

PGENE503 – Sistemas de Tempo Real

Quarta Lista de Exercícios

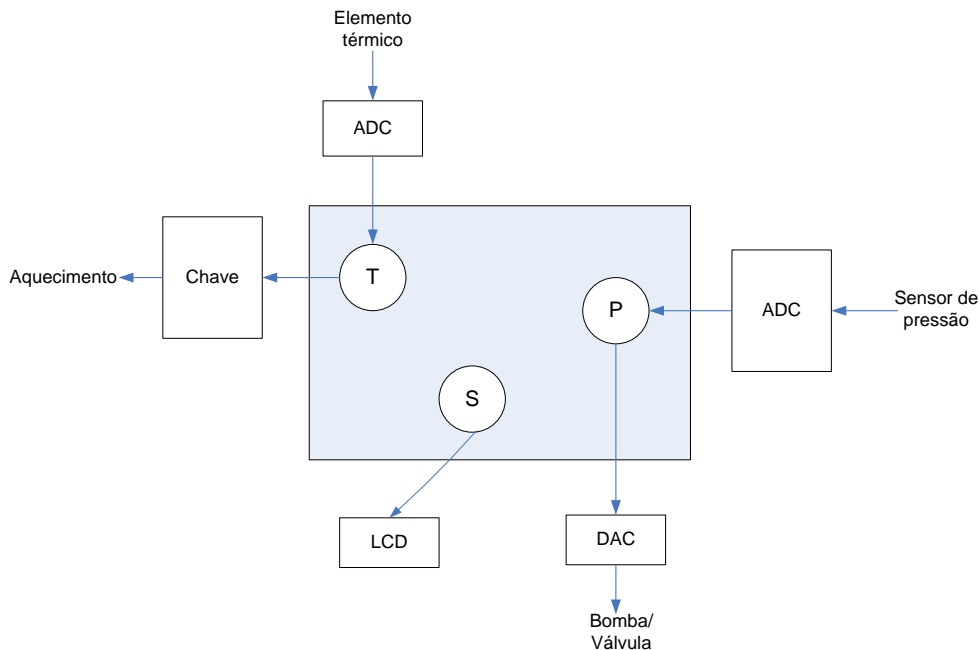
1) (**Warming-up**) Crie um procedimento chamado *One_Task*. Declare um tipo de tarefa chamado *Hello_Task* com um parâmetro de entrada chamado “mensagem” que é do tipo *Character*. O objetivo da tarefa é imprimir na tela o texto “*Hello, I am Task*” mais o caractere que é passado através do parâmetro de entrada (use a função *Put* do pacote *Ada.Text_IO*). Declare uma variável *Task_A* do tipo *Hello_Task* com a mensagem “A”.

- a) Agora estenda o programa com um parâmetro adicional chamado *Amount* que declara com que frequência o texto deveria ser mostrado na tela (estenda o corpo do tipo da tarefa apropriadamente). Declare duas variáveis *Task_A* e *Task_B* com as mensagens “A” e “B” e o número de repetições 5 e 7, respectivamente.
- b) Insira uma instrução de retardo dentro do corpo do tipo da tarefa no programa do item (a) tal que cause um curto tempo de espera na tarefa.
- c) Estenda o tipo da tarefa do programa do item (b) com um ponto de sincronização chamado de *start* o qual permite que tarefas sejam ativadas explicitamente neste ponto (por exemplo, forçando uma certa sequência).

2) (**Produtor em Ada**) Uma tarefa chamada *Producer* imprimirá caracteres na tela. Para este propósito, a tarefa receberá ordens que causam imprimir um certo caractere, um certo número de vezes com um certo tempo de retardo entre as repetições. O número de repetições é um inteiro natural com o valor *default* de 10. O retardo ocorrerá depois que cada caractere for impresso na tela. O valor *default* para este parâmetro é zero. Note que deve ser possível na sua implementação terminar a tarefa através de alguma interrupção externa.

- a) Para resolver este problema declare um tipo de tarefa chamado *Producer* com dois pontos de chamada *Order* e *Control*. O ponto de chamada *Control* deve ter um parâmetro de entrada do tipo *Boolean*. Se o valor *False* é passado então a tarefa pode receber uma nova ordem; caso contrário, se o valor *True* é passado então a tarefa terminará.
- b) Crie o corpo da tarefa *Producer*.
- c) Escreva um procedimento chamado *Process_Frame* que cria duas tarefas A e B do tipo *Producer* e então executa as seguintes chamadas:
 - 1) Tarefa A imprimirá o caractere “A” 10 vezes sem retardo.
 - 2) Tarefa B imprimirá o caractere “B” 10 vezes com um retardo de 1.0 segundo.
 - 3) Tarefa A imprimirá o caractere “A” 100 vezes sem retardo.
 - 4) Tarefa B terminará.
 - 5) Tarefa A terminará.

3) (**Programação concorrente em Ada**) Um simples sistema embarcado é mostrado na figura abaixo:



Um processo **T** lê os valores medidos a partir de um sensor de temperatura (através de um conversor analógico-digital, ADC) e liga/desliga o sistema de aquecimento (através de uma chave que é controlada por uma saída digital). O processo **P** possui uma tarefa similar, isto é, ele regula a pressão com um sensor de pressão e uma bomba ou válvula (através de um conversor digital analógico, DAC). Ambos os processos **T** e **P** devem transferir dados para um processo **S**, o qual mostra através de um display de cristal líquido (*LCD - Liquid crystal display*) os valores atuais medidos.

A tarefa deste sistema embarcado é manter a temperatura e pressão de um processo químico dentro dos limites definidos. Implemente o software para o sistema em Ada em dois diferentes estilos:

1. Use um simples programa seqüencial que não considera a natureza concorrente dos processos **T**, **P** e **S**. Um sistema operacional de tempo real não é necessário aqui.
2. Use o suporte de programação paralela em Ada e mapeie a estrutura lógica dos três processos **T**, **P** e **S** para as tarefas do Ada.

Para este propósito, utilize os pacotes descritos abaixo:

```
package DataTypes is
-- necessary type definitions
type TemperatureValue is new Integer range 10..500;
type PressureValue is new Integer range 0..750;
type HeatingPosition is (On, Off);
type ValvePosition is new Integer range 0..9;
```

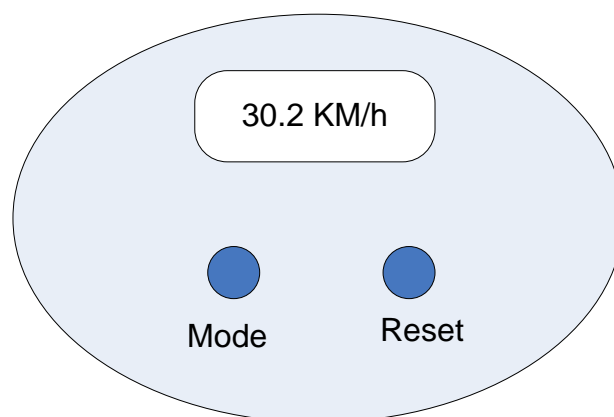
```
end DataTypes;
```

```
with DataTypes; use DataTypes;
package IO is
-- Procedures for the data communication with the environment
procedure Read_Temp(TV : out TemperatureValue); -- from ADC
procedure Read_Pressure(PV : out PressureValue); -- from ADC
procedure Set_Heating(HP : HeatingPosition); -- to switch
procedure Set_Valve(VP : ValvePosition); -- to DAC
procedure Show_Temp(TV : TemperatureValue); -- to display
procedure Show_Pressure(PV : PressureValue); -- to display
end IO;
```

```
with DataTypes; use DataTypes;
package ControlAlgorithms is
-- Procedures for generating a setpoint value based on the
measured value
procedure Calculate_Heating_Setpoint(TV : TemperatureValue;
HP : out HeatingPosition);
procedure Calculate_Valve_Setpoint(PV : PressureValue;
VP : out ValvePosition);
end ControlAlgorithms;
```

Declare as vantagens de desvantagens das duas abordagens que você implementou.

4) **(Programação concorrente em C/POSIX)** Neste exercício você desenvolverá em C/POSIX um simples computador embutido em uma bicicleta usando a plataforma LPC3250.



O sistema embarcado medirá e mostrará no *display* a velocidade atual e a quilometragem da viagem. Além disso, o sistema mostrará a quilometragem total e o tempo total da viagem. Estas funções serão realizadas por um software embarcado na plataforma LPC3250 com as seguintes especificações:

- Uma interrupção chamada *WHEEL_INT* será gerada por um sensor para cada rotação da roda frontal da bicicleta.
- O computador tem basicamente dois botões chamados *Mode* e *Reset*, os quais geram as interrupções *MODE_INT* e *RESET_INT*, respectivamente, quando eles são pressionados.
- Depois de inserir as baterias, o computador estará no estado chamado “Viagem” (o qual mostra a quilometragem atual da viagem). Se o botão *Mode* for pressionado uma ou mais vezes então o estado é sucessivamente alterado para o seguinte: “Velocidade” (velocidade atual), “Total” (quilometragem total), “Tempo” (tempo da viagem) e de volta para o estado “Viagem”.
- No estado “Viagem” a quilometragem da viagem atual é mostrada a cada 200 ms com uma tolerância de 100m. O botão *Reset* ajusta o contador de volta para zero.
- No estado “Velocidade” a velocidade atual é mostrada a cada 100ms.
- No estado “Total” a quilometragem total de todas as viagens é mostrada a cada 500ms com uma tolerância de um quilômetro. A quilometragem total não pode ser ajustada para zero (exceto no caso de remoção das baterias).
- No estado “Time” a duração atual da viagem é mostrada a cada segundo. O botão *Reset* ajusta o tempo da viagem de volta para zero.