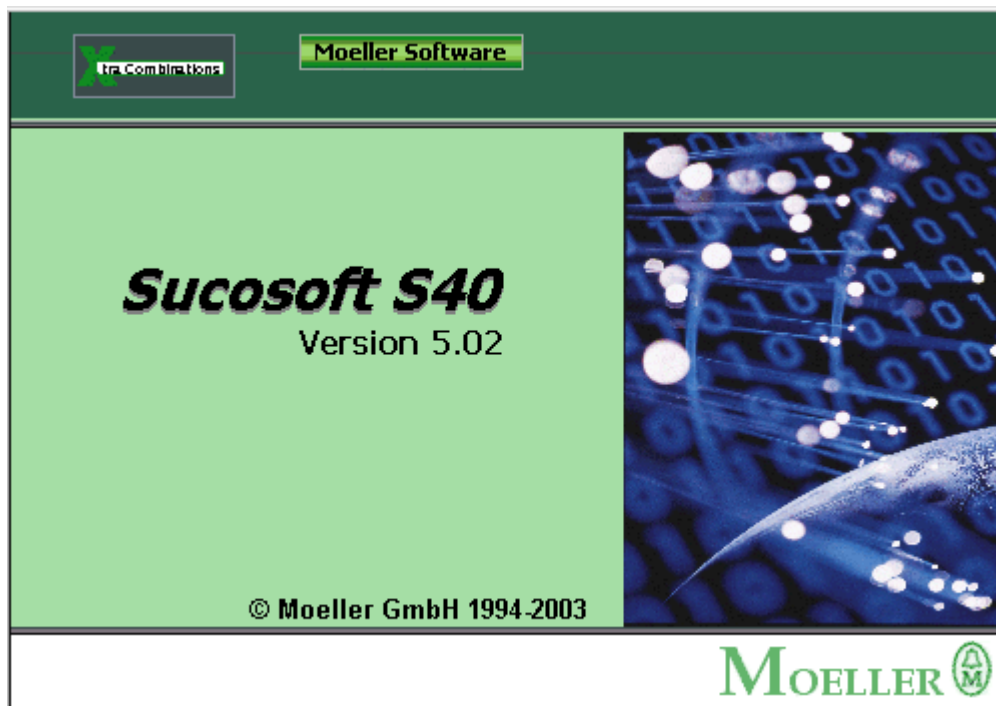


SUCOSOFT S40 V5.02



GUIA RÁPIDO DE PROGRAMAÇÃO DOS CLP'S LINHAS PS4-200, PS4-300 e PS416

INTRODUÇÃO

No mercado pode-se encontrar uma variedade de diferentes linguagens e sistemas de programação. Para cada tipo de linguagem são necessários dispendiosos treinamentos para todos os tipos de controladores, tanto para a linguagem de programação como para o sistema operacional.

O usuário não consegue trocar unidades de programas entre diferentes sistemas, porque as linguagens de programação não são compatíveis entre si. Não existe também um formato de arquivos para troca de programas.

Não é possível a reutilização de funções programadas sem que seja necessária uma reprogramação, pois o modelo de dados destes CLP's necessita de endereços físicos.

Pelo exposto acima a utilização de um controlador de outro fabricante não era viável por causa do alto custo.

O principal objetivo da IEC1131.3 é reduzir, se não eliminar, todos os problemas citados acima, criando uma estruturação e forma de programação padrão, de tal forma que o usuário não se torne dependente do fabricante.

CAPÍTULO 1 REQUISITOS E OBJETIVOS

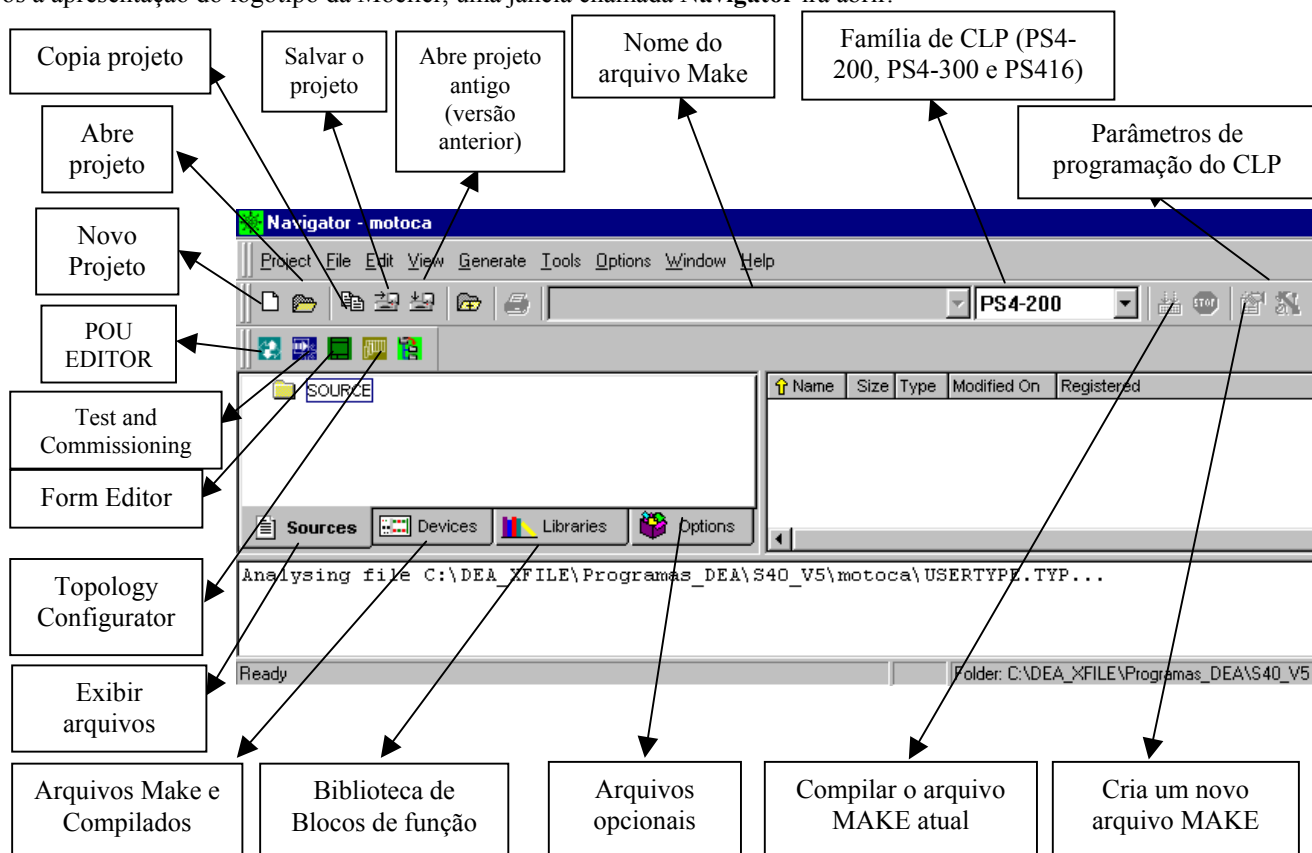
Antes de programar é necessário um conhecimento básico da linha de controladores lógicos programáveis da Moeller.

Este manual de programação rápida destina-se a auxiliar o usuário a iniciar-se na programação do Software S40 Versão 5.02 em ambiente para Windows.

CAPÍTULO 2 INICIANDO O SOFTWARE

Através do menu INICIAR (START), faça a chamada do S40.

Após a apresentação do logotipo da Moeller, uma janela chamada **Navigator** irá abrir.



POU Editor - é usado para escrever programas e blocos de função. O programa pode ser escrito em lista de instruções, diagrama de contatos e blocos lógicos e também pode-se alterar as diferentes linguagens de acordo com as necessidades e preferências.

TOPOLOGY CONFIGURATOR - é usado para definir a configuração do hardware, definindo a estrutura de toda a rede.

TEST & COMMISSIONING - é o ambiente utilizado para realizar todas as comunicações entre o PC e o CLP. Neste é possível transferir o programa para o CLP, parar ou colocar o CLP em marcha, modificar online o programa e testar as seqüências lógicas.


FORM EDITOR - é usado para visualizar e editar a documentação do programa. É possível através deste definir folhas padrão com logotipos e desenhos da empresa.

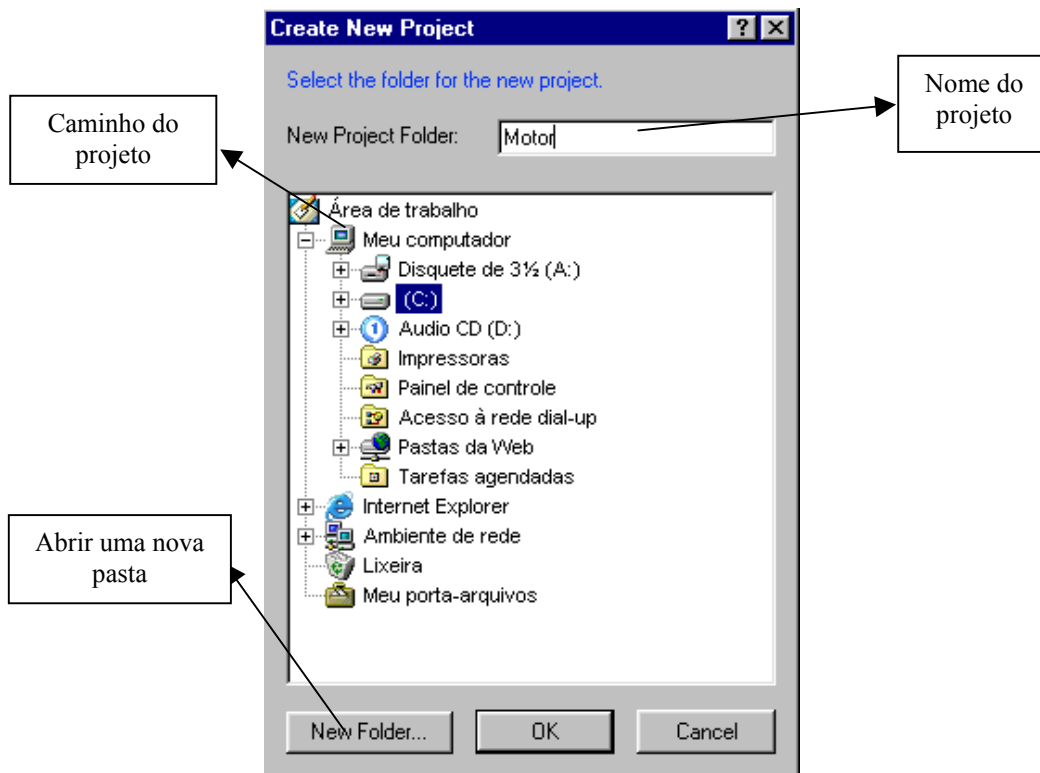
Os passos necessários para a criação de um programa são:

- **Criar um novo projeto**
- **Configurar o Hardware**
- **Criar os arquivos fontes necessários (POU)**
 - Criar as Variáveis
 - Realizar a lógica do programa
- **Criar um novo Make**
- **Compilar o programa**
- **Realizar os testes e comissionamento**
 - Transferir o programa para o CLP
 - Rodar o programa
 - Testar o programa
- **Compactar e deixar o programa fonte na memória Flash para possibilitar o UP-load.(opcional)**

CAPÍTULO 3

INICIANDO UM NOVO PROJETO

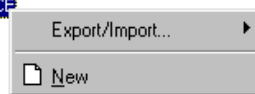
- Clique no menu Project do Navigator.
- Selecione no menu Project → New...ou clique em 



- Selecione o caminho do novo projeto, ou crie um novo diretório para guardar seus projetos.
- No nosso caso digite MOTOR e clique no botão OK.
- Todos os diretórios necessários à confecção de um programa já estão criados.
- Na tela do Navigator irá aparecer o diretório Source dentro da pasta Sources.
- Caso seja desejado poderemos criar outros diretórios para organizar nossos programas. Para isto basta clicar em cima do

SOURCE

diretório Source com o botão da direita do mouse:



clcando em New poderemos criar



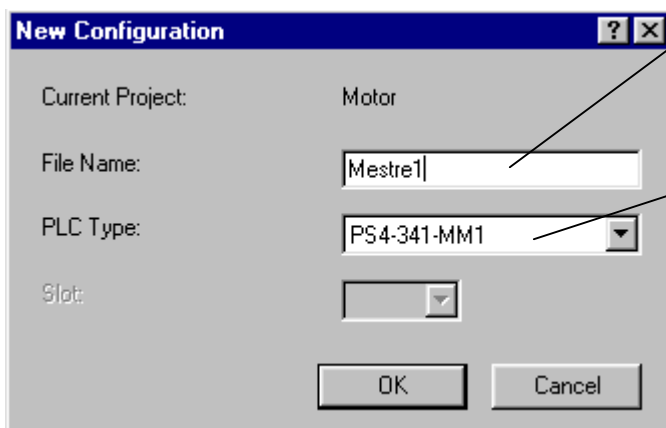
um novo diretório abaixo de SOURCE. Neste caso vamos chamar este sub-diretório de MOTORC2.

- Se deixarmos o diretório MOTORC2 selecionado todos os nossos fontes serão gravados nele. Neste exemplo vamos deixar os arquivos fonte gravados no SOURCE, para isto clique no diretório SOURCE e deixe-o selecionado.
- Um mesmo projeto pode conter vários softwares e configurações diferentes.

CAPÍTULO 4

INICIANDO UMA CONFIGURAÇÃO

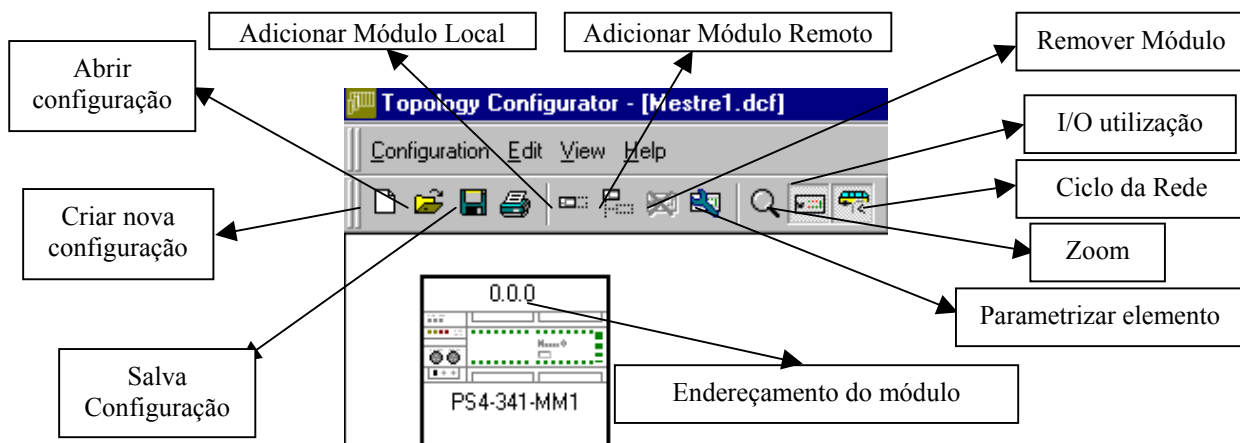
- O segundo passo é configurar o hardware. Este deve ser especificado utilizando o Topology Configurator.
- Assim que a janela do Topology Configurator abrir selecionamos : Configuration → New...
- No campo File Name , digite o seguinte nome: Mestre1 , e em CLP Type clicamos em cima do nome apresentado e selecionamos a CPU que estamos utilizando.



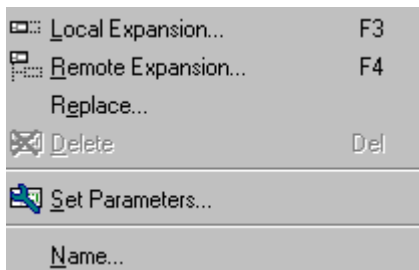
Nome do arquivo de configuração

Modelo da CPU

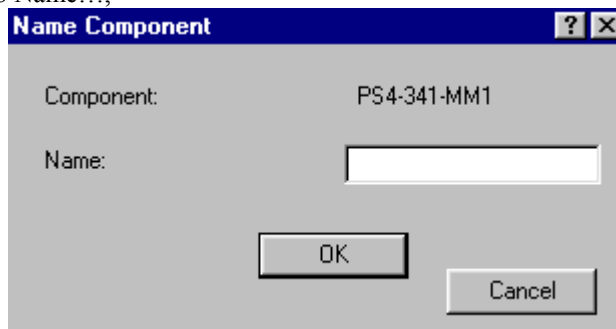
- Através do item Edit podemos escolher agregar a esta CPU módulos de expansão remota, expansão local ou configurar dados de rede da CPU, ou então através dos ícones na própria tela, estes dados estão mais detalhados nos Manuais.



- Podemos ainda dar um nome para cada elemento da nossa configuração, para fazer isto basta clicar em cima do elemento com o botão da direita do mouse.



- E agora selecionamos a opção Name...,



- Colocamos um nome para o nosso elemento e clicamos em OK.
- Salvamos a nossa configuração.
- Clicamos em Configuration → Exit, para sair do Topology Configurator. O arquivo com nossa configuração irá aparecer dentro do SOURCE.

Name	Size	Type	Modified On	Registered
Mestre1	1 KB	Topology (PS4-300)	12/12/03 12:15:08	<input checked="" type="checkbox"/>

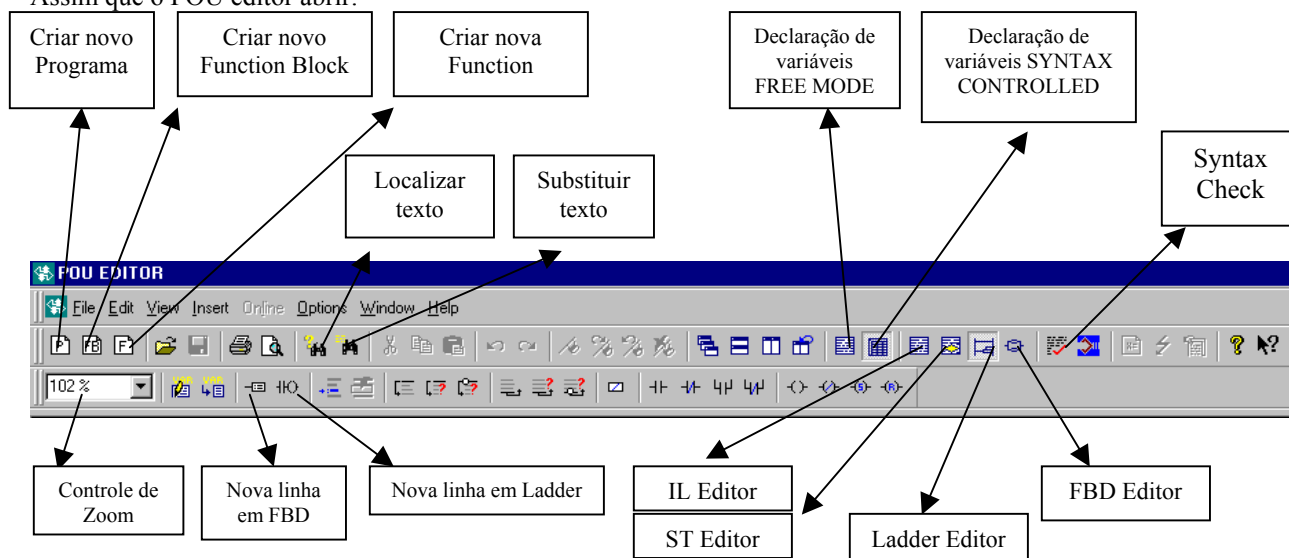
CAPÍTULO 5

COMEÇANDO A PROGRAMAR

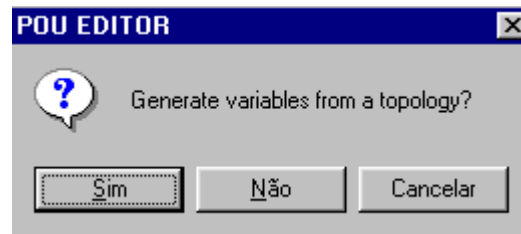
- No Navigator selecionamos agora a família da CPU desejada, no nosso caso escolheremos a família PS4-300.

PS4-300

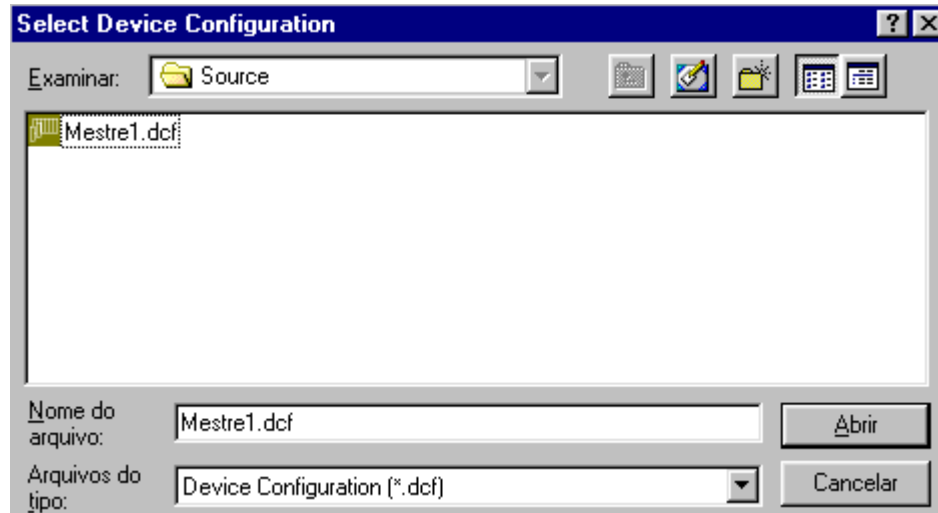
- Selecione agora o item POU-Editor.
- Assim que o POU editor abrir:



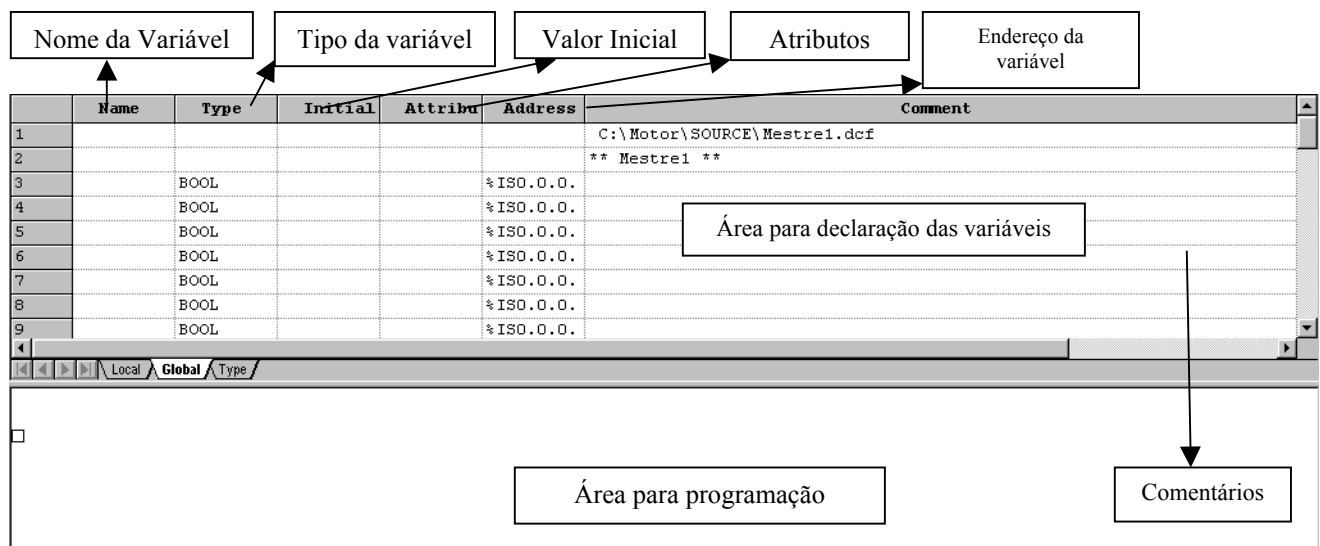
- Seleccionamos agora a opção Criar novo Programa.



- O software tem condições de gerar sozinho os endereços dos I/Os à partir da topologia criada anteriormente. Vamos seleccionar Sim.



- Agora escolhemos a topologia correspondente à este programa, no nosso caso será o arquivo Mestre1.dcf



- A área para declaração das variáveis pode ser utilizada em dois modos distintos, o Syntax Controlled e o FREE MODE, na figura acima estamos utilizando o Syntax Controlled que é a melhor opção quando se é iniciante. Se o seu programa não estiver no Syntax Controlled basta seleccionar esta opção no seu menu de ferramentas.
- Lembre-se que as variáveis que o programa importa de sua topologia estão na pasta Global!
 - A área de programação possui quatro formas de programação disponíveis: a programação com blocos lógicos (FBD), a programação em Ladder (LD), a programação em Lista de instruções (IL) e a programação estruturada (ST).
 - Através do menu Windows, podemos ainda seleccionar como iremos visualizar esta tela. Se seleccionarmos Windows → Rotate, faremos um giro na tela, a declaração de variáveis irá para o lado esquerdo da tela, e o programa permanecerá no lado direito da tela. Através das opções Windows → Tile Vertically e Windows → Tile Horizontally,

conseguimos arrumar as telas de programação. Isso é utilizado quando desejamos trabalhar e visualizar mais de um programa fonte de cada vez

DECLARANDO VARIÁVEIS

As variáveis geralmente são declaradas antes de serem usadas. A declaração consiste do nome da variável e o tipo de dado. Dependendo do tipo de dado atribuído a uma variável existe uma faixa de valores admissíveis. Por exemplo: o tipo de dado SINT possui 8 bits, e pode assumir valores entre -128 a 127, já o tipo USINT que também possui 8 bits, pode assumir valores entre 0 e 255.

Os tipos de dados elementares definidos pela IEC1131-3 são:

Binary	um ou vários bits em grupo de operações binária.
Integer	todos os números inteiros.
Real	números com ponto flutuante. (Somente PS4-300 e PS416)
String	conjunto de caracteres alfanuméricos.
Date/Time	datas, hora do dia e data e hora.

Junto com o tamanho da variável e sua faixa de valores, os tipos de dados também decidem que tipos de operação são permitidas para estas variáveis, por exemplo uma variável do tipo Integer só pode ser usado em funções aritméticas, já uma variável do tipo Binary só pode ser usada em funções lógicas.

Isto soluciona o problema do operador que testa diferentes tipos de dados na aplicação. Frequentes erros podem ser facilmente reconhecidos com uma rápida observação do programa. Temos os seguintes tipos de dados descritos a seguir:

Palavra Chave	Tipo de dado.	bits
BOOL	número booleano; aceita valores de 0 ou 1.	1
SINT	inteiro curto; com faixa de valores de -128 a 127.	8
INT	inteiro; com faixa de valores de -32768 a 32767.	16
USINT	inteiro curto sem sinal; com faixa de valores de 0 a 255.	8
UINT	inteiro sem sinal; com faixa de valores de 0 a 65535.	16
DINT	inteiro duplo, com faixa de valores de -2^{31} a 2^{31}	32 (Somente PS4-341 e PS416)
UDINT	inteiro duplo sem sinal, com faixa de valores de 0 a 2^{32}	32 (Somente PS4-341 e PS416)
TIME	duração	-
DATE	data.	-
STRING	Conjunto de caracteres com tamanho variável.	variável
BYTE	seqüência de 8 bits.	8
WORD	seqüência de 16 bits.	16
REAL	Número em ponto flutuante, com valores de $\pm 10^{-38}$ a $\pm 10^{38}$	32 (Somente PS4-341 e PS416)

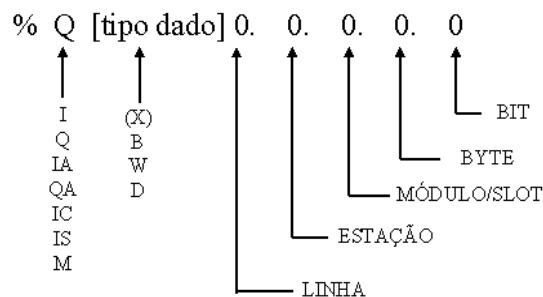
Ou seja, se desejamos fazer lógicas Booleanas devemos utilizar variáveis do tipo: BOOL, Byte ou Word; se for necessário efetuar funções aritméticas utilizamos as variáveis do tipo: INT, SINT, USINT, etc.

Nesta versão do S40, os I/Os já estão declarados no campo Global.

	Name	Type	Initial	Attribu	Address	Comment
1						C:\Motor\SOURCE\Mestre1.def
2						** PS4-341-MM1 **
3		BOOL			%I0.0.0.	
4		BOOL			%I0.0.0.	
5		BOOL			%I0.0.0.	
6		BOOL			%I0.0.0.	
7		BOOL			%I0.0.0.	
8		BOOL			%I0.0.0.	
9		BOOL			%I0.0.0.	
Local Global Type						

- Repare que o software coloca, automaticamente, no lado dos comentários (Comment) o nome do módulo a que os I/Os pertencem. Se tivéssemos outros módulos, conectados à CPU, o endereço deles apareceriam também, com seu respectivo código no comentário.
- Agora é necessário declarar as nossas entradas / saídas, primeiramente vamos localizar nosso endereço. Suponha que desejamos utilizar a primeira entrada da CPU. O seu endereço será %I0.0.0.0, veja esquema abaixo:

ENDEREÇAMENTO DIRETO DE VARIÁVEIS



Linha: é o número da linha onde o escravo está conectado, linha 0 é o rack básico, linha 1 é a linha que sai da CPU principal, linhas 2 e 3 são as linha que saem dos módulos de expansão conectados ao mestre.

Estação: Número do escravo na rede Suconet K

Módulo / Slot : Módulo → Número da expansão local conectada ao EM4-2XX ou PS4.

Slot → Número do slot no rack onde o cartão foi encaixado. (PS416)

Byte: Número do byte, se o módulo possuir mais de 16 saídas ou entradas terá 2 bytes, byte 0 e byte 1

Bit: Este é o número do bit, é o número da entrada ou saída propriamente dita, caso a variável declarada seja do tipo Byte ou Word, este número não será apresentado.

I : Entrada Digital

IB : Entrada digital (Byte)

IW : Entrada Digital (Word)

Q : Saída Digital

QB : Saída Digital (Byte)

QW : Saída Digital (Word)

IA : Entrada Analógica

QA : Saída Analógica

IS : Status do módulo



- Queremos uma entrada digital , por isso o endereço começa com %I , a entrada está na CPU (0.0.0) desejamos o byte 0 e é o primeiro bit que me interessa (0.0).
- Escreva no campo “Name” o nome desta variável ,que será **Partida**. No nome das variáveis é proibido utilizar caracteres especiais como : !#\$%&*()-+/?<.>:,|,
- O nome das variáveis podem ter até 30 caracteres , procure usar nomes auto explicativos para facilitar no momento do start-up. Se forem nomes compostos , procure escrever com Underline , por exemplo : **BOMBA_RECALQUE** ou então desta maneira: **BombaRecalque**.
- Vamos agora declarar as demais variáveis necessárias ao funcionamento do nosso programa de estrela triângulo, declare as demais variáveis como segue abaixo:

	Name	Type	Initial Value	Attribute	Address
11	Partida	BOOL			%I0.0.0.0.0
12	Parada	BOOL			%I0.0.0.0.1
13	ContatorPrincipal	BOOL			%Q0.0.0.0.0
14	ContatorEstrela	BOOL			%Q0.0.0.0.1
15	ContatorTriangulo	BOOL			%Q0.0.0.0.2

NOTA: Quando declaramos um Tag e especificamos apenas o tipo, por exemplo BOOL, e não especificamos o endereço o próprio compilador determina que se trata de um Flag por não possuir Saída / Entrada física, então automaticamente reserva uma área de memória para este Tag. Porém se for necessário relacionar este Tag a uma memória conhecida (para comunicação com uma IHM por exemplo) o endereçamento é feito como se segue:

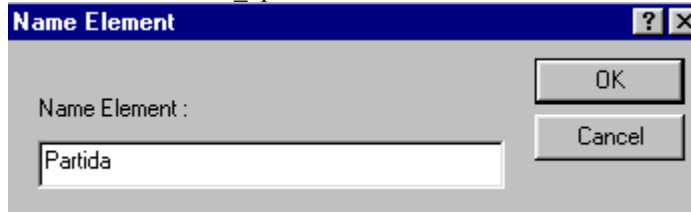
TAG1	BOOL	M0.0.0.0.0
Nome_2	BYTE	MB0.0.0.0
Qualquer_nome	WORD	MW0.0.0.0

ESCREVENDO O PROGRAMA

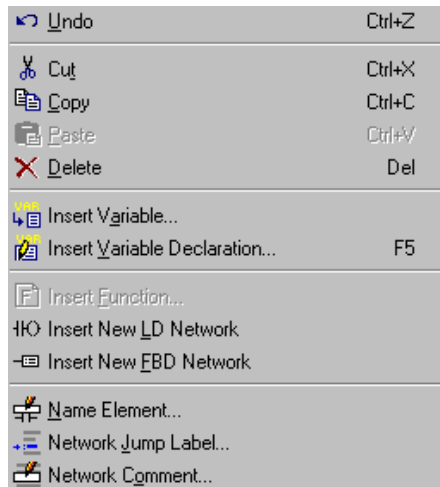
- Primeiramente vamos editar o nosso programa em Ladder, para tanto selecione na Barra de ferramentas do POU Editor a opção Ladder Editor: .
- Neste ponto estamos prontos para editar o nosso programa, vamos editar a linha de comando que liga o ContatorPrincipal.
- Primeiramente selecione a opção Nova Linha em Ladder . Uma nova linha em Ladder irá abrir na área do programa:



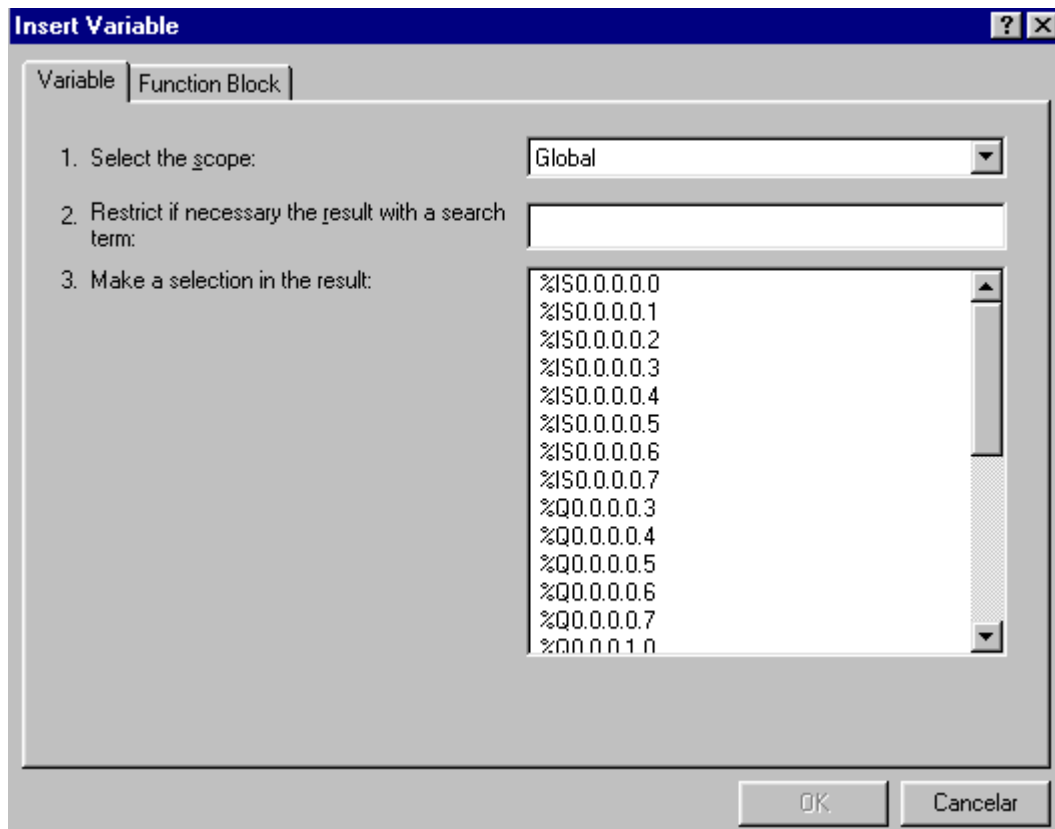
- Para darmos um nome para o contato NA undef_opd basta clicar duas vezes sobre ele:



- Coloque nele o nome Partida.
- É possível também colocar o nome através de um outro procedimento, selecione agora a bobina de saída e clique com o botão direito do mouse:



- Selecione agora a opção Insert Variable...



- Mantenha selecionada a opção Variable, na opção 1 escolha Global, o item 2 é um filtro. Por exemplo, se escrevermos a letra C na opção 2 veremos abaixo todas as variáveis que começam por C.
- Escolha agora a opção ContatorPrincipal.



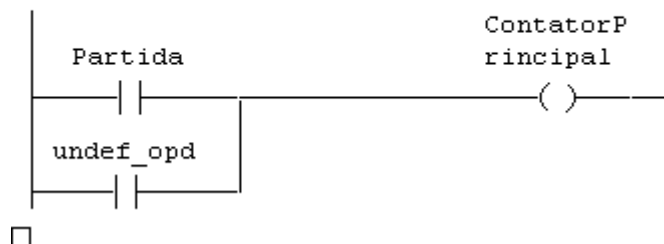
0001




- Agora nós iremos colocar um contato de selo, para tanto selecione o contato da variável Partida.
- Escolha agora a opção contato Paralelo: 4P

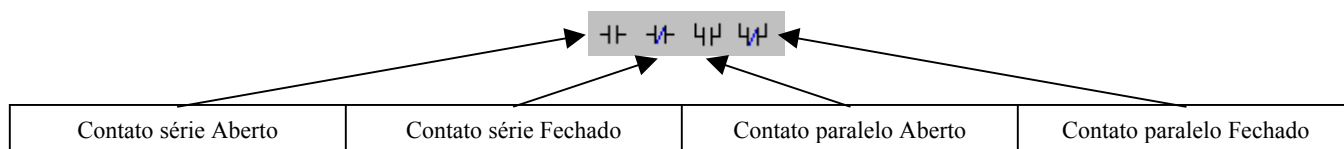


0001



- Agora colocamos neste contato o nome de ContatorPrincipal.
- Só falta colocar na lógica o botão de desliga, para isto basta clicar na linha logo à frente do nosso selo. Neste ponto selecionamos a opção: -V-

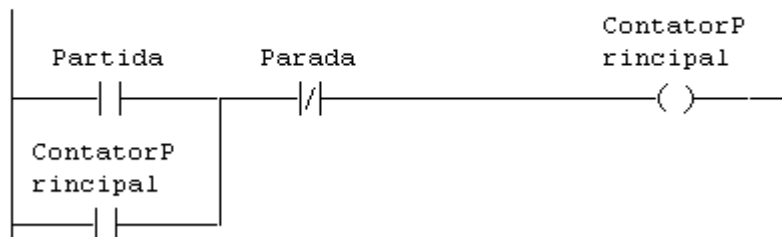
- Logo abaixo podemos ver as quatro opções que existem para inserção de contatos. Se quisermos inverter o estado de um contato já criado , basta seleciona-lo e pressionar a barra de espaço no teclado , ou selecionar a opção: 




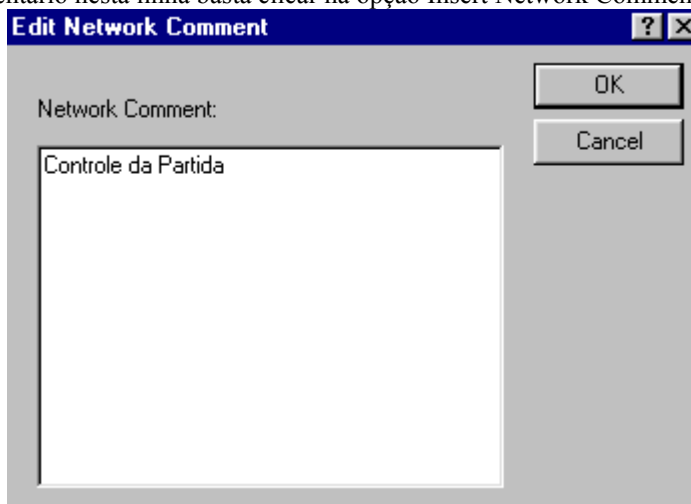
- Logo após o aparecimento do contato colocamos nele o nome da variável , **Parada**.



0001

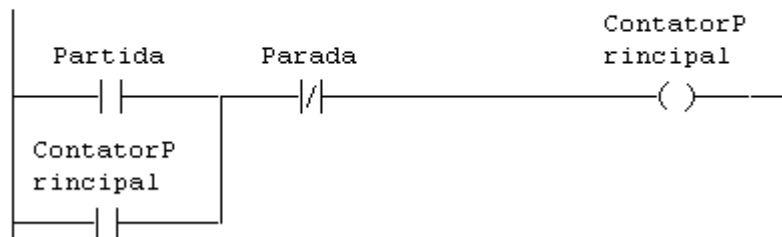


- Para colocarmos um comentário nesta linha basta clicar na opção Insert Network Comment : .



0001

Controle da Partida



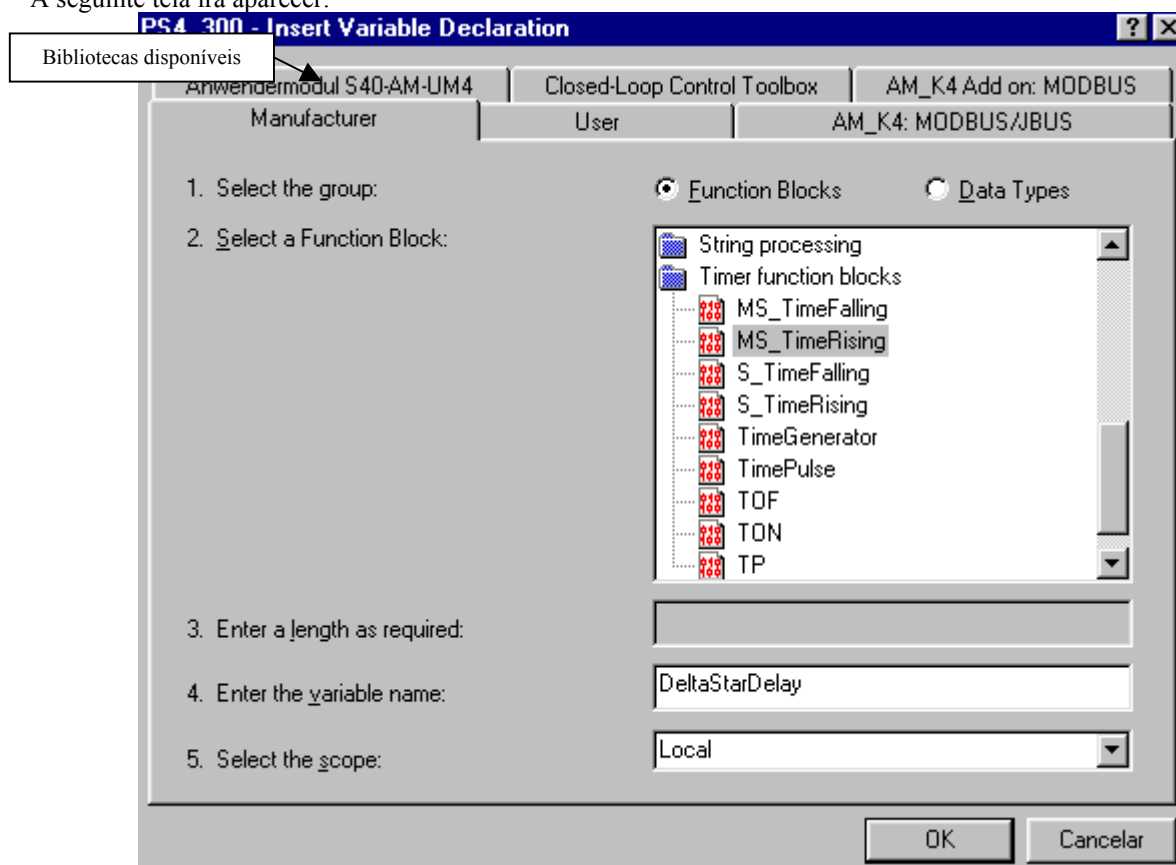
- Agora precisamos ligar os outros contadores. Para fazer a partida Estrela Triângulo são necessários dois temporizadores , um para determinar o tempo da troca entre o contator estrela e o triângulo e outro que espere um contator cair para ativar outro. Mas é necessário declara-los antes de utilizá-los.

DECLARANDO FUNCTION BLOCK's

- Vamos voltar à Área de Declaração de Variáveis, aqui nós declararemos uma nova variável.
- Clique com o botão da direita sobre a área de declaração de variáveis e selecione a opção Insert Variable Declaration... ou pressione F5.



- A seguinte tela irá aparecer:



- Em Select the group deixe ativada a opção Function Block.
- Provavelmente não aparecerão tantas bibliotecas em seu programa, somente a manufacturer (Selecione esta) , e a User. As demais bibliotecas podem ser instaladas posteriormente em seu computador , veja os manuais correspondentes para maiores dados.
- No item 2 selecione o grupo Timer function blocks. Dentro deste grupo escolhemos a opção MS_TimeRising.
- Item 4 serve para inserirmos o nome deste Function Block , neste caso DeltaStarDelay , em nosso programa. E finalmente no item 5 determinamos o escopo desta variável , podemos deixa-la como Global ou local , tanto faz. (Maiores dados sobre variáveis podem ser localizados nos manuais ou no Help do software)
- Para obter maiores informações sobre os temporizadores utilize o Help do S40. Pressione F1 e siga a indicação abaixo, você terá acesso a um help Online de todos os blocos de função standard (Que vem com o S40).

ContentsIndexSearch

Sucosoft

Language elements

Program Structure

Variables and Constants

Variables

Constants

Textual Programming Languages

Graphical Programming Languages

Functions

Function blocks

Summary, Function blocs

Edge detection

Flip-flops

Date and time

Timing FBs

MS_TimeFalling: Off-Delay Timer, Milliseconds

MS_TimeRising: On-Delay Timer, Milliseconds

S_TimeFalling: OffDelay Timer, Seconds

S_TimeRising: On-Delay Tmer, Seconds

TimeGenerator: Clock Generator

TimePulse: Clock Generator

TOF: Off Delay

TON: On Delay

TP: Pulse

Alarm FBs

Counter and comparison FBs

Register operations

Code converters

Array operations

String processing

Sequence control

Communication

Reading and writing the memory card

OS system function blocks

Peripheral access

Conversion functions

Language Extension

Sucosoft language definitions

Working with sucosoft

On-Delay Timer, Milliseconds: MS_TimeRising

Function block prototype

MS_TimeRising

BOOL

Set

OutputControl

BOOL

BOOL

ReSet

ElapsedTime

UINT

BOOL

Hold

UINT

PresetTime

Set

Start condition, rising edge

ReSet

Reset condition

Hold

Time interruption

PresetTime

Time setpoint in milliseconds

OutputControl

Control output

ElapsedTime

Time actual value in milliseconds

Description

Set

ReSet

Hold

OutputControl

(1)

(2)

(3)

(4)

(5)

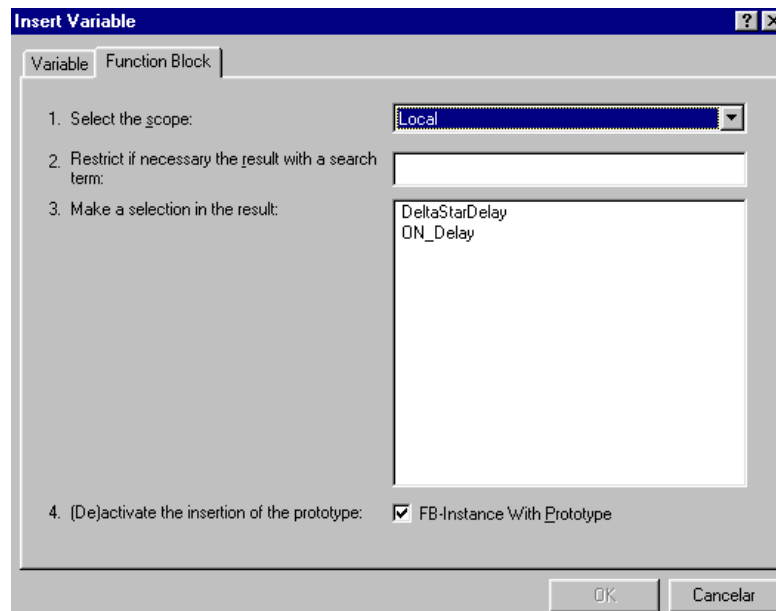
- Vamos declarar ainda mais um temporizador , como o CLP é mais rápido que o contator , caso nós aguardássemos somente o tempo de comutação , a contatora triângulo poderia ligar antes da outra contatora ter tido tempo de desligar.
- O outro temporizador será do mesmo tipo do anterior , MS_TimeRising , desta vez no nome coloque : ON_Delay.
- Nossa declaração ficará deste jeito:

	Name	Type	Initial	Attribu	Address	Comment
1	DeltaStarDelay	MS_TimeRising				
2	ON_Delay	MS_TimeRising				

LocalGlobalType

TERMINANDO O PROGRAMA MOTOR

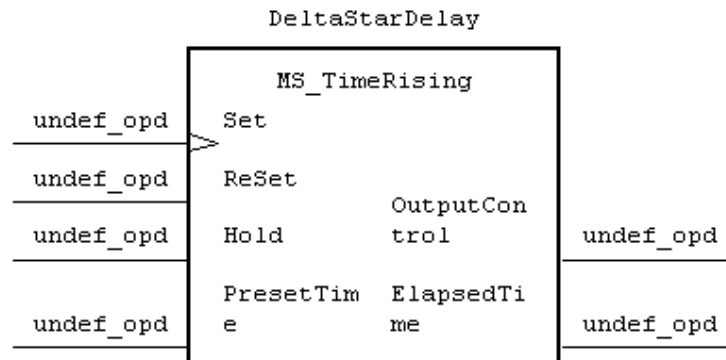
- Vamos agora terminar o nosso programa MOTOR
- Clique com o botão da direita do mouse em algum lugar da Área de Programação, quando a janela abrir, selecione a opção Insert Variable.



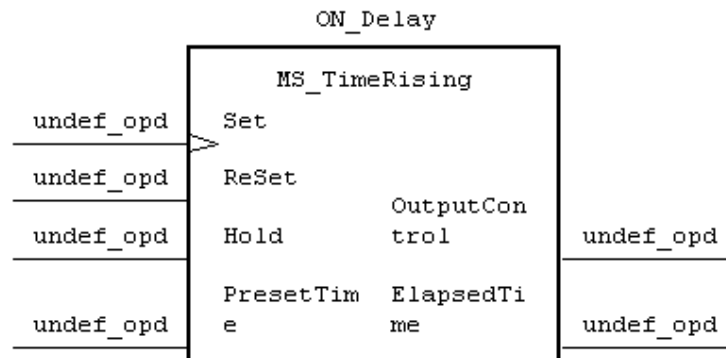
- Na pasta logo acima desta caixa escolhemos que vamos inserir um Function Block , no item 1 escolhemos em que escopo está este Function Block. Se necessário temos uma opção de filtro , opção 2 , como temos nas variáveis.
- No nosso caso escolha o DeltaStarDelay e logo em seguida insira o ON_Delay. O que aparecerá no programa é:



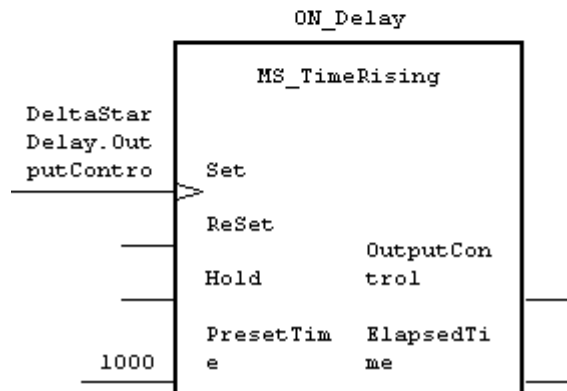
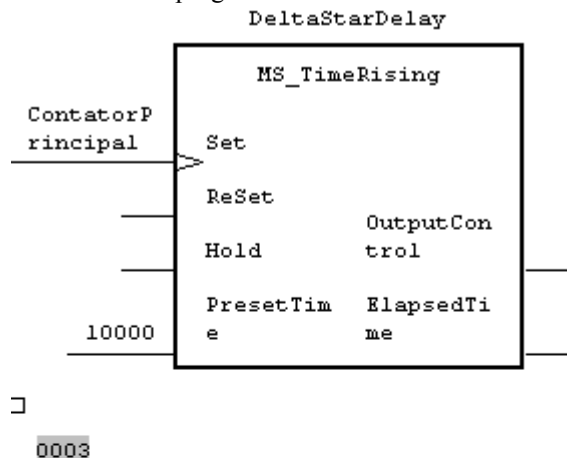
0002



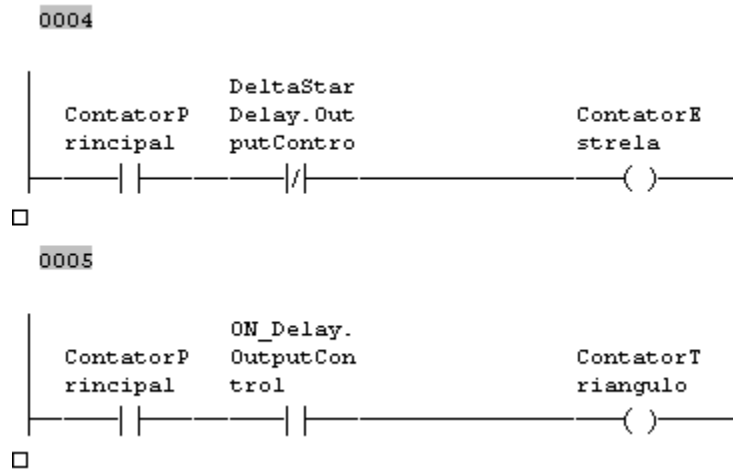
0003




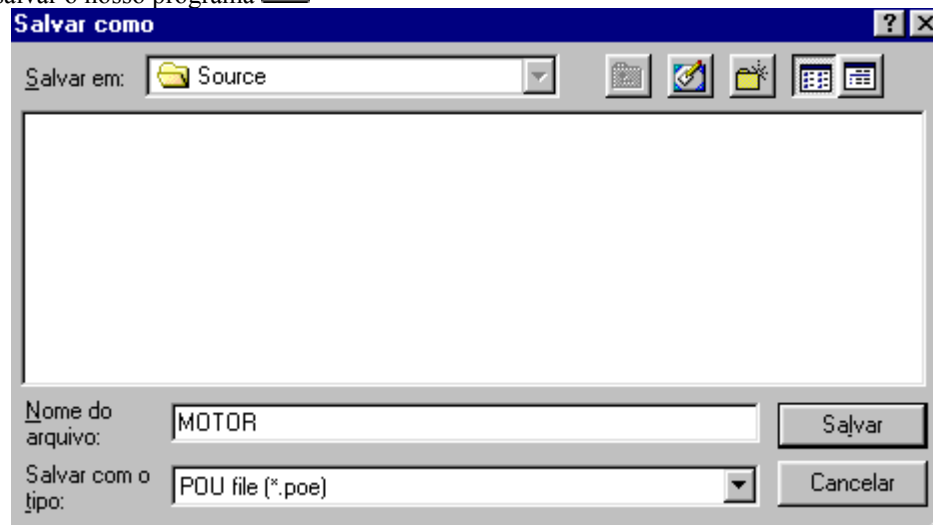
- Na entrada Set do DeltaStarDelay colocamos quem irá iniciar a contagem do tempo, no nosso caso deveremos escrever ContatorPrincipal
- Nas entradas ReSet e Hold não escreveremos nada pois elas não serão utilizadas neste programa.
- Na entrada PresetTime colocamos o tempo desejado em Milésimos de segundos. Por exemplo 10000
- A saída OutputControl é a que indica quando o tempo desejado se esgotou. Também deixamos esta em branco.
- A saída ElapsedTime indica quanto tempo já decorreu. Esta fica em branco também.
- Vamos agora acertar o outro bloco de função , o ON_Delay.
- Na entrada Set do bloco ON_Delay escreveremos o seguinte: **DeltaStarDelay.OutputControl** , quando escrevemos o nome de um bloco de função , em qualquer parte do programa , seguido de um ponto , significa que queremos pegar uma entrada/saída deste: Nome_Do_Bloco_De_Função.Saída_Entrada_Desejada
- Nas entradas ReSet e Hold não escreveremos nada pois elas não serão utilizadas neste programa.
- Na entrada PresetTime colocamos o tempo desejado em Milésimos de segundos. Por exemplo 1000
- A saída OutputControl e ElapsedTime também ficarão em branco.
- Veja abaixo como os blocos ficaram em nosso programa:



- Agora vamos escrever o resto do nosso programa, vamos colocar as linhas que fazem o acionamento dos outros contadores.
- Edite as seguintes linhas em Ladder:



- Quando declaramos um dos contatos como ON_DELAY.OutputControl este contato terá a mesma função da saída OutputControl do bloco de função ON_DELAY. Nada impede o programador de colocar um Flag na saída do temporizador e utilizar este Flag no programa.
- Vamos agora salvar o nosso programa 

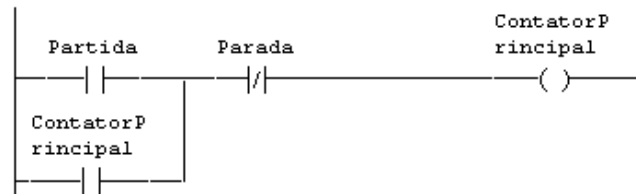


- Coloque o nome motor e salve-o na pasta Source de seu projeto. O S40 automaticamente abre este caminho para você.
- O programa final fica como segue abaixo:

]

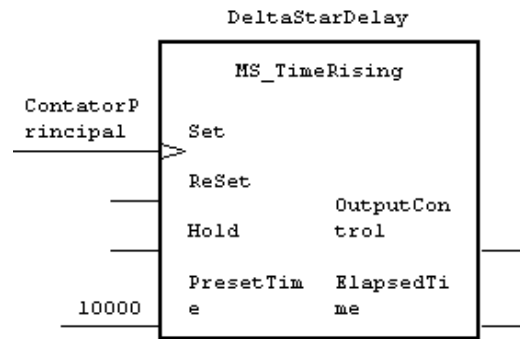
0001

Controle da Partida



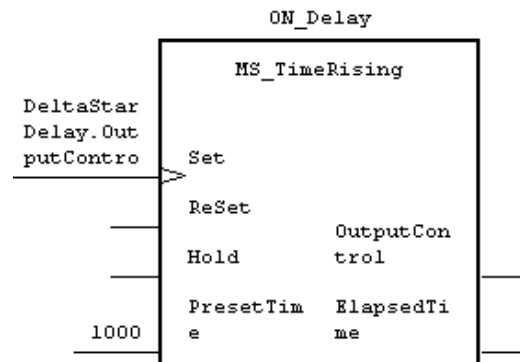
]

0002



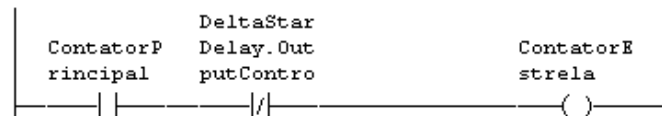
]

0003



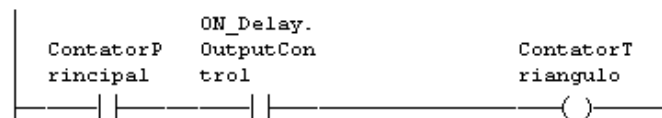
]


0004



]

0005





- Agora nós fazemos uma checagem no programa, para isso utilizamos o Syntax Check, .
- Se nenhum erro aparecer podemos fechar o POU Editor e passar então para a próxima etapa da nossa programação.

CAPÍTULO 6

COMPILANDO O PROGRAMA

- Novamente na tela do Navigator, observe que agora o seu Programa MOTOR também aparece na janela dos arquivos fonte:

↑ Name	Size	Type	Modified On	Registered
 Mestre1	1 KB	Topology (PS4-300)	15/12/03 11:42:40	<input checked="" type="checkbox"/>
 MOTOR	3 KB	Program	15/12/03 16:44:46	<input checked="" type="checkbox"/>

- Selecione agora a opção Criar um novo arquivo MAKE na barra de ferramentas do Navigator, .

Nome do programa Fonte

Nome da configuração

New Make File - PS4-300 ? X

PROGRAM POU's:



MOTOR ▼

Topology Configurations:

Mestre1 ▼

OK

Cancel

- Ao pressionarmos OK iremos criar um novo arquivo MAKE. O nome deste arquivo irá aparecer na janela superior do Navigator: **MOTOR.MAK** .
- A janela inferior do Navigator é a área de mensagem, todas as vezes que algum erro ocorrer quando estivermos gerando um novo arquivo, ou então quando estivermos compilando o nosso programa, este erro irá aparecer nesta janela. Se nenhum erro ocorrer durante a geração do arquivo MAKE a seguinte mensagem irá aparecer: **New make file created**.
- Selecionamos agora a opção Parâmetros de programação do CLP, .
- A janela muda conforme o modelo de CPU que estamos utilizando , veja abaixo as janelas correspondentes às CPUs da família PS4-200.

Senha de acesso ao programa

Versão do programa do usuário

Máximo tempo de ciclo permitido

PS4-200 - motor.MAK X

Runtime Parameters Compiler

Management

Password:

Version Number

Behaviour After NOT READY


☒ Halt

☐ Warm Start

☐ Cold Start

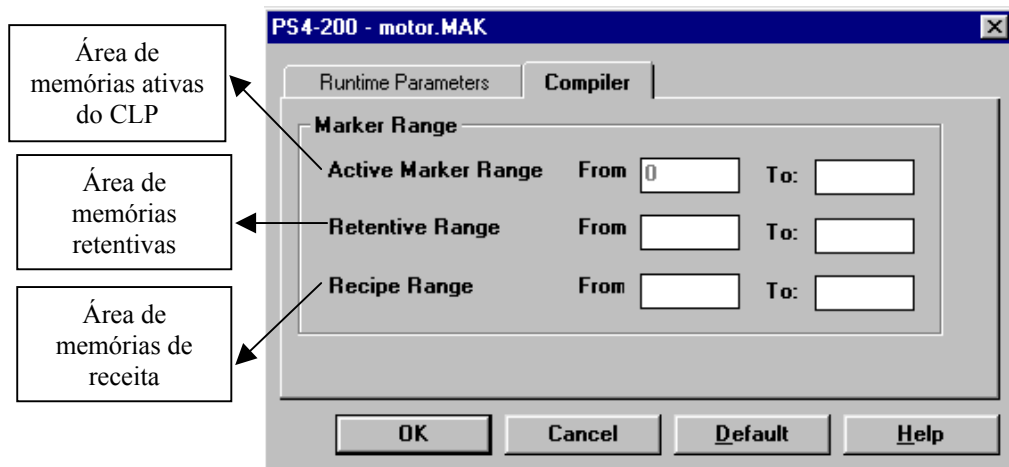
Cycle Time

Max. Cycle Time (1-255ms) ms

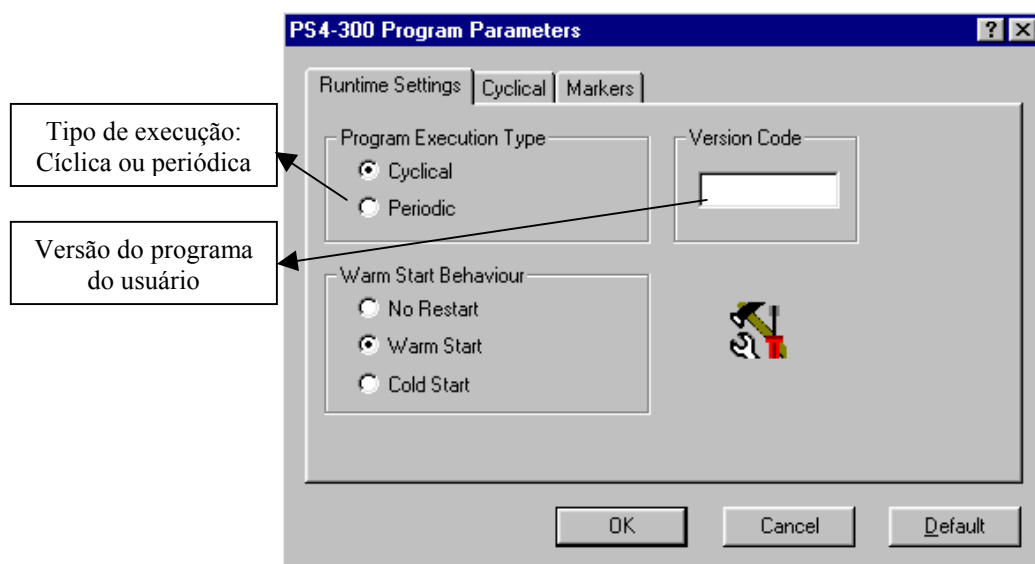


OK Cancel Default Help

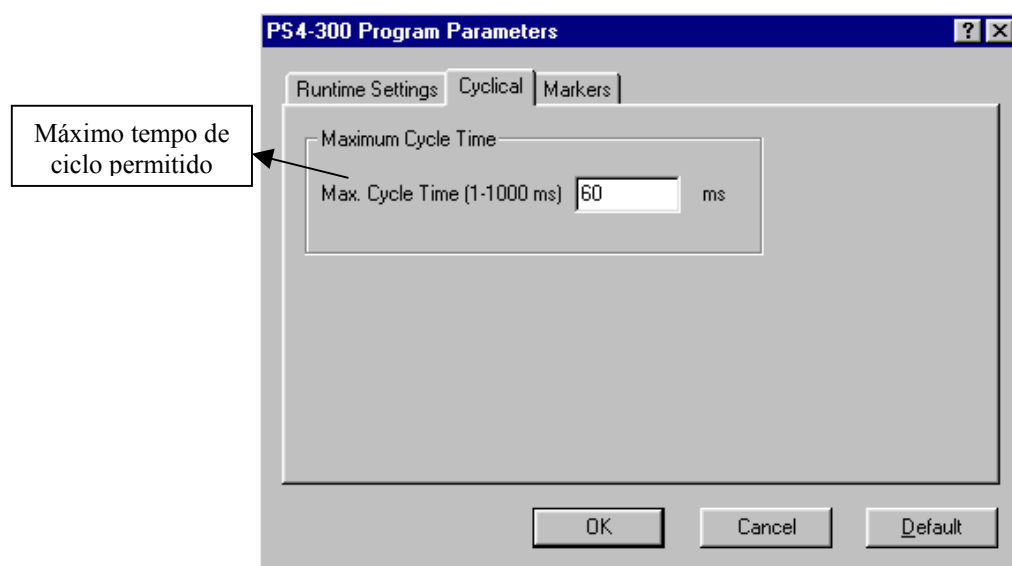
Parâmetros de Programação do CLP PS4-200



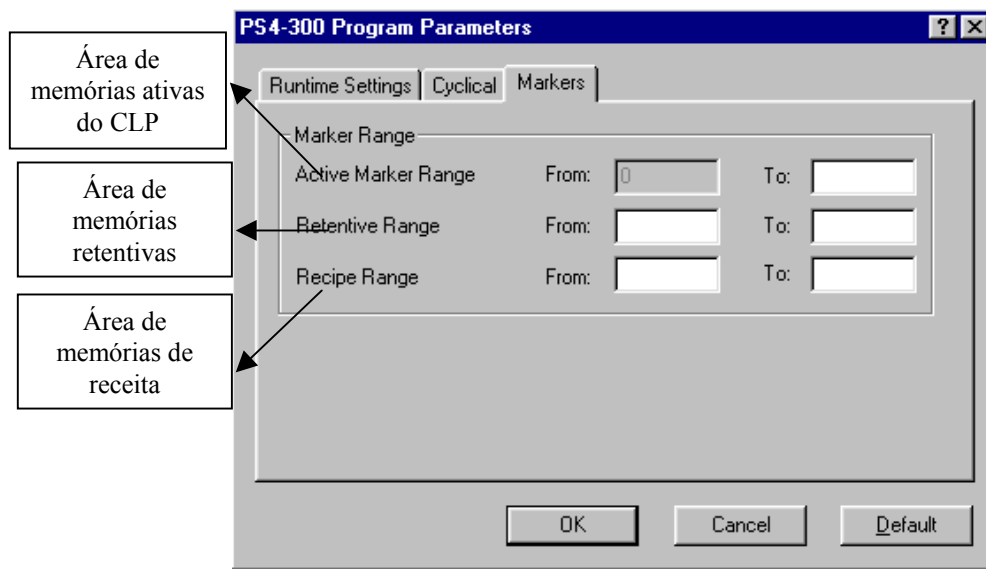
Parâmetros de Programação do CLP PS4-200




Parâmetros de Programação do CLP PS4-300



Parâmetros de Programação do CLP PS4-300



Parâmetros de Programação do CLP PS4-300

- Quando criamos um programa que usa endereços definidos de memórias devemos criar uma área de memórias ativas.
- A CPU possui uma chave de três posições, na posição HALT sempre que a CPU for ligada ela entrará em modo HALT. Se a chave for colocada na posição RUN e a CPU for ligada, todas as memórias configuradas como retentivas e receita serão preservadas, ou seja, seu conteúdo não será destruído. E por último se a chave estiver na posição RUN M-RESET e a CPU for ligada, somente as memórias de receita serão preservadas, as demais memórias serão apagadas. Isto vale para qualquer modelo PS4-200 ou PS4-300.
- No nosso caso não precisaremos declarar nenhuma memória, basta colocar no Runtime Parameters a versão do nosso programa, no caso a versão 1.00.
- Feito isto poderemos compilar o nosso programa, para isto clicamos na opção Compilar o arquivo MAKE atual, .
- Se não houver nenhum erro no nosso programa na janela inferior do Navigator irá aparecer a seguinte mensagem:

```


Open Make File: MOTOR.mak
Compilation of: MOTOR
Link Operation ...
Configuration of: MOTOR
Status of entire program 'MOTOR'
Code size =    2658 Bytes
Data size =     56 Bytes
1 function blocks with 1 instance(s)

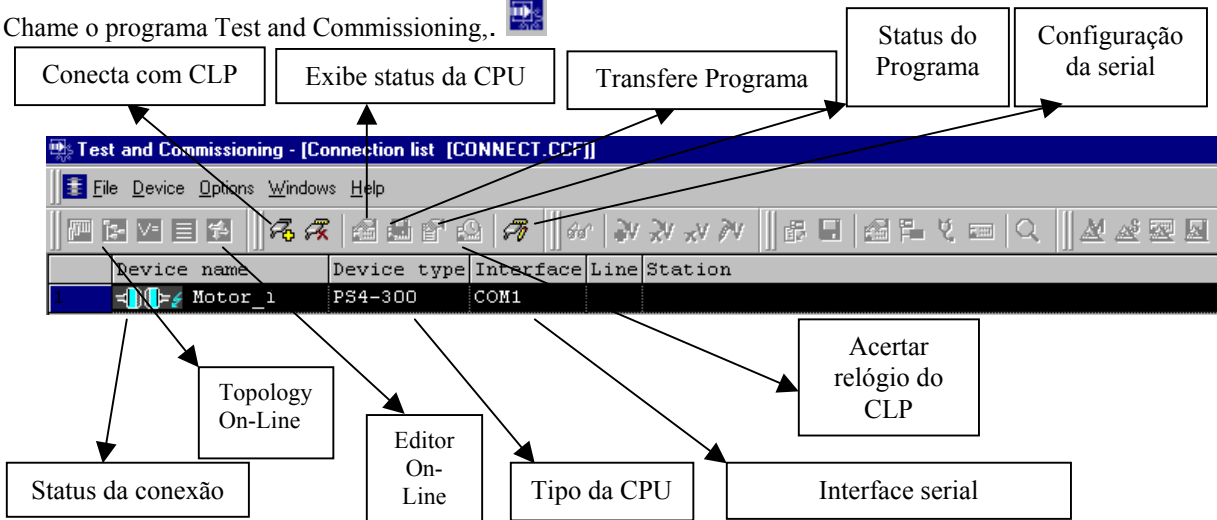
Program code generation successfully completed
  
```


- Se houver algum erro uma mensagem em vermelho irá aparecer, basta clicar nesta mensagem e a janela do POU Editor irá reabrir e selecionar a linha que contém o erro.
- Uma vez que nosso programa foi compilado com sucesso, podemos transferir o programa para o CLP e iniciar os testes

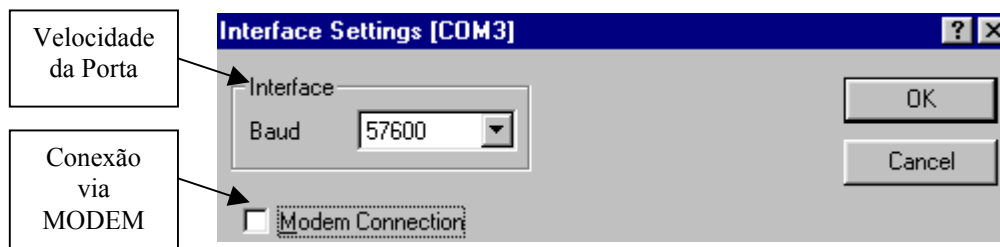
CAPÍTULO 7

TRANSFERINDO O PROGRAMA





- Ligue o seu CLP e ligue-o ao computador usando o cabo ZB4-303-KB1.
- Chame o programa Test and Commissioning. 

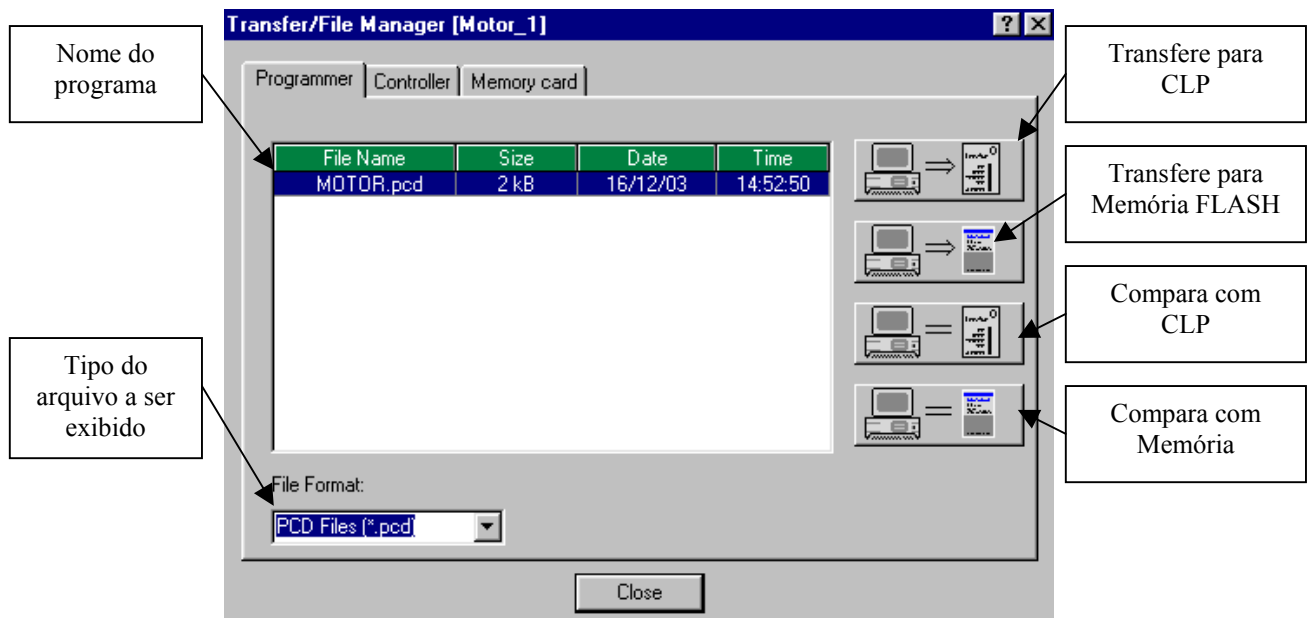


- Selecione o tipo de CPU que iremos trabalhar em primeiro lugar.
- Defina a porta serial que você está utilizando.
- Se estivermos trabalhando com um CLP da família PS4-200, podemos pular os próximos passos e ir direto para a conexão, se for uma CPU da família PS4-300 poderemos escolher a velocidade da conexão, isto é importante pois como se trata de uma CPU de maior capacidade, logicamente, ela terá programas maiores, sendo assim podemos utilizar uma conexão mais rápida para ganhar tempo.
- Clique na opção Configuração Serial : 

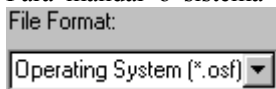


- Vamos selecionar a velocidade de 57600Baud, lembre-se isto é válida para os CLPs da família PS4-300 e PS416. Para as CPUs da família PS4-200 a velocidade será sempre de 9600 Baud.

- Então pressione Connect,  o Status da conexão irá mudar de Disconnected  para Connected .
- Para transferir o programa para o CLP clique na opção Transfere Programa : 




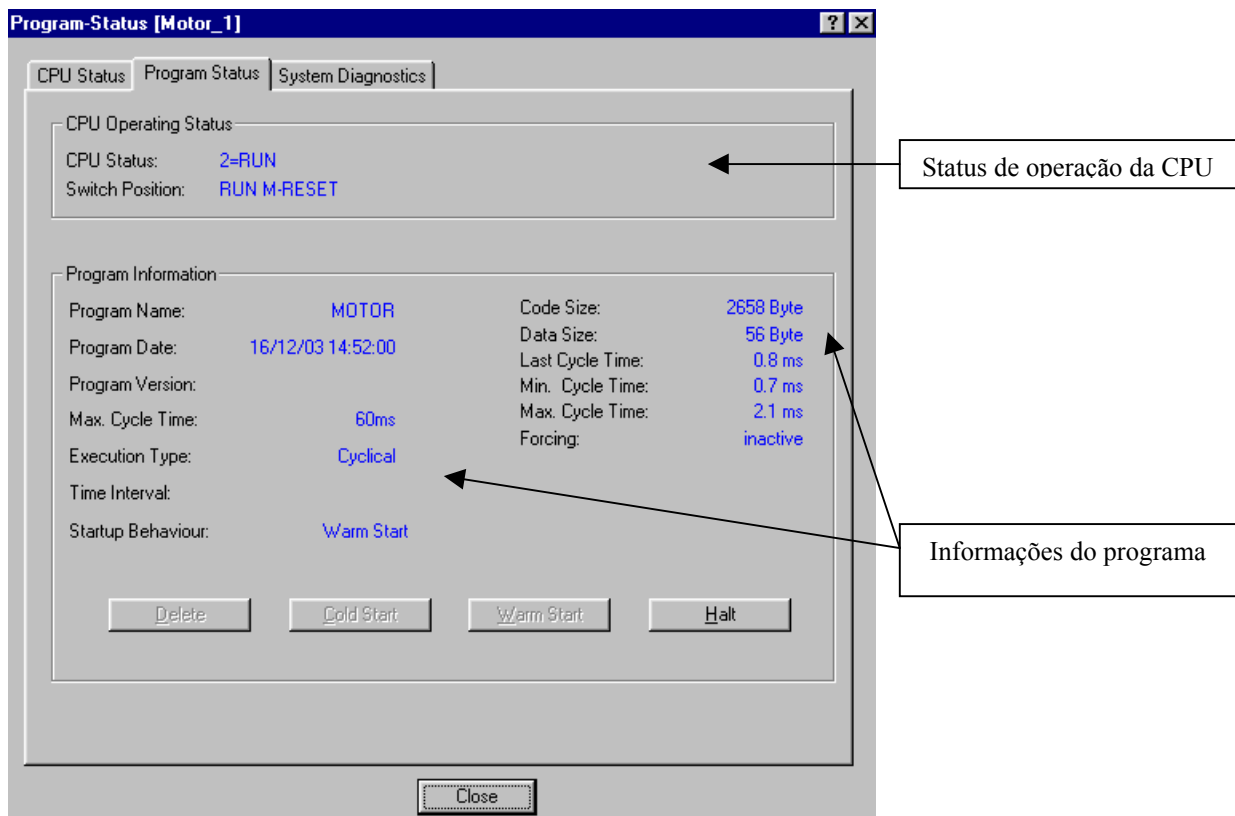
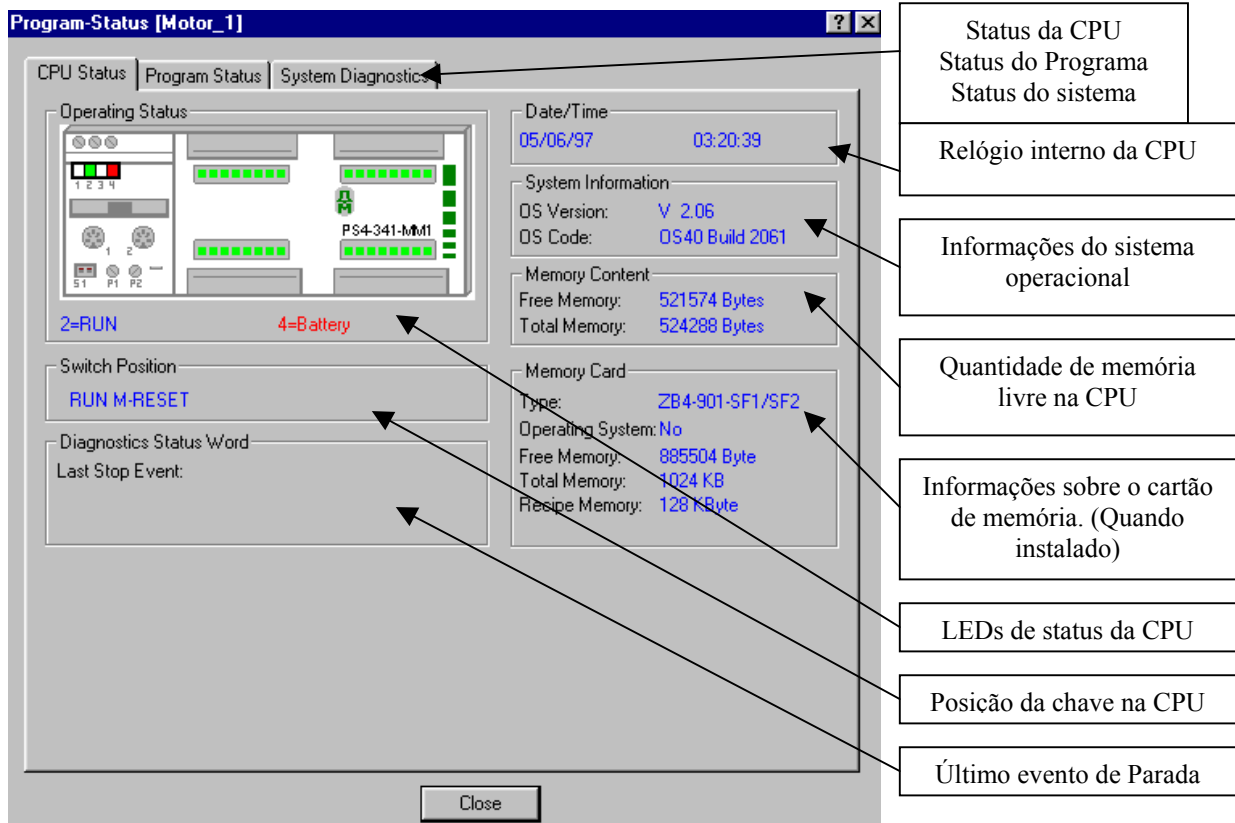
- Deixe selecionada a pasta Programmer.
- Se for um PS4-200 poderemos mandar o arquivo PCD , arquivo no formato do CLP , diretamente para a CPU. Se for um CLP da família PS4-300 ou PS416 ele vem sem o sistema operacional , por isso devemos carregar este sistema antes. Este procedimento só precisa ser feito uma vez , devendo ser repetido apenas se o CLP for desligado e estiver sem a bateria.
- Para mandar o sistema operacional , OSF , para o PS4-300 ou PS416 selecione a opção Operating System (OSF):

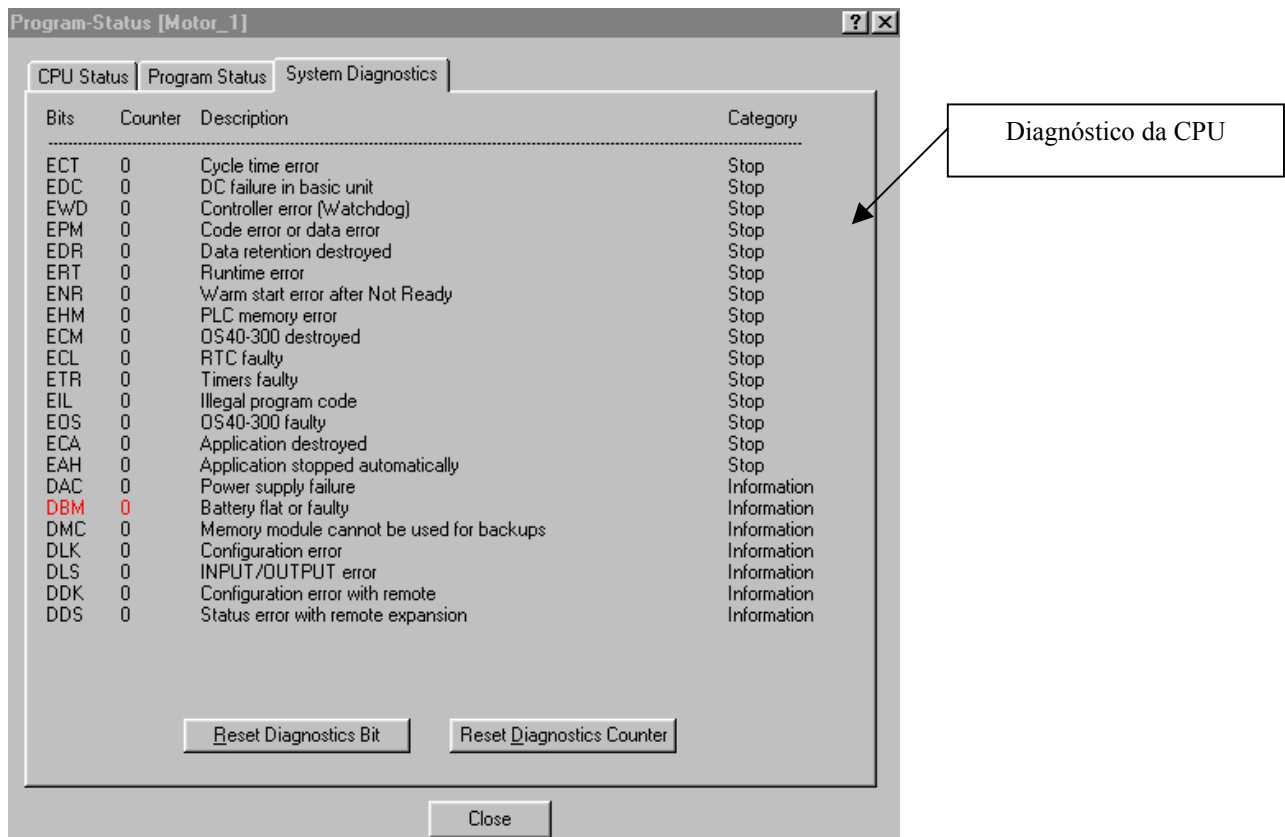


- Observe que aparecerá no campo nome programa , um arquivo chamado 341_206.OSF:


File Name	Size	Date	Time
341_206.OSF	789 kB	27/09/01	11:12:16

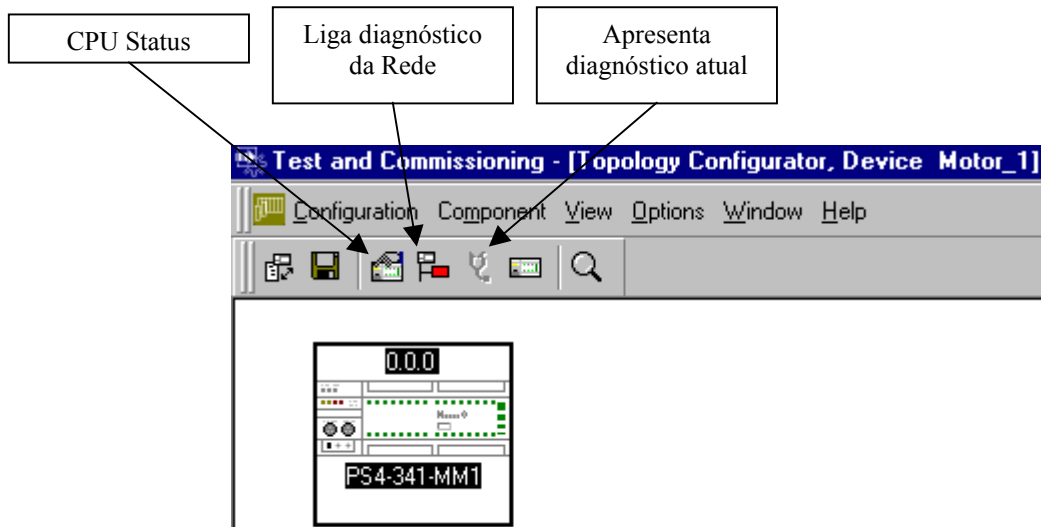
- Selecione agora o arquivo que você deseja transferir, no nosso caso é o arquivo motor.pcd, e logo em seguida selecione a opção Transfer→CLP.
- Feche agora esta janela e coloque a CPU em marcha, para isto basta abrir a tampa superior da CPU, colocar a chave na posição 3 e pressionar o botão de reset.
- Observe que os LED's que se encontram na parte superior da CPU indicam que a CPU agora está no modo RUN.
- Clique agora na opção Status da CPU... 




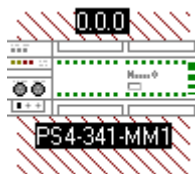



- Entre outras informações podemos ver que a CPU está no modo RUN, a chave da CPU está na posição RUN M RESET, podemos ver quanta memória está livre etc...
- Através da janela Program status podemos parar a CPU, através da tecla Halt , ou colocá-la de novo em marcha. Aqui visualizamos o tempo de ciclo da CPU , que estará em torno de 1 ms, o tamanho de nosso programa em Code Size, e etc...
- Já na janela de diagnósticos nós podemos observar os erros que estão ativos e quantas vezes determinados erros ocorreram.

- Saia desta janela e selecione o ícone Topology On-line .




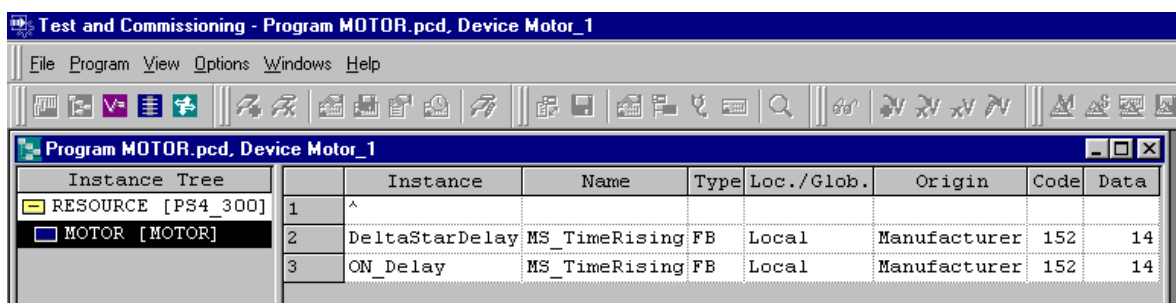
- Se ligarmos o diagnóstico de rede  e se houver qualquer diagnóstico ativo nesta unidade, ela aparecerá como segue abaixo:




- Para saber qual o diagnóstico atual da CPU / Módulo, selecione o módulo que está hachurado e em seguida a opção : Apresenta diagnóstico atual, .

MONITORANDO O PROGRAMA

- Feche agora a janela do Topology On-line e selecione a opção Open Instance tree, .



- Na janela esquerda selecione a opção MOTOR.
- Selecionamos a opção Enter Editor On Line : .

POU EDITOR - [MOTOR (online)]



File Edit View Insert Online Options Window Help

90 %

		Name	Type	Initial
1				
2				
3	0		BOOL	
4	0		BOOL	
5	0		BOOL	
6	0		BOOL	
7	0		BOOL	
8	0		BOOL	
9	1		BOOL	
10	0		BOOL	
11	0	Partida	BOOL	
12	0	Parada	BOOL	

0001

Controle da Partida

- Repare que esta tela é a mesma que temos no editor de programas , POU Editor , tanto que não conseguimos abrir esta tela se o POU Editor estiver rodando , por isso , é extremamente recomendável fechar o POU Editor e o topology configurator antes de iniciar o Debug do programa.
- Para visualizar o status de seus Tags pressione a opção status display , .
- Os contatos/Bobinas que aparecem em vermelho estão logicamente fechados/ligados.
- Do lado esquerdo da tela podemos visualizar a nossa declaração de variáveis. Na declaração também é possível visualizar o status de cada um dos TAGs
- Se precisar executar alterações On-line , execute-as normalmente utilizando as mesmas ferramentas vistas no POU Editor.
- Assim que finalizar as alterações a tecla Activate Modification irá ficar habilitada  . Ao pressionar esta tecla , as modificações serão enviadas para o CLP.

CAPÍTULO 8

CONSIDERAÇÕES FINAIS

Este manual foi criado conforme a versão mais atual do software S40. A Moeller Electric se reserva ao direito de efetuar quaisquer alterações neste manual sem aviso prévio.

Leia com atenção os manuais de programação, tente compreender mais blocos de funções através do Help do software e através do manual.

O programa aqui apresentado é apenas para efeito didático, a Moeller Electric não se responsabiliza por danos que o uso deste programa venha a causar.

Qualquer dúvida contate:

Moeller Electric Ltda

e-mail: suporte@moeller.com.br