ProModel®

Guia de Referência

MATERIAL EXTRAÍDO DO ‘REFERENCE GUIDE’ DO PROMODEL PARA CONSULTA RÁPIDA DOS USUÁRIOS DO SOFTWARE

São Paulo/SP

Telefax: (011) 5561.5353

Guia de Referência Rápida do ProModel for Windows

Editado e impresso pela BELGE – Engenharia & Sistemas

Rua Bernardino de Campos, 318

04620-001 – São Paulo – SP – Brasil

Pabx: (011) 5561.5353

e-mail : suporte@belge.com.br

http://www.belge.com.br/servicos_suporte.html

A elaboração deste 'Guia de Referência Rápida' é uma iniciativa da BELGE e visa auxiliar os usuários do software em nosso país, sejam eles empresas que utilizam o produto para aprimorar seus procedimentos e sua tomada de decisão, ou estudantes e professores que utilizam o software em suas atividades acadêmicas. Com este trabalho temos certeza de que este produto tão poderoso, amigável, flexível e mundialmente consagrado, será mais acessível aos profissionais das indústrias de manufatura, processos e serviços em nosso país.

ATENÇÃO:

Este guia de referência é apenas um extrato do 'Reference Guide' da versão 7.0 do software ProModel e não tem o objetivo de substituí-lo. Campos lógicos onde são válidos, exemplos de utilização, algumas exceções ou casos especiais não estão inclusos neste guia. Para informações mais detalhadas favor consultar os manuais originais que acompanham o software ProModel.

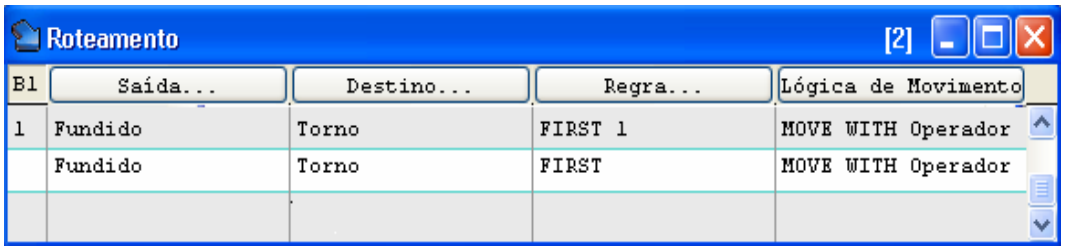
ÍNDICE

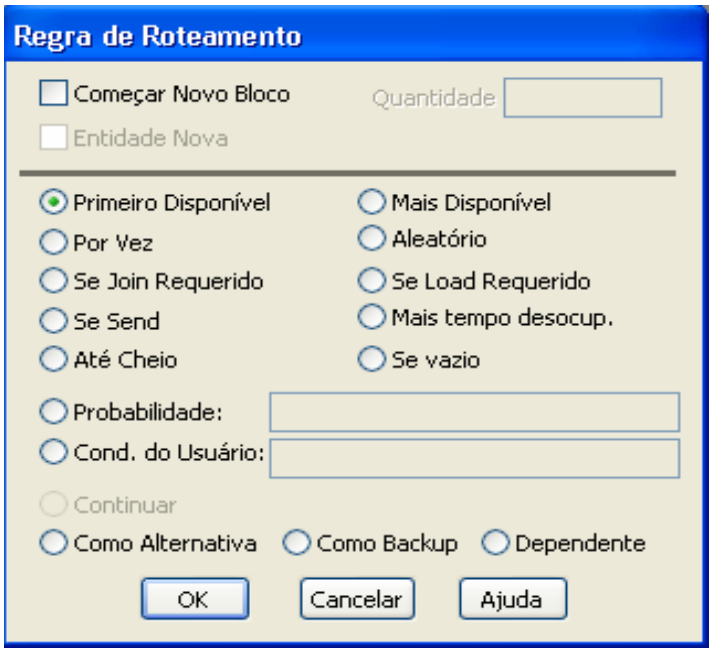
- REGRAS DE ROTEAMENTO

- COMANDOS E FUNÇÕES

REGRAS DE ROTEAMENTO
(Routing Rules)

As regras de roteamento fazem parte da construção dos processos nos modelos elaborados com o software. Para acessá-las, entre no módulo de processos (Construir/Processo). As regras de roteamento determinam o próximo local para a entidade em processamento. Você deve especificar no campo "Regra" da janela "Roteamento" (janela da direita do módulo de processos) digitando ou selecionando as regras da caixa de diálogo de regras de roteamento (Roteamento Regra) mostrada abaixo, clicando no botão Regra.





ALTERNATE

Sintaxe

ALT

Descrição

Seleciona um local como um destino alternativo se a capacidade disponível e a condição para uma rota anterior não for satisfeita. Se o local alternativo não estiver disponível, a entidade espera até que este local esteja disponível ou até que a condição da regra de roteamento anterior seja satisfeita.

Rotas alternativas são comuns a todas as rotas primárias, exceto às regras Probability (por probabilidade) e User Condition (com condição definida pelo usuário), e você pode listá-las depois do último roteamento primário mas sempre antes de qualquer regra do tipo BACKUP.

Por exemplo, uma máquina de alta velocidade pode ser preferida, mas alternativamente, uma máquina mais lenta pode ser usada quando a máquina mais rápida não estiver disponível.

BACKUP

Sintaxe

BACKUP

Descrição

Os locais especificados por esta regra são selecionados apenas se todos os outros destinos (primários ou alternativos) listados nos roteamentos anteriores de um mesmo bloco não estiverem disponíveis devido a qualquer parada por downtime, incluindo turnos. Um dos usos da regra BACKUP é no caso de uma máquina que não está funcionando, e as entidades precisam de outro destino para continuar.

CONTINUE

Sintaxe

CONT1

Descrição

Deixa uma entidade no local atual para processamento adicional. O ProModel procura na lista de processos adiante e depois desde o início da lista até que um processo seja encontrado para a entidade que sai do local atual. Se o nome da entidade não mudar, o processo original se repetirá continuamente a não ser que uma condição faça com que o bloco com a regra CONTINUE seja pulado.

A regra CONTINUE também permite mudanças dinâmicas de prioridade. Isto significa que a entidade que entra em um local com uma prioridade baixa pode ser dinamicamente promovida a uma prioridade mais alta sem sair deste local. A lógica de movimentação do bloco com a regra CONTINUE não será executada.

Não são permitidos valores de tempo ou movimentações na coluna de 'move logic' do bloco de roteamento que contenha a regra CONTINUE. Esta regra é uma maneira de rotear uma entidade de um local de capacidade única para o mesmo local sem que se crie um beco sem saída. A regra CONTINUE pode simular também um fornecimento ilimitado de matéria prima em um local.

As estatísticas para um local usando a regra CONTINUE são coletadas como uma entrada para a entidade.

DEPENDENT

Sintaxe

DEP

Descrição

Seleciona um local se, e somente se, você seleciona o roteamento que o precede imediatamente. Um roteamento ALTERNATE colocado depois de um roteamento dependente é uma alternativa ao último roteamento principal que precede a condição de dependência, e não o roteamento dependente em si.

Um roteamento dependente pode ser usado quando um processo resulta em dois tipos diferentes de entidades que devem ir para locais diferentes. Por exemplo, um roteamento dependente poderia simular uma peça com duas partes na qual apenas uma deve ser pintada. O roteamento para o processo de preparação deve rotear uma parte para a área de pintura e a outra para a área de espera. Mas a parte que deve ir para a área de espera só poderá ser roteada se a área de pintura tiver capacidade para receber a parte que irá ser pintada. Assim a parte roteada para a área de espera só vai para lá quando a outra parte for para a área de pintura.

EMPTY

Sintaxe

EMPTY {<*expressão*>}

Descrição

Seleciona um local apenas se este estiver completamente vazio. Esta regra de roteamento é similar a regra UNTIL FULL exceto pelo fato de o local precisar estar completamente vazio antes de ser inicialmente selecionado. Assim que um local que estiver vazio é selecionado, este continuará a ser selecionado até o momento em que estiver cheio. Se nenhum local estiver vazio a entidade de saída irá esperar até que algum dos locais fique vazio.

A condição EMPTY é usada para situações onde dois ou mais locais de múltipla capacidade estão sendo alimentados de uma mesma fonte e o modelador deseja que cada local seja completamente enchido e completamente esvaziado de um modo alternado.

FIRST AVAILABLE

Sintaxe

FIRST {<*expressão*>}

Descrição

Seleciona o primeiro local disponível entre um ou mais locais listados no bloco de roteamento. Especificar múltiplos roteamentos FIRST AVAILABLE em um bloco de roteamento terá o mesmo efeito de especificar um roteamento FIRST AVAILABLE seguido de um ou mais roteamentos ALTERNATE.

A regra FIRST AVAILABLE é a regra padrão. Neste caso, o roteamento FIRST AVAILABLE pode ser interpretado como um roteamento "primário".

JOIN

Sintaxe

JOIN {<*expressão*>}

Descrição

Seleciona um local assim que uma solicitação de união (JOIN) é feita naquele local através do comando JOIN. Como a entidade que vai ser unida não utiliza a capacidade do local, não há necessidade de checar a disponibilidade de capacidade no local de destino. Várias solicitações de união são atendidas de acordo com a ordem de espera, ou seja, a solicitação que estiver esperando a mais tempo será a primeira a ser atendida e assim por diante. As entidades roteadas com o roteamento JOIN não são enviadas para aquele local até que seja encontrado um comando JOIN no local de destino.

LOAD

Sintaxe

LOAD {<*expressão*>}

Descrição

Seleciona um local assim que uma solicitação de carregamento (LOAD) é feita naquele local através do comando LOAD. Como a entidade que vai ser unida não utiliza a capacidade do local, não há necessidade de checar a disponibilidade de capacidade no local de destino. Várias solicitações de carregamento são atendidas de acordo com a ordem de espera, ou seja, a solicitação que estiver esperando a mais tempo será a primeira a ser atendida e assim por diante.

LONGEST UNOCCUPIED

Sintaxe

LU {<*expressão*>}

Descrição

Seleciona um dos locais listados em um bloco de roteamentos baseado em que local ficou com menor ocupação. Se vários locais de múltipla capacidade possuírem uma ou mais entidades, o local com maior capacidade disponível será selecionado. Se nenhum local tiver capacidade disponível, o primeiro que ficar disponível será selecionado. Esta regra é útil em situações nas quais efeitos residuais precisam ser minimizados antes de uma utilização posterior em um local como por exemplo em um forno que deve esfriar até a temperatura ambiente, ou na diminuição do vapor em suspensão num ambiente de pintura.

MOST AVAILABLE

Sintaxe

MOST {<*expressão*>}

Descrição

Seleciona um dos locais listados em um bloco de roteamento, baseado no local que tiver maior capacidade disponível. Se não houver capacidade disponível em nenhum dos locais listados, o ProModel seleciona o primeiro que ficar disponível. Use a regra de roteamento MOST para equalizar filas na frente dos trabalhadores. Esta regra é útil em casos onde você precisa balancear os níveis de estoque entre várias filas.

PROBABILITY

Sintaxe

<probabilidade> {<*expressão*>}

Descrição

Seleciona aleatoriamente um local listado no bloco de roteamento baseado em uma probabilidade. Você deve usar vários roteamentos por probabilidade juntos e a soma das probabilidades deve ser igual a um, o equivale a 100%. A entidade irá esperar até que o local escolhido tenha capacidade disponível. Diferentemente de outros roteamentos primários, você pode especificar um roteamento ALTERNATE e BACKUP depois de cada roteamento PROBABILITY em um mesmo bloco de roteamento. Se o local selecionado tiver um roteamento ALTERNATE e não tiver capacidade disponível, a entidade irá esperar até que o local tenha capacidade disponível.

RANDOM

Sintaxe

RANDOM {<*expressão*>}

Descrição

Seleciona aleatoriamente um de vários locais disponíveis listados no bloco de roteamento, de forma que o ProModel pode vir a escolher qualquer local que tenha capacidade disponível. Se nenhum dos locais tiver capacidade disponível, o ProModel irá selecionar o primeiro local que ficar disponível.

SEND

Sintaxe

SEND {<*expressão*>}

Descrição

Faz com que a entidade permaneça no mesmo local até que um dos locais listados execute o comando SEND. Assim que o comando SEND é executado, o local deve ter capacidade disponível antes que a entidade seja roteada. Você pode definir vários locais alternativos com a regra SEND em um bloco.

TURN

Sintaxe

TURN {<*expressão*>}

Descrição

Seleciona rotativamente os locais listados no bloco de roteamentos que estiverem disponíveis. Se nenhum dos locais listados estiver disponível, o ProModel selecionará o primeiro que ficar disponível. Você pode listar um local mais de uma vez em um bloco de roteamento se você quiser usá-lo mais vezes que os outros.

UNTIL FULL

Sintaxe

FULL {<*expressão*>}

Descrição

Esta regra continua direcionando todas as entidades para o primeiro local especificado até que sua capacidade seja preenchida e depois passa para o segundo local especificado até que este não tenha mais capacidade e assim por diante. Se todos os locais estiverem cheios, o ProModel selecionará o primeiro que ficar disponível.

USER CONDITION

Sintaxe

<condição> <*expressão*>

Descrição

Seleciona um dos vários locais listados no bloco de roteamento baseado em uma condição definida pelo usuário. O ProModel geralmente usa vários roteamentos USER CONDITION em um mesmo bloco de roteamento. No mínimo uma das condições definidas pelo usuário deve ser verdadeira, caso contrário uma mensagem de erro será mostrada e a simulação será interrompida. Para que a entidade seja roteada para um local, este local deve ter capacidade disponível. Diferentemente de outras regras primárias, você pode especificar roteamentos ALTERNATE ou BACKUP depois de cada roteamento USER CONDITION em um mesmo bloco de roteamento.

COMANDOS E FUNÇÕES

ACCUM

Sintaxe

ACCUM <*expressão*>
ACCUM 10
ACCUM Var1

Descrição

Acumula, sem consolidar, a quantidade de entidades especificada em um determinado local. O comando ACCUM funciona como um portão que evita que as entidades sejam processadas até que um certo número delas chegue. Assim que o número de entidades for acumulado, elas passarão pelo portão e serão processadas individualmente, independente das outras.

O comando ACCUM pode ser usado para modelar situações onde várias entidades devem ser acumuladas antes de serem processadas. Por exemplo, quando uma empilhadeira coloca pallets em um caminhão, será mais eficiente acumular vários pallets antes de solicitar a empilhadeira.

ACTIVATE

Sintaxe

ACTIVATE <*subrotina*>({<*parâmetro1*>,<*parâmetro2*>...})

ACTIVATE Sub1()

Descrição

Executa uma subrotina independente. Chama a subrotina sem esperar que a subrotina chamada acabe de ser executada. Assim subrotinas independentes podem ser executadas em paralelo com a lógica que as chamou. Subrotinas independentes são subrotinas que não dependem de nenhum

local ou entidade e são executadas de acordo com o que acontece dentro da lógica que as chamou.

Use ACTIVATE para processar lógicas que tenham os comandos WAIT ou WAIT UNTIL quando você não quiser usar uma entidade para processar estes comandos. Por exemplo, um ACTIVATE na lógica de inicialização pode chamar uma subrotina que ajusta a frequência de chegada de um tipo de entidade dependendo da hora do dia.

As subrotinas independentes chamadas pelo comando ACTIVATE não podem ser funções de sistemas específicas a entidades ou locais. Se a subrotina retorna um valor, o valor é ignorado. Subrotinas externas não podem ser chamadas com o comando ACTIVATE, entretanto podem ser chamadas de dentro de uma subrotina ativada.

ANIMATE

Sintaxe

ANIMATE <*expressão*>

ANIMATE 70

ANIMATE Var1

Descrição

Ajusta a velocidade de animação da simulação. Quanto maior for o valor, mais rápida será a animação. O comando ANIMATE é utilizado principalmente para aumentar ou diminuir a velocidade de uma simulação assim que uma condição é satisfeita. Outro uso comum é para ajustar a velocidade de animação para o valor 100 (cem) na Lógica de Inicialização para avançar a simulação rapidamente para algum ponto no tempo.

ARRAYDIMS()

Sintaxe

ARRAYDIMS (<nome da matriz>)

Descrição

Retorna o número de dimensões de uma matriz.. Pode ser utilizado em qualquer lógica do Promodel.

ARRAYDIMSIZE()

Sintaxe

ARRAYDIMSIZE (<nome da matriz> , <número da dimensão>)

Descrição

Retorna o tamanho de uma determinada dimensão em uma matriz. Basta fornecer através da sintaxe acima o nome da matriz e a dimensão que se deseja saber o tamanho. Esta função pode ser utilizada em qualquer parte da lógica do Promodel.

ASSIGNMENT STATEMENT

Sintaxe

<variável, matriz ou atributo> = <expressão numérica>
Var1 = 300

Attr2 = Clock(hr) - Attr3

Descrição

Atribui o valor de uma expressão numérica a uma determinada variável, matriz ou atributo. Pode ser utilizado em qualquer parte da lógica do Promodel.

BEGIN

Sintaxe

BEGIN ou {

```
WHILE FREECAP(Loc1) > 5 DO
    BEGIN
        INC Var2,5
        WAIT 5 sec
    END
```

```
IF Var1 > 5 THEN
        {
        INC Var2,5
        WAIT 5 sec
        }
```

Descrição

Define um bloco de comandos com um correspondente comando END. BEGIN e END são quase sempre usados em conjunto com um outro comando de controle como IF-THEN e DO WHILE. Para todo comando BEGIN deve haver o comando END correspondente.

BREAK

Sintaxe

BREAK

Descrição

Sai do loop WHILE...DO, DO...WHILE, ou DO...UNTIL que está sendo executado. O próximo comando a ser executado será o que está imediatamente após ao comando END associado ao loop que está sendo executado. Se o comando BREAK for colocado fora de qualquer loop, o ProModel sairá de toda a lógica.

BREAKBLK

Sintaxe

BREAKBLK

Descrição

Sai do bloco de comando que está sendo executado. O próximo comando a ser executado será o que está imediatamente após o comando END do bloco que está sendo executado. Se um BREAKBLK é colocado fora de qualquer bloco de comando, o ProModel sairá de toda a lógica.

CAP()

Sintaxe

CAP(<*local*>)

GROUP CAP(Loc1)

Descrição

Retorna a capacidade total de um local. CAP() pode ser usado para preparar um lote de entidades para preencher um local.

CALDAY()

Sintaxe

CALDAY()

Descrição

A função CALDAY() corresponde ao dia da semana da data no calendário, seja no período de aquecimento ou a partir da data de início da simulação, no menu "Simulação > Opções". Uma vez que o CALDAY() reinicia com o advento de uma nova semana, todo dia de semana retornará o mesmo valor (isto é, quarta-feira irá sempre retornar o valor 3).

Nota

O CALDAY() funciona somente quando você seleciona a data no calendário na caixa de diálogo do menu "Simulação > Opções".

CALDOM()

Sintaxe

CALDOM()

Descrição

Esta função retorna com o dia do calendário, seja no período de aquecimento ou a partir da data de início da simulação, no menu "Simulação > Opções".. Os valores retornados por esta função serão inteiros e dentro do intervalo de 1 a 31. Pode ser utilizada em qualquer parte da lógica do Promodel, no entanto, só funcionará se o período de simulação for selecionado para rodar como calendário.

CALHOUR()

Sintaxe

CALHOUR()

Descrição

A função CALHOUR() corresponde a hora da data no calendário seja no período de aquecimento ou a partir da data de início da simulação, no menu "Simulação > Opções". Uma vez que esta função está diretamente ligada ao relógio de 24-horas mostrado na tela durante a simulação, CALHOUR() nunca vai retornar um valor maior que 23.

Nota

O CALHOUR() trabalha somente quando você seleciona uma data no calendário na caixa de diálogo sob o menu "Simulação > Opções".

CALMIN()

Sintaxe

CALMIN()

Descrição

A função CALMIN() corresponde ao minuto da data no calendário seja no período de aquecimento ou a partir da data de início da simulação, no menu "Simulação > Opções". Como esta função está diretamente ao relógio de 24 horas mostrado na tela durante a simulação, CALMIN() nunca retornará um valor maior que 59

CALMONTH()

Sintaxe

CALMONTH()

Descrição

Esta função corresponde ao minuto de uma data específica do calendário seja no período de aquecimento ou a partir da data de início da simulação, no menu "Simulação > Opções". CALMonth() sempre retornará um valor entre 1 e 12. Pode ser utilizada em qualquer parte da lógica do Promodel, no entanto, só funcionará se o período de simulação for selecionado para rodar como calendário.

Exemplo

```
If CalMonth()=12 THEN WAIT 20
Else Wait 10
```

CALYEAR()

Sintaxe

CALYEAR()

Descrição

Esta função corresponde ao ano de uma data específica do calendário seja no período de aquecimento ou a partir da data de início da simulação, no menu "Simulação > Opções". Pode ser utilizada em qualquer parte da lógica do Promodel, no entanto, só funcionará se o período de simulação for selecionado para rodar como calendário.

CHAR()

Sintaxe

CHAR(<*expressão*>)

CHAR(10)

Descrição

Retorna o caractere ASCII correspondente ao valor do resultado da expressão. Esta função é útil para gerar caracteres ASCII que não podem ser digitados, como caracteres acentuados. Esta função é quase sempre usada em conjunto com uma expressão string e o operador de concatenação ("$").

O número correspondente ao caractere ASCII varia de computador para computador dependendo do ajuste de caracteres especificado no arquivo config.sys de seu computador. Para determinar o número correspondente ao caractere ASCII no seu computador, execute o modelo CHAR.MOD que está no diretório PROMOD4/MODELS/REFS. Veja o arquivo que este modelo gera chamado CHAR.TXT no mesmo diretório.

CLOCK

Sintaxe

CLOCK({<*unidade de tempo*>})

IF CLOCK(DAY) >= 1.5 THEN PAUSE

Attr1 = CLOCK()

Descrição

Retorna o valor do tempo de simulação na unidade especificada. As unidades de tempo devem ser consistentes ao comparar valores. Se um atributo tem seu tempo em minutos, qualquer valor de tempo comparado com este atributo deve estar em minutos.

Quando nenhuma unidade de tempo é especificada nos parênteses na função CLOCK, será utilizada a unidade de tempo padrão definida no 'General Information'.

CLOSE

Sintaxe

CLOSE <*nome do arquivo*>

CLOSE Arquivo_de_Chegada

CLOSE ALL

Descrição

Fecha um arquivo que foi previamente escrito com WRITE, WRITELINE, XWRITE, ou lido com READ. Use CLOSE quando acabar de usar um arquivo para liberar recursos do sistema. O arquivo será automaticamente reaberto para ser lido ou escrito, caso tenha sido fechado. Todos os arquivos abertos são automaticamente fechados no final da simulação. Este comando é especialmente útil quando você está usando vários arquivos externos e quer economizar recursos do sistema.

COMBINE

Sintaxe

COMBINE <*expressão*> {AS <*nome da nova entidade*>}

COMBINE Var1

COMBINE 3 AS EntQ

COMBINE Var1 AS ENT(Attr1)

Descrição

Acumula e consolida uma quantidade específica de entidades em uma entidade, opcionalmente com um nome diferente. Diferentemente do comando GROUP, entidades combinadas perdem sua identidade e seus atributos e não podem ser posteriormente desagrupadas. Use COMBINE quando várias entidades devem ser combinadas, como quando oito velas para carro devem ser combinadas em uma caixa. Note que após várias entidades terem sido combinadas em um local, nenhuma estatística de qualquer das entidades combinadas naquele local poderá ser colhida.

Quando for especificado COMBINE <*expressão*> AS <*nome da nova entidade*> na lógica de operação, deve haver um outro bloco de operação no mesmo local. Neste caso, a entidade que entra no novo bloco de operação é a nova entidade especificada no comando COMBINE.

COMMENTS

Sintaxe

Sinaliza o início de uma linha de comentário. O ProModel irá ignorar todos os caracteres do resto da linha.

// Sinaliza o início de uma linha de comentário. O ProModel irá ignorar todos os caracteres do resto da linha. Este símbolo funciona exatamente como o símbolo #.

/*...*/ Sinaliza o início de várias linhas de comentário. O ProModel irá ignorar todos os caracteres após o "/*"até achar o "*/". Use este tipo de comentário para explicações longas e para evitar que o ProModel execute longas porções de lógica durante a depuração do modelo (debugging).

Descrição

Comentários são notas para o modelador dentro de blocos de lógica. O ProModel os ignora, mas eles podem ser úteis para explicar a lógica quando mais de uma pessoa estiver usando o modelo.

CONTENTS()

Sintaxe

CONTENTS(<*local*>{,<*tipo de entidade*>})

LOAD CONTENTS(Loc1)

JOIN CONTENTS(Loc1,EntA) EntA

Descrição

Retorna o número total de entidades de um local ou o número de entidades de um certo tipo que estão naquele local. Use CONTENTS() para tomar decisões baseadas na ocupação do local. Usando CONTENTS() para controlar o número de entidades em um local exige menos passos e permite mais flexibilidade que o uso de variáveis para contar o número de entidades que entram e que saem de um local. Por exemplo, a segunda sintaxe acima, faz com um comando o que normalmente exigiria vários comandos sem a função CONTENTS().

CREATE

Sintaxe

CREATE <*expressão*> { AS <*nome da entidade*>}

{TAKE {<*expressão2*>} <*recurso*>,...}

CREATE 10 AS EntX

CREATE 2 AS EntB TAKE 1 Res1, 2 Res2

CREATE Var1 AS Ent(Var2) TAKE Var3 Res(Var4)

Descrição

Cria um número específico de entidades adicionais à entidade original e copia todos os atributos da entidade original para cada nova entidade. A primeira entidade criada pode, opcionalmente, apanhar qualquer um dos recursos que estavam sendo utilizados pela entidade que a criou. As novas entidades criadas não exigem capacidade adicional do local e são processadas imediatamente.

O comando CREATE pode simular a criação de papéis que precisam ser carregados por alguém para outro local para serem aprovados enquanto a entidade base continua processando. Antes que a entidade base possa sair do local, os papéis devem ser aprovados e roteados de volta para o local original onde se juntam com a entidade base.

DEBUG

Sintaxe

DEBUG

Descrição

Ativa o depurador do ProModel (debugger). Use DEBUG para 'andar' pela lógica, comando por comando, e examinar valores de atributos e de variáveis enquanto desenvolve o modelo. Quando o modelo está funcionando, os comandos DEBUG são geralmente removidos.

DEC

Sintaxe

DEC <*nome*>{,<*expressão*>}

DEC Var1

DEC Attr1,5

Descrição

Decrementa uma variável, elemento de matriz, ou atributo, do valor especificado pela expressão numérica. Para decrementar um variável, atributo, ou elemento de matriz quando a entidade realmente sai de um local, use DEC na lógica de movimentação.

DISPLAY

Sintaxe

DISPLAY <*expressão1*> {,<*expressão2*>}

DISPLAY “Var1 = ” $ Var1 $ “e Attr1 = ” $ Attr1

DISPLAY “A quantidade atual de entidades é:”, Var1

DISPLAY “Começando agora o centésimo processo”

DISPLAY Numero_na_Fila

Descrição

Pausa a simulação e mostra uma mensagem. A simulação irá continuar quando o usuário selecionar OK. O símbolo ($) deve ser usado para colocar um valor numérico dentro da mensagem (como do primeiro exemplo de sintaxe acima). As funções ENT(), LOC(), e RES() irão mostrar o nome da entidade, local, ou recurso.

DO...UNTIL

Sintaxe

DO <*bloco de comandos*> UNTIL <*expressão*>

DO INC Var1 UNTIL Matriz1[Var1] <> 10

```
DO
        BEGIN
                INC Var2,5
                WAIT 5 sec
        END
UNTIL FreeCap(Loc1) > 5
```

Descrição

Repete um comando ou bloco de comandos continuamente enquanto a condição se mantiver falsa. DO...UNTIL é um loop com uma condição de saída, significando que o loop será sempre executado pelo menos uma vez. Use DO...UNTIL quando uma operação será sempre executada no mínimo uma vez e possivelmente mais vezes.

DO...WHILE

Sintaxe

DO <*bloco de comandos*> WHILE <*expressão*>

```
DO INC Var1 WHILE Matriz1[Var1] <> 10
DO
        BEGIN
                INC Var2,5
                WAIT 5 sec
        END
WHILE FreeCap(Loc1) > 5
```

Descrição

Repete um comando ou bloco de comandos continuamente enquanto a condição se mantiver verdadeira. DO...WHILE é um loop com uma condição de saída, significando que o loop será sempre executado pelo menos uma vez. Use DO...WHILE para processos que devem ser executados pelo menos uma vez e possivelmente outras. Tenha cuidado quando usar com uma função do sistema (por exemplo, FREECAP()) para assegurar que o sistema não entre em um loop infinito. Por exemplo, eliminando o "WAIT 5 sec" no exemplo acima irá fazer com que o sistema entre em um loop infinito porque não há um intervalo de tempo dentro do loop.

DOSLOAD

Sintaxe

DOSLOAD <*expressão*> {IN <*tempo*>} {,*prioridade*}

DOSLOAD 5,99

DOSLOAD 5 IN 30 min

DOSLOAD Var1 IN 2 Hr, 99

Descrição

Carrega um número específico de entidades e muda o nome da entidade original para o nome da última entidade que foi carregada. Este comando é fornecido para compatibilizar com os modelos feitos no ProModel*PC* (antiga versão DOS do ProModel). As entidades não podem ser descarregadas depois de serem carregadas com o comando DOSLOAD. Elas são unidas ou combinadas em uma entidade. O comando DOSLOAD automaticamente transfere as propriedades de qualquer entidade carregada para a entidade base. Se você não quiser que isto aconteça, você deve usar os comandos LOAD e RENAME.

DOWN

Sintaxe

DOWN <nome do downtime>, {<prioridade>}

Descrição

Ativa a parada de um local por este comando, desde que a parada seja pré-definida anteriormente como parada por chamada. Quando este comando é executado, será acionada a parada que tentará executar sua lógica. A execução da mesma dependerá da condição do local e da prioridade da chamada da parada. Pode ser utilizada em qualquer lógica do Promodel exceto nas lógicas de inicialização e término.

DOWNQTY()

Sintaxe

DOWNQTY(<*local*> ou <*recurso*>)

IF DOWNQTY(Loc1) > 3 THEN ROUTE 2

DISPLAY “Total de Res1 parados agora:” $ DOWNQTY(Res1)

Descrição

Retorna o número de unidades do local ou recurso parados naquele instante. Use esta função para tomar decisões baseadas em como unidades do local ou recurso estão parados. Por exemplo, se muitas unidades estão paradas, uma ordem pode chamar várias unidades de volta para o serviço.

DTDELAY()

Sintaxe

DTDELAY(<*unidade de tempo*>)

WAIT Attr1 – DTDELAY(Min)

DISPLAY “O atraso da parada foi de” $DTDELAY(Days)$ ”dias”

Retorna a diferença entre o momento da simulação em que a parada não-premptiva estava programada para ocorrer e o momento no qual ela realmente ocorreu. Use DTDELAY para determinar paradas que foram adiadas porque o trabalho ainda não estava completo. A função DTDELAY pode ser usada na lógica de parada (downtime logic) para assegurar que um local volte ao trabalho em um momento específico.

Também retorna a diferença entre o momento no qual a parada é preemptada e o momento em que ela é executada.

DTLEFT()

Sintaxe

DTLEFT <*unidade de tempo*>

Shift_Time = DTLEFT()

Descrição

A função DTLEFT retorna o valor do tempo que falta para que a parada se acabe e o local ou recurso volte ao trabalho. O valor retornado será na unidade especificada na tabela General Information a não ser que seja especificada outra unidade na função, por exemplo, DTLEFT(hr).

DYNPLOT()

Sintaxe

DYNPLOT "<nomedografico>"

Descrição

É utilizado para fazer a mudança automática para um gráfico dinâmico pré-definido. Seu uso é similar ao comando "view". São pré-definidos durante o funcionamento da simulação e podem ser chamadas de qualquer parte da lógica do Promodel.

Exemplo

Se após um período de duas horas deseja-se obter um gráfico do valor do estoque em processo (WIP) versus custo e depois de quatro horas deseja-se que este gráfico dinâmico desapareça, basta utilizar o seguinte comando:

WAIT 2 HR

DYNPLOT "Wip vs Custv"

Wait 2 HR

DYNPLOT ""

Nota:

Usando a sintaxe sem nenhum nome entre aspas faz com que a janela aberta com o gráfico dinâmico seja fechada.

END

Sintaxe

END ou }

```
WHILE FreeCap(Loc1) > 5 DO
    BEGIN
        INC Var2,5
        WAIT 5 sec
    END
```

Descrição

Define um bloco de comandos juntamente com o comando BEGIN correspondente. BEGIN e END são quase sempre usados em conjunto com outros comandos como IF-THEN e DO...WHILE. Para todo comando END deve haver um comando BEGIN correspondente.

ENT()

Sintaxe

ENT(<*número de nome-índice da entidade*>)

```
SEND 10 ENT(Var1) TO Loc1
DISPLAY "Ent A foi combinada com " $ ENT(Var1)
```

Descrição

Converte um número de nome-índice ou um valor inteiro no nome da entidade. Use esta função quando um comando ou função precisa do nome da entidade cujo nome-índice foi guardado em um atributo, variável, ou alguma outra expressão. Também pode ser usada para variar a entidade a qual o comando se refere usando uma expressão para o número de nome-índice. Quando usada em uma expressão esperando por uma série (string), como na segunda sintaxe acima, o ProModel irá converter o número de nome-índice para o nome da entidade atual.

ENTITY()

Sintaxe

ENTITY({<*expressão*>})

ENTITY()

ENTITY(Var1)

Descrição

Retorna o número de nome-índice da entidade corrente ou de uma entidade particular em um grupo de entidades. Esta função é especialmente útil em macros e subrotinas que variam dependendo da entidade que as chama. Use ENTITY() para determinar que tipo de entidade esta sendo processada quando o tipo ALL está especificado no local. Por exemplo, se uma área comum suporta várias peças diferentes com praticamente o mesmo processo, use um comando IF-THEN em conjunto com a função ENTITY() para ter blocos de comandos individuais para detalhar a operação. Esta função retorna um valor inteiro.

ENTRIES()

Sintaxe

ENTRIES(<*local*>)

DISPLAY “LocA teve ” $ ENTRIES(LocA) $ “entradas.”

Descrição

Retorna o total de entradas de um local. Esta função retorna um valor inteiro.

EXP()

Sintaxe

EXP(<*expressão*>)

Real1 = EXP(Real2)

Descrição

Retorna o exponencial de uma expressão. Esta função é equivalente a e^x.

FORLOCATION()

Sintaxe

```
FORLOCATION()
IF FORLOCATION() THEN PRIORITY 100
IF FORLOCATION() THEN
        INC Arr1[1,2]
ELSE
        INC Arr1 [2,2]
```

Descrição

Esta função retorna TRUE se o objeto que estiver executando a lógica de turno (shift) ou de parada (break) for um local.

FORMAT()

Sintaxe

FORMAT(<*expressão*>,<*dígitos antes do decimal*> {,<*dígitos após o decimal*>})

DISPLAY "O valor da Var1 é " $ FORMAT(Var1,5)

Descrição

Converte um número em um string no formato especificado. Quase sempre FORMAT() é usado com o símbolo ("$") e outro string, como na sintaxe acima. FORMAT é usualmente usado em conjunto com o comando XWRITE para produzir uma saída formatada.

FORRESOURCE()

Sintaxe

FORRESOURCE()

IF FORRESOURCE() THEN GET Res1

Descrição

Esta função retorna TRUE se o objeto que estiver executando a lógica de turno (shift) ou de parada (break) for um recurso.

FREE

Sintaxe

FREE {<*quantidade*>} <*recurso*>, {{<*quantidade*>}<*recurso*>...}

FREE Res1,2 Res2,5 Res3

FREE ALL

FREE RES(Attr1)

Descrição

Libera os recursos que estão "possuídos" pela entidade corrente. Estes recursos devem ter sido capturados pela entidade com os comandos GET ou JOINTLY GET.

FREECAP()

Sintaxe

FREECAP(<*local*>)

SEND FREECAP(Loc1) EntA TO Loc1

Descrição

Retorna a capacidade disponível de um local. Esta função retorna um valor inteiro.

FREEUNITS()

Sintaxe

FREEUNITS(<*local*> ou <*recurso*>)

USE (FREEUNITS(Res1)) Res1 FOR 5 min

Descrição

Retorna as unidades livres de um local ou recurso.

GET

Sintaxe

GET {<quantidade>} <recurso> {,<prioridade1>{,<prioridade2>}}

{AND ou OR {quantidade} <recurso> {,<prioridade1>{,<prioridade2>}}}

GET Res1

GET Crane3, 5, 32

GET Res3, 20 AND (Crane5, 15 OR Crane2, 12, 40)

GET 2 Res2

Descrição

Captura um número específico de recursos assim que estes estiverem disponíveis. Se a entidade já possui um dos recursos solicitados, a entidade vai tentar capturar uma unidade adicional deste recurso. Quando estiver capturando múltiplos recursos, cada recurso será capturado assim que estiver disponível até que todos os recursos sejam capturados.

Um recurso capturado com o comando GET em um local e depois liberado com um comando FREE em outro local não será utilizado para movimentar a entidade entre os locais a não ser que isto seja especificado com o comando MOVE WITH na lógica de movimentação. O ProModel usa a entidade que vai estar movimentando o recurso de um local para o outro e o recurso não será visível quando estiver se movendo com a entidade.

Excetuando-se na lógica de movimentação, os recursos capturados com o comando GET podem ser premptados apenas quando a entidade que estiver possuindo o recurso estiver executando um tempo de WAIT ou USE. Se o recurso é premptado do durante um destes tempos, o tempo continuará, de onde parou, quando o recurso estiver disponível.

Para todo GET deve haver o FREE correspondente ou um erro ocorrerá quando a entidade sair do sistema. Se uma entidade possui um ou mais recursos e é subsequentemente carregada ou agrupada com outra entidade, o recurso não será liberado até que ela seja descarregada ou desagrupada.

GETCOST()

Sintaxe

GETCOST()

GETCOST()

Descrição

Retorna o custo da entidade corrente que está executando a lógica. Use esta função para retornar o custo real acumulado da entidade até aquele local ou momento da simulação.

GETREPLICATIONNUM()

Sintaxe

GETREPLICATIONNUM()

Descrição

Retorna o numero da replicação atual. Pode ser utilizada em qualquer parte da lógica do Promodel.

GETRESRATE()

Sintaxe

GETRESRATE({<*recurso*>})

GETRESRATE()

GETRESRATE(Operador1)

Descrição

Retorna o fator de custo especificado na janela de custos (Cost dialog) ou através da função SETRATE() para o recurso capturado pela entidade que chamou a função. Quando usado sem o parâmetro opcional <*recurso*>, esta função retorna o fator de custo do recurso que foi capturado mais recentemente pela entidade.

Se uma entidade captura múltiplas unidades de um recurso, a função retorna o fator de custo da unidade do recurso que foi capturado mais recentemente pela entidade.

GOTO

Sintaxe

GOTO <label ID>

GOTO LabelA

Descrição

Pula para o comando identificado pelo rótulo (label) designado. Um rótulo deve seguir as regras normais para os nomes, com a exceção de que é seguido por dois pontos (:) na lógica. O comando GOTO pode ser substituído por comandos IF...THEN...ELSE.

GRAPHIC

Sintaxe

GRAPHIC <*expressão*>

GRAPHIC 2

GRAPHIC Var1

Descrição

Muda o aspecto gráfico corrente de uma entidade ou recurso. O ProModel designa gráficos da biblioteca gráfica (graphic library) para entidades e recursos através do editor de entidades e de recursos. Use o comando GRAPHIC para mostrar o resultado de um processo. Por exemplo, quando um avião tem suas asas colocadas, o aspecto gráfico pode mudar de uma fuselagem sem asas para uma fuselagem com asas.

GROUP

Sintaxe

GROUP<*expressão*> {AS <nome da entidade>}

GROUP (Var1+Var2)

GROUP 10 AS EntX

Descrição

Acumula e temporariamente consolida uma quantidade específica de entidades em uma única entidade. As entidades individuais formando o grupo mantêm sua identidade, atributos, e recursos e são separadas do grupo quando o comando UNGROUP é encontrado. A primeira entidade processada do grupo toma todos os recursos que o grupo possui. As entidades de um grupo podem ser agrupadas a um grupo maior, em outro local.

GROUPQTY()

Sintaxe

GROUPQTY({<nome da entidade>})

ORDER GROUPQTY(Part1) Part2 TO Loc1

IF GROUPQTY(Part1) > 5 THEN ...

Descrição

Retorna o número de entidades de um tipo específico em uma entidade agrupada ou carregada. Se não for especificado nenhum nome, retorna a quantidade total do grupo. Se for uma entidade carregada, irá retornar somente o número de entidades carregadas, e não a entidade base. Por exemplo, se quatro peças forem carregadas em um Pallet e se for dado ao conjunto o nome de Grupo, a função GROUPQTY() irá retornar o número de peças (por exemplo 4), o que não inclui o Pallet.

No caso de grupos híbridos com vários níveis de grupos e carregamentos misturados, a função GROUPQTY() retorna o número de entidades apenas do último nível.

IF...THEN...ELSE

Sintaxe

IF <*expressão*> THEN <comando> {ELSE <comando2>}

IF Var1 = 5 THEN WAIT 2

IF (Attr2 = 5) OR (Var5 <>0) THEN WAIT 2 min ELSE WAIT 3 min

```
IF Var1 > Attr2 THEN
        BEGIN
                Var1 = Attr2
                WAIT Attr1
        END
ELSE
        BEGIN
                INC Var1
                WAIT Attr2
        END
```

Descrição

Executa um comando ou bloco de comandos se a expressão for verdadeira. Se um comando ELSE estiver incluso e a expressão for falsa, um comando ou bloco de comandos alternativo será executado. Para que o comando IF-THEN seja quebrado em mais de uma linha, o primeiro item da linha seguinte deve ser THEN, ou AND, ou OR. O comando IF-THEN somente se aplica ao próximo comando ou bloco de comandos em uma lógica. Qualquer comando fora do BEGIN e END será executado normalmente. Veja exemplos em BEGIN e END.

INC

Sintaxe

INC <nome>{,<*expressão*>}

INC Var1

INC Attr2, 5+Var1

Descrição

Incrementa uma variável, elemento de matriz, ou atributo, pelo valor de uma expressão numérica especificada. Quando estiver contando o número de entidades que foram processadas em um local, incremente a variável no final da lógica do processo.

INCENTCOST

Sintaxe

INCENTCOST <*expressão*>

INCENTCOST 15

INCENTCOST –15

Descrição

Possibilita que você incremente o custo (positiva ou negativamente) da entidade corrente de uma certa quantidade. Use esta função para adicionar custos ao custo atual acumulado da entidade corrente.

INCLOCCOST

Sintaxe

INCLOCCOST <*expressão*>

INCLOCCOST 15

INCLOCCOST –15

Descrição

Possibilita que você incremente o custo (positiva ou negativamente) do local corrente, por uma certa quantidade. Use esta função para adicionar custos ao custo atual acumulado do local corrente.

INCRESCOST

Sintaxe

INCRESCOST <*expressão de custo*> {,<*recurso*>}

INCRESCOST 10

INCRESCOST GETRESRATE(Operador1)*20, Operador1

Descrição

Possibilita que você incremente o custo (positiva ou negativamente) de um recurso capturado pela entidade que estiver executando o comando. Use esta função para adicionar custos ao custo atual do recurso. Quando utilizado sem o parâmetro opcional *<recurso>*, este comando incrementa o fator de custo do recurso mais recentemente capturado pela entidade.

Se uma entidade tem muitas unidades de um recurso, o custo é distribuído equilibradamente para cada unidade.

INT

Sintaxe

INT <*nome1*>{=<*expressão*>,<*nome2*>=<*expressão2*>...}

INT Contador

INT Contador = 1

INT Contador = 1, Test = FREECAP(Loc2)

Descrição

Cria uma variável local do tipo inteiro. Variáveis locais trabalham quase da mesma maneira que os atributos, exceto que elas só estarão disponíveis dentro da lógica que as declarou. Uma nova variável será criada para cada entidade que encontrar o comando INT. As variáveis locais não estão diretamente disponíveis para subrotinas, os quais possuem suas próprias variáveis locais. Entretanto, uma variável local pode ser passada para uma subrotina como um parâmetro. As variáveis locais estão disponíveis dentro das macros que são referenciadas.

Use variáveis locais onde for possível para testar as variáveis nos loops WHILE...DO, DO...WHILE, e DO...UNTIL.

JOIN

Sintaxe

JOIN <*expressão*> <*nome da entidade*> {,<*prioridade*>}

JOIN 4 EntA

JOIN Var1 EntA, 1

Descrição

Une uma quantidade específica de um tipo designado de entidade à entidade corrente. As entidades unidas à entidade corrente perdem sua identidade e quaisquer recursos possuídos pelas entidades unidas serão automaticamente transferidos para a entidade corrente. Use JOIN

para simular um componente sendo montado na peça principal, tal como quando asas são montadas na fuselagem do avião.

As entidades que serão unidas, devem estar roteadas para o local corrente através da regra JOIN. A entidade corrente espera até que tenham sido roteadas entidades suficientes que preencham suas exigências, e que também tenham sido roteadas para o local corrente através da regra JOIN.

A entidade resultante conserva os atributos e o nome da entidade corrente. Para transferir atributos da entidade que vai ser unida para a entidade corrente, copie o atributo desejado para uma variável global na lógica de saída da entidade que vai ser unida. Depois designe a variável global ao atributo da entidade corrente após o comando JOIN na lógica do processo.

Todos os recursos 'possuídos' pela entidade que será unida serão transferidos para a entidade base.

Para unir uma entidade com um valor de atributo específico, a outra entidade com o mesmo valor de atributo, use o comando MATCH adicionalmente ao comando JOIN.

JOINTLY GET

Sintaxe

JOINTLY GET {<*quantidade*>} <*recurso*> {,<*prioridade1*>{,<*prioridade2*>}}

AND ou OR {<*quantidade*>} <*recurso*> {,<*prioridade1*>{,<*prioridade2*>}}}

JOINTLY GET 3 Res1,5

JOINTLY GET 2 Res1 OR 3 Res2

JOINTLY GET Crane2, 15, 20 AND (Res2 OR Res3)

JOINTLY GET 2 Res(Atributo1)

Descrição

Captura um número específico de recursos quando este número de recursos está disponível. Quando estiver capturando vários recursos, nenhum dos recursos será capturado até que todos estejam disponíveis. Se a entidade já possuir um dos recursos solicitados, a entidade ainda assim vai tentar capturar a quantidade especificada.

LAST()

Sintaxe

LAST()

Var1=LAST()

IF LAST() = 13 THEN Var3 = 0

IF LAST() = PathNet1.N1 THEN INC Var1

Descrição

Retorna o número de nome-índice do nó do qual o recurso acaba de sair. LAST() pode ser útil para escolher o gráfico apropriado ou para reajustar (reset) uma variável. Você pode também checar o número de nome-índice do último nó especificando <nome da caminho (path network)>.<nome do nó>. Por exemplo, se você quer saber se o último nó foi o N5 do caminho Net3, você pode especificar "IF LAST() = Net3.N5 THEN ..." na lógica de entrada do nó (node entry logic).

LN()

Sintaxe

LN(<*expressão*>)

Real1 = LN(Real2)

Descrição

Retorna o logaritmo natural de uma expressão.

LOAD

Sintaxe

LOAD <*expressão*> {IFF <*expressão*>} {IN <tempo>} {,<prioridade>}

LOAD 5, 99

LOAD 5 IFF Attr3 > 2 IN 5 min

LOAD Capacidade_do_Pallet

Descrição

Carrega uma quantidade específica de entidades na entidade corrente. As entidades carregadas mantêm sua identidade e podem ser descarregadas com o comando UNLOAD. As entidades carregadas devem ser roteadas para o local de carregamento usando a regra de roteamento LOAD. Entidades adicionais podem ser acrescentadas à entidade carregada anteriormente, através de comandos LOAD adicionais. Use LOAD para modelar peças colocadas em um container ou pallet, quando elas precisam ser removidas mais tarde. Se um recurso possui a entidade carregada quando as entidades são carregadas sobre a entidade base, o recurso fica com a entidade base.

LOC()

Sintaxe

LOC(<*número de nome-índice do local*>)

ORDER 10 EntA TO LOC(5)

DISPLAY “EntA chegou no” $ LOC(5)

Descrição

Converte o número de nome-índice ou valor inteiro em um nome de local. Use esta função quando um comando ou função precisa do nome de um local cujo número de nome-índice está guardado em um atributo, variável, ou alguma outra expressão. Quando usando em uma expressão string como no exemplo da segunda sintaxe acima, o ProModel irá converter o número de nome-índice para o nome do local. Se a expressão aponta para um local de múltiplas unidades, a função LOC() irá retornar o nome do local principal.

LOCSTATE()

Sintaxe

LOCSTATE (<nomedolocal>)

Descrição

Retorna um valor indicando o estado atual de determinado local. Os valores retornados irão variar entre 1 e 7 que significam respectivamente:

1 = ocioso/vazio;
2 = setup;
3 = operação;
4 = bloqueado;
5 = esperando;
6 = em atividade (múltipla capacidade);
7 = parado.

LOCATION()

Sintaxe

LOCATION()

Attr1 = LOCATION()

IF LOCATION() = 2 THEN WAIT 4 min

Descrição

Retorna o número de nome-índice do local corrente. Esta função é especialmente útil em macros e subrotinas que variam dependendo da lógica do local que as chama. Usando a função LOCATION() com o comando IF-THEN, a macro ou subrotina pode agir diferentemente dependendo do local que a chamou. Além disso, a mesma técnica pode ser usada para determinar que local está sendo executado quando ALL é usado como local de processo.

LOG

Sintaxe

LOG <*string*>,<*expressão*>

LOG “Tempo da Atividade”, Attr1

Descrição

Subtrai o valor do parâmetro ‘expressão’ do tempo de simulação corrente, e guarda o resultado com uma string em um arquivo texto chamado <nome do modelo>.LAP quando não há cenários; e L01 para o primeiro cenário, L02 para o segundo cenário, e assim por diante quando cenários estão sendo executados. O ProModel assume que o tempo guardado na expressão está na unidade de tempo padrão escolhida na janela General Information. Use o comando LOG para gravar o tempo entre um comando e outro, armazenando o tempo do primeiro comando em um atributo, variável, ou elemento de matriz com CLOCK() e usando o mesmo atributo, variável, ou elemento de matriz como a expressão no comando LOG. Use o comando LOG para determinar o tempo de produção ou ao produção em um determinado trecho da planta.

MAPARR

Sintaxe

MAPARR <nome da matriz>{TO<nome da variável>}

MAPARR Array1 TO Var10

MAPARR Array5

Descrição

Começando com uma variável que você especifica, o comando MAPARR mapeia cada célula individual de uma matriz para uma variável única (isto é, se você definir 12 células para a matriz, a matriz será mapeada para 12 variáveis). Para mostrar o valor da célula de uma matriz mapeada, crie um gráfico de variável para a variável à qual você mapeou a célula da matriz. O ProModel coleta estatísticas para uma célula da matriz através da variável à qual você mapeou a célula.

(Escolha estatísticas “Básicas” ou “Séries Temporais” para um variável mapeada, em seguida observe o comportamento da variável no módulo de resultados estatísticos).

Se você não especificar o nome da variável opcional no comando, o ProModel vai desmapear a matriz das variáveis para à qual você a mapeou originalmente. Você pode remapear matrizes usando o comando MAPARR novamente.

Vide exemplo no “Reference Manual” do ProModel, para verificar em que ordem as células da matriz referem-se às variáveis.

MATCH

Sintaxe

MATCH <*atributo*>

MATCH Attr1

Descrição

Faz com que a entidade corrente espere até que o valor do atributo especificado seja igual ao valor do mesmo atributo de outra entidade. As duas entidades devem ter um comando MATCH especificado para o mesmo nome de atributo e podem estar em qualquer local do modelo, incluindo o mesmo local e dois locais sem relação alguma. Entretanto, você deve quase sempre designar o valor do atributo para que este corresponda (MATCH) ao valor de uma variável global incrementada por cada entidade que você correspondeu (matched), como pode ser visto no exemplo do manual (reference guide).

Os locais usando o comando MATCH geralmente devem ser de múltipla capacidade pois caso contrário, eles não conseguirão processar nenhuma outra entidade até que a correspondência (match) seja feita. Além disso, os locais usando MATCH geralmente devem ser do tipo sem enfileiramento (non-queuning) para permitir que as entidades saiam quando a correspondência estiver feita.

Use o comando MATCH para corresponder duas partes específicas antes de montá-las, ou para equivaler uma ordem de trabalho com uma tarefa completada.

MOVE

Sintaxe

MOVE {FOR <*expressão de tempo*>}

MOVE FOR .25 min

MOVE FOR 3.2

MOVE

Descrição

Move a entidade para o fim de uma fila (queue) ou esteira transportadora (conveyor). Use o comando MOVE para controlar o movimento de uma entidade em uma fila ou conveyor.

Se não houver nenhum comando MOVE, quando uma entidade entra em uma fila ou conveyor, ela irá executar toda a lógica do processo definida para a entidade naquele local e depois se mover para o fim da fila ou conveyor. Para filas, seu tempo de movimentação é baseado na velocidade da entidade e no comprimento da fila.

Se uma entidade que vai ser processada em uma fila ou conveyor encontrar o comando MOVE, a entidade para de executar a lógica do processo, move-se até o fim da fila ou do conveyor durante a quantidade de tempo apropriada, e depois continua a lógica do processo. O tempo de movimentação de uma entidade em um conveyor é calculado usando a seguinte fórmula:

Tempo = (Comprimento do Conveyor – Comprimento ou Largura da Entidade)/Velocidade do Conveyor

Somente para as filas, o comando MOVE opcionalmente pode ser seguido do comando FOR e depois de um tempo de movimentação. Se um tempo de movimentação é especificado, a entidade se move através da fila durante o tempo de movimentação especificado, sem considerar a velocidade da entidade e o comprimento da fila. As entidades com menores tempos de movimentação irão alcançar, mas não ultrapassar, as com um tempo de movimentação maior.

Se uma fila não estiver vazia quando uma entidade estiver entrando, o tempo de movimentação continuará a ser contado mesmo se o gráfico da entidade parar de se movimentar. Quando o tempo de movimentação da entidade acabar, a entidade começará a executar a lógica que segue o comando MOVE e depois estará disponível para roteamento mesmo que o gráfico da entidade não pareça estar no fim da fila.

Para um conveyor, se alguma lógica adicional seguir o comando MOVE, a entidade deve avançar para a última posição do conveyor antes que a lógica restante seja executada.

MOVE FOR

Sintaxe

MOVE FOR <*tempo*>

MOVE FOR 0

MOVE FOR 2.5 + Tempo_de_Limpeza

MOVE FOR N(8,.5) + 3 sec

Descrição

Usado para especificar a quantidade de tempo exigida para a movimentação da entidade. Um tempo de movimentação igual a zero pode ser especificado para fazer com que eventos de outras entidades que estejam ocorrendo no mesmo tempo de simulação sejam processados, antes que seja processada qualquer lógica adicional para a entidade corrente. Se nenhum tipo de comando de movimentação (MOVE FOR, MOVE ON, MOVE WITH) for especificado, a entidade instantaneamente entra no local destinado e, imediatamente, começa a executar a lógica de operação para este local.

MOVE ON

Sintaxe

MOVE ON <*caminho*>

MOVE ON StatPath2

Descrição

Use este comando para mover uma entidade por um caminho (path network). As entidades não podem se mover por caminhos do tipo 'pontes rolantes' usando este comando. Veja MOVE WITH.

MOVE WITH

Sintaxe

MOVE WITH <*rec1*> {,{*prior1*}{,{*prior2*}{,{*prior3*}}}}

OR <*rec2*> {,{*prior1*}{,{*prior2*}{,{*prior3*}}}}

{FOR <*tempo*>} {THEN FREE}

MOVE WITH Ponte3, 99, 230, 425

MOVE WITH Operador1, 399 FOR 3 min

MOVE WITH Caminhao1, 99 THEN FREE

MOVE WITH Operador1 OR Operador2

Descrição

Este comando é usado para mover uma entidade usando um recurso designado, como uma ponte rolante ou uma empilhadeira. Com o operador OR, você pode designar recursos alternativos para movimentar a entidade. Neste caso, o comando captura o primeiro recurso designado na expressão que estiver disponível e realiza a movimentação. Assim que o local de destino fica disponível a entidade captura o recurso. Entretanto, se a entidade já possui um dos recursos ela irá usar este recurso.

O comando também permite que você determine uma prioridade (prior1) para acessar o recurso designado. Se a entidade já possui o recurso, esta prioridade é ignorada. Se o recurso for uma ponte rolante, você pode determinar outras duas prioridades: prior2 para a movimentação para pegar a entidade e prior3 para a movimentação da entrega da entidade. Estas prioridades são úteis em situações onde duas pontes usam o mesmo trilho na movimentação para pegar ou entregar a entidade.

Se o recurso é estático, você pode especificar um tempo (FOR <*expressão de tempo*>) para a movimentação. Se o recurso é dinâmico, um tempo (FOR <*expressão de tempo*>) não é valido. Se você utiliza (FOR <*expressão de tempo*>) com um recurso dinâmico, o ProModel ignora o tempo. O recurso irá mover-se baseado seja no tempo, seja na velocidade/distância definida no módulo de caminhos (path networks).

O recurso usado para realizar a movimentação só é liberado se a opção THEN FREE for utilizada.

NEXT()

Sintaxe

NEXT()

Var1 = NEXT()

IF NEXT() = PathNet5.N11 THEN Var5 = 3

Descrição

Retorna o número de nome-índice do nó de destino do recurso. Use NEXT() para determinar em que direção a entidade se dirige, e para escolher o aspecto gráfico apropriado. Esta função pode ser usada para controlar a interferência entre múltiplos transportadores no mesmo caminho (path network). Você também pode checar o número de nome-índice do próximo nó especificando <*nome do caminho*>.<*nome do nó*>. Por exemplo, se você quiser saber se o próximo nó é o nó N5 do caminho Net3, você poderia especificar "IF NEXT() = Net3.N5 THEN ..." na lógica do nó (node entry logic).

ORDER

Sintaxe

ORDER <*expressão*> <*entidade*> {TO <*local*>}

ORDER 10 EntA TO Loc2

ORDER Attr_Qdd_Ordem ENT(Attr_da_Entidade) TO LOC(Attr_do_Loc)

Descrição

Faz com que um número específico de entidades seja criado e colocado no sistema, em um local designado. Se o local não tiver capacidade suficiente para todas as novas entidades, o excesso será destruído. Os atributos da entidade criadora serão copiados para as entidades criadas. Use ORDER para reabastecer inventários quando uma condição particular ocorrer, como por exemplo quando o inventário atinge o nível mínimo.

OWNEDRESOURCE()

Sintaxe

OWNEDRESOURSE ({<*expressão*>})

OWNEDRESOURSE(2)

OWNEDRESOURSE(RESQTY())

OWNEDRESOURSE()

Descrição

Retorna o número de nome-índice do enésimo recurso que está sendo usado pela entidade ou pela parada (downtime) que está fazendo a função de chamada. O parâmetro da função indica a posição no recurso na lista cronológica de recursos utilizados pela entidade. Por exemplo, OWNEDRESOURCE(1) retorna o recurso utilizado a mais tempo da lista de recursos possuídos e assim por diante.

Quando usado sem um parâmetro, esta função retorna o último recurso que a entidade capturou e ainda possui. Se o valor do parâmetro está fora do alcance da lista de recursos, ou se a entidade ou parada (downtime) no momento não possui um recurso, a função irá retornar o valor 0 (zero) sem nenhum aviso ou mensagem de erro.

Um recurso premptado NÃO é removido da lista mas marcado temporariamente para indicar que o que o premptou não possui o recurso. Depois da preempção, quando o recurso continua o processo original, ele conserva sua posição original na lista.

PAUSE

Sintaxe

PAUSE {<*expressão*>}

PAUSE

PAUSE “Var1 = ” $ Var1

PAUSE “Alcançou a metade da simulação.”

Descrição

Executa uma pausa na simulação e opcionalmente mostra uma mensagem em algum ponto de interesse. Esta pausa permite ao usuário examinar o sistema com detalhes. Uma janela de informação aparecerá na tela quando a pausa ocorrer. A simulação continuará apenas quando o usuário selecionar a opção Resume Simulation no menu Simulation.

PERCENTOP()

Sintaxe

PERCENTOP (<nomedolocal>)

Descrição

Retorna a porcentagem acumulada do tempo de operação de um local de capacidade unitária. Este valor representa a porcentagem acumulativa do tempo em que um local processou uma determinada entidade até o ponto onde a função foi chamada. Se esta função for utilizada para um local de múltipla capacidade, o valor retornado será sempre zero, já que esta função só serve para locais de capacidade unitária. Pode ser utilizada em qualquer parte da lógica do Promodel.

PERCENTUTIL()

Sintaxe

PERCENTUTIL (<nomedolocal>)

Descrição

Retorna a porcentagem acumulada de utilização de um local especifico. O valor retornado representa a porcentagem acumulativa da capacidade ocupada por este local até o momento em que a função é chamada. Pode ser utilizada em locais de múltipla capacidade e capacidade unitária e pode ser usado em qualquer parte da lógica do Promodel.

PREEMPTOR()

Sintaxe

PREEMPTOR()

Var1 = PREEMPTOR()

Descrição

Identifica se é uma parada (downtime) ou uma entidade que está solicitando a preempção. A função retorna o número de nome-índice da entidade que está premptando; entretanto, a função retornará o valor 0 se o premptor for uma parada.

PRIORITY

Sintaxe

PRIORITY <*expressão*>

PRIORITY 199

Descrição

Este comando é usado para mudar a prioridade do estado off-line do local ou recurso. Se a prioridade é menor que o valor ajustado anteriormente, o sistema irá checar se você pode preemptar o recurso ou local.

PROMPT

Sintaxe

PROMPT <*expressão*>, <*nome*> {,<*escolha1*>:<*expressão1*>,

<*escolha2*>:<*expressão2*>,<*escolha3*>:<*expressão3*>...}

PROMPT “Digite o número de entidades do processo:”, Var2

PROMPT “Digite o tamanho do lote a ser feito:”, Var1, “Grande”:20, “Medio”:15 “Pequeno”:10

Descrição

Pausa a simulação e mostra seja uma mensagem com um campo de entrada, seja um menu para que se escolha uma opção. O valor digitado ou selecionado é alocado para uma variável, elemento de matriz, ou atributo em questão. Para que o comando PROMPT apresente um menu, especifique uma ou mais escolhas como na Segunda sintaxe acima. O valor atual da variável, elemento de matriz, ou atributo é usado como padrão na caixa de diálogo. Um dos usos do comando PROMPT é a de dar a opção de mudar o tempo de operação representado por uma variável durante a simulação.

RAND()

Sintaxe

RAND(<*expressão*>)

RAND(10) min

ORDER RAND(10) EntA TO Loc1

IF RAND(100) > 45 THEN ROUTE 1

Descrição

Retorna um valor aleatório n entre 0 e X ($0 \leq n < X$) onde X é o resultado da expressão.

Para gerar números aleatórios entre a expressão e um número diferente de zero use a expressão como o alcance entre o valor máximo e o valor mínimo. Depois adicione o valor mínimo ao número aleatório. Por exemplo, a expressão Attr1 = 2 + RAND(3) gera um número aleatório maior que 2 e menor que 5. Um método alternativo é usar uma distribuição uniforme.

Esta função retorna um número real mas que pode ser convertido para um inteiro. Para fazer com que a função RAND() gere valores aleatórios inteiros entre zero e um valor máximo, use a fórmula, Valor_Inteiro = RAND(X+1), onde X é o máximo valor inteiro que pode ser retornado pela função. Por exemplo, para gerar um valor aleatório e inteiro entre 0 e 6, use a fórmula, Valor_Inteiro=RAND(7). A parte RAND(7) da fórmula vai gerar um valor real entre 0 e 6.999, mas que será truncado para um número inteiro entre 0 e 6. Portanto, quando estiver gerando valores inteiros, verifique se o resultado da função RAND() está designada a um valor inteiro, ou é usada em uma expressão onde será truncada para um inteiro.

READ

Sintaxe

READ <*identificação do arquivo*>,<*nome*>

READ File1, Var1

Descrição

Lê o próximo valor numérico de um arquivo de leitura geral e designa este valor a um nome. Os arquivos externos são definidos no External Files Editor (Editor de Arquivos Externos). Quando está lendo de um arquivo, o ProModel ignora os dados não numéricos, tais como textos, e lê o próximo valor numérico. Assim, comentários e observações podem estar incluídos sem problema em um arquivo de leitura. Note que o ProModel vai ler um ponto (.) existente em um arquivo externo como sendo um valor zero. Para evitar que isto aconteça, você deve usar o símbolo de comentário (#) antes das observações e comentários que contêm um ponto. Múltiplas replicações do modelo irão continuar lendo os dados do arquivo do ponto em que a replicação anterior parou, a não ser que este seja 'zerado' com o comando RESET.

O comando READ pode ler arquivos ASCII. A maioria das planilhas eletrônicas podem converter as planilhas em arquivos ASCII (.TXT) e arquivos delimitados por vírgulas (.CSV – Comma Delimited Files).

Se o comando READ não estiver alocando os valores corretos aos nomes apropriados, pode haver algum tipo de informação numérica nas observações e comentários de cabeçalho contidos no início do arquivo. Além disso, se os valores estão sendo lidos para uma matriz, os índices da matriz podem não estar sendo incrementados corretamente entre as leituras.

Se você for ler um arquivo de leitura mais de uma vez em um modelo, ele pode precisar ser reiniciado (reset). Uma maneira de disser quando um arquivo precisa ser reiniciado (reset) é através de um marcador de fim de arquivo, tal como 9999 no final do arquivo e as seguintes duas linhas de lógica:

READ File1, Valor

IF Valor = 9999 THEN RESET File1

REAL

Sintaxe

REAL <*nome1*>{=<*expressão1*>,<*nome2*>=<*expressão2*>...}

REAL Var1

REAL Counter = 0

REAL Var1 = CLOCK(SEC), Random_Num = RAND(10)

Descrição

Cria uma variável local do tipo real. As variáveis locais funcionam quase da mesma maneira que os atributos, exceto que elas só estarão disponíveis dentro da lógica que as declara. Uma variável local será criada para cada entidade que encontra o comando REAL. As variáveis locais não estão diretamente disponíveis para subrotinas, as quais possuem suas próprias variáveis locais. Entretanto, uma variável local pode ser passada para uma subrotina como um parâmetro. As variáveis locais estão disponíveis para macros.

REAL()

Sintaxe

REAL (<*expressão*>)

Var2 = Var1 + REAL(Var3)

Attr3 = 1.05 * REAL(Var5)

Descrição

Converte um número inteiro para um real. As versões do ProModel superiores à 2.0 convertem automaticamente números inteiros para reais quando necessário. Esta função foi incluída no ProModel para se manter a compatibilidade com as versões anteriores.

RENAME

Sintaxe

RENAME {AS} <novo nome da entidade>

RENAME EntB

RENAME AS EntB

RENAME AS ENT(Var2)

Descrição

Renomeia a entidade que está sendo processada. Depois que um comando RENAME é encontrado, a entidade procura adiante na lista de processos e depois volta para o começo da lista de processos até achar um processo definido para o novo tipo de entidade no local atual. Não será executada mais nenhuma lógica para esta entidade que estiver definida para o seu nome original. Use RENAME para começar a colher estatísticas para uma entidade com um novo nome. Geralmente, o modo mais fácil e eficiente para renomear uma entidade é simplesmente usar um novo nome como entidade de saída no roteamento.

REPORT

Sintaxe

REPORT {WITH RESET} {AS <*expressão*>}

REPORT

REPORT WITH RESET

IF producao = 50 THEN REPORT AS “Relat50”

Descrição

Calcula e envia as estatísticas atuais para um banco de dados de saída. É útil para que se tenha uma ‘fotografia instantânea’ do modelo enquanto ele está sendo executado.

O comando REPORT pode ser seguindo pela opção WITH RESET que ‘zera’ as estatísticas depois que o relatório é passado para o banco de dados. Quando você usa a opção WITH RESET, você geralmente quer fornecer algum loop ou criar algum evento que irá chamar o comando REPORT em um momento apropriado.

Usado com a opção AS, o comando REPORT cria um relatório com o nome especificado na expressão, que pode ser acessado através do Módulo de Resultados quando criado um relatório de estatísticas gerais (General Stats Report) com o nome especificado na expressão.

RES()

Sintaxe

RES(<*número de nome-índice do recurso*>)

USE 10 RES(Var1) FOR 1.5 min

FREE RES(Var1)

DISPLAY “Usando agora” $ RES(Var1)

Descrição

Converte o número de nome-índice ou inteiro para o nome do recurso. Use esta função quando um comando ou função precisa do nome do recurso cujo número de nome-índice está definido em um atributo, variável, ou alguma outra expressão. Quando usado em uma expressão como a da terceira sintaxe acima, o ProModel irá retornar o nome do recurso atual.

Use RES() para designar o operador apropriado de acordo com o atributo da peça, ou para mudar as tarefas dos recursos com o passar da simulação.

RESET

Sintaxe

RESET <*identificação do arquivo*>

RESET Tempos

RESET (Tempos)

Descrição

Inicia um arquivo geral de leitura novamente à partir do início. O comando RESET é utilizado principalmente nas lógicas de Inicialização ou de Terminação (Initialization or Termination logic) para começar de novo a ler ou a escrever um arquivo geral no começo ou no fim de múltiplas replicações ou execução simples e independente. O comando RESET também pode ser usado para reler dados cíclicos em uma mesma execução da simulação. Os parênteses opcionais foram incluídos apenas para assegurar a compatibilidade com modelos mais antigos.

RESET STATS

Sintaxe

RESET STATS

IF Total = 20 THEN RESET STATS

Descrição

Zera as estatísticas da simulação. É útil em conjunto com o comando REPORT para se controlar manualmente as estatísticas para confecção de relatórios em algum caso específico ou lógica de evento.

RESOURCE()

Sintaxe

RESOURCE()

Descrição

Retorna o número de nome-índice do recurso que está sendo processado por uma lógica de parada (break logic) ou lógica fora do turno (off-shift logic).

RESOURCEUNIT ()

Sintaxe

RESOURCEUNIT()

Descrição

Retorna o numero unitário do recurso que esta sendo utilizado. Pode ser utilizada somente nas lógicas de parada e fora de turno.

RESQTY()

Sintaxe

RESQTY({<*nome do recurso*>})

IF RESQTY(Res1) > 5 THEN FREE Res1

Descrição

Retorna o número de unidades do recurso especificado que a entidade atual possui. A função RESQTY() pode ser usada para determinar a quantidade de tempo necessária para processar uma entidade baseada no número de recursos que a entidade possui.

RETURN

Sintaxe

RETURN {<*expressão*>}

RETURN

RETURN Attr1**SQRT(Attr2)

Descrição

Envia um valor, à partir de uma subrotina, para a lógica que chamou a subrotina. Da mesma maneira que os parâmetros enviam informações da lógica que chamou a subrotina para a subrotina, o comando RETURN envia informações da subrotina para a lógica que a chamou. Depois que o comando RETURN é executado, nenhuma outra lógica na subrotina é executada. Quando a subrotina retorna valores, o comando RETURN deve ser seguido de uma expressão.

Quando é usado em uma lógica que não seja uma subrotina, o comando RETURN funciona como um comando BREAK ou BREAKBLK muito poderoso. Enquanto os comandos BREAK e BREAKBLK saem apenas do loop ou bloco de comandos em que se encontram o comando RETURN sai da lógica inteira, não importando o número de loops ou blocos de comando que estiverem depois dele.

ROUND()

Sintaxe

ROUND(<*expressão*>)

Inteiro1 = ROUND(3.5)

Descrição

Arredonda a expressão para o número inteiro mais próximo. Use ROUND() quando o ProModel estiver truncando números reais para inteiros, e você quer que eles sejam arredondados.

ROUTE

Sintaxe

ROUTE <*expressão*>

ROUTE 2

ROUTE Attr1

ROUTE Dist1()

Descrição

Executa um bloco de roteamento para a entidade que está sendo processada. Os processos não continuam até que todas as entidades roteadas para um bloco particular tenham começado a executar sua lógica de movimentação. A lógica do processo pode conter vários comandos ROUTE. Estes comandos podem ser selecionados individualmente usando os comandos IF-THEN, ou podem ser processados sequencialmente, combinados ou não com outros comandos entre eles. Se qualquer comando ROUTE aparecer em uma lógica de processo, o ProModel assume que todo o roteamento será feito pelo usuário e assim não realiza o roteamento automático. Se nenhum comando ROUTE aparecer na lógica do processo, todos os blocos de roteamento serão executados automaticamente assim que a lógica do processo estiver completa.

O comando ROUTE é geralmente utilizado com os comandos IF-THEN para tomarem decisões de roteamento baseado em alguma lógica complexa que não seja disponível em alguma outra função do ProModel (tais como funções do sistema ou 'user condition routing rule'). O comando ROUTE,

se usado com os comandos IF-THEN adequadamente, irá assegurar que apenas um dos blocos de roteamento será ativado.

Este comando pode ser usado para rotear uma ou mais entidades e deixar para trás uma entidade "fantasma" que irá processar a lógica restante depois do comando ROUTE. A entidade "fantasma" é também referenciada como a entidade mãe. A entidade gerada pela entidade mãe toma a rota especificada pelo comando ROUTE. Se a entidade gerada não puder ir para o próximo local e está bloqueada, a entidade mãe também estará bloqueada e não irá continuar na execução da lógica até que a entidade gerada por ela seja desbloqueada.

SEND

Sintaxe

SEND <*expressão*><*nome da entidade*> TO <*destino*>{,<*prioridade*>}

SEND 2 EntA TO Loc2

SEND 1 Grp_A TO Processamento_do_Grp_A, 10

Descrição

Envia um número especificado de um tipo de entidade particular para um destino. As entidades a serem enviadas devem estar esperando com uma regra de roteamento SEND. A entidade que gerou o comando SEND continua processando não importando se as entidades do tipo solicitado estão esperando para serem enviadas ou não. Se nenhuma entidade naquele momento estiver esperando para ser enviada, as entidades serão enviadas à medida em que forem ficando disponíveis.

O comando SEND pode modelar um sistema baseado na demanda, e não na chegada de entidades, (chamado de sistema de 'puxada'). Os pedidos do cliente fazem com que um comando SEND seja designado para a entidade principal. A montagem principal designa comandos SEND para as submontagens. O modelo de referência SEND é um excelente exemplo desta técnica.

O comando SEND também pode ser usado como um dispositivo de controle para limitar a quantidade de estoque em processo (WIP) em certas áreas críticas. As quantidades são enviadas para a área de produção apenas quando o nível de WIP cai a um certo ponto.

SETRATE

Sintaxe

SETRATE <nome *do recurso*>,<*expressão*>,<*# da unidade*>

SETRATE Operador,25,3

Descrição

Permite que você defina a taxa regular de custos para os recursos de um modelo. Se você já definiu uma taxa regular no módulo de custos (Cost module), este comando irá substituir esta taxa. Você pode usar o comando SETRATE para designar diferentes taxas de custo para cada unidade do recurso.

SKIP

Sintaxe

SKIP

Descrição

Em lógicas de pré off-shift (antes do término no turno) ou pre-break (antes da parada), um comando SKIP faz com que qualquer lógica off-shift ou lógica principal de parada, assim como tempo fora de turno ou paradas definidas no arquivo de turno sejam ignoradas e assim o local ou recurso afetado irá continuar em operação.

Em uma lógica off-shift (fora do turno) ou lógica de parada (break logic), um comando SKIP faz com que o término de turno ou tempos de parada definidos no arquivo de turno sejam ignorados. Isto é útil quando você quer definir seu próprio tempo de parada como uma parte da lógica e não usando o tempo definido no arquivo de turno.

SOUND

Sintaxe

SOUND <*expressão*>

SOUND “Chimes.wav”

SOUND “Tada.wav”

Descrição

Executa um arquivo de som. Os arquivos de som, que têm a extensão .WAV, podem ser comprados ou criados com uma placa de som. Alguns sons, como os dos exemplos acima, já vêm com o Windows e são encontrados no diretório Windows. Use o comando SOUND para alertar o usuário do modelo que algum evento está ocorrendo.

SPLIT

Sintaxe

SPLIT <*expressão*> {AS <*novo nome da entidade*>}

SPLIT 10

SPLIT 10 AS EntX

Descrição

Divide uma entidade em um número especificado de entidades, e, opcionalmente, muda seus nomes. Todas as entidades resultantes terão os mesmos valores de atributos da entidade original. Cada entidade resultante inicia novas estatísticas. Para que uma entidade seja dividida esta deve liberar, com o comando FREE, todos os recursos capturados por ela. Use o comando SPLIT para dividir um pedaço de matéria prima em componentes, tal como dividir uma caixa de velas em velas individuais. As entidades formadas pelo comando SPLIT AS em um local não aparecerão nas estatísticas deste local.

SQRT()

Sintaxe

SQRT(<*expressão*>)

Real1 = SQRT(Real2)

Descrição

Retorna a raiz quadrada de uma expressão.

STOP

Sintaxe

STOP {<*expressão*>}

STOP

STOP “Termino normal”

Descrição

Termina a replicação corrente e, opcionalmente, mostra uma mensagem. A simulação irá continuar com a próxima replicação. Use STOP para parar uma replicação quando a simulação foi executada o tempo suficiente para fornecer dados estatísticos suficientes.

THREADNUM()

Sintaxe

THREADNUM()

IF THREADNUM() = 215 THEN DEBUG

Descrição

Toda vez que qualquer lógica é executada, executa-se linha por linha, às quais são designados números específicos. A função THREADNUM retorna o número da linha que chamou a função. Esta função é mais útil em conjunto com os comandos IF-THEN e DEBUG para ativar o depurador num processo específico. Se o modelo não mudar entre as execuções da simulação, o número de cada linha também não mudará, mas este número irá mudar de replicação para replicação.

A maioria das mudanças num modelo farão com que as linhas (threads) tenham diferentes numerações em execuções subsequentes.

TIMELEFT()

Sintaxe

TIMELEFT()

Attr1 = TIMELEFT()

Descrição

Retorna o tempo restante caso uma preempção tenha ocorrido durante um comando WAIT. O valor retornado estará na unidade de tempo padrão e deve ser checado antes que qualquer processamento ocorra, pois o valor é atualizado toda vez que uma preempção ocorre. Se o valor tiver que ser guardado para um uso posterior, este deve ser designado a um atributo da entidade ou a uma variável local.

Se várias entidades forem preemptadas em um local, o valor retornado pela função para cada entidade será o da entidade que tiver o maior tempo restante no comando WAIT.

Quando nenhuma unidade é especificada nos parênteses desta função, ela retorna o valor na unidade de tempo padrão especificada na janela de Informação Geral (General Information).

TIMESUSED()

Sintaxe

TIMESUSED(<*recurso*>)

IF TIMESUSED(Res1) > 5 THEN USE Res2 FOR 10

Descrição

Retorna o número de vezes que o recurso foi utilizado.

TRACE

Sintaxe

TRACE {<*mensagem*>} {STEP ou CONT ou OFF ou CLOSE}

TRACE “Begin Test for Resouce A”

TRACE CONT

Descrição

Ativa e desativa o rastreamento. As listas de rastreamento aparecerão em uma janela separada na tela. Use o comando TRACE para acompanhar o fluxo lógico do modelo.

STEP – Faz com que o ProModel espere por um clique no botão esquerdo do mouse para executar o próximo comando, ou faz com que rasteie continuamente enquanto o botão direito do mouse estiver pressionado. O step é o default do comando TRACE.

CONT – Rasteia continuamente sem a intervenção do usuário.

OFF – Desativa o rastreamento mas não fecha a lista de rastreamento.

CLOSE – Desativa o rastreamento e fecha a lista de rastreamento.

TRUNC()

Sintaxe

TRUNC(<*expressão*>)

Inteiro1 = TRUNC(3.9)

Descrição

Retorna uma expressão real truncada em um inteiro. Qualquer dígito a direita da vírgula será removido. O ProModel converte valores reais em inteiros truncando-os automaticamente quando necessário. Esta função foi incluída para assegurar compatibilidade com as versões mais antigas do ProModel.

UNGROUP

Sintaxe

UNGROUP { LIFO }

UNGROUP

UNGROUP LIFO

Descrição

Separa as entidades que foram agrupadas com o comando GROUP. Cada uma das entidades resultantes procura por um processo adiante e depois volta ao início da lista de processos até que um processo seja encontrado para aquele tipo de entidade no local em que foram desagrupadas. A primeira entidade processada do grupo fica com os recursos possuídos pelo grupo. Se uma entidade agrupada tem membros que também são entidades agrupadas, apenas o nível mais alto de agrupamento será desagrupado com o comando UNGROUP. Um comando UNGROUP adicional irá desagrupar outros membros agrupados.

UNITS()

Sintaxe

UNITS (<*local*> ou <*recurso*>)

PAUSE “Existem ” $ UNITS(Res1) $ “Res1s no sistema.”

Descrição

Retorna o total de unidades de um local ou recurso.

UNLOAD

Sintaxe

UNLOAD <*expressão*> {IFF <*expressão booleana*>}

UNLOAD 5

UNLOAD 5 IFF Entity() = EntA

Descrição

Descarrega uma certa quantidade de entidades, ou uma certa quantidade destas entidades dependendo de uma condição. Use o comando UNLOAD para descarregar entidades de uma entidade carregadora que foi carregada anteriormente com o comando LOAD. O ProModel processa as entidades descarregadas, na frente da entidade que as carregou. Cada uma das entidades descarregadas procura por um processo adiante e depois volta ao início da lista de processos até que um processo seja encontrado para aquele tipo de entidade no local em que foram descarregadas.

USE

Sintaxe

USE {<*quantidade*>} <*recurso*> {,<*prioridade1*>{,<*prioridade2*>}}

FOR <*tempo*>

{AND ou OR {<*quantidade*>} <*recurso*> {,<*prioridade1*>{,<*prioridade2*>}}

FOR <*tempo*> ...}

USE 2 Res2, 5 FOR 4:23:O3'

USE 2 Res1 FOR 2.0 min OR 3 Res2 FOR 1.5 min

USE Crane2 , 15, 35 AND (Res2 FOR 5 OR Res3 FOR 5)

USE Attr_Oper RES(Tipo_Attr) FOR Var_Tempo1 Hr

Descrição

Captura um recurso ou combinação de recursos assim que cada recurso fica disponível. Quando o recurso é capturado, ele é usado pelo intervalo de tempo especificado, e depois é liberado quando a duração especificada terminar. Se a entidade já possui um dos recursos especificados devido a um comando GET, JOINTLY GET, ou MOVE WITH anterior, a entidade irá ainda assim tentar capturar uma unidade adicional daquele recurso.

VARIABLE()

Sintaxe

VARIABLE()(<expressão numérica>)

VARIABLE(Attr)=124

VARIABLE(x)=VARIABLE(y)

N=VARIABLE(x)+VARIABLE(y)-1

Descrição

Converte o número de nome-índice ou inteiro em um nome de variável. Use esta função quando uma expressão numérica usar uma variável cujo número do nome-índice estiver armazenado em um atributo, matriz, ou variável.

Observação: Você não pode usar VARIABLE() em um comando PROMPT, INC, DEC OU READ.

VIEW

Sintaxe

VIEW “nome da vista”

VIEW “Cell5”

VIEW “Vista10”

Descrição

Use este comando para mudar a lista na janela de Layout, à partir de uma lógica. Uma vez que uma vista é definida no menu de exibição (View) do menu principal, você poderá usá-la na lógica.

WAIT

Sintaxe

WAIT <*expressão de tempo*>

WAIT 3 min

WAIT 0

WAIT 2.5 + Tempo_de_Limpeza

WAIT N(8,.5) + 3 sec

Descrição

Simula o tempo necessário para o processamento de uma entidade. O comando WAIT faz com que qualquer processamento posterior da entidade aguarde até que o intervalo de tempo especificado tenha passado. O resto do modelo continua sendo processado enquanto a entidade espera. Se o valor de expressão for igual a zero, a entidade atual não acabará o processamento até que todos os outros processos agendados para aquele momento da simulação tenham sido executados.

WAIT UNTIL

Sintaxe

WAIT UNTIL <*expressão*>

WAIT UNTIL Var1 > 3

WAIT UNTIL Var1 < Attr3 AND Var2 >= 5

Descrição

Aguarda o processamento da lógica corrente até que a expressão seja verdadeira. O resto do modelo continua processando durante a espera. Observe que, se a expressão é avaliada inicialmente como falsa, ela só será reavaliada quando um atributo de local, variável, ou elemento de matriz da expressão mudar. Se várias entidades estiverem esperando pela mesma condição elas serão liberadas uma por uma. Isto permite que uma entidade liberada mude o valor de uma variável que irá impedir que as outras entidades que estão esperando sejam liberadas.

WARMUP

Sintaxe

WARMUP

WARMUP

IF producao = 50 THEN WARMUP

Descrição

Instrui a simulação para encerrar o período de warmup (aquecimento) zerando todas as estatísticas e apagando arquivos seriados no tempo (time series) relevantes. Somente um comando WARMUP poderá ser usado na simulação.

CUIDADO! Se vários comandos WARMUP forem encontrados ou se um comando WARMUP for usado adicionalmente ao tempo de warmup especificado na janela de opções de execução (Run Options), o ProModel irá gerar um aviso para informá-lo que isto ocorreu. Somente o primeiro warmup encontrado (do comando, ou da opção de execução) será executado.

WHILE...DO

Sintaxe

WHILE <*expressão booleana*> DO <*bloco de comandos*>

```
WHILE Matriz1[n] <> 10 DO INC n
WHILE FREECAP(Loc1) > 5 DO
        BEGIN
          INC Var2,5
          WAIT 5 sec
        END
```

Descrição

Repete um comando ou bloco de comandos continuamente enquanto uma condição permanecer verdadeira. WHILE...DO é um entry-condition loop (loop condicional da entrada), ou seja, o loop não será executado até que uma expressão seja verdadeira.

WRITE

Sintaxe

WRITE <*nome do arquivo*>,<*string ou expressão numérica*>

{<*número máximo de dígitos antes do decimal*>,<*dígitos após a decimal*>}

Descrição

Escreve informações para um arquivo geral de escrita. O próximo item a ser escrito no arquivo aparecerá imediatamente após este item. O comando WRITE irá sempre inserir no arquivo, a não ser que o arquivo seja resetado (RESET) em múltiplas replicações ou em simulações simples e independentes. Qualquer arquivo que foi escrito com o comando WRITE automaticamente se torna um arquivo texto e terá um marcador de fim atachado automaticamente no fim do arquivo quando ele for fechado. Para maior flexibilidade de escrita, use o comando XWRITE.

WRITELINE

Sintaxe

WRITELINE <*nome do arquivo*>,<*string ou expressão numérica*>

{<*número máximo de dígitos antes do decimal*>,<*dígitos após a decimal*>}

Descrição

Escreve informações para um arquivo geral de escrita e inicia uma nova linha. O comando WRITELINE irá sempre inserir dados no arquivo a não ser que o arquivo seja resetado (RESET) em múltiplas replicações ou simulações simples e independentes. Qualquer arquivo que foi escrito com o comando WRITELINE automaticamente se torna um arquivo texto e terá um marcador de fim atachado automaticamente no fim do arquivo quando ele for fechado.

XSUB()

Sintaxe

XSUB (<*nome do arquivo*>,<*número da função ordinária*> ou <*nome da função*>

{,<*parâmetro1*>,<*parâmetro*2>...})

XSUB(Interface,1,5)

XSUB(LogDLL,"_Log_B_of_X",10,5)

Descrição

Chama uma subrotina externa dentro de um arquivo DLL. XSUB() é talvez o comando mais poderoso do ProModel, porque ao utilizá-lo o usuário pode acessar toda a funcionalidade de qualquer linguagem de programação 32 bit para Windows, tais como as linguagens C, C++, ou Pascal. O comando XSUB() pode ser usado para entradas e saídas (IO) sofisticadas de arquivo, e para tornar as simulações interativas. De fato, as subrotinas chamadas com o comando XSUB() podem fazer qualquer coisa que a linguagem em que o arquivo foi escrito permitir. Por causa de seu poder, entretanto, o comando XSUB() deve ser usado com cuidado. Quando chamado, o XSUB suspende a simulação até que a execução da subrotina externa seja concluída.

Para isso, a subrotina deve ter sido compilada dentro do DLL como exportável pelo compilador 32 bit do Windows, e ter um tipo de retorno do tipo IEEE e no formato real duplo (double real).

O XSUB() irá copiar os parâmetros que seguem o nome da função para um bloco de memória, e daí passa a função a um ponteiro para este bloco de memória.

A função pode pegar apenas um parâmetro. Mas a função pode acessar qualquer quantidade de parâmetros através da estrutura sobreposta. A função deve definir uma estrutura equivalente ao tipo e ordem dos parâmetros, e alocar um ponteiro transmitido a um ponteiro para este tipo de estrutura. Assim os ponteiros podem ser usados através de uma estrutura sobreposta. O ProModel transmite strings como ponteiros de caracteres, inteiros como valores de 4 bytes, e reais como valores IEEE de 8 bytes.

XWRITE

Sintaxe

XWRITE <*nome do arquivo*>,<*string ou expressão numérica*>

Descrição

Escreve informações para um arquivo geral de escrita em qualquer formatação que o usuário escolher. O comando XWRITE é útil para escrever em arquivos com uma formatação definida pelo usuário, enquanto que os comandos WRITE e WRITELINE visam escrever em arquivos texto com uma formatação especial. O comando XWRITE sempre insere dados ao arquivo a não ser que este seja resetado (RESET). Note que sempre que os comandos WRITE ou WRITELINE escrevem em um arquivo, o arquivo será automaticamente um arquivo texto. Não haverá nenhum marcador de fim inserido nos arquivos escritos apenas com o XWRITE. Em replicações subsequentes, itens adicionais são inseridos no fim do arquivo a não ser que o arquivo seja resetado (RESET). Nenhuma formatação é dada à expressão, embora o comando FORMAT possa ser usado para formatar os dados manualmente. Para adicionar um marcador de final de arquivo a um arquivo com formatação definida pelo usuário, use XWRITE CHAR(26).