EDITOR GRÁFICO PARA COLOCAÇÃO DE COMPONENTES EM PLACA DE CIRCUITO IMPRESSO (PCB)

Pedro Reis dos Santos[1], André Zúquete[2],
Joaquim Jorge[3] e Carlos Miguel Amaral[3]

(1) aluno IST, estagiário INESC, bolseiro JNICT
(2) aluno IST, estagiário INESC (3) investigador INESC

SUMÁRIO

O sistema para apoio ao projecto de circuitos electrónicos actualmente em desenvolvimento no **INESC** integra um conjunto de ferramentas que simplificam o trabalho do projectista e facilitam a detecção e correcção de erros de projecto. Neste artigo descreve-se um editor gráfico interactivo de colocação de componentes em placa de circuito impresso (*placer*), a sua adaptação a diversos sistemas integrados de desenvolvimento e flexibilidade relativamente à interface gráfica utilizada, permitindo a sua utilização com uma gama variada de dispositivos gráficos e postos de trabalho. Como facto importante salientamos a capacidade do editor para colocar interactivamente componentes na placa de circuito impresso em edição.

ABSTRACT

The CAD system for electronics being developed at **INESC**, includes a set of tools that simplifies and automates the project cycle making the project error free. In this article an interactive graphic editor for placement of components on printed circuit boards is presented. It's ability to adjust to several integrated systems in developement and a flexible graphic interface allows the use in a wide range of graphic devices and workstations. As an important caracteristic, the editor features interactive and automatic placement of components.

1. INTRODUÇÃO

Inicialmente a ênfase da investigação na área de **CAD** foi posta nos algorítmos e técnicas de programação dos vários programas de simplificação, verificação e automatização das diferentes fases de projecto. Assim, o projectista passou a dispôr de editores gráficos, simuladores, verificadores de regras de projecto, programas para colocação e interligação de componentes, etc.

Os programas que integram um sistema de apoio podem considerar-se agrupados em dois conjuntos distintos. Um é constituído pelos editores de esquemáticos, simuladores eléctricos ou lógicos e visualisadores de formas de onda enquanto que o outro integra os programas que descrevem o domínio físico, ou seja, verificadores de regras de desenho e colocação automática ou manual de componentes (*placer*) e a sua interligação (*router*). Esta clara separação do nível físico permite adaptar o sistema a vários tipos de projecto: **PCB**, híbridos e circuitos integrados.

Os elevados custos dos sistemas de desenho assistido por computador da actualidade impedem as empresas nacionais de apostar no seu investimento, apesar dos elevados ganhos que podem advir do seu emprego. Este trabalho surge no seguimento da implementação de um sistema deste tipo utilizando conhecimentos entretanto adquiridos noutros projectos e adaptando algumas ferramentas **CAD** disponíveis ou actualmente em desenvolvimento no **INESC** [1].

Este projecto consistiu no desenvolvimento de um editor de colocação de componentes em placa de circuito impresso que irá integrar de imediato o sistema **PCB-INESC** [2] e futuramente poderá vir a integrar o projecto **PACE** [3](Projecto Assistido por Computador para Electrónica, ainda em fase de arranque).

2. OBJECTIVOS DE PROJECTO

Pretendeu-se contruir um editor com capacidade para funcionar numa gama tão grande quanto possível de sistemas operativos sem efectuar qualquer tipo de alteração no seu código fonte. Se se garantir a portabilidade do código consegue-se poupar o por vezes enorme dispêndio de esforço e tempo. Tendo presente a mesma ideia, procurou-se tornar o editor, não só independente do dispositivo gráfico a utilizar como, permitir que um mesmo executável possa conter *drivers* para vários dispositivos gráficos, sendo estes escolhidos por configuração dinâmica deste. A necessidade de adaptar o editor a ambientes e condições de utilização muito diversas exige a possibilidade de configuração dinâmica deste. O esquema de configuração adoptado permite adaptar as características do programa sem modificar e recompilar todo o código fonte.

Para todos os parâmetros configuráveis do editor existe um valor de defeito interno. O valor destes pode ser alterado de duas formas:

- Ficheiro de configuração. Neste ficheiro cada utilizador deve explicitar os seus gostos pessoais ou características particulares da máquina onde se encontra. Por

gostos pessoais entende-se as cores usadas pelo editor e o tipo de interacção pretendido. As características particulares da máquina englobam o *driver* gráfico que pode ser usado e parâmetros dependentes do sistema operativo.

- Linha de comando. Nesta são normalmente indicadas as variáveis cujo valor é frequentemente alterado - ficheiros de comandos, de recuperação, etc.

3. INTERFACE HOMEM – MÁQUINA

Uma das caracteríticas importantes deste editor, consiste na capacidade de parametrizar os vários componentes de interface utilizador, podendo esta última ser dividida em várias zonas de interacção:

- *Input* de comandos;
- *Output* de texto. Permite apresentar texto que seja o resultado da execução de um comando ou as opções possíveis para um argumento de um comando;
- *Output* de mensagens de apoio à edição. Indicam qual o tipo do argumento seguinte do comando em introdução, consequências da execução de um comando ou outras informações de apoio (tempos de execução, etc.);
- *Output* de mensagens de erro. Indicam qualquer erro detectado durante a edição;
- *Output* de mensagens de estado. Indicam a hora corrente, as dimensões da área de colocação visível e o espaçamento da grelha de ajuste para apoio à colocação de componentes;
- Vizualização gráfica da área de colocação. Permite observar a colocação de componentes, de barreiras de proibição e de grelhas para orientação. Permite ainda apresentar outros tipos de informação gráfica adicional (malhas mínimas, gráficos de densidade de ligações, etc.);
- Utilização de menu de comandos. Permite a introdução de comandos em modo **locator**, que será posteriormente descrito.

Os quatro primeiros tipos de interacção podem ser realizados através de operações de leitura/escrita em ficheiros convencionais, ou utilizando áreas de ecran de dispositivos gráficos. Os três últimos são apenas conseguidos por intermédio de um *driver* gráfico (ver fig. 1).

Os comandos de edição foram concebidos de modo a permitir uma melhor interface utilizador-máquina para dois tipos de interacção possível – **string** e **locator**.

O modo **string** é o método usual de introdução de comandos num computador, sendo estes dados como uma sequência de caracteres através de um teclado. Pode usar qualquer terminal – gráfico ou não – para ecoar os caracteres introduzidos e consiste na indicação do nome do comando seguido dos seus argumentos. Por exemplo:

```
Prompt > MOVE U3 1200 4000
Prompt > ROTATE U3
```

A edição em modo **string** suporta caracteres especiais configuráveis de anulação e terminação de comandos mas não dispõe de facilidades de edição de linha. Para um utilizador menos experiente pode ser indicado o tipo de argumento pretendido, por exemplo:

```
Prompt > ZOOM
 Enter first corner
Prompt > 0 0
 Enter second corner
Prompt > 5000 5000
```

ou apenas:

```
Prompt > ZOOM
 Enter first corner
Prompt > 0 0 5000 5000
```

O modo **locator** só é possível numa interface suportada por um *driver* gráfico. O utilizador controla a posição de um cursor no ecran, usando dispositivos de localização (teclas de cursor, rato, etc.). A selecção de comandos é feita posicionando o cursor sobre o menu dos comandos possíveis e premindo um butão do rato quando o cursor se encontrar sobre o comando pretendido – ou de forma equivalente para os outros dispositivos físicos. Uma vez selecionado o comando, surge na zona de interacção para apoio uma mensagem de eco que indica qual o tipo de argumento esperado para esse comando. Simultaneamente, surgem na zona de *output* de texto as opções possíveis, podendo estas ser seleccionadas de forma idêntica à utilizada para introduzir comandos. De modo a facilitar a sua selecção essa zona de interacção dispõe de uma *scrollbar* para localizar a opção desejada. A *scrollbar* indica qual a percentagem e posição relativa do texto apresentado em relação ao texto total. Para apresentar o texto oculto modifica-se a posição da *scrollbar* indicando com o cursor a sua nova posição. Se o argumento desejado for um componente presente na área de colocação ou uma posição da mesma deve-se colocar o cursor sobre o componente ou na posição pretendida e premir o botão do rato (ver fig. 2). Para certos comandos existe um apoio visual constituído por *dragging* (deslocação de uma figura geométrica acompanhando o movimento do cursor) ou *rubberbanding* (desenho de uma linha ou rectângulo entre um ponto fixo e a posição do cursor).

De modo a permitir a troca de modos de *input* existe um comando no menu que permite mudar de modo **locator** para modo **string** e, no caso de se pretender a comutação inversa, um carácter de controlo configurável permite passar a modo **locator**. A troca de modos de *input* pode ser efectuada em qualquer ponto da introdução de um comando, sendo este continuado no modo e zona de interacção indicada.

4. IMPLEMENTAÇÃO

O editor foi escrito em linguagem C e o ambiente de desenvolvimento escolhido foi uma estação **VAXstation II/GPX** com um sistema de exploração **ultrix-32m**[1] (**UNIX**[2]) e uma interface gráfica **X**[3] [4], muito embora o editor tivesse como finalidade imediata a sua integração no sistema **PCB-INESC**. Este é suportado por *software* da **DATA GENERAL** modificado e adaptado no **INESC** e está actualmente instalado numa estação **DS/7500** com um sistema de exploração **AOS-VS**[4], utilizando uma inter-

[1] ultrix-32m é uma marca registada da Digital Equipment Corporation.

[2] UNIX é uma marca registada da AT&T Bell Laboratories.

[3] X Window System é uma marca registada do Massachusetts Institute of Technologie.

[4] AOS-VS é uma marca registada da Data General Corporation.

face gráfica **CGI-INESC** [5] desenvolvida no **INESC** no âmbito do projecto **CORTE**.

O código do editor foi separado em blocos com funções específicas de modo a facilitar alterações e futuros melhoramentos. Os dados necessários à edição e os seus resultados são transferidos de ou para outras aplicações por intermédio de ficheiros com um formato já definido no sistema de **PCB-INESC**. Estes ficheiros no seu conjunto formam a base de dados do sistema.

O editor é constituído por um conjunto de blocos que oferece uma base funcional necessária à construção de uma ampla variedade de editores gráficos. Estes blocos fornecem uma interface de fácil utilização que esconde pormenores de implementação quer do utilizador quer de quem desenvolve novos editores. Para tal definiu-se um conjunto de convenções que permite a introdução de novas facilidades sem colisão com as já existentes.

A cada bloco está associada uma semântica bem definida, utilizando o editor os blocos básicos que se descrevem a seguir.

4.1 Aquisição e processamento de comandos (interpretador)

Para adquirir um comando introduzido pelo utilizador – em modo **string** ou **locator** – o interpretador dispõe de um conjunto de rotinas para a sua construção e verificação sintática com base num conjunto de directivas previamente definidas. É da responsabilidade deste bloco o envio de mensagens de apoio indicando qual o tipo do argumento seguinte do comando, e de erro sintático caso o argumento não seja do tipo esperado. Os tipos de argumentos permitidos são nomes (cadeia de caracteres), números inteiros ou pontos (pares de números inteiros). O interpretador usa a zona de *output* de texto para indicar as opções possíveis do argumento de um dado comando. Estas estão previamente definidas juntamente com a sintaxe do comando.

Uma vez completamente introduzidos os argumentos do comando é chamada a rotina responsável pela sua interpretação semântica e execução. Os comandos que se consideram relevantes para a evolução da edição são guardados num ficheiro (**HistoryFile**) que permite, caso a edição termine anormalmente, recuperar o contexto da edição na altura da interrupção. Este ficheiro é apagado caso a sessão termine normalmente. Se durante a mesma se fizer uma extracção da colocação efectuada para um ficheiro, o **HistoryFile** é reinicializado pois o estado da edição pode ser recuperado a partir desse ficheiro.

4.2 Gestor de recursos

O gestor de recursos proporciona um método de associar valores a variáveis de estado do editor. Os utilizadores necessitam de meios que permitam integrar valores de defeito específicos em recursos e de um método flexível de especificar as suas preferências. Um sistema que especifique valores de defeito e permite redefini-los oferece a necessária flexibilidade e clareza. O algorítmo para determinar qual o recurso ou recursos que coincidem com o nome dado é o centro do gestor de recursos. A ideia é que os recursos são guardados subespecificados, ou seja, certos parâmteros como por exemplo o dispositivo gráfico a que se destinam podem ser omitidos se o recurso não for específico. Todos os pedidos especificam completamente o recurso que pretendem. O algorítmo de busca procura qual o nome que melhor se adapta ao pedido. Os valores de defeito dos recursos são indicados no código do editor. O seu valor pode ser alterado usando os modos de configuração do editor já indicados – ficheiro de configuração ou linha de comando. O gestor de recursos realiza também funções de um gestor de associações. O utilizador fornece na configuração do editor pares tipo–zona de interacção, sendo este responsável por fornecer pares tipo de interacção–rotina. Os recursos disponíveis do editor englobam:

- Indicação de ficheiros de dados e resultados;
- Escolha das áreas de interacção e da interface gráfica a utilizar.
- Escolha do *driver* gráfico;
- Definição de características da interface gráfica (cores, etc.);
- Outros atributos (*prompt*, etc.).

4.3 Gestor da base de dados

O gestor da base de dados é responsável pela transferência de informação contida em bases de dados do sistema de **PCB** para listas próprias guardadas e manipuladas dinamicamente em memória durante a edição. Todas as alterações das bases de dados do sistema são igualmente da sua responsabilidade.

O gestor garante a coerência de toda a informação guardada. Controla igualmente todos os acessos dos restantes blocos do editor às estruturas de dados.

4.4 Gestor gráfico

O gestor gráfico converte as descrições topológicas existentes na base de dados em informação utilizável pelos *drivers* gráficos. A partir da topologia associada a cada elemento da base de dados que pode ser vizualizado, o gestor gráfico gera um conjunto de primitivas geométricas (pontos, linhas, rectângulos, etc.) que irão ser interpretadas por um *driver* gráfico e dar origem a primitivas gráficas. Gera também primitivas geométricas não associadas a elementos da base de dados (grelhas, ponto fixo para *rubberbanding*, etc.).
O gestor gráfico possui duas interfaces possíveis com um *driver* gráfico conforme este suporte ou não segmentos. No primeiro caso uma grande parte das funções do gestor gráfico são ignoradas já que um *driver* gráfico que suporte segmentos conhece a relação entre uma topologia e as primitivas gráficas que lhe estão associadas.

O gestor gráfico é também responsável pela manutenção de informação gráfica de estado (cores, realces, etc.), e possui um conjunto de rotinas utilitárias de conversão de informação gráfica que poderão ser utilizadas por outros blocos do editor (translação e rotação de rectângulos, geração de rectângulos envolventes, etc.).

4.5 Gestor de colocação automática

Um gestor de colocação automática é um elemento muito útil para a colocação de componentes cuja posição não é questionável mas que aumentam grandemente o tempo da edição e o trabalho do utilizador (condensadores de desacoplamento, etc.).

O gestor de colocação automática mantém informação interna relativa à colocação de todos os componentes que é actualizada após a execução de certos comandos. É responsável por colocar automáticamente os componentes que lhe forem indicados e permite verificar a validade da colocação – indicando se um componente está fora da placa ou sobreposto a outro [6].

4.6 *Drivers* gráficos

Esta versão do editor dispõe de três *drivers* gráficos possíveis. A escolha da interface gráfica a utilizar é efectuada no início da edição.

O *driver* base do editor é suportado pelo pacote gráfico **CGI**, desenvolvido no **INESC** e baseado na norma gráfica **CGI**[5]. Garante uma interface idêntica com inúmeros dispositivos físicos (terminais *raster*, ploters, etc.), executa transformações *window – viewport* e suporta segmentos.

Dada a excelente performance dos pacotes gráficos baseados no **X Window System (X10** e, especialmente, a última versão **X11)** desenvolveram-se *drivers* para este sistema. . O sistema **X** é um sistema extremamente leve – não degrada a performance das aplicações – não suportando facilidades comuns a pacotes gráficos correntes (**GKS, CGI,** etc.) – transformações *window – viewport* e segmentos. Como todas as zonas de interacção utilizadas, excepto uma, são apenas texto não necessitando de transformações ou segmentação é poupado um considerável tempo de processamento. Determinante é o tempo perdido no *output* de texto gráfico – caso do **CGI** – enquanto que o sistema de fontes do X garante uma boa resposta.

5. CONCLUSÕES

Neste artigo descreveu-se a concepção e realização de um editor gráfico interactivo para colocação de componentes em placa de circuito impresso. Um esquema de configuração e gestão de recursos flexível, revela-se de grande importância para a adaptação e utilização do editor numa grande diversidade de ambientes e dispositivos. Este requesito mostra-se indispensável dada a muito rápida evolução dos componentes e sistemas **CAD** actuais.

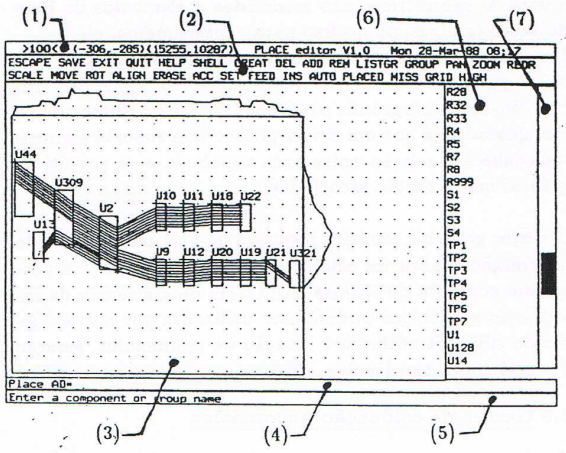


Figura 1: Exemplo de utilização de todas as zonas de interacção suportadas por um *driver*; (1) informação de estado; (2) menu de comandos; (3) área de edição; (4) linha de *input*; (5) linha de mensagens de apoio e erro; (6) janela de texto; (7) scroolbar da janela de texto.

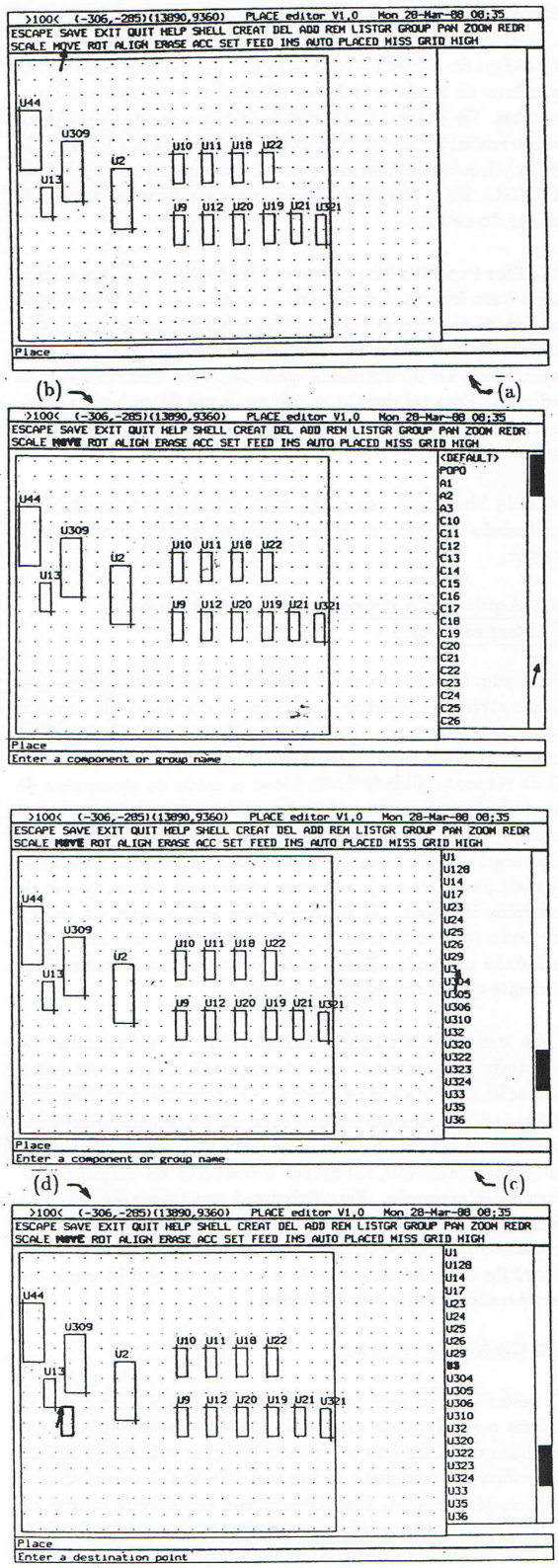
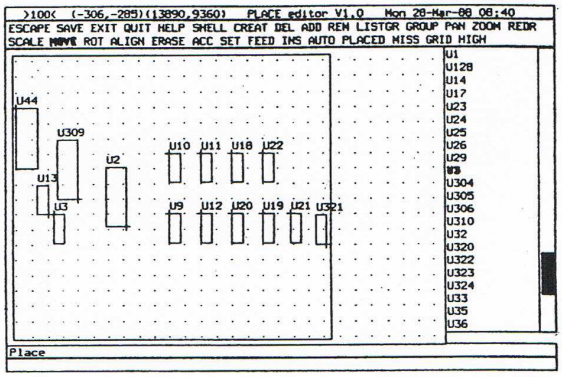


Figura 2: Sequência de um comando introduzido em modo **locator**(MOVE U3 1200 4000): (a) Escolha do comando; (b) Procura do primeiro argumento (U3); (c) Escolha do primeiro argumento (U3); (d) Escolha do segundo argumento (posição 1200 4000);



(e)

Figura 2: (continuação): (e) Resultado do comando.

Referências

[1] G.P. Saragoça, M.R. Gomes, M.H. Sarmento, J.A. Jorge, "Concepção e Desenvolvimento de um Editor Esquemático para Circuitos Elétricos e Electrónicos", ENDIEL 87.

[2] M.R. Gomes, J.L. Fernandes, "O Sistema PCB-INESC", 3º CPI 84.

[3] M.J. Silva, M.H. Sarmento, "PACE - A New Approach to Solve the Design Integration Problem", Relatório Interno do INESC, Março 1988.

[4] R.W. Scheifler, J. Gettys, "The X Window System", ACM Transactions on Graphics, April 1986.

[5] ISO/DP9636 (CGI) - "Functional Specification", ISO, December 1986.

[6] Carlos Amaral, Joaquim Jorge, "Colocação automática de componentes em placas de circuito impresso", pppac, dezembro 1987