Manual Básico Operacional CLP Telemecanique Modicon M340 /DDM16025/AMI0210/AMO0410 2030 e 2020

Presidente da FIEMG
Olavo Machado Junior

Gestor do SENAI
Petrônio Machado Zica

Diretor Regional do SENAI e
Superintendente de Conhecimento e Tecnologia
Lucio José de Figueiredo Sampaio

Gerente de Educação e Tecnologia
Edmar Fernando de Alcântara

Elaboração / Ano
Anderson Dias de Oliveira / 2010

Unidade Operacional
Centro de Formação Profissional “Orlando Chiarini”

APRESENTAÇÃO

"Muda a forma de trabalhar, agir, sentir, pensar na chamada sociedade do conhecimento."

Peter Drucker

O ingresso na sociedade da informação exige mudanças profundas em todos os perfis profissionais, especialmente naqueles diretamente envolvidos na produção, coleta, disseminação e uso da informação.

O **SENAI**, maior rede privada de educação profissional do país, sabe disso, e, consciente do seu papel formativo, educa o trabalhador sob a égide do conceito da competência***: "formar o profissional com responsabilidade no processo produtivo, com iniciativa na resolução de problemas, com conhecimentos técnicos aprofundados, flexibilidade e criatividade, empreendedorismo e consciência da necessidade de educação continuada*"**.**

Vivemos numa sociedade da informação. O conhecimento, na sua área tecnológica, amplia-se e se multiplica a cada dia. Uma constante atualização se faz necessária. Para o **SENAI**, cuidar do seu acervo bibliográfico, da sua infovia, da conexão de suas escolas à rede mundial de informações – internet - é tão importante quanto zelar pela produção de material didático.

Isto porque, nos embates diários, instrutores e alunos, nas diversas oficinas e laboratórios do **SENAI**, fazem com que as informações, contidas nos materiais didáticos, tomem sentido e se concretizem em múltiplos conhecimentos.

O **SENAI** deseja, por meio dos diversos materiais didáticos, aguçar a sua curiosidade, responder às suas demandas de informações e construir *links* entre os diversos conhecimentos, tão importantes para sua formação continuada !

Gerência de Educação e Tecnologia

Manual Básico Operacional CLP Telemecanique
Modicon M340
/DDM16025/AMI0210/AMO0410

Introdução

De formato compacto o CLP Modicon M340 oferece a flexibilidade e os serviços de um CLP de grande porte. No centro da sua aplicação, disponibiliza soluções integradas Plug and Work (Ligar e Trabalhar) com outros dispositivos Schneider Electric . Durante o desenvolvimento do projeto, a grande capacidade da ferramenta Unity irá facilitar e abreviar o seu tempo de programação.

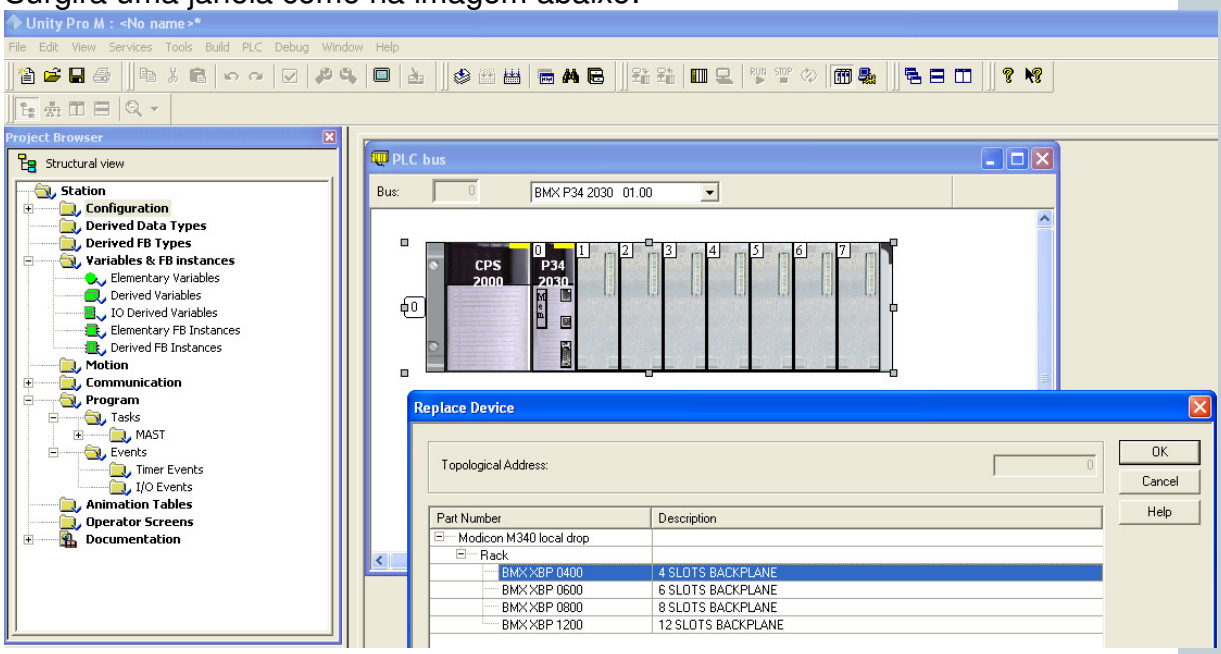
Unity Pro

Para iniciarmos a utilização do Unity Pro, devemos fazer algumas configurações para que o CLP possa conectar-se a máquina. A esmagadora maioria dos CLP's possuem este procedimento, porém claro, tendo cada um deles um programa específico para este fim.

Configuração física

Ao abrir o UNIT PRO clique em "NEW PROJECT", surgirá uma janela onde devemos escolher o modelo do cartão de comunicação do CLP, que pode ser observado no topo do módulo. No caso de nosso exemplo selecionaremos o 2030. Após selecionar o cartão devemos clicar em "Configuration" no lado esquerdo superior da tela.

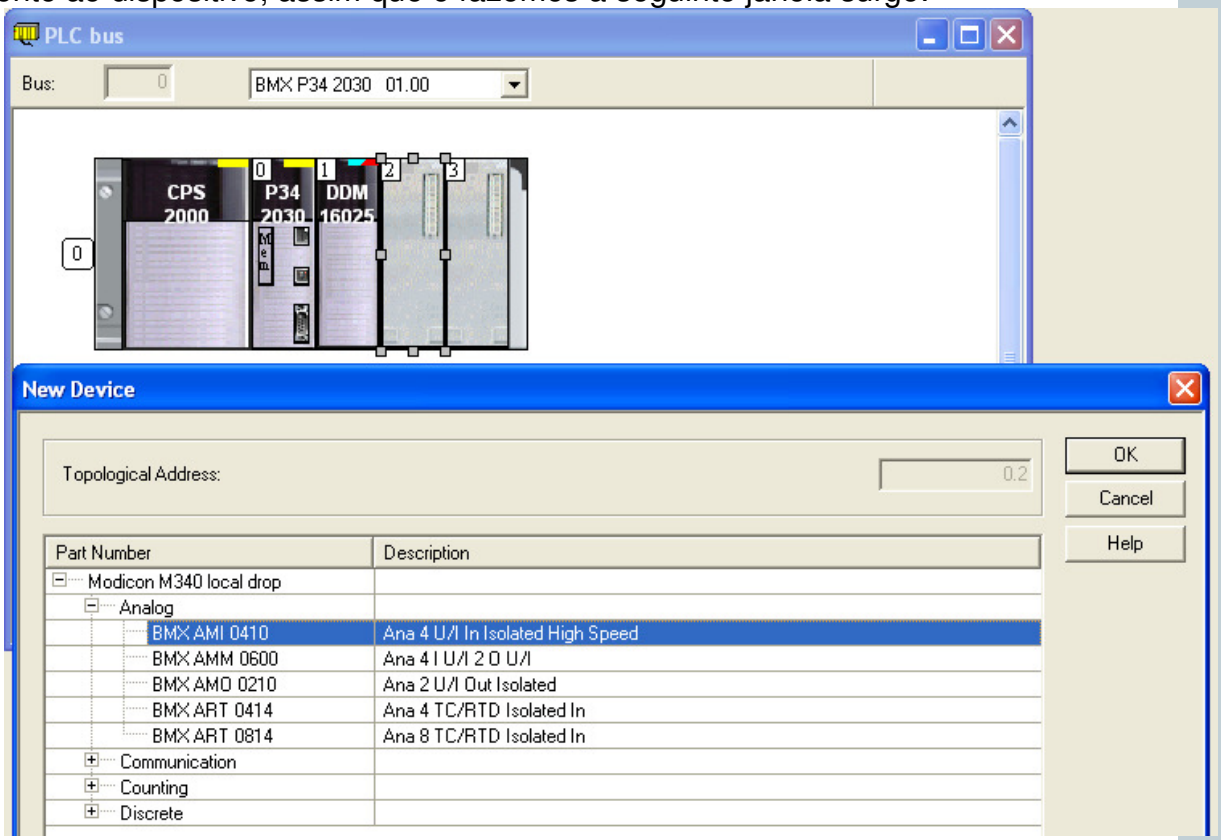
Surgirá uma janela como na imagem abaixo:



Devemos então dar um duplo clique no "0" (zero) que fica do lado esquerdo da figura do CLP, e na janela "Replace Device" selecionar o número de slots, ou módulos que o CLP possui, em nosso exemplo selecionaremos 4 slots. A maioria dos CLP Telemecanique possuem 4 slots.

Como podemos reparar na imagem acima o primeiro módulo corresponde a fonte do CLP, o segundo módulo que chamamos de "0" é a unidade central de processamento, e os demais módulos são as entradas e saídas analógicas e digitais (1,2,3,4...).

Percebemos ainda que a fonte e a CPU estão devidamente identificados, e os demais slots apresentam-se em branco, faz-se então extremamente necessário que configuremos os slots, e para tanto devemos dar um duplo clique acima do cartão referente ao dispositivo, assim que o fazemos a seguinte janela surge:



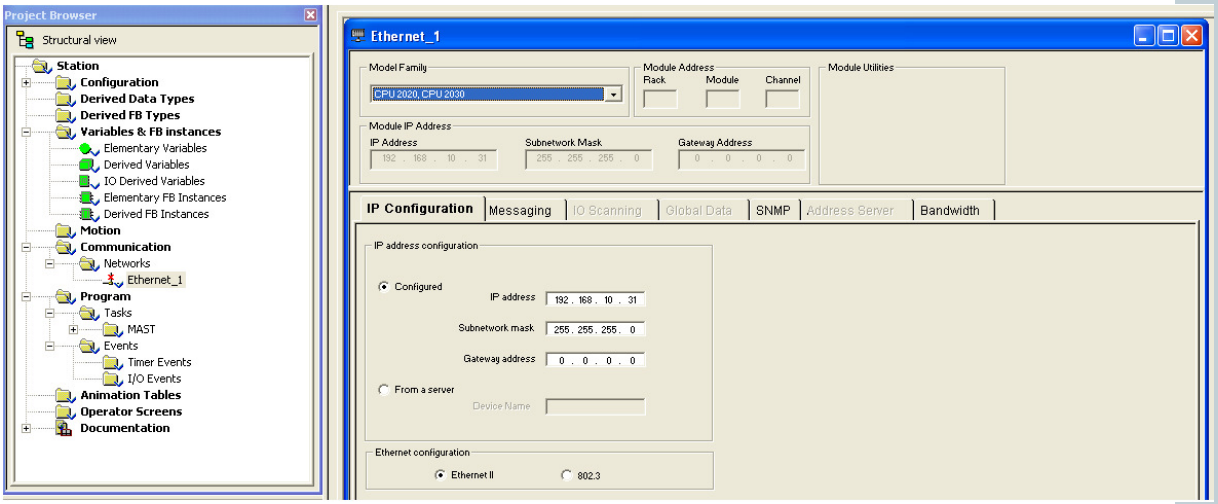
Em “NEW DEVICE” podemos selecionar o tipo de cartão que queremos inserir no CLP, pra reconhecer o cartão correto, podemos observar a numeração acima do cartão, como exemplo podemos citar o cartão de número 1, que é composto de entradas digitais, devemos escolher “Discrete” e (BMX DDM 16025). O mesmo procedimento é tomado nos demais itens, como no slot 2 entradas analógicas - escolher Analog → BMX AMI 0410 e no slot 3 – Saídas analógicas – devemos escolher BMX AMO 0210.

Nossa parte física já está quase toda configurada, devemos configurar agora a parte de rede.

Configuração de rede

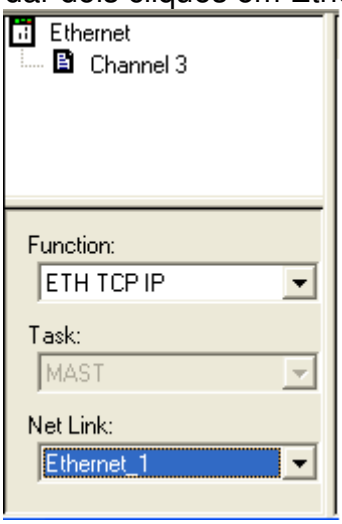
Na lista lateral esquerda devemos selecionar “Comunication” posteriormente “Networks e com o botão direito criar um “New Network” e selecionar “Ethernet”.

Surgirá um ícone de “Ethernet” abaixo de network, no canto esquerdo da tela, dê duplo clique em ethernet1. Neste ponto surgirá uma janela como a seguinte:



Devemos então selecionar o modelo da família do CLP (Model Family – CPU 2020, CPU 2030), e logo abaixo podemos inserir o endereço de IP e a máscara de sub-rede, tendo uma vez realizado o processo devemos clicar no ícone “Validate”, ou ir e “EDIT” e clicar em “Validate” para que o programa armazene as modificações.”

Finalizando podemos retornar a “Configuration”, e no primeiro cartão percebemos que possuem três interfaces de comunicação, uma USB, uma Ethernet e um CANopen, no nosso caso configuraremos o ETHERNET, e para isso podemos dar dois cliques em Ethernet, e a seguinte janela surgirá:



Clique em Chanel e posteriormente em function alternar para ETH TCP IP e depois em NetLink, selecionar Ethernet_1 e validar novamente.

Faz-se extremamente necessário que a configuração seja compilada, pois assim o programa consegue criar todas as guias e plataformas utilizadas na conexão. Podemos ir na barra de menus e no menu “BUILD” clicar em “Analyze Project”, depois em “Build Changes” e finalmente em “Rebuild All Project”.

Como último passo temos que testar a comunicação do CLP, para tanto podemos ir em PLC, “SET ADDRESS”, e inserir em:

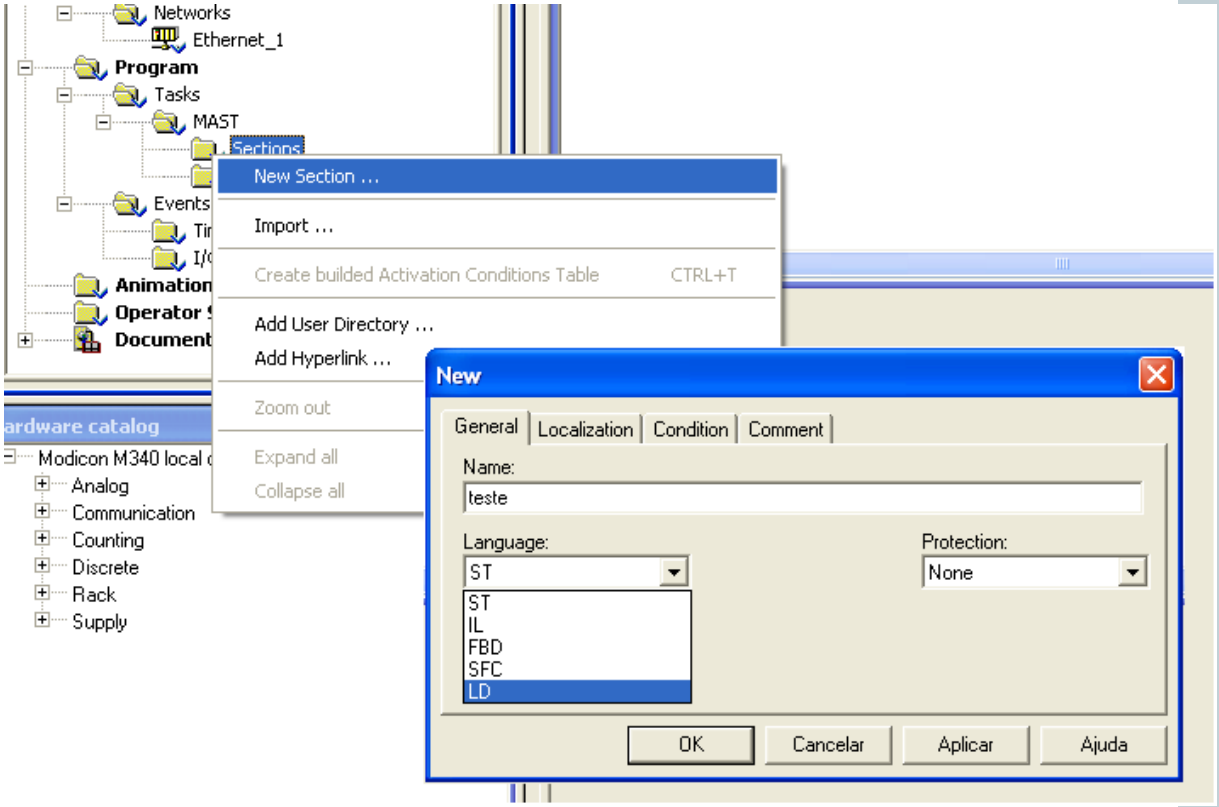
- Address → o IP do CLP
- Media → TCP-IP

Posteriormente, clicar em “test conection.” Nesta fase o CLP exibirá uma mensagem de sucesso ao conectar ou não, e neste caso devemos rever nossos processos e conexões.

Criando um programa

Na janela lateral esquerda ir em “Program” , “Tasks”, “Mast”, e finalmente, clicar com o botão direito em “Sections”, em “New section”.

- Dar nome ao programa, e em “Language” selecionar LD (ladder), assim como ilustra a figura abaixo:.



Nesta fase surgirá a tela de programação, devemos agora conhecer as ferramentas para esta atividade.

Programação

Como selecionamos a linguagem de programação em ladder, prosseguiremos com ela, posteriormente vendo as demais.

Neste programa (Unity Pro) devemos clicar no ícone e inseri-lo, através de um clique, na área de trabalho do programa, no local desejado. Vale ressaltar que deve-se colocar contatos mais a esquerda, bobinas e memórias mais a direita e demais blocos ao centro, pois desta forma podemos proporcionar uma melhor organização no programa.

- Negated Coil – Bobina de negação, funciona semelhantemente a uma porta NOT.

- Set Coil – Bobina retentiva – Uma vez alimentada mantêm-se até que seja ressetada.

- Reset Coil – Desliga um Bit (reset)

- Positive Transition Sense-coil – Bobina sensível a transição positiva (flanco positivo)

•Negative Transition Sense-coil – Bobina sensível a transição negativa (flanco negativo)

•Haltcoil – Bobina de parada = Trava todo o sistema através do impedimento da alimentação, porém, mantém todos os bits salvos.

•Call Coil – Bobina de chamado = Permite chamar ou acionar uma sub-rotina pré-definida.

Blocos de contadores e temporizadores

Memórias

BIT = corresponde a um sinal, por exemplo, um pulso provindo de uma botoeira.

BYTE = É um conjunto de 8 bits, podemos utilizá-lo para controle de módulos IO, e leitura de sinais codificados.

WORD = Baseia-se em um conjunto de 2 bytes, ou 16 bits. Utilizamos para leitura de sensores analógicos.

DOUBLE-WORD = é composta por um conjunto de 2 words ou 32 bits, e também é utilizada para leitura de sensores analógicos de grande variação.

Para inserir contadores e temporizadores temos que clicar no ícone de nome “FFB Input Assistant” ou acionar Ctrl+i.

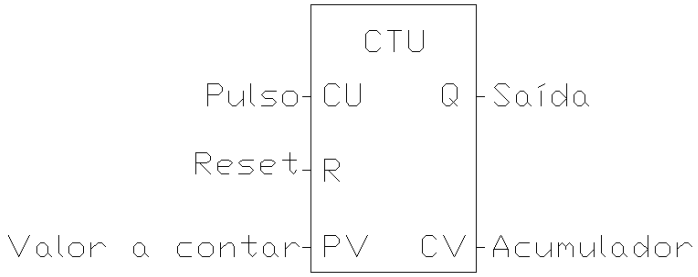
Para imputar contadores podemos digitar CTD ou CTU, e para temporizadores, podemos escrever TON ou TOF.

Blocos de função

CONTADORES (Counters)

Contadores são blocos que têm a capacidade de contar de forma crescente ou decrescentemente variações de um atuador aferido por um sensor, ou seja, é capaz de contar “n” números de processos.

CTU – Counter UP



É capaz de contar de forma crescente até que o valor de contagem seja igual ao valor que se deseja, definido anteriormente, sendo que neste instante a saída é

acionada. Esta ferramenta mantém a saída acionada e continua contando, no caso de novas incidências de sinal.

Podemos observar que temos vários elementos no bloco acima, sendo:

- CU = é nesta entrada que damos os pulsos a serem contados, sendo que estes podem provir de um sensor, por exemplo.
- Reset = toda vez que o bloco efetuar uma contagem pré-definida, a saída é acionada, porém a contagem continua em vigor. Para limpar então o bloco, podemos dar um pulso em RESET, que é responsável por apagar os dados salvos em CTU.
- PV (preset value) = é a guia onde devemos inserir o número a ser contado, por exemplo: Se queremos que o contador conte até cinco, devemos inserir tal número em PV. Quando CU receber 5 pulsos a saída será acionada.
- Q = é a saída do contador, quando PV = CV a saída é acionada.
- CV = é um acumulador, é ele que registra a contagem.

CTD – Counter DOWN

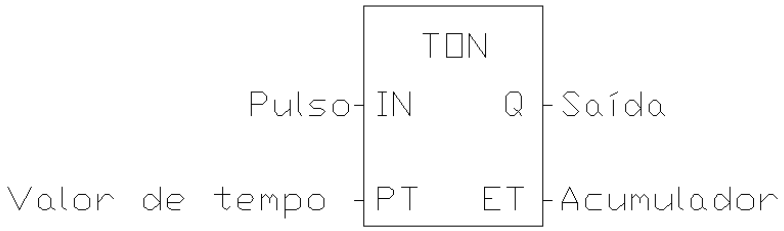
O couter down, ou contador decrescente têm praticamente a mesma configuração que o CTU, porém devemos lançar um valor maior para que possa ser decrescido no decorrer da contagem .

TIMERS (Temporizadores)

Os temporizadores são infinitamente utilizados em todos os becos da programação, e no ladder não é diferente, pois existe uma grande necessidade de contar um tempo para fins múltiplos. Desta forma podemos encontrar dois tipos básicos de temporizadores, sendo o TON (Timer on-delay) e TOF (Timer off-delay). É óbvio que existem inúmeros outros tipos de temporizadores, com especificidades inerentes a cada tipo de aplicação, os quais veremos no decorrer do curso.

TON

Este é o tipo mais utilizado e simples de temporizador, pois quando acionado, passa a contar um tempo pré-definido, e quando este tempo é contado ele aciona a saída.

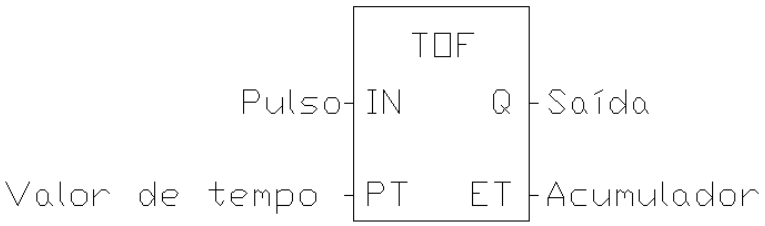


- IN = é onde devemos inserir o comando para que o operador comece a contar o tempo.
- PT = devemos inserir o tempo que desejamos, deve vir acompanhado de unidade de medida (Segundos, horas, minutos, etc)
- Q = é a saída, quando ET=PT a saída assume “1”.
- ET = Realiza a contagem de tempo interna do componente.

OBS.: Ao aplicar um pulso em “IN” ele começa a contagem de tempo, se interrompermos este pulso o acumulador é zerado. Se mantivermos o pulso o contador acionará a carga depois do tempo decorrido, e a carga ficará ligada enquanto houver pulso em IN.

TOF

Também muito utilizado, o TOF tem um papel semelhante ao do TON, a diferença é que ao inserirmos um pulso no mesmo a saída é imediatamente acionada, a medida que o tempo é contado. Ao final do tempo a saída se desaciona, esperando nova ordem.

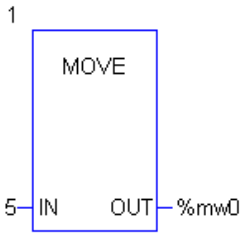


- IN = é onde devemos inserir o comando para que o operador comece a contar o tempo.
- PT = devemos inserir o tempo que desejamos, deve vir acompanhado de unidade de medida (Segundos, horas, minutos, etc)
- Q = é a saída, quando ET=PT a saída assume “0”.
- ET = Realiza a contagem de tempo interna do componente.

Existem inúmeras outras ferramentas (componentes) que veremos mais adiante, de complexibilidade também mais acentuada que as apresentadas até o momento. Estas vistas são as mais básicas, são aplicadas em todos os programas, porém ainda existem cerca de 2000 ferramentas para diversas aplicações.

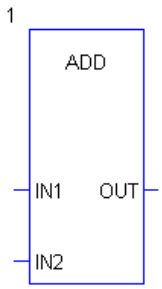
MOVE

Move um valor de um ponto a outro do programa.



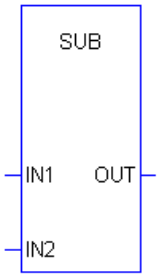
Como exemplo movimentamos o valor 5 para uma Word na memória do clp, é válido lembrar que como 5 é um número composto por vários bits, não conseguiremos salvá-lo em uma memória comum, necessitando de uma Word.

SOMADOR



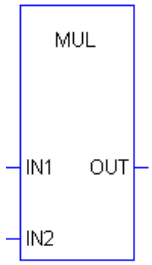
Através da inserção de valores em IN1 e IN2, podemos somar os valores, novamente precisamos utilizar words ou até Double-words.

SUBTRATOR



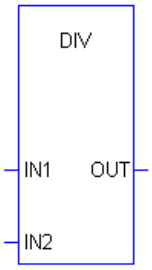
Podemos subtrair valores nas entradas IN1 e IN2, obtendo esta diferença através de out. Também devemos utilizar Word ou Double-word.

MULTIPLICADOR



Pode multiplicar dois escalares inseridos em IN1 e IN2, sendo Word ou Double-word.

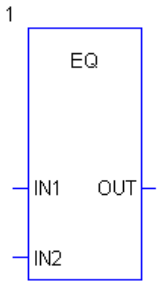
DIVISOR



Este bloco nos permite encontrar a relação entre dois números escalares. Ao inserir os números em IN1 e IN2, sendo IN1 o dividendo e IN2 o divisor, efetua-se a operação.

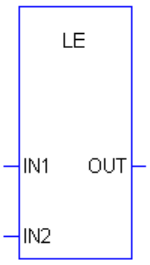
COMPARADORES

Comparador de igualdade



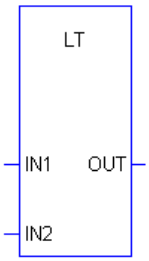
Este comparador analisa as duas entradas de sinal, IN1 e IN2. A saída Out do comparador assume valor lógico positivo, se e somente se, IN1 = IN2.

Comparador de menor igual



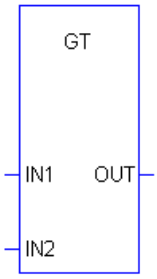
Este comparador eleva a saída OUT para 1, se IN1 for menor ou igual a IN2.

Comparador de menor



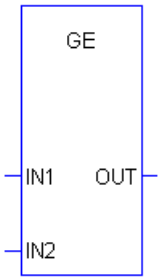
Less than – Este comparador tem por sua vez a função de aferir os valores das duas entradas, e elevar a saída para 1 se e somente se, IN1 for menor que IN2.

Comparador de maior



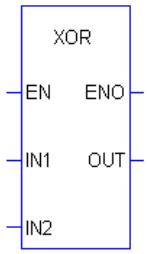
Great than – Este comparador tem por sua vez a função de aferir os valores das duas entradas, e elevar a saída para 1 se e somente se, IN1 for maior que IN2.

Comparador de maior igual



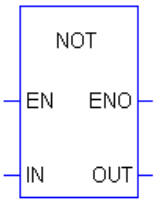
Este comparador eleva a saída OUT para 1, se IN1 for maior ou igual a IN2.

Comparador de diferença



Este bloco tem a mesma funcionalidade de uma porta lógica “ou exclusivo”. Então quando as entradas (IN1 e IN2) tiverem sinais opostos, a saída (out) terá nível lógico alto.

Comparador de negação



O bloco NOT tem como papel trocar o valor de entrada, em outras palavras: a saída estará ligada quando a entrada estiver desligada e o recíproco é verdadeiro.

UP e DOWNLOAD

A finalidade de se construir um programa é podermos gravá-lo no CLP, para que execute as tarefas solicitadas. Desta forma, quando o programa estiver pronto podemos prosseguir da seguinte maneira:

Toda modificação ou criação de programa implica em uma nova conferência de seus dados, então é preciso que compilemos o programa novamente. E para tal podemos clicar na barra de menus e no menu “BUILD” clicar em “Analyze Project”, depois em “Build Changes” e finalmente em “Rebuild All Project”, como já foi citado anteriormente.

Como segundo passo podemos ir na barra de menus e selecionar o menu “PLC”, posteriormente, clicando em “Connect”, feito isso o programa habilitará mais

alguns ícones e chaves referentes a trafego de dados, neste mesmo ponto e menu, podemos selecionar “Transfer Project to PLC”, para transportar os dados do computador para o CLP; ou clicamos em “Transfer Project From PLC, caso queiramos “baixar” o programa que está no CLP.

Todo CLP possui uma chave de “PROGRAM” (modo de programação) e “RUN” (Funcionamento), sendo que, obviamente, para programar deve estar selecionado a guia “Program” é para que o CLP funcione o programa criado, e selecionar “RUN” para que o mesmo execute as tarefas gravadas.

Principalmente alguns modelos mais antigos de CLP esta ”chave” é manual, neste modelo apresentado temos uma chave virtual que pode ser localizada no menu “PLC”, (Run e Stop), ao clicarmos em “RUN” o CLP começa a operar sozinho, podemos então retirar até o cabo de comunicação que não fará diferença alguma, citamos o exemplo da necessidade desta atitude vendo uma fábrica que apresenta ambiente hostil, onde um computador convencional não suportaria a exposição a tal ambiente, esta máquina citada seria então utilizada para somente programar o CLP, sendo retirada assim que os dados fossem transferidos sem alterar o funcionamento do CLP.

Quando quisermos modificar um programa já em um CLP, devemos nos lembrar de modificar o estado do CLP para “Stop” (parado), pois somente neste estágio devemos programar.

Temos abaixo um exemplo parcial da barra de ferramentas encontrada no Unity Pro.

Começando da esquerda para direita temos:
1 – **Go To** → Transporta a tela de maneira rápida a um ponto do programa, previamente definido.
2 – **Analyse Project** → Analisa o projeto (programa).
3 – **Build Changes** → Constrói algumas mudanças na estrutura da configuração, necessárias a comunicação do PC com o CLP.
4 – **Rebuild Project** → Reconstrói, testa e aplica as modificações e programa.
5 – **Project Browser** → Navegador de projeto – Insere a janela lateral esquerda, de configurações que utilizamos o tempo todo.
6 – **Data Search** → Nos permite localizar um componente específico pelo nome dentre o programa;
7 – **Types Library Manager** → Gerencia as bibliotecas do sistema,
8 – **Download Project** → Transfere o programa do computador para o CLP;
9 – **Upload Project** → Transfere um programa do CLP para o computador;
10 – **Conect** → Permite que o computador conecte-se ao CLP para transferência de programa.
11 – **Disconect** → Desconecta o CLP para alterações no programa,
12 – **RUN** → Permite ao CLP funcionar, sem ajuda de outro equipamento de programação.
13 – **Stop** → Coloca o CLP no modo de programação;
14 – **Animation** → Permite a criação de alguma animações;

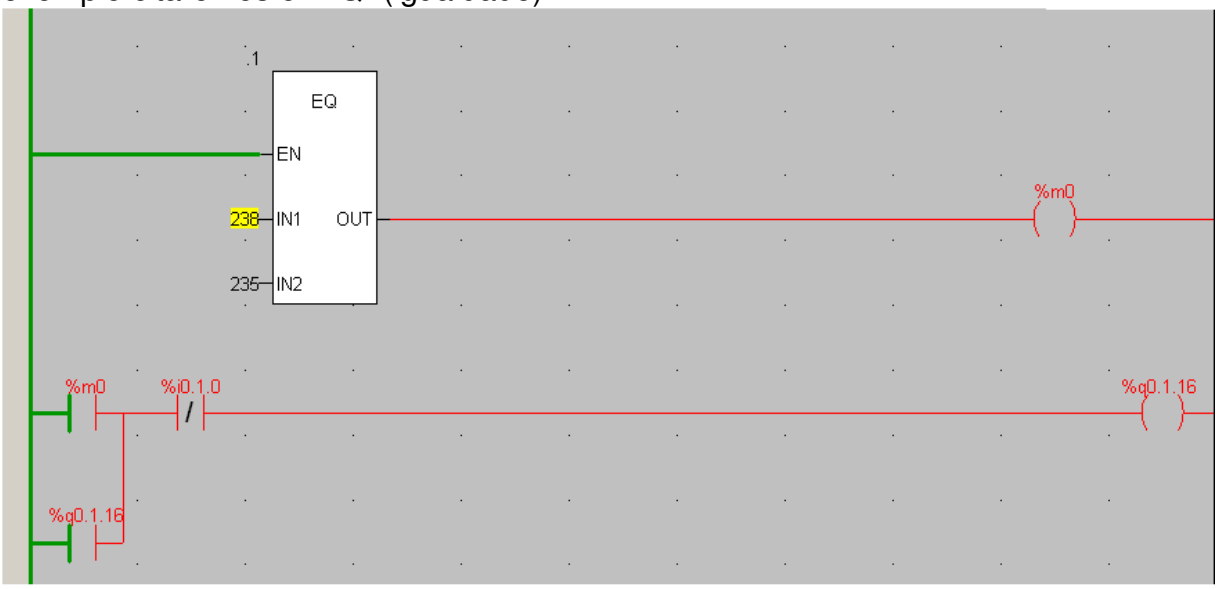
15 – **Standard Mode** → Para transferir o programa ao CLP este ícone deve estar acionado, pois permite a comunicação.
16 - **Simulation Mode** → permite somente simulação do programa, não conseguimos gravar se esta guia estiver acionada.

Trabalhando com entradas e saídas analógicas

Os CLP's também podem possuir várias entradas e saídas analógicas. Estas podem ser utilizadas quando necessitamos de "ler" valores elétricos variáveis provindos de um ou mais sensores. Como exemplo, podemos citar o trabalho de uma caldeira: Esta possui uma válvula de alívio quando a pressão interna chega a valores muito elevados, e esta válvula é acionada de forma proporcional a pressão, assim, quando um sensor de pressão variar sua tensão de saída, o CLP é capaz de ler e comparar este valor, fazendo com que a pressão seja aliviada até que se atinja um valor de segurança.

Este sistema pode ser utilizado em sensores de pressão, altura, profundidade, temperatura, força, enfim, todo tipo de sensor que capta uma variação escalar ou exponencial em sua saída.

Não existe neste sistema abordado, um bloco que seja dedicado a trabalhar com sinais analógicos, embora alguns deles sejam mais eficientes para tanto. Como exemplo citaremos o "EQ" (igualdade):



No exemplo temos um comparador de igualdade donde:
- EN – Enable = Habilita o funcionamento do EQ.
- IN1 = Entrada 1 = É onde devemos inserir o endereço da entrada analógica que queremos acionar, que neste CLP é de %I.W.0.2.0 até %I.W.0.2.3.
- IN2 = devemos digitar neste campo o valor a ser comparado, que neste caso é o 235.
- OUT = saída do bloco, quando IN1 for igual a IN2 a saída é acionada.

As entradas IW são capazes de perceber variações de tensão, para fins de teste, podemos utilizar um potenciômetro.

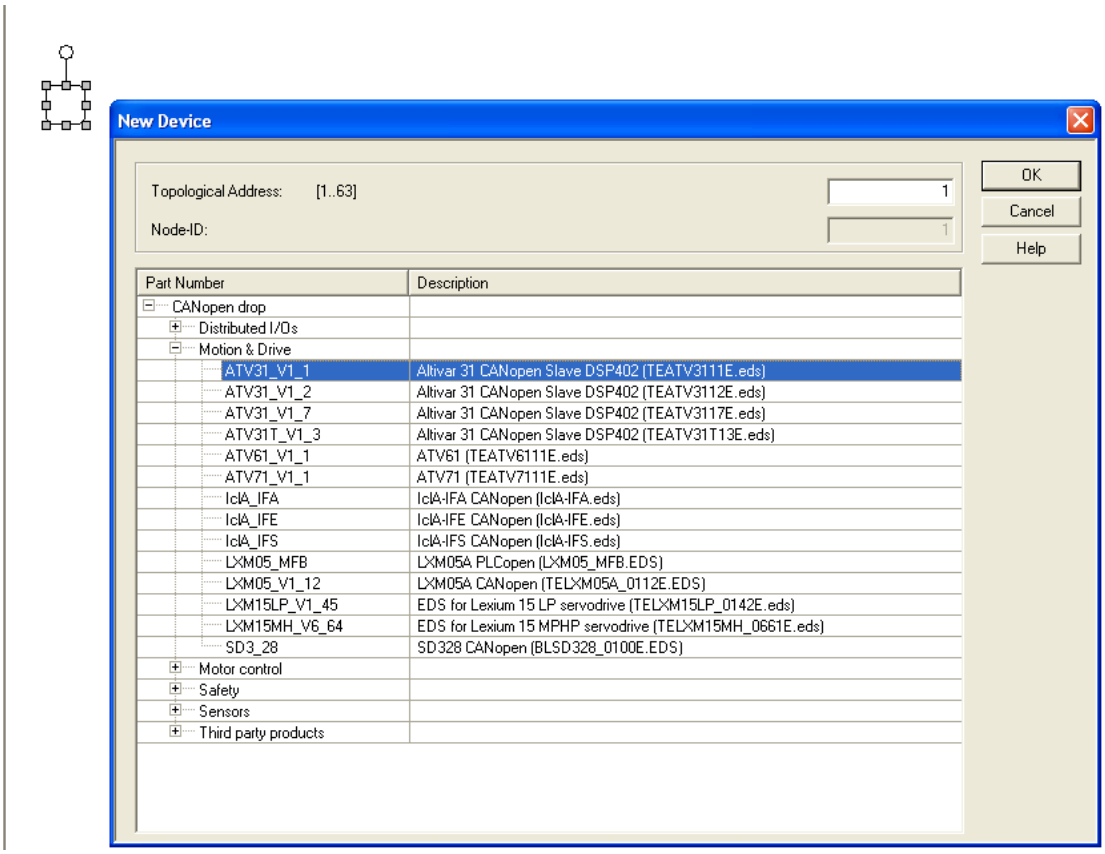
Utilização e controle do inversor de freqüência

Em muitos casos é necessário fazer com que o CLP comunique-se com o inversor de freqüência a fim de controlar velocidade de motores, em sua aplicação mais utilizada.

Podemos fazer uma configuração normal no CLP, lembrando de prestar muita atenção no modelo da CPU, que no caso de nosso estudo é a P342030 (ethernet/CANopen).

Na janela lateral esquerda que apresenta uma vista em estrutura do projeto, surgirá uma pequena via em “configuration” específica para CANopen.

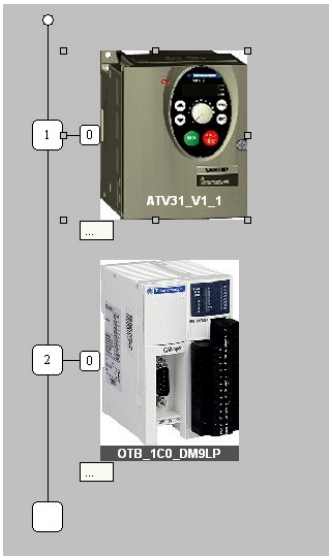
Ao clicar neste ícone uma segunda janela especifica para configuração do CANopen é aberta, e para que o CLP possa reconhecer o inversor como um equipamento válido de rede, devemos declará-lo, para tanto podemos dar duplo clique no quadrado, sendo que surgirá uma lista conforme a figura abaixo:



Na guia “Motion & Drive” na janela “New device” podemos selecionar o tipo de equipamento que utilizaremos, este tipo do equipamento pode ser encontrado no próprio equipamento , que no nosso caso é o “ATV31_V1_1”, clicando em OK, posteriormente.

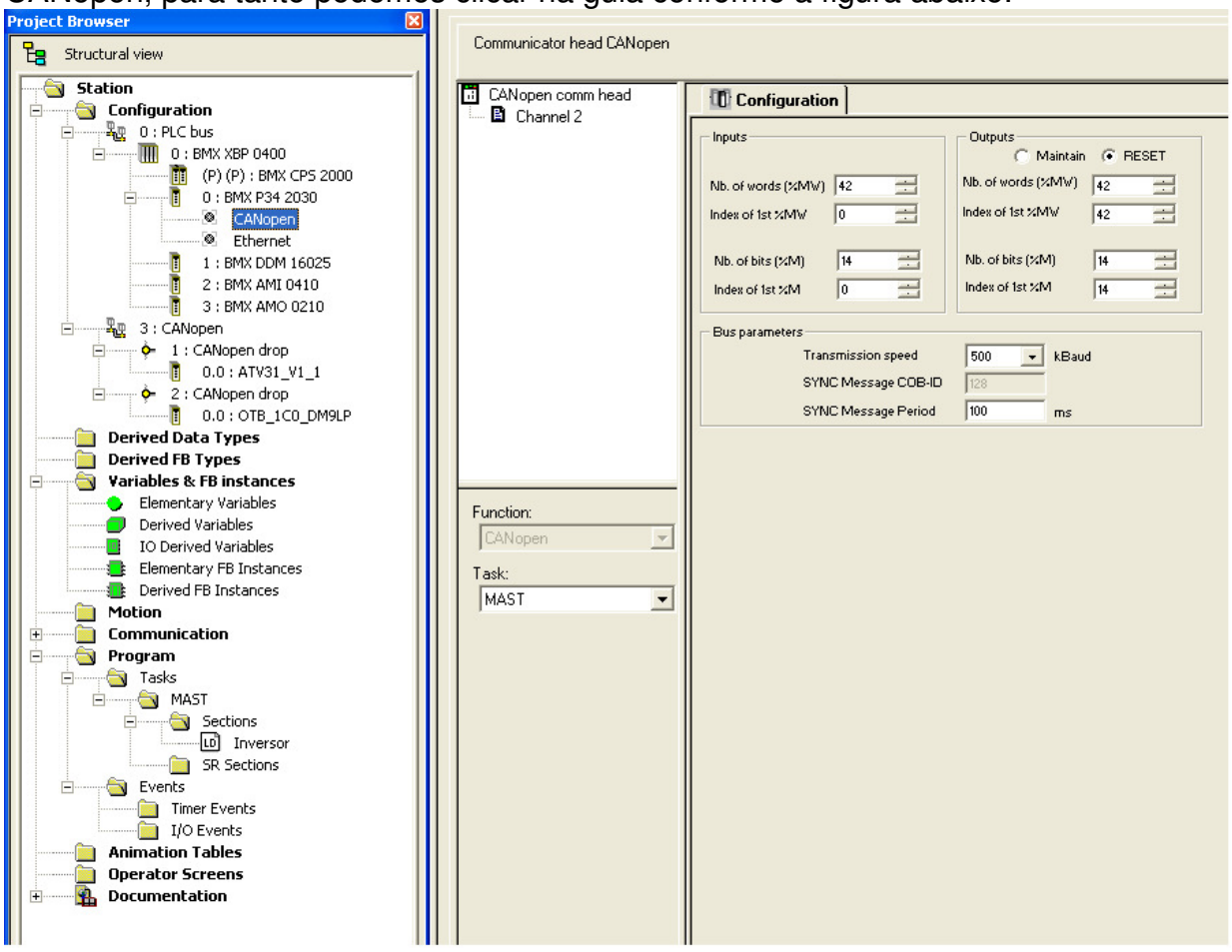
Em nosso caso possuímos ainda um comunicador “OTB_1C0_DM9LP”, que estabelece comunicação com o painel e o CLP, de maneira que também temos que “avisar” ao CLP que este equipamento também têm a necessidade de se comunicar, então novamente com um duplo clique no quadrado vazio podemos abrir a janela e selecionar o comunicador.

Após inserir deve aparecer uma tela como no exemplo abaixo:



]

Faz-se de extrema importância configurar a comunicação e memória do CANopen, para tanto podemos clicar na guia conforme a figura abaixo.



É necessário observar que o número de memórias para entrada e saída do inversor necessitam de uma banda mínima para a troca de dados, sendo assim, utilizamos o valor mínimo de 42. É importante ainda resaltar que a taxa de transmissão (Transmition speed) deve ser de acordo com a velocidade de transmissão do equipamento utilizado, em nosso caso, esta velocidade é de 500 KBaud.

Para que possamos ligar ou alterar a velocidade do motor ou ainda acionar e realizar leituras do módulo IO é necessário enviar valores para determinadas variáveis internas. Estas variáveis estão organizadas em uma tabela exclusiva de cada dispositivo chamada PDO (Process Data Object) que pode ser encontrada clicando em CANopen (1) e posteriormente no componente que se deseja visualizar os dados, como exemplificado na figura abaixo:

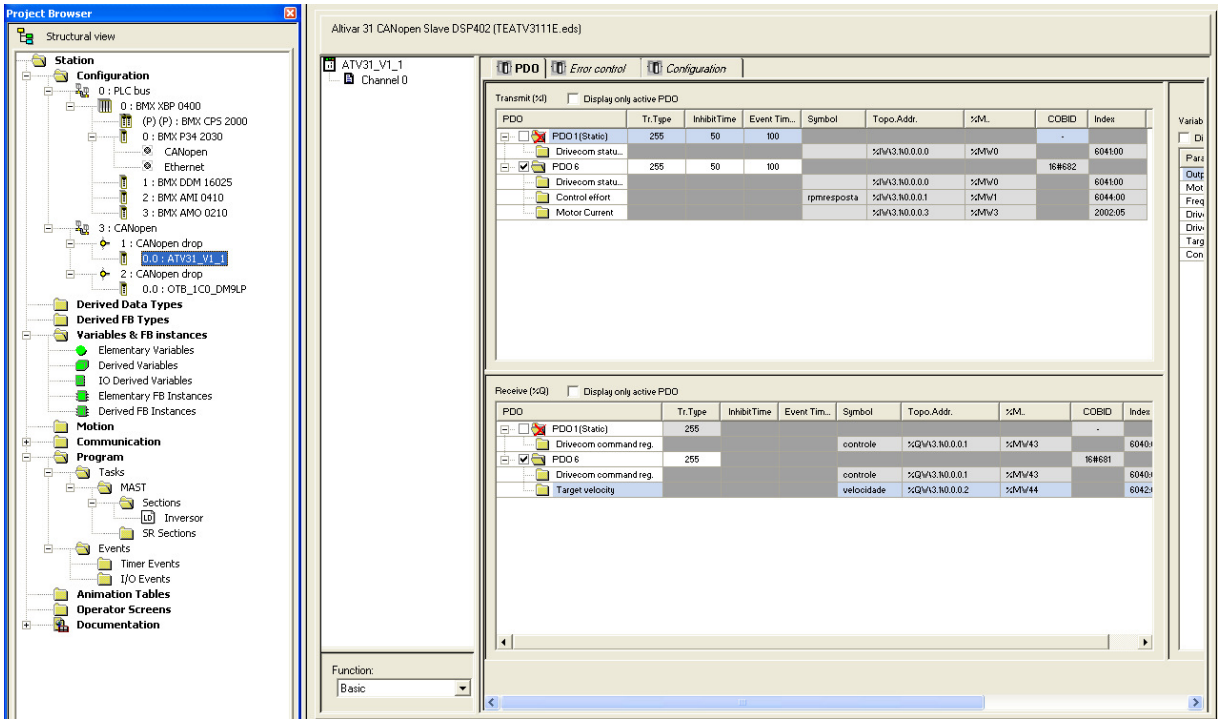
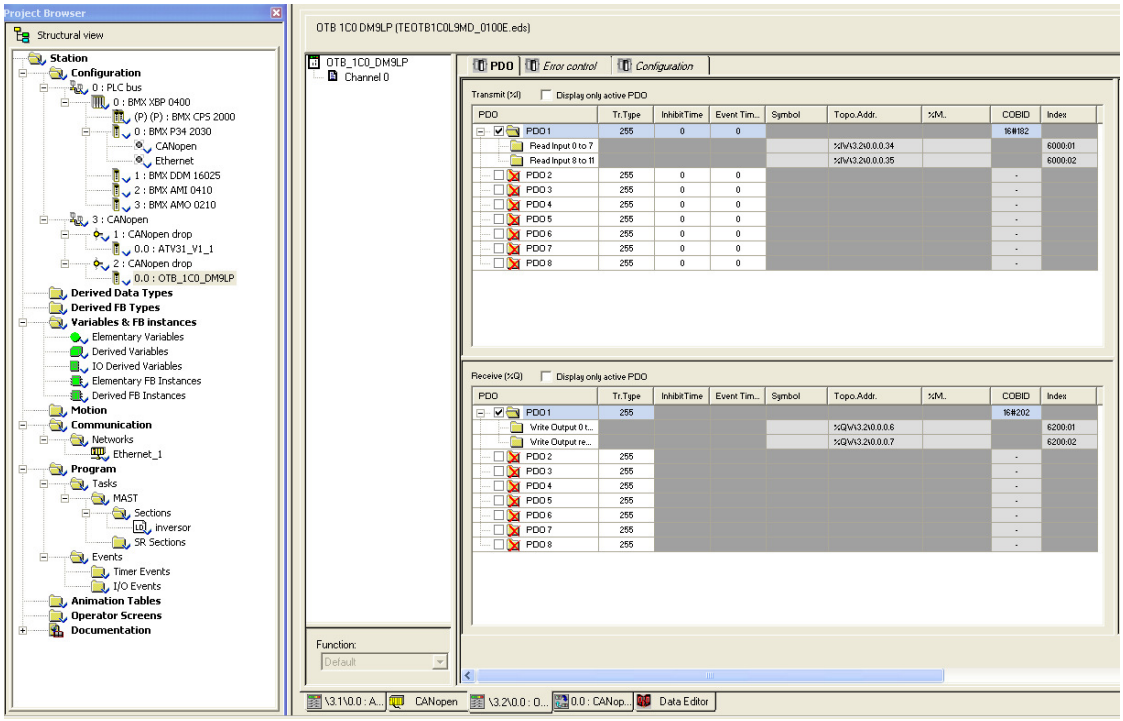


Imagem referente ao PDO do inversor.

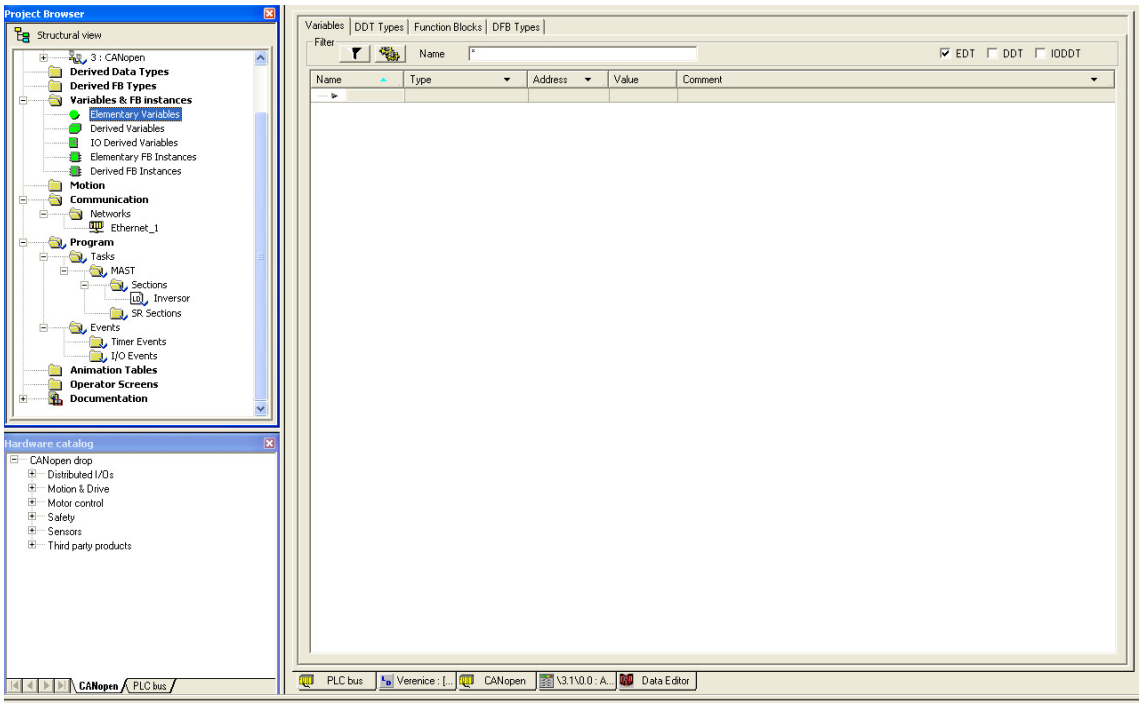
No PDO do inversor podemos observar a existência de duas telas de endereçamentos, uma referente ao recebimento de dados (Transmit (%i)), e outra referente a transmissão de dados (Receive (%Q).
É válido ressaltar que a informação acima está correta, pois devemos entender que é como se nós fossemos o equipamento.
Nesta primeira podemos citar o endereço de “Control effort” (do ING – LIT – Controle de esforço), que é responsável por nos enviar informações inerentes a velocidade do motor.
Já em “Receive” podemos encontrar dois endereçamentos referentes ao PDO ativo, sendo que “Drivecom command reg” é responsável pela criação de uma plataforma que habilita o CLP a controlar o inversor, e em “Target velocity” podemos controlar a velocidade do motor.
Todos os endereçamentos citados encontram-se na coluna “Topo. Addr”.
No canto esquerdo inferior da tela podemos encontrar um menu de nome “FUNCTION”, neste podemos selecionar a o nível de informações desejadas em “basic”, “standard” e “Extended” isso nos permite ler, além da rotação do motor,a temperatura.

Agora falaremos sobre o PDO do módulo IO (OTB_1C0_DM9LP). Neste também podemos encontrar duas telas conforme a figura abaixo:

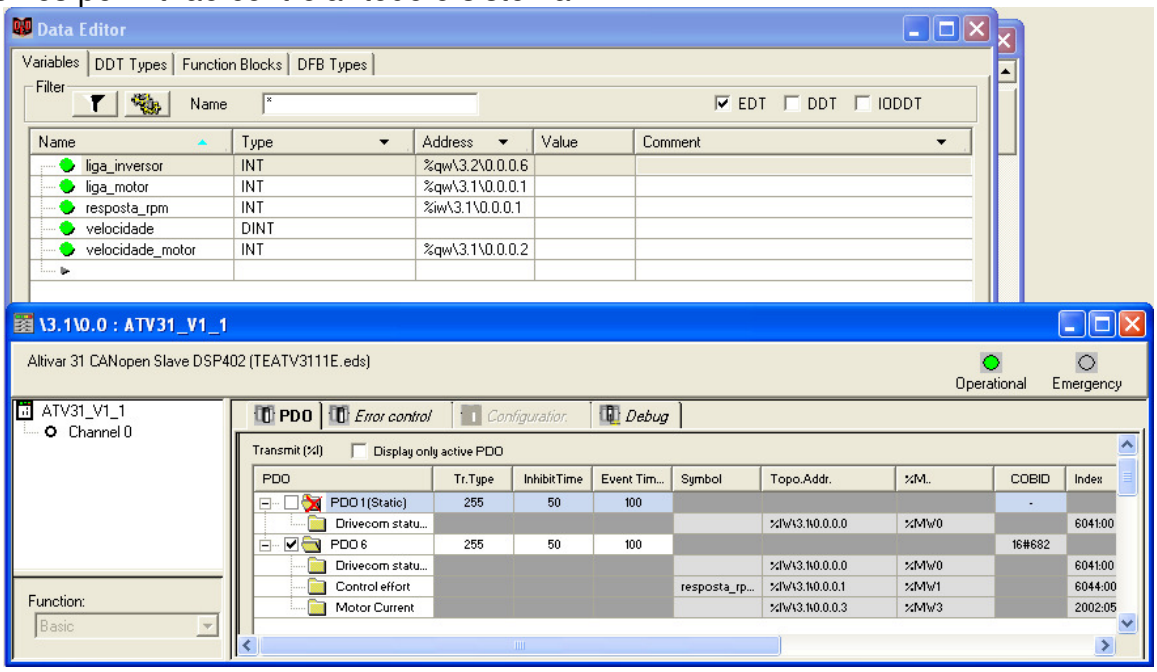


Uma é referente ao recebimento de dados (Transmit (%i)), e outra referente ao acionamento de saídas (Receive (%Q)). Em transmit encontramos duas variáveis referentes a leitura das entradas, e em receive temos variáveis referentes ao acionamento das saídas.

Tendo em vista que já conhecemos as variáveis que serão utilizadas na configuração, devemos nomeá-las pois não é possível inserir diretamente no programa. Para isso temos uma tabela onde podemos declará-las.
Para tanto podemos clicar no ícone “Elementary Variables” que se encontra na pasta “Variables &FB instances”, conforme é mostrado na figura abaixo:



Ao dar duplo clique no ícone a seguinte janela surgirá, sendo que na coluna denominada “Name” deve-se inserir o nome, e na coluna “Address” inserir os endereços que vimos nas PDO’s. Podemos consultar esta tabela a qualquer tempo, acionando-a novamente. É importante que tenhamos ciência destes endereços, pois eles nos permitirão controlar todo o sistema.

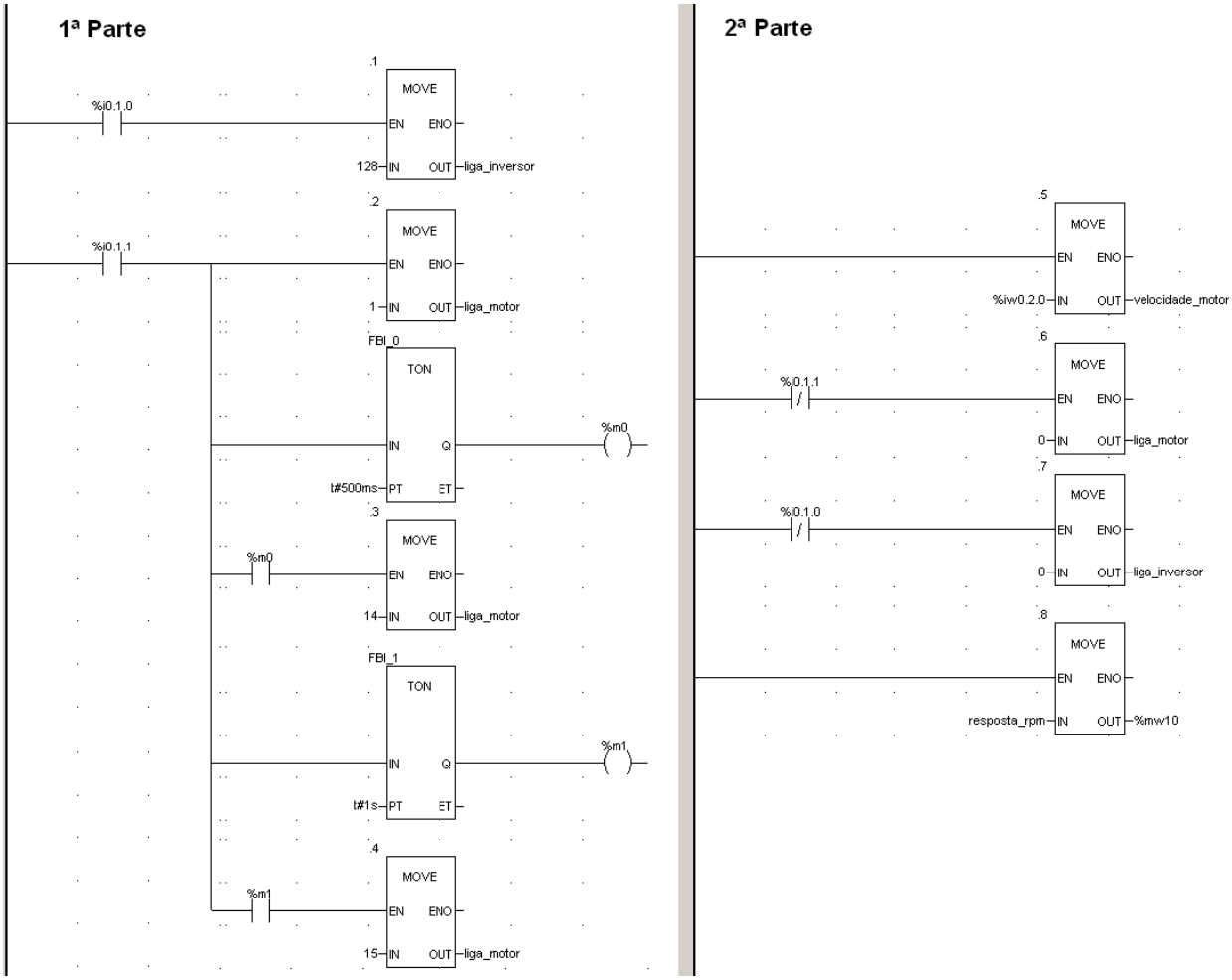


Devemos observar que em “Type” / “velocidade” a variável deve ser “DINT”, para que comporte a informação.
Finalizando o processo acima descrito, podemos iniciar a programação do inversor, que consiste em mover três números para o mesmo, e para tanto utilizaremos o bloco “move”.
Detalharemos o programa a seguir, passando primeiramente pela tabela de variáveis.

Name	Type	Address	Objetivo
Liga_inversor	INT	%qw\3.2\0.0.0.6	Sela o contator de alimentação
Liga_motor	INT	%qw\3.1\0.0.0.1	Aciona o motor
Resposta_rpm	INT	%iw\3.1\0.0.0.1	Envia ao CLP a rotação do motor
Velocidade	DINT		Alterar a velocidade para o CLP
Velocidade_motor	INT	%qw\3.1\0.0.0.2	Altera a velocidade no motor

Obs.: No item “velocidade” alteramos a velocidade no programa para alterar a rotação do motor. No programa abaixo não estamos utilizando este item, pois variamos a velocidade com o auxílio de um potenciômetro, através da entrada analógica do CLP, porém podemos modificar a velocidade manualmente no programa.

Veja um exemplo do programa a seguir:



- Linha 1 = Enviamos o valor 128 (10000000_2) para o endereço responsável pelo acionamento do inversor, denominado de "liga_inversor", acionando assim o contator responsável pela alimentação do mesmo.

- Linha 2 = Esta linha está subdividida em várias partes, pois aproveitamos o contato %i0.1.1 para todos eles. Em "MOVE 2", enviamos um byte 1, responsável por retirar o inversor do estado de "stop" e permitir sua habilitação.

Foi necessária a colocação de um temporizador, para que o inversor tenha tempo hábil para reconhecer o byte, caso não coloquemos este, o inversor pode entrar em conflito.

Enviamos então o byte "14" para o inversor que o prepara para a partida.

Novamente temos um temporizador para garantir o tempo de leitura do inversor, e transportamos finalmente o sinal 15 para concretizar a partida.

Linha 3 = Neste bloco estamos movendo para o CLP e para o inversor o valor numérico correspondente a velocidade que desejamos no motor, este valor pode ser escrito ou imputado com a ajuda de um potenciômetro, que é o nosso caso.

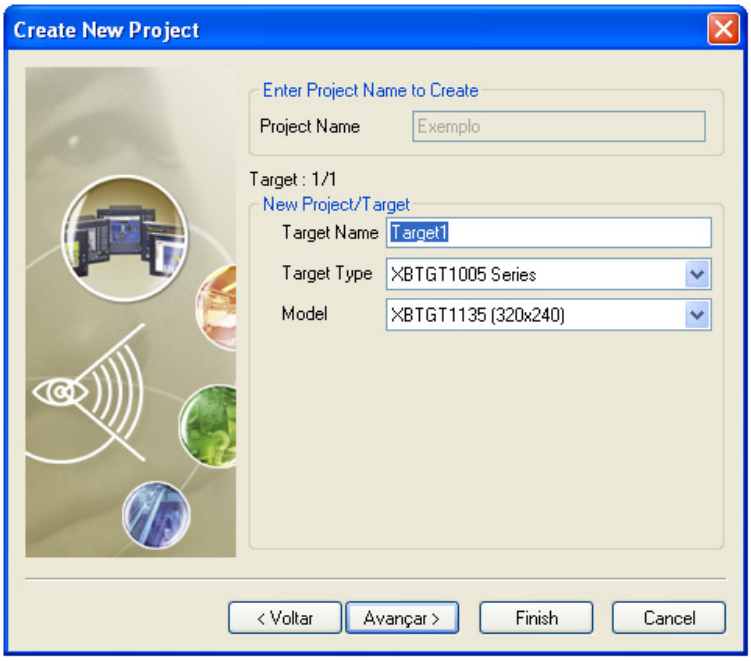
Linha 4 = Movemos o valor "0" para o "liga_motor", para desligá-lo quando necessário.

Linha 5 – Quando levamos o valor "0" para "liga_inversor", estamos na realidade desabilitando o inversor.

Linha 6 – Quando o motor gira o inversor é capaz de realizar a leitura de rotação do mesmo, desta forma utilizamos este valor a nosso favor, fazendo com que esta informação seja exibida em uma interface para que possamos monitorar.

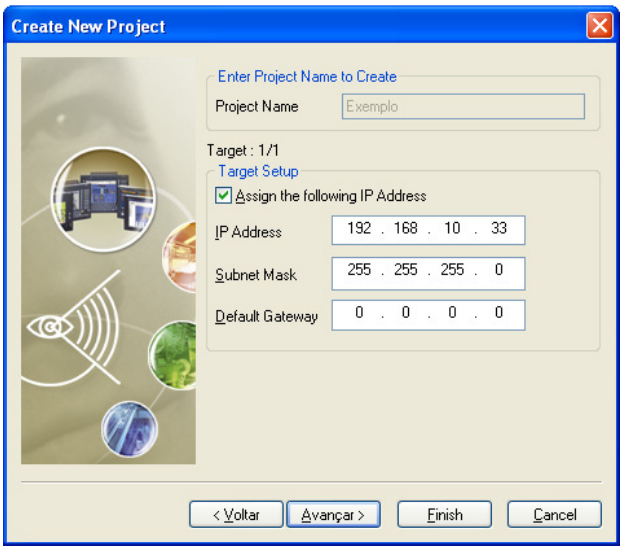
VIJEO-DESIGNER

Praticamente todas as aplicações de controladores programáveis necessitam de uma IHM, ou interface homem-máquina. O operador precisa interagir com a máquina ou processo, recebendo alarmes, comandando operações, alterando parâmetros, visualizando a situação do processo, recebendo diagnósticos do CLP, entre outras funções.
Porem é necessário a criação de uma plataforma para que esta comunicação entre homem/Máquina seja possível, desta forma então desenvolvemos uma espécie de sistema operacional para IHM , e para tanto utilizaremos o VIJEO-DESIGNER.
Devemos primeiramente clicar no ícone ao lado, uma pequena janela nos trará opções de escolha, se queremos criar um projeto novo, abrir um último projeto e abrir um projeto qualquer, podemos selecionar “Create a new Project” posteriormente em “Avançar”, e após este processo uma janela surgirá para que possamos dar um nome ao projeto e se quisermos criar uma senha para edição do mesmo, após criar o nome do projeto uma janela como a do exemplo abaixo surgirá.

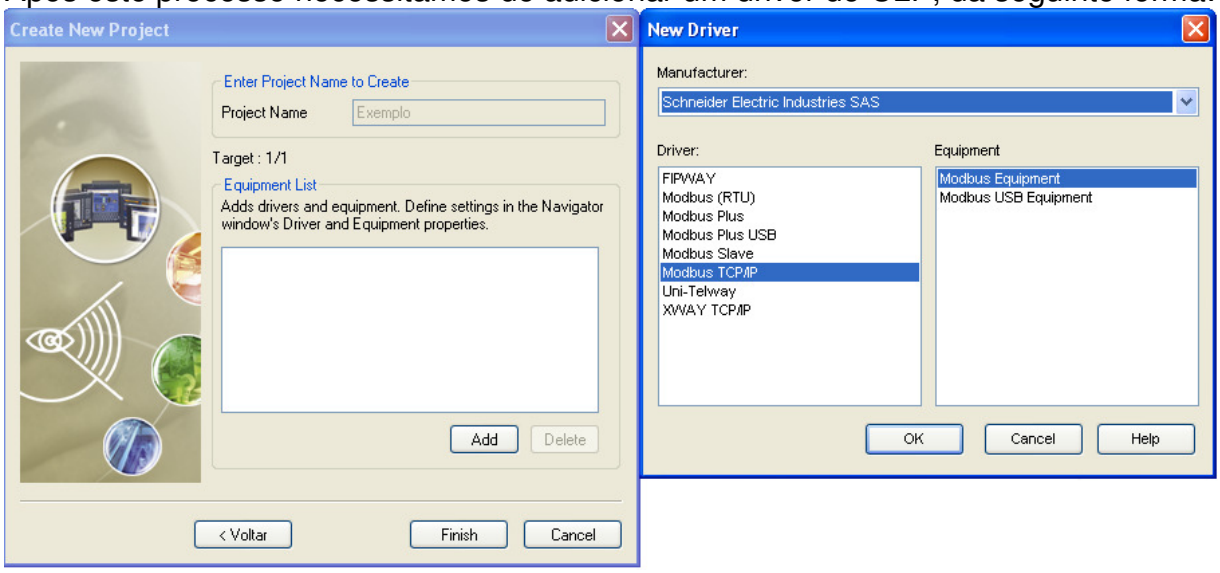


- Devemos definir um nome para o terminal “Target name”. Será mostrada árvore de diretório abaixo do nome do projeto.
- Em “target type” e “model”, selecione o modelo da IHM.

- Nesta janela abaixo, podemos assinalar a guia “Assign the following IP Address”, para que possamos agregar o endereço da IHM.
- Podemos então inserir o endereço de IP e a máscara de sub-rede de acordo com as especificações de rede local.
- Clicar em “Avançar”.

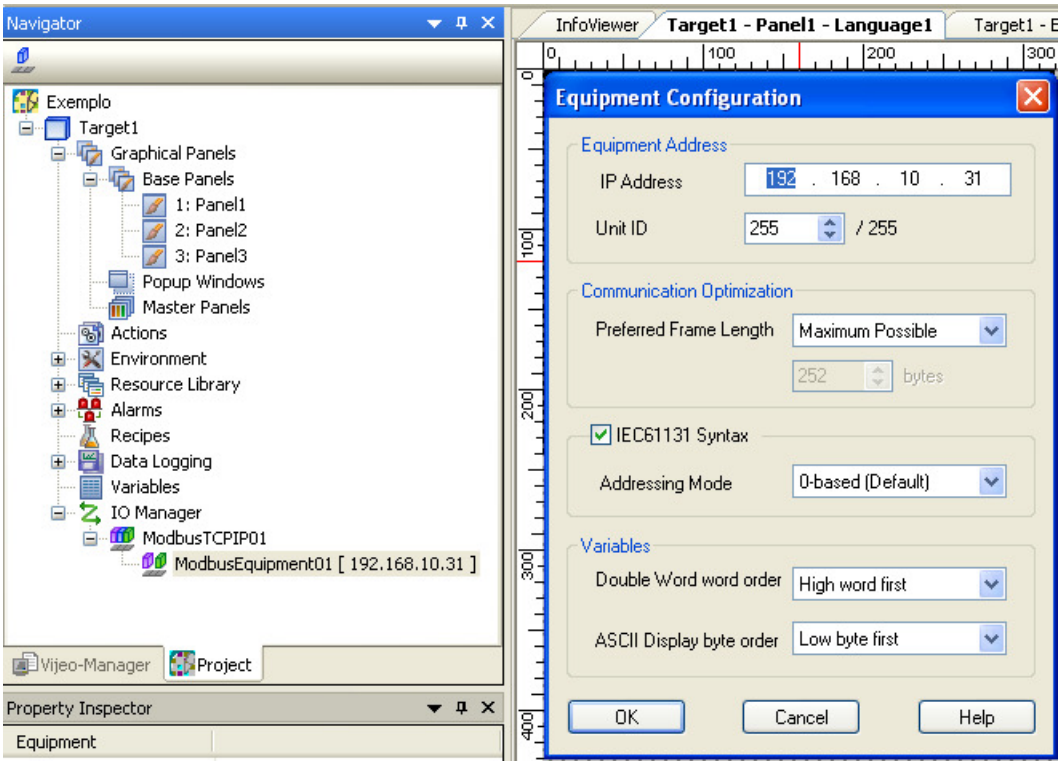


Após este processo necessitamos de adicionar um driver do CLP, da seguinte forma:



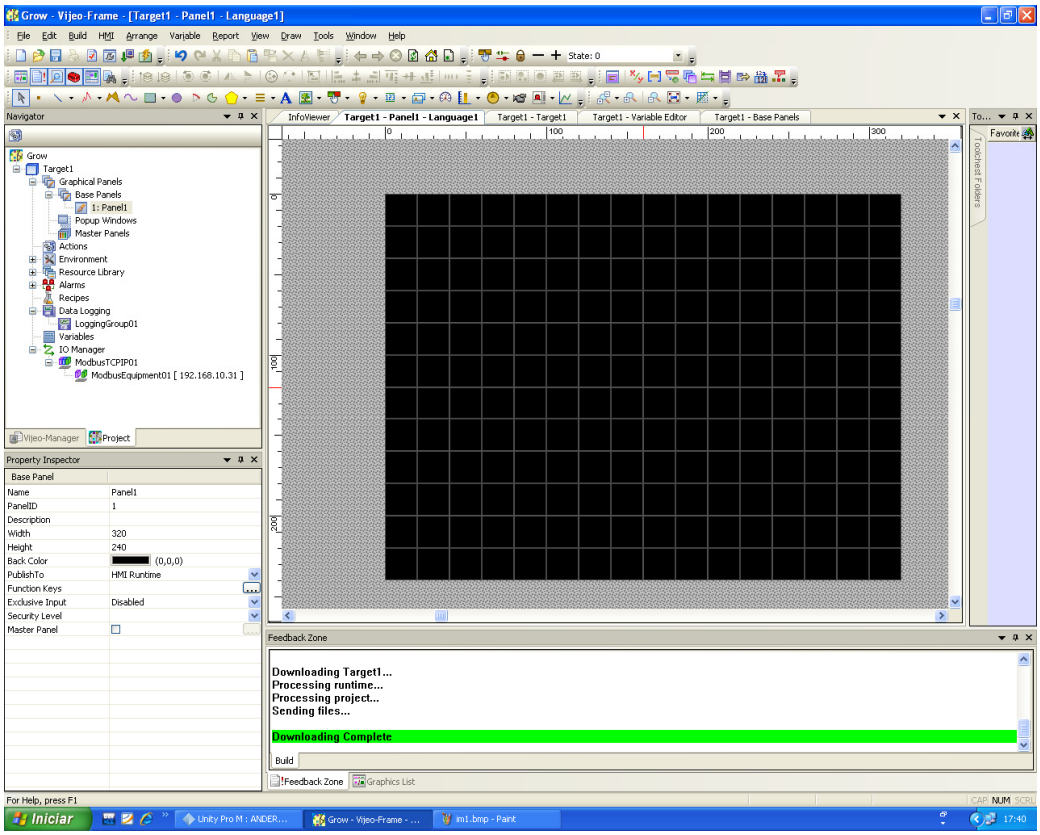
- Clicar em “Add”
- Devemos selecionar o driver desejado, que no nosso caso é Modbus TCPIP.

Para que a IHM comunique-se com o CLP devemos entrar com o endereço do CLP, desta forma, devemos clicar no ícone IO Manager, e posteriormente em “ModbusEquipament01[0.0.0.0]”, e neste momento, uma janela chamada “Equipament Configuration” será exibida, conforme a figura abaixo:



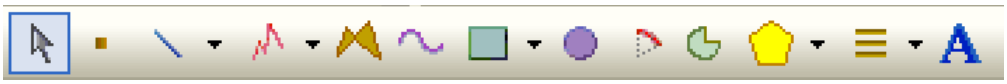
É necessário que assinalemos a sintaxe IEC61131, pois ela define o formato do endereço que o CLP irá utilizar, no caso o mesmo será na forma %Mx.

Após termos selecionado o driver, o programa criará uma plataforma e exibirá a tela de trabalho, para que possamos configurá-la para utilização da ihm, conforme a figura abaixo:



Na imagem seguinte, apresentamos uma parte da barra de ferramentas, que corresponde aos objetos gráficos, podendo citar em destaque, a “seta” que nos

permite selecionar, mover e alterar botões, e a letra ‘A’, que é utilizada para escrever textos e títulos.



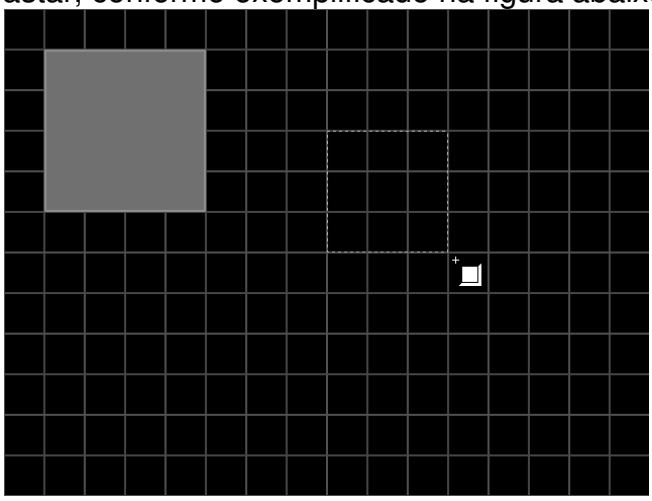
Já na figura abaixo existem ferramentas que são bastante utilizadas, sendo as mais importantes da esquerda para direita:

- Insere imagens;
- Insere botões;
- Insere lâmpadas
- Insere displays numéricos ou alpha-numéricos
- Insere display de mensagem;
- Insere um mostrador analógico de grandezas
- Insere gráfico de barras;
- Insere uma chave seletora;
- Insere vídeo;
- Insere aviso de erro, com relatório.
- Insere gráfico de tendência.

Criando botões

Faz-se necessária a criação de botões para que possamos transmitir um comando ao CLP, por exemplo, podemos ter uma memória %Mx que será acionada por este botão, e esta mesma memória, ou endereço, deve ser o mesmo para o botão e o CLP.

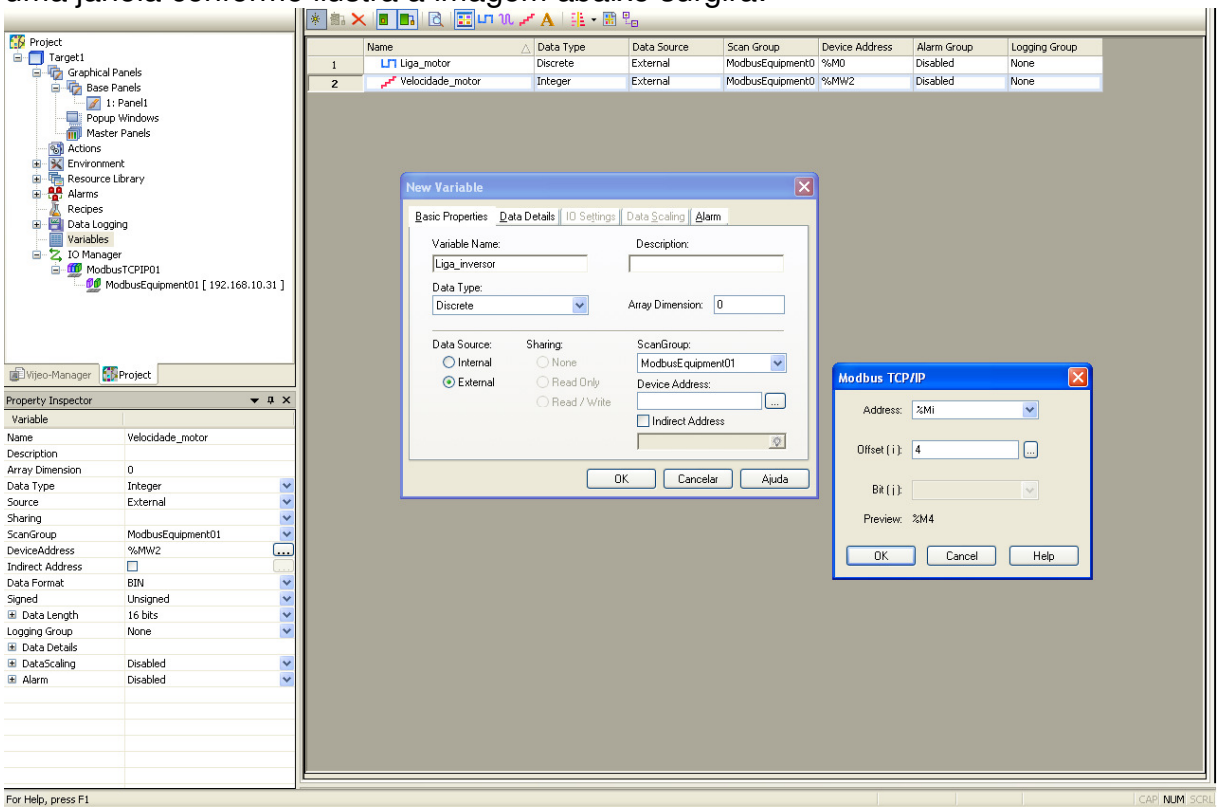
Podemos inserir um botão clicando no ícone  , desta forma o ponteiro do mouse tornar-se-á um ícone próprio da função, devemos então posicioná-lo na área de trabalho, clicar e arrastar, conforme exemplificado na figura abaixo:



Como mencionado é necessário que a variável que está ligada ao CLP, deve ser a mesma variável que está ligada ao botão.

Criando uma variável

No navegador lateral, podemos clicar na guia de nome “Variables”, neste momento uma janela conforme ilustra a imagem abaixo surgirá:



Na janela “New Variable” podemos inserir um nome qualquer para a variável em “Variable Name”, embora seja bastante comum a inserção de um nome referente a ação desempenhada, assim como no exemplo.

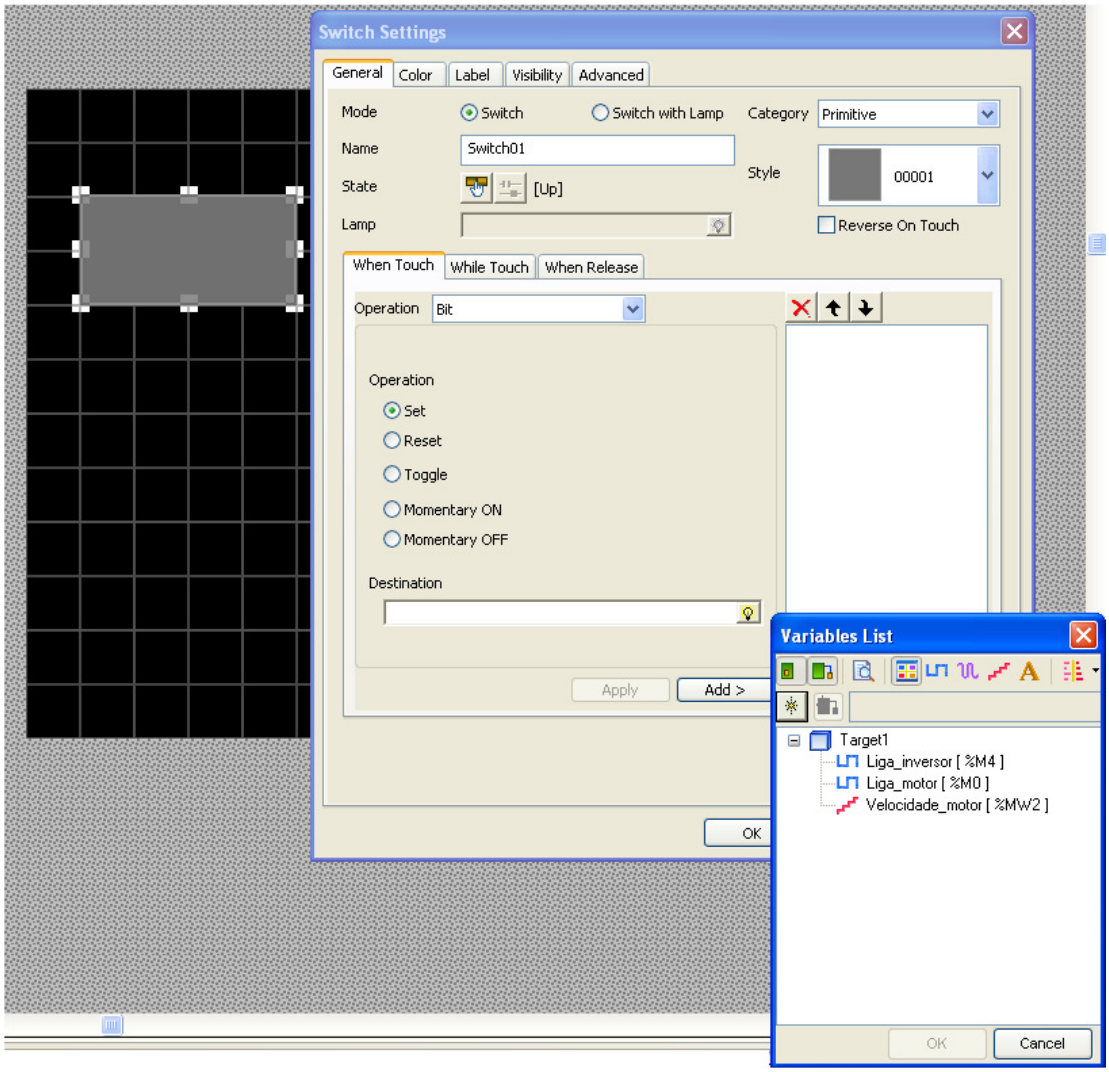
Ainda em “New Variable” devemos definir o formato da informação que o botão vai gerar em “Data Type”, podemos selecionar Discrete (Pulso digtal), Integer ( ), Float ( ), String ( ), Block Integer ( ) e Block Float ( ).

Em “Data Source” (fonte de dados) podemos selecionar “Internal”, se quisermos que o CLP utilize valores internos do programa, ou “External”, se desejamos que a origem dos dados seja externa, como botões por exemplo.

Em “Device Address” devemos introduzir o endereço ou o nome da memória que o programa utiliza para determinada variável, como exemplo podemos citar a imagem acima, onde temos a variável “Liga_inversor” atrelada a memória %m4, isto significa que no programa do CLP, %M4 aciona a área responsável pelo inversor, e assim por diante.

Finalmente podemos clicar em OK.

É necessário que atrelemos um endereço ao botão, podemos então dar duplo clique no mesmo, e a imagem abaixo surgirá:



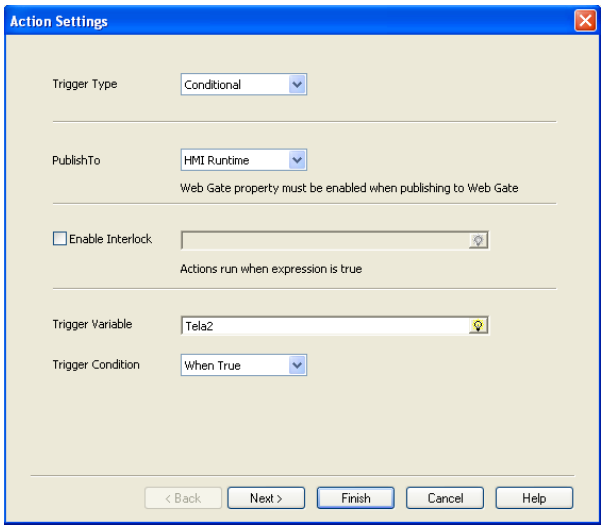
Nesta janela podemos dar um nome ao botão, aqui chamado de switch em “Name”, podemos selecionar a categoria do botão inserindo imagens tipo bitmap, e até personalizar o mesmo, nas guias “Category” e “Style”.
No campo operation selecionaremos o funcionamento do botão, tendo:
SET – Envia um pulso ao CLP, movendo um valor 1
RESET – Envia pulso negativo, movendo valor zero.
TOGGLE – Quando pressionado mantém-se neste estado até que seja pressionado novamente,
MOMENTARY ON – Mantém-se enviando pulso 1, enquanto estivermos pressionando o botão,
MOMENTARY OFF – Envia um pulso zero, enquanto mantivermos o botão pressionado.

Após ter inserido botões e imagem de fundo, podemos fazer o download para a máquina (IHM) do programa, e para tanto podemos clicar com o botão direito em “Target 1” que se encontra na barra de navegação, sendo que neste momento surgirá um menu onde encontraremos a guia “Download to (Ethernet 192.168.x.x)”, devemos então clicar sobre a guia citada, e em cerca de um minuto o programa será carregado na IHM.

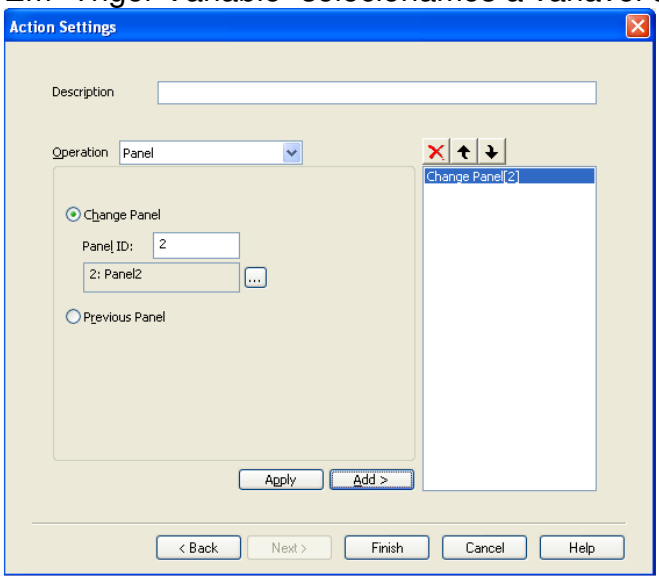
Criando e selecionando várias telas

Durante a confecção de um projeto de IHM pode vir a ocorrer que as informações a inserir em uma tela sejam numerosas demais para atrelar em uma única tela, desta forma é necessária a criação de telas extras para a inserção destes dados. Podemos prosseguir da seguinte maneira:
- Clicando com o botão direito no ícone “Base Panels”, localizado na área de navegação do Vijeo, surgirá um menu no qual devemos selecionar “New Panel”. Podemos neste ponto nomear os painéis de acordo com a nomenclatura mais adequada a especificidade do projeto.

Para fazermos a transição de telas devemos criar variáveis, conforme já mencionado, podendo ser internas ou externas.
Para que seja feita a troca de tela devemos criar ações, ou seja, funções especiais para variáveis, Para tanto podemos clicar em “Action” e “New action”, e uma tela como abaixo surgirá.

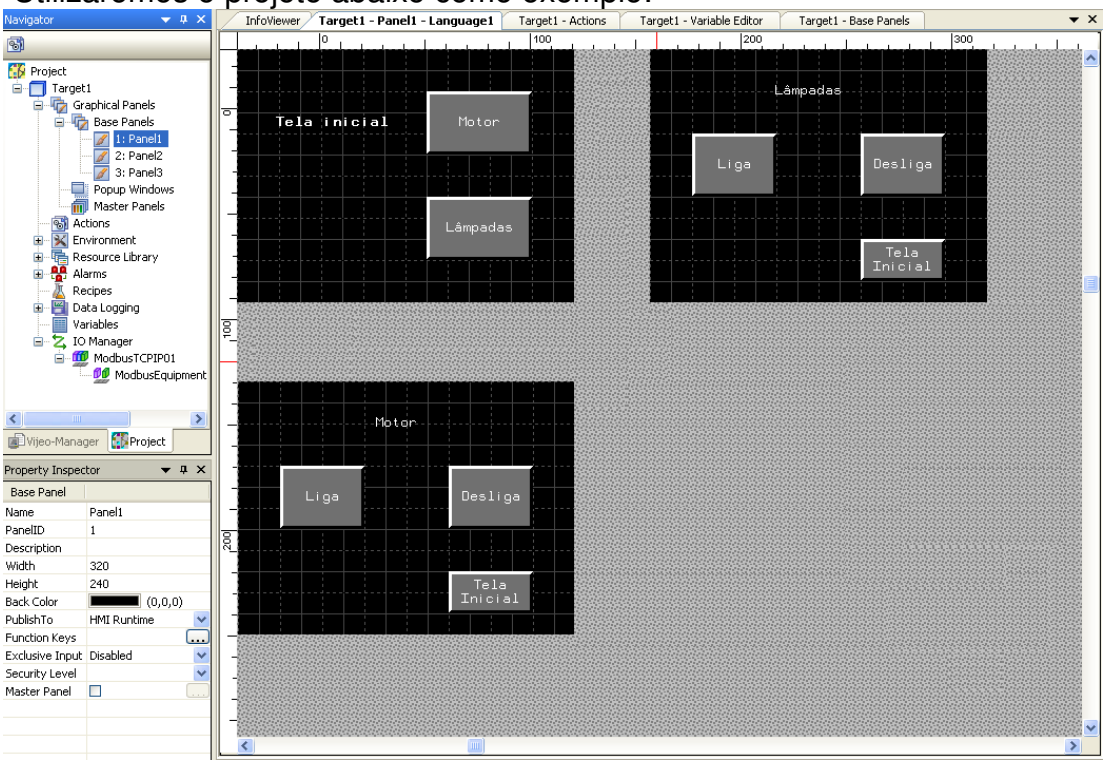


Posso criar vários tipos de ações, mas direcionando para transição de tela faremos da seguinte maneira:
Em “Triger Type” selecione “conditional”, isso significa que a transição da tela depende de uma condição, sendo esta que a variável criada seja verdadeira.
Em “Triger Variable” selecionamos a variável desejada, e clique em “next”.



Neste ponto podemos ir em “Operation” e selecionar “Panel”.
“Change panel” é utilizado para escolher o numero do painel que será exibido ao ser acionada aquela variável indicada em Triger Variable, feito isso cliquemos em “ADD”.
Para concluir, devemos clicar em “finish”.

Utilizaremos o projeto abaixo como exemplo:



Como podemos observar nosso projeto possui três telas,
Na primeira tela temos um guia que nos direciona para tela 2 e 3. Para isso temos dois botões, o botão “Motor” está atrelado com uma variável que quando acionada, nos leva para a tela 2. E para a tela 3 temos o botão lâmpadas. Caso desejarmos voltar para a tela inicial, podemos encontrar nas telas 2 e 3, um botão nos direcionando a tela principal citada.