Introduction to Industrial Communication Networks, Integration Systems and Control (PCWorx Express) - Apostilled material of the practical class - 002: PCWorx Express–IEC 61131-Pro...

Technical Report · May 2018

DOI: 10.13140/RG.2.2.21946.70089

CITATIONS
0

2 authors:

Hermes Jose Loschi
University of Campinas
53 PUBLICATIONS **37** CITATIONS
SEE PROFILE

Yuzo Iano
University of Campinas
122 PUBLICATIONS **204** CITATIONS
SEE PROFILE

Aula Prática – 002 – 1ºSem 2018

Introduction to Industrial Communication Networks, Integration Systems and Control (PCWorx Express)

Prof. Dr. Yuzo Iano
Prof. MSc. Eng. Hermes José Loschi

Designação:	Aula Prática -002-1ºSem 2018
Tema:	PCWorx Express–IEC 61131-Programming
Revisão:	v00.002-2018
Professor Responsável:	Prof. Dr. Yuzo Iano

Termos gerais e condições de uso para documentação técnica

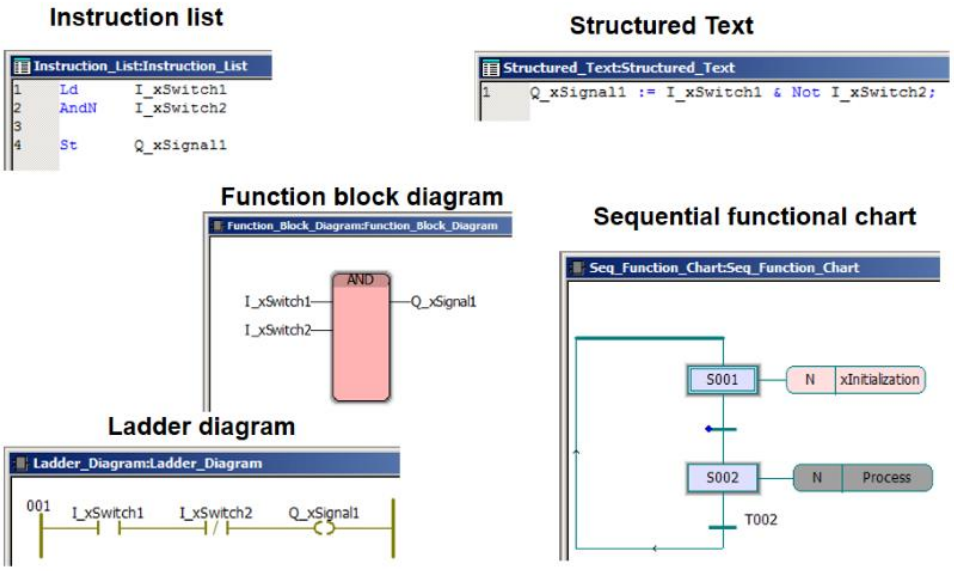
A publicação e atualização desta documentação técnica, não implicam necessariamente em qualquer obrigação por parte da Feec/Unicamp em fornecimento subsequente.

Os usuários desta documentação são responsáveis pela correta aplicação desta documentação, no âmbito dos cursos de pós-graduação da Feec/Unicamp, em conformidade com as normas e regulamentações vigentes.

Sumário

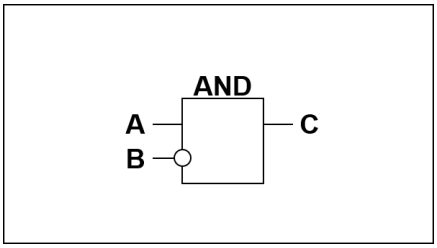
1. Linguagens de programação conforme IEC 61131

Este capítulo oferece uma visão geral dos cinco idiomas de acordo com a IEC 61131 e suas características. O PC WORX oferece os cinco idiomas descritos na IEC 61131 (conforme ilustrado na imagem abaixo):



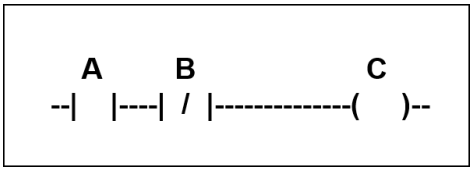
1.1. Function Block Diagram (FBD)

- Linguagem gráfica, amplamente utilizada na Europa;
- Elementos de programação na forma de blocos de função (FB);
- Os blocos de função podem ser "conectados" de forma semelhante a um diagrama de circuito;
- Usado em uma variedade de aplicações, responsáveis pelo fluxo de informações entre os componentes do sistema de controle.



1.2. Ladder Diagram (LD)

- Conjunto de símbolos de programação padronizados para sistemas de controle de relé;
- Com base no estilo de programação norte-americano, é semelhante ao padrão de diagramas de circuitos de desenho.



1.3. **Instruction List (IL)**

(Obs.: não está disponível para PCWorx Express)

- Modelo de programação *"Assembler"*, usando um acumulador;
- Por linha, um comando é permitido. Economizando um valor no acumulador.

```
LD      A
ANDN    B
ST      C
```

1.4. **Structured Text (ST)**

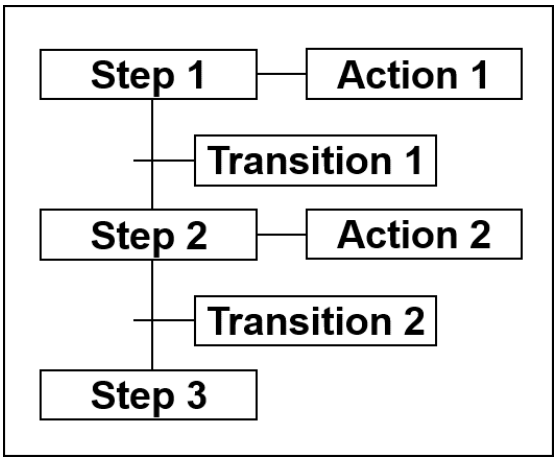
- Linguagem de alto nível, estruturada através de subprogramas;
- Instruções complexas e aninhadas.

```
C := A AND NOT B;
```

1.5. **Sequential Function Chart (SFC)**

(Obs.: não está disponível para PCWorx Express)

- Linguagem de programação gráfica, poderosa para descrever o comportamento de sequência de programas de controle;
- Usado para estruturar programas de controle;
- Linguagem de programação que permite diagnósticos rápidos;
- Elementos básicos: etapas com blocos de ação e transições
- Suporta sequências alternativas e paralelas;

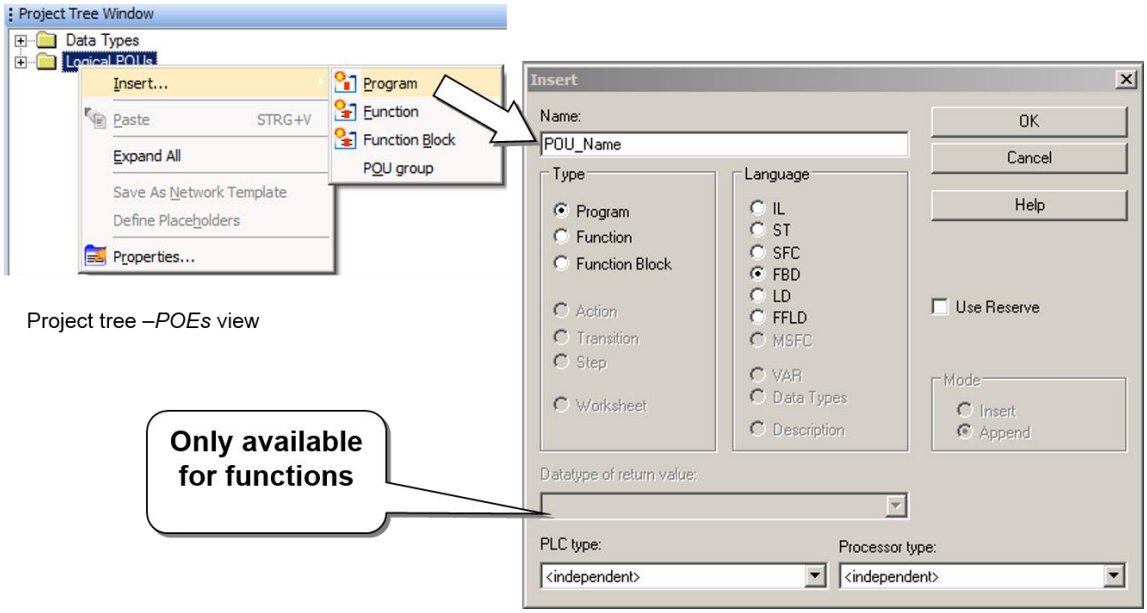


2. Program Organization Units

Este capítulo fornece uma análise sobre a estrutura de projeto no PC WORX. Isso inclui unidades de inserção e manuseio de programação, bem como a apresentação de funções (FU) e blocos de função (FB) conforme IEC 61131.

2.1. Inserting POU

Se o programa *"Main"*, que é inserido automaticamente, não está sendo utilizado, o primeiro passo para criar uma programação é inserir uma POU do tipo *"Program"*. Quando um elemento *Logical POUs* ou uma POU existente é selecionado, a inserção de uma nova POU pode ser feita através do menu de contexto ou da barra de menus (vide ilustração a seguir).



Project tree –*POEs* view

As opções de POUs estão disponíveis na seção *"Type"*. Por exemplo, nem todos as linguagens de programação são selecionáveis para cada tipo de POU disponível (Ex.: as funções (FU) não suportam SFC).

A opção "*Datatype of return value"* só está ativa para funções (FU). As funções possuem apenas um parâmetro de saída. O tipo de dados deste parâmetro é determinado através desta seleção.

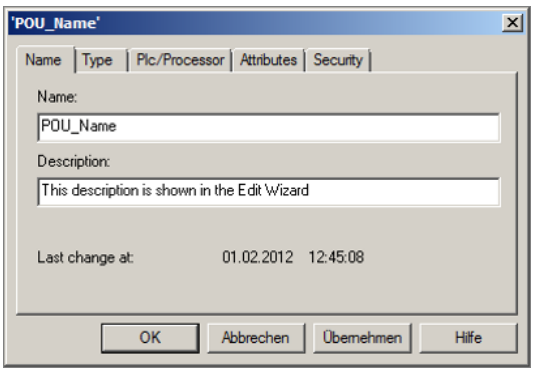
A seleção <*independent*> aplicada nas opções *"PLC type"* e *"Processor type"* devem ser apenas adaptadas se funções (FU) e blocos de função (FB) específicos forem usados. Se esta opção for usada, esta POU só pode ser operada em um CLP ou processador do tipo determinado, para esta condição em especifico.

A posição de uma POU na árvore do projeto não afeta a ordem de execução. A chamada de um programa sob a forma de uma instância de programa através de uma tarefa, ou a chamada de funções (FU) e blocos de função (FB) em planilhas de código é quem determina a sequência do programa (fluxo de processamento).

2.2. **POU Properties**

Com a exceção do idioma, as propriedades POU podem ser posteriormente adaptadas na caixa de diálogo de propriedades da POU (menu contexto). Obs. para acessar a caixa de diálogo de propriedades as *worksheets* devem estar fechadas.

Uma mudança de tipo é possível, mas, dependendo da frequência de uso da POU, isso significa um maior esforço de adaptação para o projeto. Como um padrão, o recurso (estrutura de hardware) está configurado para que a memória esteja disponível para cada POU e uma reserva individual não é necessária.



Sobre a guia *"Attributes"*.

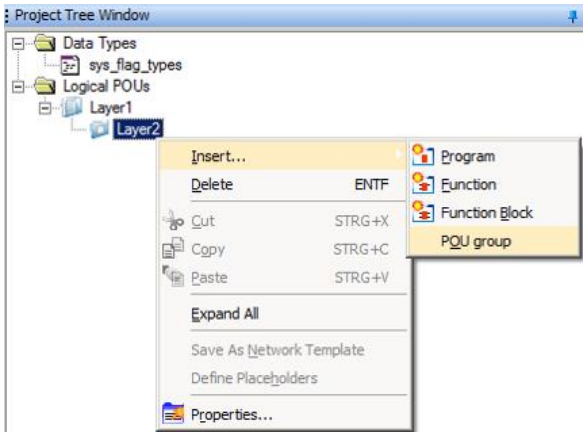
- O atributo *"Read Only"* é utilizado para funções multiusuários, mas também pode ser usado como proteção para alterações involuntárias.

Sobre a guia *"Security"* (PC Worx Professional).

- As configurações de segurança para uma POU permitem a proteção da para programação. A proteção só é ativada depois de inserir uma senha através do menu *File → Enter password*.

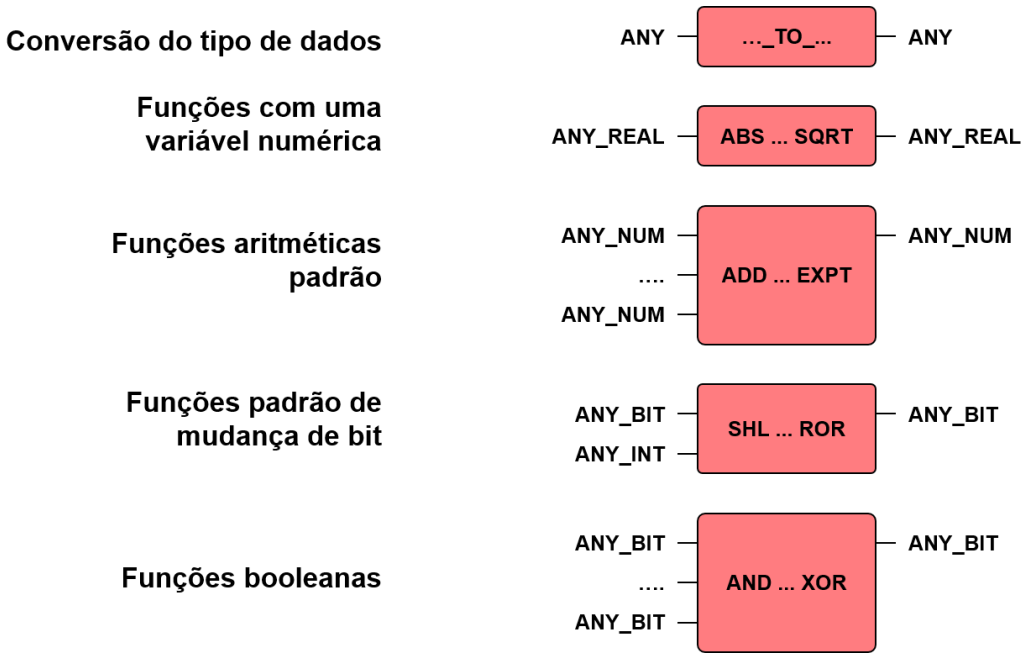
2.3. **POU Groups**

Os *POU groups* (PC Worx Professional) não afetam as atribuições de programação. Esses grupos servem como um meio para uma melhor organização e podem ser criados em uma ordem tão complexa quanto você quiser. Os *POU groups* são sempre listados como os elementos mais altos na pasta *Logical POU* e aparecem em ordem alfabética.

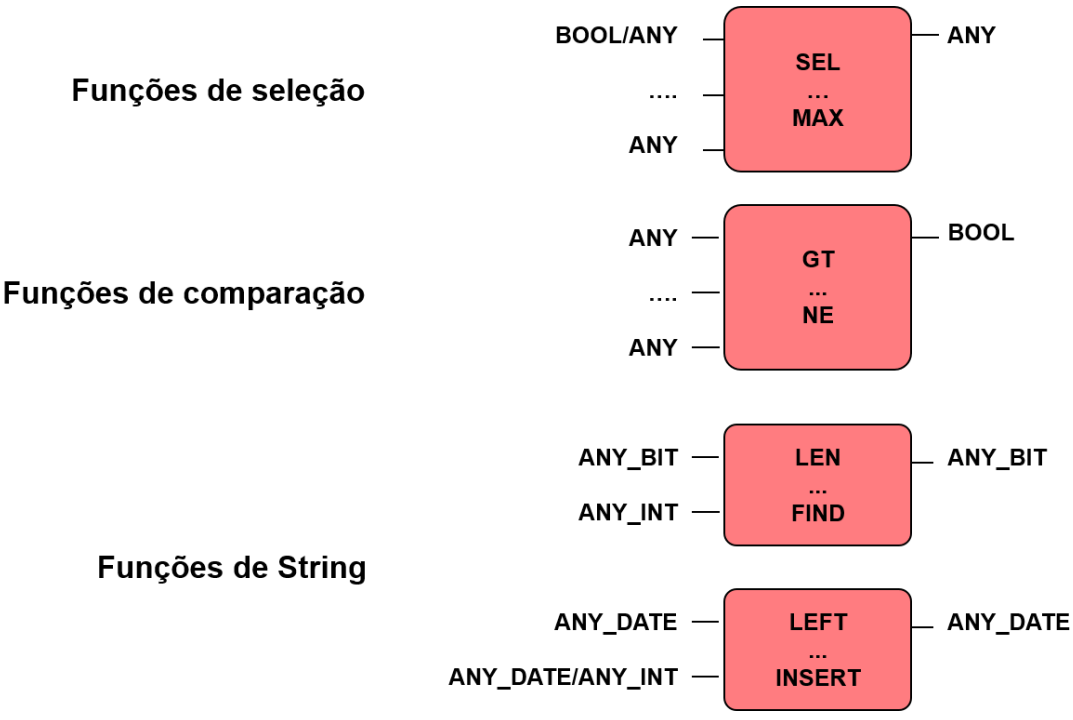


2.4. Funções presentes na IEC61131:

Após a instalação do PC WORX, umas variedades de FU estão disponíveis. Estas FU podem ser visualizadas através do *edit wizard, IEC Programming Workspace,* definidos pela IEC 61131. Devido à variedade de tipos de dados, especialmente o grupo de funções para conversão de tipo de dados é muito extensa.

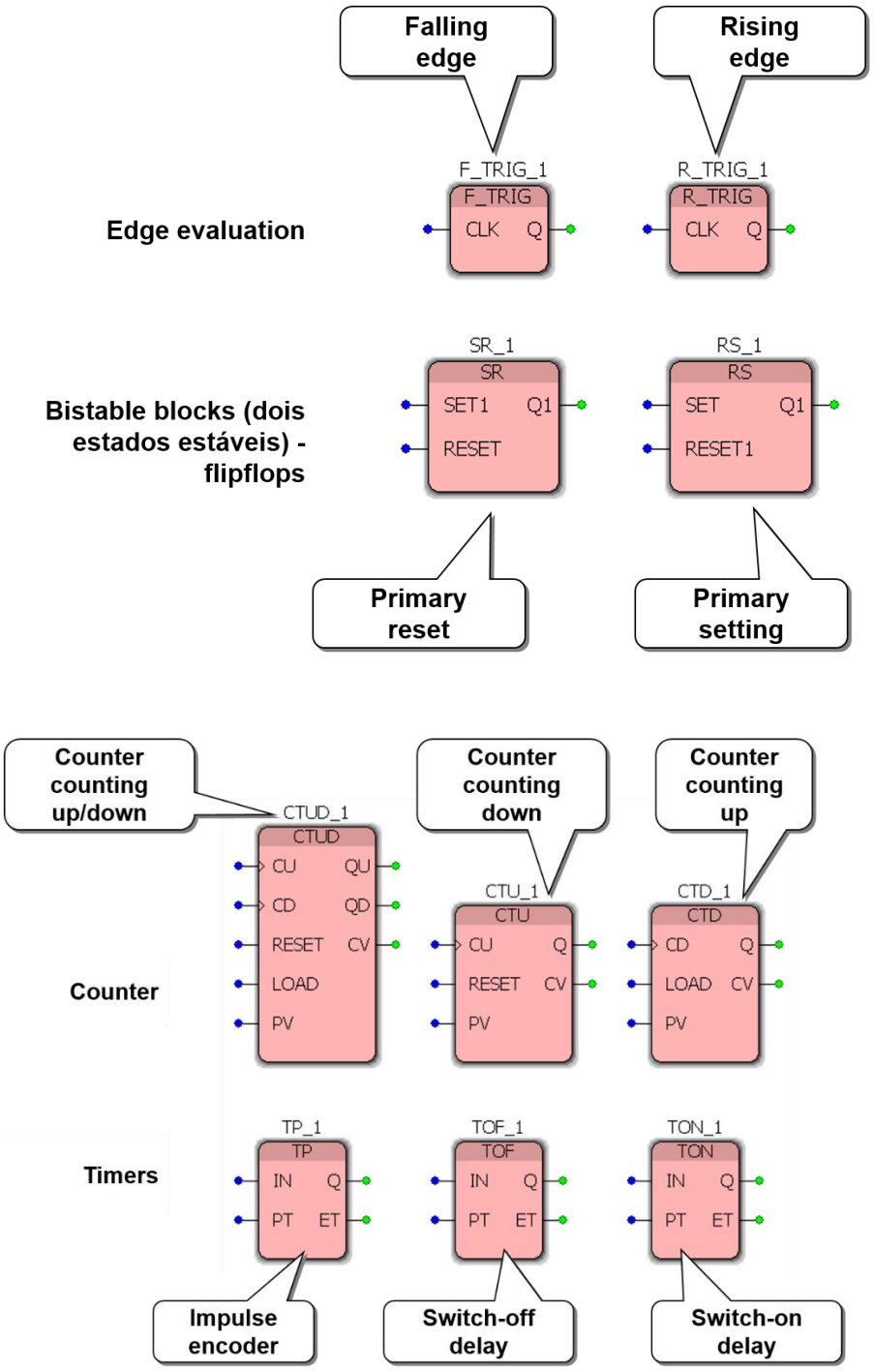


Detalhes sobre a conexão e sobre os blocos de função (FB) podem ser obtidos a partir das ajudas HTML individuais para cada bloco no PC WORX.



2.5. **Blocos de Funções presentes na IEC61131:**

A IEC 61131 define FB, que podem ser divididos em quatro grupos, conforme mostrado abaixo.

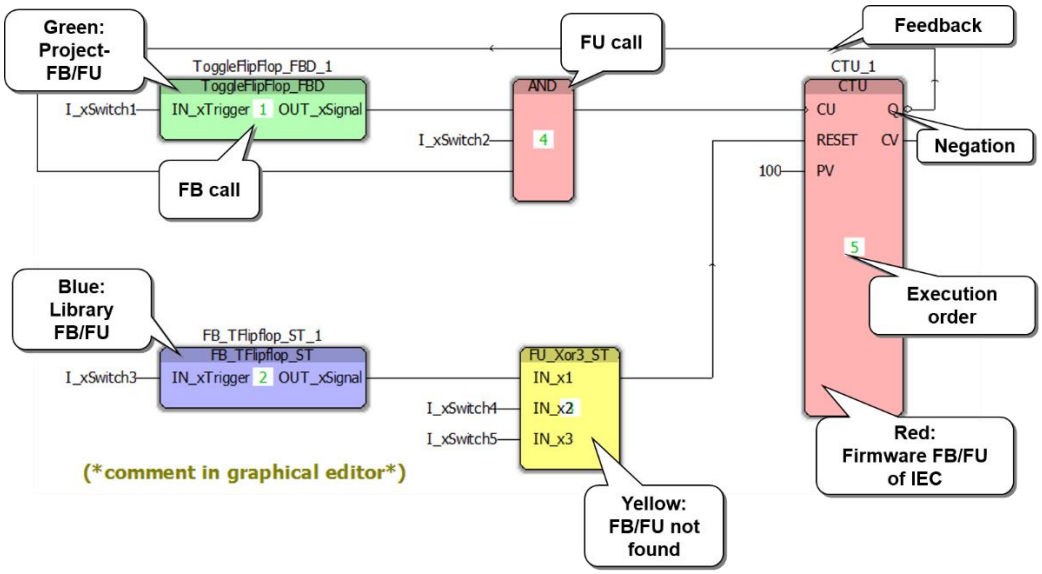


2.6. **FBD – Function Block Diagram**

Este capítulo descreve a programação no diagrama de blocos de função. Isso inclui inserir funções (FU) e blocos de função (FB), bem como acessar variáveis de diferentes validades.

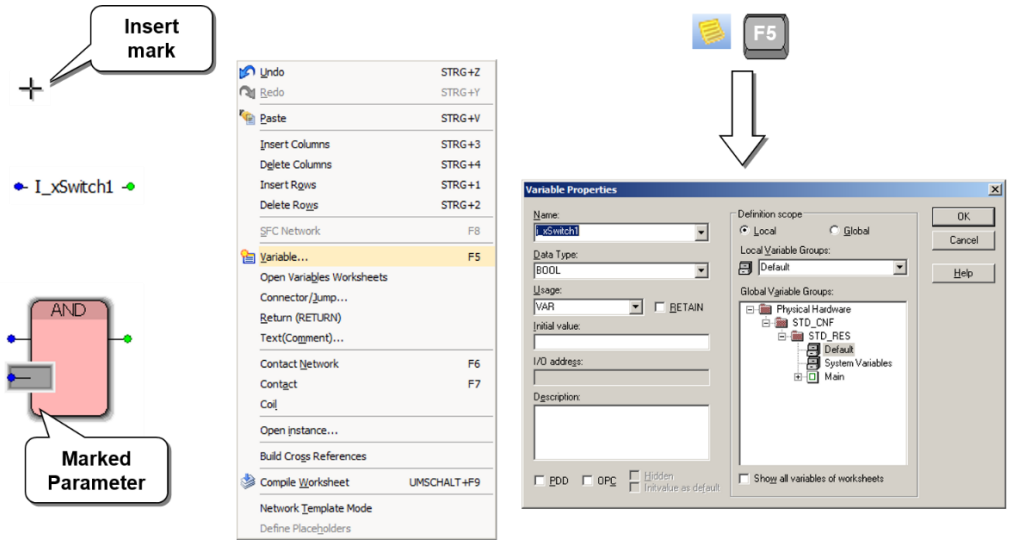
FBD (*Function Block Diagram*) é uma linguagem gráfica de programação, amplamente utilizado nas controladoras atuais. Elementos de programação são apresentados em forma de blocos de funções (FB).

Os blocos de função podem ser "ligados" da mesma forma como em um diagrama de circuitos elétricos. É utilizado em uma grande variedade de aplicações responsáveis pelo fluxo de informações entre os componentes do sistema de controle. Na imagem abaixo, vemos um exemplo de programação em FBD e a respectiva representação das cores:

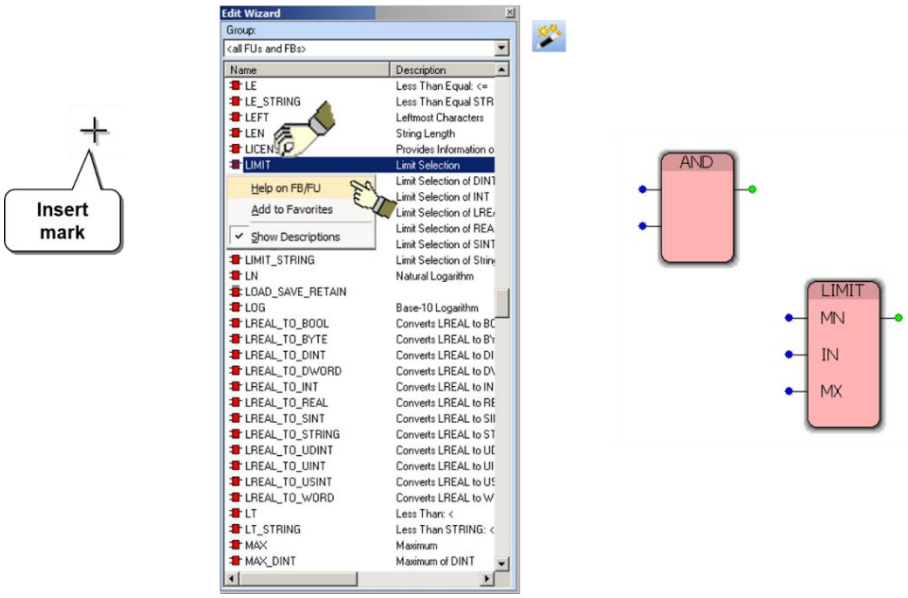


No FBD ilustrado acima, a inserção de uma variável é realizada através da caixa de diálogo da variável. Esta caixa de diálogo oferece acesso a variáveis locais e globais. No editor FBD, o primeiro passo é definir a marca de inserção ou um ponto de conexão de uma FU ou FB, que deve ser marcado antes que a caixa de diálogo possa ser exibida.

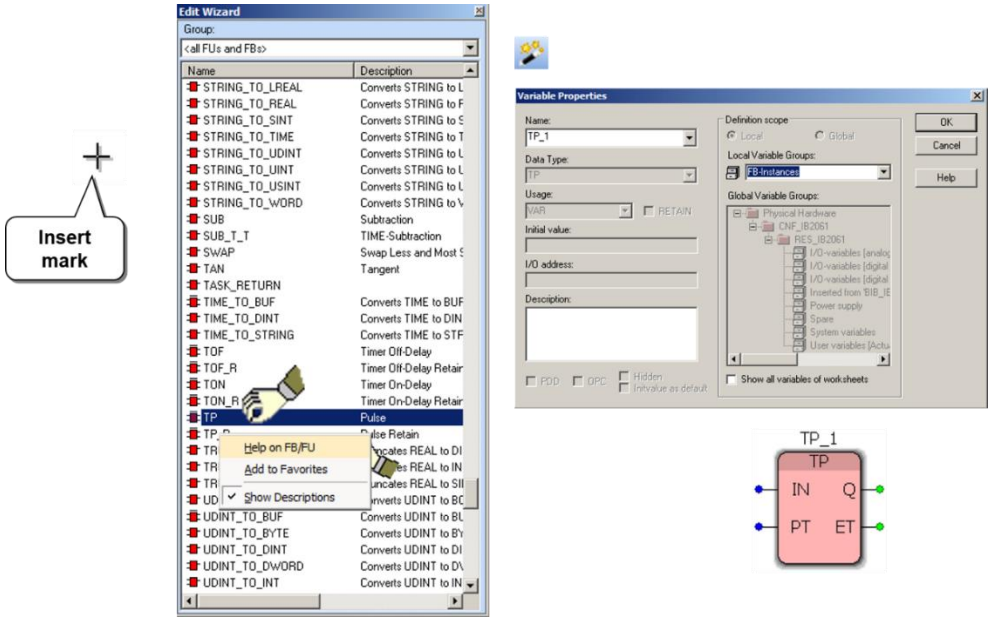
A caixa de diálogo da variável pode ser ativada através do menu de contexto, ou através da tecla F5 (configuração padrão).



Para acessar uma função através do *Edit Wizard*, primeiro a marca de inserção deve ser definida na *worksheet*. Para inserir uma marca, faça um duplo clique.



A inserção de FB é feita da mesma maneira que as FU. No entanto, antes de inserir um FB, sua instância deve ser declarada. O PC WORX propõe o nome da FB, como um incremento vinculado por um sublinhado. Este nome pode ser selecionado individualmente, isto é, relacionado à função, pelo usuário.



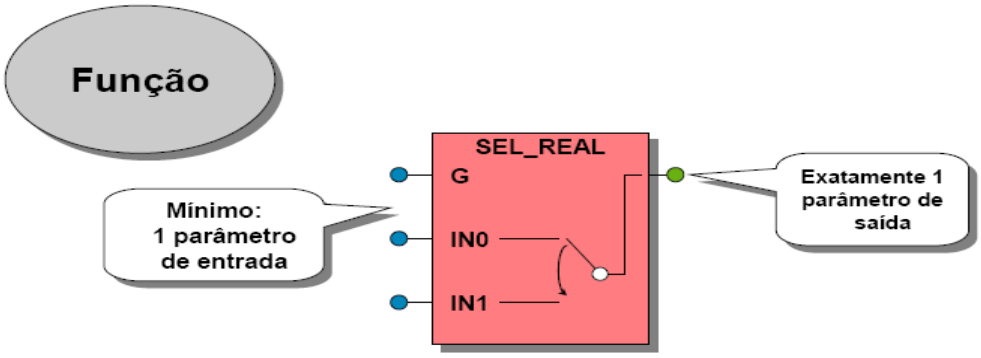
O exemplo acima mostra a caixa de diálogo de variável de um projeto, no qual um grupo de variáveis locais para instâncias FB já foi criado. Caso contrário, apenas o grupo “Default” estaria disponível.

3. Programação com FB e FU

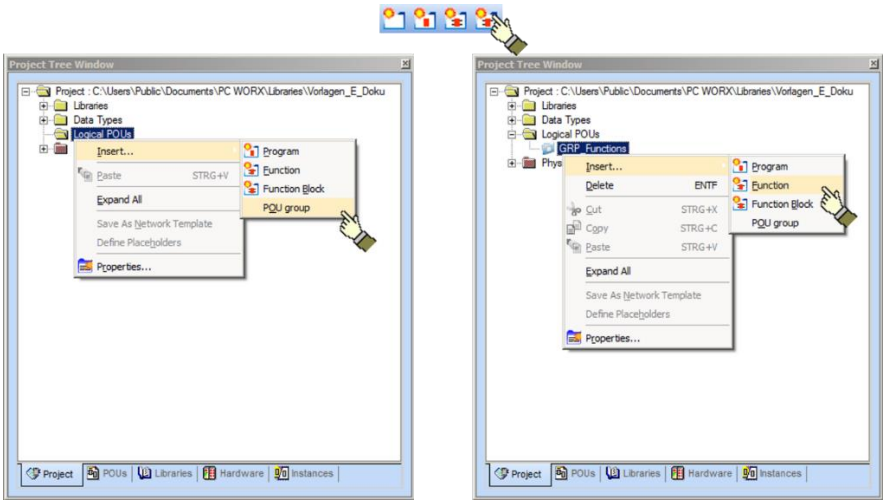
Este capítulo descreve como criar FB e FU, o que deve ser considerado e como utilizá-los em projetos únicos e disponibilizá-los para outros projetos.

3.1. Criando Funções

As funções (FU) podem ter uma ou mais entradas e devem ter uma e somente uma saída. Elas não podem conter variáveis globais e seu processamento não pode depender da influência de ciclos anteriores.



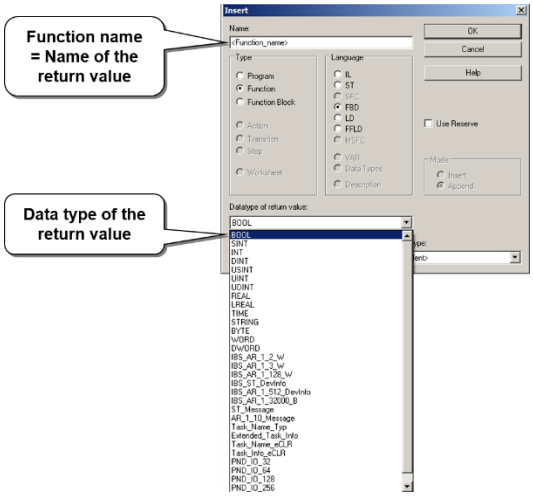
A inserção de uma FU na árvore do projeto é possível quando a pasta *Logical POUs* ou *POU group* são marcados.



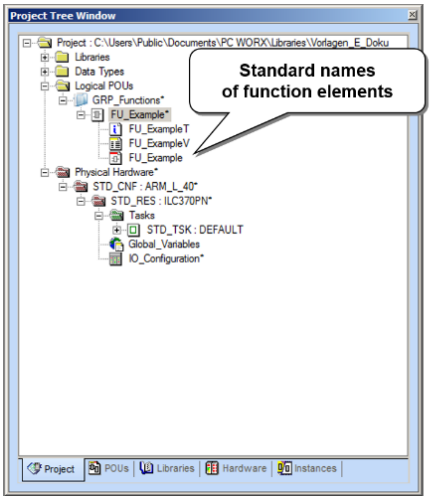
Os seguintes parâmetros devem ser configurados para inserir uma função:

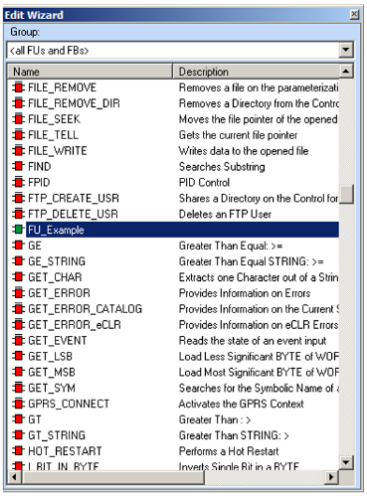
- **Nome da função**
  - Mais tarde, o nome da FU será listado para seleção da mesma no *edit wizard*. Além disso, este nome será usado como um nome de valor de retorno para a programação interna do bloco. O nome deve corresponder às convenções para nomes de variáveis.

- **Valor de retorno da função**
  - O tipo de valor de retorno da função é determinado através do tipo de dados do valor de retorno.



Depois que a função foi inserida na *Project tree*, ela é listada com o nome escolhido em três elementos básicos (vide figura abaixo). Obs.: A opção de visualização desta estrutura da arvore de projeto, é disponível apenas para a versão Professional.

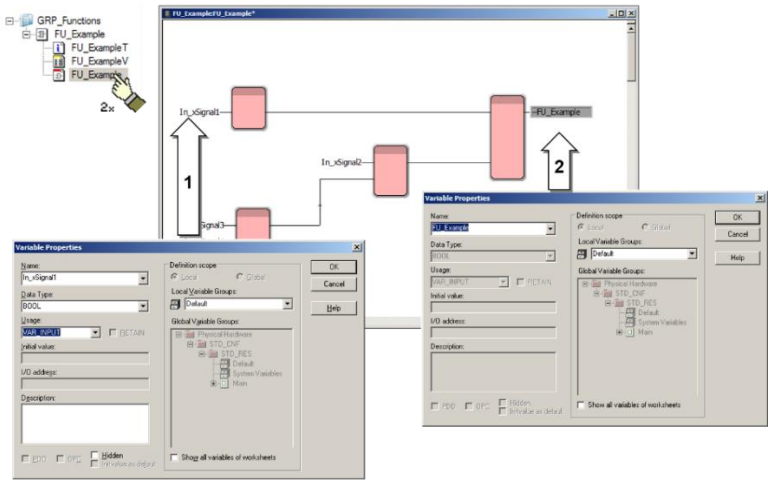




- <Nome da função> T para a folha de comentários;
- <Nome da função> V para a tabela variável;
- <Nome da função> para a primeira planilha do código;

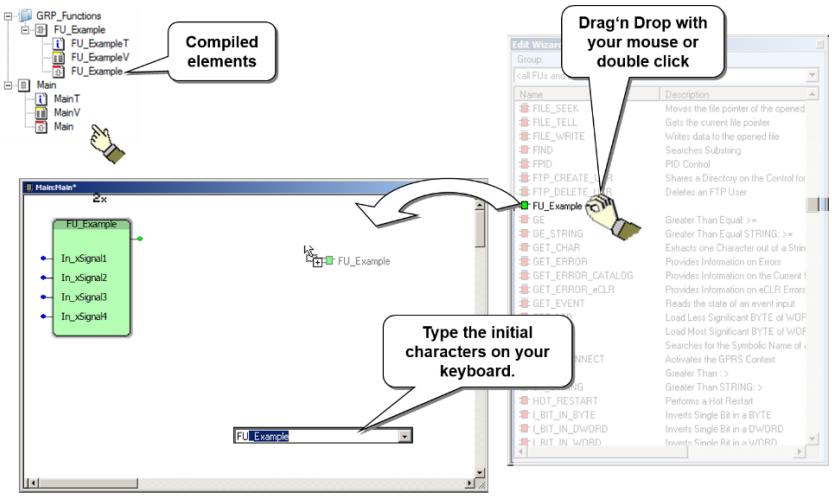
Além disso, no *Edit wizard*, a função aparece no grupo *<all FUs e FBs>* com o nome do projeto atual. Embora sejam FU, as FU criadas pelo usuário não entram no grupo *<Funções>*. Neste ponto, a FU ainda não pode ser chamada. Os asteriscos nas planilhas de código indicam que esses elementos ainda não foram compilados.

A função em si é editada em sua planilha (s) de acordo com o procedimento padrão para programação. Como mostrado abaixo, pelo menos (1) um parâmetro de entrada deve ser declarado. Além disso, o (2) valor de retorno (nome = nome da função) deve ser usado para fazer o valor computado por meio do algoritmo de função disponível para o projeto.



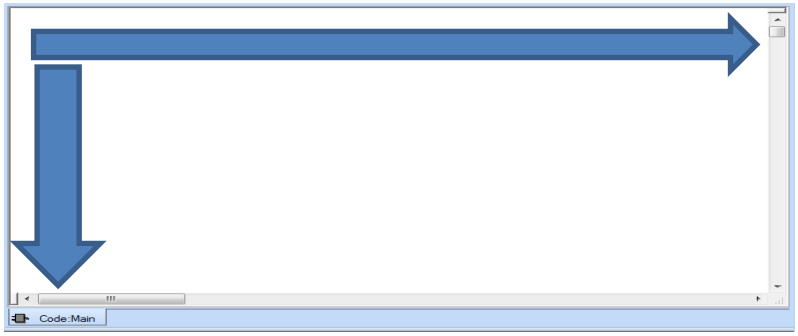
Após a compilação bem-sucedida da tabela de variáveis, uma função pode ser chamada para uso em outra *POU*.

Quando a FU é inserida, as variáveis disponíveis e os tipos de blocos parametrizáveis estão listados em uma caixa de combinação depois de inserir um caractere válido (vide imagem abaixo).



3.2. **Fluxo de Processamento**

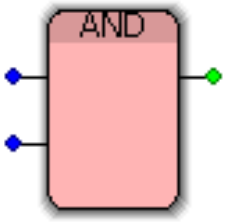
Na linguagem de programação de Diagrama de Blocos a varredura do programa é da ESQUERDA para a DIREITA e de CIMA PARA BAIXO. Sempre nesta ordem
É extremamente importante atentar esta questão para garantir a correta execução do programa.



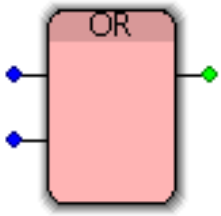
3.3. **Exercícios**

Testes e analise cada um dos blocos listados a seguir.

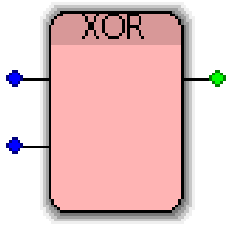
- Lógicos:



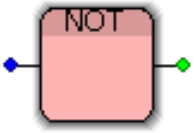
IN 1	IN 2	OUT
0	0	0
0	1	0
1	0	0
1	1	1



IN 1	IN 2	OUT
0	0	0
0	1	1
1	0	1
1	1	1

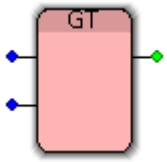


IN 1	IN 2	OUT
0	0	0
0	1	1
1	0	1
1	1	0

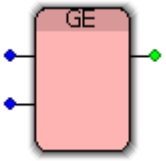


IN	OUT
0	1
1	0

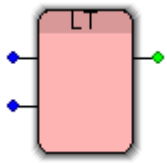
- Comparação:



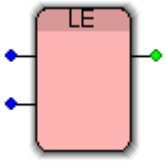
Entradas	OUT
IN1>IN2	1
OUTROS	0



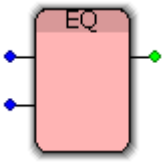
Entradas	OUT
IN1>=IN2	1
OUTROS	0



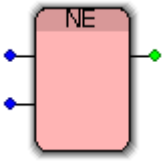
Entradas	OUT
IN1<IN2	1
OUTROS	0



Entradas	OUT
IN1<=IN2	1
OUTROS	0



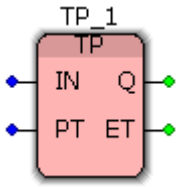
Entradas	OUT
IN1=IN2	1
OUTROS	0



Entradas	OUT
IN1≠IN2	1
OUTROS	0

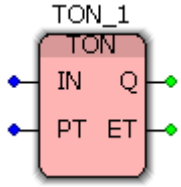
- Temporizadores:

- TP: Pulso de largura definida:



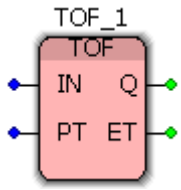
Parameter	Data types	Description
IN	BOOL	If a rising edge is detected, a pulse is created.
PT	TIME	preset time interval for the pulse
Q	BOOL	TRUE if IN = TRUE and ET < PT FALSE if IN = FALSE and ET >= PT
ET	TIME	elapsed time interval

- TON: Espera para ativar a saída:



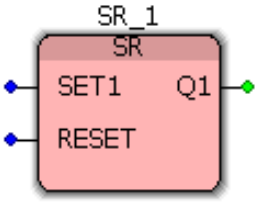
Parameter	Data types	Description
IN	BOOL	If a falling edge is detected, the off-delay timing is started.
PT	TIME	preset time interval for the delay
Q	BOOL	TRUE if IN = TRUE and ET < PT FALSE if IN = FALSE and ET >= PT
ET	TIME	elapsed time interval

- TOF: Espera para desativar a saída:

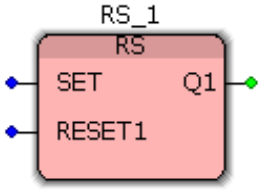


Parameter	Data types	Description
IN	BOOL	If a rising edge is detected, the on-delay timing is started.
PT	TIME	preset time interval for the delay
Q	BOOL	TRUE if IN = TRUE and ET >= PT FALSE if IN = FALSE or ET < PT
ET	TIME	elapsed time interval

- Set-Reset:

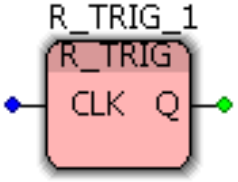


Parameter	Data types	Description
SET1	BOOL	IF TRUE Q1 is set dominant
RESET	BOOL	If TRUE Q1 is reset
Q1	BOOL	output

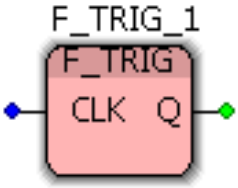


Parameter	Data types	Description
SET	BOOL	IF TRUE Q1 is set
RESET1	BOOL	IF TRUE Q1 is reset dominant
Q1	BOOL	output value

- Detecção de Bordas:



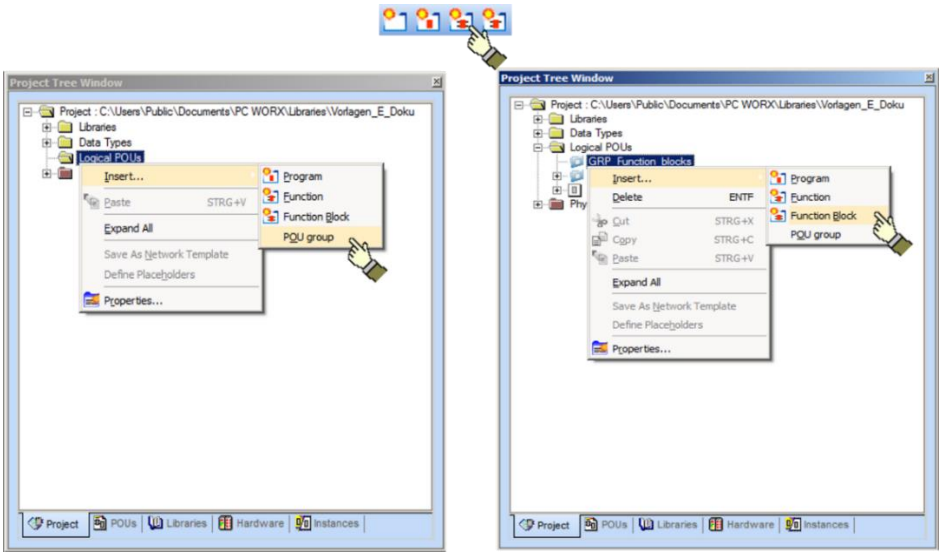
Parameter	Data types	Description
CLK	BOOL	detects a rising edge
Q	BOOL	If a rising edge is detected, Q changes from FALSE to TRUE



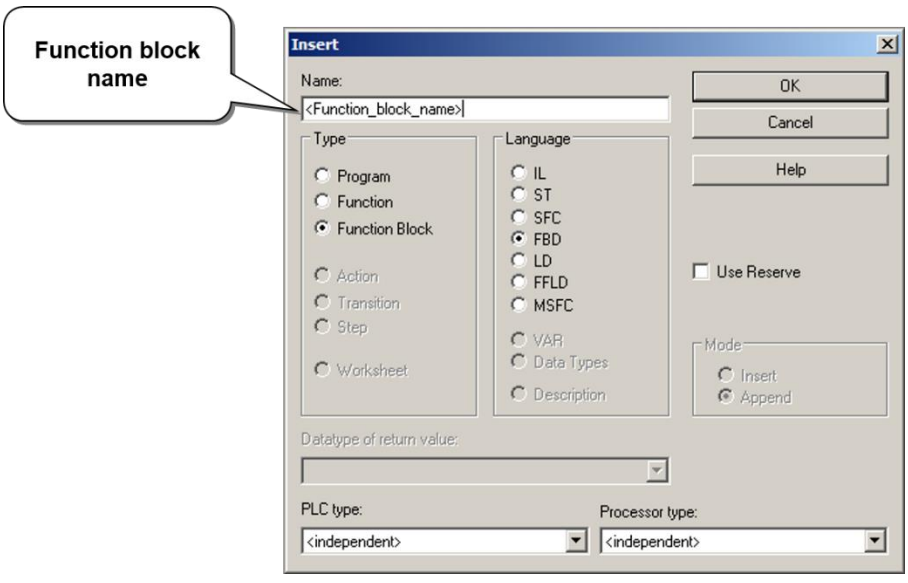
Parameter	Data types	Description
CLK	BOOL	detects a falling edge
Q	BOOL	If a falling edge is detected, Q changes from FALSE to TRUE

3.4. Criando Blocos de Funções

A inserção de um FB na árvore do projeto é possível quando a pasta *Logical POUs* ou o *POU groups* são selecionadas. Dependendo do tamanho do projeto, pode ser sensato organizar FB em *POU groups*. Esses FB também podem ser adicionados aos *POU groups* mais tarde.



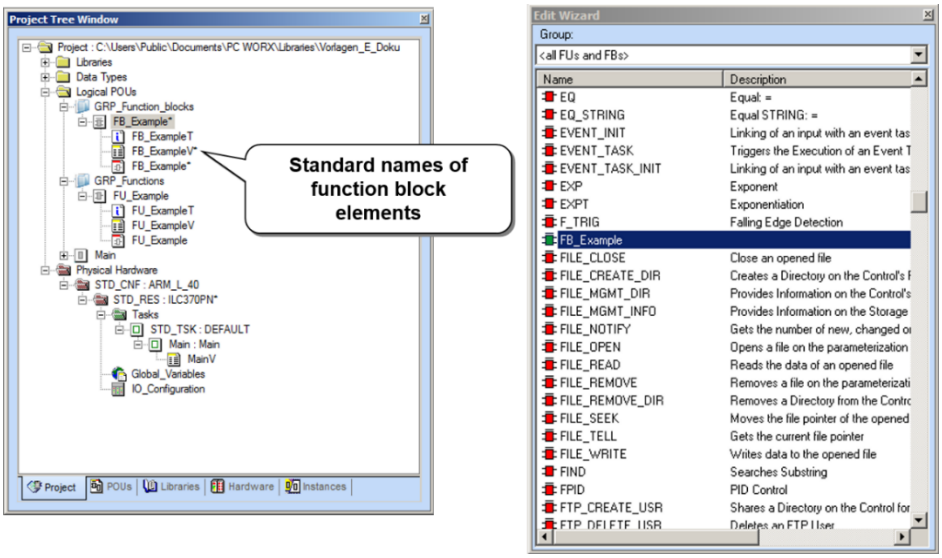
Para a inserção do FB, existe uma maior variedade de idiomas que a inserção de FU. O valor do tipo de dados de retorno não precisa ser especificado. Mais tarde, o nome do FB será listado para seleção no assistente de edição.



Depois que o FB foi inserido na árvore do projeto, ele é listado com o nome escolhido e com os três elementos básicos. Obs.: A opção de visualização desta estrutura da arvore de projeto, é disponível apenas para a versão Professional:

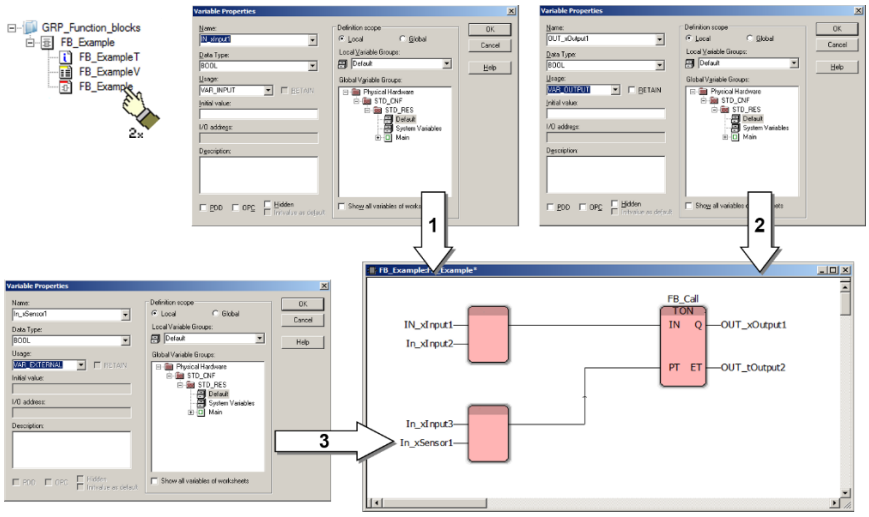
- <Nome da função> T para a folha de comentários;
- <Nome da função> V para a tabela variável;
- <Nome da função> para a primeira planilha do código;

Além disso, no assistente de edição, o FB aparece no grupo <all FUs e FBs> e no grupo com o nome do projeto atual.



Embora sejam FB, para FB criados pelo usuário não há entrada no grupo <Functions blocks>.

Neste ponto, o FB ainda não pode ser chamado. Os asteriscos na tabela de variáveis e na planilha de códigos indicam que esses elementos ainda não foram compilados.

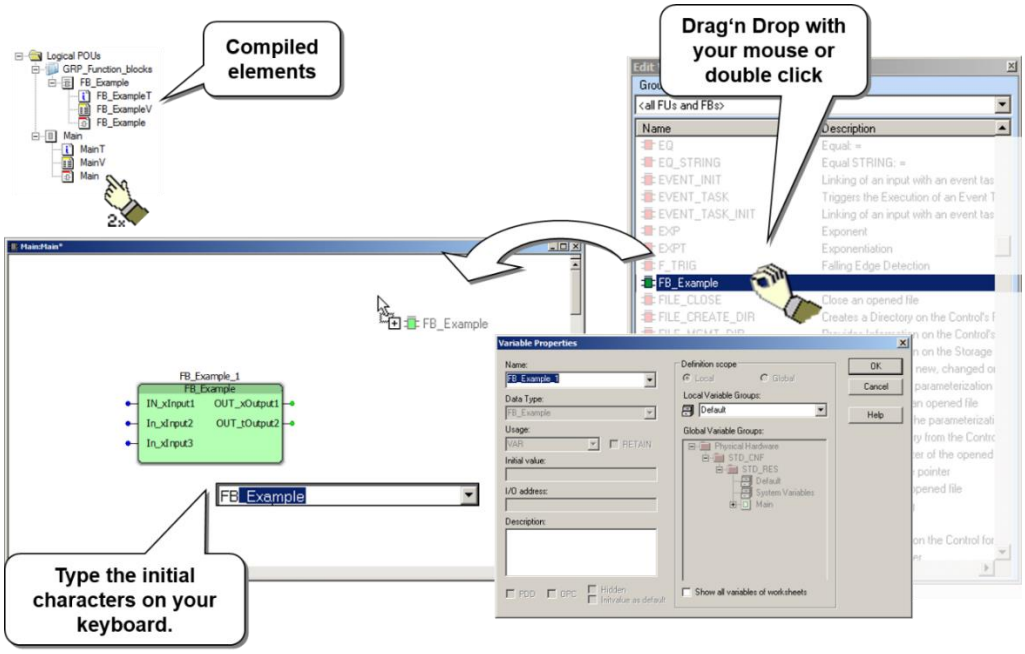


O próprio FB é editado em sua planilha (s) de acordo com o procedimento padrão para a programação. Ao contrário das FU, existe um grau significativamente maior de liberdade em relação ao design. Os parâmetros Input (1) e output (2) podem ser declarados em qualquer número e os FB também permitem o acesso a variáveis globais (3).

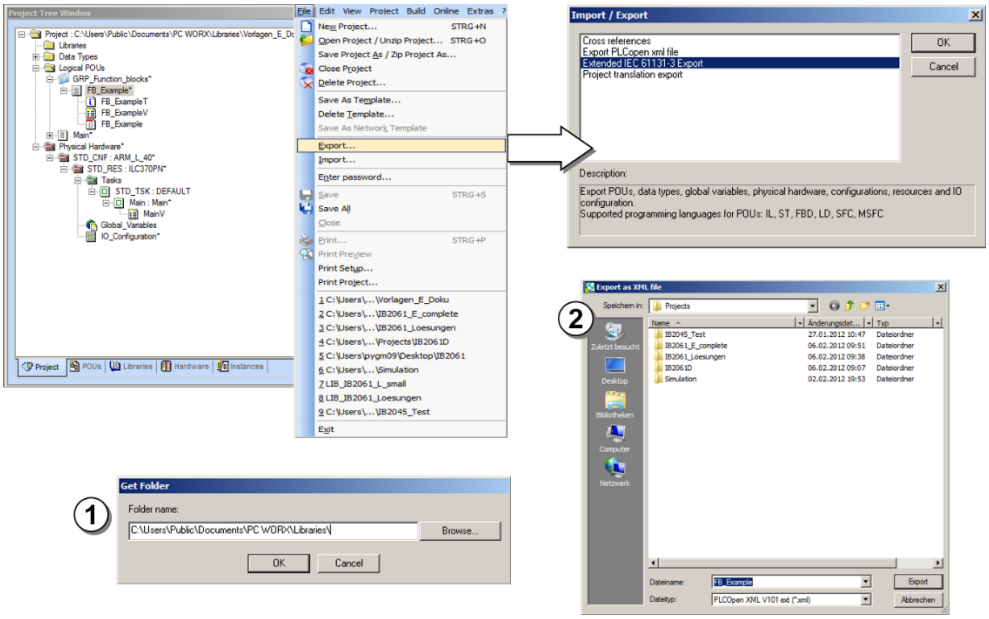
Lembre-se de que o nome do FB não deve ser usado como um nome de variável durante a programação interna do FB.

Após a compilação bem-sucedida da tabela de variáveis, o FB pode ser chamado para uso em outra *POU*.

Quando a FB é inserida, as variáveis disponíveis e os tipos de blocos parametrizáveis estão listados em uma caixa de combinação depois de inserir um caractere válido (vide imagem abaixo).

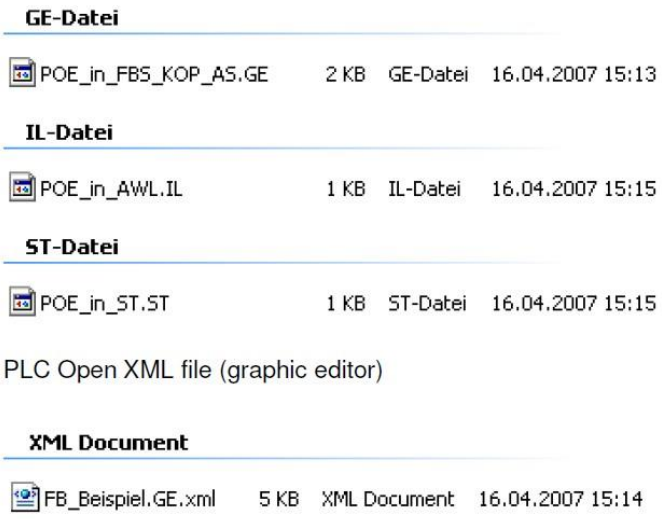


3.5. **Exportar e Importar Blocos de Funções**

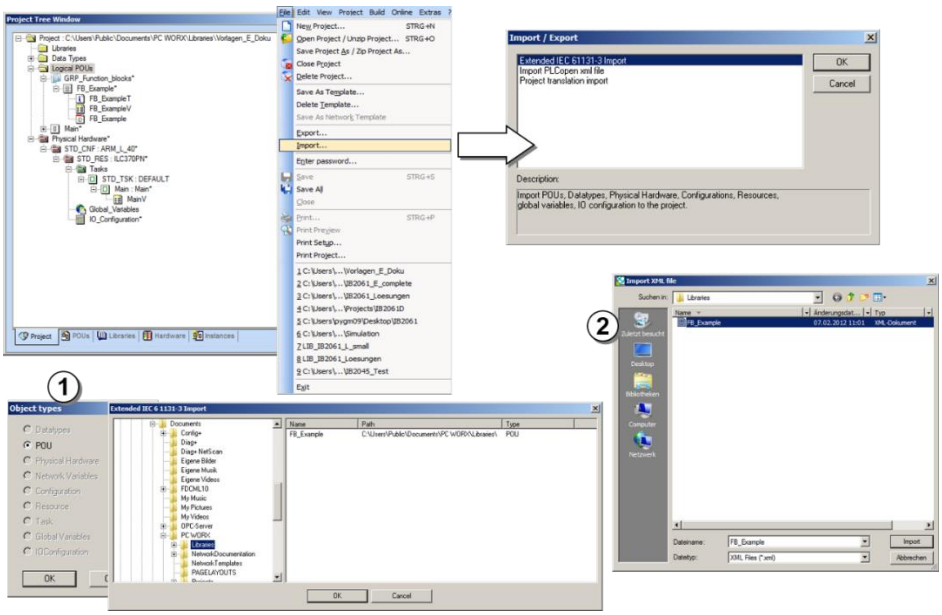


Um FB deve ser marcado na árvore do projeto antes de poder ser exportado, salvo como um arquivo ou disponibilizado para outros projetos. No menu Arquivo, selecione Exportar. Você pode então selecionar entre salvar a POU como (1) arquivo IEC 61131-3 ou como (2) arquivo PLC Open XML.

Arquivos IEC61131-3:



Para importar um bloco de (1) um arquivo IEC 61131-3 ou um (2) arquivo CLP Open XML, a pasta *Logical POUs* ou um de seus elementos deve ser selecionado (vide imagem abaixo).



4. IL *Instruction List* (lista de instruções)

Este capítulo mostra como programar em lista de instruções no PC WORX. Além de instruções simples, como operadores e funções, é descrito como os FB podem ser chamados e como a programação executada pode ser implementada por salto e retorno. **Obs.: não está disponível para PCWorx Express.**

4.1. Elemento de idioma da lista de instruções

O acesso aos elementos de idioma disponíveis em IL é principalmente suportado pelo assistente de edição. Além dos grupos de *Functions* e *Function block*, o grupo *Operators* está disponível na IL.

Assignments and operators

```
Ld    I_xInput1
AndN  I_xInput2
Not
St    Q_xOutput1
```

FU calls and arguments

```
Ld    iMin
Limit iBas, iMax
St    iScaled
```

Conditional processing

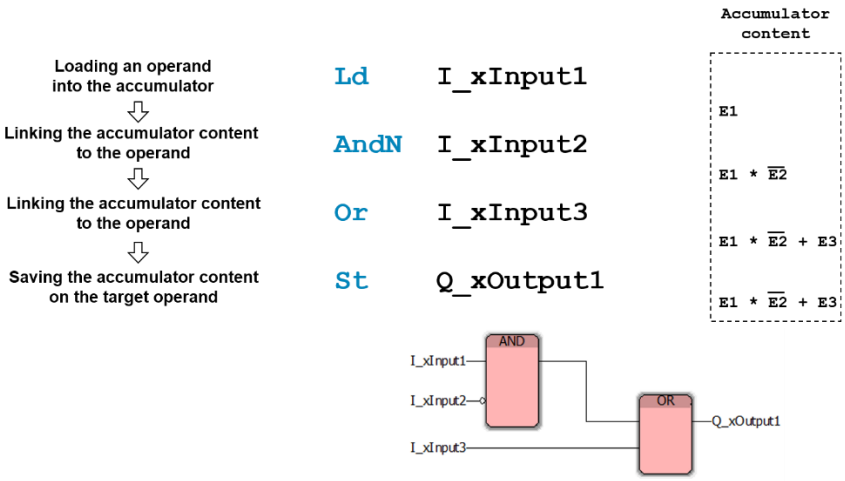
```
Ld    I_xInput1
JmpC  Jump
...
Ret
Jump:
Ld    I_xInput2
St    Q_xOutput1
```

FB calls and arguments

```
Ld    xPulse
St    CTU_CountUp.CU
Ld    iMax
St    CTU_CountUp.PV
Ld    xStop
St    CTU_CountUp.RESET

Cal   CTU_CountUp

Ld    CTU_CountUp.QV
St    iDisplay
Ld    CTU_CountUp.Q
St    xFull
```

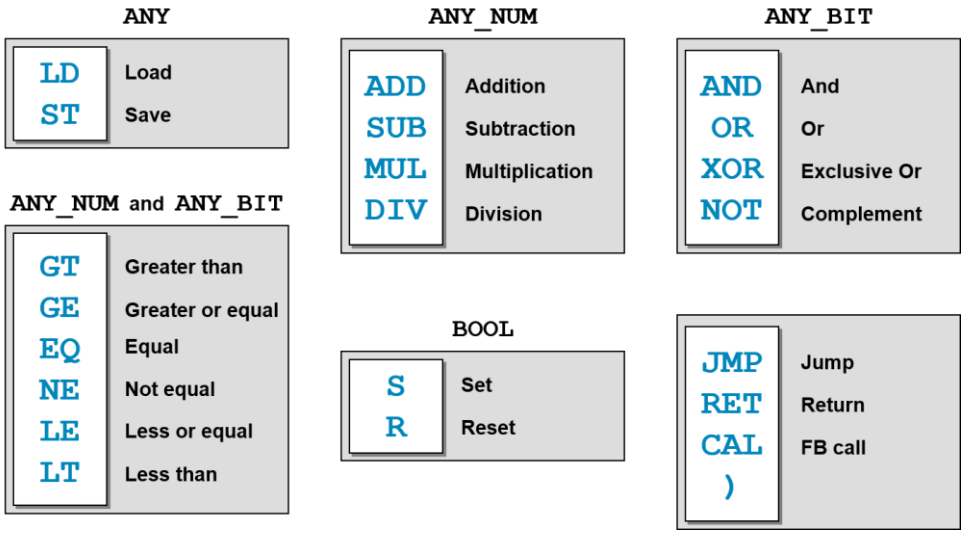
4.2. Atribuição e Operadores

A IL no PC WORX usa um acumulador, um par uniforme de carga e salva um comando para todos os tipos de dados.



4.3. Operadores na Lista de Instrução

O diagrama a seguir fornece uma visão geral dos tipos de dados e dos grupos de tipos de dados aos quais os operadores podem ser usados. A formatação de uma instrução na IL é a mesma para todos.

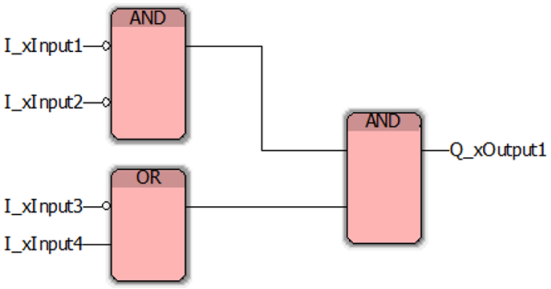


As exceções são o complemento NOT, a extremidade do bloco de função RETURN e o suporte de fechamento, que pode ser usado sem um operando na mesma linha.

4.4. Modificando os Operadores

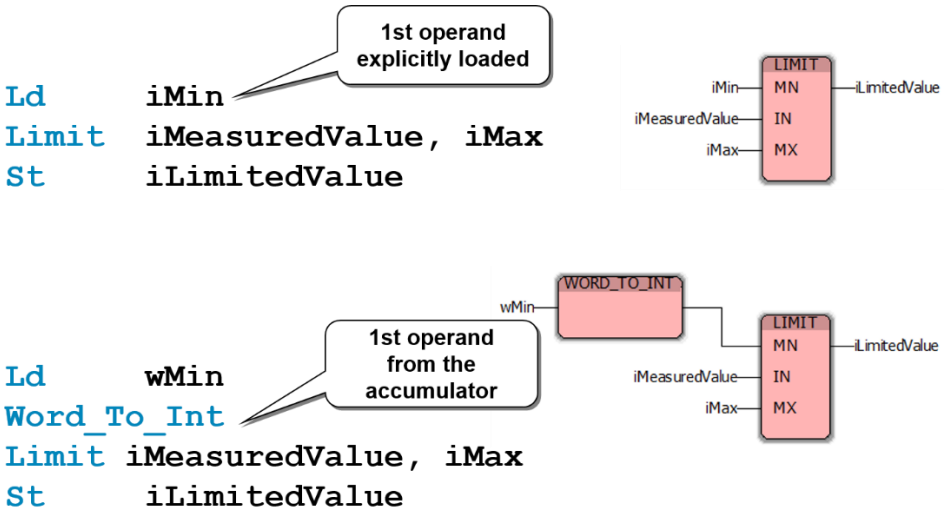
Por meio do modificador N, os operadores para acessar LD e ST, podem implementar um acesso invertido (barrado) a operandos baseados em bit. O mesmo vale para operadores booleanos.

```
LdN    I_xInput1
AndN   I_xInput2
And(   I_xInput3
Not
Or     I_xInput4
)
St     Q_xOutput1
```



4.5. Chamando funções

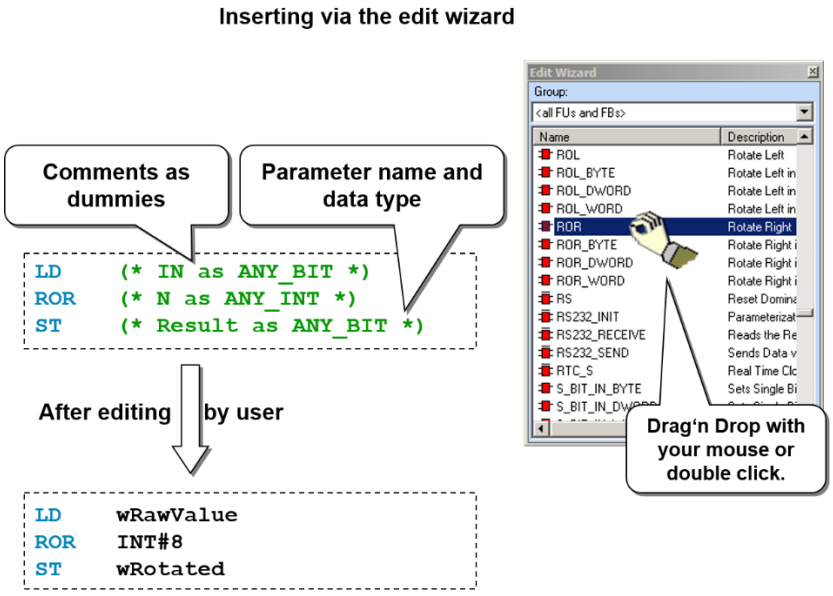
A chamada de funções na IL tem uma pequena diferença da sintaxe das operações padrões. O primeiro parâmetro pode ser carregado explicitamente via LD ou ser retirado do acumulador.



A diferença fica clara se forem usadas funções com mais de uma entrada. Os parâmetros absolutos que seguem o segundo parâmetro não são escritos em linhas separadas, mas sim por trás do nome da função, na ordem correta e separadas por vírgulas.

4.5.1. Edição no PC WORX

A edição de uma função no PC WORX pode ser realizada digitando a sintaxe da função por meio do teclado. Se os parâmetros, a ortografia da função ou a ordem dos argumentos são desconhecidos, um modelo de sintaxe pode ser inserido na programação através do assistente de edição.

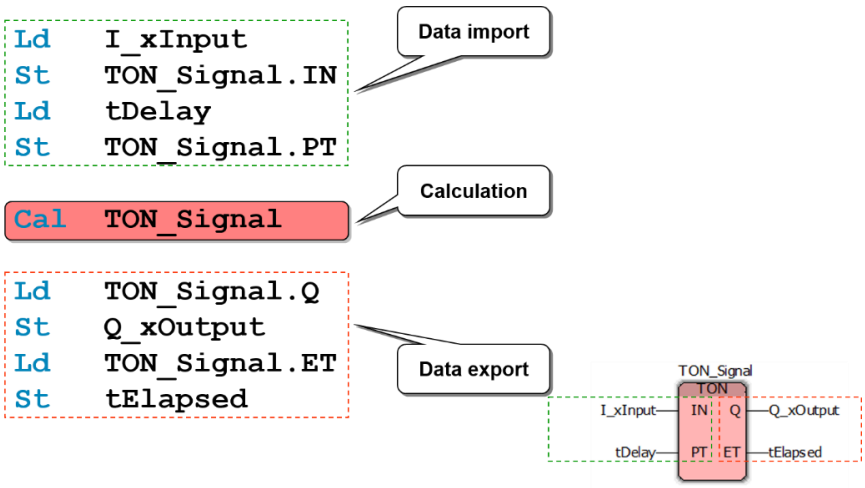


A variável de destino e os argumentos a serem transmitidos são inseridos como comentários pelo assistente de edição e devem ser substituídos por variáveis. Especialmente, as funções não podem ser executadas apenas usando o assistente de edição.

4.6. Chamando bloco de funções

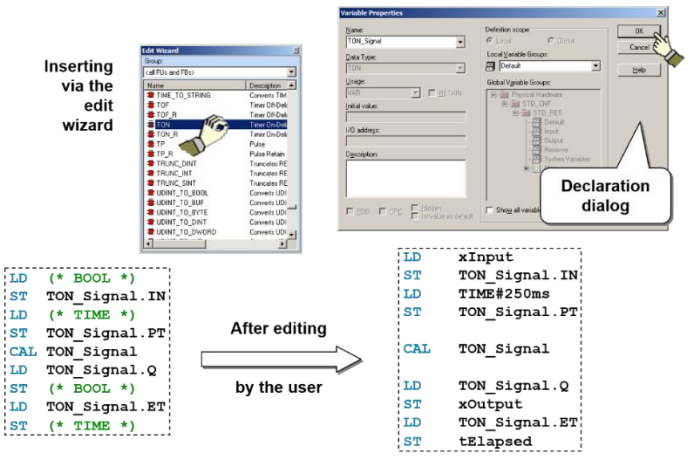
A chamada de um FB na IL é realizada, como em todas as outras línguas, em três fases:

1. Fornecimento de valores para os parâmetros de entrada (importação de dados);
2. Execução da funcionalidade do bloco, se necessário, usando dados salvos (cálculo);
3. Salvando os valores calculados através dos parâmetros de saída nas variáveis criadas (exportação de dados).



4.6.1. Edição no PC WORX

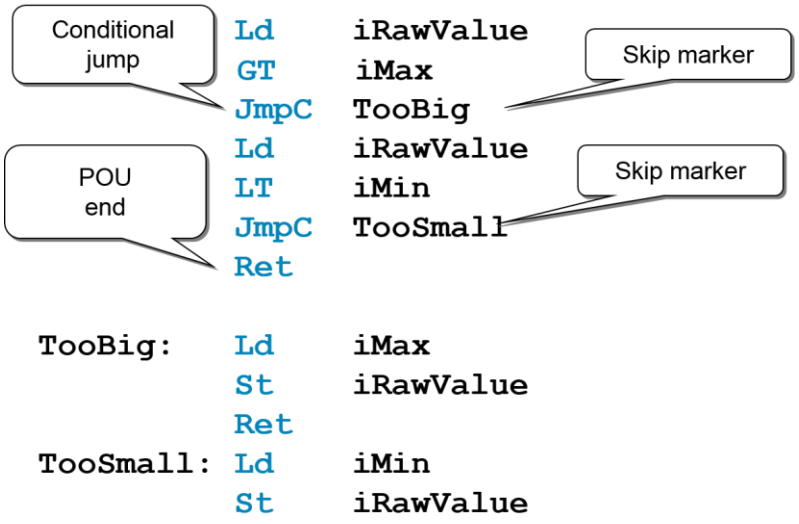
Assim como para as FU, a edição de um bloco de função no PC WORX pode ser realizada digitando a sintaxe FB através do teclado. Você deve ter em mente, no entanto, que o bloco de função, ao contrário das funções, precisa ser instanciado. Isso pode ser feito manualmente através da tabela de variáveis ou através da caixa de diálogo de declaração de variável. O tipo FB é determinado como tipo de dados.



Se o assistente de edição for usado para adicionar um FB à programação, a declaração da instância será executada como nos idiomas gráficos através da caixa de declaração da variável.

4.7. Execução de Código Condicional JMP | RET

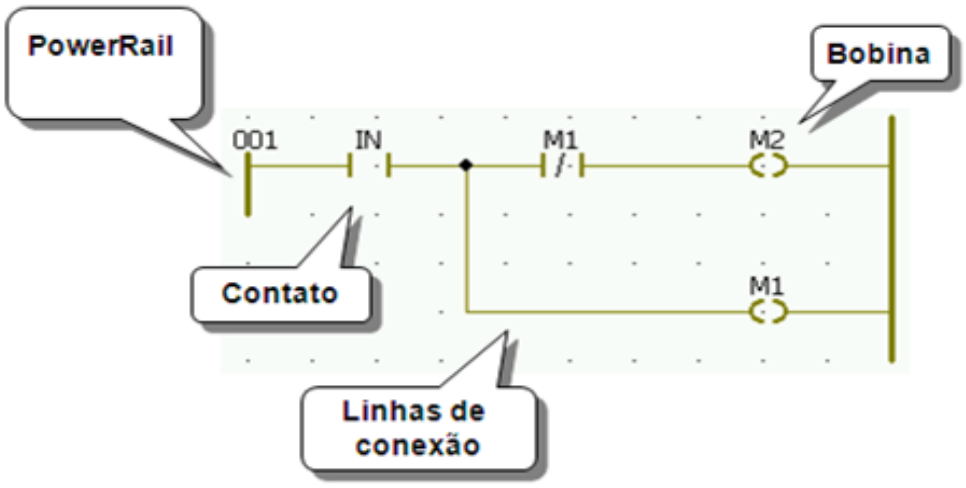
O operador de salto JMP e o operador final do bloco de função RET permitem que a ordem de execução se desvie da ordem padrão de IL. Para o JMP, o endereço de salto deve ser determinado. O marcador de salto não é declarado. Como alvo, pode ser inserido sozinho em uma linha ou, como ilustrado a seguir na frente das instruções.



O modificador C permite a execução condicional dos operadores JMP, RET e CAL.

5. LD Ladder Diagram (Diagrama Ladder)

Este capítulo descreve os elementos do *Ladder* e sua aplicação necessária para a programação. O programa na linguagem *Ladder* permite apenas lógicas do tipo boleana. Uma lógica deve sempre ser finalizada usando barras no lado direito e no lado esquerdo (ilustração abaixo).

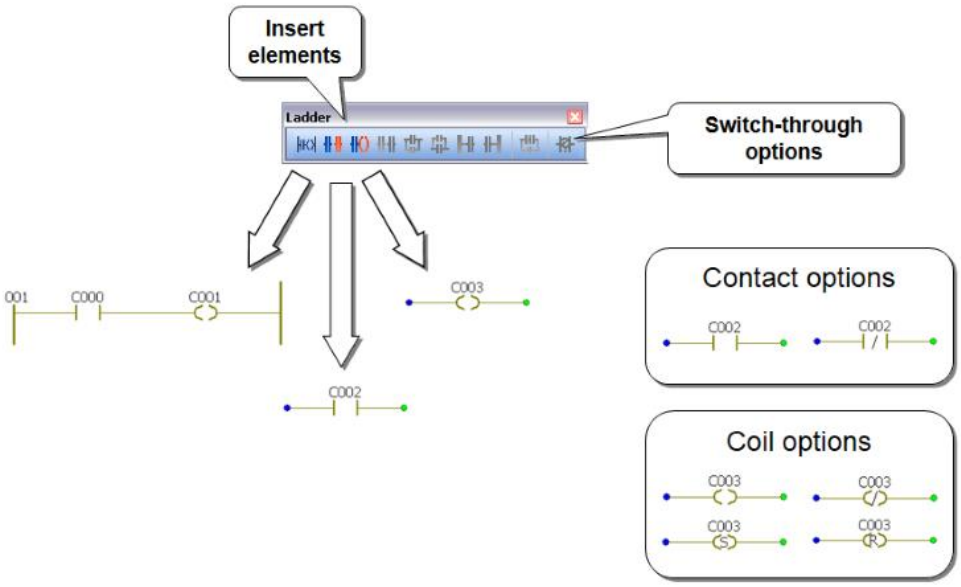


Estas barras são chamadas de PowerRail. Quando inserir novos contatos ou bobinas, as variáveis devem ser declaradas da mesma forma que no FBD.

Obs.: Nenhum contato ou bobina deve ser instalado no lado direito de uma bobina.

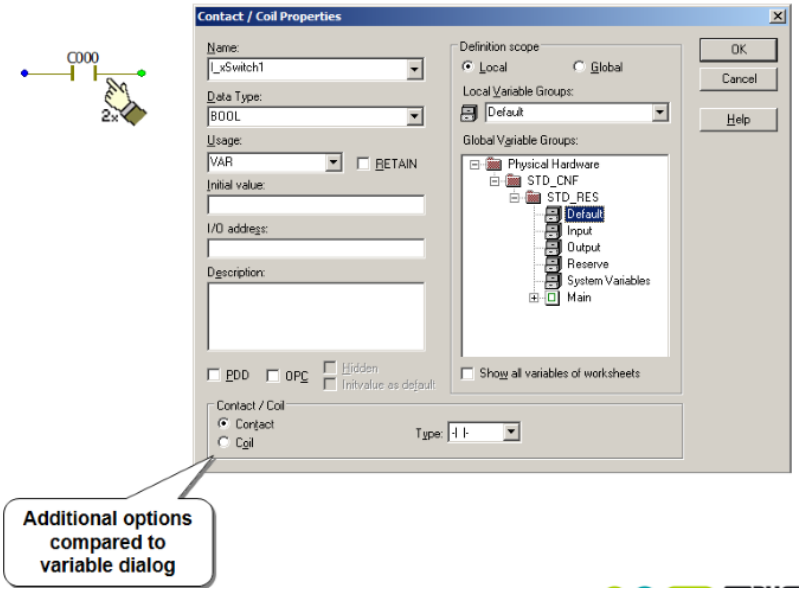
Os elementos básicos do diagrama *Ladder* podem ser inseridos e editados através da barra de menu LD.

As redes existentes podem ser estendidas e completadas usando esses botões. Dependendo do elemento selecionado na planilha, os botões correspondentes são habilitados (vide imagem abaixo).

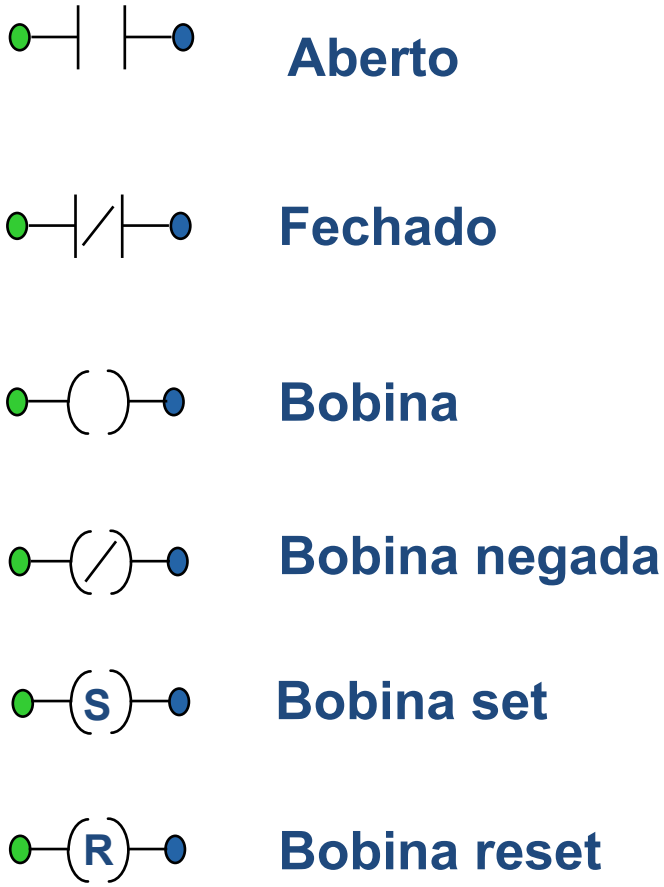


A caixa de diálogo para definir as propriedades de contato/bobina é um pouco diferente da caixa de diálogo das propriedades das variáveis. Apesar disso, todos os elementos de controle para configurar o diagrama *LD* estão disponíveis.

Obs.: apenas os tipos de dados que permitem o acesso a um parâmetro booleano podem ser selecionados.

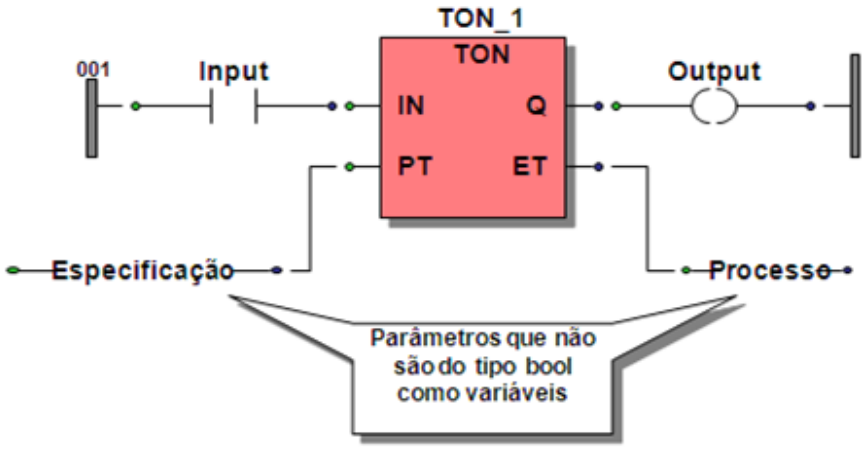


Elementos Básicos



5.1. FB no Ladder

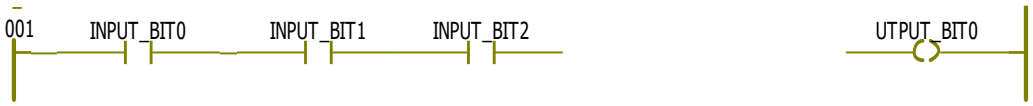
Os elementos do diagrama de bloco de funções (variáveis, funções e blocos de funções) podem ser usados no diagrama *Ladder*.



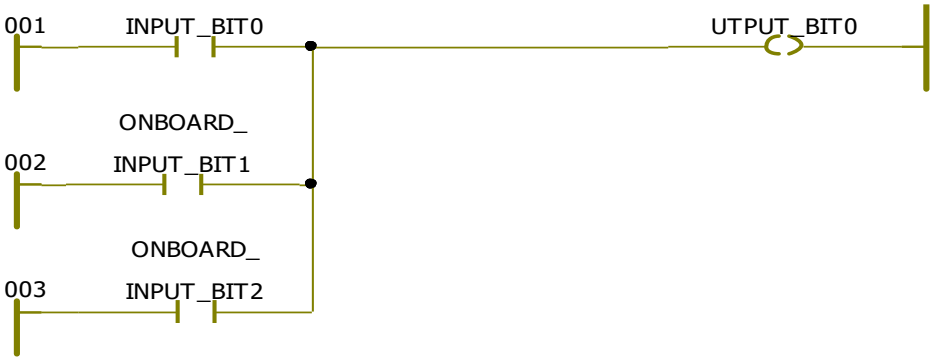
Obs.: Muitas combinações das duas linguagens dentro de uma mesma rede podem levar a resultados que não serão interpretados claramente pelo compilador e, portanto, resultando em mensagens de erro.

5.2. **Lógica**

Porta AND:

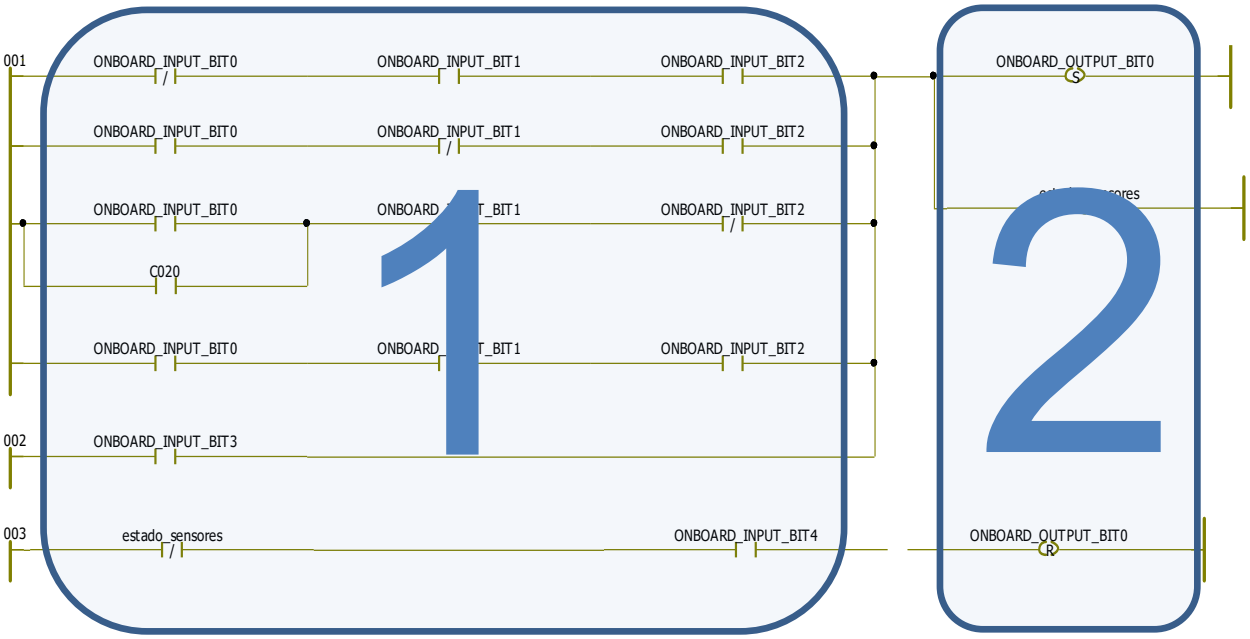


Porta OR:



5.3. **Fluxo de Processamento**

Em *Ladder*, o processamento ocorre da seguinte forma: Leitura de todos os contatos e depois aplicação dos sinais nas bobinas.



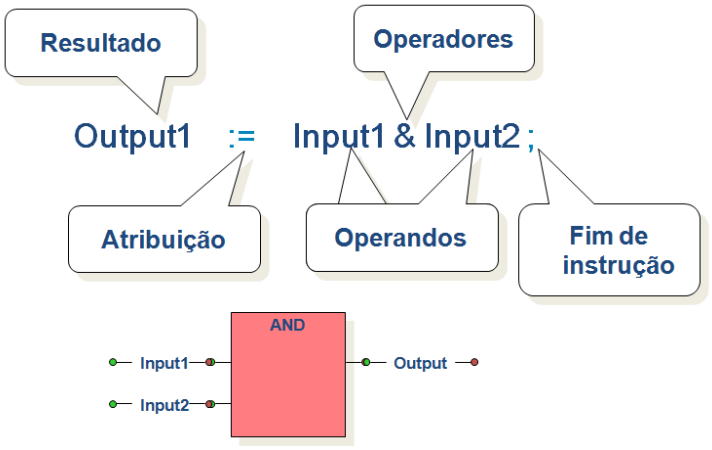
Exercício:
Vamos fazer um semáforo em Ladder, usando com acionamento a régua de chaves bit0 ~bit6.

6. ST (*Structured Text*) Texto Estruturado

Este capítulo apresenta as possibilidades de programação em ST. Estes incluem elementos como operadores FU e FB, que também estão disponíveis nas outras linguagens.

6.1. Elementos de idioma

Ao contrário do FBD, na linguagem de ST as atribuições são feitas da direita para a esquerda através do operador " := "
Uma instrução em ST não é restrita a uma linha. Pode se estender por mais linhas, porém deve terminar por um " ; ".



O acesso aos elementos de idioma disponíveis em ST, é principalmente suportado pelo assistente de edição. Além dos grupos de FU e FB, um grupo de elementos-chave estão disponíveis:

Assignments, Operators

```
Q_xOutput1 := False;
Q_xOutput2 := I_xInput2 & Not I_xInput4;
```

Requests

```
If I_xInput1 & I_xInput2 Then
   Q_xOutput1 := True;
ElsIf I_xInput2 & I_xInput3 Then
   Q_xOutput2 := True;
End_If;
```

FU calls and arguments

```
iScaled := Limit(iMin, iBase, iMax);
```

FB calls and arguments

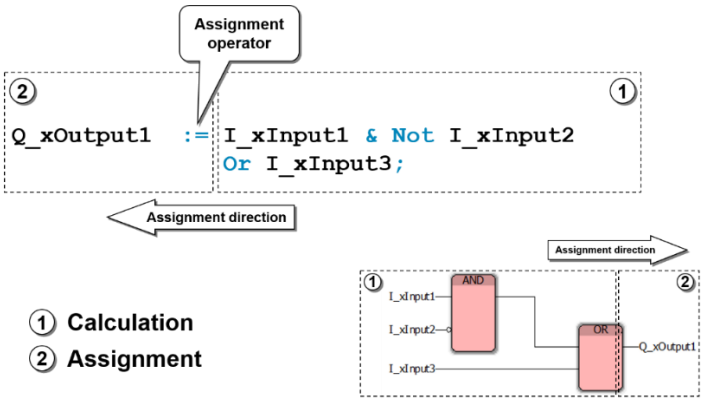
```
CTU_Output (CU    := xPulse,
            PV    := iMaxValue,
            RESET := xStop);

iValue            := CTU_Output.QV;
xFull             := CTU_Output.Q;
```

Loops

```
Repeat
   iLoop := iLoop + 1;
Until iLoop = 100
End_Repeat;
```

6.2. **Atribuições e Operadores**

A principal diferença entre linguagens gráficas e textuais é que a atribuição de expressões (constantes, variáveis e os cálculos) é realizada da esquerda para a direita em linguagens gráficas e da direita para a esquerda em linguagens textuais. No que diz respeito ao conjunto de comandos, aos operadores e aos elementos de idioma, não existem diferenças entre esses dois tipos de linguagens de programação.



Os elementos-chave que são a base para as atribuições são sempre o operador de atribuição := e o ; para terminar uma instrução. Conforme mostrado no exemplo acima, as instruções podem ser executadas em várias linhas para aumentar a clareza da programação.

6.3. **Hierarquia dos operadores**

Os operadores disponíveis em ST estão sujeitos a uma determinada ordem de execução. Isso pode ter uma importância significativa se mais de um operador for usado em uma expressão. Usando suportes, o usuário sempre pode alterar essas prioridades. No entanto, os suportes também podem ser usados para obter maior clareza na programação.

A coluna *Data type group* na lista abaixo indica para qual tipo de dados ou grupo de tipos de dados que os operadores podem ser usados.

Priority	Operation	Symbol	Data type group
	Brackets	(Ausdruck)	ANY
	Function evaluation	Funktion(Argumente)	*
	Potentialization	iZahl1 ** iZahl2 *	NUM
	Negation Complement	-iZahl NOT wCode	NUM BIT
	Multiplication Division Modulo	iZahl1 * iZahl2 * rZahl1 / rZahl2 * iZahl1 MOD iZahl2	NUM
	Addition Subtraction	iZahl1 + iZahl2 * rZahl1 - rZahl2 *	NUM
	Comparison	diA > diB | wC < wD | iE >= iF | iG <= iH	ANY
	Equality Inequality	iZahl1 = iZahl2 * rZahl1 <> rZahl2 *	ANY
	Boolean AND	xVar1 & xVar2 wCode1 AND wCode2	BIT
	Boolean exclusive OR	xVar1 XOR xVar2	BIT
	Boolean OR	bVar1 OR bVar2	BIT

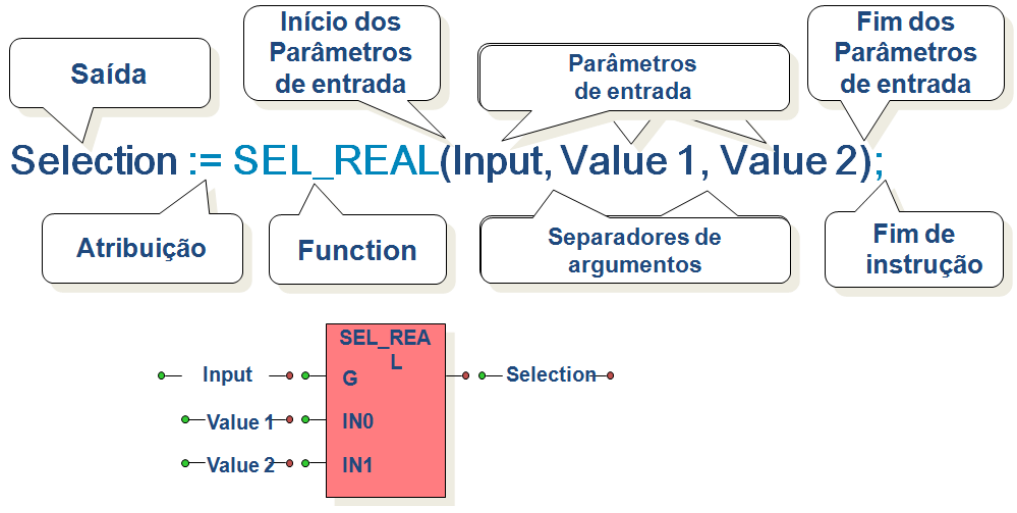
Matemáticos	Comparações	Binários	Gerais
+ Soma	= Igual	NOT Complemento	:= Atribuição
- Subtração	<> Diferente	& And	( ) Delimitador
* Multiplicação	> Maior	OR OU	(* *) Comentário
/ Divisão	>= Maior igual	XOR OU Exclusivo	
MOD Módulo	<= Menor igual		
** Expoente			

6.4. **Chamando funções**

Para o uso dos FU em ST, todos os parâmetros devem ser manipulados na ordem determinada pela FU. Similar à atribuição padrão em FBD, a variável a esquerda do sinal de atribuição é atribuída como o valor de retorno da FU. A FU pode ser inserida sem declaração.

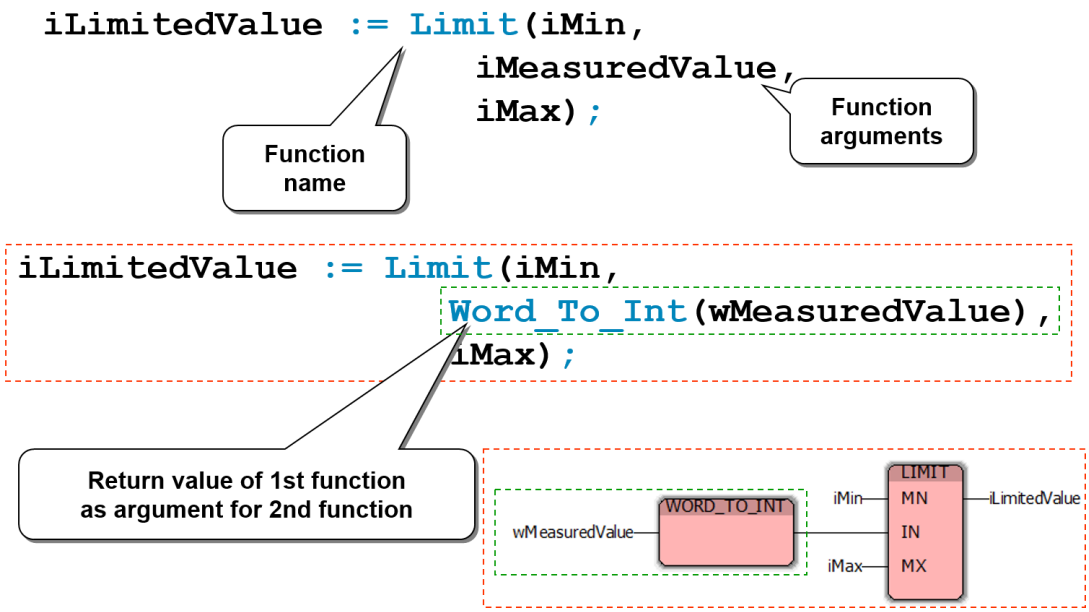
O parâmetro de entrada é manipulado entre parênteses, separado por uma vírgula. O PCWorx Express pode inserir a sintaxe exigida para o FU na *worksheet* através do assistente de edição (Edit Wizard).

De acordo com o exemplo a seguir, é possível executar uma associação de FU.



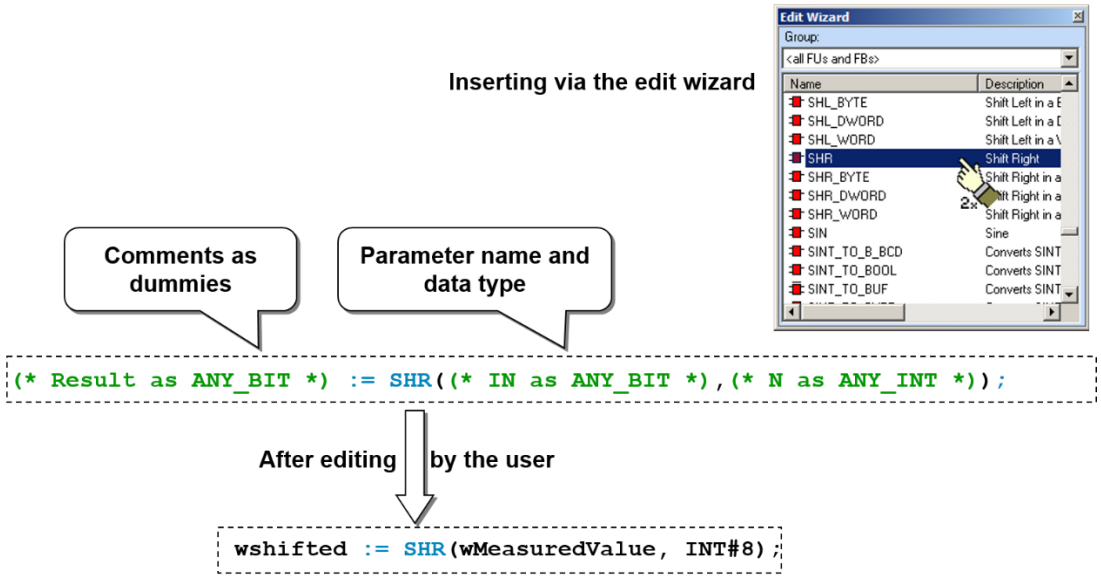
Após o nome da FU, os argumentos a serem transmitidos seguem entre colchetes, separados por vírgulas. Recomenda-se seguir o padrão de uma linha por argumento. Na aplicação mais simples, o valor de retorno da função é atribuído a uma variável de destino através do operador de atribuição.

O segundo exemplo, destacado abaixo, mostra a como as funções podem ser nítidas. O resultado do *Word_To_Int* neste exemplo é usado como um segundo argumento para a função *Limite.*



6.4.1. Edição no PC WORX

A edição de uma FU no PC WORX pode ser realizada digitando a sintaxe da FU por meio do teclado. Se os parâmetros, a ortografia da FU ou a ordem dos argumentos forem desconhecidos, um modelo de sintaxe pode ser inserido na programação através do assistente de edição.



As variáveis de destino e os argumentos a serem transmitidos são inseridos como comentários pelo assistente de edição e devem ser substituídos por variáveis. Especialmente, as FU não podem ser executadas usando o assistente de edição.

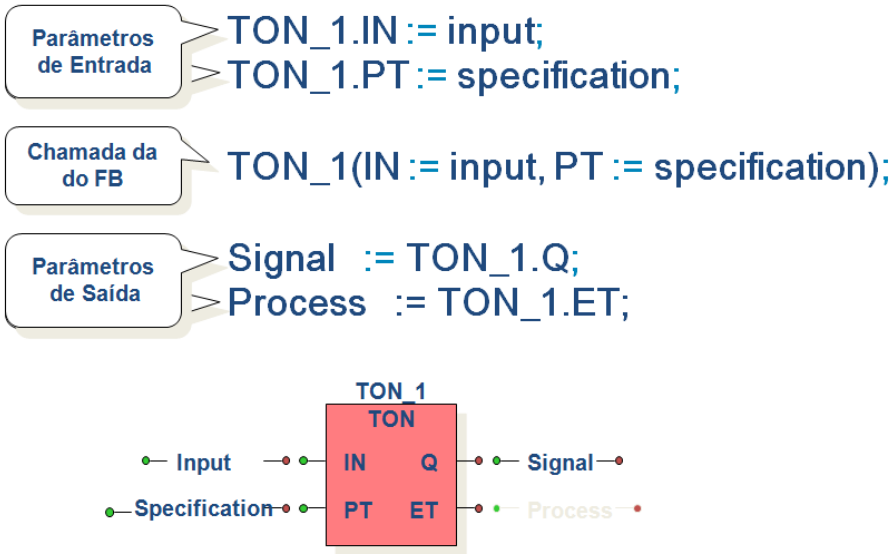
6.5. Chamando Blocos de Funções

O processamento dos FB em ST é executado através de três etapas: importação de dados, cálculo e exportação de dados. Ao contrário da IL, há uma opção de resumir as primeiras duas etapas. Por esta razão, a instância do FB é manipulada através dos valores de entrada entre parênteses.

Entretanto, este processo está relacionado ao parâmetro, conforme mostrado no exemplo TON ilustrado a seguir. Se os valores de entrada são manipulados em parênteses, as instâncias do FB não precisam ser posicionadas na frente para atribuição exclusiva. Isto é necessário, apenas, se um parâmetro da primeira linha for atribuído individualmente.

Da mesma maneira que as FU, o *PCWorx Express* também permite inserir a sintaxe do FB na sua *worksheet*. Por esta razão, a declaração para a instância do FB é chamada automaticamente como na linha de instrução. Caso contrário, ela deve ser inserida manualmente depois.

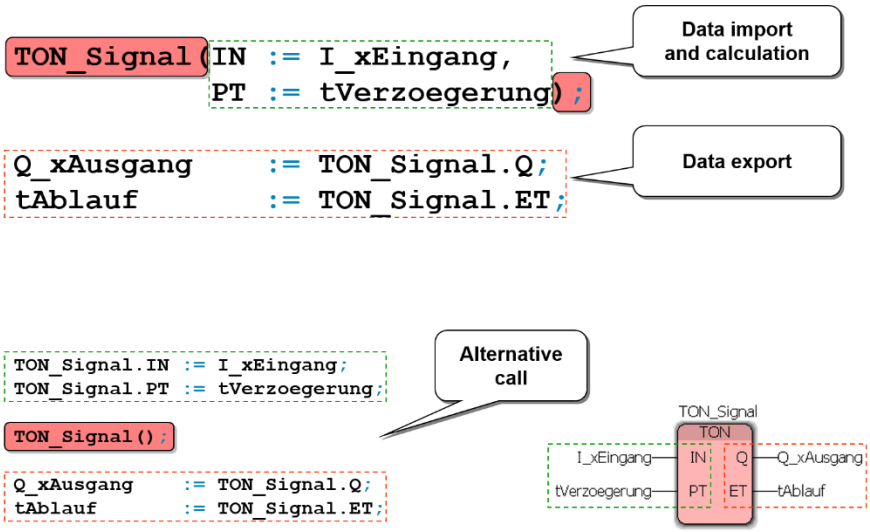
Em lugar de variáveis ou constantes, o valor de retorno das FU pode ser usado como parâmetro de entrada.



A chamada de FB em ST é realizada, como em todas as outras linguagens, em três fases:

1. Fornecimento de valores para os parâmetros de entrada (importação de dados);
2. Execução da funcionalidade do FB, se necessário, usando dados salvos (cálculo);
3. Salvando os valores calculados através dos parâmetros de saída nas variáveis criadas (exportação de dados);

Na primeira opção os parâmetros de entrada dentro de um suporte são conectados aos parâmetros formais da instância do FB. Portanto, o nome da instância não precisa estar localizado na frente deles, como é o caso da chamada alternativa.

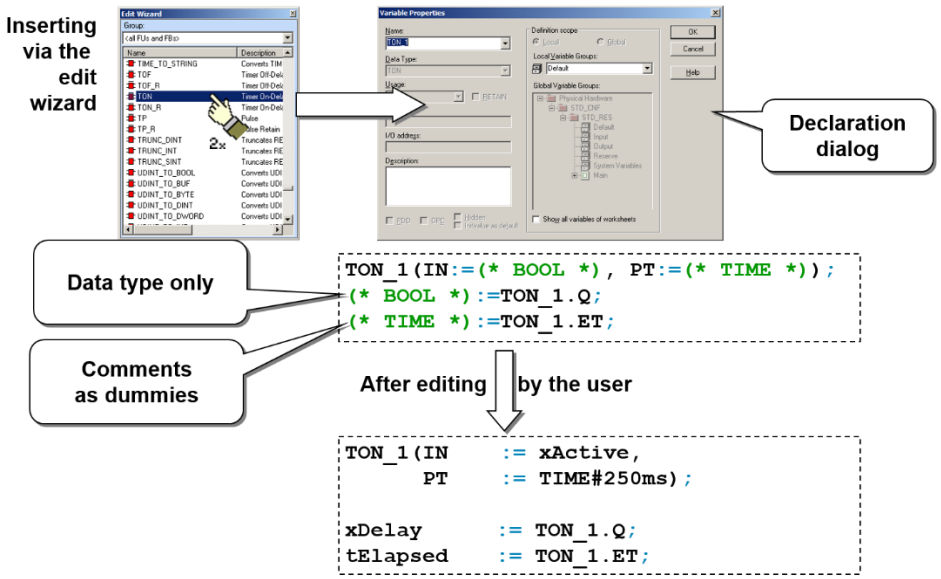


Em qualquer caso, os colchetes durante a operação de chamar o FB devem ser inseridos, mesmo que a importação de dados seja feita separadamente.

6.5.1. **Edição no PC WORX**

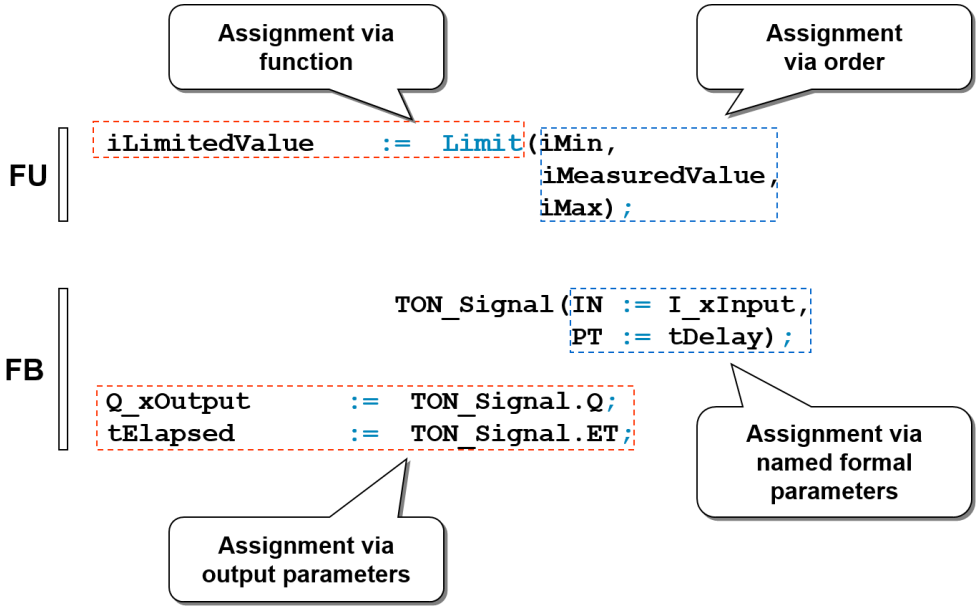
Assim como para as FU, a edição de um FB no *PCWorx Express* pode ser realizada digitando a sintaxe FB através do teclado. Você deve ter em mente, no entanto, que a FB, ao contrário das FU, precisam ser instanciadas. Isso pode ser feito manualmente através da tabela de variáveis ou através da caixa de diálogo de declaração de variável.

Se o assistente de edição for usado para adicionar uma FB à programação, a declaração da instância será implementada como nos idiomas gráficos através da caixa de diálogo de declaração da variável que se abre.



A chamada de um FB não pode ser implementada através do assistente de edição. Usando o assistente de edição, a sintaxe de uma única chamada pode ser inserida e a instância FB declarada.

6.6. Comparação entre “chamar” uma FU ou FB



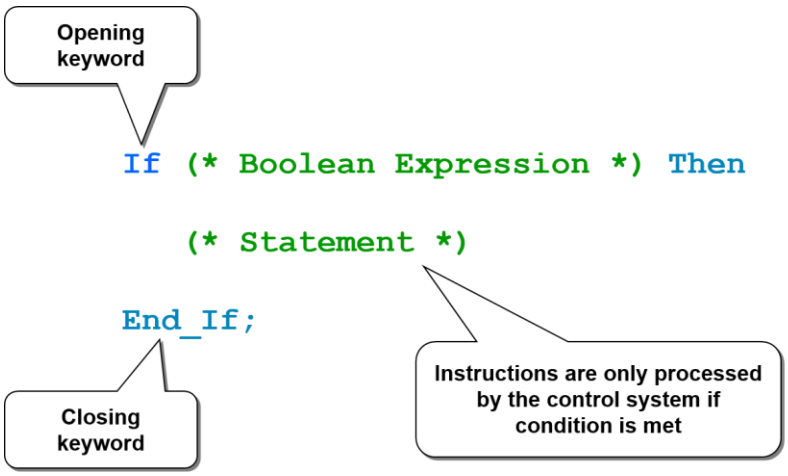
O diagrama acima é uma comparação da estrutura para chamar uma FU e FB. A FU é muito restritiva em relação às suas possibilidades de conexão, os argumentos são atribuídos através de uma ordem. Os FB são baseados no acesso aos parâmetros de entrada e saída, desde que nomeados corretamente.

6.7. Elementos de programação

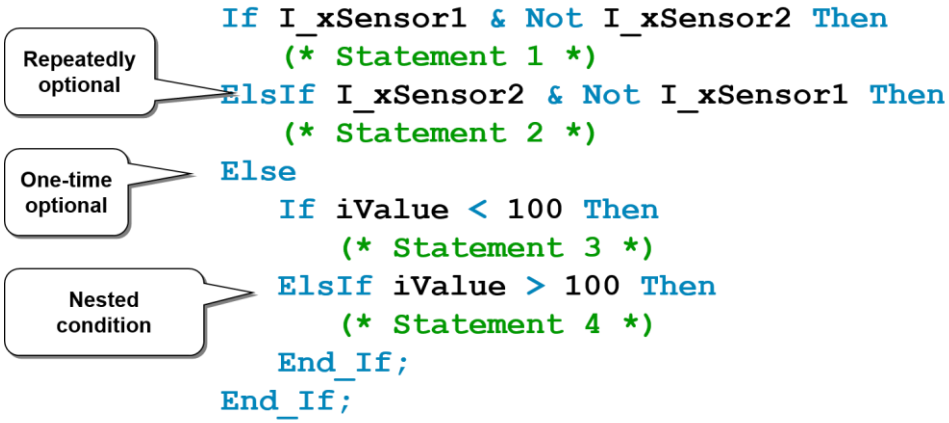
O objetivo deste item é desenvolver os elementos de programação em ST e visualizar sua aplicação em FB. Uma das vantagens da programação em ST, é que os elementos normais em linguagens padrão, como PASCAL, C, C++, JAVA, também estão disponíveis. Estes elementos serão apresentados nas próximas páginas.

6.7.1. Instrução IF

A estrutura construída na palavra-chave **IF** torna possível que o programador tenha um código executado, dependendo de uma condição booleana.



A estrutura básica pode ser estendida usando duas palavras-chave:



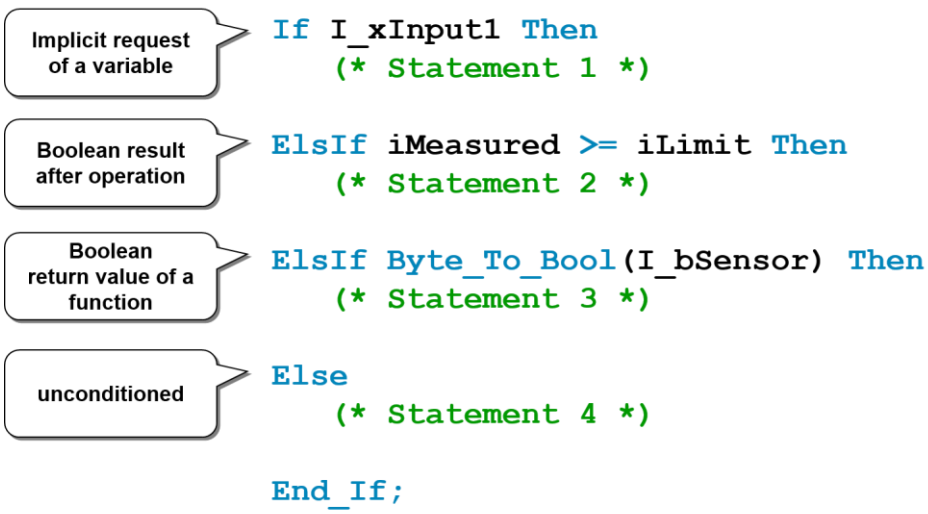
ElsIf trata-se de um pedido condicionado se a condição da abertura não for cumprida. Ele pode ser usado tantas vezes quanto possível. No entanto, dentro de uma estrutura **IF**, apenas a instrução dependendo da primeira condição encontrada é executada.

Ao contrário de **ElsIf**, se dentro de uma estrutura **IF** nenhuma das condições anteriores foram atendidas, as instruções dependendo de **Else** são executadas.

O diagrama acima mostra também um exemplo para uma estrutura **IF** aninhada. O segundo bloco **IF** só é executado se **Else** estiver ativado, e posteriormente seguindo as regras de uma estrutura **IF** independente.

Várias programações podem ser usadas como condições para os elementos e extensões (vide diagrama a seguir):

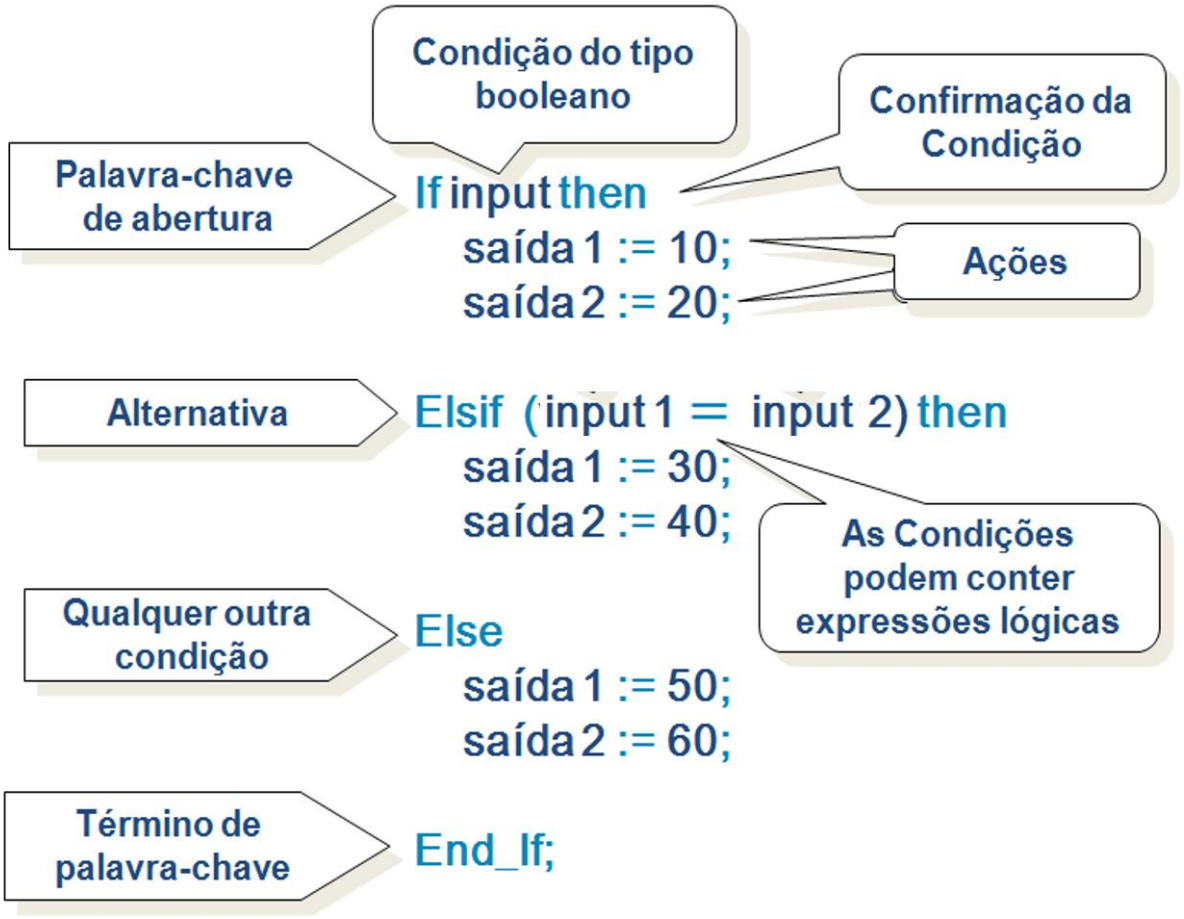
- Variáveis booleanas (com NOT para pedidos invertidos);
- Resultados booleanos das operações;
- Valores de retorno booleanos das funções;
- **Else** sem condição (uso possível dentro de uma estrutura IF somente uma vez).



Para uma instrução de decisão **IF**, podemos tomar a decisão de executar uma parte do programa através de um operador do tipo BOOL ou do resultado do tipo BOOL de uma operação.

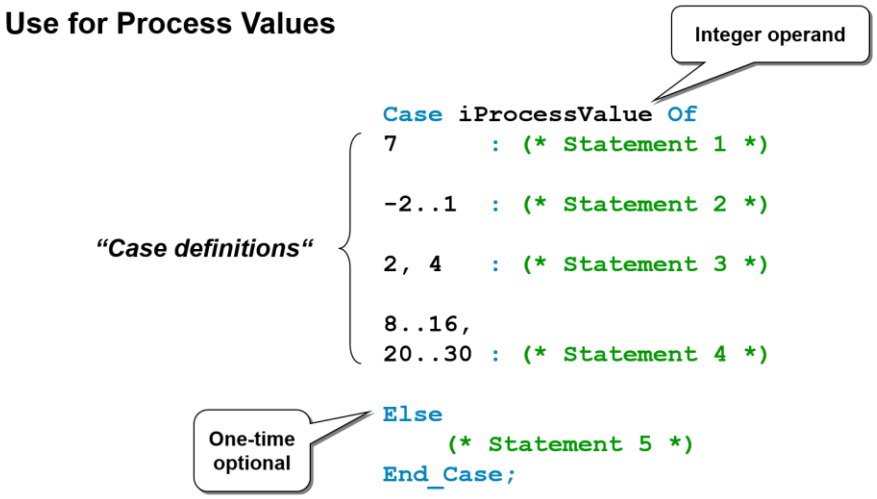
Basicamente, dentro de uma estrutura **IF**, apenas um bloco de instrução pode ser processado por vez, durante a execução do programa. É possível criar uma estrutura cascata do comando **IF** (ilustração a seguir).

Na programação de alto nível, é comum acrescentarmos tabulações para que as partes do programa apareçam na mesma altura.



6.7.2. **Instrução CASE**

A instrução **CASE** solicita o valor de uma variável do tipo de dados Inteiro. O exemplo a seguir mostra que valores individuais, valores separados por vírgulas, intervalos de valores e a combinação dos dois últimos podem ser usados como definições da instrução **CASE**.



Tal como acontece com a estrutura **IF**, o bloco de instruções cuja condição (valor) corresponde ao valor da variável solicitada é executado. Ao contrário da estrutura **IF**, em que as condições não precisam ser mutuamente exclusivas, uma sobreposição de definições em estruturas **CASE** é lida como um erro.

O bloco de instruções dependendo do opcional **Else** é executado se os valores das variáveis não coincidem com os valores solicitados.

Todos os valores devem ser selecionados a partir do intervalo de valores do tipo de dados INT.

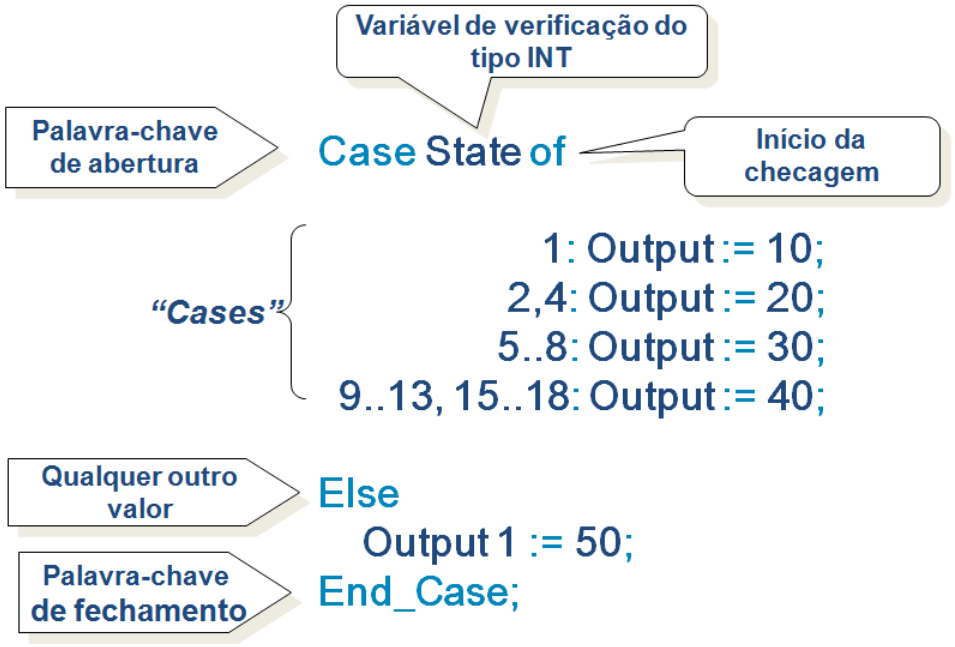
Use for Control Values

```
Case iProcessStep Of
0    : (* Perform initialization*)
       If xInit_finished Then iProcessStep := 10;
       End_If;

10   : (* Execute Process 1 *)
       If xProcess1_finished Then iProcessStep := 20;
       End_If;

20   : (* Execute Process 2*)
       If xProcess2_finished Then iProcessStep := 30;
       End_If;

(* etc *)

500  : (* Exception handling *)
   iProcessStep := 0;
End_Case;
```

Se a instrução **CASE** for usada para construir uma programação em cascata de passos textual (ilustração acima), a variável solicitada é usada para controlar a sequência. Seu status mostra a etapa atual do processo.

No exemplo acima, a variável ***iProcesStep*** é inicializada após o início do CLP com 0 e o bloco de instruções ao qual este valor é atribuído é executado até que o processo relate o fim da sequência de inicialização através da variável booleana ***xInit_finished***. Como resultado, a variável ***iProcessStep*** está configurada para 10 e no próximo ciclo do CLP, o bloco de instrução ao qual o novo valor é atribuído é executado.

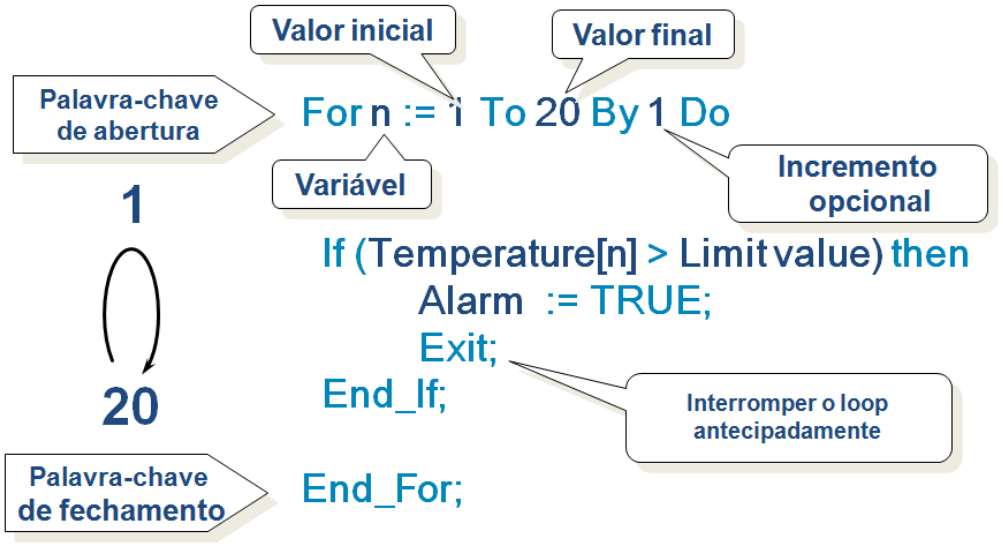
Ao contrário da instrução **IF**, a instrução **CASE** requer um valor em uma variável INT. Conforme ilustração a seguir, as execuções do programa podem ser realizadas para as seguintes situações: valores individuais, valores separados por vírgulas ou valores de variáveis de uma faixa de valor individual ou de uma das várias faixas de valor.



6.7.3. **Instrução FOR**

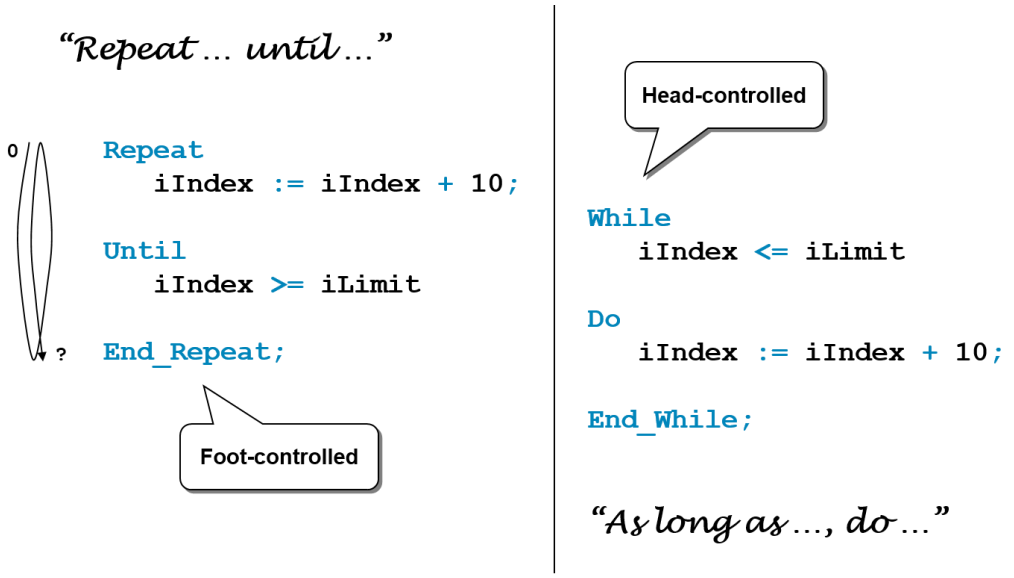
A instrução **FOR** no PCWorx Express contradiz a ordem de processamento pré-determinada em um CLP. Ele poderia ser descrito como um retorno contínuo dentro do processo do programa, até que o valor da variável atinja o valor final.

Por esta razão, depois da programação com a instrução **FOR**, a duração do processamento de todo a programação (time-out) deve ser prolongada de modo que a *task* a ser executada não tenha o tempo expirado. A instrução *Exit* permite a interrupção imediata da instrução **FOR**.



6.7.4. Instrução Repeat | While

As instruções **Repeat** e **While**, ao contrário das instruções **FOR**, não são pré-determinados. Dependendo de um valor booleano, é decidido se a programação dependente é executada novamente. O tempo de solicitação é diferente entre a instrução **Repeat** e **While.** No ciclo **While**, em primeiro lugar, a solicitação é realizada enquanto no ciclo **Repeat** as instruções dependentes são executadas uma vez antes do ciclo ser interrompido, devido à não conformidade com a condição.



No processo e na programação, é necessário assegurar que os ciclos não sejam executados indefinidamente, pois o PLC não poderá continuar a execução.

7. Data Types (Tipos de Dados)

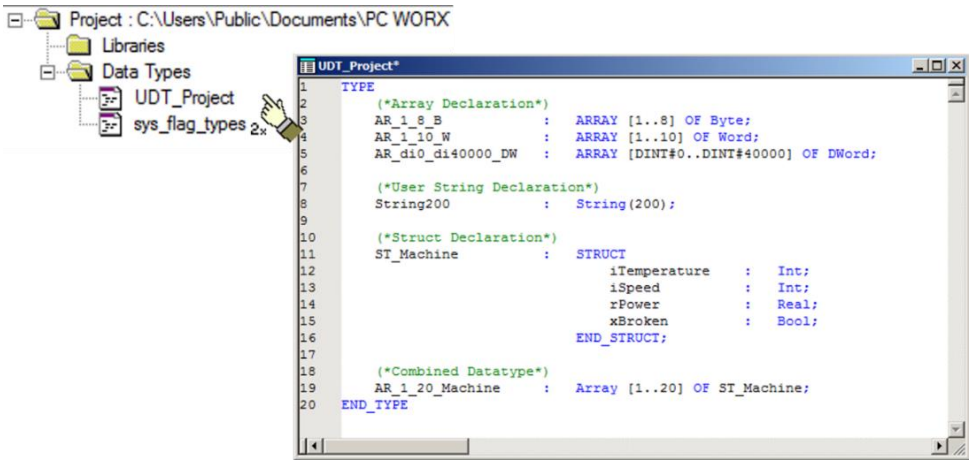
Esta seção informa sobre as diferentes classes e tipos de dados, como eles são declarados e como as variáveis com base neles podem ser usadas na programação. Nesta seção *Data Types* é possível criar tipos de dados especiais, baseados nos tipos de dados padrão, como numéricos, lógicos e textuais.

Estes tipos de dados podem ser:

- **Arrays:** Variáveis indexadas de um único tipo de dado;
- **Estruturas:** Variáveis que embarcam diversos tipos de dados;
- **Strings:** Variáveis de texto padrão, que alocam exatamente 80 BYTES (Este tamanho pode ser aqui alterado para de 1 até 250 BYTES).

A criação de novos tipos de dados por si não gera uma declaração de variável. A declaração deve ser realizada nas POUs Programas, FU e FB.

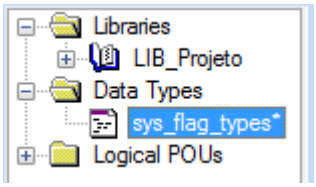
A declaração dos tipos de dados definidos pelo usuário é realizada em planilhas na pasta *Data Types*. A própria planilha pode ser renomeada, porém as declarações de tipo de dados contidas nela não devem ser alteradas.



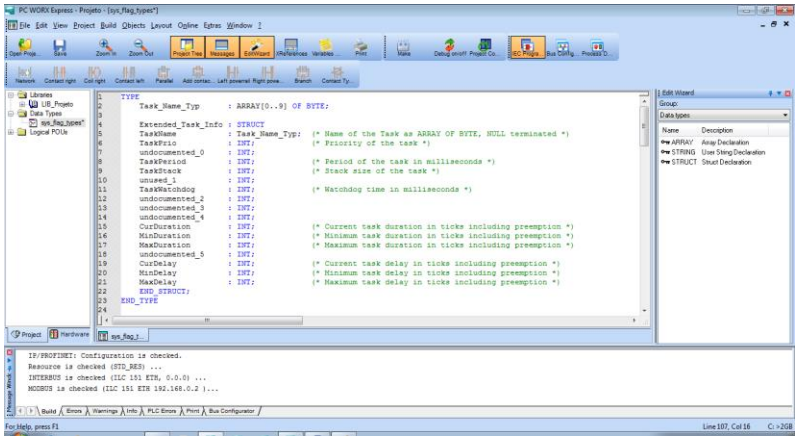
Basicamente, todas as declarações de um projeto podem ser feitas em uma planilha. No entanto, para manter tudo claramente organizado, recomenda-se a criação de planilhas de trabalho individuais relacionadas a funções.

Para criar novos *Data Types*, acesse:

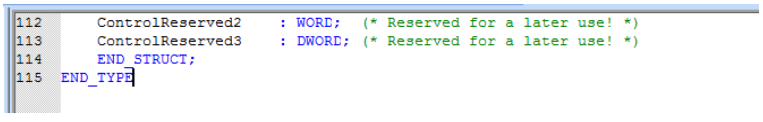
Project Tree Window -> Data Types-> sys_flag_types



Abra a POU sys_flag_types:



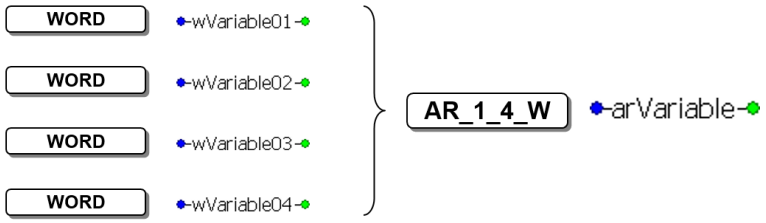
Avance o texto até ao final das declarações geradas automaticamente.



ATENÇÃO PARA NÃO ALTERAR AS DEFINIÇÕES DE TIPO DE DADOS GERADAS AUTOMATICAMENTE.

7.1. **Arrays**

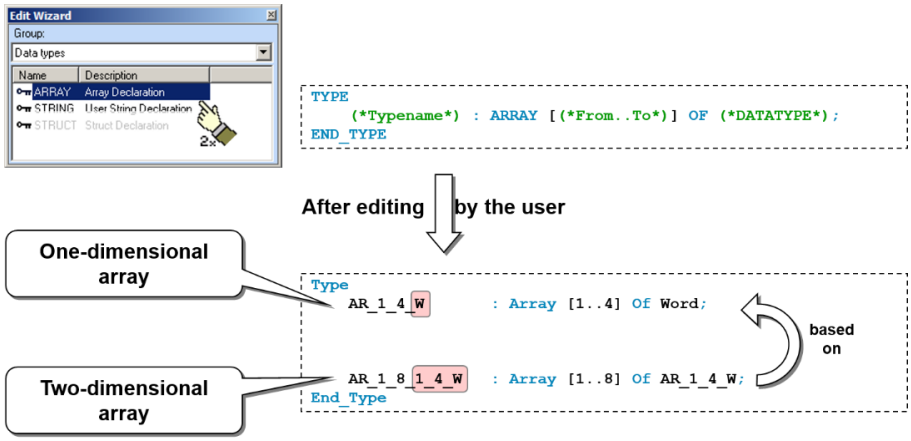
As declarações para os tipos de dados *Arrays*, resumem elementos em um tipo básico de dados. O exemplo ilustrado a seguir, resumo quatro variáveis do tipo de dados *Word* em uma variável matriz com base em um novo tipo de dados. Obs. não é possível visualizar vetores de posição do PC Worx Express.



Name example

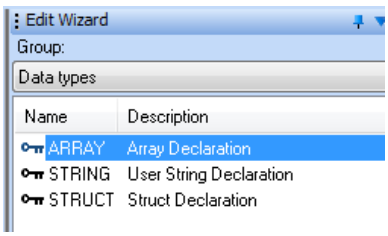
`<AR_LowerBound>_<UpperBound>_<BasicType>`

Para declarar *Arrays*, é necessário usar um determinado formato, que pode ser incluído como ajuda do assistente de edição, adaptado às declarações de tipo de dados. O nome desejado para o tipo de dados, o limite inferior e superior da matriz (entre colchetes e como valor inteiro positivo) e o tipo de dados básicos devem ser inseridos.



A ilustração acima, mostra que a declaração e, portanto, o uso posterior de variáveis desses tipos de dados, podem levar a dificuldades em relação à clareza na programação.

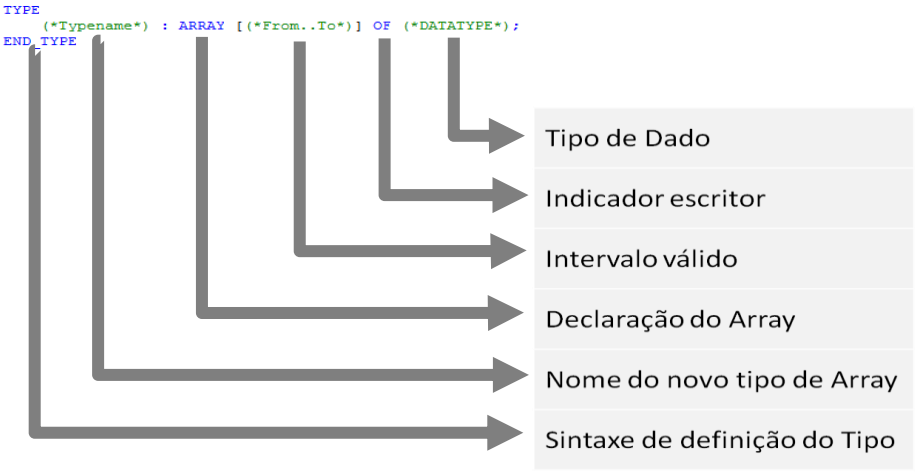
Abra o *Edit Wisard* e aplique um duplo clique em ARRAY.



Resultado:

```
112     ControlReserved2    : WORD;  (* Reserved for a later use! *)
113     ControlReserved3    : DWORD; (* Reserved for a later use! *)
114     END_STRUCT;
115 END_TYPE
116
117
118 TYPE
119     (*Typename*) : ARRAY [(*From..To*)] OF (*DATATYPE*);
120 END_TYPE
121
```

Descrição:



Exemplo de um ARRAY de INT. Neste exemplo um novo Tipo de Dado é declarado como um VETOR de 10 posições de um número inteiro.

```
117
118 TYPE
119     ar_INT_1_10: ARRAY [1..10] OF INT;
120 END_TYPE
121
```

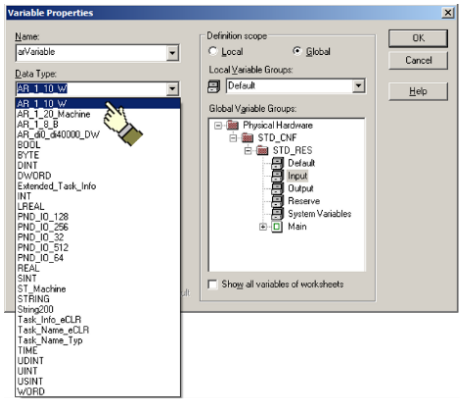
O nome do novo tipo (Type Name) pode ser qualquer um, obedecendo as regras de sintaxe do PCWorx, mas sugerimos nomes compostos intuitivos como neste caso:

ar_INT_1_10

ar -> indica tratar-se de um ARRAY
INT -> indica ser do tipo Inteiro
1_10 -> indica as posições válidas de 1 à 10

Criação de uma variável com do tipo **ar_INT_1_10**.

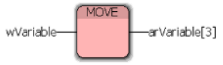
Compile o projeto e crie uma nova variável (vamos criar um FB). Observe que o tipo ar_INT_1_10 estará disponível.



Assigning two array variables of the same dimension

```
arVariable1 := arVariable2;
```

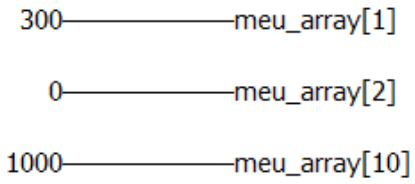
Assigning a single value to an element of an array variable using a constant



Assigning a single value to an element of an array variable using an index variable

```
Ld wVariable
St arVariable[iIndex]
```

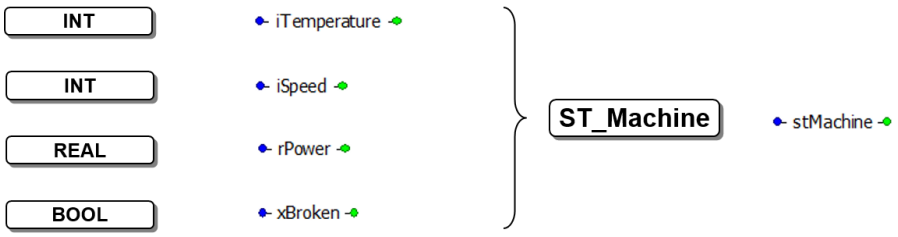
Para utilização e acesso desta variável, a posição desejada deve ser informada entre os caracteres colchetes de abertura e fechamento: Ex. meu_array[1], meu_array[2] ... meu_array[10].



7.2. **Estruturas**

As declarações de tipo de dados *Estruturas* podem resumir elementos de um ou mais tipos de dados básicos. O exemplo a seguir mostra um resumo de quatro variáveis de diferentes tipos de dados em uma variável de estrutura com base em um novo tipo de dados.

Tal como acontece com os *Arrays*, elementos do mesmo tipo podem ser resumidos se o acesso aos elementos individuais não for dado por um índice, mas por designações individuais.



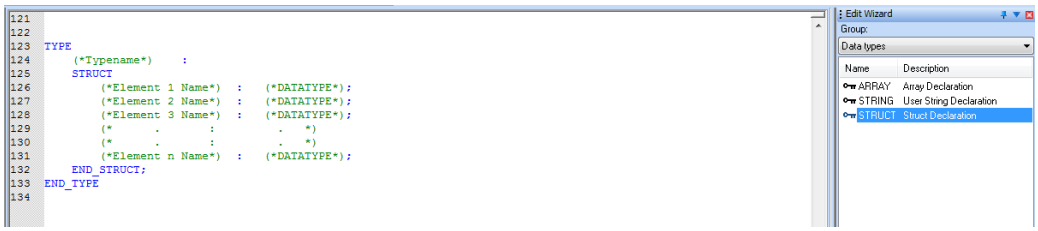
Name example

```
ST_<Function>
```

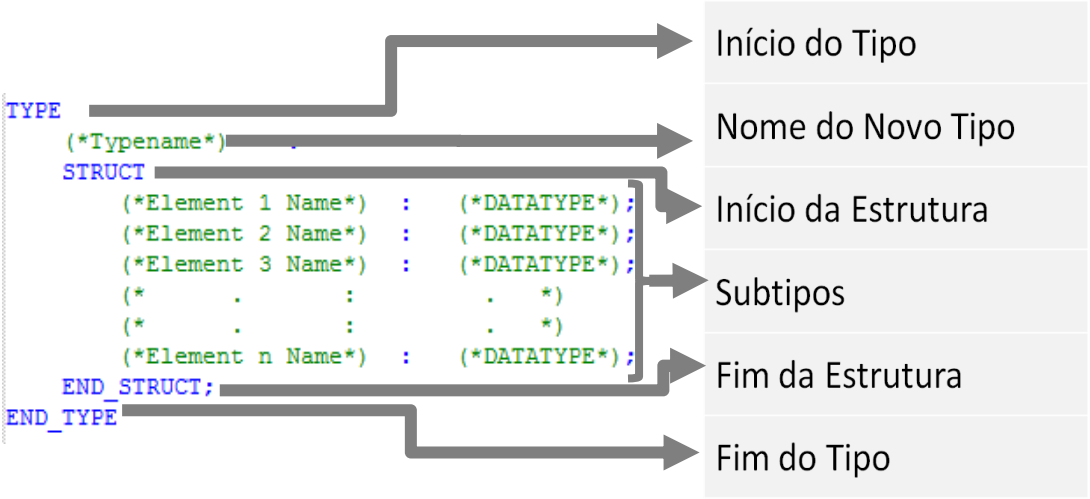
Um nome para um novo tipo de dados pode ser selecionado livremente, no entanto, assim como com todos os outros elementos na programação, recomenda-se usar uma convenção para os nomes de tipos de dados definidos pelo usuário.

Para declarar estruturas, é necessário usar um determinado formato, que pode ser chamado como ajuda através do assistente de edição, adaptado às declarações de tipo de dados. O nome do tipo de dados desejado deve ser inserido e os elementos devem ser listados. A declaração de estruturas multidimensionais é suportada.

Abra o *Edit Wisard* e aplique um duplo clique em STRUCT.



Descrição:



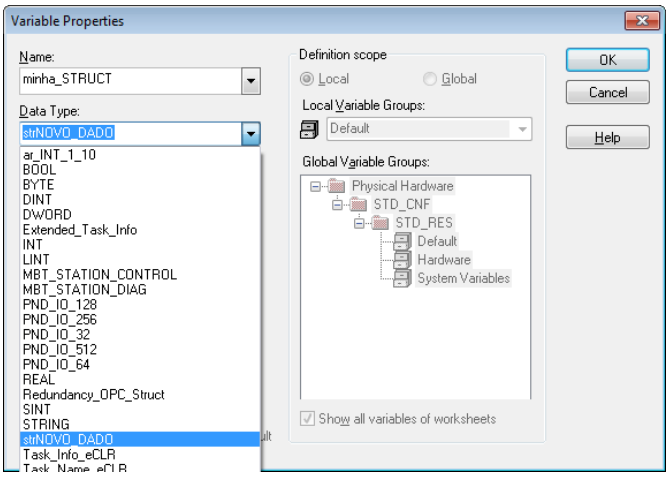
Exemplo de uma Estrutura contendo BOOL, INT, REAL, BYTE e WORD.

```
123  TYPE
124      strNOVO_DADO    :
125      STRUCT
126          Local_BOOL  :   BOOL;
127          Local_INT   :   INT;
128          Local_REAL  :   REAL;
129          Local_BYTE  :   BYTE;
130          Local_WORD  :   WORD;
131      END_STRUCT;
132  END_TYPE
133
```

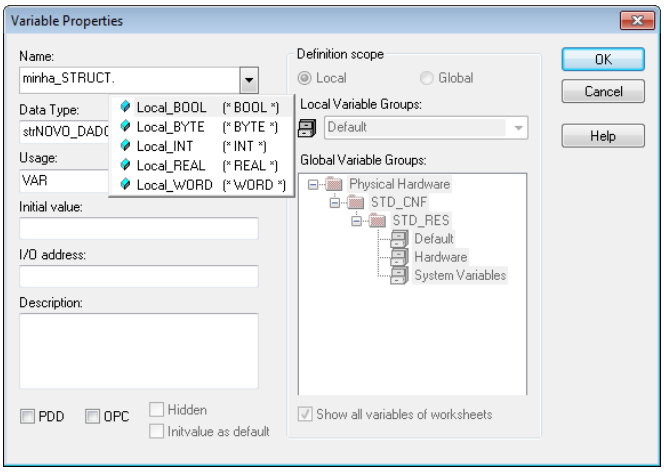
O nome do novo tipo (Type Name) pode ser qualquer um, obedecendo as regras de sintaxe do PCWorx, mas sugerimos iniciar os nomes com str:

Criação de uma variável com do tipo **str_NOVO_DADO**.

Compile o projeto, abra um Programa, Função ou FB e crie uma nova variável. Observe que o tipo str_NOVO_DADO estará disponível na lista suspensa.



Para utilização e acesso das sub-variáveis, o nome da variável deverá ser acompanhado do caractere PONTO (.) e o sub-item. Observe que na janela de variáveis, os sub-items são listados assim que o nome da variável for sucedido pelo PONTO:



Exemplos:

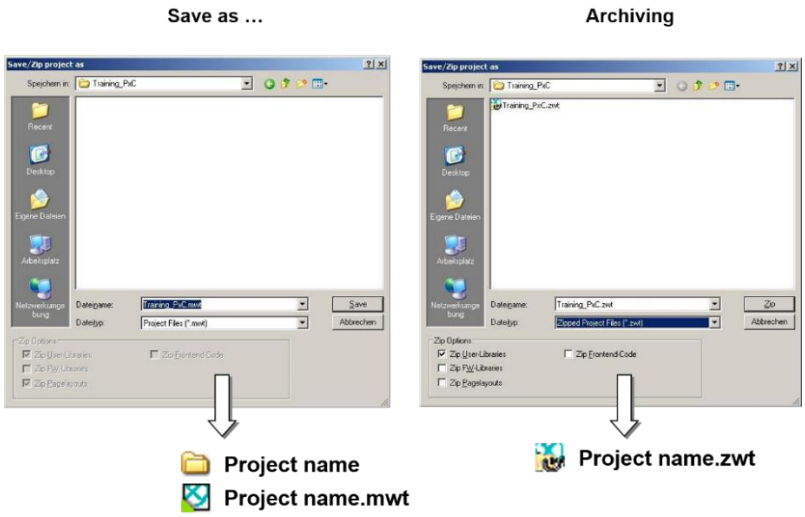
true———————————minha_STRUCT.Local_BOOL

512———————————minha_STRUCT.Local_INT

WORD#16#FFFF———minha_STRUCT.Local_WORD

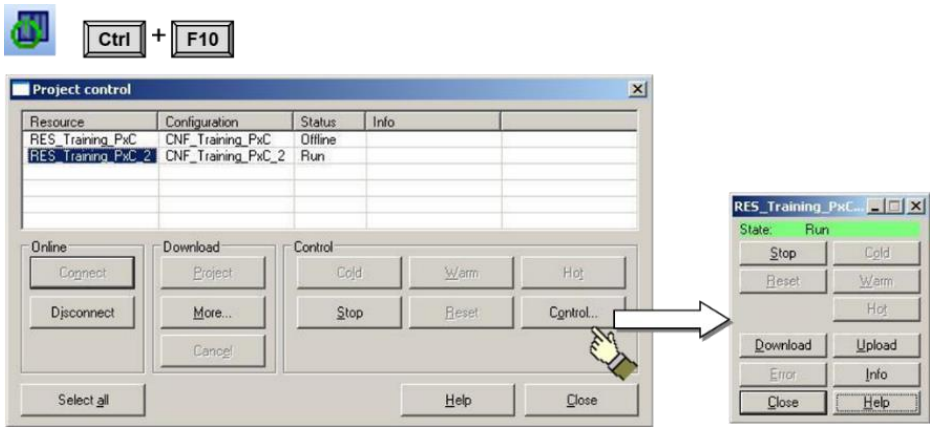
8. Gerenciamento de projetos

Para o processamento no PC WORX, os arquivos do projeto são descompactados e salvos no local de armazenamento selecionado no disco rígido do PC. Além de um arquivo de cabeçalho da extensão * .mwt, esses arquivos incluem um diretório com o mesmo nome.

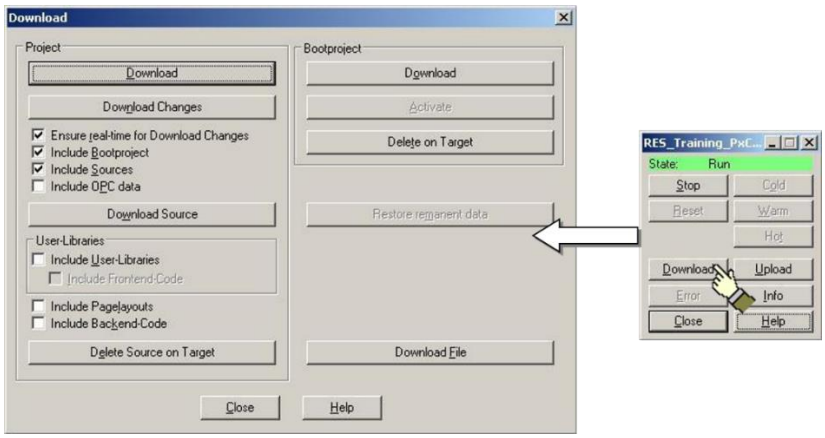


Ao selecionar a opção *Salvar como* ..., o projeto atualmente sendo processado no PC WORX é salvo com o nome selecionado e esses arquivos são usados para processamento posterior.

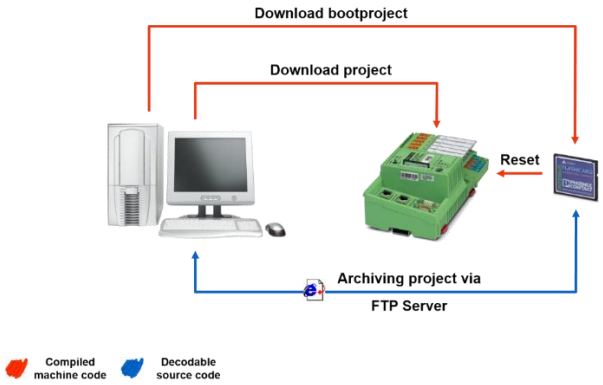
Após a compilação bem-sucedida do projeto, o código pode ser baixado para o sistema de controle através do controle do projeto. No caso de um projeto, em que mais de um sistema de controle foi configurado, o sistema de controle deve ser selecionado através da caixa de diálogo (ilustração a seguir).



A caixa de diálogo de controle individual só pode ser chamada após a conexão ao sistema de controle ter sido estabelecida via *Connect*.



A seguir uma ilustração dos fluxos de download.

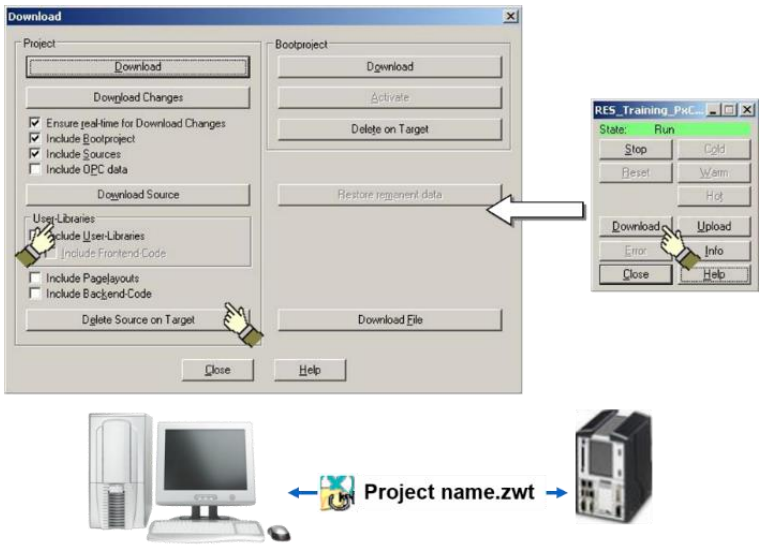


Ao manipular códigos, você pode trabalhar com dois tipos de código:

Compiled machine code
Este código pode ser interpretado por sistemas de controle ou diretamente ser executado. Além disso, ele pode ser salvo como um projeto de inicialização na memória de parametrização do sistema de controle. Se o sistema de controle for reiniciado, o código da máquina é carregado a partir da memória de parametrização e executado no sistema de controle.

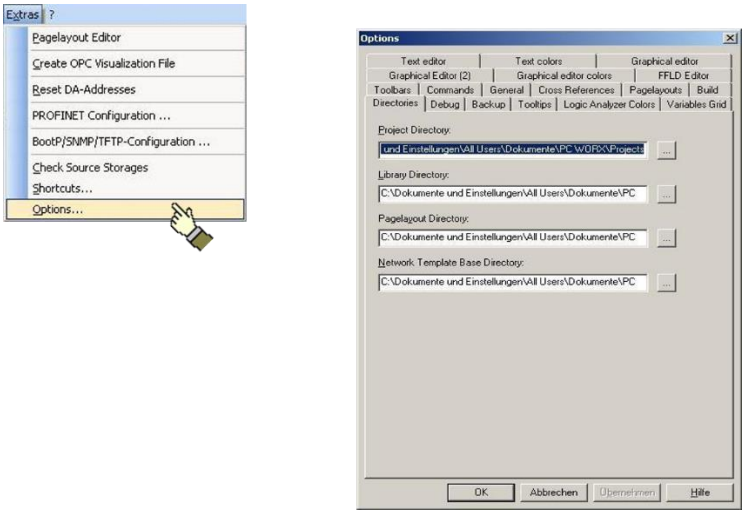
Decodable source code
Este código corresponde ao projeto arquivado (<Projectname> .zwt) salvo no disco rígido do dispositivo de programação. Este código não pode ser interpretado por sistemas de controle, serve apenas como um backup de projeto do sistema de controle.

Se forem utilizados sistemas de controle mais antigos, o procedimento descrito na próxima página para salvar o código-fonte usando um nome de arquivo padronizado deve ser seguido.



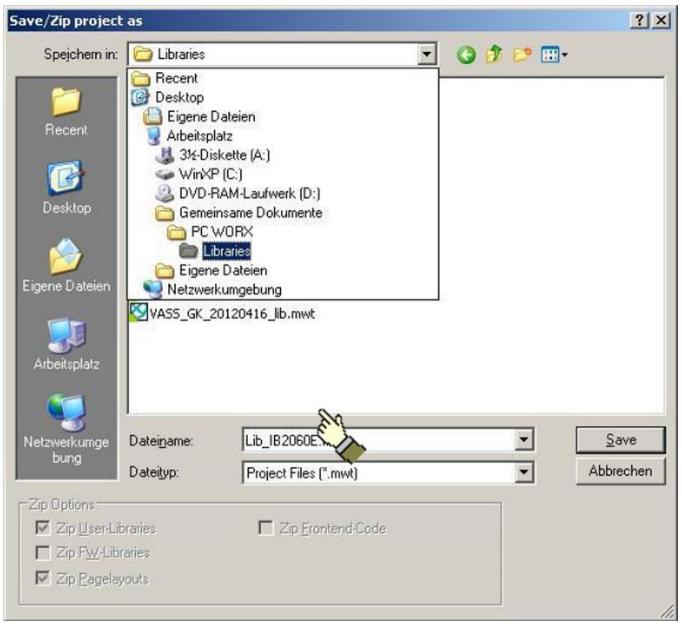
O arquivo padrão pode ser usado para salvar o código-fonte em todos os sistemas de controle (mesmo para aqueles sem um servidor FTP integrado). Através do comando *Download source* no PC WORX, o projeto atual é arquivado automaticamente e gravado na memória de parametrização com o nome ZipFile.zwt.

9. Bibliotecas



No PC WORX, o termo biblioteca refere-se a um projeto que contém componentes do programa que podem ser usados em projetos futuros. Todos os projetos desenvolvidos em PC WORX podem ser utilizados como biblioteca, ou seja, integrados em novos projetos.

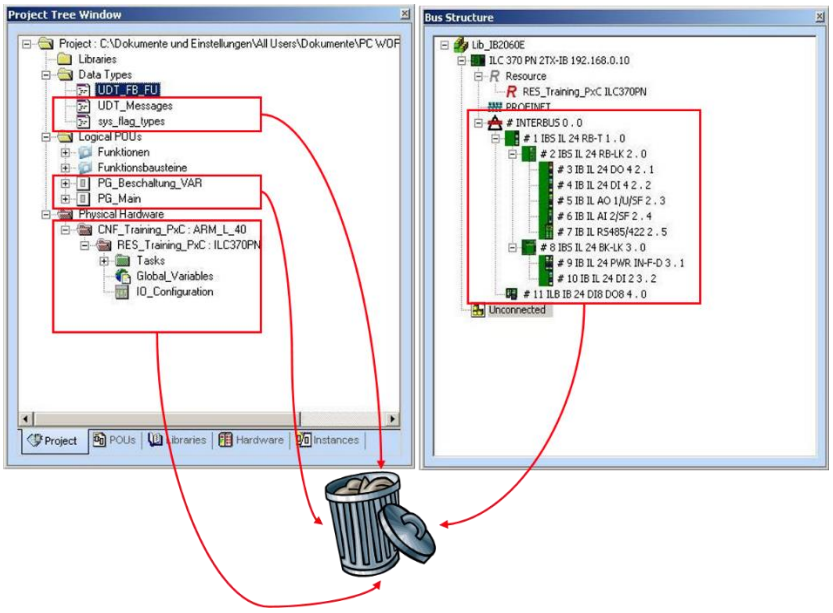
As bibliotecas são gerenciadas automaticamente pelo PC WORX e salvas em um diretório no disco rígido. Através da caixa de opções, este diretório pode ser adaptado.



Se as FU e os FB de um projeto atual devem ser disponibilizados para uso em outros projetos através de uma biblioteca, o projeto deve primeiro ser salvo com um nome de arquivo, na pasta da biblioteca.

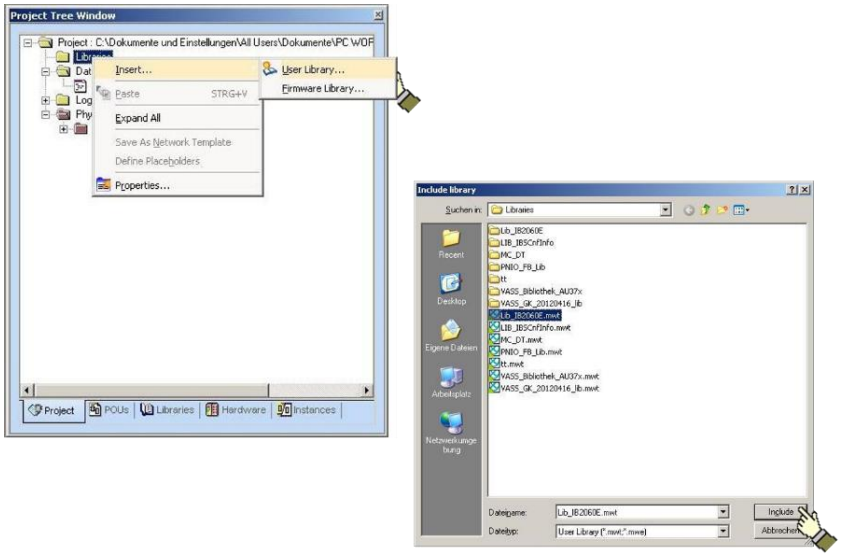
Para poder acessar as bibliotecas, através da caixa de diálogo *inserir*, o diretório selecionado anteriormente através da caixa de opções deve ser usado como o local de armazenamento.

Se as mudanças forem implementadas no projeto antes de serem salvas com o novo nome, os elementos excluídos serão irreversivelmente perdidos (ilustração a seguir).



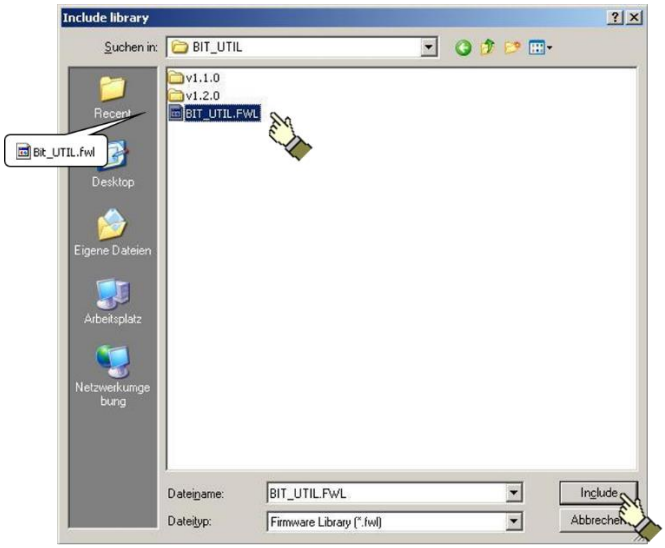
Em princípio, uma biblioteca não precisa ser criada. Em muitos casos, no entanto, é útil remover componentes não-específicos da biblioteca de um projeto. Estes incluem a configuração do barramento, bem como a estrutura do controlador, do hardware da árvore do projeto.

Como a tarefa principal das bibliotecas é fornecer FU, FB e tipos de dados correspondentes definidos pelo usuário, também é para as POUs programas.

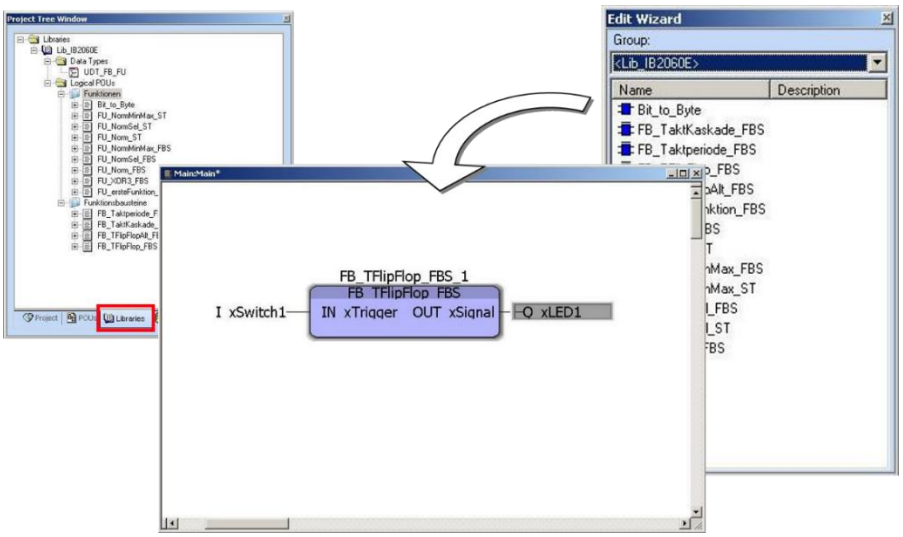


Através do menu de contexto da árvore do projeto, uma biblioteca pode ser incluída no projeto atual. Você pode incluir bibliotecas definidas pelo usuário.

Ao inserir uma biblioteca de usuários, o sistema acessa o diretório da biblioteca onde os projetos extraídos estão localizados (* .mwt), conforme ilustração a seguir.



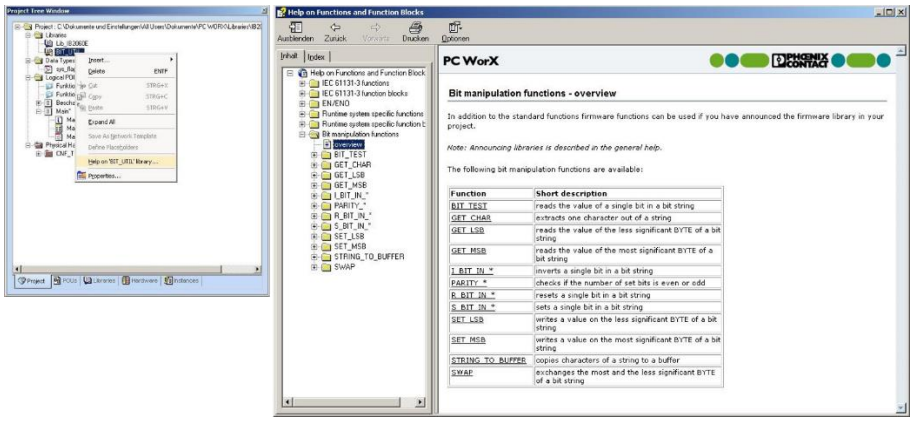
Ao incluir uma biblioteca de firmware, o PC WORX acessa o diretório fixo onde os diretórios adicionais para bibliotecas de firmware estão localizados.



As bibliotecas incluídas podem ser visualizadas através da guia bibliotecas na árvore do projeto, ou através do assistente de edição (ilustração acima). Cada biblioteca está disponível como um grupo no assistente de edição. Como padrão, os blocos de uma biblioteca que são usados em um projeto aparecerão em azul.

Uma biblioteca só pode ser incluída em um projeto após a compilação bem-sucedida do projeto da biblioteca (ilustração acima). Se necessário, o projeto correspondente no diretório da biblioteca deve ser aberto e compilado.

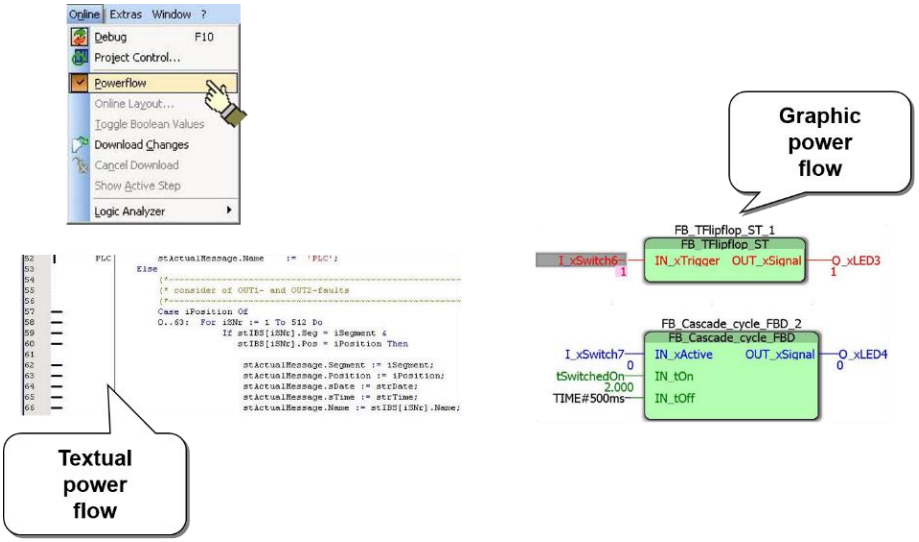
Você pode usar a caixa de diálogo de propriedades da biblioteca para verificar qual biblioteca está incluída no projeto.



Ao contrário das bibliotecas definidas pelo usuário, as bibliotecas de firmware possuem uma função de ajuda central, que pode ser chamada através do menu de contexto.

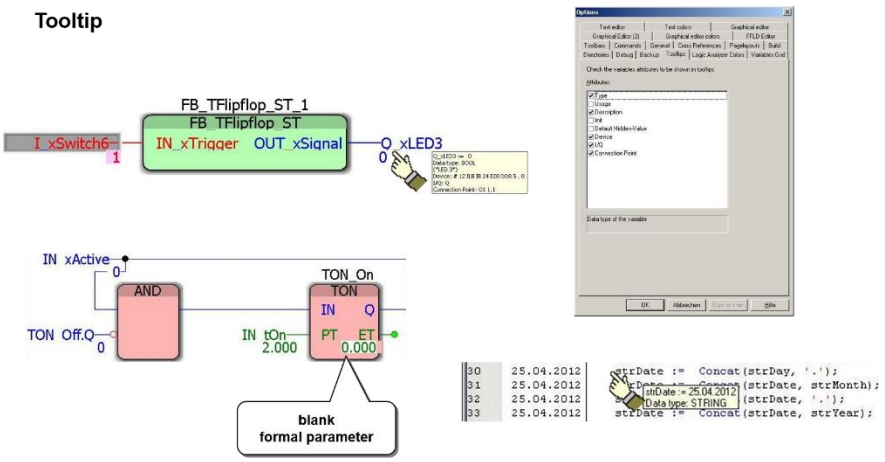
10. Testes e Debugging

Na visualização de status, você pode selecionar o formato de exibição de todas as planilhas através da caixa de diálogo de *debugging*.



Ao ativar o status *debugging*, o fluxo de energia (*Powerflow*) pode ser implementado em planilhas gráficas e de códigos de texto que estão online. Uma barra horizontal ou vertical mostra se a parte do programa correspondente está sendo processada (ilustração acima, apenas para PC WORX Professional).

Ativar o status *debugging* pode levar a um tempo de ciclo mais longo e, portanto, também a uma carga maior no processador. Durante a operação, é absolutamente necessário observar a mensagem de aviso, pois a ativação do status *debugging* pode levar a uma parada do CLP.



Além das exibições oferecidas pelos editores gráficos e textuais no status da planilha, informações adicionais podem ser exibidas quando o ponteiro do mouse é movido sobre elementos. Através da caixa de opções, você pode selecionar os detalhes de uma variável ou uma instância do FB que deve ser exibida.

Para poder exibir todos os detalhes selecionados, pressione o botão F12.