Módulo I

Desenvolvimento
Software CLP - Básico

Lista de exercícios utilizados nas vídeo aulas e manual de referência das instruções utilizadas em cada aula.

Setor de capacitação técnica Branqs Automação
Santos - SP

índice

Introdução....4
Aula 01 - Instalação da ferramenta de programação do CLP....5
Aula 02 - Campainha residencial....6
Declarações e instruções do CLP....7
Declaração: "CAMPO_ENTRADA_DIGITAL"....7
Declaração: "CAMPO_SAIDA_DIGITAL"....8
Instrução: "SEL"....9
Instrução: "MEMO"....10
Declaração de campos na IHM....11
Campo: "bool"....11
Aula 03 - Acionamento de lâmpada por interruptor de toque....12
Declarações e instruções do CLP....13
Declaração: "SINAL_DIGITAL"....13
Instrução: "SUBIDA"....14
Instrução: "E"....15
Instrução: "EN"....16
Instrução: "OU"....16
Declaração de campos na IHM....18
Aula 04 - Acionamento de lâmpada com sensor de presença....19
Declarações e instruções do CLP....20
Declaração: "CAMPO_INT"....20
Declaração: "TEMPORIZADOR"....21
Instrução: "DESCIDA"....23
Declaração de campos na IHM....24
Campo: "int"....24
Aula 05 - Controle de enchimento de caixa d'água através de bomba....25
Declarações e instruções do CLP....26
Instrução: "SED"....26
Declaração de campos na IHM....27
Aula 06 - Acionamento de lâmpada em horários programados....28
Declarações e instruções do CLP....29
Declaração: "CAMPO_CHAR"....29
Declaração: "AgendamentoSemanal"....30
Declaração de campos na IHM....32
Aula 07 - Alarme residencial....33
Declarações e instruções do CLP....34
Instrução: "if"....34
Declaração de campos na IHM....36
Campo: "Senha"....36
Aula 08 - Controle do tempo de chuveiro ligado....37
Declarações e instruções do CLP....38

Instrução: ENTAO_EXECUTA_BLOCO.....38
Declaração de campos na IHM.....39
Controle de Revisões.....40

Introdução

Este documento apresenta os exercícios que serão resolvidos através das Vídeo-Aulas disponíveis na área de treinamento do site http://www.branqs.com.br.

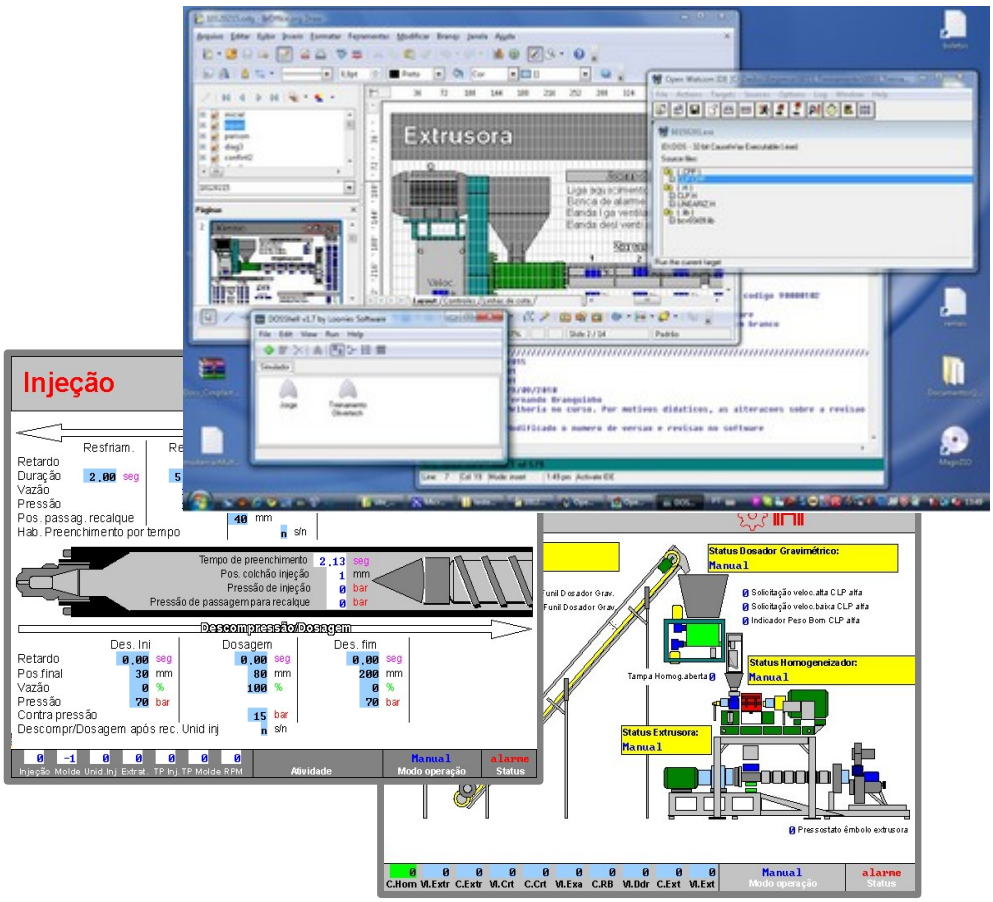
Adicionalmente, este documento também apresenta um curto manual de referência para as instruções utilizadas na solução de cada aula.

Esperamos que todos possam aproveitar o conteúdo.

Ficamos à disposição para prestar suporte através dos "comentários" do próprio YouTube.

Aula 01 - Instalação da ferramenta de programação do CLP

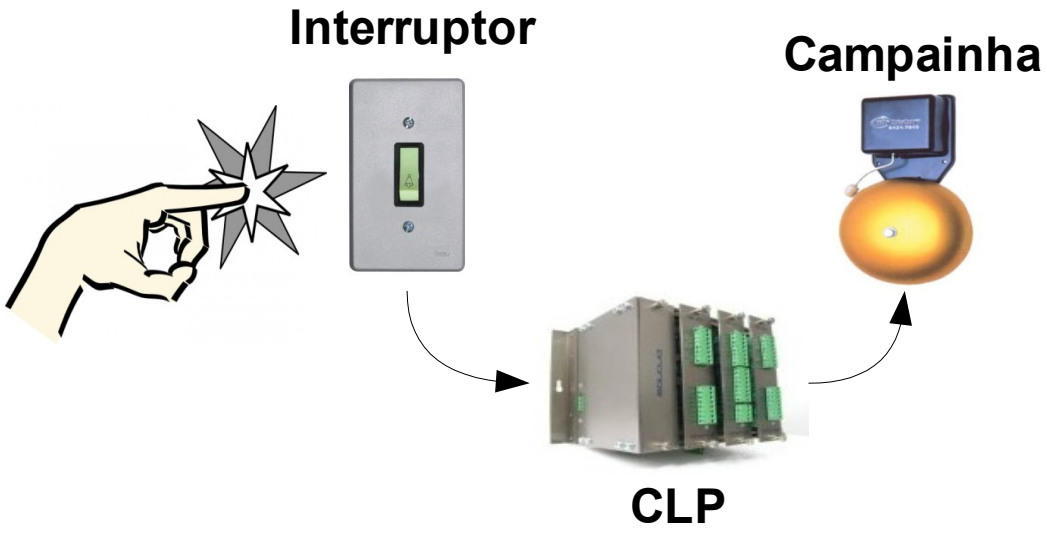
Esta aula propõe simplesmente mostrar como é feita a instalação das ferramentas necessárias para desenvolver o software do CLP branqs. Após a instalação, é demonstrado como abrir as ferramentas de edição do programa CLP, o editor de páginas da IHM e o simulador do software.



Aula 02 - Campainha residencial

Enunciado do exercício

Montar um programa de controle da campainha de uma casa automatizada. A campainha deve tocar sempre que o interruptor for pressionado. Os estados dos sinais da campainha e do interruptor devem ser apresentados na IHM.



Entradas Digitais

- Interruptor de toque

Saídas Digitais

- Campainha

Declarações e instruções do CLP

Declaração: "CAMPO_ENTRADA_DIGITAL"

Cria uma variável que assume o estado de uma entrada digital do CLP.

Sintaxe

```
CAMPO_ENTRADA_DIGITAL(nomeDaEntradaDigital)
```

A variável declarada (nomeDaEntradaDigital) possuirá um valor booleano conforme o estado da entrada digital do CLP.

Comentário

A relação entre a declaração das entradas digitais e as entradas digitais físicas existentes no CLP é feita através da ordem das declarações. A primeira variável declarada corresponderá à primeira entrada da primeira placa BC8E. A segunda variável corresponderá à segunda entrada e assim por diante.

Exemplo

DECLARA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Declaracao Entradas digitais
////////////////////////////////////////////////////////////////////////////

     CAMPO_ENTRADA_DIGITAL(interruptorCampainha)
```

Quando a primeira entrada digital do CLP for acionada, a variável "interruptorCampainha" receberá o estado lógico "1" (verdadeiro).

Declaração: "CAMPO_SAIDA_DIGITAL"

Cria uma variável que atribui seu estado lógico a uma saída digital do CLP

Sintaxe

```
CAMPO_SAIDA_DIGITAL(nomeDaSaidaDigital)
```

A variável declarada poderá receber um valor booleano (“0” ou “1”) conforme o resultado de uma lógica construída no programa. O estado dessa variável é refletido em uma saída digital do CLP.

Comentário

A relação entre a declaração das saídas digitais e as saídas digitais físicas existentes no CLP é feita através da ordem das declarações. A primeira variável declarada corresponderá à primeira saída da primeira placa BC8S. A segunda variável corresponderá à segunda saída e assim por diante.

Exemplo

DECLARA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Declaracao Saidas digitais
////////////////////////////////////////////////////////////////////////////

     CAMPO_SAIDA_DIGITAL(saidaCampainha)
```

Quando a variável "saidaCampainha" receber o estado lógico "1" (Verdadeiro), a primeira saída do CLP será acionada.

Instrução: "SEL"

Instrução normalmente utilizada para iniciar uma sentença lógica entre sinais digitais. A instrução SEL é o acrônimo da expressão "Se Ligado".

Sintaxe

```
SEL (variavelDeEstadoDigital)
```

Comentário

Ao ser executada, a instrução SEL copia o estado da variável indicada para uma outra variável interna chamada de "Estado Lógico Corrente" (ELC). O ELC é uma variável auxiliar interna do CLP usada para realizar a execução e armazenamento do resultado intermediário de uma sentença lógica. Por esse motivo, esta instrução é muito utilizada para iniciarmos uma sentença lógica. É equivalente a um "contato normalmente aberto" no início de uma linha na linguagem de programação LADDER.

Exemplo

DECLARA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Declaracao Entradas digitais
////////////////////////////////////////////////////////////////////////////

      CAMPO_ENTRADA_DIGITAL(interruptorCampainha)

////////////////////////////////////////////////////////////////////////////
//Declaracao Saidas digitais
////////////////////////////////////////////////////////////////////////////

      CAMPO_SAIDA_DIGITAL(saidaCampainha)
```

LOGICA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Comando
////////////////////////////////////////////////////////////////////////////

      SEL          (interruptorCampainha)
      MEMO         (saidaCampainha)
```

Podemos ler a lógica acima do seguinte modo:

Se ligado o interruptorCampainha
Memoriza a saidaCampainha

Ou seja, se a entrada "interruptorCampainha" for acionada, então a "saidaCampainha" também será acionada.

Instrução: "MEMO"

Instrução utilizada para concluir uma operação lógica, armazenando o resultado da sentença em uma variável que armazena um estado lógico (0 ou 1; verdadeiro ou falso). A instrução MEMO é o acrônimo da expressão "MEMORIZA".

Sintaxe

```
MEMO (variavalDeEstadoDigital)
```

Esta instrução finaliza uma sentença lógica, armazenando o "Estado Lógico Corrente" (ELC) na "variavelDeEstadoDigital".

Comentário

A instrução MEMO equivale à "Bobina de um relê" utilizado na linguagem de programação LADDER.

Exemplo

DECLARA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Declaracao Entradas digitais
////////////////////////////////////////////////////////////////////////////

      CAMPO_ENTRADA_DIGITAL(interruptorCampainha)

////////////////////////////////////////////////////////////////////////////
//Declaracao Saidas digitais
////////////////////////////////////////////////////////////////////////////

      CAMPO_SAIDA_DIGITAL(saidaCampainha)
```

LOGICA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Comando
////////////////////////////////////////////////////////////////////////////

      SEL         (interruptorCampainha)
      MEMO        (saidaCampainha)
```

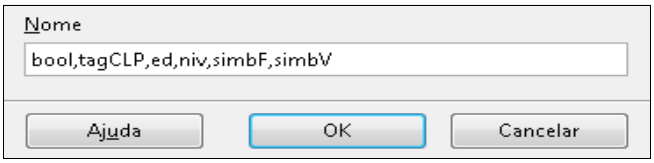
A instrução MEMO irá atribuir à variável "saidaCampainha" o resultado lógico resultante nas linhas anteriores. Considerando que no exemplo existe somente uma instrução SEL antes da instrução MEMO, então o valor da variável "interruptorCampainha" será transferido para a variável "saidaCampainha". Posteriormente serão mostrados exemplos onde a instrução MEMO atribui valores resultantes de sentenças lógicas mais complexas.

Declaração de campos na IHM

Campo: "bool"

Os campos do tipo "bool" permitem a visualização e edição do conteúdo de variáveis do CLP do tipo "CAMPO_ENTRADA_DIGITAL", "CAMPO_SAIDA_DIGITAL" e "CAMPO_CHAR"

Sintaxe



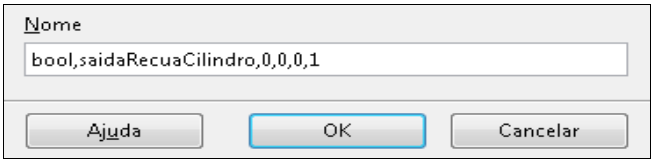
Onde:

bool: Tipo do campo
tagCLP: Nome da variável declarada no CLP
ed: Valor 0 (zero) ou 1 (um). 0 - Não permite edição, 1 - Permite edição
niv: Nível de senha exigida para edição (Valor entre 0 e 3)
simbF: Símbolo de um único digito que será apresentado no campo quando a variável possuir o estado lógico "Falso"
simbV: Símbolo de um único digito que será apresentado no campo quando a variável possuir o estado lógico "Verdadeiro"

Comentário

Os símbolos "0 e 1" ou "N e S" são frequentemente utilizados.

Exemplo

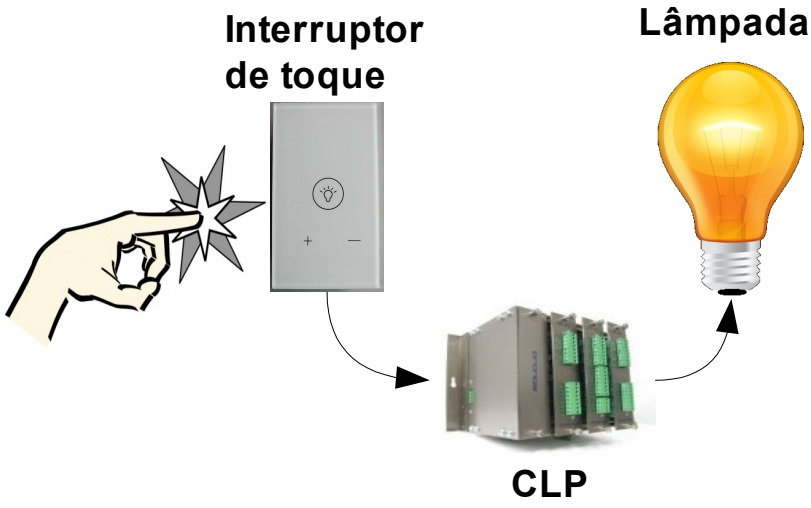


O campo acima apresentará o conteúdo da variável "saidaRecuaCilindro". Esse campo não será editável; terá o nível de proteção 0; mostrará o simbolo "0" (zero) quando a variável possuir o estado lógico "Falso" e o símbolo "1" (um) quando a variável possuir o estado lógico "Verdadeiro".

Aula 03 - Acionamento de lâmpada por interruptor de toque

Enunciado do exercício

Montar o programa de controle de uma lâmpada controlada por um interruptor de toque em uma casa automatizada. Ao tocar no interruptor, deve inverter o estado da lâmpada. Os estados das entradas e saídas digitais devem ser apresentados na IHM.



Entradas Digitais

- Interruptor de toque

Saídas Digitais

- Lâmpada

Declarações e instruções do CLP

Declaração: "SINAL_DIGITAL"

Cria uma variável capaz de armazenar um estado lógico digital, ou seja, pode assumir os estados "Falso" ou "Verdadeiro", também representados por "0" (zero) ou "1" (um).

Sintaxe

```
SINAL_DIGITAL(nomeDaVariavel)
```

Comentário

Variáveis do tipo "SINAL_DIGITAL" são largamente utilizadas como variáveis auxiliares na construção da lógica do CLP. Estas variáveis não são manipuladas pela a IHM. Ao ligarmos o CLP, o estado lógico de uma variável deste tipo é iniciado sempre com "zero".

Exemplo

DECLARA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Declaracao das entradas digitais
////////////////////////////////////////////////////////////////////////////

      CAMPO_ENTRADA_DIGITAL(entradaDigitalCilindroRecuado)

////////////////////////////////////////////////////////////////////////////
//Declaracao Campos e variaveis organizados por funcao
////////////////////////////////////////////////////////////////////////////

      SINAL_DIGITAL(cilindroRecuado)
```

LOGICA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Comando
////////////////////////////////////////////////////////////////////////////

      SEL         (entradaDigitalCilindroRecuado)
      MEMO        (cilindroRecuado)
```

Quando o sinal "entradaDigitalCilindroRecuado" possuir o estado lógico "verdadeiro", a variável "**cilindroRecuado**" também receberá o estado lógico "verdadeiro".

Instrução: "SUBIDA"

Permite verificar se ocorreu a transição de subida (0 para 1) no estado lógico de uma variável ou no resultado de uma sentença lógica.

Sintaxe

SUBIDA

Comentário

A instrução SUBIDA é extremamente útil para detectar o exato momento de acionamento de um sinal. Essa instrução avalia o resultado da sentença lógica inserida anteriormente a ela, e resulta em estado lógico "verdadeiro" somente durante um ciclo de execução do CLP, exatamente no momento de transição do estado lógico "falso" para "verdadeiro". (assistir explicação mostrada na vídeo aula).

Exemplo

DECLARA.CLP

```
////////////////////////////////////////////////////////////////////////////
// Declaracao Entradas digitais
////////////////////////////////////////////////////////////////////////////

      CAMPO_ENTRADA_DIGITAL(botao)

////////////////////////////////////////////////////////////////////////////
// Declaracao Saidas digitais
////////////////////////////////////////////////////////////////////////////

      CAMPO_SAIDA_DIGITAL(sirene)
```

LOGICA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Comando
////////////////////////////////////////////////////////////////////////////

      SEL         (botao)
      SUBIDA
      MEMO        (sirene)
```

Quando o botão for pressionado, a sirene realizará um breve toque (durante milésimos de segundos) como se emitisse um "beep". Perceba que a sirene não continuará tocando enquanto o botão estiver pressionado. Para que ela realize o breve toque novamente, será necessário soltar o botão e pressioná-lo novamente. Somente dessa forma a instrução SUBIDA irá detectar a transição do sinal (de 0 para 1) e então acionar a sirene novamente durante um único ciclo de CLP.

Instrução: "E"

Realiza uma operação lógica "E" (AND) entre o "Estado Lógico Corrente" (ELC) e a variável passada como parâmetro.

Sintaxe

```
E  (sinalDigital)
```

Comentário

O resultado da operação "AND" entre a variável passada e o "Estado Lógico Corrente" (ELC) é armazenado no próprio ELC. Esse resultado pode ser utilizado em novas operações ou ser finalmente armazenado em uma variável destino através da operação MEMO. Esta instrução equivale à utilização de uma ligação "em série" de um "contato aberto" através da linguagem LADDER.

Exemplo

DECLARA.CLP

```
////////////////////////////////////////////////////////////////////////////
// Declaracao das entradas digitais
////////////////////////////////////////////////////////////////////////////

     CAMPO_ENTRADA_DIGITAL(sensorPortaFrontalFechada)
     CAMPO_ENTRADA_DIGITAL(sensorPortaTraseiraFechada)

////////////////////////////////////////////////////////////////////////////
//Declaracao Campos e variaveis organizados por funcao
////////////////////////////////////////////////////////////////////////////

     SINAL_DIGITAL(todasPortasFechadas)
```

LOGICA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Comando
////////////////////////////////////////////////////////////////////////////

     SEL          (sensorPortaFrontalFechada)
     E            (sensorPortaTraseiraFechada)
     MEMO         (todasPortasFechadas)
```

O sinal "todasPortasFechadas" só será "verdadeiro" caso o sinal "sensorPortaFrontalFechada" e o sinal "sensorPortaTraseiraFechada" também sejam.

Instrução: "EN"

Realiza uma operação lógica "E NÃO" (AND NOT) entre o "Estado Lógico Corrente" (ELC) e a variável passada como parâmetro.

Sintaxe

```
EN   (sinalDigital)
```

Comentário

O resultado da operação "AND NOT" entre a variável passada e o "Estado Lógico Corrente" (ELC) é armazenado no próprio ELC. Esse resultado pode ser utilizado em novas operações ou ser finalmente armazenado em uma variável destino através da operação MEMO. Esta instrução equivale à utilização de uma ligação "em série" com um "contato fechado" através da linguagem LADDER.

Exemplo

DECLARA.CLP

```
////////////////////////////////////////////////////////////////////////////
// Declaracao das entradas digitais
////////////////////////////////////////////////////////////////////////////

      CAMPO_ENTRADA_DIGITAL(botaoRecuaCilindro)
      CAMPO_ENTRADA_DIGITAL(protetorAberto)

////////////////////////////////////////////////////////////////////////////
// Declaracao das saidas digitais
////////////////////////////////////////////////////////////////////////////

      CAMPO_SAIDA_DIGITAL(saidaRecuaCilindro)
```

LOGICA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Comando
////////////////////////////////////////////////////////////////////////////

      SEL          (botaoRecuaCilindro)
      EN           (protetorAberto)
      MEMO         (saidaRecuaCilindro)
```

No exemplo acima, a "saidaRecuaCilindro" só será acionada caso seja pressionado o botão e o protetor **não** esteja aberto.

Instrução: "OU"

Realiza uma operação lógica "OU" (OR) entre o "Estado Lógico Corrente" (ELC) e a variável passada como parâmetro. O resultado é armazenado no próprio ELC.

Sintaxe

```
OU   (sinalDigital)
```

Comentário

O resultado da operação "OR" entre a variável passada e o "Estado Lógico Corrente" (ELC) é armazenado no próprio ELC. Esse resultado pode ser utilizado em novas operações ou ser finalmente armazenado em uma variável destino através da operação MEMO. Esta instrução equivale à utilização de uma ligação "em paralelo" com um "contato" através da linguagem LADDER.

Exemplo

DECLARA.CLP

```
////////////////////////////////////////////////////////////////////////////
// Declaracao das entradas digitais
////////////////////////////////////////////////////////////////////////////

      CAMPO_ENTRADA_DIGITAL(botaoLigaEsteira)
      CAMPO_ENTRADA_DIGITAL(botaoRecuaCilindro)

////////////////////////////////////////////////////////////////////////////
//Declaracao Campos e variaveis organizados por funcao
////////////////////////////////////////////////////////////////////////////

      SINAL_DIGITAL(pressionouBotao)
```

LOGICA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Comando
////////////////////////////////////////////////////////////////////////////

      SEL         (botaoRecuaCilindro)
      OU          (botaoLigaEsteira)
      MEMO        (pressionouBotao)
```

Se o "botaoRecuaCilindro" ou o "botaoLigaEsteira" forem pressionados, o sinal "pressionouBotao" será verdadeiro.

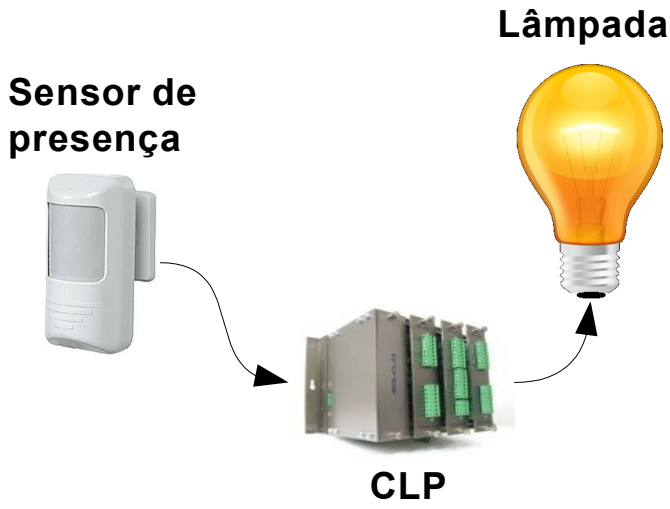
Declaração de campos na IHM

Esta aula não apresenta novos tipos de campos.

Aula 04 - Acionamento de lâmpada com sensor de presença

Enunciado do exercício

Ligar automaticamente a iluminação de um cômodo ao detectar a presença de uma pessoa. Garantir que a lâmpada fique ligada por aproximadamente 5 segundos após cada detecção de movimento realizada pelo sensor. O sistema deve permitir alterar o tempo de acionamento da lâmpada através de um campo na IHM. Os estados das entradas e saídas digitais devem ser apresentados na IHM.



Entradas Digitais

- Sensor de presença

Saídas Digitais

- Lâmpada

Declarações e instruções do CLP

Declaração: "CAMPO_INT"

Cria uma variável para armazenamento de valores numéricos inteiros, que podem ser apresentados ou modificados através do painel de operação (IHM).

Sintaxe

```
CAMPO_INT (nomeDaVariavel)
```

Comentário

Variáveis do tipo CAMPO_INT podem ser utilizadas em inúmeras ocasiões. Durante as vídeo aulas da certificação M1-DSCLPB, elas serão intensamente utilizadas para armazenar o set-point de temporizadores. Serão usadas também para armazenar contadores e até mesmo números de senhas. Estas variáveis podem ser facilmente comparadas através de operadores lógicos e manipuladas por operadores aritméticos e de atribuição da linguagem "C". Este tipo de variável é sempre inicializada com o valor zero ao iniciar a execução do programa.

Exemplo

DECLARA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Declaracao Campos e variaveis organizados por funcao
////////////////////////////////////////////////////////////////////////////

      CAMPO_INT(temporizadorRecuoDoCilindro)
```

LOGICA.CLP

```

////////////////////////////////////////////////////////////////////////////
//Comando
////////////////////////////////////////////////////////////////////////////

      temporizadorRecuoDoCilindro = 100;
```

O valor 100 é armazenado na variável "temporizadorRecuoDoCilindro".

Declaração: "TEMPORIZADOR"

Declara um temporizador para contagem de tempo pelo CLP.

Sintaxe

```
TEMPORIZADOR(nomeDoTemporizador,variavelSetPoint)
```

Onde:

nomeDoTemporizador: nome definido para o temporizador

variavelSetPoint: Variável do tipo INT que armazena o valor do tempo em centésimos de segundo a ser contado pelo temporizador (ex: 100 = 1 segundo)

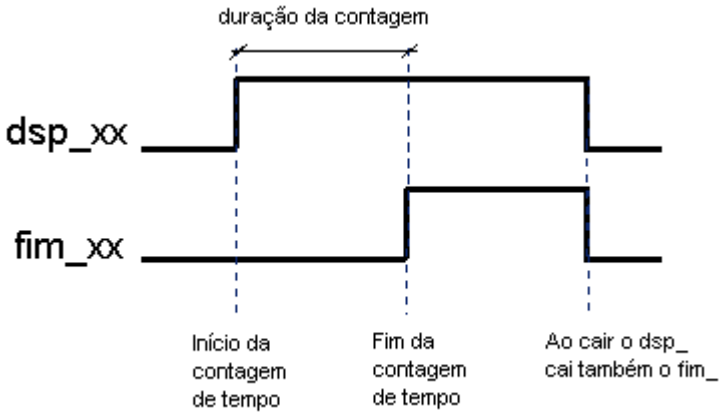
Comentário

Ao declararmos um TEMPORIZADOR, são criados automaticamente dois sinais digitais associados:

```
dsp_nomeDoTemporizador

fim_nomeDoTemporizador
```

Estes sinais serão utilizados para controlar e monitorar o uso do temporizador. O sinal com o prefixo "**dsp**_" (disparo) serve para disparar a contagem. Ao receber o estado lógico "1" ("Um" ou "Verdadeiro"), o temporizador começa a fazer a contagem do tempo. Quando for decorrido o tempo programado na variável "**variavelSetPoint**", o sinal com o prefixo "**fim**_" (Fim) passará automaticamente para o estado lógico "1". Dessa forma, a variável "**dsp**_" deverá ser alterada por uma sentença lógica existente no programa do CLP, enquanto a variável "**fim**_" será alterada automaticamente pelo próprio temporizador e será consultada para verificar se a contagem do tempo chegou ao fim. Segue abaixo uma pequena carta de tempos que demonstra o funcionamento dos sinais de "disparo" e reconhecimento de "fim" da contagem:



Adicionalmente um temporizador permite também obtermos o valor corrente de sua

contagem. Para isso, devemos utilizar o método **tempoDecorrido( )** conforme é demonstrado no exemplo a seguir.

Exemplo

DECLARA.CLP

```
////////////////////////////////////////////////////////////////////////////
// Declaracao das entradas digitais
////////////////////////////////////////////////////////////////////////////

      CAMPO_ENTRADA_DIGITAL(sensorPoteEmBaixoDoCilindro)

////////////////////////////////////////////////////////////////////////////
// Declaracao das saidas digitais
////////////////////////////////////////////////////////////////////////////

      CAMPO_SAIDA_DIGITAL(avancaCilindro)

////////////////////////////////////////////////////////////////////////////
// Declaracao Campos e variaveis organizados por funcao
////////////////////////////////////////////////////////////////////////////

      CAMPO_INT(duracaoDeAvancoDoCilindro)
      TEMPORIZADOR(tmpDuracaoDeAvancoDoCilindro,duracaoDeAvancoDoCilindro)

      CAMPO_INT(contagemDoTempo)
```

LOGICA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Comando
////////////////////////////////////////////////////////////////////////////

      duracaoDeAvancoDoCilindro = 200;

      SEL         (sensorPoteEmBaixoDoCilindro)
      MEMO        (dsp_tmpDuracaoDeAvancoDoCilindro)

      SEL         (dsp_tmpDuracaoDeAvancoDoCilindro)
      EN          (fim_tmpDuracaoDeAvancoDoCilindro)
      MEMO        (avancaCilindro)

      contagemDoTempo=tmpDuracaoDeAvancoDoCilindro.tempoDecorrido();
```

No exemplo acima, a saída "avancaCilindro" será acionada durante 2 segundos sempre que a entrada "sensorPoteEmbaixoDoCilindro" for acionada. Se a variável "contagemDoTempo" for inserida em um campo da IHM, será possível visualizar a contagem do tempo em andamento.

Instrução: "DESCIDA"

Permite verificar se ocorreu a transição de descida (1 para 0) no estado lógico de uma variável ou no resultado de uma operação lógica.

Sintaxe

DESCIDA

Comentário

Funciona de modo similar à instrução "SUBIDA", porém detectando a transição do estado de 1 para 0.

Exemplo

DECLARA.CLP

```
////////////////////////////////////////////////////////////////////////////
// Declaracao Entradas digitais
////////////////////////////////////////////////////////////////////////////
      CAMPO_ENTRADA_DIGITAL(botao)

////////////////////////////////////////////////////////////////////////////
// Declaracao Saidas digitais
////////////////////////////////////////////////////////////////////////////
      CAMPO_SAIDA_DIGITAL(sirene)

////////////////////////////////////////////////////////////////////////////
//Declaracao Campos e variaveis organizados por funcao
////////////////////////////////////////////////////////////////////////////
      SINAL_DIGITAL(pressionouBotao)
      SINAL_DIGITAL(soltouBotao)
```

LOGICA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Comando
////////////////////////////////////////////////////////////////////////////

      SEL         (botao)
      SUBIDA
      MEMO        (pressionouBotao)

      SEL         (botao)
      DESCIDA
      MEMO        (soltouBotao)

      SEL         (pressionouBotao)
      OU          (soltouBotao)
      MEMO        (sirene)
```

No exemplo acima, a sirene realizará um beep quando o botão for pressionado e também quando for liberado.

Declaração de campos na IHM

Campo: "int"

Os campos do tipo "int" permitem a visualização e edição do conteúdo de variáveis do tipo inteiro.

Sintaxe

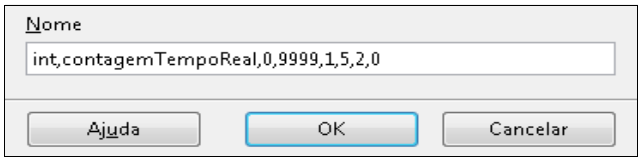
Onde:

int: Tipo do campo
tagCLP: Nome da variável declarada no CLP
vMin: Valor mínimo permitido durante edição
vMax: Valor máximo permitido durante edição
ed: Valor 0 (zero) ou 1 (um). 0 - Não permite edição, 1 - Permite edição
tam: Tamanho do campo em número de dígitos
pto: Posição do ponto decimal
niv: Nível de senha exigida para edição (Valor entre 0 e 3)

Comentário

Este campo permite visualizar e alterar o valor de variáveis declaradas como CAMPO_INT no CLP.

Exemplo

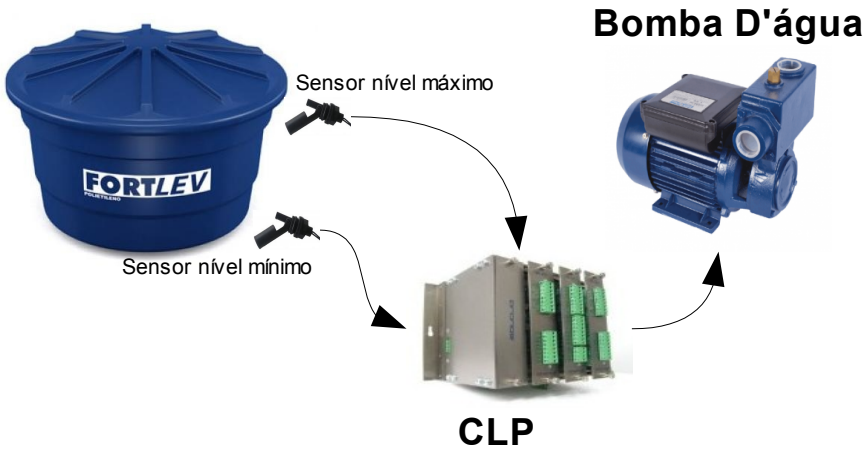


O exemplo acima apresenta o conteúdo da variável "contagemTempoReal". Essa variável pode receber valores entre 0 e 9999. O campo será "editável" e ocupará 5 posições na tela. O ponto decimal será apresentado na segunda posição. Por último, este campo não está protegido por senha, já que o último parâmetro especifica nível de proteção "0" (zero).

Aula 05 - Controle de enchimento de caixa d'água através de bomba

Enunciado do exercício

Controlar o enchimento automático de uma caixa d'água acionando uma bomba d'água sempre que o nível mínimo for atingido. A bomba deverá permanecer ligada até que o nível máximo seja completado. Cada sensor é acionado quando o nível da água atinge a altura do mesmo.



Entradas Digitais

- Sensor de nível máximo de água
- Sensor de nível mínimo de água

Saídas Digitais

- Bomba d'água

Declarações e instruções do CLP

Instrução: "SED"

Instrução normalmente utilizada para iniciar uma sentença lógica entre sinais digitais. A instrução SED é o acrônimo da expressão "Se Desigado".

Sintaxe

```
SED (variavelDeEstadoDigital)
```

Comentário

Ao ser executada, a instrução SED copia o estado "invertido" da variável indicada para a variável do "Estado Lógico Corrente" (ELC). É equivalente a um "contato fechado" no início de uma linha na linguagem de programação LADDER.

Exemplo

DECLARA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Declaracao Entradas digitais
////////////////////////////////////////////////////////////////////////////

      CAMPO_ENTRADA_DIGITAL(sensorPortaFechada)

////////////////////////////////////////////////////////////////////////////
//Declaracao Saidas digitais
////////////////////////////////////////////////////////////////////////////

      CAMPO_SAIDA_DIGITAL(saidaCampainha)
```

LOGICA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Comando
////////////////////////////////////////////////////////////////////////////

      SED           (sensorPortaFechada)
      MEMO          (saidaCampainha)
```

Podemos ler a lógica acima do seguinte modo:

Se desligado o sensorPortaFechada
Memoriza a saidaCampainha

Ou seja, se a porta estiver aberta então aciona a Campainha.

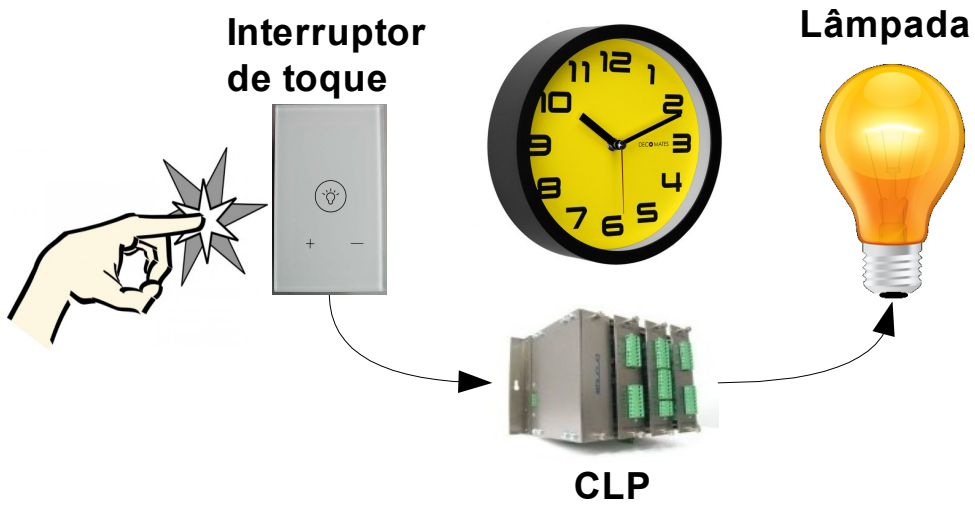
Declaração de campos na IHM

Esta aula não apresenta novos tipos de campos.

Aula 06 - Acionamento de lâmpada em horários programados

Enunciado do exercício

Permitir a definição de um horário para ligar e desligar automaticamente uma lâmpada a cada dia da semana. Manter a função de acionamento manual desenvolvido na aula 03. A definição dos horários deve ser feito através de uma página na IHM.



Entradas Digitais

- Interruptor de toque

Saídas Digitais

- Lâmpada

Declarações e instruções do CLP

Declaração: "CAMPO_CHAR"

Similar à declaração "SINAL_DIGITAL", esta declaração cria uma variável capaz de armazenar um estado lógico digital, ou seja, esta variável pode assumir os estados "Falso" ou "Verdadeiro", também representados por "0" (zero) ou "1" (um). A diferença na utilização de "CAMPO_CHAR" e "SINAL_DIGITAL" é que a variável "CAMPO_CHAR" poderá ser editada e visualizada na tela da IHM.

Sintaxe

```
CAMPO_CHAR (nomeDaVariavel)
```

Comentário

Idem a "CAMPO_SINAL_DIGITAL", porém permite sua edição e visualização diretamente na IHM.

Exemplo

Idem a "SINAL_DIGITAL"

Declaração: "AgendamentoSemanal"

A declaração "AgendamentoSemanal" cria um objeto de agendamento de eventos semanal, capaz de identificar quando o relógio do CLP atinge um determinado horário (hora e minuto) em cada dia da semana.

Sintaxe

```
AgendamentoSemanal nomeDoAgendador;
```

Após declarar o objeto "agendador", devemos programar os horários de cada evento por dia da semana. Essa programação do objeto é realizada através do método "programaAgenda()" utilizando-se os seguintes parâmetros:

```
nomeDoAgendador.programaAgenda(
 variavelHabilitaEventoDomingo,variavelHoraDomingo,variavelMinutoDomingo,
 variavelHabilitaEventoSegunda,variavelHoraSegunda,variavelMinutoSegunda,
 variavelHabilitaEventoTerca,variavelHoraTerca,variavelMinutoTerca,
 variavelHabilitaEventoQuarta,variavelHoraQuarta,variavelMinutoQuarta,
 variavelHabilitaEventoQuinta,variavelHoraQuinta,variavelMinutoQuinta,
 variavelHabilitaEventoSexta,variavelHoraSexta,variavelMinutoSexta,
 variavelHabilitaEventoSabado,variavelHoraSabado,variavelMinutoSabado);
```

Onde:

variavelHabilitaDiaSemana: Variável de estado lógico que habilita o evento naquele dia da semana.

variavelHoraDiaSemana: Variável do tipo INT que armazena a hora de ocorrência do evento naquele dia da semana.

variavelMinutoDiaSemana: Variável do tipo INT que armazena o minuto de ocorrência do evento naquele dia da semana.

Por fim utilizamos o método "verificaAtivacao()" para identificar quando o relógio do CLP atingiu o horário programado em cada evento habilitado. Ex:

```
variavelEstadoDigital=nomeDoAgendador.verificaAtivacao();
```

No exemplo acima, a "variavelEstadoDigital" receberá o estado "1" sempre que um determinado horário habilitado em um dia da semana for atingido.

Comentário

Um bom exemplo da utilização deste recurso seria considerar o acionamento de uma sirene no horário de fim de expediente de uma empresa que funciona de Segunda a Sexta-Feira. O código a seguir demonstra exatamente este exemplo:

Exemplo

DECLARA.CLP

```
////////////////////////////////////////////////////////////////////////////
// Declaracao Saidas digitais
////////////////////////////////////////////////////////////////////////////

      CAMPO_SAIDA_DIGITAL(saidaSirene)

////////////////////////////////////////////////////////////////////////////
// Declaracao Campos e variaveis organizados por funcao
////////////////////////////////////////////////////////////////////////////

      AgendamentoSemanal agendamentoHorarioFimDeExpediente;

      CAMPO_CHAR(habAgendamentoDom)
      CAMPO_CHAR(habAgendamentoSeg)
      CAMPO_CHAR(habAgendamentoTer)
      CAMPO_CHAR(habAgendamentoQua)
      CAMPO_CHAR(habAgendamentoQui)
      CAMPO_CHAR(habAgendamentoSex)
      CAMPO_CHAR(habAgendamentoSab)

      CAMPO_INT(horaFimExpedienteDom)
      CAMPO_INT(horaFimExpedienteSeg)
      CAMPO_INT(horaFimExpedienteTer)
      CAMPO_INT(horaFimExpedienteQua)
      CAMPO_INT(horaFimExpedienteQui)
      CAMPO_INT(horaFimExpedienteSex)
      CAMPO_INT(horaFimExpedienteSab)

      CAMPO_INT(minutoFimExpedienteDom)
      CAMPO_INT(minutoFimExpedienteSeg)
      CAMPO_INT(minutoFimExpedienteTer)
      CAMPO_INT(minutoFimExpedienteQua)
      CAMPO_INT(minutoFimExpedienteQui)
      CAMPO_INT(minutoFimExpedienteSex)
      CAMPO_INT(minutoFimExpedienteSab)
```

LOGICA.CLP

```
////////////////////////////////////////////////////////////////////////
// Comando
////////////////////////////////////////////////////////////////////////

      habAgendamentoDom=0;
      habAgendamentoSeg=1;
      habAgendamentoTer=1;
      habAgendamentoQua=1;
      habAgendamentoQui=1;
      habAgendamentoSex=1;
      habAgendamentoSab=0;

      horaFimExpedienteSeg=17;
      horaFimExpedienteTer=17;
      horaFimExpedienteQua=17;
      horaFimExpedienteQui=17;
      horaFimExpedienteSex=16;

      minutoFimExpedienteSeg=15;
      minutoFimExpedienteTer=15;
      minutoFimExpedienteQua=15;
      minutoFimExpedienteQui=15;
      minutoFimExpedienteSex=30;

      agendamentoLigaLampadaDoCorredor.programaAgenda(
            habAgendamentoDom,horaFimExpedienteDom,minutoFimExpedienteDom,
            habAgendamentoSeg,horaFimExpedienteSeg,minutoFimExpedienteSeg,
            habAgendamentoTer,horaFimExpedienteTer,minutoFimExpedienteTer,
            habAgendamentoQua,horaFimExpedienteQua,minutoFimExpedienteQua,
            habAgendamentoQui,horaFimExpedienteQui,minutoFimExpedienteQui,
            habAgendamentoSex,horaFimExpedienteSex,minutoFimExpedienteSex,
            habAgendamentoSab,horaFimExpedienteSab,minutoFimExpedienteSab);

      saidaSirene=agendamentoHorarioFimDeExpediente.verificaAtivacao();
```

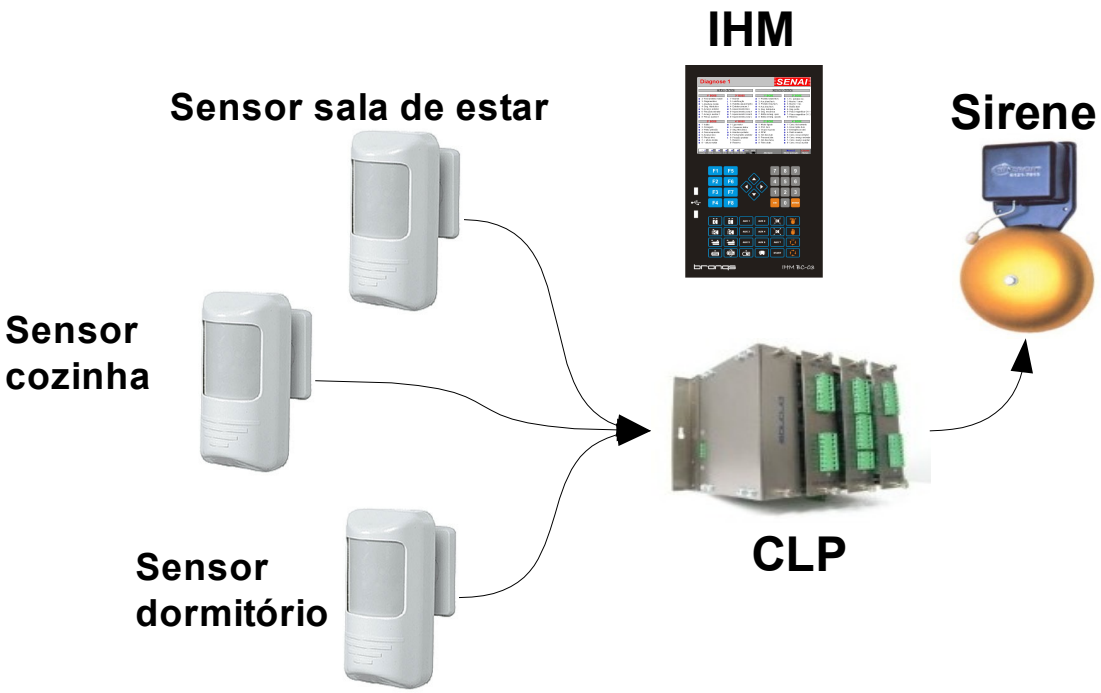
O agendador acima foi programado para acionar a sirene de segunda a quinta-feira às 17 horas e 15 minutos, enquanto na Sexta-Feira, a mesma será acionada às 16 horas e 30 minutos. Perceba que a sirene não será acionada aos sábados e domingos.

Declaração de campos na IHM

Esta aula não apresenta novos tipos de campos.

Aula 07 - Alarme residencial

Implementar um alarme residencial composto de uma sirene e três sensores de presença. A ativação e desativação do alarme deve ser feito através da digitação de uma senha na IHM. Considere que a IHM está instalada na sala de estar.



Entradas Digitais

- Sensor sala de estar
- Sensor cozinha
- Sensor dormitório

Saídas Digitais

- Sirene

Declarações e instruções do CLP

Instrução: "if"

Nesta aula mostramos que o Framework CLP branqs permite a utilização de instruções escritas na linguagem C. Este tipo de recurso estende significativamente a capacidade de processamento, podendo aproveitar os vários recursos que esta linguagem oferece.

Sintaxe

```
if (expressaoLogica)
{
      .
      . bloco 1
      .
}
else
{
      .
      . bloco 2
      .
}
```

Caso a "expressaoLogica" for verdadeira, as instruções existentes entre as primeiras chaves (bloco 1) serão executadas, caso contrário, as instruções existentes entre as segundas chaves (bloco 2) serão executadas. A instrução "else" é opcional. Dentro dos blocos delimitados pelas chaves, podem ser usadas quaisquer instruções disponíveis no framework ou na própria linguagem C.

Comentário

Caso tenha interesse em conhecer formas adicionais de trabalhar com instruções "if", basta seguir as definições existentes em qualquer tutorial para a linguagem ANSI-C.

Exemplo

DECLARA.CLP

```
////////////////////////////////////////////////////////////////////////
// Declaracao Campos e variaveis organizados por funcao
////////////////////////////////////////////////////////////////////////

      CAMPO_INT(contadorDePotesProduzidos)
      CAMPO_INT(numeroDePotesDesejados)
      CAMPO_CHAR(producaoDePotesAtingida)
```

LOGICA.CLP

```
////////////////////////////////////////////////////////////////////////
// Comando
////////////////////////////////////////////////////////////////////////

      if ( contadorDePotesProduzidos >= numeroDePotesDesejados)
            producaoDePotesAtingida=1;
      else
            producaoDePotesAtingida=0;
```

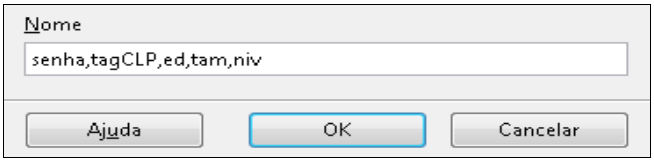
Se o valor do "contadorDePotesProduzidos" for maior ou igual ao valor do "numeroDePotesDesejados" então o sinal " producaoDePotesAtingida" receberá o estado lógico "Verdadeiro", caso contrário, receberá o estado lógico "Falso".

Declaração de campos na IHM

Campo: "Senha"

Os campos do tipo "senha" permitem a edição do conteúdo de variáveis do tipo inteiro, porém diferentemente do tipo "CAMPO_INT", um campo do tipo "Senha" apresenta asteriscos (*) no lugar dos números originalmente inseridos.

Sintaxe



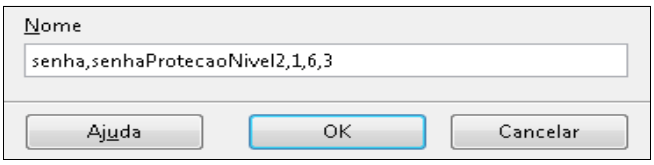
Onde:

senha: Tipo do campo
tagCLP: Nome da variável declarada no CLP
ed: Valor 0 (zero) ou 1 (um). 0 - Não permite edição, 1 - Permite edição
tam: Tamanho do campo em número de dígitos
niv: Nível de senha exigida para edição (Valor entre 0 e 3)

Comentário

Os campos do tipo “senha” utilizam variáveis declaradas como "CAMPO_INT" no CLP.

Exemplo

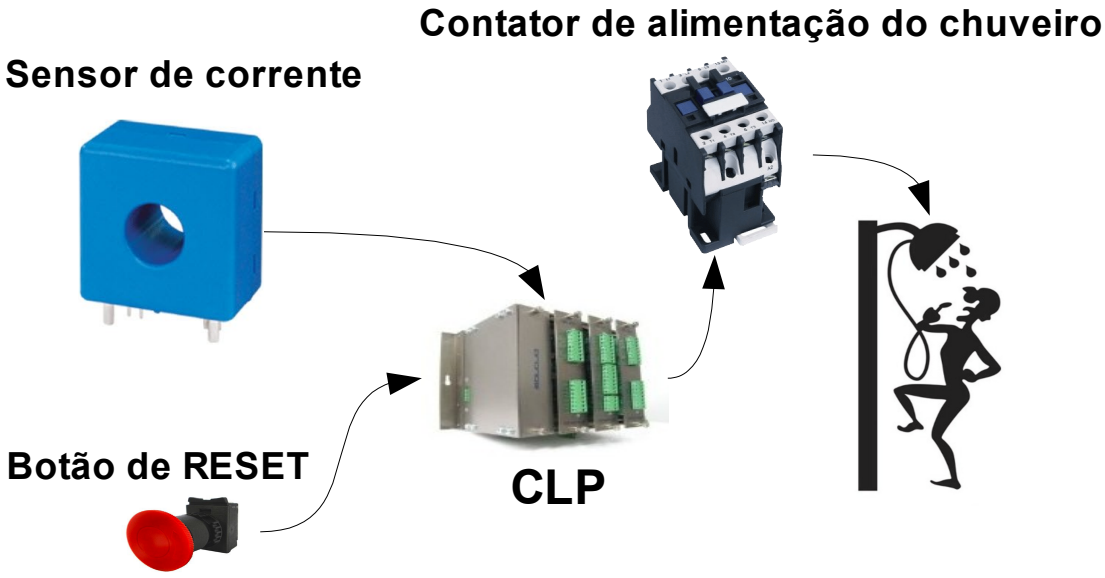


A declaração acima apresenta um campo que permitirá a edição da variável "senhaProtecaoNivel2" definida no CLP. Esse campo será editável, terá o tamanho de 6 caracteres e necessitará do nível de proteção 3 para poder ser editado. É importante lembrar que serão apresentados asteriscos (*) no lugar dos números digitados.

Aula 08 - Controle do tempo de chuveiro ligado

Enunciado do exercício

Implementar o software de um sistema que controla o tempo máximo de chuveiro ligado durante o banho de uma pessoa. O CLP monitora o tempo de chuveiro ligado através de um sensor de corrente elétrica. Caso a duração de chuveiro ligado ultrapasse o tempo limite programado na IHM, o CLP desliga a alimentação elétrica do chuveiro através de um contator. Para ligar o chuveiro novamente, é necessário pressionar o botão de RESET que fica localizado fora do banheiro.



Entradas Digitais

- Sensor de corrente
- Botão de reset

Saídas Digitais

- Contator de alimentação do chuveiro

Declarações e instruções do CLP

Instrução: "ENTAO_EXECUTA_BLOCO"

Permite executar um bloco de instruções caso o Estado Lógico Corrente (ELC) seja verdadeiro.

Sintaxe

```
ENTAO_EXECUTA_BLOCO
{
      .
      . bloco de instruções
      .
}
```

Comentário

Esta instrução funciona de forma muito parecida com a instrução "if", porém executa o bloco dependendo do estado do ELC.

Exemplo

DECLARA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Declaracao das saidas digitais
////////////////////////////////////////////////////////////////////////////

      CAMPO_SAIDA_DIGITAL(saidaAvancaCilindro)

////////////////////////////////////////////////////////////////////////////
//Declaracao Campos e variaveis organizados por funcao
////////////////////////////////////////////////////////////////////////////

      CAMPO_INT(contadorDeAvancos)
```

LOGICA.CLP

```
////////////////////////////////////////////////////////////////////////////
//Comando
////////////////////////////////////////////////////////////////////////////

      SEL         (saidaAvancaCilindro)
      SUBIDA
      ENTAO_EXECUTA_BLOCO
      {
            contadorDeAvancos = contadorDeAvancos + 1;
      }
```

Toda vez que for identificada a borda de subida no sinal "saidaAvancaCilindro", o "contadorDeAvancos" será incrementado em uma unidade.

Declaração de campos na IHM

Esta aula não apresenta novos tipos de campos.

Controle de Revisões

Revisão: 01
Data: 29/06/2014
Descrição:

- *Inserido o manual de referência das instruções utilizadas nas video-aulas.*

Revisão: 00
Data: 05/09/2013
Descrição:

- *Documento original criado para as vídeo aulas de introdução à programação de CLP.*