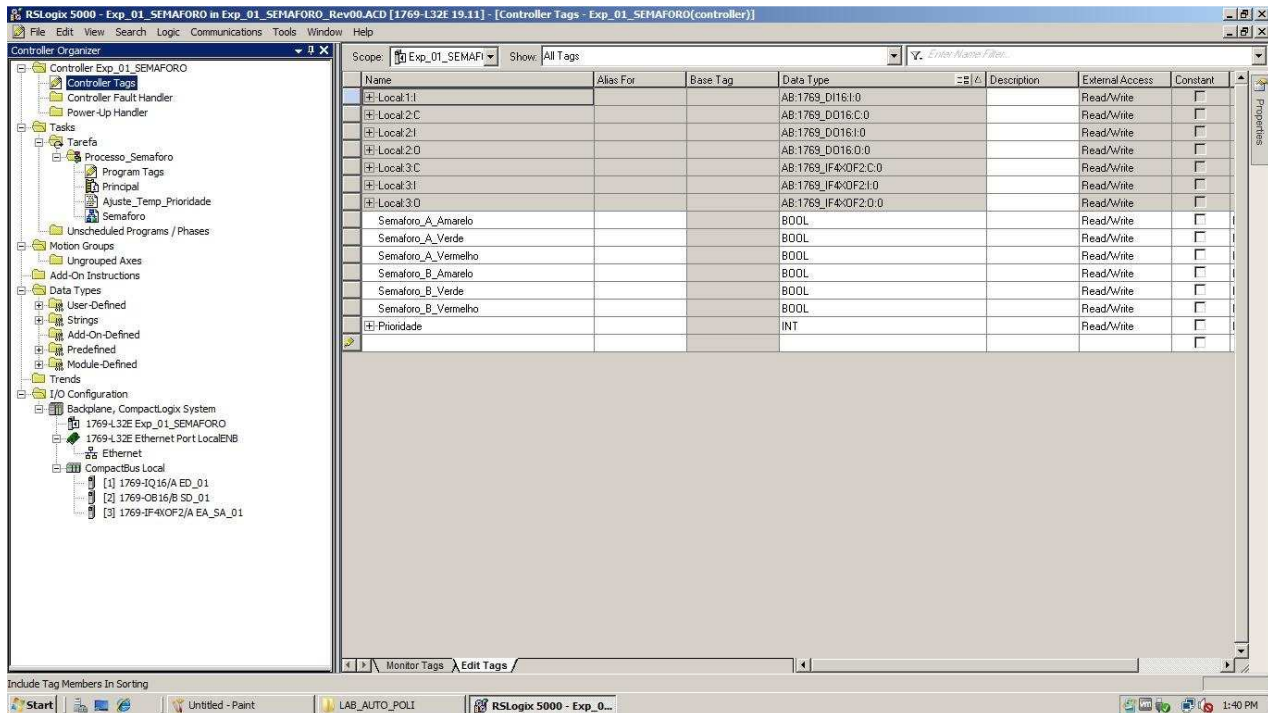


Figura 18. Estrutura do software e Base de dados

## b) Rotina Ajuste\_Temp\_Prioridade:

Esta rotina, cujo código foi escrito na linguagem Texto Estruturado, é mostrada na figura 20. Ela é constituída por uma única instrução (CASE) que define o tempo de permanência em cada estado, através do parâmetro *Action\_00x.PRE*, em função do valor atribuído para a variável *Prioridade*.

## c) Rotina Semaforo:

A figura 21 apresenta o código para essa rotina, escrito em SFC, o qual é constituído pelos quatro estados indicados na tabela 6. Cada estado possui uma única ação do tipo L (limitada no tempo). A transição em cada estado ocorre quando o tempo de duração da ação (variável *Action\_00x.T*) ultrapassa o parâmetro *Action\_00x.PRE* definido na rotina *Ajuste\_Temp\_Prioridade*.

# 1. Familiarização com o Controlador Programável

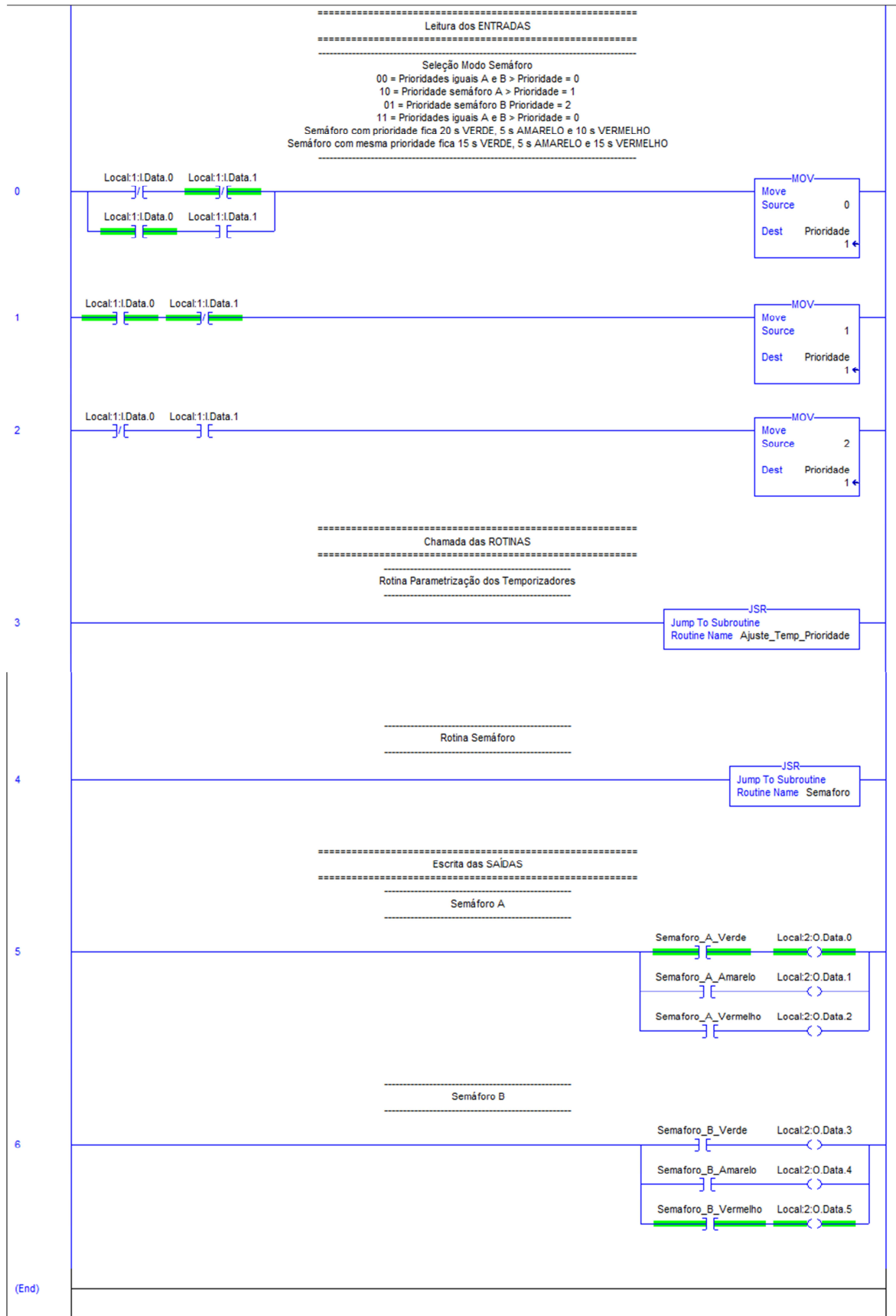
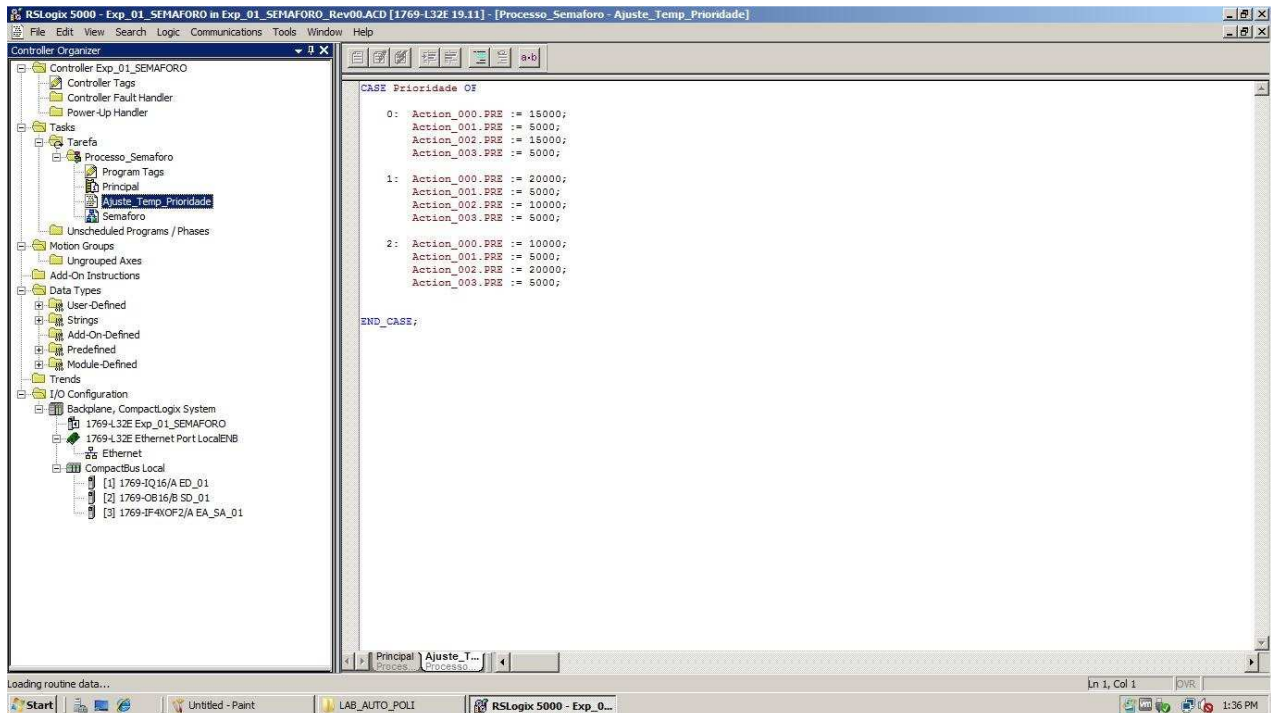


Figura 19. Código da Rotina Principal

# 1. Familiarização com o Controlador Programável



```
CASE Prioridade OF
```

```
0: Action_000.PRE := 15000;  
Action_001.PRE := 5000;  
Action_002.PRE := 15000;  
Action_003.PRE := 5000;
```

```
1: Action_000.PRE := 20000;  
Action_001.PRE := 5000;  
Action_002.PRE := 10000;  
Action_003.PRE := 5000;
```

```
2: Action_000.PRE := 10000;  
Action_001.PRE := 5000;  
Action_002.PRE := 20000;  
Action_003.PRE := 5000;
```

```
END_CASE;
```

Figura 20 Código da rotina *Ajuste\_Temp\_Prioridade*

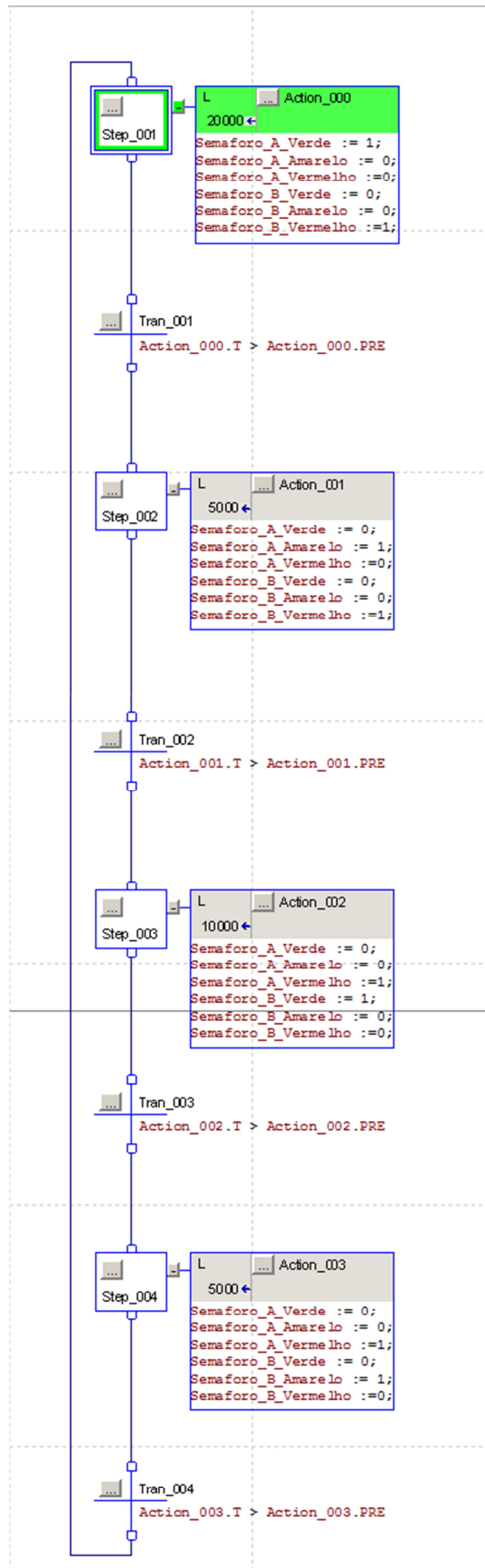


Figura 21. Código da rotina Semaforo

### 3. DIRETRIZES PARA ELABORAÇÃO DO RELATÓRIO

#### 3.1 Implementação dos sistemas: a) Partida de Pórtico com Alarme; b) Controle de Semáforo

- a) Implemente no software RSLogix 5000 os dois programas exemplos apresentados nesta apostila.
- b) Utilizando o kit didático e os demais componentes disponíveis na bancada teste o correto funcionamento desses dois exemplos.
- c) Salve todas as telas necessárias e utilize-as no relatório para documentar detalhadamente todo o desenvolvimento realizado.

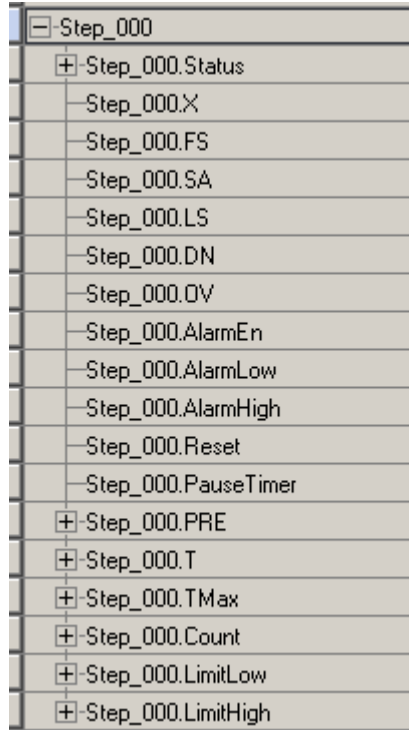
#### 3.2 Desenvolvimento do sistema Controle de Esteira com chave estrela-triângulo

- a) Leia a especificação para o sistema *Controle de Esteira com Chave Estrela-Triângulo* apresentada na apostila referente à experiência 2 – *Controle de Esteira Transportadora*, disponível no Moodle.
- b) Desenvolva e implemente no software RSLogix 5000, instalado nos computadores da Sala Energia, um aplicativo para controle desse sistema exemplo que atenda à especificação apresentada.
- c) Salve todas as telas e elabore uma documentação detalhada do aplicativo desenvolvido, a qual também deverá ser apresentada neste primeiro relatório.
- d) O código desenvolvido deverá ser trazido para a experiência 2, onde, como primeira atividade dessa aula, seu funcionamento deverá ser testado e avaliado.



## Anexo

- Linguagem SFC: TAGs associadas com um STEP



The image shows a screenshot of a software interface displaying a tree view of tags for a step named 'Step\_000'. The tree is expanded to show the following tags:

- Step_000
+ Step_000.Status
- Step_000.X
- Step_000.FS
- Step_000.SA
- Step_000.LS
- Step_000.DN
- Step_000.OV
- Step_000.AlarmEn
- Step_000.AlarmLow
- Step_000.AlarmHigh
- Step_000.Reset
- Step_000.PauseTimer
+ Step_000.PRE
+ Step_000.T
+ Step_000.TMax
+ Step_000.Count
+ Step_000.LimitLow
+ Step_000.LimitHigh

### SFC\_STEP Structure

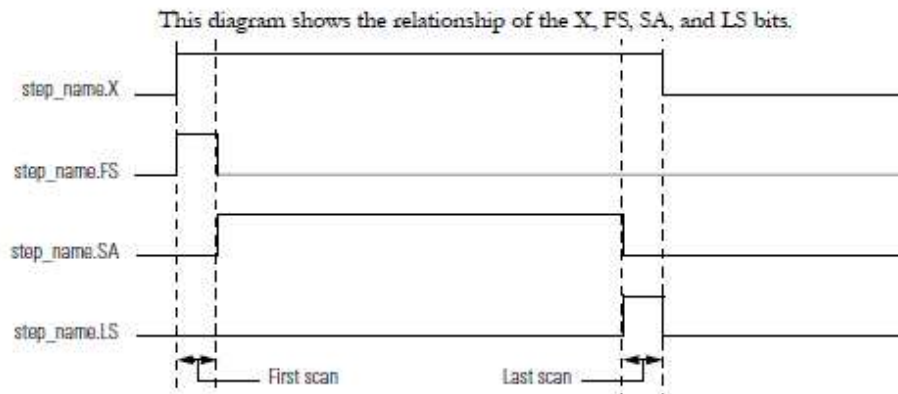
Each step uses a tag to provide information about the step. Access this information via either the Step Properties dialog box or the Monitor Tags tab of the Tags window.

If you want to	Then check or set this member	Data type	Details
Determine how long a step has been active (milliseconds)	T	DINT	When a step becomes active, the Timer (T) value resets and then starts to count up in milliseconds. The timer continues to count up until the step goes inactive, regardless of the Preset (PRE) value.
Flag when the step has been active for a specific length of time (milliseconds)	PRE	DINT	Enter the time in the Preset (PRE) member. When the Timer (T) reaches the Preset value, the Done (DN) bit turns on and stays on until the step becomes active again.  As an option, enter a numeric expression that calculates the time at runtime.
	DN	BOOL	When the Timer (T) reaches the Preset (PRE) value, the Done (DN) bit turns on and stays on until the step becomes active again.
Flag if a step did not execute long enough	LimitLow	DINT	Enter the time in the LimitLow member (milliseconds).  <ul style="list-style-type: none"> <li>• If the step goes inactive before the Timer (T) reaches the LimitLow value, the AlarmLow bit turns on.</li> <li>• The AlarmLow bit stays on until you reset it.</li> <li>• To use this alarm function, turn on (check) the AlarmEnable (AlarmEn) bit.</li> </ul> As an option, enter a numeric expression that calculates the time at runtime.
	AlarmEn	BOOL	To use the alarm bits, turn on (check) the AlarmEnable (AlarmEn) bit.
	AlarmLow	BOOL	If the step goes inactive before the Timer (T) reaches the LimitLow value, the AlarmLow bit turns on.  <ul style="list-style-type: none"> <li>• The bit stays on until you reset it.</li> <li>• To use this alarm function, turn on (check) the AlarmEnable (AlarmEn) bit.</li> </ul>

If you want to	Then check or set this member	Data type	Details
Flag if a step is executing too long	LimitHigh	DINT	<p>Enter the time in the LimitHigh member (milliseconds).</p> <ul style="list-style-type: none"> <li>If the Timer (T) reaches the LimitHigh value, the AlarmHigh bit turns on.</li> <li>The AlarmHigh bit stays on until you reset it.</li> <li>To use this alarm function, turn on (check) the AlarmEnable (AlarmEn) bit.</li> </ul> <p>As an option, enter a numeric expression that calculates the time at runtime.</p>
	AlarmEn	BOOL	To use the alarm bits, turn on (check) the AlarmEnable (AlarmEn) bit.
	AlarmHigh	BOOL	<p>If the Timer (T) reaches the LimitHigh value, the AlarmHigh bit turns on.</p> <ul style="list-style-type: none"> <li>The bit stays on until you reset it.</li> <li>To use this alarm function, turn on (check) the AlarmEnable (AlarmEn) bit.</li> </ul>
Do something while the step is active (including first and last scan)	X	BOOL	<p>The X bit is on the entire time the step is active (executing).</p> <p>Typically, we recommend that you use an action with a <i>N Non-Stored</i> qualifier to accomplish this.</p>
Do something one time when the step becomes active	FS <sup>(1)</sup>	BOOL	<p>The FS bit is on during the first scan of the step.</p> <p>Typically, we recommend that you use an action with a <i>P1 Pulse (Rising Edge)</i> qualifier to accomplish this.</p>
Do something while the step is active, <i>except</i> on the first and last scan	SA	BOOL	The SA bit is on when the step is active except during the first and last scan of the step.
Do something one time on the last scan of the step	LS <sup>(1)</sup>	BOOL	<p>The LS bit is on during the last scan of the step.</p> <p>Use this bit only if on the Controller Properties dialog box, SFC Execution tab, you set the Last Scan of Active Step to Don't Scan or Programmatic reset.</p> <p>Typically, we recommend that you use an action with a <i>P0 Pulse (Falling Edge)</i> qualifier to accomplish this.</p>
Determine the target of an SFC Reset (SFR) instruction	Reset	BOOL	<p>An SFC Reset (SFR) instruction resets the SFC to a step or stop that the instruction specifies.</p> <ul style="list-style-type: none"> <li>The Reset bit indicates to which step or stop the SFC will go to begin executing again.</li> <li>Once the SFC executes, the Reset bit clears.</li> </ul>
Determine the maximum time that a step has been active during any of its executions	TMax	DINT	Use this for diagnostic purposes. The controller clears this value only when you select the Restart Position of Restart at initial step and the controller changes modes or experiences a power cycle.
Determine if the Timer (T) value rolls over to a negative value	OV	BOOL	Use this for diagnostic purposes.

If you want to	Then check or set this member	Data type	Details	
Determine how many times a step has become active	Count	DINT	<p>This is not a count of scans of the step.</p> <ul style="list-style-type: none"> <li>The count increments each time the step becomes active.</li> <li>It increments again only after the step goes inactive and then active again.</li> <li>The count resets only if you configure the SFC to restart at the initial step. With that configuration, it resets when the controller changes from program mode to run mode.</li> </ul>	
Use one tag for the various status bits of this step	Status	DINT	<b>For this member</b>	
			<b>Use this bit</b>	
			Reset	22
			AlarmHigh	23
			AlarmLow	24
			AlarmEn	25
			OV	26
			DN	27
			LS	28
			SA	29
FS	30			
X	31			

<sup>(1)</sup> The FS and LS bits are only active during a step's execution. Once a step finishes executing the code within its actions, the FS and/or LS bits are reset. If you reference either of these bits in code outside of the SFC routine in a different part of the project, the bits are always cleared (0).



• Linguagem SFC: Qualificador para uma Action

To change when an action starts or stops, assign a different qualifier.

**Choose a Qualifier for an Action**

If you want the action to	And	Then assign this qualifier	Which means
Start when the step is activated	Stop when the step is deactivated	N	Non-Stored
	Execute only once	P1	Pulse (Rising Edge)
	Stop before the step is deactivated or when the step is deactivated	L	Time Limited
	Stay active until a Reset action turns off this action	S	Stored
	Stay active until a Reset action turns off this action	SL	Stored and Time Limited
	Or a specific time expires, even if the step is deactivated		
Start a specific time after the step is activated and the step is still active	Stop when the step is deactivated	D	Time Delayed
	Stay active until a Reset action turns off this action	DS	Delayed and Stored
Start a specific time after the step is activated, even if the step is deactivated before this time	Stay active until a Reset action turns off this action	SD	Stored and Time Delayed
Execute once when the step is activated	Execute once when the step is deactivated	P	Pulse
Start when the step is deactivated	Execute only once	P0	Pulse (Falling Edge)
Turn off (reset) a stored action	—————▶	R	Reset

- S Stored
- SL Stored and Time Limited
- DS Delayed and Stored
- SD Stored and Time Delayed