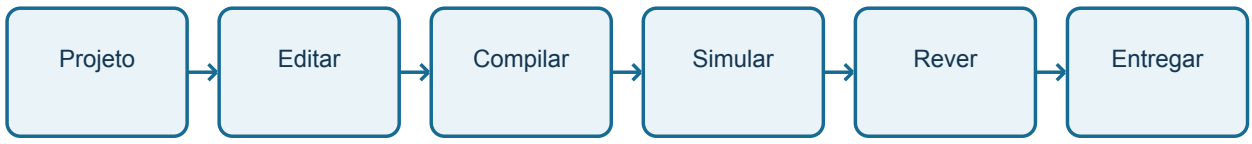
COOLMAY PLC STUDIO

Manual de programação

Programar, verificar e entregar

Guia prático detalhado

pt-BR · Edição: 30/10/2026



Índice

1. PRIMEIRO PROJETO 4

1.1 Roteiro de criação e organização 5

1.2 Símbolos, escopos e propriedade da memória 5

1.3 Salvar, reabrir e conferir 5

1.4 Tutorial: criar a unidade do primeiro ensaio 6

2. ENDEREÇAMENTO E SCAN 6

2.1 Como ler o mapa de dispositivos 6

2.2 Ciclo, ordem e estado inicial 7

2.3 Palavras, pares e retenção 7

3. LADDER / IL — PRIMEIRO ENSAIO OFFLINE 8

3.1 Série, paralelo e agrupamento — três ensaios 9

3.2 Pilha e múltiplas saídas 9

3.3 Memorização e aritmética com resultado conhecido 10

3.4 Temporização e contagem verificáveis 10

4. TEXTO ESTRUTURADO — CONTRATO IMPLEMENTADO 12

4.1 Escrever declarações e condições 13

4.2 CASE com resultados de referência 13

4.3 FOR é expansão limitada, não espera 14

4.4 Limites, chamadas e exercício completo 14

5. BLOCOS DE FUNÇÃO 15

5.1 Interface, instância e alocação 15

5.2 Parâmetros e substituição 15

5.3 Ler TON, TOF e CTU pelo comportamento atual 15

5.4 Roteiro de aceitação de uma instância 16

6. COMPILAR, SIMULAR E DIAGNOSTICAR 16

6.1 Três etapas de verificação 16

6.2 Scans manuais e tabela de observação 16

6.3 Trace e resultados reproduzíveis 17

6.4 Uso dos comandos de edição 17

6.5 Problema observado e primeira correção 17

7. CONECTAR, LER E GRAVAR 18

7.1 Preparar a conexão e preservar o original 18

7.2 Preparar, revisar, transferir e verificar 18

7.3 RUN, STOP e observação de estado 19

7.4 Recuperação após falha 19

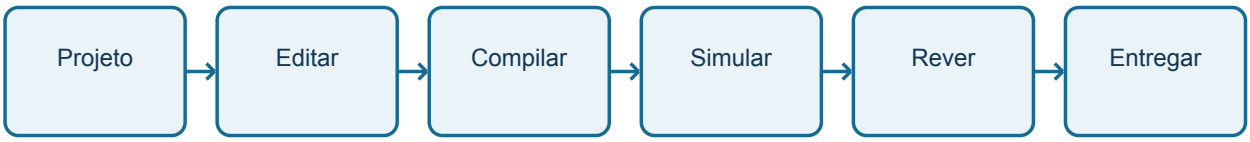
8. SALVAR E ENTREGAR 19

8.1 Dossiê de entrega e rastreabilidade 19

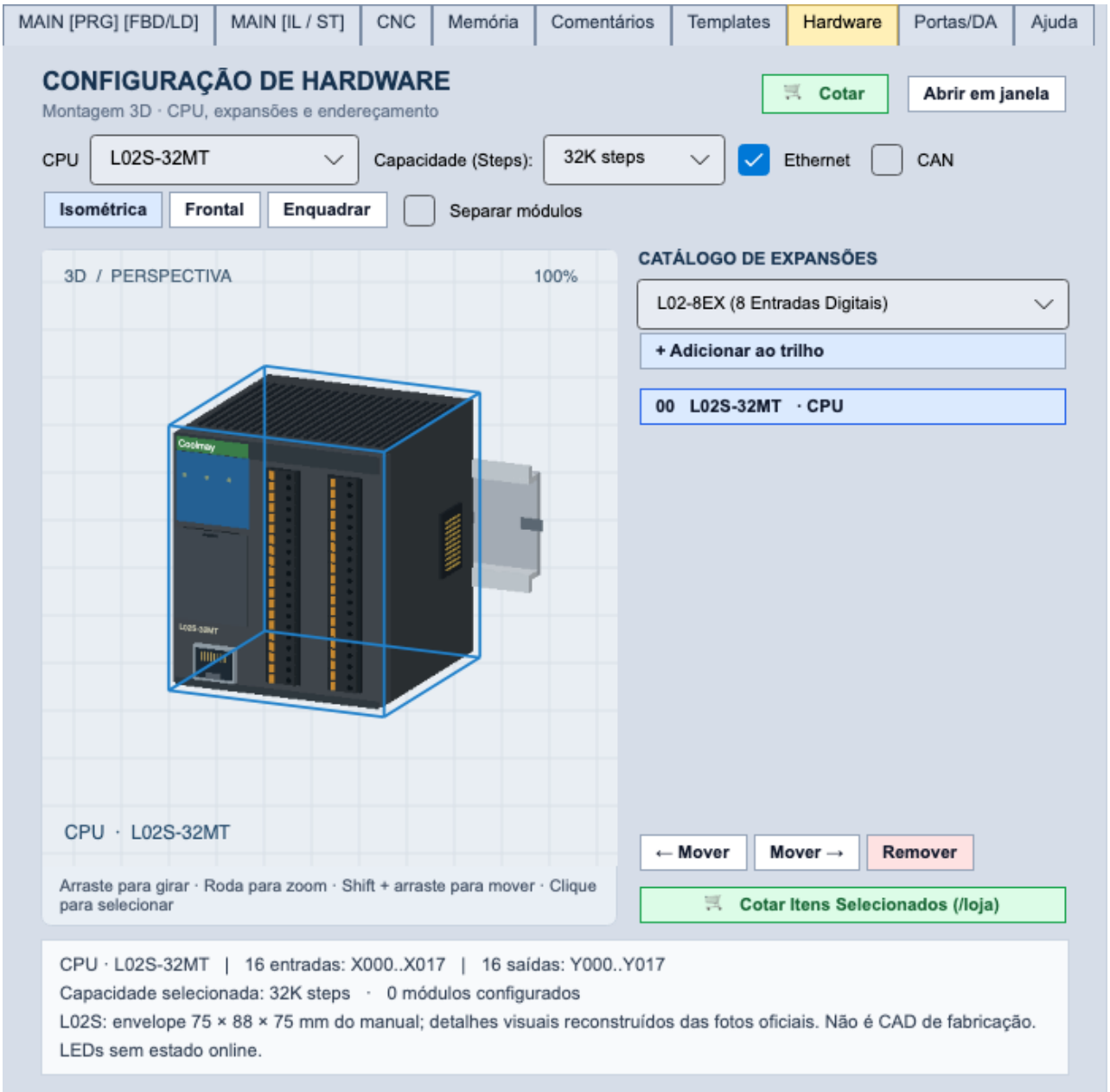
8.2 Matriz mínima de aceitação deste manual 20

8.3 Conferência antes de arquivar 20

1. PRIMEIRO PROJETO



Esquema conceitual, não medição de tempo do CLP.



Exemplo didático na IDE; sem ligação a equipamento.

Crie ou abra um projeto .cmproj. Selecione a CPU e monte os cartões no configurador de hardware. Confira a ordem dos slots e o mapa X/Y/AD/DA antes de usar os endereços. Salve uma cópia antes de importar ou

substituir programas.

Na árvore de programas, crie uma unidade e escolha sua linguagem. IL/Ladder usa a lista de instruções como fonte; o diagrama de uma unidade ST é gerado e deve ser alterado pela fonte ST. Uma unidade de programa não é o mesmo que uma instância de FB.

Edite Global1 para tags compartilhadas e os rótulos locais para cada unidade. Associe símbolos a dispositivos explícitos; um nome de tag não reserva memória física automaticamente. Evite dois escritores para a mesma saída.

1.1 Roteiro de criação e organização

Comece por um projeto de ensaio separado da instalação. No menu Projeto, use Novo Projeto Estruturado / Simples; escolha a CPU indicada na identificação do equipamento. Abra o configurador e adicione somente os módulos realmente presentes, na ordem física da esquerda para a direita. O desenho do rack e seu mapa pertencem ao projeto; selecionar outro modelo pode alterar a lista de dispositivos válidos. Confira de novo o mapa após qualquer alteração de hardware, mesmo quando os nomes das tags permanecem iguais.

Organize as unidades de programa por responsabilidade: aquisição de sinais, permissivos, sequência, comandos e diagnóstico. No gerenciador de programas locais, confira o nome, a linguagem, a habilitação e a ordem de execução de cada unidade. O programa que aparece no editor é a unidade ativa; compilar o projeto pode envolver outras unidades habilitadas. Abrir uma aba ou selecionar uma pasta não altera por si só a ordem do ciclo. Evite criar uma segunda unidade que reescreva comandos já calculados pela primeira.

1.2 Símbolos, escopos e propriedade da memória

Faça uma tabela antes de escrever a lógica: nome | tipo | dispositivo | significado | escritor. Por exemplo, Request | BOOL | X000 | pedido; Inhibit | BOOL | X001 | bloqueio; Ready | BOOL | M50 | permissão calculada; State | INT | D40 | estado. Entradas são leituras do processo; uma memória M usada como comando interno permite ensaiar sem atribuir uma saída Y. Um comentário descreve a função, mas não muda o tipo nem o dispositivo.

Global1 serve para símbolos compartilhados. Uma declaração local pertence à unidade que a contém; não confunda visibilidade do nome com isolamento do endereço. Duas tags locais associadas ao mesmo M50 continuam acessando o mesmo bit. Nomes de símbolos têm limite de 32 caracteres e cada escopo admite até 2048 símbolos. Uma associação AT liga o nome a um dispositivo já existente; não cria uma área de memória nova. Antes de reservar buffers, procure também usos pelo endereço literal, não somente pelo nome simbólico.

1.3 Salvar, reabrir e conferir

Use Salvar Projeto Como para estabelecer uma cópia de trabalho .cmproj identificável. Depois use Salvar Projeto para registrar as alterações. Reabra a cópia e confira unidades, linguagem, tags e rack antes de considerá-la um ponto de recuperação. O arquivo CSV de instruções é um intercâmbio do programa ativo; não substitui o projeto completo, as demais unidades, a biblioteca e a configuração. Antes de importar um CSV, salve uma cópia e confira se está na unidade pretendida.

Resultado esperado desta etapa: o projeto reaberto apresenta o modelo escolhido, a mesma ordem de módulos e a mesma associação das tags; uma compilação identifica a unidade de origem dos erros. Se isso falhar, resolva a organização antes de escrever a sequência de máquina. Mantenha uma versão anterior que possa ser comparada no comando Comparar projeto atual com arquivo.

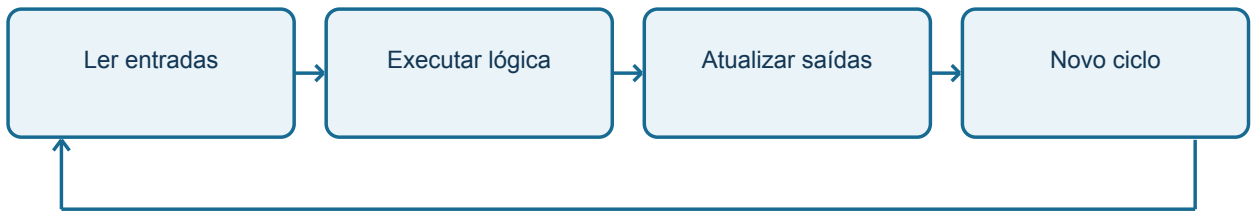
1.4 Tutorial: criar a unidade do primeiro ensaio

Passo 1: salve a cópia de trabalho e selecione a pasta Program na árvore. Use seu menu de contexto para criar um novo programa. No diálogo, preencha Nome do Novo Programa com TESTE_PERMISSO e escolha a Linguagem Fixa do POU. Confirme em Criar Programa e Abrir. O programa novo deve aparecer na pasta e ficar ativo; confira seu nome antes de colar código.

Passo 2: para o ensaio ST, escolha ST e copie somente o programa mínimo do capítulo 4, desde VAR até END_IF;. Para o ensaio IL/Ladder, use a unidade de lista de instruções e copie desde LD até END do exercício escolhido. Não cole texto explicativo nem duas instruções END intermediárias. A fonte ST deve ser editada no editor ST; não tente corrigir o diagrama gerado como se fosse a fonte original.

Passo 3: pressione F4 e leia o resultado. Se houver erro, volte à unidade e linha informadas. Se a compilação estiver correta, confira os dispositivos da saída no mapa: os exercícios usam memórias M para evitar um comando Y. Use F3 para iniciar a simulação local e faça a sequência de entradas descrita no exercício. Ao terminar, pare a simulação e salve. Reabra o projeto e confira que o mesmo código permanece na mesma unidade.

2. ENDEREÇAMENTO E SCAN



Esquema conceitual, não medição de tempo do CLP.

X e Y usam numeração octal: X007 é seguido por X010. A CPU alvo tem X000..X017 e Y000..Y017; não existem X008 ou Y009. M, S, T, C e D usam numeração decimal. K é constante decimal e H é hexadecimal.

O programa é executado na ordem das instruções. Uma atribuição posterior pode sobrescrever uma anterior no mesmo scan. OUT segue sua condição; SET memoriza e RST desliga. A retenção efetiva depende da configuração do CLP, não apenas do nome de uma variável.

INT ocupa uma word D de 16 bits com sinal. Pares D são usados por operações de 32 bits. Confira destinos adicionais de MUL/DIV, CMP/ZCP e blocos de comunicação. Não use M1400..M1535 para comunicação externa com programas ST gerados: o compilador pode reservar bits auxiliares nessa área.

2.1 Como ler o mapa de dispositivos

Para contar pontos X/Y, converta a posição para octal: os primeiros 16 pontos são 000..007 e 010..017. O ponto depois de 017 é 020. X010 representa o nono ponto, não o décimo primeiro. D10, por outro lado, é o registrador decimal 10. Esta diferença explica por que X008 é inválido mesmo quando existem mais de oito entradas. Use o mapa calculado do rack para distinguir pontos da CPU e pontos dos módulos. O exemplo L02S-32MT deste manual usa 16 entradas e 16 saídas; não estenda essa quantidade aos outros modelos.

Tabela de uso: X=entrada discreta; Y=saída discreta; M=bit interno; S=bit de estado; D=word de dados; T=temporizador; C=contador. O contato T0 ou C0 representa uma condição lógica; seu valor corrente é uma informação numérica distinta. K3 indica três em decimal e H0003 o mesmo padrão hexadecimal. Uma constante não é uma variável gravável. Os tipos definidos no editor de blocos não ampliam automaticamente os tipos aceitos pelo compilador ST principal.

2.2 Ciclo, ordem e estado inicial

Pense em uma passagem completa pelas instruções como um scan. Primeiro anote as entradas do ensaio, depois execute uma passagem e observe todas as memórias de interesse. Se uma rede calcula M20 e a seguinte lê M20, a segunda encontra o valor atualizado nessa passagem. Se duas redes escrevem a mesma bobina, o resultado depende da ordem. Para uma saída calculada continuamente, dê a ela um único escritor e calcule todos os permissivos antes da bobina.

Uma instrução OUT escreve verdadeiro ou falso conforme a condição. SET só liga quando sua condição é verdadeira; RST só limpa quando habilitado. No exemplo IL_LATCH, se X000 e X001 estiverem verdadeiros no mesmo scan, a limpeza ocorre depois da ligação e M30 termina falso. Um comando mantido por SET precisa de um caminho de reset definido. Documente também o que deve acontecer na primeira execução, depois de STOP/RUN e após perda de energia; são situações diferentes e devem ser verificadas separadamente.

2.3 Palavras, pares e retenção

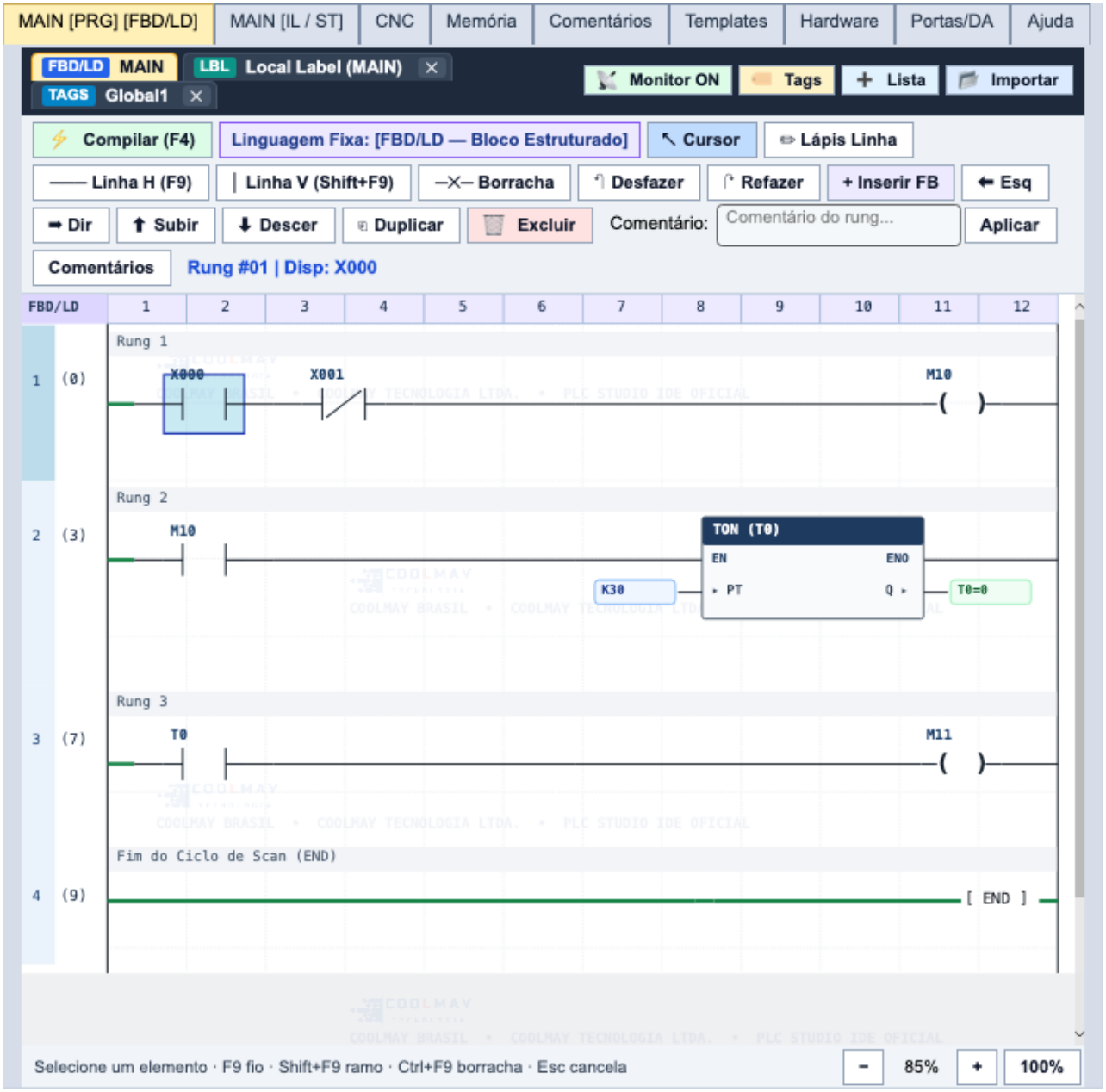
Uma word INT de 16 bits representa -32768..32767. Uma operação de 32 bits usa dispositivos consecutivos; reservar D100 não é suficiente quando a instrução também escreve D101. Destinos de multiplicação, divisão, comparação e comunicação podem ocupar vários pontos. Faça a reserva pela largura real da instrução e confira o manual do bloco. Valores de engenharia exigem unidade e escala explícitas:1000 pode representar 1000 unidades,100,0 ou 10,00; o registrador sozinho não define essa interpretação.

No modelo de retenção do simulador, M384..M7679, S500..S4095 e D128..D7999 são retentivos; T246..T255, C100..C199 e C220..C255 também recebem tratamento retentivo. A limpeza de memória não retentiva preserva essas áreas e os dispositivos de sistema. Isto descreve a simulação: confirme parâmetros e comportamento do equipamento para a instalação. Não guarde estados transitórios em área retentiva por conveniência; uma partida após energia pode recuperar um comando antigo.

3. LADDER / IL — PRIMEIRO ENSAIO OFFLINE



Contato normalmente aberto: M10 segue X000.



Exemplo didático na IDE; sem ligação a equipamento.

Insira um contato e uma bobina ou digite:

```
LD X000
OUT M10
END
```

Compile. No simulador, alterne X000: M10 deve acompanhar o contato. Este exemplo usa M10 e não aciona uma saída física. Para ligar condições em série use AND; em paralelo use OR. LDI/ANI representam condição negada. Use parênteses lógicos por redes ANB/ORB quando necessário; não remova pilhas MPS/MRD/MPP sem conferir a lógica.

Para atraso de 3 segundos usando T0 com base de 100 ms:

```
LD X000
OUT T0 K30
LD T0
OUT M11
END
```

Confira a base de tempo do temporizador escolhido no manual do CLP. Para contar eventos, use OUT C0 K10, leia o contato C0 e zere por RST C0. Teste bordas, reset e reinício. Um contato invertido representa o estado lógico lido; não prova a polaridade elétrica de um botão físico.

As ferramentas de linhas e ramos alteram lógica. Se a IDE recusar uma rede compartilhada ou ambígua, edite IL. O histórico gráfico deve acompanhar a unidade ativa; confira a fonte depois de desfazer/refazer.

3.1 Série, paralelo e agrupamento — três ensaios

IL_SERIES calcula X000 AND X001 AND NOT X002. Teste todas as oito combinações de entrada. M20 só deve ligar com X000=1, X001=1 e X002=0. Não use o nome físico de um botão para adivinhar a condição; confirme qual valor é lido quando ele é acionado.

```
LD X000
AND X001
ANI X002
OUT M20
END
```

IL_GROUP calcula (X000 OR X001) AND (X002 OR X003). Cada LD inicia um bloco lógico; ANB une os dois resultados em série. ORB faria a união em paralelo. Desenhar uma linha entre ramos muda essa expressão, portanto compare a fonte antes e depois da edição. Teste as 16 combinações: cada um dos dois grupos precisa conter pelo menos uma entrada verdadeira.

```
LD X000
OR X001
LD X002
OR X003
ANB
OUT M21
END
```

3.2 Pilha e múltiplas saídas

MPS guarda o resultado de uma condição; MRD relê sem removê-lo e MPP relê e remove. IL_STACK distribui X000 como condição comum para três saídas. Com X000=0, todas permanecem falsas. Com X000=1, M22 acompanha X001, M23 acompanha X002 e M24 acompanha X003. A alteração de uma saída não deve alterar a condição dos outros ramos. O analisador admite oito blocos LD pendentes e onze memórias MPS; MRD/MPP sem MPS e MPS sem MPP são erros de estrutura. Não tente resolver esses erros acrescentando uma bobina arbitrária.

```
LD X000
MPS
AND X001
OUT M22
MRD
AND X002
OUT M23
MPP
AND X003
OUT M24
END
```

3.3 Memorização e aritmética com resultado conhecido

Em IL_LATCH, aplique X000 por um scan e depois retire-o: M30/M31 continuam ligados. Aplique X001: ambos desligam. Aplique os dois juntos: ambos terminam desligados. Este é um teste de prioridade determinado pela ordem das instruções, não uma função de segurança física.

```
LD X000
SET M30
LD X001
RST M30
LD M30
OUT M31
END
```

IL_WORD produz D20=12, D21=15 e D22=10 em cada scan em execução. MOV copia; ADD e SUB gravam o destino indicado. M8000 é o bit de execução modelado pelo simulador. A mesma aritmética aplicada a valores de processo precisa de limites e tratamento de overflow adequados. Não suponha que a exibição de um número positivo significa ausência de overflow em operações anteriores.

```
LD M8000
MOV K12 D20
ADD D20 K3 D21
SUB D21 K5 D22
END
```

3.4 Temporização e contagem verificáveis

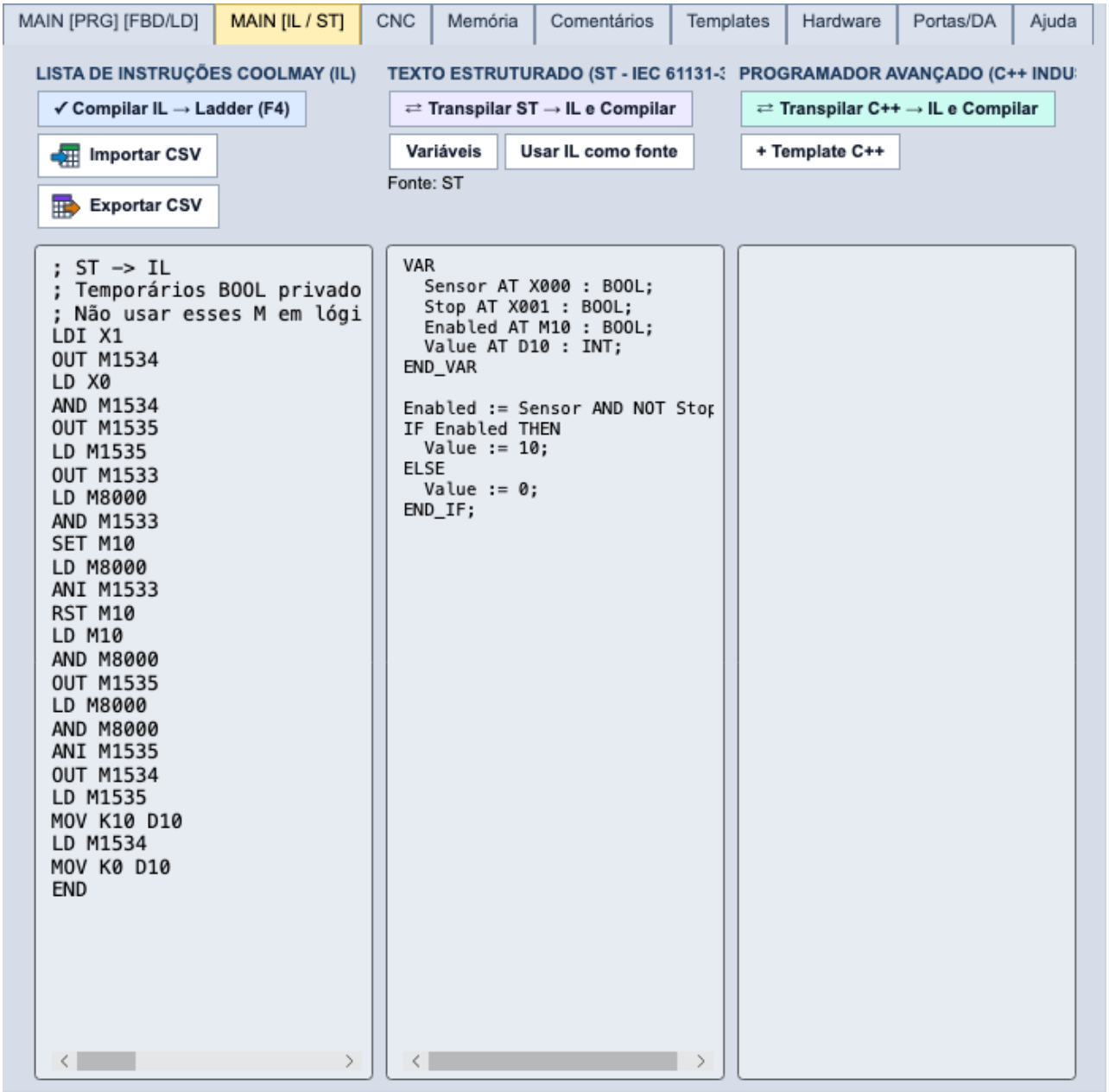
IL_TIMER usa T0 com base de 100 ms e K3: após 300 ms acumulados habilitados, M40 liga. Com X000 falso, o temporizador não retentivo e M40 retornam a zero/falso. Teste 100 ms, mais 100 ms e mais 100 ms em scans separados, depois desligue X000. O tempo acumulado depende do tempo fornecido ao simulador, não da velocidade com que se clica na tela.

```
LD X000
OUT T0 K3
LD T0
OUT M40
END
```

IL_COUNTER exige três bordas de subida de X000. Manter X000 ligado durante vários scans não deve produzir três eventos. X001 limpa o contador e inibe a contagem no mesmo scan; ao retirar o reset, produza três novos pulsos separados por scans com X000 falso. M41 deve ligar no terceiro e apagar no reset. Confira o valor corrente de C0 além do contato de conclusão.

```
LD X001
RST C0
LD X000
ANI X001
OUT C0 K3
LD C0
OUT M41
END
```

4. TEXTO ESTRUTURADO — CONTRATO IMPLEMENTADO



Exemplo didático na IDE; sem ligação a equipamento.

Exemplo mínimo:

```
VAR
  Sensor AT X000 : BOOL;
  Ligado AT M10 : BOOL;
  Valor AT D10 : INT;
END_VAR
Ligado := Sensor AND NOT X001;
IF Ligado THEN
  Valor := 10;
ELSE
  Valor := 0;
END_IF;
```

São aceitos BOOL/INT, VAR e VAR_GLOBAL com AT explícito, expressões booleanas, relações, IF/ELSIF/ELSE, CASE e SEL posicional restrito. A atribuição numérica aceita valor ou uma soma/subtração; não escreva expressões gerais com * ou / supondo suporte do compilador ST principal.

FOR exige contador e limites INT, limites/passo literais, até 256 iterações por laço e expansão total limitada. Não há alocação automática, inicializadores VAR, arrays, WHILE/REPEAT ou chamadas gerais de funções/FB nesse compilador. Chamadas diretas implementadas: SET, RST, PLSY, DRVI e ADPRW com operandos validados. TON_T0(...) não é uma chamada suportada pelo compilador ST principal.

O editor de FB possui seu próprio caminho de geração; um corpo ST no catálogo não demonstra que a mesma sintaxe seja aceita como programa ST. Compile a unidade real e revise o IL resultante.

4.1 Escrever declarações e condições

Declare cada símbolo com tipo e dispositivo explícitos. BOOL usa um dispositivo de bit; INT usa uma word D. Termine declarações e atribuições com ponto e vírgula, feche VAR com END_VAR e IF com END_IF;. Inicializadores na declaração e alocação automática não fazem parte deste contrato. Se uma tag existir também na tabela de símbolos do projeto, confira se a declaração textual aponta ao mesmo dispositivo e tipo.

A precedência é: parênteses, NOT, relações numéricas, igualdade/desigualdade, AND, XOR e OR. Use parênteses quando a intenção não for evidente. BOOL e INT não são intercambiáveis: escreva D0 > 0 para obter uma condição, em vez de usar D0 como contato. IF executa o primeiro ramo verdadeiro; ELSIF é alternativa, não um segundo IF independente. Sem ELSE, destinos não escritos conservam seu valor anterior; se precisa de falso/zero em toda passagem, atribua-o explicitamente.

4.2 CASE com resultados de referência

ST_CASE escolhe a saída a partir de Mode:0 resulta 0;1 ou 2 resulta 10;3 a 5 resulta 20; qualquer outro INT resulta-1. Listas usam vírgula e intervalos inclusivos usam dois pontos consecutivos. Os ramos não continuam para o ramo seguinte. O compilador captura a seleção antes de executar o corpo; mudar Mode dentro de um ramo não executa outro ramo nessa mesma seleção.

```
VAR
  Mode AT D20 : INT;
  Result AT D21 : INT;
END_VAR
CASE Mode OF
  0: Result := 0;
  1, 2: Result := 10;
  3..5: Result := 20;
ELSE
  Result := -1;
END_CASE;
```

Teste Mode=-1,0,1,2,3,5,6. O limite por CASE é 128 ramos e 512 rótulos/intervalos; intervalos não são expandidos um a um. Evite sobreposições e mantenha ELSE para uma condição inesperada quando a aplicação precisa de um valor definido. O dispositivo Mode deve pertencer a um INT, não a um bit.

4.3 FOR é expansão limitada, não espera

ST_FOR soma 1+2+3+4: Sum termina 10 e Index termina 5. A soma é reiniciada em cada scan, então não cresce para 20 no scan seguinte. FOR exige contador INT, início/fim/passo literais INT16 e BY diferente de zero. Há até 256 iterações por laço e 8192 operações locais expandidas no conjunto com laços. O valor do contador após o último incremento também deve caber em INT16. O corpo não pode modificar o contador nem um alias dele.

```
VAR
  Index AT D30 : INT;
  Sum AT D31 : INT;
END_VAR
Sum := 0;
FOR Index := 1 TO 4 BY 1 DO
  Sum := Sum + Index;
END_FOR;
```

Não use FOR para aguardar sensor, temporizador ou comunicação: as iterações pertencem ao mesmo scan e não representam passagem de tempo físico. Modele esperas por estados conservados entre scans. WHILE e REPEAT não são aceitos. Um erro de limite pede simplificar a lógica, dividir responsabilidades ou usar uma máquina de estados, não aumentar silenciosamente o número de instruções.

4.4 Limites, chamadas e exercício completo

A fonte ST admite até 200.000 caracteres e 20.000 tokens. Os escopos mantêm o limite de 2048 símbolos. Atribuições numéricas aceitam valor simples ou uma adição/subtração; chamadas gerais, arrays e expressões numéricas arbitrárias não são suportadas. As chamadas diretas implementadas SET, RST, PLSY, DRVI e ADPRW têm validação própria de operandos. Aceitar uma chamada na linguagem não significa simular seu efeito físico.

ST_INTERLOCK é um programa completo de permissivo. Teste Request/Inhibit=0/0,1/0,0/1,1/1: Ready e State devem resultar respectivamente falso/0, verdadeiro/1, falso/0, falso/0. Os identificadores são mantidos em todos os idiomas para que o mesmo ensaio possa ser repetido. Inspecione o IL gerado e procure as referências a M50/D40 antes de integrar o exemplo ao projeto.

```
VAR
  Request AT X000 : BOOL;
  Inhibit AT X001 : BOOL;
  Ready AT M50 : BOOL;
  State AT D40 : INT;
END_VAR
Ready := Request AND NOT Inhibit;
IF Ready THEN
  State := 1;
ELSE
  State := 0;
END_IF;
```

5. BLOCOS DE FUNÇÃO

Abra a aba Manual dos Blocos para o catálogo da biblioteca atual: descrição, pinos, tipos, valores padrão, variáveis internas, IL e corpo ST disponíveis. Crie uma instância com nome único; associe suas entradas e saídas a tags do programa. Separe memória de instâncias e reserve pares e buffers completos.

As tabelas VAR_INPUT, VAR_OUTPUT e VAR definem a interface e o estado interno; os endereços físicos são associados ao instanciar. Para editar um bloco protegido use o fluxo normal de senha; os manuais não expõem seu corpo protegido.

O catálogo inclui observações sobre os limites dos modelos TON/TOF/TP/CTU e do status Modbus. Não suponha comportamento IEC pelo nome do bloco.

5.1 Interface, instância e alocação

Uma definição de bloco descreve a operação reutilizável; uma instância é seu uso em um programa. Crie a definição pela pasta de blocos/estúdio e uma instância com nome exclusivo. Preencha VAR_INPUT com sinais e parâmetros que o chamador fornece, VAR_OUTPUT com os resultados que ele lê, e VAR com estado auxiliar. Tipos, comentários e valores padrão ajudam a preencher a chamada; não garantem que a associação de memória final seja exclusiva.

Antes de inserir duas instâncias, faça uma tabela de ocupação: instância | entradas | saídas | temporizador/contador | palavras auxiliares | bits internos. Duas instâncias usando T0 compartilham o mesmo temporizador. Um parâmetro chamado TimerDev não cria outro timer. Troque os dispositivos padrão quando eles coincidirem, incluindo memórias escondidas nos operandos do IL e pares D consecutivos. A referência ilustrada mostra pinos e tipos reais do catálogo; consulte também seu IL.

5.2 Parâmetros e substituição

Um pino BOOL normalmente recebe bit ou tag de bit; um pino de preset recebe constante ou word admitida pela instrução; TimerDev/CounterDev identificam dispositivos. Por exemplo, K30 é um preset decimal, enquanto D30 é memória variável. O nome PT_100 ms informa uma unidade do modelo; confirme o temporizador escolhido antes de converter segundos. Se deseja 3 scom base 100 ms, o preset é 30. Com outra base, o mesmo número representa outra duração.

Compile após substituir parâmetros. Confira todas as instruções emitidas, não só a primeira linha. Um pino de saída pode ser usado por outra rede e uma instrução pode produzir mais de uma word. Reserve um único escritor para cada comando; use variáveis internas para combinar resultados antes da saída física. Para bloco protegido, a interface permanece utilizável, mas o corpo e as variáveis internas não aparecem no manual; use o fluxo de senha autorizado para edição.

5.3 Ler TON, TOF e CTU pelo comportamento atual

FB_TON_DELAY executa TMR e depois OUT Q sem carregar o contato TimerDev. Assim Q pode refletir a condição de entrada em vez da conclusão do atraso. O padrão IL_TIMER do capítulo 3 lê T0 explicitamente antes de escrever M40. Compare entrada, valor corrente do timer, contato T0 e saída em cada scan: olhar somente Q não demonstra o atraso.

FB_TOF_OFF_DELAY escreve Q dependente de IN; quando IN cai, Q cai imediatamente no modelo IL. Ele não mantém a saída durante PT após a queda. FB_TP_PULSO também depende de IN: retirar o sinal encerra o pulso. FB_CTU_CONTADOR usa RESET como inibição e não executa RST CounterDev; Q não é

carregado do contato de conclusão. O padrão IL_COUNTER mostra reset e leitura de conclusão explícitos. Esses limites são funcionais, não diferenças de tradução; teste o bloco real antes de adotá-lo.

5.4 Roteiro de aceitação de uma instância

Comece com entradas falsas e observe saídas/estado interno. Ative somente uma entrada por vez; depois teste combinações, mudanças na borda, reset e permanência por vários scans. Para dois usos do mesmo bloco, altere uma instância e confirme que a outra não muda por compartilhamento acidental. Registre o dispositivo de cada pino, valores iniciais e resultado esperado. Um modelo que compila não está automaticamente aprovado para temporização, movimento, PID ou transação de rede.

6. COMPILAR, SIMULAR E DIAGNOSTICAR

Compile após cada alteração e resolva o primeiro diagnóstico na linha/coluna indicada. Confira tags não resolvidas, tipos, faixas, END e possíveis sobreposições. Faça ensaios em scan manual antes de execução contínua. Observe valores iniciais, entradas desligadas, transições e reset.

O simulador não certifica frequências de pulso, tempo físico de scan, sensores, estabilidade de PID ou troca Modbus. A prévia CNC mostra geometria; não é validação de interpolação nem movimento físico. C++ é um subconjunto voltado ao CLP e não um compilador C++ de uso geral.

6.1 Três etapas de verificação

Primeiro compile a fonte atual e leia os diagnósticos: sintaxe, tipos, símbolos, dispositivos e estrutura lógica. Depois simule os resultados previstos, usando entradas e estado inicial registrados. Por último, quando autorizado e disponível, confira a instalação real com o modelo correto. Estas etapas respondem perguntas diferentes. Um compilador pode aceitar instrução que o simulador não implementa; o simulador verifica seu conjunto suportado antes de executar e deve indicar a interrupção, em vez de tratar operação desconhecida como instrução vazia.

Ao corrigir um erro, comece pelo primeiro diagnóstico. Um END_IF ausente pode gerar vários erros posteriores. Para endereço inválido, diferencie erro octal de ponto ausente no rack. Para símbolo não resolvido, confira unidade ativa, escopo, grafia e dispositivo. Para tipos incompatíveis, decida se precisa de bit, comparação numérica ou word; não mude o tipo apenas para esconder o diagnóstico.

6.2 Scans manuais e tabela de observação

Monte uma tabela: passo | entrada antes do scan | estado anterior | tempo transcorrido | resultado esperado | resultado observado. Para IL_LATCH use cinco passos: tudo falso; pulso de start; start retirado; pulso de reset; ambos verdadeiros. Para IL_TIMER controle intervalos de 100 ms; para IL_COUNTER separe pulsos por pelo menos um scan falso. Reinicie o estado entre ensaios que exigem memória zerada. A ordem da sequência faz parte do teste.

Monitore tanto símbolos quanto dispositivos. Uma tag com valor inesperado pode estar ligada a outro endereço; comparar a tabela de símbolos com o mapa ajuda a encontrar isso. Observe também destinos auxiliares e condições antes da bobina. Em execução contínua, uma transição curta pode ocorrer entre atualizações visuais; use scans manuais e o recurso de trace quando aplicável. Não trate uma tela aparentemente estável como prova de ausência de transições.

6.3 Trace e resultados reproduzíveis

Escolha poucos sinais que expliquem a causa: entrada, permissivo, estado, timer/contador e comando. Informe duração/intervalo e condição de captura disponíveis na interface e salve o resultado de interesse. Um trace deve indicar se a fonte é simulador ou leitura online, a unidade e a revisão do programa. Se o valor só muda quando outra unidade executa, procure todos os escritores pela referência cruzada. Acrescentar mais pontos sem hipótese de diagnóstico torna a leitura mais difícil.

Para reproduzir falha, preserve a menor fonte que ainda falha, o rack, as tags, a sequência de entradas e o texto completo do diagnóstico. Não remova a rede defeituosa e declare o ensaio aprovado porque a mensagem desapareceu. Compare resultado esperado com observado em cada etapa; um teste de compilação sozinho não prova a sequência funcional.

6.4 Uso dos comandos de edição

Com o foco no contexto adequado, F3 alternar simulação, F4 compilar, F5 contato normal, Shift+F5 ramo OR, F6 contato negado, Shift+F6 ramo ORI, F7 bobina e F8 instrução selecionada. F9 seleciona linha horizontal, Shift+F9 vertical e Ctrl/Meta+F9 apagamento de linha. Escape retorna à seleção. Ctrl/Meta+F abre busca; Shift+Ctrl/Meta+F abre referência cruzada. No Ladder, Ctrl/Meta+Z desfaz, Shift+Ctrl/Meta+Z ou Ctrl/Meta+Y refaz. Confira o foco antes de usar atalhos: campos de texto e diagrama têm ações diferentes. F3 controla a simulação local; não equivale ao comando RUN físico do capítulo 7.

6.5 Problema observado e primeira correção

Mensagem / sintoma	conferir	ação
X008 ou Y009 inválido	numeração octal	use a posição correta do mapa, sem converter para decimal por aparência
Símbolo desconhecido	nome, âmbito e associação	declare o símbolo ou corrija o nome na unidade certa
Tipo BOOL/INT incompatível	bit ou palavra	use comparação para transformar valor numérico em condição
END_IF ou END_CASE ausente	fechamento dos blocos	feche o bloco correspondente antes de corrigir erros derivados
MRD/MPP sem MPS	pilha do ramo	recupere a condição comum e balanceie a estrutura
Destino muda sozinho	outros escritores e alias	procure referência cruzada e compare dispositivos físicos
Timer não atrasa Q	contato lido e modelo do bloco	confira T0 e use o padrão explícito IL_TIMER
Contador conta uma vez	entrada permanece verdadeira	produza novas bordas com scans falsos intermediários
Porta ocupada	outro cliente da conexão	conclua a operação anterior normalmente e reconecte

Estas ações orientam a investigação; preserve o diagnóstico original. Se a causa não estiver demonstrada, mantenha a falha registrada em vez de declarar a função aprovada.

7. CONECTAR, LER E GRAVAR

Selecione a conexão correta. Identifique o equipamento e leia uma cópia antes de preparar gravação. A revisão deve corresponder à fonte atual e ao modelo/capacidade de programa. Revise backup, comparação, STOP e releitura quando apresentados pela IDE.

RUN e STOP são ações explícitas; confira o estado retornado. Um arquivo de captura, simulação ou compilação bem-sucedida não prova teste físico. Um backup da área de programa não é backup de toda memória retentiva, parâmetros e estado do CLP. Não desconecte durante I/O; aguarde o cancelamento concluir.

Se houver erro: preserve o diagnóstico, confira cabo/porta/modelo e a fonte selecionada; não repita gravações supondo que ACK significa verificação integral. Para redes e cartões, consulte o manual correspondente e o mapa da montagem.

7.1 Preparar a conexão e preservar o original

Identifique o equipamento pelo modelo e pela conexão disponível. Porta USB nativa e porta serial são transportes diferentes; use a opção que corresponde ao dispositivo reconhecido pela IDE. Confira se a porta selecionada pertence à CPU pretendida. Se outro aplicativo estiver usando a conexão, encerre sua operação normalmente antes de conectar; não interprete porta ocupada como programa do CLP inválido. Não escolha parâmetros de outra família apenas porque o conector se parece.

Leia primeiro as informações e a área de programa permitida e preserve a cópia original. Anote tamanho, data, modelo informado e diagnóstico; confira se a leitura terminou completa. A leitura binária de programa não reconstrói necessariamente nomes, comentários, símbolos e organização do .cmproj. Guarde também o projeto fonte correspondente. Um backup de programa não inclui por definição todos os parâmetros, memórias retentivas e estado de processo.

7.2 Preparar, revisar, transferir e verificar

A fonte selecionada deve ser a revisão que acabou de compilar. Confira unidades habilitadas, ordem, mapa, tamanho codificado e capacidade efetiva apresentada pelo equipamento. Antes de escrever, revise a comparação e as etapas mostradas pela IDE. STOP, início da transferência, envio de blocos, encerramento e releitura são ações distintas. Um ACK confirma a etapa de protocolo a que pertence; não comprova sozinho a igualdade final entre bytes preparados e lidos.

Depois da transferência, confira o relatório de comparação/releitura. Se existir diferença, preserve os arquivos e o diagnóstico; não conclua sucesso porque a conexão permaneceu aberta. Não altere o programa enquanto uma preparação/revisão está em andamento sem refazer a etapa correspondente. Não desconecte o cabo durante I/O. Cancelamento pode precisar concluir a operação pendente antes de liberar a conexão; aguarde o estado final apresentado.

7.3 RUN, STOP e observação de estado

Escolha RUN ou STOP explicitamente e leia o retorno e o estado observado. Encerrar uma transferência não é iniciar a CPU. Uma resposta positiva e uma observação posterior do estado são evidências diferentes. Não use o botão de simulação para inferir o estado da CPU real. Registre o estado anterior e posterior e a fonte da informação. Este manual e seus exemplos não relatam ensaio físico realizado pela IDE.

Na bancada, leitura, gravação de programa, escrita de memória e mudança de estado devem ter resultados separados. Se a leitura funciona mas a gravação falha, investigue o diagnóstico de gravação; se o comando é aceito mas o estado não muda, preserve ambas as evidências. Não derive versão de firmware de marcação de placa ou identificação comercial. Para limites elétricos, cabos e módulos, consulte o documento específico do equipamento.

7.4 Recuperação após falha

Anote em que fase ocorreu o erro: abrir conexão, identificar, ler, preparar, enviar, encerrar ou verificar. Preserve o diagnóstico completo e o último estado confirmado. Verifique alimentação, cabo, conexão selecionada, ocupação da porta e compatibilidade do modelo antes de repetir. Repetir escrita sem compreender a etapa que falhou pode apagar a evidência mais útil. Retome a partir de uma fonte salva e de um original preservado; não substitua o backup original pelo resultado de uma tentativa incompleta.

8. SALVAR E ENTREGAR

Salve o .cmproj com suas unidades, tags e configuração. Confira se há proteção de arquivo ou de programa/FB e mantenha as senhas no local autorizado. Exporte o IL e os diagnósticos necessários ao comissionamento. Registre quais testes foram offline e quais ocorreram no equipamento, com modelo e versão do pacote utilizado.

8.1 Dossiê de entrega e rastreabilidade

Entregue o .cmproj que reabre corretamente, seus arquivos auxiliares necessários e a revisão identificada. Inclua lista de unidades habilitadas/ordem, CPU/módulos, mapa de tags, parâmetros relevantes, dispositivos reservados pelos blocos e resultados dos ensaios. Diferencie arquivo fonte, IL gerado, bytes preparados e bytes lidos do equipamento. Eles são artefatos relacionados, mas não intercambiáveis. Mantenha o original de leitura separado da última versão editada.

Para cada alteração, explique o gatilho e o resultado esperado: por exemplo, bloquear Request quando Inhibit=1 e escrever State=0. Uma entrega verificável permite que outra pessoa repita a tabela de teste sem adivinhar valores iniciais. Comentários e nomes devem explicar o processo; mantenha números de dispositivo e tipos visíveis na documentação de interface.

8.2 Matriz mínima de aceitação deste manual

Ensaio	sequência	resultado
IL_SERIES	oito combinações X000..X002	M20=1 somente para 1,1,0
IL_GROUP	16 combinações X000..X003	M21=(X000 OR X001) AND (X002 OR X003)
IL_STACK	16 combinações	três saídas compartilham somente X000
IL_LATCH	start, retirar, reset, ambos	memoriza, mantém, limpa, limpa
IL_WORD	um scan em execução	D20=12,D21=15,D22=10
IL_TIMER	três scans 100 ms, depois IN=0	M40 liga no terceiro, depois desliga
IL_COUNTER	três bordas separadas, reset	M41 liga na terceira, limpa comX001
ST_CASE	-1,0,1,2,3,5,6	-1,0,10,10,20,20,-1
ST_FOR	dois scans independentes	Sum=10 e Index=5 em ambos
ST_INTERLOCK	quatro combinações	Ready=Request AND NOT Inhibit; State=0 ou 1

Os programas completos estão nos capítulos 3 e 4. São exercícios independentes: não cole todos em uma unidade sem reservar endereços e remover terminações intermediárias. Compile e execute cada um em um projeto de ensaio. A matriz é um ponto de partida para a lógica mostrada; acrescente falhas de sensor, reinicialização, limites numéricos e recuperação necessários ao seu processo.

8.3 Conferência antes de arquivar

Reabra a cópia entregue, compile e repita os casos de aceitação aplicáveis. Compare o mapa final com a montagem. Confira que o relatório identifica testes offline e testes no equipamento, sem atribuir resultado físico ao simulador. Registre pendências concretas: qual função falta verificar, em qual modelo e com qual resultado esperado. Mantenha senhas pelo procedimento autorizado, sem registrá-las em diagnósticos ou comentários de código. Se um bloco está protegido, preserve a interface documentada e a origem autorizada da versão.

Quando o equipamento ou a aplicação mudar, repita os testes afetados: trocar um timer exige conferir base/preset; mudar um módulo exige conferir mapa; alterar ordem de unidades exige conferir escritores; trocar um bloco exige conferir interface e memória. Atualize a revisão da fonte e do relatório juntos. A documentação deixa de representar o sistema se descreve uma versão diferente da que foi compilada ou transferida.