Tutorial de Programação Básica

CLP E IHM

APOSTILA

SIMILAR
TECNOLOGIA E AUTOMAÇÃO

Rua Alagoas 2466 - CEP: 80630-050 - Tel.: 41 3074-0300 - Curitiba/Pr
www.similar.ind.br www.lsbrasil.com.br

- CRIANDO UM PROGRAMA BÁSICO NO CLP

Entradas e Saídas:

Em uma programação de CLP, podemos nos deparar com dois tipos de memórias de entradas e saídas digitais:

- Memórias de entradas e saídas físicas: **Essas memórias são utilizadas quando temos um equipamento ligado na entrada ou na saída física do CLP, como por exemplo, na entrada digital: um sensor ou um botão (chave) e na saída digital: um rele, por exemplo, que pode acionar um contator que por sua vez acionará um motor.**

Nos CLP's da LS a memória inicial de entrada física é a P0 que segue em uma sequência hexadecimal. Por exemplo, um CLP com 16 entradas digitais terá suas entradas nomeadas como: P0, P1, P2, P3, P4, P5, P6, P7, P8, P9, P0A, P0B, P0C, P0D, P0E, P0F.

Isso acontece também com as saídas que podem, dependendo do modelo do CLP, iniciar em P40, P41, P42, P43... e assim por diante. Isto é facilmente identificado olhando no próprio CLP a descrição impressa no equipamento.

- Memórias internas: **As memórias internas podem ser utilizadas em uma programação de CLP sem que haja necessidade de ter um equipamento físico ligado ao CLP. Quando criamos um programa de simulação, por exemplo, podemos utilizar memórias internas para realizar a simulação, haja vista que não utilizaremos equipamentos físicos ligados nas entradas ou saídas do CLP por se tratar de uma simulação.**

As memórias internas de BIT dos CLP's da LS são as memórias "M" e as memórias internas de WORD (16bits) são as "D".

Devemos lembrar que quando falamos de memória de BIT estamos falando que esta memória pode assumir apenas dois estados: Ligado ou Desligado (0 ou 1).

Já quando falamos de memórias de WORD, esta pode assumir diversos valores, por isso utilizamos esta memória para receber dados, como um valor numérico por exemplo.

MEMÓRIAS do CLP LS	
P	Entradas e saídas física do CLP
M	Memória interna para salvar dados em formato Bit
D	Memória interna para salvar dados em formato Word
K	Memórias retentivas
U	Memória reservada para receber dados de módulos especiais tais como entradas analógicas;
T	Temporizadores;
C	Contadores;
Z	Dedicado à função de indexação;
L	Reservada para indicar status de comunicação high speed link e P2P;
F	Reservada para Flags do sistema;

Para que esse assunto fique mais claro, vamos criar um programa básico de simulação no CLP e na IHM para que possamos entender melhor as memórias internas do CLP:

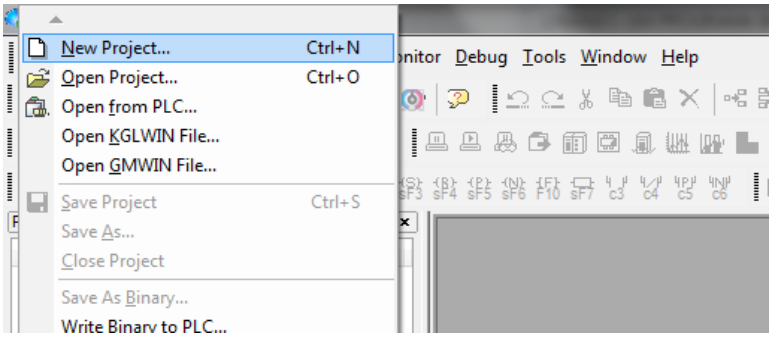
Imagine uma situação onde teremos que partir 3 motores, porém estes motores nunca poderão partir simultaneamente.

Quando o primeiro motor partir, o segundo motor deverá esperar no mínimo 5 segundos para partir (Nunca poderá partir antes dos 5 segundos).

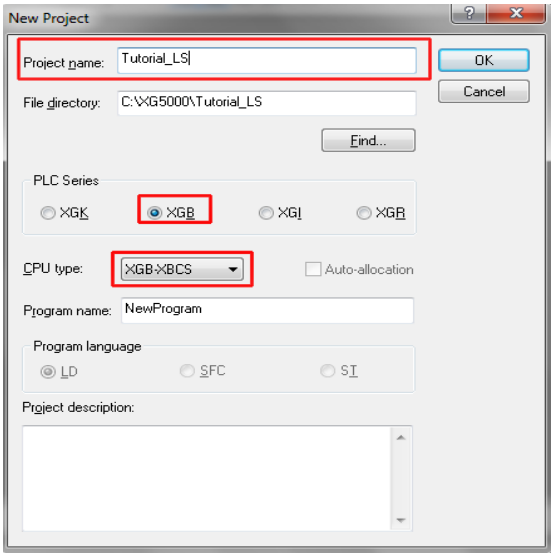
Já o terceiro motor deverá aguardar sempre, 3 segundos a mais para partir do que o segundo motor.

Vamos a partir de agora, desenvolver a lógica de programação em linguagem Ladder, para aprendermos um pouco mais sobre as memórias internas de bit e word, comparadores e operações matemáticas no CLP.

Abra primeiramente o programa XG5000 (Programa do CLP) e crie um novo projeto:

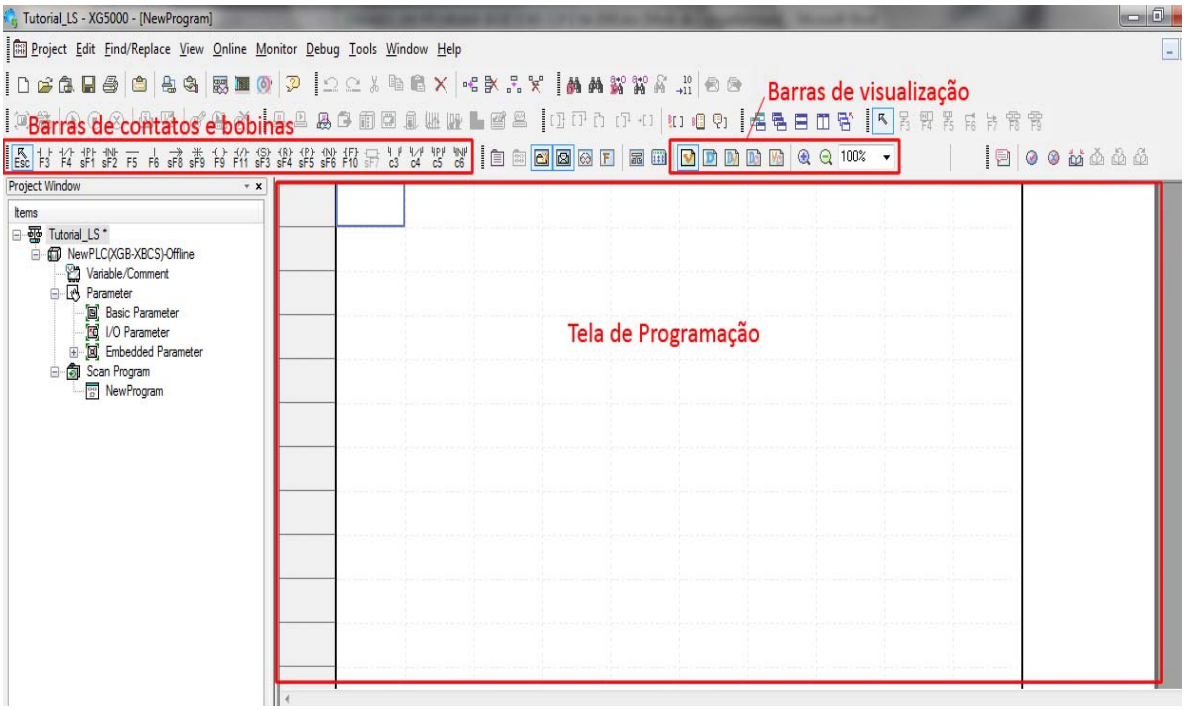


Dê um nome ao seu projeto e em seguida escolha qual o modelo de CLP da LS que você deseja utilizar, neste exemplo vamos utilizar um CLP XBC Standard:

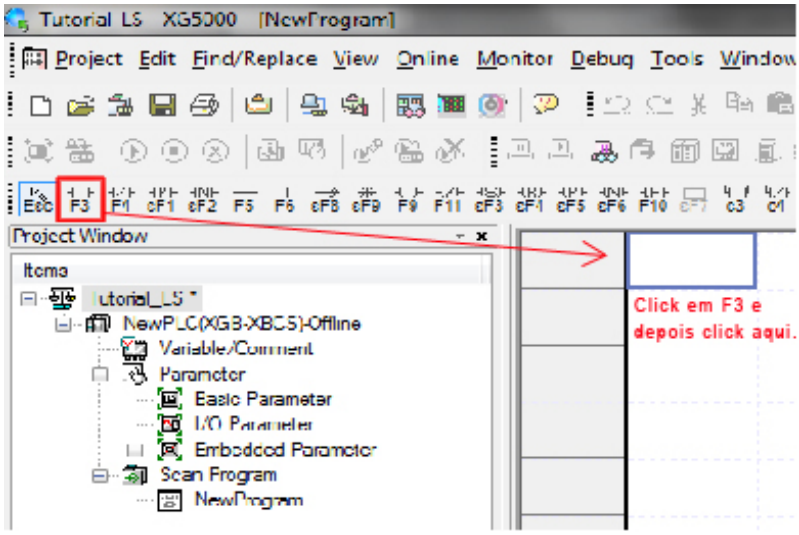


A família de CLP "XGB" da LS é divida em H (High Performance), S (Standard) e E (Economic)

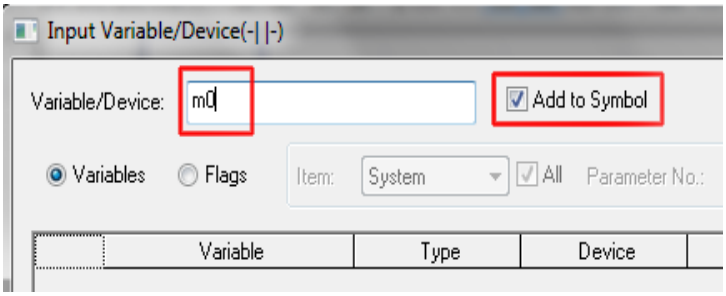
Teremos a seguinte tela:



Vamos agora inserir uma memória interna para ser o nosso botão de "LIGA" (entrada digital). Click no Contato Normalmente Aberto (NA) na barra de contatos bobinas:

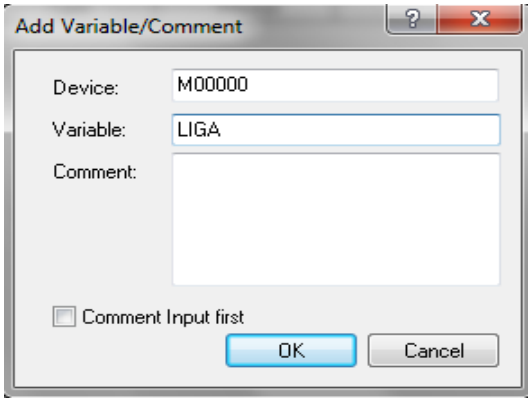


Insira este contato na tela e declare ele como uma memória M0:

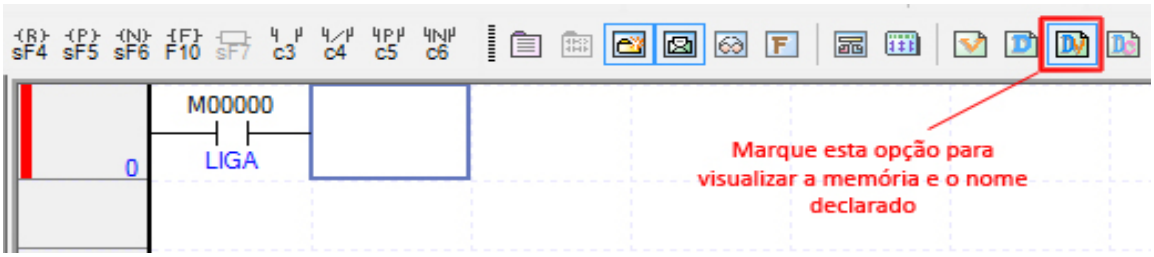


E caso não esteja marcada a opção "Add to Symbol", marque-a para podermos declarar também um nome à memória M0.

Escreva na janela seguinte "LIGA" como na imagem abaixo:



Pronto nossa primeira memória interna de bit está inserida no programa:



Como explicado anteriormente, caso tivéssemos um botão na entrada física do CLP, deveríamos então, colocar a memória de entrada digital física P0 no lugar da memória M0.

Continuando a programação vamos agora inserir nossa primeira saída. Esta saída será também, uma saída interna de memória "M1". Isto significa que não iremos acionar nada físico, pois M1 é uma memória interna. Caso fossemos acionar um rele físico, utilizaríamos uma memória digital de saída física, como por exemplo, a P40. Esta saída P40 seria responsável em acionar um rele que acionaria um contator e que por sua vez acionaria um motor.

Note que as memórias internas de bit "M" podem ser utilizadas como entradas ou saídas digitais.

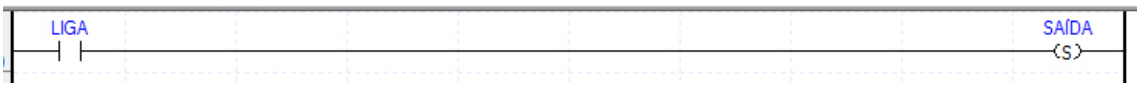
Mas para continuar os nossos estudos, precisamos entender o conceito de saídas utilizando bobinas de "Set" e "Reset".

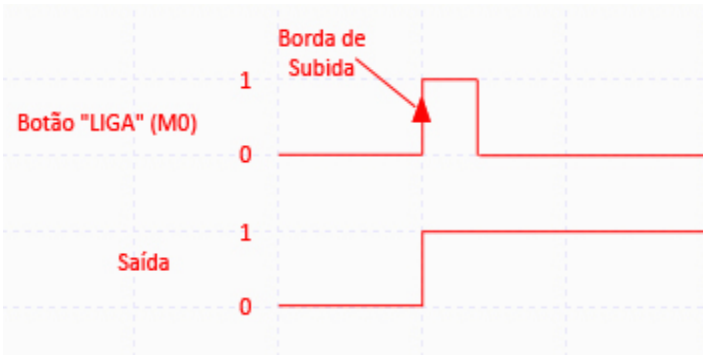
BOBINAS “SET” E RESET”

Outro ponto que devemos tomar conhecimento é no que diz respeito às bobinas de “Set” e “Reset”.

Para quem já trabalhou com as lógicas de contatores, deve conhecer bem a lógica para “Selar” o contato de uma bobina.

Na **programação do CLP não precisamos utilizar este “Contato de Selo” para que uma bobina permaneça ligada, pois temos uma bobina chamada “Bobina de Set”. Esta bobina diferente de uma bobina normal, precisa de apenas um pulso para ligar e continuar ligada. Na verdade a Bobina de Set necessita de uma borda de subida para ligar e permanecer ligada até que seja “Resetada”.**

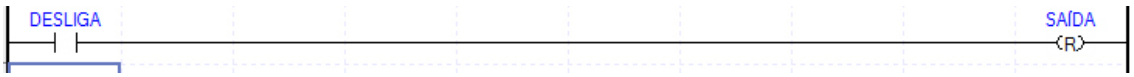


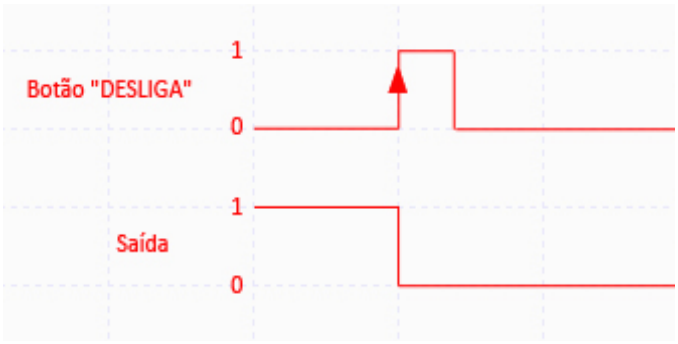


Conseguimos perceber na imagem acima, que a Saída ligou exatamente na borda de subida da entrada M0 (momento em que o operador pressionou o botão de liga) e como a saída é uma Bobina de Set, permaneceu ligada mesmo depois que a entrada M0 foi desligada (momento em que o operador soltou o botão de liga).

Para desligar a saída agora, precisamos inserir um comando de “Reset”.

Então para desligarmos a Saída usaremos um botão de "Desliga" (outra memória diferente de M0). Este botão irá dar um novo pulso, mas agora no comando "Reset" da bobina de Saída:



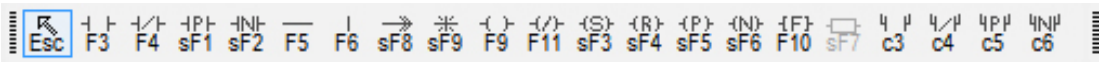


Conseguimos perceber agora que no exato momento que o botão de Desliga foi apertado (borda de subida), a Saída foi para nível logico "0", ou seja, a Saída foi "Resetada" (Desligada).

Sempre quando utilizarmos uma bobina de SET nós devemos utilizar um Push-Button (Botão Momentâneo) para acioná-la.

Este conceito de "Set e Reset" é muito importante para prosseguirmos com a nossa programação.

Contatos e Bobinas do CLP LS:



F3 **– Contato Normalmente Aberto;**

F4 – Contato Normalmente Fechado;

sF1 – Contato de Transição Positiva: Envia apenas um pulso na borda de subida: 

sF2 – Contato de Transição Negativa: Envia apenas um pulso na borda de descida: 

F5 – Cria uma linha reta na horizontal;

F6 – Cria uma linha reta na vertical;

sF8 – Linha de preenchimento horizontal;

sF9 – Inverte a função do contato;

F9 – Bobina Normalmente Aberta: Enquanto esta bobina receber sinal de nível lógico 1 ela ficará acionada, ao contrario disto, desacionará.

F11 – Bobina Normalmente Fechada: Enquanto esta bobina receber sinal de nível lógico 0 ela ficará acionada ao contrário disto, desacionará.

sF3 – Bobina de Set: Esta bobina é acionada quando recebe apenas um pulso de nível lógico 1. Diferente da bobina F9, esta ficará acionada mesmo que o pulso passe para nível lógico 0.

sF4 – Bobina de Reset: Esta bobina é responsável em colocar uma memória acionada pela função Set em nível lógico 0. Quando acionamos uma memória utilizando a função anterior sF3, em algum lugar da programação teremos que resetar essa memória utilizando esta função sF4.

sF5: - Bobina de Transição Positiva: Esta bobina é acionada apenas na borda de subida de um contato acionador e fica ligada somente no tempo de duração da borda de subida (1 scan).

sF6: - Bobina de Transição Negativa: Esta bobina é acionada apenas na borda de descida de um contato acionador e fica ligada somente no tempo de duração da borda de descida (1 scan).

sF10 – Funções: Neste ícone, estão todas as funções que podem ser utilizadas na programação do CLP, tais como: comparações, temporizadores, contadores, controle de posição, lógicas de operação, função END etc.

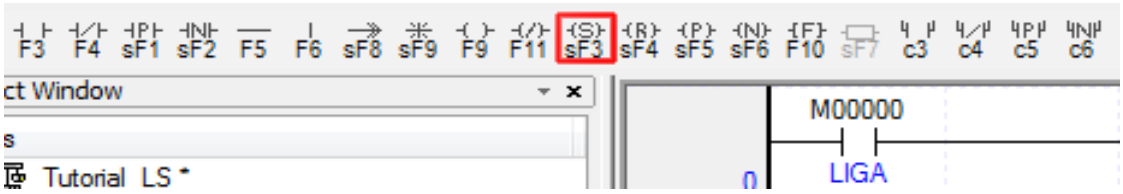
C3 – Contato Paralelo Normalmente Aberto;

C4 – Contato Paralelo Normalmente Fechado;

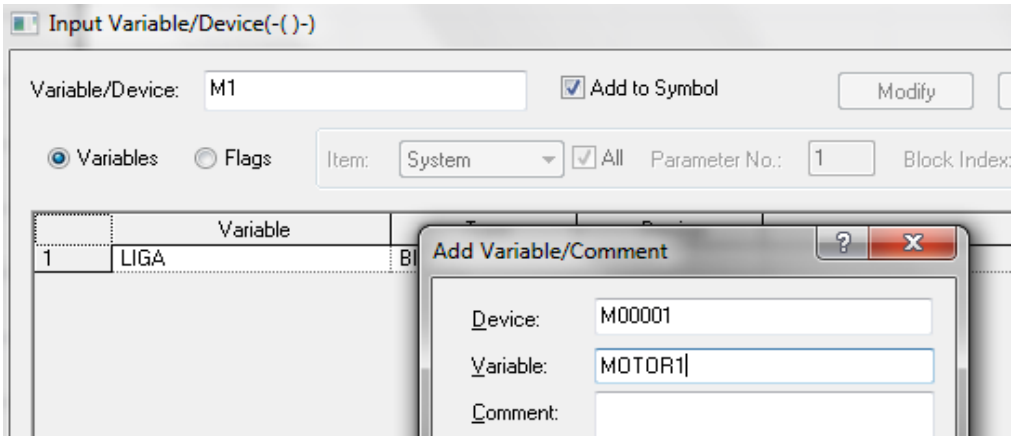
C5 – Contato Paralelo de Transição Positiva;

C6 – Contato Paralelo de Transição Negativa;

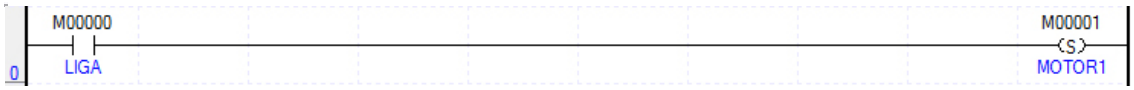
Para criar nossa saída interna com uma bobina de SET, click na bobina que possuí a letra "S" na barra de contatos e bobinas e insira na tela:



Declare como M1 e insira o nome de "MOTOR1":



Pronto, a primeira linha de programação para acionamento do primeiro motor está concluída:



A memória M1 é a nossa bobina de saída. Um pouco diferente do que temos na prática, uma bobina pode conter aqui, vários contatos auxiliares. Por ser um software, não ficamos limitados a quantidade de contatos, como por exemplo, em um contator que fisicamente possuí um número limitado de contatos auxiliares. Aqui no software podemos inserir quantos contatos auxiliares NA ou NF "desejarmos" da bobina de saída, neste caso M1. Claro, respeitando a limitação de cada modelo de CLP.

Sabendo disso, vamos agora utilizar um contato NA da bobina M1 para acionar um temporizador. Existem vários tipos de temporizadores que podem ser utilizados em uma aplicação. No nosso caso queremos aguardar um tempo de no mínimo 5 segundos para partir o segundo motor. Isso significa que precisamos de um atraso de 5 segundos na ligação desse motor. Chamamos este temporizador de retardo na ligação de "TON".

TEMPORIZADOR "TON" E "TOFF"

- Para temporizarmos algo na programação do CLP precisamos conhecer bem como funciona a montagem do temporizador no programa.

Nos CLP's da LS a montagem do temporizador com tempo "Fixo" é da seguinte maneira:

TON T0 50

Onde: - TON: Significa retardo na ligação;

- T0: Primeiro temporizador (T0, T1, T2...)

- 50: Tempo fixo de 50 décimos de segundo, ou seja, 5 segundos.

Então quando falarmos de 5 segundos coloque o número 50 no CLP.

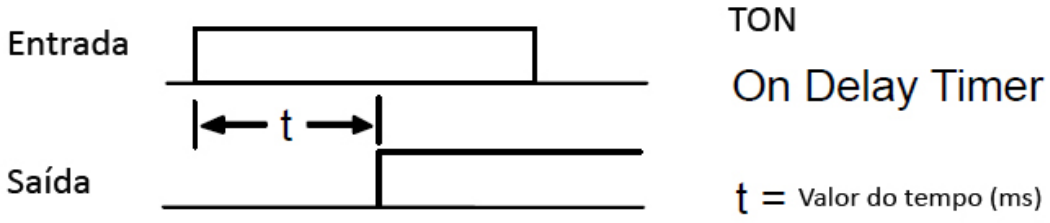
***Considere a unidade de tempo dos temporizadores "T0" a "T499" em 100ms.**

1 décimo de segundo = 100 milissegundos

50 x 100ms = 5000ms = 5 segundos.

TIMERS		
TIMER	**START**	**END**
100ms	**T0**	**T499**
10ms	**T500**	**T999**
1ms	**T1000**	**T1023**

Esquema do temporizador TON:



Neste caso acima foi inserido um número fixo de 50 décimos de segundo ou 5 segundos. Seu comportamento será da seguinte forma: Ao energizarmos e deixarmos energizado o temporizador T0, o mesmo iniciará a contagem e ao chegar em 5 segundos acionará seus contatos auxiliares. Seu funcionamento ficará mais claro quando inserirmos este temporizador no programa.

- Nos CLP's da LS a montagem do temporizador com tempo "Variável" é da seguinte maneira:

TON T0 D0

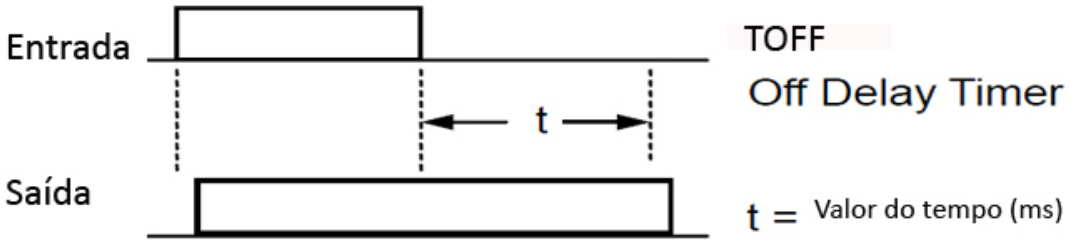
Onde: - TON: Significa retardo na ligação;

- T0: Primeiro temporizador (T0, T1, T2...)

- D0: Memória interna de dados (Word) onde será inserido um valor manualmente pela IHM.

Neste caso acima foi inserido uma memória interna de dados D0. Seu comportamento será da seguinte forma: Precisamos primeiramente inserir um valor neste temporizador, o valor inserido vai ser o valor máximo da contagem. Ao chegar nesse valor máximo o temporizador acionará um contato auxiliar T0.

O contador "TOFF" funciona ao contrário do contador visto acima. Este contador retarda o desligamento de uma saída.

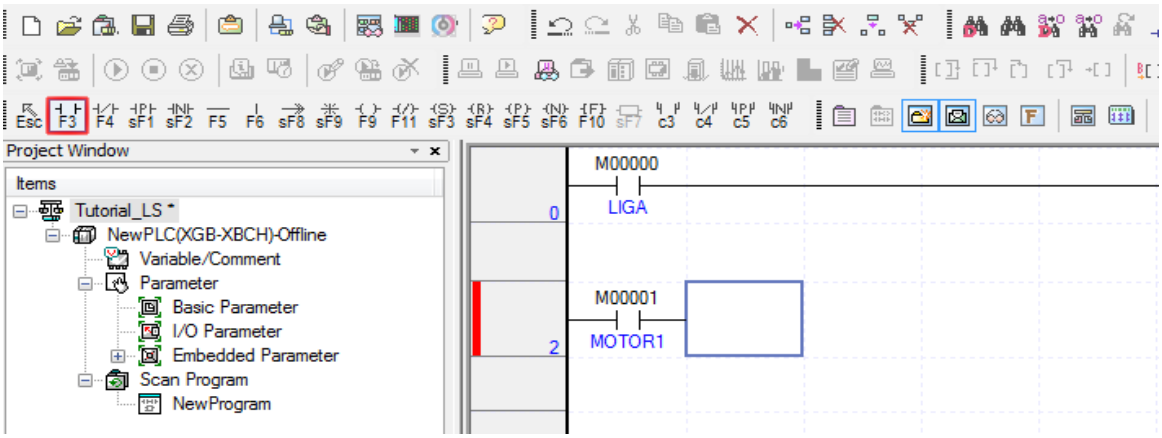


Conseguimos perceber na figura acima que a saída desligou depois de um tempo "t" que a entrada foi desligada. Houve assim, um retardo no desligamento.

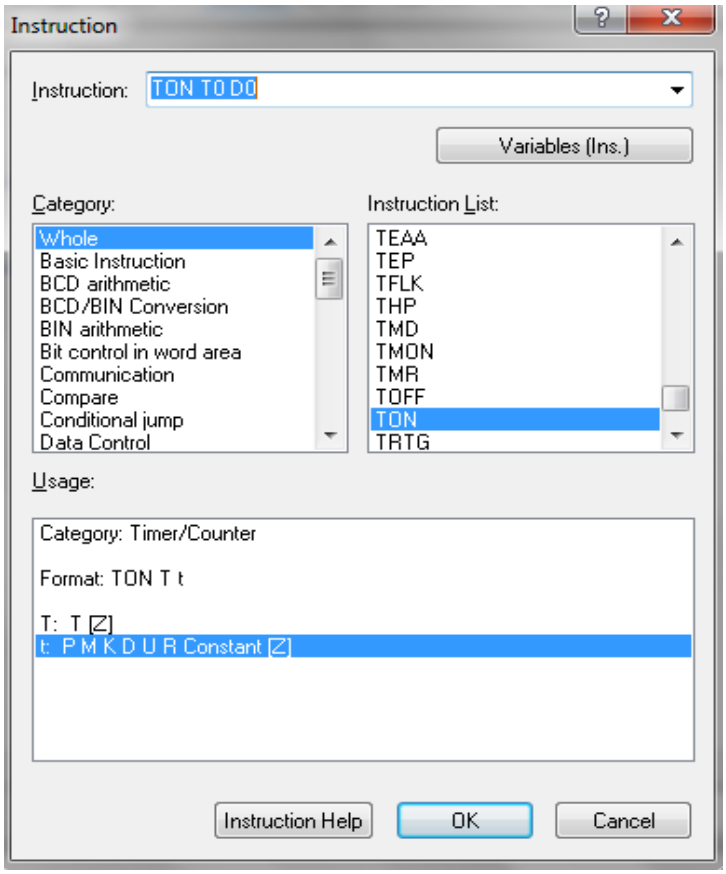
Agora que conhecemos um pouco mais sobre temporizadores, podemos colocar em prática o seu funcionamento.

Vamos então inserir um temporizador em nosso programa. Este temporizador terá a função de retardar a ligação do nosso segundo motor. É importante percebemos que quem deve acionar o nosso temporizador é um contato auxiliar NA da bobina M1.

Em uma nova linha de programação click em um novo contato aberto ou aperte F3 no teclado conforme a imagem abaixo:



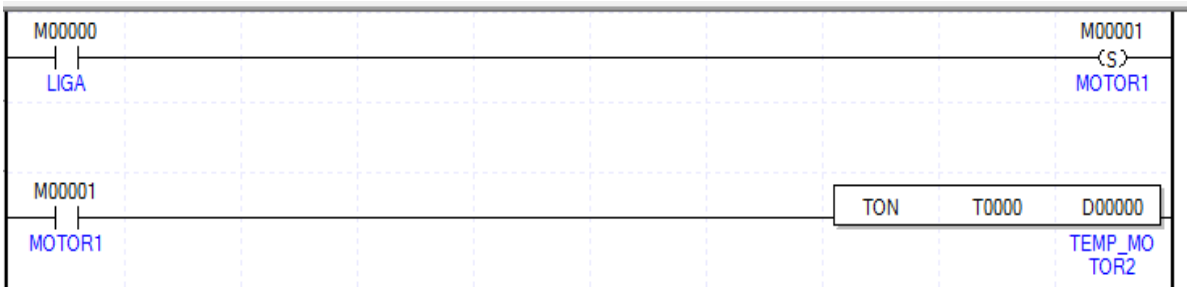
Em seguida clique nas funções {F} F10 **ou aperte F10 no teclado e monte o temporizador conforme a tela abaixo:**



TON - Nossa função de retardo de ligamento;

T0 – Primeiro temporizador;

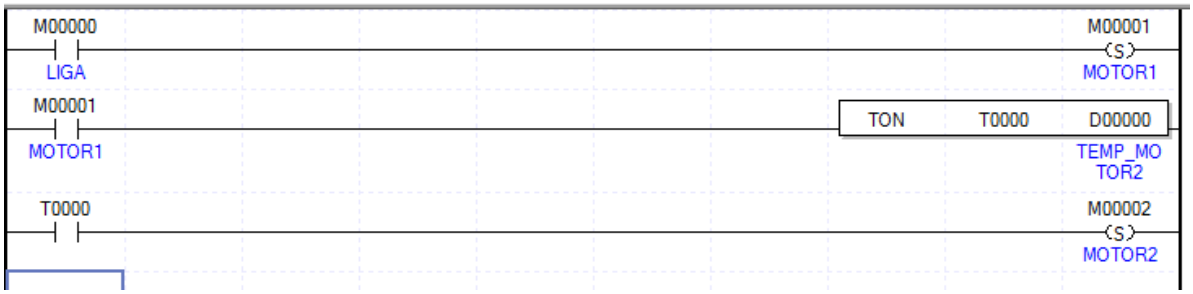
D0 – Memória onde será inserido o valor do tempo;



Observando a programação acima, conseguimos perceber que quando M1 ligar, irá acionar seu contato auxiliar que por sua vez irá iniciar a contagem de tempo do nosso temporizador. Conseguimos perceber também que utilizamos uma memória D0 no temporizador e não um tempo fixo. Isso porque futuramente iremos entrar com um valor neste temporizador, podendo assim, ajustar o tempo de partida do segundo motor, que conforme nossa aplicação, não poderá partir antes dos 5 segundos.

Agora vamos inserir, em nossa programação, um contato aberto do nosso temporizador. Este contato será responsável em partir nosso segundo motor após a contagem do tempo.

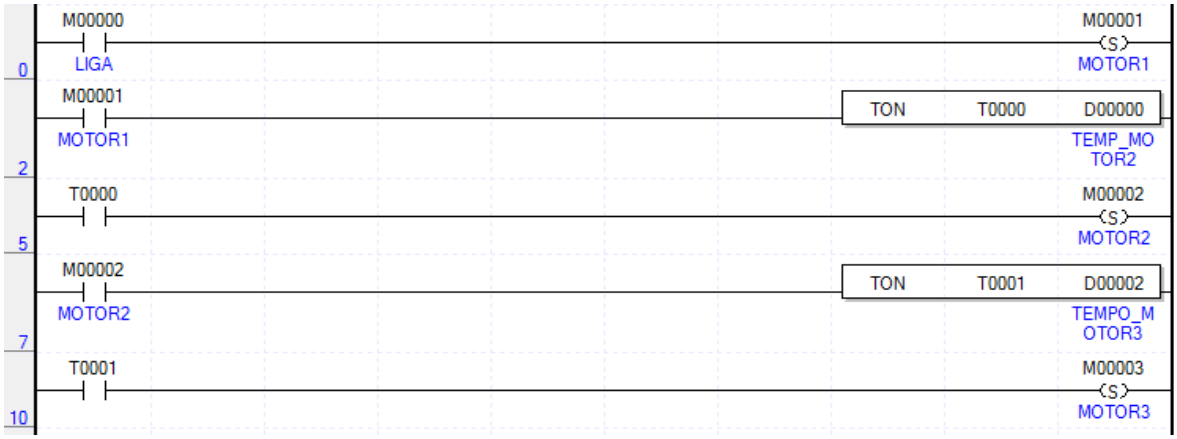
Insira um contato aberto (F3) do temporizador T0 e uma bobina de saída SET com a memória M2 e declare como MOTOR2:



Pronto, a partida do nosso segundo motor está pronta.

Agora vamos repetir a mesma linha de raciocínio para fazer a partida do terceiro motor, que também como o segundo motor, precisará ter a partida temporizada. Quem irá acionar o segundo temporizador será um contato auxiliar normalmente aberto da bobina M2.

Faça a programação conforme a imagem abaixo:



Pronto, a partida temporizada dos três motores já está pronta.

A nossa aplicação também propõem que o segundo motor não deve partir antes de 5 segundos do primeiro motor. Para garantirmos que nosso segundo motor irá obedecer esta condição, vamos fazer uma lógica que impeça e avise ao operador caso o tempo inserido no temporizador T0 seja menor que 5 segundos.

COMPARADORES

A função de comparação é muito utilizada na programação de CLP's, então vale a pena conhecermos um pouco mais sobre este assunto.

Existem vários tipos de comparações que são feitas, normalmente entre dois valores numéricos e conforme a resposta desta comparação, tomamos uma ou outra atitude.

Para quem programa em C++ conhece a função "SE" que se assemelha muito com os comparadores feitos em Ladder.

Nos CLP's da LS, temos vários símbolos utilizados na função de comparadores, os principais são:

X condition	Ccondition	Operation result
=	S1 = S2	On
<=	S1 ≤ S2	On
>=	S1 ≥ S2	On
<>	S1 ≠ S2	On
<	S1 < S2	On
>	S1 > S2	On

= **Igual**

<> **Diferente**

< **Menor**

> **Maior**

<= **Menor ou igual**

>= **Maior ou igual**

Para montarmos uma função de comparação devemos seguir a seguinte sequencia:

Ex: Se um valor X for maior que um valor Y faça uma ação:

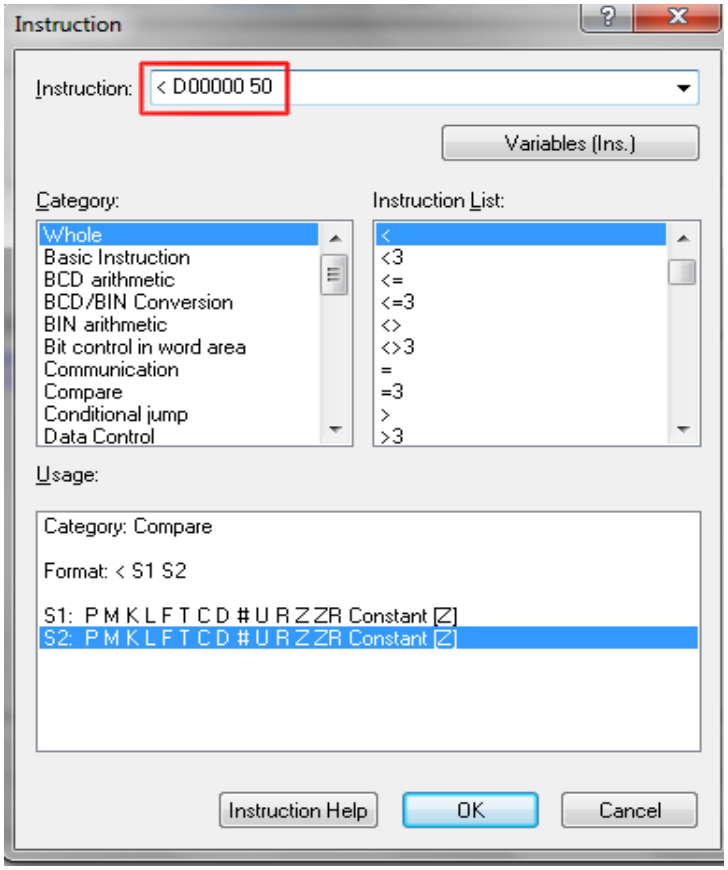
> X Y

Repare que entre o sinal de ">" e o valor "X" temos um espaço e entre o valor "X" e o valor "Y" temos outro espaço.

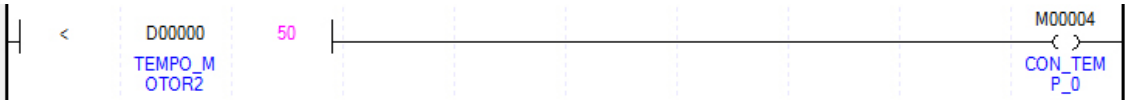
Vamos voltar a nossa programação e criar uma linha de comando com a função de comparação.

Como a comparação é uma função, devemos clicar no ícone das funções {F} F10 ou apertar F10 no teclado.

Monte a função de comparação conforme a tela abaixo:



A função acima significa que quando "D0 for menor que 50, isto é 5 segundos" a condição será aceita.



A linha acima significa que quando D0 (tempo para partir o motor2) for menor do que o valor "50" (5 segundos) a saída M4 (Condição do Tempo 0) irá acionar, caso contrário (maior do que 5 segundos) esta saída não acionará.

Está linha de programação servirá para respeitarmos a condição inicial proposta pelo programa de que não podemos acionar nenhum dos motores caso o tempo inserido de partida do motor 2 seja inferior aos 5 segundos.

Então caso o operador por descuido digitou qualquer valor abaixo do valor 5 precisamos garantir que não ocorra o acionamento dos motores. Como o acionamento dos motores segue a seqüência:

- 1° Aciona Motor 1;

- 2° Motor 1 aciona Temporizador T0;

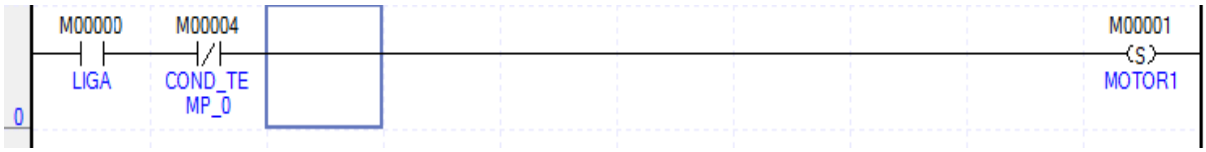
- 3° T0 aciona Motor 2;

- 4° Motor 2 aciona Temporizador T1;

- 5° T1 aciona Motor3;

Precisamos então evitar o acionamento do Motor 1, consequentemente os outros motores não irão acionar.

Para evitarmos o acionamento do motor 1, precisamos simplesmente colocar um contato fechado (NF) da bobina M4 em série com o acionamento do motor 1. Então, voltando à primeira linha de programação, teremos:



Como o contato de M4 é um contato fechado, significa que quando a bobina M4 estiver desenergizada, este contato estará fechado (NF) permitindo a partida do Motor1. Quando a condição estabelecida na linha 12 for verdadeira (< D0 50), a bobina M4 irá energizar e o contato fechado de M4 irá abrir, impedindo assim a partida do Motor1.

Além de impedirmos a partida do Motor1, podemos também enviar um aviso para o operador. Este aviso aparecerá na IHM, avisando ao operador que os motores não partiram porque o tempo de partida do Motor2 está menor que 5 segundos. Deste modo o operador ficará ciente do que está acontecendo e poderá digitar novamente o valor correto.

Para realizar esta ação, precisamos conhecer o comando "MOV". Este comando move valores fixos para uma memória ou move valores de uma memória para outra memória.

Mais adiante nesta apostila, iremos aprender como programar a IHM e mostrar mensagens ao operador. Isto será feito através de uma tabela de mensagens contida no programa da IHM. Esta tabela possui varias linhas: Linha 0, Linha 1, Linha 2 e assim por diante. Em cada linha desta podemos escrever uma mensagem, tal como:

"Partida NÃO autorizada. Tempo de partida do MOTOR2 menor do que 5 segundos".

Para que está mensagem seja mostrada na tela da IHM, precisamos mover um valor do CLP para IHM, correspondente a linha que a nossa mensagem está contida, por exemplo: Caso a mensagem foi escrita na linha 1 da tabela da IHM, precisamos enviar o número "1" para a IHM. A IHM por sua vez, mostrará o conteúdo que está na linha "1".

COMANDO MOV

Como citado acima, o comando "MOV" serve para mover um valor para uma memória. Este comando é montado da seguinte maneira: MOV 1 D4

O comando escrito desta maneira significa que estamos movendo o número "1" para a memória D4. Perceba os espaços entre o MOV e o "1" e entre o "1" e o "D4". Perceba também que utilizamos uma memória "D" para receber o valor, que como já explicado anteriormente, as memórias de word(valores) são as memórias "D" e as de bit(ligado/desligado) são as memórias "M".

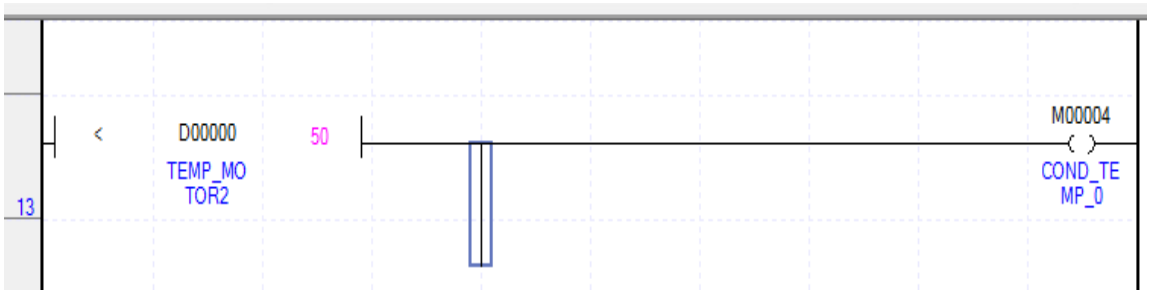
Voltando a programação, vamos inserir este comando no lugar correto, para que a mensagem lá na IHM apareça somente quando o valor de D0(tempo de partida do Motor2) for menor que 50.

Precisamos inserir este comando na frente da nossa condição < D0 50:

Para isso vamos colocar o comando em paralelo com o acionamento de M4, clicando no ícone 

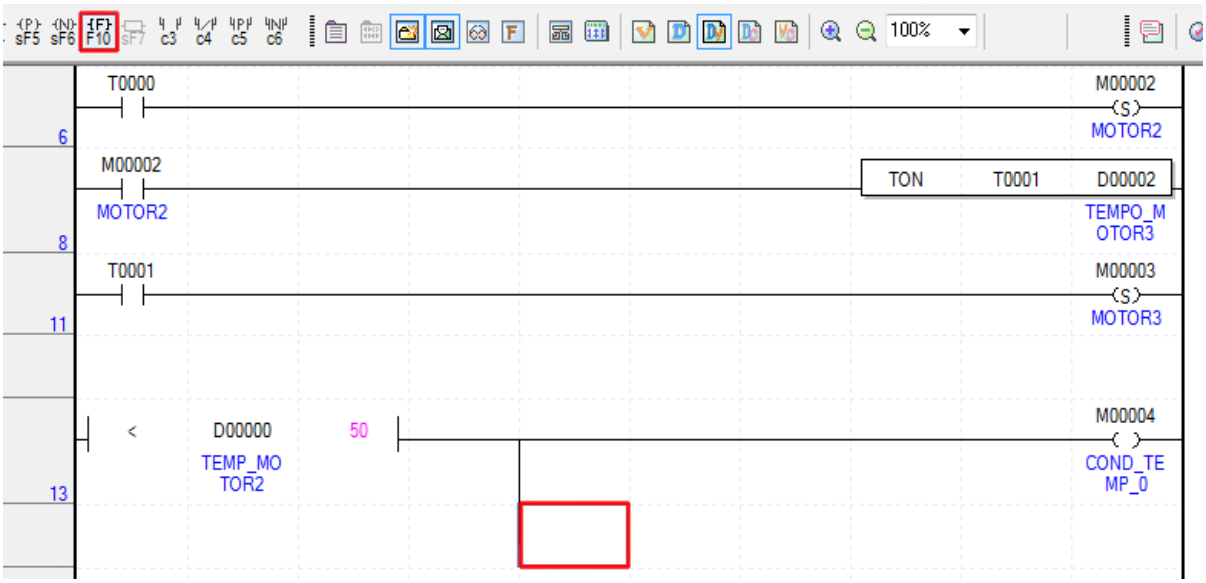
.

Em seguida clique neste ponto da linha de programação:

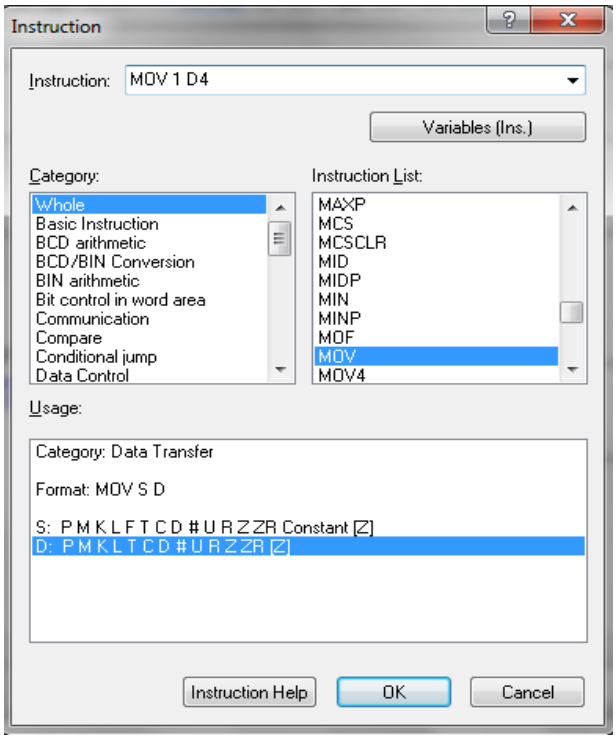


Assim criamos uma linha na vertical.

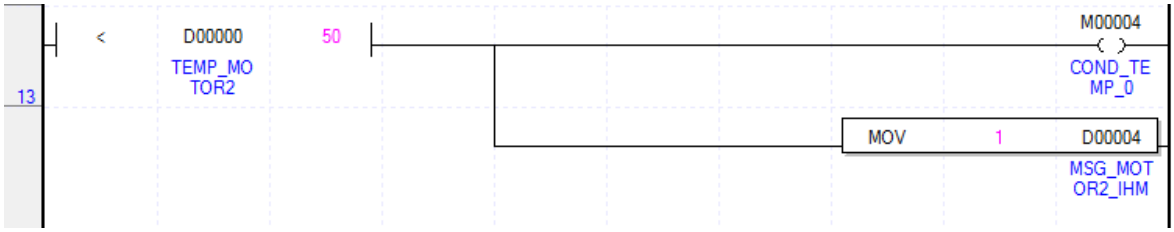
Vamos inserir o comando “MOV” clicando no ícone das funções  **e em seguida clicando do lado direito da linha vertical que acabamos de criar:**



Escreva o comando da seguinte maneira:



Teremos a seguinte linha de programação:

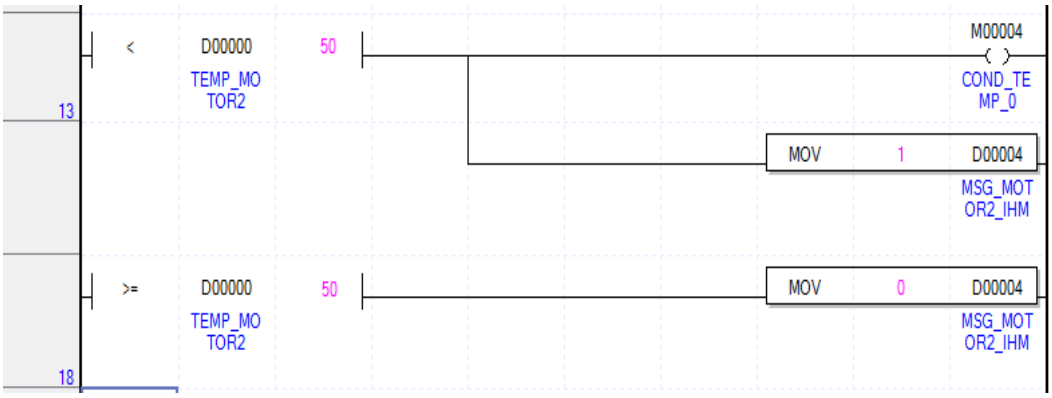


Podemos perceber que o programa só irá mover o número "1" para a memória D4 quando a condição < D0 50 for respeitada, ou seja, quando D0 for menor do que 5 segundos.

Lembrando que a tabela da IHM inicia-se com a linha 0 e neste exemplo podemos perceber que escrevemos nossa mensagem na linha 1. Isso porque, a linha 0 da tabela da IHM irá conter a mensagem "Partida Autorizada". Nesta condição devemos mover o número "0" para a memória D4.

Então precisamos criar outra condição em nossa programação. A condição de que quando D0 (tempo de partida do Motor2) for IGUAL ou MAIOR do que 5 segundos, MOVA o valor 0 (mensagem "Partida Autorizada") para a memória D4.

Crie a seguinte linha de programação:



Se o valor do tempo de partida do Motor2 for menor do que 50, ligaremos a bobina M4 e moveremos 1 para a memória D4.

Se não moveremos 0 para a memória D4.

PROGRAMAÇÃO MOTOR 3

A programação do Motor2 está completa, vamos agora iniciar a programação para o Motor 3. Na aplicação proposta no início desta apostila, temos a condição, de que o Motor 3, só deverá partir 3 segundo após a partida do motor 2. Lembrando que o tempo de partida do Motor 2 é variável, podendo ser 5, 6, 7, 8... segundos. Então não sabemos exatamente qual valor o operador irá digitar na IHM. Sabemos apenas que o tempo de partida do Motor 3 tem que ser 3 segundos a mais do que o Motor 2.

Para que consigamos respeitar esta condição, teremos que fazer uma operação matemática simples no CLP.

OPERAÇÕES MATEMÁTICAS

Os CLP's da LS nos permitem fazer operações matemáticas com números inteiros e números reais (float). Neste exemplo iremos mostrar como fazer as operações matemáticas básicas com números inteiros.

Você deve montar a função da seguinte maneira:

- Função de Adição: ADD 6 3 D5

Significa que o CLP irá somar 6 mais 3 e jogar o resultado na memória de word D5 (6 + 3 = D5)

- Função de Subtração: SUB 6 3 D5

Significa que o CLP irá subtrair 6 menos 3 e jogar o resultado na memória de word D5 (6 - 3 = D5)

- Função de Multiplicação: MUL 6 3 D5

Significa que o CLP irá multiplicar 6 vezes 3 e jogar o resultado na memória de word D5 (6 x 3 = D5)

- Função de Divisão: DIV 6 3 D5

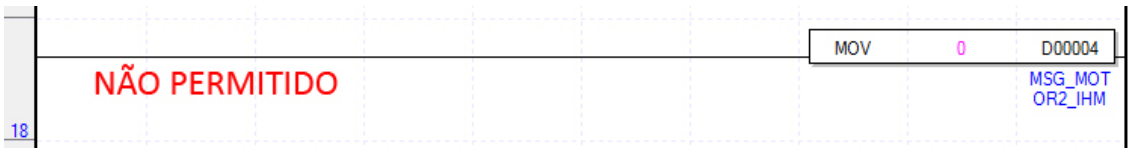
Significa que o CLP irá dividir 6 por 3 e jogar o resultado na memória de word D5 (6 / 3 = D5)

Como na aplicação proposta, o tempo do Motor 3 tem que ser sempre 3 segundos a mais do que o tempo do Motor 2, iremos utilizar a operação de adição para somarmos o valor fixo "3" com o valor inserido em D0 (tempo de partida do Motor2).

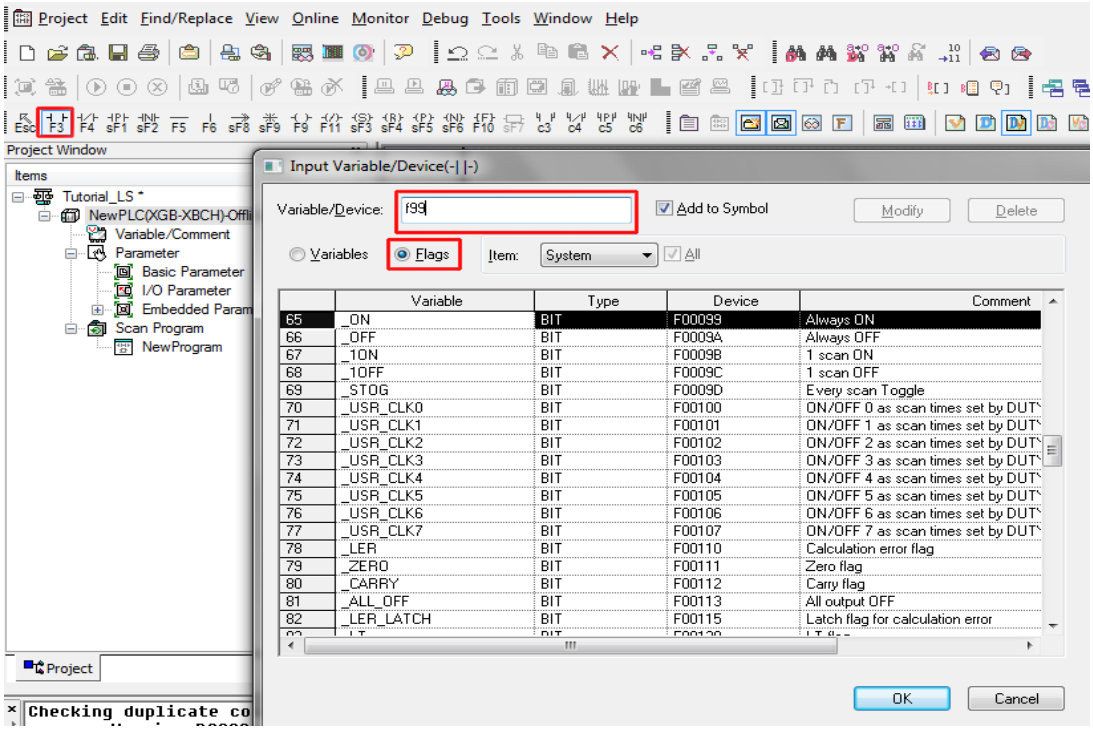
Voltando a programação, vamos criar uma linha de comando para realizar esta operação. Precisamos pensar em que momento nós queremos que esta operação de adição seja realizada. Sempre quando for inserir uma função como MOV, Operação Matemática, ou até mesmo o SET ou RESET de uma bobina, nós precisamos pensar em que condição nós queremos que esta função seja executada. Fazemos a seguinte pergunta:

A FUNÇÃO PODE SER REALIZADA O TEMPO TODO?

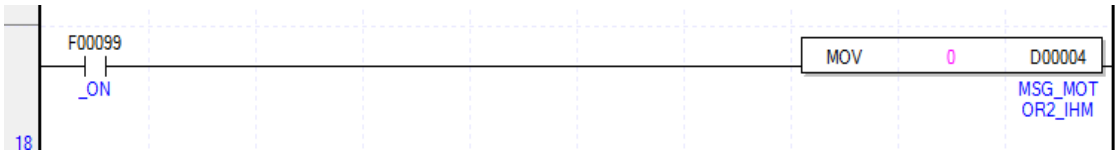
- SE SIM: utilizaremos uma Flag F99. Esta flag é um contato sempre ativo (Always ON), esta flag existe porque não podemos ligar uma função diretamente na linha esquerda de programação:



Se desejarmos que a função seja executada o tempo todo, precisamos inserir um contato F99 – Always ON que sempre ficará ativo na programação. Para isso devemos clicar em F3 > Flags e digitar F99:



Teremos a seguinte linha de programação:



**Não precisa ser feito na programação é apenas um exemplo.*

Esta linha de programação significa o valor "0" será movido o tempo todo no programa para D4, diferente da condição anterior que o valor "0" só era movido quando D0 fosse maior ou igual a 50 (>= D0 50).

Flag – São funções prontas dos CLP's da LS para facilitar a programação. Existem várias Flags para serem utilizadas na programação, as mais utilizadas são:

F99 – Always ON - Contato sempre ativo.

F9B – 1 scan ON – Contato fica ativo apenas no primeiro Scan do CLP.

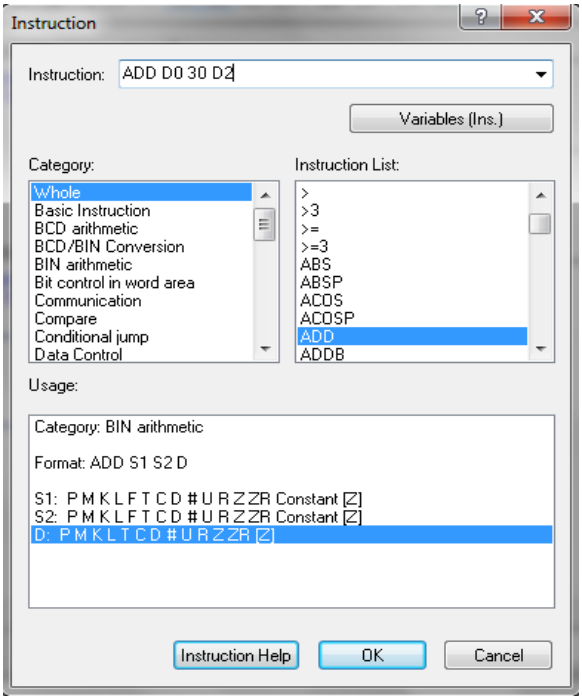
F93 – Clock de 1 segundo.

- SE NÃO: precisamos saber em que condição nós queremos que a função seja ativada. Por exemplo, quando criamos as linhas de programação onde movemos um valor para a memória D4, nós colocamos uma condição para que a função MOV fosse executada, esta condição significa que não queremos que a função MOV fique executando o tempo todo. Queremos que ela só execute quando respeitar a condição (< D0 50) ou a condição (>= D0 50).

Mas agora queremos realizar uma operação de adição, então precisamos perguntar: A função de adição pode ser realizada o tempo todo?

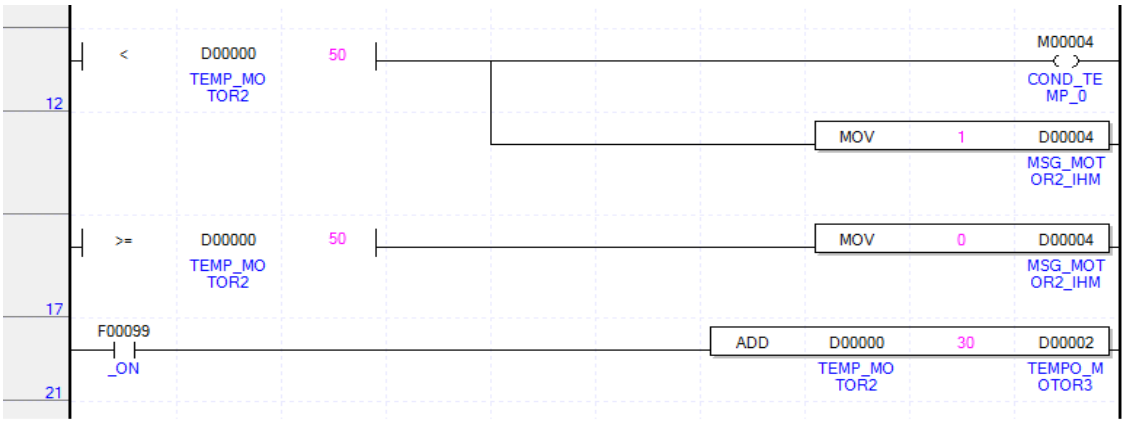
A resposta é sim, porque ao inserirmos um valor pela IHM na memória D0, automaticamente este valor será somado com o valor 30 (3 segundos), sem precisarmos dar um comando para que a função de adição seja executada. Esta explicação ficará mais clara na programação que faremos abaixo.

Então Vamos criar a linha para a função de adição na nossa programação. Click em um contato aberto (F3) e coloque uma Flag F99 como visto acima e em seguida, click nas funções F10 ou aperte F10 do teclado e monte a função de adição conforme a tela abaixo:



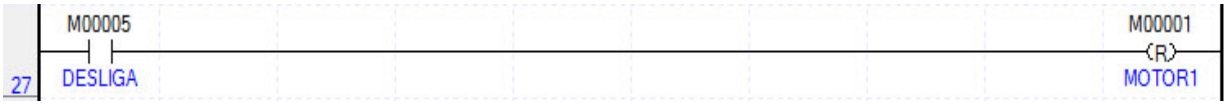
Isso significa que iremos adicionar 30 (3 segundos) em D0(independente do valor de D0) e colocar o resultado memória D2.

A linha de programação ficará da seguinte forma:

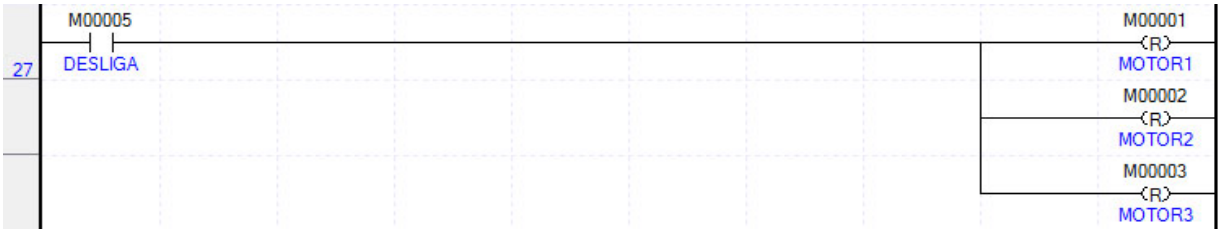


Assim que o operador digitar na IHM o tempo de partida do motor 2 (memória D0) automaticamente o CLP irá somar 3 segundos a esse tempo de partida do motor 2 e colocar o resultado em D2 que é o tempo de partida do motor 3. Isso garante que o motor 3 sempre irá partir 3 segundos após o motor 2.

Coloque mais uma linha de programação com a função de Desligar todos os motores juntos. Use como exemplo a primeira linha de programação, apenas altere o botão de liga para desliga, a memória M1 para M5 e nomeie como DESLIGA e agora troque a bobina de SET por uma bobina de RESET:

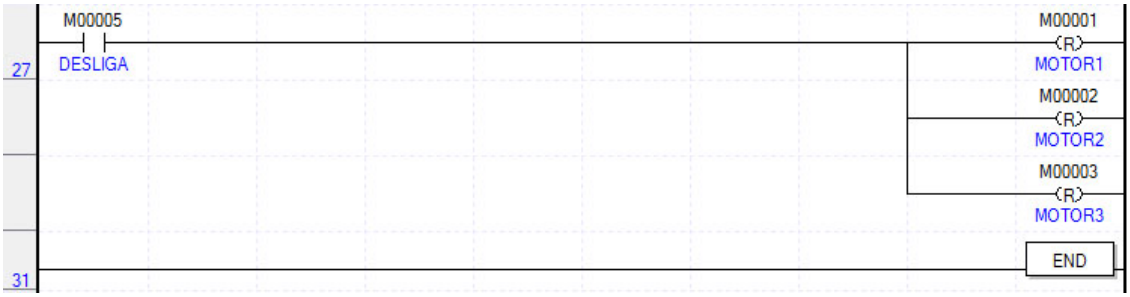


Resete também nem paralelo, as memórias M2 (Motor 2) e M3 (Motor 3):



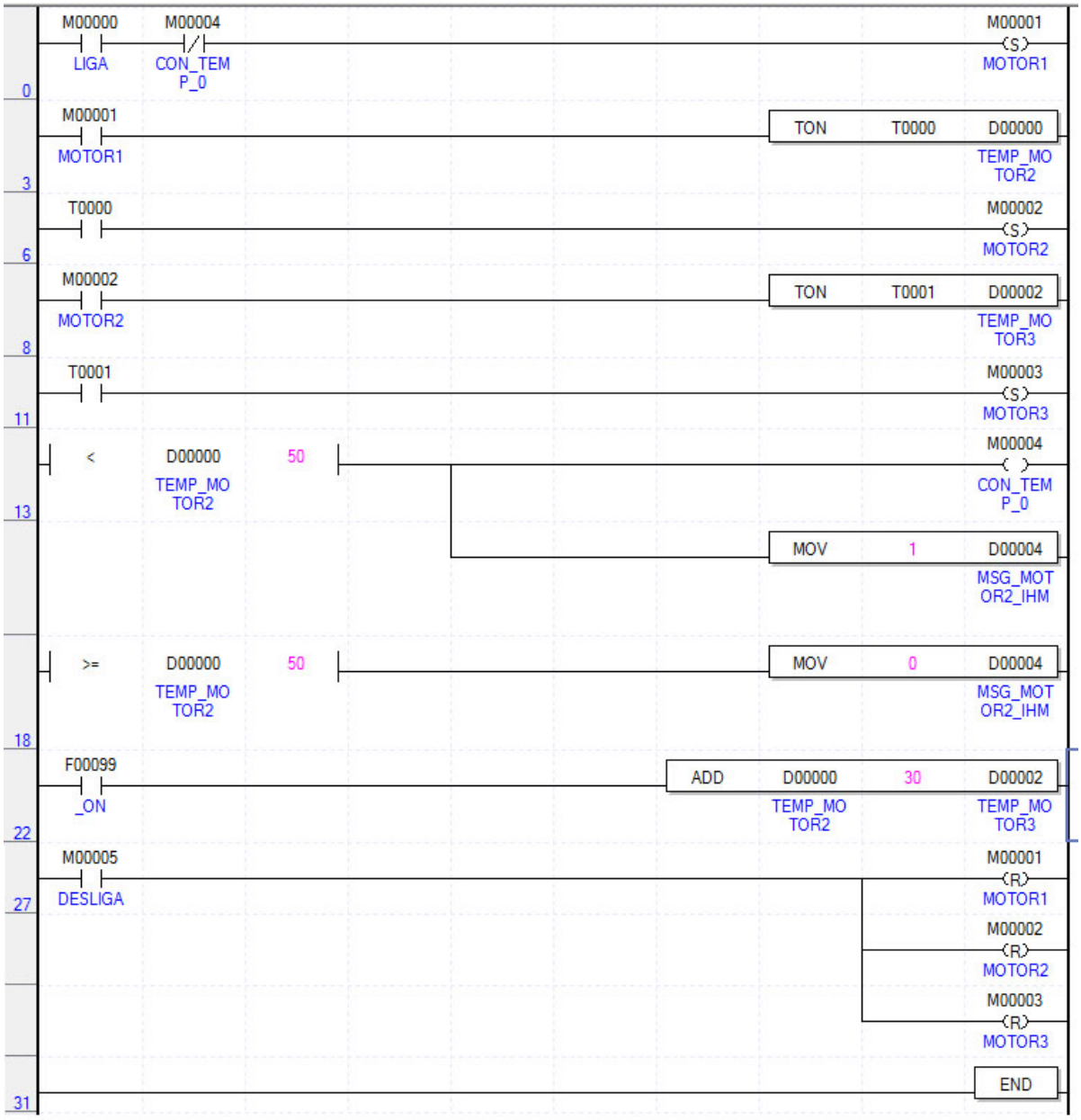
Quando o operador apertar na IHM o botão M5, as memórias M1, M2 e M3 serão Resetadas. Isto fará com todos os motores desliguem.

Click em {F} F10 **e escreva END para finalizar a programação.**



**** Todos os programas precisam ter a função END para representar a finalização da programação.***

Programação completa no CLP:

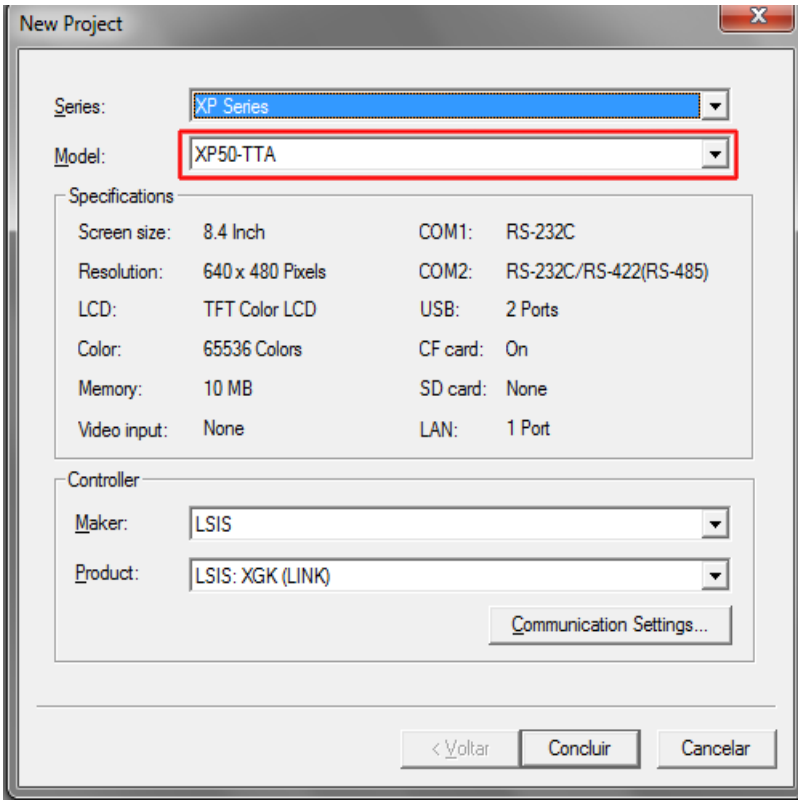


- CRIANDO UM PROGRAMA BÁSICO NA IHM

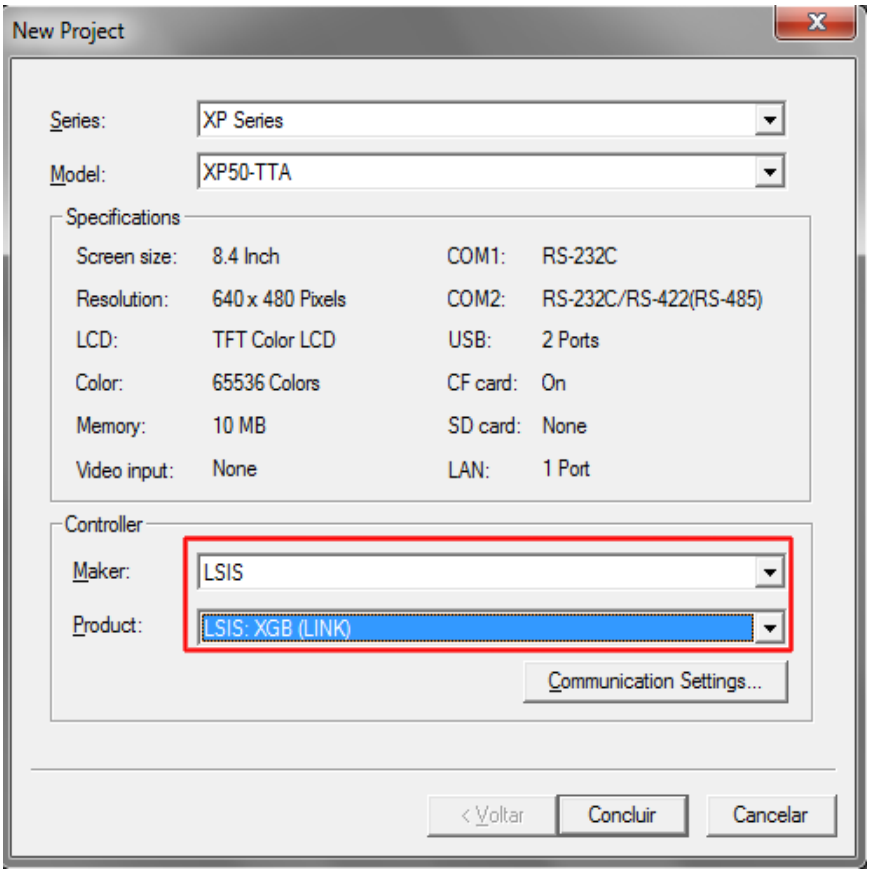
Abra o Software XP-Builder:

Cick em Create Project para criar um novo projeto:

Nesta tela precisamos configurar qual modelo de IHM iremos utilizar. Neste exemplo vamos utilizar a IHM XP 50 – TTA que é uma IHM de 8.4 polegadas da LS.



Nesta mesma tela ainda precisamos configurar qual o fabricante do equipamento que irá comunicar-se com a IHM, neste caso LSIS:



E em seguida vamos configurar a linha do CLP que estamos utilizando, neste caso o CLP é da família XGB.

Existem para essa família sempre três opções: XGB(ETHERNET) utilizado quando possuímos um módulo Ethernet acoplado ao CLP, XGB(LINK) e XGB(CPU). A diferença entre as duas últimas é que a comunicação CPU é utilizada quando comunicamos o CLP utilizando a mesma porta de programação do CLP (Mini Din). Então a porta Mini Din dos CLP's da LS podem ser utilizadas para realizar a comunicação entre CLP e PC via serial RS232 ou para comunica-se com a IHM também via RS232. Já a opção LINK serve apenas para comunicação e não para programação.

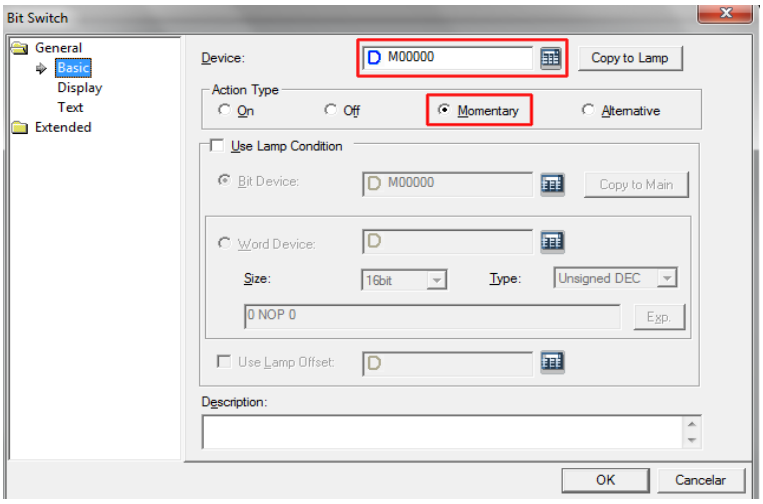
CLP: XBM-DR16S

Como estamos apenas simulando nossa aplicação, vamos escolher a opção XGB (Link) que é a mais usada na prática real.

Após o software de programação da IHM configurado vamos iniciar inserindo primeiramente o nosso botão de LIGA correspondente a primeira linha de programação feita no CLP.

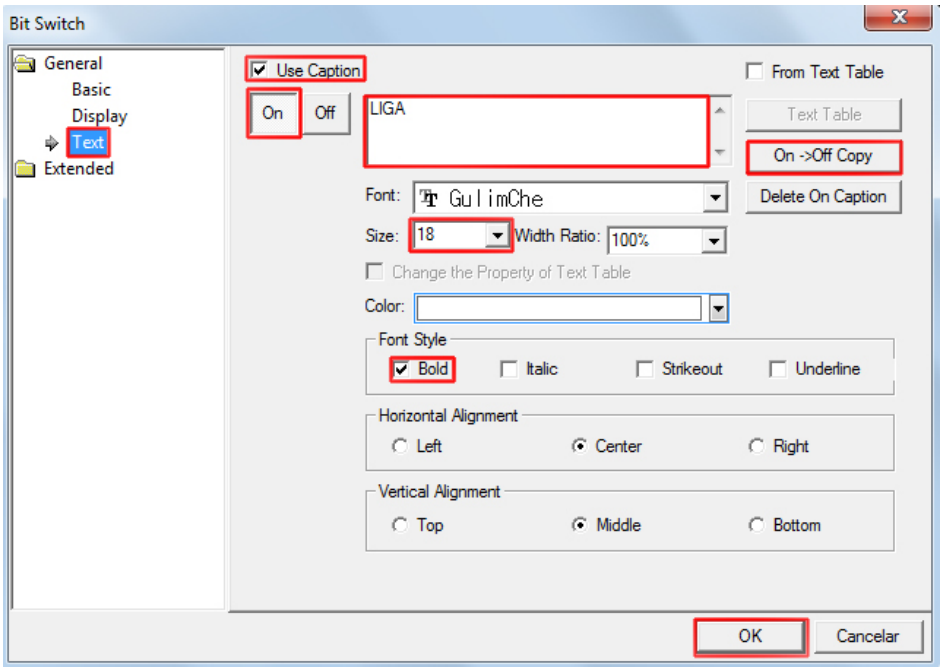
Click em "Bit Switch" que se encontra na barra de Objetos ao lado direito da tela.

Agora posicione o cursor do mouse na tela preta, onde você deseja inserir o botão e de um click na tela. Abrirá a seguinte janela:



Em "Device" você precisa inserir a memória responsável pelo botão "LIGA". Se voltarmos à primeira linha de programação do CLP, vamos perceber que a memória responsável pelo botão liga é a memória M0:

Outra opção que marcamos na tela acima foi a opção "Momentary", isto significa que o botão vai se comportar como um "Push Button" (Botão de Pulso), porque precisamos de apenas 1 pulso para Setar a bobina M1.

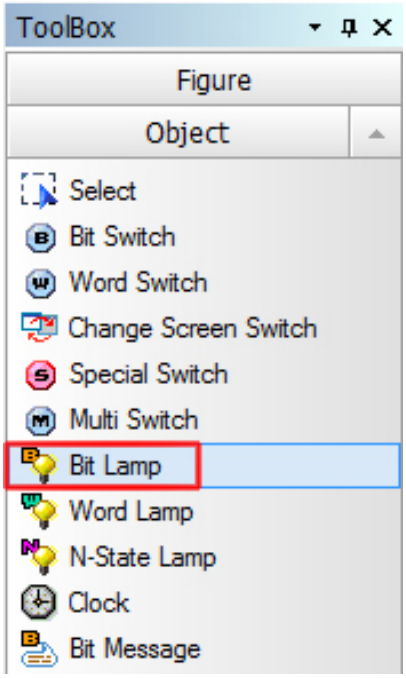


Podemos também clicar em "Text" para entrar nas propriedades de texto do botão. Marque a opção "Use Caption" e em "On" digite o texto "LIGA" para o nosso botão. Desta maneira o texto LIGA ficará visível no botão.

Click também no botão "Off" e digite um texto para ser mostrado quando o botão não estiver apertado, ou aperte o botão "On ->Off Copy" e o programa copia automaticamente o texto LIGA para a função "Off". Podemos também nesta tela alterar as propriedades da fonte.

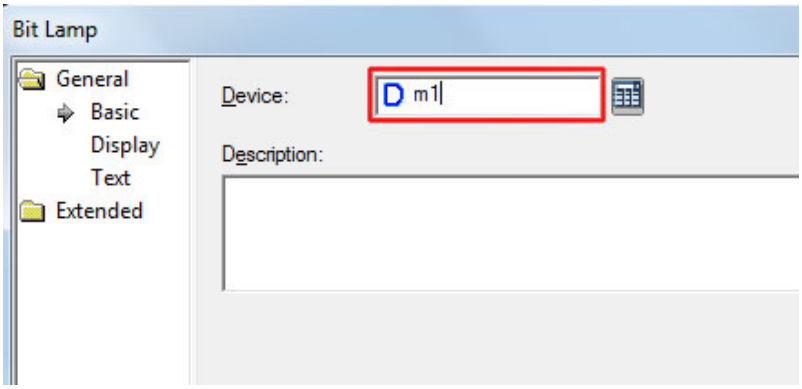
A tela deverá ficar conforme imagem abaixo:

Agora vamos inserir 3 lâmpadas para representar quando os motores 1, 2 e 3 estiverem ligados ou desligados. Para isso click em "Bit Lamp" e depois click na tela preta:

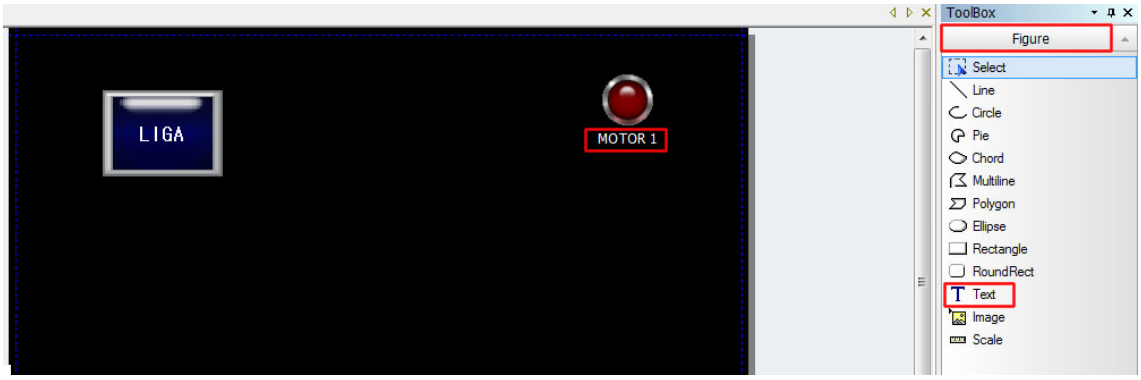


Lembre-se que a memória responsável pelo MOTOR 1 é a M1.

Então na janela que abriu vamos digitar M1 e depois clicar em OK:

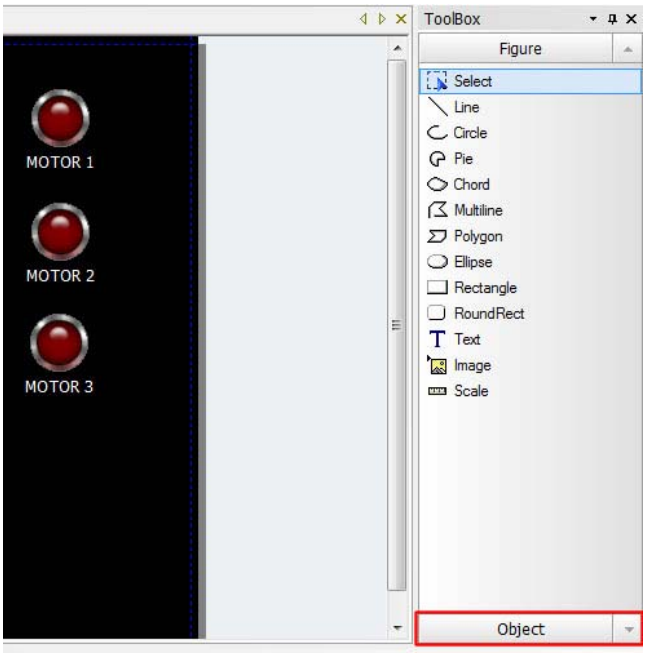


Agora click em "Figure" do lado esquerdo da tela, depois em "Text" e logo em seguida click abaixo da lâmpada e digite o texto MOTOR 1.



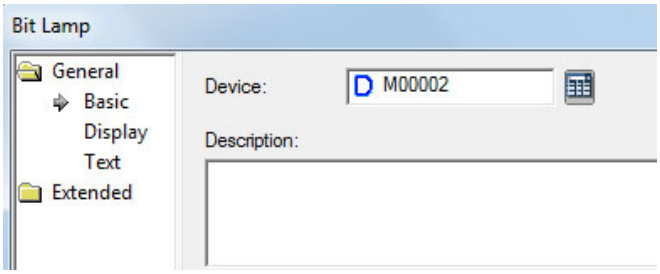
A representação do status do primeiro motor está pronta.

Click em “Object” novamente:

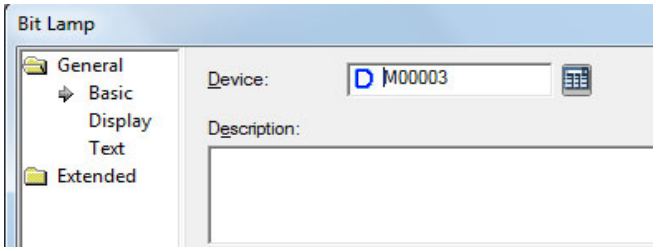


Repita o passo acima mais duas vezes para os motores 2 e 3 e onde você digitou M1, agora digite M2 para o motor 2 e M3 para o motor 3:

Para o motor 2:



Para o motor 3:

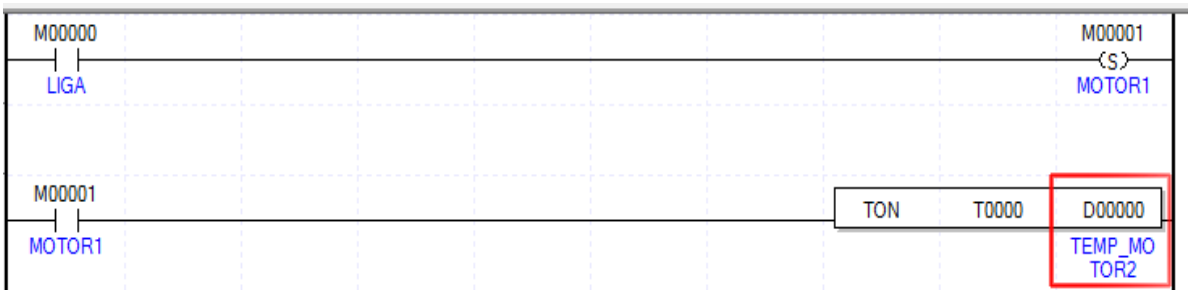


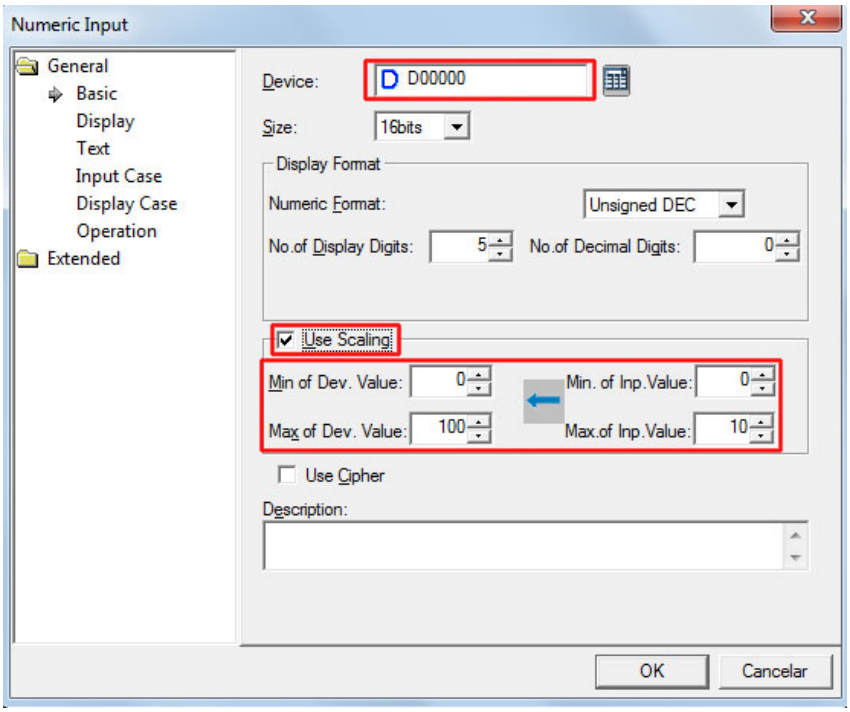
Agora precisamos criar um botão na tela da IHM para que possamos inserir o valor de partida do Motor 2. Lembrando que se o operador inserir um valor menor do que 5 segundos mostraremos um aviso na tela: *"Partida NÃO autorizada. Tempo de partida do MOTOR 2 menor do que 5 segundos".*

Como queremos inserir um valor numérico, precisamos então, clicar em "Numeric Input" (Entrada Numérica) que se encontra ao lado direito da tela do XP-Builder:

Click na tela preta e em seguida abrirá a janela para inserirmos as propriedades desse botão "Numeric Input":

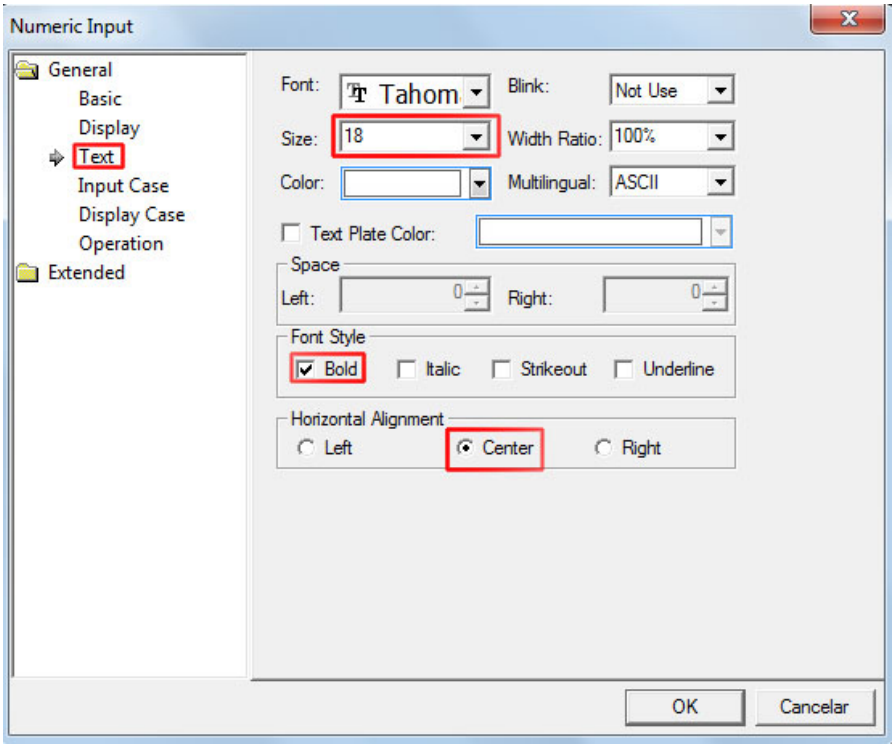
Coloque a memória responsável por receber o valor do tempo de partida do Motor 2 que pela nossa programação no CLP é a memória D0:





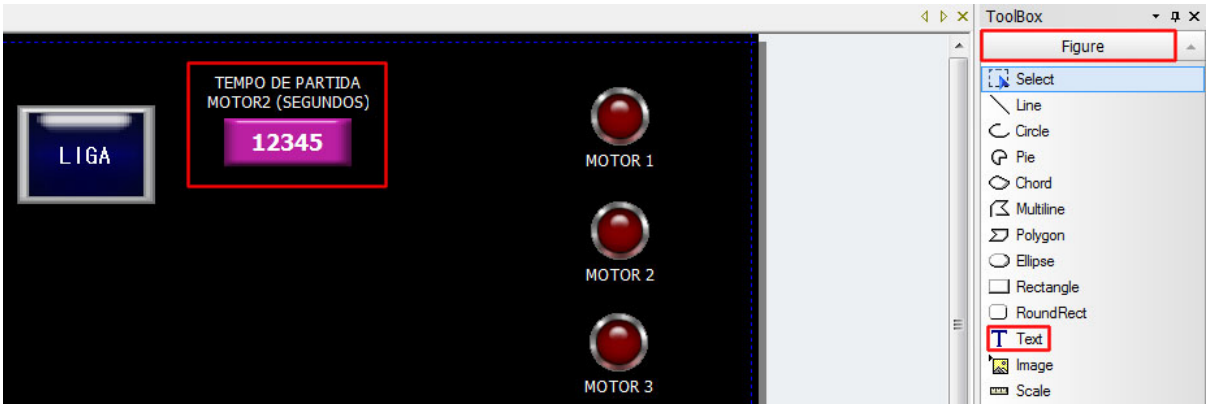
Marque a opção "Use Scaling" e configure conforme a tela acima. Esta opção tem a função de multiplicar, neste caso por 10, o valor que foi inserido no "Numeric Imput". Então quando o operador digitar, por exemplo o número 5, a IHM multiplica automaticamente esse valor por 10 e manda 50 para o CLP. Lembre-se que a unidade de tempo do CLP é de 1 décimo de segundo, neste caso o valor 50 no CLP é igual a 5 segundos.

Podemos alterar também a propriedades de texto do “Numeric Input”:

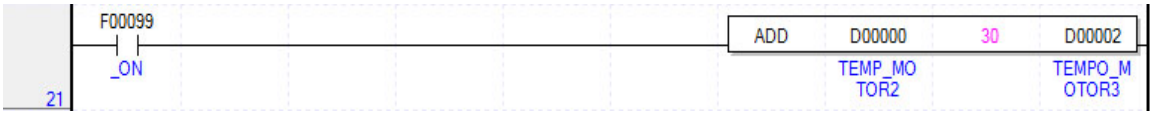


Estas propriedades são apenas para melhor visualização na tela da IHM. Click em OK.

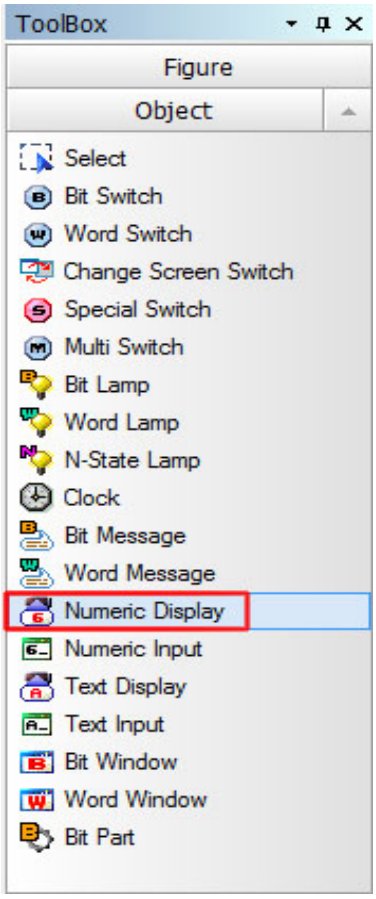
Insira um texto a cima do botão “Numeric Input” para o operador saber onde inserir o tempo do Motor 2:



Vamos agora inserir um display para visualizarmos o tempo de partida do Motor 3. Lembrando que o operador não poderá alterar o tempo de partida desde motor, apenas visualizá-lo. E conforme nossa programação, ele vai ser sempre 3 segundos a mais do que o tempo de partida do Motor 2:

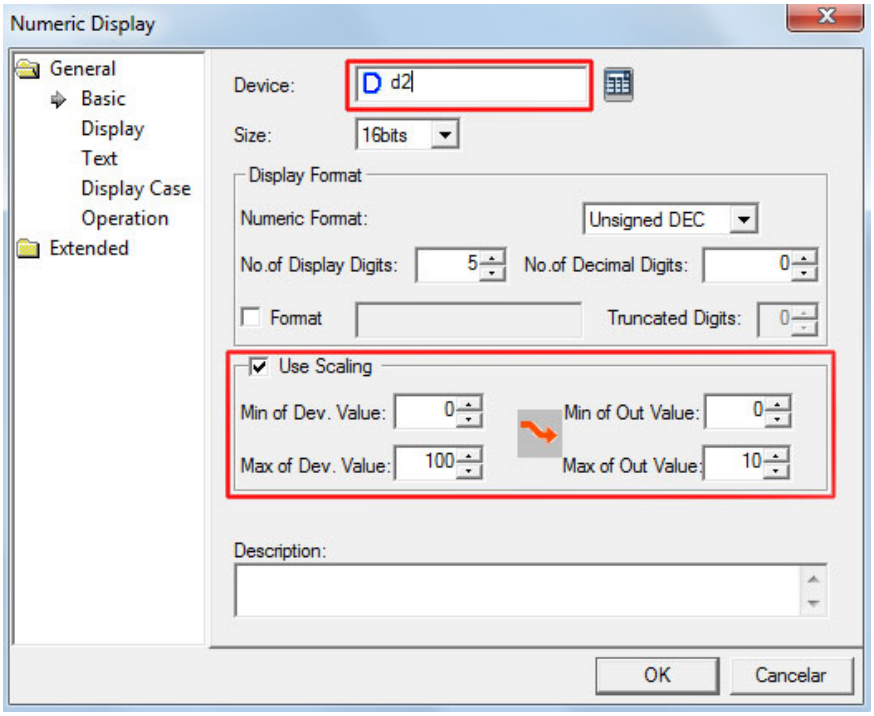


Para inserir um Display, precisamos clicar em "Numeric Display" que se encontra ao lado esquerdo da tela:



Em seguida click na tela para inserir o Display.

Irá abrir a janela de propriedades do Display para que possamos configurá-lo:



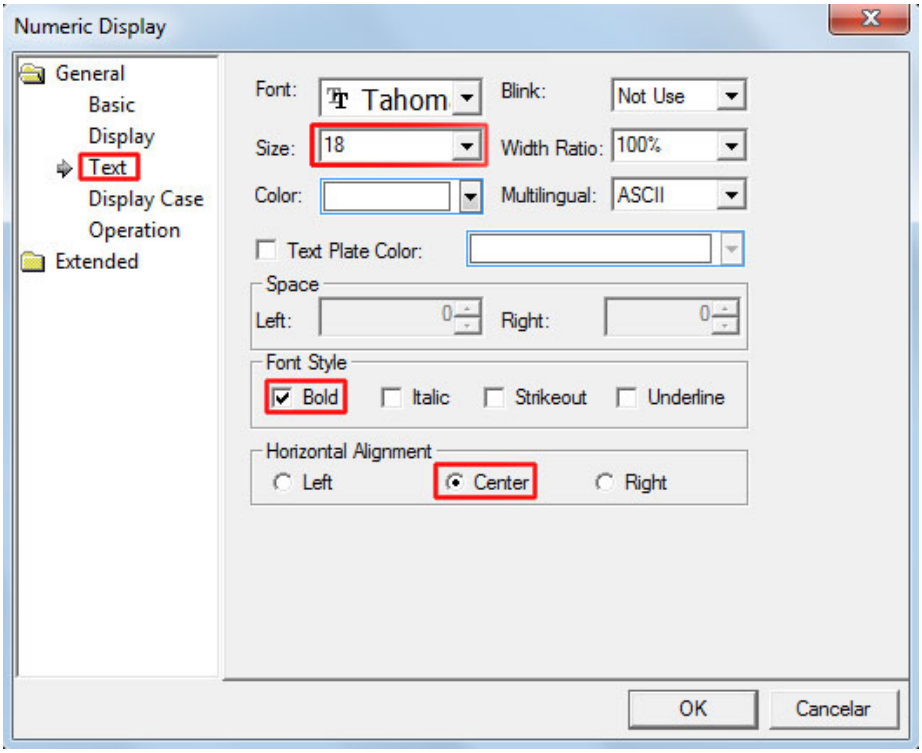
Coloque em "Device" a memória D2 responsável pelo tempo de partida do Motor 3.

Vamos também utilizar a opção de escala, porém agora como estamos visualizando um valor e não inserindo, esta opção funcionará um pouco diferente da já citada anteriormente. Agora o CLP enviará para a IHM o valor do tempo de partida do Motor 2 acrescido do valor 30, então a IHM irá dividir automaticamente este valor por 10 e mostrar no Numeric Display. Exemplo: Se o CLP enviar 80 a IHM mostrará na tela o valor 8 (em segundos). Configure a opção "Use Scaling" conforme a tela acima.

Então quando quisermos INSERIR um valor numérico na IHM, devemos utilizar a função "Numeric Input".

E quando quisermos VISUALIZAR um valor na IHM, devemos utilizar a função "Numeric Display".

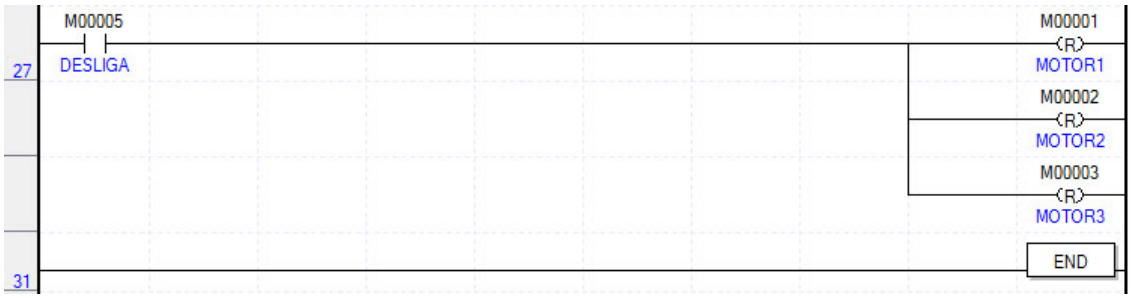
Podemos alterar também a propriedades de texto do "Numeric Display":



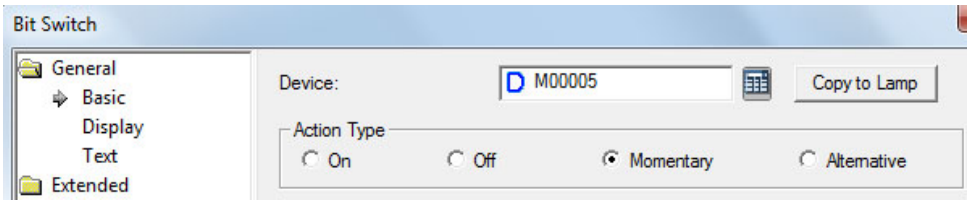
Estas propriedades são apenas para melhor visualização na tela da IHM. Click em OK.

Insira um texto a cima do display "Numeric Display" para o operador saber que é a visualização do tempo de partida do Motor 3:

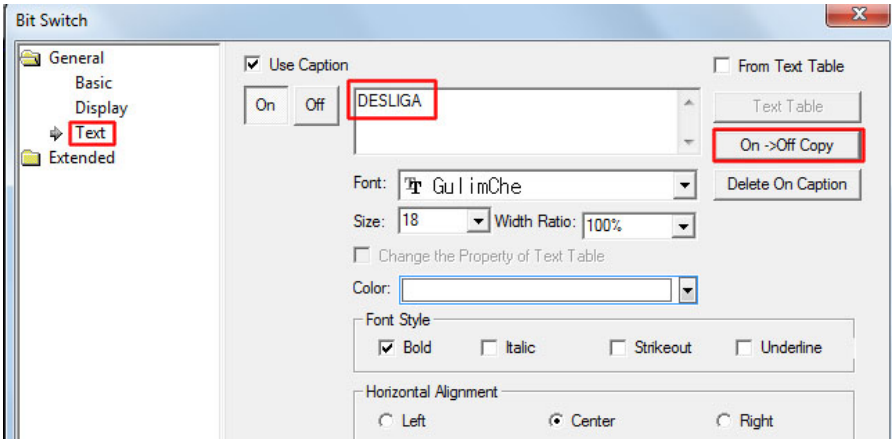
Vamos agora inserir o botão de desliga, para que possamos desligar todos os motores. O botão de desliga é o botão responsável por Resetar a memória M1, M2 e M3, memórias de partida do Motor 1, Motor 2 e Motor 3:



Como o botão de Desliga será bem parecido com o botão de Liga, você pode copiar e colar o botão de liga. Selecione o botão de Liga e aperte "Ctrl C" e depois "Ctrl V" no teclado. Depois de colado na tela, de dois clicks em cima do novo botão e altere o "Device" para M5:



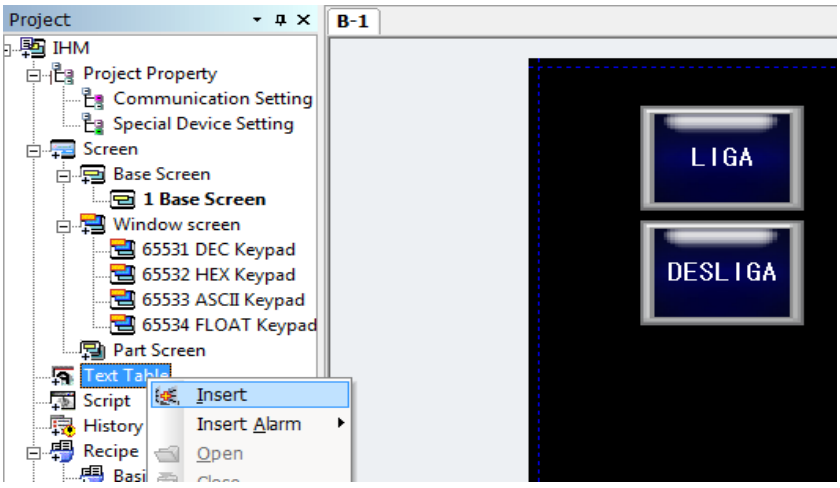
Click agora em "Text", troque o título para DESLIGA, em seguida click no botão "On ->Off Copy" e depois em OK:



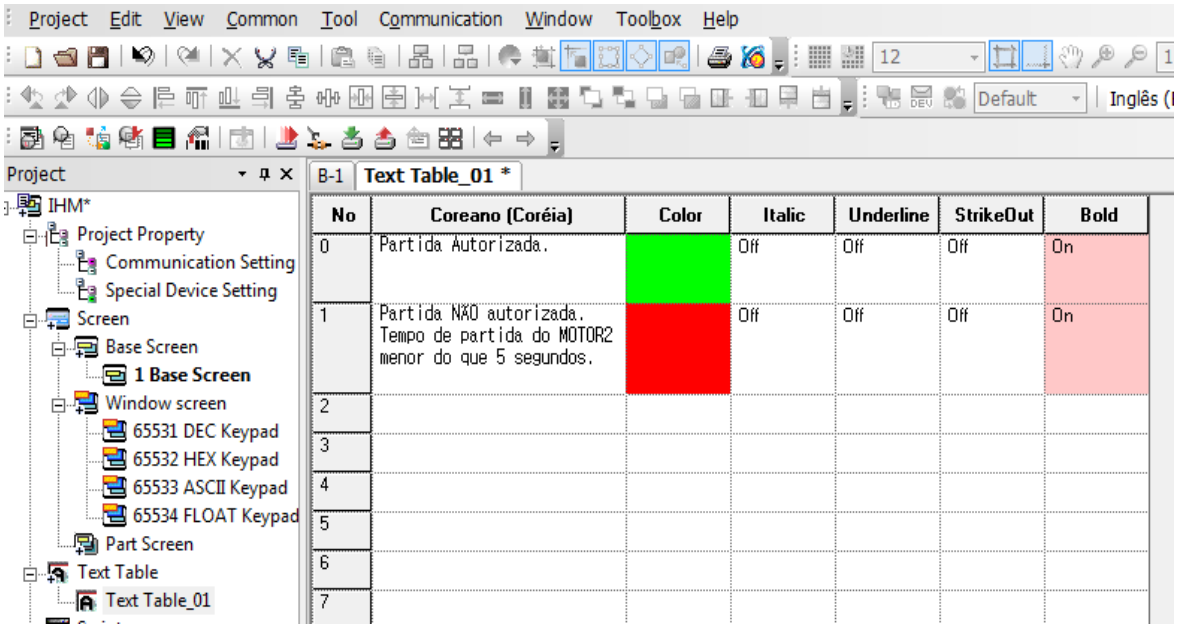
Você terá uma tela semelhante a esta:

Agora precisamos criar um Campo de mensagem, para quando o operador digitar na IHM um tempo de partida para o Motor 2, menor do que 5 segundos, o programa possa avisá-lo que o tempo está incorreto.

Primeiramente precisamos criar uma tabela com as mensagens que queremos mostrar na tela, para isso click com o botão direito em "Text Table" e em seguida "Insert" para inserirmos uma tabela.



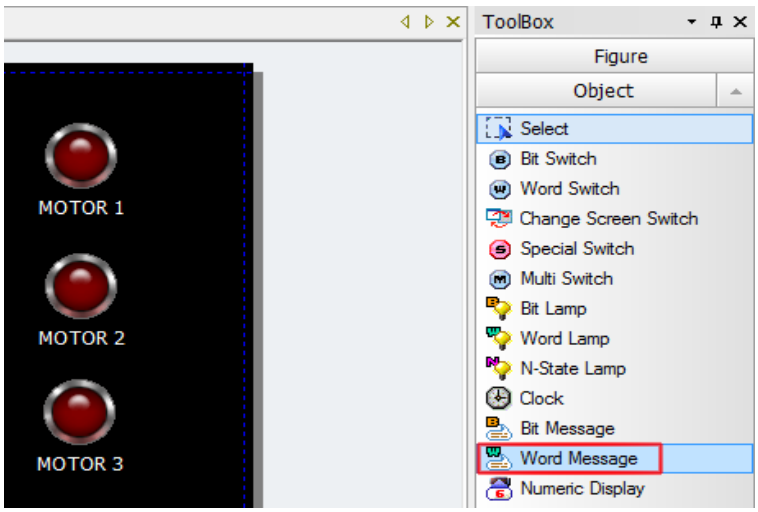
Preencha a linha 0 e a linha 1 conforme a imagem abaixo:



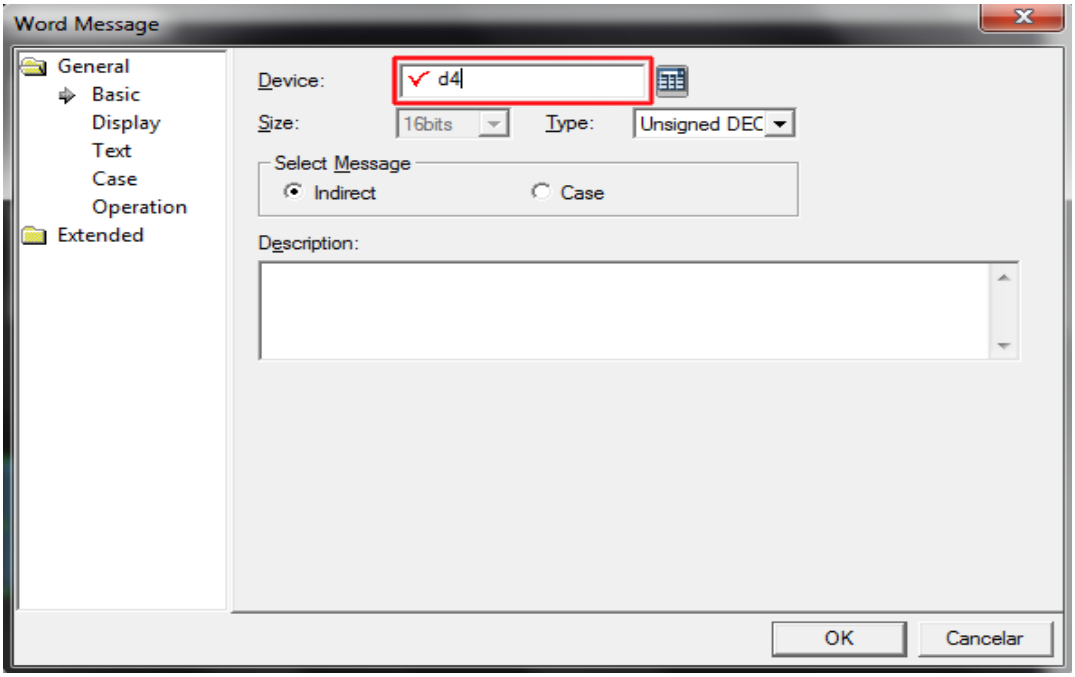
Na linha 0 digite: Partida Autorizada.

Na linha 1 Digite: Partida NÃO autorizada. (Aperte a tecla CTRL e ENTER para quebrar a linha, continue digitando) Tempo de partida do MOTOR 2 menor do que 5 segundos.

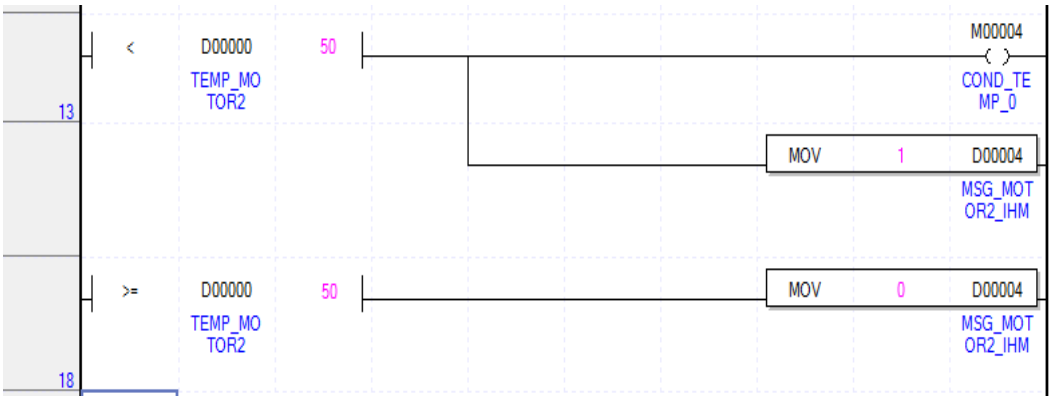
Agora vamos inserir o display que irá mostrar essas mensagens na tela, para isso click em "Word Message":



Abrirá uma janela de propriedades dessa função, em "Device" coloque a memória responsável por receber os valores da linha da tabela:



Lembrando, que a memória D4 é a memória de word colocada na nossa programação com a função de receber dois valores: 0 correspondente a linha 0 da tabela e o número 1, correspondente a linha 1 da tabela. Vamos analisar novamente a linha de programação que fizemos no CLP:



A programação acima significa que quando o tempo de partida do motor 2 (Memória D0) for menor que 50 (5 segundos) o programa irá mover 1 para a memória D4.

Este 1 corresponde a primeira linha da tabela que criamos lá na IHM com a mensagem:

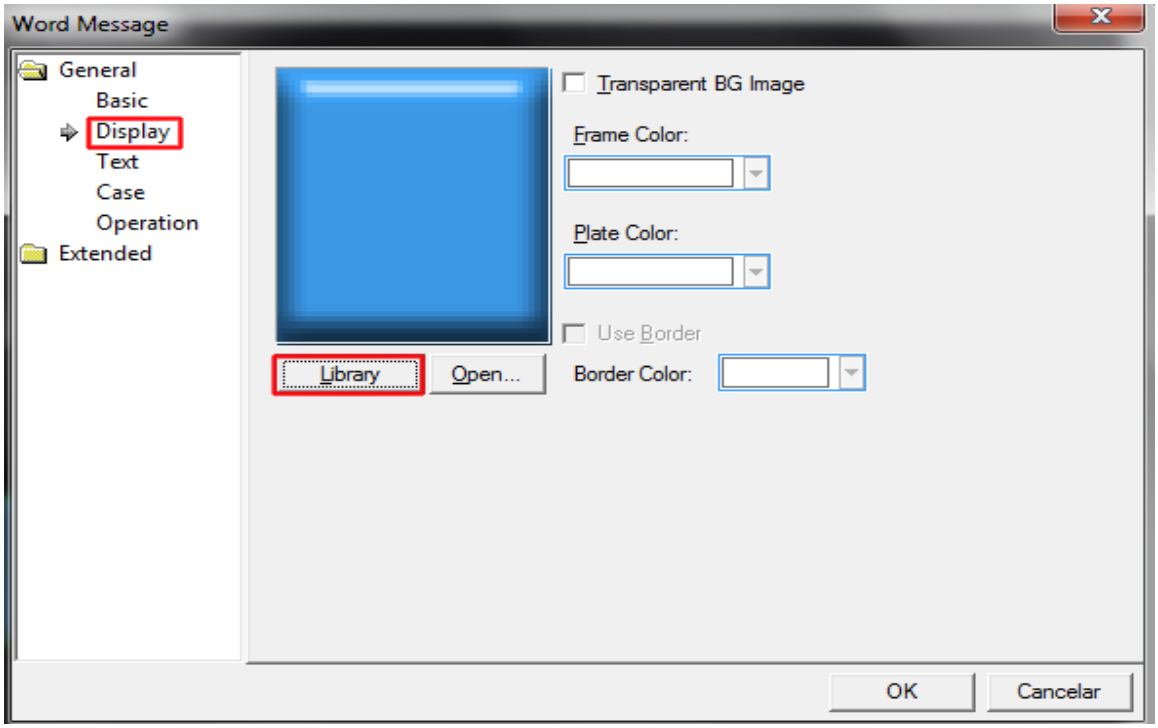
Partida NÃO autorizada.

Tempo de partida do MOTOR 2 menor do que 5 segundos.

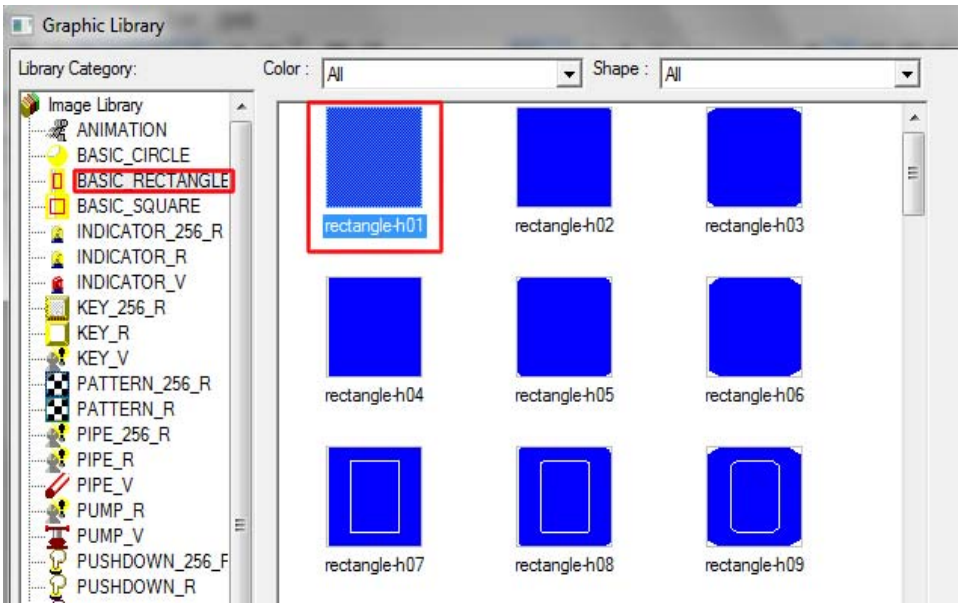
Agora quando este tempo (D0) for maior ou igual a 5 segundos o programa não moverá mais o número 1, mas sim o número 0 correspondente a linha 0 da tabela criada lá na IHM. A IHM por sua vez receberá este 0 e entenderá que deve mostrar no display, que estamos criando neste momento, a mensagem:

Partida Autorizada.

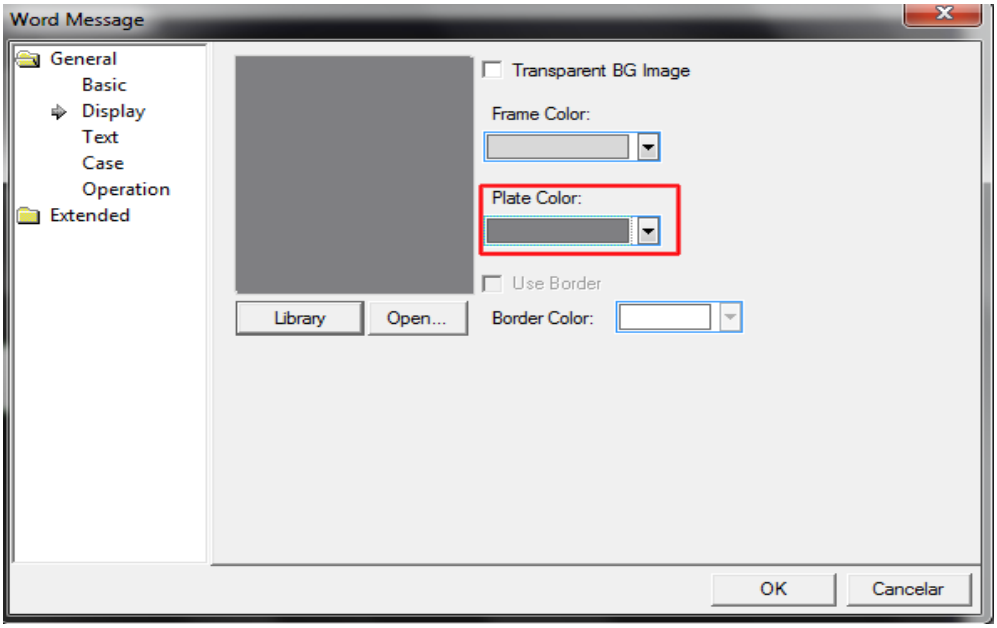
Voltando então ao programa XP-Builder, na janela de propriedades da função "Word Message" click em "Display" e depois em "Library":



Abrirá uma série de displays, botões e imagens para escolhermos, neste caso vamos escolher uma imagem bem básica conforme a imagem abaixo:

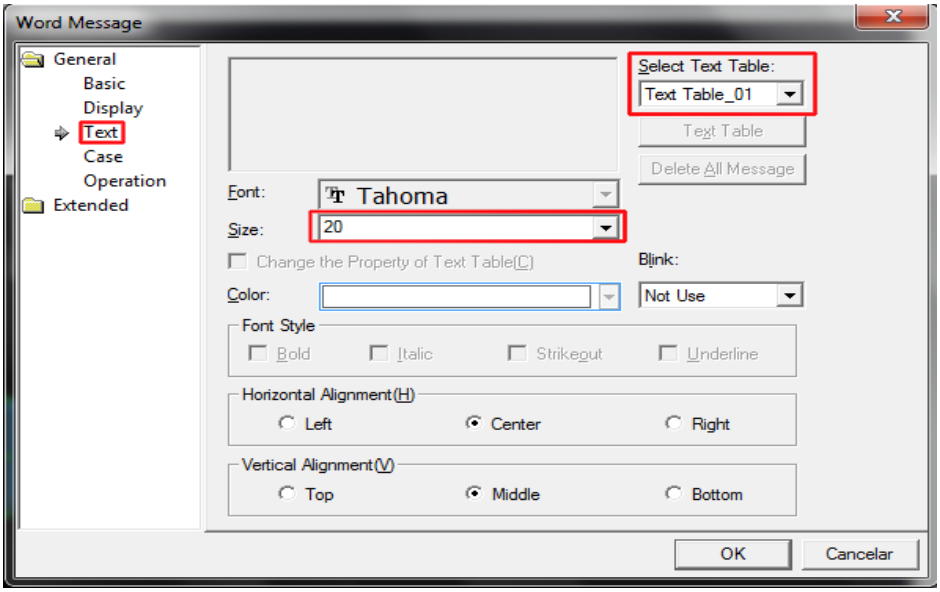


Click em OK:



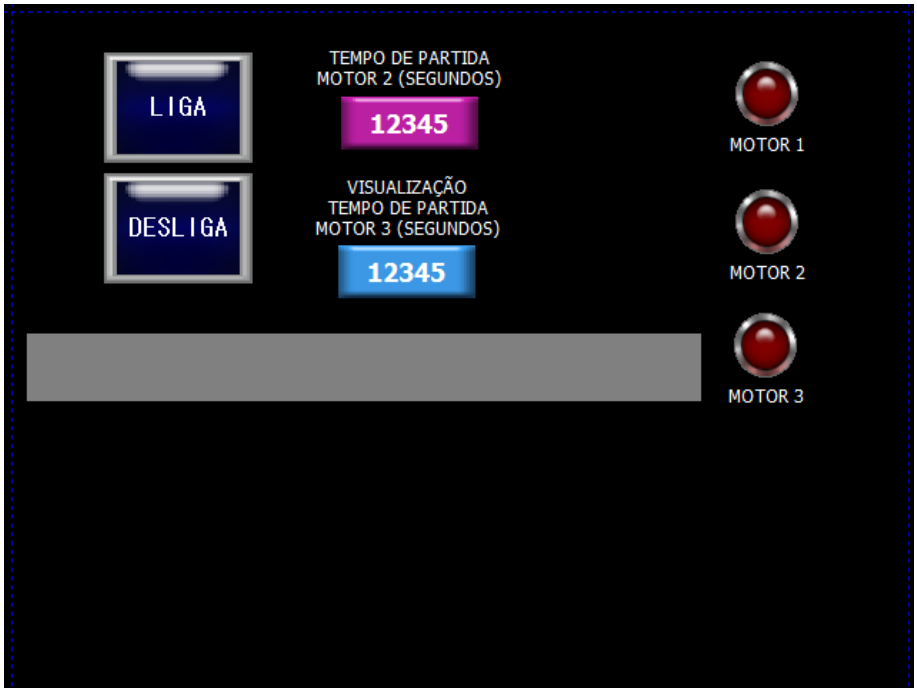
Podemos deixar na cor cinza ou em qualquer outra cor de sua preferência.

Após escolher a imagem adequada, click em “Text”:

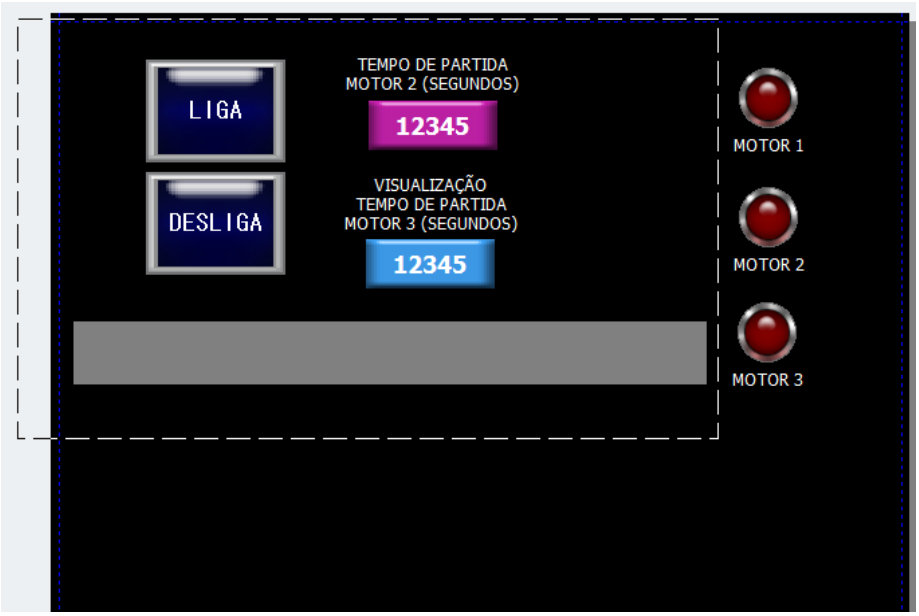


Quando selecionamos “Text Table_01”, estamos selecionando a tabela que criamos e agora a IHM puxará desta tabela, as mensagens conforme o valor que estiver na memória D4. Click em OK.

Você terá a seguinte tela:

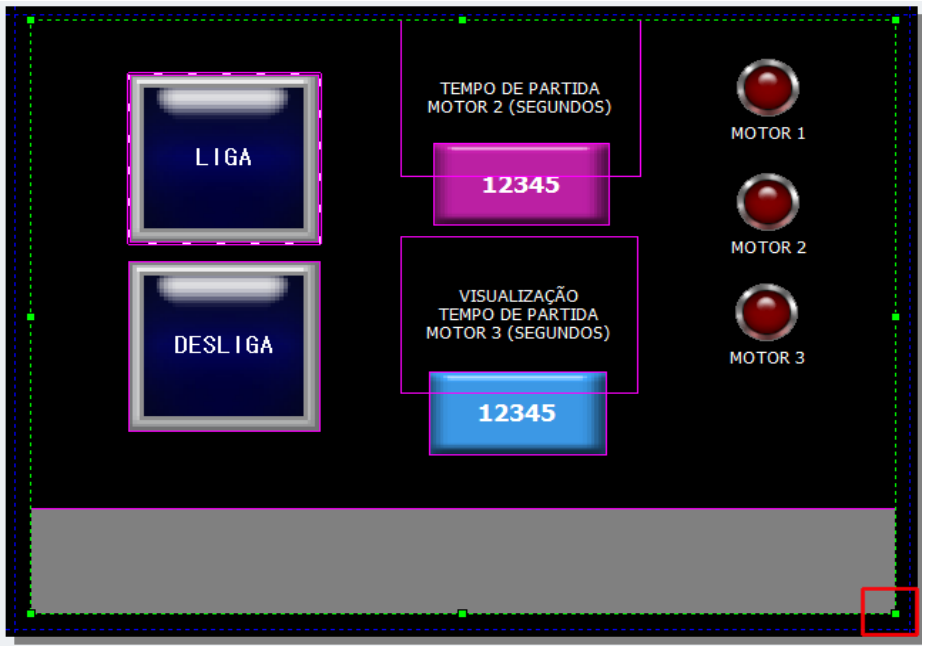


Note que podemos aumentar um pouco os objetos para preencher toda a tela da IHM. Para isso, click com o mouse na parte superior da tela do lado esquerdo e abra uma janela arrastando o ponteiro do mouse e selecionando os itens conforme a tela abaixo:



Note que as três lâmpadas não foram selecionadas.

Agora **você terá a possibilidade de aumentar a janela clicando e arrastando o quadrado verde que se encontra no canto inferior direito da seleção.**

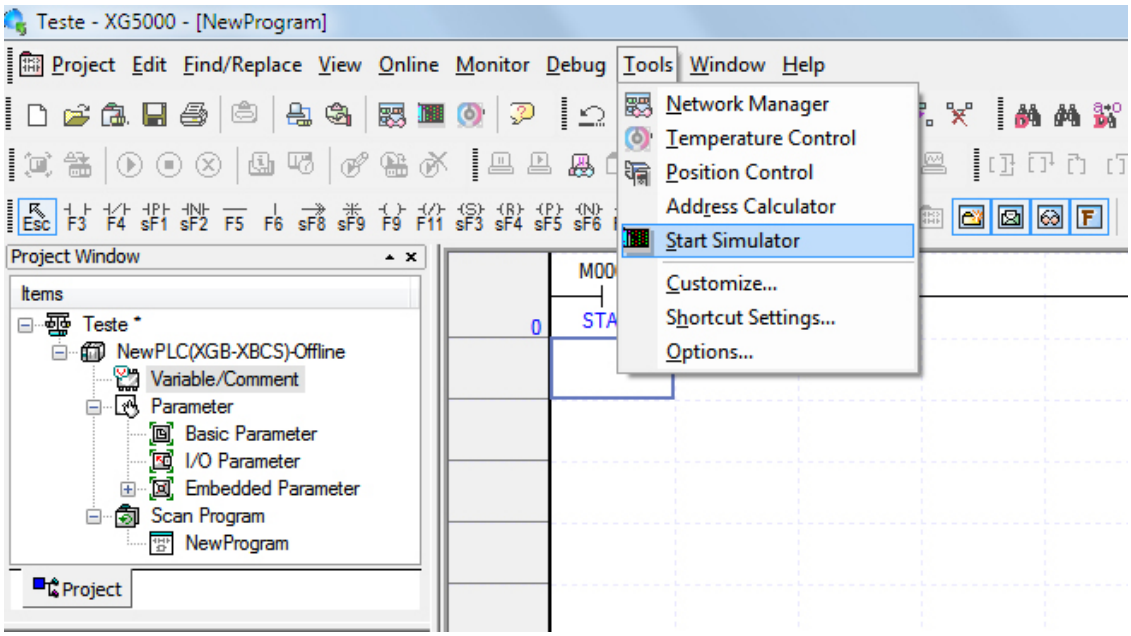


Pronto, nosso programa está completo. Vamos agora aprender a simular o Programa do CLP – XG5000 e o Programa da IHM – XP-Builder ao mesmo tempo para que possamos testar se nossa lógica está funcionando corretamente.

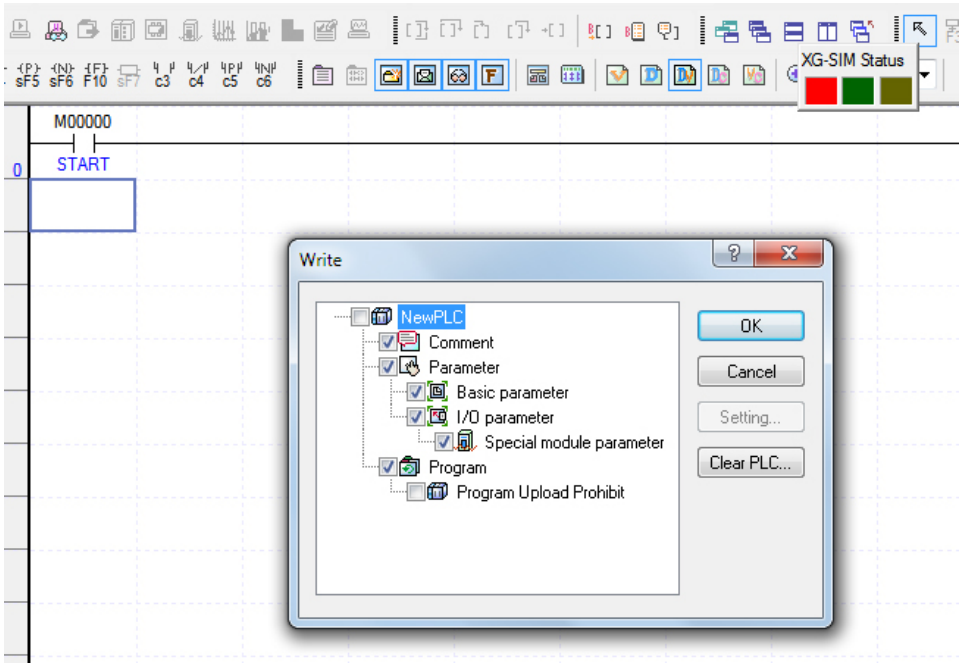
SIMULANDO O PROGRAMA DO CLP

Os CLP's da LS permitem que façamos simulações da programação sem precisar do equipamento físico. Além disso, podemos integrar a programação do CLP com a programação da IHM, simulando virtualmente ambos os dispositivos, o que nos permite testes instantâneos na programação. Não precisamos terminar toda a programação para depois testarmos. Você pode ir testando cada passo criado no programa.

Para executar o modo simulação do CLP, no XG5000, click em Tools > Start Simulation.

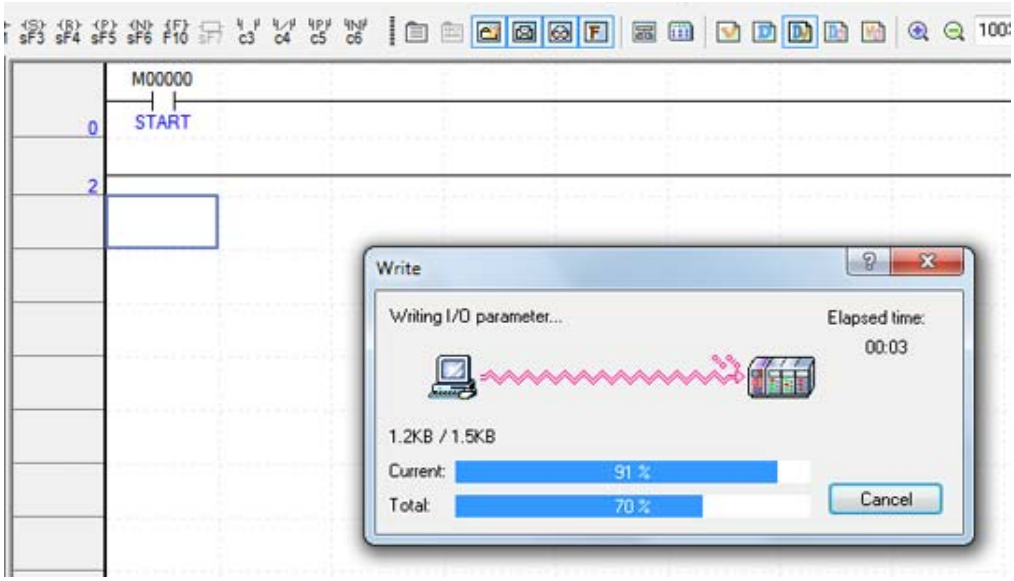


Aguarde o modo de simulação carregar.

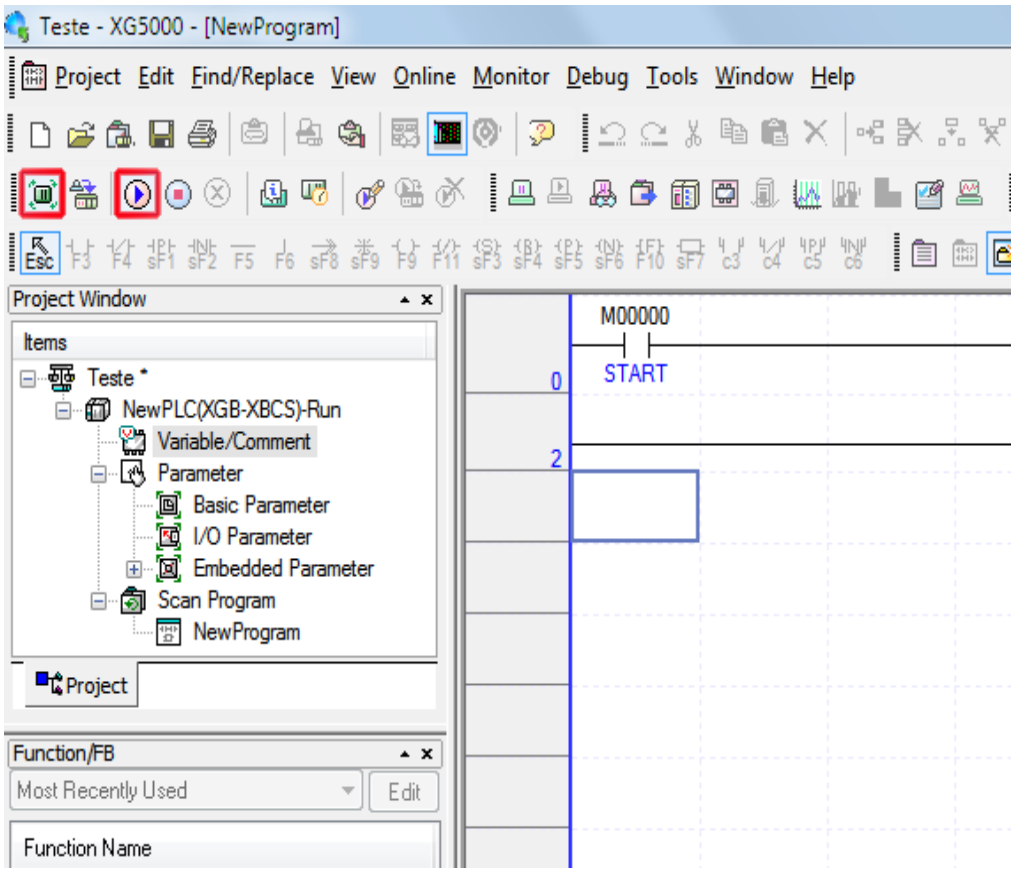


Click em OK.

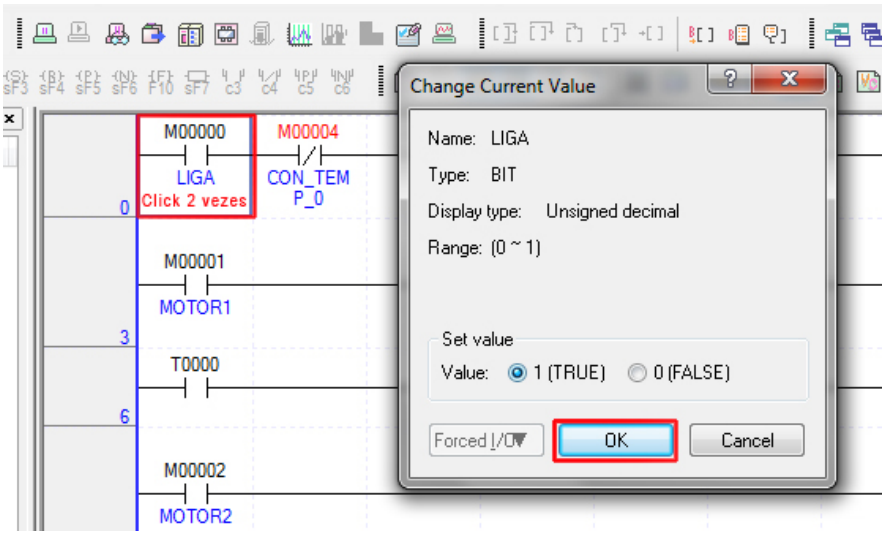
O programa será transferido para o CLP simulando na íntegra uma aplicação real.



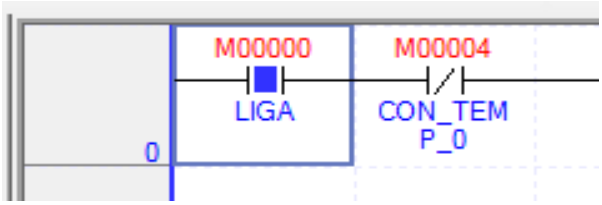
Após a transferência, click em "Run" e em seguida click em Monitor para visualizar o programa rodando no modo simulação:



Quando você clica em monitor, você consegue ver o programa rodando como se estivesse internamente no CLP. Se você clicar duas vezes em cima de um contato aberto, por exemplo, quando estiver em modo Simulação e Monitorando e em seguida clicar em OK, este contato atuará fechando e fazendo o seu papel. Mais dois clicks em cima do mesmo contato e novamente dando OK ele abrirá retornando ao seu estado original:



Note que após clicar duas vezes encima de M0 abrirá uma janela já marcada a opção "1(TRUE)", então você não precisa alterar nada. Apenas clicar em OK. E o contato normalmente aberto de M0 mudará seu estado para "1" (fechado):



Representação do contato após fechado.

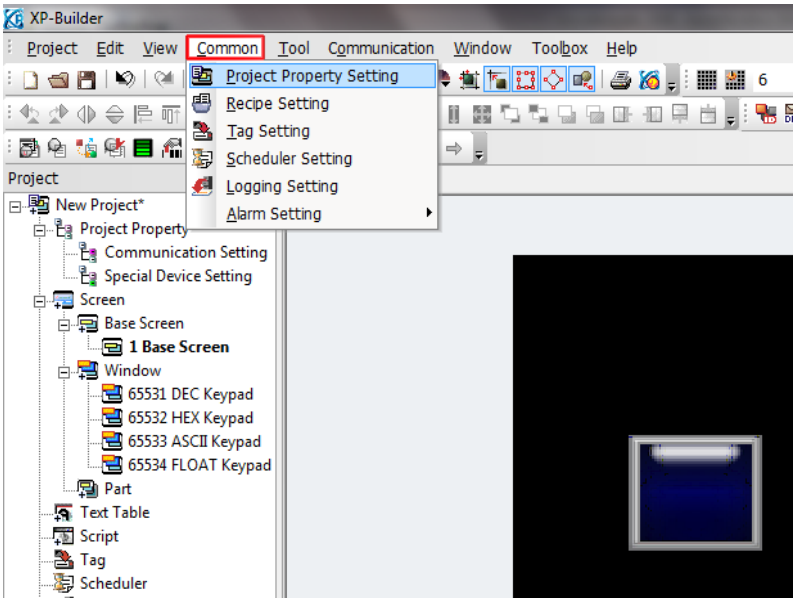
Agora se você der dois clicks novamente e em seguida clicar em OK o contato voltará para seu estado original, isto é "0" (aberto)

Deste modo estamos executando apenas a simulação do CLP.

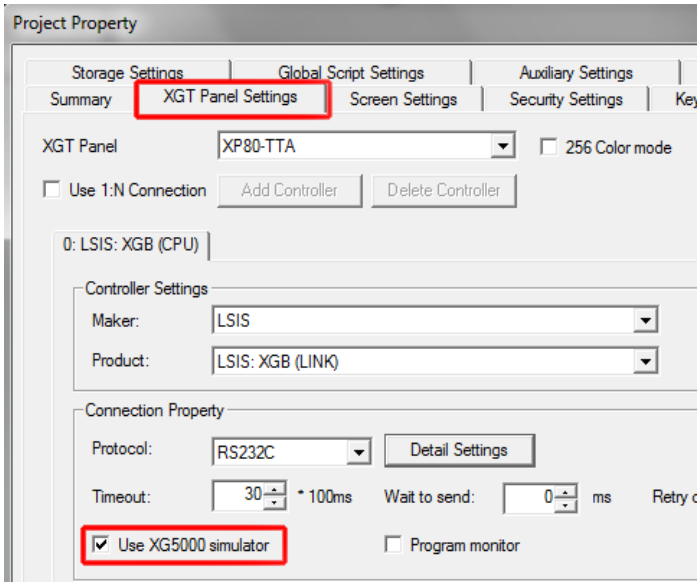
Quando simulamos apenas o programa do CLP a visualização torna-se um pouco confusa, por isso é importante simularmos também, o programa da IHM juntamente com o programa do CLP.

SIMULANDO O PROGRAMA DA IHM

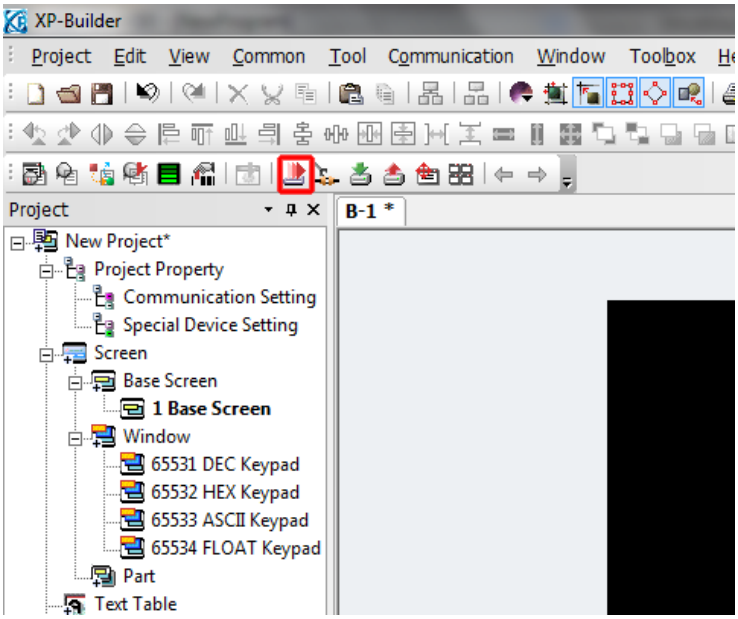
Com a telada IHM pronta podemos simular nosso programa juntamente com o programa do CLP. Para isso, no XP-Builder, click em "Common > Project Property Settings":



Na aba "XGT Panel Settings", marque a opção "Use XG5000 simulator". Isso faz com que os programas do CLP e da IHM se interajam. Click em OK.

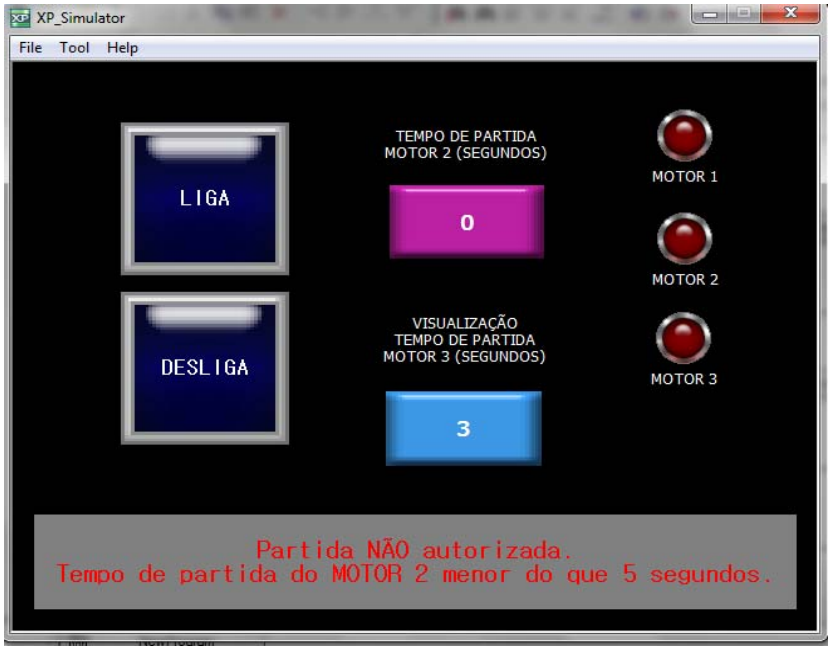


Click no botão de simulação:



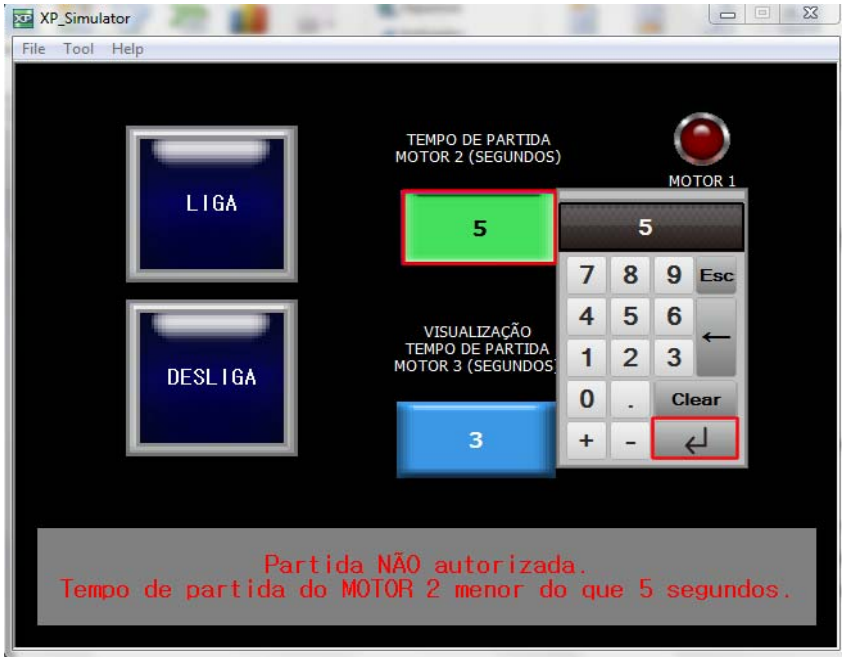
Pronto a IHM Irá abrir a tela de simulação. Só lembrando que para simularmos a IHM juntamente com o CLP, você deve iniciar primeiramente a simulação do CLP no XG5000 e só depois iniciar a simulação da IHM no XP-Builder.

Teremos a seguinte tela:



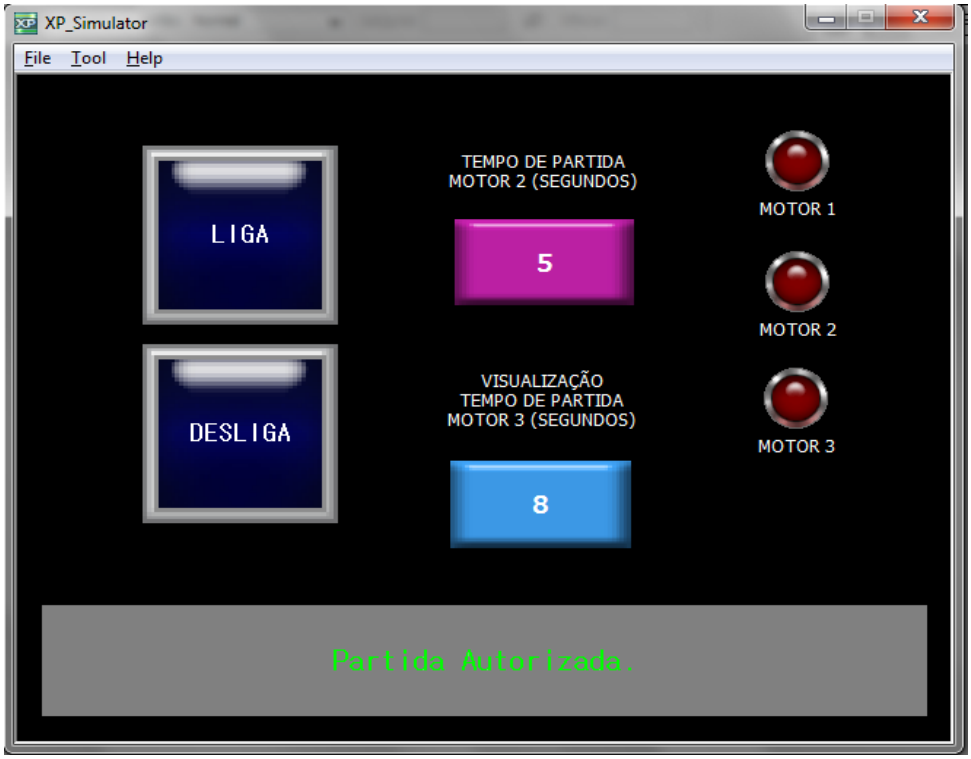
Note que o operador está recebendo a mensagem de partida não autorizada porque o tempo de partida do Motor 2 está em 0 neste momento (menor do que 5 segundos).

Podemos clicar no tempo de partida do Motor 2 (botão roxo) e inserir um valor maior do que 5 e em seguida clicar em ENTER.



Note que quando inserirmos um valor no tempo de partida do Motor 2, automaticamente nossa programação do CLP soma a este valor, 3 segundos e joga no tempo de partida do Motor 3 (display azul):

Com a partida autorizada, click em "LIGA":



O programa deve se comportar da seguinte maneira:

- O primeiro motor partirá logo que você aperta o botão LIGA;

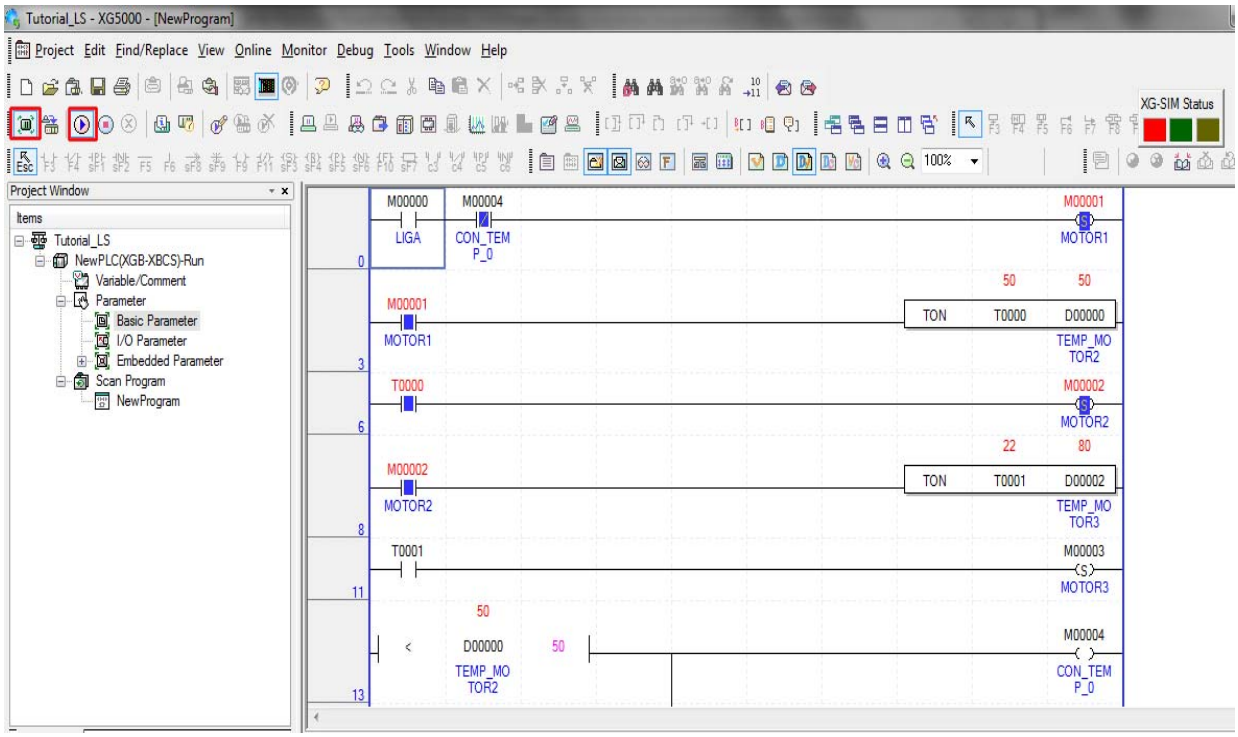
- 5 segundos depois, irá partir o segundo motor;

- E 8 segundos depois do segundo motor ter partido, partirá do terceiro motor.

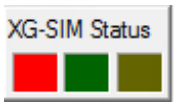
Para desligar os motores, click no botão "Desliga".

Como você está usando o modo de simulação, seu computador talvez não possua a capacidade de responder rapidamente aos clicks na tela. Então ao clicar em um botão na IHM segure o click por uns 2 segundos para ter certeza de que o seu computador processou o comando. Na prática isto não ocorre porque o processamento do CLP é muito rápido não podendo ser comparado ao processamento de um PC.

O importante agora para que se entenda bem a programação que foi feita, é verificar linha por linha e entender todos os passos que o CLP está executando na sua programação Ladder. Por isso existe no modo simulação a opção de "Monitoramento". Com essa opção ligada você consegue ver o acionamento dos botões, bobinas e temporizadores em tempo real.



Este ícone fica visível no programa para mostrar que o XG5000 está em modo simulação:



Você pode arrastá-lo para o canto da tela para não atrapalhar a visualização do programa.

BOA SORTE NOS SEUS ESTUDOS!

FIM.