Universidade Federal de Minas Gerais
Escola de Engenharia
Departamento de Engenharia Eletrônica
Laboratório de Informática Industrial

AULA 11 - PROGRAMAÇÃO SOB A NORMA IEC 61131-3 (III): PROJETO FINAL EM SFC

Objetivos: Desenvolvimento de aplicações na linguagem SFC (*Structured Function Chart*) para o CLP *CompactLogix* da Roclwell Automation.

Atividades Prévias

1. Leia atenciosamente, em casa, o texto desta prática.
2. Desenvolva previamente, ainda em casa, os programas nas linguagens SFC e ST descritos no item "Atividades de Projeto e Programação" desta prática.
3. Elabore um pré-relatório manuscrito, entre 1 e 2 páginas, descrevendo resumidamente os passos a serem executados nesta aula prática e incluindo os programas desenvolvidos.
4. Ao chegar ao laboratório, caso tenha um colega de bancada, discutam entre suas respectivas soluções e decidam, em conjunto, qual deles será o adotado na prática.

Introdução

Em aplicações de automação industrial, frequentemente um projeto é quebrado em partes menores de forma a facilitar seu desenvolvimento e também permitir que diferentes equipes possam trabalhar em paralelo. Neste sentido, a linguagem SFC (*Sequential Function Chart*) é muito útil pois permite que um projeto de software seja desenvolvido na técnica *top-down*, na qual parte-se de uma idéia genérica, sem muitos detalhes, que vai sendo sucessivamente "quebrada" em blocos menores com graus de crescentes de detalhamento, até chegar-se ao grau de refinamento desejado que possibilite o desenvolvimento dos programas.

Desta forma, com a linguagem SFC podemos ter um programa principal composto de poucos estados mas onde as ações de cada estado, por sua vez, são expressas na forma de outro diagrama SFC, e assim sucessivamente até alcançar-se o grau de refinamento desejado.

Criação de novas tarefas no RSLogix 5000

A atividade prática a ser executada nesta aula requer a criação de novas tarefas (*tasks*) a serem executadas pelo CLP *CompactLogix*. Como já visto anteriormente, tarefas correspondem a unidades de execução que encapsulam programas, e que são configuradas para que seus respectivos programas executem de forma contínua, periódica ou associados a um evento de disparo. No modelo 1769-L32E do *CompactLogix* podem ser configuradas até 6 tarefas, sendo apenas uma contínua e as demais periódicas ou por eventos. A tarefa contínua (se existente) é sempre executada em *background* (ou seja, com menor prioridade que as demais). Tarefas periódicas executam em intervalos regulares de tempo definidos pelo programador, e podem ter diferentes níveis de prioridade entre si.

Como já mencionado no guia de aula anterior que introduz a programação na norma IEC 61131 (aula 9), ao criar-se um projeto com o *RSLogix 5000* são automaticamente definidos uma tarefa (*MainTask*), um programa (*MainProgram*) e uma rotina (*MainRoutine*). A tarefa *MainTask* é definida inicialmente como uma tarefa de execução contínua, podendo contudo ser alterada para uma tarefa periódica ou por eventos se assim o desejarmos. Para adicionar novas tarefas, execute os seguintes passos:

1. Clique com o botão direito do mouse no item *Tasks*, no organizador de programas, e selecione *New Task* (Fig. 1).
2. Na nova janela que será aberta, defina o nome da nova tarefa, uma descrição para a mesma, o seu tipo (caso já exista uma tarefa continua, os tipos possíveis serão apenas *Periodic* ou *Event*) e, no caso

de tarefas periódicas, o intervalo de execução, como exemplificado na Fig. 2.

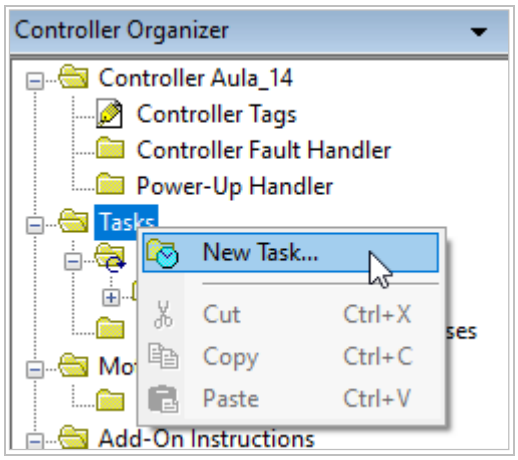

Figura 1: Criação de uma nova tarefa.

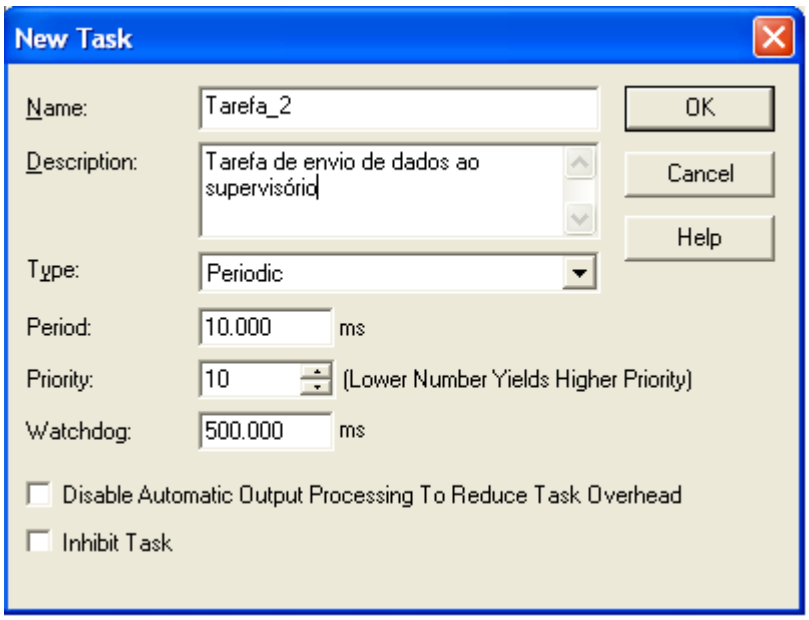

Figura 2: Definições sobre uma nova tarefa criada.

3. Criada uma nova tarefa, clique com o botão direito do mouse sobre a mesma, no organizador de programas, e defina um novo programa para a mesma como exemplificado nas Figs. 3 e 4 a seguir:

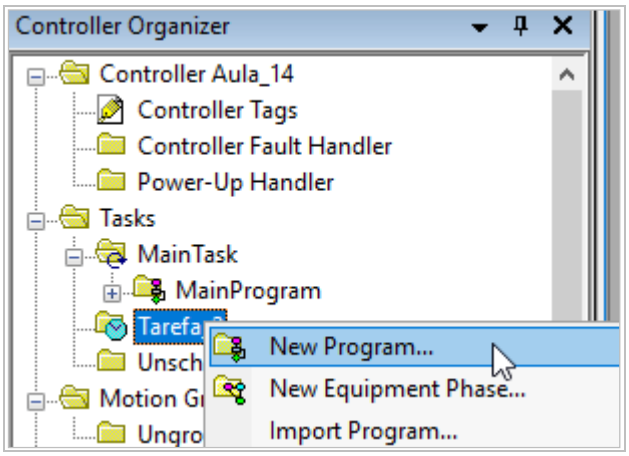

Figura 3: Criação de um programa para uma determinada tarefa.

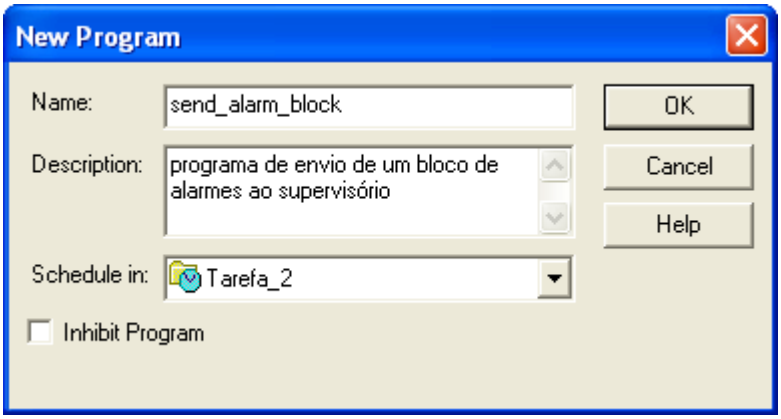

Figura 4: Definições sobre um novo programa criado.

4. Finalmente, clique com o botão direito do mouse sobre o novo programa criado e selecione *New Routine* para criar uma nova rotina para o mesmo, como exemplificado na Fig. 5. Se forem criadas duas ou mais rotinas, uma delas deve necessariamente ser definida como a principal. Este passo já foi coberto nas aulas anteriores e não será detalhado aqui.

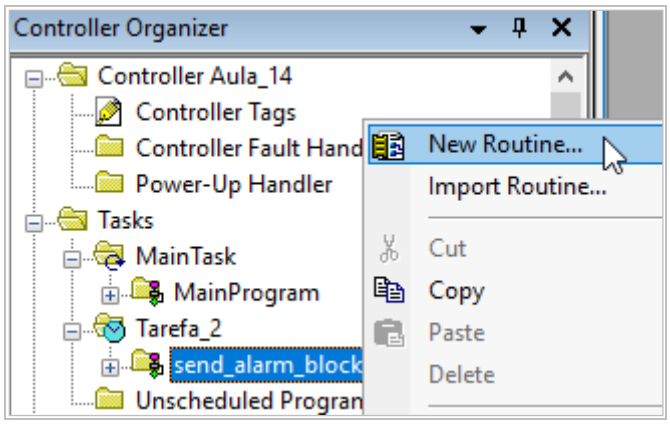

Figura 5: Criação de uma rotina para um programa de uma dada tarefa.

Atividades de Projeto e Programação

No processo industrial de fabricação de cervejas, uma etapa importante é a de pasteurização, na qual a as garrafas de cerveja já produzidas são submetidas a um choque térmico durante algum tempo, com o propósito de eliminar ou reduzir ao máximo os micro-organismos que poderiam causar sua deterioração. Este choque térmico – conhecido como pasteurização – significa aquecer a cerveja a certa temperatura por alguns segundos, e então resfria-la subitamente até a temperatura ambiente. Há diversos tipos de equipamentos para pasteurização de cervejas, desde tanques de pequena capacidade (típicos de pequenas cervejarias artesanais) até os túneis de pasteurização (empregados por grandes produtores).

Túnel de pasteurização de cervejas. Fonte: *www.smrtmachine.com*

Considere uma indústria de cervejas de médio porte em que cada lote de garrafas de cervejas é depositado em uma esteira transportadora que atravessa um túnel de pasteurização de zona única. A esteira é acionada até que o lote penetre completamente no túnel, e então desligada. Uma vez neste túnel, as garrafas são aspergidas por água quente à temperatura de 75 C durante 10 segundos, e em seguida resfriadas à temperatura de 20 C. Finalmente a esteira é novamente acionada, retirando este lote do túnel e introduzindo um próximo lote. Você deverá desenvolver uma aplicação que execute o controle automático deste processo, composta de duas tarefas:

- *MainTask* – Tarefa contínua que executará o controle automático da pasteurização. Esta tarefa contém um programa composto de duas rotinas em SFC: a principal, de nome *cervejaria*, que por sua vez chama a rotina secundária de nome *pasteurizacao*, que controla o processo de pasteurização.
- *simula_controle_temperatura* – Tarefa periódica responsável pela simulação do controle de temperatura do processo de pasteurização. Deve conter o programa de nome *controle_temperatura* que, por sua vez, contém uma rotina de mesmo nome a ser desenvolvida na linguagem ST.

Rotina *cervejaria* (em SFC)

Esta rotina deverá realizar a seguinte sequência:

1. Inicialmente o sistema encontra-se no estado desligado quando, então, aguarda o comando de partida no sistema.
2. Executa continuamente a função *pasteurizacao* que irá controlar o processo de pasteurização.
3. Aguarda o comando de desligamento do sistema.
4. Aguarda a sinalização de término do ciclo corrente de pasteurização de cervejas.
5. Retorna ao passo 1.

Rotina *pasteurizacao* (em SFC)

Esta rotina deverá realizar as seguintes ações:

1. Cancelar a sinalização de término do ciclo corrente de pasteurização.
2. Acionar a esteira por 5 segundos, de forma a introduzir o novo lote de garrafas no túnel de pasteurização.
3. Ligar o sistema de aquecimento e aguardar a temperatura alcançar 75 C.
4. Quando a temperatura alcançar 75 C, ligar a buzina e aguardar 5 segundos.
5. Ao término destes 5 segundos, desligar o sistema de aquecimento, desligar a buzina e ligar o sistema de resfriamento.
6. Aguardar a temperatura cair até 20 C, desligar o sistema de resfriamento e sinalizar o término do ciclo corrente de pasteurização.
7. Voltar ao passo 1.

Rotina *controle_temperatura* (em ST)

Esta rotina simula a lógica de controle de aquecimento e resfriamento do túnel de pasteurização. Por simplicidade, serão ignoradas as variáveis de controle e considerado apenas o sinal analógico proveniente do sensor de temperatura, a ser simulado pela variável *temperatura*. Enquanto o sistema de aquecimento estiver ligado, o programa deve incrementar o valor da variável *temperatura* a cada execução até esta alcançar 75 C, quando então permanece estável. Da mesma forma, enquanto o sistema de resfriamento estiver ligado, o programa decrementará o valor de *temperatura* a cada execução até esta alcançar 20 C. Para que a simulação destes ciclos de aquecimento e resfriamento ocorra em um tempo adequado para o teste da aplicação, defina o período de execução da tarefa *simula_controle_temperatura* como 100 ms.

Ao final do processo de criação destas tarefas e respectivos programas e rotinas, seu organizador de programas deve apresentar a seguinte aparência (Fig. 6):

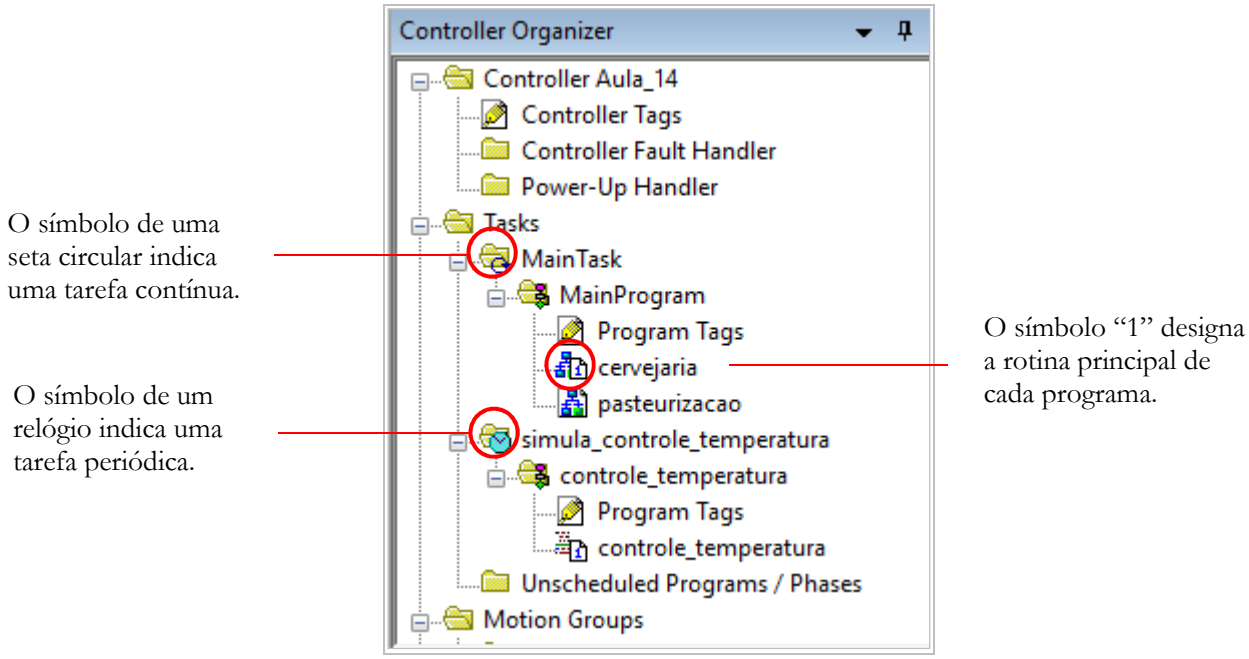

Figura 6: Tarefas, programas e rotinas definidas para esta prática.

A tabela a seguir apresenta as variáveis de entrada e saída a serem utilizadas.

Nome	Descrição	Tipo	Modo	Endereço	Módulo MICA
liga	Liga/desliga sistema	BOOL	Entrada	Local:1:I.Data.0	Liga/Desliga do Contator C1
liga_esteira	Liga/desliga esteira	BOOL	Saída	Local:3:0:Data.0	Lâmpada verde
liga_aquec	Liga/desliga sistema de aquecimento	BOOL	Saída	Local:3:0:Data.2	Lâmpada vermelha
liga_resfr	Liga/desliga sistema de resfriamento	BOOL	Saída	Local:3:0:Data.3	Lâmpada amarela
buzina	Buzina indicadora da pasteurização	BOOL	Saída	Local:3:0:Data.4	Buzina sonora
temperatura	Temperatura do túnel de pasteurização	INT	Base	----------	----------
fim_ciclo	Indica término de cada pasteurização	BOOL	Base	----------	----------

Observações Importantes:

1. Defina todos os *tags* referentes às variáveis internas e de E/S como *tags* globais, de forma que sejam visíveis a todas as rotinas do programa. Para tal, declare-os no escopo do controlador (ou seja, no escopo do projeto associado ao controlador) e não no escopo de cada programa ou rotina (figura 7). Se você errar neste passo, será obrigado a cancelar os *tags* e declará-los novamente!

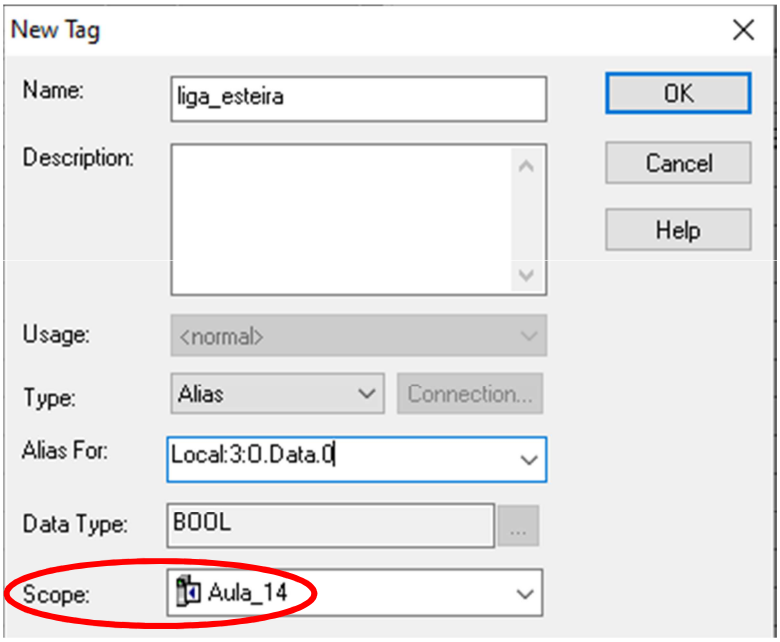

Figura 7: Escopo de declaração de variáveis (*tags*)

2. Para que a rotina principal *cervejaria* chame, no passo adequado, a rotina secundária *pasteurizacao*, use o comando *Jump to Subroutine (JSR):*

   JSR(nome_da_rotina);

   Este comando irá provocar o desvio da execução da rotina principal para a rotina referenciada. Observe, contudo, que quando o passo que chama a rotina secundária é desativado, na próxima ativação deste passo a rotina chamada reiniciará do ponto onde teve sua execução interrompida. Se desejarmos que a rotina chamada reinicie a partir de seu passo inicial, devemos adicionar a instrução *SFC Reset (SFR),* especificando a rotina a ser reinicializada e o passo inicial:

   SFR(nome_da_rotina, passo_inicial);

   A instrução SFR pode ser executada antes da instrução JSR, com o qualificador P1, ou depois da instrução JSR, com o qualificador P0. A figura abaixo exemplifica o emprego destas instruções:

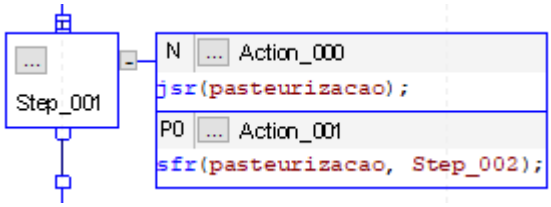

Figura 8: Passo de um diagrama SFC com as instruções *JSR* e *SFR*.

3. Observe que as diferentes tarefas colaboram entre si através das variáveis globais. O resultado coletivo das tarefas executadas é que resulta na ação de controle desejada.
4. Teste seu programa minuciosamente.

Instruções de montagem no MICA

Nesta prática, caberá a você identificar as interligações necessárias entre os sensores e atuadores do MICA e os cartões de E/S do CLP, bem como realizar as respectivas conexões necessárias. Lembre-se porém dos seguintes cuidados:

1. Com o MICA energizado, certifique-se que o botão de emergência esteja pressionado durante qualquer operação de montagem;
2. Verifique atentamente se todos os elementos de montagem estão devidamente aterrados;
3. Confira com atenção a sua montagem antes de rearmar o botão de emergência.

Referências

Rockwell Automation, *SFC programming*, 1756-pm006
Rockwell Automation, *SFC and ST programming*, 1756-pm003