ESCOLA POLITÉCNICA DA UNIVERSIDADE DE S. PAULO

DEPTO DE ENGENHARIA MECÂNICA

PMC 581 – PROJETO MECÂNICO II

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE UM SISTEMA MRP

SÃO PAULO, FEVEREIRO DE 2002.

PROF. ORIENTADOR
JOÃO PAULO M. MARCICANO

ALUNOS	Nº USP
EDUARDO STINCHI PASCALE	308305
FABIO ANDRADE DE MACEDO	2370751

5.0

ÍNDICE

Índice de Figuras

Introdução

A logística ganhou status de prioridade a ponto de os profissionais da área atuarem no primeiro escalão das grandes empresas. As estratégias e mudanças nesse segmento já são discutidas diretamente com a presidência destas empresas.

A adoção de novas metodologias nas montadoras de automóveis nos últimos anos, reduziu de 30% para 20% a participação da logística no custo final do veículo, segundo dados publicados pela Gazeta Mercantil em maio de 2000.

As decisões levam sempre em conta a logística. Ou seja, o desenho de um novo parafuso de uma peça poderá ser modificado se for possível aproveitar um já existente. Isso permite manter um mesmo fornecedor e negociar uma escala maior de encomendas, portanto, a preços mais reduzidos.

Isso reflete de forma significativa o quão grande é a necessidade de planejar e pensar logística dentro de uma organização. O estudo individual de cada atividade, de cada prioridade, de cada projeto, deve ser feito tendo em vista sempre a logística.

Neste contexto o MRP (*material requirements planning*) e o MRP-II (*manufacturing resources planning*) surgem como uma grande questão a ser entendida, desenvolvida, adaptada e implementada numa empresa.

Neste trabalho vamos estudar os conceitos e desenvolver uma metodologia de implementação de um sistema MRP-II.

Num primeiro momento, vamos nos ater ao estudo dos conceitos envolvidos nos Sistemas de Administração da Produção, tais como gestão de materiais, gestão de estoques, capacidade produtiva, etc.

A seguir, serão abordados de maneira mais específica os conceitos de MRP (planejamento das necessidades de materiais) e MRP-II (planejamento dos recursos de manufatura), para posteriormente desenvolvermos uma metodologia de implementação de um sistema MRP-II.

Sistemas de Administração da Produção

De um modo geral, podemos considerar como sistemas de administração da produção (SAP) todos os sistemas de informação envolvidos nas decisões (estratégicas ou operacionais), que se referem às questões logísticas mais básicas como:

- O que produzir e comprar
- Quanto produzir e comprar
- Quando produzir e comprar
- Com que recursos produzir

Dentre as técnicas conhecidas atualmente, o MRP-II se destaca como uma das mais importantes, devido à sua utilização em larga escala pelas grandes empresas em todo o mundo.

FUNÇÕES DO SAP

Todos os SAP, independente da técnica utilizada, devem ser capazes de auxiliar a organização a realizar as seguintes atividades[1]:

1. Planejar as necessidades futuras de capacidade produtiva da organização;
2. Planejar os materiais comprados;
3. Planejar os níveis adequados de estoques de matérias-primas, semi-acabados e produtos finais, nos pontos certos;
4. Programar atividades de produção para garantir que os recursos produtivos envolvidos estejam sendo utilizados, em cada momento, nas coisas certas e prioritárias;
5. Ser capaz de saber e de informar corretamente a respeito da situação corrente dos recursos (pessoas, equipamentos, instalações, materiais) e das ordens (de compra e produção);
6. Ser capaz de prometer os menores prazos possíveis aos clientes e depois fazer cumpri-los;
7. Ser capaz de reagir eficazmente.

[1] Corrêa, Gianesi, Caon – Planejamento, Programação e Controle da Produção – Ed. Atlas, 1997

ASPECTOS DE DESEMPENHO

Podem ser definidos como os fatores que influenciam a decisão do cliente ao escolher um fornecedor. São eles:

Custo percebido

Não envolve apenas o *preço* de um produto, mas também custos adicionais tais como transporte, manutenção de estoques (que mudam conforme as diferentes freqüências de entrega dos fornecedores), tamanhos de lote maiores que o necessário, etc.

Velocidade de Entrega

É o tempo entre a solicitação de um determinado material pelo cliente e a sua efetiva disponibilização, por parte do fornecedor, para uso.

Confiabilidade de Entrega

Refere-se à capacidade de um fornecedor de cumprir com seus prazos de entrega e também com a quantidade prometida.

Flexibilidade das Saídas

Representa a capacidade de implementar mudanças no sistema produtivo, em termos de quantidade produzida, de velocidade de produção, ou até mesmo de material produzido, conforme as inovações e variações do mercado. Em outras palavras, é a capacidade do sistema produtivo de responder de maneira eficiente a mudanças inesperadas das necessidades de produção.

Qualidade dos produtos

Diz respeito à conformidade do produto com as suas especificações de projeto.

Serviços prestados ao cliente

São atividades oferecidas pelo fornecedor visando a agregar valor ao produto, tais como informações técnicas, garantia de qualidade, assistência técnica pré e pós-venda, maior freqüência de entrega, etc.

RELAÇÃO ENTRE FUNÇÕES DO SAP E ASPECTOS DE DESEMPENHO

O quadro a seguir mostra de maneira simples e objetiva a influência de cada uma das funções do SAP (representadas pelos números da primeira coluna) nos diferentes aspectos competitivos de desempenho descritos.

	Custo percebido	Velocidade	Confiabilidade	Flexibilidade	Qualidade	Serviços
1	✓	✓	✓			
2	✓					
3	✓	✓	✓	✓		
4	✓	✓	✓			
5			✓		✓	✓
6	✓		✓			
7		✓		✓		

MRP – Planejamento das Necessidades de Materiais

O MRP é um sistema de inventário que consiste em tentar minimizar o investimento em inventário. Seu objetivo é atender o cliente da melhor forma, com o menor investimento em estoque.

Em suma, o conceito de MRP é obter o material certo, no ponto certo, no momento certo, através de um planejamento das prioridades e a Programação Mestra de Produção.

Este sistema envolve atividades como planejamento empresarial, previsão de vendas, planejamento dos recursos produtivos, planejamento da produção, planejamento das necessidades de produção, controle e acompanhamento da fabricação, compras e contabilização dos custos, além da criação e manutenção da infra-estrutura de informação industrial.

A criação e manutenção da infra-estrutura de informação industrial passam pelo cadastro de materiais, estrutura de informação industrial, estrutura do produto (lista de materiais), saldo de estoques, ordens em aberto, rotinas de processo, capacidade do centro de trabalho, entre outras.

A grande vantagem da implantação de um sistema de planejamento das necessidades de materiais é a de permitir ver, "rapidamente", o impacto de qualquer replanejamento. Assim, podem-se tomar medidas corretivas, sobre o estoque planejado em excesso, para cancelar ou reprogramar pedidos e manter os estoques em níveis razoáveis.

Além do controle do estoque, é importante a atualidade da lista de material. Através da contagem cíclica ou inventário rotativo garantimos uma maior proximidade à realidade do estoque.

CÁLCULO DA NECESSIDADE DE MATERIAIS

Para evitar falta ou excesso dos materiais envolvidos, o cálculo da necessidade de materiais pode ser demonstrado da seguinte forma:

Previsão de vendas – Estoque de produto acabado = Previsão líquida de vendas

↓

Com isso, podemos dar origem ao programa-mestre de produção.

↓

Programa mestre de produção **x** Lista de materiais = Demanda de materiais

↓

Demanda de materiais + Estoque físico – Saldo de pedidos = Necessidades de materiais

NATUREZA DA DEMANDA PARA O MRP

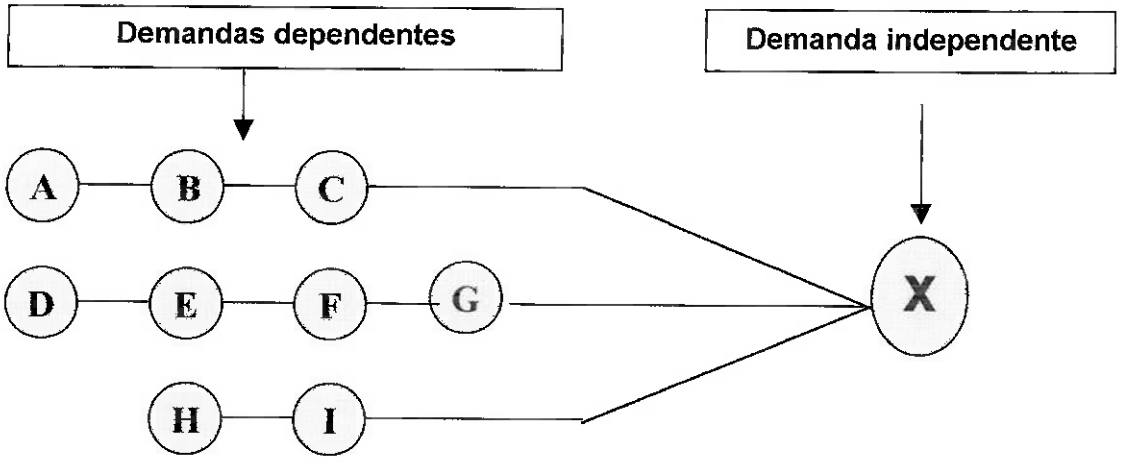
Podemos identificar claramente duas classificações para a demanda de um produto:

Independente

Quando não está relacionada com nenhum outro item. Neste caso deve ser prevista e projetada através de técnicas específicas de previsões.

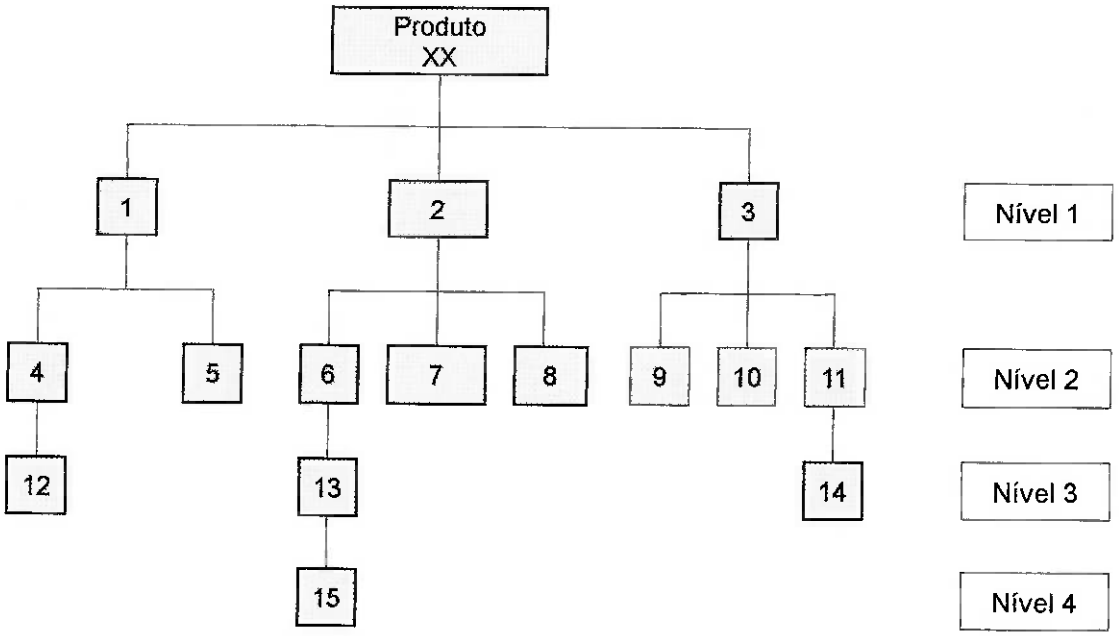
Dependente

Quando está relacionada ou depende de outro item. Esta demanda deve ser calculada.



ESTRUTURA DO PRODUTO OU LISTA DE MATERIAL

Na figura abaixo vemos a estrutura do produto explodida. Esta estrutura do produto é baseada na emissão de ordens em uma demanda calculada a partir do programa de montagens. Para que isto aconteça é necessário uma "Lista de Material", ou "Lista de Peças Estruturada".



Para o efetivo cálculo das necessidades de materiais deve-se considerar a estrutura do produto com os níveis de fabricação, a quantidade do lote de compra, o tempo de reposição para cada componente (comprado ou fabricado internamente), as necessidades das peças baseados no programa-mestre, o uso de cada peça, atentando-se para a sua utilização também em outros produtos, e o uso de cada peça, levando-se em conta que ela pode ser usada no mesmo produto em diversos níveis.

Para determinação da quantidade a comprar podemos escolher diversos métodos de acordo com as necessidades reais, tais como: Quantidade fixa, lote econômico, lote a lote, ou reposição periódica.

MRP II – Planejamento dos Recursos de Manufatura

O MRP II pode ser interpretado como uma extensão do MRP. Enquanto o MRP auxilia nas decisões de quando, quanto e o que produzir e comprar, o MRP II abrange ainda informações a respeito dos recursos produtivos, auxiliando assim nas decisões referentes a como produzir.

De maneira simplificada, poderíamos imaginar que o MRP II é apenas o MRP com cálculo de capacidade, porem o uso do MRP II implica numa lógica estruturada de planejamento que prevê uma seqüência hierárquica de cálculos, verificações e decisões, com objetivo de chegar a um plano mestre de produção que seja viável em termos de disponibilidade de material e de capacidade produtiva.

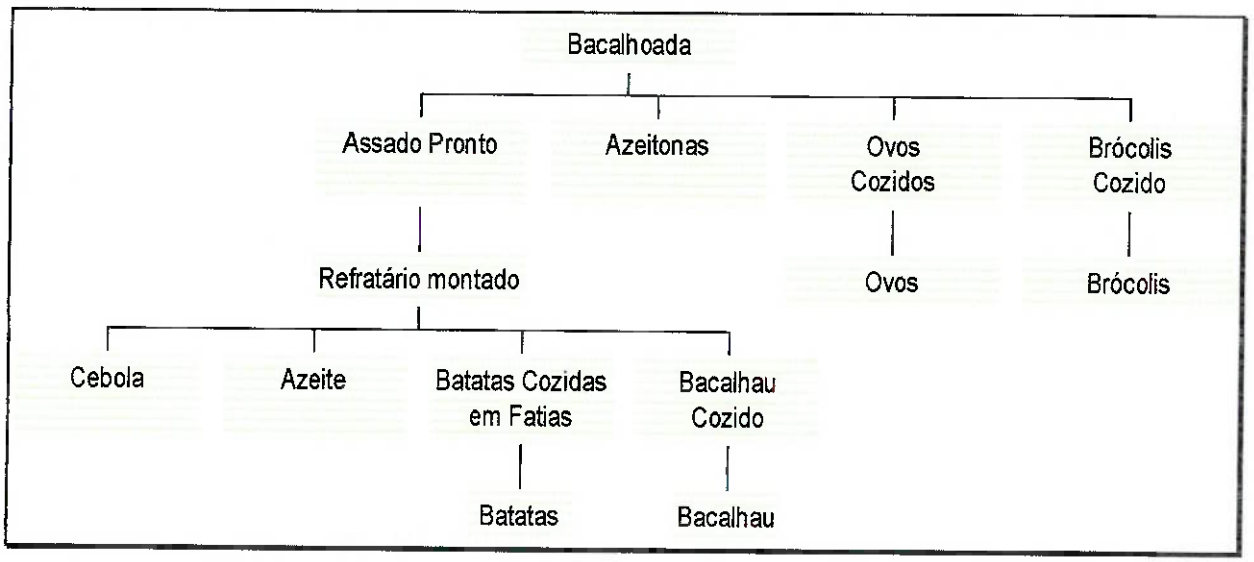
O MRP II é composto por uma série de funções que normalmente estão associadas a módulos de *softwares*.

Os principais módulos de um sistema MRP II são descritos no próximo capítulo, mas antes precisamos esclarecer alguns termos e conceitos que surgirão no decorrer do trabalho, de forma a garantir a total compreensão dos módulos e seus processos:

ESTRUTURA DO PRODUTO

Conforme já descrevemos anteriormente, a estrutura do produto representa uma visão explodida da estrutura física do produto a ser manufaturado.
Uma maneira de representar uma estrutura de produto é através de uma "Árvore de Materiais", em que são descritos os componentes e materiais em termos de relação "pai-filho".
Os itens que são manufaturados sempre apresentarão filhos em um nível inferior da árvore, o que não ocorre com os componentes comprados. Esta divisão permite que materiais idênticos sejam agrupados e calculados uma só vez.

Segue um modelo real de Estrutura de Produtos para uma Bacalhoada:



As Estruturas de Produtos têm como um de seus principais objetivos permitir que o MPS (Master Planning Schedule) seja baseado no menor número possível de itens.

Além disso, deve permitir o planejamento das prioridades das sub-montagens e outras funções conforme segue:

- emissão de ordem de cada item;
- programação de montagem dos itens.
- planejamento de prioridades;
- permitir o custeio do produto;

LEAD TIME

Podemos definir *Lead Time* como sendo o tempo que decorre entre a liberação de um pedido (produção ou compra) e o instante em que o respectivo material está pronto e disponível para uso.

O lead time (LT) deve determinar as necessidades de materiais, e para sua definição devem ser considerados tempos envolvidos nos seguintes aspectos/atividades:

LT de produção

- emissão física da ordem;
- tramitação da ordem até responsável no chão de fábrica;

- transporte de materiais enquanto a ordem está aberta;
- fila aguardando processamento nos setores produtivos;
- preparação dos equipamentos (*setup*) ou setores para processamento;
- processamento propriamente dito;
- possíveis inspeções de qualidade.

LT de compras

- emissão física da ordem;
- transformação da ordem de compra em pedido;
- envio até o fornecedor;
- entrega do fornecedor / transporte de materiais;
- recebimento e liberação;
- possíveis inspeções de recebimento e armazenagem.

É importante notar que erros na definição dos *lead times* podem ocasionar dois tipos de problema: formação de estoque indesejado (superdimensionamento do LT) ou então falta de produtos ou atraso na entrega (LT sub-dimensionado).

HORIZONTE DE PLANEJAMENTO

O horizonte de planejamento engloba todo o período de tempo sobre o qual se deseja elaborar os planos, visando a calcular as necessidades dos materiais.

Este horizonte engloba todo o tempo necessário para efetivar as decisões tomadas e deve ser tão longo quanto a influência dessas decisões sobre o negócio atual, isto é, não se deve perder tempo fazendo planos para um período que esteja tão distante do momento atual que praticamente não exerce nenhuma influência sobre as decisões atuais.

Deve ser considerado ainda dentro deste horizonte um eventual período de replanejamento, em que serão feitas análises atualizadas e possíveis correções de rota em relação ao que havia sido inicialmente planejado.

CADASTROS BÁSICOS

Antes de definir os módulos propriamente ditos, é necessário fazer uma série de cadastros básico iniciais, que servirão como fonte de informação para as funções de MRP-II propriamente ditas.

Seguem alguns exemplos de informações básicas a serem armazenadas:

Cadastro mestre de item – código dos produtos, descrição, *lead time*, estoque de segurança;
Estrutura de Produto – contém as informações referentes à estrutura do produto, conforme descrição acima;
Locais – informações a respeito do local exato de armazenagem dos itens, desde unidades fabris até as prateleiras;
Centros Produtivos – inclui horários de trabalho, índice de aproveitamento de horas disponíveis;
Calendários – converte o calendário de fábrica para o calendário de datas do ano, incluindo informações de feriados e férias;
Roteiros – seqüência de operações para fabricação de cada item incluindo *lead times*, ferramental necessário, etc...

Módulos de MRP-II a serem desenvolvidos

Nesta etapa apresentamos uma descrição teórica detalhada dos módulos a serem desenvolvidos e posteriormente inseridos no *software* que utilizaremos no exemplo de implantação do sistema MRP-II.

Para ajudar a construir uma visão estruturada dos diferentes módulos, vamos apresentá-los já divididos em três grandes grupos, grosseiramente descritos na figura abaixo, conforme o grau de planejamento exigido para sua perfeita execução.

COMANDO

Aqui estão os módulos que requerem o mais alto nível de planejamento. Normalmente é executado pela alta direção da empresa. É aqui que são tomadas as decisões mais estratégicas, que direcionam a empresa no mercado em que atua.

É importante ressaltar que os conceitos aqui apresentados podem ser amplamente empregados em diversos e diferentes ambientes de fabricação e montagem, o que demonstra a grande flexibilidade de aplicação dos sistemas MRP e MRP-II. Talvez o único ambiente em que o modelo não possa ser aplicado seja o de engenharia sob encomenda por exigir técnicas específicas na gestão dos projetos e da fabricação.

PLANEJAMENTO DE VENDAS E OPERAÇÕES (S&OP)

O processo de S&OP caracteriza-se por ser um planejamento contínuo, que sofre revisões mensais e outros ajustes em função da complexa situação do mercado (com suas alterações e picos de demanda), além das disponibilidades de recursos internos e externos dos quais depende a execução os planos da empresa.

Os principais objetivos do S&OP são:

- dar suporte ao planejamento estratégico da empresa
- garantir que os planos sejam realizáveis
- gerenciar de maneira eficaz eventuais alterações de planejamento e objetivos
- gerenciar estoques e pedidos para garantir bom nível de serviço a clientes

avaliar o desempenho real versus o planejamento anterior

Portanto, podemos dizer que o processo de S&OP está inserido entre o planejamento estratégico da empresa e os seus planos de vendas e plano mestre de produção (MPS), sendo que nele são analisados ainda planos de desenvolvimento de novos produtos, planos financeiros, etc.

O S&OP deve integrar diversas áreas de uma empresa (Vendas, Marketing, Logística, Finanças...), garantindo assim a coerência nas decisões de cada setor e visando a evitar que o desencontro de informações resulte em decisões conflitantes entre dois ou mais departamentos, o que pode até mesmo inviabilizar a execução de todo um projeto.
Exemplo: *os departamentos de Marketing e Desenvolvimento de Produto decidem-se pelo lançamento de uma grande variedade de produtos, o que exige máquinas mais flexíveis, ao mesmo tempo em que a área de manufatura resolve investir em máquinas mais produtivas, que requerem maior tempo de* setup *ou lotes mínimos de produção maiores.*

O horizonte de planejamento do S&OP pode variar dependendo do mercado em que se atua, mas normalmente é de um ou dois anos. Deve-se destacar, porém, que as revisões mensais do processo fazem com que se tenha um planejamento contínuo, evitando assim os estragos causados por um plano anual estático, que pode se tornar desatualizado rapidamente se não for adaptado às variações do mercado.

Dentro deste horizonte de planejamento é definido um sub-período, denominado Período de Congelamento, dentro do qual não se devem mudar as estratégias e objetivos previamente definidos, normalmente porque o custo de não mudar é menor do que os custos advindos de uma eventual mudança.

Alguns dos resultados práticos que devem ser atingidos a cada ciclo de S&OP são:

- estabelecimento de metas mensais de faturamento;
- projeção de lucros;
- projeção de estoques;
- quantidades mensais de produção determinadas dentro do Período de Congelamento;

MASTER PRODUCTION SCHEDULE (MPS)

O planejamento-mestre de produção, ou MPS, é um plano operacional, subordinado ao plano de vendas e operações resultante do processo de S&OP e da previsão de demanda feita na Gestão de Demanda.

O resultado deste novo processo é o plano-mestre de produção que serve como dado de entrada para o MRP, de onde sairá o plano detalhado de materiais e capacidade.

Diferentemente do MRP, que é basicamente um módulo de cálculos, o MPS tem como principal característica o fato de ser um módulo de tomada de decisão, baseada na seguinte equação:

Estoque Final = Estoque Inicial + Produção – Previsão de Vendas – Carteira

O quadro acima mostra que o estoque final será determinado a partir da decisão relativa à produção tomada no MPS. Para facilitar essa decisão, é imprescindível que a empresa tenha uma política de estoques clara, bem definida.

Alguns dos principais resultados advindos do MPS são:

- minimização da produção em horas-extras,
- minimização dos gastos em compras,
- redução dos investimentos em inventário,

O quadro abaixo[2] mostra as principais diferenças entre o S&OP e o MPS

Características	S&OP	MPS
Horizonte de planejamento	12 a 24 meses	2 a 5 meses
Período de re-planejamento	1 a 2 meses	1 semana
Item planejado	Famílias de Produtos	Produtos Finais
Participantes	Alta Administração e Diretorias (diversas áreas)	Gerências de Manufatura, Marketing e Vendas

É importante ressaltar que o MPS deve ser uma desagregação do S&OP, já que este é elaborado tendo em vista objetivos de prazo mais longo que o MPS. Assim, é claro que as decisões do MPS não podem estar em desacordo com as decisões tomadas pelo S&OP.

[2] Corrêa, Gianesi, Caon – Planejamento, Programação e Controle da Produção – Ed. Atlas, 1997

O MPS é uma declaração do que irá ser produzido e, além de estar coerente com o S&OP, deve ser realista, e para isso faz-se paralelamente ao processo do plano-mestre um cálculo grosseiro de capacidade produtiva, o RCCP (*Rough-Cut Capacity Planning*) com o objetivo de evitar futuros re-trabalhos quando for processado o planejamento detalhado de capacidade (CRP).

Em outras palavras, podemos dizer que o RCCP procura garantir que o MPS seja aproximadamente viável, no que diz respeito à capacidade produtiva.
O RCCP será explicado com mais detalhes no módulo de Planejamento de Capacidade.

GESTÃO DE DEMANDA

Este é um módulo de extrema importância num sistema MRP-II, pois é ele o responsável pela introdução de informações sobre as atividades do mercado no processo de planejamento. Apesar de não ser intuitivamente fácil de aceitar, é possível e necessário gerenciar e influenciar a demanda de determinados produtos, de modo a ocupar a capacidade de produção de maneira mais eficaz.
Corrêa e Gianesi definiram as cinco principais áreas que compõem a Gestão de Demanda, a saber:
Previsão de demanda – através da utilização de modelos matemáticos aplicados sobre bases de dados contendo histórico de vendas, obviamente levando em consideração os fatores externos e internos que têm influência direta na demanda, como promoções de vendas e marketing ou ainda uma eventual sazonalidade devido ao clima ou às condições econômicas
Comunicação com o mercado – se houver um canal de comunicação com o mercado que seja eficiente, a empresa pode obter informações preciosas como atividades da concorrência, antecipar o lançamento de um novo produto, etc. e utilizar estas informações na atividade de planejamento e gestão da demanda
Influência sobre a demanda – este recurso pode e deve ser utilizado pela empresa, tanto sobre a demanda de um período futuro (realização de promoções para aumentar as vendas) quanto sobre vendas já concretizadas (parcelamento de entrega, por exemplo). Conforme já dissemos anteriormente, esse recurso é importantíssimo, à medida que permite a melhor ocupação de toda a capacidade de produção da empresa.
Comprometimento com prazos – através de uma boa gestão sobre a demanda, cria-se a possibilidade de prometer e cumprir prazos de entrega, o que proporciona um bom desempenho no quesito confiabilidade de entrega, cuja importância já foi demonstrada no início deste trabalho.

Priorização e Alocação – caso a empresa não tenha capacidade de atender à demanda dos seus clientes (seja por problemas de produção ou de entrega), faz-se necessário estabelecer critérios e mecanismos que determinem quais são os clientes que ficarão sem receber tudo o que pediram para que outros possam ser atendidos. Uma opção bastante utilizada consiste em atender a todos os clientes parcialmente através de uma divisão justa da mercadoria produzida, em função do histórico de compras de cada um.

Operacionalmente falando, temos portanto os seguintes processos como etapas da Gestão de Demanda:

- Previsão de vendas
- Cadastramento de pedidos
- Comprometimento de datas de entrega
- Avaliação do nível de serviço prestado ao cliente
- Planejamento de necessidades divididas por fábricas e por centros de distribuição
- Distribuição física dos produtos aos clientes e Centros de Distribuição

Uma boa Gestão de Demanda deve ser capaz de produzir como principal contribuição para a empresa um plano de vendas detalhado, que esteja de acordo com a capacidade produtiva e ainda sirva para direcionar as atividades realizadas pelos departamentos da área comercial (Marketing/Vendas).

MOTOR

Neste bloco são incluídos os módulos que transformam as decisões tomadas pelo bloco de comando em ações práticas a serem executadas pelas "rodas" da empresa.

PLANEJAMENTO DE NECESSIDADE DE MATERIAIS (MRP)

Conforme já vimos anteriormente, o MRP é um módulo que, a partir das decisões tomadas no MPS (em termos de produção de produtos acabados) faz o cálculo das necessidades de materiais ou, mais especificamente, as quantidades e momentos de liberação e término de cada ordem de produção.

Caso existam alguns materiais com problema de disponibilidade, o próprio MRP já emite relatórios com mensagens de ação para que o programador faça os devidos ajustes na programação como, por exemplo:

- *Apressamento de recebimentos programados de compras*
- *Contratação de turnos extras de trabalho*
- *Sub-contratação de serviços*

Para isto são utilizadas, além das informações advindas do MPS, as informações contidas nos cadastros básicos (estruturas de produto, parâmetros dos itens, etc.)

O MRP gera, portanto, o plano detalhado de produção e compras, cuja viabilidade é assegurada pelo processo de capacidade de produção (CRP) que está explicado a seguir.

Estes dois processos formam o motor do MRP-II, pois são eles que fazem os cálculos que geram os planos detalhados a serem executados, a partir da direção principal que lhes é passada pelos processos de S&OP e MPS.

PLANEJAMENTO DE CAPACIDADE

O planejamento de capacidade é um processo desenvolvido em paralelo com o planejamento de materiais, e tem o objetivo de garantir que os planos de produção gerados pelos processos de S&OP, MPS e MRP sejam viáveis em termos da capacidade produtiva da empresa.

Através do planejamento de capacidade pode-se evitar a ocorrência de problemas como custos adicionais advindos da ociosidade de máquinas (no caso de excesso de capacidade produtiva) ou então de queda na qualidade de serviços ao cliente, pressão sobre os funcionários de fábrica (caso a empresa não tenha capacidade de produzir tudo o que foi planejado).

O planejamento de capacidade é realizado em todos os níveis de planejamento de materiais, e o seu grau de detalhe aumenta conforme avançamos no processo.

Destacamos assim o RCCP (*Rough-Cut Capacity Planning*) e o CRP (*Capacity Requirements Planning*) realizados respectivamente em paralelo ao MPS e ao MRP.

O RCCP é um planejamento grosseiro da capacidade, com o principal objetivo de dar suporte ao plano-mestre de produção, subsidiando as decisões de quanto produzir de cada produto, especialmente nos casos em que a capacidade produtiva da empresa não permite que o volume de venda previsto no S&OP seja atingido.

Por ser realizado de maneira mais superficial, o RCCP não leva em consideração todos os recursos envolvidos na produção. A análise é feita em cima de recursos críticos da empresa, que são identificados em função das seguintes características:

- É um gargalo?;
- Executa um processo de difícil sub-contratação?;
- É sensível ao *mix* de produtos?;
- É uma ferramenta especial de produção?;
- Requer funcionamento contínuo?;
- Requer tempo de *set-up* muito longo?

Além disso, o RCCP antecipa necessidades de capacidade de recursos cuja mobilização demore alguns meses.

Por fim, uma das mais importantes contribuições do RCCP é gerar um plano de produção de produtos finais que seja aproximadamente viável para evitar a perda de tempo com o processamento do MRP e CRP, para que, então, sejam descobertos graves problemas de excesso de capacidade que obriguem o re-processamento do MPS.

Analogamente ao RCCP, o CRP tem praticamente os mesmos objetivos, porém num nível de detalhe maior.

Assim, o CRP deve gerar um plano detalhado de produção e compras que seja realmente viável, por meio de ajustes efetuados no plano original sugerido pelo MRP, para que este possa ser enfim liberado para a execução pela fábrica.

O CRP utiliza informações de centros produtivos, roteiros e tempos, calculando as necessidades de capacidade para cada centro, período a período gerando um gráfico de carga, onde podem ser identificados excessos de necessidade de capacidade (estouro) ou ociosidade.

De posse desse gráfico, o programador pode então tomar as providências necessárias como adiamento de ordens ou provisão da capacidade necessária, de forma a garantir que a empresa consiga cumprir o plano elaborado pelo MRP.

RODAS

Aqui temos os módulos relacionados a duas importantes atividades dentro do sistema: **execução** e **controle**, sendo que esta última é responsável pela realimentação de todo o processo.

SHOP FLOOR CONTROL (SFC)

O controle de chão de fábrica faz a interface entre o planejamento e a fábrica. Em outras palavras, realiza a sequenciação das ordens recebidas.

Em termos práticos, o SFC libera a ordem de produção e então o sistema faz a alocação dos materiais utilizados, dando início a uma seqüência de operações conforme segue:

- Desconta os materiais utilizados do estoque disponível
- Informa tempos efetivamente gastos nas operações
- Materiais efetivamente utilizados
- Momentos de término de cada operação
- Controle de utilização de recursos (comparação *real x padrão*)
- Acompanha a evolução da ordem de produção no decorrer do *lead-time*

Por ser um módulo operacional e não de planejamento, o SFC é considerado – ao lado do módulo de compras – a Roda do MRP-II. O resultado obtido após o processamento do módulo SFC é um programa detalhado de produção, que é transmitido diretamente ao chão de fábrica.

O SFC é ainda responsável por realimentar o sistema de duas maneiras
Status – indica onde estão as ordens, verifica as quantidades produzidas, etc.
Alerta – sinalizam eventuais problemas do plano de materiais, normalmente quando não é possível executar detalhadamente o plano agregado recebido do MRP.

COMPRAS

Analogamente ao SFC, este módulo faz a interface entre o planejamento e os fornecedores (tanto de componentes como de matérias-primas).

As atividades são:

- Negociação de programações de entrega com os fornecedores
- Abertura das ordens de compra
- Emissão e acompanhamento dos pedidos
- Fechamento das ordens de compra
- Atualização dos registros de estoque.

Desenvolvimento do Software

Levando em consideração problemas com o licenciamento de softwares mais robustos, optamos por desenvolver o nosso software tomando como base o Microsoft Access, que é de amplo acesso no mercado.
Esta decisão acaba por facilitar a própria aplicação do software em empresas de médio porte, que já dispõem dos requisitos legais para a utilização do Microsoft Office.
A programação dos módulos e macros componentes será toda feita em Visual Basic.

FUNCIONALIDADES

O nosso software se propõe a atuar como um completo sistema de apoio aos conceitos de MRP-II apresentados. Suas principais aplicações serão:

- *Planejamento de Materiais (Compra e Produção);*
- *Planejamento de Capacidade Produtiva;*
- *Controle total do sistema e conseqüente realimentação dos processos de planejamento.*

Além disso, o software será desenvolvido de modo genérico, permitindo assim a sua aplicação aos mais diversos ramos da indústria, sem que para isso sejam necessárias grandes alterações na sua composição original.

ESTRUTURA

Segue uma breve relação das tabelas desenvolvidas para o programa, bem como os dados nelas contidos:

TABELAS BÁSICAS

Minha Empresa
Descrição: Tabela em que podem ser encontrados todos os dados da própria empresa, que devem constar em todas as Notas Fiscais (de Entrada e Saída).
Nome Fantasia
Razão Social
CNPJ
Inscrição Estadual

Endereço Completo
Telefone (PABX)
Telefone FAX

Funcionários

Descrição: Esta tabela deverá conter todos os dados pessoais referentes aos funcionários da empresa, a saber:

Código de Funcionário
Nome / Sobrenome
Identificação (RG)
CPF
N° Cart. Trabalho
Data Nascimento
Endereço Residencial Completo
Telefone Residencial
Telefone Celular
Contato Familiar
Grau de Parentesco
Foto
e-mail pessoal
e-mail empresa
Data Admissão
Departamento
Ramal
Chefe

Fornecedores

Descrição: Esta tabela contém todos os dados jurídicos das empresas fornecedoras, bem como os principais dados referentes às pessoas de contato.

Código de Fornecedor
Nome Fantasia
Razão Social
CNPJ
Inscrição Estadual
Ramo de Atividade
Data de Cadastro

Endereço Completo
Telefone (PABX)
Telefone FAX
Data Último Serviço
Número Serviços Prestados
Contato (Nome/Sobrenome)
Departamento
Cargo
Telefone / Ramal
e-mail

Clientes

Descrição: Assim como na tabela de fornecedores, esta tabela conterá todos os dados referentes aos clientes atendidos pela empresa, com algumas pequenas diferenças.

Código de Cliente
Nome Fantasia
Razão Social
CNPJ
Inscrição Estadual
Ramo de Atividade
Data de Cadastro
Endereço Completo (Faturamento)
Endereço Completo (Entrega)
Telefone (PABX)
Telefone FAX
Código de Desconto Geral Cadastrado
Data Último Pedido
Número Pedidos Realizados
Contato (Nome/Sobrenome)
Departamento
Cargo
Telefone / Ramal
e-mail

Transportadoras

Descrição: Contém os dados cadastrais das transportadoras contratadas e suas respectivas áreas de atuação.

Código de Transportadora
Nome Fantasia
Razão Social
CNPJ
Inscrição Estadual
Data de Cadastro
Endereço Completo
Telefone (PABX)
Telefone FAX
Data Último Serviço Prestado
Região de Atuação
Contato (Nome/Sobrenome)
Departamento
Cargo
Telefone / Ramal
e-mail

Produtos

Descrição: Tabela que traz todas as informações referentes aos produtos manufaturados, comprados e comercializados pela empresa.

Código Interno
Código de Barras (EAN-13)
Nome do Produto
Descrição do Produto
Preço Unitário (sem Descontos e sem Impostos)
Lote Mínimo (Compra ou Produção)
Estoque de Segurança
Dimensões (3D)
Peso
Lead Time (Compra ou Produção)
Figura

Equipamentos

Descrição: Aqui se encontram as informações referentes a todos os equipamentos de Produção da empresa, conforme segue.

Número Patrimônio

Nome de Referência

Operação Realizada

Operador Responsável (Funcionário)

Produto

Tempo de Setup

Tempo Médio Entre Falhas (MTBF)

Tempo Médio Para Reparar (MTTR)

Eficiência de Operação (%)

Status

Processos

Descrição: Contém os dados referentes às diversas operações realizadas pela empresa, não apenas em termos de fabricação, mas também em termos de compras

Código da Operação

Nome da Operação

Produto

Equipamento Relacionado

Lead Time

Status

TABELAS FUNDAMENTAIS

Pedidos de Compra

Descrição: Apresenta todas as informações (históricas e atuais) referentes aos pedidos de compra efetuados pela empresa.

Número do Pedido

Código do Fornecedor

Funcionário Responsável

Data Pedido

Prazo Entrega

Custo Estimado

Status do Pedido

Data Recebimento

Custo Real

Pedidos de Venda

Descrição: Todo e qualquer material negociado e vendido pela empresa deverá gerar um pedido de venda com as seguintes informações.

Número do Pedido (Interno)
Código do Cliente
Funcionário Responsável
Número de Pedido (do Cliente)
Data Pedido
Prazo Entrega
Produtos
Quantidade
Valor do Pedido
Prazo de Pagamento
Status do Pedido
Data Embarque
Nota Fiscal Correspondente

Ordens de Produção

Descrição: Aqui estão armazenadas as informações relativas a todas as ordens de produção expedidas.

Número da Ordem de Produção
Produto
Quantidade
Equipamentos
Processos
Funcionário Solicitante
Data da Ordem
Prazo de Entrega
Status
Código de Falha

Estoques

Descrição: Nesta tabela se encontram as quantidades em estoque de todos os produtos da empresa, assim como os seus respectivos lotes de segurança.

Código do Produto
Descrição do Produto
Estoque de Segurança
Quantidade em Estoque
Unidade de Medida

Notas Fiscais

Descrição: Todas as Notas Fiscais de Venda da empresa (não inclui as devoluções) ficam armazenadas nesta tabela

Número da NF
Pedido de Venda Referente
Valor Frete
Valor Impostos
Descontos Aplicados
Valor Final da Nota
Data Emissão
Data Recebimento Mercadoria
Status
Ocorrência

Falhas/Ocorrências

Descrição: Apresenta um histórico de todas as falhas e ocorrências registradas nas operações da empresa, envolvendo todas as etapas do processo (compras, produção, vendas, transporte e pós-venda).

Número da Ocorrência
Tipo de Falha
Descrição da Falha
Departamento Responsável
Data Registro
Providência Tomada
Status

Planejamentos

Descrição: Tabela que registra os últimos planejamentos de processos produzidos pelo sistema.

Número do Planejamento

Data Planejamento

Horizonte Planejado

Produto

Lead Times

TABELAS AUXILIARES

Estruturas de Produto

Descrição: Tabela que relaciona todos os produtos listados e seus respectivos pais, permitindo que sejam construídas e visualizadas as árvores de produto, bem como se calculem as necessidades de material (compra ou produção) equivalentes.

Código do Relacionamento

Código Produto

Código Pai

Quantidade no Pai

Nível

Unidades de Medida

Código

Tipo de Medida (metros, kg, ...)

Status de Ordens de Produção

Código

Situação (Aberta, Recebida, Em Processamento, Fechada)

Status de Pedidos

Código

Situação (Aberto, Enviado, Entregue, Recebido, Fechado)

Status de Processos

Código

Situação (Em Espera, Em Andamento, Finalizado, Falha)

Descontos

Código

Tipo do Desconto (Quantidade, Fidelidade, Especial)

Valor do Desconto

Impostos

Produtos

Tipo Imposto (IPI, ICMS, PIS/COFINS,...)

Valor Imposto (%)

Custos de Frete

Código

Região de Entrega

Transportadora

Custo Fixo

Custo Variável (km rodado)

Tipos de Falhas

Código

Tipo (Avaria no Transporte, Defeito de Fabricação, Entrega Parcial, ...)

Métodos de Transporte

Código

Método (Aéreo, Rodoviário, Ferroviário,...)

Moeda

Código

Moeda (R$, US$, €$, £$)

Demanda

Descrição: Tabela que armazena temporariamente as informações de demanda utilizadas nos planejamentos de processos

Planejamento

Produto

Período

Demanda Estimada

Horizontes de Planejamento

Unidade de Tempo

Número de Períodos

FORMULÁRIOS

Os formulários referentes às informações básicas necessárias para o funcionamento do *software* já foram desenvolvidos e serão apresentados mais adiante:

PRINCIPAIS RELACIONAMENTOS

Os relacionamentos entre as diversas tabelas apresentadas estão representados na figura da próxima página.

Analisando brevemente estes relacionamentos, pode-se obter uma idéia de como está estruturada a base de dados do software, o que é fundamental para a compreensão dos algoritmos dos principais códigos de programação do nosso software, que são os códigos do planejamento de necessidades e do planejamento de capacidade.

Por exemplo, temos a tabela de estrutura do produto, que armazena as informações de todos os relacionamentos pai-filho existentes nos diversos produtos da empresa.

Podemos destacar ainda os relacionamentos entre as tabelas *Equipamentos / Rotas / Processos*, fundamentais no desenvolvimento dos cálculos de planejamento de capacidade produtiva.

RELATÓRIOS DE CONTROLE

O software ainda nos permitirá obter de maneira rápida alguns relatórios de controle, com dados totalmente confiáveis, uma vez que a fonte de dados destes relatórios é a própria base única sobre a qual estarão operando todos os departamentos da empresa.

Dentre estes relatórios, podemos destacar os seguintes:

- *Planejamento Mestre de Produção*
- *Simulações Instantâneas de Necessidades (MRP-I)*
- *Necessidades de Compra / Produção (Produtos em Falta)*
- *Histórico de Vendas (por clientes / por produtos)*
- *Controle de Estoques*
- *Status de Produção*
- *Falhas de Produção (por equipamento / por produto)*
- *Pedidos Pendentes (Compra ou Venda)*
- *Ocorrências (Serviço ao Cliente)*

PRINCIPAIS ALGORITMOS

Os dois algoritmos mais importantes do software, sob o ponto de vista conceitual do MRP-II estarão inseridos no sistema de acordo com as figuras a seguir:

PLANEJAMENTO DE NECESSIDADES

O algoritmo esquematizado acima aparece de forma mais detalhada abaixo:

Parâmetros de Entrada (inseridos pelo usuário)
Produto
Horizonte de Planejamento
Previsão de vendas (por período)

Parâmetros de Cálculo (obtidos pelo software automaticamente)
Quantidades em Estoque
Lead Times
Quantidades em Processamento (Produção / Compras Pendentes)

Resultado do Planejamento (fornecido pelo Software)

Relatório de Compras (com as quantidades de cada produto e sub-produto)
Emissão das Ordens de Produção (a serem confirmados pelo usuário)
Emissão dos Pedidos de Compra (a serem confirmados pelo usuário)
Relatório de Reposição de Estoques (já considerando os produtos a serem utilizados na produção planejada)

PLANEJAMENT DE CAPACIDADE

O algoritmo esquematizado acima aparece de forma mais detalhada abaixo:

Parâmetros de Entrada (obtidos pelo *software* a partir do planejamento de necessidades)

Necessidades de Produção
Horizonte de Planejamento

Parâmetros de Cálculo (obtidos pelo software automaticamente)

Lead Times de Produção
Lead Times de Compras
Capacidade dos Equipamentos e Processos
Prazo de Entrega para o(s) produto(s) em questão

Resultado do Planejamento (fornecido pelo Software)

Elaboração do Plano de Produção Otimizado
Relatório de Providências extra-ordinárias para o cumprimento do plano de produção
Emissão das Ordens de Produção (a serem confirmados pelo usuário)

Implantação do sistema MRP-II

Ao se decidir por implantar o sistema MRP-II, a empresa deve tomar alguns cuidados. A efetiva implantação do sistema é um processo demorado, contínuo e que exige mudanças profundas no comportamento e na metodologia de trabalho de praticamente todos os funcionários da empresa.

Uma característica fundamental e indispensável é que todos os departamentos devem se utilizar de uma única base de dados, bem estruturada, que garanta a interdependência entre os departamentos, para evitar que haja desencontro de informações.

A alteração parte da completa reformulação no sistema de planejamento da empresa – o que afeta diretamente o trabalho da alta administração (diretoria e superintendência). Cabe à alta administração definir onde a empresa quer chegar; o que e quanto deve ser melhorado.

Só então deve-se partir para a análise dos *softwares* disponíveis no mercado, não esquecendo de levar em consideração a sua maleabilidade para que ele possa ser totalmente adaptado às particularidades da empresa.

Um erro muito comum – e caro – é iniciar o processo a partir da compra de um *software*, para posterior adaptação às características da empresa. Muitas vezes, descobre-se tarde demais que as alterações necessárias para fazer esta adaptação são caras demais, e então corre-se o risco de trabalhar com um sistema totalmente inadequado, o que praticamente assegura o fracasso da implantação.

Vale ressaltar que ter um *software* bom é uma condição necessária, mas não suficiente para o sucesso da implantação.

Uma vez escolhido o *software* mais adequado, é necessário providenciar treinamento aos funcionários, não apenas no que diz respeito à utilização do *software*, mas também em função dos conceitos teóricos aplicados por trás do programa.

Aí sim pode-se partir para a implantação propriamente dita, seguindo alguns procedimentos explicitados a seguir:

- *Preparação do projeto de implantação*

- *Programa de treinamento*
- *Desenho procedimental do sistema de planejamento*
- *Revisão dos processos logísticos*
- *Garantia da acurácia da base de dados*
- *Elaboração de Procedimentos*
- *Corte do sistema antigo e entrada do novo sistema*

Algumas premissas básicas para a suficiência ligadas ao processo de implantação do sistema são:

- *Comprometimento da alta direção com os objetivos da implantação*
- *Treinamento intensivo e continuado em todos os níveis*
- *Gerenciamento adequado do processo de implantação.*

A respeito do gerenciamento do processo, existem diversas estruturas de equipe de implantação, mas podemos ressaltar algumas condições básicas, como a necessidade de haver um gerente dedicado exclusivamente ao projeto, além do fundamental envolvimento das diretorias dos diversos departamentos da empresa, indispensável para garantir que o projeto possa seguir o seu curso sem entraves em nenhuma área.

Uma sugestão de estrutura com as respectivas descrições das funções será desenvolvida nas próximas etapas do projeto.

Apresentação do Software

A partir dos formulários desenvolvidos, foram feitas algumas formatações para melhor apresentação do software junto ao seu usuário final e acesso mais fácil às suas funções através de menus e sub-menus de fácil manipulação.

Ao longo do processo de simulação e entrada de dados, verificou-se a necessidade de mais algumas tabelas e formulários para entrada de dados que são utilizados nos cálculos de necessidades MRP.

Alguns destes formulários e também alguns relatórios de saída estão apresentados nos itens a seguir.

MENU PRINCIPAL

Esta é a tela inicial do programa, onde através do menu à esquerda, tem-se o acesso aos sub-menus da direita que são os principais formulários de entradas de dados e relatórios de saídas:

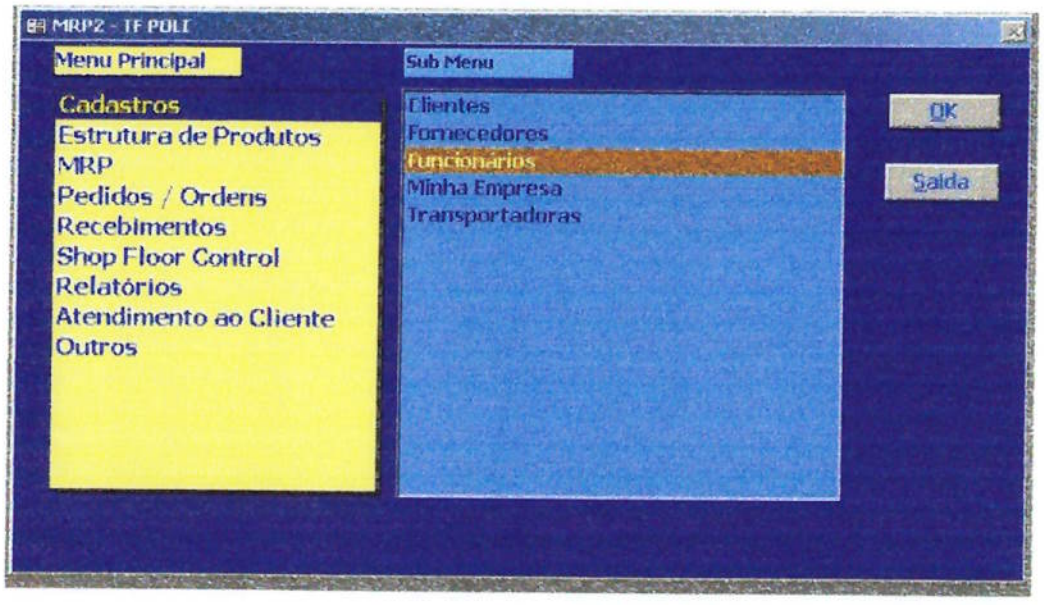


Figura 1 - menu principal

CADASTROS BÁSICOS

Aqui neste sub-menu encontram-se os formulários para entrada de dados referentes à empresa, funcionários, fornecedores, clientes e transportadoras.

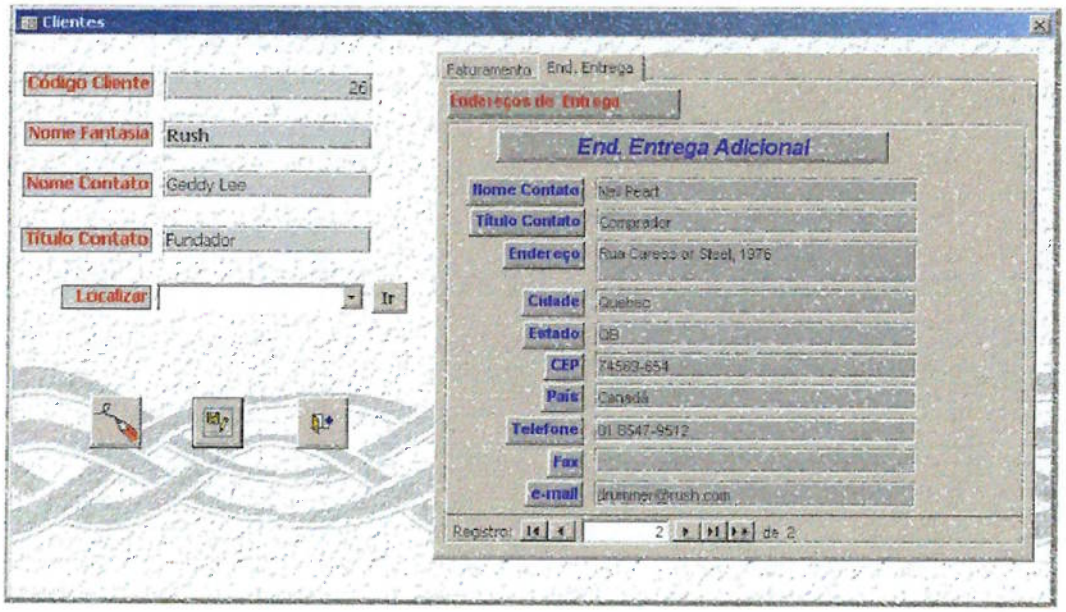


Figura 2 - cadastro de clientes

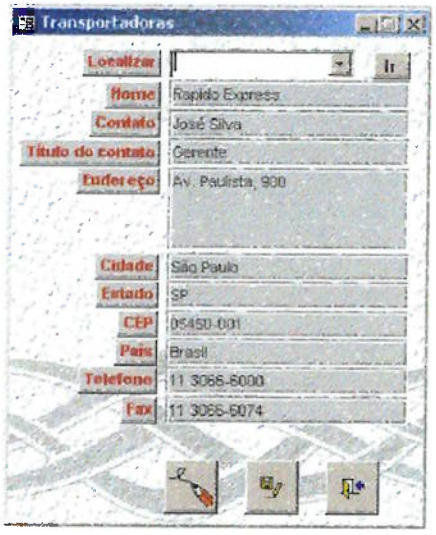


Figura 3 - cadastro de transportadoras

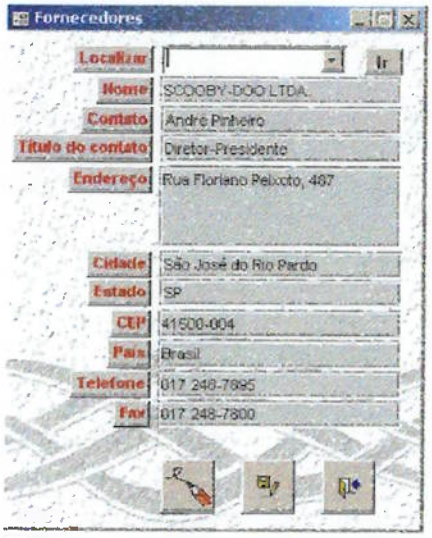


Figura 4 - cadastro de fornecedores

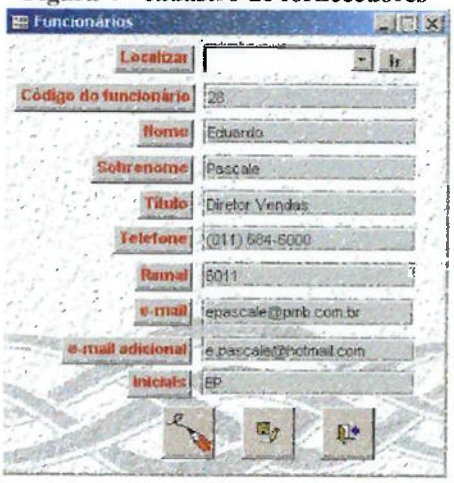


Figura 5 - cadastro de funcionários

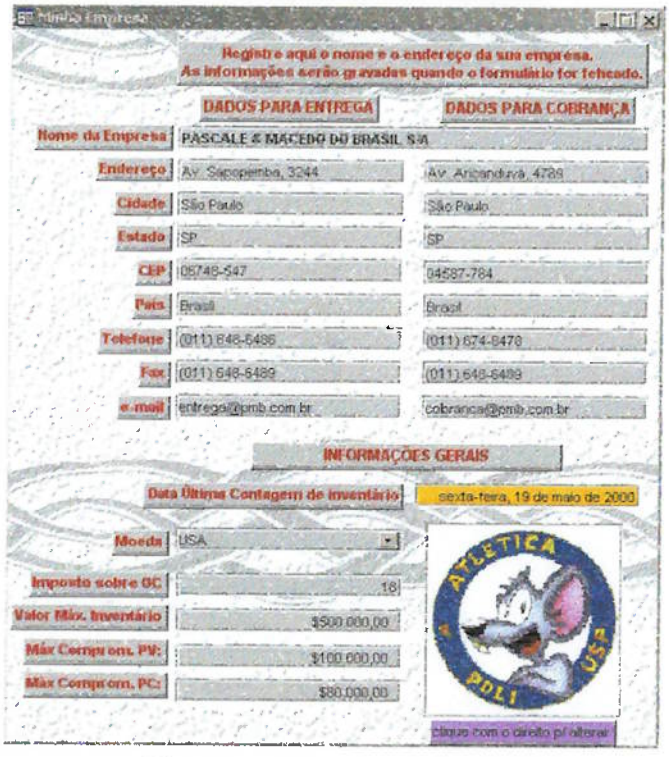


Figura 6 - cadastro minha empresa

ESTRUTURA DE PRODUTO (EP)

Na Estrutura de Produto (EP) são encontrados os formulários para entrar e editar os dados sobre produtos (cadastro mestre de produtos), bem como suas estruturas, criando estruturas multi-níveis de produtos que são levadas em conta nos cálculos MRP. Abaixo estão demonstradas as telas de entrada de informações sobre produtos e de criação de níveis na estrutura do produto.

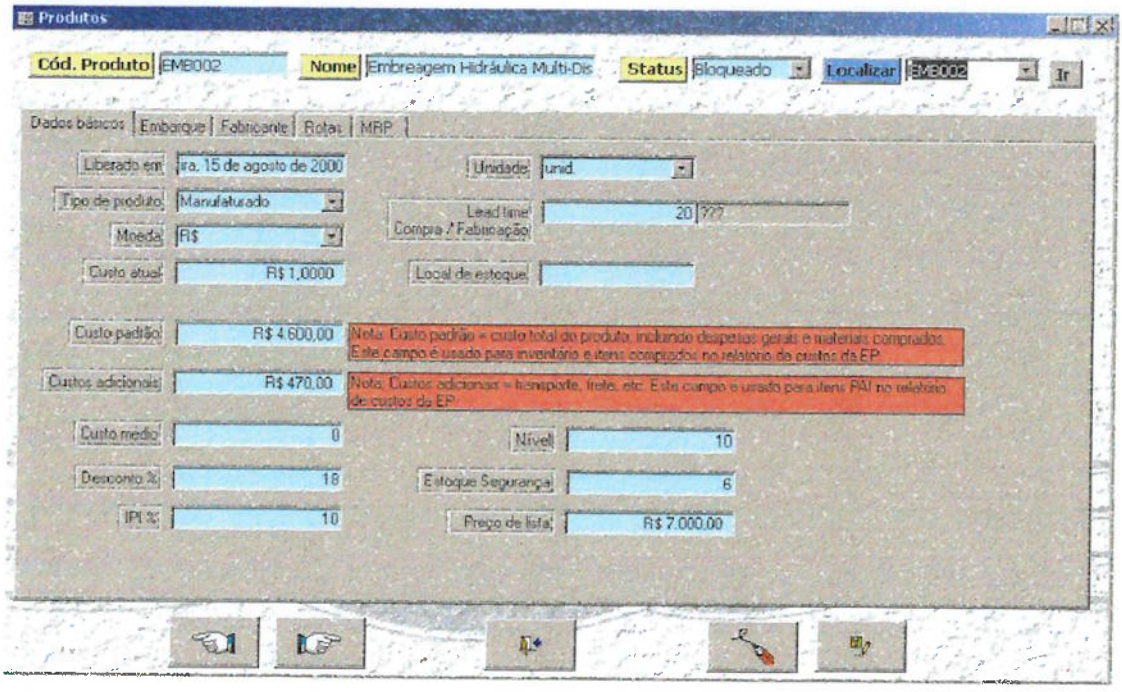


Figura 7 – cadastro mestre de produtos

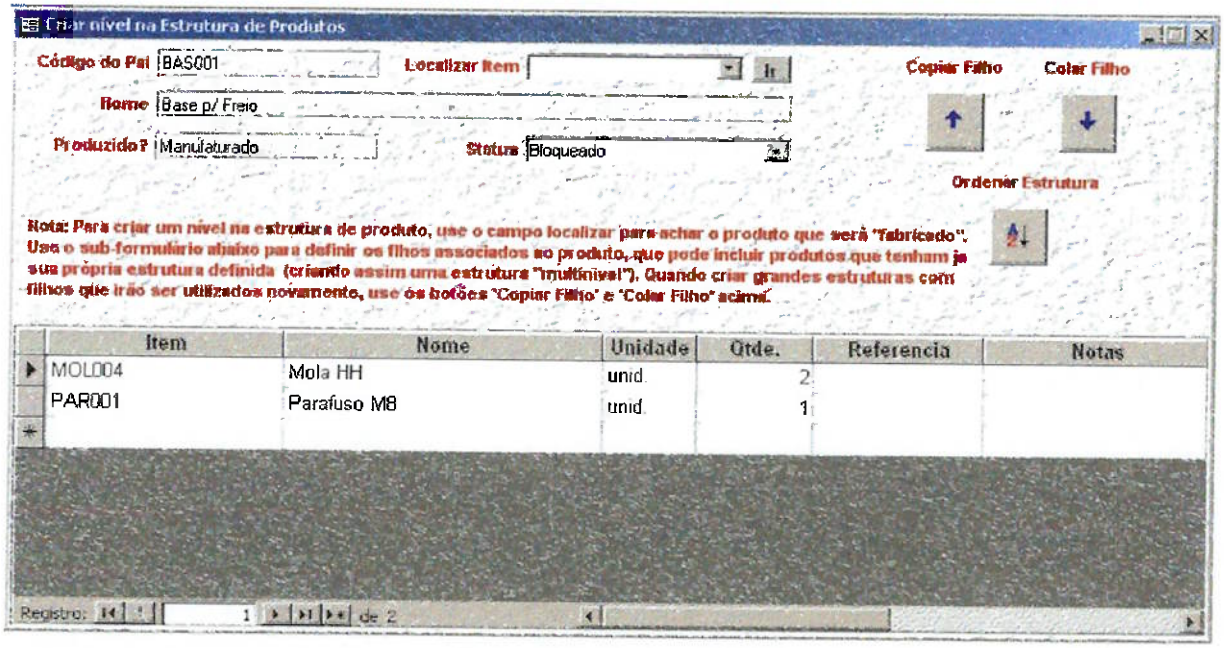


Figura 8 – criar nível na EP

Dentro deste menu também se encontra uma ferramenta muito útil que, através da entrada de um produto qualquer, localiza todos os itens onde ele é usado, ou seja, seus "Pais" na EP.

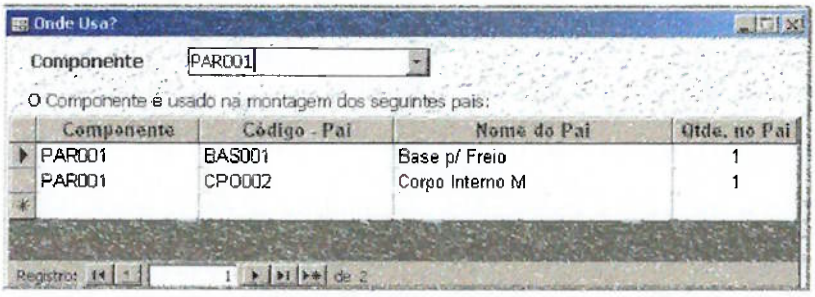


Figura 9 – localizar pais

ORDENS / PEDIDOS

Aqui geram-se todos os pedidos e ordens de compra, venda e produção da empresa.

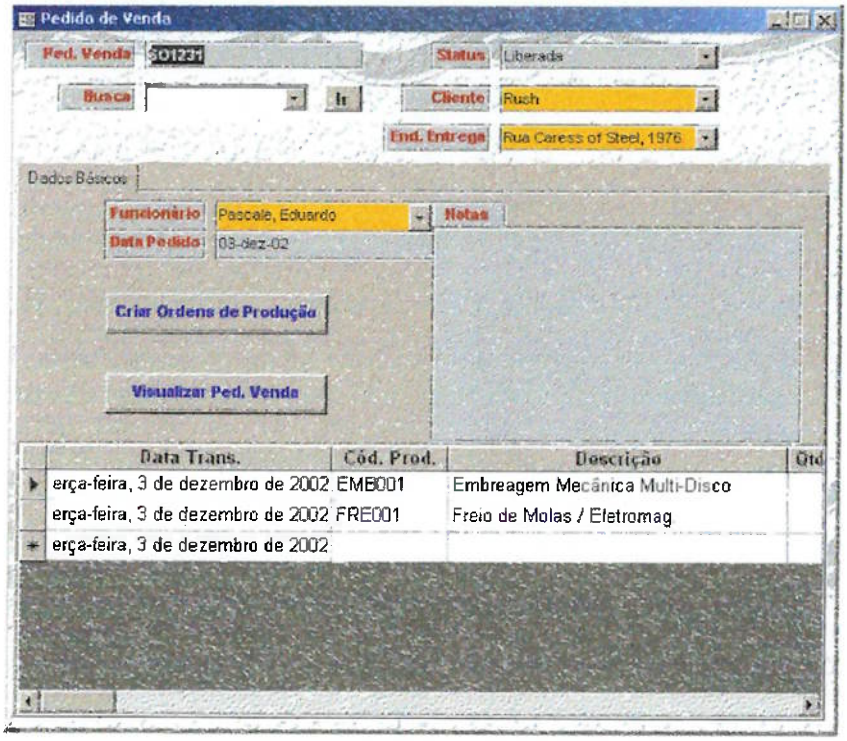


Figura 10 – pedido de venda

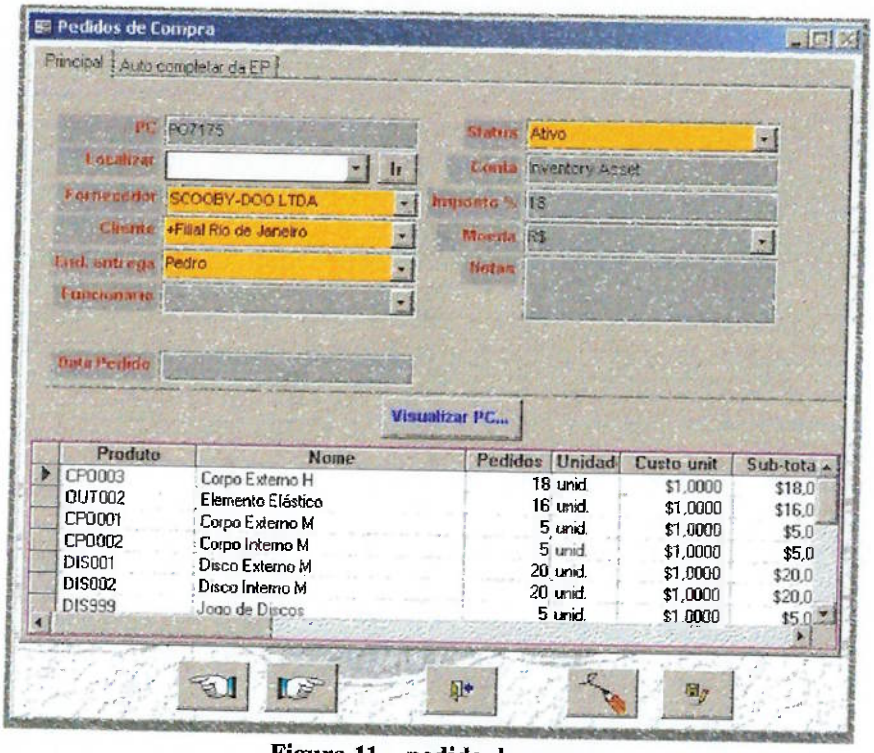


Figura 11 – pedido de compra

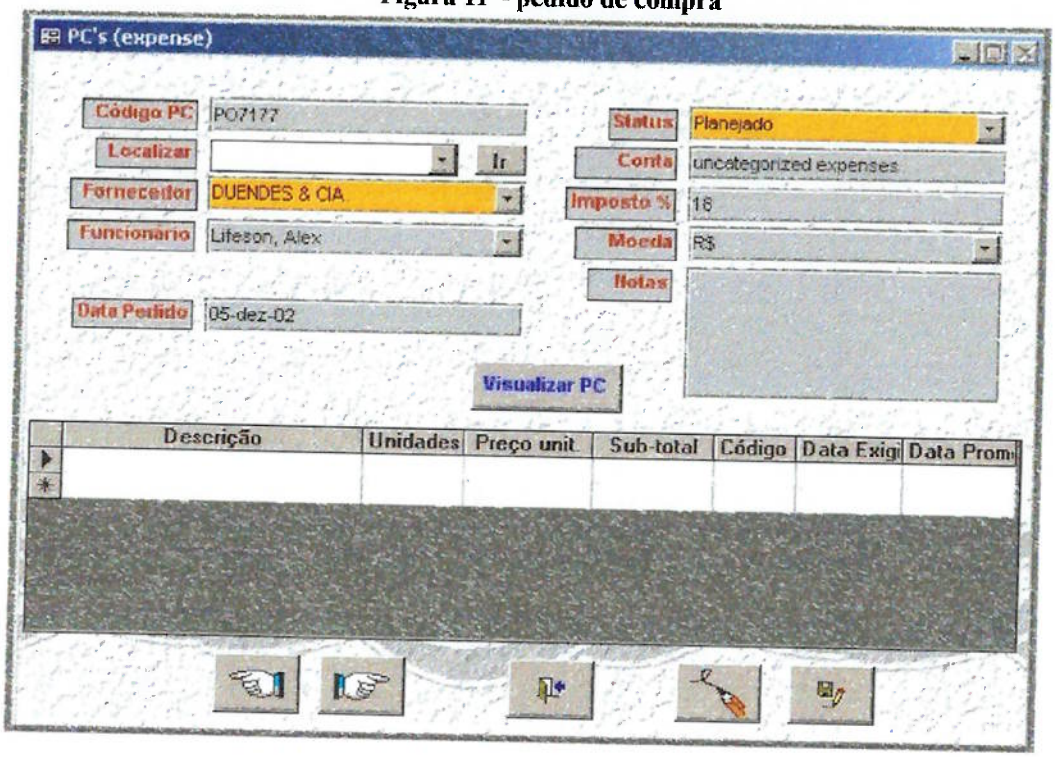


Figura 12 – pedido de compra

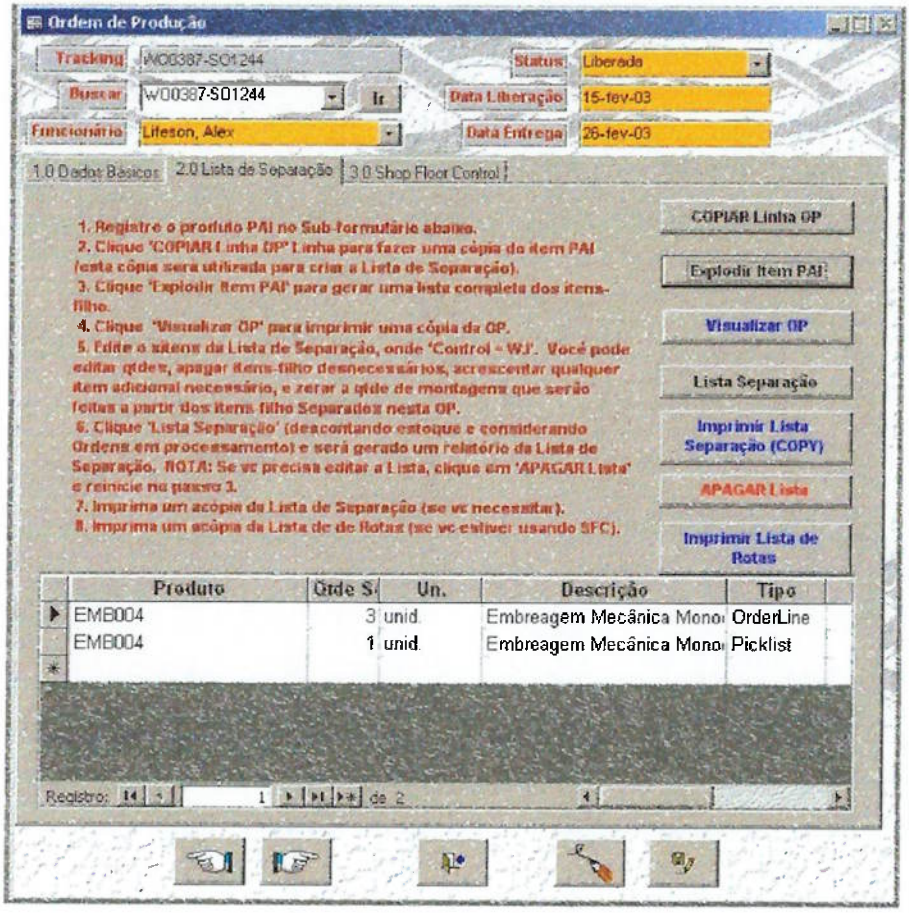


Figura 13 – ordem de produção

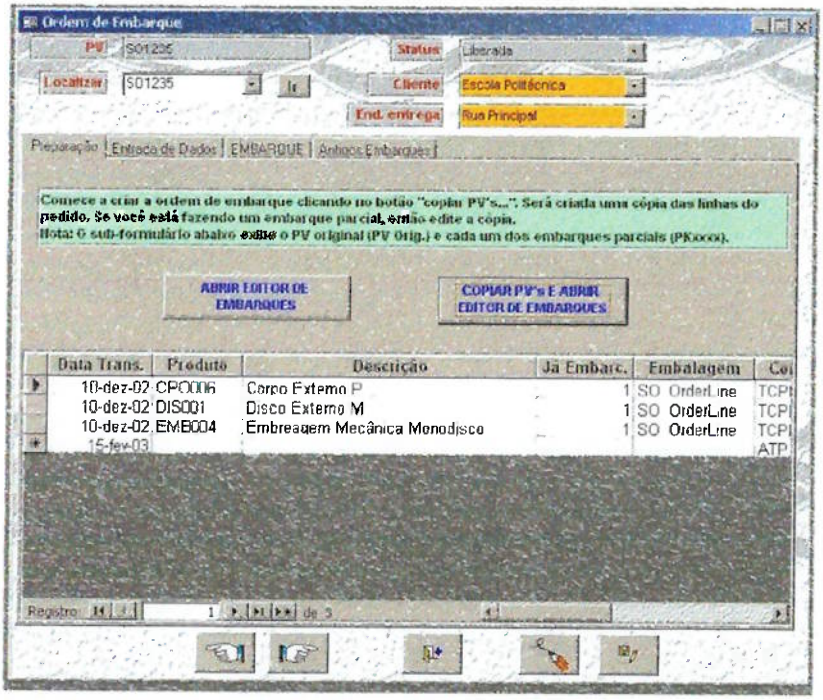


Figura 14 – ordem de embarque

RECEBIMENTO DE ORDENS / PEDIDOS

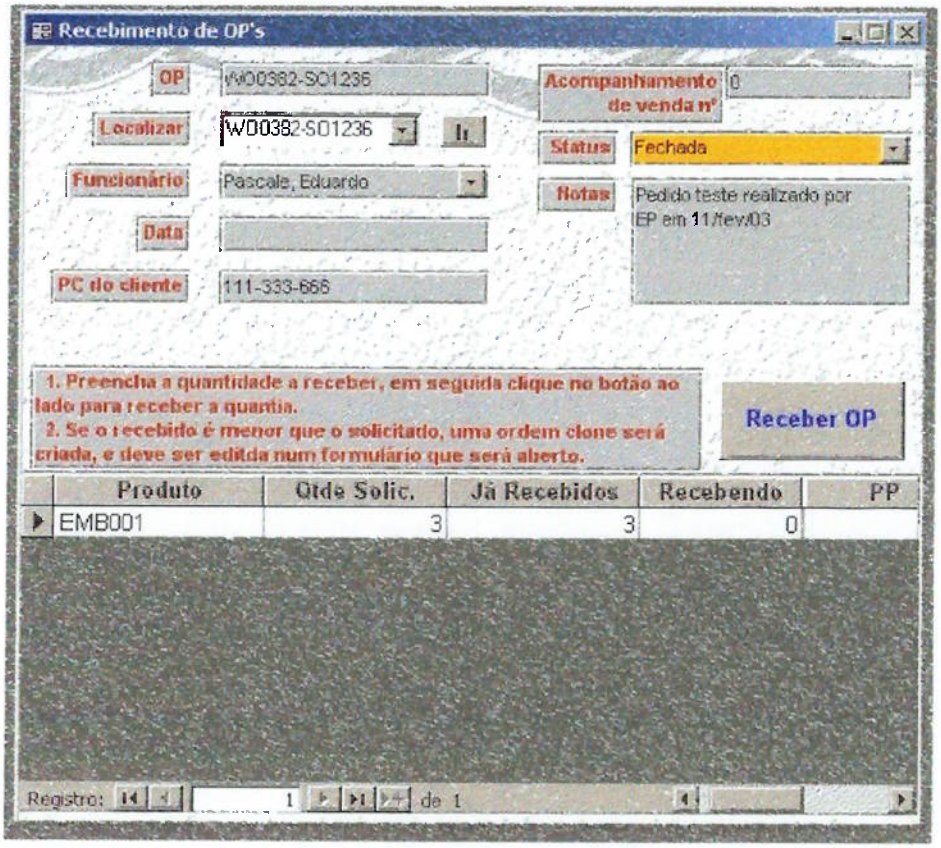


Figura 15 – recebimento de OP´s

Figura 16 – recebimento parcial de OP´s

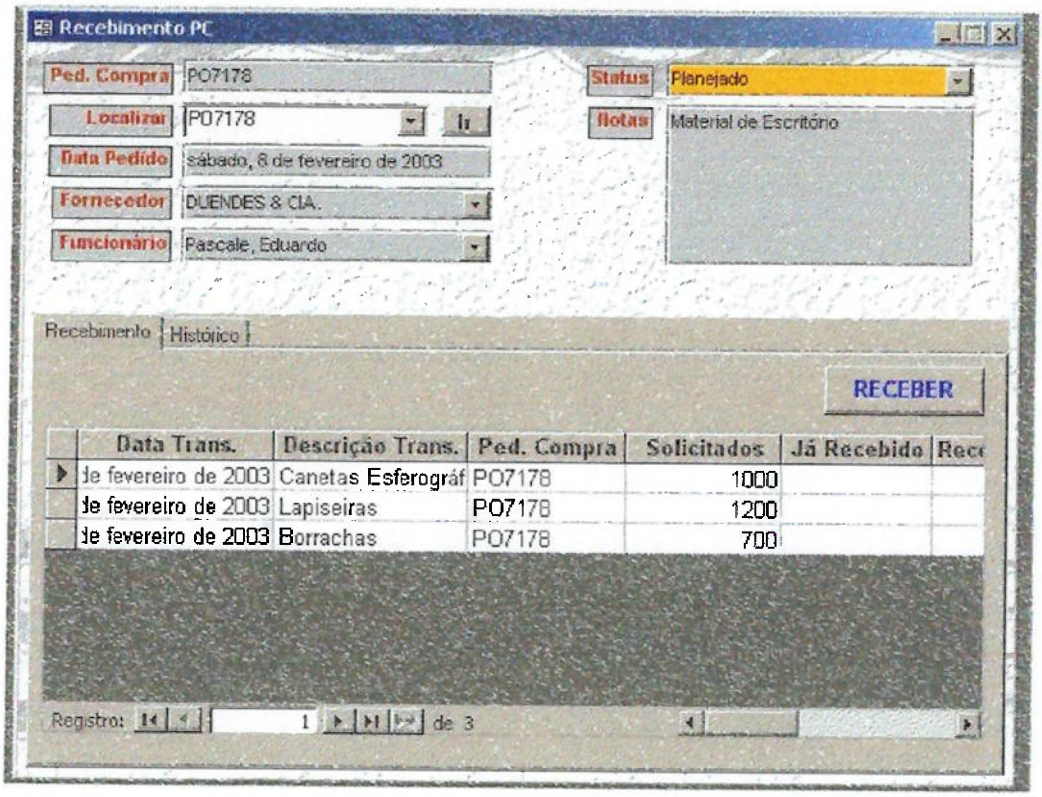


Figura 17 – recebimento de PC´s

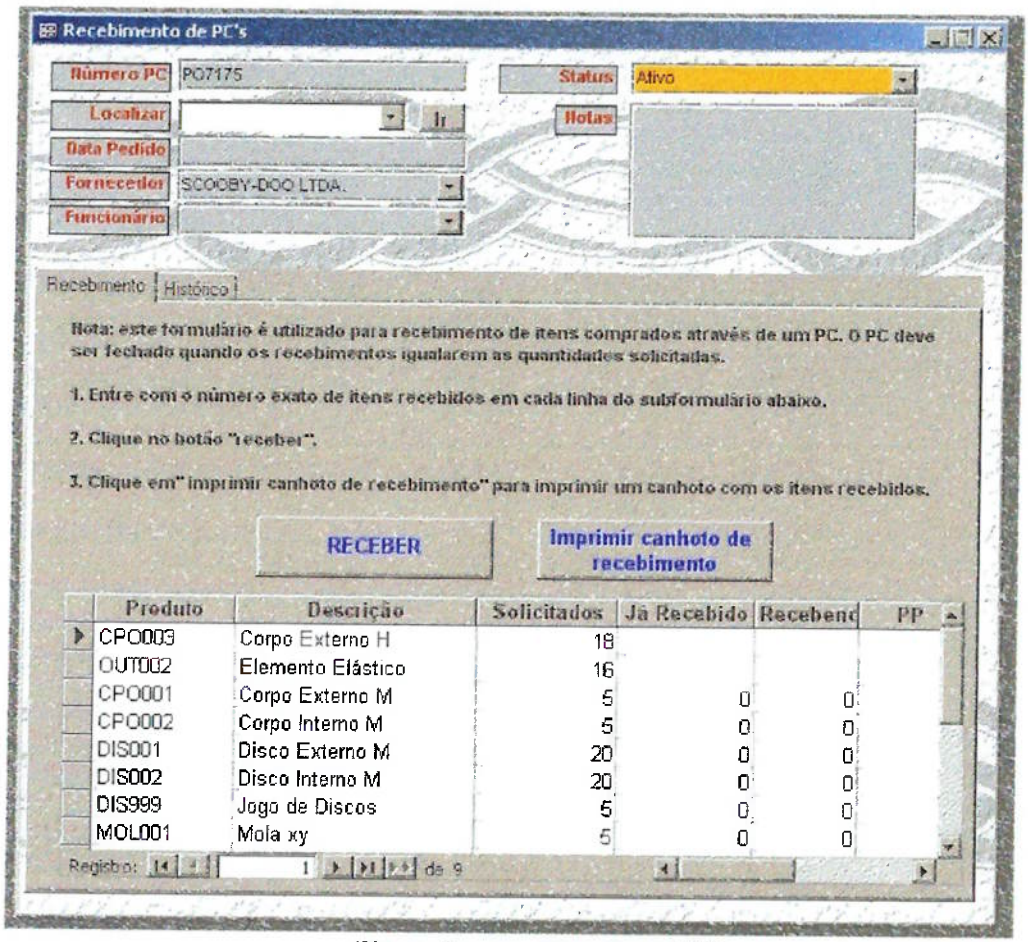


Figura 18 – recebimento de PC´s

DADOS PARA PLANEJAMENTO

Antes de fazer as simulações do MRP o usuário deve fazer as definições de períodos e horizontes de planejamento através dos formulários abaixo:

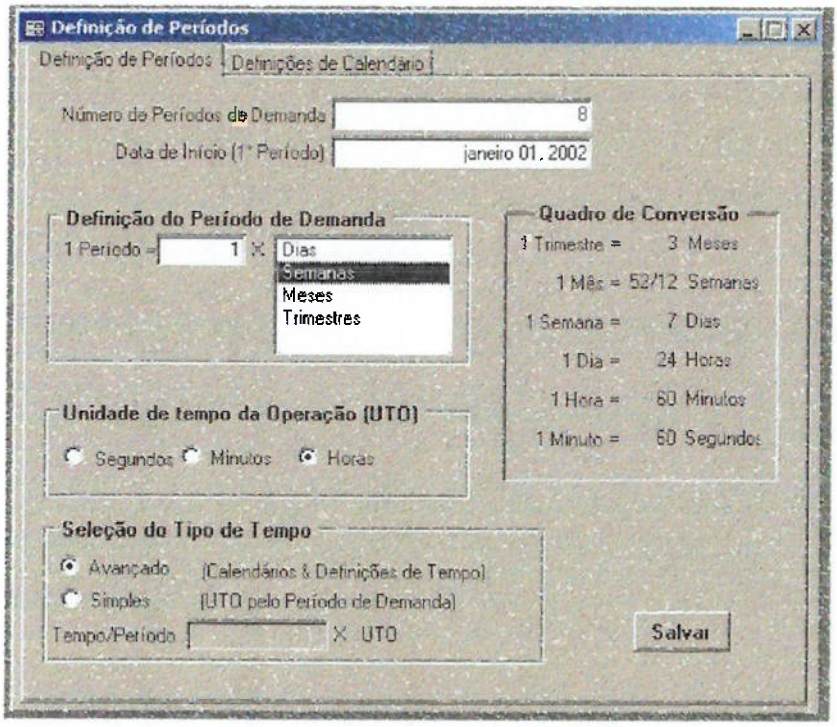


Figura 19 – definição de períodos

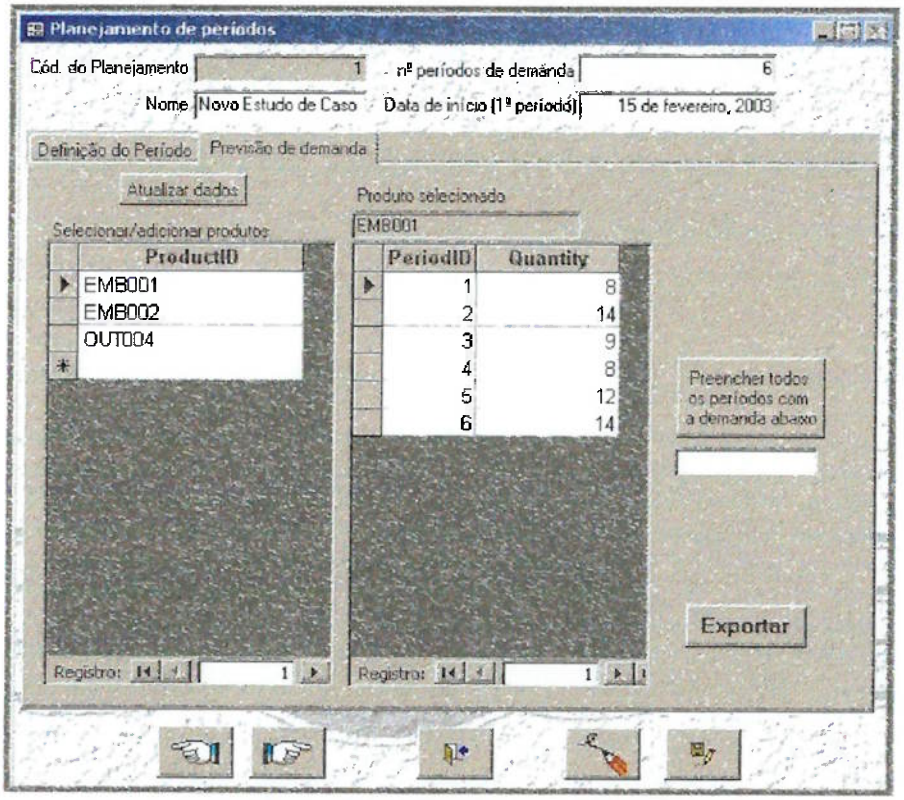


Figura 20 – definição de planejamentos

Figura 21 – importar arquivos para MRP

DADOS PARA CÁLCULO MRP

Aqui encontra-se o coração do software. Aqui o usuário insere dados de entrada informando horizonte de planejamento, demanda prevista, lotes, etc... e o programa gera um planejamento e todas as ordens necessárias para cumpri-lo. Os formulários onde são inseridos tais dados e as saídas são expostos a seguir:

Figura 22 – simulação de MRP

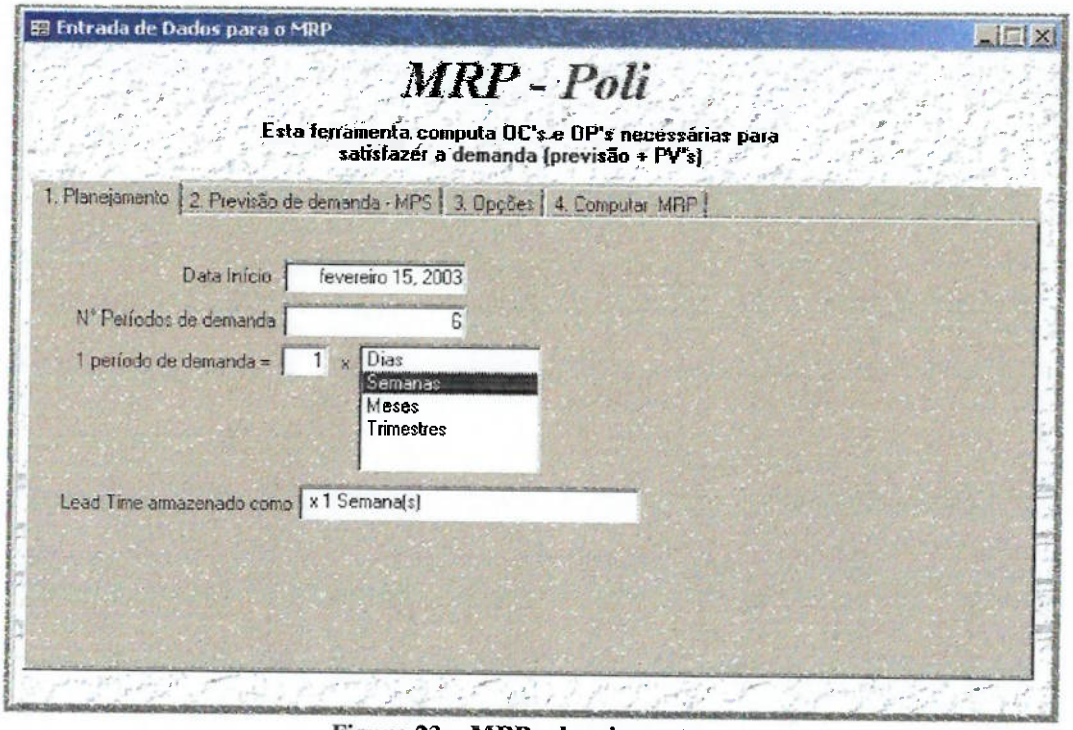


Figura 23 – MRP planejamento

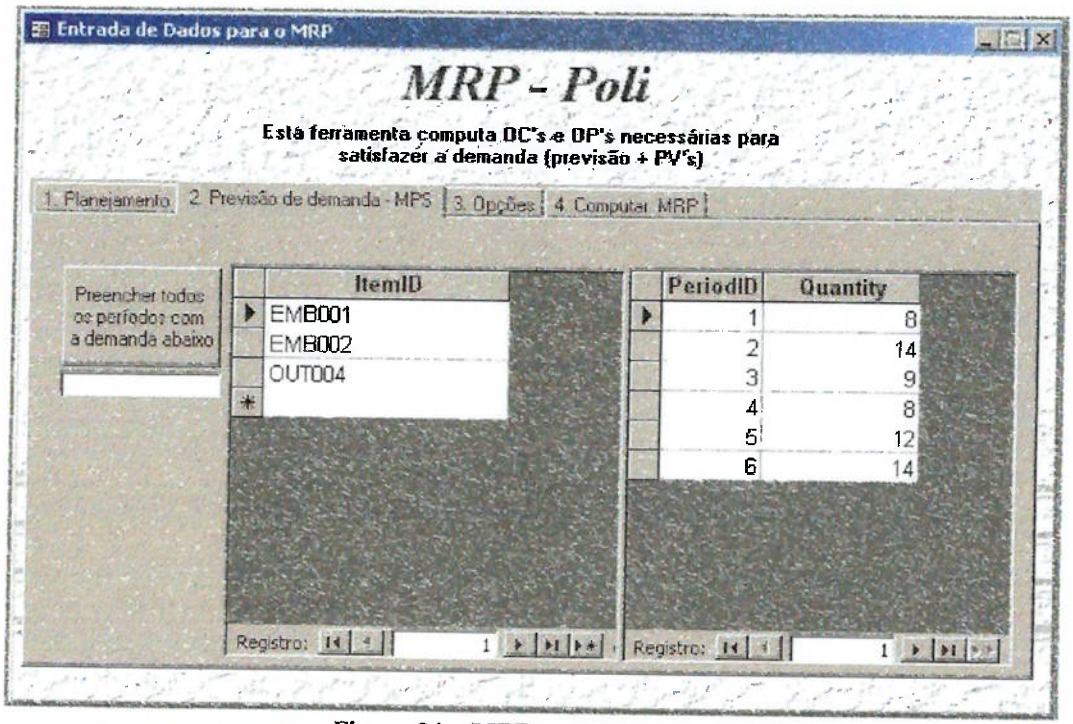


Figura 24 – MRP previsão de demanda

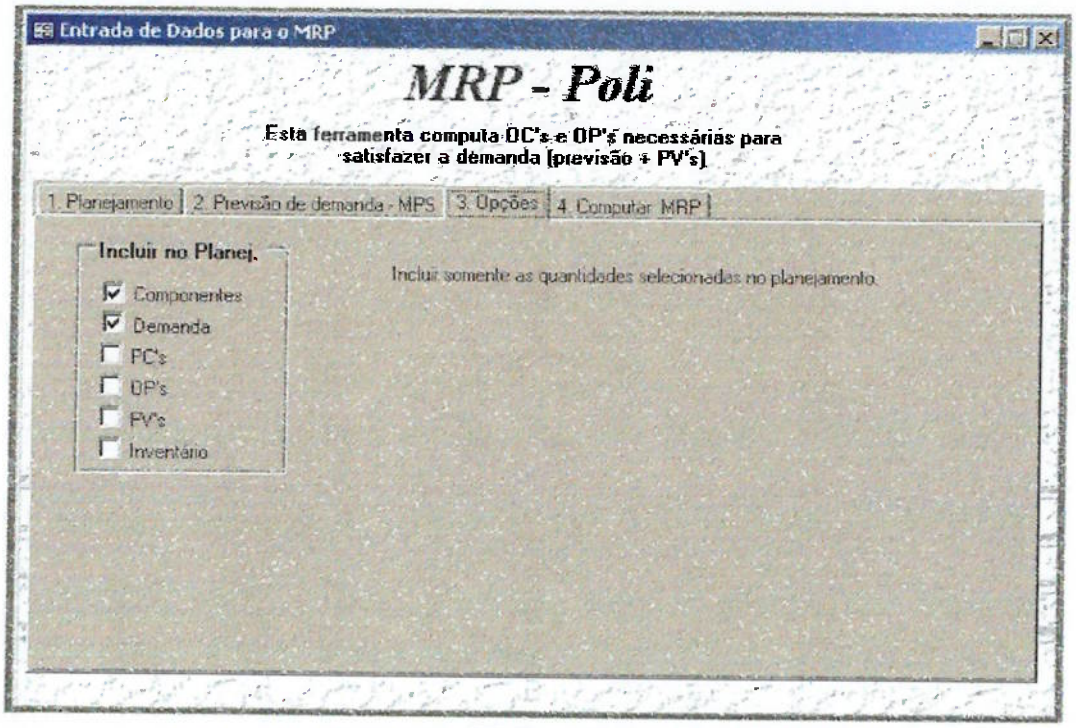


Figura 25 – MRP opções para o planejamento

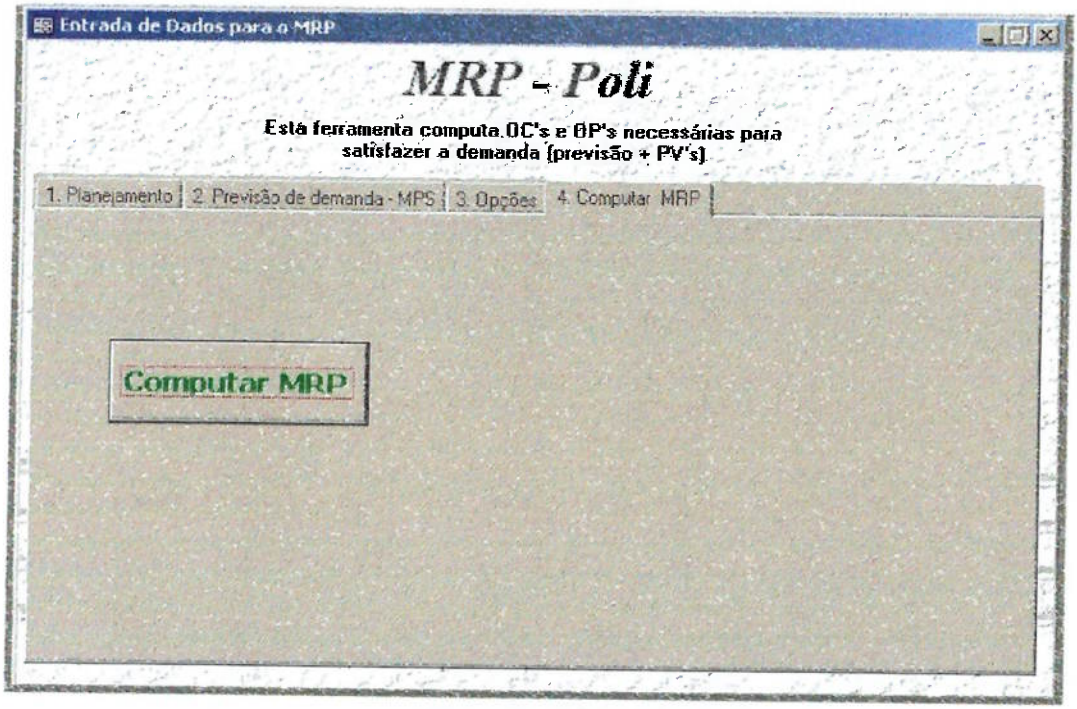


Figura 26 – MRP computar

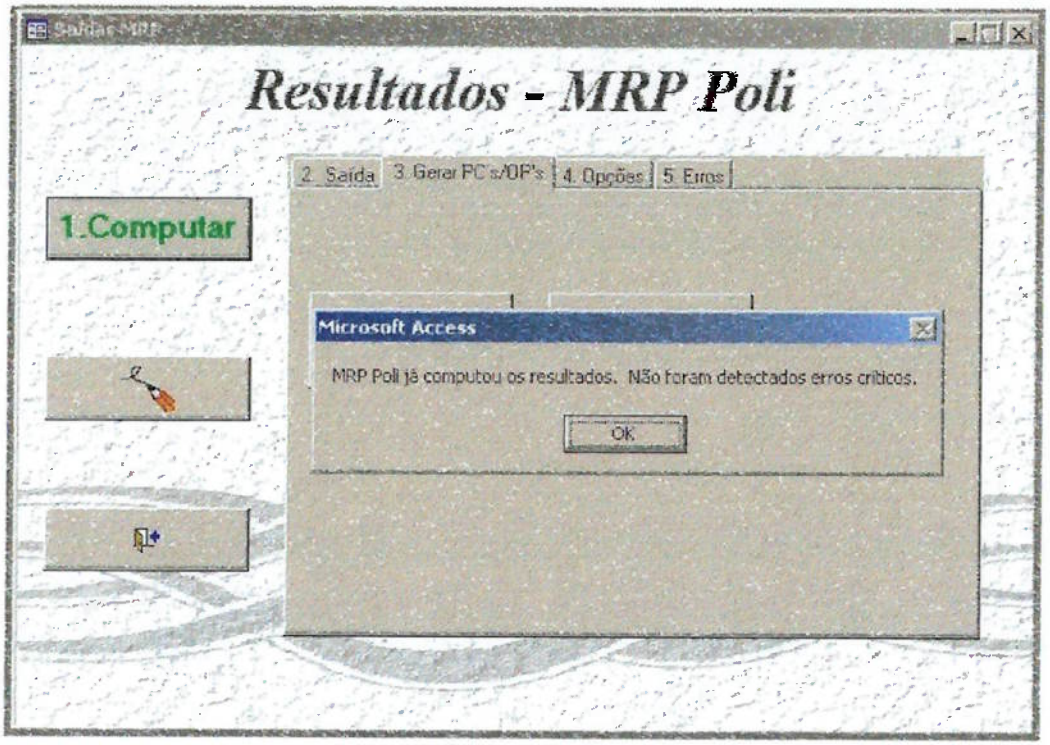


Figura 27 – MRP saída ok

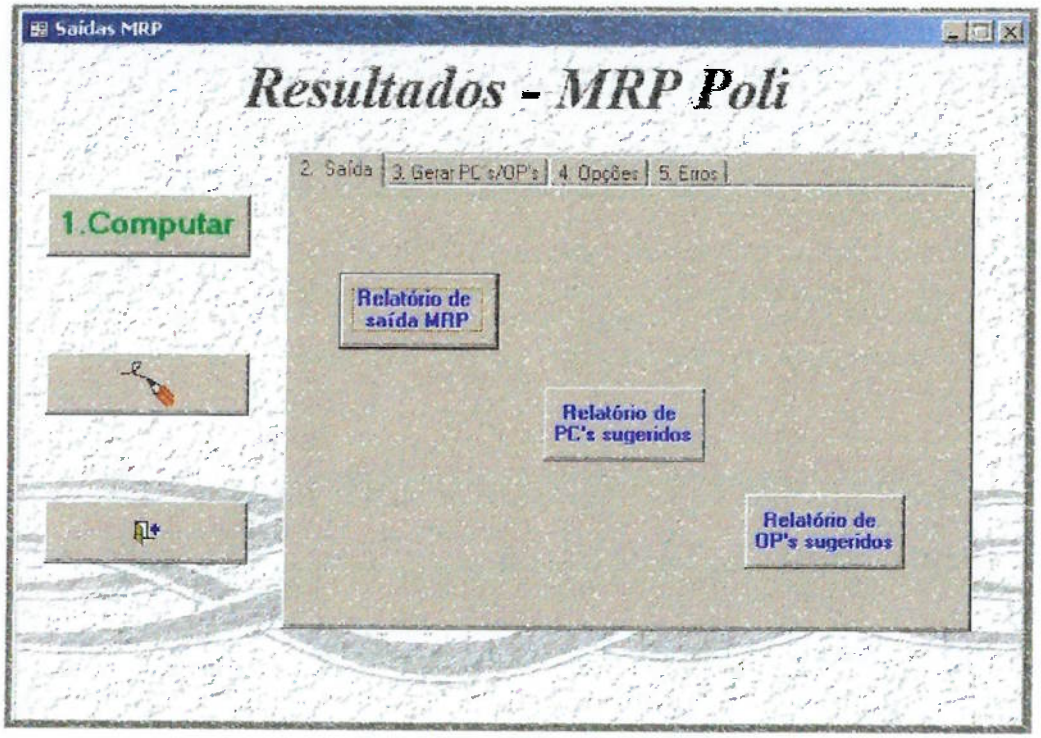


Figura 28 – MRP gerar relatórios

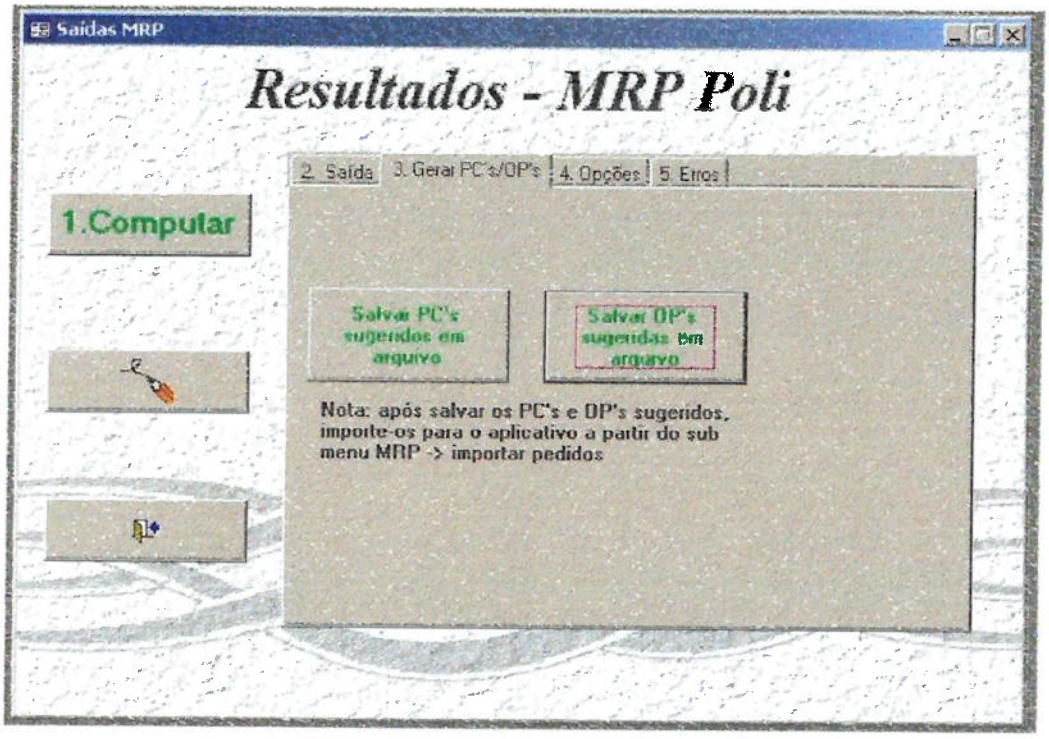


Figura 29 – MRP gerar pedidos e ordens

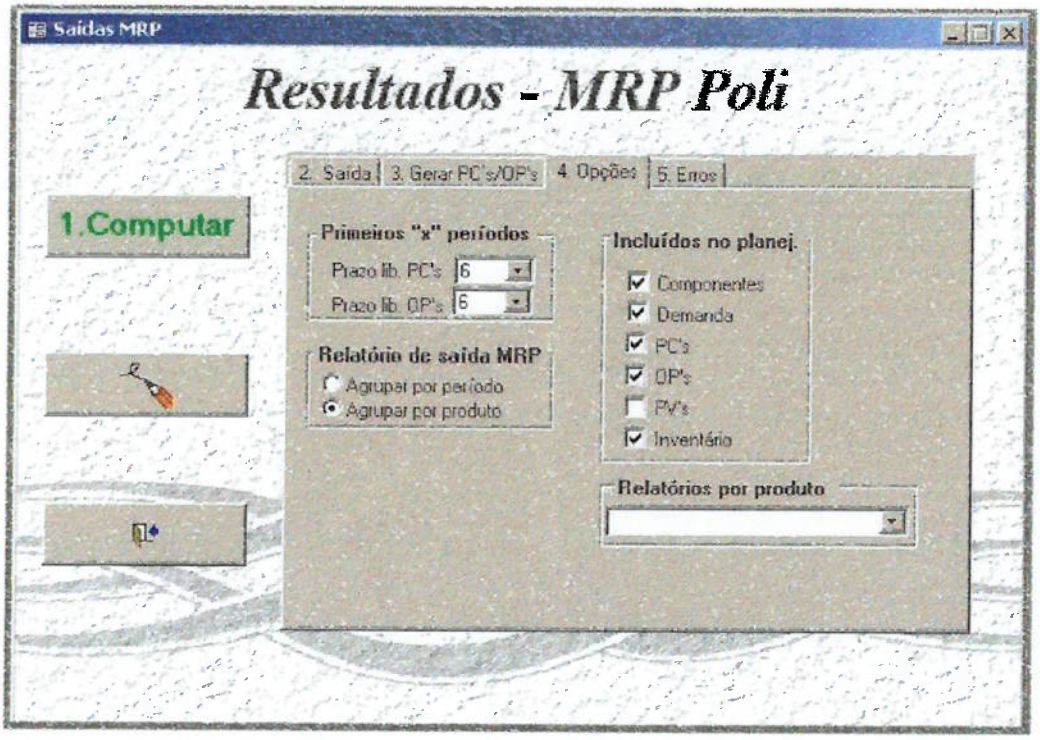


Figura 30 – MRP opções para relatórios

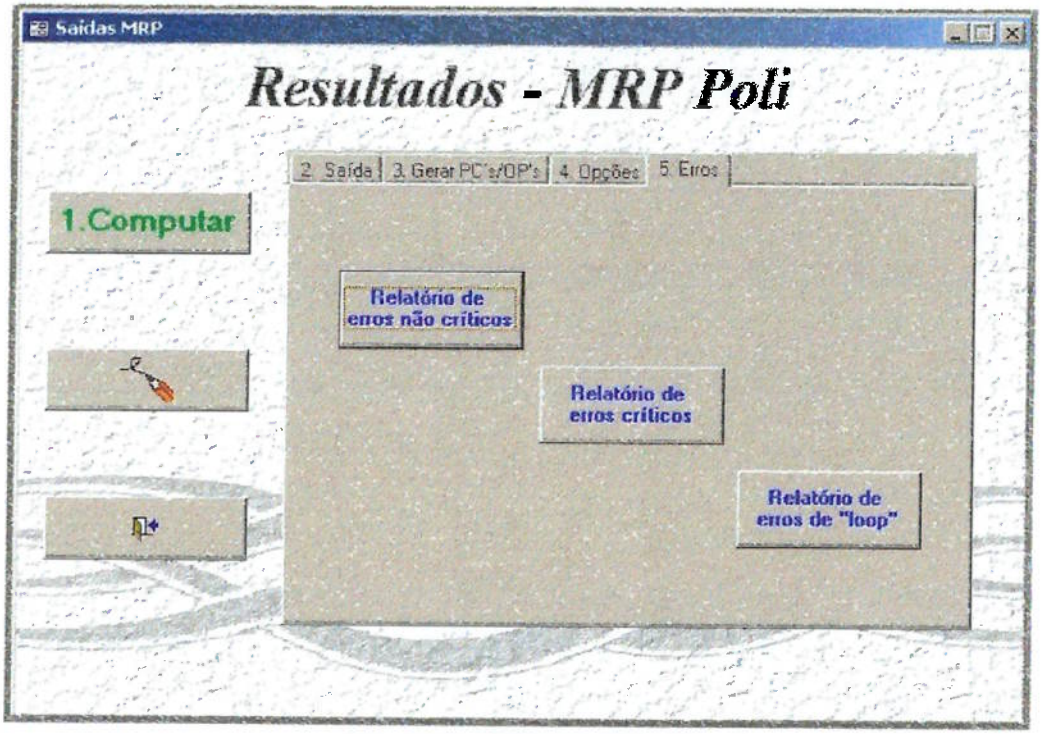


Figura 31 – MRP relatórios de erros

SHOP FLOOR CONTROL

No menu SFC são encontrados formulários para entradas de equipamentos e operações.

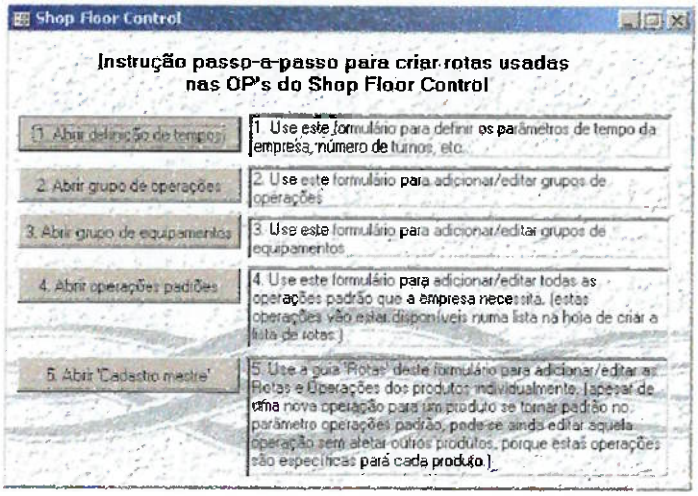


Figura 32 – Shop Floor Control

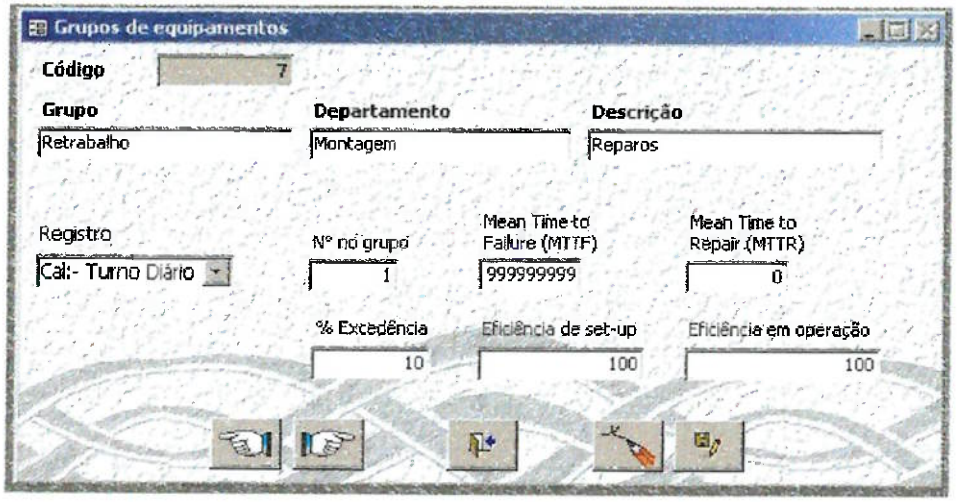


Figura 33 – grupos de equipamentos

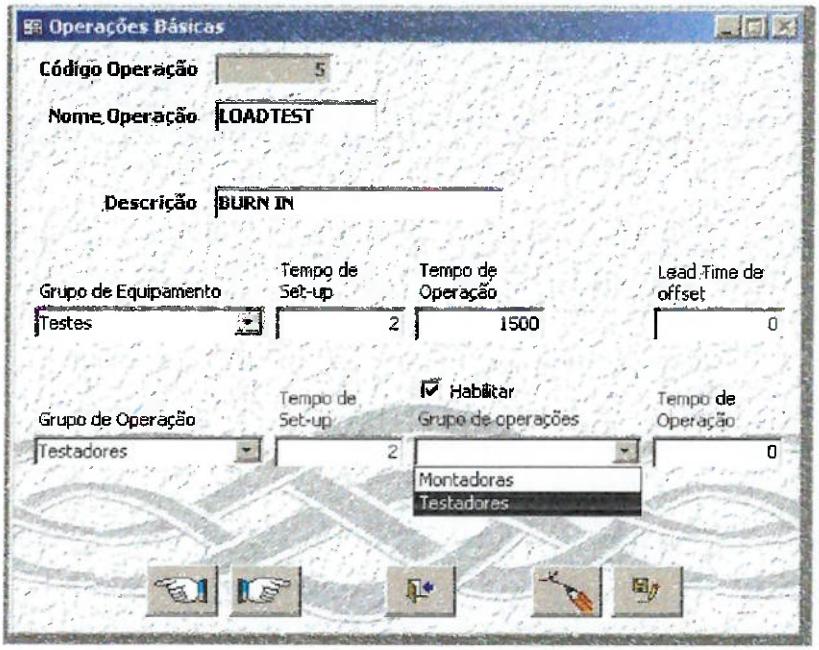


Figura 34 – operações básicas

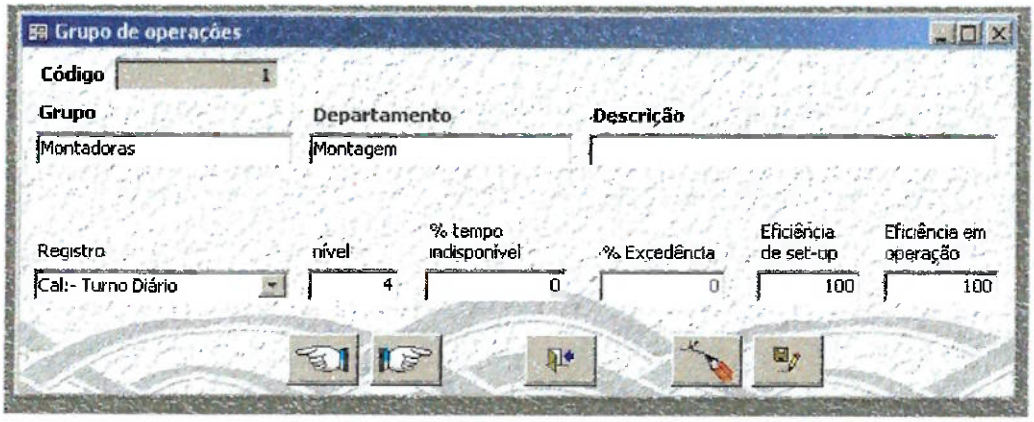


Figura 35 – grupos de operações

FORMULÁRIOS EXTRAS

Alguns formulários muito úteis ao usuário:

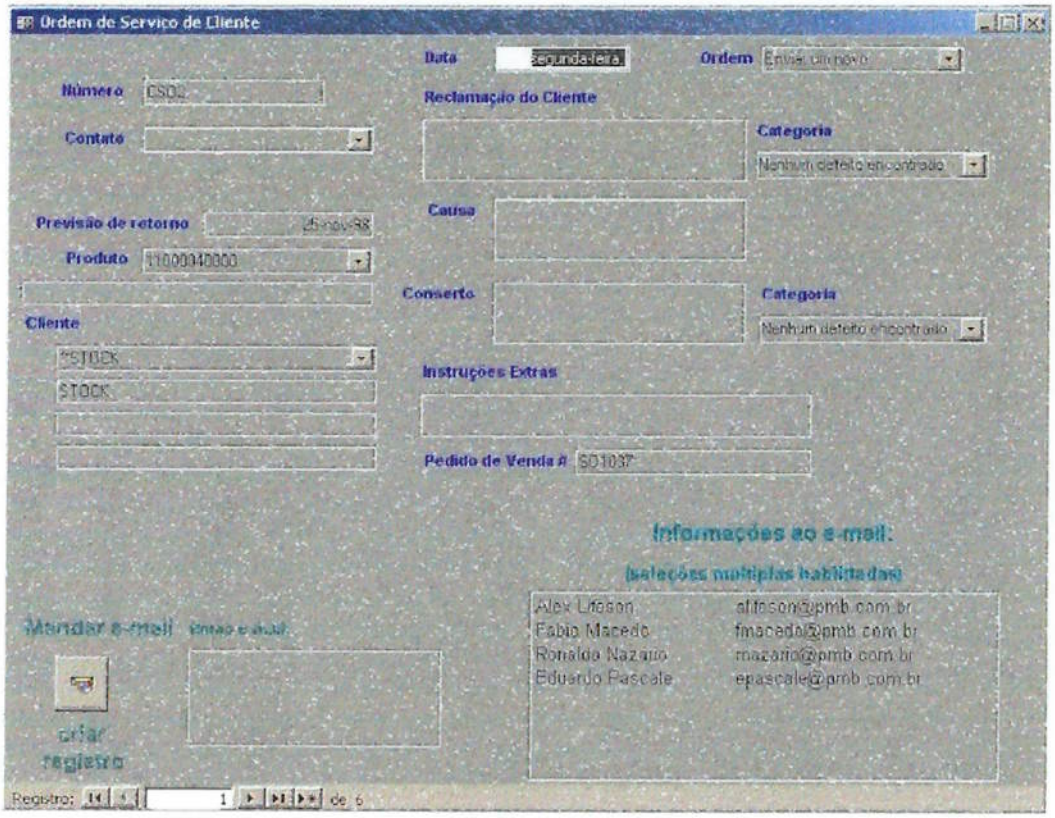


Figura 36 – atendimento ao cliente

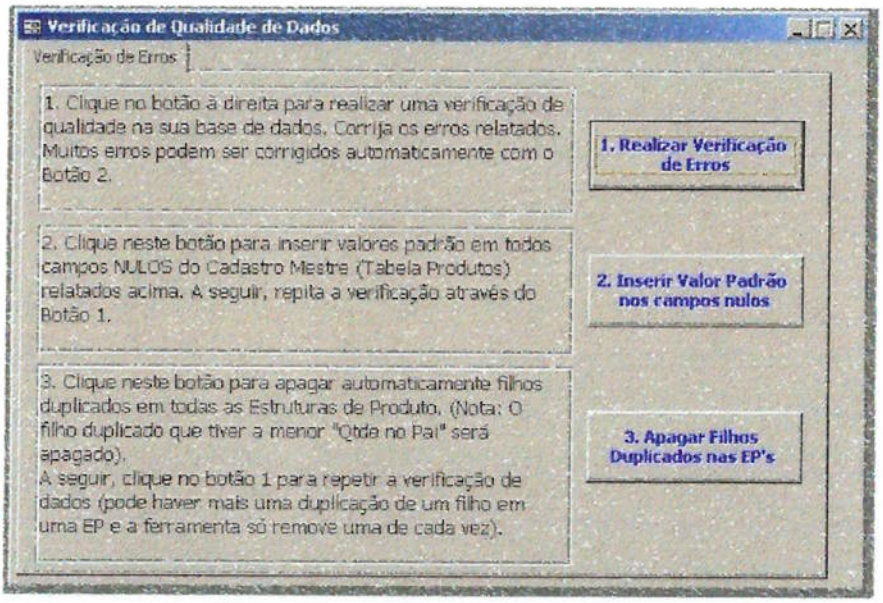


Figura 37 – verificação de dados do programa

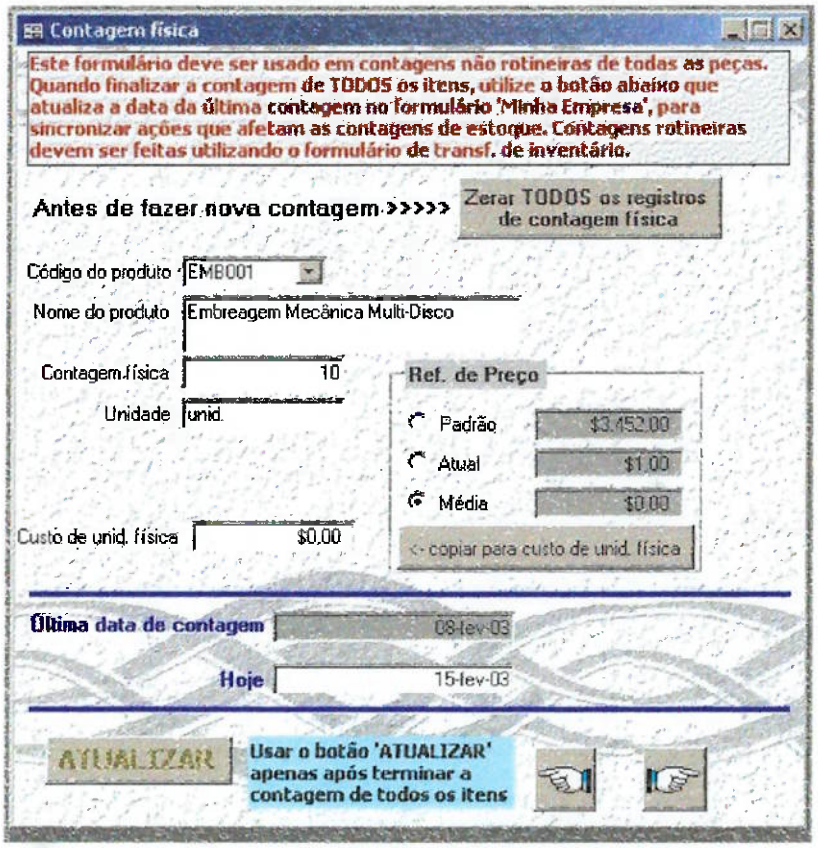


Figura 38 – contagem física de inventário

Figura 39 – transferência de inventário

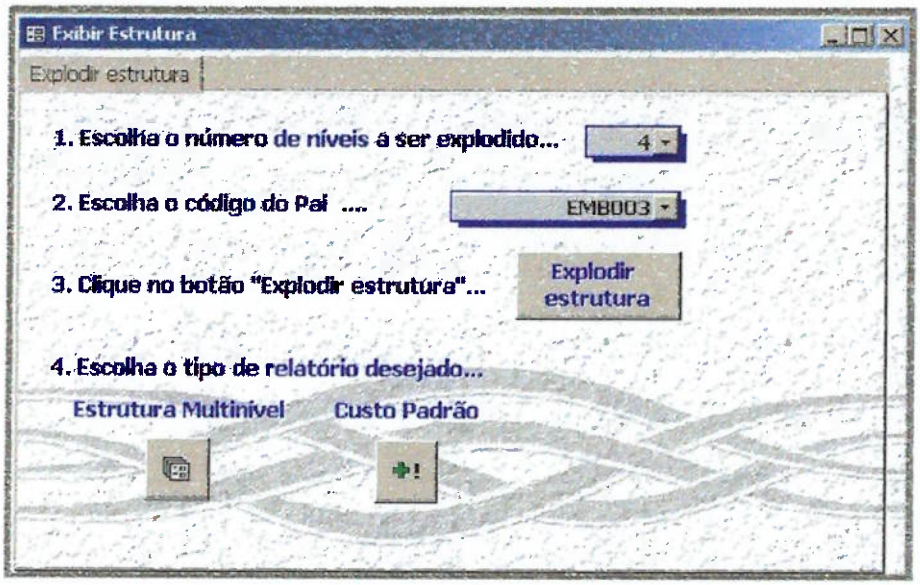


Figura 40 – exibir estrutura do produto

RELATÓRIOS

Para gerar diversos tipos de relatórios, a respeito de clientes, pedidos, ordens, produtos e suas estruturas, etc... Alguns são exemplificados abaixo e outros estão exibidos no estudo de caso, logo abaixo.

Figura 41 – status de material

Status de Material

Atualizado até:
terça-feira, 3 de dezembro de 2002

Notas:
Prontos = Matéria-prima em estoque
Alocados = Matéria-prima em estoque, porém já alocada para pedidos de clients através de ordens de produção
DPP = Qtde "Prontos" - Qtde "Alocados"
Solicitados = Materiais solicitados em PCS, porém ainda não recebido

Cód. Produto	Nome Produto	Prontos	Alocados	DPP	Solicitados
BAS001	Base p/ Freio	31	0	31	0
CPO001	Corpo Externo M	6	0	6	0
CPO002	Corpo Interno M	8	0	8	0
CPO003	Corpo Externo H	345	0	345	0
CPO004	Corpo Interno H	412	0	412	0
CPO005	Corpo Interno P	36	0	36	0
CPO006	Corpo Externo P	0	0	0	0
CPO007	Corpo Interno p/ Freio	0	0	0	0
DIS001	Disco Externo M	0	0	0	0
DIS002	Disco Interno M	0	0	0	0
DIS003	Disco de Fricção H	0	0	0	0
DIS004	Disco de Fricção p/ Freio	0	0	0	0
DIS005	Disco Único M	0	0	0	0
DIS999	Jogo de Discos	0	0	0	0
EMB001	Embreagem Mecânica Multi-Disco	0	0	0	0
EMB002	Embreagem Hidráulica Multi-Disco	0	0	0	0

Figura 42 – relatório: status de material

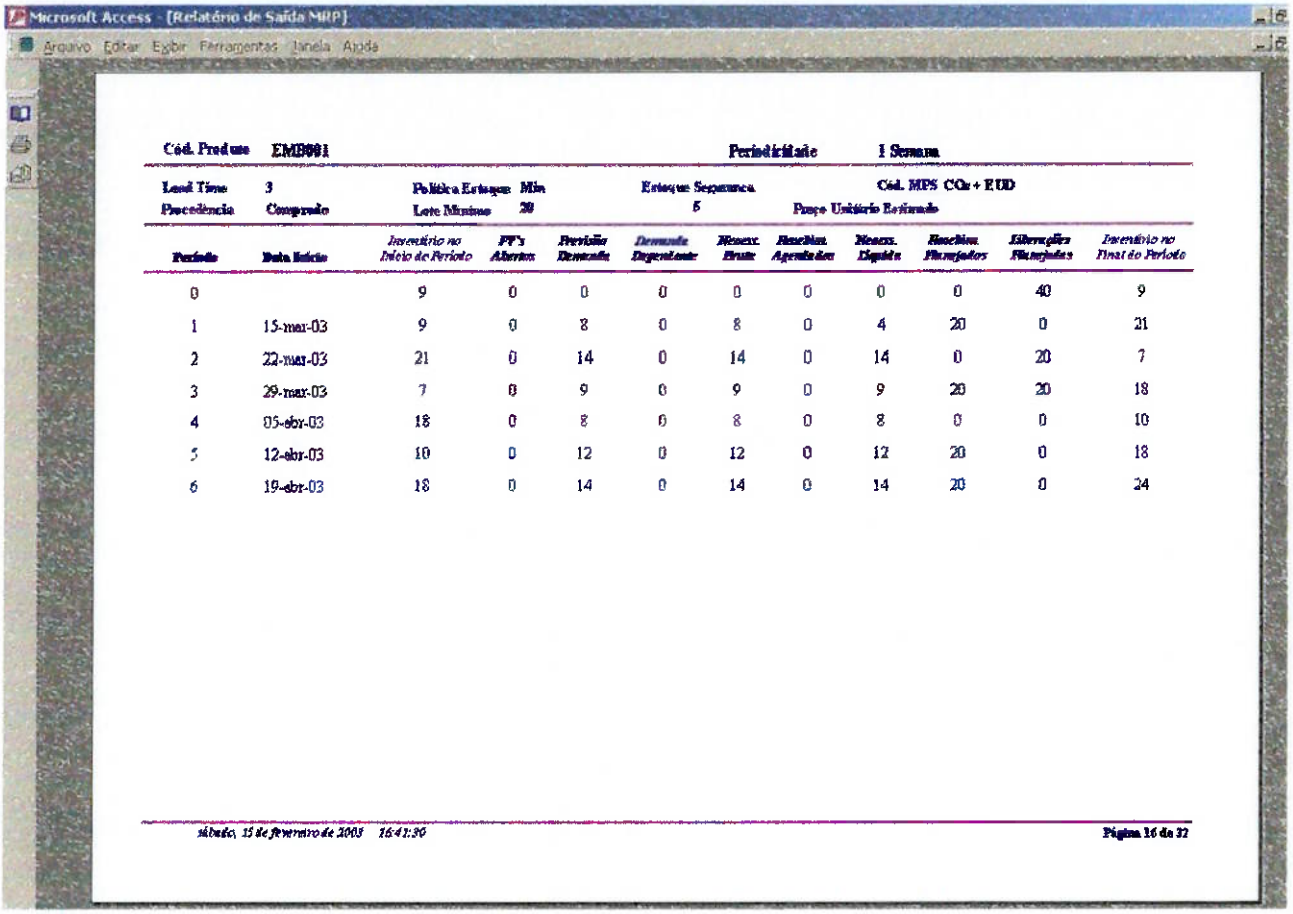


Figura 43 – relatório: saída do MRP

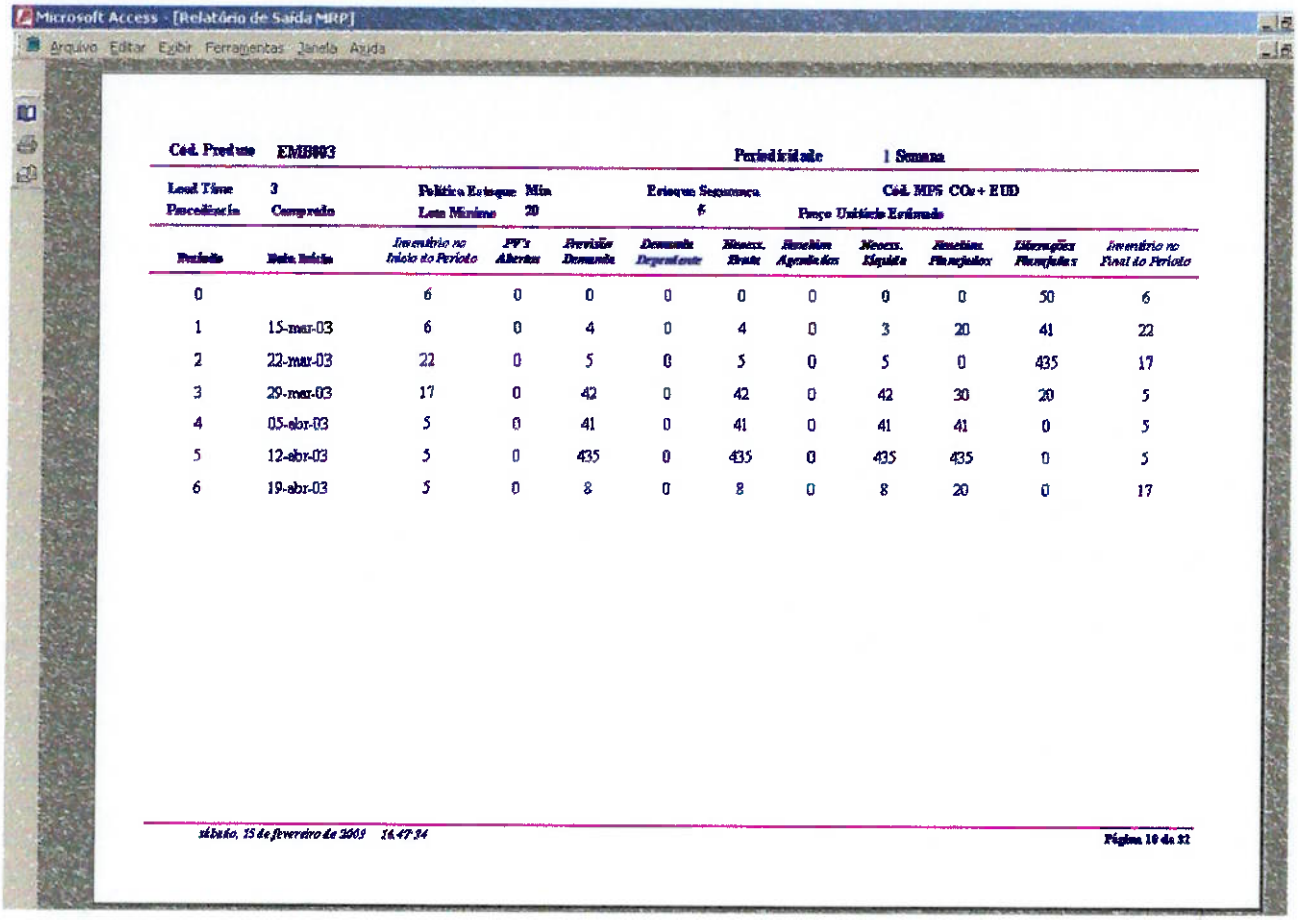
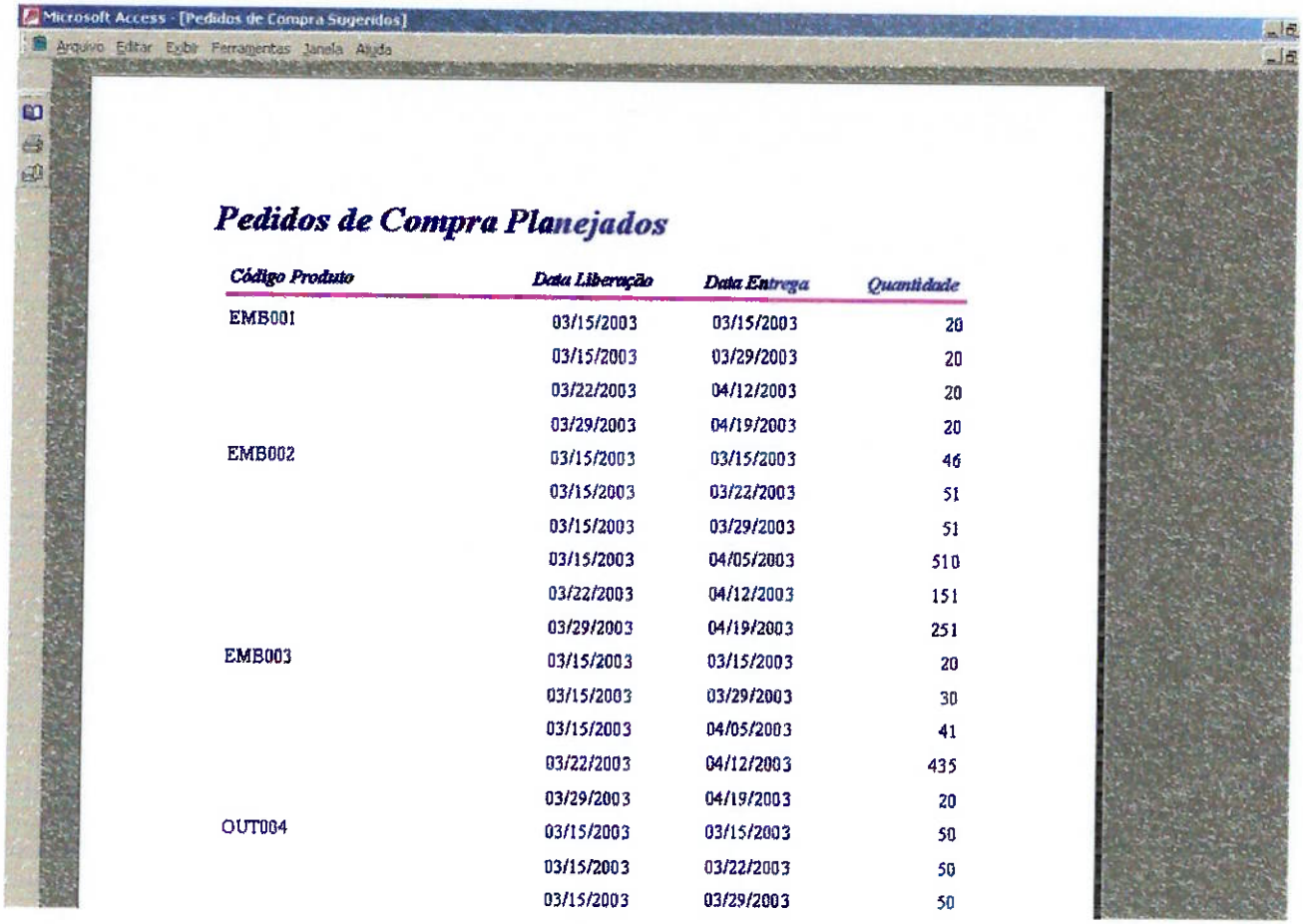


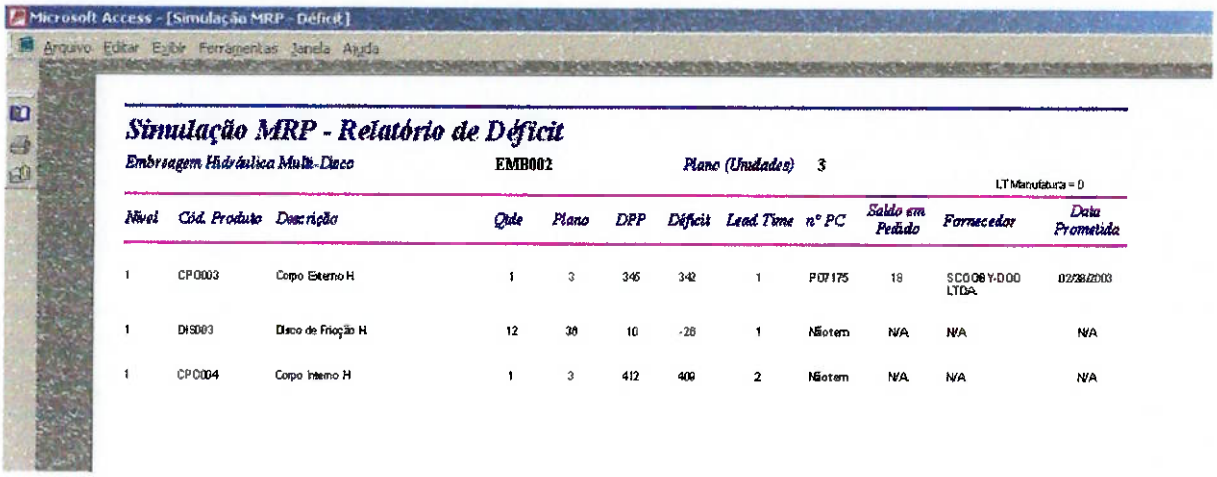
Figura 44 – relatório: saída do MRP



Pedidos de Compra Planejados

Código Produto	Data Liberação	Data Entrega	Quantidade
EMB001	03/15/2003	03/15/2003	20
	03/15/2003	03/29/2003	20
	03/22/2003	04/12/2003	20
	03/29/2003	04/19/2003	20
EMB002	03/15/2003	03/15/2003	46
	03/15/2003	03/22/2003	51
	03/15/2003	03/29/2003	51
	03/15/2003	04/05/2003	510
	03/22/2003	04/12/2003	151
	03/29/2003	04/19/2003	251
EMB003	03/15/2003	03/15/2003	20
	03/15/2003	03/29/2003	30
	03/15/2003	04/05/2003	41
	03/22/2003	04/12/2003	435
	03/29/2003	04/19/2003	20
OUT004	03/15/2003	03/15/2003	50
	03/15/2003	03/22/2003	50
	03/15/2003	03/29/2003	50

Figura 45 – relatório: saída do MRP, pedidos sugeridos



Microsoft Access - [Simulação MRP - Déficit]

Arquivo Editar Exibir Ferramentas Janela Ajuda

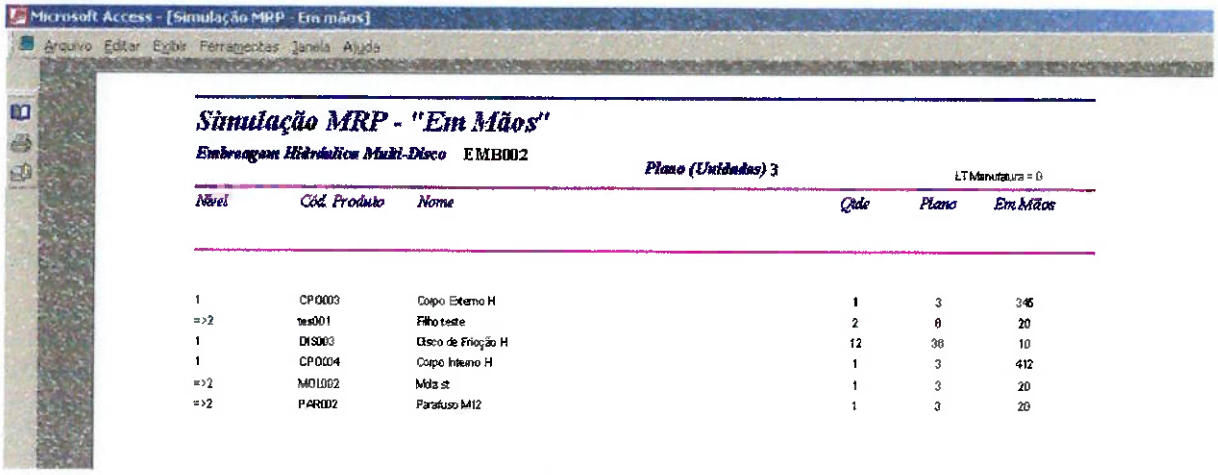
Simulação MRP - Relatório de Déficit

Embreagem Hidráulica Multi-Disco EMB002 Plano (Unidades) 3

LT Manufatura = 0

Nível	Cód. Produto	Descrição	Qtde	Plano	DPP	Déficit	Lead Time	nº PC	Saldo em Pedido	Fornecedor	Data Prometida
1	CPO003	Corpo Externo H	1	3	345	342	1	PO7175	18	SCOOBY-DOO LTDA	02/28/2003
1	DIS003	Disco de Fricção H	12	36	10	-26	1	Não tem	N/A	N/A	N/A
1	CPO004	Corpo Interno H	1	3	412	409	2	Não tem	N/A	N/A	N/A

Figura 46 – relatório: simulação MRP déficit



Microsoft Access - [Simulação MRP - Em mãos]

Arquivo Editar Exibir Ferramentas Janela Ajuda

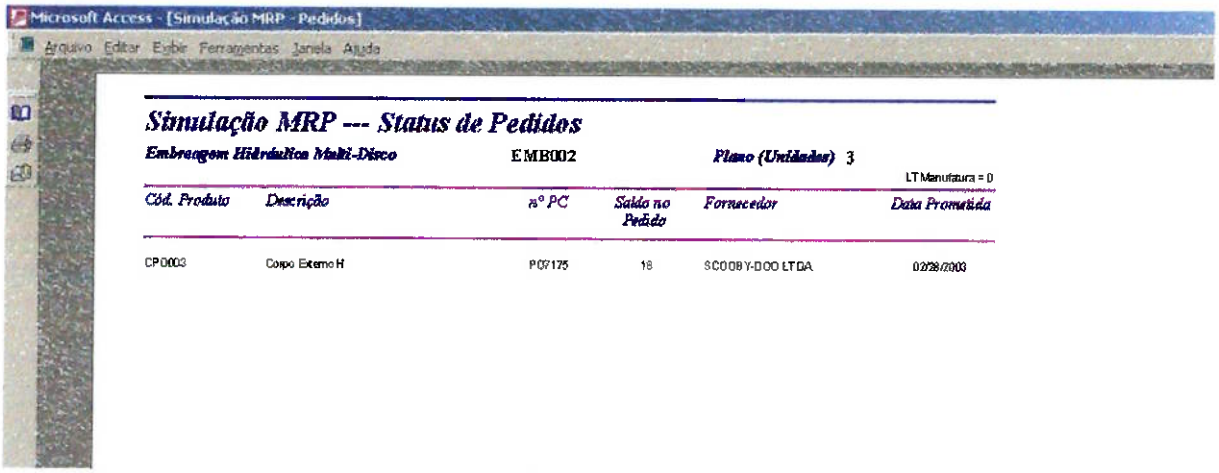
Simulação MRP - "Em Mãos"

Embreagem Hidráulica Multi-Disco EMB002 Plano (Unidades) 3

LT Manufatura = 0

Nível	Cód. Produto	Nome	Qtde	Plano	Em Mãos
1	CPO003	Corpo Externo H	1	3	345
=>2	test001	Filho teste	2	6	20
1	DIS003	Disco de Fricção H	12	36	10
1	CPO004	Corpo Interno H	1	3	412
=>2	MOL002	Mola st	1	3	20
=>2	PAR002	Parafuso M12	1	3	20

Figura 47 – relatório: simulação MRP "em mãos"



Microsoft Access - [Simulação MRP - Pedidos]

Arquivo Editar Exibir Ferramentas Janela Ajuda

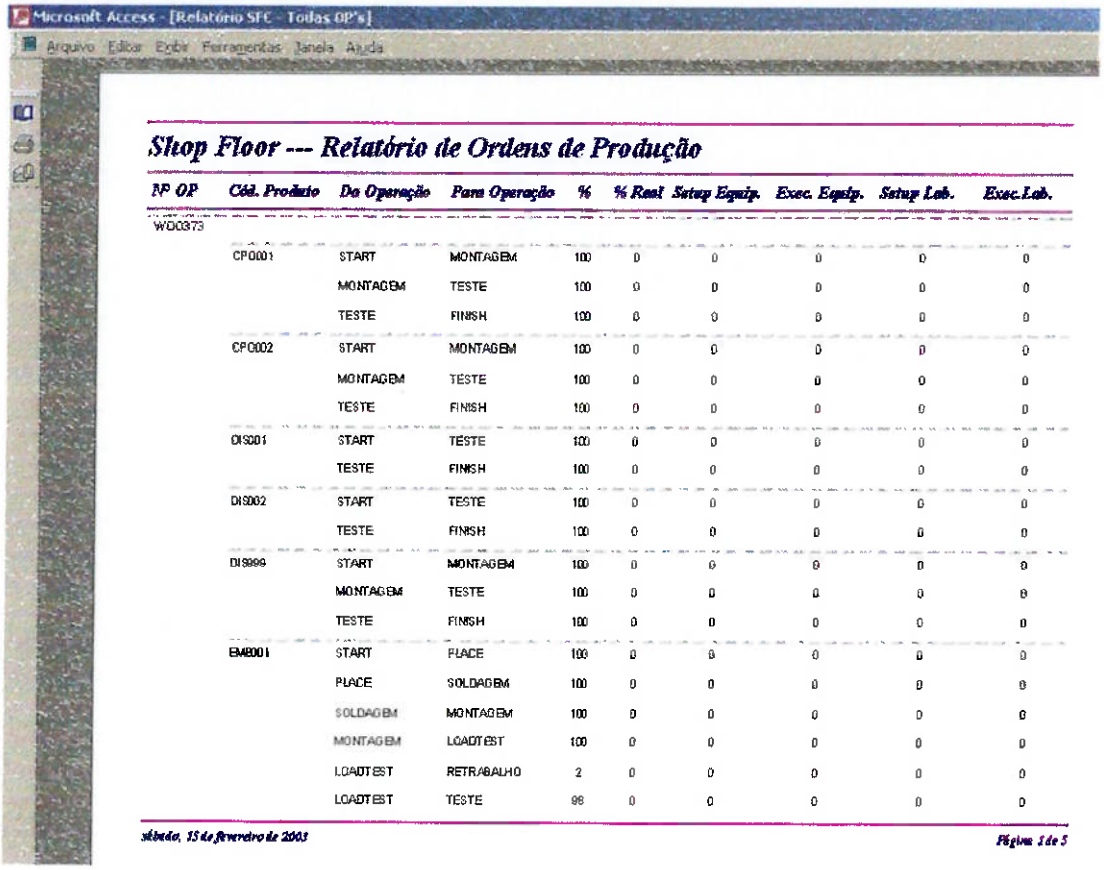
Simulação MRP --- Status de Pedidos

Embreagem Hidráulica Multi-Disco EMB002 Plano (Unidades) 3

LT Manufatura = 0

Cód. Produto	Descrição	nº PC	Saldo no Pedido	Fornecedor	Data Prometida
CPO003	Corpo Externo H	PO7175	18	SCOOBY-DOO LTDA	02/28/2003

Figura 48 – relatório: simulação MRP status de pedidos



Microsoft Access - [Relatório SFC - Todas OP's]

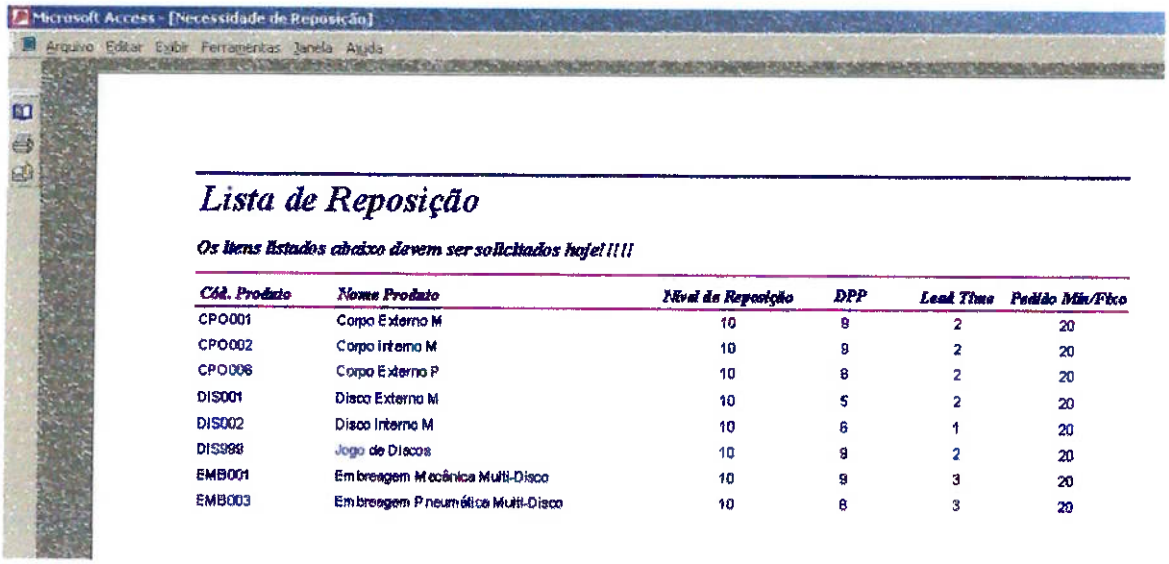
Arquivo Editar Exibir Ferramentas Janela Ajuda

Shop Floor --- Relatório de Ordens de Produção

NºOP	*Cód. Produto*	*De Operação*	*Para Operação*	*%*	*% Real*	*Setup Equip.*	*Exec. Equip.*	*Setup Lab.*	*Exec.Lab.*
WO0373									
	CPO001	START	MONTAGEM	100	0	0	0	0	0
		MONTAGEM	TESTE	100	0	0	0	0	0
		TESTE	FINISH	100	0	0	0	0	0
	CPO002	START	MONTAGEM	100	0	0	0	0	0
		MONTAGEM	TESTE	100	0	0	0	0	0
		TESTE	FINISH	100	0	0	0	0	0
	DIS001	START	TESTE	100	0	0	0	0	0
		TESTE	FINISH	100	0	0	0	0	0
	DIS002	START	TESTE	100	0	0	0	0	0
		TESTE	FINISH	100	0	0	0	0	0
	DIS999	START	MONTAGEM	100	0	0	0	0	0
		MONTAGEM	TESTE	100	0	0	0	0	0
		TESTE	FINISH	100	0	0	0	0	0
	EMB001	START	PLACE	100	0	0	0	0	0
		PLACE	SOLDAGEM	100	0	0	0	0	0
		SOLDAGEM	MONTAGEM	100	0	0	0	0	0
		MONTAGEM	LOADTEST	100	0	0	0	0	0
		LOADTEST	RETRABALHO	2	0	0	0	0	0
		LOADTEST	TESTE	98	0	0	0	0	0

sábado, 15 de fevereiro de 2003 *Página 1 de 5*

Figura 49 – relatório: SFC status das OP´s



Microsoft Access - [Necessidade de Reposição]

Arquivo Editar Exibir Ferramentas Janela Ajuda

Lista de Reposição

Os itens listados abaixo devem ser solicitados hoje!!!!

Cód. Produto	*Nome Produto*	*Nível de Reposição*	*DPP*	*Lead Time*	*Pedido Mín/Fixo*
CPO001	Corpo Externo M	10	9	2	20
CPO002	Corpo Interno M	10	9	2	20
CPO006	Corpo Externo P	10	8	2	20
DIS001	Disco Externo M	10	5	2	20
DIS002	Disco Interno M	10	6	1	20
DIS999	Jogo de Discos	10	9	2	20
EMB001	Embreagem Mecânica Multi-Disco	10	9	3	20
EMB003	Embreagem Pneumática Multi-Disco	10	8	3	20

Figura 50 – relatório: Lista de reposição

Estudo de Caso

Para analisar as funcionalidades do software foi feito um estudo de caso para uma dada empresa que fabrica acoplamentos industriais.

Feito o cadastro geral da empresa, clientes, fornecedores, transportadoras e funcionários, foram inseridos seus produtos com suas estruturas através das EP's. Então através do SFC os equipamentos e operações da fábrica foram criados.

Os formulários com alguns destes dados de entrada foram exibidos nas páginas anteriores.

Com tudo isso cadastrado no sistema, o usuário já pode fazer um planejamento através do software, no sub-menu MRP. Neste estudo os dados de entrada do MRP foram:

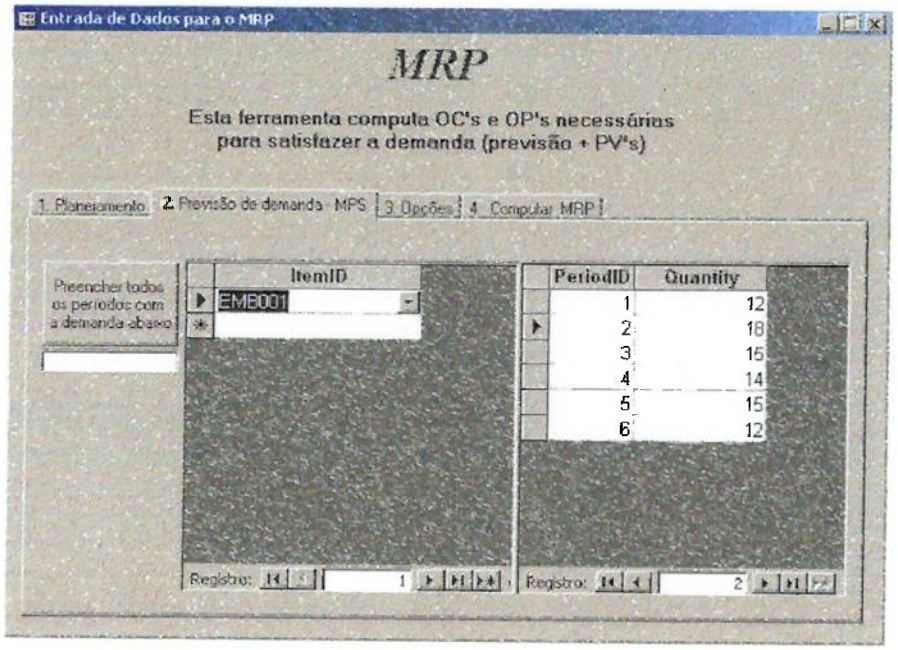


Figura 51 – entrada de dados para MRP

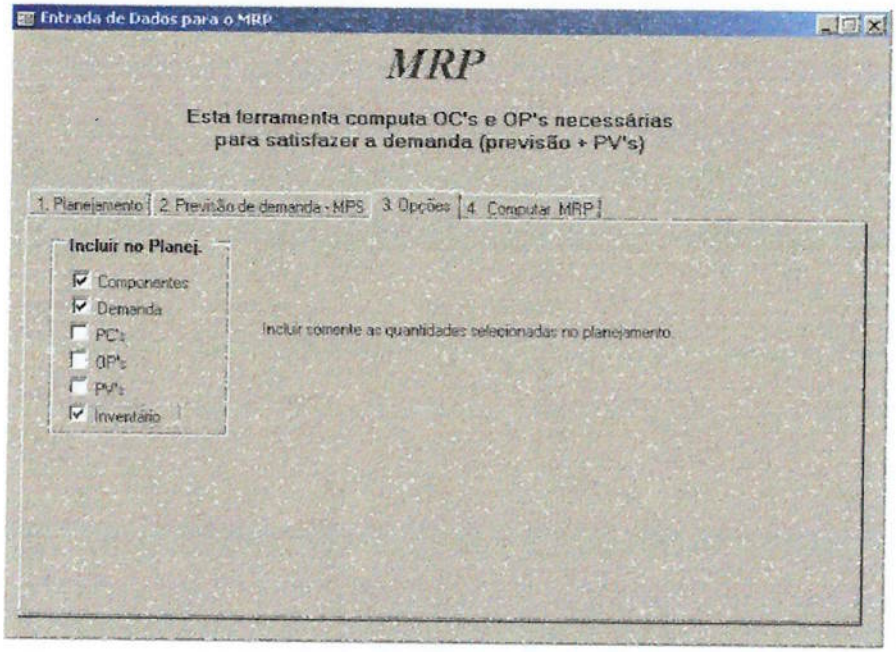


Figura 52 – entrada de dados para MRP

Com o cálculo feito, alguns relatórios são gerados com as seguintes informações:

Figura 53 – gerar saídas do MRP

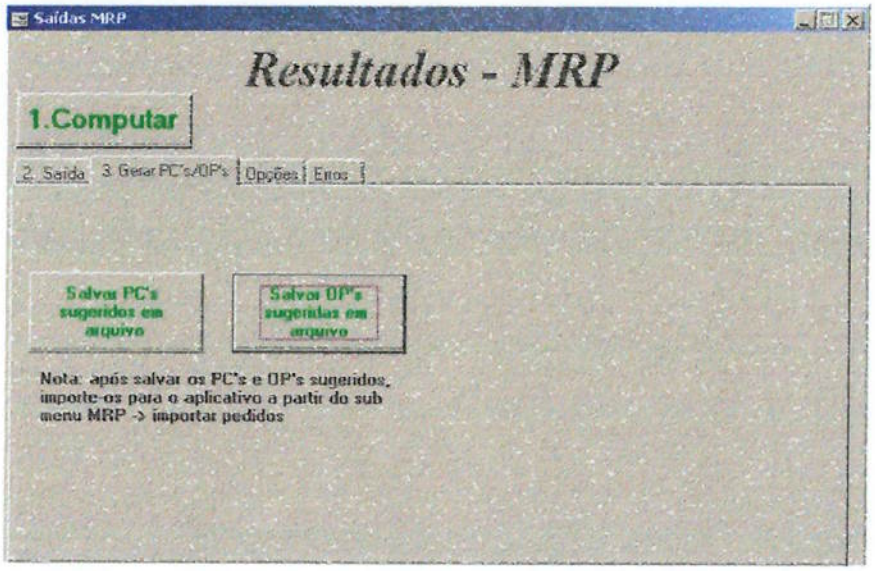


Figura 54 – avaliar saídas do MRP

Conclusão

O trabalho desenvolvido ao longo do ano atingiu a maior parte dos seus objetivos, considerando que conseguimos desenvolver um *software* de controle e gerenciamentos de estoque e produção, princípios básicos do MRP.

A versão apresentada tem uma interface com o usuário bastante amigável, sendo muito simples de desenvolver os planejamentos e obter os resultados pretendidos.

A maior dificuldade encontrada foi o desenvolvimento de um algoritmo robusto para o planejamento de capacidades, que completaria o *software* como um programa de *Manufacturing Resources Planning*, o MRP-II.

Considerando as dificuldades encontradas no decorrer do primeiro semestre, em que o projeto não se desenvolveu como era inicialmente esperado, podemos dizer que o resultado final é muito bom, sendo fruto de um grande empenho nas atividades do segundo semestre

Assim, ficamos muito satisfeitos com o resultado alcançado, pois nos permitiu aprofundar os conhecimentos em duas áreas distintas: os conceitos de Planejamento Programação e Controle da Produção, que englobam o MRP, e o desenvolvimento de um software em Visual Basic, utilizando como base o Microsoft Access.

Por fim, acreditamos ter deixado este projeto em boas condições para que seja completado com o Planejamento de Capacidades, podendo inclusive já ser aplicado para as funções mais básicas como o *Material Requirements Planning*, o MRP-I.

Bibliografia

CORRÊA, H. L.; GIANESI, I. G. N. ; CAON, M. Planejamento, programação e controle da produção – MRP II/ERP. São Paulo, Atlas, 1997.

CORRÊA, H. L.; GIANESI, I. G. N. MRP II, Just In Time e OPT: Um enfoque estratégico. São Paulo, Atlas, 1993.

IWASA, F. T.; INOSHITA, R. T.; Estudo e implementação de um sistema MRP II, São Paulo, 2001 Trabalho de formatura – Escola Politécnica. Universidade de São Paulo.

SLACK, N.; CHAMBERS, S.; HARLAND, C.; HARRISON, A.; JOHNSTON, R.; Administração da Produção – Edição Compacta, São Paulo, Atlas, 1999.

BERRY, W. L..; Production and Invetory Management, v.13, 2nd Quarter, 1972.

BERRY, W. L..; MABERT, V. A.; Internal Journal of Operations and Production Management, v.12, nº 6, 1992.

GROOVER, M. P.; Automation, Production Systems and Computer-aided Manufacturing, Englewood Cliffs, Prentice-Hall, 1980.

PIRES, L. G. R.; Estudo do Sistema de Planejamento dos Recursos de Manufatura (MRP II) Atraves de suas Principais Variáveis, São Paulo, 1995.

HIGGINS, P.; Manufacturing Planning and Control Beyond MRP II, London, Chapman & Hall, 1996.

WIGHT, O. W.; The Executive's Guide to Succesful MRP II, New York, Prentice-Hall, 1993.

www.gianesi.com.br

www.unisoma.com/mrp.htm

Anexos

Seguem, anexos, os principais códigos fonte do programa.

CRIAR ESTRUTURA DE PRODUTO

```
Option Compare Database
Public strTmp As String
Public strdelete As String
Option Explicit

Private Sub CopyKids_Click()
Dim dbsCurrent As DAO.Database
Dim strParentID As String

If Not IsNull(ParentID) Then
    Set dbsCurrent = CurrentDB()

    strParentID = Forms!BOM!ParentID

    DoCmd.Hourglass True

     dbsCurrent.Execute "DELETE FROM Children_tmp;"

    dbsCurrent.Execute "INSERT INTO [Children_tmp]" _
    & "SELECT W.[Gen_Details],W.[ParentID],W.[LineNumber],W.[ProductID]," _
    & "W.[QtyInParent],W.[VendorPartNumber],W.[ReferenceDesignations],W.[Notes],W.[Details]" _
    & "FROM [Where_Used] W " _
    & "WHERE [ParentID]=""" & ReplaceString(strParentID, Chr(34), Chr(34) & Chr(34)) & """ " _
    & "ORDER BY [ProductID] ASC;"

    DoCmd.Hourglass False
End If

End Sub

Private Sub Form_AfterDelConfirm(Status As Integer)
    Select Case Status
        Case acDeleteOK
            MsgBox "O processo ocorreu normalmente."
        Case acDeleteCancel
            MsgBox "O usuário cancelou o processo."
        Case acDeleteUserCancel
            MsgBox "O usuário cancelou o processo."
    End Select
End Sub

Private Sub Form_BeforeDelConfirm(Cancel As Integer, Response As Integer)
    Response = acDataErrContinue
    If MsgBox("Apagar este Registro?", vbOKCancel) = vbCancel Then
        Cancel = True
        Exit Sub
    End If
    Dim dbsCurrent As DAO.Database
    Dim strParentID As String
    Dim wrkJet As DAO.Workspace
     Set dbsCurrent = CurrentDB()

    On Error Resume Next

    strParentID = Strdelete
    DoCmd.Hourglass True
    dbsCurrent.Execute "DELETE * FROM [Where_Used] WHERE ([Where_Used].[ParentID] = """ &
ReplaceString(strParentID, Chr(34), Chr(34) & Chr(34)) & """);"
    DoCmd.Hourglass False

End Sub

Private Sub Form_BeforeUpdate(Cancel As Integer)
Dim strOld As String
Dim strNew As String

If Not ((Me![Status] = "Bloqueado") Or (Me![Status].OldValue = "Bloqueado")) Then
    MsgBox ("É necessária autorização para alterar uma Estrutura de Produto Ativa ou Obsoleta!")
    DoCmd.CancelEvent
    Me.Undo
    Exit Sub
End If

End Sub

Private Sub Form_Delete(Cancel As Integer)
strTmp = Me!ParentID
strdelete = strTmp
End Sub
```

```
Private Sub Form_LostFocus()
On Error Resume Next
strTmp = Me!ParentID
End Sub

Private Sub ItemFind_Click()
If Not IsNull([Combo55]) Then
    Me.autoid.SetFocus
    DoCmd.FindRecord [Combo55]
    Combo55.SetFocus
Else
    MsgBox "Por favor selecione um produto."
End If
End Sub

Private Sub ParentID_BeforeUpdate(Cancel As Integer)
On Error GoTo quit
If Not IsNull(Forms![BOM].ParentID) Then
Dim strValue As String
strValue = DLookup("[ProductName]", "Products", "[Products].[ProductID]= Forms![BOM].ParentID")
End If
MsgBox "Este código de Pai é inválido, provavelmente porque está duplicado (ie já foi registrado).
Procure por ele através da função 'Localizar' (Binóculo)."
DoCmd.CancelEvent
Me.Undo
Forms![BOM]![ParentID].SetFocus
Exit Sub
quit:
End Sub

Private Sub PasteChildren_Click()
Dim dbsCurrent As DAO.Database
Dim strParentID As String

If Not IsNull([Forms]![BOM]![ParentID]) Then
    Set dbsCurrent = CurrentDB()
    strParentID = Forms!BOM!ParentID
    DoCmd.Hourglass True

    dbsCurrent.Execute "INSERT INTO [Where_Used]"
    & "SELECT [Children_tmp].[Gen_Details],""" & ReplaceString(strParentID, Chr(34), Chr(34) &
Chr(34)) & """ as [ParentID],[Children_tmp].[LineNumber],[Children_tmp].[ProductID]," _&
"[Children_tmp].[QtyInParent],[Children_tmp].[VendorPartNumber],[Children_tmp].[ReferenceDesignations]
,[Children_tmp].[Notes],[Children_tmp].[Details]" _
    & "FROM [Children_tmp];"
    dbsCurrent.Execute "DELETE * FROM Where_Used WHERE ([Where_Used].[ProductID] is NULL);"
    On Error Resume Next
    DoCmd.Save
    On Error GoTo 0

    Dim iRecordNumber As Long
     iRecordNumber = Me.CurrentRecord
     Me.Refresh
     On Error Resume Next
     DoCmd.GoToRecord , , acGoTo, iRecordNumber
     On Error GoTo 0
    DoCmd.Hourglass False
End If
End Sub

Private Sub Sort_Click()
On Error GoTo Err_Sort_Click
    Me![Where_Used subform2].Requery
Exit_Sort_Click:
    Exit Sub
Err_Sort_Click:
    MsgBox Err.Description
    Resume Exit_Sort_Click
End Sub


****************************************
SELECT Products.*, Products.ProductID, Products.ProductName, Products.Manufacturer, Products.Released,
Products.ControlCharacters, Products.PhysicalCount, Products.DynamicCount, Products.UnitsOfMeasure,
Products.Released
FROM Products
WHERE (((Products.Manufacturer) Like "M*"))
ORDER BY Products.ProductID;
****************************************
```

RELATÓRIO DE ATENDIMENTO AO CLIENTE

```
Option Compare Database
Option Explicit
Private Sub Form_Load()
    DoCmd.GoToRecord , , acNewRec
    CboCustomer.SetFocus

End Sub

Private Sub email_RMA_Click()

Const EmailColumn = 1
Const Quote = """"
Const MsgErrorText = "Você não selecionou um destinatário para enviar a mensagem."

Dim dbCurrent As DAO.Database

Dim strRMANumber As String
Dim FirstInfo As String
Dim EmailTo As String
Dim counter As Integer

DoCmd.Hourglass True

strRMANumber = Me!RMANumber & " - " & Me!CustomerName
FirstInfo = "CSTMR:" & Me!CustomerName & "|NOTES:" & Me!TxtEmailNotes & "|ETA:" & Me!Repair_Date &
"|ITEM:" & Me!ProductName
FirstInfo = FirstInfo & "|" & Me!RMANumber & "|VGI:" & Me!FieldContact

EmailTo = ""
With Me![LstEmailNames]
    For counter = 0 To .ListCount
        If (.Selected(counter) = True) Then
            If EmailTo = "" Then
                EmailTo = .Column(EmailColumn, counter)
            Else
                EmailTo = EmailTo & ", " & .Column(EmailColumn, counter)
            End If
        End If
    Next
End With

If EmailTo = "" Then
    MsgBox MsgErrorText, vbOKOnly + vbExclamation, "Mensagem Não Enviada!"
Else
    DoCmd.SendObject acSendNoObject, , acFormatTXT, EmailTo, , , strRMANumber, FirstInfo, True
End If

DoCmd.Hourglass False

End Sub

Private Sub Form_Current()

    Dim counter As Integer
    For counter = 0 To Me![LstEmailNames].ListCount

        Me![LstEmailNames].Selected(counter) = False

    Next
End Sub
```

CADASTRO DE CLIENTES

```
Option Compare Database
Option Explicit

Private Sub CustomersShipTo_subform_Enter()
Me.CustomersShipTo_subform.Requery
End Sub

Private Sub Form_AfterInsert()
DoCmd.SetWarnings False
DoCmd.OpenQuery "CustomersExtraShipTo_Append"
DoCmd.SetWarnings True
End Sub

Private Sub Form_AfterUpdate()
On Error Resume Next
DoCmd.SetWarnings False
DoCmd.OpenQuery "CustomersExtraShipTo_Update"
DoCmd.SetWarnings True
End Sub

Private Sub Form_Load()
DoCmd.GoToRecord , , acLast
End Sub

Private Sub ItemFind_Click()
If Not IsNull([ItemLookUP]) Then
    Me.CustomerID.SetFocus
    DoCmd.FindRecord ([ItemLookUP])
    ItemLookUP.SetFocus
Else
    MsgBox "Por favor escolha um cliente para ser exibido."
End If
End Sub

Private Sub NovoRegistro_Click()
On Error GoTo Err_NovoRegistro_Click

    DoCmd.GoToRecord , , acNewRec

Exit_NovoRegistro_Click:
    Exit Sub
Err_NovoRegistro_Click:
    MsgBox Err.Description
    Resume Exit_NovoRegistro_Click
End Sub

Private Sub SalvarRegistro_Click()
On Error GoTo Err_SalvarRegistro_Click
    DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, , acMenuVer70
Exit_SalvarRegistro_Click:
    Exit Sub
Err_SalvarRegistro_Click:
    MsgBox Err.Description
    Resume Exit_SalvarRegistro_Click
End Sub

Private Sub Sair_Click()
On Error GoTo Err_Sair_Click

    DoCmd.Close
    DoCmd.OpenForm "Menu Inicial"

Exit_Sair_Click:
    Exit Sub
Err_Sair_Click:
    MsgBox Err.Description
    Resume Exit_Sair_Click

End Sub
```

CADASTRO DE OPERAÇÕES

```
Option Compare Database
Option Explicit

Private Sub EnableLabourIDRunOnly_Click()
    LabourIDRunOnly.Enabled = EnableLabourIDRunOnly
    If Not EnableLabourIDRunOnly Then
        LabourIDRunOnly = Null
    End If
End Sub

Private Sub Form_BeforeUpdate(Cancel As Integer)
    If IsNull(EquipmentID) And IsNull(LabourID) Then
        MsgBox "Os campos 'Grupo de Equipamento' e 'Groupo de Operação' não podem estar ambos vazios."
        Cancel = True
        EquipmentID.SetFocus
    End If
End Sub

Private Sub Form_Current()
    EnableLabourIDRunOnly = Not IsNull(LabourIDRunOnly)
    LabourIDRunOnly.Enabled = EnableLabourIDRunOnly
End Sub

Private Sub OperationName_BeforeUpdate(Cancel As Integer)
    If Not IsNull(Me.OperationName) And Nz(Me.OperationName <> Me.OperationName.OldValue, True) Then
        If dCount("[OperationName]", "Operations", "[OperationName] = " & Quote(Me.OperationName) & "
And isnull([ProductID])") > 0 Then
            MsgBox "Não é permitido um nome de Operação duplicado."
            Cancel = True
            OperationName.SelStart = 0
            OperationName.SelLength = Len(OperationName)
        End If
    End If
End Sub

Private Sub Fechar_Click()
On Error GoTo Err_Fechar_Click

    DoCmd.Close , , acSavePrompt
    DoCmd.OpenForm "GFCConfig"

Exit_Fechar_Click:
    Exit Sub

Err_Fechar_Click:
    MsgBox Err.Description
    Resume Exit_Fechar_Click
End Sub

Private Sub SalvaRegistro_Click()
On Error GoTo Err_SalvaRegistro_Click


    DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, , acMenuVer70

Exit_SalvaRegistro_Click:
    Exit Sub

Err_SalvaRegistro_Click:
    MsgBox Err.Description
    Resume Exit_SalvaRegistro_Click

End Sub
```

EXIBIR ESTRUTURA DE PRODUTO

```
Option Compare Database
Option Explicit

Dim DBObject As New EB.EBdefinitionsClass
Dim XObject As New EB.EBexplodeClass

Private Sub Command12_Click()
    DoCmd.OpenReport "Exploded_BOM_Picklist3", acViewPreview
End Sub

Private Sub cmdOption1_Click()
    DoCmd.OpenForm "ItemMaster"
End Sub

Private Sub cmdOption2_Click()
    DoCmd.OpenQuery "WhereUsed_InvalidLines"
End Sub

Private Sub cmdShowInvalidProductID_Click()
    If dCount("*", "WhereUsed_MissingProductID(ALL)") + dCount("*", "WhereUsed_InvalidLines") > 0 Then
        DoCmd.Close acQuery, "WhereUsed_MissingProductID(ALL)"
        DoCmd.Close acQuery, "WhereUsed_InvalidLines"
        DoCmd.OpenQuery "WhereUsed_MissingProductID(ALL)"
        DoCmd.OpenQuery "WhereUsed_InvalidLines"
    Else
        MsgBox "Não há produtos inválidos."
    End If
End Sub



Private Sub Command16_Click()
Dim dbs As Database
Dim rst As Recordset
Dim levelCurrent As Double
Dim levelPrevious As Double
Dim levelLoop As Double
Dim parentQuantity As Double
Dim manufactured As String

Set dbs = CurrentDB

DoCmd.SetWarnings False
dbs.Execute "DELETE * FROM Standard_Costs;"
DoCmd.OpenQuery "Standard_Costs_AppendTable", acNormal, acEdit

Set rst = dbs.OpenRecordset("SELECT Row, [Level], QtyInParent, StandardUnitPrice, lineCost FROM
Standard_Costs ORDER BY row;")

Do While Not (rst.EOF)
    levelCurrent = rst![Level]
    parentQuantity = rst!QtyInParent
    rst.MoveNext
    If Not (rst.EOF) Then
        levelPrevious = levelCurrent
        levelCurrent = rst![Level]
        If (levelCurrent > levelPrevious) Then
            levelLoop = levelCurrent
            Set rst = Filter_Recordset(rst, parentQuantity, levelLoop)
            If (Not (rst.EOF)) Then
                levelCurrent = rst![Level]
            Else
                Exit Do
            End If
        End If
    Else
        Exit Do
    End If
Loop

dbs.Execute "UPDATE Standard_Costs " _
        & "SET UnitCost = StandardUnitPrice, lineCost = StandardUnitPrice * QtyInParent, " _
        & "OverheadLineCost = OverheadCost * QtyInParent, InputLineCost = NULL " _
        & "WHERE Manufacturer LIKE " & """P*""" & ";"
dbs.Execute "UPDATE Standard_Costs " _
        & "SET UnitCost = OverheadCost, lineCost = OverheadCost * QtyInParent, " _
        & "OverheadLineCost = OverheadCost * QtyInParent, InputLineCost = StandardUnitPrice " _
        & "WHERE Manufacturer LIKE " & """M*""" & ";"
```

```
rst.Close
Set dbs = Nothing

On Error Resume Next
DoCmd.Minimize
DoCmd.OpenReport "Exploded_BOM_stdcost3", acViewPreview
On Error GoTo 0
DoCmd.SetWarnings True

End Sub
Private Sub Command18_Click()

Dim dbs As Database
Dim rst As Recordset
Dim levelCurrent As Double
Dim levelPrevious As Double
Dim levelLoop As Double
Dim parentQuantity As Double

DoCmd.SetWarnings False
DoCmd.OpenQuery "BOM MakeTable", acNormal, acEdit

Set dbs = CurrentDB
Set rst = dbs.OpenRecordset("BOM", dbOpenTable)

Do While Not (rst.EOF)
    levelCurrent = rst!Level
    parentQuantity = rst!QtyInParent
    rst.MoveNext
    If Not (rst.EOF) Then
        levelPrevious = levelCurrent
        levelCurrent = rst!Level
        If (levelCurrent > levelPrevious) Then
            levelLoop = levelCurrent
            Set rst = Filter_Recordset_MultiLevelBOM(rst, parentQuantity, levelLoop)
            If (Not (rst.EOF)) Then
                levelCurrent = rst!Level
            Else
                Exit Do
            End If
        End If
    Else
        Exit Do
    End If
Loop

On Error Resume Next
DoCmd.OpenReport "BOM Report", acViewPreview
On Error GoTo 0
DoCmd.SetWarnings True

End Sub

Private Sub Command18_Click()

Dim dbs As Database
Dim rst As Recordset
Dim levelCurrent As Double
Dim levelPrevious As Double
Dim levelLoop As Double
Dim parentQuantity As Double

DoCmd.SetWarnings False
DoCmd.OpenQuery "Multilevel BOM Query", acNormal, acEdit

Set dbs = CurrentDB
Set rst = dbs.OpenRecordset("Multilevel BOM", dbOpenTable)

Do While Not (rst.EOF)
    levelCurrent = rst!Level
    parentQuantity = rst!QtyInParent
    rst.MoveNext
    If Not (rst.EOF) Then
        levelPrevious = levelCurrent
        levelCurrent = rst!Level
        If (levelCurrent > levelPrevious) Then
            levelLoop = levelCurrent
            Set rst = Filter_Recordset_MultiLevelBOM(rst, parentQuantity, levelLoop)
            If (Not (rst.EOF)) Then
                levelCurrent = rst!Level
            Else
```

```
                Exit Do
            End If
        End If
    Else
        Exit Do
    End If
Loop

On Error Resume Next
DoCmd.Minimize
DoCmd.OpenReport "Exploded_BOM1", acViewPreview
On Error GoTo 0
DoCmd.SetWarnings True

End Sub



Private Sub Explode_Click()
On Error GoTo Err_Explode_Click
Dim strValue As String
Dim mLevel As Integer
Dim dbsCurrent As DAO.Database
Set dbsCurrent = CurrentDB()

If IsNull(Forms!DMR_Explosion!ProductID1) Then
    MsgBox "Por favor selecione um produto antes."
    Exit Sub
Else
    strValue = Forms!DMR_Explosion!ProductID1
End If

If IsNull(Forms!DMR_Explosion!maxLevel) Then
    MsgBox "Por favor entre um nível."
    Exit Sub
Else
    mLevel = Forms!DMR_Explosion!maxLevel
End If
DoCmd.Hourglass True
XObject.mlevels = mLevel
XObject.ParentID = strValue
XObject.DatabaseCurrent = dbsCurrent

XObject.Explode
DoCmd.SetWarnings False
DoCmd.OpenQuery "Exploded_BOM Query1 New"
DoCmd.OpenQuery "Exploded_BOM Query"
DoCmd.SetWarnings True
If dCount("*", "WhereUsed_MissingProductID(ALL)") + dCount("*", "WhereUsed_InvalidLines") > 0 Then
    MsgBox "A tabela 'Usado em...', que armazena as Estruturas de Produto, contém alguns Códigos de
Produto inválidos." & vbCrLf & "Por favor corrija este dados antes de prosseguir, ou as EPs exibidas
podem conter erros." & vbCrLf & vbCrLf & "Para maiores informações sobre como corrigi-los, use a
Verificação de Qualidade de Dados a partir do Menu Inicial -> Outros.", vbExclamation, "DADOS
INVÁLIDOS!"
End If

Exit_Explode_Click:
    DoCmd.Hourglass False
    Exit Sub
Err_Explode_Click:
    If (Err.Number = 3021) Then
        MsgBox "Erro : " & Err.Number & ", " & Err.Description & ".  A causa provável é: o item
selecionado não possui filhos."
    Else
        MsgBox "Erro : " & Err.Number & ",  " & Err.Description
    End If
    Resume Exit_Explode_Click
End Sub

Private Sub Form_Load()
Dim dbsCurrent As DAO.Database
Dim strValue As String

    Set dbsCurrent = CurrentDB()
    strValue = DLookup("[Userdefinable 3]", "My Company Information")
    strValue = strValue & "DMR_fe.mdb"
    DBObject.DatabaseCurrent = dbsCurrent
    DBObject.DatabaseName = strValue
    DBObject.SetDatabase

End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
    Set DBObject = Nothing
    Set XObject = Nothing
End Sub

Function Filter_Recordset(temp_rst As Recordset, aQuantity As Double, aLevelLoop As Double) As
Recordset

Dim levelCurrent As Double
Dim levelPrevious As Double
Dim levelLoop As Double
Dim parentQuantity As Double

levelCurrent = temp_rst!Level
Do While ((levelCurrent >= aLevelLoop) And Not (temp_rst.EOF))
    With temp_rst
    .Edit
    !QtyInParent = !QtyInParent * aQuantity
    !linecost = !QtyInParent * !StandardUnitPrice
    .update
    parentQuantity = temp_rst!QtyInParent
    .MoveNext
    End With
    If Not (temp_rst.EOF) Then
        levelPrevious = levelCurrent
        levelCurrent = temp_rst!Level
        If (levelCurrent > levelPrevious) Then
            levelLoop = levelCurrent
            Set temp_rst = Filter_Recordset(temp_rst, parentQuantity, levelLoop)
            If Not (temp_rst.EOF) Then
                levelCurrent = temp_rst!Level
            Else
                Exit Do
            End If
        End If
    Else
        Exit Do
    End If
Loop
Set Filter_Recordset = temp_rst
End Function

Function Filter_Recordset_CurrentCosts(temp_rst As Recordset, aQuantity As Double, aLevelLoop As
Double) As Recordset

Dim levelCurrent As Double
Dim levelPrevious As Double
Dim levelLoop As Double
Dim parentQuantity As Double

levelCurrent = temp_rst!Level
Do While ((levelCurrent >= aLevelLoop) And Not (temp_rst.EOF))
    With temp_rst
    .Edit
    !QtyInParent = !QtyInParent * aQuantity
    !linecost = !QtyInParent * !UnitPrice
    .update
    parentQuantity = temp_rst!QtyInParent
    .MoveNext
    End With
    If Not (temp_rst.EOF) Then
        levelPrevious = levelCurrent
        levelCurrent = temp_rst!Level
        If (levelCurrent > levelPrevious) Then
            levelLoop = levelCurrent
            Set temp_rst = Filter_Recordset_CurrentCosts(temp_rst, parentQuantity, levelLoop)
            If (Not (temp_rst.EOF)) Then
                levelCurrent = temp_rst!Level
            Else
                Exit Do
            End If
        End If
    Else
        Exit Do
    End If
Loop
Set Filter_Recordset_CurrentCosts = temp_rst
End Function
```

```
Function Filter_Recordset_AverageCosts(temp_rst As Recordset, aQuantity As Double, aLevelLoop As
Double) As Recordset

Dim levelCurrent As Double
Dim levelPrevious As Double
Dim levelLoop As Double
Dim parentQuantity As Double

levelCurrent = temp_rst!Level
Do While ((levelCurrent >= aLevelLoop) And Not (temp_rst.EOF))
    With temp_rst
    .Edit
    !QtyInParent = !QtyInParent * aQuantity
    !linecost = !QtyInParent * !NewAvg
    .update
    parentQuantity = temp_rst!QtyInParent
    .MoveNext
    End With
    If Not (temp_rst.EOF) Then
        levelPrevious = levelCurrent
        levelCurrent = temp_rst!Level
        If (levelCurrent > levelPrevious) Then
            levelLoop = levelCurrent
            If temp_rst.EOF Then
                Stop
            End If
            Set temp_rst = Filter_Recordset_AverageCosts(temp_rst, parentQuantity, levelLoop)
            If (Not (temp_rst.EOF)) Then
                levelCurrent = temp_rst!Level
            Else
                Exit Do
            End If
        End If
    Else
        Exit Do
    End If
Loop
Set Filter_Recordset_AverageCosts = temp_rst
End Function

Function Filter_Recordset_MultiLevelBOM(temp_rst As Recordset, aQuantity As Double, aLevelLoop As
Double) As Recordset

Dim levelCurrent As Double
Dim levelPrevious As Double
Dim levelLoop As Double
Dim parentQuantity As Double

levelCurrent = temp_rst!Level

Do While ((levelCurrent >= aLevelLoop) And Not (temp_rst.EOF))
    With temp_rst
    .Edit
    !QtyInParent = !QtyInParent * aQuantity
    .update
    parentQuantity = temp_rst!QtyInParent
    .MoveNext
    End With
    If Not (temp_rst.EOF) Then
        levelPrevious = levelCurrent
        levelCurrent = temp_rst!Level
        If (levelCurrent > levelPrevious) Then
            levelLoop = levelCurrent
            Set temp_rst = Filter_Recordset_MultiLevelBOM(temp_rst, parentQuantity, levelLoop)
            If (Not (temp_rst.EOF)) Then
                levelCurrent = temp_rst!Level
            Else
                Exit Do
            End If
        End If
    Else
        Exit Do
    End If
Loop
Set Filter_Recordset_MultiLevelBOM = temp_rst
End Function

Private Sub Form_Close()
    DoCmd.OpenForm "Menu Inicial"
End Sub
```

CADASTRO FUNCIONÁRIOS

```
Option Compare Database
Option Explicit

Private Sub Form_Load()
    If Me.OpenArgs = "GotoNew" And Not IsNull([EmployeeID]) Then
        DoCmd.DoMenuItem acFormBar, 3, 0, , acMenuVer70
    End If
End Sub

Private Sub ItemFind_Click()
If Not IsNull([ItemLookUP]) Then
    Me.EmployeeID.SetFocus
    DoCmd.FindRecord ([ItemLookUP])
    ItemLookUP.SetFocus
Else
    MsgBox "Por favor selecione um funcionário para exibir."
End If
End Sub
Private Sub NovoRegistro_Click()
On Error GoTo Err_NovoRegistro_Click


    DoCmd.GoToRecord , , acNewRec

Exit_NovoRegistro_Click:
    Exit Sub

Err_NovoRegistro_Click:
    MsgBox Err.Description
    Resume Exit_NovoRegistro_Click

End Sub
Private Sub SalvarRegistro_Click()
On Error GoTo Err_SalvarRegistro_Click


    DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, , acMenuVer70

Exit_SalvarRegistro_Click:
    Exit Sub

Err_SalvarRegistro_Click:
    MsgBox Err.Description
    Resume Exit_SalvarRegistro_Click

End Sub
Private Sub Sair_Click()
On Error GoTo Err_Sair_Click


    DoCmd.Close
    DoCmd.OpenForm "Menu Inicial"

Exit_Sair_Click:
    Exit Sub

Err_Sair_Click:
    MsgBox Err.Description
    Resume Exit_Sair_Click

End Sub
```

HISTÓRICO DE TRANSAÇÕES

```
Option Compare Database
Option Explicit

Private Type SortItem
    strExternalName As String
    strInternalName As String
End Type

Private Const conReport = "Inventory_Transactions_History_Report"
Private Const conTable = "[Inventory Transactions]"
Private Const conQuery_Base = "PO_Receiving"
Private Const conQuery_Filter = conQuery_Base & "_Filter"
Private Const conQuery_GroupBy = conQuery_Base & "_GroupBy"

Private Const conSpecific = 0
Private Const conAll = 1
Private Const conBlank = 2
Private Const conNotBlank = 3
Private Const conMaxSortListSize = 4

Private strSortList(conMaxSortListSize) As SortItem 'Store the sort order list
Private sqlProductID_Condition As String
Private sqlOrderTransfer_Condition As String
Private sqlInvoiceNum_Condition As String
Private sqlPPNum_Condition As String
Private sqlReceivedDate_Condition As String
Private sqlSortBy_SortOrder As String
Private sqlGroupLeader_Select As String
Private sqlGroupBy_Condition As String
Private sqlGroup_GroupBy As String
Private sqlGroupID_Select As String
Private sqlFilter_WHERE As String

Private sqlFullFilterStmt As String
Private sqlFullGroupByStmt As String


Private Sub Execute_Click()
    Dim dbsCurrent As DAO.Database
    Dim strTemp As String
    Dim strValue As String
    Dim rstRpt As DAO.Recordset
    Dim strSQL As String
    Dim sqlGROUP_BY As String
    Dim sqlORDER_BY As String
    Dim intLoop As Integer

    Set dbsCurrent = CurrentDB()
    DoCmd.Hourglass True
    DoCmd.SetWarnings False

    On Error Resume Next
    DoCmd.Close acReport, conReport, acSavePrompt
    On Error GoTo 0

    Call PullFullSQL

    dbsCurrent.Execute "DELETE * FROM [ReportDetailOrganizer] " _
                     & "WHERE (ReportName = """ & conReport & """);"

    If sqlFilter_WHERE <> "" Then sqlFilter_WHERE = " And " & sqlFilter_WHERE

    sqlGROUP_BY = ""
    If (sqlGroup_GroupBy <> "") Then
      sqlGROUP_BY = " GROUP BY TransactionID, ReportName, " & sqlGroup_GroupBy
    End If

    If (sqlSortBy_SortOrder <> "") Then
      sqlORDER_BY = " ORDER BY " & sqlSortBy_SortOrder
    Else
      sqlORDER_BY = ""
    End If

    strSQL = "INSERT INTO ReportDetailOrganizer " _
"SELECT TransactionID,""" & conReport & """ AS ReportName " _
"FROM [Inventory Transactions] " _
"WHERE (([Inventory Transactions].CustomerID) <> 0) " _
"AND (UnitsShipped <> 0)" _
sqlFilter_WHERE & " " _
```

```
& sqlORDER_BY & ";"

    dbsCurrent.Execute strSQL

    strSQL = "SELECT [ReportDetailOrganizer].TransactionID,OrderBy,GroupBy,TransactionDate," _
& "ProductID,PP,InvoiceNumber,GroupByField,GroupByTitle,MiscNum1,PurchaseOrderID,SalesOrderID " _
& "FROM [ReportDetailOrganizer] INNER JOIN [Inventory Transactions] ON " _
& "[ReportDetailOrganizer].TransactionID = [Inventory Transactions].TransactionID " _
& "WHERE (ReportName=""" & conReport & """) " _
& sqlORDER_BY & ";"

    Set rstRprt = dbsCurrent.OpenRecordset(strSQL)

If (rstRprt.RecordCount > 0) Then
    Dim strGroupByField As String
    Dim strOldGroupBy As String
    Dim strNewGroupBy As String
    Dim strGroupBy As String
    Dim strGroupByTitle As String
    Dim lngOrderBy As Long
    Dim NewestTransactionDate As String
    Dim NewTransactionDate As String

    strGroupByField = strSortList(0).strInternalName
    Select Case strGroupByField
      Case "ProductID"
        strGroupByTitle = ""
      Case "InvoiceNumber"
        strGroupByTitle = "InvoiceNum"
      Case "PP"
        strGroupByTitle = "PPNum"
      Case "TransactionDate"
        strGroupByTitle = ""
      Case "PurchaseOrderID,SalesOrderID"
        strGroupByTitle = ""
    End Select


    strOldGroupBy = ""
    strNewGroupBy = ""
    strGroupBy = ""
    lngOrderBy = 0

    With rstRprt

      .MoveFirst
      If (sqlORDER_BY <> "") Then
        Do Until .EOF

          Select Case strGroupByField
            Case "ProductID"
              strGroupBy = !ProductID
            Case "InvoiceNumber"
              strGroupBy = iif(IsNull(!InvoiceNumber), "", !InvoiceNumber)
            Case "PP"
              strGroupBy = iif(IsNull(!PP), "", !PP)
            Case "TransactionDate"
              strGroupBy = !TransactionDate
            Case "PurchaseOrderID,SalesOrderID"
              If (!PurchaseOrderID Like "PO*") Then
                strGroupBy = !PurchaseOrderID
              Else
                strGroupBy = !SalesOrderID
              End If
          End Select

          .Edit
          !GroupBy = ReplaceString(strGroupBy, Chr(34), Chr(34) & Chr(34))
          !OrderBy = lngOrderBy
          !GroupByTitle = strGroupByTitle
          !GroupByField = strGroupByField
          .update
          lngOrderBy = lngOrderBy + 1

        .MoveNext
      Loop
    End If
  End With

End If
  rstRprt.Close
```

```
    DoCmd.Hourglass False
    DoCmd.SetWarnings True

On Error GoTo Execute_Click_Err

DoCmd.Minimize
DoCmd.OpenReport conReport, acPreview, "", ""


Execute_Click_Exit:
    Exit Sub

Execute_Click_Err:
    MsgBox Error$
    Resume Execute_Click_Exit
    On Error GoTo 0

    Set dbsCurrent = Nothing
End Sub

Private Sub Form_Close()
    DoCmd.SetWarnings False
    DoCmd.Close acReport, "Inventory_Transactions_History_Report", acSavePrompt
    DoCmd.SetWarnings True
    DoCmd.OpenForm "Menu Inicial"

End Sub

Private Sub Form_Load()
    ProductID.ColumnWidths = str(ProductID.Width) & ";2"""

    strSortList(0).strExternalName = "TransactionDate"
    strSortList(0).strInternalName = "TransactionDate"
    strSortList(1).strExternalName = "ItemID"
    strSortList(1).strInternalName = "ProductID"
    strSortList(2).strExternalName = "Order/TransferID"
    strSortList(2).strInternalName = "PurchaseOrderID,SalesOrderID"
    strSortList(3).strExternalName = "Invoice Number"
    strSortList(3).strInternalName = "InvoiceNumber"
    strSortList(4).strExternalName = "PP#"
    strSortList(4).strInternalName = "PP"
    Call ReformSortList
End Sub

Private Sub fraInvoiceNum_AfterUpdate()
    InvoiceNum.Enabled = iif(fraInvoiceNum = conSpecific, True, False)
End Sub

Private Sub fraPPNum_AfterUpdate()
    PPNum.Enabled = iif(fraPPNum = conSpecific, True, False)
End Sub

Private Sub fraProductID_AfterUpdate()
    ProductID.Enabled = iif(fraProductID = conSpecific, True, False)
End Sub

Private Sub FromDate_BeforeUpdate(Cancel As Integer)
If (Not (IsNull(Me!ToDate) Or IsNull(Me!FromDate))) Then
  If (DateDiff("d", Me!FromDate, Me!ToDate) < 0) Then
    MsgBox "A data de início não pode ser depois da data final."
    Me.Undo
    Cancel = True
  End If
End If
End Sub


Private Sub MoveDown_Click()
    Dim intCurrentIndex As Integer
    intCurrentIndex = SortList.ListIndex
    If intCurrentIndex < conMaxSortListSize And intCurrentIndex >= 0 Then
        Call SwitchSortOrder(intCurrentIndex, intCurrentIndex + 1)
        Call ReformSortList
        SortList = SortList.ItemData(intCurrentIndex + 1)
    End If
End Sub

Private Sub MoveUp_Click()
```

```
    Dim intCurrentIndex As Integer
    intCurrentIndex = SortList.ListIndex
    If intCurrentIndex > 0 Then
        Call SwitchSortOrder(intCurrentIndex, intCurrentIndex - 1)
        Call ReformSortList
        SortList = SortList.ItemData(intCurrentIndex - 1)
    End If
End Sub

Sub SwitchSortOrder(int1 As Integer, int2 As Integer)
    Dim strTemp As String
    strTemp = strSortList(int1).strExternalName
    strSortList(int1).strExternalName = strSortList(int2).strExternalName
    strSortList(int2).strExternalName = strTemp
    strTemp = strSortList(int1).strInternalName
    strSortList(int1).strInternalName = strSortList(int2).strInternalName
    strSortList(int2).strInternalName = strTemp
End Sub

Sub ReformSortList()
    Dim intLoop As Integer
    Dim strRowSource As String
    strRowSource = ""
    For intLoop = 0 To conMaxSortListSize
        strRowSource = strRowSource & Quote(strSortList(intLoop).strInternalName) & ";" &
Quote(strSortList(intLoop).strExternalName) & ";"
    Next intLoop
    SortList.RowSource = strRowSource
End Sub

Sub ProductID_SectionUpdate()
    Select Case fraProductID
        Case conAll
            sqlProductID_Condition = ""
        Case conSpecific
            If IsNull(ProductID) Then
                sqlProductID_Condition = ""
            Else
                sqlProductID_Condition = "((" & conTable & ".ProductID)=" &
Quote(ReplaceString(ProductID, Chr(34), Chr(34) & Chr(34))) & ")"
            End If
    End Select
End Sub


Sub InvoiceNum_SectionUpdate()
    Select Case fraInvoiceNum
        Case conAll
            sqlInvoiceNum_Condition = ""
        Case conBlank
            sqlInvoiceNum_Condition = "((" & conTable & ".InvoiceNumber) Is Null)"
        Case conNotBlank
            sqlInvoiceNum_Condition = "((" & conTable & ".InvoiceNumber) Is NOT Null)"
        Case conSpecific
            If IsNull(InvoiceNum) Then
                sqlInvoiceNum_Condition = ""
            Else
                sqlInvoiceNum_Condition = "((" & conTable & ".InvoiceNumber)=" & Quote(InvoiceNum) &
")"
            End If
    End Select
End Sub

Sub PPNum_SectionUpdate()
    Select Case fraPPNum
        Case conAll
            sqlPPNum_Condition = ""
        Case conBlank
            sqlPPNum_Condition = "((" & conTable & ".PP) = 'NA')"
        Case conNotBlank
            sqlPPNum_Condition = "((" & conTable & ".PP) <> 'NA')"
        Case conSpecific
            If IsNull(PPNum) Then
                sqlPPNum_Condition = ""
            Else
                sqlPPNum_Condition = "((" & conTable & ".PP)=" & Quote(PPNum) & ")"
            End If
    End Select
End Sub

Sub ReceivedDate_SectionUpdate()
```

```
    Dim strAdjustedFromDate As String
    Dim strAdjustedToDate As String

    strAdjustedFromDate = "#" & FromDate & "#"
    strAdjustedToDate = "#" & ToDate & "#"
    sqlReceivedDate_Condition = BuildSQLCondition("And", iif(IsNull(FromDate), "", "(" & conTable &
".TransactionDate) >= " & strAdjustedFromDate), _
                                                  iif(IsNull(ToDate),   "",   "("   &   conTable   &
".TransactionDate) <= " & strAdjustedToDate))
End Sub

Sub OrderTransfer_SectionUpdate()
    Select Case OrderTransfer
    Case 1: 'all
      sqlOrderTransfer_Condition = ""
    Case 2: 'PO
      If (Len(ComboFromPO) > 0) Then
        sqlOrderTransfer_Condition = conTable & ".PurchaseOrderID = """ & ComboFromPO & """"
      Else
        sqlOrderTransfer_Condition = conTable & ".PurchaseOrderID LIKE """ & "PO*" & """"
      End If
    Case 3: 'WO
      If (Len(ComboFromWO) > 0) Then
        sqlOrderTransfer_Condition = conTable & ".SalesOrderID = """ & ComboFromWO & """"
      Else
        sqlOrderTransfer_Condition = conTable & ".SalesOrderID LIKE """ & "WO*" & """"
      End If
    Case 4: 'IT
      If (Len(ComboFromIT) > 0) Then
        sqlOrderTransfer_Condition = conTable & ".SalesOrderID = """ & ComboFromIT & """"
      Else
        sqlOrderTransfer_Condition = conTable & ".SalesOrderID LIKE """ & "IT*" & """"
      End If
    Case 5: 'SO
      If (Len(ComboFromSOSH) > 0) Then
        sqlOrderTransfer_Condition = conTable & ".SalesOrderID = """ & ComboFromSOSH & """"
      Else
        sqlOrderTransfer_Condition = "(" & conTable & ".SalesOrderID LIKE """ & "SO*" & """ " _
                                   & "OR " & conTable & ".SalesOrderID LIKE """ & "SH*" & """)" _
      End If
    Case 6 'NO
      sqlOrderTransfer_Condition = conTable & ".PurchaseOrderID IS NULL"
    End Select
End Sub

Private Sub ReportSubtotal_SectionUpdate()
    Dim strGroupBy

    sqlGroup_GroupBy = ""
    sqlGroupLeader_Select = ""
    sqlGroupBy_Condition = ""

End Sub

Sub SortBy_SectionUpdate()
    Dim intLoop As Integer
    Call ReformSortList
    sqlSortBy_SortOrder = conTable & "." & strSortList(0).strInternalName & " DESC"
    For intLoop = 1 To conMaxSortListSize
        sqlSortBy_SortOrder   =   sqlSortBy_SortOrder   &   ",   "   &   conTable   &   "."   &
strSortList(intLoop).strInternalName & " DESC"
    Next intLoop
End Sub

Sub BuildFullSQL()

    Dim sqlFilterSELECT As String
    Dim sqlFilterFROM As String
    Dim sqlFilterWHERE As String
    Dim sqlFilterORDERBY As String
    Dim sqlGroupBySELECT As String
    Dim sqlGroupByFROM As String
    Dim sqlGroupByGROUPBY As String
    Dim sqlGroupByHAVING As String
    Dim sqlGroupByORDERBY As String

    Call ProductID_SectionUpdate
    Call OrderTransfer_SectionUpdate
    Call InvoiceNum_SectionUpdate
    Call PFNum_SectionUpdate
    Call ReceivedDate_SectionUpdate
```

```
    Call ReportSubtotal_SectionUpdate
    Call SortBy_SectionUpdate


    sqlFilterSELECT = conTable & ".*, " & sqlGroupLeader_Select
    sqlFilterFROM = conTable
    sqlFilterWHERE = BuildSQLCondition("AND", sqlProductID_Condition, sqlInvoiceNum_Condition, _
                                       sqlPPNum_Condition,                    sqlReceivedDate_Condition,
sqlOrderTransfer_Condition)
    sqlFilterORDERBY = ReplaceString(sqlSortBy_SortOrder, "TransactionDate", "TransactionDate, " &
conTable & ".TransactionID")

    sqlFullFilterStmt = BuildSQLFullStmt(sqlFilterSELECT, sqlFilterFROM, , sqlFilterWHERE,
sqlFilterORDERBY)

    sqlFilter_WHERE = sqlFilterWHERE


    sqlGroupBySELECT = sqlGroupID_Select & ", " & conQuery_Filter & ".GroupLeader as
GroupLeader_GroupBy, Sum(" & conQuery_Filter & ".LineCost) AS GroupCost"
    sqlGroupByFROM = conQuery_Filter
    sqlGroupByGROUPBY = sqlGroup_GroupBy
    sqlGroupByHAVING = iif(sqlGroupBy_Condition = "", "", "(" & sqlGroupBy_Condition & ")")

    sqlFullGroupByStmt = BuildSQLFullStmt(sqlGroupBySELECT, sqlGroupByFROM, sqlGroupByGROUPBY,
sqlGroupByHAVING)

End Sub


Private Sub OrderTransfer_AfterUpdate()
  ComboFromPO.Enabled = False
  ComboFromWO.Enabled = False
  ComboFromIT.Enabled = False
  ComboFromSOSH.Enabled = False

  Select Case OrderTransfer
    Case 1: 'all
    Case 2: 'PO
      ComboFromPO.Enabled = True
    Case 3: 'WO
      ComboFromWO.Enabled = True
    Case 4: 'IT
      ComboFromIT.Enabled = True
    Case 5: 'SO
      ComboFromSOSH.Enabled = True
  End Select
End Sub

Private Sub ToDate_BeforeUpdate(Cancel As Integer)
If (Not (IsNull(Me!ToDate) Or IsNull(Me!FromDate))) Then
  If (DateDiff("d", Me!FromDate, Me!ToDate) < 0) Then
    MsgBox "A data de início não pode ser depois da data final."
    Me.Undo
    Cancel = True
  End If
End If
End Sub
```

FORMULÁRIO DE TRANSFERÊNCIA DE INVENTÁRIO

```
Option Compare Database
Option Explicit
Public strTmp As String
Public strdelete As String
Const cIncomplete = "Incomplete"
Const cComplete = "Complete"
Const DefaultEmailSelection = 1

Private Sub Command222_Click()
On Error Resume Next
    DoCmd.Minimize
    DoCmd.OpenReport "Inventory Transfer", acViewPreview
End Sub

Private Sub CustomerID_AfterUpdate()
Dim dbsCurrent As DAO.Database
Dim strTmp As String
If (IsNull(Me.CustomerID.OldValue)) Then Exit Sub

strTmp = Me.CustomerID.OldValue  'get old value before committing
DoCmd.RunCommand acCmdSaveRecord 'Commit changes in buffer to database
Set dbsCurrent = CurrentDB
dbsCurrent.Execute ("UPDATE [Inventory Transactions] " _
  & "SET CustomerID = " & Me.CustomerID & " " _
  & "WHERE SalesOrderID = """ & Me!SalesOrderID & """ " _
  & "AND CustomerID = " & strTmp & ";")

Set dbsCurrent = Nothing

End Sub

Private Sub Form_AfterDelConfirm(Status As Integer)
    Select Case Status
        Case acDeleteOK
            MsgBox "O registro foi apagado normalmente."
        Case acDeleteCancel

MsgBox "O usuário cancelou o processo."
        Case acDeleteUserCancel
            MsgBox "O usuário cancelou o processo."
    End Select
End Sub

Private Sub Form_BeforeDelConfirm(Cancel As Integer, Response As Integer)
    Response = acDataErrContinue
    If MsgBox("Apagar este Registro?", vbOKCancel) = vbCancel Then
        Cancel = True
        Exit Sub
    End If
    Dim dbsCurrent As DAO.Database
    Dim strSalesOrderID As String
    Dim wrkJet As DAO.Workspace
     Set dbsCurrent = CurrentDB()

    On Error Resume Next
    Dim strTmp As String
    strSalesOrderID = strdelete
    DoCmd.Hourglass True
    dbsCurrent.Execute "DELETE * FROM [Inventory Transactions] WHERE ([Inventory
Transactions].[SalesOrderID] = """ & strSalesOrderID & """);"
    DoCmd.Hourglass False
End Sub

Private Sub Form_Close()
DoCmd.OpenForm "Menu Inicial"
End Sub

Private Sub Form_Delete(Cancel As Integer)
strTmp = Me!SalesOrderID
strdelete = strTmp
End Sub

Private Sub ItemFind_Click()
If Not IsNull([ItemLookUP]) Then
    SalesOrderID.SetFocus
    DoCmd.FindRecord ([ItemLookUP])
    ItemLookUP.SetFocus
Else
    MsgBox "Por favor selecione um PV para exibir."
```

```
    End If

    End Sub

    Private Sub SalesOrderID_BeforeUpdate(Cancel As Integer)
        Dim decision As Integer
        Dim prefix As String
        Dim rstID As DAO.Recordset
        Dim dbsCurrent As Database
        Dim strSQL As String
        Dim strValue As String

        Set dbsCurrent = CurrentDB()

        decision = MsgBox("Você quer realmente alterar o código desta Transferência de Inventário?",
    vbYesNo)
        If decision = 7 Then
            Me.Undo
        Else
            If IsNull(Me!SalesOrderID) Then
                MsgBox "Não é possível substituir o código da Transferência de Inventário por um valor
    nulo.", vbOKOnly
                Me.Undo
                Exit Sub
            End If
            prefix = Left(Forms!InventoryTransfer!SalesOrderID.Value, 2)
            If prefix <> "IT" Then
                MsgBox "Este código não é válido.  Ele deve começar com IT.", vbOKOnly
                Me.Undo
            Else
                strValue = Me!SalesOrderID
                strSQL = "SELECT * FROM [Sales Orders] WHERE ([Sales Orders]![SalesOrderID] = """ &
    strValue & """)"
                Set rstID = dbsCurrent.OpenRecordset(strSQL)
                If rstID.RecordCount <> 0 Then
                    MsgBox "Este código já existe.", vbOKOnly
                    Me.Undo
                End If
                rstID.Close
            End If
        End If
    End Sub

    Private Sub SaveReceipts_Click()
    Dim dbsCurrent As DAO.Database
    Dim rstDMR As DAO.Recordset
    Dim strSQL As String
    Dim strValue As String
    Dim iTransferFromID As Integer
    Dim iCustomerID As Integer


        Set dbsCurrent = CurrentDB()
        If Me!StatusID > 3 Then
        Dim RetValue As Integer
        RetValue = MsgBox("Não é possível salvar o recibo se o registro está Fechado/Arquivado.  ",
    vbOKCancel)
        End
        End If
        strValue = Me!SalesOrderID
        strSQL = "SELECT * FROM [Inventory Transactions] WHERE (([Inventory Transactions]![SalesOrderID] =
    """ & strValue & """) AND ([Inventory Transactions]![TransferFromID] > 0))"

        Set rstDMR = dbsCurrent.OpenRecordset(strSQL)
        If rstDMR.RecordCount > 0 Then
        MsgBox ("Os recibos de TI já foram gerados!")
        rstDMR.Close
        Exit Sub
        End If
        rstDMR.Close

        iTransferFromID = Me!TransferFromID
        iCustomerID = Me!CustomerID
        Me!StatusID = 2
    dbsCurrent.Execute "INSERT INTO [Inventory Transactions] ( TransactionDate,ProductID, SalesOrderID,
    UnitsShipped,TransferFromID,CustomerID,SerialNumber,PP  )" & "SELECT Now() AS [TDate],[Inventory
    Transactions].[ProductID], [Inventory Transactions].[SalesOrderID],[Inventory
    Transactions].[UnitsShippedPG] AS [Shipd]," & iTransferFromID & " AS [FromID]," & iCustomerID & " AS
    [ToID],[Inventory Transactions].[SerialNumber],[Inventory Transactions].[PP]" & "FROM [Inventory
    Transactions]" & "WHERE ((([Inventory Transactions].[SalesOrderID]) = """ & strValue & """ ));"
    MsgBox ("Recibos de TI completados com sucesso!")
```

```
End Sub

Private Sub SalesOrderID_LostFocus()
Dim strTmp As String
On Error Resume Next
strTmp = Me!SalesOrderID
End Sub

Private Sub WorkOrdersSubform_Exit(Cancel As Integer)

    If (Me![WorkOrdersSubform].Locked = False) Then ChangeButtons (False)

End Sub
Sub ChangeButtons(Change As Boolean)
    Change = True

End Sub

Private Sub TransferFromID_AfterUpdate()
Dim dbsCurrent As DAO.Database
Dim strTmp As String
If (IsNull(Me.TransferFromID.OldValue)) Then Exit Sub

strTmp = Me.TransferFromID.OldValue  'get old value before committing
DoCmd.RunCommand acCmdSaveRecord 'Commit changes in buffer to database
Set dbsCurrent = CurrentDB
dbsCurrent.Execute ("UPDATE [Inventory Transactions] " _
  & "SET TransferFromID = " & Me.TransferFromID & " " _
  & "WHERE SalesOrderID = """ & Me!SalesOrderID & """ " _
  & "AND TransferFromID = " & strTmp & ";")

Set dbsCurrent = Nothing

End Sub
```

STATUS DE PRODUTOS

```
Option Compare Database
Option Explicit
Dim DBObject As New EBdefinitionsClass
Dim IObject As New EBupdatedynamiccountClass
Dim crypobject As New EBCrypKeyClass


Private Sub AvailableToPromise_Click()
DoCmd.Requery
On Error Resume Next
DoCmd.Minimize
DoCmd.OpenReport "Available to Promise", acViewPreview
On Error GoTo 0
End Sub


Private Sub Form_Open(Cancel As Integer)

Dim dbsCurrent As DAO.Database
Dim intLocation As Integer
Dim strReportName As String
Dim strValue As String

Set dbsCurrent = CurrentDB()
strValue = DLookup("[Userdefinable 3]", "My Company Information")
strValue = strValue & "EBM_fe.mdb"
DBObject.DatabaseCurrent = dbsCurrent
DBObject.DatabaseName = strValue
    DBObject.SetDatabase


intLocation = 1
strValue = DLookup("[InventoryType]", "[Customers]", "[Customers]![CustomerID] = " & intLocation & "")
IObject.Winxx = DLookup("[Windows]", "[VersionControl]")
IObject.InventoryType = strValue
IObject.InventoryLocation = intLocation
IObject.InventoryEndDate = Now()
IObject.DatabaseCurrent = dbsCurrent
IObject.Winxx = DLookup("[Windows]", "[VersionControl]")
IObject.UpdateDynamicCount
DoCmd.Requery
End Sub

Private Sub Form_Unload(Cancel As Integer)
Set DBObject = Nothing
Set IObject = Nothing
End Sub
Private Sub Command11_Click()
On Error GoTo Err_Command11_Click

    DoCmd.GoToRecord , , acNext

Exit_Command11_Click:
    Exit Sub

Err_Command11_Click:
    Resume Exit_Command11_Click

End Sub
Private Sub Command12_Click()
On Error GoTo Err_Command12_Click


    DoCmd.GoToRecord , , acPrevious

Exit_Command12_Click:
    Exit Sub

Err_Command12_Click:
    Resume Exit_Command12_Click

End Sub

Private Sub ItemFind_Click()
If Not IsNull([ItemLookUP]) Then
    Me.autoid.SetFocus
    DoCmd.FindRecord [ItemLookUP]
    ItemLookUP.SetFocus
Else
```

```
    MsgBox "Por favor selecione um produto para exibir."
End If
End Sub

Private Sub Form_Close()
    DoCmd.Restore
    DoCmd.OpenForm "Menu Inicial"
End Sub
```

```
SELECT [Available To Promise (All Products)].ProductID, [Available To Promise (All Products)].ProductName, [Available To Promise (All Products)].DynamicCount, [Available To Promise (All Products)].WOUnitsOrdered, [Available To Promise (All Products)].[SumOfTotal B/O], [Available To Promise (All Products)].ATP, [Available To Promise (All Products)].autoid
FROM [Available To Promise (All Products)]
GROUP BY [Available To Promise (All Products)].ProductID, [Available To Promise (All Products)].ProductName, [Available To Promise (All Products)].DynamicCount, [Available To Promise (All Products)].WOUnitsOrdered, [Available To Promise (All Products)].[SumOfTotal B/O], [Available To Promise (All Products)].ATP, [Available To Promise (All Products)].autoid;
```

CADASTRO MESTRE

```
Option Compare Database
Option Explicit

Private Sub Form_Current()
On Error Resume Next
    Me.routings_subform.Form![FromOperationID].Requery
    Me.routings_subform.Form![ToOperationID].Requery
    Call LotSizePolicy_AfterUpdate
    MakeEditable (IsEditable())
End Sub

Private Sub ItemFind_Click()
If Not IsNull([ItemLookUP]) Then
    Me.autoid.SetFocus
    DoCmd.FindRecord ([ItemLookUP])
    ItemLookUP.SetFocus
Else
    MsgBox "Por favor selecione um produto para exibir."
End If

End Sub

Private Sub Label189_Click()

End Sub

Private Sub ItemLookUP_GotFocus()
  MakeEditable (True)
End Sub

Private Sub ItemLookUP_LostFocus()
  MakeEditable (IsEditable())
End Sub

Private Sub LotSizePolicy_AfterUpdate()
    If LotSizePolicy = 0 Then
        MinimumOrderSize.Enabled = False
    Else
        MinimumOrderSize.Enabled = True
    End If
End Sub

Private Sub Operations_subform_Enter()
    If IsNull(ProductID) Then
        MsgBox ("Por favor preencha o Cód. Produto antes")
        ProductID.SetFocus
    End If
End Sub

Private Sub ProductID_AfterUpdate()
    Me.Refresh
End Sub

Private Sub ProductID_BeforeUpdate(Cancel As Integer)
On Error GoTo quit
If Not IsNull(Forms![ItemMaster].ProductID) Then
Dim strValue As String
strValue = DLookup("[ProductName]", "Products", "[Products].[ProductID]=
Forms![ItemMaster].ProductID")
End If
MsgBox "Este Código não é válido, provavelmente porque está duplicado ou é nulo. Entre um número
diferente."
DoCmd.CancelEvent
Me.Undo
Forms![ItemMaster]![ProductID].SetFocus
Exit Sub
quit:

End Sub

Private Sub Released_GotFocus()
  MakeEditable (True)
End Sub

Private Sub Released_LostFocus()
  DoCmd.RunCommand acCmdSaveRecord
  MakeEditable (IsEditable())
End Sub
```

```
Private Sub Routings_subform_Enter()
    If IsNull(ProductID) Then
        MsgBox ("Por favor preencha o Cód. Produto antes")
        ProductID.SetFocus
    End If
End Sub

Private Function IsEditable() As Boolean
  If (Me!Released = "Bloqueado") Then
    IsEditable = True
  Else
    IsEditable = False
  End If
End Function

Private Sub MakeEditable(editable As Boolean)
    Me.allowEdits = editable
    Me.Item_Document.Form.allowEdits = editable
    Me.Item_Document.Form.AllowAdditions = editable
    Me.Item_Document.Form.AllowDeletions = editable
    Me.QVL_subform.Form.allowEdits = editable
    Me.QVL_subform.Form.AllowAdditions = editable
    Me.QVL_subform.Form.AllowDeletions = editable
    Me.Operations_subform.Form.allowEdits = editable
    Me.Operations_subform.Form.AllowAdditions = editable
    Me.Operations_subform.Form.AllowDeletions = editable
    Me.routings_subform.Form.allowEdits = editable
    Me.routings_subform.Form.AllowAdditions = editable
    Me.routings_subform.Form.AllowDeletions = editable
End Sub
Private Sub SalvaRegistro_Click()
On Error GoTo Err_SalvaRegistro_Click


    DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, , acMenuVer70

Exit_SalvaRegistro_Click:
    Exit Sub

Err_SalvaRegistro_Click:
    MsgBox Err.Description
    MsgBox "A causa provável é que este é um produto ativo."
    Resume Exit_SalvaRegistro_Click

End Sub
Private Sub AdicionaRegistro_Click()
On Error GoTo Err_AdicionaRegistro_Click


    DoCmd.GoToRecord , , acNewRec
    ProductID.SetFocus
Exit_AdicionaRegistro_Click:
    Exit Sub

Err_AdicionaRegistro_Click:
    MsgBox Err.Description
    Resume Exit_AdicionaRegistro_Click

End Sub
Private Sub ProximoRegistro_Click()
On Error GoTo Err_ProximoRegistro_Click


    DoCmd.GoToRecord , , acNext

Exit_ProximoRegistro_Click:
    Exit Sub

Err_ProximoRegistro_Click:
    MsgBox Err.Description
    Resume Exit_ProximoRegistro_Click

End Sub
Private Sub RegistroAnterior_Click()
On Error GoTo Err_RegistroAnterior_Click


    DoCmd.GoToRecord , , acPrevious

Exit_RegistroAnterior_Click:
    Exit Sub
```

```
Err_RegistroAnterior_Click:
    MsgBox Err.Description
    Resume Exit_RegistroAnterior_Click

End Sub
Private Sub Fechar_Click()
On Error GoTo Err_Fechar_Click

    DoCmd.Close ', , acSavePrompt
    DoCmd.OpenForm "Menu Inicial"

Exit_Fechar_Click:
    Exit Sub

Err_Fechar_Click:
    MsgBox Err.Description
    Resume Exit_Fechar_Click

End Sub
```

MENU INICIAL

```
Option Compare Database   'Use database order for string comparisons.
Option Explicit
Dim DBObject As New EBdefinitionsClass
Dim crypobject As New EBCrypKeyClass
Public vValue As Variant

Private Sub Exit_Click()
Set crypobject = Nothing
Set DBObject = Nothing

Application.quit
End Sub

Private Sub Form_Load()
Dim dbsCurrent As DAO.Database
Dim strValue As String
Set dbsCurrent = CurrentDB()
strValue = DLookup("[Userdefinable 3]", "My Company Information")
strValue = strValue & "DbM_fe.mdb"
DBObject.DatabaseCurrent = dbsCurrent
DBObject.DatabaseName = strValue
DBObject.SetDatabase
vValue = "augustusjohn"
End Sub

Private Sub Form_Open(Cancel As Integer)
    Dim strMinBe As String
    Dim strCurBe As String
    strMinBe = "DbMv4.3.1.4_be"
    strCurBe = Nz(DLookup("[UserDefinable 1]", "[My Company Information]"), "")
    If (strCurBe = "" Or strMinBe > strCurBe) Then
        Call OpenGenericTextBox("O aplicativo não pode rodar com o arquivo de backup atual", _
            "Voce está atualizando o aplicativo, e deve atualizar também o backup atual para que o
programa funcione.", _
            "ETAPA  1" _
            & vbCrLf & "  Todos os usuários devem fechar o aplicativo." _
            & vbCrLf & vbCrLf & "ETAPA  2" _
            & vbCrLf & "   Faça uma cópia do do arquivo backup da base de dados (C:\Arquivo de
Programas\EIFA_bu.mdb). Atualize o backup para a última versão" _
            & "através do formulário 'Controle de Verificação da Versão'." _
            & vbCrLf & vbCrLf & "ETAPA  3" _
            & vbCrLf & " Instale então esta versão (v4321 ou mais recente) para todos os usuários. " _
            & vbCrLf & vbCrLf _
            & "Retome agora a operação normal. NOTA: esta atualização não vai danificar nem
desatualizar nenhum dado da sua base atual.")
    End If
End Sub

Private Sub SelectExample_DblClick(Cancel As Integer)

OK_Click

End Sub
Private Sub SelectTopic_AfterUpdate()

Me!SelectExample.Requery

End Sub

Private Sub OK_Click()
    On Error Resume Next
    If vValue <> "augustusjohn" Then
        DoCmd.Close acForm, vValue, acSaveYes
    End If
    Screen.ActiveForm.Visible = False
    vValue = Me!SelectExample.Column(1)
    If Me!SelectExample.Column(2) = "Form" Then
    DoCmd.Requery
        If    Me!SelectExample.Column(1)    =    "WebBrowseWeb"    Or    Me!SelectExample.Column(1)    =
"WebBrowseable" Then
            Dim objIE3Status As New clsIE30Status
            If objIE3Status.IE30Present = False Then
                MsgBox "Você precisa de Microsoft Internet Explorer 3.0 ou mais recente instalado.",
vbCritical, "Erro ao carregar o Browser"
                Exit Sub
            End If
        End If

        DoCmd.OpenForm Me!SelectExample.Column(1)
```

```
        If Err = 2501 Then Exit Sub
        If Me!ShowMe = True Then
            'ShowHelpAPI
        End If
    ElseIf Me!SelectExample.Column(2) = "Report" Then

        Dim strValue As String
        Dim iTopicID As Integer
        If Me!SelectExample Like "Help*" Then
        iTopicID = CInt(Me!SelectTopic)
        strValue = DLookup("Description", "ExampleTopics", "TopicID = " & iTopicID)
        strValue = "[Help].[ScreenName] = '" & strValue & "'"
        DoCmd.OpenReport "Help_erplite", acViewPreview, , strValue
        Exit Sub
        End If
        DoCmd.OpenReport Forms![Menu Inicial]!SelectExample.Column(1), acViewPreview
        If Me!ShowMe = True Then
            'ShowHelpAPI
        End If
    ElseIf Me!SelectExample.Column(2) = "Dialog" Then
            DoCmd.OpenForm "EmployeeSalesDialogBox"
    End If

End Sub
```

GERAÇÃO DE PC's E OP's

```
Option Compare Database
Option Explicit

Dim strOrderTypeShort As String '"PO" or "WO"
Dim strOrderTypeLong As String '"Purchase Orders" or "Mfg/Work Orders"

Private Sub cmdAbort_Click()
    DoCmd.Close
End Sub

Private Sub cmdAll_Click()
    CurrentDB.Execute "UPDATE MP_Import_POWO_tmp SET MP_Import_POWO_tmp.Import = True;"
    Me.MP_Import_POWO_tmp_subform.Form.Refresh
End Sub

Private Sub cmdAutoGrouping_Click()

    Dim rstPOWOSortByDate As DAO.Recordset
    Dim intCurrentGroup As Integer
    Dim dteCurrentReleaseDate As Date
    Dim dteCurrentDueDate As Date

    Call cmdRemoveGrouping_Click
    Set rstPOWOSortByDate = CurrentDB.OpenRecordset("SELECT MP_Import_POWO_tmp.*" & " FROM
MP_Import_POWO_tmp" & " WHERE ((MP_Import_POWO_tmp.Import) = True)" & " ORDER BY
MP_Import_POWO_tmp.ReleaseDate, MP_Import_POWO_tmp.DueDate, MP_Import_POWO_tmp.ID;")

    intCurrentGroup = 0
    With rstPOWOSortByDate
        Do While Not .EOF
            dteCurrentReleaseDate = ![ReleaseDate]
            dteCurrentDueDate = ![DueDate]
            intCurrentGroup = intCurrentGroup + 1

            Do
                .Edit
                ![Grouping] = Format(intCurrentGroup, "000")
                .Update
                .MoveNext
                If .EOF Then Exit Do
            Loop While ![ReleaseDate] = dteCurrentReleaseDate And ![DueDate] = dteCurrentDueDate
        Loop
    End With
    Set rstPOWOSortByDate = Nothing
    Me.MP_Import_POWO_tmp_subform.Form.Refresh

End Sub

Private Sub cmdNone_Click()
    CurrentDB.Execute "UPDATE MP_Import_POWO_tmp SET MP_Import_POWO_tmp.Import = False;"
    Me.MP_Import_POWO_tmp_subform.Form.Refresh
End Sub

Private Sub cmdOK_Click()
    Select Case strOrderTypeShort
        Case "PO"
            If GeneratePlannedPO() Then
                DoCmd.Close
            End If
        Case "WO"
            If GeneratePlannedWO() Then
                DoCmd.Close
            End If
        Case Else
            MsgBox "Erro de programação..."
    End Select
End Sub

Private Sub cmdRemoveGrouping_Click()
    CurrentDB.Execute "UPDATE MP_Import_POWO_tmp SET MP_Import_POWO_tmp.Grouping = Null;"
    Me.MP_Import_POWO_tmp_subform.Form.Refresh
End Sub

Private Sub Form_Load()
    Me.Caption = ReplaceString(Me.Caption, "[strOrderTypeLong]", strOrderTypeLong)
    Me.lblHeader.Caption = ReplaceString(Me.lblHeader.Caption, "[strOrderTypeLong]", strOrderTypeLong)
    Me.lblDescription.Caption = ReplaceString(Me.lblDescription.Caption, "[strOrderTypeLong]",
strOrderTypeLong)
    Me.lblTableTitle.Caption = strOrderTypeLong
```

```
    Me.cmdOK.Caption = ReplaceString(Me.cmdOK.Caption, "[strOrderTypeShort]", strOrderTypeShort)
    Me.MP_Import_POWO_tmp_subform![lblGrouping].Caption                                    =
ReplaceString(Me.MP_Import_POWO_tmp_subform![lblGrouping].Caption,              "[strOrderTypeShort]",
strOrderTypeShort)
End Sub

Private Sub Form_Open(Cancel As Integer)
    Select Case Me.OpenArgs
        Case "PO"
            strOrderTypeShort = "PO"
            strOrderTypeLong = "Purchase Orders"
        Case "WO"
            strOrderTypeShort = "WO"
            strOrderTypeLong = "Mfg/Work Orders"
        Case Else
            strOrderTypeShort = "??"
            strOrderTypeLong = "????"
    End Select
End Sub

Function GeneratePlannedPO() As Boolean
On Error GoTo Err_GeneratePlannedPO

    Dim wspCurrent As DAO.Workspace
    Dim dbsCurrent As DAO.Database
    Dim rstPOImported As DAO.Recordset
    Dim rstPOParents As DAO.Recordset, rstPOChildren As DAO.Recordset
    Dim strCurrentPOID As String
    Dim strCurrentPOGroup As String
    Dim blnTransStarted As Boolean

    Dim varSupplierID_DV As Variant
    Dim intPST_DV As Integer
    Dim varStatusID_DV As Integer
    Dim strUserdefinable1_DV As String
    Dim strUserdefinable2_DV As String
    Dim lngCustomerID_DV As Long
    Dim varCustomerShipToID_DV As Variant

    DoCmd.Hourglass True

    intPST_DV = Nz(DLookup("[CurrencyRate]", "[My Company Information]"), 0)
    varStatusID_DV = DLookup("[StatusID]", "Status_PO", "[Status]='Planned'")
    strUserdefinable1_DV = "Inventory"
    strUserdefinable2_DV = "USA"
    lngCustomerID_DV = 1
    varCustomerShipToID_DV = DLookup("[CustomerExtraID]", "CustomersExtraShipTo", "[CustomerID] = 1
And [Default] = True")

    GeneratePlannedPO = True
    Set wspCurrent = DBEngine.Workspaces(0)
    Set dbsCurrent = CurrentDB
    Set rstPOImported = dbsCurrent.OpenRecordset(SQLDependsOnOption())
    Set rstPOParents = dbsCurrent.OpenRecordset("Purchase Orders")
    Set rstPOChildren = dbsCurrent.OpenRecordset("Inventory Transactions")

    wspCurrent.BeginTrans
    blnTransStarted = True
    With rstPOImported
        Do While Not .EOF

            strCurrentPOID = GetNextSN("PO") & "P"
            strCurrentPOGroup = Nz(![Grouping], "NoGroup")

            With rstPOParents
                .AddNew
                ![PurchaseOrderID] = strCurrentPOID
                ![PST] = intPST_DV
                ![StatusID] = varStatusID_DV
                ![Userdefinable 1] = strUserdefinable1_DV
                ![UserDefinable 2] = strUserdefinable2_DV
                ![CustomerID] = lngCustomerID_DV
                ![CustomerShipToID] = varCustomerShipToID_DV
                ![OrderDate] = rstPOImported![ReleaseDate]
                .Update
            End With

            Do
                With rstPOChildren
                    .AddNew
                    ![TransactionDate] = Now()
```

```
                    ![ProductID] = rstPOImported![ProductID]
                    ![PurchaseOrderID] = strCurrentPOID
                    ![UnitPrice] = rstPOImported![StandardUnitPrice]
                    ![UnitsOrdered] = rstPOImported![SumOfOrderQuantity]
                    ![DateRequired] = rstPOImported![DueDate]
                    ![DatePromised] = rstPOImported![DueDate]
                    ![Currency] = rstPOImported![Currency]
                    .update
                End With

                dbsCurrent.Execute "UPDATE Products SET Products.UnitPrice = " & ![StandardUnitPrice]
& " WHERE (((Products.ProductID)=" & Quote(ReplaceString(![ProductID], Chr(34), Chr(34) & Chr(34))) &
"));"

                .MoveNext
                If .EOF Then Exit Do
            Loop While Nz(![Grouping] = strCurrentPOGroup, False)

        Loop
    End With

    wspCurrent.CommitTrans
    DoCmd.Hourglass False
    MsgBox "Todos os Pedidos de Compra foram criados." & vbCrLf & vbCrLf & "Você precisa utilizar o
formulário 'PC - inventário' para adicionar mais informações e ativá-los.", vbInformation

Exit_GeneratePlannedPO:
    Set wspCurrent = Nothing
    Set dbsCurrent = Nothing
    Set rstPOParents = Nothing
    Set rstPOChildren = Nothing
    Set rstPOImported = Nothing
    Exit Function

Err_GeneratePlannedPO:
    GeneratePlannedPO = False
    If blnTransStarted Then
        wspCurrent.Rollback
    End If
    DoCmd.Hourglass False
    MsgBox Err.Description, vbCritical
    MsgBox "Erro durante a criação do PC!  Não foi criado nenhum PC."
    Resume Exit_GeneratePlannedPO

End Function

Function GeneratePlannedWO() As Boolean
On Error GoTo Err_GeneratePlannedWO

    Dim wspCurrent As DAO.Workspace
    Dim dbsCurrent As DAO.Database
    Dim rstWOImported As DAO.Recordset
    Dim rstWOParents As DAO.Recordset, rstWOChildren As DAO.Recordset
    Dim strCurrentWOID As String
    Dim strCurrentWOGroup As String
    Dim blnTransStarted As Boolean

    'Variables to store Purchase Orders record default values
    Dim lngTransFromID_DV As Long
    Dim lngCustomerID_DV As Long
    Dim lngEndCustomerID_DV As Long
    Dim varEmployeeID_DV As Variant
    Dim strCustomerPONumber_DV As String
    Dim strSalesTrackingNumber_DV As String
    Dim varStatusID_DV As Variant
    Dim dblUnitsShipped_DV As Double
    Dim strControl_DV As String

    DoCmd.Hourglass True

    lngTransFromID_DV = 0
    lngCustomerID_DV = 20
    lngEndCustomerID_DV = 0
    strCustomerPONumber_DV = "0"
    strSalesTrackingNumber_DV = "0"
    varStatusID_DV = DLookup("[StatusID]", "Status_WO", "[Status]='Planned'")
    dblUnitsShipped_DV = 0
    strControl_DV = "ZW"
    GeneratePlannedWO = True
    Set wspCurrent = DBEngine.Workspaces(0)
    Set dbsCurrent = CurrentDB
```

```
    Set rstWOImported = dbsCurrent.OpenRecordset(SQLDependsOnOption())
    Set rstWOParents = dbsCurrent.OpenRecordset("Sales Orders")
    Set rstWOChildren = dbsCurrent.OpenRecordset("Inventory Transactions")

    wspCurrent.BeginTrans
    blnTransStarted = True
    With rstWOImported
        Do While Not .EOF
            strCurrentWOID = GetNextSN("WO") & "F"
            strCurrentWOGroup = Nz(![Grouping], "NoGroup")
            With rstWOParents
                .AddNew
                ![SalesOrderID] = strCurrentWOID
                ![TransferFromID] = lngTransFromID_DV
                ![CustomerID] = lngCustomerID_DV
                ![EndCustomerID] = lngEndCustomerID_DV
                ![OrderDate] = rstWOImported![ReleaseDate]
                ![CustomerPONumber] = strCustomerPONumber_DV
                ![SalesTrackingNumber] = strSalesTrackingNumber_DV
                ![StatusID] = varStatusID_DV
                ![ReleaseDate] = rstWOImported![ReleaseDate]
                ![DueDate] = rstWOImported![DueDate]
                .Update
            End With

            Do
                With rstWOChildren
                    .AddNew
                    ![TransactionDate] = Now()
                    ![ProductID] = rstWOImported![ProductID]
                    ![TransferFromID] = lngTransFromID_DV
                    ![CustomerID] = lngCustomerID_DV
                    ![UnitsOrdered] = rstWOImported![SumOfOrderQuantity]
                    ![UnitsShipped] = dblUnitsShipped_DV
                    ![UnitsShippedFG] = rstWOImported![SumOfOrderQuantity]
                    ![SalesOrderID] = strCurrentWOID
                    ![Control] = strControl_DV
                    ![DateRequired] = rstWOImported![DueDate]
                    ![DatePromised] = rstWOImported![DueDate]
                    .Update
                End With

                .MoveNext
                If .EOF Then Exit Do
            Loop While Nz(![Grouping] = strCurrentWOGroup, False)

        Loop
    End With

    wspCurrent.CommitTrans
    DoCmd.Hourglass False
    MsgBox "Todas as Ordens de Produção foram criadas." & vbCrLf & vbCrLf _
        & "Você precisa utilizar o formulário 'Ordens de Produção' para adicionar mais informações
e ativá-las.", vbInformation

Exit_GeneratePlannedWO:
    Set wspCurrent = Nothing
    Set dbsCurrent = Nothing
    Set rstWOParents = Nothing
    Set rstWOChildren = Nothing
    Set rstWOImported = Nothing
    Exit Function

Err_GeneratePlannedWO:
    GeneratePlannedWO = False
    If blnTransStarted Then
        wspCurrent.Rollback
    End If
    DoCmd.Hourglass False
    MsgBox Err.Description, vbCritical
    MsgBox "Erro durante a criação da OP!  A Ordem não foi criada."
    Resume Exit_GeneratePlannedWO
End Function

Function SQLDependsOnOption() As String
    SQLDependsOnOption = CurrentDB.QueryDefs("MP_Import_POWO_Summing").SQL
    If chkAutoSum Then
        SQLDependsOnOption = ReplaceString(SQLDependsOnOption, ", MP_Import_POWO_tmp.ID", "")
    End If
End Function
```

PLANEJAMENTO DE NECESSIDADES – ENTRADA DE DADOS

```
Option Compare Database
Option Explicit

Private Const conSourceQueries As Integer = 1
Private Const conSourceFiles As Integer = 2
Private Const consourcedefault As Integer = conSourceQueries
Private Const conEBPlannerDLLCopyright As String = "MRP Poli"

Private boolPreserveWOcmb As Boolean
Private boolPreservePOcmb As Boolean

Private Sub cmdExit_Click()
    On Error Resume Next
    DoCmd.Close acForm, "MP_MRPOutput_form"
    On Error GoTo 0
    DoCmd.Close acForm, Me.Name
End Sub


Private Sub cmdFillQuantity_Click()
    On Error GoTo Err_cmdFillQuantity_Click
    Dim dbsCurrent As DAO.Database
    Set dbsCurrent = CurrentDB()

    If (IsNull(Me.txtFillQuantity)) Then
        MsgBox "Por favor insira a quantidade antes de clicar o botão."
        GoTo Exit_cmdFillQuantity_Click
    End If
    If (Not IsNumeric(Me.txtFillQuantity)) Then
        MsgBox "Por favor insira um 'número' para a quantidade."
        GoTo Exit_cmdFillQuantity_Click
    End If
    If (IsNull(Me.MP_MRPInput_ProductID_Subform!ProductID)) Then
        MsgBox "Nenhum item foi selecionado. Escolha um produto da lista."
        GoTo Exit_cmdFillQuantity_Click
    End If

    dbsCurrent.Execute ("UPDATE Demands " _
                    & "SET Quantity = " & CDbl(Me.txtFillQuantity) & " " _
                    & "WHERE SchedulingPlanID = " & Me!SchedulingPlanID & " " _
                    &               "AND             ProductID             =          """                &
ReplaceString(Me.MP_MRPInput_ProductID_Subform!ProductID, Chr(34), Chr(34) & Chr(34)) & """ " _
                    & "AND PeriodID > 0;")

Exit_cmdFillQuantity_Click:
    Set dbsCurrent = Nothing
    Me.Refresh
    Me.MP_MRPInput_PeriodID_Subform.Form.Refresh
    Exit Sub
Err_cmdFillQuantity_Click:
    MsgBox "Erro no preenchimento dos números.  " & Err.Number & ": " & Err.Description
    Resume Exit_cmdFillQuantity_Click
End Sub

Private Sub cmdGetOutput_Click()
    Dim strOpenArgs As String
    Dim dbsCurrent As DAO.Database
    Dim rstDups As DAO.Recordset
    Dim strSQL As String
    Dim rstNegativeCounts As Recordset
    Set dbsCurrent = CurrentDB()
    Set  rstDups  =  dbsCurrent.OpenRecordset("SELECT  DISTINCTROW  First(Products.autoid)  AS  [autoid
Field],  Count(Products.autoid)  AS  NumberOfDups  FROM  Products  GROUP  BY  Products.autoid  HAVING
(((Count(Products.autoid))>1));", dbOpenDynaset)
    If rstDups.RecordCount > 0 Then
        Dim Msg, Style, Title, Help, Ctxt, Response, MyString
        Msg = "Para continuar com " & conEBPlannerDLLCopyright & ", o aplicativo precisa reparar
códigos duplicados no 'Cadastro Mestre'. É altamente recomendável que você faça uma cópia de segurança
dos dados (EPPA_bu.mdb) antes de prosseguir com o reparo. O reparo irá ainda APAGAR as previsões de
demanda, que deverão ser registradas novamente. Você quer prosseguir?"
        Style = vbYesNo + vbCritical + vbDefaultButton2
        Title = "AUTO-REPARO"
        Help = "DEMO.HLP"
        Ctxt = 1000
        Response = MsgBox(Msg, Style, Title)
        If Response = vbYes Then
            MyString = "Yes"
            Else
            MyString = "No" ' Perform some action.
```

```
            Exit Sub
        End If
        DoCmd.SetWarnings False
        DoCmd.OpenQuery "Make_ProductsCopy"
        dbsCurrent.Execute ("DELETE * FROM [Products]")
        DoCmd.OpenQuery "ProductsCopy_Query"
        DoCmd.DeleteObject acTable, "ProductsCopy"
        DoCmd.SetWarnings True

    End If

    Call UpdateDynamicCount(1)
    strSQL = "SELECT Products.*, Products.DynamicCount FROM Products WHERE
(((Products.DynamicCount)<0));"
    Set rstNegativeCounts = dbsCurrent.OpenRecordset(strSQL)
    If (rstNegativeCounts.RecordCount > 0) And (Me.chkComponents) Then MsgBox "O programa detectou
algumas contagens de inventário negativas. Estes valores serão considerados como 'zero' no " &
conEBPlannerDLLCopyright & ". Você pode querer considerar estes números negativos antes de
prosseguir."

    strOpenArgs = "SP=" & CStr(Me!SchedulingPlanID)
    strOpenArgs = strOpenArgs & "#PD=" & Me.cmbPO_Periods
    strOpenArgs = strOpenArgs & "#WD=" & Me.cmbWO_Periods
    strOpenArgs = strOpenArgs & "#DS=" & CStr(Me.fraDataSource)
    strOpenArgs = strOpenArgs & "#C=" & iif(Me.chkComponents, "T", "F")
    strOpenArgs = strOpenArgs & "#D=" & iif(Me.chkDemand, "T", "F")
    strOpenArgs = strOpenArgs & "#P=" & iif(Me.chkPOs, "T", "F")
    strOpenArgs = strOpenArgs & "#W=" & iif(Me.chkWOs, "T", "F")
    strOpenArgs = strOpenArgs & "#S=" & iif(Me.chkSOs, "T", "F")
    strOpenArgs = strOpenArgs & "#I=" & iif(Me.chkInventory, "T", "F")
    DoCmd.RunCommand acCmdSaveRecord
    DoCmd.OpenForm "MP_MRPOutput_form", , , , , , strOpenArgs
    DoCmd.Close acForm, Me.Name, acSaveNo
End Sub


Private Sub DemandPeriodCount_AfterUpdate()
    On Error GoTo Err_DemandPeriodCount_AfterUpdate
    Dim dbsCurrent As DAO.Database
    Dim rstDemandPeriodsToAdd As DAO.Recordset
    Dim rstDemandProductsToAdd As DAO.Recordset
    Dim strCurrentProductID As String
    Dim intLoop As Integer

    Set dbsCurrent = CurrentDB()

    If (Me!DemandPeriodCount.OldValue > Me!DemandPeriodCount.Value) Then
        dbsCurrent.Execute ("DELETE * FROM Demands " _
                          & "WHERE SchedulingPlanID = " & Me!SchedulingPlanID.Value & " " _
                          & "AND PeriodID > " & Me!DemandPeriodCount.Value & ";")
    ElseIf (Me!DemandPeriodCount.OldValue < Me!DemandPeriodCount.Value) Then
        Set rstDemandProductsToAdd = dbsCurrent.OpenRecordset("SELECT Demands.ProductID " _
                                                            & "FROM Demands" _
                                                            & " WHERE (((Demands.SchedulingPlanID)="
& Me!SchedulingPlanID & ") AND ((Demands.PeriodID)=0));")
        Set rstDemandPeriodsToAdd = dbsCurrent.OpenRecordset("Demands")

        With rstDemandProductsToAdd
            Do While Not .EOF
                strCurrentProductID = ![ProductID]
                With rstDemandPeriodsToAdd
                    For intLoop = Me!DemandPeriodCount.OldValue + 1 To Me!DemandPeriodCount
                        .AddNew
                        ![PeriodID] = intLoop
                        ![ProductID] = strCurrentProductID
                        ![SchedulingPlanID] = Me!SchedulingPlanID
                        .Update
                    Next intLoop
                End With
                .MoveNext
            Loop
        End With
        rstDemandPeriodsToAdd.Close
        Set rstDemandPeriodsToAdd = Nothing
        rstDemandProductsToAdd.Close
        Set rstDemandProductsToAdd = Nothing
    Else
    End If
    Set dbsCurrent = Nothing
    DoCmd.RunCommand acCmdSaveRecord 'Force no undo
    Me.Refresh
```

```
        Call Update_DemandPeriodsAhead(Me!DemandPeriodCount)

    Exit_DemandPeriodCount_AfterUpdate:
        Exit Sub
    Err_DemandPeriodCount_AfterUpdate:
        MsgBox "Houve um erro durante a atualização do período de demanda" & vbCrLf & Err.Number & " = " &
    Err.Description
        Resume Exit_DemandPeriodCount_AfterUpdate
    End Sub

    Private Sub DemandPeriodUnit_AfterUpdate()
        If (Me!DemandPeriodUnit = 0 Or Me!DemandPeriodUnit = 1) Then
            Me.DemandPeriodUnitSize.Enabled = True
            If (Me!DemandPeriodUnitSize < 1) Then
                Me!DemandPeriodUnitSize = 1
            End If
        Else
            Me!DemandPeriodUnitSize = 1
            Me.DemandPeriodUnitSize.Enabled = False
        End If
    End Sub

    Private Sub Form_AfterInsert()
        If (Me!DemandPeriodUnit = 0 Or Me!DemandPeriodUnit = 1) Then
            Me.DemandPeriodUnitSize.Enabled = True
            If (Me!DemandPeriodUnitSize < 1) Then
                Me!DemandPeriodUnitSize = 1
            End If
        Else
            Me!DemandPeriodUnitSize = 1
            Me.DemandPeriodUnitSize.Enabled = False
        End If
    End Sub

    Private Sub Form_Current()
        Call Update_DemandPeriodsAhead(Me!DemandPeriodCount)
        Call EmptyTempTables
    End Sub

    Private Sub Form_Load()
        boolPreserveWOcmb = False
        boolPreservePOcmb = False
        If (Len(Nz(Me.OpenArgs, "")) > 0) Then Call Load_MeOpenArgs
    End Sub

    Private Sub SchedulingPlan_Find_Click()
        If Not IsNull(Me.SchedulingIDLookUp) Then
            Me.SchedulingPlanID.SetFocus
            DoCmd.FindRecord (Me.SchedulingIDLookUp)
            Me.SchedulingIDLookUp.SetFocus
            Me.Refresh
        Else
            MsgBox "Por favor selecione um Planejamento para exibir."
        End If
    End Sub

    Private Sub Load_MeOpenArgs()
        Dim strParam As String
        Dim strParamID As String
        Dim strParamVal As String
        Dim strRemaining As String
        Dim lngSchedulingPlanID As Long
        Dim strSchedulingPlanName As String
        Dim intDemandPeriodCount As Integer

        strRemaining = Me.OpenArgs
        strSchedulingPlanName = ""
        intDemandPeriodCount = 1


        Do While (Len(strRemaining) > 0)
            strParam = ParsePop(strRemaining, "#")
            If (Len(strParam) > 0) Then
                strParamVal = strParam
                strParamID = ParsePop(strParamVal, "=")
                Select Case (strParamID)
                    Case "SP": 'SchedulingPlanID
                        If (Len(strParamVal) < 1) Then
                        ElseIf (IsNull(DLookup("SchedulingPlanID", "SchedulingPlans", "SchedulingPlanID =
    & strParamVal))) Then
                        Else
```

```
                        lngSchedulingPlanID = Val(strParamVal)
                        Me.SchedulingIDLookUp = lngSchedulingPlanID
                        Call SchedulingPlan_Find_Click
                        intDemandPeriodCount    =    Nz(DLookup("DemandPeriodCount",    "SchedulingPlans",
"SchedulingPlanID = " & lngSchedulingPlanID), 1)
                        Call Update_DemandPeriodsAhead(intDemandPeriodCount)
                    End If
                Case "PD": 'POsDue cmbBox
                    If (IsNumeric(strParamVal)) Then
                        If (Val(strParamVal) < intDemandPeriodCount And Val(strParamVal) > 0) Then
                            Me.cmbPO_Periods = Val(strParamVal)
                            boolPreservePOcmb = True
                        Else
                            Me.cmbPO_Periods.Value = intDemandPeriodCount
                        End If
                    Else
                        Me.cmbPO_Periods.Value = intDemandPeriodCount
                    End If
                Case "WD": 'WOsDue cmbBox
                    If (IsNumeric(strParamVal)) Then
                        If (Val(strParamVal) < intDemandPeriodCount And Val(strParamVal) > 0) Then
                            Me.cmbWO_Periods = Val(strParamVal)
                            boolPreserveWOcmb = True
                        Else
                            Me.cmbWO_Periods.Value = intDemandPeriodCount
                        End If
                    Else
                        Me.cmbWO_Periods.Value = intDemandPeriodCount
                    End If
                Case "DS": 'DataSource
                    If    Not    (strParamVal    =    CStr(conSourceQueries)    Or    strParamVal    =
CStr(conSourceFiles)) Then
                        Me.fraDataSource = Val(strParamVal)
                    End If
                Case "C":  'Components chkbox
                    If (strParamVal = "F") Then
                        Me.chkComponents = False
                    ElseIf (strParamVal = "T") Then
                        Me.chkComponents = True
                    End If
                Case "D":  'Demand chkbox
                    If (strParamVal = "F") Then
                        Me.chkDemand = False
                    ElseIf (strParamVal = "T") Then
                        Me.chkDemand = True
                    End If
                Case "P":  'POs chkbox
                    If (strParamVal = "F") Then
                        Me.chkPOs = False
                    ElseIf (strParamVal = "T") Then
                        Me.chkPOs = True
                    End If
                Case "W":  'WOs chkbox
                    If (strParamVal = "F") Then
                        Me.chkWOs = False
                    ElseIf (strParamVal = "T") Then
                        Me.chkWOs = True
                    End If
                Case "S":  'SOs chkbox
                    If (strParamVal = "F") Then
                        Me.chkSOs = False
                    ElseIf (strParamVal = "T") Then
                        Me.chkSOs = True
                    End If
                Case "I":  'Inventory chkbox
                    If (strParamVal = "F") Then
                        Me.chkInventory = False
                    ElseIf (strParamVal = "T") Then
                        Me.chkInventory = True
                    End If
                Case Else:
                    'Do nothing
            End Select
        End If

    Loop
End Sub

Private Sub Update_DemandPeriodsAhead(newValue As Integer)
    Dim intCmbWOPeriods As Integer
    Dim intCmbPOPeriods As Integer
```

```
    Dim strNewRowSource As String
    Dim intLoop As Integer
    Dim intSetToVal As Integer

    intCmbWOPeriods = Val(Me.cmbWO_Periods)
    intCmbPOPeriods = Val(Me.cmbPO_Periods)
    Me.cmbWO_Periods = "1"
    Me.cmbPO_Periods = "1"

    strNewRowSource = "1"
    For intLoop = 2 To newValue
        strNewRowSource = strNewRowSource & ";" & intLoop
    Next intLoop
    Me.cmbWO_Periods.RowSource = strNewRowSource
    Me.cmbPO_Periods.RowSource = strNewRowSource

    If (boolPreserveWOcmb = True And Not (newValue < intCmbWOPeriods)) Then
        Me.cmbWO_Periods = CStr(intCmbWOPeriods)
        boolPreserveWOcmb = False
    Else
        Me.cmbWO_Periods = CStr(newValue)
    End If
    If (boolPreservePOcmb = True And Not (newValue < intCmbPOPeriods)) Then
        Me.cmbPO_Periods = CStr(intCmbPOPeriods)
        boolPreservePOcmb = False
    Else
        Me.cmbPO_Periods = CStr(newValue)
    End If
    Me.cmbWO_Periods = "1"
    Me.cmbPO_Periods = "1"
    Me.cmbWO_Periods.Requery
    Me.cmbPO_Periods.Requery
End Sub

Private Sub EmptyTempTables()
    Dim dbsCurrent As DAO.Database
    Set dbsCurrent = CurrentDB()
    dbsCurrent.Execute ("DELETE * FROM MP_MRPOutput;")
    dbsCurrent.Execute ("DELETE * FROM MP_POsDue;")
    dbsCurrent.Execute ("DELETE * FROM MP_WOsDue;")
    dbsCurrent.Execute ("DELETE * FROM MP_InternalDataErrorLog;")
    dbsCurrent.Execute ("DELETE * FROM MP_DataErrorsLog;")
    Set dbsCurrent = Nothing
End Sub

Private Function DemandPeriodDefinitionStored() As String
    Select Case (DLookup("[DemandPeriodUnit]", "TimeDefinition"))
        Case 0
            DemandPeriodDefinitionStored = " x " & Nz(DLookup("[DemandPeriodUnitSize]",
"[TimeDefinition]"), 0) & " Dia(s)"
        Case 1
            DemandPeriodDefinitionStored = " x " & Nz(DLookup("[DemandPeriodUnitSize]",
"[TimeDefinition]"), 0) & " Semana(s)"
        Case 2
            DemandPeriodDefinitionStored = " x 1 Mês"
        Case 3
            DemandPeriodDefinitionStored = " x 1 Trimestre"
        Case Else
            DemandPeriodDefinitionStored = " x ???"
    End Select
End Function

Private Function DemandPeriodDefinitionCurrent() As String
    Select Case (Me!DemandPeriodUnit)
        Case 0
            DemandPeriodDefinitionCurrent = "" & Nz(Me!DemandPeriodUnitSize, 0) & " Dia(s)"
        Case 1
            DemandPeriodDefinitionCurrent = "" & Nz(Me!DemandPeriodUnitSize, 0) & " Semana(s)"
        Case 2
            DemandPeriodDefinitionCurrent = " x 1 Mês"
        Case 3
            DemandPeriodDefinitionCurrent = " x 1 Trimestre"
        Case Else
            DemandPeriodDefinitionCurrent = " x ???"
    End Select
End Function

Private Sub Form_Close()
DoCmd.OpenForm "Menu Inicial"
End Sub
```

PLANEJAMENTO DE NECESSIDADES – SAÍDA

```
Option Compare Database
Option Explicit

Private Const conSourceQueries As Integer = 1
Private Const conSourceFiles As Integer = 2
Private Const consourcedefault = conSourceQueries
Private Const conAcc97 As Integer = 8
Private Const conAcc2000 As Integer = 9
Private Const conCurAcc As Integer = conAcc97
Private Const conInputForm As String = "MP_MRPInput_form"
Private Const conLoadingMsgForm As String = "AAA_Loading"
Private Const conEBPlannerDLLCopyright As String = "MRP Poli"
Private strMeOpenArgs As String
Private lngSchedulingPlanID As Long
Private boolErrorFindingSchedulingPlan As Boolean
Private intAccessVersion As Integer
Private PlannerObj As Object

Private Sub cmdCompute_Click()
    Call Compute_PlannerObj
End Sub

Private Sub cmdExit_Click()
    If (IsLoaded(conInputForm)) Then
        DoCmd.Close acForm, conInputForm
    End If
    DoCmd.Close acForm, Me.Name, acSaveNo
End Sub

Private Sub cmdReload_Click()
    Call Load_PlannerObj
End Sub

Private Sub cmdRetrieve_Click()
    Call Output_PlannerObj
End Sub

Private Sub cmdReturn_Click()
    DoCmd.OpenForm conInputForm, , , , , , strMeOpenArgs
    DoCmd.Close acForm, Me.Name
End Sub

Private Sub Form_Open(Cancel As Integer)
    On Error GoTo Err_Form_Open
    Dim strAccessVersion As String
    strAccessVersion = SysCmd(acSysCmdAccessVer)
    intAccessVersion = CInt(ParsePop(strAccessVersion, "."))

    strMeOpenArgs = Nz(Me.OpenArgs, "")
    Call Load_MeOpenArgs
    Call Load_PlannerObj

Exit_Form_Open:
    Exit Sub

Err_Form_Open:
    Resume Exit_Form_Open
End Sub


Private Sub Form_Unload(Cancel As Integer)
    Call Unload_PlannerObj
    If (intAccessVersion = conAcc97) Then
        If (IsLoaded(conLoadingMsgForm)) Then
            DoCmd.Close acForm, conLoadingMsgForm
        End If
    End If
End Sub

Private Sub Report_Errors_Critical_Click()
    On Error GoTo Err_Report_Errors_Critical_Click
    Dim dbsCurrent As DAO.Database
    Dim rstCountBad As DAO.Recordset
    Dim intErrorCount As Integer

    Set dbsCurrent = CurrentDB()
    Set   rstCountBad   =   dbsCurrent.OpenRecordset("SELECT   *   "   &   "FROM   MP_InternalDataErrorLog,
MP_InternalDataErrorCode " & "WHERE MP_InternalDataErrorLog.ErrorID = MP_InternalDataErrorCode.ErrorID
" & "AND Importance > 0;")
```

```
    If (rstCountBad.RecordCount = 0) Then
        intErrorCount = 0
    Else
        rstCountBad.MoveLast
        intErrorCount = rstCountBad.RecordCount
    End If
    rstCountBad.Close
    Set rstCountBad = Nothing
    Set dbsCurrent = Nothing

    If (intErrorCount > 0) Then
        On Error Resume Next
        DoCmd.Close acReport, "MP_CriticalError_Report"
        On Error GoTo Err_Report_Errors_Critical_Click
        DoCmd.Minimize
        DoCmd.OpenReport "MP_CriticalError_Report", acViewPreview
    Else
        MsgBox "Nenhnum erro reportado.  Recarregue os dados para verificar se há erros novamente."
    End If


Exit_Report_Errors_Critical_Click:
    Exit Sub

Err_Report_Errors_Critical_Click:
    If Err <> 2501 And Err <> 2465 Then
        MsgBox "Erros não relacionados durante verificação.  '" & Err.Number & "' = " &
Err.Description
    End If
    Resume Exit_Report_Errors_Critical_Click
End Sub

Private Sub Report_Errors_Dll_Click()
    On Error GoTo Err_Report_Errors_Dll_Click

    If (dCount("*", "MP_DataErrorsLog") > 0) Then
        On Error Resume Next
        DoCmd.Close acReport, "MP_DllError_Report"
        On Error GoTo Err_Report_Errors_Dll_Click
        DoCmd.Minimize
        DoCmd.OpenReport "MP_DllError_Report", acViewPreview
    Else
        MsgBox "Nenhnum erro reportado.  Recarregue os dados para verificar se há erros novamente."
    End If


Exit_Report_Errors_Dll_Click:
    Exit Sub

Err_Report_Errors_Dll_Click:
    If Err <> 2501 And Err <> 2465 Then '2051 = Err. The open report action was cancelled
        MsgBox "Erros não relacionados durante verificação.  '" & Err.Number & "' = " &
Err.Description
    End If
    Resume Exit_Report_Errors_Dll_Click
End Sub

Private Sub Report_Errors_NonCritical_Click()
    On Error GoTo Err_Report_Errors_NonCritical_Click
    Dim dbsCurrent As DAO.Database
    Dim rstCountBad As DAO.Recordset
    Dim intErrorCount As Integer

    Set dbsCurrent = CurrentDB()
    Set rstCountBad = dbsCurrent.OpenRecordset("SELECT * " & "FROM MP_InternalDataErrorLog,
MP_InternalDataErrorCode " & "WHERE MP_InternalDataErrorLog.ErrorID = MP_InternalDataErrorCode.ErrorID
" & "AND Importance < 0;")
    If (rstCountBad.RecordCount = 0) Then
        intErrorCount = 0
    Else
        rstCountBad.MoveLast
        intErrorCount = rstCountBad.RecordCount
    End If
    rstCountBad.Close
    Set rstCountBad = Nothing
    Set dbsCurrent = Nothing

    If (intErrorCount > 0) Then
        On Error Resume Next
        DoCmd.Close acReport, "MP_NonCriticalError_Report"
        On Error GoTo Err_Report_Errors_NonCritical_Click
```

```
        DoCmd.Minimize
        DoCmd.OpenReport "MP_NonCriticalError_Report", acViewPreview
    Else
        MsgBox "Nenhnum erro reportado.  Recarregue os dados para verificar se há erros novamente."
    End If


Exit_Report_Errors_NonCritical_Click:
    Exit Sub

Err_Report_Errors_NonCritical_Click:
    If Err <> 2501 And Err <> 2465 Then '2051 = Err. The open report action was cancelled
        MsgBox "Erros não relacionados durante verificação.  '" & Err.Number & "' = " &
Err.Description
    End If
    Resume Exit_Report_Errors_NonCritical_Click
End Sub

Private Sub Report_MRP_Ouput_Click()
On Error GoTo Err_Report_MRP_Ouput_Click
    Dim returnVal As Integer
    If (PlannerObj Is Nothing) Then
        MsgBox "Os dados não foram preenchidos corretamente.  " & vbCrLf & "Corrija os erros, e então
retorne para este formulário."
        GoTo Exit_Report_MRP_Ouput_Click
    End If
    If (Not PlannerObj.IsRunMRPRan) Then
        MsgBox "Os resultados ainda não foram computados. Por favor clique no botão 'Computar'."
        GoTo Exit_Report_MRP_Ouput_Click
    End If
    If (PlannerObj.NonCriticalErrorCount > 0) Then
        MsgBox "Erros não-críticos foram encontrados. Por favor vá para a guia 'Erros' para verificar
os dados rejeitados."
    End If

    DoCmd.Hourglass True
    returnVal = PlannerObj.GetMRPOutput()
    Me.Refresh
    DoCmd.Hourglass False
    If returnVal <> 0 Then GoTo PlannerErr_Report_MRP_Ouput_Click

    If (IsNull(Me.ItemLookUP)) Then
        If (Me.fraProductID = 1) Then         'Show by Period
            DoCmd.Minimize
            DoCmd.OpenReport "MP_MRPOutput_byPeriod_Report", acViewPreview
        ElseIf (Me.fraProductID = 2) Then       'Show by Product/Item
            DoCmd.Minimize
            DoCmd.OpenReport "MP_MRPOutput_byProduct_Report", acViewPreview
        End If
    Else
        If (Me.fraProductID = 1) Then         'Show by Period
            DoCmd.Minimize
            DoCmd.OpenReport "MP_MRPOutput_byPeriod_Report_Filtered", acViewPreview
        ElseIf (Me.fraProductID = 2) Then       'Show by Product/Item
            DoCmd.Minimize
            DoCmd.OpenReport "MP_MRPOutput_byProduct_Report_Filtered", acViewPreview
        End If
    End If

Exit_Report_MRP_Ouput_Click:
    DoCmd.Hourglass False
    Exit Sub

Err_Report_MRP_Ouput_Click:
    If Err <> 2501 And Err <> 2465 Then '2051 = Err. The open report action was cancelled
        MsgBox Err.Description
    End If
    Resume Exit_Report_MRP_Ouput_Click

PlannerErr_Report_MRP_Ouput_Click:
    MsgBox "" & conEBPlannerDLLCopyright & " Erro = " & returnVal & ".  Não é possível exibir o
Relatório."
    Call Unload_PlannerObj
    Resume Exit_Report_MRP_Ouput_Click
End Sub


Private Sub Report_POsDue_Click()
On Error GoTo Err_Report_POsDue_Click
    Dim returnVal As Integer
    If (PlannerObj Is Nothing) Then
```

```
        MsgBox "Os dados não foram preenchidos corretamente.  " & vbCrLf & "Corrija os erros, e então
 retorne para este formulário."
        GoTo Exit_Report_POsDue_Click
    End If
    If (Not PlannerObj.IsRunMRPRan) Then
        MsgBox "Os resultados ainda não foram computados. Por favor clique no botão 'Computar'."
        GoTo Exit_Report_POsDue_Click
    End If
    If (PlannerObj.NonCriticalErrorCount > 0) Then
        MsgBox "Erros não-críticos foram encontrados. Por favor vá para a guia 'Erros' para verificar
 os dados rejeitados."
    End If

    DoCmd.Hourglass True
    returnVal = PlannerObj.GetPOsDue(cmbPO_Periods.Value)
    DoCmd.Hourglass False
    returnVal = 0
    If returnVal <> 0 Then GoTo PlannerErr_Report_POsDue_Click

    If (IsNull(Me.ItemLookUP)) Then
        DoCmd.Minimize
        DoCmd.OpenReport "MP_POsDue_Report", acViewPreview
    Else
        DoCmd.Minimize
        DoCmd.OpenReport "MP_POsDue_Report_Filtered", acViewPreview
    End If

Exit_Report_POsDue_Click:
    DoCmd.Hourglass False
    Exit Sub

Err_Report_POsDue_Click:
    If Err <> 2501 And Err <> 2465 Then '2051 = Err. The open report action was cancelled
        MsgBox Err.Description
    End If
    Resume Exit_Report_POsDue_Click

PlannerErr_Report_POsDue_Click:
    MsgBox "" & conEBPlannerDLLCopyright & " Erro = " & returnVal & ".  Não é possível exibir o
Relatório."
    Call Unload_PlannerObj
    Resume Exit_Report_POsDue_Click
End Sub

Private Sub Report_WOsDue_Click()
On Error GoTo Err_Report_WOsDue_Click
    Dim returnVal As Integer
    If (PlannerObj Is Nothing) Then
        MsgBox "Os dados não foram preenchidos corretamente.  " & vbCrLf & "Corrija os erros, e então
retorne para este formulário."
        GoTo Exit_Report_WOsDue_Click
    End If
    If (Not PlannerObj.IsRunMRPRan) Then
        MsgBox "Os resultados ainda não foram computados. Por favor clique no botão 'Computar'."
        GoTo Exit_Report_WOsDue_Click
    End If
    If (PlannerObj.NonCriticalErrorCount > 0) Then
        MsgBox "Erros não-críticos foram encontrados. Por favor vá para a guia 'Erros' para verificar
os dados rejeitados."
    End If

    DoCmd.Hourglass True
    returnVal = PlannerObj.GetWOsDue(cmbWO_Periods.Value)
    DoCmd.Hourglass False
    If returnVal <> 0 Then GoTo PlannerErr_Report_WOsDue_Click

    If (IsNull(Me.ItemLookUP)) Then
        DoCmd.Minimize
        DoCmd.OpenReport "MP_WOsDue_Report", acViewPreview
    Else
        DoCmd.Minimize
        DoCmd.OpenReport "MP_WOsDue_Report_Filtered", acViewPreview
    End If

Exit_Report_WOsDue_Click:
    DoCmd.Hourglass False
    Exit Sub

Err_Report_WOsDue_Click:
    If Err <> 2501 And Err <> 2465 Then '2051 = Err. The open report action was cancelled
        MsgBox Err.Description
```

```
    End If
    Resume Exit_Report_WOsDue_Click

PlannerErr_Report_WOsDue_Click:
    MsgBox "" & conEBPlannerDLLCopyright & " Erro = " & returnVal & ".  Não é possível exibir o
Relatório."
    Call Unload_PlannerObj
    Resume Exit_Report_WOsDue_Click
End Sub

Private Sub Load_PlannerObj()
    On Error GoTo Err_Load_PlannerObj
    If (boolErrorFindingSchedulingPlan) Then
        MsgBox "Não foi possível carregar os dados.  Ocorreu um erro ao abrir o formulário, ao
carregar a identificação do Planejamento." & vbCrLf & vbCrLf & "Por favor volte ao formulário de
entrada de dados e corrija as informações."
        GoTo Exit_Load_PlannerObj
    End If
    Dim intReturnVal As Integer

    DoCmd.Hourglass True
    If (intAccessVersion = conAcc97) Then
        DoCmd.OpenForm conLoadingMsgForm
    End If
    '****Loading Planner Code****
    If (PlannerObj Is Nothing) Then
        Set PlannerObj = CreateObject("EB.EBPlannerDLL")
    End If

    PlannerObj.IncludeComponents = Me.chkComponents
    PlannerObj.IncludeDemand = Me.chkDemand
    PlannerObj.IncludePOs = Me.chkPOs
    PlannerObj.IncludeWOs = Me.chkWOs
    PlannerObj.IncludeSOs = Me.chkSOs
    PlannerObj.IncludeInventory = Me.chkInventory
    PlannerObj.AccessVersion = intAccessVersion
    PlannerObj.DatabaseCurrent = CurrentDB()
    PlannerObj.SchedulingName = Nz(DLookup("SchedulingPlanName", "SchedulingPlans", "SchedulingPlanID
= " & lngSchedulingPlanID), "")
    intReturnVal = PlannerObj.ReLoadData(Me.fraDataSource)
    '****Finished Loading Planner****
    DoCmd.Hourglass False
    If (intReturnVal <> 0) Then
        GoTo Exit_Load_PlannerObj
    End If

    If (intAccessVersion = conAcc97) Then
        If (IsLoaded(conLoadingMsgForm)) Then
            DoCmd.Close acForm, conLoadingMsgForm
        End If
    End If
    Call MakeABeep

    If (Not PlannerObj.IsDLLFilled) Then
        If (PlannerObj.CriticalErrorCount > 0) Then
            MsgBox "Erros Críticos foram encontrados. Por favor vá para a guia 'Erros' no formulário a
seguir e corrija todos os erros reportados."
            GoTo Exit_Load_PlannerObj
        End If
    End If
    If (PlannerObj.NonCriticalErrorCount > 0) Then
        MsgBox "Erros não-críticos foram encontrados. Por favor vá para a guia 'Erros' para verificar
os dados rejeitados."
    End If
    MsgBox "Carregamento de Dados para o Planejador foi completado. Clique no botão 'Computar' para
calcular os resultados."

Exit_Load_PlannerObj:
    DoCmd.Hourglass False
    Exit Sub
Err_Load_PlannerObj:
    DoCmd.Hourglass False
    If (intAccessVersion = conAcc97) Then
        If (IsLoaded(conLoadingMsgForm)) Then
            DoCmd.Close acForm, conLoadingMsgForm
        End If
    End If
    MsgBox "Erro '" & Err.Number & "' durante o carregamento dos dados no " & conEBPlannerDLLCopyright
    Call Unload_PlannerObj
    Resume Exit_Load_PlannerObj
Planner_Err_Load_PlannerObj:
```

```
    DoCmd.Hourglass False
    If (intAccessVersion = conAcc97) Then
        If (IsLoaded(conLoadingMsgForm)) Then
            DoCmd.Close acForm, conLoadingMsgForm
        End If
    End If
    MsgBox "Erro '" & intReturnVal & "' ao tentar carregar os dados. As informações não foram
carregadas."
    Call Unload_PlannerObj
    GoTo Exit_Load_PlannerObj
End Sub

Private Sub Compute_PlannerObj()
    On Error GoTo Err_Compute_PlannerObj
    Dim intReturnVal As Integer
    If (Not boolErrorFindingSchedulingPlan And Not (PlannerObj Is Nothing)) Then
        If (Not (PlannerObj.CriticalErrorCount > 0 Or PlannerObj.DllErrorCount > 0)) Then
            If (PlannerObj.IsDLLFilled) Then
                If (PlannerObj.NonCriticalErrorCount > 0) Then
                    MsgBox "Erros não-críticos foram encontrados. Por favor vá para a guia 'Erros'
para verificar os dados rejeitados."
                End If
                DoCmd.Hourglass True
                intReturnVal = PlannerObj.RunMRP
                DoCmd.Hourglass False
                If (intReturnVal <> 0) Then GoTo Planner_Err_Compute_PlannerObj
                MsgBox conEBPlannerDLLCopyright & " já computou os resultados.  Não foram detectados
erros críticos."
            Else
                MsgBox "Primeiro, você deve carregar as informações no " & conEBPlannerDLLCopyright &
".  Volte para o formulário de Entrada de Dados e tente novamente."
            End If
        Else
            If (PlannerObj.CriticalErrorCount > 0) Then
                MsgBox "Foram encontrados erros. Por favor vá para a guia 'Erros' no formulário a
seguir e corrija todos os erros reportados."

            ElseIf (PlannerObj.DllErrorCount > 0) Then
                MsgBox "Foram encontrados erros. Por favor vá para a guia 'Erros' no formulário a
seguir e corrija todos os erros reportados."
            End If
        End If
    Else
        MsgBox "Os dados não foram preenchidos corretamente.  " & vbCrLf & "Corrija os erros, e então
retorne para este formulário."
    End If

Exit_Compute_PlannerObj:
    DoCmd.Hourglass False
    Exit Sub

Err_Compute_PlannerObj:
    DoCmd.Hourglass False
    MsgBox "Erro '" & Err.Number & "' durante a execução do planejamento MRP em " &
conEBPlannerDLLCopyright & ".  Os resultados não foram computados."
    Call Unload_PlannerObj
    Resume Exit_Compute_PlannerObj

Planner_Err_Compute_PlannerObj:
    DoCmd.Hourglass False
    MsgBox "Erro '" & intReturnVal & "' ocorreu ao tentar computar o MRP.  Os resultados não puderam
ser computados."
    Call Unload_PlannerObj
    GoTo Exit_Compute_PlannerObj
End Sub


Private Sub Output_PlannerObj()
    On Error GoTo Err_Output_PlannerObj
    Dim intReturnVal As Integer
    If (Not boolErrorFindingSchedulingPlan And Not (PlannerObj Is Nothing)) Then
        If (PlannerObj.IsRunMRPRan) Then
            DoCmd.Hourglass True
            intReturnVal = PlannerObj.GetMRPOutput()
            If (intReturnVal <> 0) Then GoTo Planner_Err_Output_PlannerObj
            Me.Refresh
            intReturnVal = PlannerObj.GetPOsDue(Me.cmbPO_Periods)
            If (intReturnVal <> 0) Then GoTo Planner_Err_Output_PlannerObj
            intReturnVal = PlannerObj.GetWOsDue(Me.cmbWO_Periods)
            If (intReturnVal <> 0) Then GoTo Planner_Err_Output_PlannerObj
            DoCmd.Hourglass False
```

```
            Call MakeABeep
            MsgBox "Finalizado o processo de recuperação de dados a partir de " &
conEBPlannerDLLCopyright
        Else
            MsgBox "Os resultados ainda não foram computados. Por favor clique no botão 'Computar'."
        End If
    Else
        MsgBox "Não é possível computar devido a erros anteriores."
    End If

Exit_Output_PlannerObj:
    DoCmd.Hourglass False
    Exit Sub

Err_Output_PlannerObj:
    DoCmd.Hourglass False
    MsgBox "Erro '" & Err.Number & "' ao carregar a saída, ocorreu em " & conEBPlannerDLLCopyright &
".  Os Resultados não puderam ser recuperados."
    Call Unload_PlannerObj
    Resume Exit_Output_PlannerObj

Planner_Err_Output_PlannerObj:
    DoCmd.Hourglass False
    MsgBox "Erro '" & intReturnVal & "' ocorreu ao tentar carregar a saída de " &
conEBPlannerDLLCopyright & ".  Os Resultados não puderam ser recuperados."
    Call Unload_PlannerObj
    GoTo Exit_Output_PlannerObj
End Sub


Private Sub Unload_PlannerObj()
    If (Not (PlannerObj Is Nothing)) Then
        On Error Resume Next
        Call PlannerObj.Unload
        Set PlannerObj = Nothing
    End If
End Sub

Private Sub Load_MeOpenArgs()
    Dim strParam As String
    Dim strParamID As String
    Dim strParamVal As String
    Dim strRemaining As String
    Dim strSchedulingPlanName As String
    Dim intNumPOsDuePeriods As Integer
    Dim intNumWOsDuePeriods As Integer
    Dim intDemandPeriodCount As Integer
    Dim intSourceID As Integer

    strRemaining = strMeOpenArgs
    boolErrorFindingSchedulingPlan = False
    strSchedulingPlanName = ""
    intDemandPeriodCount = 1


    Do While (Len(strRemaining) > 0)
        strParam = ParsePop(strRemaining, "&")
        If (Len(strParam) > 0) Then
            strParamVal = strParam
            strParamID = ParsePop(strParamVal, "=")
            Select Case (strParamID)
                Case "SP": 'SchedulingPlanID
                    If (Len(strParamVal) < 1) Then
                        MsgBox "Não foi enviado nenhum Planejamento." & vbCrLf & vbCrLf & "Não há
dados para serem exibidos."
                        boolErrorFindingSchedulingPlan = True
                        intDemandPeriodCount = 1
                        Call Update_DemandPeriodsAhead(intDemandPeriodCount)
                    ElseIf (IsNull(DLookup("SchedulingPlanID", "SchedulingPlans", "SchedulingPlanID =
" & strParamVal))) Then
                        MsgBox "O Planejamento enviado não é válido." & vbCrLf & vbCrLf & "Não há
dados para serem exibidos."
                        boolErrorFindingSchedulingPlan = True
                        intDemandPeriodCount = 1
                        Call Update_DemandPeriodsAhead(intDemandPeriodCount)
                    Else
                        lngSchedulingPlanID = Val(strParamVal)
                        intDemandPeriodCount = Nz(DLookup("DemandPeriodCount", "SchedulingPlans",
"SchedulingPlanID = " & lngSchedulingPlanID), 1)
                        Call Update_DemandPeriodsAhead(intDemandPeriodCount)
                    End If
```

```
                Case "PD": 'POsDue cmbBox
                    If (IsNumeric(strParamVal)) Then
                        If (Val(strParamVal) < intDemandPeriodCount And Val(strParamVal) > 0) Then
                            Me.cmbPO_Periods = Val(strParamVal)
                        Else
                            Me.cmbPO_Periods.Value = intDemandPeriodCount
                        End If
                    Else
                        Me.cmbPO_Periods.Value = intDemandPeriodCount
                    End If
                Case "WD": 'WOsDue cmbBox
                    If (IsNumeric(strParamVal)) Then
                        If (Val(strParamVal) < intDemandPeriodCount And Val(strParamVal) > 0) Then
                            Me.cmbWO_Periods = Val(strParamVal)
                        Else
                            Me.cmbWO_Periods.Value = intDemandPeriodCount
                        End If
                    Else
                        Me.cmbWO_Periods.Value = intDemandPeriodCount
                    End If
                Case "DS": 'DataSource
                    If Not (strParamVal = CStr(conSourceQueries) Or strParamVal =
CStr(conSourceFiles)) Then
                        intSourceID = consourcedefault
                    Else
                        intSourceID = Val(strParamVal)
                    End If
                    Me.fraDataSource = intSourceID
                Case "C":  'Components chkbox
                    If (strParamVal = "F") Then
                        Me.chkComponents = False
                    ElseIf (strParamVal = "T") Then
                        Me.chkComponents = True
                    End If
                Case "D":  'Demand chkbox
                    If (strParamVal = "F") Then
                        Me.chkDemand = False
                    ElseIf (strParamVal = "T") Then
                        Me.chkDemand = True
                    End If
                Case "P":  'POs chkbox
                    If (strParamVal = "F") Then
                        Me.chkPOs = False
                    ElseIf (strParamVal = "T") Then
                        Me.chkPOs = True
                    End If
                Case "W":  'WOs chkbox
                    If (strParamVal = "F") Then
                        Me.chkWOs = False
                    ElseIf (strParamVal = "T") Then
                        Me.chkWOs = True
                    End If
                Case "S":  'SOs chkbox
                    If (strParamVal = "F") Then
                        Me.chkSOs = False
                    ElseIf (strParamVal = "T") Then
                        Me.chkSOs = True
                    End If
                Case "I":  'Inventory chkbox
                    If (strParamVal = "F") Then
                        Me.chkInventory = False
                    ElseIf (strParamVal = "T") Then
                        Me.chkInventory = True
                    End If
                Case Else:
                    'Do nothing
            End Select
        End If

    Loop
    If (boolErrorFindingSchedulingPlan = False) Then
        strSchedulingPlanName = Nz(DLookup("SchedulingPlanName", "SchedulingPlans", "SchedulingPlanID
= " & lngSchedulingPlanID), "")
        If (Len(strSchedulingPlanName) = 0) Then
            MsgBox "O nome do Planejamento não é válido." & vbCrLf & "Volte para o formulário de
Entrada de Dados e tente novamente." & vbCrLf & vbCrLf & "Não há dados para serem exibidos."
            boolErrorFindingSchedulingPlan = True
        End If
    End If
End Sub
```

```
Private Sub Update_DemandPeriodsAhead(newValue As Integer)
    Dim strNewRowSource As String
    Dim intLoop As Integer

    Me.cmbWO_Periods = "1"
    Me.cmbPO_Periods = "1"
    strNewRowSource = "1"
    For intLoop = 2 To newValue
        strNewRowSource = strNewRowSource & ";" & intLoop
    Next intLoop

    Me.cmbWO_Periods.RowSource = strNewRowSource
    Me.cmbPO_Periods.RowSource = strNewRowSource

    Me.cmbWO_Periods.Requery
    Me.cmbPO_Periods.Requery

    Me.cmbWO_Periods = CStr(newValue)
    Me.cmbPO_Periods = CStr(newValue)
End Sub

Private Sub EmptyTempTables()
    Dim dbsCurrent As DAO.Database
    Set dbsCurrent = CurrentDB()
    dbsCurrent.Execute ("DELETE * FROM MP_MRPOutput")
    dbsCurrent.Execute ("DELETE * FROM MP_WOsDue")
    dbsCurrent.Execute ("DELETE * FROM MP_POsDue")
    dbsCurrent.Execute ("DELETE * FROM MP_TimeDefinition")
    dbsCurrent.Execute ("DELETE * FROM MP_Products")
    dbsCurrent.Execute ("DELETE * FROM MP_Where_Used")
    dbsCurrent.Execute ("DELETE * FROM MP_Demand")
    dbsCurrent.Execute ("DELETE * FROM MP_Inventory")
    dbsCurrent.Execute ("DELETE * FROM MP_PurchaseOrders")
    dbsCurrent.Execute ("DELETE * FROM MP_SalesOrders")
    dbsCurrent.Execute ("DELETE * FROM MP_WorkOrders")
    Set dbsCurrent = Nothing
End Sub

Private Sub Save_POsDue_Click()
    On Error GoTo Err_Save_POsDue_Click

    Dim returnVal As Integer
    If (PlannerObj Is Nothing) Then
        MsgBox "Os dados não foram preenchidos corretamente.  " & vbCrLf & "Corrija os erros, e então
retorne para este formulário."
        GoTo Exit_Save_POsDue_Click
    End If
    If (Not PlannerObj.IsRunMRPRan) Then
        MsgBox "Os resultados ainda não foram computados. Por favor clique no botão 'Computar'."
        GoTo Exit_Save_POsDue_Click
    End If

    DoCmd.Hourglass True
    returnVal = PlannerObj.SaveToFilePOsDue(Me.cmbPO_Periods)
    DoCmd.Hourglass False
    If returnVal <> 0 Then GoTo PlannerErr_Save_POsDue_Click
    MsgBox ("O Pedido de Compra foi gravado com sucesso!")


Exit_Save_POsDue_Click:
    DoCmd.Hourglass False
    Exit Sub

Err_Save_POsDue_Click:
    MsgBox "Erro Número '" & Err.Number & "' " & Err.Description
    GoTo Exit_Save_POsDue_Click

PlannerErr_Save_POsDue_Click:
    MsgBox "Erro no Planejador = " & returnVal & ".  Não é possível completar este procedimento."
    Call Unload_PlannerObj
    GoTo Exit_Save_POsDue_Click
End Sub

Private Sub Save_WOsDue_Click()
    On Error GoTo Err_Save_WOsDue_Click

    Dim returnVal As Integer
    If (PlannerObj Is Nothing) Then
        MsgBox "Os dados não foram preenchidos corretamente.  " & vbCrLf & "Corrija os erros, e então
retorne para este formulário."
```

```
        GoTo Exit_Save_WOsDue_Click
    End If
    If (Not PlannerObj.IsRunMRPRun) Then
        MsgBox "Os Resultados ainda não foram computados. Por favor clique no botão 'Computar'."
        GoTo Exit_Save_WOsDue_Click
    End If

    DoCmd.Hourglass True
    returnVal = PlannerObj.SaveToFileWOsDue(Me.cmbWO_Periods)
    DoCmd.Hourglass False
    If returnVal <> 0 Then GoTo PlannerErr_Save_WOsDue_Click
    MsgBox ("A Ordem de Produção foi gravada com sucesso!")


Exit_Save_WOsDue_Click:
    DoCmd.Hourglass False
    Exit Sub

Err_Save_WOsDue_Click:
    MsgBox "Erro Número '" & Err.Number & "' " & Err.Description
    GoTo Exit_Save_WOsDue_Click

PlannerErr_Save_WOsDue_Click:
    MsgBox "Erro no Planejador = " & returnVal & ".  Não foi possível completar este procedimento."
    GoTo Exit_Save_WOsDue_Click
End Sub
Private Sub SairForm_Click()
On Error GoTo Err_SairForm_Click


    DoCmd.Close
    DoCmd.OpenForm "Menu Inicial"

Exit_SairForm_Click:
    Exit Sub

Err_SairForm_Click:
    MsgBox Err.Description
    Resume Exit_SairForm_Click

End Sub

Private Sub MakeABeep()
End Sub
```

SIMULAÇÃO MRP

```
Option Compare Database
Option Explicit
Public strValue As String
Dim DBObject As New EB.EBdefinitionsClass
Dim XObject As New EB.EBexplodeClass
Dim IObject As New EBupdatedynamiccountClass

Private Sub BuildPlan_LostFocus()
On Error GoTo Err_BuildPlan_LostFocus
Dim dbsCurrent As DAO.Database
Dim strValue As String
Dim mLevel As Integer
Dim strMsgbox As String
Set dbsCurrent = CurrentDB()
strValue = Me!ProductID
mLevel = 7
XObject.mlevels = mLevel
XObject.ParentID = strValue
XObject.DatabaseCurrent = dbsCurrent
XObject.Winxx = DLookup("[Windows]", "[VersionControl]")
XObject.Explode
Dim intLocation As Integer

intLocation = 1
strValue = DLookup("[InventoryType]", "[Customers]", "[Customers]![CustomerID] = " & intLocation & "")

IObject.InventoryType = strValue
IObject.InventoryLocation = intLocation
IObject.InventoryEndDate = Now()
IObject.DatabaseCurrent = dbsCurrent
IObject.Winxx = DLookup("[Windows]", "[VersionControl]")
IObject.UpdateDynamicCount

Exit_BuildPlan_LostFocus:
    Exit Sub

Err_BuildPlan_LostFocus:
    strMsgbox = "Erro n°'" & Err.Number & "' ocorreu.   " & Err.Description & "  "
    If (Err.Number = 3021) Then
        strMsgbox = strMsgbox & "  A causa provável é que este produto não tem filhos."
    End If
    MsgBox strMsgbox
    Resume Exit_BuildPlan_LostFocus
End Sub

Private Sub Command33_Click()

If IsNull([Forms]![MRP_lite]![ProductID]) Then
    MsgBox "Por favor selecione um produto antes."
    Exit Sub
End If

If IsNull([Forms]![MRP_lite]![BuildPlan]) Then
    MsgBox "Por favor insira a quantidade a ser produzida."
    Exit Sub
End If

Call AllocatedCounts

Dim dbs As Database
Dim rst As Recordset
Dim levelCurrent As Double
Dim levelPrevious As Double
Dim levelLoop As Double
Dim parentQuantity As Double

DoCmd.SetWarnings False
DoCmd.OpenQuery "MRPlite_Whatif_ATP_Query", acNormal, acEdit

Set dbs = CurrentDB
Set rst = dbs.OpenRecordset("MRPlite_Whatif_ATP", dbOpenTable)

Do While Not (rst.EOF)
    levelCurrent = rst!Level
    parentQuantity = rst!QtyInParent
    rst.MoveNext
    If Not (rst.EOF) Then
        levelPrevious = levelCurrent
```

```
        levelCurrent = rst!Level
        If (levelCurrent > levelPrevious) Then
            levelLoop = levelCurrent
            Set rst = Filter_Recordset_MultiLevelBOM(rst, parentQuantity, levelLoop)
            If (Not (rst.EOF)) Then
            levelCurrent = rst!Level
            End If
        End If
    Else
        Exit Do
    End If
Loop

On Error Resume Next
DoCmd.Minimize
DoCmd.OpenReport "Exploded_BOM_MRP_available", acViewPreview
On Error GoTo 0
DoCmd.SetWarnings True

End Sub

Private Sub Command35_Click()

If IsNull([Forms]![MRP_lite]![ProductID]) Then
    MsgBox "For favor selecione um produto antes."
    Exit Sub
End If

If IsNull([Forms]![MRP_lite]![BuildPlan]) Then
    MsgBox "Por favor insira a quantidade a ser produzida."
    Exit Sub
End If

DoCmd.Hourglass True
DoCmd.SetWarnings False
DoCmd.OpenQuery "OnOrder make table", acNormal, acEdit
On Error Resume Next
DoCmd.Minimize
DoCmd.OpenReport "Exploded_BOM_MRP_OrderStatus", acViewPreview
On Error GoTo 0
DoCmd.SetWarnings True
DoCmd.Hourglass False

End Sub

Private Sub Command36_Click()

If IsNull([Forms]![MRP_lite]![ProductID]) Then
    MsgBox "Por favor selecione um produto antes."
    Exit Sub
End If

If IsNull([Forms]![MRP_lite]![BuildPlan]) Then
    MsgBox "Por favor insira a quantidade a ser produzida."
    Exit Sub
End If

Dim dbs As Database
Dim rst As Recordset
Dim levelCurrent As Double
Dim levelPrevious As Double
Dim levelLoop As Double
Dim parentQuantity As Double

DoCmd.Hourglass True
DoCmd.SetWarnings False
DoCmd.OpenQuery "MRPlite_Whatif_OnHand_Query", acNormal, acEdit

Set dbs = CurrentDB
Set rst = dbs.OpenRecordset("MRPlite_Whatif_OnHand", dbOpenTable)

Do While Not (rst.EOF)
    levelCurrent = rst!Level
    parentQuantity = rst!QtyInParent
    rst.MoveNext
    If Not (rst.EOF) Then
        levelPrevious = levelCurrent
        levelCurrent = rst!Level
        If (levelCurrent > levelPrevious) Then
            levelLoop = levelCurrent
            Set rst = Filter_Recordset_MultiLevelBOM(rst, parentQuantity, levelLoop)
```

```
            If (Not (rst.EOF)) Then
            levelCurrent = rst!Level
            End If
        End If
    Else
        Exit Do
    End If
Loop

On Error Resume Next
DoCmd.Minimize
DoCmd.OpenReport "Exploded_BOM_MRP_available1", acViewPreview
On Error GoTo 0
DoCmd.SetWarnings True
DoCmd.Hourglass False

End Sub


Private Sub Form_Load()
Dim dbsCurrent As DAO.Database
Dim strValue As String
Set dbsCurrent = CurrentDB()
strValue = DLookup("[Userdefinable 3]", "My Company Information")
strValue = strValue & "DbM_fe.mdb"
DBObject.DatabaseCurrent = dbsCurrent
DBObject.DatabaseName = strValue
    DBObject.SetDatabase
End Sub

Private Sub Form_Unload(Cancel As Integer)
Set DBObject = Nothing
Set XObject = Nothing
Set IObject = Nothing
End Sub

Private Sub Shortages_Click()
On Error GoTo Err_Shortages_Click
    Dim dbs As Database
    Dim rst As Recordset
    Dim Shortage As Long
    Dim levelParent As Integer
    Dim levelChild As Integer
    Dim strMsgbox As String

    If IsNull([Forms]![MRP_lite]![ProductID]) Then
        MsgBox "Por favor selecione um produto antes."
        Exit Sub
    End If

    If IsNull([Forms]![MRP_lite]![BuildPlan]) Then
        MsgBox "Por favor insira a quantidade a ser produzida."
        Exit Sub
    End If

    DoCmd.SetWarnings False
    DoCmd.OpenQuery "OnOrder make table", acNormal, acEdit
    DoCmd.OpenQuery "MRPlite_Whatif_Shortages_Query1", acNormal, acEdit

    Set dbs = CurrentDB
    Set rst = dbs.OpenRecordset("MRPlite_Whatif_Shortages", dbOpenTable)

Dim levelCurrent As Double
Dim levelPrevious As Double
Dim levelLoop As Double
Dim parentQuantity As Double

Do While Not (rst.EOF)
    levelCurrent = rst!Level
    parentQuantity = rst!QtyInParent
    rst.MoveNext
    If Not (rst.EOF) Then
        levelPrevious = levelCurrent
        levelCurrent = rst!Level
        If (levelCurrent > levelPrevious) Then
            levelLoop = levelCurrent
            Set rst = Filter_Recordset_Shortages(rst, parentQuantity, levelLoop)
            If (Not (rst.EOF)) Then
            levelCurrent = rst!Level
            End If
        End If
```

```
        Else
            Exit Do
        End If
    Loop

        rst.MoveFirst

        Do While Not (rst.EOF)
            Shortage = rst!Shortage
            levelParent = rst!Level
            If Shortage > 0 Then
                rst.MoveNext
                If Not (rst.EOF) Then
                    levelChild = rst!Level
                Else
                    Exit Do
                End If
                Do While levelChild > levelParent
                    rst.Delete
                    rst.MoveNext
                    If Not (rst.EOF) Then
                        levelChild = rst!Level
                    Else
                        Exit Do
                    End If
                Loop
            ElseIf Not (rst.EOF) Then
                rst.MoveNext
            Else
                Exit Do
            End If
        Loop

        On Error Resume Next
        DoCmd.Minimize
        DoCmd.OpenReport "Exploded_BOM_Shortages", acPreview, "", ""
        On Error GoTo 0
        DoCmd.SetWarnings True

Exit_Shortages_Click:
    Exit Sub

Err_Shortages_Click:
    strMsgbox = "Erro n°'" & Err.Number & "' ocorreu.    " & Err.Description & "  "
    If (Err.Number = 3021) Then
        strMsgbox = strMsgbox & "  A causa provável é que este produto não tem filhos."
    End If
    MsgBox strMsgbox
    Resume Exit_Shortages_Click
End Sub

Function Filter_Recordset_MultiLevelBOM(temp_rst As Recordset, aQuantity As Double, aLevelLoop As
Double) As Recordset

DoCmd.Hourglass True

Dim levelCurrent As Double
Dim levelPrevious As Double
Dim levelLoop As Double
Dim parentQuantity As Double

levelCurrent = temp_rst!Level
Do While (levelCurrent >= aLevelLoop)
    With temp_rst
    .Edit
    !QtyInParent = !QtyInParent * aQuantity
    !BuildPlan1 = Me.BuildPlan * !QtyInParent
    .update
    parentQuantity = temp_rst!QtyInParent
    .MoveNext
    End With
    If Not (temp_rst.EOF) Then
        levelPrevious = levelCurrent
        levelCurrent = temp_rst!Level
        If (levelCurrent > levelPrevious) Then
            levelLoop = levelCurrent
            Set temp_rst = Filter_Recordset_MultiLevelBOM(temp_rst, parentQuantity, levelLoop)
            If (Not (temp_rst.EOF)) Then
            levelCurrent = temp_rst!Level
            Else
            Exit Do
```

```
            End If

        End If
    Else
        Exit Do
    End If
Loop

Set Filter_Recordset_MultiLevelBOM = temp_rst

DoCmd.Hourglass False
End Function

Function Filter_Recordset_Shortages(temp_rst As Recordset, aQuantity As Double, aLevelLoop As Double)
As Recordset

Dim levelCurrent As Double
Dim levelPrevious As Double
Dim levelLoop As Double
Dim parentQuantity As Double

levelCurrent = temp_rst!Level
Do While (levelCurrent >= aLevelLoop)
    With temp_rst
    .Edit
    !QtyInParent = !QtyInParent * aQuantity
    !BuildPlan1 = Me.BuildPlan * !QtyInParent
    !Shortage = !ATP - !BuildPlan1
    .update
    parentQuantity = temp_rst!QtyInParent
    .MoveNext
    End With
    If Not (temp_rst.EOF) Then
        levelPrevious = levelCurrent
        levelCurrent = temp_rst!Level
        If (levelCurrent > levelPrevious) Then
            levelLoop = levelCurrent
            Set temp_rst = Filter_Recordset_Shortages(temp_rst, parentQuantity, levelLoop)
            If (Not (temp_rst.EOF)) Then
            levelCurrent = temp_rst!Level
            Else
            Exit Do
            End If
        End If
    Else
        Exit Do
    End If
Loop

Set Filter_Recordset_Shortages = temp_rst

End Function

Private Sub Form_Close()
DoCmd.OpenForm "Menu Inicial"
End Sub
```

IMPORTAÇÃO DE DADOS - MRP

```
Option Compare Database
Option Explicit

Const conTargetPOWOTable = "MP_Import_POWO_tmp"

Private Sub chkPO_AfterUpdate()
    POFullPath.Enabled = chkPO
End Sub

Private Sub chkWO_AfterUpdate()
    WOFullPath.Enabled = chkWO
End Sub

Private Sub cmdImport_Click()
    If chkPO Then
        If Not IsNull(POFullPath) Then
            If POFullPath <> "" And Len(Dir(POFullPath)) > 0 Then
                If Not Import_PO() Then
                    Exit Sub
                End If
                DoCmd.OpenForm "MP_GeneratePOWO", , , , , acDialog, "PO"
            Else
                MsgBox "Caminho inválido para PC's." & vbCrLf & "PC's não serão importados",
vbCritical
            End If
        End If
    End If

    If chkWO Then
        If Not IsNull(WOFullPath) Then
            If WOFullPath <> "" And Len(Dir(WOFullPath)) > 0 Then
                If Not Import_WO() Then
                    Exit Sub
                End If
                DoCmd.OpenForm "MP_GeneratePOWO", , , , , acDialog, "WO"
            Else
                MsgBox "Caminho inválido para OP's." & vbCrLf & "OP's não serão importadas",
vbCritical
            End If
        End If
    End If
End Sub

Private Sub cmdShowErrors_Click()
    If TableExist(CurrentDB, conImportErrorTable) Then
        DoCmd.OpenTable conImportErrorTable, , acReadOnly
    Else
        MsgBox "Nenhum erro foi encontrado."
    End If
End Sub

Private Sub Form_Load()
    Call chkPO_AfterUpdate
    Call chkWO_AfterUpdate
End Sub

Private Sub POBrowse_Click()
    Dim msaof As MSA_OPENFILENAME

    msaof.strDialogTitle = "Encontre o arquivo de 'Pedido de Compra' para importar"
    msaof.strFilter = MSA_CreateFilterString("Text files", "*.txt")

    MSA_GetOpenFileName msaof

    If Trim(msaof.strFullPathReturned) <> "" Then
        POFullPath = Trim(msaof.strFullPathReturned)
        chkPO = True
        Call chkPO_AfterUpdate
    End If
End Sub

Private Sub WOBrowse_Click()
    Dim msaof As MSA_OPENFILENAME

    msaof.strDialogTitle = "Encontre o arquivo de 'Ordem de Produção' para importar"
    msaof.strFilter = MSA_CreateFilterString("Text files", "*.txt")

    MSA_GetOpenFileName msaof
```

```
    If Trim(msaof.strFullPathReturned) <> "" Then
        WOFullPath = Trim(msaof.strFullPathReturned)
        chkWO = True
        Call chkWO_AfterUpdate
    End If
End Sub

Sub FillInProductIDForAllRecord(strTableName As String)
    Dim dbsCurrent As DAO.Database
    Dim rstTemp As DAO.Recordset
    Dim strCurrentProductID As String

    Set dbsCurrent = CurrentDB
    Set rstTemp = dbsCurrent.OpenRecordset(strTableName)
    With rstTemp
        Do While Not .EOF
            If IsNull(![ProductID]) Then
                .Edit
                ![ProductID] = strCurrentProductID
                .update
            Else
                strCurrentProductID = ![ProductID]
            End If
            .MoveNext
        Loop
    End With

    Set rstTemp = Nothing
    Set dbsCurrent = Nothing

End Sub

Function Import_PO() As Boolean
    Import_PO = TxtToTable(POFullPath, conTargetPOWOTable & " Import Specification",
conTargetPOWOTable, acImportFixed, 11, True)
    If Import_PO Then
        Call RemoveFirstXRecords(conTargetPOWOTable, 3)
        Call FillInProductIDForAllRecord(conTargetPOWOTable)
    End If
End Function

Function Import_WO() As Boolean
    Import_WO = TxtToTable(WOFullPath, conTargetPOWOTable & " Import Specification",
conTargetPOWOTable, acImportFixed, 11, True)
    If Import_WO Then
        Call RemoveFirstXRecords(conTargetPOWOTable, 3)
        Call FillInProductIDForAllRecord(conTargetPOWOTable)
    End If
End Function

Private Sub Form_Close()
DoCmd.OpenForm "Menu Inicial"
End Sub
```

CONTAGEM INVENTÁRIO

```
Option Compare Database
Option Explicit
Dim NeedUpdate As Boolean 'Check if the LastPhyysicalCountDate have been updated since the last
modification

Private Sub cmdZeroPhysicalCount_Click()
  If (Not (vbOK = MsgBox("Esta ação vai zerar a contagem física de TODOS os itens.  " _
    & vbCrLf & "Só deve ser feita imediatamente antes de registrar uma nova contagem física." _
    & vbCrLf & "Esta ação não pode ser desfeita automaticamente!!  Prosseguir mesmo assim?",
vbOKCancel, "Zerar Contagem"))) Then
        MsgBox "Processo abortado. Não foram realizadas mudanças na contagem."
        Exit Sub
  End If

  Dim dbsCurrent As DAO.Database
  Set dbsCurrent = CurrentDB()
  dbsCurrent.Execute "UPDATE Products SET PhysicalCount = 0;"
  dbsCurrent.Close
  Set dbsCurrent = Nothing
End Sub

Private Sub Standard_Click()
PriceReference = 1
End Sub

Private Sub Current_Click()
PriceReference = 2
End Sub

Private Sub Average_Click()
PriceReference = 3
End Sub

Private Sub Copy_Click()
Select Case PriceReference
    Case 1
        AvgUnitPrice = Standard
    Case 2
        AvgUnitPrice = Current
    Case 3
        AvgUnitPrice = Average
End Select
End Sub

Private Sub PhysicalCount_AfterUpdate()
NeedUpdate = True
End Sub

Private Sub Form_Current()
ProductIDLookUp = ProductID
End Sub

Private Sub Form_Open(Cancel As Integer)
NeedUpdate = True

End Sub


Private Sub Form_Unload(Cancel As Integer)
If False Then 'NeedUpdate Then  'Feature disabled for now
    Select Case MsgBox("Foram realizadas alterações após a última data modificada.  Você quer
atualizar agora?", vbQuestion + vbYesNoCancel, "Atualização")
        Case vbCancel
            DoCmd.CancelEvent
        Case vbYes
            UpdateNow_Click
        Case Else
        'Do nothing. Continue the to close the form
    End Select
End If
End Sub

Private Sub ProductIDLookUp_AfterUpdate()
If Not IsNull(ProductIDLookUp) Then
    Me.Autoid.SetFocus
    DoCmd.FindRecord (Me.ProductIDLookUp.Column(1))
    ProductIDLookUp = ProductID
    ProductName.SetFocus
End If
```

```
End Sub

Private Sub ProductName_GotFocus()
If Not IsNull(Me.ProductID) Then
    ProductIDLookUp = Me.ProductID
End If
End Sub

Private Sub UpdateNow_Click()
Dim rstCompany As DAO.Recordset
Set rstCompany = CurrentDB.OpenRecordset("My Company Information")
With rstCompany
    .Edit
    ![LastPhysicalCountDate] = Date
    .update
End With
Set rstCompany = Nothing
Me.Refresh
LastModified = Date
NeedUpdate = False
End Sub


Private Sub Form_Close()
DoCmd.OpenForm "Menu Inicial"
End Sub
Private Sub RegAnterior_Click()
On Error GoTo Err_RegAnterior_Click


    DoCmd.GoToRecord , , acPrevious

Exit_RegAnterior_Click:
    Exit Sub

Err_RegAnterior_Click:
    MsgBox Err.Description
    Resume Exit_RegAnterior_Click

End Sub
Private Sub ProximoReg_Click()
On Error GoTo Err_ProximoReg_Click

    DoCmd.GoToRecord , , acNext

Exit_ProximoReg_Click:
    Exit Sub

Err_ProximoReg_Click:
    MsgBox Err.Description
    Resume Exit_ProximoReg_Click

End Sub
```

RECEBIMENTO PC

```
Option Compare Database
Option Explicit

Public strTmp As String
Public strdelete As String
Private Sub Command91_Click()
On Error Resume Next
DoCmd.Minimize
DoCmd.OpenReport "PO_Receive", acViewPreview
End Sub

Private Sub Form_AfterDelConfirm(Status As Integer)
Select Case Status
        Case acDeleteOK
            MsgBox "O Processo ocorreu normalmente."
        Case acDeleteCancel

MsgBox "O usuário cancelou o processo"
        Case acDeleteUserCancel
            MsgBox "O usuário cancelou o processo"
    End Select
End Sub

Private Sub Form_BeforeDelConfirm(Cancel As Integer, Response As Integer)
    Response = acDataErrContinue
    If MsgBox("Apagar este registro?", vbOKCancel) = vbCancel Then
        Cancel = True
        Exit Sub
    End If
    Dim dbsCurrent As DAO.Database
    Dim strPurchaseOrderID As String

     Set dbsCurrent = CurrentDB()

    On Error Resume Next

    strPurchaseOrderID = strdelete
    DoCmd.Hourglass True
    dbsCurrent.Execute "DELETE * FROM [Inventory Transactions] WHERE ([Inventory
Transactions].[PurchaseOrderID] = """ & strPurchaseOrderID & """);"
    DoCmd.Hourglass False

End Sub

Private Sub Form_Current()
    strTmp = iif(IsNull(Me!PurchaseOrderID), "Null", Me!PurchaseOrderID)

End Sub

Private Sub Form_Activate()
On Error GoTo Err_Form_Activate
    DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, , acMenuVer70

Exit_Form_Activate:
    Exit Sub

Err_Form_Activate:
    MsgBox Err.Description
    Resume Exit_Form_Activate
End Sub

Private Sub Form_Delete(Cancel As Integer)
strTmp = Me!PurchaseOrderID
strdelete = strTmp
End Sub

Private Sub Form_LostFocus()
On Error Resume Next
strTmp = Me!PurchaseOrderID
End Sub

Private Sub ItemFind_Click()
If Not IsNull([ItemLookUP]) Then
    Text23.SetFocus
    DoCmd.FindRecord ([ItemLookUP])
    ItemLookUP.SetFocus
Else
    MsgBox "Por favor entre um pedido."
End If
```

```
End Sub

Private Sub SaveReceipts_Click()
Dim dbsCurrent As DAO.Database
Dim iLevel As Integer
Dim kLevel As Integer
Dim mLevel As Integer
Dim iNumber As Integer
Dim rstDMR As DAO.Recordset
Dim rstDMR_sort As DAO.Recordset
Dim wrkJet As DAO.Workspace
Dim strProductID As String
Dim strValue As String
Dim qdfExplode As DAO.QueryDef
Dim jLevel As Integer
Dim dUnitsSold As Double
Dim intValue As Integer

    Set dbsCurrent = CurrentDB()
    strValue = Forms![PO Receiving]!PurchaseOrderID
    If Me!StatusID > 3 And Me!StatusID <> DLookup("[StatusID]", "Status_PO", "[Status]='Drop Ship'")
Then
        MsgBox ("Este PC está fechado!!")
        Exit Sub
    End If
DoCmd.SetWarnings False
dbsCurrent.Execute "INSERT INTO [Inventory Transactions] ( TransactionDate,ProductID," _
& "PurchaseOrderID, UnitsShipped,TransferFromID,CustomerID, UnitPrice, ModelNumber,PP)" _
& "SELECT Now() AS [TDate],[Inventory Transactions].[ProductID]," _
& "[Inventory Transactions].[PurchaseOrderID], [UnitsReceived] AS [Shipd]," _
& "'8' AS [FromID], '1' AS [ToID], [Inventory Transactions].[UnitPrice], " _
& "[Inventory Transactions].[ModelNumber],[Inventory Transactions].[PP]" _
& "FROM [Inventory Transactions]" _
& "WHERE ((([Inventory Transactions].[PurchaseOrderID]) = """ & strValue & """ )) " _
& "AND [Inventory Transactions].[UnitsReceived] <> 0;"

On Error Resume Next
DoCmd.DeleteObject acTable, "DynamicInventory"
 dbsCurrent.Execute "SELECT [ProductID]," _
& "(Sum(IIf([CustomerID] =1,[UnitsShipped],0))) AS [Recd]," _
& "(Sum(IIf([TransferFromID] =8,[UnitsShipped],0))) AS [Shipd]," _
& "[Recd] AS [Total]" _
& "INTO [DynamicInventory]" _
& "FROM [Inventory Transactions]" _
& "WHERE ([PurchaseOrderID] = """ & strValue & """)" _
& "GROUP BY [ProductID];"

DoCmd.OpenQuery "POReceipts_InvTrans_update"
DoCmd.SetWarnings True

Dim iRecordNumber As Long
.RecordNumber = Me.CurrentRecord
le.Refresh
oCmd.GoToRecord , , acGoTo, iRecordNumber

nd Sub

rivate Sub Form_Close()
oCmd.OpenForm "Menu Inicial"
nd Sub
```

PEDIDO DE COMPRAS

```
Option Compare Database
Option Explicit
Public strTmp As String
Public strdelete As String
Dim XObject As New EB.FBexplodeClass


Private Sub Command84_Click()
    MsgBox Me.NewRecord
End Sub

Private Sub cmdExplode_Click()
If Not (Me![Status] = "1") Then
    MsgBox ("É necessária autorização para alterar um Pedido de Compra Ativo, Fechado ou Arquivado!")
    Exit Sub
End If

If (IsNull(Me.ExplodeLevel) Or IsNull(Me.ExplodeProductID) Or IsNull(Me.ExplodeQty)) Then
  MsgBox  "Campos  em  branco  devem  ser  preenchidos  antes  de  explodir.    Por  favor  preencha
Nível/Qtde/Produto. "
  Exit Sub
End If

If (IsNull(Me.SupplierID)) Then
    MsgBox "Por favro selecione um fornecedor no campo apropriado na guia 'Principal' antes de auto-
preencher da partir de uma EP."
    Exit Sub
End If

Dim dbsCurrent As DAO.Database
Dim rst As DAO.Recordset
Dim strPOID As String
Dim strValue As String
Dim dblQty As Double
Dim mLevel As Integer
Dim lngSupplierID As Long

Dim levelCurrent As Long
Dim levelPrevious As Long
Dim levelLoop As Long
Dim parentQuantity As Double

Set dbsCurrent = CurrentDB()
mLevel = Me.ExplodeLevel
strPOID = Me!PurchaseOrderID
lngSupplierID = Me!SupplierID
strValue = Me.ExplodeProductID
dblQty = Me.ExplodeQty

  DoCmd.SetWarnings False
  DoCmd.Hourglass True

  XObject.mlevels = mLevel
  XObject.ParentID = strValue
  XObject.DatabaseCurrent = dbsCurrent
  On Error GoTo Err_cmdExplode_Click
  XObject.Winxx = DLookup("[Windows]", "[VersionControl]")

  XObject.Explode

  dbsCurrent.Execute "UPDATE Exploded_BOM SET Name = """ & strPOID & """;"
  dbsCurrent.Execute "UPDATE Exploded_BOM SET QtyInParent = QtyInParent * " & dblQty & " WHERE Level =
;"

  Set rst = dbsCurrent.OpenRecordset("Exploded_BOM", dbOpenTable)


  Do While Not (rst.EOF)
    levelCurrent = rst!Level
    parentQuantity = rst!QtyInParent

    rst.Edit
    rst!VendorPartNumber = DLast("VendorItemNumber", "QVL", "SupplierID = " & Me!SupplierID & " AND
roductID = """ & ReplaceString(rst!ProductID, Chr(34), Chr(34) & Chr(34)) & """")
    rst.update
    rst.MoveNext
    If Not (rst.EOF) Then
        levelPrevious = levelCurrent
        levelCurrent = rst!Level
```

```
        If (levelCurrent > levelPrevious) Then
            levelLoop = levelCurrent
            Set rst = Filter_Recordset_MultiLevelBOM(rst, parentQuantity, levelLoop)
            If (Not (rst.EOF)) Then
            levelCurrent = rst!Level
            End If
        End If
    Else
        Exit Do
    End If
  Loop

  DoCmd.OpenQuery "PO_Exploded_Append_Query"
  rst.Close


Exit_cmdExplode_Click:
  DoCmd.SetWarnings True
  DoCmd.Hourglass False
  Set dbsCurrent = Nothing
  Me.Purchase_Orders_Subform.Requery
  Exit Sub

Err_cmdExplode_Click:
  If (Err.Number = 3021) Then
    MsgBox "Este produto não possui filhos. Não há itens para gerar."
  Else
    MsgBox Err.Description
  End If
  GoTo Exit_cmdExplode_Click
End Sub


Private Sub Currency_AfterUpdate()
 Dim temp As String

        temp = Me![Currency]

        Me![Purchase Orders Subform].SetFocus
        DoCmd.GoToRecord , , acFirst

        While (Not Me![Purchase Orders Subform].Form.NewRecord)

                Me![Purchase Orders Subform]![ProductsCurrency] = temp
                Me![Purchase Orders Subform]![ITCurrency] = temp

            DoCmd.GoToRecord , , acNext
        Wend

        DoCmd.GoToRecord , , acLast

End Sub

Private Sub Form_AfterDelConfirm(Status As Integer)
Select Case Status
        Case acDeleteOK
            MsgBox "O processo ocorreu normalmente."
        Case acDeleteCancel

MsgBox "UO usuário cancelou o processo."
        Case acDeleteUserCancel
            MsgBox "O usuário cancelou o processo."
    End Select
nd Sub

rivate Sub Form_BeforeDelConfirm(Cancel As Integer, Response As Integer)
    Response = acDataErrContinue
    If MsgBox("Apagar este Registro?", vbOKCancel) = vbCancel Then
        Cancel = True
        Exit Sub
    End If
    Dim dbsCurrent As DAO.Database
    Dim strPurchaseOrderID As String
    Dim intPurchaseOrderID As Integer
    Dim wrkJet As DAO.Workspace
     Set dbsCurrent = CurrentDB()

   DoCmd.Hourglass True
    dbsCurrent.Execute "DELETE * FROM [Inventory Transactions] " _
    & "WHERE ([Inventory Transactions].[PurchaseOrderID] =""" & strdelete & """);"
   DoCmd.Hourglass False
```

```
End Sub

Private Sub Form_BeforeUpdate(Cancel As Integer)
Dim update As Integer
Dim strNew As String
strNew = Me![Status]

If Not ((Me![Status] = "1") Or (Me![Status].OldValue = "1")) Then
    MsgBox ("É necessária autorização para alterar um Pedido de Compra Ativo, Fechado, Arquivado ou
'Drop Ship'!")
    Me.Undo
    Location.Requery
    Exit Sub
End If
End Sub


Private Sub Form_Close()
  Me.Filter = "StatusID <> 7"
  DoCmd.OpenForm "Menu Inicial"
End Sub

Private Sub Form_Current()

    strTmp = iif(IsNull(Me!PurchaseOrderID), "Null", Me!PurchaseOrderID)
    Location.Requery
End Sub
Private Sub Form_Activate()

On Error GoTo Err_Form_Activate
    DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, , acMenuVer70

Exit_Form_Activate:
    Exit Sub

Err_Form_Activate:
    MsgBox Err.Description
    Resume Exit_Form_Activate
End Sub

Private Sub Form_Delete(Cancel As Integer)
strTmp = Me!PurchaseOrderID
strdelete = strTmp
End Sub

Private Sub Form_Load()
    ShipTo.Requery
End Sub

Private Sub Form_LostFocus()
On Error Resume Next
strTmp = Me!PurchaseOrderID
End Sub

Private Sub Form_Open(Cancel As Integer)
  Me.Filter = "StatusID <>7"
  DoCmd.ApplyFilter "Purchase Orders(Inventory)", "StatusID <>7"
End Sub

Private Sub ItemFind_Click()
If Not IsNull([ItemLookUP]) Then
    PurchaseOrderID.SetFocus
    DoCmd.FindRecord ([ItemLookUP])
    ItemLookUP.SetFocus
Else
    MsgBox "Por favor selecione um Pedido de Compra para exibir."
End If

End Sub

Private Sub ItemUnArchive_Click()
If Not IsNull([ItemLookUpArchived]) Then
    Dim dbsCurrent As DAO.Database
    Dim strIDtmp As String
    Set dbsCurrent = CurrentDB
    strIDtmp = ItemLookUpArchived
    dbsCurrent.Execute "UPDATE [Purchase Orders] " _
      & "SET StatusID = 4 " _
      & "WHERE PurchaseOrderID = """ & strIDtmp & """;"
```

```
    Me.Requery
    Me.ItemLookUpArchived.Requery
    Me.ItemLookUP.Requery


    PurchaseOrderID.SetFocus
    DoCmd.FindRecord ([strIDtmp])
    ItemLookUP.SetFocus
Else
    MsgBox "Por favor selecione um Pedido de Compra para desarquivar."
End If


End Sub

Private Sub Location_GotFocus()
Location.Requery
End Sub

Function Filter_Recordset_MultiLevelBOM(temp_rst As Recordset, aQuantity As Double, aLevelLoop As
Long) As Recordset

Dim levelCurrent As Long
Dim levelPrevious As Long
Dim levelLoop As Long
Dim parentQuantity As Double
On Error GoTo 0
levelCurrent = temp_rst!Level
Do While (levelCurrent >= aLevelLoop)
    temp_rst.Edit
    temp_rst!QtyInParent = temp_rst!QtyInParent * aQuantity
    temp_rst!VendorPartNumber = DLast("VendorItemNumber", "QVL", "SupplierID = " & Me!SupplierID & "
AND ProductID = """ & ReplaceString(temp_rst!ProductID, Chr(34), Chr(34) & Chr(34)) & """")
    temp_rst.update
    parentQuantity = temp_rst!QtyInParent
    temp_rst.MoveNext
    If Not (temp_rst.EOF) Then
        levelPrevious = levelCurrent
        levelCurrent = temp_rst!Level
        If (levelCurrent > levelPrevious) Then
            levelLoop = levelCurrent
            Set temp_rst = Filter_Recordset_MultiLevelBOM(temp_rst, parentQuantity, levelLoop)
           If (Not (temp_rst.EOF)) Then
           levelCurrent = temp_rst!Level
           Else
           Exit Do
           End If
        End If
    Else
        Exit Do
    End If
Loop

Set Filter_Recordset_MultiLevelBOM = temp_rst

End Function


Private Sub Purchase_Orders_Subform_Enter()
    If Me.NewRecord Then
        MsgBox "Você deve preencher a info do Pedido de Compra antes de adicionar um produto."
        Me![SupplierID].SetFocus
        Exit Sub
    End If
    If IsNull(Me![PurchaseOrderID]) Then
        Me![OrderDate] = Date
    End If
End Sub

Private Sub Preview_Click()
On Error GoTo Err_Preview_Click
    Me.Refresh
    If IsNull(Me![PurchaseOrderID]) Then
        MsgBox "Entre as informações do Pedido de Compra antes de Visualizar."
    Else
        DoCmd.Minimize
        DoCmd.OpenReport "Purchase Order", acPreview, , "[Purchase Orders].[PurchaseOrderID]= """ &
Me![PurchaseOrderID] & """"
    End If
Exit_Preview_Click:
    Exit Sub
```

```
Err_Preview_Click:
    If Err <> 2501 And Err <> 2465 Then
        MsgBox Err.Description
    End If
    Resume Exit_Preview_Click
End Sub

Private Sub PurchaseOrderID_BeforeUpdate(Cancel As Integer)
    Dim decision As Integer
    Dim prefix As String
    Dim strValue As String
    Dim strSQL As String
    Dim rstID As DAO.Recordset
    Dim dbsCurrent As Database

    Set dbsCurrent = CurrentDB()
    decision = MsgBox("Você tem certeza de que quer alterar o número deste Pedido de Compra?",
vbYesNo)
    If decision = 7 Then
        Me.Undo
    Else
        If IsNull(Me!PurchaseOrderID) Then
            MsgBox "Não é permitido substitulr um número de Pedido por um valor nulo", vbOKOnly
            Me.Undo
            Exit Sub
        End If
        prefix = Left(Forms![Purchase Orders(inventory)]!PurchaseOrderID.Value, 2)
        If prefix <> "PO" Then
            MsgBox "Este código não é válido. Ele deve começar com 'PO'.", vbOKOnly
            Me.Undo
        Else
            strValue = Me!PurchaseOrderID
            strSQL = "SELECT * FROM [Purchase Orders] WHERE ([Purchase Orders]![PurchaseOrderID] = """
& strValue & """)"
            Set rstID = dbsCurrent.OpenRecordset(strSQL)
            If rstID.RecordCount <> 0 Then
                MsgBox "Este código já existe.", vbOKOnly
                Me.Undo
            End If
            rstID.Close
        End If
    End If
End Sub

Private Sub ShipTo_AfterUpdate()
Location.Requery
Location = Location.ItemData(0)
End Sub

Private Sub Status_AfterUpdate()
Me.Status.Requery
End Sub

Private Sub Status_Change()
Me.Status.Requery
End Sub

Private Sub StatusCriteria_AfterUpdate()
Dim strPOID As String
strPOID = Me!PurchaseOrderID
Select Case StatusCriteria
Case 1:
  Me.Filter = "StatusID <>7"
  DoCmd.ApplyFilter "Purchase Orders(inventory)", "StatusID <>7"
Case 2:
  Me.Filter = ""
  DoCmd.ApplyFilter "Purchase Orders(inventory)", ""
ase 3:
  Me.Filter = "StatusID =7"
  DoCmd.ApplyFilter "Purchase Orders(inventory)", "StatusID =7"
  On Error GoTo 0
nd Select
e.Requery
urchaseOrderID.SetFocus
oCmd.FindRecord ([strPOID])
nd Sub

ivate Sub SupplierID_Change()
e.Purchase_Orders_Subform!VendorNumber.Requery
nd Sub
```

PEDIDO DE COMPRAS – 2

```
Option Compare Database
Public strTmp As String
Public strdelete As String
Option Explicit

Private Sub Form_AfterDelConfirm(Status As Integer)
Select Case Status
        Case acDeleteOK
            MsgBox "O processo ocorreu normalmente."
        Case acDeleteCancel

MsgBox "O usuário cancelou o processo."
        Case acDeleteUserCancel
            MsgBox "O usuário cancelou o processo."
    End Select
End Sub

Private Sub Form_BeforeDelConfirm(Cancel As Integer, Response As Integer)
    Response = acDataErrContinue
    If MsgBox("Apagar este Registro?", vbOKCancel) = vbCancel Then
        Cancel = True
        Exit Sub
    End If
    Dim dbsCurrent As DAO.Database
    Dim strPurchaseOrderID As String
    Dim wrkJet As DAO.Workspace
     Set dbsCurrent = CurrentDB()

    On Error Resume Next

    strPurchaseOrderID = strdelete
    DoCmd.Hourglass True
    dbsCurrent.Execute "DELETE * FROM [Inventory Transactions] WHERE ([Inventory
Transactions].[PurchaseOrderID] = """ & strPurchaseOrderID & """);"
    DoCmd.Hourglass False
End Sub

Private Sub Form_BeforeUpdate(Cancel As Integer)
Dim strNew As String
strNew = Me![Status]

If Not ((Me![Status] = "1") Or (Me![Status].OldValue = "1")) Then
    MsgBox ("É necessária autorização para alterar um Pedido de Compra Ativo, Fechado ou Arquivado!")

    Me.Undo
    Exit Sub
End If
End Sub

Private Sub Form_Current()
    strTmp = iif(IsNull(Me!PurchaseOrderID), "Null", Me!PurchaseOrderID)

End Sub
Private Sub Form_Activate()
On Error GoTo Err_Form_Activate
    DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, , acMenuVer70

Exit_Form_Activate:
    Exit Sub

Err_Form_Activate:
    MsgBox Err.Description
    Resume Exit_Form_Activate
End Sub

Private Sub Form_Delete(Cancel As Integer)
strTmp = Me!PurchaseOrderID
strdelete = strTmp
End Sub

Private Sub Form_LostFocus()
On Error Resume Next
strTmp = Me!PurchaseOrderID
End Sub


Private Sub Purchase_Orders_Subform_Enter()
    If IsNull(Me![PurchaseOrderID]) Then
        Me![OrderDate] = Date
```

```
    End If
End Sub


Private Sub Command46_Click()
On Error GoTo Err_Preview_Click
    DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, , acMenuVer70
    Dim strValue
    strValue = Me!PurchaseOrderID
    If IsNull(Me![PurchaseOrderID]) Then
        MsgBox "Entre as informações do Pedido de Compra antes de visualizar."
    Else
        DoCmd.Minimize
        DoCmd.OpenReport "PO_N1", acPreview, , "[Purchase Orders].[PurchaseOrderID]= """ & strValue &
"""
    End If

Exit_Preview_Click:
    Exit Sub

Err_Preview_Click:
    If Err <> 2501 Then
        MsgBox Err.Description
    End If
    Resume Exit_Preview_Click

End Sub

Private Sub ItemFind_Click()
If Not IsNull([ItemLookUP]) Then
    PurchaseOrderID.SetFocus
    DoCmd.FindRecord ([ItemLookUP])
    ItemLookUP.SetFocus
Else
    MsgBox "Por favor selecione um Pedido de Compra para exibir."
End If

End Sub

Private Sub PurchaseOrderID_BeforeUpdate(Cancel As Integer)
    Dim decision As Integer
    Dim prefix As String
    Dim strValue As String
    Dim strSQL As String
    Dim rstID As DAO.Recordset
    Dim dbsCurrent As Database

    Set dbsCurrent = CurrentDB()

   decision = MsgBox("Você tem certeza de que quer alterar o número deste Pedido de Compra?",
vbYesNo)
    If decision = 7 Then
        Me.Undo
   Else
        If IsNull(Me!PurchaseOrderID) Then
            MsgBox "Não é permitido substituir um número de Pedido por um valor nulo", vbOKOnly
            Me.Undo
            Exit Sub
        End If
        prefix = Left(Me!PurchaseOrderID.Value, 2)
        If prefix <> "PO" Then
            MsgBox "Este código não é válido. Ele deve começar com 'PO'.", vbOKOnly
            Me.Undo
        Else
            strValue = Me!PurchaseOrderID
            strSQL = "SELECT * FROM [Purchase Orders] WHERE ([Purchase Orders]![PurchaseOrderID] = """
strValue & """)"
            Set rstID = dbsCurrent.OpenRecordset(strSQL)
            If rstID.RecordCount <> 0 Then
                MsgBox "Este código já existe.", vbOKOnly
                Me.Undo
            End If
            rstID.Close
        End If
   End If
nd Sub
```

VERIFICAÇÃO DE QUALIDADE DE DADOS

```
Option Compare Database
Option Explicit

Private Const conExpenseOnly = "99999999999"

Private Sub cmdOption1_Click()
    DoCmd.OpenForm "ItemMaster"
End Sub

Private Sub cmdOption2_Click()
    DoCmd.OpenQuery "WhereUsed_InvalidLines"
End Sub

Private Sub cmdShowInvalidProductID_Click()
    If dCount("*", "WhereUsed_MissingProductID(ALL)") + dCount("*", "WhereUsed_InvalidLines") > 0 Then
        DoCmd.Close acQuery, "WhereUsed_MissingProductID(ALL)"
        DoCmd.Close acQuery, "WhereUsed_InvalidLines"
        DoCmd.OpenQuery "WhereUsed_MissingProductID(ALL)"
        DoCmd.OpenQuery "WhereUsed_InvalidLines"
    Else
        MsgBox "Não existem produtos inválidos."
    End If
End Sub

Private Sub Command0_Click()
    Dim dbsCurrent As Database
    Set dbsCurrent = CurrentDB()
    Dim stDocName As String

    DoCmd.SetWarnings False

    dbsCurrent.Execute "DELETE * FROM QClog;"

    DoCmd.OpenQuery "QC:Products_musts_0"
    DoCmd.OpenQuery "QC:Products_musts_1"
    DoCmd.OpenQuery "QC:Products_musts_2"
    DoCmd.OpenQuery "QC:Products_musts_3"
    DoCmd.OpenQuery "QC:Products_musts_4"
    DoCmd.OpenQuery "QC:Products_musts_5"
    DoCmd.OpenQuery "QC:Products_musts_6"
    DoCmd.OpenQuery "QC:Products_musts_7"
    DoCmd.OpenQuery "QC:SupplierGid_Nullvalue"
    DoCmd.OpenQuery "QC:Customerid_Nullvalue"
    DoCmd.OpenQuery "QC:Where_Used_Nulls_0"
    DoCmd.OpenQuery "QC:Where_Used_Nulls_1"
    DoCmd.OpenQuery "QC:Where_Used_Nulls_2"
    DoCmd.OpenQuery "QC:QtyInParent_0"
    DoCmd.OpenQuery "QC:Where_Used_InvalidProductID"
    DoCmd.OpenQuery "QC:Purchase_Orders_PO"
    DoCmd.OpenQuery "QC:Sales_Orders"
    DoCmd.OpenQuery "QC:ECO"
    DoCmd.OpenQuery "QC:NCMR_RMA"
    DoCmd.OpenQuery "QC_Where_Used_Duplicates"
    DoCmd.OpenQuery "QC_Where_Used_DuplicatesDeleteList"
    DoCmd.OpenQuery "QC_Where_Used_DuplicatesAppendQClog"

    On Error GoTo Error999

    stDocName = DLookup("[ProductName]", "[Products]", "ProductID = '99999999999'")
    On Error GoTo 0
    On Error GoTo Error_InvLoc
    stDocName = DLookup("[CustomerID]", "[Customers]", "[CustomerName] = '~stores'")
    On Error GoTo 0
    DoCmd.Minimize
    DoCmd.OpenReport "QClog Query Report", acViewPreview

    DoCmd.SetWarnings True

    Exit Sub

Error_InvLoc:
    dbsCurrent.Execute "INSERT INTO QClog ( ErrorID, TableName ) SELECT 5 AS Error, 'Customers' AS
Label;"
    Resume Next

Error999:
    dbsCurrent.Execute "INSERT INTO QClog ( ErrorID, TableName ) SELECT 4 AS Error, 'Products' AS
Label;"
    Resume Next
```

```
End Sub

Private Sub Command6_Click()
    'Check if ProductID '99999999999' is deleted, if is: add back to database
    DoCmd.SetWarnings False
    If (0 = Nz(dCount("ProductID", "[Products]", "ProductID = """ & conExpenseOnly & """), 0)) Then
      DoCmd.OpenQuery ("Append_Expense_99999999999")
    End If
    DoCmd.SetWarnings True

    DoCmd.SetWarnings False
    DoCmd.OpenQuery "Update_nullvalues_to_defaultvalues_in_ControlCharacter"
    DoCmd.OpenQuery "Update_nullvalues_to_defaultvalues_in_Currency"
    DoCmd.OpenQuery "Update_nullvalues_to_defaultvalues_in_CustUnitPrice"
    DoCmd.OpenQuery "Update_nullvalues_to_defaultvalues_in_Leadtime"
    DoCmd.OpenQuery "Update_nullvalues_to_defaultvalues_in_Manufacturer"
    DoCmd.OpenQuery "Update_nullvalues_to_defaultvalues_in_Margin"
    DoCmd.OpenQuery "Update_nullvalues_to_defaultvalues_in_PhysicalCount"
    DoCmd.OpenQuery "Update_nullvalues_to_defaultvalues_in_Release"
    DoCmd.OpenQuery "Update_nullvalues_to_defaultvalues_in_Minordersize"
    DoCmd.OpenQuery "Update_nullvalues_to_defaultvalues_in_Reorderlevel"
    DoCmd.OpenQuery "Update_nullvalues_to_defaultvalues_in_Rev"
    DoCmd.OpenQuery "Update_nullvalues_to_defaultvalues_in_Standardprice"
    DoCmd.OpenQuery "Update_nullvalues_to_defaultvalues_in_Unitprice"
    DoCmd.OpenQuery "Update_nullvalues_to_defaultvalues_in_UnitsOfMeasure"
    DoCmd.OpenQuery "Update_nullvalues_to_defaultvalues_in_TaxRate"
    DoCmd.SetWarnings True

End Sub

Private Sub Command9_Click()

    DoCmd.SetWarnings False
    DoCmd.OpenQuery "QC_Where_Used_Duplicates"
    DoCmd.OpenQuery "QC_Where_Used_DuplicatesDeleteList"
    DoCmd.OpenQuery "QC_Where_Used_DuplicatesUnmatched"
    DoCmd.OpenQuery "QC_Where_Used_DuplicatesClearOld"
    DoCmd.OpenQuery "QC_Where_Used_DuplicatesAppendNew"
    DoCmd.SetWarnings True

End Sub
Private Sub Form_Close()
DoCmd.OpenForm "Menu Inicial"
End Sub
```

PEDIDO DE VENDAS / ORDEM DE EMBARQUE

```
Option Compare Database
Option Explicit
Dim DBObject As New EB.EBdefinitionsClass
Dim SSOObject As New EBexplodeSSOClass
Const cIncomplete = "Incomplete"
Const cComplete = "Complete"
Const DefaultEmailSelection = 1
Public strTmp As String
Public strdelete As String

Private Sub CmbPackingOrderID_Click()
Me.CmbPackingOrderID.Requery
End Sub

Private Sub Command369_Click()
Call CmdExplodeVds_Click
End Sub

Private Sub CustomerID_AfterUpdate()
Dim dbsCurrent As DAO.Database
Dim strTmp As String

If (IsNull(Me.CustomerID.OldValue)) Then Exit Sub

Location.Requery
strTmp = Me.CustomerID.OldValue  'get old value before committing
DoCmd.RunCommand acCmdSaveRecord 'Commit changes in buffer to database
Set dbsCurrent = CurrentDB

dbsCurrent.Execute ("UPDATE [Inventory Transactions] " _
    & "SET CustomerID = " & Me.CustomerID & iif(IsNull(Me.Location), "", ",ShipToID =" & Me.Location)
& " " & "WHERE SalesOrderID = """ & Me!SalesOrderID & """ " & "AND CustomerID = " & strTmp & ";")

Set dbsCurrent = Nothing
End Sub

Private Sub DeleteShipment_Click()
Dim dbsCurrent As DAO.Database
Dim strPKID As String

Set dbsCurrent = CurrentDB
If (IsNull(Me.CmbPackingOrderID)) Then
    MsgBox "Você deve selecionar uma ordem de embalagem para poder apagar o embarque."
    Exit Sub
End If

On Error GoTo Err_DeleteShipment_NoPKID
    strPKID = Me.CmbPackingOrderID

On Error GoTo Err_DeleteShipment

If (vbNo = MsgBox("Esta ação pode invalidar alguns relatórios de embarque.  Apagar o embarque com
ordem de embalagem """ & strPKID & """?", vbYesNo)) Then
    MsgBox "Abortado. O embarque não foi apagado."
    Exit Sub
End If

dbsCurrent.Execute "UPDATE [Inventory Transactions] " _
    & "SET TransferFromID = 10, UnitsShipped = 0 " _
    & "WHERE PackingOrderID = """ & strPKID & """;"

Me.ShippingOrder_Subform.Requery
Me.CmbPackingOrderID.Requery
MsgBox "O registro foi apagado."

Exit_DeleteShipment:
    Exit Sub

Err_DeleteShipment:
    MsgBox Error
    GoTo Exit_DeleteShipment

Err_DeleteShipment_NoPKID:
    MsgBox "Ordem de embalagem não selecionada. Você deve selecionar uma ordem de embalagem para
apagar."
    GoTo Exit_DeleteShipment

End Sub
```

```
Private Sub EditShipmentNew_Click()
On Error GoTo Err_EditShipmentNew_Click
    Dim strMeOpenArgs As String

    DoCmd.RunCommand acCmdSaveRecord
    Me.Refresh
    Me.SalesOrderID.SetFocus

    strMeOpenArgs = "SO=" & Me!SalesOrderID
    DoCmd.OpenForm "Shipment Subform", , , , , , strMeOpenArgs
    DoCmd.Close acForm, Me.Name

Exit_EditShipmentNew_Click:
    Exit Sub

Err_EditShipmentNew_Click:
    MsgBox Err.Description
    Resume Exit_EditShipmentNew_Click

End Sub

Private Sub EditShipmentOld_Click()
On Error GoTo Err_EditShipmentOld_Click

    DoCmd.RunCommand acCmdSaveRecord
    Me.Refresh
    Me.SalesOrderID.SetFocus
    DoCmd.OpenForm "Shipment Subform"

Exit_EditShipmentOld_Click:
    Exit Sub

Err_EditShipmentOld_Click:
    MsgBox Err.Description
    Resume Exit_EditShipmentOld_Click

End Sub

Private Sub Form_Close()
    DoCmd.SetWarnings False
    DoCmd.Close acForm, "ShippingSN", acSavePrompt
    DoCmd.SetWarnings True
    DoCmd.OpenForm "Menu Inicial"
End Sub

Private Sub Form_Load()
  Dim dbsCurrent As DAO.Database
  Dim strValue As String
  Set dbsCurrent = CurrentDB()
  strValue = DLookup("[Userdefinable 3]", "My Company Information")
  strValue = strValue & "DBM_fe.mdb"
  DBObject.DatabaseCurrent = dbsCurrent
  DBObject.DatabaseName = strValue
  DBObject.SetDatabase
  If (Len(Nz(Me.OpenArgs, "")) > 0) Then Load_MeOpenArgs
End Sub

Sub IsComplete()

    Me![TglIncomplete].Value = False
    Me![TglIncomplete].Locked = False
    Me![TglIncomplete].Caption = cComplete
    Me![ShippingOrder Subform].Locked = True
    ChangeTextStatus (True)

End Sub

Sub IsIncomplete()

End Sub

Sub ValidationCheck(Valid As Boolean, Complete As Boolean)
Dim QtyRequired As Integer
Dim QtyShipped As Integer
Dim Count As Integer
Dim ErrorFlag As Boolean
Dim TransDate As Date
Dim TransDateNull As Boolean
Dim ShippedDate As Date
Dim ShippedDateNull As Boolean
```

```
    Exit Sub

    Me![ShippingOrder Subform].SetFocus
    DoCmd.GoToRecord , , acFirst

    ErrorFlag = False
    Complete = True
    Count = 0

    While (Not Me![ShippingOrder Subform].Form.NewRecord) And (Not ErrorFlag)

        If IsNull(Me![ShippingOrder Subform]![QtyRequired]) Then
            QtyRequired = 0
        Else
            QtyRequired = Val(Me![ShippingOrder Subform]![QtyRequired])
        End If

        If IsNull(Me![ShippingOrder Subform]![QtyShipped]) Then
            QtyShipped = 0
        Else
            QtyShipped = Val(Me![ShippingOrder Subform]![QtyShipped])
        End If

        If (Me![ShippingOrder Subform]![TransactionDate] <> "") Then
            TransDate = Format(Me![ShippingOrder Subform]![TransactionDate])
            TransDateNull = False
        Else
            TransDateNull = True
        End If

        If (Me![ShippingOrder Subform]![ShippedDate] <> "") Then
            ShippedDate = Format(Me![ShippingOrder Subform]![ShippedDate])
            ShippedDateNull = False
        Else
            ShippedDateNull = True
        End If

        If (Not ShippedDateNull) And (Not TransDateNull) Then
            If (ShippedDate < TransDate) Then
                ErrorFlag = True
                Me![ShippingOrder Subform]![TransactionDate].SetFocus
            End If
        End If
        If (QtyRequired < 0) Then
            ErrorFlag = True
            Me![ShippingOrder Subform]![QtyRequired].SetFocus
        End If
        If (QtyShipped <> QtyRequired) And (QtyShipped <> 0) Then
            ErrorFlag = True
            Me![ShippingOrder Subform]![QtyShipped].SetFocus
        End If
        If (ShippedDateNull = False) And ((QtyRequired > QtyShipped) Or (QtyRequired = 0)) Then
            ErrorFlag = True
            Me![ShippingOrder Subform]![ShippedDate].SetFocus
        End If
        If (ShippedDateNull = True) And (QtyRequired = QtyShipped) And (QtyRequired <> 0) Then
            ErrorFlag = True
            Me![ShippingOrder Subform]![ShippedDate].SetFocus
        End If
        If (ShippedDateNull = True) Or (QtyRequired <> QtyShipped) Then Complete = False
        If (QtyRequired = 0) Then Complete = False

        Count = Count + 1
        If ErrorFlag = False Then DoCmd.GoToRecord , , acNext

    Wend

    If Count = 0 Then
        Valid = False
    Else
        Valid = Not ErrorFlag
    End If

    If (Valid = False) Then Complete = False

End Sub

Sub ChangeButtons(Change As Boolean)

    Change = True
    Me!Shipping_email.Enabled = Change
```

```
End Sub

Private Sub CmdAutoShip_Click()

    Dim TempDate As Date
        TempDate = Me![txtShippedDate]
        Me![ShippingOrder Subform].SetFocus
        DoCmd.GoToRecord , , acFirst
        While (Not Me![ShippingOrder Subform].Form.NewRecord)

            If     (IsNull(Me![ShippingOrder     Subform]![ShippedDate]))     And     (Me![ShippingOrder
Subform]![ShippedDate].Locked = False) Then
                Me![ShippingOrder Subform]![ShippedDate] = Format(TempDate, "Medium Date")
            End If
            DoCmd.GoToRecord , , acNext
        Wend
        DoCmd.GoToRecord , , acLast

End Sub

Private Sub CmdLock_Click()
    Dim strValue As String
    Dim dbsCurrent As DAO.Database

    Set dbsCurrent = CurrentDB()
    Me![ShippingOrder Subform].SetFocus

    strValue = Me![SalesOrderID]

    dbsCurrent.Execute "UPDATE [Inventory Transactions] SET [Control] = [Control] & """ & "L" & """
WHERE (([Inventory Transactions]![SalesOrderID] = """ & strValue & """) AND ([Inventory
Transactions].[Control] not Like """ & "*L*" & """));"
End Sub

Private Sub CmdTransfer_Click()

    Const MsgText = "Você gostaria de limpar do formulário principal os valores copiados?"
    Const msgTitle = "Cópia Completa."

    Dim ClearMainForm As Integer
    Dim ShippedDate As Date
    Dim OrderDate As Date
    Dim ETADate As Date
    Dim DeliveredDate As Date
    Dim ShippedDateNull As Boolean
    Dim OrderDateNull As Boolean
    Dim ETADateNull As Boolean
    Dim DeliveredDateNull As Boolean
    Dim CustomerPO As String
    Dim SalesTracking As String
    Dim ShipperTracking As String
    Dim ShippingMethod As Long

        If Not IsNull(Me![txtShippedDate]) Then ShippedDate = Me![txtShippedDate]
        ShippedDateNull = IsNull(Me![txtShippedDate])

        If Not IsNull(Me![OrderDate]) Then OrderDate = Me![OrderDate]
        OrderDateNull = IsNull(Me![OrderDate])

        If Not IsNull(Me![ETADate]) Then ETADate = Me![ETADate]
        ETADateNull = IsNull(Me![ETADate])

        If Not IsNull(Me![DeliveredDate]) Then DeliveredDate = Me![DeliveredDate]
        DeliveredDateNull = IsNull(Me![DeliveredDate])

        If Not IsNull(Me![CustomerPONumber]) Then
            CustomerPO = Me![CustomerPONumber]
        Else
            CustomerPO = ""
        End If

        If Not IsNull(Me![SalesTrackingNumber]) Then
            SalesTracking = Me![SalesTrackingNumber]
        Else
            SalesTracking = ""
        End If

        If Not IsNull(Me![ShipperTrackingNumber]) Then
            ShipperTracking = Me![ShipperTrackingNumber]
        Else
```

```
            ShipperTracking = ""
        End If

        If Not IsNull(Me![ShippingMethodID]) Then
            ShippingMethod = Me![ShippingMethodID]
        Else
            ShippingMethod = 0
        End If

        Me![ShippingOrder Subform].SetFocus
        DoCmd.GoToRecord , , acFirst

        While (Not Me![ShippingOrder Subform].Form.NewRecord)

            If (Me![ShippingOrder Subform]![ShippedDate].Locked = False) Then

                If Not ShippedDateNull And IsNull(Me![ShippingOrder Subform]![ShippedDate]) Then
                    Me![ShippingOrder Subform]![ShippedDate] = Format(ShippedDate, "Medium Date")
                End If

                If Not OrderDateNull And IsNull(Me![ShippingOrder Subform]![OrderedDate]) Then
                    Me![ShippingOrder Subform]![OrderedDate] = Format(OrderDate, "Medium Date")
                End If

                If Not ETADateNull And IsNull(Me![ShippingOrder Subform]![ETADate]) Then
                    Me![ShippingOrder Subform]![ETADate] = Format(ETADate, "Medium Date")
                End If

                If Not DeliveredDateNull And IsNull(Me![ShippingOrder Subform]![ETADate]) Then
                    Me![ShippingOrder Subform]![DeliveredDate] = Format(DeliveredDate, "Medium Date")
                End If

                If IsNull(Me![ShippingOrder Subform]![CustomerPO]) Then Me![ShippingOrder
Subform]![CustomerPO] = CustomerPO
                If IsNull(Me![ShippingOrder Subform]![SalesTracking]) Then Me![ShippingOrder
Subform]![SalesTracking] = SalesTracking
                If IsNull(Me![ShippingOrder Subform]![ShippingTracking]) Then Me![ShippingOrder
Subform]![ShippingTracking] = ShipperTracking
                If IsNull(Me![ShippingOrder Subform]![ShippingMethod]) Then Me![ShippingOrder
Subform]![ShippingMethod] = ShippingMethod

            End If

            DoCmd.GoToRecord , , acNext

        Wend
        DoCmd.GoToRecord , , acLast
        ClearMainForm = MsgBox(MsgText, vbYesNo + vbQuestion + vbDefaultButton2, msgTitle)

        If ClearMainForm = vbYes Then
            Me![txtShippedDate].SetFocus
            Me![txtShippedDate].text = ""
            Me![OrderDate].SetFocus
            Me![OrderDate].text = ""
            Me![ETADate].SetFocus
            Me![ETADate].text = ""
            Me![DeliveredDate].SetFocus
            Me![DeliveredDate].text = ""
            Me![CustomerPONumber].SetFocus
            Me![CustomerPONumber].text = ""
            Me![SalesTrackingNumber].SetFocus
            Me![SalesTrackingNumber].text = ""
            Me![ShipperTrackingNumber].SetFocus
            Me![ShipperTrackingNumber].text = ""
            Me![ShippingMethodID].SetFocus
            Me![ShippingMethodID].text = ""

        End If

    Dim strValue As String
    Dim dbsCurrent As Database

  Set dbsCurrent = CurrentDB()

    strValue = Me![SalesOrderID]
    dbsCurrent.Execute "UPDATE [Inventory Transactions] SET [Control] = [Control] & """ & "L" & """
WHERE (([Inventory Transactions]![SalesOrderID] = """ & strValue & """) AND ([Inventory
Transactions].[Control] not Like """ & "*L*" & """));"

End Sub
```

```
Private Sub CmdValCheck_Click()

    Dim Valid As Boolean
    Dim Finished As Boolean

    Const ErrorMessage = "O registro atual não é válido."
    Const ErrorTitle = "NÃO É VÁLIDO."

    ValidationCheck Valid, Finished
    Valid = True
    If (Valid = False) Then
        MsgBox ErrorMessage, vbOKOnly + vbExclamation, ErrorTitle
        ChangeButtons (False)
    Else
        DoCmd.GoToRecord , , acLast
        ChangeButtons (True)
    End If

    If (Finished = True) Then
        IsComplete
    Else
        IsIncomplete
    End If

End Sub

Private Sub Delete_Sales_Orders_Record_and_subform_records_Click()
Dim dbsCurrent As DAO.Database
Dim strSalesOrderID As String
Dim wrkJet As DAO.Workspace
  Set dbsCurrent = CurrentDB()

  strSalesOrderID = Forms![Shipping Order]!SalesOrderID
  DoCmd.Hourglass True
    dbsCurrent.Execute "DELETE * FROM [Sales Orders] WHERE ([Sales Orders].[SalesOrderID]=""" &
strSalesOrderID & """);"
    dbsCurrent.Execute "DELETE * FROM [Inventory Transactions] WHERE ([Inventory
Transactions].[SalesOrderID] = """ & strSalesOrderID & """);"
    DoCmd.Hourglass False


DoCmd.Save acForm, "Sales Order"
DoCmd.GoToControl "SalesOrderID"
DoCmd.GoToRecord , , acFirst
End Sub

Private Sub Command378_Click()
Dim TempDate As Date

        TempDate = Me![txtShippedDate2]
        Me![ShippingOrder Subform].SetFocus
        DoCmd.GoToRecord , , acFirst

        While (Not Me![ShippingOrder Subform].Form.NewRecord)
            If (IsNull(Me![ShippingOrder Subform]![ShippedDate])) And (Me![ShippingOrder
ubform]![ShippedDate].Locked = False) Then
                Me![ShippingOrder Subform]![ShippedDate] = Format(TempDate, "Medium Date")
            End If
            DoCmd.GoToRecord , , acNext
        Wend
        DoCmd.GoToRecord , , acLast

nd Sub

ivate Sub Command419_Click()
m TempDate As Date
        TempDate = Me![txtShippedDate1]

        Me![ShippingOrder Subform].SetFocus
        DoCmd.GoToRecord , , acFirst

        While (Not Me![ShippingOrder Subform].Form.NewRecord)

            If (IsNull(Me![ShippingOrder Subform]![ShippedDate])) And (Me![ShippingOrder
bform]![ShippedDate].Locked = False) Then
                Me![ShippingOrder Subform]![ShippedDate] = Format(TempDate, "Medium Date")
            End If
            DoCmd.GoToRecord , , acNext
        Wend
        DoCmd.GoToRecord , , acLast
```

```
End Sub

Private Sub CopySOItems_Click()
If (Me!ShipmentModified = 0) Then
  If (vbCancel = MsgBox("Esta ação vai criar um novo 'Número de Embalagem' Rev. Você quer continuar?",
vbOKCancel, "Novo Rev")) Then
    Exit Sub
  End If
End If

Dim dbsCurrent As DAO.Database
Dim iLevel As Integer
Dim kLevel As Integer
Dim mLevel As Integer
Dim iNumber As Integer
Dim rstDMR As DAO.Recordset
Dim rstDMR_sort As DAO.Recordset
Dim wrkJet As DAO.Workspace
Dim strValue As String
Dim strProductID As String
Dim qdfExplode As DAO.QueryDef
Dim jLevel As Integer
Dim iCustomerID As Integer
Dim strSQL As String
Dim iRecordNumber As Integer

    iRecordNumber = Me.CurrentRecord
    DoCmd.SetWarnings False
    Set dbsCurrent = CurrentDB()
    strValue = Forms![Sales_Shipping Order]!SalesOrderID

    iCustomerID = iif(IsNull(Forms![Sales_Shipping Order]![CustomerID]), 0, Forms![Sales_Shipping
Order]![CustomerID])
    If iCustomerID <= 0 Then
    MsgBox ("Impossível copiar porque o campo 'Cliente' está em branco. " _
    & "Selecione um cliente da lista e em seguida clique 'Copiar e Editar' novamente!")
    DoCmd.Hourglass False
    Exit Sub
    End If

    mLevel = 4


    iLevel = 0
    iNumber = 0
    strSQL = "SELECT * FROM [Inventory Transactions] WHERE (([Inventory Transactions]![SalesOrderID] =
""" & strValue & """) AND ([Inventory Transactions].[Control] Like """ & "*0*" & """))"

    Set rstDMR = dbsCurrent.OpenRecordset(strSQL)
    If rstDMR.RecordCount = 0 Then
    MsgBox ("Não há produtos pai para copiar. ")
    rstDMR.Close
    DoCmd.Hourglass False
    Exit Sub
    End If
    rstDMR.Close

dbsCurrent.Execute "DELETE ShipChk.* FROM ShipChk;"
dbsCurrent.Execute "DELETE ShipChk_Sorted.* FROM ShipChk_Sorted;"

dbsCurrent.Execute "INSERT INTO ShipChk "
& "SELECT ProductID, CustomerID, ShipToID, UnitsShippedFG as [UnitsSold],""" & iLevel & """ as
[Level], "
& "Control,""" & strValue & """ as [SalesOrderID],OrderLine,TransactionID " _
& "FROM [Inventory Transactions] "
& "WHERE [SalesOrderID]=""" & strValue & """ AND ([Inventory Transactions].[Control] Like """ & "*0*"
& """) AND ([Inventory Transactions].[Control] Not Like """ & "*W*" & """);"

DoCmd.OpenQuery "ShipChk_Append_vb1_SHO"

Set_PackingOrderID

DoCmd.SetWarnings True
DoCmd.Hourglass False
me![ShippingOrder Subform].Requery
DoCmd.GoToRecord , , acGoTo, iRecordNumber

DoCmd.RunCommand acCmdSaveRecord
EditShipmentNow_Click

End Sub
```

```
Private Sub Form_BeforeDelConfirm(Cancel As Integer, Response As Integer)

    Response = acDataErrContinue
    Dim strValue As String
    strValue = Left(strdelete, 2)


    If strValue <> "SH" Then
        MsgBox ("Apenas ordens de embarque do tipo SHxxxx podem ser apagadas a partir deste
formulário.")
        DoCmd.CancelEvent
        Exit Sub
    End If
    If MsgBox("Apagar este Registro?", vbOKCancel) = vbCancel Then
        Cancel = True
        Exit Sub
    End If
    Dim dbsCurrent As DAO.Database
    Dim strSalesOrderID As String
    Dim wrkJet As DAO.Workspace
     Set dbsCurrent = CurrentDB()

    On Error Resume Next

    strSalesOrderID = strdelete
    DoCmd.Hourglass True
    dbsCurrent.Execute "DELETE * FROM [Inventory Transactions] WHERE ([Inventory
Transactions].[SalesOrderID] = """ & strSalesOrderID & """);"
    DoCmd.Hourglass False

End Sub

Private Sub Form_AfterDelConfirm(Status As Integer)
    Select Case Status
        Case acDeleteOK
            MsgBox "O processo ocorreu normalmente."
        Case acDeleteCancel
            MsgBox "O usuário cancelou o processo."
        Case acDeleteUserCancel
            MsgBox "O usuário cancelou o processo."
    End Select
End Sub

Private Sub Form_Current()
Dim counter As Integer
Dim Valid As Boolean
Dim Finished As Boolean
Me.Location.Requery
Me.CmbPackingOrderID.Requery
DoCmd.GoToControl "SalesOrderID"

strTmp = iif(IsNull(Me!SalesOrderID), "Null", Me!SalesOrderID)

    For counter = 0 To Me![LstEmailNames].ListCount
        Me![LstEmailNames].Selected(counter) = False
    Next

    Me![LstEmailNames].Selected(DefaultEmailSelection) = True

    Exit Sub
    ValidationCheck Valid, Finished

    DoCmd.GoToRecord , , acFirst

    Me!SalesOrderID.SetFocus

    If (Not Valid) Or (Me.NewRecord = True) Then
        Me![TglIncomplete].Value = True
    Else
        ChangeButtons (True)
    End If

    If (Me![TglIncomplete].Value = True) Then
        IsIncomplete
    Else
        IsComplete
    End If

nd Sub
```

```
Private Sub Form_Delete(Cancel As Integer)
strTmp = Me!SalesOrderID
strdelete = strTmp

End Sub

Private Sub Form_Open(Cancel As Integer)
  On Error GoTo FormNotOpen
FormNotOpen:
    Dim Valid As Boolean
    Dim Finished As Boolean

    Const ErrorMessage = "O registro atual não é válido."
    Const ErrorTitle = "Não Válido"

        ChangeButtons (True)

End Sub

Private Sub Form_Unload(Cancel As Integer)
Set DBObject = Nothing
Set SSOObject = Nothing
End Sub

Private Sub ItemFind_Click()
If Not IsNull([ItemLookUP]) Then
    SalesOrderID.SetFocus
    DoCmd.FindRecord ([ItemLookUP])
    ItemLookUP.SetFocus
Else
    MsgBox "Por favor selecione uma ordem de embarque para exibir."
End If

End Sub

Private Sub Location_Exit(Cancel As Integer)
Dim dbsCurrent As DAO.Database
If (IsNull(Me.Location)) Then Exit Sub

Set dbsCurrent = CurrentDB
dbsCurrent.Execute ("UPDATE [Inventory Transactions] " _
  & "SET ShipToID =" & Me!Location & " " _
  & "WHERE SalesOrderID = """ & Me!SalesOrderID & """ " _
  & "AND CustomerID = " & Me!CustomerID & ";")

Set dbsCurrent = Nothing

End Sub

Private Sub Location_GotFocus()
Location.Requery
End Sub

Private Sub Open_ShippingSN_Click()
On Error GoTo Err_Open_ShippingSN_Click

    Dim stDocName As String
    Dim stLinkCriteria As Strin:

    stDocName = "ShippingSN"

    stLinkCriteria = "[SalesOrderID]=" & "'" & Me![SalesOrderID] & "'"
    DoCmd.OpenForm stDocName, , , stLinkCriteria

xit_Open_ShippingSN_Click:
    Exit Sub

rr_Open_ShippingSN_Click:
    MsgBox Err.Description
    Resume Exit_Open_ShippingSN_Click


nd Sub

rivate Sub ReportCommercialInvoice_Click()
portPreview "Commercial Invoice", iif(IsNull(Me.CmbPackingOrderID), "", Me.CmbPackingOrderID)
nd Sub

rivate Sub ReportFCC_Click()
portPreview "FCCMonitor", iif(IsNull(Me.CmbPackingOrderID), "", Me.CmbPackingOrderID)
nd Sub
```

```
Private Sub ReportFDA_Click()
ReportPreview "FDAMonitor", iif(IsNull(Me.CmbPackingOrderID), "", Me.CmbPackingOrderID)
End Sub

Private Sub ReportInvoice_Click()
ReportPreview "Invoice", iif(IsNull(Me.CmbPackingOrderID), "", Me.CmbPackingOrderID)
End Sub

Private Sub ReportNAFTA_Click()
ReportPreview "NAFTA", iif(IsNull(Me.CmbPackingOrderID), "", Me.CmbPackingOrderID)
End Sub

Private Sub ReportPackingList_Click()
ReportPreview "PackingList", iif(IsNull(Me.CmbPackingOrderID), "", Me.CmbPackingOrderID)
End Sub

Private Sub ReportShippingCheckList_Click()
ReportPreview "ShippingCheckList", iif(IsNull(Me.CmbPackingOrderID), "", Me.CmbPackingOrderID)
End Sub

Private Sub ReportShippingLabels_Click()
ReportPreview "Shipping Labels", iif(IsNull(Me.SalesOrderID), "", Me.SalesOrderID)
End Sub

Private Sub ReportPreview(strReportName As String, strPKID As String)
On Error GoTo Err_ReportPreview
If (Len(strPKID) = 0) Then
  MsgBox "Para visualizar o relatório você deve selecionar a Ordem de embalagem/embarque."
  Exit Sub
End If

If (strReportName = "Shipping Labels") Then
  DoCmd.OpenReport strReportName, acPreview, , "SalesOrderID = """ & strPKID & """"
Else
  DoCmd.OpenReport strReportName, acPreview, , "PackingOrderID = """ & strPKID & """"
End If

Exit_ReportPreview:
  Exit Sub

Err_ReportPreview:
  MsgBox Error
  GoTo Exit_ReportPreview
End Sub


Private Sub SalesOrderID_LostFocus()
On Error Resume Next
strTmp = Me!SalesOrderID
End Sub

Private Sub Shipping_email_Click()
Const NewLine = vbNewLine
Const TempRptFileName = "C:\TempRpt.TXT"
Const EmailColumn = 2
Const MsgErrorText = "Você não selecionou um destinatário para enviar a mensagem."

Dim dbsCurrent As DAO.Database

Dim rstShip As DAO.Recordset

Dim strRMANumber As String
Dim FirstInfo As String
Dim strSalesOrderID As String
Dim strCustomerName As String
Dim strShippingMethod As String
Dim ReportString As String
Dim TempString As String
Dim EmailTo As String

Dim iCustomerID As Long
Dim iShippingMethodID As Long

Dim ReportFile As Integer
Dim counter As Integer

Set dbsCurrent = CurrentDB()
iCustomerID = Me!CustomerID
DoCmd.Hourglass True
```

```
 Dim strValue As String

 strValue = Forms![Sales_Shipping Order]!SalesOrderID
 Set rstShip = dbsCurrent.OpenRecordset("SELECT * FROM Customers ", dbOpenSnapshot)
 With rstShip
     .MoveFirst
     Do While Not .EOF
         If rstShip![CustomerID] = iCustomerID Then
             strCustomerName = rstShip!CustomerName
         End If
         .MoveNext
     Loop
 End With

 iShippingMethodID = Me!ShippingMethodID
 Set rstShip = dbsCurrent.OpenRecordset("SELECT * FROM [Shipping Methods] ", dbOpenSnapshot)

 With rstShip
     .MoveFirst
     Do While Not .EOF
         If rstShip![ShippingMethodID] = iShippingMethodID Then
             strShippingMethod = rstShip!ShippingMethod
         End If
         .MoveNext
     Loop
 End With

 strRMANumber = "Shipment: " & Me!SalesOrderID & "/: " & strCustomerName

 FirstInfo = "CSTMR:" & strCustomerName & "|NOTES:" & Me!email_message & "|ETA:" & Me!ETADate &
 "|ORDER:" & Me!SalesOrderID
 FirstInfo = FirstInfo & "|DLVRDdate" & Me!DeliveredDate & "|SO:" & Me!SalesTrackingNumber & "|VIA:" &
 strShippingMethod & "|WAYBILL:" & Me!ShipperTrackingNumber
 FirstInfo = FirstInfo & NewLine & NewLine & "PACKING LIST FOLLOWS:" & NewLine & NewLine
 EmailTo = ""
     With Forms![Sales_Shipping Order]![LstEmailNames]
     For counter = 0 To .ListCount
         If (.Selected(counter) = True) Then
             If EmailTo = "" Then
                 EmailTo = .Column(EmailColumn, counter)
             Else
                 EmailTo = EmailTo & ", " & .Column(EmailColumn, counter)
             End If
         End If
     Next
 End With

 If EmailTo = "" Then
     MsgBox MsgErrorText, vbOKOnly + vbExclamation, "Por favor selecione um destinatário. O e-mail não
 foi enviado."
 Else
     DoCmd.SendObject acSendReport, "PackingList1", acFormatRTF, EmailTo, , , strRMANumber, FirstInfo &
 vbCrLf & vbCrLf, True
 End If

 rstShip.Close
 DoCmd.Hourglass False
 End Sub

 Private Sub CmdExplodeVGs_Click()
 If (Me!ShipmentModified = 0) Then
  If (vbCancel = MsgBox("Esta ação vai criar um novo 'Número de Embalagem' Rev. Você quer continuar?",
 vbOKCancel, "Novo Rev")) Then
    Exit Sub
  End If
 End If

 Dim dbsCurrent As DAO.Database
 Dim iLevel As Integer
 Dim kLevel As Integer
 Dim mLevel As Integer
 Dim iNumber As Integer
 Dim rstDMR As DAO.Recordset
 Dim rstDMR_sort As DAO.Recordset
 Dim wrkJet As DAO.Workspace
 Dim strValue As String
 Dim strProductID As String
 Dim qdfExplode As DAO.QueryDef
 Dim jLevel As Integer
 Dim iCustomerID As Integer
 Dim iShipToID As Integer
```

```
Dim strSQL As String

Set dbsCurrent = CurrentDB()
DoCmd.Hourglass True
DoCmd.SetWarnings False
strValue = Forms![Sales_Shipping Order]!SalesOrderID
iShipToID = Me!ShipToID

iCustomerID = iif(IsNull(Forms![Sales_Shipping Order]![CustomerID]), 0, (Forms![Sales_Shipping
Order]![CustomerID]))
If iCustomerID <= 0 Then
    MsgBox ("Impossível explodir porque o campo 'Transferir para' está em branco. " _
    & "Selecione um destino a partir da lista e clique 'Explodir' novamente!")
    DoCmd.Hourglass False
    DoCmd.SetWarnings False
    Exit Sub
End If

iLevel = 0
iNumber = 0
    strSQL = "SELECT * FROM [Inventory Transactions] WHERE (([Inventory Transactions]![SalesOrderID] =
""" & strValue & """) AND ([Inventory Transactions].[TransferFromID] = 16) AND ([Inventory
Transactions].[Control] not Like """ & "*L*" & """))"

    Set rstDMR = dbsCurrent.OpenRecordset(strSQL)
    If rstDMR.RecordCount > 0 Then
        MsgBox (" A explosão da Ordem de Embarque já foi gerada, ou então apenas itens do Pedido de
Venda foram inseridos!")
        rstDMR.Close
        DoCmd.Hourglass False
        Exit Sub
    End If
    rstDMR.Close
    strSQL = "SELECT * FROM [Inventory Transactions] WHERE (([Inventory Transactions]![SalesOrderID] =
""" & strValue & """) AND ([Inventory Transactions].[TransferFromID] = 1) AND ([Inventory
Transactions].[Control] not Like """ & "*L*" & """)AND ([Inventory Transactions].[Control] not Like
""" & "*W*" & """))"

    Set rstDMR = dbsCurrent.OpenRecordset(strSQL)
    If rstDMR.RecordCount = 0 Then
        MsgBox (" Não há itens válidos para explodir! Pressione o botão 'Copiar itens'!")
        rstDMR.Close
        DoCmd.Hourglass False
        Exit Sub
    End If
    rstDMR.Close
dbsCurrent.Execute "DELETE * FROM ShipChk ;"

SSOObject.SalesOrderID = strValue
SSOObject.CustomerID = iCustomerID
SSOObject.DatabaseCurrent = dbsCurrent
SSOObject.ExplodeSSO
DoCmd.SetWarnings False
dbsCurrent.Execute ("UPDATE [ShipChk] SET ShipToID=" & iShipToID & ";")

Set rstDMR = dbsCurrent.OpenRecordset("SELECT OrderLine " _
  & "FROM [Inventory Transactions] " _
  & "WHERE SalesOrderID = """ & Me!SalesOrderID & """ " _
  & "AND (([Inventory Transactions].TransferFromID)=0 " _
  & "Or ([Inventory Transactions].TransferFromID)=1) " _
  & "AND (([Inventory Transactions].Control) Not Like """ & "*W*" & """) " _
  & "AND (([Inventory Transactions].Control) Not Like """ & "*L*" & """) " _
  & "AND (([Inventory Transactions].Control) Not Like """ & "*O*" & """) " _
  & "ORDER BY TransactionID;")

et rstDMR_sort = dbsCurrent.OpenRecordset("ShipChk", dbOpenTable)

stDMR.MoveFirst
stDMR_sort.MoveFirst
o
 Do
   rstDMR_sort.Edit
   rstDMR_sort!OrderLine = rstDMR!OrderLine
   rstDMR_sort.update
   rstDMR_sort.MoveNext
   If (rstDMR_sort.EOF) Then Exit Do
 Loop Until (rstDMR_sort!Level = 1)  'if Next!Level = 1 then Need to raise OrderLine #
 rstDMR.MoveNext
oop Until (rstDMR.EOF)
stDMR.Close
stDMR_sort.Close
```

```
DoCmd.OpenQuery "ShipChk_Append_vbl"
Get_PackingOrderID

DoCmd.Hourglass False
DoCmd.SetWarnings True
Forms![Sales_Shipping Order].Refresh
End Sub

Private Sub TglIncomplete_Click()

    IsIncomplete

End Sub
Private Sub CmdRMACommercial_Click()
On Error GoTo Err_CmdRMACommercial_Click

    Dim stDocName As String

    stDocName = "Commercial Invoice - RMA"
    DoCmd.OpenReport stDocName, acPreview, , "[Sales Orders]![SalesOrderID]=[Forms]![Shipping
Order]![SalesOrderID]"

Exit_CmdRMACommercial_Click:
    Exit Sub

Err_CmdRMACommercial_Click:
    MsgBox Err.Description
    Resume Exit_CmdRMACommercial_Click

End Sub
Private Sub CmdPreviewForPrinting_Click()
On Error GoTo Err_CmdPreviewForPrinting_Click

    Dim stDocName As String

    stDocName = "Shipping Orders"
    DoCmd.OpenReport stDocName, acViewNormal, , "[Sales Orders]![SalesOrderID]=[Forms]![Sales_Shipping
Order]![SalesOrderID]"

Exit_CmdPreviewForPrinting_Click:
    Exit Sub

Err_CmdPreviewForPrinting_Click:
    MsgBox Err.Description
    Resume Exit_CmdPreviewForPrinting_Click

End Sub
Private Sub CmdShippingSignoff_Click()
On Error GoTo Err_CmdShippingSignoff_Click

    Dim stDocName As String

    stDocName = "Shipping Signoff"
    DoCmd.OpenReport stDocName, acPreview

Exit_CmdShippingSignoff_Click:
    Exit Sub

Err_CmdShippingSignoff_Click:
    MsgBox Err.Description
    Resume Exit_CmdShippingSignoff_Click

End Sub


Private Sub CommandSHIP_Click()

On Error GoTo Err_CommandSHIP_Click
DoCmd.SetWarnings False

   Const NewLine = vbNewLine
Const TempRptFileName = "C:\TempRpt.TXT"
Const EmailColumn = 2
Const MsgErrorText = "Você não selecionou um destinatário para enviar a mensagem."

Dim strkMANumber As String
Dim FirstInfo As String
Dim strSalesOrderID As String
Dim iCustomerID As Long
Dim iShipToID As Long
```

```
Dim strCustomerName As String
Dim iShippingMethodID As Long
Dim strShippingMethod As String
Dim rstShip As DAO.Recordset
Dim dbsCurrent As DAO.Database
Dim ReportFile As Integer
Dim ReportString As String
Dim TempString As String
Dim EmailTo As String
Dim counter As Integer
Dim strSQL As String
Dim strValue As String
Dim rstDMR As DAO.Recordset
Dim iRecordNumber As Long


    Set dbsCurrent = CurrentDB()
    iCustomerID = Me!CustomerID
    iShipToID = Me!ShipToID
    counter = Me!StatusID
    strValue = Me![SalesOrderID]
    iRecordNumber = Me.CurrentRecord

     strSQL = "SELECT * FROM [Inventory Transactions] " _
     & "WHERE (([Inventory Transactions]![SalesOrderID] = """ & strValue & """) " _
     & "AND ([Inventory Transactions]![UnitsShippedFG] > 0 ) " _
     & "AND (([Inventory Transactions]![Control] Not Like """ & "*L*" & """) " _
     & "AND ([Inventory Transactions]![Control] Not Like """ & "*0*" & """) " _
     & "AND ([Inventory Transactions]![Control] Not Like """ & "*W*" & """))) " _
     & "ORDER BY TransactionID;"

    Set rstDMR = dbsCurrent.OpenRecordset(strSQL)
    If rstDMR.RecordCount = 0 Then
    MsgBox ("Não existem itens 'enviáveis'! Leia as instruções da guia 'Preparação Embarque'.")
    rstDMR.Close
    Exit Sub
    End If

    Select Case (Me!ShipmentModified)
    Case -1: 'hasn't been printed
    Case 0: 'hasn't been changed since printing, do nothing
    Case 1: 'has changed since printing, update Rev
      Check_Update_PackingOrderID
    End Select
    'Reset Modified because new shipment will be created after this one
    Me!ShipmentModified = -1


    rstDMR.MoveFirst
    Do
      If (rstDMR!TransferFromID = 1) Then 'Don't include exploded/field tracking items
        rstDMR.Edit
        If (IsNull(rstDMR!OrderLine)) Then
          rstDMR!UnitsReceived = Null
          rstDMR!UnitsReceivedFG = Null
        Else
          rstDMR!UnitsReceivedFG = rstDMR!UnitsShippedFG + Nz(DSum("UnitsShippedFG", "[Inventory
Transactions]",
            "SalesOrderID = """ & Me!SalesOrderID & """ AND ProductID = """ &
ReplaceString(rstDMR!ProductID, Chr(34), Chr(34) & Chr(34))
            & """ AND OrderLine = " & rstDMR!OrderLine & " AND Control NOT LIKE " & """*0*""" _
            & " AND Control NOT LIKE " & """*W*""" & " AND UnitsShipped = 0" _
            & " AND TransferFromID <> 10 AND TransactionID < " & rstDMR!TransactionID))
          rstDMR!UnitsReceived = rstDMR!UnitsReceivedFG
        End If
        rstDMR.update
      End If
      rstDMR.MoveNext
    Loop Until (rstDMR.EOF)
    rstDMR.Close
    'Update Inv. Transactions with items shipped
    DoCmd.Hourglass True

    DoCmd.SetWarnings False

    strValue = Forms![Sales_Shipping Order]!SalesOrderID
dbsCurrent.Execute "INSERT INTO [Inventory Transactions] ( TransactionDate,Control,ProductID," _
  & "SalesOrderID,UnitsOrdered, UnitsShipped,UnitsShippedFG," _
  & "TransferFromID,CustomerID,ShipToID,OrderLine,ShipFromID, PackingOrderID )" _
  & "SELECT Now() AS [TDate],""" & "T" & """,[ProductID]," _
  & "[SalesOrderID], [UnitsOrdered], [UnitsShippedFG],[UnitsShippedFG]," _
```

```
    & "1 AS [FromID], " & iCustomerID & " as [ToID], " & iShipToID & " as
[LocationToID],OrderLine,[Inventory Transactions].[ShipFromID], [PackingOrderID] " _
    & "FROM [Inventory Transactions]" _
    & "WHERE ([SalesOrderID] = """ & strValue & """ ) " _
    & "AND ([UnitsShippedFG] <> 0 ) " _
    & "AND ([Control] not Like """ & "*O*" & """) " _
    & "AND ([Control] not Like """ & "*W*" & """) " _
    & "AND ([Control] not Like """ & "*L*" & """) " _
    & "AND ([TransferFromID] = 1));"

'Transfer shipping details data to shipped records

    Const MsgText = "Você gostaria de limpar do formulário principal os valores copiados?"
    Const msgTitle = "Cópia Completa."

    Dim ClearMainForm As Integer
    Dim ShippedDate As Date
    Dim OrderDate As Date
    Dim ETADate As Date
    Dim DeliveredDate As Date
    Dim ShippedDateNull As Boolean
    Dim OrderDateNull As Boolean
    Dim ETADateNull As Boolean
    Dim DeliveredDateNull As Boolean
    Dim CustomerPO As String
    Dim SalesTracking As String
    Dim ShipperTracking As String
    Dim ShippingMethod As Long

dbsCurrent.Execute "UPDATE [Inventory Transactions] " _
& "SET [Control] = [Control] & """ & "L" & """ " _
& "WHERE (([Inventory Transactions]![SalesOrderID] = """ & strValue & """) " _
& "AND ([Inventory Transactions].[Control] not Like """ & "*L*" & """) " _
& "AND ([Inventory Transactions].[Control] not Like """ & "*W*" & """) " _
&  "AND  (([Inventory  Transactions].[Control]  not  Like  """  &  "*O*"  &  """)  OR  ([Inventory
Transactions].[Control]  Like """ & "*R*" & """)));"

 DoCmd.Hourglass False

Me.[Sales Orders.StatusID] = 3
Me.CommandSHIP.SetFocus
Me.ShippingOrder_Subform.Requery
MsgBox ("Alterações no invetário de embarque realizadas!!")
DoCmd.SetWarnings True

Exit_CommandSHIP_Click:
    DoCmd.Hourglass False
    DoCmd.SetWarnings True
    Exit Sub

Err_CommandSHIP_Click:
    MsgBox Err.Description
    Resume Exit_CommandSHIP_Click

End Sub
Private Sub Command354_Click()
On Error GoTo Err_Command354_Click


    Screen.PreviousControl.SetFocus
    DoCmd.FindNext

Exit_Command354_Click:
    Exit Sub

Err_Command354_Click:
    MsgBox Err.Description
    Resume Exit_Command354_Click

End Sub

Private Sub ShippingOrder_Subform_Enter()
If IsNull(Me!EmployeeID) Or IsNull(Me!ShippingMethodID) Or IsNull(Me!CustomerID) Then
    MsgBox ("Não é possível adicionar itens porque o campo 'Funcionário'/'Transportadora'/'End.
Entrega' está em branco.  " _
    & "For favor preencha todos os campos 'laranja' antes de adicionar items.")
    Me!CustomerID.SetFocus
End If
End Sub

Private Sub StatusID_Change()
```

```
  If (Me.StatusID < Me.StatusID.OldValue) Then
    If (vbCancel = MsgBox("Alterar o status para um anterior pode corromper seus dados.  Prosseguir
mesmo assim?", vbOKCancel, "Aviso!  Status Anterior")) Then
      Me.StatusID = Me.StatusID.OldValue
    End If
  End If
  StatusID.Requery
End Sub

Private Sub txtShippedDate1_Change()
Forms![Sales_Shipping Order]!txtShippedDate = Me!txtShippedDate1
End Sub

Private Sub txtShippedDate2_Change()
Me!txtShippedDate = Me!txtShippedDate2
End Sub

Private Sub Set_PackingOrderID()
Dim dbsCurrent As DAO.Database
Dim rstPKID As DAO.Recordset
Dim strPKID As String
Set dbsCurrent = CurrentDB

Set rstPKID = dbsCurrent.OpenRecordset("SELECT PackingOrderID " _
  & "FROM [Inventory Transactions] " _
  & "WHERE SalesOrderID = """ & Me!SalesOrderID & """ " _
  & "AND Control NOT LIKE " & """*0*""" & " " _
  & "AND Control NOT LIKE " & """*W*""" & " " _
  & "AND Control NOT LIKE " & """*L*""" & " " _
  & "AND PackingOrderID IS NOT NULL " _
  & "AND UnitsShipped = 0;")

If (rstPKID.RecordCount > 0) Then
  rstPKID.MoveFirst
  strPKID = rstPKID!PackingOrderID
Else
  strPKID = GetNextSN("PK")
  Me!ShipmentModified = -1
End If

dbsCurrent.Execute ("UPDATE [Inventory Transactions] " _
  & "SET PackingOrderID = """ & strPKID & """ " _
  & "WHERE SalesOrderID = """ & Me!SalesOrderID & """ " _
  & "AND Control NOT LIKE " & """*0*""" & " " _
  & "AND Control NOT LIKE " & """*W*""" & " " _
  & "AND Control NOT LIKE " & """*L*""" & " " _
  & "AND PackingOrderID IS NULL " _
  & "AND UnitsShipped = 0;")


If (Me!ShipmentModified = 0) Then Me!ShipmentModified = 1

Set dbsCurrent = Nothing
End Sub

Private Sub Check_Update_PackingOrderID(Optional Cancel As Integer = 0)
Dim dbsCurrent As DAO.Database
Dim rstPKID As DAO.Recordset

Set dbsCurrent = CurrentDB

Select Case (Me!ShipmentModified)
Case -1:
Case 0:
Case 1:
  Dim strOldRev As String
  Dim strNewRev As String

  Set rstPKID = dbsCurrent.OpenRecordset("SELECT PackingOrderID " _
    & "FROM [Inventory Transactions] " _
    & "WHERE SalesOrderID = """ & Me!SalesOrderID & """ " _
    & "AND Control NOT LIKE " & """*W*""" & " AND Control NOT LIKE " & """*0*""" & " " _
    & "AND Control NOT LIKE " & """*L*""" & "AND UnitsShipped = 0;")

  If (rstPKID.RecordCount > 0) Then
    rstPKID.MoveLast
    strOldRev = rstPKID!PackingOrderID
    rstPKID.Close

    strNewRev = GetNextRev(strOldRev)
```

```
    dbsCurrent.Execute "UPDATE [Inventory Transactions] " _
      & "SET PackingOrderID = """ & strNewRev & """ " _
      & "WHERE PackingOrderID = """ & strOldRev & """ " _
      & "AND Control NOT LIKE " & """*O*""" & " " _
      & "AND Control NOT LIKE " & """*W*""" & ";"

   Me!ShipmentModified = 0

    MsgBox "Novo Rev completado."
  End If '(rstPKID.RecordCount > 0)
End Select  '(Me!ShipmentModified)

Set dbsCurrent = Nothing

End Sub


Public Function RevOldShipment(strOldRev As String) As String
Dim strNewRev As String
Dim dbsCurrent As DAO.Database

Set dbsCurrent = CurrentDB
strNewRev = GetNextRev(strOldRev)

dbsCurrent.Execute "UPDATE [Inventory Transactions] " _
  & "SET PackingOrderID = """ & strNewRev & """ " _
  & "WHERE PackingOrderID = """ & strOldRev & """ " _
  & "AND Control NOT LIKE " & """*O*""" & " " _
  & "AND Control NOT LIKE " & """*W*""" & ";"

RevOldShipment = strNewRev
Set dbsCurrent = Nothing

Me.CmbPackingOrderID.Requery
Me.CmbPackingOrderID = strNewRev
Me.ShippingOrder_Subform.Requery
End Function


Private Sub Load_MeOpenArgs()
On Error GoTo Err_Load_MeOpenArgs
    Dim strParam As String
    Dim strParamID As String
    Dim strParamVal As String
    Dim strRemaining As String
    Dim strSalesOrderID As String

    strRemaining = Me.OpenArgs
    strSalesOrderID = ""

    Do While (Len(strRemaining) > 0)
        strParam = ParsePop(strRemaining, "#")
        If (Len(strParam) > 0) Then
            strParamVal = strParam
            strParamID = ParsePop(strParamVal, "=")
            Select Case (strParamID)
                Case "SO": 'SalesOrderID
                    If (Len(strParamVal) < 1) Then
                    ElseIf (IsNull(DLookup("[SalesOrderID]", "[Sales Orders]", "SalesOrderID = """ &
trParamVal & """"))) Then
                    Else
                        Me.ItemLookUP = strParamVal
                        Call ItemFind_Click
                        If (Me.SalesOrderID <> strParamVal) Then
                          MsgBox "Não foi possível encontrar o Pedido de Venda ou Ordem de Embarque =
"" & strParamVal & """", , "Pedido não Encontrado"
                        End If
                    End If
                Case Else:
                    'Do nothing
            End Select
        End If
    Loop
 Exit Sub
rr_Load_MeOpenArgs:
 Exit Sub
nd Sub
```

PEDIDO DE VENDAS

```
Option Compare Database
Option Explicit
Public strTmp As String
Public strdelete As String
Public itmp As Long
Public idelete As Long
Const cIncomplete = "Incomplete"
Const cComplete = "Complete"
Const DefaultEmailSelection = 1


Sub ChangeTextStatus(Change As Boolean)


End Sub
Sub IsComplete()

    Me![TglIncomplete].Value = False
    Me![TglIncomplete].Locked = False
    Me![TglIncomplete].Caption = cComplete
    Me![SalesOrderSubform].Locked = True
    ChangeTextStatus (True)

End Sub

Sub IsIncomplete()


End Sub

Sub ValidationCheck(Valid As Boolean, Complete As Boolean)

    Dim QtyRequired As Integer
    Dim QtyShipped As Integer
    Dim Count As Integer
    Dim ErrorFlag As Boolean
    Dim TransDate As Date
    Dim TransDateNull As Boolean
    Dim ShippedDate As Date
    Dim ShippedDateNull As Boolean
    Exit Sub

    Me![SalesOrderSubform].SetFocus
    DoCmd.GoToRecord , , acFirst

    ErrorFlag = False
    Complete = True
    Count = 0

    While (Not Me![SalesOrderSubform].Form.NewRecord) And (Not ErrorFlag)

        If IsNull(Me![SalesOrderSubform]![QtyRequired]) Then
            QtyRequired = 0
        Else
            QtyRequired = Val(Me![SalesOrderSubform]![QtyRequired])
        End If

        If IsNull(Me![SalesOrderSubform]![QtyShipped]) Then
            QtyShipped = 0
        Else
            QtyShipped = Val(Me![SalesOrderSubform]![QtyShipped])
        End If

        If (Me![SalesOrderSubform]![TransactionDate] <> "") Then
            TransDate = Format(Me![SalesOrderSubform]![TransactionDate])
            TransDateNull = False
        Else
            TransDateNull = True
        End If

        If (Me![SalesOrderSubform]![ShippedDate] <> "") Then
            ShippedDate = Format(Me![SalesOrderSubform]![ShippedDate])
            ShippedDateNull = False
        Else
            ShippedDateNull = True
        End If

        If (Not ShippedDateNull) And (Not TransDateNull) Then
            If (ShippedDate < TransDate) Then
```

```
                ErrorFlag = True
                Me![SalesOrderSubform]![TransactionDate].SetFocus
            End If
        End If
        If (QtyRequired < 0) Then
            ErrorFlag = True
            Me![SalesOrderSubform]![QtyRequired].SetFocus
        End If
        If (QtyShipped <> QtyRequired) And (QtyShipped <> 0) Then
            ErrorFlag = True
            Me![SalesOrderSubform]![QtyShipped].SetFocus
        End If
        If (ShippedDateNull = False) And ((QtyRequired > QtyShipped) Or (QtyRequired = 0)) Then
            ErrorFlag = True
            Me![SalesOrderSubform]![ShippedDate].SetFocus
        End If
        If (ShippedDateNull = True) And (QtyRequired = QtyShipped) And (QtyRequired <> 0) Then
            ErrorFlag = True
            Me![SalesOrderSubform]![ShippedDate].SetFocus
        End If
        If (ShippedDateNull = True) Or (QtyRequired <> QtyShipped) Then Complete = False
        If (QtyRequired = 0) Then Complete = False

        Count = Count + 1
        If ErrorFlag = False Then DoCmd.GoToRecord , , acNext

    Wend

    If Count = 0 Then
        Valid = False
    Else
        Valid = Not ErrorFlag
    End If

    If (Valid = False) Then Complete = False

End Sub

Sub ChangeButtons(Change As Boolean)
    Change = True
End Sub

Private Sub CmdAutoShip_Click()

    Dim TempDate As Date

        TempDate = Me![txtShippedDate]

        Me![SalesOrderSubform].SetFocus
        DoCmd.GoToRecord , , acFirst

        While (Not Me![SalesOrderSubform].Form.NewRecord)

            If                    (IsNull(Me![SalesOrderSubform]![ShippedDate]))                    And
(Me![SalesOrderSubform]![ShippedDate].Locked = False) Then
                Me![SalesOrderSubform]![ShippedDate] = Format(TempDate, "Medium Date")
            End If
            DoCmd.GoToRecord , , acNext
        Wend

        DoCmd.GoToRecord , , acLast
End Sub

Private Sub CmdLock_Click()
    Dim strValue As String
    Dim dbsCurrent As DAO.Database

  Set dbsCurrent = CurrentDB()
    Me![SalesOrderSubform].SetFocus

    strValue = Me![SalesOrderID]

    dbsCurrent.Execute "UPDATE [Inventory Transactions] SET [Control] = [Control] & """ & "L" & """
WHERE (([Inventory Transactions]![SalesOrderID] = """ & strValue & """) AND ([Inventory
Transactions].[Control] not Like """ & "*L*" & """));"

End Sub

Private Sub CmdTransfer_Click()

    Const MsgText = "Você gostaria de limpar o formulário principal dos itens copiados?"
```

```
    Const msgTitle = "Cópia Completa."

    Dim ClearMainForm As Integer
    Dim ShippedDate As Date
    Dim OrderDate As Date
    Dim ETADate As Date
    Dim DeliveredDate As Date
    Dim ShippedDateNull As Boolean
    Dim OrderDateNull As Boolean
    Dim ETADateNull As Boolean
    Dim DeliveredDateNull As Boolean
    Dim CustomerPO As String
    Dim SalesTracking As String
    Dim ShipperTracking As String
    Dim ShippingMethod As Long

        If Not IsNull(Me![txtShippedDate]) Then ShippedDate = Me![txtShippedDate]
        ShippedDateNull = IsNull(Me![txtShippedDate])

        If Not IsNull(Me![OrderDate]) Then OrderDate = Me![OrderDate]
        OrderDateNull = IsNull(Me![OrderDate])

        If Not IsNull(Me![ETADate]) Then ETADate = Me![ETADate]
        ETADateNull = IsNull(Me![ETADate])

        If Not IsNull(Me![DeliveredDate]) Then DeliveredDate = Me![DeliveredDate]
        DeliveredDateNull = IsNull(Me![DeliveredDate])

        If Not IsNull(Me![CustomerPONumber]) Then
            CustomerPO = Me![CustomerPONumber]
        Else
            CustomerPO = ""
        End If

        If Not IsNull(Me![SalesTrackingNumber]) Then
            SalesTracking = Me![SalesTrackingNumber]
        Else
            SalesTracking = ""
        End If

        If Not IsNull(Me![ShipperTrackingNumber]) Then
            ShipperTracking = Me![ShipperTrackingNumber]
        Else
            ShipperTracking = ""
        End If

        If Not IsNull(Me![ShippingMethodID]) Then
            ShippingMethod = Me![ShippingMethodID]
        Else
            ShippingMethod = 0
        End If

        Me![SalesOrderSubform].SetFocus
        DoCmd.GoToRecord , , acFirst

        While (Not Me![SalesOrderSubform].Form.NewRecord)

            If (Me![SalesOrderSubform]![ShippedDate].Locked = False) Then

                If Not ShippedDateNull And IsNull(Me![SalesOrderSubform]![ShippedDate]) Then
                    Me![SalesOrderSubform]![ShippedDate] = Format(ShippedDate, "Medium Date")
                End If

                If Not OrderDateNull And IsNull(Me![SalesOrderSubform]![OrderedDate]) Then
                    Me![SalesOrderSubform]![OrderedDate] = Format(OrderDate, "Medium Date")
                End If

                If Not ETADateNull And IsNull(Me![SalesOrderSubform]![ETADate]) Then
                    Me![SalesOrderSubform]![ETADate] = Format(ETADate, "Medium Date")
                End If

                If Not DeliveredDateNull And IsNull(Me![SalesOrderSubform]![ETADate]) Then
                    Me![SalesOrderSubform]![DeliveredDate] = Format(DeliveredDate, "Medium Date")
                End If

                If                    IsNull(Me![SalesOrderSubform]![CustomerPO])                    Then
e![SalesOrderSubform]![CustomerPO] = CustomerPO
                If                   IsNull(Me![SalesOrderSubform]![SalesTracking])                  Then
e![SalesOrderSubform]![SalesTracking] = SalesTracking
                If                 IsNull(Me![SalesOrderSubform]![ShippingTracking])                 Then
e![SalesOrderSubform]![ShippingTracking] = ShipperTracking
```

```
            If                  IsNull(Me![SalesOrderSubform]![ShippingMethod])                  Then
Me![SalesOrderSubform]![ShippingMethod] = ShippingMethod

            End If

            DoCmd.GoToRecord , , acNext

        Wend

        DoCmd.GoToRecord , , acLast

        ClearMainForm = MsgBox(MsgText, vbYesNo + vbQuestion + vbDefaultButton2, msgTitle)

        If ClearMainForm = vbYes Then

            Me![txtShippedDate].SetFocus
            Me![txtShippedDate].text = ""

            Me![OrderDate].SetFocus
            Me![OrderDate].text = ""

            Me![ETADate].SetFocus
            Me![ETADate].text = ""

            Me![DeliveredDate].SetFocus
            Me![DeliveredDate].text = ""

            Me![CustomerPONumber].SetFocus
            Me![CustomerPONumber].text = ""

            Me![SalesTrackingNumber].SetFocus
            Me![SalesTrackingNumber].text = ""

            Me![ShipperTrackingNumber].SetFocus
            Me![ShipperTrackingNumber].text = ""

            Me![ShippingMethodID].SetFocus
            Me![ShippingMethodID].text = ""

        End If

    Dim strValue As String
    Dim dbsCurrent As Database

  Set dbsCurrent = CurrentDB()

    strValue = Me![SalesOrderID]

    dbsCurrent.Execute "UPDATE [Inventory Transactions] SET [Control] = [Control] & """ & "L" & """
WHERE  (([Inventory  Transactions]![SalesOrderID]  =  """  &  strValue  &  """)  AND  ([Inventory
Transactions].[Control] not Like """ & "*L*" & """));"

End Sub

Private Sub CmdValCheck_Click()

    Dim Valid As Boolean
    Dim Finished As Boolean

    Const ErrorMessage = "O registro atual não é válido."
    Const ErrorTitle = "Erro!"

    ValidationCheck Valid, Finished
    Valid = True
    If (Valid = False) Then
        MsgBox ErrorMessage, vbOKOnly + vbExclamation, ErrorTitle
        ChangeButtons (False)
    Else
        DoCmd.GoToRecord , , acLast
        ChangeButtons (True)

    End If

    If (Finished = True) Then
        IsComplete
    Else
        IsIncomplete
    End If

End Sub
```

```
Private Sub Delete_Sales_Orders_Record_and_subform_records_Click()
Dim dbsCurrent As DAO.Database
Dim strSalesOrderID As String
Dim wrkJet As DAO.Workspace
  Set dbsCurrent = CurrentDB()

  strSalesOrderID = Forms![Shipping Order]!SalesOrderID
  DoCmd.Hourglass True
    dbsCurrent.Execute "DELETE * FROM [Sales Orders] WHERE ([Sales Orders].[SalesOrderID]=""" &
strSalesOrderID & """);"
    dbsCurrent.Execute "DELETE * FROM [Inventory Transactions] WHERE ([Inventory
Transactions].[SalesOrderID] = """ & strSalesOrderID & """);"
    DoCmd.Hourglass False


DoCmd.Save acForm, "Sales Order"
DoCmd.GoToControl "SalesOrderID"
DoCmd.GoToRecord , , acFirst
End Sub

Private Sub Command356_Click()
Dim orderLineCount As Integer
Dim childLineCount As Integer
Dim strSQL As String
Dim rstInvTrans As DAO.Recordset
Dim dbsCurrent As DAO.Database

If (Me.StatusID1 <> 2) Then
    MsgBox "As Ordens de Produção só podem ser criadas quando o status do pedido de venda atual for
""" & "Liberado" & """."
    Exit Sub
End If

If (IsNull(Me.CustomerID)) Then
    MsgBox "Por favor selecione um cliente para este Pedido de Venda."
    Exit Sub
End If

If (IsNull(Me.EmployeeID)) Then
    MsgBox "Por favor selecione um funcionário para este Pedido de Venda."
    Exit Sub
End If

Set dbsCurrent = CurrentDB()
strSQL = "SELECT [Inventory Transactions].Control, [Inventory Transactions].SalesOrderID " _
    & "FROM [Inventory Transactions] " _
    & "WHERE ([Inventory Transactions].[SalesOrderID] LIKE """ & "*-" & SalesOrderID & """);"
Set rstInvTrans = dbsCurrent.OpenRecordset(strSQL, dbOpenForwardOnly)
orderLineCount = rstInvTrans.RecordCount
If (orderLineCount > 0) Then
    MsgBox "Já foram geradas Ordens de Produção para este Pedido de Venda."
    Exit Sub
End If

strSQL = "SELECT [Inventory Transactions].Control, [Inventory Transactions].SalesOrderID, [Inventory
Transactions].PackingOrderID " _
    & "FROM [Inventory Transactions] " _
    & "WHERE ([Inventory Transactions].[SalesOrderID] = """ & SalesOrderID & """) " _
    & "AND ([Inventory Transactions].[PackingOrderID] Is Null " _
    & "AND [Inventory Transactions].Control NOT LIKE ""*V*"");"
Set rstInvTrans = dbsCurrent.OpenRecordset(strSQL, dbOpenForwardOnly)
orderLineCount = rstInvTrans.RecordCount
If (orderLineCount = 0) Then
    MsgBox "Não existem pedidos prévios neste Pedido de Venda para criar Ordens de Produção sub-
sequentes."
    Exit Sub
End If

strSQL = "SELECT [Inventory Transactions].Control, [Inventory Transactions].SalesOrderID, [Inventory
Transactions].PackingOrderID " _
    & "FROM [Inventory Transactions] " _
    & "WHERE ([Inventory Transactions].[SalesOrderID] = """ & SalesOrderID & """) " _
    & "AND ([Inventory Transactions].[PackingOrderID] Is Null " _
    & "AND [Inventory Transactions].Control LIKE ""*V*"");"
Set rstInvTrans = dbsCurrent.OpenRecordset(strSQL, dbOpenForwardOnly)
childLineCount = rstInvTrans.RecordCount

rstInvTrans.Close
dbsCurrent.Close
```

```
   If (orderLineCount > 0) Then
     If (childLineCount > 0) Then
       Call CreateWOsFromSO_Pkg(Me!SalesOrderID)
     ElseIf (childLineCount = 0) Then
       Call CreateWOsFromSO_Std(Me!SalesOrderID)
     End If
   End If
End Sub

Private Sub CustomerID_AfterUpdate()
Dim dbsCurrent As DAO.Database
Dim strTmp As String
If (IsNull(Me.CustomerID.OldValue) Or IsNull(Me.CustomerID)) Then Exit Sub

strTmp = Me.CustomerID.OldValue  'get old value before committing
DoCmd.RunCommand acCmdSaveRecord 'Commit changes in buffer to database
Set dbsCurrent = CurrentDB
dbsCurrent.Execute ("UPDATE [Inventory Transactions] " _
  & "SET CustomerID = " & Me.CustomerID & " " _
  & "WHERE SalesOrderID = """ & Me!SalesOrderID & """ " _
  & "AND CustomerID = " & strTmp & ";")

Set dbsCurrent = Nothing
If (strTmp <> Me.CustomerID) Then
  Me.ShipToID.Requery
  On Error Resume Next
  Me!ShipToID = DLookup("CustomerExtraID", "CustomersExtraShipTo", "CustomerID=" & Me!CustomerID & "
AND Default=-1")
End If
End Sub

Private Sub Form_BeforeDelConfirm(Cancel As Integer, Response As Integer)

    Response = acDataErrContinue
    Dim strValue As String
    strValue = Left(strdelete, 2)
    If strValue <> "SO" Then
        MsgBox ("Apenas Pedidos de Venda do tipo SOxxxx podem ser apagados a partir do Formulário de
PV.")
        DoCmd.CancelEvent
        Exit Sub
    End If

    strValue = strdelete
    Dim iValue As Long
    iValue = idelete

    strValue = DLookup("[Status]", "Status_WO", "StatusID = " & iValue)
    If strValue <> "Planned" Then
        MsgBox ("Um Pedido de Venda não pode ser apagado depois de ter sido liberado.")
        DoCmd.CancelEvent
        Exit Sub
    End If
    If MsgBox("Pedido de Venda (Estimado) " & strdelete & vbCrLf & vbCrLf & "Apagar este Registro?",
vbOKCancel) = vbCancel Then
        Cancel = True
        Exit Sub
    End If
    Dim dbsCurrent As DAO.Database
    Dim strSalesOrderID As String
    Dim wrkJet As DAO.Workspace
     Set dbsCurrent = CurrentDB()

   On Error Resume Next

   strSalesOrderID = strdelete
   DoCmd.Hourglass True
   dbsCurrent.Execute "DELETE * FROM [WO_SFC_Routings] WHERE ([WO_SFC_Routings].[SalesOrderID] LIKE
"" & "*-" & strSalesOrderID & """);"
   dbsCurrent.Execute     "DELETE     *     FROM     [Inventory     Transactions]     WHERE     ([Inventory
ransactions].[SalesOrderID] LIKE """ & "*-" & strSalesOrderID & """);"

   dbsCurrent.Execute "DELETE * FROM [WO_SFC_Routings] WHERE ([WO_SFC_Routings].[SalesOrderID] = """
strSalesOrderID & """);"
   dbsCurrent.Execute     "DELETE     *     FROM     [Inventory     Transactions]     WHERE     ([Inventory
ransactions].[SalesOrderID] = """ & strSalesOrderID & """);"
   DoCmd.Hourglass False

nd Sub
```

```
Private Sub Form_AfterDelConfirm(Status As Integer)
    Select Case Status
        Case acDeleteOK
            Dim strSalesOrderID As String
            strSalesOrderID = strdelete
            Dim dbsCurrent As DAO.Database
            Set dbsCurrent = CurrentDB()
            dbsCurrent.Execute "DELETE * FROM [Sales Orders] WHERE ([Sales Orders].[SalesOrderID] LIKE
""" & "*-" & strSalesOrderID & """);"
            Set dbsCurrent = Nothing
            MsgBox "O Pedido foi apagado sem problemas."
        Case acDeleteCancel

MsgBox "Usuário cancelou o processo. O Pedido não foi apagado."
        Case acDeleteUserCancel
            MsgBox "Usuário cancelou o processo."
    End Select
End Sub


Private Sub Form_BeforeUpdate(Cancel As Integer)
If (Me!StatusID1.Value <> 9 And Me!StatusID1.Value <> 5) Then
    If Me!StatusID1.OldValue > 2 Then
        Dim strMsg As String
        If (vbCancel = MsgBox("Mudar dados do seu Pedido de Venda agora pode invalidar Ordens de
Produção e Ordens de Embarque associadas a ele. Clique OK para prosseguir com a alteração.")) Then
         Me.Undo
        End If
    End If
End If
End Sub

Private Sub Form_Close()
    DoCmd.OpenForm "Menu Inicial"

End Sub

Private Sub Form_Current()

    Dim counter As Integer
    Dim Valid As Boolean
    Dim Finished As Boolean

    strTmp = iif(IsNull(Me!SalesOrderID), "Null", Me!SalesOrderID)
    itmp = iif(IsNull(Me!StatusID1), 1, Me!StatusID1)

    Exit Sub
    ValidationCheck Valid, Finished

    DoCmd.GoToRecord , , acFirst

    Me!SalesOrderID.SetFocus

    If (Not Valid) Or (Me.NewRecord = True) Then
        Me![TglIncomplete].Value = True
    Else
        ChangeButtons (True)
    End If
    If (Me![TglIncomplete].Value = True) Then
        IsIncomplete
    Else
        IsComplete
    End If
 Me.ShipToID.Requery
nd Sub

rivate Sub Form_Delete(Cancel As Integer)
trTmp = Me!SalesOrderID
trdelete = strTmp
tmp = Me!StatusID1
delete = itmp
nd Sub

rivate Sub Form_Open(Cancel As Integer)

   Dim Valid As Boolean
   Dim Finished As Boolean

   Const ErrorMessage = "O registro atual não é válido."
   Const ErrorTitle = "Registro Inválido"
       ChangeButtons (True)
```

```
End Sub

Private Sub ItemFind_Click()
If Not IsNull([ItemLookUP]) Then
    SalesOrderID.SetFocus
    DoCmd.FindRecord [ItemLookUP]
    ItemLookUP.SetFocus
Else
    MsgBox "Por favor selecione um Pedido de Venda para buscar."
End If
End Sub

Private Sub SalesOrderID_BeforeUpdate(Cancel As Integer)
    Dim decision As Integer
    Dim prefix As String
    Dim strValue As String
    Dim strSQL As String
    Dim rstID As DAO.Recordset
    Dim dbsCurrent As Database

    Set dbsCurrent = CurrentDb()
    decision = MsgBox("Tem certeza de que quer alterar o Código deste Pedido de Venda?", vbYesNo)
    If decision = 7 Then
        Me.Undo
    Else
        If IsNull(Me!SalesOrderID) Then
            MsgBox "Não é possível substituir o código do pedido por um valor nulo", vbOKOnly
            Me.Undo
            Exit Sub
        End If
        prefix = Left(Forms!SalesOrder!SalesOrderID.Value, 2)
        If prefix <> "SO" Then
            MsgBox "Este código é inválido. Ele deve começar com 'SO'.", vbOKOnly
            Me.Undo
        Else
            strValue = Me!SalesOrderID
            strSQL = "SELECT * FROM [Sales Orders] WHERE ([Sales Orders]![SalesOrderID] = """ &
strValue & """)"
            Set rstID = dbsCurrent.OpenRecordset(strSQL)
            If rstID.RecordCount <> 0 Then
                MsgBox "Este código já existe. Por favor escolha outro.", vbOKOnly
                Me.Undo
            End If
            rstID.Close
        End If
    End If
End Sub

Private Sub SalesOrderID_LostFocus()
On Error Resume Next
strTmp = Me!SalesOrderID
End Sub

Private Sub SalesOrderSubform_Enter()

If    (IsNull(Me!EmployeeID)    Or    IsNull(Me!ShippingMethodID)    Or    IsNull(Me!CustomerID)    Or
IsNull(Me.ShipToID)) Then
    MsgBox ("Não é possível adicionar um item porque um dos campos Funcoinário/Cliente/End. Entrega
está em branco.  "
    & "Por favor preencha todos os campos em laranja antes de adicionar itens ao pedido.")
    Me!CustomerID.SetFocus
End If
End Sub

Private Sub SalesOrderSubform_Exit(Cancel As Integer)

    If (Me![SalesOrderSubform].Locked = False) Then ChangeButtons (False)

End Sub

Private Sub ShipToID_GotFocus()
  Me.ShipToID.Requery
End Sub

Private Sub StatusID1_Change()
StatusID1.Requery
End Sub
```

PLANEJAMENTO DE DEMANDAS

```
Option Compare Database

Const conBasic = False
Const conAdvanced = True

Private Sub cmdRefreshData_Click()
    Me.Refresh
End Sub

Private Sub DemandPeriodCount_AfterUpdate()
    Dim dbsCurrent As DAO.Database
    Dim rstDemandPeriodsToAdd As DAO.Recordset
    Dim rstDemandProductsToAdd As DAO.Recordset
    Dim intLoop As Integer
    Dim strCurrentProductID As String

    DoCmd.Hourglass True
    Set dbsCurrent = CurrentDB

    If Me.DemandPeriodCount < Me.DemandPeriodCount.OldValue Then
        dbsCurrent.Execute "DELETE Demands.*" _
                           & " FROM Demands" _
                           & " WHERE (((Demands.SchedulingPlanID)=" & Me.SchedulingPlanID & ") AND
((Demands.PeriodID)>" & Me.DemandPeriodCount & "));"
    Else
        Set rstDemandProductsToAdd = dbsCurrent.OpenRecordset("SELECT Demands.ProductID " _
                                                              & "FROM Demands" _
                                                              & " WHERE (((Demands.SchedulingPlanID)="
& Me.SchedulingPlanID & ") AND ((Demands.PeriodID)=0));")
        Set rstDemandPeriodsToAdd = dbsCurrent.OpenRecordset("Demands")

        With rstDemandProductsToAdd
            Do While Not .EOF
                strCurrentProductID = ![ProductID]
                With rstDemandPeriodsToAdd
                    For intLoop = Me.DemandPeriodCount.OldValue + 1 To Me.DemandPeriodCount
                        .AddNew
                        ![PeriodID] = intLoop
                        ![ProductID] = strCurrentProductID
                        ![SchedulingPlanID] = Me.SchedulingPlanID
                        .Update
                    Next intLoop
                End With
            .MoveNext
            Loop
        End With
    End If

    [Forms]![SchedulingPlans]![Demands_subform].[Form].Requery
    Set rstDemandProductsToAdd = Nothing
    Set rstDemandPeriodsToAdd = Nothing
    Set dbsCurrent = Nothing
    DoCmd.Hourglass False
nd Sub

rivate Sub DemandPeriodUnit_AfterUpdate()
    If Me.TimeSelectionType = conAdvanced Then
        If Me.DemandPeriodUnit = 2 Or Me.DemandPeriodUnit = 3 Then
            AdvancedDemandPeriodUnitSize = 1
            Me.AdvancedDemandPeriodUnitSize.Enabled = False
        Else
            Me.AdvancedDemandPeriodUnitSize.Enabled = True
        End If
    End If
ıd Sub

ivate Sub Demands_ProductSelect_subform_Enter()
    If IsNull(Me.SchedulingPlanName) Then
        MsgBox "Favor preencha o nome do planejamento primeiramente"
        Me.SchedulingPlanName.SetFocus
    End If
ıd Sub

ivate Sub Export_Click()
    If IsNull(Me.SchedulingPlanName) Then
        MsgBox "Por favor preencha o nome do planejamento primeiramente"
        Me.SchedulingPlanName.SetFocus
    Else
        DoCmd.RunCommand acCmdSaveRecord
```

```
        PeriodCF = DemandPeriodConversionFactor()
        OpCF = OperationTimeConversionFactor()
        Call Export_Main
    End If
End Sub

Private Sub FillAll_Click()
    If Not (IsNull(Me.AllQuantity) Or IsNull(Me.CurrentProductID) Or IsNull(Me.SchedulingPlanID)) Then
        Dim dbsCurrent As DAO.Database
        Set dbsCurrent = CurrentDB
        dbsCurrent.Execute "UPDATE Demands SET Demands.Quantity = " & Me.AllQuantity _
                           & " WHERE (((Demands.ProductID)=" & Quote(Me.CurrentProductID) & ") AND
((Demands.SchedulingPlanID)=" & Me.SchedulingPlanID & "));"
        Set dbsCurrent = Nothing
        [Forms]![SchedulingPlans]![Demands_ProductSelect_subform].[Form].Refresh
        [Forms]![SchedulingPlans]![Demands_subform].[Form].Refresh
    End If
End Sub

Private Sub Form_BeforeInsert(Cancel As Integer)
    DemandPeriodUnit = DLookup("[DemandPeriodUnit]", "TimeDefinition")
End Sub

Private Sub Form_Current()
    Call TimeSelectionType_AfterUpdate
    Call DemandPeriodUnit_AfterUpdate
End Sub

Private Sub TimeSelectionType_AfterUpdate()
    BasicDemandPeriodUnitSize.Enabled = Not TimeSelectionType
End Sub

Function DemandPeriodConversionFactor() As Double

    Dim blnDefaultTimeSelectionType As Boolean
    Dim lngDefaultDemandPeriodUnitSize As Long
    Dim intDefaultDemandPeriodUnit As Integer
    Dim intDefaultOperationTimeUnit As Integer
    Dim strDefaultUnit As String
    Dim strCurrentUnit As String

    blnDefaultTimeSelectionType = DLookup("[TimeSelectionType]", "TimeDefinition")
    lngDefaultDemandPeriodUnitSize = DLookup("[DemandPeriodUnitSize]", "TimeDefinition")
    intDefaultDemandPeriodUnit = DLookup("[DemandPeriodUnit]", "TimeDefinition")
    intDefaultOperationTimeUnit = DLookup("[OperationTimeUnit]", "TimeDefinition")

    If blnDefaultTimeSelectionType = conAdvanced Then
        strDefaultUnit = DemandPeriodUnitName(intDefaultDemandPeriodUnit)
    Else
        strDefaultUnit = OperationTimeUnitName(intDefaultOperationTimeUnit)
    End If

    If TimeSelectionType = conAdvanced Then
        strCurrentUnit = DemandPeriodUnitName(DemandPeriodUnit)
    Else
        strCurrentUnit = OperationTimeUnitName(OperationTimeUnit)
    End If

    DemandPeriodConversionFactor = TimeConversionFactor(lngDefaultDemandPeriodUnitSize,
trDefaultUnit, DemandPeriodUnitSize, strCurrentUnit)

nd Function

unction OperationTimeConversionFactor() As Double
    Dim strDefaultUnit As String
    Dim strCurrentUnit As String

    strDefaultUnit = OperationTimeUnitName(DLookup("[OperationTimeUnit]", "TimeDefinition"))
    strCurrentUnit = OperationTimeUnitName(OperationTimeUnit)
    OperationTimeConversionFactor = TimeConversionFactor(1, strDefaultUnit, 1, strCurrentUnit)
nd Function

rivate Sub SalvaRegistro_Click()
n Error GoTo Err_SalvaRegistro_Click

    DoCmd.DoMenuItem acFormBar, acRecordsMenu, acSaveRecord, , acMenuVer70

xit_SalvaRegistro_Click:
    Exit Sub

rr_SalvaRegistro_Click:
```

```
    MsgBox Err.Description
    Resume Exit_SalvaRegistro_Click

End Sub
Private Sub AdicionaRegistro_Click()
On Error GoTo Err_AdicionaRegistro_Click
    DoCmd.GoToRecord , , acNewRec
    SchedulingPlanName.SetFocus
Exit_AdicionaRegistro_Click:
    Exit Sub
Err_AdicionaRegistro_Click:
    MsgBox Err.Description
    Resume Exit_AdicionaRegistro_Click
End Sub

Private Sub ProximoRegistro_Click()
On Error GoTo Err_ProximoRegistro_Click
    DoCmd.GoToRecord , , acNext
Exit_ProximoRegistro_Click:
    Exit Sub
Err_ProximoRegistro_Click:
    MsgBox Err.Description
    Resume Exit_ProximoRegistro_Click
End Sub

Private Sub RegistroAnterior_Click()
On Error GoTo Err_RegistroAnterior_Click
    DoCmd.GoToRecord , , acPrevious
Exit_RegistroAnterior_Click:
    Exit Sub
Err_RegistroAnterior_Click:
    MsgBox Err.Description
    Resume Exit_RegistroAnterior_Click
End Sub

Private Sub Fechar_Click()
On Error GoTo Err_Fechar_Click

    DoCmd.Close , , acSavePrompt
    DoCmd.OpenForm "Menu Inicial"
Exit_Fechar_Click:
    Exit Sub
Err_Fechar_Click:
    MsgBox Err.Description
    Resume Exit_Fechar_Click

End Sub
```

START UP

```
Option Compare Database
Option Explicit


Private Sub cmdCloseForm_Click()
    DoCmd.Close
End Sub

Private Sub Form_Activate()
    DoCmd.ShowToolbar "Form View", acToolbarNo
End Sub

Private Sub Form_Deactivate()
    DoCmd.ShowToolbar "Form View", acToolbarWhereApprop
End Sub

Private Sub Form_Open(Cancel As Integer)
    If IsLoaded("Menu Inicial") Or IsLoaded("SolutionsHyperForm") Then
        cmdCloseForm.Visible = True
    Else

        cmdCloseForm.Visible = False
        Me.TimerInterval = 4800
    End If
End Sub

Private Sub Form_Timer()
     Dim iMonth As Integer
     Dim vKey As Variant

    If Me.TimerInterval <> 0 Then
        Me.TimerInterval = 0
    End If
    Dim lData As Boolean
    lData = DisplayStartup
    If DisplayStartup = True Then
       DoCmd.OpenForm "EULA"

    Else

        DoCmd.OpenForm "Timer", , , , , acHidden
    End If
    DoCmd.Close acForm, "Startup"
End Sub
```

DEFINIÇÃO DE PERÍODOS

```
Option Compare Database
Option Explicit

Const conBasic = False
Const conAdvanced = True

Dim blnOldTimeSelectionType As Boolean
Dim lngOldDemandPeriodUnitSize As Long
Dim intOldDemandPeriodUnit As Integer
Dim intOldOperationTimeUnit As Integer
Dim intUserAns As Integer

Private Sub cmdSave_Click()
    Me.Refresh
End Sub

Private Sub DemandPeriodUnit_AfterUpdate()
    If Me.TimeSelectionType = conAdvanced Then
        If Me.DemandPeriodUnit = 2 Or Me.DemandPeriodUnit = 3 Then
            Me.AdvancedDemandPeriodUnitSize = 1
            Me.AdvancedDemandPeriodUnitSize.Enabled = False
        Else
            Me.AdvancedDemandPeriodUnitSize.Enabled = True
        End If
    End If
End Sub


Private Sub Form_AfterUpdate()
    If intUserAns = vbYes Then
        Call DoConversion
    End If
End Sub

Private Sub Form_BeforeUpdate(Cancel As Integer)

    intUserAns = vbNo

    blnOldTimeSelectionType = DLookup("[TimeSelectionType]", "TimeDefinition")
    lngOldDemandPeriodUnitSize = DLookup("[DemandPeriodUnitSize]", "TimeDefinition")
    intOldDemandPeriodUnit = DLookup("[DemandPeriodUnit]", "TimeDefinition")
    intOldOperationTimeUnit = DLookup("[OperationTimeUnit]", "TimeDefinition")

    If blnOldTimeSelectionType <> Me.TimeSelectionType _
       Or lngOldDemandPeriodUnitSize <> Me.DemandPeriodUnitSize _
       Or intOldDemandPeriodUnit <> Me.DemandPeriodUnit _
       Or intOldOperationTimeUnit <> Me.OperationTimeUnit Then
        'Prompt user for confirmation
        intUserAns = MsgBox("Estas modificações na Definição de período vão afetar os dados já
egistrados." & vbCrLf & "Você quer converter automaticamente todos os dados registrados conforme a
ova definição?", vbYesNoCancel + vbQuestion, "AVISO DE MODIFICAÇÃO")
        If intUserAns = vbCancel Then
            Cancel = True
            Me.Undo
        End If
    End If

nd Sub

rivate Sub Form_Current()
    Call TimeSelectionType_AfterUpdate
    Call DemandPeriodUnit_AfterUpdate
nd Sub

rivate Sub Form_Open(Cancel As Integer)
    Dim rstForm As Recordset
    Set rstForm = Me.RecordsetClone
    With rstForm
        If .RecordCount = 0 Then
            .AddNew
            .update
            Me.Refresh
        End If
    End With
d Sub

rivate Sub TimeSelectionType_AfterUpdate()
    BasicDemandPeriodUnitSize.Enabled = Not TimeSelectionType
    AdvancedDemandPeriodUnitSize.Enabled = TimeSelectionType
```

```
    DemandPeriodUnit.Enabled = TimeSelectionType
End Sub

Sub DoConversion()

    Dim strOldDemandUnit As String
    Dim strNewDemandUnit As String
    Dim strOldOpUnit As String
    Dim strNewOpUnit As String
    Dim dblDemandCF As Double
    Dim dblOpCF As Double

    If blnOldTimeSelectionType = conAdvanced Then
        strOldDemandUnit = DemandPeriodUnitName(intOldDemandPeriodUnit)
    Else
        strOldDemandUnit = OperationTimeUnitName(intOldOperationTimeUnit)
    End If
    If Me.TimeSelectionType = conAdvanced Then
        strNewDemandUnit = DemandPeriodUnitName(Me.DemandPeriodUnit)
    Else
        strNewDemandUnit = OperationTimeUnitName(Me.OperationTimeUnit)
    End If
    dblDemandCF = TimeConversionFactor(lngOldDemandPeriodUnitSize, strOldDemandUnit,
Me.DemandPeriodUnitSize, strNewDemandUnit)

    strOldOpUnit = OperationTimeUnitName(intOldOperationTimeUnit)
    strNewOpUnit = OperationTimeUnitName(Me.OperationTimeUnit)
    dblOpCF = TimeConversionFactor(1, strOldOpUnit, 1, strNewOpUnit)

    MsgBox "Conversão do Período : " & lngOldDemandPeriodUnitSize & " " & strOldDemandUnit & " --> " &
Me.DemandPeriodUnitSize & " " & strNewDemandUnit & " = " & dblDemandCF & vbCrLf & vbCrLf _
        & "Conversão da Operação: " & "1" & " " & strOldOpUnit & " --> " & "1" & " " & strNewOpUnit
& " = " & dblOpCF

    Call UpdateRelatedTables(dblDemandCF, dblOpCF)

End Sub

Sub UpdateRelatedTables(dblDemandCF As Double, dblOpCF As Double)

    Dim dbsCurrent As DAO.Database
    Set dbsCurrent = CurrentDb

    dbsCurrent.Execute "UPDATE Products " _
                     & " SET Products.LeadTime = nz([LeadTime]*" & dblDemandCF & ",0);"

    dbsCurrent.Execute "UPDATE Operations" & " SET Operations.EquipmentSetupTime =
nz([EquipmentSetupTime]*" & dblOpCF & ",0), " & "Operations.EquipmentRunTime = nz([EquipmentRunTime]*"
& dblOpCF & ",0), " & "Operations.LabourSetupTime = nz([LabourSetupTime]*" & dblOpCF & ",0), " &
"Operations.LabourRunTime = nz([LabourRunTime]*" & dblOpCF & ",0), " & "Operations.LeadTimeOffset =
nz([LeadTimeOffset]*" & dblDemandCF & ",0)" & " WHERE (((Operations.Permanent)=False));"

    dbsCurrent.Execute "UPDATE Equipments"
                     & " SET Equipments.MTTF = nz([MTTF]*" & dblOpCF & ",0), " _
                       & "Equipments.MTTR = nz([MTTR]*" & dblOpCF & ",0);"

    Set dbsCurrent = Nothing
End Sub

Private Sub Form_Close()
    DoCmd.OpenForm "SFCConfig"
End Sub
```

ORDEM DE PRODUÇÃO

```
Option Compare Database
Option Explicit
Public strTmp As String
Public strdelete As String
Public itmp As Integer

Dim WOObject As New EBexplodeWOClass
Dim XObject As New EB.EBexplodeClass
Dim DBObject As New EB.EBdefinitionsClass
Dim IObject As New EBupdatedynamiccountClass


Private Sub Command518_Click()
Call CopyWOItems_Click

End Sub

Private Sub CopyWOItems_Click()
Dim iLevel As Integer
Dim kLevel As Integer
Dim mLevel As Integer
Dim jLevel As Integer
Dim iNumber As Integer
Dim iCustomerID As Integer
Dim strSQL As String
Dim strCtl As String
Dim strValue As String
Dim strProductID As String
Dim dbsCurrent As DAO.Database
Dim rstDMR As DAO.Recordset
Dim rstDMR_sort As DAO.Recordset
Dim wrkJet As DAO.Workspace
Dim qdfExplode As DAO.QueryDef

On Error GoTo Err_CopyWOItems_Click

DoCmd.SetWarnings False
Set dbsCurrent = CurrentDB()

strCtl = "ZW"
strValue = Forms!WorkOrders!SalesOrderID

mLevel = 4
iLevel = 0
iNumber = 0
strSQL = "SELECT * FROM [Inventory Transactions] WHERE (([Inventory Transactions]![SalesOrderID] = """
& strValue & """) "
&       "AND       (IIF([Inventory       Transactions].[UnitsOrdered]       is       Null,0,[Inventory
Transactions].[UnitsOrdered])=0 ) "
&   "AND   (([Inventory   Transactions].[Control]   Like   """   &   "*0*"   &   """)   OR   ([Inventory
Transactions].[Control] Like """ & "*2*" & """)))"
Set rstDMR = dbsCurrent.OpenRecordset(strSQL)
If rstDMR.RecordCount > 0 Then
    MsgBox ("Não existem pais para copiar ou as quantidades não foram registradas.  ")
    rstDMR.Close
    DoCmd.Hourglass False
    Exit Sub
End If
rstDMR.Close
Me!StatusID = 2

CustomerID = iif(IsNull(Forms![WorkOrders]![CustomerID]), 0, Forms![WorkOrders]![CustomerID])
If iCustomerID <= 0 Then
    MsgBox ("Não é possível copiar porque o campo 'Transferir Para' está em branco.  "
    & "Selecione um local através da seta ao lado do campo e clique em 'Copiar' novamente!")
    DoCmd.Hourglass False
    Exit Sub
End If

dbsCurrent.Execute "DELETE ShipChk.* FROM ShipChk;"
dbsCurrent.Execute "DELETE ShipChk_Sorted.* FROM ShipChk_Sorted;"

dbsCurrent.Execute "INSERT INTO ShipChk (ProductID, CustomerID, [UnitsSold], [level], "
 "Control, [SalesOrderID], PP)"
 "SELECT ProductID, CustomerID, UnitsShippedFG as [UnitsSold],""" & iLevel & """ as [level], "
 "Control,""" & strValue & """ as [SalesOrderID], PP "
 "FROM [Inventory Transactions] "
 "WHERE (([Inventory Transactions].[SalesOrderID] = """ & strValue & """) "
 "AND (([Inventory Transactions].[Control] Like """ & "*0*" & """) "
```

```
& "OR ([Inventory Transactions].[Control] Like """ & "*Z*" & """)};"

DoCmd.OpenQuery "ShipChk_Append_vb1_WO_copy"

DoCmd.SetWarnings True
DoCmd.Hourglass False
Me![WorkOrdersSubform].Requery

Exit_CopyWOItems_Click:
    Exit Sub

Err_CopyWOItems_Click:
    MsgBox Err.Description
    Resume Exit_CopyWOItems_Click
End Sub
Private Sub Report_Close()
Set DBObject = Nothing
Set IObject = Nothing
Set WOObject = Nothing
End Sub

Private Sub AdjustForATP_Click()
If Me!StatusID > 1 Then
    MsgBox "O status da Ordem de Produção deve ser 'Planejada' para permitir o uso da função DPI."
    Exit Sub
End If
DoCmd.Hourglass True
Call AllocatedCounts
DoCmd.SetWarnings False
DoCmd.OpenQuery "ATPupdateDynamicCount", acNormal, acEdit
DoCmd.SetWarnings True
itmp = 0
Call Adjust_Click
DoCmd.Hourglass False
Me!StatusID = 2
If itmp = 0 Then
 MsgBox "Os ajustes baseados na contagem de inventário DPP estão completos."
End If
End Sub
Private Sub AdjustPorOnHand_Click()
Dim dbsCurrent As DAO.Database
Dim iLevel As Integer
Dim kLevel As Integer
Dim mLevel As Integer
Dim iNumber As Integer
Dim rstDMR As DAO.Recordset
Dim rstDMR_sort As DAO.Recordset
Dim wrkJet As DAO.Workspace
Dim strValue As String
Dim strProductID As String
Dim qdfExplode As DAO.QueryDef
Dim jLevel As Integer
Dim dUnitsSold As Double
Dim lCustomerID As Integer
Dim strControl As String
Dim strSQL As String
Dim lNeeded As Long
Dim lATP As Long
Dim ratio As Double
Dim strManufacturer As String
Dim varBookmark As Variant


Set dbsCurrent = CurrentDB()
DoCmd.Hourglass True
strValue = DLookup("[Userdefinable 3]", "My Company Information")
strValue = strValue & "DRM_fe.mdb"
DBObject.DatabaseCurrent = dbsCurrent
DBObject.DatabaseName = strValue
    DBObject.SetDatabase
Dim intLocation As Integer

intLocation = 1
strValue = DLookup("[InventoryType]", "[Customers]", "[Customers]![CustomerID] = " & intLocation & "")

DBObject.InventoryType = strValue
DBObject.InventoryLocation = intLocation
DBObject.InventoryEndDate = Now()
DBObject.DatabaseCurrent = dbsCurrent
DBObject.Winxx = DLookup("[Windows]", "[VersionControl]")
DBObject.UpdateDynamicCount
```

```
itmp = 0
Call Adjust_Click
DoCmd.Hourglass False
Me!StatusID = 2
If itmp = 0 Then
  MsgBox "Os ajustes baseados na contagem de inventário 'Em Mãos' estão completos."
End If
Set DBObject = Nothing
Set IOObject = Nothing

End Sub

Private Sub Adjust_Click()
Dim dbsCurrent As DAO.Database
Dim iLevel As Integer
Dim kLevel As Integer
Dim mLevel As Integer
Dim iNumber As Integer
Dim rstDMR As DAO.Recordset
Dim rstDMR_sort As DAO.Recordset
Dim wrkJet As DAO.Workspace
Dim strValue As String
Dim strProductID As String
Dim qdfExplode As DAO.QueryDef
Dim jLevel As Integer
Dim dUnitsSold As Double
Dim iCustomerID As Integer
Dim strControl As String
Dim strSQL As String
Dim iNeeded As Long
Dim iATP As Long
Dim ratio As Double
Dim strManufacturer As String
Dim varBookmark As Variant
Dim iTemp As Long

Set dbsCurrent = CurrentDB()

strValue = DLookup("[Userdefinable 3]", "My Company Information")
strValue = strValue & "DkM_fe.mdb"
DBObject.DatabaseCurrent = dbsCurrent
DBObject.DatabaseName = strValue
DBObject.SetDatabase

strValue = Forms!WorkOrders!SalesOrderID

    strSQL = "SELECT * FROM [Inventory Transactions] WHERE (([Inventory Transactions]![SalesOrderID] =
"" & strValue & """) AND ([Inventory Transactions]![Control] Like """ & "*JJ*" & """ ))"

    Set rstDMR = dbsCurrent.OpenRecordset(strSQL)
    If rstDMR.RecordCount > 0 Then
     MsgBox ("Você não pode fazer ajuste na OP neste momento!")
     itmp = 1
    rstDMR.Close
    Exit Sub
    End If

    rstDMR.Close

trSQL = "SELECT * FROM [Inventory Transactions] WHERE (([Inventory Transactions]![SalesOrderID] = """
   strValue & """) AND ([Inventory Transactions]![TransferFromID] >= 0) AND ([Inventory
ransactions]![PPnumber] >= 0 ))"

et rstDMR = dbsCurrent.OpenRecordset(strSQL)

stDMR.MoveFirst
o While Not rstDMR.EOF
arBookmark = rstDMR.Bookmark
trProductID = rstDMR!ProductID
trManufacturer = DLookup("[Manufacturer]", "Products", "[ProductID] = """ &
eplaceString(strProductID, Chr(34), Chr(34) & Chr(34)) & """")

f strManufacturer = "Manufactured" Then

   iNeeded = rstDMR!UnitsShippedFG

   iATP = DLookup("nz([DynamicCount],0)", "Products", "[ProductID] = """ &
eplaceString(strProductID, Chr(34), Chr(34) & Chr(34)) & """")
   ratio = 0
   If iATP < 0 Then
   iATP = 0
```

```
    End If
    If iNeeded > 0 Then
        ratio = CDbl(iNeeded - iATP) / CDbl(iNeeded)
    End If
    If ratio < 0 Then
        ratio = 0
    End If
    iLevel = rstDMR!PPnumber

    rstDMR.Edit
        rstDMR!Control = rstDMR!Control & "J"
        If (iNeeded > iATP) Then
            rstDMR!UnitsShippedFG = iATP
            dbsCurrent.Execute "UPDATE Products SET DynamicCount = 0 WHERE ProductID = """ &
ReplaceString(strProductID, Chr(34), Chr(34) & Chr(34)) & """;"
            If iLevel > 0 And strManufacturer = "Manufactured" Then
                rstDMR!UnitsOrdered = iATP
            End If
        Else
            rstDMR!UnitsShippedFG = iNeeded
            iTemp = iATP - iNeeded
            dbsCurrent.Execute "UPDATE Products SET DynamicCount = """ & iTemp & """ WHERE ProductID =
""" & ReplaceString(strProductID, Chr(34), Chr(34) & Chr(34)) & """;"
            If iLevel > 0 And strManufacturer = "Manufactured" Then
                rstDMR!UnitsOrdered = iNeeded
            End If
        End If
    rstDMR.update

    varBookmark = rstDMR.Bookmark
    rstDMR.MoveNext

    Do While Not rstDMR.EOF

        kLevel = rstDMR!PPnumber

        If kLevel <= iLevel Then
            Exit Do
        Else
            iNeeded = rstDMR!UnitsShippedFG
            iNeeded = CDbl(iNeeded) * ratio
            strProductID = rstDMR!ProductID
            rstDMR.Edit
            rstDMR!UnitsShippedFG = iNeeded
            rstDMR!UnitsOrdered = iNeeded
            rstDMR.update
        End If
        rstDMR.MoveNext

    Loop

End If

rstDMR.Bookmark = varBookmark
rstDMR.MoveNext

Loop
Set rstDMR = Nothing
Set DBObject = Nothing


DoCmd.Requery
End Sub

Private Sub CmdExplodeVGs_Click()

Dim iLevel As Integer
Dim kLevel As Integer
Dim mLevel As Integer
Dim jLevel As Integer
Dim iNumber As Integer
Dim iCustomerID As Integer
Dim ExplodeLevel As Integer
Dim levelLoop As Double
Dim dUnitsSold As Double
Dim levelCurrent As Double
Dim levelPrevious As Double
Dim parentQuantity As Double
Dim strSQL As String
Dim strCtl As String
Dim strValue As String
```

```
Dim curZW As Long
Dim currentWO As String
Dim strControl As String
Dim strProductID As String
Dim explodeChildren As String
Dim wrkJet As DAO.Workspace
Dim rstDMR As DAO.Recordset
Dim rstDMR_sort As DAO.Recordset
Dim rstDMR_tmp As DAO.Recordset
Dim qdfExplode As DAO.QueryDef
Dim dbsCurrent As DAO.Database

On Error GoTo Err_Explode_Click
If IsNull(Me!EmployeeID) Then
    MsgBox " Preencha o Código de Funcionário na Guia 'Dados Básicos' antes de Explodir."
    Exit Sub
End If

Set dbsCurrent = CurrentDB()

strCtl = "ZW"
strValue = Me!SalesOrderID
mLevel = 7
iLevel = 0
iNumber = 0

strSQL = "SELECT * FROM [Inventory Transactions] WHERE (([Inventory Transactions].[SalesOrderID] = """
&  strValue  &  """)  AND  ([Inventory  Transactions].[TransferFromID]  =  0)  AND  ([Inventory
Transactions].[PPnumber] > 1 ))"
Set rstDMR = dbsCurrent.OpenRecordset(strSQL)
If rstDMR.RecordCount > 0 Then
    MsgBox ("A explosão da OP já foi gerada!")
    rstDMR.Close
    DoCmd.Hourglass False
    Exit Sub
End If
rstDMR.Close

strSQL = "SELECT * FROM [Inventory Transactions] WHERE (([Inventory Transactions].[SalesOrderID] = """
& strValue & """) " _
&       "AND       (IIF([Inventory       Transactions].[UnitsOrdered]       is       Null,0,[Inventory
Transactions].[UnitsOrdered])=0 ) " _
& "AND ([Inventory Transactions].[Control] Like """ & "*J*" & """))"
Set rstDMR = dbsCurrent.OpenRecordset(strSQL)
If rstDMR.RecordCount > 0 Then
    MsgBox ("Não existem itens Copiados para explodir ou então as quantidades não foram registradas.")
    rstDMR.Close
    DoCmd.Hourglass False
    Exit Sub
End If
rstDMR.Close

DoCmd.Hourglass True
DoCmd.SetWarnings False

    dbsCurrent.Execute "DELETE * FROM ShipChk_tmp;"
    dbsCurrent.Execute "INSERT INTO ShipChk_tmp (SalesOrderID, UnitsSold, " _
      & "[Products_ProductID], PP, " _
      & "CustomerID, [Level], Units_tmp) " _
      & "SELECT SalesOrderID, UnitsOrdered AS UnitsSold, " _
      & "[Inventory Transactions].ProductID AS Products_ProductID, PP, " _
      & "CustomerID, 1 AS [Level], TransactionID AS Units_tmp " _
      & "FROM [Inventory Transactions] " _
      & "WHERE [Inventory Transactions].SalesOrderID = """ & strValue & """ " _
      & "AND [Inventory Transactions].[Control] LIKE " & """*WJ*""" & " " _
      & "ORDER BY TransactionID;"
    dbsCurrent.Execute "DELETE * FROM [Inventory Transactions] WHERE ((SalesOrderID = """ & strValue &
""") AND ([Control] Like """ & "*J*" & """));"

    Set rstDMR_tmp = dbsCurrent.OpenRecordset("SELECT Units_tmp AS AutoID FROM ShipChk_tmp ORDER BY
nits_tmp;")
    If (rstDMR_tmp.RecordCount > 0) Then rstDMR_tmp.MoveFirst

    Do While Not (rstDMR_tmp.EOF)
        dbsCurrent.Execute "DELETE * FROM ShipChk;"
        dbsCurrent.Execute "INSERT INTO ShipChk (SalesOrderID, UnitsSold, ProductID, " _
                           & "CustomerID, [Level], PP) " _
                           &   "SELECT   SalesOrderID,   UnitsSold,   ShipChk_tmp.Products_ProductID   AS
roductID, " _
                           & "CustomerID, [Level], PP " _
                           & "FROM ShipChk_tmp " _
```

```
                & "WHERE Units_tmp = " & rstDMR_tmp!autoid & ";"

Set rstDMR = dbsCurrent.OpenRecordset("ShipChk", dbOpenTable)
If Not (rstDMR.EOF) Then rstDMR.MoveNext
Do While Not (rstDMR.EOF)
    rstDMR.Delete
    rstDMR.MoveNext
Loop
rstDMR.MoveFirst

If (NoChildren(rstDMR!ProductID)) Then
    dbsCurrent.Execute "UPDATE ShipChk INNER JOIN Products " _
        & "ON ShipChk.ProductID = Products.ProductID " _
        & "SET ShipChk.Explode = Products.Manufacturer;"
Else
    If (Right(strValue, 1) = "P") Then
        ExplodeLevel = 1
    Else
        ExplodeLevel = 30
    End If

    XObject.mlevels = ExplodeLevel
    XObject.ParentID = rstDMR!ProductID
    XObject.DatabaseCurrent = dbsCurrent
    XObject.Explode

    rstDMR.Close

    DoCmd.OpenQuery "ShipChk_BOMSorted", acNormal, acEdit

    Set rstDMR = dbsCurrent.OpenRecordset("ShipChk", dbOpenTable)

    rstDMR.MoveFirst
    Do While Not (rstDMR.EOF)
        levelCurrent = rstDMR!Level
        parentQuantity = rstDMR!UnitsSold
        rstDMR.MoveNext
        If Not (rstDMR.EOF) Then
            levelPrevious = levelCurrent
            levelCurrent = rstDMR!Level
            If (levelCurrent > levelPrevious) Then
                levelLoop = levelCurrent
                Set rstDMR = Filter_Recordset_MultiLevelBOM(rstDMR, parentQuantity, levelLoop)
                If (Not (rstDMR.EOF)) Then
                    levelCurrent = rstDMR!Level
                Else
                    Exit Do
                End If
            End If
        Else
            Exit Do
        End If
    Loop

  currentWO = strValue

    dbsCurrent.Execute "UPDATE ShipChk INNER JOIN Products " _
        & "ON ShipChk.ProductID = Products.ProductID " _
        & "SET ShipChk.Explode = Products.Manufacturer;"
    strSQL = "SELECT ID, Explode, [level] AS delevel FROM [ShipChk] " _
        & "WHERE [ShipChk].[SalesOrderID] = """ & currentWO & """ " _
        & "ORDER BY ID;"

    Set rstDMR = dbsCurrent.OpenRecordset(strSQL)
    If (rstDMR.RecordCount > 0) Then
        rstDMR.MoveFirst
        rstDMR.MoveNext
    End If

    Do While Not (rstDMR.EOF)
        levelCurrent = rstDMR!delevel
        explodeChildren = Nz(rstDMR!Explode, "Purchased")
        rstDMR.MoveNext
        If Not (rstDMR.EOF) Then
            levelPrevious = levelCurrent
            levelCurrent = rstDMR!delevel
            If (levelCurrent > levelPrevious) And (explodeChildren = "Mftd (Not Exploded)")
hen
                Do While (levelCurrent > levelPrevious)
                    rstDMR.Delete
```

```
                        rstDMR.MoveNext
                        If Not (rstDMR.EOF) Then
                            levelCurrent = rstDMR!delevel
                        Else
                            Exit Do
                        End If
                    Loop
                End If
            Else
                Exit Do
            End If
        Loop
    End If '(NoChildren(rstDMR!ProductID)) Then

    DoCmd.OpenQuery "ShipChk_Append_vbl_WO"
    rstDMR_tmp.MoveNext
Loop

If ((rstDMR Is Nothing)) Then rstDMR.Close
DoCmd.SetWarnings True
DoCmd.Hourglass False
Me![WorkOrder.Subform].Requery
Me![Child247].Requery
Set WOObject = Nothing

Exit_Explode_Click:
    DoCmd.SetWarnings True
    DoCmd.Hourglass False
    Exit Sub

Err_Explode_Click:
    MsgBox "Ocorreu um erro ao tentar processar este comando. Por favor use o tópico 'Ajuda' no Manual
para a Operação deste botão."
    Resume Exit_Explode_Click

End Sub

Private Sub CmdPreviewForPrinting_Click()
On Error Resume Next

    Dim stDocName As String
    Dim strValue As String

    stDocName = "Work Order"
    strValue = Me!SalesOrderID
    DoCmd.Minimize
    DoCmd.OpenReport stDocName, acPreview, , "[Sales Orders]![SalesOrderID]= """ & strValue & """"

Exit_CmdPreviewForPrinting_Click:
    Exit Sub

Err_CmdPreviewForPrinting_Click:
    MsgBox Err.Description
    Resume Exit_CmdPreviewForPrinting_Click

End Sub

Private Sub Command248_Click()
Call CmdExplodeVGs_Click

End Sub

Private Sub Command249_Click()
Call CmdPreviewForPrinting_Click
End Sub

Private Sub Command250_Click()
Call Picklist_Click
End Sub

Private Sub Command252_Click()
Call PrintPicklist_Click
End Sub

Private Sub Command254_Click()
Call AdjustForATP_Click
End Sub

Private Sub Command285_Click()
Call AdjustForOnHand_Click
End Sub
```

```
Private Sub Command291_Click()
DoCmd.Minimize
DoCmd.OpenReport "RoutingReport_WO", acViewPreview
End Sub

Private Sub Command292_Click()
DoCmd.OpenReport "RoutingReport_WO", acViewPreview
End Sub

Private Sub Command305_Click()
DoCmd.OpenReport "Picklist Shortages", acViewPreview
End Sub

Private Sub Command306_Click()
DoCmd.Minimize
DoCmd.OpenReport "Picklist Shortages", acViewPreview
End Sub

Private Sub Command311_Click()
DoCmd.Minimize
DoCmd.OpenReport "RoutingReport_AllWO's", acViewPreview
End Sub

Private Sub Command315_Click()
Call DELETEPicklist_Click
End Sub

Private Sub Form_Close()
DoCmd.OpenForm "Menu Inicial"
End Sub

Private Sub Form_Load()
Dim dbsCurrent As DAO.Database
Dim strValue As String
Set dbsCurrent = CurrentDB()
strValue = DLookup("[Userdefinable 3]", "My Company Information")
strValue = strValue & "DbM_fe.mdb"
DBObject.DatabaseCurrent = dbsCurrent
DBObject.DatabaseName = strValue
DBObject.SetDatabase
End Sub

Private Sub Filter_AfterUpdate()
strtmp = Forms![WorkOrders]!Filter
End Sub

rivate Sub Form_AfterDelConfirm(Status As Integer)
    Select Case Status
        Case acDeleteOK
            MsgBox "O processo ocorreu normalmente."
        Case acDeleteCancel

sgBox "Usuário cancelou o processo."
        Case acDeleteUserCancel
            MsgBox "Usuário cancelou o processo."
    End Select
nd Sub

rivate Sub Form_BeforeDelConfirm(Cancel As Integer, Response As Integer)
    Response = acDataErrContinue
    Dim strValue As String
    strValue = Left(strdelete, 2)
    If strValue <> "WO" Then
        MsgBox ("Apenas as Ordens de Produção do tipo WOxxxx podem ser apagadas a partir do Formulário
rdens de Produção'.")
        DoCmd.CancelEvent
        Exit Sub
    End If

    If MsgBox("Apagar este Registro?", vbOKCancel) = vbCancel Then
        Cancel = True
        Exit Sub
    End If
    Dim dbsCurrent As Database
    Dim strSalesOrderID As String
    Dim wrkJet As Workspace
     Set dbsCurrent = CurrentDB()

    On Error Resume Next
    Dim strTmp As String
```

```
    strSalesOrderID = strdelete
    DoCmd.Hourglass True
    dbsCurrent.Execute "DELETE * FROM [Sales Orders] WHERE ([Sales Orders].[SalesOrderID]=""" &
strSalesOrderID & """);"
    dbsCurrent.Execute "DELETE * FROM [Inventory Transactions] WHERE ([Inventory
Transactions].[SalesOrderID] = """ & strSalesOrderID & """);"
    dbsCurrent.Execute "DELETE * FROM [WO_SFC_Routings] WHERE ([WO_SFC_Routings].[SalesOrderID] = """
& strSalesOrderID & """);"
    DoCmd.Hourglass False

End Sub

Private Sub Form_Delete(Cancel As Integer)
strTmp = Me!SalesOrderID
strdelete = strTmp
End Sub

Private Sub ItemFind_Click()
If Not IsNull([ItemLookUP]) Then
    SalesOrderID.SetFocus
    DoCmd.FindRecord ([ItemLookUP])
    ItemLookUP.SetFocus
Else
    MsgBox "Por favor selecione uma Ordem de Produção pra buscar."
End If
End Sub

Private Sub Picklist_Click()
Dim dbsCurrent As DAO.Database
Dim iLevel As Integer
Dim kLevel As Integer
Dim mLevel As Integer
Dim iNumber As Integer
Dim rstDMR As DAO.Recordset
Dim rstDMR_sort As DAO.Recordset
Dim wrkJet As DAO.Workspace
Dim strValue As String
Dim strProductID As String
Dim qdfExplode As QueryDef
Dim jLevel As Integer
Dim dUnitsSold As Double
Dim strSQL As String
Dim sfcSubform As SubForm
Dim advWOSF As SubForm

   Dim checkWOID As Long
   checkWOID = InStr(Me.SalesOrderID, "-WO")
   If (checkWOID <> 0) Then
      MsgBox "Você não pode selecionar uma OP 'clone' uma vez que os produtos já foram selecionados
a OP original. "
          & "Se você já explodiu esta OP 'clone', prossiga para o Recebimento quando estiver
ronto.", vbExclamation, "Operação Inválida"
      Exit Sub
   End If

   Set dbsCurrent = CurrentDB()
   Me!StatusID = 3

   strValue = Forms!WorkOrders!SalesOrderID
   strSQL = "SELECT * FROM [Inventory Transactions] WHERE (([Inventory Transactions]![SalesOrderID] =
" & strValue & """) AND ([Inventory Transactions]![UnitsOrdered] Is Null))"

   Set rstDMR = dbsCurrent.OpenRecordset(strSQL)
   If rstDMR.RecordCount > 0 Then
   MsgBox ("Você precisa registrar a Qtde para todos os itens, ou você já gerou uma lista de
paração.")
   rstDMR.Close
   Exit Sub
   End If
   rstDMR.Close

   strSQL = "SELECT * FROM [Inventory Transactions] WHERE (([Inventory Transactions]![SalesOrderID] =
" & strValue & """) AND ([Inventory Transactions]![Control] Like """ & "*WO*" & """))"

   Set rstDMR = dbsCurrent.OpenRecordset(strSQL)
   If rstDMR.RecordCount = 0 Then
   MsgBox ("Você precisa explodir a OP antes de gerar uma Lista de Separação!")
   rstDMR.Close
   Exit Sub
   End If
   rstDMR.Close
```

```
    strSQL = "SELECT * FROM [Inventory Transactions] WHERE (([Inventory Transactions]![SalesOrderID] =
""" & strValue & """) AND ([Inventory Transactions]![CustomerID] = 2))"

    Set rstDMR = dbsCurrent.OpenRecordset(strSQL)
    If rstDMR.RecordCount > 0 Then
    MsgBox ("A Lista de Separação já foi gerada!")
    rstDMR.Close
    Exit Sub
    End If
    rstDMR.Close

DoCmd.Hourglass True
Call AllocatedCounts
DoCmd.SetWarnings False
DoCmd.OpenQuery "ATUpdateDynamicCount_neqs", acNormal, acEdit
DoCmd.SetWarnings True

dbsCurrent.Execute "INSERT INTO [Inventory Transactions] ( TransactionDate,ProductID, " _
& "SalesOrderID, UnitsShipped,UnitsShippedFG,TransferFromID,CustomerID,Control,PPnumber,PP ) " _
& "SELECT Now() AS [TDate],[Inventory Transactions].[ProductID], " _
& "[Inventory Transactions].[SalesOrderID], " _
& "IIf([Products].[Manufacturer] Like """ & "*Manu*" & """ and [Inventory Transactions].[Control] Not
Like """ & "*J*" & """,0,[UnitsShippedFG]) AS [Shipd], " _
& "UnitsShippedFG,'1' AS [FromID],'2' AS [ToID],IIF([Inventory Transactions].[Control] Like """ &
"*W*" & """, " _
& "[Inventory Transactions].[Control],[Inventory Transactions].[Control] & """ & "W" & """) as
[CTRL],[Inventory Transactions].[PPnumber], [Inventory Transactions].[PP] " _
& "FROM [Products] INNER JOIN [Inventory Transactions] ON [Products].[ProductID] = [Inventory
Transactions].[ProductID] " _
& "WHERE ([Inventory Transactions].[SalesOrderID] = """ & strValue & """) " _
& "AND ((([Inventory Transactions].[Control] Not Like """ & "*0*" & """) AND ([Inventory
Transactions].[Control] Not Like """ & "*Z*" & """)) AND ([Inventory Transactions].[Control] Like """
& "*W*" & """)) " _
& "ORDER BY ([Inventory Transactions].[TransactionID]);"
DoCmd.OpenReport "WO Picklist", acViewPreview, , "([SalesOrderID] =
Forms![WorkOrders]![SalesOrderID])"
DoCmd.Hourglass False

DoCmd.SetWarnings False
Set sfcSubform = [Forms]![WorkOrders]![WO_SFC_Routings subform]
Set advWOSF = [Forms]![WorkOrders]![Child247]
DoCmd.OpenQuery "WO_SFC subform query", acNormal, acEdit
sfcSubform.Requery
advWOSF.Requery
DoCmd.SetWarnings True

End Sub

Private Sub PrintPicklist_Click()

Dim dbsCurrent As DAO.Database
Dim iLevel As Integer
Dim kLevel As Integer
Dim mLevel As Integer
Dim iNumber As Integer
Dim rstDMR As DAO.Recordset
Dim rstDMR_sort As DAO.Recordset
Dim wrkJet As DAO.Workspace
Dim strValue As String
Dim strProductID As String
Dim qdfExplode As DAO.QueryDef
Dim jLevel As Integer
Dim dUnitsSold As Double
Dim strSQL As String

    Set dbsCurrent = CurrentDB()
    Me!StatusID = 3

    strValue = Me!SalesOrderID
    strSQL = "SELECT * FROM [Inventory Transactions] WHERE (([Inventory Transactions]![SalesOrderID] =
" & strValue & """) AND ([Inventory Transactions]![CustomerID] = 2))"

    Set rstDMR = dbsCurrent.OpenRecordset(strSQL)
    If rstDMR.RecordCount = 0 Then
    MsgBox ("A Lista de Separação ainda não foi gerada! Clique no botão Lista de Separação.")
    rstDMR.Close
    Exit Sub
    End If
    rstDMR.Close
```

```
On Error Resume Next
DoCmd.Minimize
DoCmd.OpenReport "WO_Picklist", acViewPreview, , "([SalesOrderID] = """ & strValue & """)"

End Sub

Private Sub SalesOrderID_BeforeUpdate(Cancel As Integer)
    Dim decision As Integer
    Dim prefix As String
    Dim rstID As DAO.Recordset
    Dim dbsCurrent As Database
    Dim strSQL As String
    Dim strValue As String
    Dim result As Long

    Set dbsCurrent = CurrentDB()
    result = InStr(Me.SalesOrderID, "-WO")
    If result <> 0 Then
        MsgBox "Não foi possível alterar o código da OP 'clone' por erros na qualidade dos dados.",
vbCritical
        Me.Undo
        Exit Sub
    End If
    decision = MsgBox("Você quer realmente alterar o código desta OP?", vbYesNo)
    If decision = 7 Then
        Me.Undo
    Else
        If IsNull(Me!SalesOrderID) Then
            MsgBox "Não é possível substituir o código da OP por um valor nulo", vbOKOnly
            Me.Undo
            Exit Sub
        End If
        prefix = Left(Forms!WorkOrders!SalesOrderID.Value, 2)
        If prefix <> "WO" Then
            MsgBox "Este código é inválido. Ele deve começar com 'WO'.", vbOKOnly
            Me.Undo
        Else
            strValue = Me!SalesOrderID
            strSQL = "SELECT * FROM [Sales Orders] WHERE ([Sales Orders]![SalesOrderID] = """ &
strValue & """)"
            Set rstID = dbsCurrent.OpenRecordset(strSQL)
            If rstID.RecordCount <> 0 Then
                MsgBox "Este código já existe. Por favor escolha outro.", vbOKOnly
                Me.Undo
            End If
            rstID.Close
        End If
    End If
End Sub

Private Sub SalesOrderID_LostFocus()
Dim strTmp As String
On Error Resume Next
strTmp = Me!SalesOrderID
End Sub

Private Sub StatusID_Change()
StatusID.Requery
End Sub

Private Sub Shortages__Click()
On Error GoTo Err_Shortages__Click
    Dim stDocName As String
    Dim stLinkCriteria As String

    stDocName = "MRP_lite"
    DoCmd.OpenForm stDocName, , , stLinkCriteria
Exit_Shortages__Click:
    Exit Sub
Err_Shortages__Click:
    MsgBox Err.Description
    Resume Exit_Shortages__Click

End Sub
Private Sub EditFlag_Click()
Dim dbsCurrent As DAO.Database
Dim iLevel As Integer
Dim kLevel As Integer
Dim mLevel As Integer
Dim iNumber As Integer
Dim rstDMR As DAO.Recordset
```

```
Dim rstDMR_sort As DAO.Recordset
Dim wrkJet As DAO.Workspace
Dim strValue As String
Dim strProductID As String
Dim qdfExplode As DAO.QueryDef
Dim jLevel As Integer
Dim dUnitsSold As Double
Dim strSQL As String

    Set dbsCurrent = CurrentDB()
On Error GoTo Err_EditFlag_Click
strValue = Me!SalesOrderID
dbsCurrent.Execute "UPDATE Products " _
& "INNER JOIN [Inventory Transactions] " _
& "ON Products.ProductID = [Inventory Transactions].ProductID " _
& "SET [Inventory Transactions].Control = [Control] & """ & "J" & """ " _
& "WHERE (([Inventory Transactions].SalesOrderID) = """ & strValue & """) " _
& "AND (([Products.Manufacturer)= 'manufactured') " _
& "AND (([Inventory Transactions].TransferFromID)=0 Or ([Inventory Transactions].TransferFromID)=1) " _
& "AND (([Inventory Transactions].CustomerID)<>2)) OR (([Products.Manufacturer)='purchased') " _
& "AND (([Inventory Transactions].TransferFromID)=0 Or ([Inventory Transactions].TransferFromID)=1) " _
& "AND (([Inventory Transactions].CustomerID)<>2) " _
& "AND (([Inventory Transactions].UnitsOrdered)>0)));"

Exit_EditFlag_Click:
    Exit Sub
Err_EditFlag_Click:
    MsgBox Err.Description
    Resume Exit_EditFlag_Click
End Sub
Private Sub DELETEPicklist_Click()
On Error GoTo Err_Command313_Click
Dim dbsCurrent As DAO.Database
Dim iLevel As Integer
Dim kLevel As Integer
Dim mLevel As Integer
Dim lNumber As Integer
Dim rstDMR As DAO.Recordset
Dim rstDMR_sort As DAO.Recordset
Dim wrkJet As DAO.Workspace
Dim strValue As String
Dim strProductID As String
Dim qdfExplode As DAO.QueryDef
Dim jLevel As Integer
Dim dUnitsSold As Double
Dim iCustomerID As Integer
Dim strControl As String
Dim strSQL As String
Set dbsCurrent = CurrentDB()
strValue = Forms!WorkOrders!SalesOrderID

Dim checkWOID As Long
heckWOID = InStr(Me.SalesOrderID, "-WO")
f (checkWOID <> 0) Then
    MsgBox "Listas de Separação não podem ser aplicadas para OP 'clones'.", vbExclamation, "Operação
nválida"
    Exit Sub
nd If
trSQL = "SELECT * FROM [Inventory Transactions] " _
    & "WHERE [Inventory Transactions]![SalesOrderID Like """ & strValue & """ " _
    & "AND [Inventory Transactions]![TransferFromID = 1" _
    & "AND [Inventory Transactions]![CustomerID = 2;"
et rstDMR = dbsCurrent.OpenRecordset(strSQL)
f Not (rstDMR.RecordCount > 0) Then
    MsgBox "Não existe Lista de Separação para esta OP."
    Exit Sub
nd If
stDMR.Close

trSQL = "SELECT * FROM [Inventory Transactions] WHERE SalesOrderID Like ""*" & "-" & strValue &
e"";"
et rstDMR = dbsCurrent.OpenRecordset(strSQL)
f rstDMR.RecordCount > 0 Then
    Dim answer As Long
    answer = MsgBox("O Programa detectou que iniciou recebimento parcial para esta OP. Ao apagar a
ista de Separação " & "você retornará todos os produtos 'em processamento' desta OP para os
epósitos, e a contagme de inventário " & "pode ser afetada se OP 'clones' forem recebidas sem que se
ie uma nova Lista de Separação para esta OP. " & "Você quer continuar?", vbOKCancel, "AVISO!")
    If answer = 2 Then
        MsgBox "Processo abortado. A lista NÃO foi apagada."
        Exit Sub
```

```
    End If
End If
rstIMk.Close
dbsCurrent.Execute "DELETE * FROM [Inventory Transactions] " _
& "WHERE (SalesOrderID = """ & strValue & """ AND CustomerID = 2);"

dbsCurrent.Execute "DELETE * FROM [WO_SFC_ROutings] WHERE (SalesOrderID = """ & strValue & """);"
Me.WO_SFC_Routings_subform.Requery

Me!StatusID = 2
MsgBox "A Lista de Separação foi apagada com sucesso."
Exit_Command313_Click:
    Exit Sub
Err_Command313_Click:
    MsgBox Err.Description
    Resume Exit_Command313_Click

End Sub

Private Sub WorkOrdersSubform_Enter()
If IsNull(Me!EmployeeID) Then
    MsgBox ("Não é possível adicionar um item porque o campo Funcoinário está em branco.  " _
    & "Por favor preencha todos os campos em laranja antes de adicionar itens ao pedido.")
    Me!EmployeeID.SetFocus
End If

End Sub

Private Function NoChildren(strProductID As String) As Boolean
Dim dbsCurrent As DAO.Database
Dim rstTemp As DAO.Recordset

Set dbsCurrent = CurrentDB
Set rstTemp = dbsCurrent.OpenRecordset("SELECT ParentID FROM Where_Used WHERE ParentID = """ &
ReplaceString(strProductID, Chr(34), Chr(34) & Chr(34)) & """;")

If (rstTemp.RecordCount = 0) Then
  NoChildren = True
Else
  NoChildren = False
End If

rstTemp.Close
Set dbsCurrent = Nothing

End Function

Function Filter_Recordset_MultiLevelBOM(temp_rst As Recordset, aQuantity As Double, aLevelLoop As
Double) As Recordset
Dim levelCurrent As Double
Dim levelPrevious As Double
Dim levelLoop As Double
Dim parentQuantity As Double
levelCurrent = temp_rst!Level
Do While ((levelCurrent >= aLevelLoop) And Not (temp_rst.EOF))
    With temp_rst
    .Edit
    !UnitsSold = !UnitsSold * aQuantity
    .update
    parentQuantity = temp_rst!UnitsSold
    .MoveNext
    End With
    If Not (temp_rst.EOF) Then
        levelPrevious = levelCurrent
        levelCurrent = temp_rst!Level
        If (levelCurrent > levelPrevious) Then
            levelLoop = levelCurrent
            Set temp_rst = Filter_Recordset_MultiLevelBOM(temp_rst, parentQuantity, levelLoop)
            If (Not (temp_rst.EOF)) Then
                levelCurrent = temp_rst!Level
            Else
                Exit Do
            End If
        End If
    Else
        Exit Do
    End If
Loop
Set Filter_Recordset_MultiLevelBOM = temp_rst
End Function
```